

HTML (Hyper Text Markup Language)

- ❑ E' un linguaggio per la descrizione formale di **ipertesti multimediali** (e, in generale, interfacce utente grafiche)
- ❑ Utilizzato nel WWW come rappresentazione delle pagine (tipo MIME text/html)
- ❑ Portabile e leggibile (basato su testo)
- ❑ Principali versioni:
 - ❖ HTML 2.0 (rfc 1866) Novembre 1995
 - ❖ HTML 3.2 (w3c) Gennaio 1997
 - ❖ HTML 4.0 (w3c) Dicembre 1997
 - ❖ HTML 4.01 (w3c) Dicembre 1999 - vers. Finale
 - ❖ HTML 5 (w3c + whatwg) Iniziato 2004, finito 2014

Cosa è possibile fare

- ❑ Definire il layout di una pagina web
- ❑ Definire il modo in cui la pagina dovrà essere presentata all'utente
- ❑ Inserire collegamenti ipertestuali ad altre pagine web
- ❑ Inserire parti eseguibili nella pagina web

Si può realizzare il cliente di un'applicazione distribuita

Esempio di file HTML

```
<!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//IT>
<HTML>
<HEAD>
<TITLE>Universita` e Ricerca sul WWW</TITLE>
</HEAD>

<BODY bgcolor="#FFFFFF">

<H1 ALIGN=center> UNIVERSIT&Agrave; E RICERCA SUL WWW</H1>
<HR SIZE=4 WIDTH=60% ALIGN=CENTER>

<P>
<UL>
<LI><A HREF="http://cru1.dsi.uniroma1.it/">CRUI</A>
Conferenza dei Rettori delle Universit&grave; Italiane
<LI><A HREF="http://www.murst.it/">MURST</A> Ministero
dell'Università e della Ricerca Scientifica e Tecnologica
</UL>

</BODY>
</HTML>
```

HTML come applicazione SGML

- ❑ SGML (Standard Generalized Markup Language) è un sistema per la descrizione di linguaggi di markup:
 - ❖ basato su annotazioni (markup) che servono per rappresentare elementi strutturali, di presentazione, ecc.
- ❑ HTML è descrivibile tramite SGML
- ❑ HTML è definito da:
 - ❖ SGML Declaration: caratteri e delimitatori validi
 - ❖ Document Type Definition (DTD) : sintassi dei costrutti di markup
 - ❖ La specifica della semantica dei costrutti di markup

Caratteri

- ❑ **Character Set** (insieme ordinato di caratteri astratti)
 - ❖ Vista l'universalità del web, un set che contenga solo i caratteri ASCII non è sufficiente.
 - ❖ HTML usa UCS (Universal Character Set - ISO 10646), che contiene esattamente tutti i caratteri Unicode.
- ❑ **Character Encoding** (corrispondenza tra sequenze di caratteri e sequenze di byte)
 - ❖ In HTML possono essere usate diverse codifiche (esempio: ISO-8859-1, noto anche come Latin1)
 - ❖ Esistono diverse modalità per specificare la codifica usata in un documento HTML

Character References

- ❑ Una codifica può rappresentare direttamente solo un sottoinsieme del set di caratteri UCS
- ❑ I caratteri che non hanno rappresentazione diretta possono essere indicati tramite i *character references* (rappresentazioni basate su sequenze di escape)
- ❑ Esempi:

Carattere	Sequenze di escape	
&	&	&
<	<	<
>	>	>

ELEMENTI e TAG

- Un documento contiene una serie di ELEMENTI (titoli, paragrafi, liste, ecc.)
- Un tag è un simbolo usato per rappresentare e delimitare gli elementi.
- Esempi:

`<P>Questo e' un paragrafo</P>`

`<P>Paragrafo contenente del
testo enfaticizzato</P>`

`
`

Alcuni caratteri speciali

- ❑ ` ` ("spazio")
- ❑ `<` (<) `>` (>)
- ❑ `&` (&)
- ❑ `"` ("")
- ❑ `à` (à) `Á` (Á)
- ❑ `è` (è) `é` (é)
- ❑ `®` (®)
- ❑ `Ä` (Ä) `ü` (ü)

Sintassi generale degli elementi

<nome_elemento attributi> contenuto </nome_elemento>

Tag di inizio

Tag di fine

- ❑ Per alcuni elementi il tag di fine è opzionale (finiscono col tag di fine dell'elemento successivo)
- ❑ Per gli elementi HEAD e BODY anche il tag di inizio è opzionale
- ❑ Per alcuni elementi il contenuto è vuoto
- ❑ **Non esiste errore di sintassi nei tag**
 - ❖ Tag errati vengono semplicemente ignorati e considerati parte del testo della pagina

TAG con attributi

- ❑ Alcuni elementi ammettono attributi, che vengono specificati nel TAG di inizio con una sintassi del tipo:

nome = valore

- ❑ Il valore deve essere racchiuso tra apici singoli o doppi (a meno che non sia costituito solo da caratteri base)
- ❑ Esempi:

```
<img src='http://home/mydir/xxx.gif' >
```

```
<input type=image src="map.gif">
```

Commenti

□ Vengono delimitati da:

<!-- (inizio)

--> (fine)

□ Possono essere inseriti dovunque

□ Esempi:

<!--Ecco un commento-->

<p> <!-- questo NON e' un commento-->

Alcuni Tipi di dato base HTML

- ❑ **Stringhe di testo**
 - ❑ **Identificatori** (iniziano per lettera, possono contenere caratteri alfanumerici e i caratteri speciali - _ : .)
 - ❑ **URI**
 - ❑ **Colori:** possono assumere due forme
 - ❖ il nome del colore (Es. **Black**, **Silver**, ecc.)
 - ❖ un carattere # seguito da un numero esadecimale su 6 cifre
- Es: **Black** = "#000000"
- Gray** = "#808080"

Lunghezze

□ Possono assumere diverse forme:

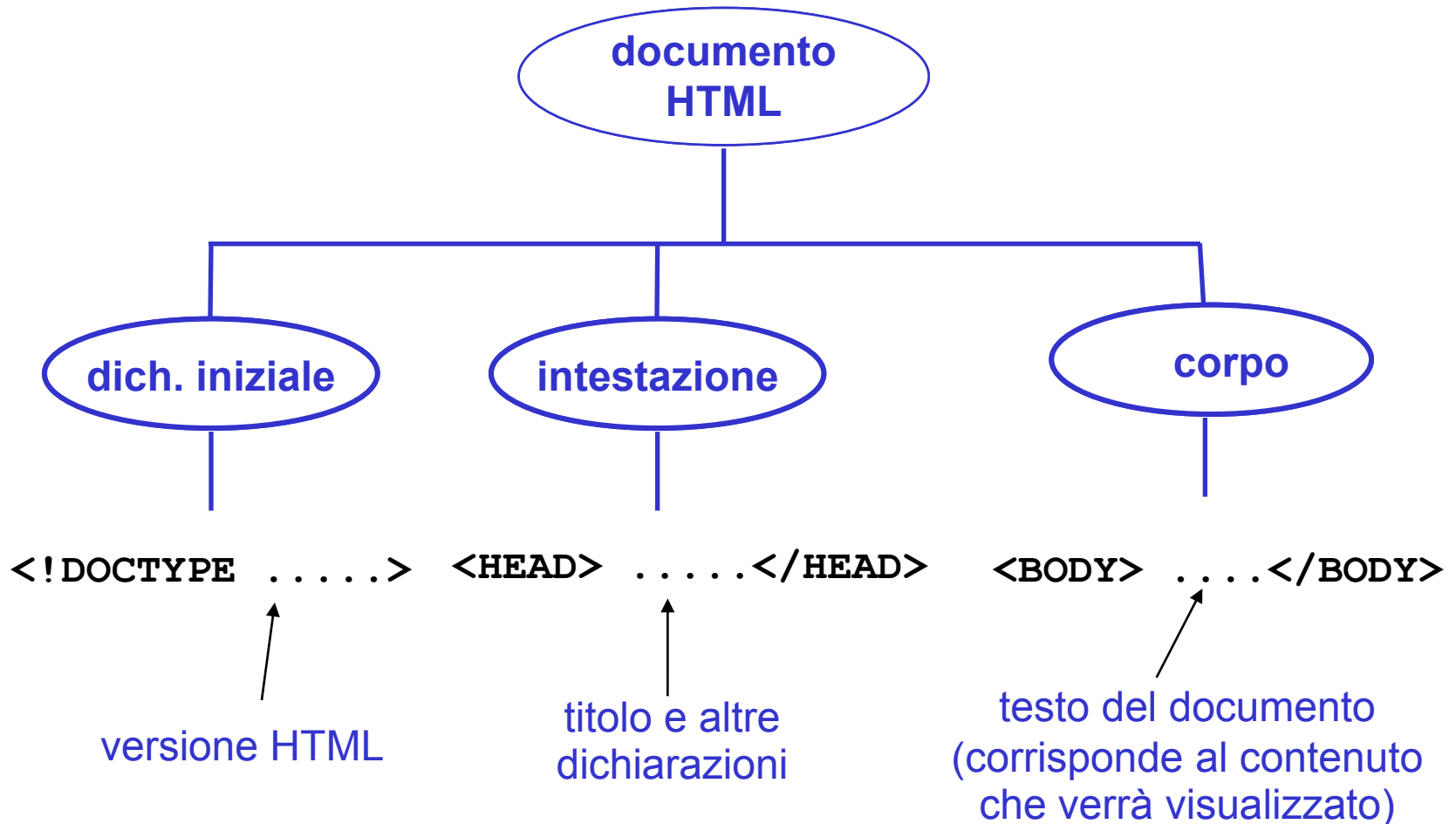
- ❖ 50 (un numero di pixel)
- ❖ 50% (una percentuale dello spazio disponibile)
- ❖ 50* (una lunghezza relativa)

