ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

HTML (II parte)

Fabio Vitali

CdS in Informatica

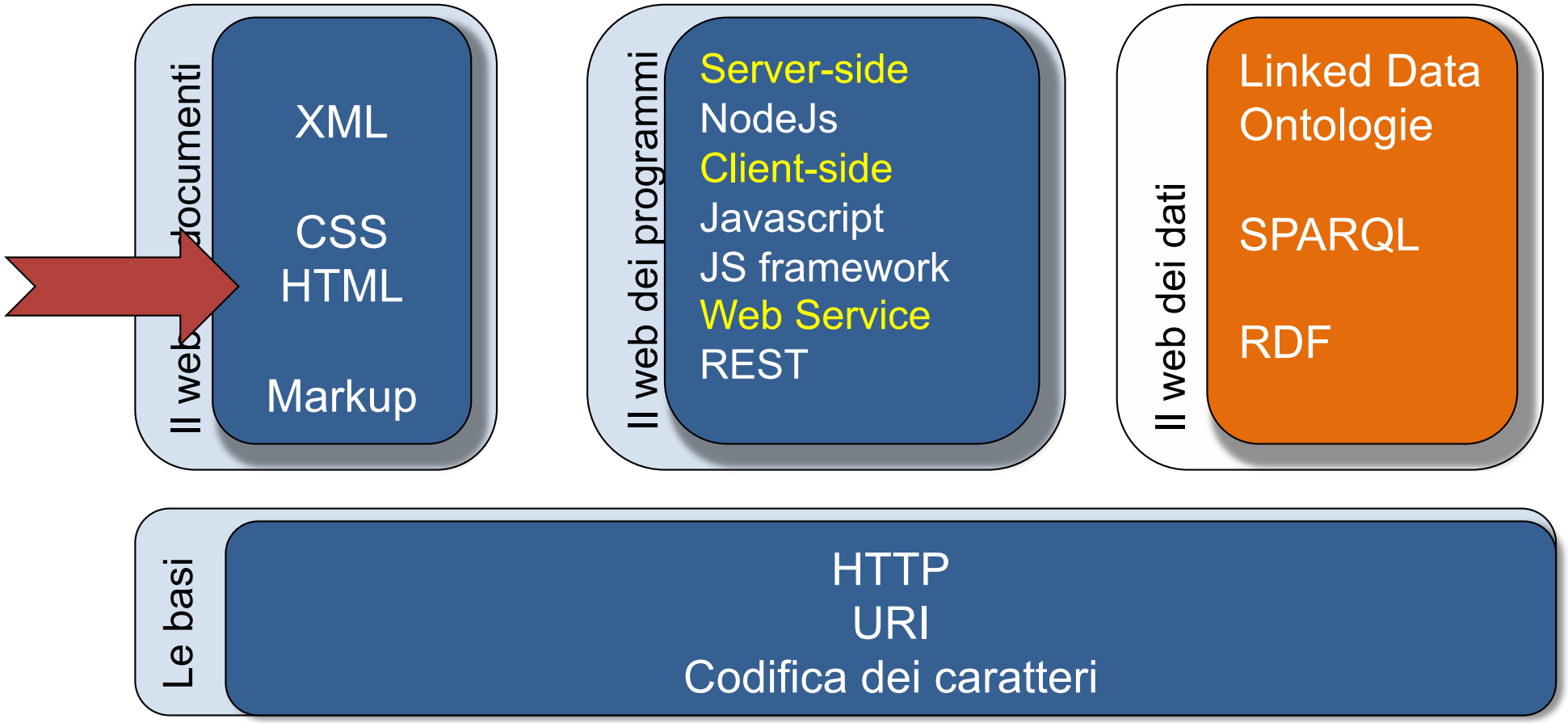
CdS in Informatica per il Management

Introduzione

Oggi esaminiamo in breve:

- HTML (concetti avanzati)
- Il Document Object Model

Argomenti delle lezioni



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

HTML – altri aspetti

HTML – altri aspetti

Adesso parliamo di:

- *La struttura di un documento HTML*
- *Elementi inline*
- *Elementi di blocco e di lista*
- *Elementi generici*
- *Elementi di struttura*
- Link ed immagini
- Tipi di dati
- Entità predefinite
- Peculiarità sintattiche
- Tabelle
- Form
- Il contenuto dell'elemento HEAD
- Il DOM

Link ipertestuali (anchors) (1)

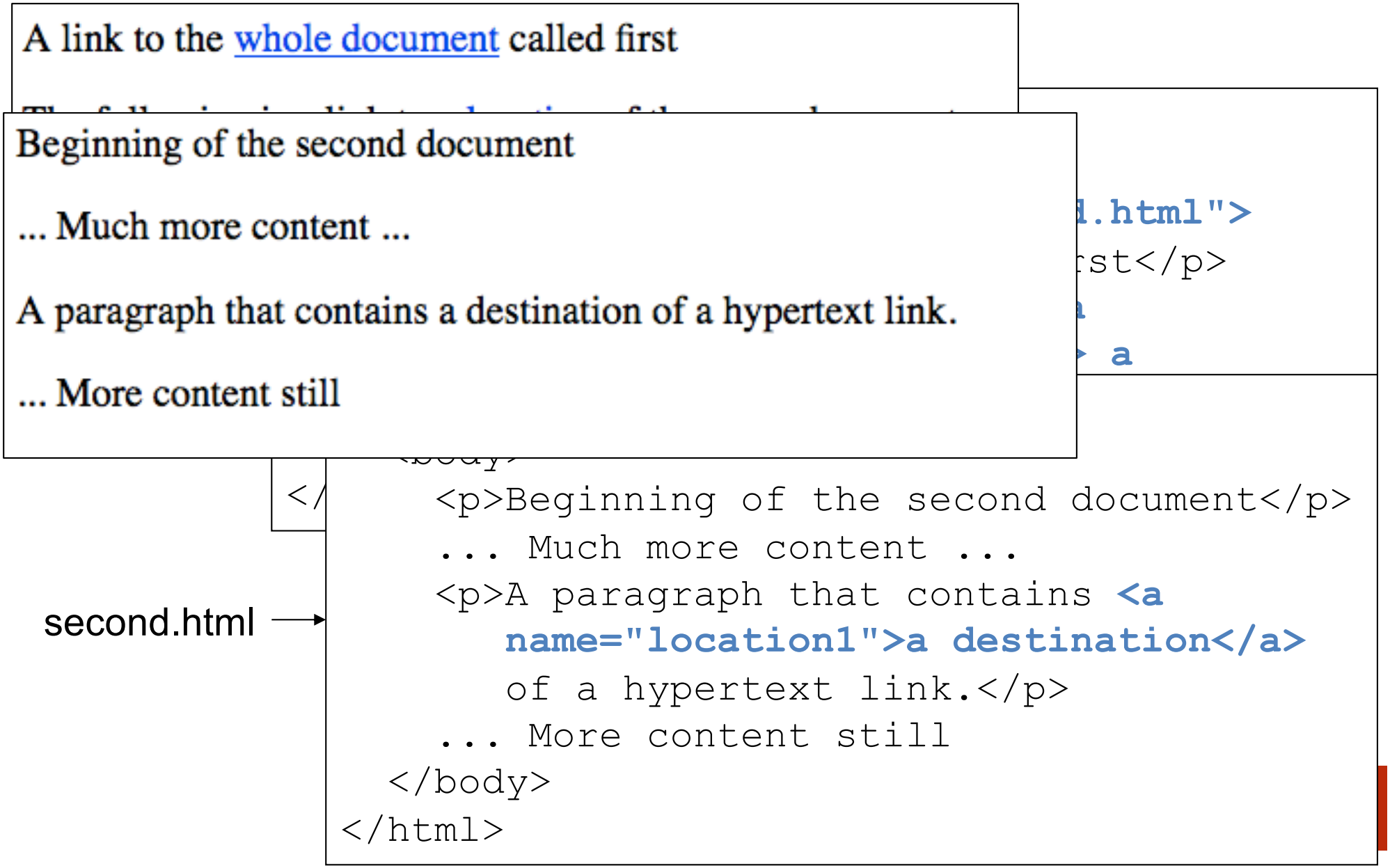
I link sono definiti con elementi <a> (àncore nel documento).

