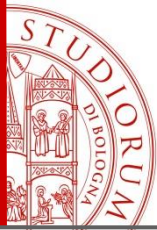


HTML5

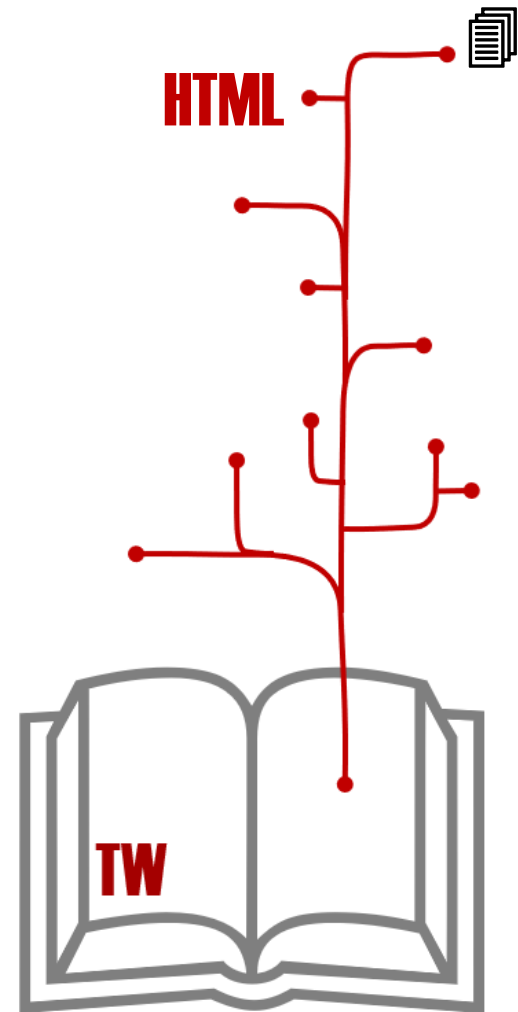
quarta parte





Argomenti

- HTML:
 - Elementi:
 - Interactive
 - Come scrivere codice corretto
 - Esempio di domanda del compito

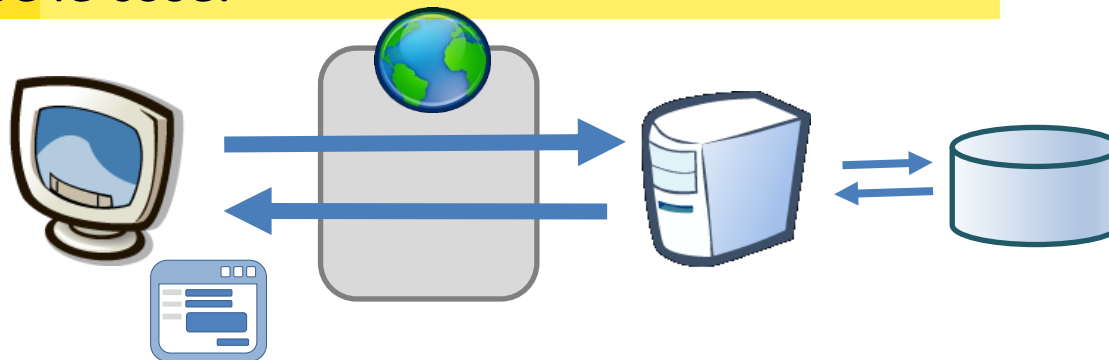


- Relativamente alla categoria Interactive, introduciamo le **form**, meccanismo di input (e in parte di output) di valori da elaborare.
- Gli elementi che vedremo sono:
 - `<form>`
 - `<input>`
 - `<textarea>`
 - `<select>`
 - `<option>`
 - `<output>`



<form>

- Una **form** è una parte della pagina Web che contiene controlli di input (form control), come per esempio campi di testo, bottoni e checkbox.
- Le form sono realizzate dall'elemento **<form>**.
- In HTML5 la form può anche avere campi per l'output.
- La form:
 - può essere processata a lato client (via Javascript),
 - può essere processata a lato server,
 - entrambe le cose.



<form>

- Attributi di **<form>** sono:
 - **action**: specifica l'URL dell'applicazione server-side che riceverà i dati.
 - **method**: specifica il metodo HTTP che deve essere usato per i dati, può essere:
 - **GET**: richiede di processare i dati a una specifica applicazione server.
 - **POST**: sottomette (invia) i dati da processare ad una specifica applicazione server.

```
<form action="http://www.google.com/search"
      method="get">
```

```
</form>
```

Specifica il metodo HTTP usato per l'invio:

- GET (default): i parametri vengono aggiunti all'URL dopo un ? (query string).

