prof. Ivaldi Giuliano

Sommario

RICORDARSI DI APRIRE E CHIUDERE OGNI TAG !

Tag fondamentali

Tag in inglese vuol dire "etichetta"

<!doctype html>	identificazione documento HTML5
<html> </html>	inizio e fine documento html
<head> </head>	inizio e fine intestazione
<title> </title>	inizio e fine titolo posto sulla “barra del titolo” del browser (es, Mozilla Firefox, Google Chrome, Internet Explorer) utile per rendersi “visibile” ai motori di ricerca ed è quindi da inserire in ogni documento HTML che si crea, dentro il tag <head>
<body> </body>	inizio e fine corpo del sito
bgcolor="#cod colore"	attributo di <body>, colora lo sfondo del sito con "cod colore" (deprecato in HTML5) (es. <body bgcolor="blue"> , colora lo sfondo della pagina di blu)
background=”percorso/nome immagine”	mette ”nome immagine” come sfondo (deprecato in HTML5) (es. <body background=”images/immagine.jpg”> , mette ”immagine.jpg” come sfondo)
text=”#cod colore”	colora tutto il testo del sito con “cod colore” (es. <body text=”yellow”> , colora il testo del corpo della pagina di giallo)
<!-- commento -->	codice per inserire un commento

Formattazione del testo

** **	inizio e fine intestazione livello 1 (si può arrivare fino al livello 6)
** **	inizio e fine *paragrafo* saltando una riga
** **	inizio *line break* (inserisce un "a capo", non ha tag di chiusura)
** **	inizio e fine grassetto (bold)
** **	inizio e fine corsivo (italic)
****	inizio e fine apice
****	inizio e fine pedice
<hr>	disegna una riga ed inserisce un’interruzione di pagina
color=”#cod colore”	colorazione (es, <hr color=”blue”> , inserisce un filetto orizzontale e lo colora di blu)

Oss :

- “/” indica il termine di un tag
- le righe vuote in blocco note non vengono visualizzate dal browser
- in HTML5 è indifferente scrivere in maiuscolo o minuscolo, ma è consigliato il minuscolo
- per aggiungere caratteri speciali si usa il formato : &#cod-ASCII (es, &#65 = "A",   = spazio), oppure per gestire le lettere accentate aggiungere nell'intestazione, tag <head>, il tag <meta charset>
- l'attributo **"align"** (allineamento), anche se deprecato in HTML5, può però tornare utile, soprattutto all'inizio, quando non si conoscono ancora i fogli di stile (CSS): *align* è abbastanza generico e si può utilizzare con buona parte dei tag (es, , <hr>, , <td>), basta aggiungerlo prima di chiudere il tag e può assumere i valori *"left", "center" e "right"* (es,).

CODICE COLORI						
I colori si possono inserire tramite il loro codice o il loro nome in inglese						
White #FFFFFF	Black #000000	Red #FF0000	Green #00FF00	Blue #0000FF	Magenta #FF00FF	Cyan #00FFFF
Yellow #FFFF00	Aquamarine #70DB93	Dark Green #2F4F2F	Light Blue #C0D9D9	Light Green #32CD32	Orange #E47833	Pink #BC8F8F
Summer Sky #236B8E	Turquoise #ADEAEA	Violet #4F2F4F	Thistle #D8BFD8	Goldenrod #DBDB70	Maroon #8E236B	Lightest Grey #DDDDDD
Lighter Grey #A8A8A8	Medium Grey #545454	Darker Grey #454545	Darkest Grey #333333			

Esempio, creazione di una prima pagina web, creare con un editor di testo (es. Blocco Note) un file e rinominarlo cambiando estensione da “.txt” in “.html”, all’interno del file scrivere il seguente codice HTML

```
<!doctype html>
<html>
        <head>                                  <!- -sezione di intestazione - ->
                <meta charset>
                <title> Prova </title>
        </head>
        <body bgcolor="yellow" text="blue">     <!-- il corpo della pagina sarà colorato di giallo e il testo di blu -->
                <h1 align="center"> Questo è un Titolo </h1>
                <hr color="blue">               <!-- disegna una riga colorata di blu -->
                <br> <br>                       <!-- va a capo due volte -->
                <p> Questo è un paragrafo </p>
                <p> <b> Questo è un paragrafo in grassetto </b> </p>
        </body>
</html>
```

Dopo aver salvato, cliccare due volte sull’icona del file e aprirlo, il file sarà automaticamente aperto con il browser e si potrà vedere la pagina web corrispondente al codice HTML che si è digitato

Le Ancore (Link o Collegamenti ipertestuali)

** nome link **
crea un link di nome “Nome Link” che punta all’indirizzo contenuto nel tag

** indirizzo_e-mail **
crea un link all’indirizzo di posta contenuto nel tag (nel momento in cui ci si cliccherà sopra si aprirà una e-mail con l’indirizzo di destinazione già predisposto con quello indicato nel tag)

** nome link **
crea un link di nome “Nome Link” che punta alla pagina html contenuta nel tag

** nome link **
crea un link di nome “Nome Link” che punta al file audio contenuto nel tag

Oss : se il link è interno alla stessa pagina la sintassi sarà la seguente
** nome link **
e laddove si vorrà inserire l’ancora, che non verrà visualizzata, si digiterà
****

se nel tag <a href…> si aggiunge l’attributo -> **target=”nome_bersaglio”**
il “contenuto” a cui punta il link verrà caricato in “nome_bersaglio”.
I possibili valori di "nome_bersaglio" sono:
_self, apre il collegamento nello stesso frame o finestra
_blank, apre il collegamento in una nuova scheda
_parent, apre il collegamento nella finestra genitore della finestra
_top, apre il collegamento nel frame o finestra più importante
framename

Es,
 Google motore di ricera
crea un link di nome “Google motore di ricerca” che punta all’indirizzo www.google.it

 mariorossi@virgilio.it
crea un link all’indirizzo di posta del signor Rossi (nel momento in cui ci si cliccherà sopra, si aprirà una e-mail con l’indirizzo di destinazione del signor Rossi)

 Aggiornamenti
cliccando sul link Aggiornamenti si aprirà una nuova scheda che punterà alla pagina news.html

 Volare , crea un link di nome “Volare” che punta alla canzone volare

```
<a href=”#sezione_lettura” > Nome link </a>
  …codice HTML…
<a name=”sezione_lettura”></a>
```

Oss: prima di chiudere il sito aggiungere data,ora dell’ultimo aggiornamento ed il nome e cognome del proprietario del sito con indirizzo e-mail eventualmente sottoforma di link.

Elenchi puntati o numerati

** ** (unordered list) inizio e fine elenco non ordinato
proprietà (o attributo) "type" consente di specificare quale punto elenco si intende utilizzare.
Valori type = "disc" (puntino-default), "square" (quadratino), "circle" (cerchio vuoto)

** ** (ordered list) inizio e fine elenco ordinato
Valori type = "1" (numeri decimali - default), "A" (lettere maiuscole), "a" (lettere minuscole), "I" (numeri romani maiuscoli), “i” (numeri romani minuscoli)

** ** (list item) identifica una voce dell’elenco

<dl> </dl> (definition list) inizio e fine elenco tipo glossario
<dt> </dt> (definition tag) consente di assegnare un titolo al paragrafo
<dd> </dd> (definition descriptor) consente di assegnare una descrizione che verrà visualizzata indentata a destra rispetto al titolo (dt)

Es,

```
<p>  Componenti fondamentali di un computer </p>
<ol type=“1”>
<li> CPU </li>
<ul type="disc">
        <li> CU </li>
        <li> ALU </li>
</ul>
<li> Memoria centrale </li>
<ul type="disc">
        <li> RAM </li>
        <li> ROM </li>
</ul>
<li> Dispositivi di Input/Output </li>
        <ul type="disc">
                <li>Dispositivi di Input </li>
                <li>Dispositivi di Output </li>
                <li>Dispositivi di Input/Output </li>
        <ul type="circle">
          <li>Memorie di massa (Hard Disk, Floppy Disk, CD, DVD) </li>
        </ul>
  </ul>
</ol>

<p> Glossario </p>
<dl>
        <dt> CPU </dt>
        <dd> Central Processing Unit </dd>
        <dt> CU </dt>
        <dd> Control Unit </dd>
        <dt> ALU </dt>
        <dd> Arithmetic Logic Unit </dd>
        <!-- Commento : il glossario continua… -->
</dl>
```

Le immagini

<img>

src="percorso\nome_immagine" indica quale immagine dovrà essere caricata
alt=”titolo immagine” visualizzerà titolo immagine durante il caricamento, utile anche per indicizzare l’immagine e renderla “visibile” ai motori di ricerca
width=”n° pixel” larghezza immagine
height=”n° pixel” altezza immagine

Oss: le proprietà dell’immagine si possono ottenere facendo clic destro sull’immagine

Es,

<img src="images\immagine.jpg" alt=”titolo dell’immagine”>
inserisce l’immagine ”immagine.jpg” presente nella cartella ”images”, introducendo durante la fase di caricamento dell’immagine una breve descrizione.

Immagini mappate

Sono dei link rappresentati da immagini grafiche e sono un valido ausilio ai semplici link testuali, utili comunque per rendersi “visibili” ai motori di ricerca.
Per realizzarle, seguire i seguenti passaggi :

- costruire con un programma di grafica (es. paint) la barra delle immagini-link ed annotarsi le coordinate in pixel, fornite dal programma stesso, degli angoli in alto a sinistra ed in basso a destra di ogni immagine inserita nella barra appena realizzata, che deve iniziare dalle coordinate 0,0
- aggiungere nella pagina/file html in cui si vuole creare la barra-link il seguente codice html :

```
<map name="barra" title="titolo replicato su tutte le aree sensibili della mappa">
    <area shape=”rect” coords="49,15,179,130"   href="italiano.html">
    <area shape=”rect” coords="211,15,345,130" href="storia.html">
    <area shape=”rect” coords="380,15,510,130" href="matematica.html" title="titolo relativo alla singola area">
    <area shape=”rect” coords="544,15,676,130" href="informatica.html">
    …altre coordinate ed altri collegamenti…
</map>
```

laddove si vuole inserire la barra nella pagina inserire

```
<img src="images\link_mappati.jpg" width="722" height="120" usemap="#barra">
```

Oss :

- le coordinate di COORDS sono di esempio come i file html di HREF, il nome e percorso della barra “images\link_mappati.jpg” e tutti i valori degli altri attributi.
- il nome indicato da USEMAP deve coincidere con quello dato alla barra in <MAP NAME> a cui si aggiunge davanti il “#”
- le altre opzioni di shape oltre rect sono circle (xcentro, ycentro, raggio) e poly (servono 4 coordinate separate da virgola)
- inserendo, prima di coords, l’opzione ALT=”nome” si dà un nome all’immagine che verrà visualizzato passandoci sopra con il mouse

Tag

E' un contenitore molto simile a <p>, però mentre <p> è utilizzato normalmente per i testi, è un contenitore generico in grado di contenere qualsiasi oggetto (testo, immagini, video, etc...) ed è usato per suddividere una pagina html in più sezioni. Presenta il solo attributo "align". Deprecato in HTML5.0.

Caratteristiche:

- se non si specifica la posizione o l'allineamento i tag DIV vengono posizionati uno sotto l'altro esattamente come i paragrafi (default: position = static).
- se non si specificano le dimensioni i tag DIV occupano come larghezza l'intera riga e coma altezza lo spazio necessario a contenere quanto inserito all'interno, esattamente come per i paragrafi. Se dentro il tag DIV c'è un'immagine, il tag occupa un'altezza pari a quella dell'immagine.
- se si specificano width e height, questi rappresentano non la reale dimensione del box, ma l'area utile interna. A questi valori occorre ancora sommare l'eventuale dimensione del padding e del bordo.
- sui tag DIV si può realizzare il cosiddetto posizionamento assoluto in un certo punto della pagina
- i tag DIV possono essere visualizzati uno a fianco all'altro impostando la proprietà, float = left; (o right)

Es,

Paragraph 1
Paragraph 2
Paragraph 3
Paragraph 4

```
<div style="border:solid;"  align="right">
        <p>Paragraph 1</p>
        <p>Paragraph 2</p>
        <p>Paragraph 3</p>
        <p>Paragraph 4</p>
</div>
```

(Oss: principali tag deprecati in HTML5: font, center, u (underline, CSS equivalente text-decoration: underline), s (barrato, CSS equivalente text:decoration: line-through), strike(uguale a s), big, small, pre, tt, div, frame, frameset, noframe, acronym (sostittuito da abbr), applet (sostituito da object). Principali attributi deprecati in HTML5: align, valign, background, bgcolor, border, cellpadding, cellspacing. Mantenuti i tag b ed i, ma da utilizzare solo come ultima risorsa.)

