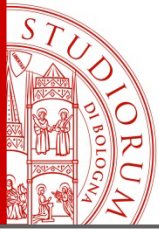


HTML5

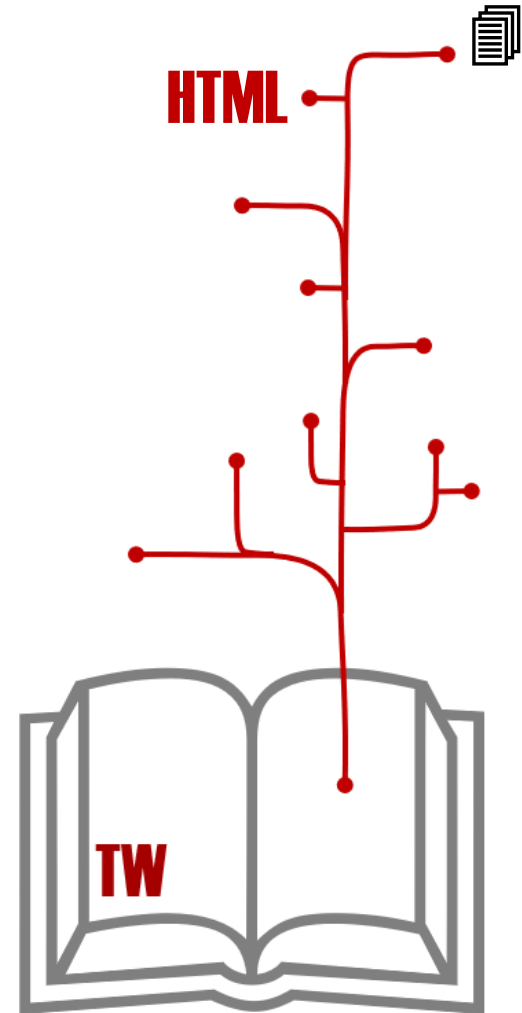
Link, Tabelle e Liste

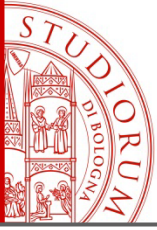




Argomenti

- HTML:
 - Elementi:
 - Link
 - Flow





Commenti

- I commenti nel codice HTML sono inseriti tra `<!-- e -->`
- Esempio:
`<!-- Questo è un commento. I commenti non sono visualizzati dal browser -->`



- L'elemento **<a>** consente di inserire àncore nel documento, ovvero **punti di partenza di un link**.
 - la destinazione si specifica con un URI attraverso l'attributo **href**.
 - Nelle precedenti versioni di HTML, con **<a>** si realizzavano anche i **punti di arrivo** di un link (raggiungibili con # come frammento interno di un URI), usando l'attributo **name**. Questa soluzione in HTML5 è stata **superata**: al suo posto si usa l'attributo **id**, associandolo a **qualunque elemento**.

<a>

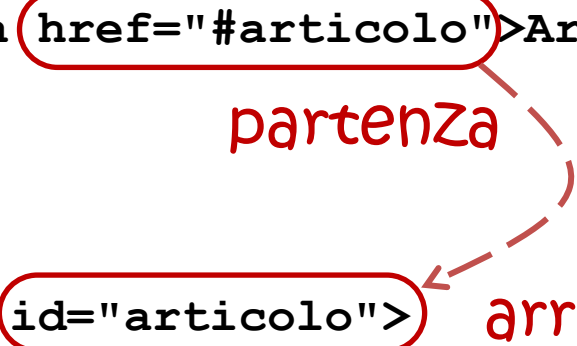
- Esempi:

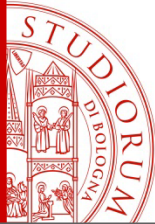
```
<nav>
  <ul>
    <li> <a href="/">Home</a> </li>
    <li> <a href="/news">News</a> </li>
    <li> <a href="http://www.google.it">Google</a> </li>
    <li> <a href="#articolo">Articolo</a> </li>
  </ul>
</nav>
```

partenza

```
<article id="articolo">
  <p> testo dell'articolo .... </p>
</article>
```

arrivo





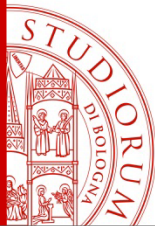
Risorsa

- Una **risorsa** è qualunque struttura che sia oggetto di scambio tra applicazioni all'interno del Web. Il concetto di risorsa è **indipendente dal meccanismo di memorizzazione** effettiva o dal tipo di contenuto.
- Anche se molti identificatori fanno riferimento a **file** memorizzati in un **file system gerarchico**, questo tipo di risorsa non è l'unico. Una risorsa potrebbe essere:
 - in un file system relazionale (per esempio VM di IBM)
 - in un database, e l'URI essere la chiave di ricerca
 - il risultato dell'elaborazione di un'applicazione, e l'URI essere i parametri di elaborazione.
 - una risorsa non elettronica (un libro, una persona, un pezzo di produzione industriale), e l'URI essere il suo nome Uniforme
 - un concetto astratto (la grammatica di un linguaggio)

