HTML: HyperText Markup Language

Angelo Di Iorio
Università di Bologna

Hello World HTML

Dichiarazione del tipo del documento
(versione HTML e schema di "validazione")

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Hello World</title>
    </head>

    <body>
        <h1>Hello World!</h1>
    </body>

</html>
```

HEAD
Informazioni sul documento

BODY
Contenuto vero e proprio

Premesse sintattiche

- Maiuscolo/minuscolo:
  - HTML non è sensibile al maiuscolo/minuscolo (XHTML lo è e vuole tutto in minuscolo).
- Whitespace
  - HTML *collassa* tutti i caratteri di whitespace (SPACE, TAB, CR, LF) in un unico spazio. Questo permette di organizzare il sorgente in modalità leggibile senza influenzare la visualizzazione su browser.
  - L'entità   permette di inserire spazi non collassabili.
- Parsing e (non) buona forma
  - HTML rilassa diversi vincoli sulla buona forma, che invece erano imposti in XHTML

Entità in HTML

HTML definisce un certo numero di entità per quei caratteri che sono:

- proibiti perché usati in HTML (<, >, &, “, ecc.)
- proibiti perché non presenti nell’ASCII a 7 bit.

Ad esempio:

– amp	&	quot	“
– lt (less than)	<		gt (greater than) >
– reg	®	nbsp (non-breaking space)	
– Aelig	Æ	Aacute	Á
– Agrave	À	Auml	Ä
– aelig	æ	aacute	á
– agrave	à	auml	ä
– ccedil	ç	ntilde	ñ

– ecc.

Attributi globali: *coreattrs*

Sono attributi globali quelli definiti su tutti gli elementi del linguaggio HTML.

I *coreattrs* costituiscono attributi di qualificazione e associazione globale degli elementi. Per lo più per CSS e link ipertestuali.

- **id**: un identificativo unico (su tutto il documento)
- **style**: un breve stile CSS associato al singolo elemento
- **class**: una lista (separata da spazi) di nomi di classe (per attribuzione semantica e di stile CSS)
- **title**: un testo secondario associato all'elemento (per accessibilità e informazioni aggiuntive)

Attributi globali: *i18n* e *eventi*

- Gli attributi i18n *(internationalization)* garantiscono l'internazionalizzazione del linguaggio e la coesistenza di script diversi
  - **lang**: una codifica dei linguaggi umani (stringa a due caratteri: it, en, fr, etc. da RFC1766)
  - **dir**: uno dei due valori ltr (left-to-right) o rtl (right-to-left) per indicare la direzione di flusso secondario del testo.

- Gli attributi di evento permettono di associare script a particolari azioni sul documento e sui suoi elementi:
  - **onclick**, **ondoubleclick**, **onmouseover**, **onkeypress**, ecc.

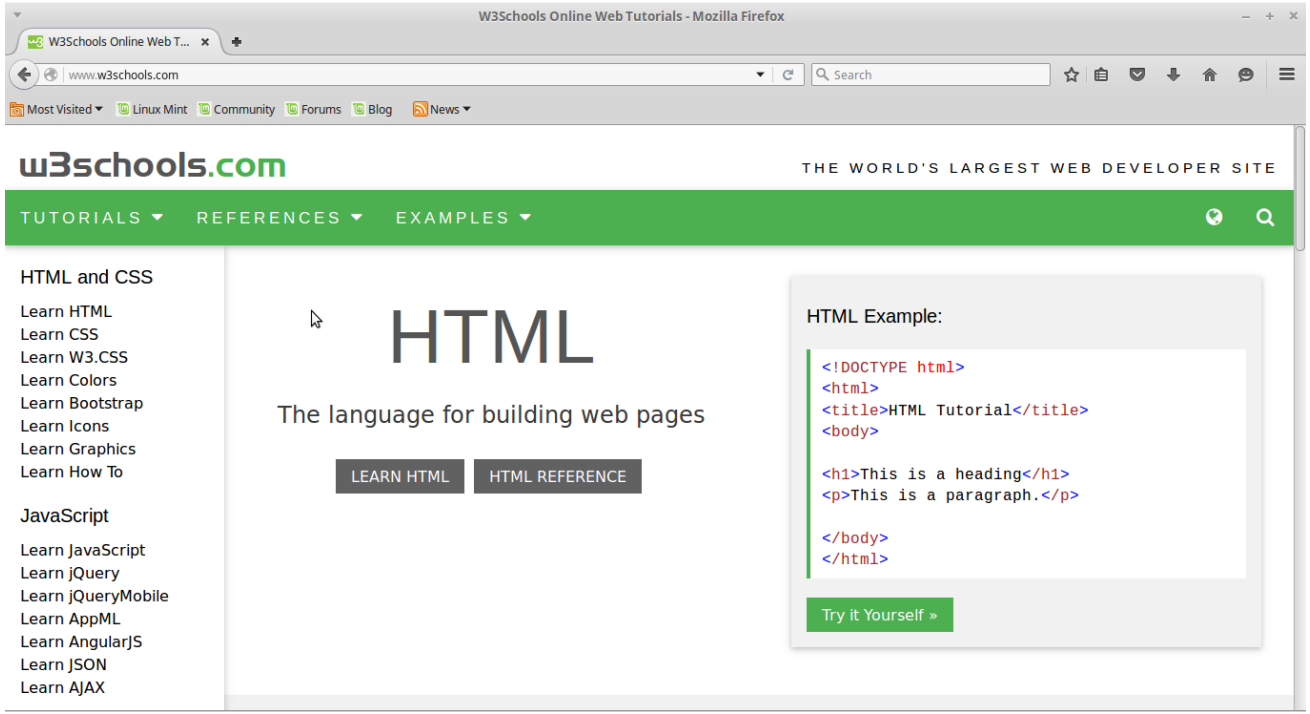