Tag <audio>

Permette il caricamento di un file audio (in formato mp3, wav o ogg) direttamente dal codice html della pagina.

es,

```
<audio src="myFile.mp3" autoplay="autoplay" controls="controls"> Il tuo browser non supporta l'audio gestito con HTML5 </audio>
```

Attributi

src	è il riferimento al contenuto multimediale da riprodurre
controls="controls"	è un boolean che consente di visualizzare i controlli standard offerti dal browser per la classica gestione di play, pause, seek e volume, di cui ogni contenuto multimediale normalmente necessita
autoplay="autoplay"	l'audio partirà automaticamente al caricamento della pagina
loop="loop"	il brano verrà ripetuto automaticamente dopo ogni terminazione
preload="none"/"auto"	il valore "auto" forza il browser a scaricare l'intero file prima di iniziare la riproduzione. In caso di impostazione dell'attributo "autoplay", l'attributo "preload" viene ignorato

All'interno del tag si può indicare come testo il markup da visualizzare nel caso il browser non dovesse supportare HTML5 (ormai comunque supportato da tutti i browser)

Il tag <source>

il tag <audio> supporta anche la possibilità di gestire più di un sorgente, in tal caso si omette l'attributo src e si annidano più tag **<source>** all'interno del tag <audio>. Il tag <source> consente di definire risorse alternative per gli elementi multimediali in base alle capacità di riproduzione del browser. Il browser cercherà di scaricare il primo file audio, se questo per qualche motivo dovesse fallire o fosse in un formato non supportato dal browser, automaticamente passerà a scaricare il file audio successivo.
L'attributo **type** indica l'Internet media type (altrimenti noto come MIME type) del documento.

es,

```
<audio controls="controls">
        <source src="primo_file.ogg" type="audio/ogg">
        <source src="secondo_file.mp3" type="audio/mp3">
        Il tuo browser non supporta l'audio gestito con HTML5
</audio>
```

Formati supportati dai Browser			
Browser	**Ogg Vorbis**	**MP3**	**Wav**
Firefox 3.6+	v		v
Safari 5+		v	v
Chrome 6	v	v	
Opera 10.5+	v		v
Internet Explorer 9		v	v

Tag <video>

Simile al tag <audio>. Permette il caricamento di un file video (in formato mp4, webM o ogg) direttamente dal codice html della pagina.

es,

```
<video src="myFile.ogg" autoplay="autoplay" controls="controls"> Il tuo browser non supporta i video gestiti con HTML5 </video>
```

```
<video width="320" height="240" autoplay="autoplay" controls="controls">
        <source src="movie.mp4" type="video/mp4">
        <source src="movie.ogg" type="video/ogg">
        Il tuo browser non supporta i video gestiti con HTML5
</video>
```

Attributi

src, controls, autoplay, loop e preload sono gli stessi del tag audio

width e **height**	dimensioni di visualizzazione del player. Questi attributi sono vivamente consigliati, altrimenti il browser, non potendo sapere le dimensioni del filmato, andrà ad occupare tutto lo spazio necessario modificando il layout della pagina.
muted="muted"	soppressione dell'audio
poster="url"	indica un'immagine da visualizzare mentre il video viene scaricato o fino a quando l'utente non preme il tasto di avvio della riproduzione

All'interno del tag si può indicare come testo il markup da visualizzare nel caso il browser non dovesse supportare HTML5 (ormai comunque supportato da tutti i browser).

Il tag <source> è lo stesso del tag audio

Formati supportati dai Browser			
Browser	**MP4**	**WebM**	**Ogg**
Firefox		v	v
Safari	v		
Chrome	v	v	v
Opera		v	v
Internet Explorer	v		

Le tabelle

<table> </table> inizio e fine gestione tabella
cols="numero" indica il numero di colonne della tabella
width="numero" indica la larghezza complessiva della tabella (esprimibile anche in % rispetto alla pagina)
height=”numero” indica l’altezza a cui verrà posta la tabella
colspan=”numero” permette di creare celle che occupano più colonne
rowspan=”numero” permette di creare celle che occupano più righe
border=”numero” definisce le dimensioni del bordo della tabella, “0” per rendere la tabella invisibile (deprecato in HTML5)
<tr> </tr> formattazione righe (Table Row).
Una riga può contenere celle di intestazione o di dati
<th> </th> cella di intestazione con testo grassetto e centrato
<td> </td> formattazione celle di dati (Table Data), con testo normale allineato a sinistra

Proprietà disponibili	<table>	<tr>	<th> e <td>
width	v		v
height	v	v	v
colspan			v
rowspan		v	

Es,

```
<table>
    <tr>
        <td>Colonna 1</td>
        <td>Colonna 2</td>
    </tr>
    <tr>
        <td>Dato 1,1</td>
        <td>Dato 1,2</td>
    </tr>
    <tr>
        <td>Dato 2,1</td>
        <td>Dato 2,2</td>
    </tr>
    <tr>
        <td>Dato 3,1</td>
        <td>Dato 3,2</td>
    </tr>
</table>
```

Colonna 1	Colonna 2
Dato 1,1	Dato 1,2
Dato 2,1	Dato 2,2
Dato 3,1	Dato 3,2

Creazione di una barra di navigazione

```
<table align="center" border="0" cols="5" width="60%">
    <tr align="center">
        <td> <a href="index.html" target="_blank"> Homepage </a> </td>
        <td width=1%> | </td>
        <td> <a href="pagina2.html" target="_blank"> Pagina 2 </a> </td>
        <td width=1%> | </td>
        <td> <a href="pagina3.html" target="_blank"> Pagina 3 </a> </td>
    </tr>
</table>
```

il risultato del codice dell'esempio sarà il seguente

Homepage | Pagina 2 | Pagina 3

```
<table border="4" cols="3" width="80%" align="center">
	<tr>
		<th width=25% rowspan=3>Promossi</th>
		<td align=center colspan=2>Voti</td>
	</tr>
	<tr>
		<td>Rossi</td>
		<td>29</td>
	</tr>
	<tr>
		<td>Verdi</td>
		<td>28</td>
	</tr>
	<tr>
		<th width=25% rowspan=2>Respinti</th>
		<td>Bianchi</td>
		<td>25</td>
	</tr>
	<tr>
		<td>Gialli</td>
		<td>21</td>
	</tr>
```

Promossi	Voti	
	Rossi	29
	Verdi	28
Respinti	Bianchi	25
	Gialli	21

Negli esempi sono presenti alcune proprietà deprecate in HTML5 come border, align

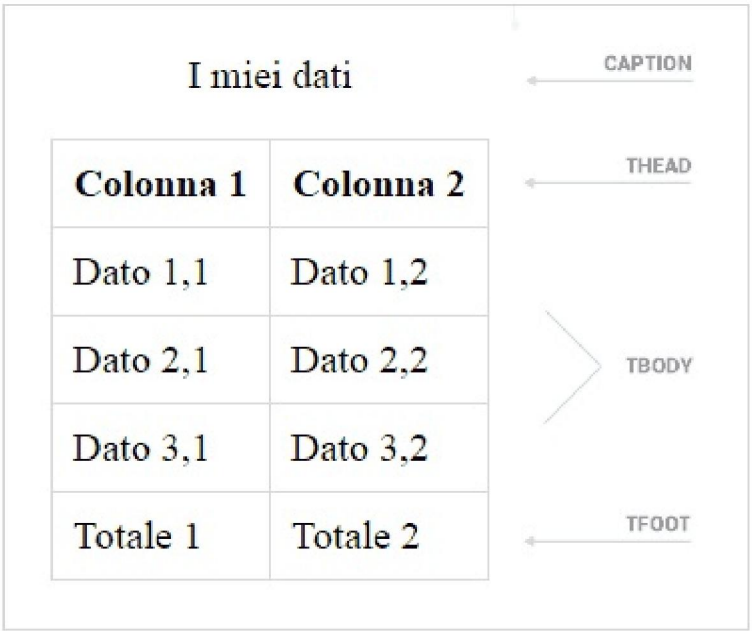
Un template più completo

Per scrivere una tabella che fornisca una rappresentazione più chiara dei dati introduciamo un template leggermente più ricco con i tag...

<caption> </caption> introduce una didascalia dei dati
<thead> </thead> serve per raggruppare le righe che rappresentano l'intestazione della tabella.
<tbody> </tbody> raggruppa le righe che contengono il corpo della tabella, spesso con i dati veri e propri.
<tfoot> </tfoot> racchiude le righe che utilizziamo come footer della tabella, in cui si possono inserire dei dati di riepilogo, somme, medie, etc.

Es,

```
<table>
	<caption>
		<p>I miei dati</p>
	</caption>
	 <thead>
		<tr>
			<th>Colonna 1</th>
			<th>Colonna 2</th>
		</tr>
	</thead>
	 <tbody>
		<tr>
			<td>Dato 1,1</td>
			<td>Dato 1,2</td>
		</tr>
		<tr>
			<td>Dato 2,1</td>
			<td>Dato 2,2</td>
		</tr>
		<tr>
			<td>Dato 3,1</td>
			<td>Dato 3,2</td>
		</tr>
	</tbody>
	<tfoot>
		<tr>
			<td>Totale 1</td>
			<td>Totale 2</td>
		</tr>
	</tfoot>
</table>
```



I moduli (Form)

<form> </form> inizio e fine gestione form

action=”nome_pagina” indica al browser a quale indirizzo o pagina HTML inviare i dati del form, in corrispondenza del click sul pulsante di submit. I dati vengono inviati alla nuova pagina nel formato "Name = Value". Il contenuto dei controlli che non hanno il "name" non viene inviato, come anche per i controlli "disabled"

target=”nome_bersaglio” indica dove verrà caricata la pagina a cui punta l'attributo action (vedere le Ancore)

method=”get” o "post" specifica il metodo per accedere all'URL dichiarato in action, due le possibilità: get o post. Il metodo get: preferito quando sono in gioco pochi parametri, può utilizzare solo parametri testuali. Metodo post: è il più usato, è utilizzato per inviare grosse quantità di dati, può essere inviato qualsiasi formato di informazioni, possono così essere scambiati tra Client e Server anche files con immagini, sonoro, video, eccetera, e permette di "nascondere" i dati inviati

<!-- Tipi di controlli -->

<input> inserisce degli oggetti (caselle di testo,bottoni,…) nel form, non ha tag di chiusura

type attributo che specifica il tipo di controllo che si intende inserire

- **type=”text”** inserisce una casella di testo (textbox)
- **type=”password”** casella di testo per le password (con asterischi)
- **type="hidden"** casella di testo nascosta (utile per memorizzare informazioni senza visualizzarle
- **type="file"** casella di testo in cui si richiede all'utente di scrivere il nome di un file. Invece di scriverlo direttamente l'utente lo può selezionare utilizzando il tipico "Sfoglia" di Windows
- **type=”radio”** inserisce un bottone radio (casella con pallino)
- **type=”checkbox”** inserisce un checkbox (casella da spuntare)
- **type=”submit”** inserisce un bottone che in corrispondenza del click del mouse invia al server il contenuto della form. Richiama la pagina indicata in action
- **type=”reset”** inserisce un bottone di azzeramento di tutti i campi del modulo

name=”nome_controllo” nome del controllo

value="testo" di tipo testo, rappresenta il contenuto del controllo (valore trasmesso in corrispondenza del submit)

disabled consente di abilitare/disabilitare un controllo

tabindex=numero ordine di tabulazione del controllo

type=”text” o "password" o "hidden" o "file"

- **maxlenght=numero** specifica il numero massimo di caratteri digitabili all'interno del textbox
- **size=”num_caratteri”** dimensione larghezza casella in numero caratteri. Raggiunto "num_caratteri" il testo comincia a scorrere verso sinistra
- **readonly** consente di utilizzare la casella di testo in sola lettura

es, Password <input type="password" name="password">

type="radio" o "checkbox"

- **checked** se specificato, il bottone radio o il checkbox saranno selezionati di default

es,
```
<p> Avete un computer?
    <input type="radio" name="optComputer" value="si" checked> Si
    <input type="radio" name="optComputer" value="no"> No
</p>
Verrà restituito "optComputer=Si" oppure "optComputer=No"

<input type="checkbox" name="chkMaggiorenne" value="si"> Maggiorenne
Se l'opzione è selezionata verrà restituita la stringa "chkMaggiorenne=si", altrimenti stringa vuota
```

type=”submit” o "reset"

- **value="testo"** in questo caso value rappresenta la caption (testo contenuto) del pulsante