I 3 tipi di lunghezza possono convivere: prima vengono allocati i pixel e le percentuali. Lo spazio rimasto viene diviso proporzionalmente alle lunghezze relative.

Esempio: uno spazio di 200 pixel deve essere diviso in 4 campi con lunghezze: 20,3*,30%,5*

=> Le lunghezze risultanti saranno (in pixel):
20,45,60,75

Struttura di un documento HTML



L'elemento iniziale

□ Specifica

- ❖ qual è il DTD della versione HTML usata
- ❖ l'URI del DTD

□ Esempi:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"  
"http://www.w3.org/TR/html4/strict.dtd">
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01  
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
```

Attributi di <HTML>

□ DIR

- ❖ Specifica la direzione nella quale mostrare il testo (sin.-des. **default**, alto-basso, des.-sin.)
- ❖ Dipende dalla lingua e dal contenuto

□ Lang

- ❖ Specifica la lingua utilizzata
- ❖ Teoricamente, il browser dovrebbe usare questa informazione per rendere meglio il contenuto

Gli elementi dell'intestazione

- **Titolo** (ogni documento HTML deve contenerne uno):

`<TITLE>Documento di prova</TITLE>`

- **Indirizzo base per gli URL relativi**

`<BASE HREF="http://myhost/mydir/index">`

Tag <meta>

- ❑ Appare nella sezione **<HEAD>**
- ❑ Serve per
 - ❖ Memorizzare informazioni
 - ❖ Caricare/scaricare la pagina
 - ❖ ...
- ❑ Usato anche nei motori di ricerca
- ❑ sono coppie nome-valore (**NAME-CONTENT**)
 - ❖ **NAME** è il nome della proprietà
 - ❖ **CONTENT** è il valore associato

Uso di <META>

□ Tipici valori per NAME:

- ❖ **author** : autore del documento
- ❖ **description**: descrizione della pagina
- ❖ **keywords**: parole che descrivono il contenuto (usate dai motori di ricerca)
- ❖ **generator**: software che ha generato il documento

□ Esempi

```
<META name="author" content="Mario Rossi">
```

```
<META name="description"
```

```
content="Tecnologie di Sviluppo per il Web">
```

```
<META name="keywords" content="HTTP, HTML, CSS, PHP">
```

```
<META name="generator" content="MS WORD">
```

Refresh della pagina

- ❑ Bisogna usare l'attributo **HTTP-EQUIV**
- ❑ Si può indicare entro quanti secondi ricaricare la pagina

```
<META HTTP-EQUIV="refresh"  
CONTENT="30">
```

- ❑ Si può indicare quale altra pagina ricaricare ed entro quanti secondi

```
<META HTTP-EQUIV="refresh"  
CONTENT="30",URL="y">
```

y può essere anche un file audio

Altri parametri

- ❑ **HTTP-EQUIV** può essere usato per dare indicazioni al browser
- ❑ Come se fossero delle intestazioni aggiuntive rispetto alla risposta HTTP
- ❑ Esempio

```
<META HTTP-EQUIV="Content-type"  
  CONTENT="text/html";CHARSET="GB2312">
```

Bisogna usare i caratteri cinesi di GB2312 per il testo

Premessa

- ❑ HTML è nato originariamente per descrivere sia la struttura del contenuto sia la sua formattazione/impaginazione
 - ❖ Struttura: titoli, paragrafi, tabelle, ecc.
 - ❖ Formattazione/impaginazione: tipo di font, colore del font, bordi degli oggetti, layout, ecc.
- ❑ Per maggiore flessibilità oggi si usa
 - ❖ **HMTL** per descrivere struttura e contenuto (es. testo)
 - ❖ **Style sheets (es. CSS)** per descrivere formattazione/impaginazione (molto più potente di quanto presente in HTML)
- ❑ Comunque molti browser supportano ancora impaginazione descritta in HTML (uso sconsigliato, anche se alcuni attributi sono di uso comune)

Tag <BODY>

- ❑ Contiene il corpo della pagina
- ❑ Ha degli attributi, il cui uso è però deprecato

Testo Formattato

□ Titoli di paragrafo (section headings)

Sono previsti 6 livelli di titoli denominati H1 ... H6

```
<H1>Gli animali</H1>
```

```
<H2>I mammiferi</H2>
```

□ Capoverso

```
<P>Testo del capoverso:  
puo' estendersi su piu' linee</P>
```

□ Testo pre-formattato

