Manuale di HTML

Prof. Rocco Labbi

ISIS "CARLO MAZZARINO"

Una Premessa **3**

1. Le sezioni di un documento HTML <HEAD> e <BODY> **3**

2. Intestare il documento <HEAD> **4**

3. Il tag body BODY **5**
3.1. BGCOLOR 5
3.2. BACKGROUND 7
3.3. TEXT 7
3.4. LINK 7
3.5. ALINK 8
3.6. VLINK 8
3.7. BGPROPERTIES 8

4. Formattare il testo **8**

5. Paragrafi e Giustificazione **12**
Creare paragrafi con <P> 12
Come andare a capo
 12
Linee orizzontali con <HR> 13

6. Inserire immagini nel documento **16**

7. Come creare elenchi puntati e numerati **17**
Elenchi NON ordinati (puntati) 19

8. I link **20**

9. Le tabelle **23**

10. I FORM **36**

11. I frame **46**

12. I CSS **54**
12.1. Come è fatto un CSS: regole e commenti 55
12.2. Com'è fatta una regola 55
12.3. Commenti 56
12.4. Proprietà singole e a sintassi abbreviata 56
12.5. I selettori 57
12.5.1. Selettore di elementi (type selector) 57
12.5.2. Raggruppamento 57
12.5.3. Selettore universale (universal selector) 58
12.5.4. Selettore del discendente (descendant selector) 58
12.5.5. Selettore del figlio (child selector) 58
12.5.6. Pseudo-classi dei links 59
12.6. Id e classi: due selettori speciali 59
12.6.1. Classe 60
12.6.2. ID 61
12.7. Le proprietà dei caratteri (font) 61
12.8. Proprietà dei colori 62
12.9. Proprietà dei testi 63
12.10. Proprietà dei bordi 64
12.11. Tabelle 65

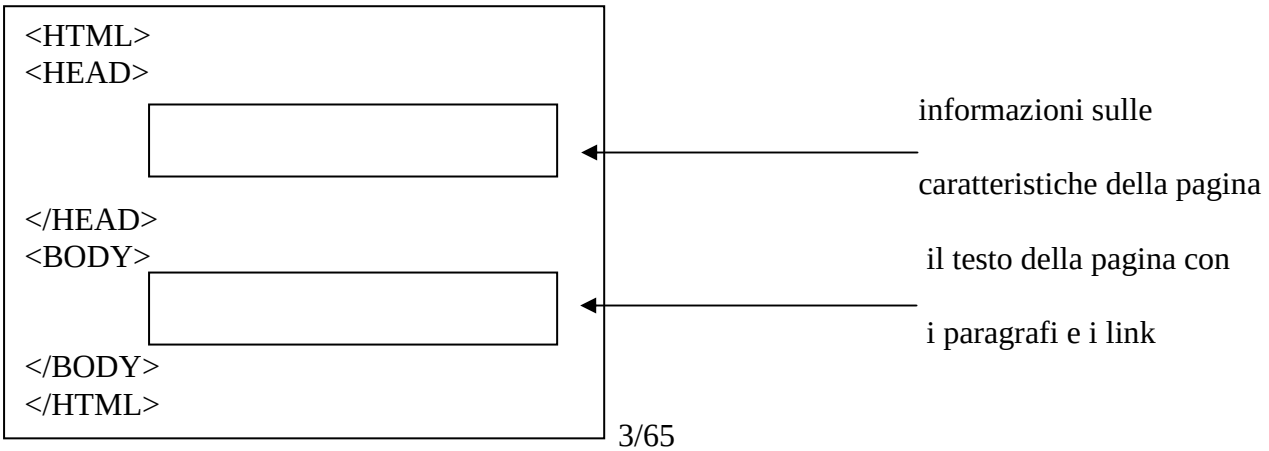
Una Premessa

In questo manuale ci occuperemo di come sia possibile comporre le usuali pagine web che siamo abituati a visualizzare con il nostro browser. Studieremo il linguaggio **HTML** (*Hiper Text Markup Language*). In italiano l’acronimo può essere tradotto con *linguaggio per ipertesti a marcatori*. Vediamo cosa occorre per creare una pagina Web :

1. **Conoscere l'HTML**
   Sarà lo scopo di questo manuale
2. **Un editor per scrivere il codice**
   Oggi il mercato propone software del genere in continuazione, ma per scrivere il codice si può fare anche con il notepad o blocco note di Windows. Ci sono sostanzialmente due tipologie di Editor:
   - Editor Testuali
     Sono editor in cui bisogna scrivere manualmente il codice.
   - Editor WYSIWYG
     L'acronimo WYSIWYG sta per :what you see is what you get, cioè ciò che si vede è ciò che si ottiene in un browser Web. Questi editor permettono di lavorare tramite una interfaccia grafica, non sul codice ma su oggetti con il semplice trascinamento del mouse. Quindi tutto ciò che si fa su questa interfaccia, verrà tradotto automaticamente in html. Questo tipo di editor è di grande aiuto per i principianti, ma hanno di contro che in presenza di errore, difficilmente riescono a risolverlo, inoltre si rischia di creare siti fotocopia.
3. **Un FTP**
   File Transfer Protocol, è un programma che permette di trasferire le pagine create, nel proprio sito Web. Se ne trovano diversi e parecchi gratuiti su internet.
4. **Uno spazio Web**
   E un'area su un server, che fa capo al nostro nome di dominio, nel quale andremo a trasferire e quindi immagazzinare le pagine del nostro sito Web.
   Anche in questo caso si trovano diversi provider che offrono spazio Web gratuito, ma con alcune limitazioni, cioè il nome di dominio del tipo http://www.nomeprovider.it/nome_mio_sito e difficilmente supportano linguaggi di script. Comunque per iniziare va bene anche un servizio del genere.

1. Le sezioni di un documento HTML <HEAD> e <BODY>

Una pagina web è composta da due sezioni: **l’intestazione**(HEAD) ed il **corpo**(BODY) secondo la struttura delineata in seguito:



2. Intestare il documento <HEAD>

Lo scopo dell'HTML è quello di fornire, attraverso comandi chiamati TAG , una formattazione al documento, oltre all'inserimento di immagini ed altri elementi multimediali (filmati, applet ecc.). Il lavoro che uno sviluppatore Web produce all'interno di un documento HTML è indirizzato a fornire tutte le informazioni necessarie al browser per interpretare correttamente la pagina.
Un documento HTML si divide in due parti fondamentali: l'intestazione e il corpo del documento.
E' facile comprendere che il corpo del documento contiene tutti gli elementi della pagina: il testo, le immagini, le applet Java, il codice Javascript e quant'altro viene materialmente visualizzato dal browser.
Al contrario, l'intestazione contiene una serie di informazioni necessarie al browser per una corretta interpretazione del documento, ma non visualizzate all'interno dello stesso. L'intestazione, quindi, ha un ruolo non apparente ma sicuramente fondamentale. Per esempio il titolo della pagina,i termini chiave per i motori di ricerca,
HTML
HEAD
TITLE

Vediamo una descrizione :

<HTML>
Tutti gli elementi ed il contenuto di un documento HTML sono compresi all'interno dei marcatori <HTML></HTML> i quali hanno il compito di aprire e chiudere il file. I tag <HTML></HTML> indicano al browser che il documento è marcato in HTML. Esistono comunque due ragioni per inserire sempre il tag <HTML></HTML> all'interno del documento:
HTML non è l'unico linguaggio di contrassegno presente sul web (si pensi per es. ad XML).
Gli utenti che usano vecchi browser rischiano di visualizzare un documento formattato male.

<HEAD>

Gli elementi <HEAD></HEAD> sono posti immediatamente dopo l'apertura del tag <HTML> e racchiudono l'intestazione vera e propria del documento; cioè tutte le informazioni necessarie al browser, al Webserver ed ai motori di ricerca.
Si tratta del primo elemento letto dal browser. All'interno di <HEAD></HEAD> va inserito il titolo del documento e altre informazioni.
Ecco la sintassi HTML di un documento con i comandi visti:
<HTML>
<HEAD><TITLE> </TITLE></HEAD>
</HTML>
L'elemento **<TITLE></TITLE>** è il più utilizzato all'interno del tag <HEAD>, in quanto fornisce il titolo alla pagina. Il titolo viene visualizzato dai browser nell'intestazione di pagina.
In caso di salvataggio dell'URL con "Aggiungi a preferiti" (per MsIe) e "Aggiungi Segnalibro" (per Netscape) il tag TITLE dà il nome al collegamento, cioè quando si salva l'indirizzo, il browser assegna allo stesso quanto presente all'interno di **<TITLE></TITLE>**.
Il contenuto riportato tra i tag <TITLE></TITLE> è anche utilizzato da alcuni motori di ricerca per indicizzare la pagina e trovare parole chiave.
La corretta sintassi per il tag TITLE è la seguente:
<TITLE>La mia prima home page**</TITLE>**

3. Il tag body BODY

Nella precedente lezione abbiamo visto come creare da zero un documento HTML e come impostare il titolo. L'operazione successiva all'impostazione del documento è la definizione del colore o dell'immagine di sfondo, oltre ai colori dei link attivi e visitati.
<BODY>
L'elemento <BODY> è posto in posizione immediatamente successiva alla chiusura del tag </HEAD> e comunque all'interno degli elementi <HTML></HTML>; ha un tag di apertura e uno di chiusura, ed all'interno di esso si sviluppa il corpo del documento. Se l'elemento <HEAD> conteneva metadati (cioè non materialmente visualizzati dal browser) la funzione del tag <BODY> è quella di mostrare gli oggetti (testo, immagini, suoni, applet ecc) della pagina.
La sintassi corretta per l'elemento <BODY> è la seguente:
<BODY> Contenuto del documento </BODY>
Il tag <BODY> è utilizzato, oltre che per fornire al browser indicazioni sulla posizione degli oggetti nel documento, anche per impostare vari attributi di visualizzazione per il documento. Di seguito vediamo quali.

3.1. BGCOLOR

L'attributo BGCOLOR imposta un colore unitario di sfondo. La sintassi corretta è la seguente:
<BODY BGCOLOR="red">
E' possibile sostituire al nome in inglese, valori esadecimali. Per esempio, il colore rosso (red) si sostiuisce in questo modo:
<BODY BGCOLOR="#ff0000">
L'utilità dei colori esadecimali si ha laddove non si vuole un colore standard ma sfumato o con diversa tonalità. I più diffusi editor HTML prevedono palette per la definizione di colori esadecimali, mentre Paint Shop Pro fornisce, oltre al colore, anche il corrispondente valore esadecimale da copiare/incollare.
Vediamo i più importanti:

	black="#000000"		green="#008000"
	silver="#C0C0C0"		lime="#00FF00"
	gray="#808080"		olive="#808000"
	white="#FFFFFF"		yellow="#FFFF00"
	maroon="#800000"		navy="#000080"
	red="#FF0000"		blue="#0000FF"
	purple="#800080"		teal="#008080"
	fuchsia="#FF00FF"		aqua="#00FFFF"

Colori a prevalenza di ROSSO

DarkRed (#8B0000)	SaddleBrown (#8B4513)	Brown (#A52A2A)	Sienna (#A0522D)	Firebrick (#B22222)
Crimson (#DC143C)	Tomato (#FF6347)	OrangeRed (#FF4500)	DarkOrange (#FF8C00)	Orange (#FFA500)
Coral (#FF7F50)	LightCoral (#F08080)	LightSalmon (#FFA07A)	Salmon (#FA8072)	DarkSalmon (#E9967A)
IndianRed (#CD5C5C)	Gold (#FFD700)	Pink (#FFC0CB)	LightPink (#FFB6C1)	MistyRose (#FFE4E1)
HotPink (#fFF9B4)	DeepPink (#FF1493)	PaleVioletRed (#DB7093)	Medium VioletRed (#C71585)	RosyBrown (#BC8f8f)
Wheat (#F5DEB3)	Burlywood (#DEB887)	Tan (#D2B48C)	Goldenrod (#DAA520)	Chocolate (#D2691E)
DarkKhaki (#BDB76B)	Khaki (#F0E68C)	PaleGoldenrod (#EEE8AA)	DarkGoldenrod (#B8860B)	Moccasin (#FFE4B5)
NavajoWhite (#FFDEAD)	Bisque (#FFE4C4)	PapayaWhip (#FFEFD5)	Oldlace (#FDF5E6)	Cornsilk (#FFF8DC)
AntiqueWhite (#FAEBD7)	FloralWhite (#FFFAF0)	Seashell (#FFF5EE)	LavenderBlush (#FFF0F5)	SandyBrown (#F4A460)

Colori a prevalenza di VERDE

SpringGreen (#00FF7F)	LawnGreen (#7CFC00)	Medium SpringGreen (#00FA9A)	GreenYellow (#ADFF2F)	Chartreuse (#7FFF00)
PaleGreen (#98FB98)	LightGreen (#90EE90)	YellowGreen (#9ACD32)	LimeGreen (#32CD32)	ForestGreen (#228B22)
DarkGreen (#006400)	OliveDrab (#6B8E23)	Dark OliveGreen (#556B2F)	SeaGreen (#2E8B57)	Medium SeaGreen (#3CB371)
Light SeaGreen (#20B2AA)	Dark SeaGreen (#8FBC8B)	Aquamarine (#7FFFD4)	Medium Aquamarine (#66CDAA)	Medium Turquoise (#48D1CC)
Turquoise (#40E0D0)	MintCream (#F5FFFA)			

Colori a prevalenza di BLU

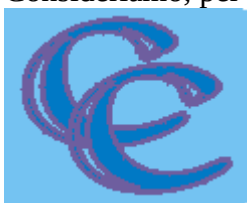
DeepSkyBlue (#00BFFF)	DodgerBlue (#1E90FF)	Cornflower Blue (#6495ED	Medium SlateBlue (#7B68EE)	RoyalBlue (#4169E1)
SlateBlue (#6A5ACD)	Dark Turquoise (#00CED1)	MediumBlue (#0000CD)	Dark SlateBlue (#483D8B)	SteelBlue (#4682B4)
DarkBlue (#00008B)	Indigo (#4B0082)	MidnightBlue (#191970)	SkyBlue (#87CEEB)	LightSkyBlue (#87CEFA)
LightBlue (#ADD8E6)	PowderBlue (#B0E0E6)	LightSteelBlue (#B0C4DE)	CadetBlue (#5F9EA0)	MediumPurple (#9370DB)
BlueViolet (#8A2BE2)	DarkViolet (#9400D3)	DarkOrchid (#9932CC)	MediumOrchid (#BA55D3)	AliceBlue (#F0F8FF)

Colori con due o tre componenti uguali e prevalenti

DarkGray (#A9A9A9)	DimGray (#696969)	SlateGray (#708090)	Light SlateGray (#778899)	LightGrey (#D3D3D3)
DarkCyan (#008B8B)	Gainsboro (#DCDCDC)	LightCyan (#E0FFFF)	Light Goldenrod Yellow (#FAFAD2)	Thistle (#D8BFD8)
Lavender (#E6E6FA)	Azure (#F0FFFF)	Dark SlateGray (#2F4F4F)	GhostWhite (#F8F8FF)	Beige (#F5F5DC)
Snow (#FFFAFA)	LightYellow (#FFFFE0)	Ivory (#FFFFF0)	Pale Turquoise (#AFEEEE)	

3.2. BACKGROUND

BACKGROUND ha una funzione simile a BGCOLOR, ma mentre il secondo mostra una tinta unica del colore, il primo visualizza sullo sfondo un'immagine in formato grafico .gif o .jpg. Consideriamo, per esempio, di voler costruire uno sfondo con l'immagine seguente:



L'immagine si chiama marchio.gif ed è depositata nella stessa directory del documento. La sintassi corretta per impostare l'immagine come sfondo è:
<BODY BACKGROUND="marchio.gif">
Il browser visualizza l'immagine sfondo.gif e la ripete su ogni punto dello schermo disponibile. In altre parole non si limita a visualizzare l'immagine una sola volta, magari al centro della pagina, ma riempie ogni spazio disponibile. Per questa ragione è assolutamente necessario creare uno sfondo che, se ripetuto, non presenti soluzione di continuità, ma un aspetto il più possibile uniforme.
E' bene scegliere un'immagine di sfondo che non infastidisca la lettura e che sia il più possibile coerente con il colore del testo. Per esempio, inserire un testo arancione su uno sfondo rosso non renderebbe leggibile il testo. Sempre meglio usare il colore nero per il testo e tinte leggere per lo sfondo.