<select name=”nome”> gestione menù a tendina (listbox o combobox)

```
<option value=”1°nome”> 1°testo da inserire </option>
<option value=”2°nome”> 2°testo da inserire </option>
……
</select>
```

<textarea> </textarea> crea una casella di testo a righe multiple (es, commenti)

name=”nome” nome casella di testo

rows=”n° righe” definisce il numero di righe

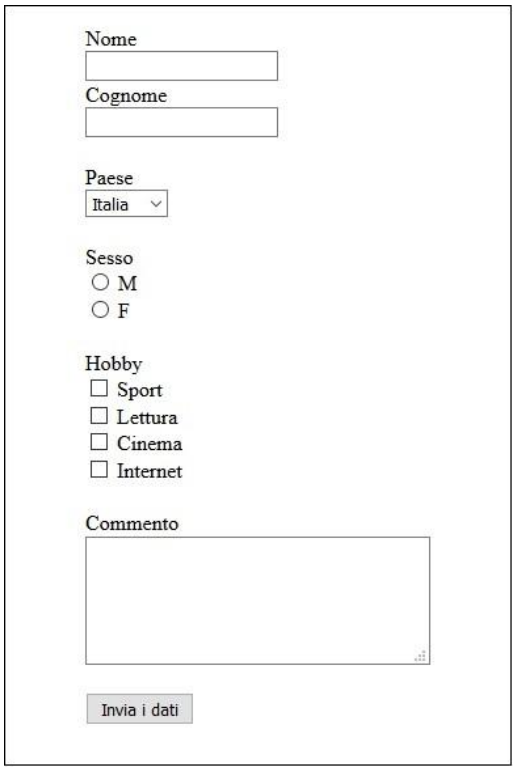
cols=”n° colonne” definisce il numero di colonne

wrap="on" o "off" se viene impostato a "off" i caratteri possono estendersi oltre il valore di "cols" con comparsa automatica della barra di scorrimento orizzontale(fino a quando non viene premuto INVIO)

Es,

Vediamo un esempio di un ipotetico form HTML per la raccolta di dati anagrafici:

```
<form name="modulo" method="post" action="esegui.php">

        <!-- CASELLE DI TESTO -->
        Nome<br>
        <input type="text" name="txt_nome"><br>
        Cognome<br>
        <input type="text" name="txt_cognome"><br><br>

        <!-- SELECTBOX -->
        Paese<br>
        <select name="paese">
                <option value="I">Italia</option>
                <option value="E">Estero</option>
        </select><br><br>

        <!-- RADIO -->
        Sesso<br>
        <input type="radio" name="sesso" value="M"> M<br>
        <input type="radio" name="sesso" value="F"> F<br>

        <!-- CHECKBOX -->
        Hobby<br>
        <input type="checkbox" name="hobby" value="S"> Sport<br>
        <input type="checkbox" name="hobby" value="L"> Lettura<br>
        <input type="checkbox" name="hobby" value="C"> Cinema<br>
        <input type="checkbox" name="hobby" value="I"> Internet<br>

        <!-- TEXTAREA -->
        Commento<br>
        <textarea name="commento" rows="5" cols="30"></textarea>
        <br><br>

        <!-- SUBMIT -->
        <input type="submit" name="invia" value="Invia i dati">
</form>
```



I META TAG

I META Tag servono per memorizzare nella sezione di head, informazioni relative al contenuto della pagina, come ad esempio le informazioni per l'indicizzazione dei motori di ricerca (in sostanza per rendersi visibili ai motori di ricerca). Non hanno in genere tag di chiusura.

Il metatag "meta"

Il metatag "meta" ha due attributi principali

name	utilizzato per memorizzare contenuti descrittivi della pagina
http-equiv	utilizzato per impostare ulteriori caratteristiche della pagina

In entrambi i casi sono seguiti dall'attributo "**content**" che memorizza il contenuto del meta tag

<meta name="tipo di attributo" content="descrizione">

name="keywords"	elenco di parole chiave del sito o della pagina utilizzabile dai motori di ricerca per aggiornare i loro cataloghi. E' raccomandato il minuscolo. **<meta name = ”keywords” content=”inserire,qui,le,parole,chiave,separate,da,virgole">** es, <meta name = ”keywords” content=”continenti, europa, america, asia, africa, oceania">
name="description"	descrizione sintetica del sito che verrà visualizzata dal motore in caso di esito positivo della ricerca **<meta name = ”description” content = “descrizione sintetica max 200 caratteri”>** es, <meta name = ”description” content = “sito di geografia, che tratta dei cinque continenti”>
name="author"	nome di colui che ha creato il sito **<meta name="author" content="nome autore">** es, <meta name="author" content="Rossi Mario">
name="copyright"	informazioni sul copyright **<meta name="copyright" content="informazioni sul copyright">** es, <meta name="copyright" content="Copyright (c) 2050 Rossi Mario, all rights reserved.">

<meta http-equiv="tipo di attributo" content="descrizione">

http-equiv="content-type"	indica il formato della pagina **<meta http-equiv="content-type" content="text/html">**
http-equiv="content-language"	linguaggio utilizzato nel contenuto della pagina **<meta http-equiv="content-language" content="it-IT">**
http-equiv="expires"	data e ora oltre la quale la pagina è da considerare obsoleta; in corrispondenza della scadenza i server proxy elimineranno la pagina dalla cache. Il valore 0 significa no-cache. **<meta http-equiv="expires" content="data ora">** es, <meta http-equiv="expires" content="Wed, 9 Nov 2011 12:00:00 GMT">
http-equiv="refresh"	provoca un refresh della pagina (come se si cliccasse sul pulsante Aggiorna) **<meta http-equiv="refresh" content="n°secondi per refresh;url=pagina per redirect">** es, <meta http-equiv="refresh" content="5"> provoca un refresh ogni 5 secondi <meta http-equiv="refresh" content="5; url=NuovaPagina.html"> provoca un refresh ogni 5 secondi, ed in corrispondenza del refresh viene eseguito un "redirect" alla NuovaPagina.html (in sostanza viene caricata "NuovaPagina.html")
http-equiv="pragma"	avvisa i client (proxy e browser) che la pagina non deve essere messa in cache **<meta http-equiv="pragma" content="no-cache">**
http-equiv="content-script-type"	linguaggio di scripting utilizzato nel documento **<meta http-equiv="content-script-type" content="text/javascript">**

N.B.: Un server **proxy** è un server (inteso come sistema informatico o applicazione) che funge da intermediario per le richieste da parte dei client alla ricerca di risorse su altri server. Un client si connette al server proxy, richiedendo qualche servizio (ad esempio un file, una pagina web o qualsiasi altra risorsa disponibile su un altro server), e quest'ultimo valuta ed esegue la richiesta in modo da semplificare e gestire la sua complessità.

Il metatag "link"

Consente di collegare la pagina corrente ad una risorsa esterna, utilizzato soprattutto per collegare la pagina con il relativo foglio di stile.
Due sono gli attributi obbligatori:

rel	indica il tipo di risorsa a cui si sta accedendo
href	indica la destinazione

Es,

```
<link rel="stylesheet" href="percorso file .css">
<link rel="stylesheet" href="stylesheets/default.css">
```

Oss: i metatag vengono ripetuti per tutte le pagine del sito, con parole chiave e descrizioni diverse.

HTML 5.0

```
<html>
  <head>
  </head>
  <body>
    <div id="header">
        --- Titolo e Testata ---
    </div>
    <div id="nav">
       --- Voci di Menu ---
    </div>
    <div id="section">
       <div class="article">
          --- Un Post ---
       </div>
    </div>
    <div id="aside">
       --- Altre Voci ---
    </div>
    <div id="footer">
       --- Piè di pagina ---
    </div>
  </body>
</html>
```

All'inizio le pagine web erano strutturate attraverso le tabelle, si dovevano costruire strutture complesse spezzandole all'interno di una griglia fatta da infiniti <tr> e <td>, con scarsa qualità e flessibilità del risultato. Poi venne introdotto il tag che permise un modello di costruzione del documento pensato per separare in modo chiaro i contenuti tipici di una pagina web: grazie alla combinazione tra questo elemento e i CSS nacquero pagine con codici HTML eleganti e leggibili come nell'esempio a fianco. All'interno di un costrutto come quello dell'esempio è tutto molto semplice da interpretare: basta seguire il flusso dei rientri di pagina facendosi guidare dai valori semantici attribuiti a id e class.
Passarono anni e il modello cominciò a mostrare le sue prime debolezze; in primo luogo ci si accorse che da un lato non vi era una regola collettiva e quello che per uno sviluppatore era 'body' per l'altro poteva benissimo essere 'corpo'; inoltre si realizzò che né il browser né tantomeno i motori di ricerca avrebbero mai potuto beneficiare di questa divisione semantica, proprio per colpa di quell'arbitrarietà che permetteva a milioni di siti web di essere organizzati in strutture simili ma sempre leggermente diverse tra loro e per questo non raggruppabili secondo schemi automatici. Emerse in questo modo uno dei più grandi problemi dell'HTML4: l'impossibilità di descrivere il significato delle informazioni di una pagina web in un formato interpretabile da altri software. L'HTML5 nacque per gestire ed incanalare tutte queste problematiche.

Il nuovo Content Model HTML5

Non più solo div

Possiamo adesso provare a confrontare l'esempio precedente con i nuovi elementi messi a disposizione dall'HTML5:

```
<html>
  <head>
  </head>
  <body>
    <div id="header">
        --- Titolo e Testata ---
    </div>
    <div id="nav">
       --- Voci di Menu ---
    </div>
    <div id="section">
      <div class="article">
         --- Un Post ---
      </div>
      <div class="article">
         --- Un altro Post ---
      </div>
    </div>
    <div id="aside">
       --- Altre Voci ---
    </div>
   <div id="footer">
       --- Piè di pagina ---
   </div>
  </body>
</html>
```

```
<!doctype html>
<html>
  <head>
  </head>
  <body>
    <header>
        --- Titolo e Testata ---
    </header>
    <nav>
        --- Voci di Menu ---
    </nav>
    <section>
       <article>
          --- Un Post ---
       </article>
       <article>
          --- Un altro Post ---
       </article>
    </section>
    <aside>
        --- Altre Voci ---
    </aside>
    <footer>
        --- Piè di pagina ---
    </footer>
  </body>
</html>
```

Header
Il tag **<header>** serve a rappresentare **un gruppo di ausili *introduttivi* o di navigazione**, relativamente a un'intera pagina, una sezione, un articolo. Tale definizione, seppure apparentemente vaga, contiene in sé i concetti chiave per comprendere appieno la funzione di questo tag:

1. L'elemento <header> è un contenitore per altri elementi.
2. L'elemento <header> non va confuso con quella che è la testata/intestazione principale di un documento (quella che oggi si definisce in genere con il tag <head>).
3. La natura e gli scopi dell'elemento <header> non dipendono dalla sua posizione nel documento, ma dai suoi contenuti (ausili alla navigazione o elementi *introduttivi*).
4. Il suo uso non è obbligatorio e in alcuni casi può risultare superfluo se non utilizzato in maniera appropriata.

Esempio,

```
<header>
        <h1>Questo è un titolo</h1>
        <h2>Questo è un sottotitolo</h2>
        ...resto del codice...
</header>
```

Section
Il tag **<section>**, secondo la definizione presente nella specifica HTML5, rappresenta una **sezione generica di un documento o applicazione**. L'elemento <section>, in questo contesto, individua un raggruppamento tematico di contenuti, ed in genere contiene un titolo introduttivo <h1>.
Vediamo quindi quali sono i punti fondamentali da ricordare riguardo il suo utilizzo:

1. L'elemento <section> **non deve** essere utilizzato in sostituzione del per impostare graficamente la pagina; inoltre è fortemente consigliato utilizzare i anche quando risultano più convenienti per gli script.
2. L'elemento <section> **non deve** essere preferito all'elemento <article> quando i contenuti possono essere ripubblicati anche su altri siti.
3. L'elemento <section> e l'elemento <article> non sono indipendenti ed esclusivi: possiamo avere sia un <article> all interno di un <section> che viceversa.