Vantaggi: semplice, cache-friendly, permette il bookmarking di URL con i parametri.

Svantaggi: lunghezza dei dati limitata (dipende dal browser/server), visibilità dei dati (non usare per informazioni sensibili).

- POST: i parametri vengono inviati nel corpo della richiesta.

Vantaggi: non soggetto ai limiti di lunghezza del URL, più sicuro per dati sensibili, necessario per l'upload di file (in combinazione con enctype="multipart/form-data").

Svantaggi: non cacheable di default, non bookmarkable.

GET e POST



- **GET**: richiede di processare i dati a una specifica applicazione server.
 - Invia dati testuali, aggiungendo una query string alla URL richiesta. I dati, tutti **testuali**, sono **visibili** nella URL.
 - La query string è separata dalla URL da ? ed è composta da coppie nome=valore separate tra loro da & (+ sostituisce gli spazi).
 - Si può usare per mettere una chiamata a server tutta nascosta nell'HTML (parametri inclusi).
 - Rimane in cache, resta nella history, può andare nei bookmark.
 - **Non va bene**: per dati troppo voluminosi, non testuali o che è opportuno non vedere (per esempio una password).
- Esempio:

<https://www.google.com/search?q=tecnologie+web>

POST



- **POST**: sottomette (invia) i dati da processare ad una specifica applicazione server.
 - Non ha restrizioni su dimensione e tipo di dati che vengono inviati al server. In particolare può inviare anche dati di tipo `multipart/form-data` incapsulando formati binari.
 - i dati sono inviati nel body del messaggio HTTP e non sono visibili, quindi è adatto al passaggio di dati riservati, come per esempio una password.
 - Quando non necessario, per l'utente è più scomodo di **GET** perché non rimane in cache, non resta nella history, non può andare nei bookmark.
 - Con HTTP **NON** è sicuro!



Attributi comuni

- Tra i tanti, noi vediamo:
 - **name**, specifica il nome del controllo, usato anche nella sottomissione della form, è definito mediante l'attributo.
 - **required** è un attributo booleano dei controlli che indica che è obbligatorio inserire un valore per quel campo.
 - **autofocus** mette il focus su questo controllo.
 - **autocomplete** è un attributo booleano di input che consente di attivare l'autocompletamento. Può essere messo anche a livello di form, il supporto del browser va verificato!

E-mail:
paola.salomoni@un...

Invia richiesta

DOMANDA 1:

Per trasmettere in modo sicuro una password è opportuno usare

- ☐ HTTP e GET
- ☐ HTTPS e GET
- ☐ HTTP e POST
- ☒ HTTPS e POST

<label>

- Ogni controllo deve avere una **<label>** che descriva il controllo all'utente:
 - La presenza di **<label>** è fondamentale per l'accessibilità, se non si vuole che sia visibile, va resa invisibile ma senza rimuoverla.
 - La label può essere associata al controllo:

- Annidando il controllo nella **<label>**.

```
<form>
  <p><label>Customer name:
    <input...../></label></p>
</form>
```

- Mettendo un **id** nel controllo e relazionandolo alla **<label>** mediante l'attributo **for**.

```
<p><label for="CN">Customer name: </label>
  <input .... id="CN" /></p>
</form>
```





<fieldset>

- L'elemento `<fieldset>` raggruppa più controlli che hanno semantica comune.
- Esempio:

`<form>`

```
<fieldset> <legend>dati personali:</legend>
```

```
<label>Nome:
```

```
<input type="text" name="nome"/></label><br/>
```

```
<label>Email:
```

```
<input type="email" name="mail"/> </label ><br/>
```

```
<label>Data di nascita:
```

```
<input type="date" name="date"/></label >
```

```
</fieldset><br/>
```

```
<input type="submit" value="Invia"/>
```

`</form>`

dati personali: _____

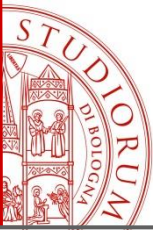
Nome:

Email:

Data di nascita

<input>

- L'elemento `<input>` consente di inserire nella pagina molti tipi diversi di controllo (alcuni introdotti da HTML5).
 - Il **tipo** è specificato mediante l'attributo **type** che può assumere i valori `hidden`, `text`, `search`, `tel`, `url`, `email`, `password`, `date`, `time`, `number`, `range`, `color`, `checkbox`, `radio`, `file`, `submit`, `image`, `reset`, `button`.
 - Visualizzazione e modalità di interazione variano da browser a browser.
 - Per provare singolarmente tutti i tipi su w3school:
http://www.w3schools.com/tags/att_input_type.asp



reset, submit e button

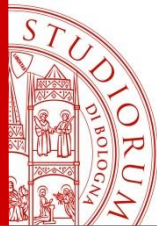
- Il valore **reset** per l'attributo **type** genera un bottone per ripristinare i valori di default della form
- Il valore **submit** per l'attributo **type** genera un bottone per sottomettere la form
- Il valore **button** per l'attributo **type** genera un bottone a cui può essere associato uno script Javascript attraverso l'attributo **onclick**.
- In tutti e tre gli elementi, attraverso l'attributo **value** viene modificato il testo sul bottone.

```
</form>  
  <input type="button" value="Bottone" onclick="..." />  
  <input type="reset" value="Reimposta" />  
  <input type="submit" value="Invia" />  
</form>
```

Bottone

Reimposta

Invia



image

Oltre ai normali campi del form, il browser aggiunge automaticamente due parametri:

- nome.x : coordinata orizzontale (pixel) in cui l'utente ha cliccato
- nome.y : coordinata verticale (pixel) in cui l'utente ha cliccato

Qui nome è il valore dell'attributo name (se non presente, i parametri non vengono inviati).

- Il tipo **image** ha la stessa funzione di un bottone (se non specificato, ^{ha la funzione} diversamente di **submit**).
 - L'immagine è usata per la resa grafica.
 - Va inserito l'elemento alt per rendere accessibile il bottone.
 - Invia le coordinate del click (x e y) al server.
- Esempio:

```
<form action="demo_form.asp">  
  <label>nome: <input type="text" name="fname" />  
  </label><br/>  
  <input type="image" src="img_submit.gif"  
    alt="Submit" width="48" height="48" />  
</form>
```

nome:



http://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_input_type_image



hidden

- Gli input di tipo `hidden` sono inseriti nella form per creare una associazione variabile valore invisibile all'utente.
- Il controllo `<input>` con `type="hidden"` non viene visualizzato.
- Esempio:

```
<form action="...">  
  <input type="hidden" name="country"  
        value="Italy" />  
  <input type="submit" value="Invia" />  
</form>
```



http://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_input_type_hidden



password

- Gli input di tipo **password** sono inseriti nella form per creare una associazione variabile valore invisibile all'utente.
- Quando l'utente riempie il controllo, la visualizzazione della password viene oscurata.
- Esempio:

```
<form action="...">  
  <label>Password:  
    <input type="password" name="pwd"  
      maxlength="8" /></label>  
  <br/><br/>  
  <input type="submit" value="Invia richiesta"/>  
</form>
```

Password: ●●●●●●●●

Invia richiesta

http://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_password

Esempio per mostrare il contenuto inserito nell'input con **type="password"**:

https://www.w3schools.com/howto/howto_js_toggle_password.asp



radio

- Gli input di tipo **radio** realizzano un radio button. Le diverse opzioni possibili sono definite inserendo più **radio** con lo stesso **name**.

- Esempio:

Viene messo Value perchè all'invio del Form si manda gender=opzione

```
<form action="demo_form.asp">  
  <label>  
    <input type="radio" name="gender" value="male"/>  
    Male</label> <br/>  
  <label>  
    <input type="radio" name="gender" value="female"/>  
    Female</label> <br/>  
  <label>  
    <input type="radio" name="gender" value="other"/>  
    Other</label> <br/>  
  
  <input type="submit" value="Submit"/>  
</form>
```

☐ Male
☐ Female
☐ Other

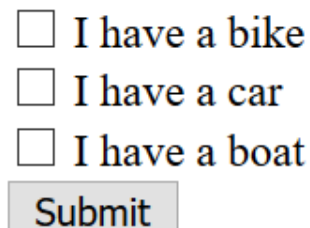
http://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_input_type_radio

checkbox

- Gli input di tipo **checkbox** realizzano una checkbox. Le diverse opzioni possibili sono definite inserendo più **checkbox** con lo stesso **name**.
- Esempio:

Viene messo Value perchè all'invio del Form si manda vehicle=opzione

```
<form action="demo_form.asp">  
  <label><input type="checkbox" name="vehicle1"  
    value="Bike"/>I have a bike</label><br/>  
  <label><input type="checkbox" name="vehicle2"  
    value="Car"/> I have a car</label><br/>  
  <label><input type="checkbox" name="vehicle3"  
    value="Boat"/> I have a boat</label><br/>  
  <input type="submit" value="Submit"/>  
</form>
```