3.3. TEXT

Se non stabilito diversamente il colore del testo del documento è nero, in quando i browser assegnano quel colore di default. Grazie all'attributo TEXT è possibile assegnare al testo un colore diverso dal nero. Questa la giusta sintassi:
<BODY BACKGROUND="marchio.gif" TEXT="white">
Anche in questo caso i colori possono esprimersi in nomi o valori esadecimali. All'interno del documento è possibile marcare parte del testo in colori differenti da quello impostato su TEXT.

3.4. LINK

Se non stabilito diversamente il colore dei link (non ancora visitati) è blue. Grazie all'attributo LINK è possibile definire colori differenti:
<BODY BACKGROUND="marchio.gif" LINK="white">

Tutti i link della pagina non saranno più blue ma bianchi (white). Tale colore può essere espresso in valori esadecimali.

3.5. ALINK

Quando cliccati i link assumono un colore diverso da quello impostato su LINK (o dal blue di default). Grazie a ALINK (la A sta per Active) è possibile modificare questo colore:
<BODY BACKGROUND="marchio.gif" ALINK="yellow">

3.6. VLINK

Quando un URL associato ad un link viene visitato, quest'ultimo assume un colore diverso da quello di LINK (o dal blue di default). Grazie a VLINK (la V sta per Visited) è possibile agire su questo colore:
<BODY BACKGROUND="marchio.gif" VLINK="green">

3.7. BGPROPERTIES

Trattando dell'attributo BACKGROUND abbiamo sottolineato come le immagini richiamate siano disposte sullo schermo disponibile. Se, comunque, la pagina è tanto lunga da attivare lo scroller laterale, lo sfondo (e l'immagine associata) scorre insieme alla pagina.
Grazie alla proprietà BGPROPERTIES è possibile rendere lo sfondo immobile rispetto allo scroller di pagina. Questa la sintassi corretta:
<BODY BACKGROUND="marchio.gif" BGPROPERTIES="fixed">
Questo espediente funziona solo con MsIe e non con Netscape che invece continua a scrollare la pagina.

4. Formattare il testo

Possiamo scrivere in **grassetto**.

```
<BODY>
Questo è un testo in<B>grassetto</B>
</BODY>
```

Quello che stiamo dicendo al browser è di visualizzare in grassetto `<B>` da questo punto in poi `</B>` fino a questo punto.

Lo stesso principio si applica per il *corsivo*...

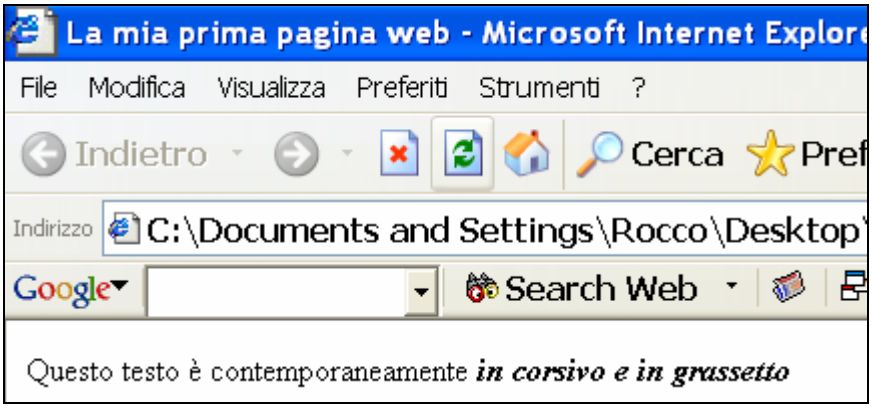
```
<BODY>
Questo è un testo in<B>grassetto</B> e in <I>corsivo</I>
</BODY>
```

...e <u>sottolineato</u>.

```
<BODY BGCOLOR="#FFFFFF">
Questo è un testo in<B>grassetto</B> in <I>corsivo</I> e <U>sottolineato</U>
</BODY>
```

I tags appena descritti possono essere combinati insieme.

```
<BODY BGCOLOR="#FFFFFF">
Questo testo è contemporaneamente <I><B>in corsivo e in grassetto</B></I>
</BODY>
```



Questo è un primo esempio di *tags annidati*. Per essi valgono le stesse proprietà delle parentesi usate in matematica, ossia l'ultimo ad essere stato aperto deve necessariamente essere il primo ad essere chiuso. Ad esempio...

```
<PRIMO> <SECONDO>  </PRIMO> </SECONDO>
```
Errato

```
<PRIMO> <SECONDO> </SECONDO> </PRIMO>
```
Corretto

I FONT

Il tag che permette di cambiare il tipo di carattere utilizzato è il tag FONT che possiede gli attributi FACE, SIZE e COLOR rispettivamente per impostare il tipo,la grandezza e il colore del carattere che si vuole utilizzare.

Vediamo qualche esempio

```
<BODY BGCOLOR="#FFFFFF">
Esempio di testo in <FONT FACE=" comic sans ms "> comic sans ms </FONT>
<br>
Esempio di testo in <FONT FACE=" arial "> arial </FONT>
</BODY>
```

I font possono avere 7 grandezze

```
<BODY BGCOLOR="#FFFFFF">
<FONT SIZE=1> Esempio di testo piccolissimo </FONT><br>
<FONT SIZE=2> Esempio di testo piccolo </FONT><br>
<FONT SIZE=3> Esempio di testo normale </FONT><br>
```

```
<FONT SIZE=4> Esempio di testo medio </FONT><br>
<FONT SIZE=5> Esempio di testo largo </FONT><br>
<FONT SIZE=6> Esempio di testo grande </FONT><br>
<FONT SIZE=7> Esempio di testo grandissimo </FONT><br>
</BODY>
```

E' possibile cambiare il colore dei font.

```
<BODY BGCOLOR="#FFFFFF">
Colore <FONT COLOR="green">verde</FONT>
Colore <FONT COLOR="red">rosso</FONT>
Colore <FONT COLOR="blue">blu</FONT>

</BODY>
```

Ovviamente, è possibile usare più di un `ATTRIBUTO` nello stesso `<TAG>`...

```
<BODY BGCOLOR="#FFFFFF">
testo <FONT COLOR="#FF0000" FACE="ARIAL" SIZE="7">arial e grandezza 7</FONT>
</BODY>
```

Testo arial e grandezza 7

Vediamo un altro esempio in cui sono annidati diversi tag

```
<BODY BGCOLOR="#FFFFFF">
testo <U><I><B><FONT COLOR="#FF0000" FACE="ARIAL" SIZE="7">sottolineato, in
corsivo, in grassetto, arial e grandezza 7</FONT></B></I></U>
</BODY>
```

Testo ***sottolineato, in corsivo, in grassetto, arial e grandezza 7***

5. Paragrafi e Giustificazione

Creare paragrafi con <P>

Il tag <P> indica al browser un nuovo paragrafo del testo,si passa quindi su una nuova riga, il posizionamento può esser a destra, a sinistra,giustificato o al centro.Per default l'allineamento è a sinistra.
Ecco i comandi giusti per allineare :
<P ALIGN="left">
Definisce un paragrafo e allinea sulla sinistra.

<P ALIGN="right">
Definisce un paragrafo e allinea sulla destra.

<P ALIGN="justify">
Definisce un paragrafo e lo giustifica sia a destra che a sinistra.

<P ALIGN="center">
Definisce un paragrafo ed allinea al centro.

Come andare a capo

**
** è un tag di interruzione di riga. Ha un funzionamento simile al paragrafo visto in precedenza, ma a differenza di <P> non inizia un nuovo paragrafo. La sua funzione è simile alla pressione del tasto "invio" della tastiera. Va usato singolarmente senza tag di chiusura.

Posizionare il testo con <DIV ALIGN> e <CENTER>
L'elemento <DIV> viene utilizzato per allineare il testo in posizione orizzontale a sinistra, destra e

centro della pagina. L'attributo ALIGN è fondamentale a questo scopo.
<DIV ALIGN="left">Testo e immagini a sinistra**</DIV>**
Sposta gli elementi contenuti tra i suoi tag sulla sinistra.

<DIV ALIGN="right">Testo e immagini a destra**</DIV>**
Sposta gli elementi sulla destra.

<DIV ALIGN="center">Testo e immagini a centro**</DIV>**
Sposta gli elementi in posizione centrale.

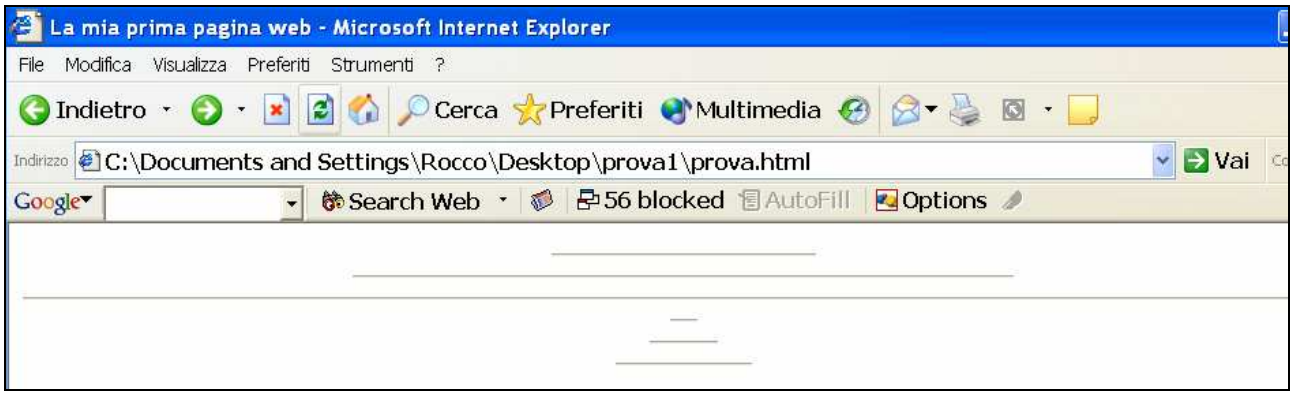
Il tag **<CENTER>** ha un funzionamento del tutto simile a **<DIV ALIGN="center">**
L'uso di **<CENTER>** è molto semplice:
<CENTER>Testo da centrare**</CENTER>**
ecco il testo centrato :
Testo da centrare

Linee orizzontali con <HR>

Le linee orizzontali sono un ottimo strumento per dividere parti del documento e rendere il testo più leggibile. Vediamone qualche esempio:

```
<BODY>
<HR WIDTH=20%>
<HR WIDTH=50%>
<HR WIDTH=100%>
<HR WIDTH=20>
<HR WIDTH=50>
<HR WIDTH=100>
</BODY>
```

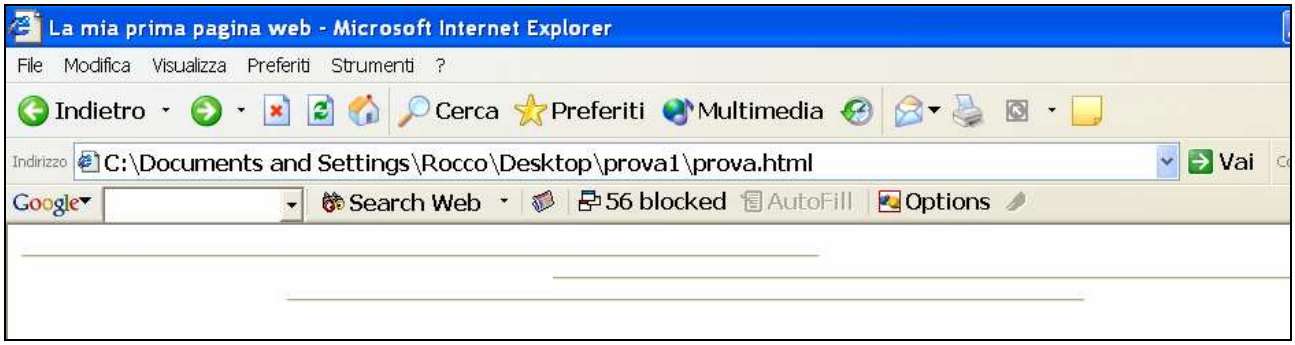
La resa del browser è la seguente



Il seguente esempio si spiega da è abbastanza intuitivo:
Questo esempio si spiega da solo.

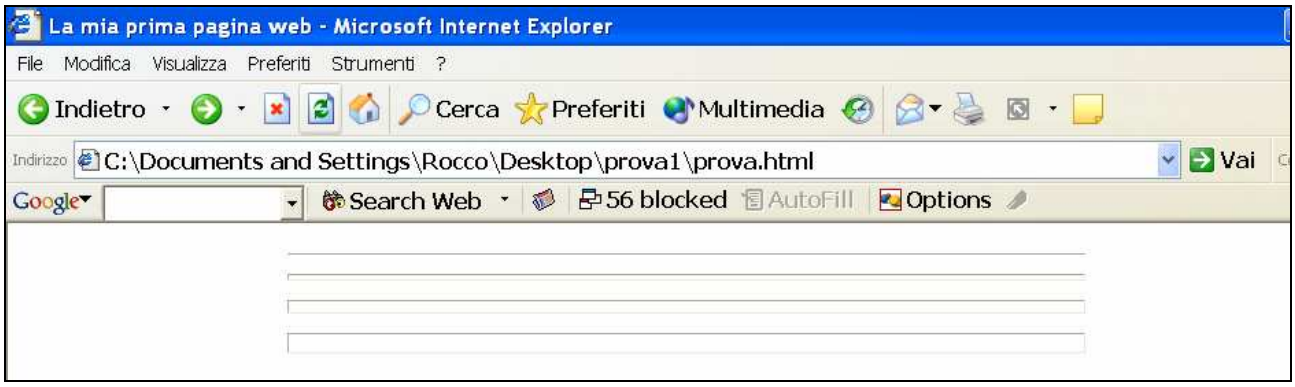
```
<BODY>
<HR WIDTH=60% ALIGN=LEFT>
<HR WIDTH=60% ALIGN=RIGHT>
<HR WIDTH=60% ALIGN=CENTER>
```

```
</BODY>
```



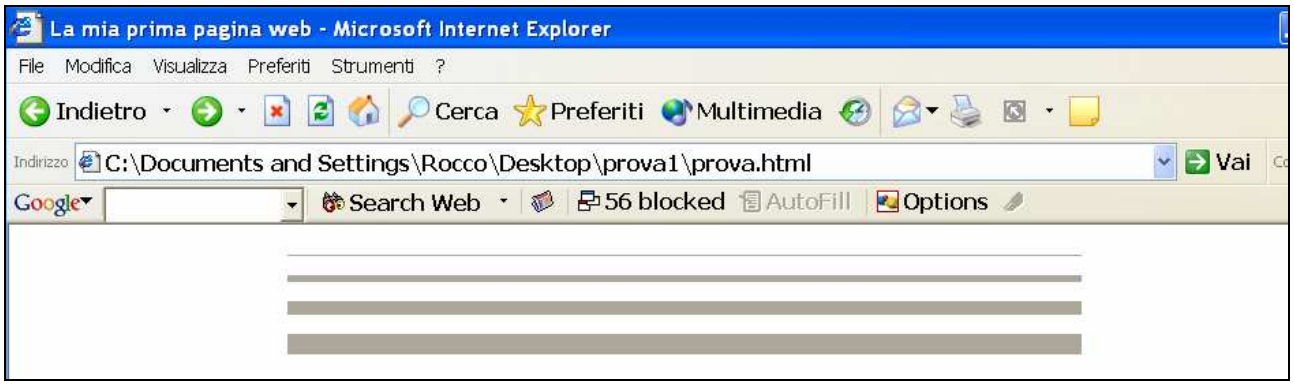
E' possibile anche definire lo spessore...

```
 <BODY>
<HR WIDTH=60% SIZE=1>
<HR WIDTH=60% SIZE=5>
<HR WIDTH=60% SIZE=10>
<HR WIDTH=60% SIZE=15>
</BODY>
```



E puoi decidere di renderla più spessa.

```
 <BODY BGCOLOR="#FFFFFF">
<HR WIDTH=60% SIZE=1 NOSHADE>
<HR WIDTH=60% SIZE=5 NOSHADE>
<HR WIDTH=60% SIZE=10 NOSHADE>
<HR WIDTH=60% SIZE=15 NOSHADE>
</BODY>
```