<a> è sintatticamente un elemento inline (dentro ai blocchi, come <b>, <i>, ecc.). 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)



Immagini

Le immagini inline sono definite con l'elemento <img>. Formati tipici: jpeg, gif, png.

<img> è un elemento vuoto, completamente definito dai suoi attributi.

```
<img src="pippo.gif" alt="picture of Pippo">
```

Attributi:

- **src** (required): l'URL della risorsa contenente l'immagine.
- **alt**: testo alternativo se l'immagine non è mostrata
- **name**: un nome per riferirsi all'immagine
- **width**: forza una larghezza in pixel per l'immagine.
- **height**: forza una altezza in pixel per l'immagine.

Immagini (2)

```
<body class="container-fluid">
  <h1>Images</h1>
  <p>Images are managed by the &lt
  The <tt>src</tt> attribute is us
  attribute specifies an alternati
  height in pixels.</p>
  <p>The image can be: </p>
  <ol>
      <li>... in the same folder a
          <img src="flower.jpg" al
      </li>
      <li>... in a different folde
          <img src="images/cake.jp
      </li>
      <li>... on a different serve
          <img src="https://c1.sta
          alt="a cat" height="150"
      </li>
  </ol>
</body>
```

Images

Images are managed by the <img> element, which is empty (no content, no end tag). The `src` attribute is used to specify the URI of the image file. The `alt` attribute specifies an alternative text. The `height` attribute specifies the height in pixels.

The image can be:

1. ... in the same folder as the HTML document:

2. ... in a different folder on the same server:

3. ... on a different server:

Immagini (3)

```
<body class="container-fluid">
  <h1>Images - stretching</h1>
  <p>Specifing height and/or width lets control dimensions simply:
  <ol>
      <li>Specify one dimension: the
          <img src="images/cake.jpg"
      </li>
      <li>Specify both dimensions: t
          proportions and will be st
          <img src="images/cake.jpg"
      </li>
      <li>Specify no dimensions: the
          <img src="images/cake.jpg"
      </li>
  </ol>
</body>
```

Images - stretching

Specifing height and/or width lets control dimensions simply:

1. Specify one dimension: the image will be resized proportionally around that dimension.

2. Specify both dimensions: the image will be resized regardless of the original proportions and will be stretched.

3. Specify no dimensions: the image will be shown in the original dimensions.

Immagini responsive

Attraverso l'attributo `srcset` è possibile indicare più risorse per la stessa immagine, da usare alternativamente.

A seconda dello user agent scegliamo immagini con dimensioni (e quindi risoluzioni) diverse, e non dobbiamo affidarci al ridimensionamento algoritmico dell'immagine con l'inevitabile perdita di dettaglio.

```
<img srcset="kitten-480w.jpg 480w, kitten-1080w.jpg 1080w"
     sizes="(max-width: 800px) 480px, 1080px"
     src="kitten-800w.jpg"
     alt="A pretty kitten">
```

- L'attributo `srcset` indica quali risorse usare e la dimensione "reale" di ciascuna risorsa.
- L'attributo `sizes` indica quali tipi di schermo associare a ciascuna risorsa.
- L'attributo `src` è un fallback nel caso che il browser non conosca `srcset`.

L'elemento <figure>

```
<h1>Figure</h1>
<figure>
    <img src="images/cake.jpg" alt="A cake" width="150">
    <figcaption>A cake</figcaption>
</figure>
```

Figure

A cake

Embedding

Simili a `<img>`, permettono di arricchire la pagina HTML con molteplici tipi di contenuti alternativi.

- `<video>` per fare embedding di un video
- `<audio>` per fare embedding di un audio
- `<object>` per fare embedding di oggetti multimediali. Richiede di specificare un plugin a cui affidare la visualizzazione del dato.
- `<embed>` per fare embedding di oggetti multimediali. Non richiede plug-in.
- `<iframe>` per fare embedding di una pagina HTML all'interno di un'altra pagina HTML.

```
<iframe width="300" height="200"
src="https://www.openstreetmap.org/export/embed.html?bbox=1
1.320%2C44.490%2C11.370%2C44.500"></iframe>
```

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.