Gli elementi di HTML

Gli elementi di HTML sono organizzati secondo alcune categorie:

- Struttura complessiva (HTML, HEAD, BODY)
- Struttura del contenuto (SECTION, NAV, FOOTER, ecc.)
- Elementi di blocco e liste (P, H1, H2, DIV, UL, OL, ecc.)
- Elementi inline (B, I, SPAN, ecc.)
- Elementi speciali (A, IMG, HR, BR)
- Tabelle (TABLE, TR, TD, TH)
- Form (FORM, SELECT, OPTION, INPUT)

HTML - W3 Schools

https://www.w3schools.com/html/



Elementi inline (1)

- Gli elementi *inline* (o di carattere) non spezzano il blocco (non vanno a capo) e si includono liberamente l'uno dentro all'altro. Non esistono vincoli di contenimento.
- Si dividono in elementi *fontstyle* e elementi *phrase*.
- I tag *fontstyle* forniscono informazioni specifiche di rendering. Molti sono deprecati e si suggerisce comunque sempre di usare gli stili CSS.
  - TT (TeleType, font monospaziato, ad es. Courier),
  - I (corsivo),
  - B grassetto,
  - U (sottolineato - deprecato),
  - S e STRIKE (testo barrato - deprecato),
  - BIG, SMALL (testo più grande e più piccolo)

Elementi inline (2)

I tag *phrase* (di fraseazione o idiomatici) aggiungono significato a parti di un paragrafo.

- EM (enfasi)
- STRONG (enfasi maggiore)
- DFN (definizione)
- CODE (frammento di programma)
- SAMP (output d'esempio)
- KBD (testo inserito dall'utente)
- VAR (variabile di programma)
- CITE (breve citazione)
- Q (citazione lunga)
- ABBR e ACRONYM (abbreviazioni ed acronimi)
- SUP e SUB (testo in apice e in pedice)
- BDO (bidirectional override)
- SPAN (generico elemento inline)

Elementi di blocco

I tag di blocco definiscono l'esistenza di blocchi di testo che contengono elementi inline.

Elementi base:

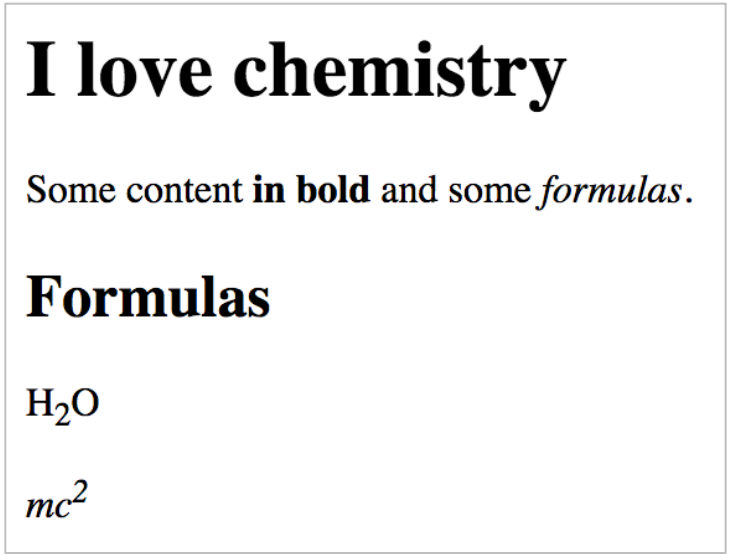
- P (paragrafo),
- DIV (generico blocco),
- PRE (blocco preformattato),
- ADDRESS (indicazioni sull'autore della pagina),
- BLOCKQUOTE (citazione lunga)

Blocchi con ruolo strutturale

- H1, H2, H3, H4, H5, H6 (intestazione di blocco)

Esercizio

- Scrivere il codice HTML per visualizzare nel browser il seguente contenuto:



I love chemistry

Some content **in bold** and some *formulas*.

Formulas

H_2O

mc^2

Elementi per liste

Le liste di elementi sono contenitori di elementi omogenei per tipo.

- UL: Lista a pallini di <LI>; Attributo type (disc, square, circle)
- OL: lista a numeri o lettere di <LI>; attributi start (valore iniziale) e type (1, a, A, i, I).
- DIR, MENU: liste compatte, poco usate
- DL: lista di definizioni <DT> (definition term) e <DD> (definition data)