URI

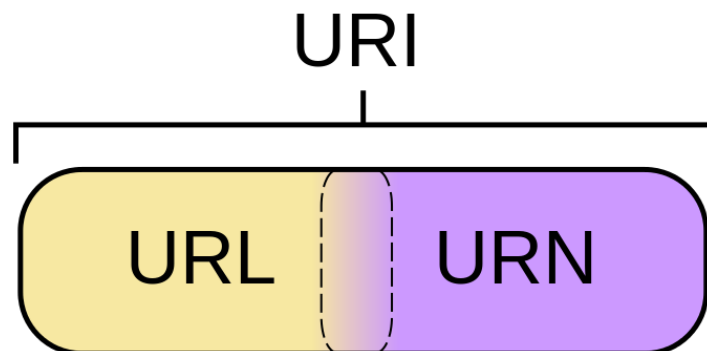
- Gli **URI (Uniform Resource Identifier)** sono una sintassi usata in WWW per definire i nomi e gli indirizzi di oggetti (risorse) su Internet.
 - Gli URI risolvono il problema di creare un meccanismo ed una sintassi di accesso **unificata** alle risorse di dati disponibili via rete.
 - Questi oggetti sono considerati accessibili tramite l'uso di **protocolli** esistenti, inventati appositamente, o ancora da inventare.
 - Tutte le istruzioni d'accesso ai vari specifici oggetti disponibili secondo un dato protocollo sono codificate come **una stringa di indirizzo**

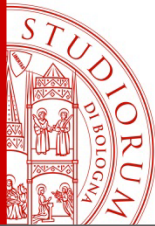




URI, URL e URN

- Gli **Uniform Resource Identifier (URI)** sono definiti come:
 - **Uniform Resource Locator (URL)**: una sintassi che contiene informazioni immediatamente utilizzabili per accedere alla risorsa (ad esempio, il suo indirizzo di rete)
 - **Uniform Resource Names (URN)**: una sintassi che permetta una etichettatura **permanente e non ripudiabile** della risorsa, indipendentemente dal riportare informazioni sull'accesso.





Sintassi degli URI

- Un URI è diviso in:

URI → `schema : [// authority] path [? query] [# fragment]`

- Lo **schema** (negli URL è il protocollo) è identificato da una stringa arbitraria (ma registrata presso **IANA**) usata come prefisso.
 - IANA (Internet Assigned Numbers Authority) è un organismo che ha, tra l'altro, la responsabilità nell'assegnazione degli indirizzi IP.
 - IANA ha un registro apposito per gli schemi URI:
<http://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml>
 - Tra gli schemi URI registrati/specificati da una RFC: http, https, ftp, mailto, rtsp, h323, snmp,...



Sintassi degli URI

- Un URI è diviso in:

URI → `schema : [// authority] path [? query] [# fragment]`

- L' **authority** individua un'organizzazione gerarchica dello spazio dei nomi a cui sono delegati i nomi (che possono essere ulteriormente delegati).
 - L'autorità è a sua volta divisa in:
authority → `[userinfo @] host [: port]`
 - La parte **userinfo** non deve essere presente se lo schema non prevede identificazione personale.
 - La parte **host** è o un nome di dominio o un indirizzo IP.
 - La **port** può essere omessa se ci si riferisce ad una well-known port (per http è la porta 80).

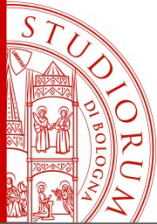


Sintassi degli URI

- Un URI è diviso in:

URI → `schema : [// authority] path [? query] [# fragment]`

- La parte **path** è la parte identificativa della risorsa all'interno dello spazio di nomi identificato dallo schema e (se esistente) dalla authority.
 - Poiché la presenza di una parte authority evidenzia uno spazio di nomi gerarchico sotto la gestione dell'autorità, in questi casi la parte path è divisa in blocchi separati da slash "/", ciascuno dei quali è un componente del path organizzato in **gerarchia**.
 - In questo caso diventano significativi gli pseudo componenti "." e "..".



Sintassi degli URI

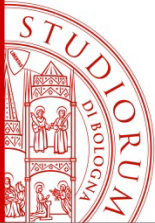
- Un URI è diviso in:

URI → `schema : [// authority] path [? query] [# fragment]`

- La parte **query** individua un'ulteriore specificazione della risorsa all'interno dello spazio di nomi identificato dallo schema.
 - Di solito questi sono parametri passati all'URI (un processo) per specificare un risultato dinamico, come l'output di una query su un motore di ricerca.
 - La parte query è tutto quello che sta dopo "?" e prima di "#".
 - Tipicamente ha la forma:

`nome1=valore1&nome2=valore+in+molte+parole`

- La parte **fragment** individua una risorsa secondaria (una risorsa associata, dipendente o in molti casi un frammento) della risorsa primaria. E' tutta la parte che sta dopo a "#".