```
<PRE>
```

```
xxxx
```

```
yyyy
```

```
zzzz
```

```
</PRE>
```


Esempio di titoli

`<body>`

`<h1> Titolo 1 </h1>`

`<h2> Titolo 2 </h2>`

`<h3> Titolo 3 </h3>`

`<h4> Titolo 4 </h4>`

`<h5> Titolo 5 </h5>`

`<h6> Titolo 6 </h6>`

`</body>`



Testo Formattato (II)

□ Interruzioni di linea

- ❖ Non corrispondono alle interruzioni del testo HTML
- ❖ Vengono inserite con un apposito TAG:

Essere o non essere.
Questo è ...

□ Barre orizzontali

<HR>

<HR SIZE=4 WIDTH=60% ALIGN=CENTER>

Uso dei Font

□ Testo *enfattizzato*

Testo enfattizzato

□ Testo *in corsivo*

Testo <I>in corsivo</I>

□ Testo in grassetto

Testo in grassetto

□ Testo tipo macchina da scrivere

Testo <TT>tipo macchina da scrivere</TT>

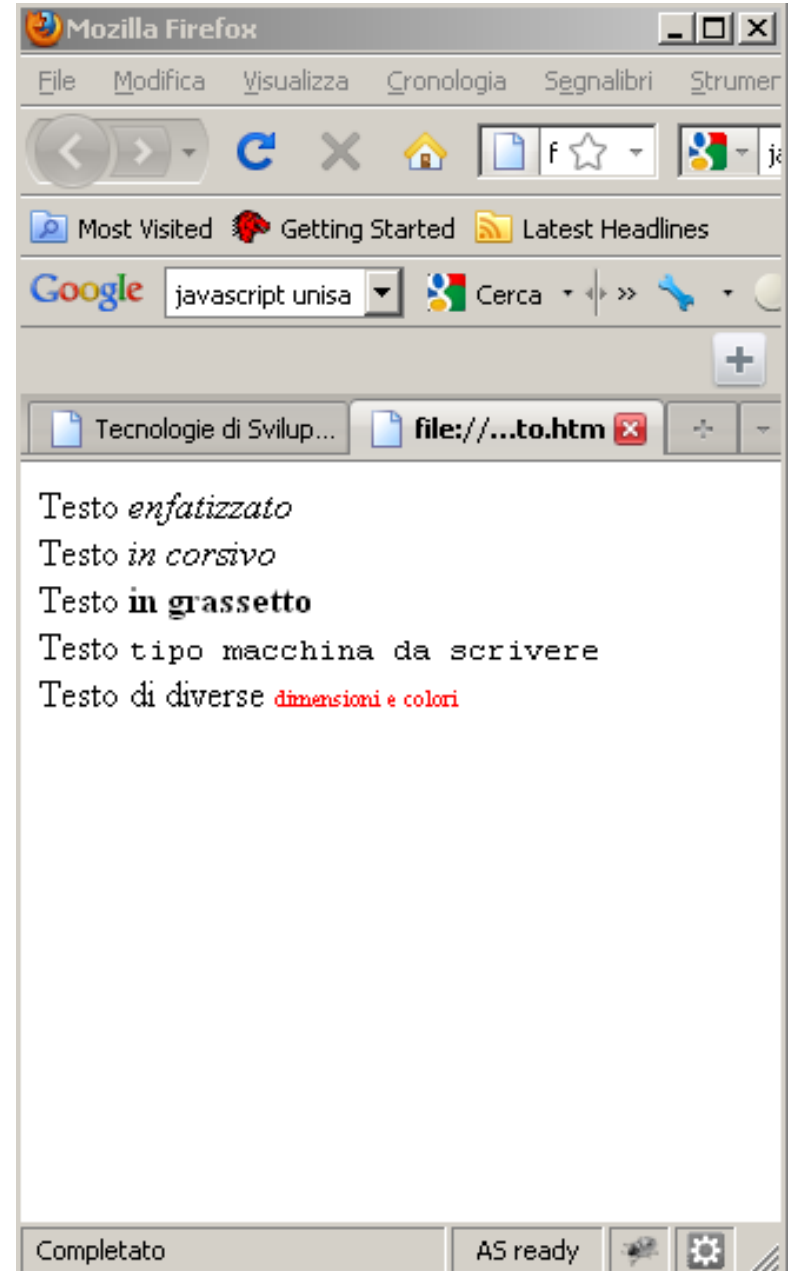
□ Testo di diverse dimensioni e colori (sconsigliato)

Testo di diverse <FONT SIZE=-2

COLOR=red>dimensioni e colori

Esempio font

```
<body>
Testo <EM>enfaticizzato</EM>
<br>
Testo <I>in corsivo</I>
<br>
Testo <B>in grassetto</B>
<br>
Testo <TT>tipo macchina
da scrivere</TT> <br>
Testo di diverse
<FONT SIZE=-2 COLOR=red>
dimensioni e colori
</FONT>
</body>
```



Liste

- ❑ Esempio di Lista ordinata (numerata): ordered list (OL)

```
<OL>  
<LI>Primo elemento  
<LI>Secondo elemento  
</OL>
```

- ❑ Cambiando i TAG di inizio e fine si ottengono altri tipi di lista:

	lista non ordinata
<DIR>	lista a elenco (può essere multicolonna)
<MENU>	lista a menù
<DL>	lista a definizione. Invece di usa i tag <DT> (definition term) e <DD> (definition description)

Esempio

```
<OL>
<LI>Primo elemento
<LI>Secondo elemento
</OL>
<UL>
<LI>Primo elemento
<LI>Secondo elemento
</UL>
<MENU>
<LI>Primo elemento
<LI>Secondo elemento
</MENU>
<DL>
<DT>Primo</DT> <DD>elemento</DD>
<DT>Secondo <DD> elemento
</DL>
```

1. Primo elemento
2. Secondo elemento

- Primo elemento
- Secondo elemento

- Primo elemento
- Secondo elemento

Primo
 elemento
Secondo
 elemento

Immagini

- ❑ I formati riconosciuti dai browser sono
 - ❖ JPEG (.jpeg, .jpg)
 - ❖ GIF (.gif)
 - ❖ Portable Network Graphics (.png)

- ❑ Vengono normalmente inseriti con il tag **IMG**

Attributi di IMG (1)

❑ **SRC="uri"**

❖ URI dell'immagine da inserire

❑ **ALT="testo"**

❖ Testo da visualizzare nel caso in cui non si possa visualizzare l'immagine

❑ **BORDER="numero"**

❖ Spessore (in pixel) del bordo attorno all'immagine

❑ **WIDTH="numero", HEIGHT="numero"**

❖ Larghezza ed altezza (in pixel) dell'immagine

Attributi di IMG (2)

□ **VSPACE="numero" HSPACE= "numero"**

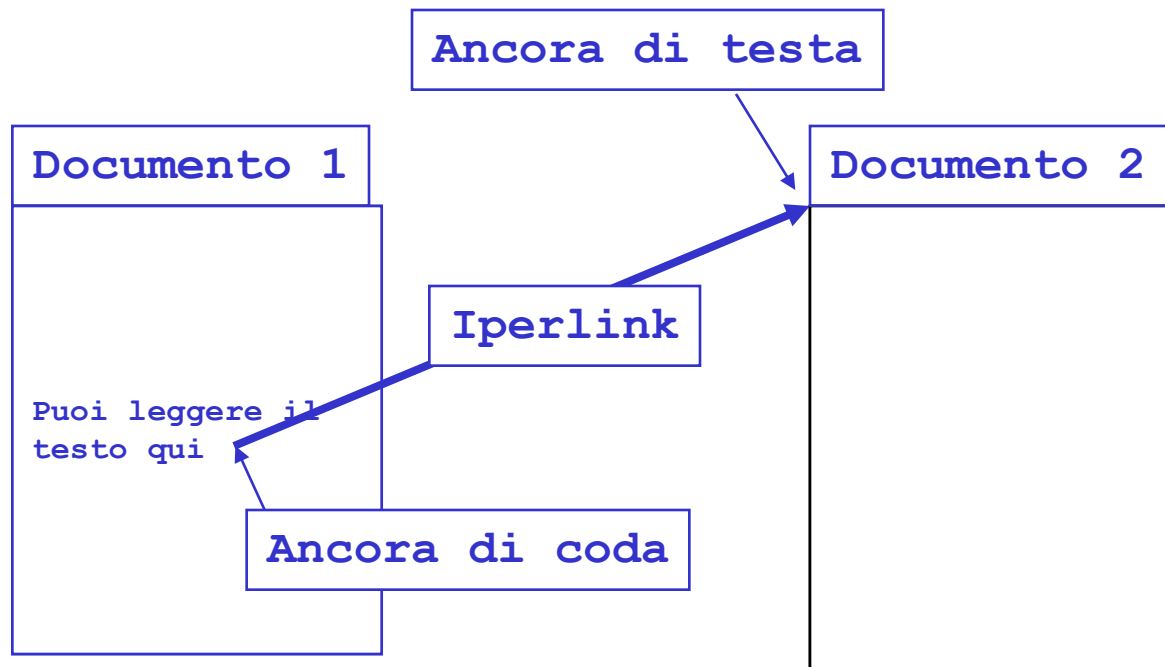
- ❖ Spazio da lasciare verticalmente (sopra e sotto) e orizzontalmente (a destra e sinistra) attorno all'immagine

□ **ALIGN= "posizione" (deprecato)**

- ❖ **LEFT** imm. allineata al margine sin. del testo
- ❖ **RIGHT** imm. allineata al margine des. del testo
- ❖ **TOP** testo allineato al margine sup. dell'imm.
- ❖ **MIDDLE** testo allineato al centro dell'imm.
- ❖ **BOTTOM** testo allineato al margine inf. dell'imm.

Iperlink

- ❑ Sono relazioni tra due **ancore**: quella di testa e quella di coda
- ❑ Permettono la navigazione



Ancore di testa

- Vi si fa riferimento usando un URL, opzionalmente seguito da un indicatore di frammento, che specifica una parte o vista della risorsa

<http://www.polito.it/corsi.html#p2>

URL

Frammento

- Per definire l'inizio di un frammento all'interno di un documento HTML si usa un elemento ancora:

```
<A NAME="p2">
```

Inserimento di Iperlink

- ❑ Per inserire un iperlink navigabile con un click del mouse si inserisce un elemento ancora contenente la specificazione dell'ancora di testa cui si vuole puntare:

```
<A HREF="http://www.polito.it/corsi.html#p2">
```

- ❑ Le immagini sono iperlink la cui risorsa (ancora di testa) viene caricata automaticamente (senza click del mouse):

```
<IMG SRC= "aircraft.gif" ALIGN=TOP>
```

```
<IMG SRC= "triangle.xbm" ALT="Warning:"> Be  
sure to..
```

Come può essere l'URI

□ Relativo

- ❖ `aircraft.gif`
- ❖ `../img.jpeg`

□ Assoluto

- ❖ `http://www.polito.it/corsi.html`
- ❖ `http://www.polito.it/corsi.html#p2`

Ancora con nome

□ È possibile dare dei nomi a parti di documenti web

□ Esempio

` Testo `



□ L'identificatore deve essere **unico nel documento**

□ Per indirizzare UniqueID si usa

`.....`

`.....`

Esempio

<BODY>

Info su Argomento 1.

Info su Argomento 2.

Info su Argomento 3.

<HR> [...altro...]

<H2> Argomento 1</H2>

Testo relativo all'argomento 1. [...altro...]

<H2> Argomento 2</H2>

Testo relativo all'argomento 2. [...altro...]

<H2> Argomento 3</H2>

Testo relativo all'argomento 3.

</BODY>

Risultato

Info su [Argomento 1](#).

Info su [Argomento 2](#).

Info su [Argomento 3](#).

[...altro...]

Argomento 1

Testo relativo all'argomento 1. [...altro...]

Argomento 2

Testo relativo all'argomento 2. [...altro...]

Argomento 3

Testo relativo all'argomento 3.

Attributi

❑ **TARGET="nome"**

- ❖ Indica quale è il frame in cui verrà caricato il documento

❑ **TITLE="nome"**

- ❖ Titolo del documento indirizzato
- ❖ Fornisce il subject in caso di spedizione email

❑ **ACCESSKEY**

- ❖ Il documento viene caricato non al click del mouse, ma quando si preme **ALT+tasto** specificato

Attributi (2)

□ **TABINDEX="numero"**

- ❖ Il valore è un intero positivo
- ❖ Serve a specificare l'ordine con cui il browser sposterà il **focus** quando si schiaccia il tasto TAB
- ❖ Utile per usabilità del sito (es. navigazione tramite tastiera)

```
<a href="home1.html" tabindex=3> Documenti </a><BR>  
<a href="home2.html"    tabindex=1> trasparenze</a><BR>  
<a href="home3.html"    tabindex=2> Registrazioni </a>
```



Spedizione di email nei link

- L'azione specificata in un'ancora può consistere nella spedizione di un messaggio
- Esempio

```
<A HREF="mailto:ciminiera@polito.it">
```

```
Cliccando qui si spedisce un  
messaggio </A>
```



Tabelle

- ❑ Una tabella è una struttura per allineare elementi in righe e colonne
- ❑ Ogni tabella può avere una didascalia (caption)
- ❑ Sono configurabili vari attributi. Per esempio:
 - ❖ **width** (larghezza tabella)
 - ❖ **border** (larghezza bordi)
- ❑ Si possono formattare in vario modo le celle
- ❑ Si possono distinguere celle di tipo header (**<TH>**) e celle di tipo data (**<TD>**) .

Esempio di Tabella

```
<TABLE border="1"
summary="This table gives some statistics about fruit
flies: average height and weight, and percentage
with red eyes (for both males and females).">
<CAPTION><EM>A test table with merged cells</EM></CAPTION>
<TR><TH rowspan="2"><TH colspan="2">Average
<TH rowspan="2">Red<BR>eyes
<TR><TH>height<TH>weight
<TR><TH>Males<TD>1.9<TD>0.003<TD>40%
<TR><TH>Females<TD>1.7<TD>0.002<TD>43%
</TABLE>
```

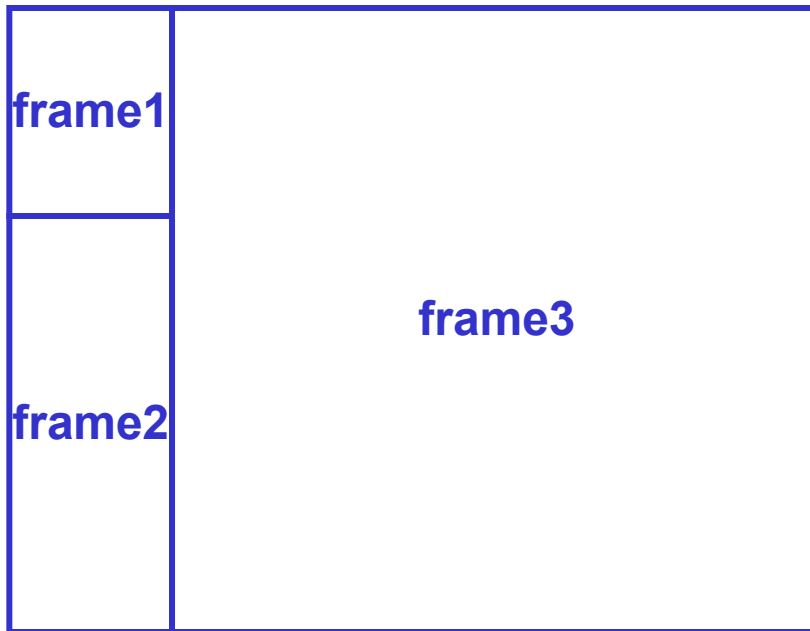
A test table with merged cells

	Average		Red eyes
	height	weight	
Males	1.9	0.003	40%
Females	1.7	0.002	43%

Frame

- ❑ Permettono di presentare documenti costituiti da viste multiple in frame indipendenti
- ❑ Il layout dei frame viene specificato da un elemento FRAMESET che *sostituisce* l'elemento HTML
- ❑ Un singolo FRAMESET definisce un layout a matrice o griglia rettangolare, in cui ogni casella contiene un elemento FRAME o un FRAMESET annidato.
- ❑ Annidando i FRAMESET si possono formare strutture complesse

Esempio



```
<FRAMESET cols="20%, 80%">  
  <FRAMESET rows="100, 200">  
    <FRAME src="frame1.html">  
    <FRAME src="frame2.html">  
  </FRAMESET>  
  <FRAME src="frame3.html">  
</FRAMESET>
```

- ❑ Matrice 1X2 il cui primo elemento è una matrice 2X1

Struttura

```
<FRAMESET lista_attributi>  
<FRAME SRC="URL" lista_attributi>  
<FRAME SRC="URL" lista_attributi>  
<FRAME SRC="URL" lista_attributi>  
</FRAMESET>  
<NOFRAMES>  
<BODY> Codice HTML visualizzato da  
browser che non supporta frames  
</BODY> </NOFRAMES>
```


Principali Attributi

□ FRAMESET

- ❖ **rows**= lista delle lunghezze delle righe (se assente, 1 sola riga)
- ❖ **cols**= lista delle larghezze delle colonne (se assente, 1 sola colonna)
- ❖ Rows e cols possono apparire insieme (caricati per riga)