```
<ul>
   <li>Primo</li>
   <li>Secondo</li>
   <li>Terzo</li>
</ul>


<dl>
   <dt>Lista</dt>
   <dd>Un contenitore
      di elementi</dd>
   <dt>Phrase</dt>
   <dd>Elementi inline di
     tipo idiomatico</dd>
</dl>
```

Esercizio

- Scrivere il codice HTML per visualizzare nel browser il seguente contenuto:

- Universe
  - Mister X
  - Mister Y
- UniBO
  1. Angelo Di Iorio
  2. Silvio Peroni

Elementi generici

- ** ** e **** sono cosiddetti *elementi generici*: privi di caratteristiche semantiche o presentazionali predefinite, assumono quelle desiderate con l'aiuto dei loro attributi (style, class, lang).
- ** ** mantiene la natura di elemento blocco, e **** la natura di elemento inline, ma ogni altra caratteristica è neutra.

```
<div id="introduction">
    <p>
     <span class="name">Angelo Di Iorio</span> works
     at <span class="place">Bologna</span>
    </p>
    …
   </div>
```

Link ipertestuali (anchors) (1)

- I link sono definiti con elementi **<a>** (àncore nel documento).
- **<a>** è sintatticamente un elemento inline. L'unica limitazione è che gli elementi **<a>** non possono annidarsi.
- Attributi:
  - ***href***: specifica l'URI della destinazione. Quindi `<a href="xxx"> ... </a>` è l'ancora di partenza di un link.
  - ***name***: specifica un nome che può essere usato come ancora di destinazione di un link. Quindi `<a name= "xxx"> ... </a>` è l'ancora finale di un link.

Link ipertestuali (anchors) (2)

A link to the whole document called first

The following is a link to a location of the same document.

first.html →

```
<body>
  <p>A link to the <a href="second.html">
     whole document</a> called first</p>
  <p>The following is a link to <a
     href="second.html#location1"> a
     location</a> of the same document.</p>
 </body>
</html>
```

Link ipertestuali (anchors) (3)

Beginning of the second document

... Much more content ...

A paragraph that contains a destination of a hypertext link.

... More content still

second.html →

```
    <p>Beginning of the second document</p>
    ... Much more content ...
    <p>A paragraph that contains <a
       name="location1">a destination</a>
       of a hypertext link.</p>
    ... More content still
  </body>
</html>
```

Esercizio

- Scrivere il codice HTML di una pagina **`seasons.html`** che contiene una lista non ordinata di 4 elementi, uno per ogni stagione (*spring*, *summer*, *autumn*, *winter*)
- Ogni voce contiene un link alla pagina **`<season-name>.html`** contenuto nella directory **`seasons/`**
- Ogni pagina **`<season-name>.html`** contiene un link alla pagina principale
- Usare URL relativi

Immagini

Le immagini inline sono definite attraverso l'elemento IMG.
Formati tipici: JPEG, GIF, PNG.

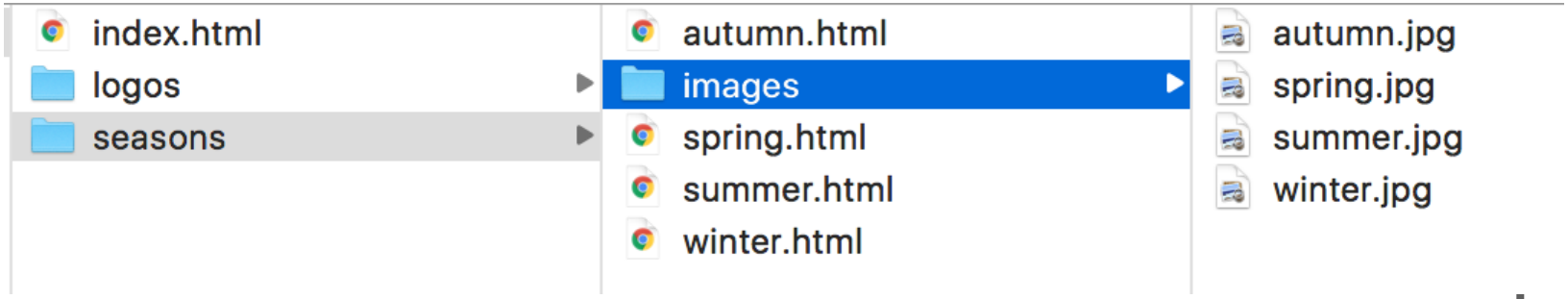
```
<img src="bologna.jpg" alt="Foto di Bologna">
```

Alcuni attributi:

- **SRC (obbligatorio): l'URL (assoluto o relativo) del file contenente l'immagine**
- ALT: testo alternativo in caso di mancata visualizzazione dell'immagine
- NAME: un nome usabile per riferirsi all'immagine
- WIDTH: forza una larghezza dell'immagine.
- HEIGHT: forza una altezza dell'immagine.
- ALIGN, BORDER, VSPACE, HSPACE: deprecati, specificano il rendering.

Esercizio

- Modificare le pagine `<season-name>.html` in modo che contengano ognuna un'immagine che rappresenta la stagione.
- Usare URL relativi assumendo che i file siano organizzati con la seguente struttura (usare immagini arbitrarie)



Elemento `figure`

- HTML5 ha introdotto anche un elemento specifico per aggiungere un'immagine ad un documento, corredata da una didascalia, in un unico blocco semanticamente rilevante
- Diverso da IMG che è un elemento inline

```
<figure>
        <img src="pictures/botticelli.jpg"
             alt="La Primavera di Botticelli">
        <figcaption>La Primavera di Botticelli
        </figcaption>
</figure>
```

Elementi di struttura

- HTML 4 fornisce pochi elementi per strutturare i contenuti in sezioni e sottosezioni: il contenitore gerarchico **`<div>`** (cui associare un ruolo tramite l'attributo **`@class`**) e gli elementi **`<h1>,..,<h6>`** (che però non sono annidati)

- HTML 5 (ma la maggior parte erano già presenti in XHTML 2.0) introduce nuovi elementi con una semantica precisa, per organizzare il documento in **contenitori**:
  - **`<section>`**: un contenitore generico annidabile.
  - **`<article>`**: una parte del documento *self-contained* e pensata per essere ri-usata e distribuita come unità atomica (post di un blog, news, feed, etc.)
  - **`<aside>`**: una sezione collegata al testo ma separata dal flusso principale (note a margine, sidebar, incisi, pubblicità, etc.)
  - **`<header>`** e **`<footer>`**: elementi iniziali e finali di un documento
  - **`<nav>`**: liste di navigazione

HTML 4 corretto ma...