Infine è possibile anche cambiare il colore attraverso l’attributo color

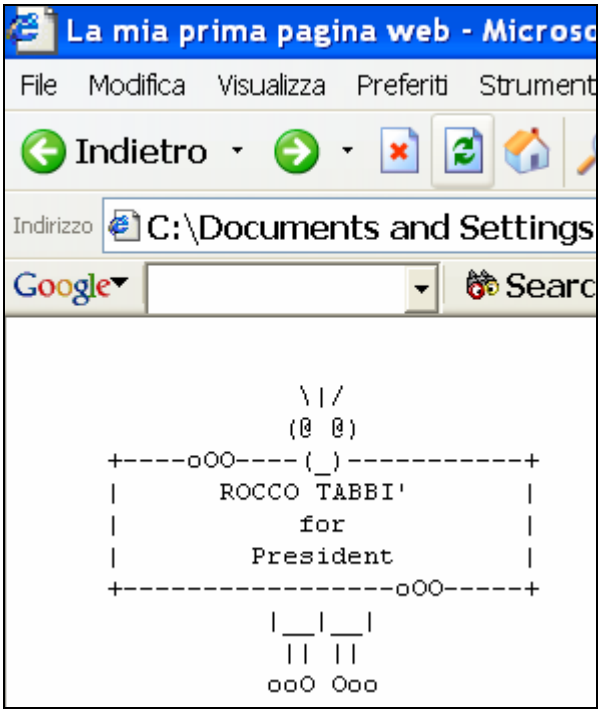
Il tag <pre> preformatted

Serve a visualizzare esattamente quello che si è scritto. Ad esempio:

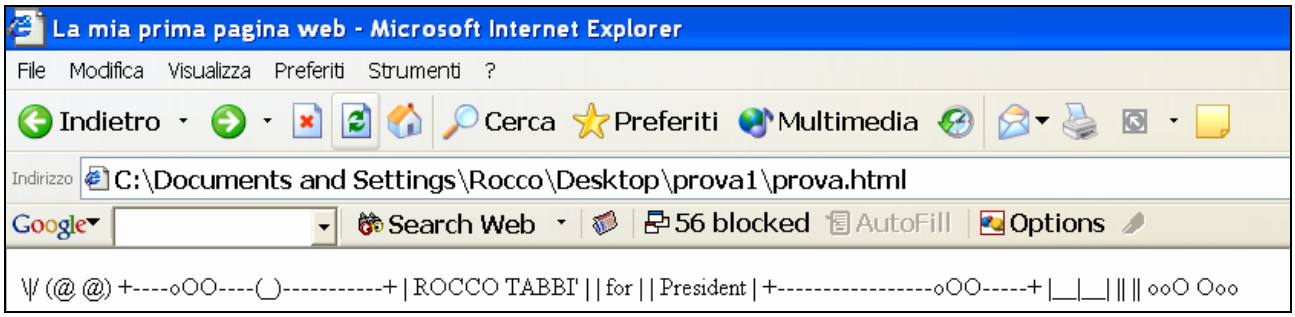
```
<BODY>
<PRE>

                \|/
               (@ @)
     +----oOO----(_)-----------+
     |      ROCCO TABBI'       |
     |           for           |
     |        President        |
     +-----------------oOO-----+
               |__|__|
                || ||
               ooO Ooo
</PRE>
</BODY>
```

Verrà visualizzato nel seguente modo:



mentre mancando il tag <pre> la visualizzazione sarebbe la seguente:



6. Inserire immagini nel documento

Ora vediamo come inserire immagini in una pagina web. Supponiamo di voler inserire la seguente immagine di nome ciofs.gif

Il tag specifico per inserire un'immagine è questo: `<IMG>`.

Dobbiamo specificare il percorso (dove si trova "fisicamente" l'immagine)

```
<BODY>
<IMG SRC="ciofs.gif">
</BODY>
```

Nota che il codice specifica non solo *il nome* dell'immagine, ma non *dove* si trova. Il codice dell'esempio precedente, `"ciofs.gif"`, significa che il browser cercerà l'immagine `ciofs.gif` nella stessa cartella in cui si trova il documento html. Se per caso l'immagine si trova in una posizione differente si deve inserire il relativo pathname.

Nell'esempio precedente l'immagine è stata inserita nella sua dimensione reale, se si vuole variarla si devono utilizzare gli attributi height (altezza) e width(larghezza). Il loro valore può essere espresso in pixel o in percentuale. E' importante inserire le dimensioni dell'immagine poiché in questo modo il browser predispone lo spazio necessario ad ospitarla e rende lo scaricamento della pagina più veloce.

Allarghiamo l'immagine…..

```
<BODY>
<IMG SRC="ciofs.gif" WIDTH=300 HEIGHT=60>
</BODY>
```

adesso restringiamola

```
<BODY>
<IMG SRC="ciofs.gif" WIDTH=60 HEIGHT=100>
</BODY>
```

7. Come creare elenchi puntati e numerati

Gli elenchi sono comodi per organizzare le informazioni all'interno di pagine Web.

Elenchi ordinati (numerati)

Gli elenchi ordinati sono costiuiti da un singolo tag di apertura e chiusura **<OL></OL>** e tanti tag di apertura per quante sono le voci di menu **<LI>.**

Questa è la corretta sintassi per creare elenchi ordinati:

<OL>
<LI> Prima voce di menu

<LI> Seconda voce di menu
<LI> Terza voce di menu
<LI> Quarta voce di menu
</OL>

1. Prima voce di menu
2. Seconda voce di menu
3. Terza voce di menu
4. Quarta voce di menu

Dall'esempio si nota come è possibile omettere il tag <BR> per il ritorno a capo, visto che è automaticamente inserito da <LI>.

Indicizzazione alfabetica maiuscola:
<OL TYPE=A>
<LI> Prima voce di menu
<LI> Seconda voce di menu
<LI> Terza voce di menu
</OL>

A. Prima voce di menu
B. Seconda voce di menu
C. Terza voce di menu

Indicizzazione alfatabetica minuscola:
<OL TYPE=a>
<LI> Prima voce di menu
<LI> Seconda voce di menu
<LI> Terza voce di menu
</OL>

a. Prima voce di menu
b. Seconda voce di menu
c. Terza voce di menu

Indicizzazione con numeri romani maiuscoli:
<OL TYPE=I>
<LI> Prima voce di menu
<LI> Seconda voce di menu
<LI> Terza voce di menu
</OL>

I. Prima voce di menu
II. Seconda voce di menu
III. Terza voce di menu

Indicizzazione con numeri romani minuscoli:
<OL TYPE=i>
<LI> Prima voce di menu
<LI> Seconda voce di menu
<LI> Terza voce di menu
</OL>

i. Prima voce di menu
ii. Seconda voce di menu
iii. Terza voce di menu

Elenchi NON ordinati (puntati)

Gli elenchi non ordinati funzionano in modo simile a quelli ordinati. La differenza di fondo è che le risorse indicizzate non sono legate da stretti rapporti di successione gerarchica, per cui non sono previsti elenchi progressivi quali numeri o lettere. Gli elenchi non ordinati (o puntati) si compongono di un tag unico di apertura e chiusura **<UL></UL>** e tanti tag di elenco per quante sono le voci da indicizzare
La corretta sintassi per definire un elenco puntato è la seguente:

<UL>
<LI> Prima voce di menu
<LI> Seconda voce di menu
<LI> Terza voce di menu
</UL>

- Prima voce di menu
- Seconda voce di menu
- Terza voce di menu

Se non definito diversamente l'elenco è costituito da una serie di pallini. E' possibile indicare diversi tipi di elenco:
I pallini pieni visti in precedenza sono ottenibili con **disc**:
<UL TYPE=disc>
<LI> Prima voce di menu
<LI> Seconda voce di menu
<LI> Terza voce di menu
</UL>

- Prima voce di menu
- Seconda voce di menu
- Terza voce di menu

L'attributo circle imposta pallini vuoti all'interno:
<UL TYPE=circle>
<LI> Prima voce di menu
<LI> Seconda voce di menu
<LI> Terza voce di menu
</UL>

- Prima voce di menu
- Seconda voce di menu
- Terza voce di menu

L'attributo square imposta elenchi definiti da quadratini pieni:
<UL TYPE=square>
<LI> Prima voce di menu
<LI> Seconda voce di menu
<LI> Terza voce di menu
</UL>

- Prima voce di menu
- Seconda voce di menu
- Terza voce di menu

8. I link

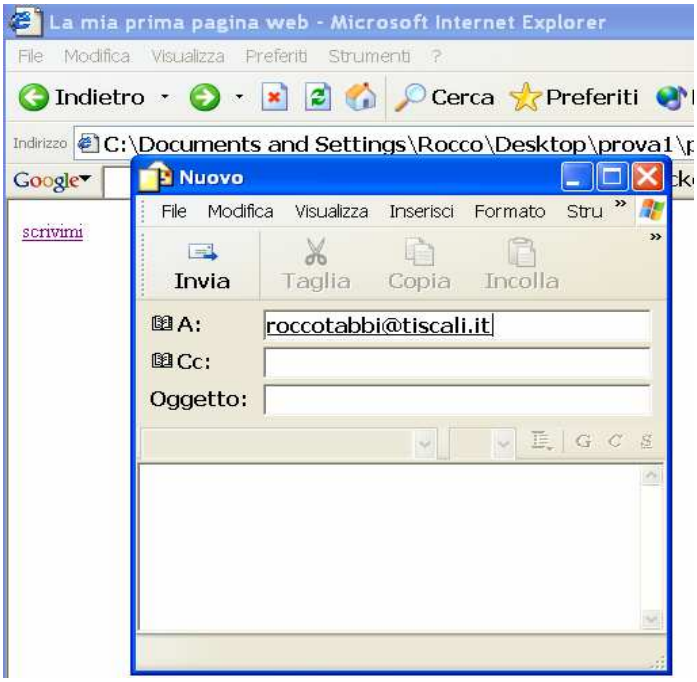
Il tag che permette i collegamenti ipertestuali è il tag **<A>** (la A sta per Ancora) che ha bisogno di un tag di apertura e chiusura e al suo interno è possibile inserire testo, immagini o altri elementi multimediali. Perchè funzioni, l'elmento **<A>** deve essere associato ad altri attributi. Il più importante di questi è **HREF** (abbreviazione di Hypertext Reference),cioè a chi si riferisce contenente l'URL o la pagina da raggiungere.
Tale collegamento può essere ad un altro sito (esempio il sito del ciofs di gela)
Ecco la sintassi corretta:
<A HREF=" HREF="http://www.ciofsfpgela.it/">visita il sito del ciofs</A>
Nel codice sopra citato cliccando sul testo "visita il sito del ciofs" (compreso tra i tag A in apertura e chiusura) si raggiunge l'URL http://www.ciofsfpgela.it indicato dall'attributo HREF.
I collegamenti possono essere a pagine dello stesso sito.
In questo caso la sintassi è la seguente:
<A HREF="chisiamo.html">Chi siamo</A>

In questo caso la pagina chisiamo.html fa parte dello stesso sito e si trova nella stessa cartella della pagina in cui è inserito il link.

Un link oltre che verso un sito,può essere fatto verso una casella di posta elettronica, vediamo di seguito l'esempio :

<A HREF="mailto:qualcuno@suo.indirizzo.email">
Apre il client mail(di solito outlook express) per spedire una lettera all'indirizzo specificato.

Se per esempio volessi farmi spedire qualcosa, io dovrei inserire :
<body>
<A HREF=”mailto:roccotabbi@tiscali.it"> Scrivimi****
</body>
E il risultato sarebbe il seguente



E' possibile sostituire al testo un'immagine ottenendo un effetto identico. L’immagine-link verrà bordata con gli stessi colori riservati ai link. Per poter togliere tali bordi è opportuno inserire l’attributo border.

```
<BODY>
<A HREF="chi siamo.html"><IMG SRC="immagine.gif" WIDTH=82 HEIGHT=68
BORDER=0></A>
</BODY>
```

Questa funzione è molto importante; infatti immagina di dover pubblicare un book fotografico. Il caricamento di immagini comporta una notevole perdita di tempo durante il caricamento da parte del browser. Si rischia così che l’utente connesso abbandoni il sito prima che queste siano state caricate completamente. Si può ovviare a tale problema, pubblicando nella pagina delle miniature(riducendo notevolmente così il tempo di caricamento) che però risultano link alle relative immagini con le dimensioni reali.
Vediamo il procedimento da adottare:

Comincia con il tag `<IMG>`.

```
<BODY>
<IMG SRC="meravigliosa1.gif" WIDTH=87 HEIGHT=60>
<IMG SRC="meravigliosa2.gif" WIDTH=87 HEIGHT=60>
</BODY>
```

Aggiungi il tag `<A>` con il relativo URL ed il gioco è fatto.

```
<BODY>
<a href=”meravigliosa1.gif”><IMG SRC="meravigliosa1.gif" WIDTH=87 HEIGHT=60></a>
<a href=”meravigliosa2.gif”><IMG SRC="meravigliosa2.gif" WIDTH=87 HEIGHT=60></a>
</BODY>
```

Se vuoi puoi togliere la cornice, anche se puoi lasciarla per far capire ai tuoi visitatori che si tratta di un link.

```
<BODY>
<a href=”meravigliosa1.gif”><IMG SRC="meravigliosa1.gif" WIDTH=87 HEIGHT=60
border=0></a>
<a href=”meravigliosa2.gif”><IMG SRC="meravigliosa2.gif" WIDTH=87 HEIGHT=60
border=0></a>
</BODY>
```

Un altro tipo di link consente di collegare *punto specifico* di una pagina. Ecco come fare. Il primo passo è stabilire quale punto della pagina linkare e inserirla tra i tag `<A>`.

<A>Punto**</A>** di arrivo

Ora, diamo a questo punto un NOME (NAME).

```
<A NAME="pippo">Punto</A> di arrivo
```

Quello che hai fatto è marcare il punto. Ora deve essere assegnato il riferimento.

Cominciamo a costruire il link.

```
Clicca <A HREF="lesson04.html#pippo">qui</A> per essere trasportato al punto
dove si trova la frase punto di arrivo
```

9. Le tabelle

Le tabelle rivestono un ruolo fondamentale nella creazione delle pagine web, poiché grazie ad esse riusciamo a posizionare gli oggetti all’interno delle stesse. Una tabella è composta da righe e colonne, quindi creando una tabella occorre specificare una nuova riga con **<TR>** e poi all'interno di essa le colonne con **<TD>**, questi sono dei tag che vanno chiusi con **</TR>** e **</TD>**. Ma prima di passare alla creazione di righe e colonne, bisogna specificare che si sta per creare una tabella con il tag **<TABLE>** che va chiuso con **</TABLE>.** Il tutto si ottiene giocando in pratica su questi TRE tags:

<TABLE>
: Il tag principale. Utilizzato per indicare al browser "questa é una tabella", assieme ad altri attributi come le dimensioni, i bordi ecc...

<TR>
: TableRow definisce una riga orizzontale di <TD> (TableData) *celle*.

<TD>

Questo è un esempio di una tabella con 3 righe e 3 colonne

<table>

<tr>	<td>oggetti </td>	<td>oggetti </td>	<td>oggetti </td>	</tr>
<tr>	<td>oggetti </td>	<td>oggetti </td>	<td>oggetti </td>	</tr>
<tr>	<td>oggetti </td>	<td>oggetti </td>	<td>oggetti </td>	</tr>

</table>

Iniziamo a costruire una tabella passo passo.