```
<FRAMESET COLS="30%,70%"ROWS="20%,80%">
```

```
<FRAME SRC="uno.html" >
```

```
<FRAME SRC="due.html" >
```

```
<FRAME SRC="tre.html" >
```

```
<FRAME SRC="quattro.html" >
```

```
</FRAMESET>
```

Altri attributi

- ❑ Attributi **non HTML** ma riconosciuti da Explorer e Firefox (o meglio dai loro «webkit»: webpage rendering library)
 - ❖ **BORDER=<numero>**
 - Indica lo spessore della cornice per tutti i frame figli
 - ❖ **BORDERCOLOR=<colore>**
 - Colore del cordo del frameset
 - ❖ **FRAMEBORDER= 1 (sì) 0 (no)**
 - Indica se bisogna usare o meno un bordo per i frames figli

Attributi di un frame (1)

- ❑ **src** URI del contenuto iniziale del frame
- ❑ **noresize** inibisce il cambio di dimensione
- ❑ **scrolling** "yes", oppure "no", oppure "auto", indica se deve apparire una barra laterale
- ❑ **frameborder** "1" (frame con bordo), oppure "0" (senza bordo)
- ❑ **marginwidth** larghezza del margine orizzontale interno (in pixel)
- ❑ **marginheight** altezza del margine verticale interno (in pixel)

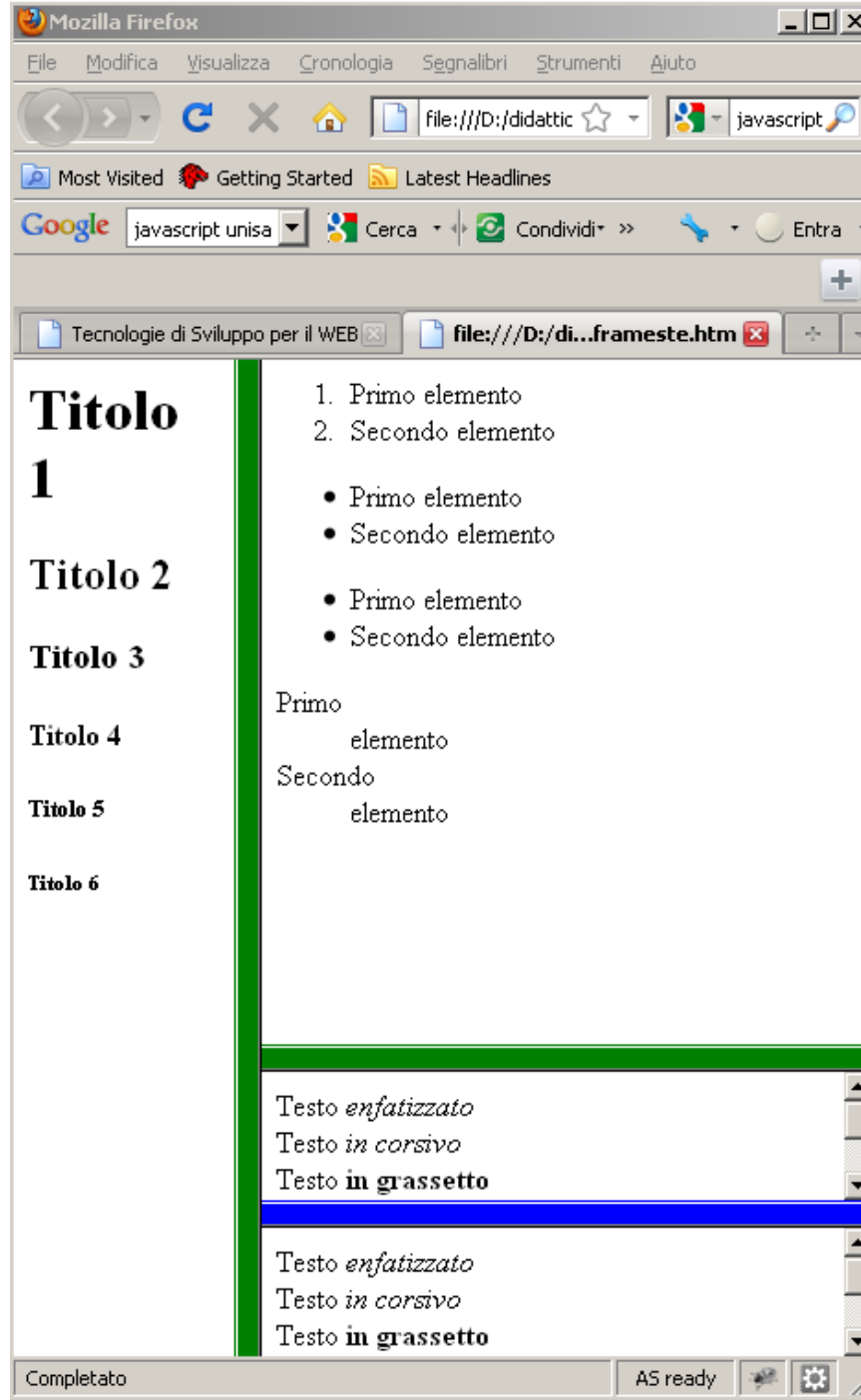
Attributi di un frame (2)

- ❑ **name** definisce un nome per il frame
- ❑ **bordercolor="colore"** colore del bordo (non standard)

Esempio

```
<FRAMESET COLS="120,*" BORDER="15">  
<FRAME SRC="titoli.htm" NAME="indice"  
    BORDERCOLOR="green">  
<FRAMESET ROWS="*,70,70">  
<FRAME SRC="liste.htm" NAME="main"  
    BORDERCOLOR=green>  
<FRAME SRC="testo.htm" NAME="bottom1"  
    BORDERCOLOR=blue>  
<FRAME SRC="testo.htm" NAME="bottom2"  
    BORDERCOLOR=red>  
</FRAMESET>  
</FRAMESET>
```

Risultato



Collegamento tra frame diversi

- ❑ Negli elementi che causano l'apertura di un nuovo documento (A, LINK, AREA, ecc.) è possibile specificare il frame dove esso deve essere aperto, con l'attributo

`target=` *nome del frame*

- ❑ Esempio:

```
<A HREF="nuovo.html" target="left">
```

```
Apertura </A>
```

- ❑ Se non esiste un frame con il nome specificato, il documento viene caricato in una nuova finestra

Identificazione dei frame

- ❑ Un frame può essere identificato con il proprio nome, assegnatogli nel tag di inizio con l'attributo name
- ❑ In alternativa, valgono i seguenti nomi predefiniti:
 - _self** Il frame stesso in cui si trova il riferimento
 - _parent** Il frameset di massimo livello all'interno del file HTML del frame
 - _top** Il frame di primo livello nella gerarchia di annidamento (la finestra principale)
 - _blank** Una nuova finestra

Esempio di prova (1)

□ File di partenza

```
<frameset rows="200,*">  
<frame name="alto" src="./iniziale.htm">  
<frame name="basso"  
  src="framesetinterno.htm">  
</frameset>
```

□ File framesetinterno.htm

```
<FRAMESET COLS="*,*,*">  
<frame name="bassosin" src="./iniziale.htm">  
<frame name="bassocen"  
  src="./testgerarchia.htm">  
<frame name="bassodes" src="./iniziale.htm">  
</FRAMESET>
```

Esempio di prova (2)

□ File iniziale.htm

```
<BODY>
```

```
<h1 align=center> Testo di partenza </h1>
```

```
</BODY>
```

□ File testgerarchia.htm

```
<ul><li> Carica in <a href="liste.htm" target="_top"> top </a>
```

```
<li> Carica in <a href="liste.htm" target="_blank"> una nuova  
pagina </a>
```

```
<li> Carica in <a href="liste.htm" target="_self"> se stesso  
</a>
```

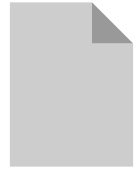
```
<li> Carica in <a href="liste.htm" target="_parent"> parent  
</a>
```

```
<li> Carica in <a href="liste.htm" target="alto"> alto </a>
```

```
<li> Carica in <a href="liste.htm" target="bassosin"> basso a  
sinistra </a>
```