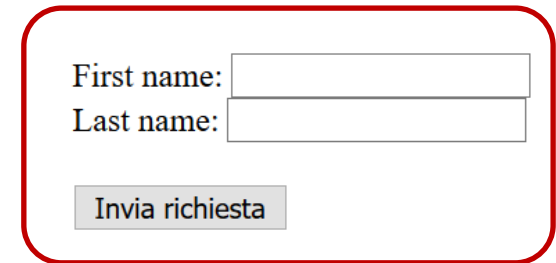
☐ I have a bike
☐ I have a car
☐ I have a boat

http://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_input_type_checkbox

text

- Gli input di tipo **text** sono utilizzati come controlli per l'input di testo (generico). Per tipi particolari di testo (per esempio mail, colori o date), esistono controlli specifici.

```
<form action="demo_form.asp"/>  
  <label>First name:  
    <input type="text" name="fname"/>  
</label><br/>  
  <label>Last name:  
    <input type="text" name="lname"/>  
</label><br/>  
  <input type="submit" value="Invia richiesta"/>  
</form>
```



http://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_input_type_text



search

- Il tipo **search** consente di inserire testi che diventano chiavi di ricerca.
 - Come nel tipo testo, si può inserire un testo qualunque (senza pattern specifici).
 - In alcuni browser (es. Safari) il campo è visualizzato in modo da evidenziare la sua funzione.

- Esempio

```
<form action="demo_form.asp">  
  <label>Search Google:  
  <input type="search" name="googlesearch"/>  
</label><br/>  
  <input type="submit"  
    value="Invia richiesta"/>  
</form>
```



http://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_input_type_search

color

- Il tipo **color** consente di scegliere un colore, attraverso un color picker che consente di selezionarlo.

- Esempio:

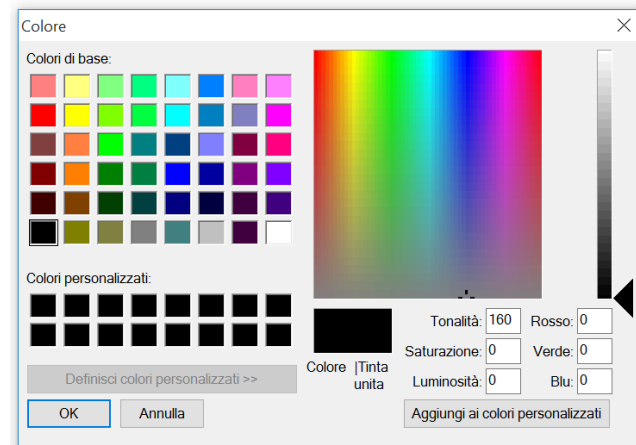
```
<form action="demo_form.asp">
  <label>Select your favorite color:
  <input type="color" name="favcolor"/>
  <label><br/>
  <input type="submit" value="Invia richiesta"/>
</form>
```

- PROVA!

Select your favorite color:



Invia richiesta



Data e ora

- Per codificare data e ora sono disponibili vari tipi di **input**:

– **time**: orario

Select a time: -- : --

Select a time: 02 : 00

– **week**: settimana dell'anno

week: Settimana --, ----

agosto 2015

Settimana	lun	mar	mer	gio	ven	sab	dom
31	27	28	29	30	31	1	2
32	3	4	5	6	7	8	9
33	10	11	12	13	14	15	16
34	17	18	19	20	21	22	23
35	24	25	26	27	28	29	30
36	31	1	2	3	4	5	6

– **date**: data

date: gg/mm/aaaa

agosto 2015

lun	mar	mer	gio	ven	sab	dom
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

– **datetime**: data e orario

– **datetime-local**: data e orario locali

- Vengono selezionati in modo diverso da browser a browser, in alcuni casi (negli esempi è Chrome) l'immissione è più efficace.