Scrivi i tags di tabella. Questi significano semplicemente "qui comincia una tabella" e "qui finisce la tabella".

```
<HTML>
<HEAD>
<TITLE>Esempio di tabella</TITLE>
</HEAD>
<BODY>

<TABLE>
</TABLE>
```

```
</BODY>
</HTML>
```

Ogni tabella ha bisogno di perlomeno UNA riga.

```
<HTML>
<HEAD>
<TITLE> Esempio di tabella </TITLE>
</HEAD>
<BODY>

<TABLE>
<TR>
</TR>
</TABLE>

</BODY>
</HTML>
```

E ovviamente ogni riga necessita di perlomeno una *cella* di dati.

```
<HTML>
<HEAD>
<TITLE> Esempio di tabella </TITLE>
</HEAD>
<BODY>

<TABLE>
<TR>
<TD></TD>
</TR>
</TABLE>

</BODY>
</HTML>
```

D'ora in poi s’intendono sottintesi i tag principali della pagina(html,head, title, body)

```
<TABLE>
<TR>
<TD></TD>
</TR>
</TABLE>
```

Adesso ci servono gli oggetti da inserire nella cella. Ad esempio il testo CIOFS.

```
<TABLE>
<TR>
<TD>CIOFS</TR>
</TABLE>
```

Aprendo la pagina con il browser si ottiene

```
CIOFS
```

Dov’è la tabella? C’è ma non si vede; poiché non abbiamo indicato al browser di mettere il bordo.

```
<TABLE BORDER=1>
<TR>
<TD>CIOFS</TD>
</TR>
</TABLE>
```

CIOFS

Possiamo ingrossare il bordo nel seguente modo:

```
<TABLE BORDER=5>
<TR>
<TD>CIOFS</TD>
</TR>
</TABLE>
```

CIOFS

Lo possiamo anche togliere

```
<TABLE BORDER=0>
<TR>
<TD>CIOFS</TD>
</TR>
</TABLE>
```

tornando così nella situazione iniziale che è poi il valore di default.

Se non viene specificata alcuna dimensione la tabella verrà visualizzata grande quel tanto che basta.

```
<TABLE BORDER=3>
<TR>
<TD>CIOFS FP GELA</TD>
</TR>
</TABLE>
```

CIOFS FP GELA

In ogni caso possiamo specificare la larghezza della tabella in base alla dimensione totale della finestra del browser.

```
<TABLE BORDER=3 WIDTH=100%>
<TR>
<TD> CIOFS FP GELA </TD>
</TR>
</TABLE>
```

CIOFS FP GELA

Un altro esempio è il seguente:

```
<TABLE BORDER=3 WIDTH=75%>
<TR>
```

```
<TD> CIOFS FP GELA </TD>
</TR>
</TABLE>
```

CIOFS FP GELA

Adesso omettiamo il simbolo %

```
<TABLE BORDER=3 WIDTH=75>
<TR>
<TD>CIOFS</TD>
</TR>
</TABLE>
```

CIOFS

Ora 100.

```
<TABLE BORDER=3 WIDTH=100>
<TR>
<TD>CIOFS</TD>
</TR>
</TABLE>
```

CIOFS

Si è intuito che ci sono due modi differenti di specificare la dimensione di un tabella. Il primo in percentuale il secondo in pixel.
Lo stesso ragionamento vale anche per l’altezza, l’attributo utilizzato è HEIGHT.

E' possibile inoltre allineare i dati in senso orizzontale con i seguenti attributi:

```
<TABLE BORDER=3 WIDTH=100 HEIGHT=75>
<TR>
<TD ALIGN=CENTER>CIOFS</TD>
</TR>
</TABLE>
```

CIOFS

```
<TABLE BORDER=3 WIDTH=100 HEIGHT=75>
<TR>
<TD ALIGN=RIGHT>CIOFS</TD>
</TR>
</TABLE>
```

CIOFS

```
<TABLE BORDER=3 WIDTH=100 HEIGHT=75>
<TR>
<TD ALIGN=LEFT>CIOFS</TD>
</TR>
</TABLE>
```

CIOFS

Come vedi il valore di default è `ALIGN=LEFT`.
L’allineamento è possibile darlo anche in verticale nel seguente modo:

```
<TABLE BORDER=3 WIDTH=100 HEIGHT=225>
<TR>
<TD ALIGN=LEFT VALIGN=TOP>CIOFS</TD>
</TR>
</TABLE>
```

CIOFS

```
<TABLE BORDER=3 WIDTH=100 HEIGHT=225>
<TR>
<TD ALIGN=LEFT VALIGN=BOTTOM>CIOFS</TD>
</TR>
</TABLE>
```

CIOFS

```
<TABLE BORDER=3 WIDTH=100 HEIGHT=225>
<TR>
<TD ALIGN=LEFT VALIGN=MIDDLE>CIOFS</TD>
</TR>
</TABLE>
```

CIOFS

Per default i dati sono inseriti nel mezzo.

Nelle tabelle oltre al testo possiamo inserire anche delle immagini:

```
<TABLE BORDER=3 WIDTH=100 HEIGHT=225>
<TR>
<TD ALIGN=LEFT VALIGN=MIDDLE><IMG SRC="prova.gif" WIDTH=32 HEIGHT=32></TD>
</TR>
</TABLE>
```

E’ consigliabile inserire gli attributi HEIGHT e WIDTH in tutte le immagini.

Finora abbiamo preso in considerazione tabelle con una riga ed una cella adesso generalizziamo il discorso.

```
<TABLE BORDER=3 WIDTH=100 HEIGHT=75>
<TR>
```

```
<TD ALIGN=LEFT VALIGN=MIDDLE>QUI</TD>
</TR>
</TABLE>
```

QUI

Ovviamente possiamo togliere gli attributi di allineamento ottenendo il medesimo risultato.

```
<TABLE BORDER=3 WIDTH=100 HEIGHT=75>
<TR>
<TD>QUI</TD>
</TR>
</TABLE>
```

QUI

Adesso allarghiamo la tabella.

```
<TABLE BORDER=3 WIDTH=300 HEIGHT=75>
<TR>
<TD>QUI</TD>
</TR>
</TABLE>
```

QUI

Supponiamo che QUI abbia la visita di suo fratello QUO che pretende la sua cella personale.

```
<TABLE BORDER=3 WIDTH=300 HEIGHT=75>
<TR>
<TD>QUI</TD>
<TD>QUO</TD>
</TR>
</TABLE>
```

QUI	QUO

Comunque è più opportuno dimensionare orizzontalmente le celle in questo modo:

```
<TABLE BORDER=3 WIDTH=300 HEIGHT=75>
<TR>
<TD WIDTH=150>QUI</TD>
<TD WIDTH=150>QUO</TD>
</TR>
</TABLE>
```

QUI	QUO

Gli attributi di `LARGHEZZA` possono anche essere espressi come valori percentuale.

```
<TABLE BORDER=3 WIDTH=300 HEIGHT=75>
<TR>
```

```
<TD WIDTH=50%>QUI</TD>
<TD WIDTH=50%>QUO</TD>
</TR>
</TABLE>
```

QUI	QUO

Essendo QUI arrivato per primo forse è meglio assegnarli una cella più grande

```
<TABLE BORDER=3 WIDTH=300 HEIGHT=75>
<TR>
<TD WIDTH=80%>QUI</TD>
<TD WIDTH=20%>QUO</TD>
</TR>
</TABLE>
```

QUI	QUO

Come se non bastasse è arrivato pure QUA. Dobbiamo sistemare anche lui:

```
<TABLE BORDER=3 WIDTH=300 HEIGHT=75>
<TR>
<TD WIDTH=60%>QUI</TD>
<TD WIDTH=20%>QUO</TD>
<TD WIDTH=20%>QUA</TD>
</TR>
</TABLE>
```

QUI	QUO	QUA

Paperina, paperino e paperone vogliono stare insieme ai loro nipotini e decidono di piazzarsi nel piano più basso(una nuova riga)

```
<TABLE BORDER=3 WIDTH=300 HEIGHT=75>

<TR>
<TD WIDTH=60%>QUI</TD>
<TD WIDTH=20%>QUO</TD>
<TD WIDTH=20%>QUA</TD>
</TR>

<TR>
<TD>Paperina</TD>
<TD>Paperino</TD>
<TD>Paperone</TD>
</TR>

</TABLE>
```

QUI	QUO	QUA
Paperina	Paperino	Paperone

L’attributo WIDTH l’abbiamo usato solo per le prime tre celle, poiché esso viene ereditato dalle celle più in basso

Conoscendo il carattere di paperone è facile che si stufa e che se ne vada. La struttura della tabella resta inalterata:

```
<TABLE BORDER=3 WIDTH=300 HEIGHT=75>

<TR>
<TD WIDTH=60%>QUI</TD>
<TD WIDTH=20%>QUO</TD>
<TD WIDTH=20%>QUA</TD>
</TR>

<TR>
<TD>Paperino</TD>
<TD>Paperina</TD>
</TR>

</TABLE>
```

QUI	QUO	QUA
Paperina	Paperino	

Risistemiamo tutto come prima e leviamo tutti gli attributi tranne il `BORDER`.

```
<TABLE BORDER=3>

<TR>
<TD>QUI</TD>
<TD>QUO</TD>
<TD>QUA</TD>
</TR>

<TR>
<TD>Paperina</TD>
<TD>Paperino</TD>
<TD>Paperone</TD>
</TR>

</TABLE>
```

QUI	QUO	QUA
Paperina	Paperino	Paperone

Introduciamo adesso altri due attributi: `CELLPADDING` e `CELLSPACING`. Sono entrambi definiti all'interno del Tag `<TABLE>` iniziale. `CELLPADDING` é lo spazio che c'é fra il bordo della cella e il contenuto della cella stessa.

```
<TABLE BORDER=3 CELLPADDING=12>

<TR>
<TD>QUI</TD>
<TD>QUO</TD>
```

```
<TD>QUA</TD>
</TR>

<TR>
<TD>Paperina</TD>
<TD>Paperino</TD>
<TD>Paperone</TD>
</TR>

</TABLE>
```

QUI	QUO	QUA
Paperina	Paperina	Paperone

Il valore di default per questo attributo é 1. La ragione per cui il default é 1 e non 0 é che altrimenti il testo starebbe schiacciato contro i bordi della cella.

Se sostituiamo `CELLSPACING` al posto di `CELLPADDING` otteniamo un effetto un po' diverso.

```
<TABLE BORDER=3 CELLSPACING=12>

<TR>
<TD>QUI</TD>
<TD>QUO</TD>
<TD>QUA</TD>
</TR>

<TR>
<TD>Paperino</TD>
<TD>Paperina</TD>
<TD>Paperone</TD>
</TR>

</TABLE>
```

QUI	QUO	QUA
Paperina	Paperino	Paperone

Il valore di default per `CELLSPACING` é 2.

Possiamo combinare i due attributi.

```
<TABLE BORDER=3 CELLSPACING=12 CELLPADDING=12>

<TR>
<TD>QUI</TD>
<TD>QUO</TD>
<TD>QUA</TD>
</TR>
```

```
<TR>
<TD>Paperina</TD>
<TD>Paperino</TD>
<TD>Paperone</TD>
</TR>

</TABLE>
```

QUI	QUO	QUA
Paperina	Paperino	Paperone

Una caratteristica dei browser più recenti è la possibilità di specificare un colore di sfondo per ciascuna cella, riga o per l'intera tabella. Per fare questo si usa l'attributo `BGCOLOR` proprio come nel tag `<BODY>`.

```
<TABLE BORDER=3 BGCOLOR="#FFCC66">

<TR>
<TD>Qui</TD>
<TD>Quo</TD>
<TD>Qua</TD>
</TR>

<TR>
<TD>Paperina</TD>
<TD>PAperino</TD>
<TD>Paperone</TD>
</TR>

</TABLE>
```

Qui	Quo	Qua
Paperina	Paperino	Paperone

```
<TABLE BORDER=3>

<TR BGCOLOR="#FF9999">
<TD>Qui</TD>
<TD>Quo</TD>
<TD>Qua</TD>
</TR>

<TR BGCOLOR="#99CCCC">
<TD>Paperina</TD>
<TD>Paperino</TD>
<TD>Paperone</TD>
</TR>

</TABLE>
```

Qui	Quo	Qua
Paperina	Paperino	Paperone

```
<TABLE BORDER=3>

<TR BGCOLOR="#FFCCFF">
<TD>Qui</TD>
<TD>Quo</TD>
<TD>Qua</TD>
</TR>

<TR>
<TD BGCOLOR="#FF0000">Paperina</TD>
<TD>Paperino</TD>
<TD BGCOLOR="#3366FF">Paperone</TD>
</TR>

</TABLE>
```

Qui	Quo	Qua
Paperina	Paperino	Paperone

Un'ultima cosa riguardo ai colori di sfondo nelle tabelle... un colore di sfondo di cella (`<TD>`) copre quello di riga (`<TR>`) e uno di riga (`<TR>`) copre uno di tabella (`<TABLE>`).

Un browser cerca di interpretare nel migliore modo possibile le istruzioni che riceve. Se c'è qualcosa che non è stato specificato in un modo o nell'altro, la maggior parte dei browser cercherà di visualizzare la pagina nella maniera che ritiene migliore. E' quindi estremamente importante che un autore di pagine HTML specifichi esplicitamente più cose possibile, specialmente gli attributi cruciali perchè le pagine vengano visualizzate correttamente. E' anche molto importante visualizzare il proprio lavoro utilizzando i browser più comuni. Dal momento che moltissimi usano Netscape, questo è un buon punto di partenza.

Introduciamo adesso `COLSPAN` (eSPANsione su COLonne) and `ROWSPAN` (espansione su righe). Immaginiamo che QUI se la prenda a morte con QUO e lo scaraventi fuori dalla tabella. Ecco quello che succede in questo caso.

```
<TABLE BORDER=3>

<TR>
<TD>QUI</TD>
<TD>QUA</TD>
</TR>

<TR>
<TD>Paperina</TD>
<TD>Paperino</TD>
<TD>Paperone</TD>
</TR>

</TABLE>
```

QUI	QUA	
Paperina	Paperino	Paperone

Viene lasciato un buco e QUA scivola di lato a riempire il vuoto.

Se però vogliamo che QUI si impossessi dello spazio che prima era di QUO, allora dobbiamo utilizzare l'attributo `COLSPAN`.

```
<TABLE BORDER=3>

<TR>
<TD COLSPAN=2>QUI</TD>
<TD>QUA</TD>
</TR>

<TR>
<TD>Paperina</TD>
<TD>Paperino</TD>
<TD>Paperone</TD>
</TR>

</TABLE>
```

QUI		QUO
Paperina	Paperino	Paperone

E se qui vuol cacciare anche QUO e impossessarsi del suo spazio il codice diventa:

```
<TABLE BORDER=3>

<TR>
<TD COLSPAN=3 ALIGN=CENTER>QUO</TD>
</TR>

<TR>
<TD>Paperina</TD>
<TD>Paperino</TD>
<TD>Paperone</TD>
</TR>

</TABLE>
```

QUI		
Paperina	Paperino	Paperone

Ora ripuliamo la tabella e rimettiamoci QUO e QUA per passare al `<ROWSPAN>`.

```
<TABLE BORDER=3>

<TR>
<TD>QUI</TD>
<TD>QUO</TD>
```

```
<TD>QUA</TD>
</TR>

<TR>
<TD>Paperina</TD>
<TD>Paperino</TD>
<TD>Paperone</TD>
</TR>

</TABLE>
```

QUI	QUO	QUA
Paperina	Paperino	Paperone

Come è facile immaginare, `<ROWSPAN>` funziona esattamente come `<COLSPAN>` tranne per il fatto che si espandono righe invece che colonne.
Se leviamo Paperina e assegnamo a QUI la sua parte di spazio, otteniamo questo.

```
<TABLE BORDER=3>

<TR>
<TD ROWSPAN=2>QUI</TD>
<TD>QUO</TD>
<TD>QUA</TD>
</TR>

<TR>
<TD>PAperino</TD>
<TD>Paperone</TD>
</TR>

</TABLE>
```

QUI	QUO	QUA
	PAperino	Paperone

Naturalmente è anche possibile utilizzare una combinazione dei due attributi.

```
<TABLE BORDER=3>

<TR>
<TD ROWSPAN=2>QUI</TD>
<TD COLSPAN=2>QUA</TD>
</TR>

<TR>
<TD>Paperino</TD>
<TD>Paperone</TD>
</TR>

</TABLE>
```

QUI	QUA

	Paperino	Paperone

10. I FORM

Il Tag che definisce l'esistenza di un form è il seguente:

```
 <HTML>
<HEAD>
<TITLE>Joe's the handsomest guy I know</TITLE>
</HEAD>
<BODY>

 <FORM>
</FORM>

 </BODY>
</HTML>
```

Dopo aver inserito il tag FORM si deve specificare al Browser *dove* inviare i dati che riceve e *come* inviarli. Esistono due opzioni:

1. mandare i dati ad uno script cgi per processarli.
2. far si che i dati vengano inviati come E-mail

Noi ci concentreremo sul secondo metodo(il cosiddetto *mailto)* il quale richiede che nel `<FORM>` vengano specificati i seguenti attributi.

```
<HTML>
<HEAD>
<TITLE>I FORM</TITLE>
</HEAD>
<BODY>

 <FORM METHOD=POST ACTION="mailto:xxx@xxx.xxx">
</FORM>

 </BODY>
</HTML>
```

Le parole `FORM`, `METHOD`, `POST` & `ACTION` non importa che siano maiuscole *ma* deve esserci uno spazio fra ciascun attributo.. fra `FORM` & `METHOD`, fra `POST` & `ACTION`.