Esempo,

```
<article>
        <section>
                <h1>Titolo 1</h1>
                <p>Testo correlato al titolo 1.</p>
        </section>
        <section>
                <h1>Titolo 2</h1>
                <p>Testo correlato al titolo 2.</p>
        </section>
</article>
```

Article
Il tag **<article>** rappresenta una **sezione autonoma in un documento, pagina, applicazione o sito**; infatti è potenzialmente ridistribuibile o riutilizzabile, e quindi ripubblicabile in parte o interamente in diverse pagine. Esso può identificare il post di un forum, un articolo di una rivista o di un giornale, l'articolo di un blog, un commento, un widget interattivo, o qualsiasi cosa che abbia un contenuto indipendente.
Prima di vedere un esempio concreto bisogna chiarire alcuni concetti riguardo il suo utilizzo:

1. quando gli elementi <article> sono nidificati, gli <article> interni rappresentano gli articoli che sono in linea di principio relativi al contenuto dell'<article> esterno. Ad esempio, un blog che accetta commenti dagli utenti potrebbe rappresentarli come <article> figli annidati all'interno dell'elemento padre <article>;
2. le informazioni relative all'autore dell'<article> non devono essere replicate all'interno degli elementi nidificati all'interno dello stesso;
3. l'elemento <time> con l'attributo pubdate può essere utilizzato per definire la data di pubblicazione dell'<article>;
4. l'elemento <section> e l'elemento <article> non sono indipendenti ed esclusivi: possiamo avere sia un <article> all interno di un <section> che viceversa.

Esempio,

```
<article>
        <header>
                <h1>Titolo articolo</h1>
                 <time pubdate datetime="2011-10-09T14:28-08:00"></time></p>
        </header>
        <p>Contenuto dell'articolo</p>
        <footer>
                <p>Informazioni riguardo l'autore</p>
        </footer>

</article>
```

Nav

Il tag **<nav>** è uno degli elementi introdotti nelle specifiche HTML5 di più facile comprensione. Infatti, rappresenta una sezione di una pagina che **contiene link (collegamenti) ad altre pagine o a parti interne dello stesso documento**; quindi, in breve, una sezione contenente **link di navigazione**. Può essere indipendente oppure contenuto in un altro elemento, come <header> ad esempio.

Per poter utilizzare correttamente l'elemento <nav> dobbiamo ricordare i seguenti punti:

1. solo sezioni che sono costituite da **grandi blocchi di navigazione** sono appropriati per l'elemento <nav>
2. i link a pie' di pagina e le breadcumb **non devono** essere inseriti in una sezione <nav>

   La parola breadcrumb significa, letteralmente, "bricciola di pane". I menu di questo tipo indicano il percorso gerarchico per giungere ad una data pagina all'interno di un sito web.
   Ad esempio: Home > Musica > Rock > Artisti, dove ogni parola è, ovviamente, un link.
   In questo modo si facilità la navigazione degli utenti ed il lavoro degli spider dei motori di ricerca. L'utilizzo dei menu breadcrumb è consigliatissimo.
3. l'elemento <nav> **non sostituisce** i link inseriti in elementi come gli <ul> o gli <ol> ma deve racchiuderli.

Esempio,

```
<nav>
    <ul>
        <li>Questo è un link</li>
        <li>Questo è un link</li>
        <li>Questo è un link</li>
        <li>Questo è un link</li>
        ...resto del codice...
    </ul>
</nav>
```

Aside

L'elemento **<aside>** rappresenta una sezione di una pagina costituita da **informazioni che sono marginalmente correlate al contenuto dell'elemento padre che la contiene**, e che potrebbero essere considerate distinte da quest'ultimo. Questo è ciò che viene indicato nelle specifiche HTML5, ma è facile immaginare l'utilità del tag <aside> semplicemente pensandolo come un **contenitore di approfondimento** in cui possiamo inserire gruppi di link, pubblicità, bookmark e così via.

Esempio,

```
<aside>
    <h1>Questi sono dei contenuti di approfondimento marginali rispetto al contenuto principale</h1>
    <nav>
        <h2>Link a pagine correlate al contenuto</h2>
        <ul>
            <li>Informazione correlata al contenuto</li>
            <li>Informazione correlata al contenuto</li>
            <li>Informazione correlata al contenuto</li>
        </ul>
    </nav>
    <section>
        <h2>Pubblicità</h2>
        …immagini pubblicitarie…
    </section>
</aside>
```

Footer

L'elemento **<footer>** deve contenere in genere **informazioni sulla sezione che lo contiene** come: i dati di chi ha scritto i contenuti, collegamenti ai documenti correlati, i dati di copyright, e così via…

Riguardo il suo apporto semantico ad una pagina web sembra essere tutto chiaro, ma più complesso è il suo utilizzo a livello pratico:

1. Non necessariamente deve essere inserito solo alla fine di un documento.
2. Quando contiene intere sezioni, esse rappresentano: appendici, indici, note, accordi di licenza e altri contenuti del genere.
3. Non introduce una nuova sezione e quindi non è rilevante per l'outliner.
4. All'interno di una pagina web possono essere presenti diversi <footer> anche più di uno per lo stesso elemento.

Esempio,

```
<footer>
    <small>©2030 Autore contenuto. Design by Designer sito </small>
</footer>
```

Il tag Hgroup
L'elemento **<hgroup>** rappresenta **l'intestazione di una sezione**. L'elemento viene utilizzato per raggruppare un insieme di elementi da <h1> a <h6>, quando il titolo ha più livelli, come sottotitoli, titoli alternativi o slogan. Racchiudendo un tag <h1> - <h6> all'interno di un medesimo hgroup, l'outliner riconosce come un titolo solamente l'heading con il valori più alto (h1) e considera tutti gli altri elementi sottotitoli.

Esempio,
```
<hgroup>
        <h1>Questo è il titolo</h1>
        <h2>Questo è un sottotitolo</h2>
</hgroup>
```

Come scrivere gli attributi

Attributi 'vuoti': non è necessario definire un valore per l'attributo, basta il nome, il valore si ricava implicitamente dalla stringa vuota.

Secondo le regole XHTML:
```
<input checked="checked" />
```
In HTML5:
```
<input checked>
```

Attributi con virgolette: è possibile usare la sintassi che prevede l'uso delle virgolette per racchiudere il valore di un attributo. Esempio:

Attributi con apostrofo: il valore di un attributo può essere racchiuso tra due apci (o apostrofi). Esempio:

Attributi senza virgolette: è possibile in HTML5 definire un attributo senza racchiudere il valore tra virgolette.
Esempio:

Attributi globali: possone essere applicati a qualisiasi tag HTML, essi sono, **id, style, class, title** (specifica un tooltip di aiuto), **accesskey** (specifica una combinazione di tasti veloci per spostare il focus sul tag), **tabindex** (specifica l'ordine di spostamento del focus sul controllo).

Chiusura abbreviata
Per i tag che non hanno tag di chiusura esiste la possibilità, ma non è obbligatoria, della chiusura abbreviata "/>"
```
Es,     <input type="text" name="cognome" />
        <img src=images/immagine.jpg alt=”titolo dell'immagine” />
```

Outliner
E' un nuovo plug HTML5 che può essere aggiunto ad un browser per mostrare l'**outline** (cioè la struttura o sommario) del documento. L'outliner non solo identifica correttamente i vari livelli di profondità, ma per ognuno di essi riesce anche a recuperare il titolo adeguato. In questo ambito è importante utilizzare sempre almeno un tag appartenente alla categoria **heading content** (h1-h6) all'interno di ogni Sectioning Tag, in modo da non avere un outline (sommario) popolato da voci senza titolo. L'inserimento di un titolo non è comunque obbligatorio. Le sezioni senza titolo rimangono tali ma non generano errori di validazione. I Sectioning Tag riconosciuti dall'outliner sono: Section, Article, Nav, Aside.

Proprietà STYLE - I fogli di stile - Cascading Style Sheets (CSS)

Servono per definire gli stili (come quelli di Word).
L'HTML, nonostante i vari comandi, non permette approfondite formattazioni, così da HTML 4.0 si è deciso di introdurre un nuovo attributo "style", che permette, che permette di definire delle **proprietà di stile** che possono essere applicate a qualsiasi tag HTML, sempre allo stesso modo e con gli stessi valori. Normalmente le proprietà di stile vengono scritte in un file esterno rispetto al file HTML (in un file con estensione **.css**) consentendo in questo modo una netta separazione fra i contenuti (scritti nel file .html) e la formattazione grafica (impostata nel file .css).

Le proprietà di stile possono essere scritte in tre posizioni differenti:

1. direttamente all'interno del tag HTML (stili INLINE)
   <nome_tag style="nome_proprietà1: valore; nome_proprietà2: valore;"> </nome_tag>

   Es,

   ```
   <p style="color:red; font-size:30pt;"> Questa è una prova di stile INLINE </p>
   ```

   il punto e virgola dopo l'ultima voce è facoltativo

2. nell'intestazione (head) della pagina (stili INCORPORATI nella pagina)

   ```
   <head>
   <style type=”text/css”>
       nome_tag {
                   nome_proprietà1: valore;
                   nome_proprietà2: valore;
                   ...altre proprietà...
                 }
   </style>
   </head>
   ```

   Es,

   ```
   <head>
   <style type=”text/css”>
       p {
           color: green;
           font-family: verdana, helvetica, sans-serif;
         }
   </style>
   </head>
   <body>
       <p> Questo paragrafo viene scritto in verdana di colore verde </p>
   </body>
   ```

   In una tabella si richiamerà nei tag <tr> e <td>
   Questa scrittura è più compatta e leggibile della precedente, ma occorre ripetere gli stili in ogni pagina.

3. in un apposito file esterno (fogli di stile ESTERNI memorizzati in un file .css, è il metodo più usato)
   Gli stili così definiti saranno visibili in tutte le pagina del sito
   Tutte le pagine HTML del sito, nella sezione head, dovranno richiamare il file .css mediante il tag **link**

   ```
   <head>
       <link type = "text/css"   rel = "stylesheet"   href = "sottocartella/nome_mio_file.css">
   </head>
   ```

   All'interno del file .css il tag style può essere omesso, iniziando subito a scrivere gli stili

   ```
   /* file .css */
   nome_tag {
               nome_proprietà1: valore;
               nome_proprietà2: valore;
               ...altre proprietà...
             }
   ```

   Es,

   creare una sottocartella del sito, chiamarla ad esempio 'stylesheets', all'interno creare un file di testo e rinominarlo con estensione .css (ad esempio 'default.css') e scrivervi un selettore di tipo come segue,

   ```
   p {
       font-family: arial;                 /* font utilizzato, Arial */
       background: lightyellow;            /* sfondo, giallo */
       color: darkgreen;
       font_size: 14 px;
       text-align: center;
     }
   ```

   Per definire un commento all'interno del file .css si usa la seguente dicitura: /* Questo è un commento */

   Dopodiché scrivere un normale file HTML, nell'<head> inserire la linea di codice indicata poco sopra per includere il file dei fogli di stile, inserirvi all'interno un paragrafo ed aprire la pagina con un browser web per valutare gli effetti.

Ridefinizione di un singolo attributo HTML

E' anche possibile ridefinire i singoli attributi di un tag scrivendo

```
tag:attributo {
        nome_proprietà1: valore;
        nome_proprietà2: valore;
        ...altre proprietà...
    }
```

Es,

```
a:hover  {
                color:yellow;
        }
```

Tag SPAN

Se si deve usare uno stile solo in un certo frangente (es. all'interno di una frase) si può usare il seguente comando:

** testo da modificare **

Es,

```
<span style="background: lightblue"> testo da modificare
    </span>
```

I selettori di stile

Le proprietà di stile possono essere definite mediante tre diversi **selettori:**

- selettori di tipo, cioè ridefinizione di un tag html come negli esempi precedenti
- selettori di classe, cioè ridefinizione di uno stile associabile a uno o più tag html
- selettori di elemento, (o selettori assoluti) cioè ridefinizione di uno stile associabile ad un unico tag html tramite "id" (identificatore)

I selettori di classe

Talvolta, all'interno della stessa pagina, occorre ridefinire lo stile di un tag in più modi differenti a seconda dei contesti. A tal fine occorre definire un selettore di classe in grado di creare tante classi differenti dello stesso tag, associabili poi, a seconda dei contesti, a istanze diverse dello stesso tag.

```
/* file .css */
nome_tag.nome_selettore_di_classe       {
                                        nome_proprietà1: valore;
                                        nome_proprietà2: valore;
                                        ...altre proprietà...
                                }
```

I selettori potranno poi essere associati alle varie istanze del tag mediante l'utilizzo dell'attributo **class.**

```
<!-- file HTML -->
<nome_tag   class = "nome_selettore_di_classe" > </nome_tag>
```

Es,

```
/* file .css */
p.titolo {
        color : red;
        font-size: 30pt;
       }

p.sottotitolo  {
        color : RGB(125, 125, 255);
        font-size: 20pt;
    }

<!-- file HTML -->
<body>
        <p class = "titolo"> Questo è un titolo </p>
        <p class = "sottotitolo"> Questo è un sottotitolo </p>
</body>
```