Data e ora

- Esempio:

```
<form action="demo_form.asp">
  <label> time: <input type="time"
    name="usr_time"/></label><br/><br/>
  <label> week: <input type="week"
    name="week_year"/></label><br/><br/>
  <label> date: <input type="date"
    name="day"/></label> <br/><br/>
  <label> date and time: <input
    type="datetime" name="dt"/>
  </label> <br/><br/>
  <label>local date and time:
    <input type="datetime-local"
      name="ldt"/></label> <br/><br/>
  <input type="submit" value="Invia"/>
</form>
```

time: --:--

week: Settimana --, ----

date: gg/mm/aaaa

date and time:

local date and time: gg/mm/aaaa --:--

Invia



tel, url, email

- Per inserire dati personali (e non solo) sono previsti tipi specifici per **tel**, **url** e **email**
- Esempio:

```
<form action="demo_form.asp">
  <p> Inserisci i tuoi dati </p>
  <label> e-mail: <input type="email"
    name="usremail"/></label><br/><br/>
  <label> telefono: <input type="tel"
    name="usrtel"/></label><br/><br/>
  <label>homepage: <input type="url"
    name="homepage"/></label><br/><br/>
  <input type="submit" value="Invia"/>
</form>
```

Inserisci i tuoi dati

E-mail:

Telephone:

homepage:



number e range

- Tipi di controllo per i numeri sono:
 - **number**: consente di inserire un numero
 - **range**: consente di scegliere in un range
- Per entrambi si possono definire:
 - **min**: specifica il valore minimo
 - **max**: specifica il valore massimo
- Esempio:

```
<form action="demo_form.asp">  
<label>range (0-10)</label>  
  <input type="range" name="points"  
min="0" max="10"/></label><br/><br/>  
<label>numero (1-5):</label>  
  <input type="number" name="quantity"  
min="1" max="5"/></label><br/><br/>  
<input type="submit" value="Invia"/>  
</form>
```

range (0-10)

numero (1-5):

range (0-10)

numero (1-5):

Inserire un numero

When UI designer go crazy...

Please enter your phone number:



file

- Il tipo **file** consente di selezionare un file:
 - Possono essere caricati più file ma il **name** deve essere senza path (anche se si carica una directory).
 - Esiste uno stato che consente di monitorare l'upload

- Esempio:

```
<form action="demo_form.asp">  
  <label>Selezionare file:  
    <input type="file"  
      name="file1"/>  
  </label><br/><br/>  
  <input type="submit"  
    value="Invia richiesta"/>  
</form>
```

Selezionare file : Sfoglia... Nessun file selezionato.

Invia richiesta

Selezionare file : Sfoglia... 0.introduzione.pptx

Invia richiesta

http://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_input_type_file



<textarea>

- L'elemento **<textarea>** definisce un controllo di input per testi multilinea.
 - Gli attributi **cols** e **rows** specificano la dimensione della **<textarea>**.

- Esempio:

```
<form>
```

```
<label>testo libero
```

```
<br/><textarea rows="4" cols="50">
```

```
il testo inserito tra inizio e fine  
elemento è il valore di default della textarea
```

```
</textarea></label><br/><br/>
```

```
<input type="submit" value="Submit"/>
```

```
</form>
```

testo libero

il testo inserito tra inizio e fine elemento è il
valore di default della textarea

Submit

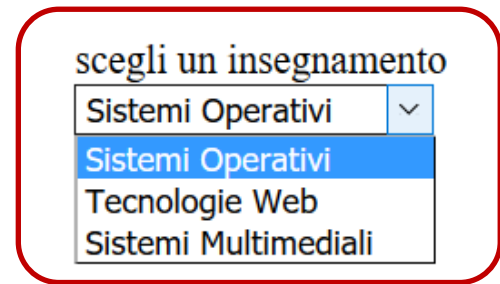
http://www.w3schools.com/tags/tryit.asp?filename=tryhtml_textarea

<select> e <option>

- L'elemento **<select>** è usato per realizzare menù a tendina.
- Le diverse opzioni sono introdotte attraverso l'elemento **<option>**.

- Esempio:

```
<form>
<label> scegli un insegnamento<br/>
<select name="insegnamento">
  <option value="SO">Sistemi Operativi</option>
  <option value="TW">Tecnologie Web</option>
  <option value="SM">Sistemi Multimediali</option>
</select></label>
</form>
```



scegli un insegnamento

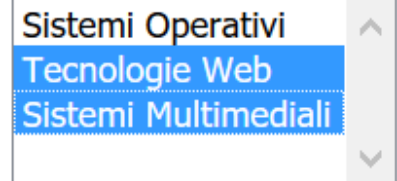
Sistemi Operativi	▼
Sistemi Operativi	
Tecnologie Web	
Sistemi Multimediali	

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

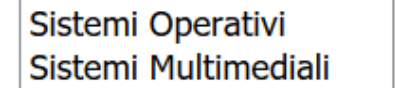
<select> e <option>

- Attributi per <select> sono:
 - **multiple**, booleano per effettuare selezioni multiple
 - **size**, definisce le opzioni da mostrare all'utente
- Attributi per <option> sono:
 - **selected**, booleano è true se quella <option> è il valore di default, altrimenti la prima option è il default. Se si vuole un default nullo, si deve aggiungere una <option> vuota.
 - **value**, indica il valore quando diverso dal testo della opzione, per esempio un codificato o abbreviato.