```
<li> Carica in <a href="liste.htm" target="bassodes"> basso a  
destra </a>
```

```
</ul>
```



Caricamento di più frames

- ❑ Si possono ricaricare più frames con un solo click del mouse
 - ❖ Si possono mettere tutti i frames in un frameset, si carica tutto il frameset
 - ❖ Si utilizza del codice Javascript

Deep linking

- ❑ Termine che si riferisce al fatto di caricare all'interno di un frame la pagina di un altro sito
- ❑ Pratica di dubbia legalità
- ❑ Si può evitare:
 - ❖ Facendo aprire le pagine in una nuova finestra (**_blank**) o nella pagina principale (**_top**)
 - ❖ Usando Javascript

IFRAME (In line FRAME)

- ❑ Tag che permette di definire un frame all'interno di una pagina convenzionale (senza frame) - utile per un «preview»
- ❑ Il nuovo frame si comporta come un'immagine (con barre di scorrimento)
- ❑ Non si deve usare **frameset**
- ❑ Attributi
 - ❖ **Align, name, src, frameborder, height, width, maginwidth, marginheight, scrolling**



Frame default

- ❑ Se non si vuole specificare sempre il frame di destinazione, se ne può indicare uno default
- ❑ Nella sezione **<HEAD>** si inserisce il tag **<BASE>**
- ❑ Esempio
<BASE TARGET="main">

Contenuto alternativo

- Un elemento FRAMESET può contenere un elemento NOFRAMES per specificare cosa deve essere visualizzato nel caso in cui non siano supportati i frame

- Esempio:

```
<NOFRAMES>
```

```
<P>This frameset document contains:
```

```
<UL>
```

```
<LI><A href="contents_of_frame1.html">Some neat  
contents</A>
```

```
<LI><IMG src="contents_of_frame2.gif" alt="A neat  
image">
```

```
<LI><A href="contents_of_frame3.html">Some other neat  
contents</A>
```

```
</UL></NOFRAMES>
```

Inclusione di Oggetti Multimediali

- ❑ Fino alla versione 3, gli unici oggetti multimediali che potevano essere inclusi in un documento HTML erano le immagini (elementi IMG) e gli applet Java (elementi APPLET)
- ❑ Dalla versione 4 è stato introdotto l'elemento OBJECT, che consente maggiore flessibilità:
 - ❖ Permette di trattare uniformemente tutti gli oggetti esterni multimediali (compresi immagini e applet)
 - ❖ Consente di inserire la visualizzazione di elementi esterni tipo plug-ins (Java applets, ActiveX, PDF, Flash)
 - ❖ Si presta ad includere anche tipi di oggetti che saranno definiti in futuro
 - ❖ Presente anche in HTML5 seppur con meno attributi

L'Elemento OBJECT

□ Permette di specificare:

- ❖ L'implementazione dell'oggetto (il codice) - se necessario
 - **classid** URI del codice
 - **codetype** content type del codice (tipo MIME)
- ❖ I dati associati all'oggetto (il file dell'immagine, la serializzazione dell'applet Java da ricostruire, ecc.)
 - **data** URI dei dati
 - **type** content type dei dati
- ❖ Eventuali dati necessari a run time (tipicamente dati per l'inizializzazione degli applet)
 - Elementi **<PARAM>** annidati
- ❖ La Modalità di visualizzazione
 - **height, width, ecc.**

Esempio

❑ Inclusione di Contenuto Flash (semplificato)

```
<OBJECT type="application/x-shockwave-flash"  
  data="movie_name.swf" width="550" height="400">  
  <PARAM name="movie" value="moviename.swf"/>  
</OBJECT>
```

Form

- ❑ Un FORM è un modulo che l'utente può compilare tramite il browser
- ❑ A compilazione terminata il browser invia il modulo compilato al server, che accede ad una risorsa e restituisce come risposta una pagina HTML
- ❑ Struttura di un form:

```
<FORM ACTION="http://www.org.sample"  
  METHOD=GET>  
  ...  
  ...  
</FORM>
```

Metodo HTTP usato

Risorsa cui accedere

Campi del FORM ("controls")

Attributi (1)

❑ ACTION

- ❖ URI del file eseguibile (o script) a cui indirizzare i dati del form

❑ METHOD

- ❖ Indica il metodo HTTP che si vuole usare per inviare la richiesta
- ❖ Sono ammessi solo **GET** e **POST**

❑ ENCTYPE

- ❖ Specifica come i dati del form sono codificati nella richiesta, quando si usa POST (per GET è standard)

Attributi (2)

❑ **TARGET**

- ❖ Indica in quale frame deve apparire la pagina di risposta

❑ **ID**

- ❖ Nome del form (utile in JavaScript per accedere ai suoi elementi)

❑ **NAME**

- ❖ Vecchio attributo, sostituito da ID

Campi INPUT

- ❑ Sono generici campi per l'input di dati.
- ❑ Vengono creati con elementi INPUT (senza contenuto)
- ❑ Possono assumere diverse forme:
 - ❖ Text Field/Password Field
 - ❖ Check Box
 - ❖ Radio Button
 - ❖ Image Pixel
 - ❖ Hidden Field
 - ❖ Button (Submit/Reset/Generic)
 - ❖ File Selection

Attributi

❑ **NAME**

- ❖ Identifica il controllo

❑ **ID**

- ❖ Come per l'analogo attributo di FORM

❑ **VALUE**

- ❖ Valore **iniziale** del controllo

❑ **TYPE**

- ❖ Identifica il tipo di controllo (v. lista di prima)

Attributi (2)

☐ **READONLY**

- ❖ Attributo booleano (se appare è vero, se no è falso)
- ❖ Il contenuto del controllo è a sola lettura, ma verrà comunque inviato al server, come se fosse stato inserito dall'utente
- ❖ Ad es. dati inseriti precedentemente che vengono ripresi nel nuovo form e rispediti al server

☐ **DISABLED**

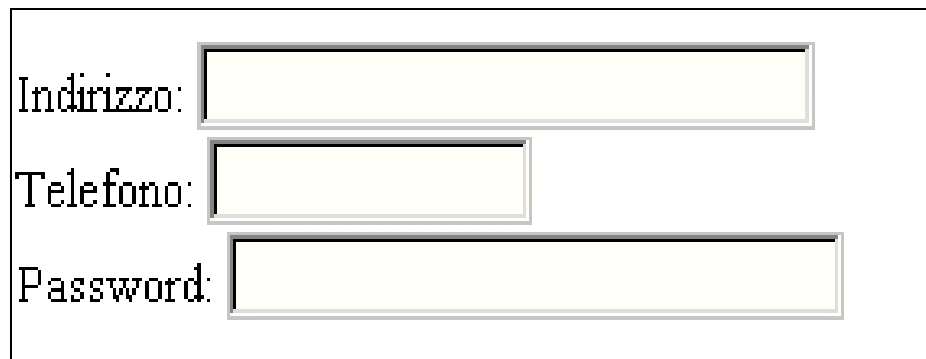
- ❖ Attributo booleano
- ❖ Il controllo non viene mai mostrato, ma il suo contenuto viene inviato nella richiesta HTTP
- ❖ Può servire per passare valori da un form all'altro

Text Field e Password Field

Indirizzo: `<INPUT TYPE=TEXT NAME=indirizzo>`
`
`

Telefono: `<INPUT TYPE=TEXT NAME=telefono`
`SIZE=10 MAXLENGTH=10>`

Password: `<INPUT TYPE=PASSWORD NAME=passwd>`



Indirizzo:

Telefono:

Password:

Check Box

```
<p>A quali generi sei interessato? <BR>
<INPUT TYPE=CHECKBOX NAME=genere
  VALUE=west>western<BR>
<INPUT TYPE=CHECKBOX NAME=genere
  VALUE=thr>thriller<BR>
<INPUT TYPE=CHECKBOX NAME=genere
  VALUE=fant>fantasy<BR>
```

A quali generi sei interessato?

☐ western

☐ thriller

☐ fantasy

Radio Button