```
<h1>Benvenuto nella mia home!</h1>
<p>Si divide in due parti: short-bio e post
giornalieri...</p>
<h2>Bio</h2>
<p>Nato a, laureato il, dott
<h2>9 marzo 2017</h2>
<p>Che giornata grigia.</p>
<h2>13 marzo 2017</h2>
<p>Una bella giornata di sol
```

Benvenuto nella mia home!

Si divide in due parti: short-bio e post giornalieri...

Bio

Nato a, laureato il, dottorato il, etc.

9 marzo 2017

Che giornata grigia.

13 marzo 2017

Una bella giornata di sole.

…molto meglio (HTML 4)

```
<div id="main">
      <h1>Benvenuto nella mia home!</h1>
      <p>Si divide in due parti: short-bio e post
         giornalieri...</p>
      <div id="bio">
         <h1>Bio</h1>
         <p>Nato a, laureato il, dottorato il, etc.</p>
      </div>
      <div id="posts">
         <div id="09-03-17">
            <h1>09 marzo 2017</h1>
            <p>Che giornata grigia.</p>
         </div>
         <div id="13-03-17">
            <h1>13 marzo 2017</h1>
            <p>Una bella giornata di sole.</p>
         </div>
      </div>
</div>
```

Ancora meglio

```
<section id="main">
    <h1>Benvenuto nella mia home!</h1>
    <p>Si divide in due parti: short-bio e post
       giornalieri...</p>

    <section id="bio">
        <h1>Bio</h1>
        <p>Nato a, laureato il, dottorato il, etc.</p>
    </section>

    <section id="posts">
        <article id="09-03-17">
            <h1>9 marzo 2017</h1>
            <p>Che giornata grigia.</p>
        </article>

        <article id="13-03-17">
            <h1>13 marzo 2017</h1>
            <p>Una bella giornata di sole.</p>
        </article>
    </section>
</section>
```

Header e footer

<header>: contiene l'intestazione della sezione corrente o dell'intera pagina ed è solitamente usato per tabelle di contenuti, indici, form di ricerca, intestazioni.

- Può essere anche usato come contenitore degli headings H1, .., H6.

<footer>: contiene la "parte conclusiva" della sezione corrente o dell'intera pagina. E' usato principalmente per mostrare informazioni (testuali, non metadati) sugli autori della pagina, copyright, produzione, licenze

- Non deve essere visualizzato necessariamente a fondo pagina
- Può essere usato anche per contenere intere sezioni come appendici, allegati tecnici, colophon, etc.

Liste di navigazione

- HTML 5 riprende da XHTML 2.0 anche le “liste di navigazione” ossia particolari sezioni dedicate a raggruppare link alla pagina corrente (o a sezioni di) o ad altre pagine
- Si usa l’elemento <**nav**> molto spesso in combinazione con <**header**> e <**footer**>.
- Le sezioni <**nav**> sono molto utili per l’accessibilità, in quanto possono essere più facilmente identificate e accedute anche da utenti disabili (tramite screenreaders o altri ausili)
- Possono essere usate anche per attivare/disattivare le funzionalità di navigazione in base allo user-agent (browser) che sta accedendo alla pagina.

```
<footer>
   <nav>
     <h1>Navigation</h1>
     <ul>
        <li><a href="/">Home</a></li>
        <li><a href="/standards/">Standards</a></li>
        <li><a href="/participate/">Participate</a></li>
        <li><a href="/membership">Membership</a></li>
        <li><a href="/Consortium/">About W3C</a></li>
     </ul>
   </nav>
   <nav>
     <h1>Contact W3C</h1>
     <ul>
        <li><a href="/Consortium/contact">Contact</a></li>
        <li><a href="/Help/">Help and FAQ</a></li>
        <li><a href="/Consortium/sup">Donate</a></li>
        <li><a href="/Consortium/siteindex">Site Map</a></li>
     </ul>
   </nav>
   …
</footer>
```

Tabelle

- Le tabelle vengono specificate **riga per riga**.
- Di ogni riga si possono precisare gli elementi, che sono o intestazioni o celle normali.
- Una tabella può anche avere una didascalia, un'intestazione ed una sezione conclusiva.
- E' possibile descrivere insieme le caratteristiche visive delle colonne.
- Le celle possono occupare più righe o più colonne
  - Si usano gli attributi `@rowspan` e `@colspan` per indicare il numero di righe o colonne occupate dalla cella

Tabella semplice

```
<body>
  <h1>Tables</h1>
      <table border="1">
        <tr>
          <th>Salesman</th><th>Jan</th>
          <th>Feb</th> <th>Mar</th>
        </tr>
        <tr>
           <th>John Smith</th> <td>12000</td>
           <td>13000</td><td>15000</td>
        </tr>
        <tr>
           <th>Alice Green</th><td>7000</td>
           <td>9000</td><td>11000</td>
        </tr>
        <tr>
           <th>Hugh Brown</th><td>25000</td>
           <td>23000</td><td>30000</td>
        </tr>
      </table>
</body>
```

Tables

Salesman	Jan	Feb	Mar
John Smith	12000	13000	15000
Alice Green	7000	9000	11000
Hugh Brown	25000	23000	30000

Tabella più complessa