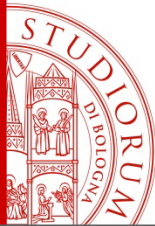


Esempio: ftp

- La sintassi della parte specifica è:

```
ftp://[user[:password]@]host[:port]/path
```

- dove:
 - Le parti **user** e **password** sono utente e password per l'accesso ad un server FTP. La mancanza di user fa partire automaticamente una connessione anonima.
 - Si tende a scoraggiare l'uso della password nell'URI, in quanto evidente situazione di scarsa sicurezza. Tuttavia lo schema lo prevede come parte facoltativa.
 - Le parti **Host**, **port** e **path** sono l'indirizzo del server, la porta di connessione ed il nome del file dell'oggetto ricercato, come per HTTP. La porta di default è 21.



Caratteri ammessi

- I caratteri degli URI sono **riservati, non riservati, escaped**:

curi : unreserved | reserved | escaped

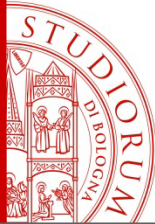
- I caratteri **non riservati** sono alfanumerici e alcuni caratteri di punteggiatura privi di ambiguità

unreserved: uppercase | lowercase | digit | punctuation

punctuation: - _ ! ~ * ' ()

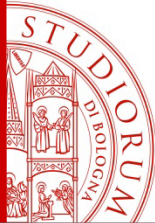
- I caratteri **riservati** sono caratteri che hanno delle funzioni particolari in uno o più schemi di URI. In questo caso vanno usati direttamente quando assolvono alle loro funzioni, e escaped quando sono invece parte della stringa identificativa naturale

reserved: ; / ? : @ & = + \$,



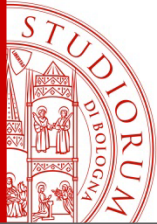
Caratteri riservati

- Sono caratteri riservati:
 - **%** é il **codice di escape**, e serve per l'utilizzo di caratteri particolari nell'URI, precedendone il codice esadecimale. Ad esempio, per utilizzare un carattere “%” nell'URI bisogna usare la stringa “%25”. Alcuni caratteri speciali o riservati o in generale non sicuri (es. quelli superiori al codice ASCII 127) possono essere specificati tramite codifica esadecimale introdotta dal carattere di escape.
 - **/**, **.** e **..** Sono usati per l'identificazione di sottoparti di uno schema **gerarchico**.
 - **#** serve per delimitare l'URI di un oggetto da un identificatore di un **frammento** interno alla risorsa considerata
 - **?** serve per separare l'URI di un oggetto su cui è possibile fare una **query** (un database, per esempio), dalla stringa usata per specificare la query.
 - **+** All'interno della query é usato al posto dello **spazio** (che non è mai usato per nessuna ragione).
 - ***** ha un significato **speciale** all'interno di schemi specifici.
 - **!** ha un significato **speciale** all'interno di schemi specifici.



URI reference

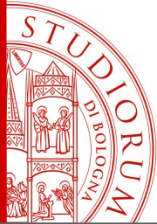
- Un **URI assoluto** contiene tutte le parti predefinite dal suo schema, esplicitamente precisate.
- Un URI gerarchico può però anche essere **relativo**, (in questo caso è detto **URI reference**) e riportare solo una parte dell'URI assoluto corrispondente, tagliando progressivamente cose da sinistra.
- Un URI reference fa sempre riferimento ad un **URI di base** (ad esempio, l'URI assoluto del documento ospitante l'URI reference) rispetto al quale fornisce porzioni differenti.
- Per esempio, l'URI reference `pippo.html` posto dentro al documento di URI `http://www.sito.com/dir1/dir2/pluto.html` fa riferimento al documento il cui URI assoluto è `http://www.sito.com/dir1/dir2/pippo.html`



URI assoluti e URI reference

- Un **URI assoluto** contiene tutte le parti predefinite dal suo schema, esplicitamente precisate.
- Un **URI gerarchico** può però anche essere **relativo** (detto tecnicamente un **URI reference**) ed in questo caso riportare solo una parte dell'URI assoluto corrispondente, tagliando progressivamente cose da sinistra.
- Un URI reference fa sempre riferimento ad un **URI di base** (ad esempio, l'URI assoluto del documento ospitante l'URI reference) rispetto al quale fornisce porzioni differenti.

Es.: l'*URI reference* **pippo.html** posto dentro al documento di URI **<http://www.sito.com/dir1/dir2/pluto.html>** fa riferimento al documento il cui URI assoluto è **<http://www.sito.com/dir1/dir2/pippo.html>**



Risolvere URI relativi

- Risolvere un URI relativo significa identificare l'URI assoluto sulla base dell'URI relativo stesso e, di solito, dell'URI di base.
- Dato l'URI di base `http://www.sito.com/dir1/doc1.html`:
 - se l'URI inizia con uno **schema**, è URI assoluto: `http://www.sito2.com/dir2/doc2.html` porta a `http://www.sito2.com/dir2/doc2.html`
 - se l'URI inizia con "#", è un frammento interno allo stesso documento di base: `#ancora1` porta a `http://www.sito.com/dir1/doc1.html#ancora1`
 - se l'URI inizia con "/", allora è un path assoluto all'interno della stessa autorità del documento di base, e gli va applicata la stessa parte autorità: `/dir3/doc3.html` porta a `http://www.sito.com/dir3/doc3.html`
 - se l'URI inizia con "..", (livello superiore di gerarchia): viene eliminato insieme all'elemento precedente. `./doc6.html` porta a `http://www.sito.com/dir1/./doc4.html` che è equivalente a `http://www.sito.com/dir1/doc4.html`
 - se l'URI inizia con ".", (stesso livello di gerarchia): `../doc7.html` porta a `http://www.sito.com/dir1/./doc5.html` che è equivalente a `http://www.sito.com/doc5.html`
 - **Altrimenti**, si estrae il path assoluto dell'URI di base, meno l'ultimo elemento, e si aggiunge in fondo l'URI relativo: `doc6.html` porta a `http://www.sito.com/dir1/doc6.html` mentre `dir7/doc7.html` porta a `http://www.sito.com/dir1/dir7/doc7.html`