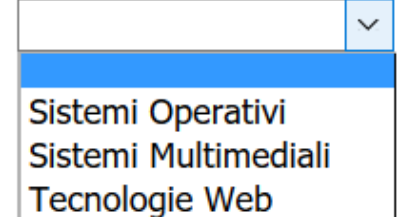
scegli un insegnamento



scegli un insegnamento



scegli un insegnamento





<optgroup>

- L'elemento <optgroup> è usato per raggruppare le <option> in una <select>:

- Esempio:

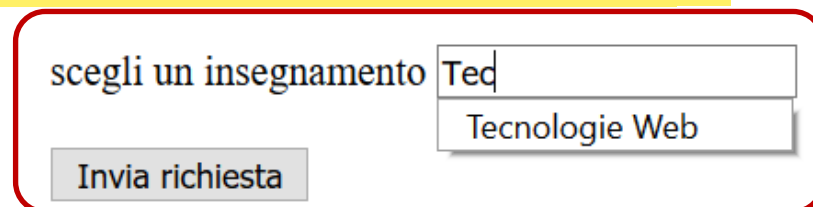
```
<form>
<label> scegli un insegnamento<br/>
<select name="insegnamento">
  <optgroup label="2014/15">
    <option value="SO">Sistemi Operativi</option>
    <option value="SM">Sistemi Multimediali</option>
  </optgroup>
  <optgroup label="2015/16">
    <option value="TW">Tecnologie Web</option>
    <option value="SM">Sistemi Multimediali</option>
  </optgroup>
</select></label>
</form>
```

scegli un insegnamento	
Sistemi Operativi	▼
2014/15	
Sistemi Operativi	
Sistemi Multimediali	
2015/16	
Tecnologie Web	
Sistemi Multimediali	

<datalist>

- L'elemento **<option>**, insieme all'elemento **<datalist>** può essere usato anche per predisporre dei suggerimenti per i valori di un campo testuale.

- Esempio:



```
<form>
```

```
<label> scegli un insegnamento
```

```
<input list="corsi" name="insegnamento"/></label>
```

```
<datalist id="corsi">
```

L'attributo list dell'<input> deve contenere l'id del corrispondente <datalist>. In questo modo il browser sa dove trovare le opzioni da suggerire:

```
<option value="SO">Sistemi Operativi</option>
```

```
<option value="TW">Tecnologie Web</option>
```

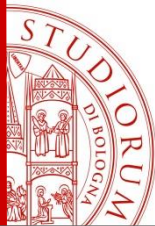
```
<option value="SM">Sistemi Multimediali</option>
```

```
</datalist><br/><br/>
```

```
<input type="submit" value="Invia richiesta"/>
```

```
</form>
```

http://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_datalist



Perché usare oninput sul form
Centralizzi il codice in un solo punto, senza dover aggiungere listener a ciascun controllo. Utile se hai più slider, input o textarea e vuoi calcolare qualcosa che dipende da tutti loro.

<output>

Grazie al bubbling (propagazione) degli eventi, quando tu sposti lo slider:

- Il browser genera un input event sull'<input type="range">.
- L'evento "risale" (bubble) ai suoi antenati fino al <form>, attivando il gestore definito in oninput sul form stesso — non serve metterlo direttamente sull'<input>

- L'elemento **<output>** rappresenta il risultato di un calcolo o azione dell'utente.

- Esempio:

```
<form oninput="(x).value=parseInt(a.value)">
```

```
<label for="a">slider: <br/>
```

```
0<input type="range" id="a" value="50"/>100</label> <br/>
```


```
il valore dello slider è: <output name="(x)"
```

```
for="a"></output>
```

```
</form>
```

L'<output> è un contenitore HTML5 pensato per visualizzare il risultato di un calcolo o un'informazione generata dallo script.
Attribuendo name="x", il codice inline x.value=... accede direttamente a questo elemento (il nome corrisponde a document.forms[0].elements['x']).

slider:

0  100

il valore dello slider è: 33

Note: The output tag is not supported in Internet Explorer.

http://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_output

Seconda domanda

**BONUS**

DOMANDA 2:

L'utente deve poter scegliere una quantità da 1 a 5. Quale soluzione NON risponde alla richiesta?