```
<p>Corso di appartenenza:<BR>
<INPUT TYPE=RADIO NAME=corso
  VALUE=inf>INF<BR>
<INPUT TYPE=RADIO NAME=corso
  VALUE=eln>ELN<BR>
<INPUT TYPE=RADIO NAME=corso VALUE=alt
  CHECKED>Altro<BR>
```

Corso di appartenenza:

- ☐ INF
- ☐ ELN
- ☒ Altro

Button

Ve ne sono di 3 tipi

SUBMIT scatena le operazioni di invio dati del form

RESET re-inizializza tutti i controlli del form

BUTTON non ha azioni standard associate in HTML, si deve usare uno script

```
<INPUT TYPE=SUBMIT VALUE=INVIA>
```

```
<INPUT TYPE=RESET VALUE=AZZERA>
```

```
<INPUT TYPE=BUTTON VALUE=OK>
```



Invio form

- ❑ Possono esserci più pulsanti di tipo SUBMIT
- ❑ Il valore associato al pulsante viene inviato nella richiesta come gli altri parametri

```
<INPUT type=submit name=azione  
value=Aggiorna>
```

```
<INPUT type=submit name=azione  
value=Aggiungi>
```

```
<INPUT type=submit name=azione  
value=Cancella>
```

File Selection (1)

- ❑ Serve per caricare un proprio file sul server
- ❑ Da usare con **POST** (modifica lo stato del server)

<p>Upload del file:

```
<INPUT TYPE=FILE NAME=filename  
VALUE="pippo.txt">
```

Upload del file:

File Selection (2)

- ❑ Esiste un attributo **ACCEPT** che permette di specificare i tipi di file gestiti
- ❑ Controlli dei browser non affidabili
- ❑ Lista di tipi MIME accettati, incluse eventuali wild card
- ❑ Esempio che accetta qualsiasi tipo di file di testo

ACCEPT="text/*"

Tipo IMAGE

□ Sintassi

```
<INPUT TYPE=IMAGE NAME="nome"  
  SRC=URI_immagine>
```

- Viene visualizzata l'immagine specificata dall'URI
- Funziona come un tasto SUBMIT
- Nella richiesta vengono inviate le coordinate del punto del mouse nella forma **nome.x**, **nome.y** (in pixel a partire dall'angolo in alto a sinistra)

Altri campi

- ❑ Oltre agli elementi INPUT sono previsti altri elementi per realizzare altri tipi di campi:
 - ❖ Menù di selezione
 - ❖ Text Area
 - ❖ Altri tipi di Bottoni

Menù di selezione

<p>Corso di appartenenza:

<SELECT NAME=corso SIZE=3>

<OPTION VALUE=inf>INF

<OPTION VALUE=eln>ELN

<OPTION VALUE=alt

SELECTED>Altro

</SELECT>

Con size=1 (default)
diventa un menù a tendina

Corso di appartenenza: 

Corso di appartenenza: 

Ancora su SELECT

- ❑ Normalmente, si comporta come un radio button

```
<INPUT TYPE="radio" VALUE=inf> INF
```

- ❑ Se appare l'attributo booleano **MULTIPLE** si comporta come un checkbox (permette selezioni multiple)



Textarea (1)

- ❑ Serve per inserire testo su più righe
- ❑ Utilizzata di solito con POST
- ❑ Può anche contenere inizialmente un testo default
- ❑ Attributi principali:
 - ❖ **NAME** nome del controllo
 - ❖ **ROWS** numero di righe della finestra
 - ❖ **COLS** numero di colonne della finestra

Text Area(2)

`<p>Suggerimenti`

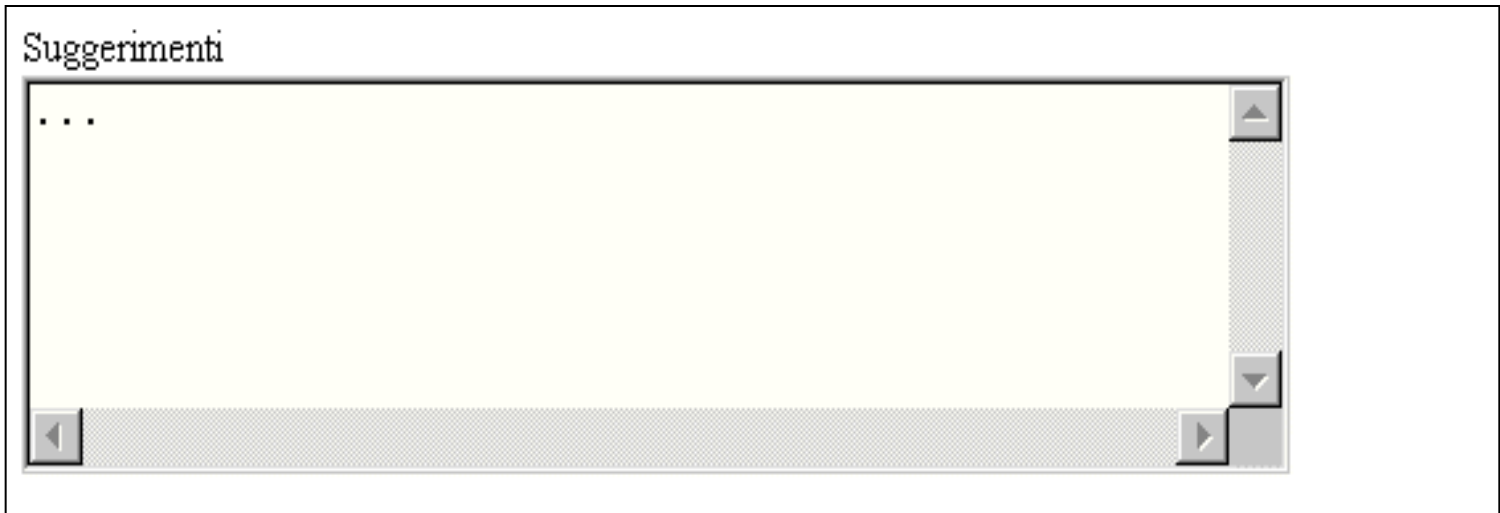
`<TEXTAREA NAME=suggerimenti ROWS=6`

`COLS=50>`

`...`

`</TEXTAREA>`

Il testo eventualmente inserito qui
è il contenuto iniziale della text area

A screenshot of a web browser window. The title bar of the browser is not visible. The main content area shows a text input field. Above the text field, the word "Suggerimenti" is displayed. The text field itself is a rectangular box with a light yellow background. Inside the box, the text "..." is visible. The text field has a standard scrollbar on the right side, with a vertical track and a scrollable area. The scrollbar is currently at the top position.

Form: Considerazioni

- ❑ I form sono un semplice modo per creare interfacce grafiche senza necessità di far girare uno specifico programma
- ❑ Forniscono
 - ❖ campi di input
 - ❖ meccanismi di scelta/selezione
 - ❖ Pulsanti
- ❑ In unione coi meccanismi successivi consentono di realizzare «applicazioni» dotate di interfaccia grafica

Protocollo HTTP

- ❑ Protocollo applicativo semplice e leggero per sistemi informativi distribuiti e iper-mediali
- ❑ Specificato negli RFC (2616, v 1.1 giugno 1999)
- ❑ Alla base del web
- ❑ Oltre al trasferimento dei contenuti supporta:
 - ❖ Descrizione dei contenuti (tipo ecc.)
 - ❖ Utilizzo di politiche di caching
 - ❖ Meccanismi di autenticazione

HTTP: Caratteristiche

- ❑ Protocollo applicativo
- ❑ Semplice (richiesta / risposta)
- ❑ Privo di stati
 - ❖ Server: non necessario ricordare richieste precedenti
- ❑ Indipendente dal contenuto
 - ❖ Può trasportare dati arbitrari (dimensione, codifica)

HTTP: Struttura base

- ❑ Richiesta e risposta sono formate da
 - ❖ Intestazione (header), terminata da coppia CR-LF (0x0d, 0x0a, ossia \r\n in notazione C)
 - ❖ Corpo