E' anche possibile definire un **selettore di classe generico**, cioè senza anteporre davanti il nome di nessun tag, ma semplicemente cominciando a scrivere direttamente il puntino. Si crea così un selettore generico che potrà essere associato a qualsiasi tag html (compatibilmente con gli stili definiti al suo interno).

```
/* file .css */
. nome_selettore_di_classe      {
                                nome_proprietà1: valore;
                                nome_proprietà2: valore;
                                ...altre proprietà...
                        }
<!-- file HTML -->
<nome_tag   class = "nome_selettore_di_classe" >  </nome_tag>
```

Es,

```
/* file .css */
.nuovoStile {
        color : red;
        font-size: 24pt;
     }

.nuovoStileBold  {
        color : red;
        font-size: 24pt;
        font-weight: bold;
    }

<!-- file HTML -->
<body>
        <p class = "nuovoStile"> Questo è il nuovo stile <span class = "nuovoStileBold"> creato  </span></p>
</body>
```

All'interno del selettore di classe, oltre agli stili del tag, è possibile ridefinire anche un singolo attributo html:

```
/* file .css */
nome_tag. nome_selettore_di_classe: nome_attributo  {
                                                        nome_proprietà1: valore;
                                                        ...altre proprietà...
                                                    }
<!-- file HTML -->
<nome_tag   class = "nome_selettore_di_classe">  </nome_tag>
```

Es,

```
/* file .css */
a.colore: hover {
                    color : yellow;
                }
<!-- file HTML -->
<a class = "colore"> Quando il mouse passerà qui la scritta diventerà gialla </a>
```

I selettori di elemento (selettori assoluti)

Dato un tag avente un identificativo **"id"** univoco, il selettore di elemento consente di associare uno stile a quella singola istanza dell'elemento. Il selettore di elemento è rappresentato con un simbolo "#" anteposto al nome dell'istanza.

```
/* file .css */
/* definire lo stile di base di un certo livello "nome_tag_principale#nome_selettore_di_elemento" */
nome_tag_principale#nome_selettore_di_elemento {
                                                    nome_proprietà1: valore;
                                                    ...altre proprietà...
                                                }
```

/* ridefinire le caratteristiche di un tag secondario tutte le volte che viene utilizzato all'interno del livello "nome_tag_principale#nome_selettore_di_elemento" */

```
nome_tag_principale#nome_selettore_di_elemento   nome_tag_secondario {
                                                        nome_proprietà1: valore;
                                                        ...altre proprietà...
                                                    }
```

/* ridefinire le caratteristiche di un attributo di un tag secondario tutte le volte che viene utilizzato all'interno del livello "nome_tag_principale#nome_selettore_di_elemento" */

```
nome_tag_principale#nome_selettore_di_elemento   nome_tag_secondario: nome_attributo {
                                                        nome_proprietà1: valore;
                                                        ...altre proprietà...
                                                    }
<!-- file HTML -->
<nome_tag_principale   id = "nome_selettore_di_elemento">
        <nome_tag_secondario>  </nome_tag_secondario>
</nome_tag>
```

Es,

```
/* file .css */
div#liv1 {
        clear:both;
        margin: 0 auto;
        border-top: 2px solid #dadada;
        width: 760px;
        text-align: center;
        color: #808080;
        font-size: 0,9 em;
}

div#liv1 p {
        padding: 0;
        margin: 0;
}

div#liv1 a {
        color: #808080;
        background-color: #CCC;
}

div#liv1 a: hover {
                color: #ff0000;
}

<!-- file HTML -->
<div id = "liv1">
        <a href = "Pagina2.html"> Questo link è scritto con stile liv a </a>
</div>
```

Oss. Essendo "id" univoco nella pagina, ogni selettore di elemento può essere associato ad un unico elemento html.
L'inserimento del nome_tag_principale davanti al nome_selettore_di_elemento è facoltativo, ma migliora la leggibilità.
Dopo nome_tag_secondario è possibile aggiungere altri sottolivelli (es. nome_tag_terziario)
Non è consentito assegnare agli "id" dei nomi numerici.
Le regole di stile possono essere assegnate contemporaneamente a più tag/classi. A tal fine occorre separare i tag mediante una **virgola.**

```
Es,     div, p, ol, ul, .riga {
                                nome_proprietà1: valore;
                            }
```

Proprietà e attributi CSS3

/* Questo è un commento */

Unità di misura

relativa **em** (ems), indica un fattore di ridimensionamento del carattere rispetto alle dimensioni impostate nel tag genitore. Ad esempio em=1.5 significa incrementare di un 50% rispetto al genitore (default 16px=12pt), indipendentemente dal tag a cui viene applicato (ad esempio se viene applicato ad H2, applica un incremento del 50% rispetto al valore di H2 del genitore)
% dimensione percentuale rispetto alla dimensione del genitore

assoluta **px** (pixel), **pt** (points; 1pt = 1/72pollice), **in** (inches; 1in = 2,54cm), **cm, mm, pc** (picas; 1pc = 12pt)

Attenzione, tra il valore numerico e l'unità di misura non devono essere lasciati degli spazi.
Utilizzare dimensioni fisse non è una tecnica consigliata, soprattutto per i problemi di scalabilità nei vari browser. E' sempre consigliabile invece utilizzare unità di misura relative come percentuale o em che meglio si adattano ai vari browser.

Elementi Block e elementi Inline

I tag **block** sono contenitori che occupano un blocco di spazio sulla pagina (div, p, h, li, table, eccetera...).
Sono preposti a contenere testo o altri oggetti:

- hanno come default la proprietà **display:block;**
- non accettano altri oggetti sulla stessa riga (salvo utilizzo della proprietà float)
- riconoscono le proprieta widht e height e text-align, ma non vertical-align

i tag **inline** sono quegli elementi span, a, img, b, i, u c, td che possono essere contenuti all'interno di un paragrafo e che quindi devono essere mostrati in riga con gli altri elementi:

- hanno come default la proprietà **display:inline;**
- possono essere inseriti all'interno di una riga come il normale testo
- non riconoscono le proprietà width, height e text-align, ma riconoscono vertical-align.

E' sempre possibile modificare qualunque tag da block a inline

Width ed Height

il valore di default è **auto**. Il valore percentuale (es, width: 50%;) è riferito rispetto alle dimensioni del genitore, per height dei tag HTML5 (header, nav, ...) il valore percentuale non funziona se il tag non è all'interno di un altro contenitore (genitore).
Per **height: auto;** significa che l'altezza sarà quella determinata dal contenuto.

Il testo

color: #codice_colore;	cambia il colore del testo con #codice_colore
text-align: left; (o center o right o justify);	allineamento testo, applicabile solo agli elementi block
text-decoration: none;	nessuna sottolineatura
underline;	sottolineatura sotto
overline;	sottolineatura sopra
line-through	testo barrato
letter-spacing: valore_in_pixel;	definisce una spaziatura addizionale fra le lettere
es, letter-spacing: 3px;	
word-spacing: valore_in_pixel;	definisce una spaziatura addizionale fra le parole
es, word-spacing: 10px;	
text-indent: valore_in_pixel;	indentazione (incolonnamento) testo
valore_in_percentuale;	il valore in percentuale è riferito al valore di width
text-transform: none;	nessuno
capitalize;	converte in maiuscolo il primo carattere di ogni parola
uppercase;	converte in maiuscolo l'intera parola
lowercase;	converte in minuscolo l'intera parola
vertical-align: baseline;	allinea le "basi" dell'elemento e del genitore
middle;	allinea il punto medio verticale dell'elemento e la base del genitore
sub;	pedice
super;	apice
text-top;	allineamento relativo punto più alto dell'elemento e il font del genitore
text-bottom;	allineamento relativo punto più basso dell'elemento e il font del genitore
top;	allinea il punto più alto dell'elemento con il più alto elemento sulla linea
bottom;	allinea il punto più basso dell'elemento con il più basso elemento sulla linea

vertical-align vertical-align serve quando si ha una riga più alta del testo (ad esempio perché c'è un elemento inline più grande, come un'immagine), per indicare dove si deve mettere il testo in relazione a suddetto elemento (e si applica all'elemento in questione non al contenitore). Applicabile solo a elementi inline.

line-height: normal; o **number, lunghezza, percentage**

line-height definisce l'altezza della riga, **normal**, significa line-height=font-size, cioè la riga avrà una altezza pari a font-size. Eventuali bordi vengono tracciati sul perimetro del testo. Impostando un valore inferiore rispetto a font-size, eventuali bordi avrebbero interferenza con il testo ed il testo sfocerebbe nella zona del margine. **Number** è un numero puro (senza unità di misura e senza %), che produce una altezza pari a N*font-size. **Percentage** è riferito al font-size. **Il testo viene centrato verticalmente rispetto al valore di line-height.**

word-wrap: break-word; permette di spezzare una parola a capo che altrimenti deborderebbe fuori dal contenitore

es,
```
p {
    word-wrap: break-word;
}
```

text-shadow: 1°parametro 2°parametro 3°parametro 4°parametro;

consente di aggiungere un'ombreggiatura al testo
1°parametro, indica la profondità dell'ombra sull'asse x (verso destra). Valori negativi indicano verso sinistra.
2°parametro, indica la profondità dell'ombra sull'asse y (verso il basso). Valori negativi verso l'alto
3°parametro, indica il raggio di sfumatura applicato al testo dell'ombra
4°parametro, indica il colore dell'ombra

es,
```
h1 {
    text-shadow: 2px 2px 0px black;
}
h2 {
    text-shadow: 3px 3px 2px #333333, 6px 7px 4px #30F744;
}
```

è possibile anche assegnare più di un'ombra all'interno della stessa regola separandole con virgola, come in questo esempio, in cui vengono create un'ombra grigia e un'ombra verde.

I font del testo

font-family: family-name, generic-family

es, font-family: Verdana, Arial, Helvetica, sans-serif;

font-family nomi dei font separati da virgola. Se il browser dispone del primo font, utilizza quello, altrimenti passa al secondo e così via. Si mettono davanti i font più specifici ed in coda quelli più generici. Non c'è valore di default, che dipende dalle impostazioni del browser.

Generic-family: serif, sans-serif, monospace
cursive, fantasy
Font-family: Verdana, Arial, Helvetica, sans-serif;
Font-family: Times New Roman, TImes, serif;

font-style: normal; o **italic, oblique** (simile all'italic, ma più inclinato)

font-weight: normal; o **light, lighter, bold, bolder, 100, 200, 300, 400, 500, 600, 700, 800, 900**
Spessore del carattere, 100=light, 400=normal, 700=bold, 900=bolder

font-size: absolute-size; o relative-siza, length, percentage
absolute-size, **xx-small, x-small, small, medium, large, x-large, xx-large**
relative-size, **larger, smaller**
length, valore dei font size in **pt** o **px**
percentage, percentuale rispetto all'attuale element's font. Per element's font si intende il size attuale del font stabilito dal font genitore (eventuale tag esterno). Se i tag esterni non definiscono il font, per element's font si intende il font-size di default

es, font-size: xx-small;

font-variant: normal; o **small-caps** (anche le minuscole sono scritte in maiuscolo, ma sono leggermente più piccole)

font: elenco proprietà; consente di specificare tutte le proprietà di font in un'unica dichiarazione, separandole con uno spazio

es, font: bold 34px Arial, Helvetica, sans-serif;

Gli sfondi

background-color: #codice_colore; cambia il colore di sfondo con #codice_colore

es, background: #FF0000;

background-color: rgb (num1, num2, num3); cambia il colore di sfondo con il comando rgb (red-green-blue), il colore viene impostato dai valori num1, num2, num3 (i tre numeri rappresentano 1 byte e quindi valori compresi fra 0 255)

es, background: rgb (0, 255, 0);

background-image: url("../nome_cartella/nome_immagine"); imposta come sfondo l'immagine "nome_immagine"

es, background-image: url("../images/sfondo.jpg"); imposta come sfondo l'immagine "sfondo.jpg" presente nella sottocartella "images"

background-image: url("nome_immagine1") url("nome_immagine2") ... url("nome_immagineN") ;

sfondi con immagini multiple, le immagini vengono caricate una sopra l'altra, la prima indicata sarà in primo piano, l'ultima sotto tutte. Per ciascuna immagine è possibile anche definire dei posizionamenti diversi.