Tabelle (2)

```
<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

Tabelle (3)

```
<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

Un altro tipo di tabella

- Le celle possono occupare più righe o più colonne.
- Ogni colspan e rowspan toglie la cella corrispondente nella colonna o nella riga successiva.
- Utili per fare tabelle di layout (fortemente criticate per ragioni di accessibilità)

Un altro tipo di tabella (2)

```
<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	

Form

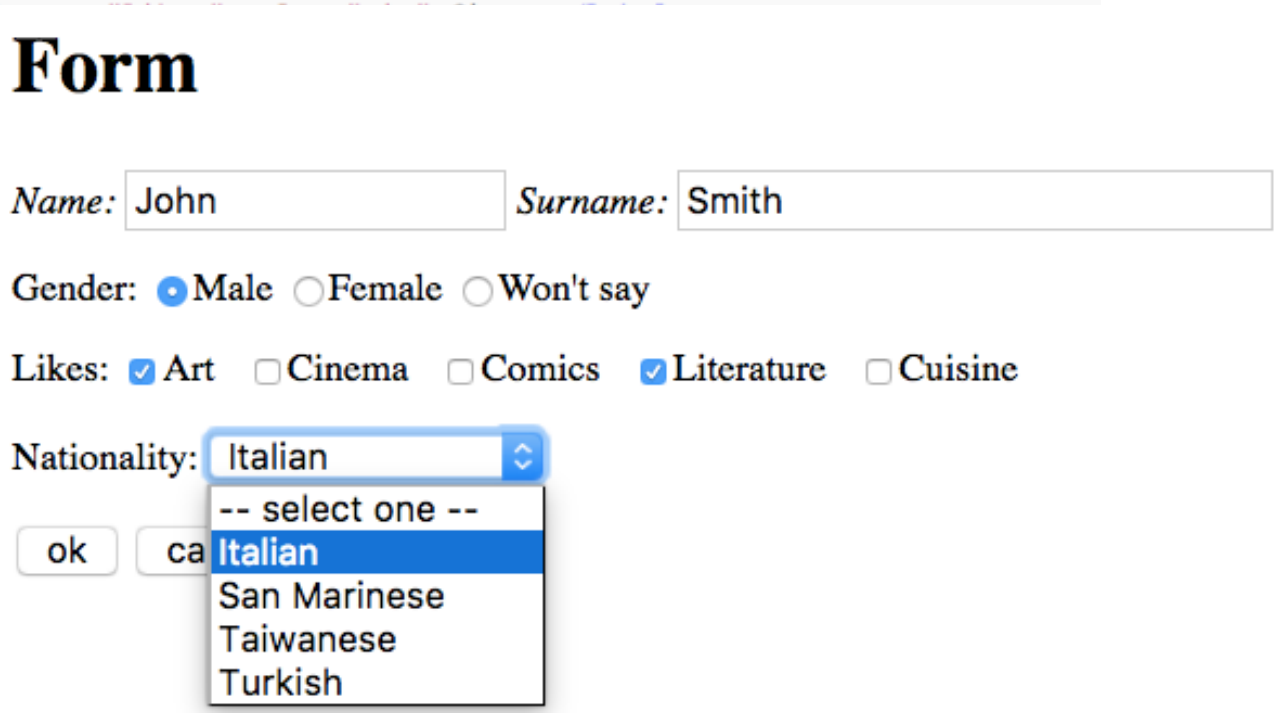
Con i FORM si utilizzano le pagine HTML per inserire valori che vengono poi elaborati sul server. I FORM sono legati ad applicazioni server-side

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.

Form (2)

```
<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"
      <label><input type="checkbox"
      <label><input type="checkbox"
      <label><input type="checkbox"
    </p>
    <p>
        <label>Nationality: <select
            <option value="">-- sel
            <option value="italian"
            <option value="sanmarin
            <option value="taiwanes
            <option value="turkish"
        </select></label>
    </p>
    <p><input type=submit name="sub
        <input type=reset name="can
    </p>
</form>
```



Form (3)

Gli elementi di un form sono:

- `<form>`: il contenitore dei widget del form
  - ***Attributi***:
    - `method`: il metodo HTTP da usare (GET, POST)
    - `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 part del widget.

Interactive content: form

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

L'oggetto **`input`** è arricchito con molti 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
- **`required`**: il form non può essere sottomesso senza una valore per questo elemento.
- **`readonly`**: è un elemento del form non modificabile.
- **`list`**: il valore deve appartenere ad uno dei valori ammessi (specificati in un elemento **`<datalist>`** associato.