Sfortunatamente i dati verranno spediti in un formato utile al computer ma poco allo sfortunato utente...

```
NOME=Rocco+Tabbì&INDIRIZZO=Via+di+qua+37&CITTA’=Butera&PROVINCIA=C
altanissetta
```

Quello che serve è un programma che li trasformi in un formato più "umano"...

```
NOME=Rocco Tabbì
INDIRIZZO=Via di qua 37
CITTA'=Butera
PROVINCIA=Caltanissetta
```

Mailto Formatter è un eccellente piccolo programma di utilità freeware che svolge tale compito in modo eccellente.

L'esempio visto poc'anzi mostra che un modulo non è altro che una serie di *nomi* (`NOME`, `INDIRIZZO`, etc) accoppiati a *valori* (`Rocco Tabbì`, Via di qua 37, etc).La sola vera variabile è *come* possiamo fare ad ottenere detti valori.

D'ora in poi, per maggiore chiarezza, verranno scritti solo quello che sta *fra* i due tag `<FORM>`. Si tralasceranno i tag head, body, title e form. Ovviamente in un documento è necessario che questi tags ci siano.

La più comune **tipologia** [`TYPE`] di **immissione dati** in un modulo [`<INPUT>`] è il semplice **testo** [`TEXT`].

`<INPUT TYPE=TEXT>`

Ogni input ha bisogno di un **nome** [`NAME`].

```
<INPUT TYPE=TEXT NAME="INDIRIZZO">
```

Quando un utente scrive nel modulo il suo indirizzo (ad esempio Via di qua 37), questo diventerà automaticamente il *valore* dell'input e verrà accoppiato ad INDIRIZZO cosicché il risultato finale, dopo aver fatto passare i dati attraverso il Mailto Formatter, sarà `INDIRIZZO=Via di qua 37`.

Se vogliamo possiamo assegnare un `VALUE` a piacere di default.

```
<INPUT TYPE=TEXT NAME="INDIRIZZO" VALUE="Via di là 73">
```

Via di là 73

In questo modo il valore `Via di là` sarà accoppiato automaticamente con il nome INDIRIZZO, *a meno che* l'utente non lo modifichi.
Note- controlla bene di mettere le virgolette dove sono state specificate.

Ora possiamo specificare la dimensione della casella di immissione dati.

```
<INPUT TYPE=TEXT NAME="ADDRESS" VALUE=" Via di là 73" SIZE=10>
```

Via di là 73

```
<INPUT TYPE=TEXT NAME="ADDRESS" VALUE=" Via di là 73" SIZE=20>
```

Via di là 73

```
<INPUT TYPE=TEXT NAME="ADDRESS" VALUE=" Via di là 73" SIZE=30>
```

Via di là 73

Come si può vedere il valore di default è 20. Probabilmente già sai che il *valore di default* è il valore che viene assegnato ad una variabile se non è specificato alcun valore particolare.

Andiamo avanti e leviamo `VALUE=" Via di là 73"`.

```
<INPUT TYPE=TEXT NAME="INDIRIZZO" SIZE=30>
```

Se vogliamo, possiamo specificare un numero massimo di caratteri da immettere utilizzando il seguente attributo.

```
<INPUT TYPE=TEXT NAME="ADDRESS" SIZE=30 MAXLENGTH=10>
```

Molto simile al `TYPE=TEXT` è il `TYPE=PASSWORD`. Si tratta esattamente della stessa cosa tranne per il fatto che vengono visualizzati degli asterischi al posto dell'input reale. Il browser poi ti *spedirà* i dati reali, soltanto che non li *visualizzerà*.

```
<INPUT TYPE=PASSWORD>
```

Ad ogni `<INPUT>` deve essere associato un `NAME`.

```
<INPUT TYPE=PASSWORD NAME="USER PASSWORD">
```

Gli attributi di `SIZE`, `VALUE`, e `MAXLENGTH` funzionano anche qui. A questo proposito, un `<TAG>` dice al browser di fare qualcosa. Un `ATTRIBUTO` sta dentro al `<TAG>` e specifica al browser *come* farlo.

Introduciamo adesso i pulsanti di selezione e le caselle di scelta. I pulsanti permettono all'utente di scegliere una fra diverse possibilità. Le caselle di scelta gli permettono di scegliere una o più fra *tutte* le opzioni possibili.
Tanto per iniziare costruiamo qualche pulsante.

```
<INPUT TYPE=RADIO NAME="BEST FRIEND">
```

Ora aggiungiamone altri due.

```
<INPUT TYPE=RADIO NAME="BEST FRIEND">
<INPUT TYPE=RADIO NAME="BEST FRIEND">
<INPUT TYPE=RADIO NAME="BEST FRIEND">
```

Mettiamo un interruzione di linea fra l'una e l'altra.

```
<INPUT TYPE=RADIO NAME="BEST FRIEND"><BR>
<INPUT TYPE=RADIO NAME="BEST FRIEND"><BR>
<INPUT TYPE=RADIO NAME="BEST FRIEND"><P>
```

Nota che ciascun input ha lo stesso nome. La ragione sarà chiara in seguito.

A ciascuno dei pulsanti va assegnato un valore [VALUE].

```
<INPUT TYPE=RADIO NAME="BEST FRIEND" VALUE="QUI"><BR>
<INPUT TYPE=RADIO NAME="BEST FRIEND" VALUE="QUO"><BR>
<INPUT TYPE=RADIO NAME="BEST FRIEND" VALUE="QUA"><P>
```

Ora contrassegniamo ciascun pulsante.

```
<INPUT TYPE=RADIO NAME="BEST FRIEND" VALUE="QUI"> primo nipote<BR>
<INPUT TYPE=RADIO NAME="BEST FRIEND" VALUE="QUO"> secondo nipote<BR>
<INPUT TYPE=RADIO NAME="BEST FRIEND" VALUE="QUA"> terzo nipote<P>
```

primo nipote

secondo nipote

terzo nipote

I pulsanti adesso sono praticamente pronti. Si può abbellire un po' il tutto aggiungendo una frase prima dei pulsanti e, se si desidera, scegliere uno dei pulsanti come default (opzionale).

```
Chi è il tuo migliore amico?<br>

<INPUT TYPE=RADIO NAME="BEST FRIEND" VALUE="QUI" CHECKED> primo nipote <BR>
<INPUT TYPE=RADIO NAME="BEST FRIEND" VALUE="QUO"> secondo nipote <BR>
<INPUT TYPE=RADIO NAME="BEST FRIEND" VALUE="QUA"> terzo nipote <P>
```

Chi è il tuo migliore amico?

primo nipote

secondo nipote

terzo nipote

Fine modulo

L'utente ovviamente può selezionare solo 1 opzione. La scelta sarà spedita come coppia nome/valore. `BEST FRIEND=QUI`(o uno qualsiasi dei pulsanti scelto).

Costruire delle Caselle di Scelta è all'incirca la stessa cosa. Cominciamo con:

```
<INPUT TYPE=CHECKBOX NAME="QUI">
```

Aggiungiamone altre tre ma stavolta diamo a ciascuna un `NAME` differente. (Puoi anche aggiungere degli a capo se preferisci)

```
<INPUT TYPE=CHECKBOX NAME="QUI"><BR>
<INPUT TYPE=CHECKBOX NAME="QUO"><BR>
<INPUT TYPE=CHECKBOX NAME="QUA"><BR>
<INPUT TYPE=CHECKBOX NAME="QUE"><P>
```

Ogni casella ha lo stesso `VALUE`.

```
<INPUT TYPE=CHECKBOX NAME="QUI" VALUE="SI"><BR>
<INPUT TYPE=CHECKBOX NAME="QUO" VALUE="SI"><BR>
<INPUT TYPE=CHECKBOX NAME="QUA" VALUE="SI"><BR>
<INPUT TYPE=CHECKBOX NAME="QUE" VALUE="SI"><P>
```

Nota- nelle Caselle di Scelta il `NAME` cambia e il `VALUE` rimane sempre lo stesso, mentre con i Pulsanti il `VALUE` cambia e il `NAME` rimane sempre uguale.
OK, contrassegnamo ciasuna casella.

```
<INPUT TYPE=CHECKBOX NAME="QUI" VALUE="SI"> QUI<BR>
<INPUT TYPE=CHECKBOX NAME="QUO" VALUE="SI"> QUO<BR>
<INPUT TYPE=CHECKBOX NAME="QUA" VALUE="SI"> QUA<BR>
<INPUT TYPE=CHECKBOX NAME="QUE" VALUE="SI"> QUE<P>
```

☐ QUI
☐ QUO
☐ QUA
☐ QUE
Fine modulo

Infine, magari si vuole aggiungere qualcosetta prima delle caselle e forse anche assegnare qualche valore di default.

```
Quali dei seguenti sono tuoi amici?<BR>
<INPUT TYPE=CHECKBOX NAME="QUI" VALUE="SI" CHECKED> QUI<BR>
<INPUT TYPE=CHECKBOX NAME="QUO" VALUE="SI"> QUO<BR>
<INPUT TYPE=CHECKBOX NAME="QUA" VALUE="SI" CHECKED> QUA<BR>
<INPUT TYPE=CHECKBOX NAME="QUE" VALUE="SI"> QUE<P>
```

Quali dei seguenti sono tuoi amici?
☑ QUI
☐ QUO
☑ QUA
☐ QUE
L'utente può scegliere 1, 2, nessuna o tutte le possibili opzioni. La scelta effettuata verrà inviata come coppie nome/valore...

```
QUI=SI
QUA=SI
```
(o qualunque cosa venga scelta. Se la scelta è NESSUNA, non viene mandato niente)

Ovviamente in un form si possono fare più domande sullo stesso gruppo

Quali dei seguenti sono amici?	Quali dei seguenti sono parenti?	Quali dei seguenti sono conoscenti?
☐ QUI	☐ QUI	☐ QUI
☐ QUO	☐ QUO	☐ QUO
☐ QUA	☐ QUA	☐ QUA
☐ QUE	☐ QUE	☐ QUE

In un singolo form è opportuno che non si ripetano mai gli stessi `NAME`. Dunque, in questo caso, meglio usare *un nome diverso* per ciascuna domanda.

Ecco qui il codice HTML per le tre domande.

```
<CENTER>
<TABLE WIDTH=600 BORDER=1 CELLSPACING=1><TR>

<TD WIDTH=199>
Quali dei seguenti sono amici?<BR>
```

```
<INPUT TYPE=CHECKBOX NAME="Amici?..QUI" VALUE="SI"> QUI<BR>
<INPUT TYPE=CHECKBOX NAME="Amici?..QUO" VALUE="SI"> QUO<BR>
<INPUT TYPE=CHECKBOX NAME="Amici?..QUA" VALUE="SI"> QUA<BR>
<INPUT TYPE=CHECKBOX NAME="Amici?..QUE" VALUE="SI"> QUE<P>
</TD>

<TD WIDTH=200>
Quali dei seguenti sono parenti?<BR>
<INPUT TYPE=CHECKBOX NAME="Parenti?..QUI" VALUE="SI"> QUI<BR>
<INPUT TYPE=CHECKBOX NAME="Parenti?..QUO" VALUE="SI"> QUO<BR>
<INPUT TYPE=CHECKBOX NAME="Parenti?..QUA" VALUE="SI"> QUA<BR>
<INPUT TYPE=CHECKBOX NAME="Parenti?..QUE" VALUE="SI"> QUE<P>
</TD>

<TD WIDTH=199>
Quali dei seguenti sono conoscenti?<BR>
<INPUT TYPE=CHECKBOX NAME="Conoscenti?..QUI" VALUE="SI"> QUI<BR>
<INPUT TYPE=CHECKBOX NAME="Conoscenti?..QUO" VALUE="SI"> QUO<BR>
<INPUT TYPE=CHECKBOX NAME="Conoscenti?..QUA" VALUE="SI"> QUA<BR>
<INPUT TYPE=CHECKBOX NAME="Conoscenti?..QUE" VALUE="SI"> QUE<P>
</TD>

</TR></TABLE>
</CENTER>
```

Facciamo finta che un utente selezioni queste caselle...

Quali dei seguenti sono amici?	Quali dei seguenti sono parenti?	Quali dei seguenti sono conoscenti?
☑ QUI	☐ QUI	☑ QUI
☑ QUO	☐ QUO	☑ QUO
☑ QUA	☑ QUA	☐ QUA
☐ QUE	☑ QUE	☑ QUE

...In questo caso verranno inviati i le seguenti coppie di nome/valore.

```
Amici?..QUI=SI
Amici?..QUO=SI
Amici?..QUA=SI
PArenti?...QUA=SI
Parenti?...QUE=SI
Conoscenti?...QUI=SI
Conoscenti?...QUO=SI
Conoscenti?...QUE=SI
```

Il prossimo tipo di input è la Lista a Scorrimento. Con questo tipo di input bisogna usare `<SELECT>` invece di `<INPUT>` *e* c'è anche un tag di chiusura. Costruiamone una.

```
<SELECT>
</SELECT>
```

Non dimentichiamo di assegnargli un nome.

```
<SELECT NAME="MIGLIOR AMICO">
</SELECT>
```

Ora aggiungiamo qualche *opzione*.

```
<SELECT NAME="MIGLIOR AMICO">
<OPTION>QUI
<OPTION>QUO
<OPTION>QUA
<OPTION>QUE
</SELECT>
```

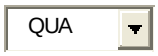
E assegniamo a ciascuna opzione [`<OPTION>`] un valore [`VALUE`].

```
<SELECT NAME="BEST FRIEND">
<OPTION VALUE="QUI">QUI
<OPTION VALUE="QUO">QUO
<OPTION VALUE="QUA">QUA
<OPTION VALUE="QUE">QUE
</SELECT>
```

L'opzione di default è quella che sta per prima.

E' comunque possibile specificare un altro default.

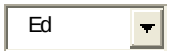
```
<SELECT NAME="BEST FRIEND">
<OPTION VALUE="QUI">QUI
<OPTION VALUE="QUO">QUO
<OPTION VALUE="QUA" Selected>QUA
<OPTION VALUE="QUE">QUE
</SELECT>
```



Una Lista Scorrevole viene costruita in maniera molto simile. Innanzitutto aggiungiamo qualche alro nome.

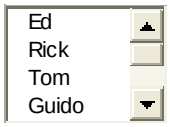
```
<SELECT NAME="BEST FRIEND">
<OPTION VALUE="Ed">Ed
<OPTION VALUE="Rick">Rick
<OPTION VALUE="Tom">Tom
<OPTION VALUE="Guido">Guido
```

```
<OPTION VALUE="Horace">Horace
<OPTION VALUE="Reggie">Reggie
<OPTION VALUE="Myron">Myron
</SELECT>
```



Tutto quello che dobbiamo fare per trasformarlo in una lista scorrevole è aggiungere l'attributo `SIZE` al tag `<SELECT>`.

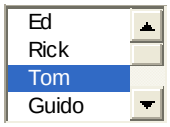
```
<SELECT NAME="BEST FRIEND" SIZE=4>
<OPTION VALUE="Ed">Ed
<OPTION VALUE="Rick">Rick
<OPTION VALUE="Tom">Tom
<OPTION VALUE="Guido">Guido
<OPTION VALUE="Horace">Horace
<OPTION VALUE="Reggie">Reggie
<OPTION VALUE="Myron">Myron
</SELECT>
```



L'attributo `SIZE` specifica semplicemente quante opzioni vengono visualizzate assieme nella finestra. Non è meraviglioso?

Anche questa volta il valore di default è la prima opzione [`<OPTION>`] specificata, e di nuovo possiamo modificarlo scegliendone uno.