```
<body>
  <h1>Tables</h1>
      <table border="2" width="80%" cellpadding="5" cellspacing="0">
          <caption>Sales in first quarter</caption>
  <col style="background-color:#FFFFBB" width="70%">
  <col span="3" width="10%">
          <thead>
              <tr>
                  <th>Salesman</th><th>Jan</th>
                  <th>Feb</th> <th>Mar</th>
              </tr>
          </thead>
          <tfoot>
              <tr>
                  <th>Totals</th><th>34000</th>
                  <th>45000</th> <th>56000</th>
              </tr>
          </tfoot>
          <tbody>
              <tr>
                  <th>John Smith</th> <td>12000</td>
                  <td>13000</td><td>15000</td>
              </tr>
              <tr>
                  <th>Alice Green</th><td>7000</td>
                  <td>9000</td><td>11000</td>
              </tr>
              <tr>
                  <th>Hugh Brown</th><td>25000</td>
                  <td>23000</td><td>30000</td>
              </tr>
          </tbody>
      </table>
</body>
```

Tables

Sales in first quarter

Salesman	Jan	Feb	Mar
John Smith	12000	13000	15000
Alice Green	7000	9000	11000
Hugh Brown	25000	23000	30000
Totals	34000	45000	56000

Celle su più righe/colonne

```
<body>
  <h1>Tables</h1>
      <table border=2 cellpadding="5" cellspacing="0" width="70%">
      <tr>
          <td>First row <br> First column</td>
          <td colspan="2">First row<br>Second column</td>
      </tr>
      <tr>
          <td colspan="2">Second row <br> First column</td>
          <td rowspan="2">Second row <br> Third column</td>
      </tr>
      <tr>
          <td>Third row <br> First column</td>
          <td>Third row <br> Second column</td>
      </tr>
      </table>
</body>
```

Tables

First row First column	First row Second column	
Second row First column		Second row Third column
Third row First column	Third row Second column	

Elementi e attributi per tabelle

- TABLE: tabella
  - `@border`, `@cellpadding` e `@cellspacing` per indicare bordi e spazi tra le celle e nelle celle (deprecati, meglio usare CSS)
- TR: riga
- TD e TH: cella semplice o di intestazione
  - `@rowspan` e `@colspan` per indicare celle che occupa più righe o colonne
- THEAD, TBODY, TFOOT: raggruppano righe rispettivamente in intestazione, body e footer della tabella
- CAPTION: didascalia
- COLGROUP, COL: specificano proprietà di gruppi colonne

Esercizio

- Scrivere il codice HTML per visualizzare nel browser il seguente contenuto:

Tabella dipendenti

	Distribuzione	Media	
		RAL	Ore
Manager	**2%**	100.000 €	-
Impiegati	**20%**	40.000 €	40
Operai	**78%**	35.000 €	

FORM

Form

- Con i FORM si utilizzano le pagine HTML per inserire valori che vengono poi elaborati sul server. I FORM sono legati ad applicazioni server-side
- Il browser raccoglie dati dall'utente con un form. Crea una connessione HTTP con il server, specificando una ACTION (cioè un applicazione che funga da destinatario) a cui fare arrivare i dati. Il destinatario riceve i dati, li elabora e genera un documento di risposta, che viene spedito, tramite il server HTTP, al browser.
- I controlli tipati e nominati vengono usati per l'inserimento dei dati nei form: campi di inserimento dati, pulsanti, bottoni radio, checkbox, liste a scomparsa, ecc.