```
GET / HTTP/1.1
Host: media.polito.it
```

```
POST /query HTTP/1.1
Host: example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 30

firstname=Mario&lastname=Rossi
```

URL

- ❑ Tutte le transazioni HTTP sono dirette a risorse specificate attraverso un URL
 - ❖ Uniform Resource Locator (RFC 2396)
 - ❖ Esempi:
 - ❖ `http://www.polito.it`
 - ❖ `http://localhost:8000/myfolder`

Richieste

- ❑ Prima riga: Azione URL versione
- ❑ Righe successive: altri headers, riga bianca, corpo

```
POST /query HTTP/1.1
Host: example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 30

firstname=Mario&lastname=Rossi
```



Tipo MIME

HTTP: Azioni possibili

□ GET

- ❖ Richiede l'invio della risorsa corrispondente all'URL
- ❖ Non c'è corpo
- ❖ Possibili diverse richieste (condizionali, parziali)

□ HEAD

- ❖ Come GET ma solo header (in richiesta e risposta)

□ POST

- ❖ Invia le informazioni contenute nel corpo della richiesta alla risorsa indicata e richiede la risposta
- ❖ La risorsa è tipicamente un programma
- ❖ Tipicamente modifica lo stato sul server

Risposte

- ❑ Prima riga: Versione Codicestato Descrizione
- ❑ Righe successive: altri headers, riga bianca, corpo (se presente)

```
HTTP/1.1 200 Ok
Content-Type: text/html
Server: Apache
Content-Length: 123
```

```
<HTML>...</HTML>
```

- ❑ Content-Length: se non c'e' si deve chiudere connessione
 - ❖ Altrimenti non si saprebbe quando termina il corpo
- ❑ Codici di stato e descrizione (esempio):
 - 200 Ok
 - 301 Moved permanently
 - 403 Forbidden
 - 500 Internal server error

Codifica e Invio di un FORM

- ❑ Quando l'utente schiaccia il bottone submit, si svolgono le seguenti operazioni:
 - ❖ Vengono identificati i cosiddetti "successful controls"
 - ❖ Viene costruito il Form Data Set (sequenza di coppie nome-valore, una per ciascun successful control)
 - ❖ Il Form Data Set viene codificato
 - ❖ Il Form Data Set codificato viene inviato tramite il metodo HTTP specificato
- ❑ La specifica HTML 4 definisce un insieme minimo di codifiche e metodi di invio, ma non ne esclude altri.

Codifica Standard

- ❑ La codifica standard (default) è descritta dal tipo MIME

application/x-www-form-urlencoded

- ❑ Il Form Data Set viene convertito in una stringa con una sintassi che la rende inseribile in un URL:

`nome1=valore1&nome2=valore2& . . .`

- ❑ Per permettere l'uso dei caratteri speciali e dei caratteri '=' e '&', questi vengono trasformati con sequenze di escape:

- ❖ lo spazio viene trasformato in '+'
- ❖ i caratteri speciali vengono trasformati in '%xx'

Altre Codifiche

- ❑ L'attributo **enctype** specifica la codifica da usare
- ❑ Si può scegliere **multipart/form-data** (rfc2388):
 - ❖ Risulta più efficiente nel caso di dati binari o non ASCII di grosse dimensioni
 - ❖ Si basa sul tipo MIME multipart (sequenza di parti, ciascuna con propri attributi: Content-type, Charset, ecc.)
 - ❖ Ogni parte rappresenta un successful control e deve avere un attributo

Content-Disposition: form-data;
name="nome_campo"

Esempio: Form

```
<FORM action="http://server.com/cgi/handle"
      enctype="multipart/form-data" method="post">
<P>
```

```
What is your name? <INPUT type="text" name="submit-
      name"><BR>
```

```
What files are you sending? <INPUT type="file"
      name="files"><BR>
```

```
<INPUT type="submit" value="Send"> <INPUT
      type="reset">
```

```
</FORM>
```

Esempio: Codifica del Data Set

Content-Type: multipart/form-data; boundary=AaB03x

--AaB03x

Content-Disposition: form-data; name="submit-name"

Larry

--AaB03x

Content-Disposition: form-data; name="files";

filename="file1.txt"

Content-Type: text/plain

... contents of file1.txt ...

--AaB03x--

Metodi di Invio

□ Metodo Get

- ❖ Viene eseguito il metodo GET, inviando come URI la concatenazione dell'attributo action con il Data Set codificato, separati da un carattere '?'
- ❖ L'unica codifica ammessa in questo caso è application/x-www-form-urlencoded

□ Metodo Post

- ❖ Viene eseguita una transazione POST usando il valore dell'attributo action e inviando un messaggio che contiene la codifica del Data Set

Inclusione di Script

- ❑ Un documento HTML può contenere codice
 - ❖ scritto in un qualunque linguaggio di scripting (ovviamente il browser deve essere in grado di interpretarlo)
 - ❖ in grado di accedere agli elementi HTML del documento e di intercettare eventi (mouse, tastiera, ...)
- ❑ Gli script sono tipicamente usati:
 - ❖ come meccanismi di Client-Side Programming
 - ❖ per modificare "al volo" il contenuto del documento HTML quando questo viene caricato (realizzazione di HTML «dinamico»)

Script Eseguiti al Caricamento

- ❑ Vengono eseguiti una sola volta, al caricamento del documento
- ❑ Vengono inseriti in elementi SCRIPT
- ❑ Possono avere un contenuto alternativo, specificato in un corrispondente elemento

<NOSCRIPT>

```
<SCRIPT type="text/javascript">
function my_onload() {
. . .
}
var win =
window.open("http://...")
if (win) win.onload = my_onload
</SCRIPT>
<NOSCRIPT>
<P><A href="http://..."> click
</A>
</NOSCRIPT>
```

```
<SCRIPT type="text/vbscript"
src="http://someplace.com/vbcalc">
</SCRIPT>
```

Supporto per Pagine Dinamiche

- ❑ Se uno script eseguito al caricamento produce come output un testo HTML,
 - ❖ l'elemento SCRIPT contenente lo script viene sostituito dal testo prodotto durante il caricamento
 - ❖ Il nuovo testo viene interpretato (può a sua volta contenere elementi script)
- ❑ Il caricamento può quindi dar luogo a sostituzioni ricorsive

```
<TITLE>Test Document</TITLE>  
<SCRIPT type="text/javascript">  
document.write("<p><b>Hello World!<\b>")  
</SCRIPT>
```

Script Eseguiti al Verificarsi di Eventi

- ❑ Vengono inseriti come valori di particolari attributi degli elementi HTML ai quali sono associati gli eventi.
- ❑ Esempi di attributi:
 - ❖ **onload** (termine del caricamento di un frameset o di una finestra)
 - ❖ **onunload**
 - ❖ **onclick** (click del mouse sull'elemento)
 - ❖ **ondblclick** (doppio click del mouse sull'elemento)

```
<INPUT NAME="num"  
onchange="if (!checkNum(this.value, 1, 10))  
{this.focus();this.select();} else  
{thanks()}"  
VALUE="0">
```

La Specifica del Linguaggio

❑ Negli elementi SCRIPT

- ❖ E' obbligatorio inserire un attributo type che specifica il linguaggio usato

❑ Se vengono usati script attivati in corrispondenza di eventi

- ❖ E' necessario che il documento abbia un linguaggio di scripting di default, che può essere specificato

- come meta-informazione:

```
<META http-equiv="Content-Script-Type"  
content="text/tcl">
```

- con un attributo dell'header HTTP

```
Content-Script-Type: text/javascript
```


Accesso agli Elementi HTML

- ❑ Ogni linguaggio di scripting abilitato per l'inclusione in HTML ha meccanismi propri di accesso agli elementi del documento
- ❑ In generale, il riferimento ad un elemento avviene sempre tramite il suo attributo **name** oppure **id**