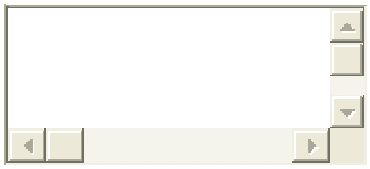
```
<SELECT NAME="BEST FRIEND" SIZE=4>
<OPTION VALUE="Ed">Ed
<OPTION VALUE="Rick">Rick
<OPTION VALUE="Tom" SELECTED>Tom
<OPTION VALUE="Guido">Guido
<OPTION VALUE="Horace">Horace
<OPTION VALUE="Reggie">Reggie
<OPTION VALUE="Myron">Myron
</SELECT>
```



Non ho la minima idea del *perchè* qualcuno possa essere interessato a utilizzare il tag `SELECTION` per fare una lista scorrevole. Tuttavia la possibilità esiste e non ho retto alla tentazione di mostrarvi come si fa. ☺

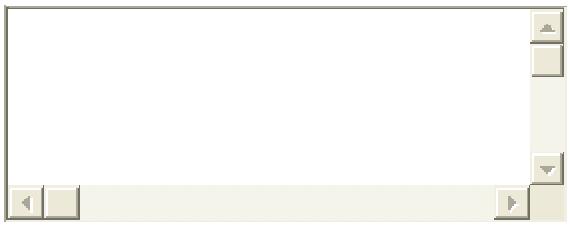
Un tipo di input estremamente utile è il `<TEXTAREA>`.

```
<TEXTAREA NAME="COMMENTS">
</TEXTAREA>
```

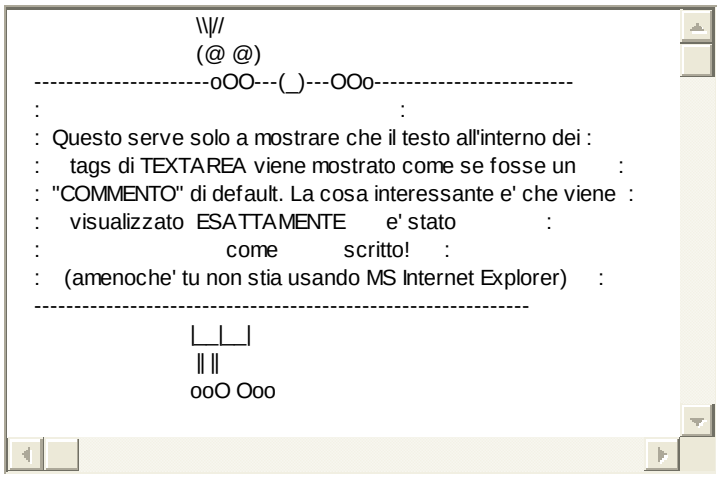


Puoi determinare la dimensione del riquadro in questo modo...

```
<TEXTAREA NAME="COMMENTS" ROWS=6 COLS=50>
</TEXTAREA>
```



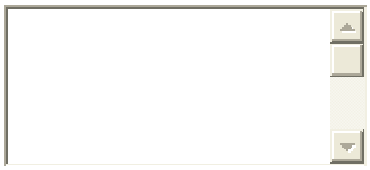
`ROWS` è l'altezza, `COLS` la larghezza.



Un attributo che è spesso bene includere in `<TEXTAREA>` è `WRAP`. Alcuni browsers non lo interpretano, ma in questo caso si limitano ad ignorarlo.

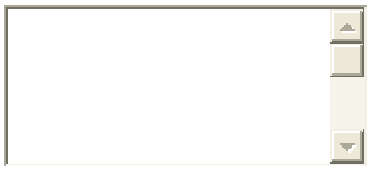
Andiamo avanti e scriviamo nel riquadro

```
<TEXTAREA NAME="COMMENTS" ROWS=3 COLS=30 WRAP=VIRTUAL>
</TEXTAREA>
```



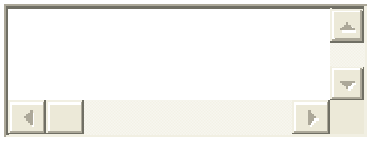
`WRAP=VIRTUAL` significa che il testo nel riquadro va a capo, ma viene *spedito* come una sola riga.

```
<TEXTAREA NAME="COMMENTS" ROWS=3 COLS=30 WRAP=PHYSICAL>
</TEXTAREA>
```



`WRAP=PHYSICAL` significa che il testo nel riquadro va a capo, *e* viene *spedito* allo stesso modo.

```
<TEXTAREA NAME="COMMENTS" ROWS=3 COLS=30 WRAP=OFF>
</TEXTAREA>
```



Questo è il valore di default.
`WRAP=OFF` significa che il testo nel riquadro *non* va a capo, ma viene *spedito* esattamente nello stesso modo in cui è stato scritto (come l'omino che c'era in un textarea precedente).

11. I frame

Dividere una pagina in frames attualmente è estremamente semplice. Il concetto di base è più o meno questo: ogni frame è un semplice e completo documento html. Se vuoi dividere la tua pagina in due frames affiancati allora dovresti mettere un documento html completo nel frame sinistro ed un'altro documento completo nel frame destro. In più hai bisogno di scrivere un *terzo* documento html. Questo *DOCUMENTO MASTER* contiene i tags `<FRAME>` che specificano *cosa* va messo e *dove*. In effetti, questa è l'unica funzione del documento master.

Per i frames ci sono solo due tags principali fra cui scegliere: **<FRAMESET>** e **<FRAME>**.

Creiamo una prima pagina web:

```
<HTML>
<HEAD>
<TITLE> </TITLE>
</HEAD>
<BODY>
Lisa
</BODY>
</HTML>
```

Salviamo la pagina con il nome lisa.html

Creiamo una seconda pagina web

```
<HTML>
<HEAD>
<TITLE> </TITLE>
</HEAD>
<BODY>
Terri
</BODY>
</HTML>
```

Salviamolo nella stessa cartella come `terri.html`.

Ora facciamo lo stesso per Kim, Tina, Shannon, e Beth. Salviamoli come abbiamo fatto per gli altri. Ora dovrebbe esserci una cartella che contiene 6 documenti html completi ed indipendenti.

Adesso costruiamo la pagina master. Partiamo con questo.

```
<HTML>
<HEAD>
<TITLE> Master Page</TITLE>
</HEAD>

<BODY>
</BODY>

</HTML>
```

Rimuoviamo i tags `<BODY>`. La pagina master non li usa...

```
<HTML>
<HEAD>
<TITLE> Master Page</TITLE>
</HEAD>



</HTML>
```

---

usiamo invece i tags `<FRAMESET>`.

```
<HTML>
<HEAD>
<TITLE>Master Page</TITLE>
</HEAD>

<FRAMESET>
</FRAMESET>
```

```
</HTML>
```

Concentriamo la nostra attenzione sul tag frameset

```
<FRAMESET>
</FRAMESET>
```

Salviamolo nella cartella (con tutte le ragazze) come `index.html`.

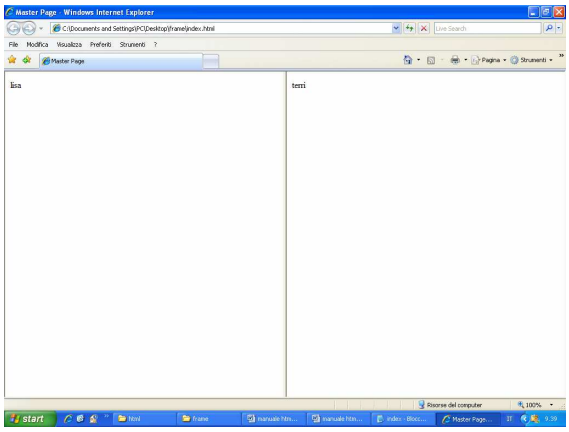
Ora cominciamo a definire come devono apparire le cose. Occorre dire al browser di dividere la finestra principale in 2 colonne, ognuna delle quali occupi il 50% della finestra piena.

```
<FRAMESET COLS="50%,50%">
</FRAMESET>
```

Dobbiamo dire al browser cosa mettere in ogni frame.

```
<FRAMESET COLS="50%,50%">
  <FRAME SRC="lisa.html">
  <FRAME SRC="terri.html">
</FRAMESET>
```

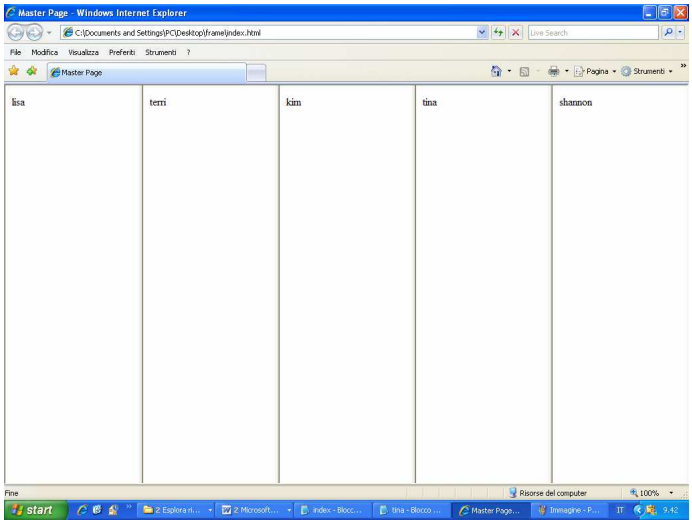
Otterremo:



Adesso dividiamo la pagina ad esempio in 5 colonne

```
<FRAMESET COLS="20%,20%,20%,20%,20%">
  <FRAME SRC="lisa.html">
  <FRAME SRC="terri.html">
  <FRAME SRC="kim.html">
  <FRAME SRC="tina.html">
  <FRAME SRC="shannon.html">
</FRAMESET>
```

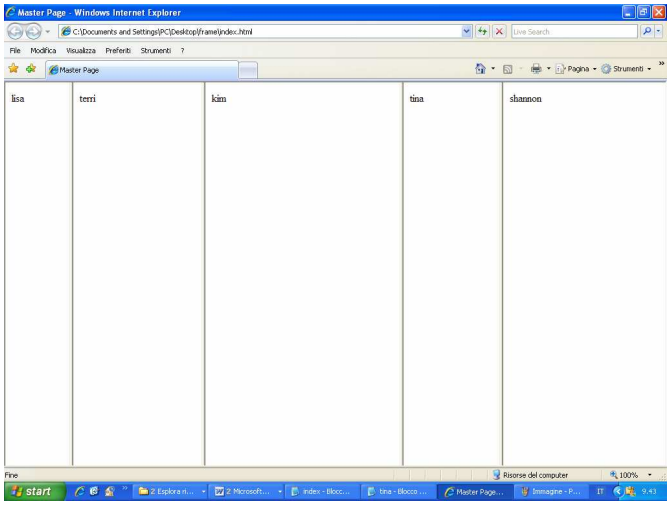
Otterremo:



Naturalmente è ovvio che possiamo dividere in frames di differente misura.

```
<FRAMESET COLS="10%,20%,30%,15%,25%">
  <FRAME SRC="lisa.html">
  <FRAME SRC="terri.html">
  <FRAME SRC="kim.html">
  <FRAME SRC="tina.html">
  <FRAME SRC="shannon.html">
</FRAMESET>
```

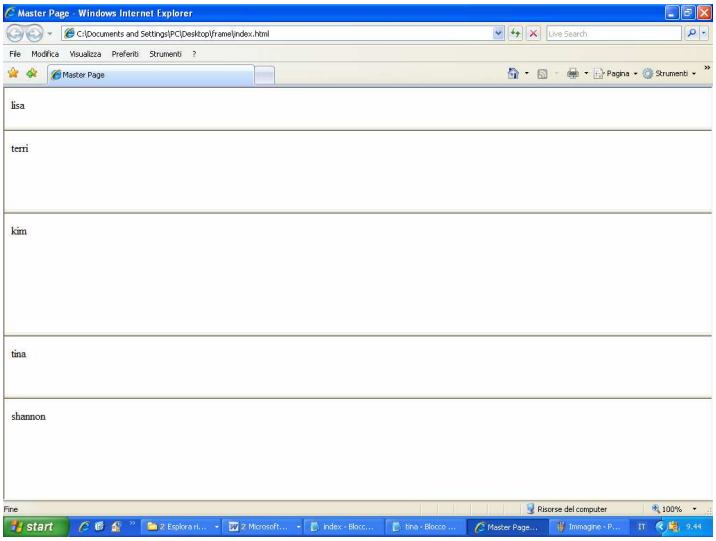
Otterremo:



Se specifichiamo di dividere in righe (ROWS) anzichè colonne (COLS) otteniamo qualcosa di completamente diverso.

```
<FRAMESET ROWS="10%,20%,30%,15%,25%">
  <FRAME SRC="lisa.html">
  <FRAME SRC="terri.html">
  <FRAME SRC="kim.html">
  <FRAME SRC="tina.html">
```

```
  <FRAME SRC="shannon.html">
</FRAMESET>
```

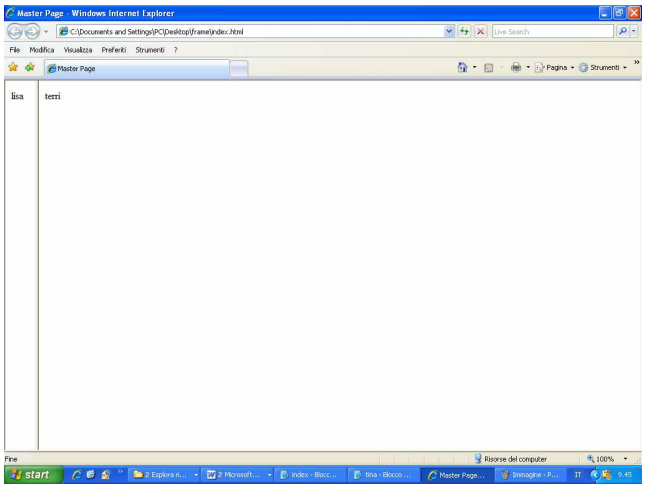


Torniamo ai nostri 2 frames, divisi equamente in colonne.

```
<FRAMESET COLS="50%,50%">
  <FRAME SRC="lisa.html">
  <FRAME SRC="terri.html">
</FRAMESET>
```

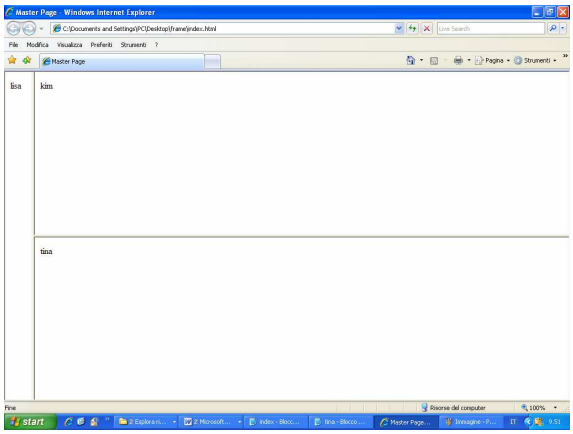
Possiamo specificare 50 pixels invece del 50%. E possiamo usare * al posto dei numeri. Il significato di * è "ciò che resta".

```
<FRAMESET COLS="50,*">
  <FRAME SRC="lisa.html">
  <FRAME SRC="terri.html">
</FRAMESET>
```



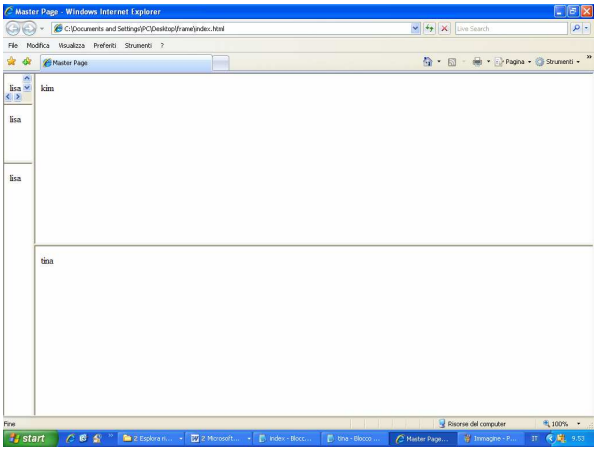
Adesso dividiamo il frame occupato da Kim a metà orizzontalmente. Per prima cosa dobbiamo sostituire Kim con un'altra coppia di tags `<FRAMESET>` `ed inserire altri due frame`.

```
<FRAMESET COLS="50,*">
  <FRAME SRC="lisa.html">
  <FRAMESET ROWS="50%,50%">
     <FRAME SRC="kim.html">
     <FRAME SRC="tina.html">
  </FRAMESET>
</FRAMESET>
```



