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

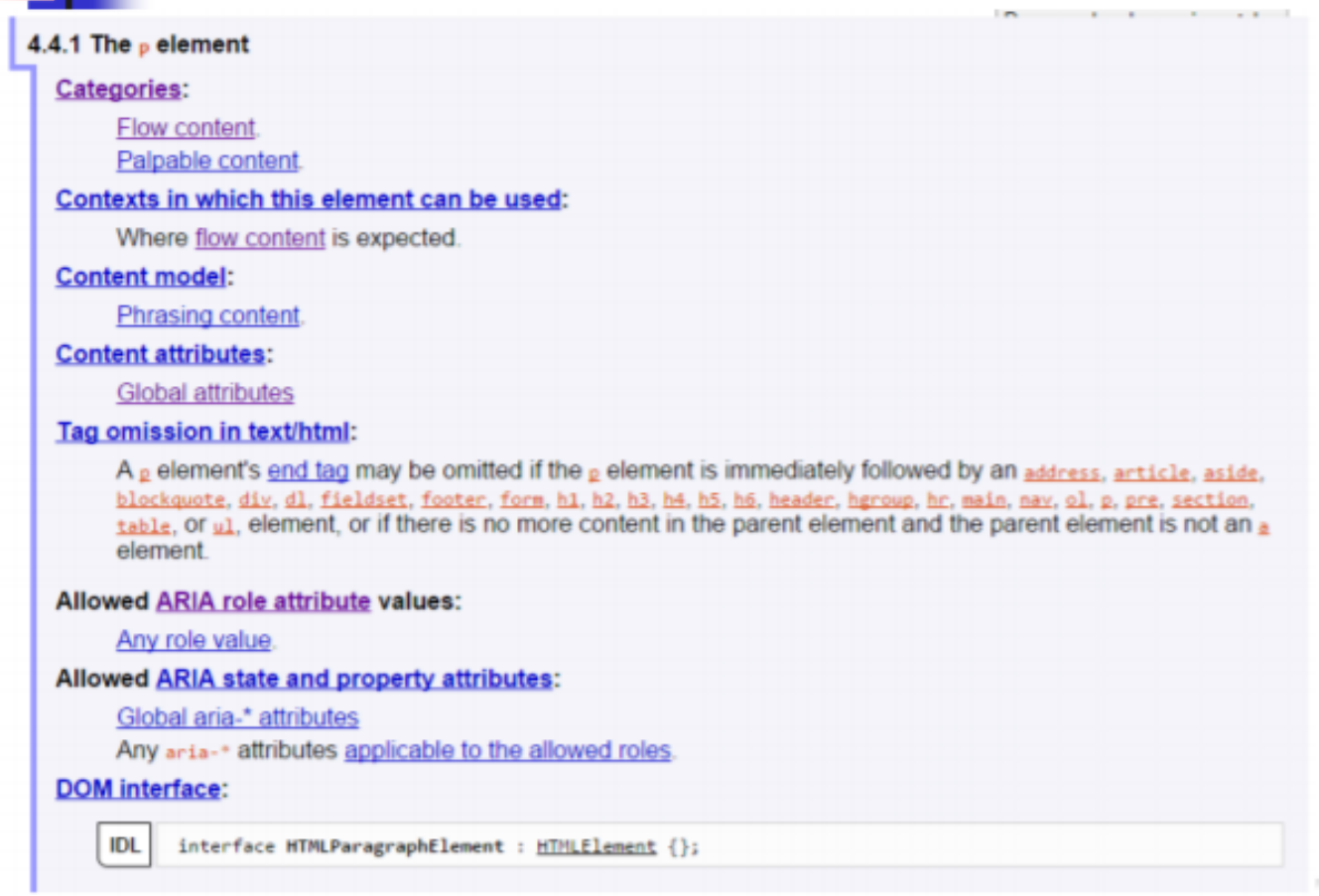
I riferimenti a caratteri sono forme di markup con cui rappresentare caratteri. Esistono tre tipi di riferimenti:

- 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

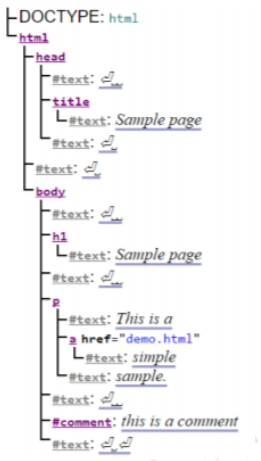
- **La categoria dell'elemento**

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).