```
<h1>Form</h1>
<form method="get" action="http://www.site.com/serverside.py">
    <p>
        <label><i>Name:</i> <input type="text" name="name" value="John" size="15"></label>
        <label><i>Surname:</i> <input type="text" name="surname" value="Smith" size="25"></label>
    </p>
    <p>Gender:
        <label><input type="radio" name="gender" value="m" checked>Male</label>
        <label><input type="radio" name="gender" value="f">Female</label>
        <label><input type="radio" name="gender" value="x">Won't say</label>
    </p>
    <p>Likes:
      <label><input type="checkbox" name="likes" value="art" checked>Art</label>
      <label><input type="checkbox" name="likes" value="cin">Cinema</label>
      <label><input type="checkbox" name="likes" value="com">Comics</label>
      <label><input type="checkbox" name="likes" value="lit" checked>Literature</label>
      <label><input type="checkbox" name="likes" value="cui">Cuisine</label>
    </p>
    <p>
        <label>Nationality: <sele
            <option value="">-- s
            <option value="italic
            <option value="sanmar
            <option value="taiwar
            <option value="turkis
        </select></label>
    </p>
    <p><input type=submit name="s
        <input type=reset name="c
    </p>
</form>
```

Form

Name: John *Surname:* Smith

Gender: Male Female Won't say

Likes: Art Cinema Comics Literature Cuisine

Nationality: Italian

ok ca

-- select one --
Italian
San Marinese
Taiwanese
Turkish

Form

Gli elementi di un form sono:

- `<form>`: il contenitore dei widget del form
  - ***Attributi***:
    - `method`: il metodo HTTP da usare (GET, POST in HTML)
    - `action`: l'URI dell'applicazione server-side da chiamare
- `<input>, <select>, <textarea>`: i widget del form.
  - ***Attributi***:
    - `name`: il nome del widget usato dall'applicazione server-side per determinare l'identità del dato
    - `type`: il tipo di widget (input, checkbox, radio, submit, cancel, etc.)
    - ***N.B.***: Tutte le checkboxes e tutti i radio buttons dello stesso gruppo condividono lo stesso nome.
- `<button>`: un bottone cliccabile (diverso dal submit)
- `<label>`: la parte visibile del widget.

Un passo indietro: caratteri ammessi negli URI

- I caratteri degli URI possono essere:
  - **unreserved**: utilizzabili liberamente, sono alfanumerici caratteri alfanumerici (lettere maiuscole e minuscolo, cifre decimali) e alcuni caratteri di punteggiatura non ambigui -_!~*'()
  - **reserved**: che hanno delle funzioni particolari in uno o più schemi di URI. In questo caso vanno usati direttamente quando assolvono alle loro funzioni e devono essere "*escaped*" per essere usati come parte della stringa identificativa naturale
    - `;/?:@&=+$`
  - **escaped**: caratteri riservarti di cui si fa *escaping* per usarli nelle stringhe identificative (attenzione alle successive elaborazioni degli URI e dei caratteri riservati)

Caratteri Escaped (1)

- I caratteri escaped fanno riferimento alle seguenti tipologie di caratteri:
  - I caratteri non US-ASCII (cioè ISO Latin-1 > 127)
  - I caratteri di controllo: US-ASCII < 32
  - I caratteri unwise: { } | \ ^ []`
  - I delimitatori: spazio<>#%"
  - I caratteri riservati quando usati in contesto diverso dal loro uso riservato
- In questo caso i caratteri vanno posti in maniera escaped, secondo la seguente sintassi:

  %XX, dove XX è il codice esadecimale del carattere

Caratteri Escaped (2)

Esempi:

- **http://www.site.com/?name=Fran%C3%A7ois**
- **http://www.site.com/script.php?msg=Hello%20World**

Nota: i due URI

http://www.alpha.edu/a/b/c/d

http://www.alpha.edu/a/b/c%2Fd

non sono uguali, perché, benché il codice esadecimale corrisponda al carattere "/", nel primo caso esso ha significato gerarchico, e nel secondo fa parte del nome dell'ultima sottoparte della gerarchia, "c/d".

La codifica *application/x-www-form-urlencoded*

- E' usata per spedire coppie `<chiave-valore>` in operazioni di POST da un form HTML
- Si può usare sia nell'URL che nel body della richiesta

```
http://www.site.com/serverside.py?name=John&surname=
   Smith&gender=m&likes=art&likes=lit&nationality=ita
   lian
```

  - i codici non alfanumerici sono sostituiti da '%HH' (HH: codice esadecimale del carattere),
  - gli spazi sono sostituiti da '+',
  - i nomi dei controlli sono separati da '&',
  - il valore è separato dal nome da '='

Un'altra parentesi: MIME

- MIME ridefinisce il formato del corpo di messaggio SMTP per superare alcuni limiti del protocollo
- Qui ci interessa una caratteristica specifica di MIME: un messaggio può contenere **parti di tipo diverso** (es. un messaggio di tipo testo e un attachment binario).
- In questo caso si creano dei sottomessaggi MIME per ciascuna parte e il messaggio MIME complessivo diventa "**multi-parte**", qualificando e codificando in maniera diversa ciascuna sottoparte.

La codifica *multipart/form-data*

- I dati spediti via form possono essere codificati anche con un messaggio **multiparte**, usando la codifica *multipart/form-data*
- Si può usare per qualunque campo del form ma si usa principalmente per spedire file
- Il messaggio è diviso in blocchi di dati, delimitati da *boundary*, e ognuna può avere *Content-type* diverso

```
POST
Accept-Encoding: gzip,deflate
Content-Type: multipart/form-data; boundary="----=_Part_0_1962658601.1561976915821"
MIME-Version: 1.0
Content-Length: 220778
Host: qacomplete.smartbear.com
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.5.2 (Java/1.8.0_181)