URI shortener

- Con la nascita di Twitter e il limite dei 140 caratteri, sorge il problema di inserire nei tweet link ad URL anche piuttosto lunghi.
- Servizi come bit.ly, tr.im, o goo.gl permettono di creare URL molto brevi corrispondenti a URL molto lunghi
- Il servizio è un semplice rewriter (redirect, per essere più precisi) che automaticamente suggerisce un nome opaco molto breve



goo.gl/T655I →

`http://www.ted.com/talks/tim_berners_lee_the_year_open_data_went_worldwide.html`

Prima domanda

**BONUS**

DOMANDA 1:

A quale di questi elementi è semanticamente corretto che punti il link `` ?

- ☐ ``
- ☐ ``
- ☐ `<section id="primaparte">`
- ☐ `<br name="primaparte"/>`

- Abbiamo visto tutti i sottoinsiemi di Flow indicati dal W3C, ma mancano alcuni elementi che sono classificati solo come **Flow**.
- In particolare:
 - Gli **elementi di gestione delle tabelle**:
`<table>`, `<caption>`, `<tr>`, `<td>`, `<th>`,
`<colgroup>`, `<thead>`, `<tfoot>`, `<tbody>`.
 - Gli **elementi di gestione delle liste**:
``, ``, `<dl>`, ``, `<dt>`, `<dd>`

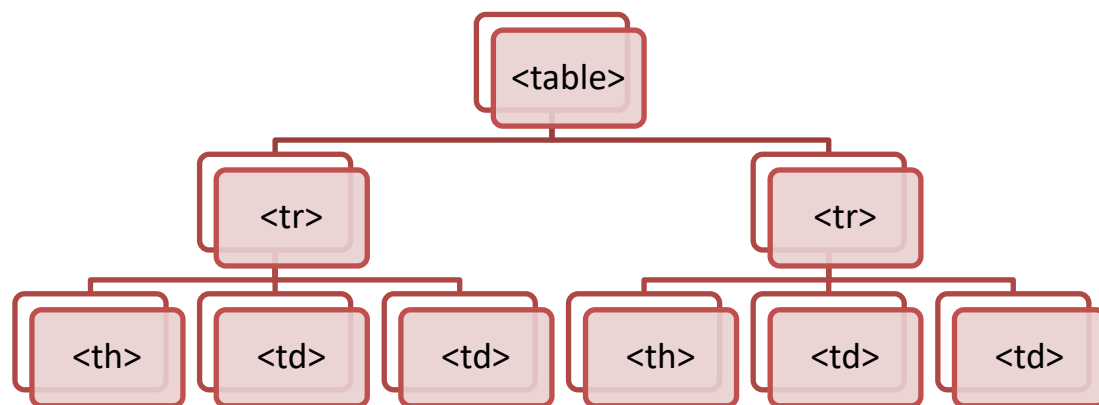


Struttura delle tabelle

- Le **tabelle** HTML5:
 - Sono realizzate attraverso l'elemento `<table>`
 - sono organizzate per righe, realizzate attraverso l'elemento `<tr>`, *table row*.
 - ciascuna riga è poi divisa in celle.
- Le celle possono essere:
 - Celle normali, in questo caso sono rese dall'elemento `<td>`, *table data*.
 - Celle di intestazione, che invece sono realizzate con l'elemento `<th>`, *table header*

Struttura delle tabelle

- Ogni tabella è composta di righe `<tr>`
 - Ogni riga è composta di celle di intestazione `<th>` o di contenuto `<td>`

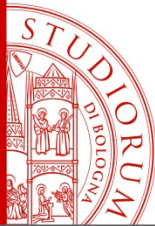


prof.	Silvia Mirri	Catia Prandi
corso	TW	Mobile

```

<table>
  <tr>
    <th>prof.</th>
    <td>Silvia Mirri</td>
    <td>Catia Prandi</td>
  </tr>
  <tr>
    <th>corso</th>
    <td>TW</td>
    <td>Mobile</td>
  </tr>
</table>

```

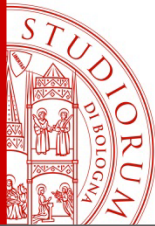


<table>

- Esempio con intestazione orizzontale (<th> prima cella di ogni riga):

```
<table>
  <tr>
    <th>Month</th>
    <td>January</td>
    <td>February</td>
  </tr>
  <tr>
    <th>Savings</th>
    <td>$100</td>
    <td>$80</td>
  </tr>
</table>
```