```
<input type="text" list="province">
<datalist id="province">
  <option value="BO">Bologna</option>
  <option value="MO">Modena</option>
  <option value="FC">Forlì Cesena</option>
</datalist>
```

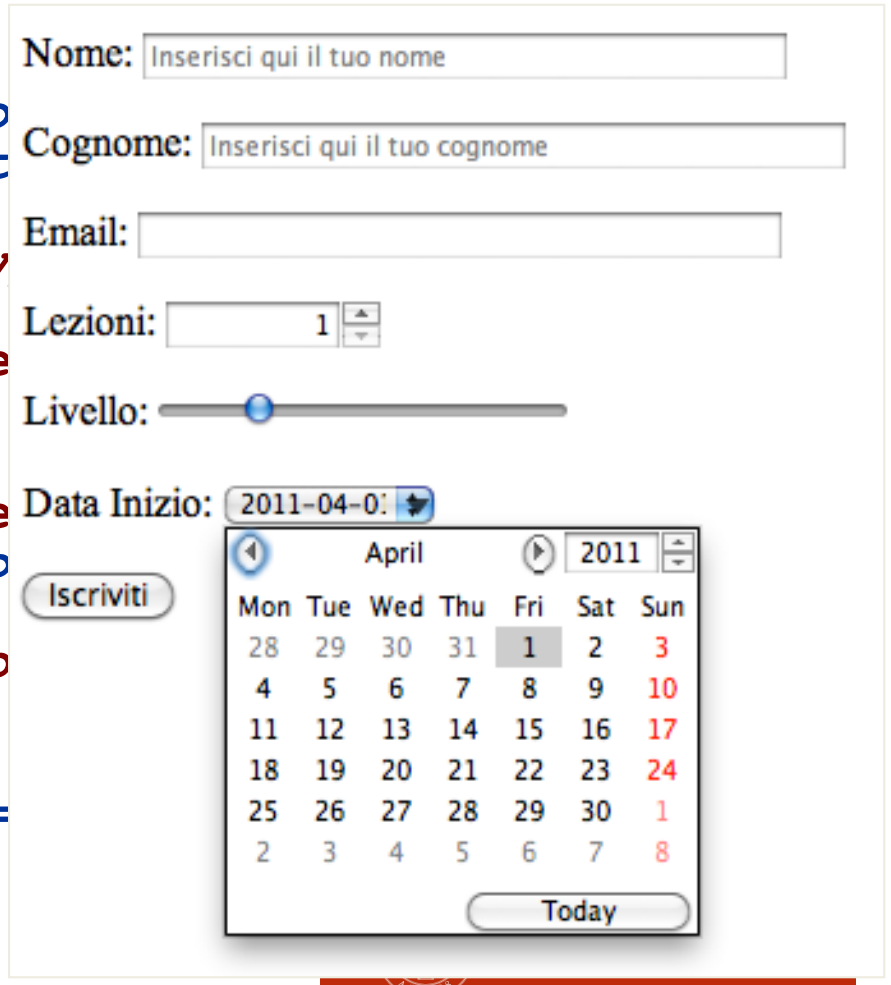
Alcuni nuovi tipi di input

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

- **`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 il valore minimo, massimo e l'unità di modifica in altri attributi di **`input`**.
- **`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`** (orario), etc.
- **`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 nome" size="40"
autofocus></p>

<p>Cognome: <input name="cogno
placeholder="Inserisci qui il t

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

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

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

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

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



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Approfondimenti sulla sintassi di HTML

Attributi globali

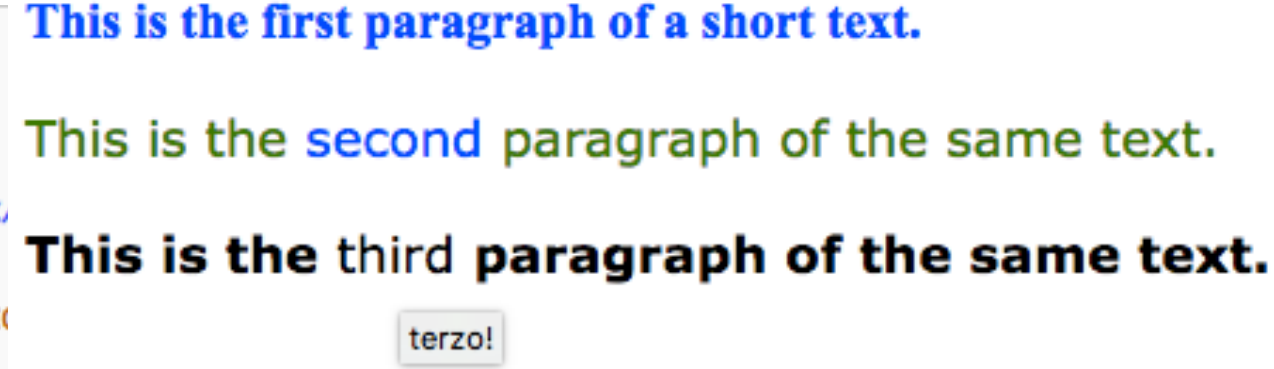
Sono attributi globali quelli definiti su tutti gli elementi del linguaggio HTML.

Essi 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)
- **Class**: una lista (separata da spazi) di nomi di classe (per attribuzione semantica e di stile CSS)

- **Style**: un breve stile CSS associato al singolo elemento
- **Title**: un testo secondario associato all'elemento (per accessibilità e informazioni aggiuntive)

Attributi globali (2)




```
<!DOCTYPE html>
<html>
  <head>
    <title>Table elements </
    <meta charset="utf-8">
    <meta name="viewport" c
    <style>
        #p1 { font-family: "Times New Roman", "Times", serif;}
        #p2, #p3 { font-family: "Verdana", "Arial", sans-serif; }
        .first { color: blue; }
        .second {color: green; }
        .special {font-weight: bold; }
        .notspecial {font-weight: normal; }
    </style>
  </head>
  <body>
    <p id="p1" class="first special">This is the first paragraph of a short text.</p>
    <p id="p2" class="second">This is the <span id="s1" class="first">second</span>
    paragraph of the same text.</p>
    <p id="p3" class="special">This is the <span id="s2" class="notspecial"
    title="terzo!">third</span> paragraph of the same text.</p>
  </body>
</html>
```

Attributi globali: altri tipi

Gli attributi i18n *(internationalization)* garantiscono l'internazionalizzazione del linguaggio e la coesistenza di alfabeti

- **lang**: l linguaggio umano (RFC1766 a due caratteri: it, en, fr, etc.)
- **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 interattività permettono di dare indicazioni su come rendere interattivi gli elementi corrispondenti (per form o accessibilità):

- **accesskey** (tasto di tastiera per focalizzare l'elemento)
- **autofocus** (l'elemento viene focalizzato all'avvio)
- **tabindex** (l'elemento è raggiungibile via tab dopo *n* pressioni)
- **inputmode** (quale tastiera visualizzare quando l'elemento è focalizzato)

Gli attributi di evento permettono di associare script a particolari azioni sul documento e sui suoi elementi:

- **Onclick**, **ondoubleclick**, **onmouseover**, **onkeypress**, ecc.

Attributi data-

Attributi personalizzati che possono essere utilizzati da applicazioni e script Javascript senza inquinare lo spazio dei nomi.

Hanno la forma data-xxx. Ad es. in Bootstrap:

```
<button id="first" data-bs-toggle="modal" data-bs-target="#exampleModal">
    show modal
</button>
```

Non tutti i caratteri sono leciti, le maiuscole non sono lecite, i trattini vengono automaticamente convertiti in maiuscole nel dataset.

Accedibile via CSS esattamente come scritto:

- `button['data-bs-toggle'] { color: blue; }`

Accedibile via Javascript con l'array dichiarativo `element.dataset.xxx` .

```
let a = document.getElementById('first').dataset['bsToggle']
```

vale "modal"

Attributi ARIA

HTML è accessibile per default.

Tuttavia nell'uso comune esistono tante cattive abitudini che creano pagine web perfettamente funzionanti per i non disabili ma impossibili da usare per i disabili.

Ad esempio:

```
<button class="button" onclick="doSomething()">Clicca qui</button>
```

e

```
<span class="button" onclick="doSomething()">Clicca qui</span>
```

sono visivamente identici ma il secondo non è accessibile dai non-vedenti.

La specifica WAI-ARIA (*Web Accessibility Initiative - Accessible Rich Internet Applications*) fornisce molte tecniche per rendere accessibili pagine HTML

- Attributo `role` per caratterizzare la semantica attesa dell'elemento
- Attributo `tabindex` per specificare la possibilità di selezionare l'elemento con la tastiera
- Attributi `aria-*` per specifiche funzionalità ai vari elementi a seconda del loro scopo.

Dedicheremo due lezioni ad ARIA nella seconda metà del corso. Questo elemento è accessibile:

```
<span role="button" class="button" onclick="doSomething()"
tabindex="0">Clicca qui</span>
```

Entità in HTML

HTML definisce un certo numero di entità per quei caratteri che sono proibiti perché usati in HTML (<, >, &, “, ecc.) o proibiti perché non presenti nell’ASCII a 7 bit.

HTML	Shown as
`libert&agrave;`	libertà
`&copy; 2018 Fabio Vitali`	© 2018 Fabio Vitali
`Fran&ccedil;ois`	François
`5 > 3 & 5 < 12`	5 > 3 & 5 < 12

Ad esempio:

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

ecc.

In aggiunta, si possono usare entità numeriche:

- Libertà Libertà
- 你好 你好

Tipi di dati: i colori

In molte situazioni (sfondi, caratteri, ecc.) è possibile specificare un colore. HTML fornisce due modi per farlo:

- **Codice RGB prefissato da un carattere di hash**. Si usano due caratteri esadecimali ciascuno per esprimere la quantità di Rosso, Verde e Blu del colore (00 significa assenza, FF significa presenza massima). E' possibile descrivere 4096 colori diversi.
- **Nome**: sono definiti 16 nomi di colori: black, silver, gray, white, maroon, red, purple, fuchsia, green, lime, olive, yellow, navy, blue, teal, aqua. N.B. Microsoft definisce 256 nomi di colori, che IE accetta, ma che Netscape ignora. Questo causa incompatibilità tra i browser.

Tuttavia HTML 4 depreca l'uso esplicito di colori nel documento HTML, e suggerisce di usare i fogli di stile.

- `<font color="#FF0000">testo in rosso</font>`
- `<body bgcolor="#008080">sfondo teal</body>`
- `<td bgcolor="yellow">sfondo giallo</td>`

Tipi di dati: le lunghezze

HTML usa le lunghezze per tutti gli oggetti con una presenza sulla pagina di dimensioni determinate (immagini, tabelle, frame, ecc.) Si usano tre tipi di lunghezze:

- ***Pixel***: una dimensione in punti di schermo. È un numero assoluto.

```
<img src="immagine.gif" width="150">
```

- ***Percentuali***: una dimensione in proporzione alla dimensione del contenitore. È un numero seguito da un carattere %.

```
<table width="100%">…</table>
```

- ***Multi-lunghezze***: una sequenza di valori di lunghezza. In questo caso il carattere '*' divide la quantità restante in parti uguali (dopo aver tolto le lunghezze esplicite in pixel e percentuale). È una lista separata da virgole di valori numerici assoluti. La si usa ad esempio nei gruppi di colonne di una tabella.

```
<colgroup width="150,25%,*,*">
```

Peculiarità sintattiche di HTML

Maiuscolo/minuscolo:

- HTML non è sensibile al maiuscolo/minuscolo (XHTML lo è e vuole tutto in minuscolo). Editor di antica concezione scrivevano i tag in maiuscolo (per meglio distinguerli dal testo), i nuovi in minuscolo (per meglio transitare verso XHTML).

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.

Estensioni del linguaggio

- I browser permettono (ignorandoli), mentre i validatori segnalano elementi e attributi che non appartengono al linguaggio

```
<p pluto="paperino"><pippo>testo</pippo></p>
```

Bizzarrie sintattiche

- Dovute perlopiù al fatto che HTML è un documento SGML, e SGML permette queste cose. Viceversa, XHTML è un documento XML, che queste cose non le permette.

Bizzarrie sintattiche di HTML

(rimosse in XHTML, rimesse in HTML LS) (1)

Virgolette negli attributi

- Tutti i valori degli attributi che sono TOKEN (cioè parole di caratteri e numeri senza spazi e che inizino con un carattere) possono essere posti senza virgolette

```
<p align=center>testo al centro</p>
```

dovrebbe essere

```
<p align="center">testo al centro</p>
```

Bizzarrie sintattiche di HTML

(rimosse in XHTML, rimesse in HTML LS) (2)

Assenza dei tag

Alcuni tag (ad esempio <HTML>, <BODY>), possono essere omessi

```
<p>Questo &egrave; un documento valido in HTML,
ma non in XHTML</p>
```

dovrebbe essere

```
<html>
  <body>
    <p>Questo &egrave; un documento valido sia
in HTML, sia in XHTML</p>
  </body>
</html>
```

Bizzarrie sintattiche di HTML

(rimosse in XHTML, rimesse in HTML LS) (3)

Omissibilità del tag di chiusura

Di alcuni elementi (ad esempio, <P>, <LI>, <OPTION>) può essere omesso il tag finale (è implicitamente dedotto dal contesto)

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

dovrebbe essere

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

Bizzarrie sintattiche di HTML

(rimosse in XHTML, rimesse in HTML LS) (4)

Attributi di solo valore

Alcuni attributi (nowrap, selected, etc.) vengono omessi, scrivendo solo il valore equivalente:

```
<select>
  <option>prima voce
  <option selected>seconda voce
  <option>terza voce
</select>
```

dovrebbe essere

```
<select>
  <option>prima voce</option>
  <option selected="selected">seconda
voce</option>
  <option>terza voce</option>
</select>
```

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Altre caratteristiche

- <head>
- <object>
- <canvas>
- web components

I tag di **<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

I tag di **<head>**:

<title>: il titolo del documento

L'elemento <title> contiene semplice testo (non può contenere elementi HTML) che definiscono il titolo del documento.

Il contenuto dell'elemento <title> ha tre scopi:

- Viene posto come titolo della finestra nei sistemi operativi a finestra
- Viene utilizzato come nome illustrativo della pagina nei bookmark
- Viene utilizzato come 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.

I tag di `<head>`:

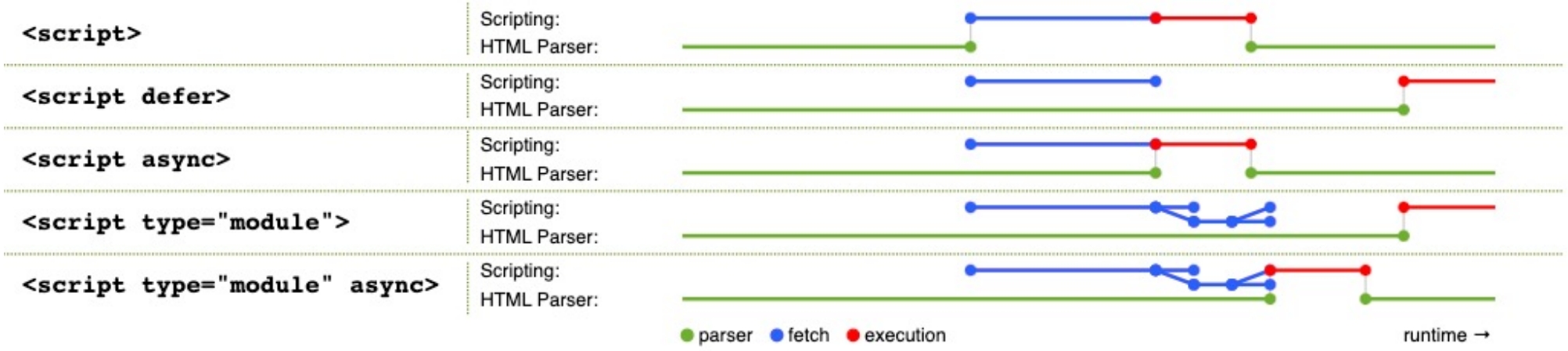
`<script>`: codice e dati interni ed esterni

Con il tag `<script>` si definiscono blocchi di codice di un linguaggio di script. A volte può esser utile mettere questo codice in una risorsa a parte (es. condivisa), e riferirvi esplicitamente.