------=_Part_0_1962658601.1561976915821
Content-Type: application/zip
Content-Disposition: form-data; name="Title"
```

Esercizio

- Scrivere il codice HTML per visualizzare nel browser il seguente contenuto:

Rispondi alle seguenti domande e clicca su **Invia** per spedire i risultati.

1. Quali tra queste aree di ricerca interessavano Leonardo (possibile scegliere anche piú di una risposta)?
   a. ☐ Matematica
   b. ☐ Fisica
   c. ☐ Numismatica
2. Chi ha inventato il telefono? Marconi

Invia Annulla

Interactive content: form

HTML 5 introduce molte novità per velocizzare, semplificare e controllare l'inserimento dei dati da parte dell'utente

L'oggetto **`input`** è arricchito con due attributi che permettono di controllare il focus e aggiungere suggerimenti agli utenti:

- **`placeholder`**: contiene una stringa che sarà visualizzata in un campo di testo se il focus non è su quel campo
- **`autofocus`**: indica il campo sul quale posizionare il focus al caricamento del form.

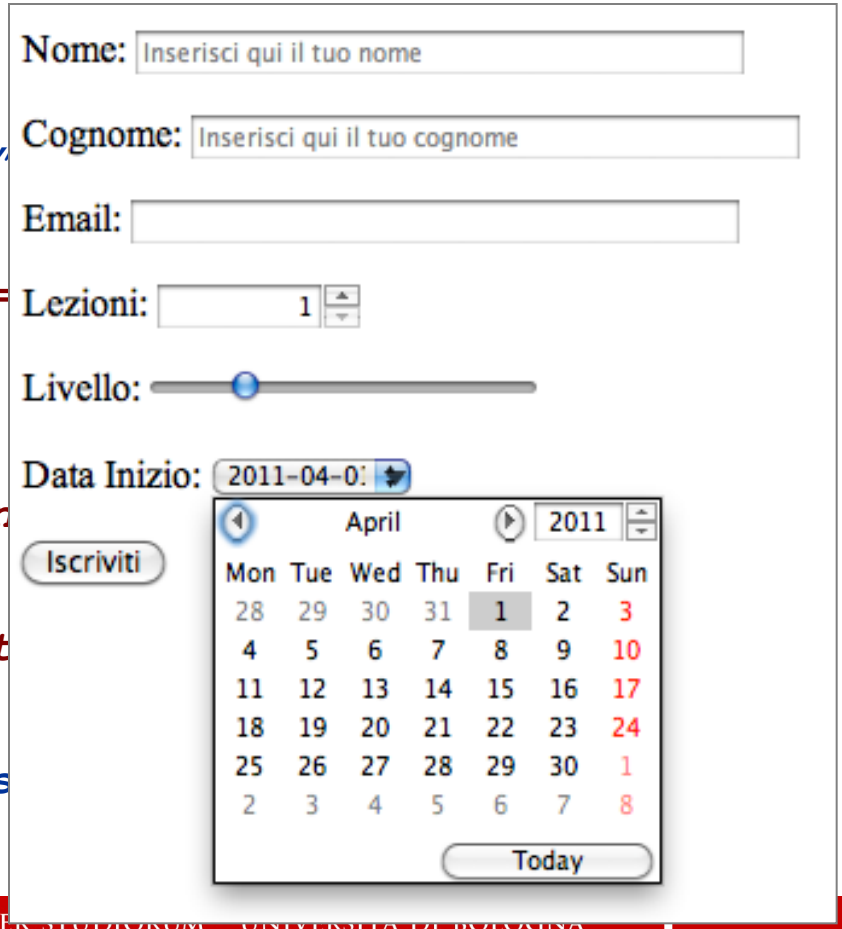
In HTML 5 sono introdotti molti nuovi tipi per l'oggetto **`input`**. I browser forniscono widget diversi nell'interfaccia in base al tipo del campo, indicato nell'attributo **`@type`**

Alcuni nuovi tipi di input

- **`email:`** in fase di validazione il browser verifica se il testo inserito contiene il simbolo '@' e il dominio è (sintatticamente) corretto
- **`url`**: si verifica che il testo inserito segue le specifiche degli URL
- **`number`**: il browser visualizza bottoni per incrementare/decrementare il valore. E' possibile specificare minimo, massimo e unità di modifica in altri attributi
- **`range`**: il browser visualizza uno slider per incrementare o decrementare un valore numerico. E' possibile specificare il valore minimo, massimo e l'unità di modifica.
- **`date`**: richiede al browser la visualizzazione di un calendario tra cui selezionare una data. Esistono vari tipi collegati come **`month`**, **`week`**, **`time`**
- **`search`**: testo renderizzato in modo diverso su alcuni browser (iPhone)
- **`color`**: usato per mostrare una tavolozza di colori, da cui selezionare un codice *RGB*

Esempio

```
<p>Nome: <input name="nome" type="text"
placeholder="Inserisci qui il tuo
autofocus></p>

<p>Cognome: <input name="cognome"
placeholder="Inserisci qui il tuo

<p>Email:<input name="mail" type=

<p>Lezioni: <input type="number"
step="1" value="1"></p>

<p>Livello: <input type="range" n
max="10" step="1" value="3"></p>

<p>Data Inizio: <input name="birt
value="2011-04-01"></p>

<p><input type="submit" value="Is
```



Validazione automatica

HTML 5 prevede anche la possibilità (anzi, è il **default**) di validare un form client-side, senza ricorrere a funzioni Javascript aggiuntive.

Dopo l'evento *submit* i dati inseriti nel form sono verificati in base al tipo di ogni campo (email, URL, etc.). Si può catturare l'evento *invalid* e implementare comportamenti specifici.

E' possibile inoltre indicare i campi obbligatori, tramite l'attributo **`required`** dell'oggetto **`input`**.

L'attributo **`novalidate`** può essere invece usato per indicare di non validare l'intero form o uno specifico campo.

```
<input name="mail" type="email" required>
<input name="url" type="url" novalidate>
```

HEAD

<head>

L'elemento <head> contiene delle informazioni che sono rilevanti per tutto il documento. Esse sono:

- **<title>:** il titolo del documento
- **<link>**: link di documenti a tutto il documento
- **<script>**: librerie di script
- **<style>**: librerie di stili
- **<meta>**: meta-informazioni sul documento
- **<base>**: l'URL da usare come base per gli URL relativi

`<title>`: il titolo del documento

- Contiene semplice testo (non elementi HTML) che definisce il titolo del documento
  - titolo della finestra (o tab) del browser
  - nome illustrativo della pagina nei bookmark
  - nome illustrativo della pagina nei motori di ricerca. Inoltre i motori di ricerca trattano con più importanza il contenuto dell'elemento title rispetto al resto del contenuto della pagina.
- Deve essere unico e non ha attributi specifici, oltre *coreattrs*

```
<head>
    <title>Corso di Tecnologie Web</title>
  ...
```

<link>, <script> e <style>

- Gli elementi SCRIPT e STYLE si possono definire, rispettivamente, blocchi di funzioni di un linguaggio di script e blocchi di stili di un linguaggio di stylesheet.