Month	January	February
Savings	\$100	\$80



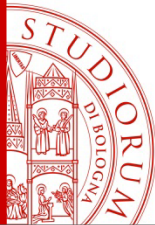
<table>

- Esempio con intestazione verticale (prima riga con tutte le celle <th>):

```
<table>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>February</td>
    <td>$80</td>
  </tr>
</table>
```

Month	Savings
January	\$100
February	\$80

http://www.w3schools.com/tags/tryit.asp?filename=tryhtml_table_test

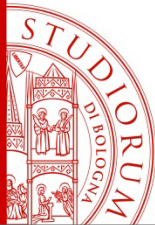


<caption>

- Esempio:

```
<table>
  <caption>Monthly savings</caption>
  <tr>
    <th>Month</th><th>Savings</th>
  </tr>
  <tr>
    <td>January</td><td>$100</td>
  </tr>
  <tr>
    <td>February</td><td>$50</td>
  </tr>
</table>
```

http://www.w3schools.com/tags/tryit.asp?filename=tryhtml_caption_test



<td><th>

- Una cella di tipo **<td>** o **<th>** può occupare più righe o più colonne utilizzando rispettivamente l'attributo **rowspan** e **colspan**

- Esempio:

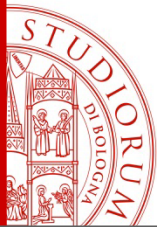
```
<table>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>February</td>
    <td>$80</td>
  </tr>
  <tr>
    <td colspan="2">Sum: $180</td>
  </tr>
</table>
```

Month	Savings
January	\$100
February	\$80
Sum: \$180	

http://www.w3schools.com/tags/tryit.asp?filename=tryhtml_td_colspan

<td><th>

- Una cella di tipo **<td>** o **<th>** può fare riferimento (tramite l'attributo **headers**), ad altre celle, per specificare che queste rappresentano una intestazione della cella corrente:
 - Lo scopo di questo sistema di relazioni tra celle è quello di supportare gli screen reader usati dalle persone non vedenti nel riferire correttamente alle celle intestazione di una certa cella. Rivediamo questo attributo parlando di accessibilità.
 - **headers** deve avere come valore la lista degli **id** delle intestazioni per la cella **SEPARATI DA SPAZIO**
- Per migliorare la strutturazione semantica e quindi l'**accessibilità** della tabella si possono usare anche elementi strutturali **<colgroup>**, **<thead>**, **<tfoot>**, **<tbody>**.



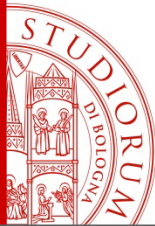
tabelle

- Tabella con nome e cognome, mail e telefono

Nome	Email	Telefono
Paola Salomoni	paola.salomoni@unibo.it	0547 338813
Silvia Mirri	silvia.mirri@unibo.it	0547 338892
Catia Prandi	catia.prandi2@unibo.it	0547 338892

- Potremmo anche compattare il telefono nelle ultime due righe?

Nome	Email	Telefono
Paola Salomoni	paola.salomoni@unibo.it	0547 338813
Silvia Mirri	silvia.mirri@unibo.it	0547 338892
Catia Prandi	catia.prandi2@unibo.it	



<table>

```
<table>
<tr>
  <th>Nome</th>
  <th>mail</th>
  <th>Telefono</th>
</tr>
<tr>
  <th> Paola Salomoni </th>
  <td><a href="mailto:paola.salomoni@unibo.it">
    paola.salomoni@unibo.it</a></td>
  <td> 0547 338813 </td>
</tr>
<tr>
  <th> Silvia Mirri </th>
  <td> <a href="mailto:silvia.mirri@unibo.it">
    silvia.mirri@unibo.it</a></td>
  <td> 0547 338892 </td>
</tr>
<tr>
  <th> Catia Prandi </th>
  <td> <a href="mailto:catia.prandi2@unibo.it">
    catia.prandi2@unibo.it</a></td>
  <td> 0547 338892</td>
</tr>
</table>
```



Versione senza ripetizioni

```
...  
<tr>  
  <th id="sm"> Silvia Mirri</th>  
  <td> <a href="mailto:silvia.mirri@unibo.it">  
    silvia.mirri@unibo.it</a></td>  
  <td rowspan="2" headers="sm cp"> 0547 338892 </td>  
</tr>  
<tr>  
  <th id="cp"> Catia Prandi</th>  
  <td> <a href="mailto:catia.prandi2@unibo.it">  
    catia.prandi2@unibo.it</a></td>  
</tr>  
...
```

Esempio

- **ESERCIZIO 1 COMPITO**

Scrivere il codice di un documento HTML accessibile il cui body contiene solo una sezione che include la seguente tabella con caption «orario delle lezioni»:

Orario delle lezioni

	Lunedì	Martedì	Mercoledì
9-10	Tecnologie Web	Analisi	Sistemi multimediali
10-11		Sistemi multimediali	
11-12	Algebra		Fisica