```
<script type="application/javascript"> ... </script>
<script type="application/javascript" src="lib/script.js"></script>
```

Attributi:

- `type="module"`: usa la sintassi JS dei moduli introdotta per facilitare la componentizzazione degli script Javascript (cambia la sintassi JS)
- `defer`: scarica lo script in parallelo ma lo esegue alla fine del caricamento
- `async` : scarica lo script in parallelo ma lo esegue appena arriva disponibile



I tag di `<head>`:

`<link>`, `<style>`: rappresentazione dello stile

Con i tag `<link>` e `<style>` si possono definire, rispettivamente, blocchi blocchi interni ed esterni di stili di un linguaggio di stylesheet.

A volte può esser utile mettere esternamente queste specifiche, e riferirvi esplicitamente.

In questo caso si usa il tag LINK, che permette di creare un link esplicito al documento esterno di script e/o di stili.

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

I tag di **<head>**:
<meta>: meta-informazioni (1)

Le meta-informazioni sono informazioni sul documento, piuttosto che informazioni del documento. Il tag <meta> è un meccanismo generale per specificare meta-informazioni sul documento HTML.

Ci sono tre tipi di meta-informazioni definibili con il tag <meta>:

- 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.

I tag di **<head>**:
<meta>: meta-informazioni (2)

```
<html>
<head>
  <meta charset="utf-8">
  <title>An example of a head element with meta tags</title>
  <meta http-equiv="expires" content="23 Mar 2019 01:00:00"/>
  <meta http-equiv="Content-Language" content="en-US"/>
  <meta name="DC.Creator" content="Fabio Vitali">
  <meta name="DC.Title" content="Using meta tags">
  <meta name="DC.Date" content="2019-03-04">
  <meta name="DC.Format" content="text/html">
  <meta name="DC.Language" content="en">
  <meta name="viewport" content="width=500, initial-scale=1">
</head>
<body>
  ...
</body>
</html>
```

I tag di **<head>**:
<meta>: meta-informazioni (3)

```
<meta name="viewport" content="width=500, initial-scale=1">
```

Il **<meta name="viewport">** è una proposta Apple in corso di standardizzazione per rendere pagine non progettate per il mobile leggibili anche su un device mobile.

Attraverso questo <meta> imponiamo al device in questione (tipicamente uno smartphone) le caratteristiche da utilizzare per il viewport invece di quelle fisiche, in modo che le pagine non risultino illeggibili sui browser e alla peggio scrollando si acceda alla parte di documento nascosto.

Nell'esempio sopra, si impone di considerare la larghezza della pagina a 500 px e considerare i pixel CSS della pagina tali e quali ai pixel dello schermo.

I tag di <head>:
<base>: URL relativi ed assoluti

Ogni documento HTML visualizzato in un browser ha associato un URL. Questo può appartenere allo schema di naming http://, ftp://, o anche file://. Tipicamente sono schemi gerarchici.

Spesso accade che esistano degli oggetti dipendenti dalla pagina (immagini, stili, script, applet, link a pagine secondarie, ecc.), che appartengono allo "stesso dominio" della pagina di partenza.

E' data allora possibilità, nello specificare l'URL della risorsa secondaria, di affidarsi ad un URL relativo, che si basa sull'URL del documento di partenza.

<object>: embedded content

HTML5 estende notevolmente le possibilità di integrazione di contenuti multimediali nelle pagine

Elementi come **<canvas>**, **<audio>**, **<video>**, **<math>** permettono di “includere” contenuti con i quali, ed è questa la vera novità e il punto di forza, è possibile interagire in modo avanzato

Il modello ad eventi di DOM è esteso con eventi specifici che permettono la costruzione di applicazioni sofisticate client-side. Agli eventi di aggiungono sofisticate API di manipolazione degli oggetti

Guardiamo in particolare:

- Canvas
- Embedding di contenuti audio e video

Canvas

Una delle più importanti innovazioni di HTML 5 è la possibilità di disegnare direttamente sulla pagina interagire con gli oggetti multimediali

L'elemento **`<canvas>`** definisce un'area rettangolare in cui disegnare direttamente immagini bidimensionali e modificarle in relazione a eventi, tramite funzioni Javascript.

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.

HTML Canvas 2D Context

Gli oggetti non sono disegnati direttamente sul canvas ma all'interno del **contesto**, recuperato tramite un metodo Javascript dell'elemento `<canvas>` chiamato `getContext()`

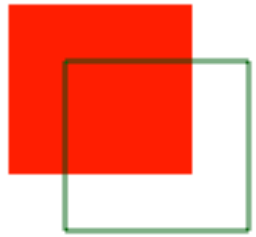
Questo metodo è parte di una vasta libreria utile per disegnare figure, colorarle, trasformarle, etc.

Questa API, inizialmente nota come *canvas API* e tuttora integrata con questo nome in "HTML Living Standard", è diventata una specifica W3C distinta da HTML, anche se usa le stesse definizioni core, le stesse classi e lo stesso background di riferimento

"HTML Canvas 2D Context" è una Recommendation W3C dal 19 Novembre 2015

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

HTML 5 permette di includere video in una pagina senza richiedere plug-in esterni (Flash, Real Player, Quicktime)

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 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>
```

WebComponents

Prendendo ispirazione dai sistemi a componenti come Angular, React e Vue, anche la sintassi standard di HTML ha introdotto i *web component*.