es,
```
background-image: url ("img1.jpg")  url ("img2.jpg");
background-position: top lef, bottom left;
background-repeat: no repeat, no-repeat;
```

background-repeat: repeat; ripete l'immagine di sfondo sia in orizzontale che in verticale riempiendo la pagina
repeat-x; ripete l'immagine di sfondo soltanto in orizzontale
repeat-y; ripete l'immagine di sfondo soltanto in verticale
no-repeat; non ripete l'immagine

background-attachment: fixed; determina se lo sfondo deve rimanere fisso
scroll; o muoversi con il testo

background-position: percentuale; o **lunghezza, top, center, bottom, left, center, right**
specifica l'allineamento dell'immagine di sfondo rispetto al contenitore. Per default l'allineamento è left top (0%, 0%) e l'mmagine si estende per le sue dimensioni
lunghezza indica le coordinate assolute x y di posizionamento dell'immagine

es, background-position: bottom right;

background: elenco_proprietà; consente di specificare tutte le proprietà di background in un'unica dichiarazione, separandole con uno spazio

es, background: url(../images/main_bkgd.png) repeat-y top left;

background-clip permette di applicare un certo colore di sfondo ad una singola porzione di area del contenitore, invece che all'intero contenitore

background-clip: border-box; lo sfondo si estende per l'intera area (anche al di sotto dei bordi). Default
padding-box; lo sfondo ricopre il padding, ma non i bordi
content-box; lo sfondo ricopre solo la cosiddetta "area del contenuto"

background-size: valore1 valore2; o **cover, contain**
consente di visualizzare l'immagine soltanto su una porzione di sfondo del contenitore. La dimensione dell'area da coprire può essere espressa in pixel o in percentuale rispetto alle dimensioni del box.
Il valore **cover** fa si che l'immagine copra interamente l'area del contenitore. Se immagine e contenitore non hanno le stesse proporzioni, una dimensione dell'immagine eccederà le dimensioni del contenitore e non verrà visualizzata.
Il valore **contain** adatta l'immagine fino a quando una delle due dimensioni non arriva a coprire completamente l'area a disposizione. Sull'altra dimensione rimarrà una parte di contenitore 'scoperto'

es, background-size: 50% 50%; l'immagine viene ridimensionata in modo da coprire il 50% del contenitoresia in orizzontale che in verticale con una ovvia deformazione dell'immagine stessa

background-size: 50%; impostando la sola dimensione x, l'immagine coprirà il 50% in orizzontale del contenitore, mentre l'altezza y dell'immagine sarà calcolata in automatico senza provocare deformazione. Se la dimensione y risultante va a sforare l'altezza a sua disposizione, la parte eccedente non viene visualizzata.

Margini, bordi e riempimento

Margini

I margini rappresentano il margine esterno di un elemento. Consentono di distanziare/avvicinare un elemento rispetto agli elementi vicini. Applicabile soltanto ai tag di tipo block, cioè ai contenitori.

margin-top: valore; margine alto, per il valore vedere le unità di misura all'inizio di questa sezione
margin-right: valore; margine destro
margin-bottom: valore; margine basso
margin-left: valore; margine sinistro
margin: valori; consente di specificare tutti e quattro i margine nel seguente ordine, top, right, bottom, left.
I valori devono essere separati da uno spazio.
Specificando un solo parametro, il valore viene applicato a tutti e quattro i margini.
Specificando 2 parametri, il primo viene applicato a top/bottom, il secondo a left/right.
Specificando 3 parametri il significato è, top, right/left, bottom
E' possibile utilizzare il valore **auto** per eseguire la centratura (tipicamente centratura orizzontale), in tal caso l'oggetto viene centrato rispetto all'oggetto genitore

es,
```
p {
        margin-top:100px;
        margin-left:150px;
        margin-right: 50px
}

margin: 20px 10px 10px 0px;
```

Bordi

border-top-width:thin; o **medium, thick, valore** spessore del bordo superiore
border-right-width:thin; o **medium, thick, valore** spessore del bordo destro
border-bottom-width:thin; o **medium, thick, valore** spessore del bordo inferiore
border-left-width:thin; o **medium, thick, valore** spessore del bordo sinistro
border-width: valori; consente di specificare tutti e quattro i bordi nel seguente ordine, top, right, bottom, left. I valori devono essere separati da uno spazio. Specificando un solo parametro, il valore viene applicato a tutti e quattro i bordi. Specificando 2 parametri, il primo viene applicato a top/bottom, il secondo a left/right.

es, border-top-width: thick;
border-width: 10pt 20pt 10pt 20pt;

border-top-color: #codice_colore; colore del bordo superiore
border-right-color: #codice_colore; colore del bordo destro
border-left-color: #codice_colore; colore del bordo sinistro
border-bottom-color: #codice_colore; colore del bordo inferiore
border-color: #codice_colore; colore dei bordi. Per default viene assunto il valore della proprietà color. Può contenere 1, 2, 4 valori nel seguente ordine top, right, bottom, left. Specificando un solo colore, il valore viene applicato a tutti e quattro i bordi.

es, border-color: blue;

border-top-style: none; o **dotted**, **dashed, solid, double, groove, ridge, inset, outset** stile bordo superiore
border-right-style: none; o **dotted**, **dashed, solid, double, groove, ridge, inset, outset** stile bordo destro
border-bottom-style: none; o **dotted**, **dashed, solid, double, groove, ridge, inset, outset** stile bordo inferiore
border-left-style: none; o **dotted**, **dashed, solid, double, groove, ridge, inset, outset** stile bordo sinistro
border-style: none; o **dotted** (a puntini) **dashed** (a linette)
solid (solido) **double** (doppio)
groove (solcato) **ridge** (in rilievo)
inset (rivolto verso l'interno) **outset** (rivolto verso l'esterno)

es,
```
h1 {
    border-width: thick;
    border-color:#FF0000;
    border-style:double
}
```

border-top: width style color; imposta in un colpo solo width, style e color del top-border
border-right: width style color; imposta in un colpo solo width, style e color del right-border
border-bottom: width style color; imposta in un colpo solo width, style e color del bottom-border
border-left: width style color; imposta in un colpo solo width, style e color del left-border
border: width style color; imposta in un colpo solo width, style e color di tutti e quattro i border

es,
```
a:active {
    border: thick double red
}
```

border-top-left-radius: valore1, valore2; definisce l'arrotondamento dell'angolo superiore sinistro di un elemento
border-top-right-radius: valore1, valore2; definisce l'arrotondamento dell'angolo superiore destro di un elemento
border-bottom-right-radius: valore1, valore2; definisce l'arrotondamento dell'angolo inferiore destro di un elemento
border-bottom-left-radius: valore1, valore2; definisce l'arrotondamento dell'angolo inferiore sinistro di un elemento

I valori possono essere espressi da un numero accompagnato da un'unità di misura o in percentuale.È possibile definire uno o due valori. Se si definiscono due valori diversi, il primo imposta la misura del raggio orizzontale, il secondo quello del raggio verticale. L'uso di due valori diversi consente di ottenere angoli ellittici. Se si usa un solo valore si applica la stessa dimensione al raggio orizzontale e a quello verticale,ottenendo angoli circolari

border-radius: valore1, valore2, valore3, valore4; consente di arrotondare in un colpo solo, tutti i bordi di un elemento. I 4 parametri indicano il raggio di curvatura di ciascuno spigolo (superiore sinistro, superiore destro, inferiore sinistro, inferiore destro). Se i 4 valori sono tutti uguali si può utilizzare la notazione breve indicando un solo valore.

es,
```
#box1  {
    border-top-right-radius: 20px;
}
#box2 {
    border-top-right-radius: 20px 10px;
}
#box3 {
    border-radius: 20px 40px 60px 80px;
}
```

box-shadow: 1°parametro 2°parametro 3°parametro 4°parametro 5°parametro;

consente di aggiungere un'ombreggiatura a qualunque oggetto.
1°parametro, indica la profondità dell'ombra sull'asse x (verso destra). Valori negativi indicano verso sinistra.
2°parametro, indica la profondità dell'ombra sull'asse y (verso il basso). Valori negativi verso l'alto
3°parametro, indica il raggio di sfumatura applicato al testo dell'ombra. 0 indica senza sfumatura.
4°parametro, indica il livello di diffusione (spread) dell'ombra. Più i valori sono alti, più l'ombra tenderà ad espandersi, in tutte le direzioni. Se invece si usano valori negativi, l'ombra tende a contrarsi, fino a scomparire del tutto. Deafult 0.
5°parametro, indica il colore dell'ombra
Il 3° e 4°parametro sono facoltativi.

```
es,   div {
          box-shadow: 5px 5px 10px 2px #dedede;
      }
      div {
          box-shadow: inset 10px 10px #dedede;
      }
```

in questo caso il parametro **inset** crea un effetto ombra renderizzato verso l'interno, cioè a destra del bordo sinistro e sotto il bordo superiore

```
      div {
          box-shadow: 3px 3px 2px #333333, 6px 7px 4px #30F744;
      }
```

è possibile anche assegnare più di un'ombra all'interno della stessa regola separandole con virgola, come in questo esempio, in cui vengono create un'ombra grigia e un'ombra verde.

Il riempimento o padding

Il riempimento o padding è lo spazio interno che separa il contenuto dal bordo. Il padding non influisce sulla distanza dell'elemento dagli altri elementi

padding-top: valore;	distanza fra il contenuto dell'elemento ed il bordo superiore
padding-right: valore;	distanza fra il contenuto dell'elemento ed il bordo destro
padding-bottom: valore;	distanza fra il contenuto dell'elemento ed il bordo inferiore
padding-left: valore;	distanza fra il contenuto dell'elemento ed il bordo sinistro
padding: valori;	consente di specificare tutti quattro i padding

```
es,   h4 {
          margin-top: 50px;
          margin-left: 100px;
          margin-bottom: 50px;
          margin-right: 100px;
          border-width:5px;
          border-color: red;
          border-style : solid;
          padding:30px 100px 30px 100px;
          text-align:justify
      }
```

Elenchi puntati e numerati

list-style-type: none; o **disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha, upper-alpha**
applicabile a UL e OL indica il tipo di marker da utilizzare.
Impostando margin:0 e padding:0 il list-style-type viene automaticamente impostato a none

```
es,   ol {
          list-style-type: upper-alpha;
      }
```

list-style-image: none; o **url("percorso immagine")**
indica l'immagine da utilizzare come marker al posto dei puntini.
Prioritaria rispetto a list-style-type.

list-style-position: inside; o **outside**
inside fa si che la seconda riga continui sotto il puntino

list-style: valori; consente di impostare in un colpo solo le tre proprietà precedenti

Aspetto dei link (pseudoclassi)

a:link { }	proprietà link a riposo
a:hover { }	proprietà link a contatto con il mouse
a:active { }	proprietà link al click del mouse
a: visited { }	proprietà link visitati

Le tabelle

table { } gestione proprietà tabelle
th { } gestione proprietà intestazioni
tr { } gestione proprietà righe
td { } gestione proprietà celle

es,

```
table {
        background-color : #FFD5D5;
        border : medium solid #A52A2A;
        font-family : Arial, Helvetica, sans-serif;
        font-size : 14px;
        color : #FF3333;
        font : bold
}

td {
        background-color : #FFD5D5;
        border : medium solid #A52A2A;
        font-family : Arial, Helvetica, sans-serif;
        font-size : 14px;
        color : #FF3333;
        font : bold
}

td.green {
                background:green;
                border: 10px inset lime
}

td.purple {
                background:#FF8C00;
                border: 10px inset orange;
}
```

Il posizionamento di un contenitore

position: static; segue il normale posizionamento HTML
absolute; posizionamento assoluto mediante top e left. Se in quel punto ci sono già altri oggetti si avrà una sovrapposizione che dovrà essere gestita tramite z-order. Gli elementi con position Absolute non vengono conteggiati nel normale rendering della pagina per cui gli elementi possono essere sovrapposti. Con position:absolute è indispensabile assegnare anche width. position:absolute è incompatibile con la proprietà float. O si usa uno oppure l'altro
relative; impostando position:relative su un elemento, top e left non saranno più riferiti alla pagina ma alla posizione iniziale dell'elemento (molto usato nelle animazioni). Inoltre il posizionamento Assoluto dei tag interni al contenitore sarà riferito non più alla pagina esterna, ma al contenitore. Usando Absolute senza un Relative esterno, il posizionamento diventa assoluto rispetto alla pagina
fixed; posizionamento assoluto insensibile allo scroll della pagina. Un elemento fixed viene automaticamente visualizzato davanti rispetto a quelli non-fixed, senza bisogno di z-order

left: lunghezza; o **percentuale, auto** posizione x dell'elemento
right: lunghezza; o **percentuale, auto**
Definendo una sola delle proprietà, l'altra viene desunta automaticamente in base a width. L'impostazione position:absolute; right:30px; esegue un allineamento a destra a 30px di distanza dal bordo destro del genitore

top: lunghezza; o **percentuale, auto**
bottom: lunghezza; o **percentuale, auto**
Definendo una sola delle proprietà, l'altra viene desunta automaticamente in base a height. L'impostazione position:absolute; bottom:30px; esegue un allineamento in basso 30px al di sopra del bordo inferiore del genitore

display consente di modificare il modo di visualizzare gli elementi html da inline a block e viceversa
display: none; no display, l'elemento non viene visualizzato
block; l'elemento viene visualizzato come block, riempiendo tutta l'area a sua disposizione e con line break prima e dopo
inline; l'elemento viene visualizzato come inline (no line break)

float: left; o **right, none**

la proprietà float consente di rimuovere un elemento dal normale flusso del documento e ancorarlo su uno dei lati (destro o sinistro) del suo elemento contenitore. Il contenuto che circonda l'elemento scorrerà intorno ad esso sul lato opposto rispetto a quello indicato come valore float. Gli elementi float non vengono conteggiati nel normale rendering della pagina, cioè il contenitore non "sente" gli elementi float contenuti al suo interno, a meno che sia dichiarato anche lui stesso di tipo float (oppure overflow:auto).