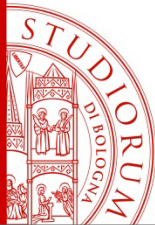
Note

- Lasciate stare la presentazione, senza CSS non si vede il bordo (io l'ho messo solo per farvi vedere meglio la tabella).
- Esiste una caption (Orario delle lezioni) che inserita dentro la tabella, prima delle celle.
- Attenzione ai rowspan e agli attributi da inserire per garantire l'accessibilità
- La struttura ha sostanzialmente 4 righe e 4 colonne ma ci saranno delle celle mancanti dovute ai rowspan



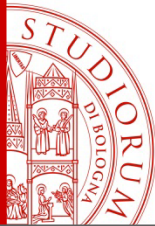
Soluzione

```
<!DOCTYPE html>
<html lang="it">
<head>
<title>Tabella accessibile</title>
</head>
<body>
<section>
  <table>
    <caption>Orario delle lezioni</caption>
    <thead>
      <tr>
        <th></th>
        <th id="lun">Lunedì</th>
        <th id="mar">Martedì</th>
        <th id="mer">Mercoledì</th>
      </tr>
    </thead>
```



Soluzione

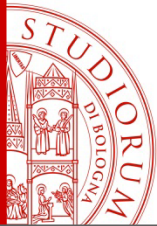
```
<tbody>
  <tr>
    <th id="9_10">9-10</th>
    <td rowspan="2" headers="lun 9_10 10_11">
      Tecnologie Web</td>
    <td headers="mar 9_10">Analisi</td>
    <td rowspan="2" headers="mer 9_10 10_11">
      Sistemi multimediali</td>
  </tr>
  <tr>
    <th id="10_11">10-11</th>
    <td rowspan="2" headers="mar 10_11 11_12 ">
      Sistemi multimediali</td>
  </tr>
  <tr>
    <th id="11_12">11-12</th>
    <td headers="lun 11_12">Algebra</td>
    <td headers="mer 11_12">Fisica</td>
  </tr>
</tbody>
</table>
</section>
</body>
</html>
```



Esempio con **scope**

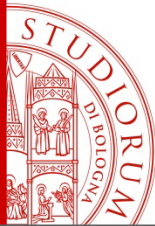
Orario delle lezioni

	Lunedì		Martedì	
	Aula 2.1	Laboratorio 2.2	Aula 2.1	Laboratorio 2.2
9-10	Sistemi multimediali	Programmazione	Analisi	Sistemi Multimediali
10-11		Tecnologie Web	Programmazione	
11-12	Algebra		Fisica	



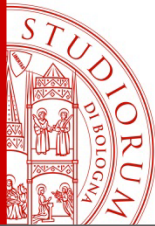
Esempio con `scope`

```
<!DOCTYPE html>
<html lang="it">
<head>
<title>Tabella accessibile</title>
<style>
    table, td, th {
        border: solid black 1px;
    }
</style>
</head>
<body>
<section>
    <table>
        <caption>Orario delle lezioni</caption>
        ...
```



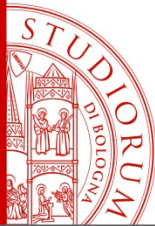
Esempio con scope

```
<thead>
  <tr>
    <th></th>
    <th id="lun" scope="colgroup" colspan="2">Lunedì</th>
    <th id="mar" scope="colgroup" colspan="2">Martedì</th>
  </tr>
  <tr>
    <th></th>
    <th id="aula21_1" headers="lun" scope="col">
      Aula 2.1</th>
    <th id="lab22_1" headers="lun" scope="col">
      Laboratorio 2.2</th>
    <th id="aula21_m" headers="mar" scope="col">
      Aula 2.1</th>
    <th id="lab22_m" headers="mar" scope="col">
      Laboratorio 2.2</th>
  </tr>
</thead>
```



Esempio con scope

```
<tbody>
  <tr>
    <th id="9_10" scope="row">9-10</th>
    <td rowspan="2" headers="lun 9_10 10_11 aula21_1">
      Sistemi multimediali</td>
    <td headers="lun 9_10 lab22_1">Programmazione</td>
    <td headers="mar 9_10 aula21_m">Analisi</td>
    <td headers="mar 9_10 10_11 11_12 lab22_m" rowspan="3">
      Sistemi Multimediali</td>
  </tr>
  <tr>
    <th id="10_11" scope="row">10-11</th>
    <td rowspan="2" headers="lun 10_11 11_12 lab22_1">
      Tecnologie Web</td>
    <td headers="mar 10_11 aula21_m">Programmazione</td>
  </tr>
```



Esempio con `scope`

```
<tr>
  <th id="11_12" scope="row">11-12</th>
  <td headers="lun 11_12 aula21_1">Algebra</td>
  <td headers="mar 11_12 aula21_m">Fisica</td>
</tr>
</tbody>
</table>
</section>
</body>
</html>
```