- *Custom element*: posso estendere il vocabolario degli elementi di HTML con quelli che più mi piacciono.
- *Shadow DOM*: posso creare un mini-DOM protetto a cui non si applicano script, stili e eventi del documento principale
- *HTML template*: posso associare ad un custom element un frammento HTML che va automaticamente a rimpiazzare il custom element di riferimento.

Per esempio: `<my-modal id="m1" title="Hello world">Some content</my-modal>`

può essere usato con un template come:

```
<div class="modal" tabindex="-1">
  <div class="modal-header">
    <h5 class="modal-title"><slot name="title"></h5>
    <button type="button" class="btn-close" aria-label="Close"></button>
  </div>
  <div class="modal-body"><slot></slot></div>
  <div class="modal-footer">
    <button type="button">Close</button>
  </div>
</div>
```

Rispetto ai sistemi a componenti tradizionali però è un modello un po' macchinoso. Ne riparliamo più avanti.

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Il Document Object Model (DOM)

Document Object Model

Adesso parliamo di:

- Struttura programmatica del documento HTML
- I principali oggetti del DOM
- Manipolazione del DOM

Una questione spinosa: il parsing di HTML 5

Il WhatWG ha definito una volta per tutte i meccanismi di parsing ed interpretazione del codice HTML.

Le specifiche “HTML Living Standard” definiscono un algoritmo (piuttosto complesso) per fare il parsing di qualunque documento HTML, anche dei documenti mal formati, sulla base di ciò che i browser già facevano.

In realtà dalla prospettiva WAI questi documenti non sono propriamente “mal formati” ma semplicemente “non strict”. Sono validi a tutti gli effetti, tanto quanto i documenti XHTML!

Pragmaticamente potremmo dire: “l’importante è arrivare ad una struttura dati in memoria unica su cui costruire applicazioni”. E' a questo scopo che la vera attenzione da parte del WHATWG è la costruzione di una struttura dati chiamata Document Object Model (DOM), a cui (in maniera più o meno precisa) sia possibile arrivare a partire dalla stringa HTML e da cui si possa generare nuovamente una altra stringa HTML.

Un problema risolto o uno in più?

Aver uniformato l'algoritmo di parsing non è un deterrente per creare pagine "ben formate", al contrario lascia maggiore libertà e margine di errore agli sviluppatori.

Se esistono pagine errate è perché, in primo luogo, ci sono stati sviluppatori che hanno creato pagine non valide e usato il "funziona sul mio browser" come strumento di convalida.

Questa brutta pratica è nata dal fatto che, in mancanza di regole semplici (SGML) i browser hanno sempre accettato tutti i documenti e fatto del loro meglio per visualizzarli.

Il vero problema è che in questo modo prolifereranno le pagine non corrette e sarà più complesso estrarre i dati e implementare manipolazioni automatiche dei contenuti.

La cosa si complicherà ancora di più quando (coi sistemi a componenti) lo stesso concetto di markup perderà valore a vantaggio di sintassi miste Javascript/Markup/CSS/whatever create ad hoc e mai standardizzate.

Il Document Object Model

Il Document Object Model è un **interfaccia di programmazione (API)** per documenti sia HTML sia XML.

Definisce la struttura logica dei documenti ed il modo in cui si accede e si manipola un documento.

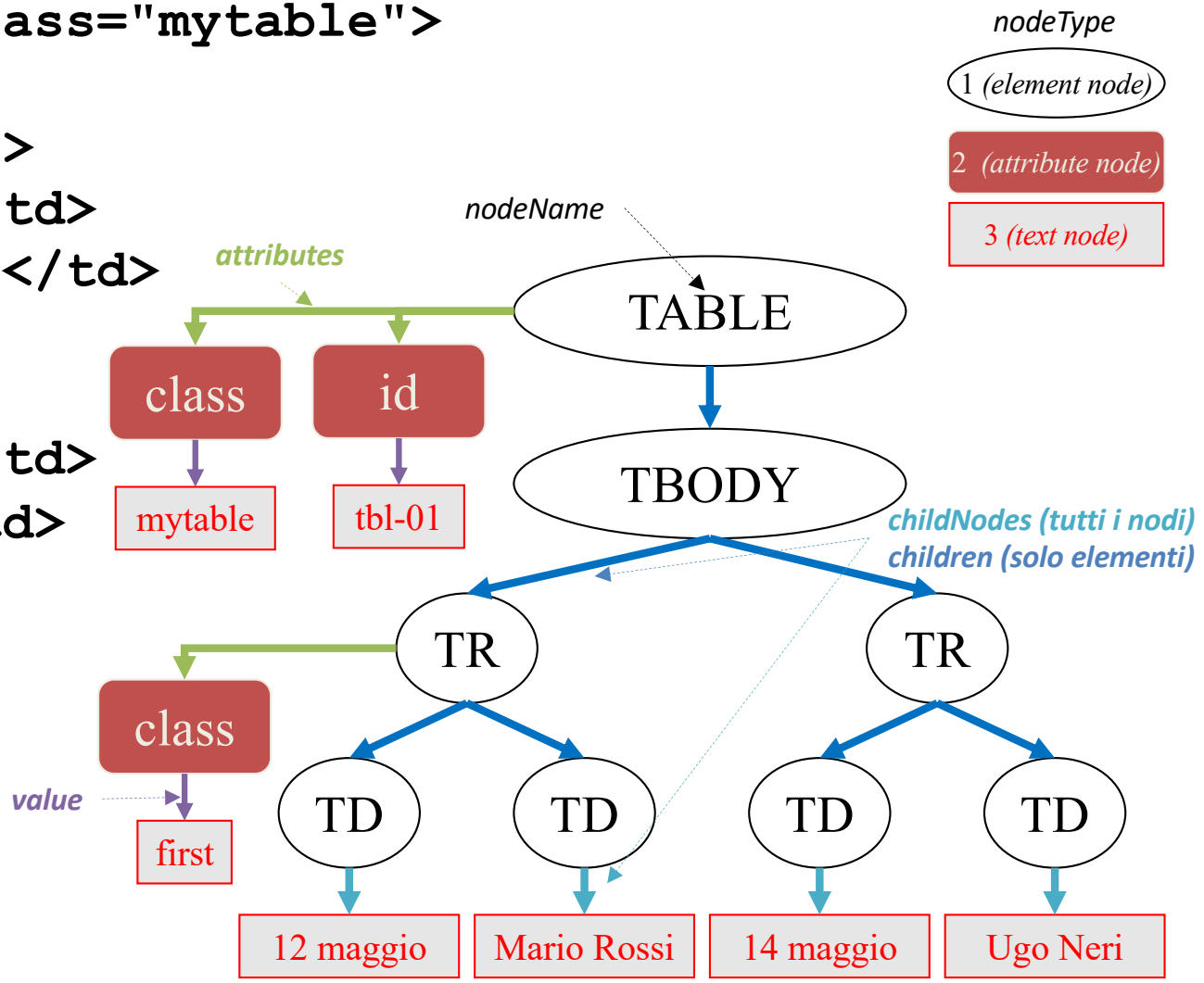
Utilizzando DOM i programmatori possono costruire documenti, navigare attraverso la loro struttura, e aggiungere, modificare o cancellare elementi.