Quando si imposta float:left; su un elemento, questo non si estende più per tutta la riga, ma solo per la larghezza necessaria per visualizzare il suo contenuto. L'elemento in sostanza si comporta come un elemento inline, ma mantiene tutte le caratteristiche degli elementi block, come ad esempio width ed height.

Gli elementi successivi, se non sono float, vengono "sovrapposti" agli elementi float (che non sono conteggiati nel rendering della pagina), a meno che non vengano dichiarati overflow:auto, nel qual caso ciascuno andrà ad occupare l'intero spazio orizzontale a sua disposizione. Se anche il secondo ha float:letf; non occuperà più l'intera riga ed il terzo verrà visualizzato in coda al secondo e così via.

Un elemento può flottare a destra o a sinistra rispetto al genitore. Consente ad esempio, di inserire all'interno di un livello una immagine allineata a sinistra e circondata dal testo.

Impostando cioè float:left; e width:xx ad un certo oggetto, gli oggetti html successivi vengono posizionati alla sua destra anziché sotto (finché ce ne stanno). Volendo gestire più livelli su colonne successive, è possibile impostare sui vari livelli foat:left e questi verranno visualizzati uno a fianco all'altro. Nel caso di soli 3 livelli, si può impostare sul primo float:left; sul 2° float:right; ed il terzo verrà automaticamente posizionato in mezzo

clear: none; o **left, right, both**

la proprietà clear serve per controllare il comportamento degli elementi che seguono elementi flottanti. Per default gli elementi successivi si muovono in modo da riempire lo spazio disponibile lasciato libero quando un oggetto viene flottato da un lato. Impostando clear: both; sull'oggetto successivo rispetto a quello floating, lo riporta al di sotto dell'oggetto floating. Clear: left; interrompe il solo floating sinistro

Proprietà overflow (overflow-x e overflow-y)

ha senso quando il testo interno eccede le dimensioni di un contenitore a dimensioni fisse.

overflow: auto; o **visible, hidden, scroll**
overflow-x: auto; o **visible, hidden, scroll**
overflow-y: auto; o **visible, hidden, scroll**

auto, se si impostano width ed height, viene visualizzata una barra di scorrimento solo se il testo eccede le dimensioni del contenitore. Se non si impostano width ed height, overflow:auto fa si che il contenitore si estenda fisicamente per tutta l'area a sua disposizione, "sentendo" anche gli elementi float interni. overflow:auto; rappresenta un'ottima alternativa a float:left; nel caso di un ultimo elemento di una sequenza float, oppure nel caso di contenitore di elementi float. In quest'ultimo caso però, rispetto a float, ha il vantaggio che il contenitore non ha il vincolo di essere ancorato a sinistra e può rimanere a centro pagina.

visible, la porzione eccedente le dimensioni del box viene mostrata eccedendo le dimensioni del box
hidden, la porzione eccedente le dimensioni del box non viene mostrato
scroll, sul contenitore vengono applicate le barre di scorrimento sia verticale, sia orizzontale

Proprietà cursor, per cambiare forma al cursore del mouse

all-scroll	col-resize	crosshair	e-resize
hand	help	move	n-resize
ne-resize	no-drop	not-allowed	nw-resize
pointer	progress	row-resize	s-resize
se-resize	sw-resize	text	vertical-text
w-resize	wait		

es,

```
div: hover
        {
                cursor: pointer;
        }
```

Altre proprietà

z-index: valore_numerico;	introduce, oltre a x e y, un terzo asse relativo alla profondità. Elementi con z-index maggiore vengono visualizzati davanti (sopra) rispetto ad elementi con z-index minore. Il default è zero. Utile per sovrapporre immagini che hanno position:absolute; o position:relative; Il testo solitamente ha z-index:0; Assegnando ad una immagine z-index:-1; ed una posizione assoluta, questa viene visualizzata al di sotto del testo presente in quella posizione.
visibility: visible;	o **hidden** consente di mostrare/nascondere un elemento
opacity: valore;	valore è un numero con la virgola tra 0 (transparent) e 1 (default, completamente solido). Utile per rendere semitrasparente un contenitore rispetto allo sfondo, la trasparenza viene ereditata da tutti gli elementi interni al box (compreso il testo ed eventuali immagini)

HTML5 Reset Stylesheet

Quando si progetta un foglio di stile, bisogna tenere conto anche degli stili integrati di default in ogni browser, stili che assegnano un valore iniziale alle proprietà di ogni elemento. Purtroppo in questo caso non esiste uno standard, così i valori iniziali variano da browser a browser. Per risolvere il problema, sono stati realizzati diversi fogli di stile da includere nei progetti mirati ad azzerare le proprietà personalizzate dei vari browser. Questi fogli di stile sono chiamati CSS Reset e vanno inseriti prima di tutte le altre regole CSS del progetto.

```
/* Reset Stylesheet */

html, body, div, span, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
abbr, address, cite, code,
del, dfn, em, img, ins, kbd, q, samp,
small, strong, sub, sup, var,
b, i,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, figcaption, figure,
footer, header, hgroup, menu, nav, section, summary,
time, mark, audio, video {
	    margin:0;
	    padding:0;
	    border:0;
	    outline:0;
	    font-size:100%;
	    vertical-align:baseline;
	    background:transparent;
}

body {
	    line-height:1;
}

article,aside,details,figcaption,figure,
footer,header,hgroup,menu,nav,section {
	    display:block;
}

nav ul {
	    list-style:none;
}

blockquote, q {
	    quotes:none;
}

blockquote:before, blockquote:after,
q:before, q:after {
	    content:'';
	    content:none;
}

a {
	    margin:0;
	    padding:0;
	    font-size:100%;
	    vertical-align:baseline;
	    background:transparent;
}

ins {
	    background-color:#ff9;
	color:#000;
	text-decoration:none;
}

mark {
	background-color:#ff9;
	color:#000;
	font-style:italic;
	font-weight:bold;
}

del {
	 text-decoration: line-through;
}

abbr[title], dfn[title] {
	border-bottom:1px dotted;
	cursor:help;
}

table {
	border-collapse:collapse;
	border-spacing:0;
}

hr {
	display:block;
	height:1px;
	border:0;
	border-top:1px solid #cccccc;
	margin:1em 0;
	padding:0;
}

input, select {
	vertical-align:middle;
}
```

Tipi di Layout delle pagine Web

I **layout** delle pagine web si suddividono sostanzialmente in tre categorie:

- **Layout fisso**
  E' generalmente studiato per una dimensione standard e pertanto non è possibile variare la larghezza ridimensionando la finestra del browser o cambiando la dimensione del testo. La larghezza dei vari componenti che compongono la pagina è specificata solitamente in **unità di misura assoluta** generalmente espressa in pixel (**px**). Mediante questa tecnica si riesce ad essere precisi senza scendere in calcoli complicati, si può essere sicuri che ciò che si codifica verrà reso uguale indipendentemente dalla macchina utilizzata, ma il prodotto risulterà essere dimensionato secondo le nostre esigenze e non secondo quelle dell'utente, questo significa che un layout fisso avrà sempre le stesse dimensioni anche se il dispositivo utilizzato dall'utente avrà dimensioni diverse: se più piccole il lavoro perderà i suoi requisiti di leggibilità e armonia che lo distinguevano visualizzandolo attraverso un dispositivo avente dimensioni ideali; non solo, utilizzando i pixel il sito non sarà scalabile modificando la dimensione del font. Allo stesso modo la pagina a grandi risoluzioni risulterà "vuota" ai lati. Il layout fisso è senza dubbio più facile da realizzare e offre maggiori possibilità di personalizzazione grafica rispetto al layout liquido, ma bisogna considerare che un buon layout fisso deve essere progettato per servire bene la maggior parte degli utenti, soprattutto senza scrolling orizzontale. Vi sono tre principali larghezze utilizzate per questo tipo di **layout**: 600px, 760px e 960px che tengono conto rispettivamente delle più diffuse risoluzioni del monitor: 640x480 pixel - 800x600 pixels - 1024x768 pixels.

- **Layout fluido (o liquido)**
  Una pagina web caratterizzata da un layout fluido è una pagina web che occupa tutto lo spazio disponibile della finestra del browser qualunque sia la risoluzione del monitor, adattandosi sempre alle misure dello schermo dell'utente. La larghezza dei vari componenti che compongono la pagina è specificata solitamente in **percentuale (%)**. Ridimensionando la finestra del browser anche la pagina web verrà ridimensionata, il disegno della pagina web non è però influenzato dai cambiamenti del formato testo. Il layout fluido è più versatile per l'utente poiché si presta ad uno spettro più ampio di risoluzioni. La sua realizzazione può risultare un po' più complessa rispetto al layout fisso, per le limitate possibilità grafiche soprattutto in casi di layout liquidi con colonne percentuali. Inoltre in schermi molto larghi, la lettura di righe lunghissime potrebbe rivelarsi problematica, ed è universalmente riconosciuto che un layout eccessivamente allargato risulta essere sgradevole alla vista. Il rappresentare, inoltre, le dimensioni in percentuali, potrebbe generare processi di arrotondamento differenti da browser a browser che potrebbero causare una resa non omogenea del sito. Infine questo tipo di layout consente una gestione migliore in fase di stampa della pagina web a differenza delle pagine con layout fissi che sovente hanno come risultato il taglio dei contenuti su uno o entrambi i lati.

- **Layout elastico**
  Il layout elastico usa il **dimensionamento in unità relative** (**em**), sia per il testo, che per la larghezza degli elementi che compongono la pagina web. Si differenzia dal fluido in quanto non si adatta solo alla risoluzione del monitor ma ridimensiona anche tutti i contenuti secondo le preferenze dell'utente cosicché solo ridimensionando il carattere del browser è possibile agire sulla larghezza della pagina. L'unità di misura relativa em, nata in ambito tipografico, rappresentava la dimensione della lettera M maiuscola per un determinato tipo di font. Nel mondo dei CSS, 1em corrisponde al valore font-size impostato per quel dato elemento o ereditato da un elemento a lui genitore. Dato che normalmente i browser settano la misura di default del font a 16 px, un em corrisponderà a quella dimensione. Lo svantaggio di questo sistema di dimensionamento è che i calcoli si complicano facilmente, inoltre dal momento che em eredita la dimensione dal suo elemento genitore, si avranno situazioni in cui 1em produrrà del testo sempre più piccolo o più grande man mano che gli elementi vengono annidati uno nell'altro.

Vi è poi anche il **layout ibrido** che è una miscela di due (o di tre) dei layout citati sopra. Un esempio potrebbe essere il caso in cui la larghezza della pagina web è di tipo fluido (espressa in %) mentre la colonna che contiene il menu di navigazione è di tipo elastico e quindi con dimensioni espresse in em.

Javascript

Definizioni utili

Script: è un programma, un insieme di comandi, che viene interpretato da un certo applicativo, che nel caso di JavaScript è costituito dal browser. JavaScript è dunque un linguaggio che consente di scrivere del codice all'interno di una pagina HTML, codice che verrà interpretato dal browser al momento della visualizzazione della pagina.