- ☐ select con opzioni da 1 a 5
- ☐ checkbox da 1 a 5
- ☒ range da 1 a 5
- ☐ number limitato da 1 a 5

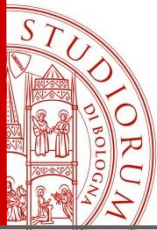


Esempio

- **ESERCIZIO 1:**

Scrivere il codice HTML5 accessibile e semanticamente corretto per realizzare una form che invia con metodo GET i seguenti dati:

- Gruppo di campi di anagrafica: Nome, cognome, email del referente del gruppo
- Gruppo di campi per la scelta del tipo di elaborato, con cui si specificano: il tipo di elaborato, a scelta alternativa tra semplice, medio, complesso; il numero di membri del gruppo, con quantità tra 1 e 4



Soluzione

```
<!DOCTYPE html>
<html lang="it">
  <head>
    <meta charset="utf-8"/>
    <title>Primo esercizio</title>
  </head>
  <body>
    <form action="script.php" method="GET">
      <fieldset>
        ... Primo fieldset
      </fieldset>
      <fieldset>
        ... Secondo fieldset
      </fieldset>
      <input type="submit" value="Invia"/>
    </form>
  </body>
</html>
```



Soluzione

```
<fieldset>
  <legend>Anagrafica</legend>
  <label>Nome:
    <input type="text" name="nome"
      autocomplete="on" placeholder="nome.."
      required/></label>
  <label>Cognome:
    <input type="text" name="cognome"
      autocomplete="on" placeholder="cognome.."
      required/></label>
  <label>Email referente:
    <input type="email" name="email"
      autocomplete="on" placeholder="email.."
      required/></label>
</fieldset>
```



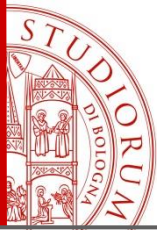
Soluzione

```
<fieldset>
  <legend>Tipo elaborato</legend>
  <label>Tipo elaborato:
    <select name="tipoElaborato">
      <option value="semplice">Semplice</option>
      <option value="medio">Medio</option>
      <option value="complesso">Complesso</option>
    </select>
  </label>
  <label>Numero membri:
    <input type="number" name="nrMembri" min="1"
      max="4" required/>
  </label>
</fieldset>
```



Scrivere il codice

- Per le consegne di TW il codice deve esser:
 1. **Well formed (XHTML)**
 2. **Basato sulla semantica degli elementi e degli attributi**
 3. **Con presentazione separata dal contenuto**
 4. **Accessibile (livello WCAG AA)**
- Per farlo, si possono seguire queste indicazioni:
 - Scrivere usando la **corretta semantica, senza elementi o attributi presentazionali**
 - Controllare con **Validator**
 - Controllare con **AChecker**
 - **Controllare a mano**



Controllare con validator

- Per validare il codice usiamo il validatore del W3C, <https://validator.w3.org/>



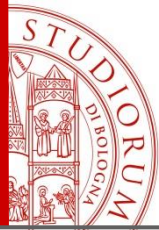
The screenshot shows the W3C Markup Validation Service interface. At the top, there's a blue header with the W3C logo and the text "Markup Validation Service" and "Check the markup (HTML, XHTML, ...) of Web documents". Below this, there are three tabs: "Validate by URI" (selected), "Validate by File Upload", and "Validate by Direct Input". Under the "Validate by URI" tab, there's a section titled "Validate by URI" with the instruction "Validate a document online:". Below this, there's a label "Address:" followed by a text input field. A link "More Options" is visible below the input field. At the bottom of the form, there is a "Check" button.

- ATTENZIONE**, è valido ma non è detto che sia accessibile e quindi lo strumento non è sufficiente a dire che le consegne del compito sono giuste!

Controllare con Achecker

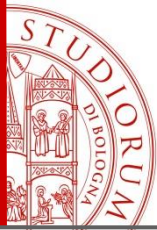
- Dobbiamo ancora introdurre bene l'accessibilità del Web (nelle prossime settimane), ma intanto introduciamo lo strumento di validazione: Achecker.