Esempio reale

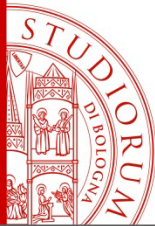
```
<table class="">
  <thead class="fc-head">
    <tr>
      <td class="fc-head-container fc-widget-header">
        <div class="fc-row fc-widget-header" style="border-right-width: 1px; margin-right: 16px;">
          <table class="">
            <thead>
              <tr>
                <th class="fc-axis fc-widget-header" style="width: 18px;"></th>
                <th class="fc-day-header fc-widget-header fc-mon fc-past" data-date="2020-09-28">
                  <span>Lun 28/9</span>
                </th>
                <th class="fc-day-header fc-widget-header fc-tue fc-past" data-date="2020-09-29">
                  <span>mar 29/9</span>
                </th>
                <th class="fc-day-header fc-widget-header fc-wed fc-past" data-date="2020-09-30">
                  <span>mer 30/9</span>
                </th>
                <th class="fc-day-header fc-widget-header fc-thu fc-past" data-date="2020-10-01">
                  <span>gio 1/10</span>
                </th>
                <th class="fc-day-header fc-widget-header fc-fri fc-today" data-date="2020-10-02">
                  <span>ven 2/10</span>
                </th>
                <th class="fc-day-header fc-widget-header fc-sat fc-future" data-date="2020-10-03">
                  <span>sab 3/10</span>
                </th>
                <th class="fc-day-header fc-widget-header fc-sun fc-future" data-date="2020-10-04">
                  <span>dom 4/10</span>
                </th>
              </tr>
            </thead>
          </table>
        </div>
      </td>
    </tr>
  </thead> ...
```

Ci sono due tabelle una dentro l'altra!

Nella tabella più interna c'è solo un <thead>, ma non c'è un <tbody> della tabella

C'è un <div> dentro ad un <td>

E questo è solo il <thead> della tabella più esterna

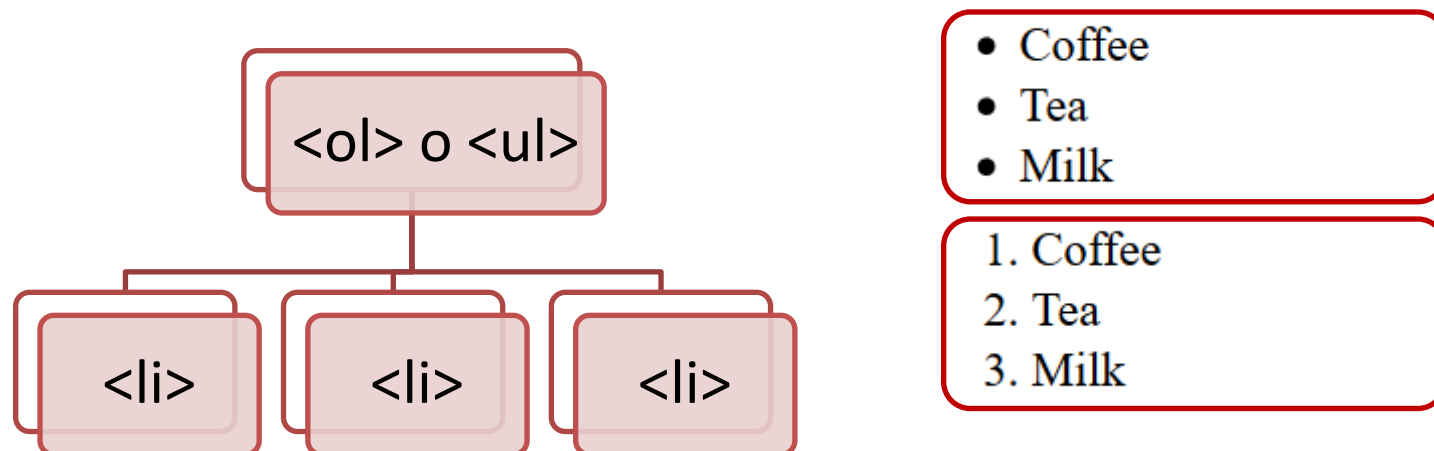


Liste

- In HTML5 sono previsti tre tipi di liste:
 - Liste non ordinate, definite da ``
(*unordered list*)
 - Liste ordinate, definite da ``
(*ordered list*)
 - Liste di definizioni, definite da `<dl></dl>`
(*definition list*)
- Nelle liste ordinate e non ordinate ogni item è definito da ``

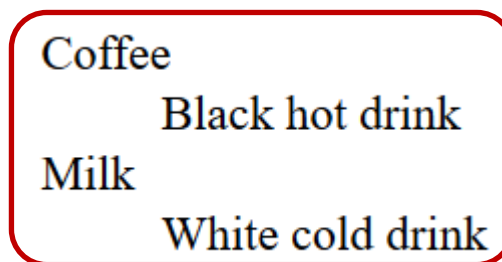
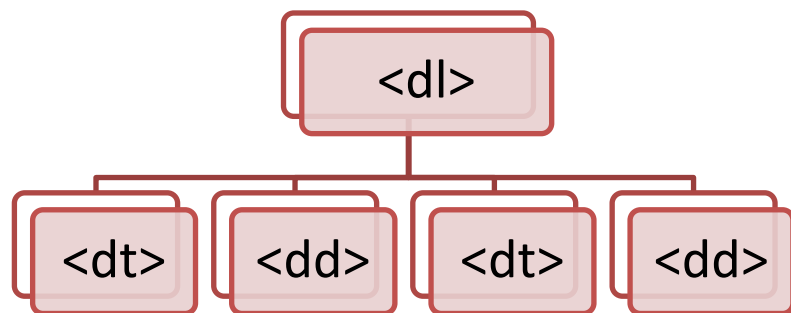
Struttura delle liste

- Per liste ordinate (``) e non ordinate (``), ogni item è definito da un ``



Struttura delle liste

- Per liste di definizioni (`<dl>`) non si usa `` ma due elementi che specificano termine e definizione `<dt>` e `<dd>`.