Dividiamo Lisa orizzontalmente in 3 parti. Metteremo Lisa in tutte e tre. Ecco tutti i cambiamenti in un colpo solo.

```
<FRAMESET COLS="50,*">
  <FRAMESET ROWS="50,100,*">
    <FRAME SRC="lisa.html">
    <FRAME SRC="lisa.html">
    <FRAME SRC="lisa.html">
  </FRAMESET>
  <FRAMESET ROWS="50%,50%">
    <FRAME SRC="kim.html">
    <FRAME SRC="tina.html">
  </FRAMESET>
</FRAMESET>
```



Possiamo inserire delle immagini nel frame se lo vogliamo.

```
<FRAMESET COLS="50%,50%">
  <FRAME SRC="world.gif" WIDTH=146 HEIGHT=162>
  <FRAME SRC="terri.html">
</FRAMESET>
```

Inseriamo la figura nel frame. Prima cosa portiamo la finestra di sinistra alla larghezza di 146 pixels. Dal momento che usiamo una misura assoluta dovremmo rendere gli altri frames elastici.

```
<FRAMESET COLS="146,*">
  <FRAME SRC="world.gif" WIDTH=146 HEIGHT=162>
  <FRAME SRC="terri.html">
</FRAMESET>
```

Potrebbe verificarsi la visualizzazione della barra di scorrimento. Essa può essere gestita dall'attributo scrolling

Le barre di scorrimento *(scrollbars)* che puoi vedere possono essere specificate come `YES`, `NO` o `AUTO`. `YES` significa che saranno messe le scrollbars- sia che siano necessarie o meno. `NO` significa che non ci saranno scrollbars, anche se il contenuto del frame è più grande del frame stesso il browser semplicemente mostrerà tutto quello che può. `AUTO` è il default. Se le scrollbars sono necessarie appariranno, se non sono necessarie non verrano mostrate. Impostiamo dunque le nostre scrollbars.

```
<FRAMESET COLS="146,*">
  <FRAMESET ROWS="162,*">
    <FRAME SRC="world.gif" WIDTH=146 HEIGHT=162 SCROLLING=NO>
    <FRAME SRC="lisa.html">
  </FRAMESET>
  <FRAME SRC="terri.html">
</FRAMESET>
```

I prossimi due attributi che analizzeremo sono relativi ai *margini*. Il browser fornisce automaticamente ad ogni frame un po' di spazio vuoto attorno al contenuto. Questo è normalmente necessario per questioni di estetica. Si può controllare la misura di questi margini usando `MARGINWIDTH` e `MARGINHEIGHT`. Questi attributi controllano i margini sinistro & destro e alto & basso rispettivamente. Se non si vogliono tali margini basta impostarli entrambi a 1. (1 è il minimo)

```
<FRAMESET COLS="146,*">
  <FRAMESET ROWS="162,*">
    <FRAME SRC="world.gif" WIDTH=146 HEIGHT=162 SCROLLING=NO
     MARGINWIDTH=1 MARGINHEIGHT=1>
    <FRAME SRC="lisa.html">
  </FRAMESET>
  <FRAME SRC="terri.html">
</FRAMESET>
```

Inoltre è possibile dimensionare il bordo mediante l'attributo border e il suo colore mediante l'attributo bordercolor

Prima cosa cambiamo lo spessore del bordo.

```
<FRAMESET COLS="154,*" BORDER=20>
  <FRAMESET ROWS="170,*">
    <FRAME SRC="world.gif" WIDTH=146 HEIGHT=162 SCROLLING=NO
     MARGINWIDTH=1 MARGINHEIGHT=1>
    <FRAME SRC="lisa.html">
```

```
  </FRAMESET>
  <FRAME SRC="terri.html">
</FRAMESET>
```

Poi possiamo cambiare il *colore* dei bordi.

```
<FRAMESET COLS="154,*" BORDER=20 BORDERCOLOR="#FF0000">
  <FRAMESET ROWS="170,*">
    <FRAME SRC="world.gif" WIDTH=146 HEIGHT=162 SCROLLING=NO
     MARGINWIDTH=1 MARGINHEIGHT=1>
    <FRAME SRC="lisa.html">
  </FRAMESET>
  <FRAME SRC="terri.html">
</FRAMESET>
```

Possiamo eliminare i bordi nei singoli `<FRAMESET>` con `FRAMEBORDER`.

```
<FRAMESET COLS="154,*" BORDER=20 BORDERCOLOR="#FF0000">
  <FRAMESET ROWS="170,*" FRAMEBORDER=NO >
    <FRAME SRC="world.gif" WIDTH=146 HEIGHT=162 SCROLLING=NO
     MARGINWIDTH=1 MARGINHEIGHT=1>
    <FRAME SRC="lisa.html">
  </FRAMESET>
  <FRAME SRC="terri.html">
</FRAMESET>
```

Possiamo evitare che il browser esegua il ridimensionamento *(resizing)* del frame.

```
<FRAMESET COLS="154,*" BORDER=20 BORDERCOLOR="#FF0000">
  <FRAMESET ROWS="170,*" FRAMEBORDER=NO >
    <FRAME SRC="world.gif" WIDTH=146 HEIGHT=162 SCROLLING=NO
     MARGINWIDTH=1 MARGINHEIGHT=1>
    <FRAME SRC="lisa.html">
  </FRAMESET>
  <FRAME SRC="terri.html" NORESIZE>
</FRAMESET>
```

Dopo questa carrellata di attributi tra i frame introduciamo il *linking* tra i frames. Esso viene fatto con il famoso tag A seguito da href. Se viene utilizzato solo questo tag con l’attributo href la pagina si aprirà nello stesso frame. Se invece voglio indirizzarla ad un altro frame devo prima assegnare un name nella pagina master ad ogni frame ed utilizzare quindi all’interno del tag A l’attributo target. Supponiamo di aver assegnato il name ad un frame ad esempio pippo, allora affinché una pagina vernga indirizzata in questo frame dovremo inserire il seguente codice:

```
<A HREF="shannon.html" TARGET="Pippo">Shannon</A>
```

Questo provocherà l'apertura del link nella window che si chiama `pippo`.

`Vi sono 4 valori predefiniti per target.` Questi sono `_self`, `_blank`, `_parent` e `_top`. Questi sono gli unici target che possono cominciare con qualcosa di diverso dai caratteri

alfanumerici. In più ogni target comincia con un underscore_ cioè *nessuno* dei 'target magici' sarà ignorato. Per quello che dobbiamo fare, `_top` è l'unico che ci interessi per il momento.

Nota- E' importante specificare `TARGET="_top"` piuttosto che `TARGET="_TOP"`. Di norma l'HTML non è sensibile alle maiuscole/minuscole *(case-sensitive)* ma in questo caso lo è. Usare `_TOP` al posto di `_top` qualche volta può provocare l'apertura del link in una nuova finestra del browser al posto della finestra a tutto schermo. Dal momento che l'abbiamo menzionato, questo sarebbe il compito di `TARGET="_blank"`.... ovvero: caricare il contenuto del link in una nuova finestra del browser.

12. I CSS

La sigla CSS vuol dire "**Cascading Style Sheets**" ovvero "**Fogli di Stile a Cascata**", si chiamano così perchè le regole css vengono applicate "a cascata agli elementi che compongono la pagina". Ci forniscono un modo rapido e preciso per visualizzare gli oggetti che compongono una pagina web, applicando delle semplici regole che vedremo nel corso di questa dispensa.

Il loro punto di forza sta nella semplicità di utilizzo, e nella flessibilità che danno ai documenti prodotti. Supponiamo ad esempio di avere una pagina in cui c'è del testo che si ripete (ad esempio dentro una tabella) e di voler colorare il testo di rosso:

In HTML scriveremmo:

```
<table>
  <tr>
    <td> <font color="red">testo</font> </td>
  </tr>
  <tr>
    <td> <font color="red">testo</font> </td>
  </tr>
  <tr>
    <td> <font color="red">testo</font> </td>
  </tr>
  <tr>
    <td> <font color="red">testo</font> </td>
  </tr>
</table>
```

Possiamo ben immaginare che qualora volessimo cambiare il colore del testo, dovremmo effettuare pesanti modifiche sulle pagine. Tale problema è risolvibile attraverso i CSS.

Esistono 4 metodi per utilizzare i css:
1. *richiamarli direttamente nelle pagine come stile dell’elemento usando l’attributo style*
2. *utilizzare l’elemento* ***<style>*** *nell’head dei nostri documenti HTML*
3. *richiamando una pagina di stili esterna attraverso l’elemento* ***<link>***
4. *utilizzando la direttiva @import in* ***<style>****.*

Tutti e 4 i metodi permettono di fare la stessa cosa anche se con alcune differenze. Il vantaggio di utilizzare fogli di stili esterni è comunque notevole: nel caso in cui volessimo cambiare look ad un documento ci basterà modificare il foglio di stile.
Molto simile è il comportamento della direttiva @import che in più esclude alcuni vecchi browser che non hanno implementato i css in modo corretto.
Gli altri due metodi ci possono esser utili per provare al volo ciò che abbiamo pensato: sono molto veloci e pratici.

Altro punto importante che iniziamo ad accennare è l’ereditarietà e la priorità delle regole impostate dai css: se diamo due stesse regole allo stesso elemento, magari una nel foglio di stile esterno ed una nell’elemento **<style>** il motore interpreterà quello a maggior priorità.
In tal caso la regola inserita in <style> andrà a *sovrascrivere* quella del css esterno.
Vediamo come inserire *praticamente* i nostri stili.
1. Richiamare i css nelle pagine utilizzando l’attributo style.
L’uso in tal caso è molto semplice.
Supponiamo di voler colorare di rosso un paragrafo. Con questo metodo basta dichiararlo direttamente nell’elemento :
Hello World!
ed il gioco è fatto.
2. Utilizzare l’elemento <style> nell’head dei nostri documenti HTML.
Con questo metodo è sufficiente inserire le proprietà direttamente nell’ *head*:

```
<html>
<head>
<style type=”text/css” media=”all”>
p {color:red; }
</style>
</head>
<body>
<p>Hello World!</p>
</body>
</html>
```

Style può avere due attributi:
type obbligatorio, che ha la funzione di specificare il tipo; in pratica sarà però sempre e solo *text/css* e **media** che serve a specificare a quale piattaforma applicare un foglio di stile
3. richiamare una pagina di stili esterna attraverso l’elemento <link>.
<link rel=’stylesheet’ type=’text/css’ href=’css.css’ media=’all’>
Questo metodo prevede 4 possibili attributi: **type** e **media** con la stessa funzione appena vista, **href** che serve a specificare i percorso del foglio di stile (obbligatorio) e **rel** che specifica la relazione tra il foglio di stile ed il documento html. Può assumere due valori: *stylesheet* e *alternate stylesheet*.
4. utilizzare la direttiva @import in <style>
Questo metodo, prevede di indicare il percorso del foglio di testo entro parentesi tonde nell’elemento style nella forma

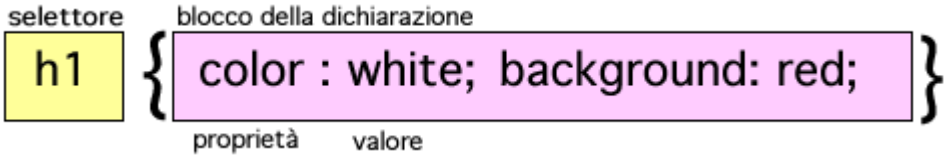
```
<style>
@import url(http://miosito.ext/css.css);
</style>
```

12.1. Come è fatto un CSS: regole e commenti

Iniziamo con l'analisi degli elementi costitutivi di un foglio di stile. Un foglio di stile non è altro che un insieme di regole, accompagnate, volendo, da qualche nota di commento. Andiamo innanzitutto a spiegare cos'è e com'è fatta una regola.

12.2. Com'è fatta una regola

Figura 1. Struttura di una regola



L'illustrazione mostra la tipica struttura di una regola CSS. Essa è composta da due blocchi principali:

- **il selettore**
- **il blocco delle dichiarazioni**

Il selettore serve a definire la parte del documento cui verrà applicata la regola. In questo caso, ad esempio, verranno formattati tutti gli elementi **<H1>**: lo sfondo sarà rosso, il colore del testo bianco.

Il blocco delle dichiarazioni è delimitato rispetto al selettore e alle altre regole da **due parentesi graffe**. Al suo interno possono trovare posto più dichiarazioni. Esse sono sempre composte da una coppia:

- **proprietà**
- **valore**

La proprietà definisce un aspetto dell'elemento da modificare (margini, colore di sfondo, etc) secondo il valore espresso. Proprietà e valore devono essere separati dai **due punti**. Una limitazione fondamentale da rispettare è questa: per ogni dichiarazione non è possibile indicare più di una proprietà, mentre è spesso possibile specificare più valori. Questa regola è pertanto errata:

```
body {color background: black;}
```

Mentre questa è perfettamente valida e plausibile:

```
p {font: 12px Verdana, arial;}
```

Ancora, se in un blocco si definiscono più dichiarazioni, come nell'esempio in figura 1, esse vanno separate dal **punto e virgola**. Il linguaggio non impone che si metta il punto e virgola dopo l'ultima dichiarazione, ma alcuni browser più datati lo richiedono: aggiungetelo sempre per sicurezza e per una maggiore compatibilità.

Gli spazi bianchi lasciati all'interno di una regola non influiscono sul risultato. Il consiglio, anzi, è di lasciare sempre uno spazio tra le varie parti per una migliore leggibilità.

12.3. Commenti

Per inserire parti di commento in un CSS racchiudetelo tra questi segni:

- **/*** come segno di apertura
- ***/** come segno di chiusura

12.4. Proprietà singole e a sintassi abbreviata

Nelle definizione delle regole è possibile fare uso di **proprietà singole** e **proprietà a sintassi abbreviata**. Con questa espressione traduciamo il termine inglese **shorthand properties** reso spesso, alla lettera, con il termine *scorciatoie*.

Le proprietà singole sono la maggior parte: impostano per un dato elemento o selettore un singolo aspetto. Con le **shorthand properties**, è possibile invece definire con una sola dichiarazione un insieme di proprietà. Chiariamo con un esempio.

Ogni elemento presenta sui suoi quattro lati un certo margine rispetto a quelli adiacenti. È possibile definire per ciascuno di essi un valore usando quattro proprietà singole separate:

- **margin-top**
- **margin-right**
- **margin-bottom**
- **margin-left**

La regola sarebbe questa:

```
div { margin-top: 10px;
      margin-right: 5px;
      margin-bottom: 10px;
      margin-left: 5px;
}
```

Lo stesso risultato si può ottenere usando la proprietà a sintassi abbreviata **margin**:

```
div {margin: 10px 5px 10px 5px;}
```

Approfondiremo nel corso dell'analisi delle proprietà usi e costrutti sintattici di ciascuna. Per il momento ci limitiamo all'elenco:

```
background  | border   | border-top | border-right | border-bottom
border-left | cue      | font       | list-style   | margin
outline     | padding  | pause
```

12.5. I selettori

Fondamentalmente una regola CSS viene applicata ad un **selettore**. La parola parla da sé: si tratta di una semplice dichiarazione che serve a **selezionare** la parte o le parti di un documento soggette ad una specifica regola. Quella che segue è una lista commentata dei vari tipi di selettore.

12.5.1. Selettore di elementi (type selector)

È il più semplice dei selettori. È costituito da uno qualunque degli elementi di HTML.

Sintassi

h1 {color: #000000;}
p {background: white; font: 12px Verdana, arial, sans-serif;}
table {width: 200px;}

12.5.2. Raggruppamento

È possibile nei CSS raggruppare diversi elementi al fine di semplificare il codice. Gli elementi raggruppati vanno separati da una **virgola**.

Il raggruppamento è un'operazione molto conveniente. Pensate a questo scenario:

h1 {background: white;}
h2 {background: white;}
h3 {background: white;}

Tutti e tre gli elementi hanno uno sfondo bianco. Invece di scrivere tre regole separate si può fare così:

h1, h2, h3 {background: white;}

12.5.3. Selettore universale (universal selector)

Anche nei CSS abbiamo un jolly. Il selettore universale serve a selezionare tutti gli elementi di un documento. Si esprime con il carattere ***** (asterisco).