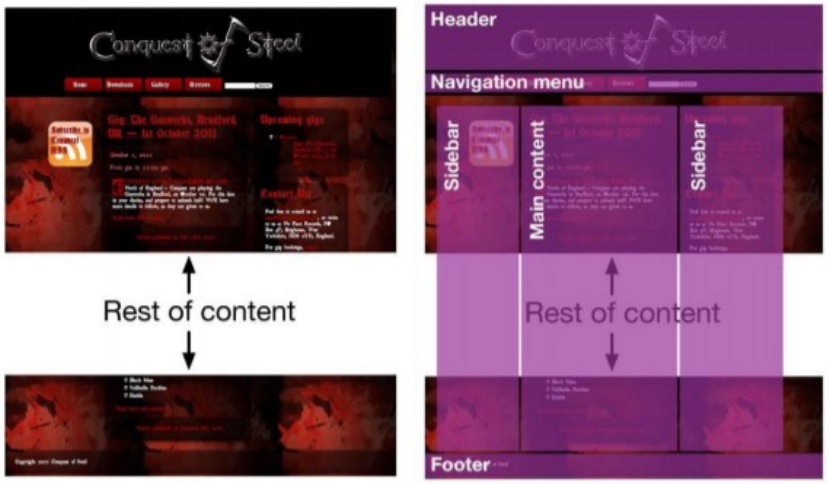
Le sezioni sono tre:

- **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)

Spiegazioni elementi HTML	
Nome	**Spiegazione**
head	- 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*)

title	- 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**:
	`<head>`
	`  <title>Titolo del documento</title>`
	`</head>`

meta	- 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**:
	`<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>`

base	- 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**:
	`<head>`
	`  <base href="https://ifrax.it/index.php">`
	`</head>`
	`<body>`
	`  [Ancora alla pagina https://ifrax.it/registri/index.php]`
	`  <a href="registri/index.php">Registri</a>`
	`</body>`

link	- Elemento con cui è possibile specificare le relazioni tra il nostro documento e risorse esterne.

<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)

</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.
	`<section>`
	`<!-- ... -->`
	`<p>Last modified: 2001-04-23</p>`
	`<p>Author: fred@example.com</p>`
	`</section>`
	Technically correct, but not appropriate
	`<section>`
	`<!-- ... -->`
	`<footer>Last modified: 2001-04-23</footer>`
	`<address>Author: fred@example.com</address>`
	`</section>`
	Better
	- **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>`
	The pre element
	This is the `Panel` constructor:
	`function Panel(element, canClose, closeHandler) {`
	`  this.element = element;`
	`  this.canClose = canClose;`
	`  this.closeHandler = function () { if (closeHandler) closeHandler() };`
	`}`

blockquote	- Il blockquote racchiude contento estratto da fonti esterne

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

- Possiamo porre, all'interno di `blockquote`, l'elemento `cite` per indicare la fonte.

- **Esempio**:

```
<blockquote>
      The people recognize themselves in their commodities;
they find their soul in their automobile, hi-fi set, split-
level home, kitchen equipment. — <cite><a
href="http://en.wikipedia.org/wiki/Herbert_Marcuse">Herbert
Marcuse</a></cite>
</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*

</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 `figure`**:

[Esempio con paragrafo]

```
<figure>
      <p>'Twas brillig, and the slithy toves<br>
      Did gyre and gimble in the wabe;<br>
      All mimsy were the borogoves,<br>
      And the mome raths outgrabe.</p>
      <figcaption><cite>Jabberwocky</cite> (first verse). Lewis Carroll,
1832-98</figcaption>
</figure>
```

'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]

```
<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>
```

</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**:

```
[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>
```

</td></tr>
<tr><td>ul</td><td>

- Elemento che contiene liste non ordinate

- **Esempio**:

```
<ul>
      <li>Norway</li>
      <li>Switzerland</li>
      <li>United Kingdom</li>
      <li>United States</li>
</ul>
```

- 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**:

<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}).

span	- 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 `div` e `span` presentano differenze importanti a livello di CSS.
br	- Tag con cui si indica un *line-break* (andare a capo).
wbr	- 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 `br`**: `<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>` So then he pointed at the tiger and screamed "thereisnowayyouareevergoingto catchme"!
Edits	
ins	- 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.

del	- 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 `ins`**:
	`<ul>`
	`<li><del datetime="2009-10-10T23:38-07:00">Apple</del></li>`
	`<li>Orange</li>`
	`<li><del>Pear</del></li>`
	`<li><ins>Teal</ins></li>`
	`<li><del>Lemon</del><ins>Yellow</ins></li>`
	`<li><ins>Olive</ins></li>`
	`</ul>`
	• ~~Apple~~
	• Orange
	• ~~Pear~~
	• Teal
	• ~~Lemon~~Yellow
	• Olive

Embedded content	
img	- 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.

- **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

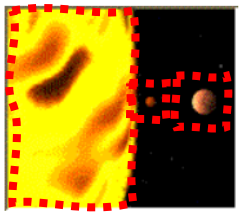
con essi indichiamo

- 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:

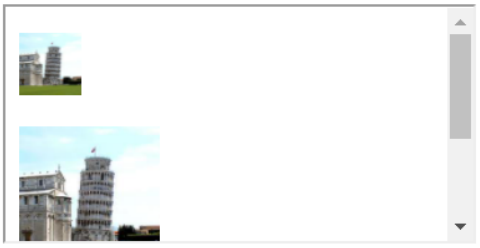


Evidenziate col tratteggio le parti cliccabili

`iframe`

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

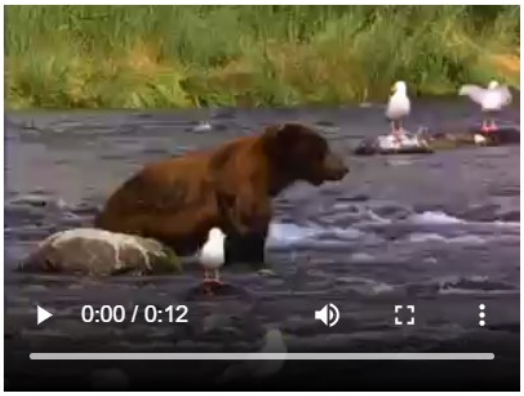
<table>
<tr><td></td><td>
- L’attributo srcdoc definisce un contenuto annidato in HTML (se presente l’attributo src si darà priorità al contenuto in srcdoc)
- Con l’attributo name assegnamo un nome all’iframe (cosa che può essere utile con l’attributo target dell’elemento a.
- Possiamo utilizzare gli 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.

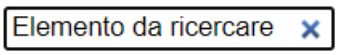
  - Selettore di file:
    - Controllo che permette all'utente di selezionare files e inviarli attraverso un form.

- **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

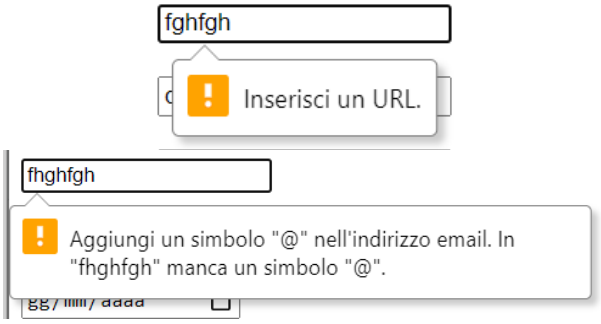
      - `search`, campo di ricerca senza line breaks

        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.

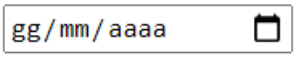
- Questi input esteticamente sono uguali all'input text. Eventuali controlli sono svolti dal browser e non li possiamo dare per scontato. Google Chrome, per esempio, notifica se un qualcosa non è stato inserito correttamente.



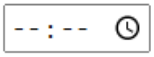
- `password`, testo senza line breaks e nascosto da asterischi (informazioni sensibili)

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?).

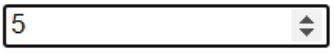
- `date`, permette di inserire una data



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

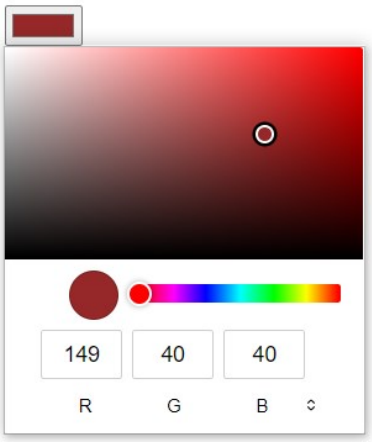


- `number`, valore numerico



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

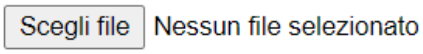
- `color`, si permette di scegliere un colore dalla cosiddetta paletta.



- `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



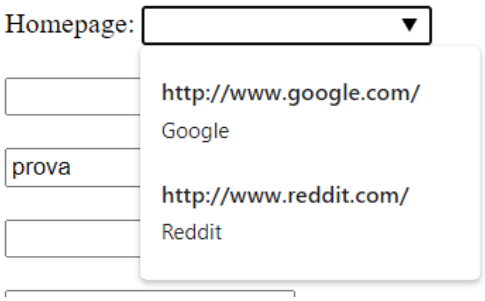
- `submit`, bottone per la sottomissione del form.

- `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>
```