```
<style type="text/css">
 p {color:red;}
</style>
```

- A volte può esser utile mettere esternamente queste specifiche, e riferirvi esplicitamente.
- In questo caso si usa LINK, che permette di creare un link esplicito e tipato al documento esterno.

```
<link rel="stylesheet" type="text/css"
href="style1.css"/>
```

`<meta>`: meta-informazioni (1)

- Il tag <meta> è un meccanismo generale per specificare meta-informazioni (proprietà) sul documento HTML
- Due attributi principali: **`@name`** e **`@content`**

```
<html>
<head>
  <meta charset="utf-8">
  <title>My document</title>
  <meta http-equiv="Content-Language" content="en-US"/>
  <meta name="keywords" content="HTML, Web, ">
  <meta name="author" content="Angelo Di Iorio">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  …
```

`<meta>`: meta-informazioni (2)

- Tre tipi di meta-informazioni definibili:
  - La codifica caratteri utilizzata nel file:

    ```
    <meta charset="utf-8">
    ```

  - Intestazioni HTTP: la comunicazione HTTP fornisce informazioni sul documento trasmesso, ma il suo controllo richiede accesso al server HTTP. Con il tag META si può invece fornire informazione "stile-HTTP" senza modifiche al server.

    ```
    <meta http-equiv="expires" content="Sat, 23 Mar 2019
    14:25:27 GMT">
    ```

  - Altre meta-informazioni: i motori di ricerca usano le meta-informazioni (ad esempio "Keyword") per organizzare al meglio i documenti indicizzati

<base>: URL relativi ed assoluti

- Ogni documento HTML visualizzato in un browser ha associato un URL ed è possibile esprimere le risorse collegati tramite URL assoluti o relativi
- L'element BASE permette di indicare l'URL di base da usare per risolvere URI relativi, nell'attributo **@href**
- Se non specificato si usa l'URL del documento corrente ma è possibile esplicitarlo

```
<base href="www.cs.unibo.it/corsi/tw/">
```

Embedded content

Embedded content

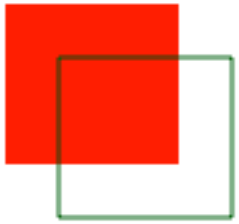
- HTML5 estende notevolmente le possibilità di **includere contenuti multimediali** nelle pagine e **permette di interagire con questi elementi in modo sofisticato**
- Alcuni elementi:
  - `<canvas>`: immagini bidimensionali
  - `<audio>`: file audio
  - `<video>`: file video
  - `<math>`: frammenti MathML per espressioni matematiche
- Il modello ad eventi di DOM è esteso con eventi specifici che permettono la manipolazione degli oggetti
- Noi non guardiamo in dettaglio questi elementi ed API

Canvas

- L'elemento **`<canvas>`** definisce un'area rettangolare in cui disegnare direttamente immagini bidimensionali e modificarle in relazione a eventi, tramite funzioni Javascript.
- La *CanvasAPI* descrivere questi metodi di manipolazione
- La larghezza e l'altezza del canvas sono specificati tramite gli attributi **`width`** e **`height`** dell'elemento **`<canvas>.`**
- Le coordinate **`(0,0)`** corrispondono all'angolo in alto a sinistra.
- Gli oggetti non sono disegnati direttamente sul canvas ma all'interno del **contesto**, recuperato tramite un metodo Javascript dell'elemento **`<canvas>`** chiamato **`getContext()`**

Esempio

```
function draw(){
   var canvas = document.getElementById('c1');
   if (canvas.getContext){
      var ctx = canvas.getContext('2d');
      ctx.fillStyle = "rgb(255,0,0)";
      ctx.fillRect (20, 20, 65, 60);
      ctx.strokeStyle = "rgb(0, 100, 20)";
      ctx.strokeRect (40, 40, 65, 60);
   }
}

<canvas id="c1" onLoad="draw();"
   width="175" height="175">
</canvas>
```



Video e audio

- L'elemento **`<video>`** specifica un meccanismo generico per il caricamento di *file* e *stream* video, più alcune proprietà DOM per controllarne l'esecuzione
- Ogni elemento **`<video>`** in realtà può contenere diversi elementi **`<source>`** che specificano diversi file, tra i quali il browser sceglie quello da eseguire.
- L'elemento **`<audio>`** è usato allo stesso modo per i contenuti sonori
- Non esiste tuttavia una codifica universalmente accettata ma è necessario codificare il video (o audio) in più formati, per renderlo realmente *cross-browser*.

Esempio video

```
<video width="400px" controls autoplay>
  <source src="video.mp4" type="video/mp4"
codecs="avc1.42E01E, mp4a.40.2">
  <source src='video.ogv' type='video/ogg"
codecs="theora, vorbis">
  <track kind="subtitles" src="video.en.vtt"
srclang="en" label="English">
  <track kind="subtitles" src="video.it.vtt"
srclang="it" label=”Italian">
</video>
```

Conclusioni

- Qui abbiamo visto gli elementi principali di HTML e brevemente esempi di contenuti embedded ed API
- Le specifiche di HTML infatti includono un numero sempre maggiore di API per interagire con i contenuti client-side
- Molte altre API e librerie sono disponibili nei browser ma non le guardiamo in dettaglio