Ogni componente di un documento HTML o XML può essere *letto, modificato, cancellato o aggiunto* utilizzando il Document Object Model.

Struttura di un DOM

Sia dato un documento HTML come il seguente:

```
<table id="tbl-01" class="mytable">
  <tbody>
    <tr class="first">
      <td>12 maggio</td>
      <td>Mario Rossi</td>
    </tr>
    <tr>
      <td>14 maggio</td>
      <td>Ugo Neri</td>
    </tr>
  </tbody>
</table>
```



Oggetti del DOM

Il core del DOM definisce alcune classi fondamentali per i documenti HTML e XML, e ne specifica proprietà e metodi.

La classe principale di DOM è *DOMNode*, di cui la maggior parte delle altre classi è una sottoclasse.

Le classi principali definiti nel DOM sono:

- DOMDocument : il documento di cui si sta parlando
- DOMElement: ogni singolo elemento del documento
- DOMAttr: ogni singolo attributo del documento
- DOMText: ogni singolo nodo di testo del documento
- DOMComment, DOMProcessingInstruction, DOMCDATASection, DOMDocumentType, ecc.

DOM Node

DOMNode specifica i metodi per accedere a tutti gli elementi di un nodo di un documento, inclusi il nodo radice, il nodo documento, i nodi elemento, i nodi attributo, i nodi testo, ecc. Semplificando:

membri

- nodeName *(uppercase string)*
- nodeType *(number)*
- children *(array)*
- childNodes *(array)*
- parentNode *(elementNode)*
- attributes *(array)*

metodi

insertBefore()
replaceChild()
removeChild()
appendChild()
hasChildNodes()
hasAttributes()

N.B.: non è vero! Sono HTMLCollection, NodeList e NamedNodeMap rispettivamente, con metodi aggiuntivi, ma Array.from() restituisce un array.

DOM Document

DOMDocument specifica i metodi per accedere al documento principale, tutto compreso. È equivalente alla radice dell'albero (non all'elemento radice!!!). Semplificando:

membri	**metodi**
docType	createElement()
documentElement	createAttribute()
	createTextNode()
	getElementsByTagName()
	getElementById()

DOM Element

DOMElement specifica i metodi e i membri per accedere a qualunque elemento del documento. Semplificando:

membri	**metodi**
nodeName	getAttribute()
	setAttribute()
	removeAttribute()

… e analogamente per le altre classi ed interfaccia del DOM.

Selettori in DOM

I metodi standard in DOM per accedere ai nodi di un documento sono essenzialmente:

- **getElementById**: solo ovviamente se l'elemento ha un id
- **getElementsByName**: se l'elemento ha un attributo name
- **getElementsByTagName**: tutti gli elementi con nome specificato

Il successo di JQuery ha portato nel tempo a proporre ed implementare in DOM anche altri nuovi selettori:

- **getElementsByClassName**; cerca tutti gli elementi di classe specificata
- **querySelector**: accetta un qualunque selettore CSS e restituisce il primo elemento trovato - del tutto equivalente a $()[0] in JQuery
- **querySelectorAll**: accetta un qualunque selettore CSS e restituisce tutti gli elementi trovati - del tutto equivalente a $() in JQuery

Esempio di manipolazione del DOM (in Javascript)

oppure anche
`document.querySelector("#table1")`

```
var table = document.getElementById('table1');
var nodes = table.childNodes()
for (var x=0; x<.nodes.length; x++)
  { ... }

var row = document.createElement("tr");
var item = document.createElement("td");
var text = document.createTextNode("testo della cella")
item.appendChild(text);
row.appendChild(item);
table.appendChild(row);
```

innerHTML e outerHTML

Il DOM per HTML (non generale!) permette di leggere/scrivere interi elementi, trattandoli come stringhe:

- **innerHTML**: legge/scrive il contenuto di un sottoalbero (escluso il tag dell'elemento radice)
- **outerHTML**: legge/scrive il contenuto di un elemento (incluso il tag dell'elemento radice)

```
// sia dato un HTML:
<div id="p1">
  <p>Paragrafo!</p>
</div>

Accesso ai valori attuali:
let a = document.getElementById("p1");
let b = a.innerHTML;                    // -> <p>Paragrafo!</p>
let c= a.outerHTML;                     // -> <div id="d"><p>Paragrafo!</p></div>
Modifica dei valori attuali:
a.innerHTML = "<ul><li>Lista!</li></ul>";
```

Conclusioni

In questa lezione abbiamo cercato di fare chiarezza tra:

- HTML
- (X)HTML 5
- HTML 4.x
- XHTML 1.0
- DOM

Riferimenti

The HTML Living Standard:
`http://www.whatwg.org/html`

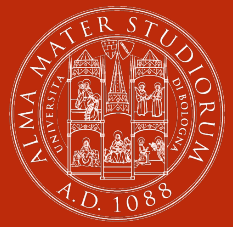
W3C HTML5:
`http://www.w3.org/TR/html5/`

D. Raggett, A. Le Hors, I. Jacobs, *HTML 4.01 Specification*, W3C Recommendation 24 December 1999,
`http://www.w3.org/TR/html401`

M. Altheim et alii, *Modularization of XHTML*, W3C Recommendation 10 April 2001, W3C Recommendation 10 April 2001,
`http://www.w3.org/TR/xhtml-modularization`

M. Altheim, S. McCarron, *XHTML™ 1.1 - Module-based XHTML*, W3C Recommendation 31 May 2001,
`http://www.w3.org/TR/xhtml11`

J. Axelsson et alii, *XHTML™ 2.0*, W3C Working Draft 31 January 2003,
`http://www.w3.org/TR/2003/WD-xhtml2-20030131`



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Fabio Vitali

Corso di tecnologie web

Fabio.vitali@unibo.it