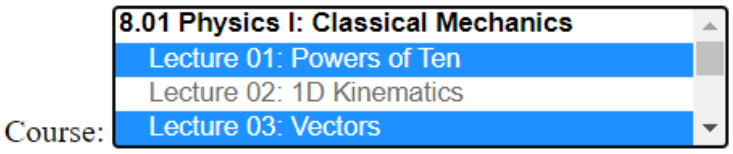
- `button`
  elemento che permette di creare un bottone primitivo (esiste sia l'input di tipo button che l'elemento button). L'elemento è più flessibile rispetto all'input: in particolare possiamo inserire delle immagini all'interno del bottone. Vediamo i seguenti attributi:
  - `disabled`, permette 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)

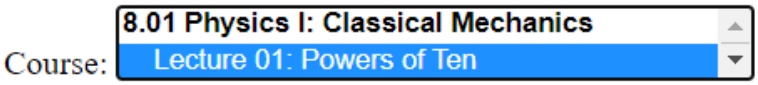
- `select`, `option` e `optgroup`
  L'elemento select delimita il codice relativo a un menu. Sono disponibili i seguenti attributi:
  - `disabled`, disabilita il menu a tendina

  - `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

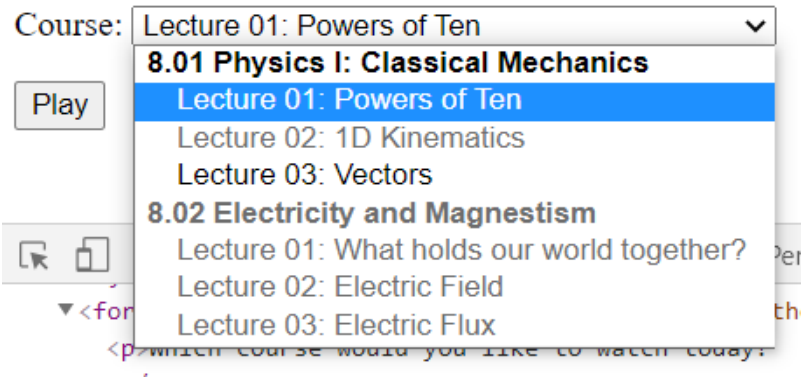
  - `selected`, permette di impostare un'opzione di default (attributo booleano, l'utente troverà quell'opzione già selezionata)

  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>
    ```

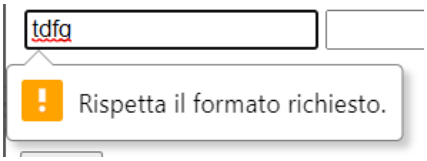
- Possiamo includere i seguenti attributi:
  - `placeholder`
    contenuto che permette all'utente di capire cosa inserire in quel controllo.

  - `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.

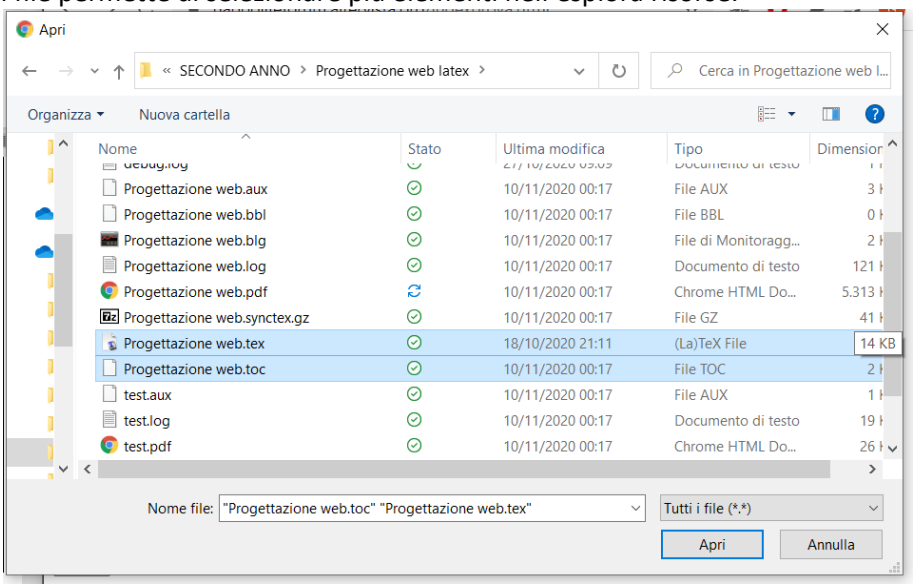
```
▼<td>
    <input required="required" name="3.pid" value
    pattern="[A-Z0-9]+"> == $0
  </td>
```