Sintassi

* { color: black; }

Associa a qualsiasi elemento della pagina il colore nero

12.5.4. Selettore del discendente (descendant selector)

Questa opzione ci permette di associare regole a particolari elementi se e solo se contenuti in altri elementi definiti:
div strong { color: red; }
Associa il colore rosso solo agli elementi *strong* contenuti in div.
<body>
Questo non sarà rosso in quanto non contenuto in un un div
Il testo successivo, **questo in particolare** sarà mostrato in rosso
Anche **questo testo** sarà mostrato in rosso; non è importante che i due elementi siano *adiacenti o figli*.

12.5.5. Selettore del figlio (child selector)

Seleziona un elemento che sia figlio diretto di un altro.

Il child selector è solo in apparenza simile al descendant selector. La differenza sta nella relazione di discendenza tra gli elementi, che in questo caso deve essere di primo livello. Potrebbe essere una buona soluzione a molti problemi, purtroppo non è supportato, così come il selettore adiacente, da Internet Explorer.
div > strong { color: red; }
Es:
<body>
Il testo successivo, **questo in particolare** sarà mostrato in rosso
Anche **questo testo** NON sarà mostrato in rosso in quanto tra div e strong c’è un altro elemento, mentre **questo testo** vicecersa lo sarà.
</body>

12.5.6. Pseudo-classi dei links

E’ probabilmente una delle feature più interessanti dei Css: permette di associare ai links (ed anche agli altri elementi, tranne :link e :visited, in teoria; in pratica visto il non supporto di IE solo all’elemento A) regole diverse a seconda se il link **non è stato visitato (:link), già seguito (:visited), attivo (:active), al passaggio del mouse (:hover) e se selezionato (:focus).**
Es:
a:link { color: blue; }
a:visited { color: purple; }
a:active { color: black; }
a:hover { color: red; }
a:focus { color: green; }

12.6. Id e classi: due selettori speciali

I CSS non sarebbero uno strumento così potente senza questi tipi di selettori. **Classi** e **ID** sono davvero una delle chiavi per sfruttare al meglio questo linguaggio.

Partiamo dall'inizio. In HTML esistono due attributi fondamentali applicabili a tutti gli elementi: sono **class** e **id**. Dichiarare questi attributi a prescindere dai CSS non ha alcun senso e non modifica in alcun modo la presentazione della pagina. Ecco un esempio.

....

Come vedete non succede nulla. Il problema è che il valore dell'attributo **class** deve trovare una corrispondenza in un foglio di stile. Se invece viene definito un CSS incorporato creando un selettore di tipo **classe** e assegnando ad esso il nome **testorosso**:

<style type="text/css">
.testorosso {
font: 12px arial, Helvetica, sans-serif;
color: #FF0000;
}
</style>

Il testo del nostro paragrafo sarà ora formattato secondo i nostri desideri: testo rosso, carattere arial. dimensione di 12px.

Lo stesso meccanismo è valido per i selettori di tipo **ID**. Ma con una sola fondamentale differenza: è ad essa che dovete fare riferimento per scegliere se usare una classe o un ID. In un documento HTML l'attributo **id** è usato per identificare in **modo univoco** un elemento. In pratica, se assegno ad un paragrafo l'id "testorosso", non potrò più usare questo valore nel resto della pagina. Di conseguenza, l'ID **#testorosso** dichiarato nel CSS trasformerà solo quel paragrafo specifico. Una singola classe, al contrario, può essere assegnata a più elementi, anche dello stesso tipo.

In un documento potrò avere senza problemi questa situazione:

```
<p class="testorosso">....</p>
<div class="testorosso">....</div>
<table class="testorosso">...</table>
<p class="testorosso">....</p>
```

La classe **.testorosso** presente nel CSS formatterà allo stesso modo il testo dei paragrafo, del div e della tabella.

Ma non questa:

```
<p id="testorosso">....</p>
<div id="testorosso">...</div>
```

Concludendo: una classe consente di superare le limitazioni intrinseche nell'uso di un selettore di elementi. Se imposto questa regola:

```
p {color: red;}
```

tutti i paragrafi della mia pagina avranno il testo rosso. E se volessi diversificare? Avere, ad esempio, anche paragrafi con il testo nero? Sarei prigioniero della regola iniziale. Scrivo due classi, una per il rosso e una per il nero, le applico di volta in volta secondo le mie necessità e il gioco è fatto.

La strategia dovrà dunque essere questa. Se uno stile va applicato ad un solo specifico elemento usate un **ID**. Se invece prevedete di usarlo più volte ovvero su più elementi definite nel CSS una classe.

Chiariti i concetti di base, passiamo ad analizzare usi e sintassi.

12.6.1. Classe

Per definire una classe si usa far precedere il nome da un semplice punto:

.nome_della_classe

Questa è la sintassi di base. Un selettore classe così definito può essere applicato a tutti gli elementi di un documento HTML.

Esiste un secondo tipo di sintassi:

<elemento>**.**nome_della_classe

Esso è più restrittivo rispetto alla sintassi generica. Se infatti definiamo questa regola:

```
p.testorosso {color: red;}
```

lo stile verrà applicato solo ai paragrafi che presentino l'attributo **class="testorosso"**. Anche qui è importante stabilire un minimo di strategia. Il secondo tipo di sintassi va usato solo se pensate di applicare una classe ad uno specifico tipo di elemento (solo paragrafi o solo div, e così via). Se invece ritenete di doverla applicare a tipi diversi usate la sintassi generica.

Una terza possibile modalità è quella che prevede la dichiarazione di classi multiple:

p.testorosso.grassetto {color:red; font-weight:bold;}

Questa regola applicherà gli stili impostati a tutti gli elementi in cui siano presenti (in qualunque ordine) i nomi delle classi definiti nel selettore. Avranno dunque il testo rosso e in grassetto questi paragrafi:

..
...

ma non questo, perchè solo uno dei nomi è presente come valore di **class**:

...

12.6.2. ID

La sintassi di un selettore **ID** è semplicissima. Basta far precedere il nome dal simbolo di cancelletto **#**:

#nome_id

Con questa regola:

#titolo {color: blue;}

assegniamo il colore blue all'elemento che presenti questa definizione:

<h1 id="titolo">...</h1>

Come per le classi è possibile usare una sintassi con elemento:

p#nome_id

In realtà questa modalità è assolutamente superflua. Se l'id è univoco non abbiamo alcun bisogno di distinguere l'elemento cui verrà applicata. Inoltre: la sintassi generica si rivela più razionale e utile. Se si dichiara un **ID** semplice è possibile assegnarlo a qualunque tipo di elemento. Posso usarlo su un paragrafo, ma se poi cambio idea posso passare tranquillamente ad un div senza dover modificare il foglio di stile. Usando la seconda sintassi, invece, sono costretto a rispettare l'elemento definito nel selettore.

12.7. Le proprietà dei caratteri (font)

Vediamo come cambiare la forma estetica dei caratteri tipografici utilizzati nelle pagine web.
• ***font-family***
• ***font-style***
• ***font-variant***
• ***font-weight***
• ***font-size***
• ***font***
font:

font: <size> [/ <line-height>] | <family> | [<style> | <variant> | <weight>]
es: font: 12px/18px bolder italic arial;
Con font è possibile indicare in forma compatta tutte le proprietà dei caratteri con le stesse regole che trovate successivamente.
font-family:
font-family: <nome carattere> | <font generico>;
Es: font-family: Verdana, Arial, sans-serif;
Utilizzando font-family riusciremo a dare un aspetto ai caratteri utilizzati nella pagina. I browser mostreranno il primo carattere, da sinistra, tra quelli utilizzati dal sistema operativo in uso. Nell’esempio precedente stamperà i caratteri in verdana, se non installato, arial altrimenti utilizzerà il font di default per il gruppo sans-serif. Ricordate quindi di provare tutti i caratteri aggiungendone uno la volta a sinistra per controllare l’aspetto estetico avendo cura di indicare una famiglia generica alla fine.
Se il font ha un nome composto da due o più termini, ad es. Times new roman, occorre metterlo tra ".
Es: font-family: "Times new Roman", serif;
font-size:
font-size: <dimensione> | xx-small | x-small | small | medium | large | x-large | xx-large | larger | smaller
dove dimensione può essere una lunghezza, oppure una percentuale.
font-style:
font-style: normal | italic | oblique;
Es: font-style: italic;
Serve a dare lo stile ai caratteri:. Sostituisce il tag html <i>.
font-variant:
font-variant: normal | small-caps;
Tale soluzione serve a dare l’effetto *"maiuscoletto"* al testo: le lettere maiuscole restano tali, quelle minuscole son riprodotte in maiuscolo ma in un corpo più piccolo.
font-weight:
font-weight: normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900;
Es: font-weight: 700;
Con tale proprietà facciamo stampare i caratteri con peso diverso. In effetti i browser attuali distinguono tra il peso standard e il bold. Pertanto la scelta si riduce a scegliere tra normal e bold, tenendo in mente che bolder e lighter, attributi relativi possono aumentare o diminuire il peso del carattere rispetto all’elemento contenitore. Sostituisce il tag html <b>.

12.8. Proprietà dei colori

Colore e background
Il colore è applicato alle due proprietà: color e background, che definiscono il colore del testo e lo sfondo.
color:
color: <valore>;
background:
background: <color> | <url> | <repeat> | <position> | <scroll>
Es: background: #F00 url(’immaginedisfondo’) fixed no-repeat center left fixed;
<color> può assumere un valore come su indicato, può essere transparent per indicare uno sfondo trasparente o none per non dare nessuno sfondo.
<url> è il percorso dell’immagine di sfondo, da usare nella forma url (<valore>), dove valore può esser inserito tra ‘, ", o senza, e può esser un percorso assoluto o **relativo alla**

posizione del css.
<repeat> può esser uno dei seguenti valori: **repeat**, opzione di default che fa ripetere l’immagine in orizzontale e verticale, **no-repeat**: nessuna ripetizione; **repeat-x, repeaty:** permettono la ripetizione lungo l’asse orizzontale (x) o verticale (y).
<position> indica la posizione dell’immagine di sfondo. può assumere valori numerici, percentuali o uno dei seguenti: **top, center, bottom** per il posizionamento verticale, **left, center, right** per quello orizzontale.
<scroll> può assumere i valori **scroll** (opz. di default) e **fixed** per indicare se l’immagine debba seguire o no lo scrolling della pagina.
Le stesse proprietà possono esser indicate singolarmente con:
• background-color: <color>;
• background-image: <url>;
• background-position: <position>;
• background-repeat: <repeat>;
• background-attachment: <scroll>;

12.9. Proprietà dei testi

I vantaggi dell’uso dei CSS viene resa evidente dalle proprietà dei testi, proprietà che permettono il controllo tipografico accurato, semplice per tutti ma allo stesso tempo potente. Con i css infatti è possibile controllare qualunque aspetto tipografico permettendo una resa avanzata dei testi usando le seguenti proprietà:
text-align:
text-align: left | center | right | justify
Esempi:
{text-align:justify;}
Permette di allineare il testo rispettivamente a sinistra, centrato, a destra e giustificato.
text-decoration:
text-decoration: none | underline | overline | line-through | blink
Esempi:
a {text-decoration:underline;}
a:visited {text-decoration:line-through;}
a:hover {text-decoration:none;}
per stampare il testo: senza nessuna decorazione, sottolineato, sopralineato, barrato e lampeggiante. L’uso di blink è da evitare poiché potrebbe creare problemi fisici ad alcuni e perché, obiettivamente, è fastidioso non poco.
text-transform:
text-transform: none | uppercase | lowercase | capitalize
permette di trasformare il testo in: tutto maiuscolo (uppercase), tutto minuscolo (lowercase), ogni prima lettera in maiuscolo (capitalize) o lasciare il testo formattato come da xHTML.
line-height:
line-height: normal | <valore> | <valore percentuale>
Esempi:
{line-height: 20px;}
{line-height: 200%;}
per impostare l’interlinea del testo.
word-spacing:
word-spacing: normal | <lunghezza>
permette di aumentare o diminuire lo spazio tra le parole. Di default ha valore 0.
Impostando un valore positivo tale lunghezza si aggiunge alla distanza solita,

impostandone uno negativo si dimuisce. **Attenzione**:
{word-spacing:10px;}
imposta una distanza tra le parole di 10px PIU’ il valore standard, non 10px in assoluto.
letter-spacing:
letter-spacing: normal | <lunghezza>
Come sopra. Di default ha valore 0 che sta ad indicare lo spazio tra le lettere standard. Anche in questo caso la lunghezza indicata va a sottrarsi o ad aggiungersi al valore di default.
text-indent:
text-indent: <lunghezza> | <percentuale>
Permette di stabilire il rientro del capoverso di un paragrafo o di un qualunque blocco di testo.
vertical-align:
vertical-align: baseline | bottom | middle | sub | super | text-bottom | text-top | top | <valore> | <percentuale>
Inziamo subito a dire cosa NON è: non corrisponde al valign delle tabelle **SE non usato nelle tabelle**. Non allinea al centro di un blocco un’immagine, tanto per esser chiari, ma indica l’allineamento verticale di un elemento online rispetto ai contigui. Per esser chiari, fa ciò che si fa con gli apici o pedici.
I possibili valori accettati sono:
• **baseline** (allinea l’elemento alla linea base dell’elemento contiguo)
• **bottom** (l’elemento viene allineato con il più basso degli elementi della linea)
• **middle** (l’elemento viene piazzato al centro dell’elemento contenitore)
• **sub** (pedice)
• **super** (apice)
• **text-bottom** (l’elemento viene allineato con la parte bassa della linea di testo)
• **text-top** (l’elemento viene allineato con la parte alta della linea di testo)
• **top** (l’elemento viene allineato con il più alto degli elementi della linea)
• ***valore*** (il tal caso viene allineato secondo il valore indicato. accetta anche valori negativi)
• ***percentuale*** (allinea l’elemento di un x% rispetto al valore del line-height dichiarato.)
Esempi:
{vertical-align:middle;}
white-space:
{white-space: normal | pre | nowrap}
Se nel testo xhtml si inseriscono più spazi successivi, o anche ritorni a capo, questi saranno ignorati e verrà mostrato a schermo un solo spazio. Con tale proprietà si modifica tale comporamento. Con **normal** due o più spazi o ritorni capo vengono mostrati in un unico carattere di spazio, **no-wrap** fa sì che il ritorno a capo del testo si ripercuota nella pagina mostrata, mentre più spazi successivi vengono stampati come un unico spazio, con **pre** si imposta il ritorno a capo e gli spazi così come scritti nel codice.

12.10. Proprietà dei bordi

I bordi possono esser definiti da tre proprietà:
• border-width: <dimensione>;
• border-color: <colore>;
• border-style: <stile>;
che vengono applicati a tutti i bordi del box.
<Dimensione> può essere una lunghezza come definitita in questo articolo oppure i

valori
• **thin** (sottile)
• **medium** (medio)
• **thick** (spesso)
<Stile> può essere:
• **none** (nessuno)
• **solid** (continuo)
• **double** (doppio)
• **dashed** (tratteggiato)
• **dotted** (punteggiato)
• **inset** (incassato)
• **outset** (in rilievo)
• **groove** (scanalato [in basso])
• **ridge** (scanalato [in alto])
<Colore> infine può essere qualsiasi colore

12.11. Tabelle

I css permettono design tableless, cioè senza tabelle, che ritornano finalmente ad assumere il valore per le quali son state progettate: mostrare dati tabellari, statistici. Anche per le tabelle esistono specifiche proprietà CSS:
table-layout:
table-layout: auto | fixed
che permette alle tabelle, alle righe o alle colonne delle stesse, di adattarsi al contenuto o di restare fisse in base, in altezza o larghezza a ciò che si è indicato.
empty-cells:
empty-cells: show | hide
per scegliere se mostrare o meno le celle vuote.
border-collapse:
border-collapse: collapse | separate
indica se far *collassare* i bordi in un unico bordo o separarli in più livelli.
border-spacing:
border-spacing: <lunghezza> <lunghezza>
da usare ovviamente con *border-collapse:separate* per indicare lo spazio tra i bordi. Si possono specificare due valori: 1 per la spaziatura destra e sinistra ed il secondo per quella in alto e basso; se si indica un solo valore vale per tutti e 4 i bordi.
caption-side:
caption-size: top | right | bottom | left
per indicare in che lato mostrare il valore di caption della tabella.