The screenshot shows the AChecker Web Accessibility Checker interface. At the top, there are links for 'Login' and 'Register'. The main heading is 'Web Accessibility Checker'. Below this, there are three tabs: 'Web Page URL', 'HTML File Upload', and 'Paste HTML Markup'. The 'Web Page URL' tab is selected, showing an 'Address:' input field and a 'Check It' button. Below the input field, there is an 'Options' section with checkboxes for 'Enable HTML Validator', 'Enable CSS Validator', and 'Show Source'. Underneath, there is a 'Guidelines to Check Against' section with radio buttons for various standards: BITV 1.0 (Level 2), WCAG 1.0 (Level A), WCAG 2.0 (Level A), Section 508, WCAG 1.0 (Level AA), WCAG 2.0 (Level AA) (which is selected), Stanca Act, WCAG 1.0 (Level AAA), and WCAG 2.0 (Level AAA). At the bottom of the options, there is a 'Report Format' section with radio buttons for 'View by Guideline' (selected) and 'View by Line Number'. On the right side of the interface, there is an advertisement for Google with a 'Report this ad' button and a 'Why this ad?' link. At the bottom of the page, there is a welcome message: 'Welcome to AChecker. This tool checks single HTML pages for conformance with accessibility standards to ensure the content can be accessed by everyone. See the Handbook link to the upper right for more about the Web Accessibility Checker.'



Controllo manuale

- Ogni volta che si fa una correzione, le validazioni automatiche vanno rifatte entrambe!
- La validazione automatica:
 - A volte ha dei bug
 - Non può controllare tutto (le figure e la necessità di alternative testuali, per esempio)
- Alla fine va riverificato manualmente che il codice rispetti tutti e 4 i punti.



Esempio di ESERCIZIO (compito)

- Scrivere un documento HTML valido con codice HTML5 accessibile e semanticamente corretto per realizzare la tabella seguente, con *caption* «Valutazione Esercizi»:

Valutazione esercizi

Esercizio	Valutazione	
	minimo	massimo
Esercizio 1	3	5
Esercizio 2	5	7
Esercizio 3	7	9



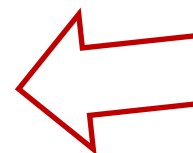
Esempio

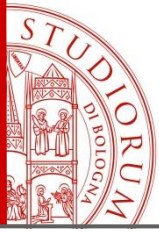
```
<!DOCTYPE html>
<html>
<head>
  <link href="fogliostile.css" rel="stylesheet" />
  <title>Esercizio</title>
</head>
<body>
<table>
<caption> Valutazione esercizi </caption>
<tr>
  <th rowspan="2"> Esercizio </th>
  <th colspan="2" id="val">Valutazione</th>
</tr>
<tr>
  <th id="min"> minimo </th>
  <th id="max"> massimo </th>
</tr>
...
```

Mancano
<thead> e
<tbody>!



Mancano scope
e colgroup!





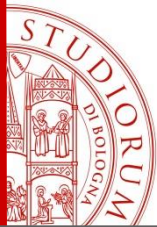
Esempio

```
<tr>
  <th id="es1">Esercizio 1</th>
  <td headers="val min es1">3</td>
  <td headers="val max es1">5</td>
</tr>
<tr>
  <th id="es2">Esercizio 2</th>
  <td headers="val min es2">5</td>
  <td headers="val max es2">7</td>
</tr>
<tr>
  <th id="es3">Esercizio 3</th>
  <td headers="val min es3">7</td>
  <td headers="val max es3">9</td>
</tr>
</table>
</body>
</html>
```



Esempio

```
<!DOCTYPE html>
<html>
<head>
  <link href="fogliostile.css" rel="stylesheet" />
  <title>Esercizio</title>
</head>
<body>
<table>
<caption> Valutazione esercizi </caption>
  <thead>
    <tr>
      <th rowspan="2"> Esercizio </th>
      <th colspan="2" scope="colgroup" id="val">Valutazione</th>
    </tr>
    <tr>
      <th id="min" scope="col"> minimo </th>
      <th id="max" scope="col"> massimo </th>
    </tr>
  </thead>
```

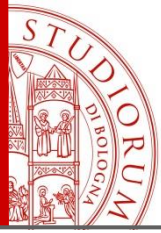


Esempio

```
<tbody>
  <tr>
    <th id="es1" scope="row">Esercizio 1</th>
    <td headers="val min es1">3</td>
    <td headers="val max es1">5</td>
  </tr>
  <tr>
    <th id="es2" scope="row">Esercizio 2</th>
    <td headers="val min es2">5</td>
    <td headers="val max es2">7</td>
  </tr>
  <tr>
    <th id="es3" scope="row">Esercizio 3</th>
    <td headers="val min es3">7</td>
    <td headers="val max es3">9</td>
  </tr>
</tbody>
</table>
</body>
</html>
```

Concludendo ...





Domande?