- `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

```
▼<td>
    " $"
    <input required="required" type="number" min="0"
    step="0.01" name="3.pprice" value> == $0
  </td>
▶<td>…</td>
```

- `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>
```

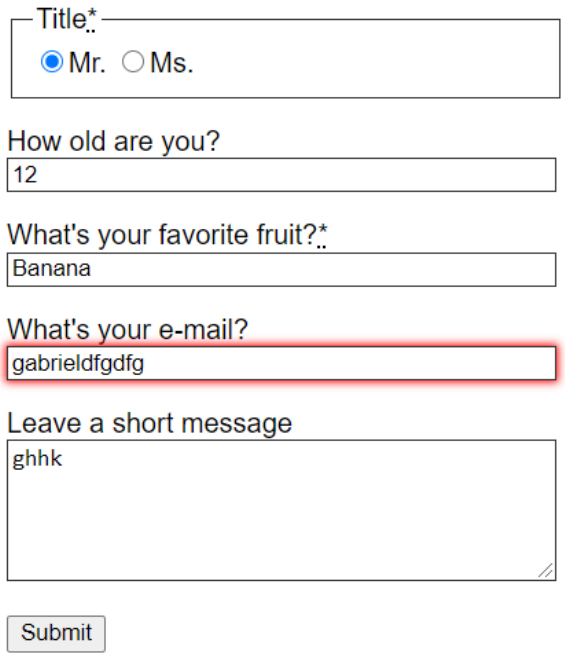
    Supponiamo che l'utente invii due file, un file di testo e un immagine. Otteniamo

```
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.

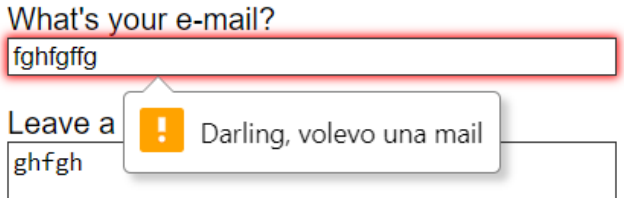
- Prendiamo il seguente codice:




```
<!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.

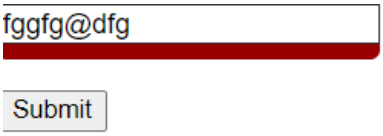
  Vediamo un esempio:

```
<!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
```

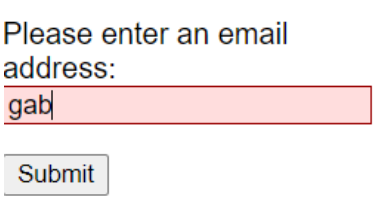
```
                    error.className = "error"; // Reset the visual state
of the message
                }
            }, false);

            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`.

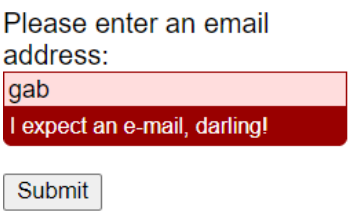


        In aggiunta si chiama la funzione `preventDefault()` per bloccare la sottomissione del form.

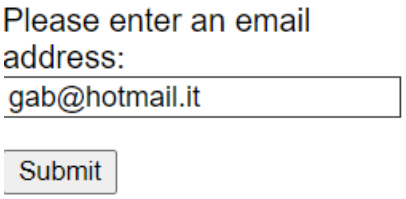
    - Ogni volta che viene captato un evento `input` si verifica la validità della mail e si rimuovono eventuali messaggi d'errore (oltre a ripristinare la classe iniziale dello span). Questo serve per rimuovere il messaggio di errore dopo aver posto un primo valore sbagliato: non appena l'utente indicherà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