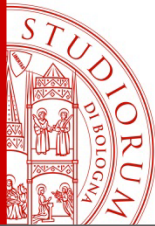


- Per ogni punto elenco, si deve annidare un elemento nella lista
- Esempio:

```
<ul>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ul>
```

- Coffee
- Tea
- Milk

http://www.w3schools.com/tags/tryit.asp?filename=tryhtml_lists4



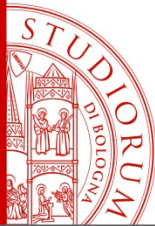
- Nelle liste ordinate possono essere specificati:
 - start: il valore iniziale della numerazione
 - type: il tipo di numerazione utilizzata
 - reversed: la numerazione è inversa

- Esempio:

```
<ol start="50">  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ol>
```

50. Coffee
51. Tea
52. Milk

- http://www.w3schools.com/tags/tryit.asp?filename=tryhtml_lists



<dl></dl>

- Le liste definite da <dl></dl> sono liste **associative**, pensate per correlare un concetto (il termine) con uno o più altri termini (la sue definizioni), che sono specificati da:
 - <dt> termini descrittivi,
 - <dd> una o più definizioni.
- A ogni termine possono corrispondere più definizioni.

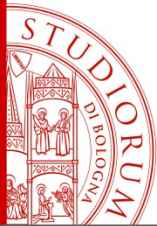
Authors

John

Luke

Editor

Frank



<dl></dl>

- Esempio:

```
<dl>  
  <dt>Coffee</dt>  
  <dd>Black hot drink</dd>  
  <dt>Milk</dt>  
  <dd>White cold drink</dd>  
</dl>
```

Coffee	Black hot drink
Milk	White cold drink

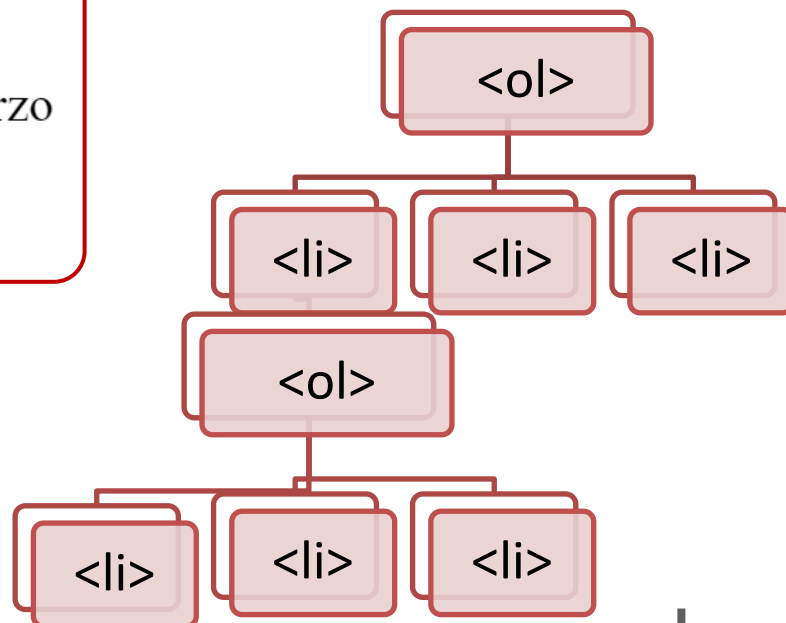
http://www.w3schools.com/tags/tryit.asp?filename=tryhtml_dd_test

Liste annidate

- Ovviamente le liste possono essere annidate l'una nell'altra

```
<ul>
  <li> primo
    <ul>
      <li>A</li>
      <li>B</li>
      <li>terzo</li>
    </ul>
  </li>
  <li>secondo</li>
  <li>terzo</li>
</ul>
```

- primo
 - A
 - B
 - terzo
- secondo
- terzo



Seconda domanda

**BONUS**

DOMANDA 2:

Considerare il seguente codice. Come viene visualizzata la lista corrispondente dal browser?

```
<ol>
  <li> Tecnologie Web
    <ul> <li> HTML </li>
        <li> URI </li>
    </ul> </li>
  <li> Reti
    <ul> <li> HTTP </li></ul> </li>
</ol>
```

Seconda domanda

**BONUS**

- Tecnologie Web
 - HTML
 - URI
- Reti
 - HTTP



1. Tecnologie Web
 - HTML
 - URI
2. Reti
 - HTTP



1. Tecnologie Web
 - HTML
2. Reti
 - URI
 - HTTP



1. Tecnologie Web
 1. HTML
2. Reti
 1. URI
 2. HTTP

Domande?