Linguaggi compilati e linguaggi interpretati
Un programma viene sviluppato **scrivendo il codice sorgente** in un opportuno linguaggio. La differenza tra compilazione ed interpretazione è molto importante ed influisce sia sulle prestazioni che sulle possibilità del linguaggio stesso.
Linguaggi come il C, C++, Delphi, Visual Basic sono **linguaggi compilati** e seguono questi passi: si scrive il codice in un editor, al più utilizzando un ambiente di sviluppo IDE(Integrated Development Environment) che ne facilita la creazione, questo codice viene poi controllato per verificare che non ci siano errori e poi viene compilato, ovvero ogni istruzione viene trasformata nel corrispondente codice in linguaggio macchina che può essere, così, eseguito dal processore; questi sono i linguaggi compilati che vengono detti anche linguaggi imperativi, ed hanno il vantaggio di prestazioni migliori.
I **linguaggi interpretati**, invece, seguono una strada diversa, il codice sorgente viene, appunto, interpretato al volo e vengono, quindi, eseguite le istruzioni così come descritte nel codice sorgente; un esempio su tutti è il PHP il cui codice viene "elaborato" e restituisce una pagina html pura. La potenza di questo genere di linguaggi è, di fatto, l'alta portabilità e l'immediatezza tra quello che scriviamo e quello che viene presentato all'esecuzione del programma, ma rimangono dei problemi come la ricerca di errori nel codice sorgente o il carico di lavoro maggiore per il processore (che ogni volta deve elaborare la pagina).
Un **linguaggio che è a metà strada tra queste metodologie è Java** che è sia compilato che interpretato; il codice sorgente viene compilato in un formato intermedio (chiamato bytecode), il quale a sua volta viene interpretato dalla Java Virtual Machine (JVM), che ha il compito di interpretare "al volo" le istruzioni bytecode in istruzioni per il processore; la JVM viene sviluppata per ogni Sistema Operativo e permette di astrarre la macchina virtuale creata dal SO ad un livello di standardizzazione superiore (data di fatto dalla creazione della virtual machine sopra un'altra virtual machine) che rende, in pratica, JAVA altamente portabile.
Questa metodologia implica la possibilità di controllare eventuali errori del codice sorgente (grazie alla compilazione), di creare programmi relativamente leggeri (il bytecode è un formato che crea file di dimensioni ragionevoli), ma ha la pecca di avere delle prestazioni non proprio soddisfacenti, questo perché il codice viene interpretato dalla JVM che a sua volta deve delegare l'esecuzione vera e propria al Sistema Operativo.

JavaScript

JavaScript è un linguaggio di scripting orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione, in siti web e applicazioni web, di effetti dinamici interattivi tramite funzioni di script invocate da eventi innescati a loro volta in vari modi dall'utente sulla pagina web in uso (mouse, tastiera, caricamento della pagina ecc...). Tali funzioni di script possono essere opportunamente inserite in file HTML o in appositi file separati con estensione *.js* poi richiamati in file HTML.
Java, JavaScript e JScript: JavaScript fu originariamente sviluppato dalla Netscape Communications con il nome di Mochan e successivamente di LiveScript, ma in seguito è stato rinominato JavaScript ed è stato formalizzato con una sintassi più vicina a quella del linguaggio Java; JavaScript è stato standardizzato per la prima volta nel 1997. La scelta del nome si rivelò fonte di grande confusione, non c'è una vera relazione tra Java e JavaScript, le loro somiglianze sono soprattutto nella sintassi (derivata in entrambi i casi dal linguaggio C), ma le loro semantiche sono piuttosto diverse, e in particolare i loro object model non hanno relazione e sono ampiamente incompatibili. Dato il successo di JavaScript come linguaggio per arricchire le pagine web, Microsoft sviluppò un linguaggio compatibile, conosciuto come JScript.
JavaScript engine: interprete o compilatore integrato agli ambienti di sviluppo. Fino a qualche anno fa JavaScript era un linguaggio esclusivamente interpretato. L'esigenza di performance sempre maggiori ha poi condotto alla creazione di engine che effettuassero una compilazione in tempo reale (JIT, Just In Time compilation**)** in bytecode o addirittura in codice macchina.

Aspetti strutturali
Le caratteristiche principali di JavaScript sono:

- è un linguaggio interpretato (in JavaScript *lato client*, l'interprete è incluso nel browser che si sta utilizzando)
- la sintassi è relativamente simile a quella del C, del C++ e del Java
- definisce le funzionalità tipiche dei linguaggi di programmazione ad alto livello (strutture di controllo, cicli, ecc.) e consente l'utilizzo del paradigma ad oggetti (object oriented)
- è un linguaggio debolmente tipizzato
- è un linguaggio debolmente orientato agli oggetti

In JavaScript lato client, il codice viene eseguito direttamente sul client e non sul server. Il vantaggio di questo approccio è che, anche con la presenza di script particolarmente complessi, il web server non viene sovraccaricato a causa delle richieste dei client. Di contro, nel caso di script che presentino un codice sorgente particolarmente grande, il tempo per lo scaricamento può diventare abbastanza lungo, inoltre ogni informazione che presuppone un accesso a dati memorizzati in un database remoto deve essere rimandata ad un linguaggio che effettui esplicitamente la transazione, per poi restituire i risultati ad una o più variabili JavaScript.
A differenza di altri linguaggi, quali il C o il C++, che permettono la scrittura di programmi completamente stand-alone, JavaScript viene utilizzato soprattutto in quanto linguaggio di scripting, integrato, quindi, all'interno di un altro programma. L'esempio tipico (e, forse, il più noto) di *programma ospite* per uno script JavaScript è quello del browser. Un browser tipicamente incorpora un interprete JavaScript, quando viene visitata una pagina web che contiene il codice di uno script JavaScript, quest'ultimo viene portato in memoria primaria ed eseguito dall'interprete contenuto nel browser.
Le interfacce che consentono a JavaScript di rapportarsi con un browser sono chiamate DOM (*Document Object Model*, in italiano *Modello a Oggetti del Documento*). Molti siti web usano la tecnologia JavaScript lato client per creare potenti applicazioni web dinamiche.
Attenzione! Se si compie qualche errore, non verrà segnalato nulla dall'interprete, semplicemente il codice non funzionerà.

Esempi di applicazione,

- inserimento di messaggi a scorrimento che cambiano sulla barra di stato del browser
- esecuzione di calcoli matematici e di conteggi di lettere o parole (ed usi simili)
- inserimento di messaggi di avviso all'interno di una pagina di un sito
- rendere le immagini animate e cambiare la loro visualizzazione quando si passa il mouse sopra di esse
- identificare il tipo di browser in utilizzo e visualizzare un contenuto adatto e diverso per ogni browser
- ricercare i plug-in installati e avvisare l'utente se è richiesto un ulteriore plug-in per continuare

A seconda del browser utilizzato JavaScript (ma anche altri linguaggi di programmazione) viene visualizzato in maniera diversa, può capitare infatti che i colori siano diversi al passaggio del mouse, il testo si visualizzi in modo inesatto o nei casi più gravi di incompatibilità lo script potrebbe non essere visualizzato o mandare il browser in crash.

Sintassi di base,

- JavaScript è case sensitive, fa differenza fra minuscole e maiuscole
- le istruzioni terminano con un punto e virgola ";"
- le variabili dichiarate fuori dalle funzioni hanno visibilità globale, ma non vengono inizializzate
- Inserire commenti, si usano // questo è un commento su riga singola
  /* questo è un commento su riga multipla */

Inserire codice JavaScript in una pagina HTML

Esistono essenzialmente tre modi per farlo:

- inserire **codice inline**;
- scrivere **blocchi di codice nella pagina**;
- importare file con codice **JavaScript esterno**.

Codice inline

Consiste nell'inserire direttamente le istruzioni JavaScript nel codice di un elemento HTML, assegnandolo ad un attributo che rappresenta un evento.

Esempi, <button type="button" onclick="alert('Ciao!')">Cliccami</button>
in questo caso viene assegnato all'attributo onclick dell'elemento button la stringa alert('Ciao!'); l'attributo onclick rappresenta l'evento del clic sul pulsante del mouse, quindi in corrispondenza di questo evento verrà analizzato ed eseguito il codice JavaScript assegnato, nel caso specifico verrà visualizzato un box con la scritta "Ciao!".

Un altro approccio per l'inserimento di codice inline, utilizzabile però soltanto con i link, è quello mostrato nel seguente esempio, Cliccami
In questo caso viene indicato al browser di interpretare il link come la richiesta di esecuzione del codice JavaScript specificato, e non come un collegamento ad un'altra pagina.

Blocchi di codice, il tag <script>

L'approccio inline può risultare immediato perché mette direttamente in relazione il codice da eseguire con un elemento HTML, ma risulta scomodo quando il codice è più complesso e si ha necessità di definire variabili e funzioni: in questi casi si può ricorrere al tag **<script>** per inserire blocchi di codice in una pagina HTML.

Esempio, <script>alert('Ciao!')</script>

Quando il parser HTML del browser esaminerà la pagina, riconoscerà il tag <script> come blocco di codice e ne passerà il contenuto all'engine JavaScript, che lo eseguirà. Si possono inserire blocchi di codice, e i relativi tag <script>, nella sezione <head> o nella sezione <body> della pagina HTML.

JavaScript esterno

Il terzo approccio, quello più consigliato, consiste nel collegare alla pagina HTML del codice JavaScript presente in un file esterno. Questa tecnica permette di agganciare script e librerie in modo detto *non intrusivo*, con il vantaggio di una separazione netta tra la struttura del documento e il codice, come accade per i fogli di stile CSS, che separano struttura e presentazione.

Per inserire un file JavaScript esterno si utilizza il tag <script>, con gli attributi type e src, nella sezione <head> come per i CSS.

Esempio, <script type="text/javascript" src="nome_sottocartella/index.js"> </script>

La gestione degli eventi

Gli attributi di evento sono quelli che consentono di richiamare le funzioni utente scritte nella sezione JavaScript.
Come gli altri attributi HTML, per attivarli basta inserirli in un tag HTML prima della chiusura ">".

Esempio, scrivere del codice JavaScript esterno e richiamarlo nella pagina HTML

creare una sottocartella in cui inserire un file index.js, nel file scrivere

```
function esegui() {
        alert("Questa e' una prova");
}
```

nel file HTML scrivere

```
... nell'head ...
<script type="text/javascript" src="javascripts/index.js"></script>

... nel body ...
<a href="#" onClick="esegui()"> Home </a>
```

cliccando sul link "Home" non avverrà un normale collegmento ipertestuale, ma verrà eseguita la funzione "esegui()", che visualizzerà semplicemente il messaggio "Questa è una prova"

Dichiarazione delle variabili
Sintassi, var nome_variabile;
Le variabili non sono tipizzate, non bisogna quindi specificarne il tipo (intero, decimale, carattere, stringhe), inoltre durante uno script il tipo di dato che può contenere una variabile può cambiare. Le variabili si possono anche non dichiarare (dichiarazione implicita), ma in questo caso vengono dichiarate come globali al loro primo utilizzo, saranno quindi visibili in tutte le funzioni e questo può generare errori, oltre alla confusione nel caso di variabili con lo stesso nome, ma usi diversi (esempio, la stessa variabile trattata in un frammento di codice come numerica e in un altro come stringa). Nonostante quindi una certa flessibilità si consiglia comunque di dichiarare sempre le variabili ed evitare cambi di tipo: per aiutarci Javascript ci viene incontro con il comando **"use strict";** da inserire all'inizio del codice, così ogni assenza di dichiarazione di variabili sarà intercettata dall'engine e verrà segnalato un errore.
Il linguaggio JavaScript ricava il tipo di una variabile in base al valore contenuto, si hanno quindi cinque tipi di dati primitivi, numeri, stringhe, booleani, null e undefined ed un tipo di dato complesso, gli oggetti. Tutti gli altri elementi previsti dal linguaggio, come ad esempio gli array (vettori) e le funzioni, sono in realtà oggetti. A dire il vero, anche i tipi di dato "primitivi" hanno degli oggetti corrispondenti con relative proprietà e metodi: JavaScript converte automaticamente un tipo primitivo nel corrispondente oggetto quando si utilizza un suo metodo o una sua proprietà, questo contribuisce a creare non poca confusione nella distinzione tra dati primitivi e oggetti, tanto che non sono in pochi a semplificare sostenendo che in realtà in JavaScript tutto è oggetto.

Quando si dichiara una variabile senza specificare un valore, a questa viene assegnato il valore **undefined**.

Variabili **numeriche**: JavaScript ha un unico tipo di dato numerico, cioè non c'è distinzione ad esempio, fra intero e decimale. Tutti i valori numerici sono rappresentati come numeri in virgola mobile, ma se non è specificata la parte decimale il numero viene trattato come intero.
Esempi,

```
var interoPositivo = 123;
var numeroDecimale = 0.52;
```

Oltre alla classica notazione in base dieci, si possono rappresentare i numeri in notazione esadecimale**:**
Esempio, var numeroEsadecimale = 0x123; //equivalente a 291, un numero esadecimale inizia con "0x"

L'insieme dei numeri rappresentabili in JavaScript cade nell'intervallo compreso tra $-1.79769*10^{308}$ e $1.79769*10^{308}$. Ogni valore che va al di fuori dell'intervallo rappresentabile non genera un'eccezione ma viene rappresentato da due valori speciali:

- *Infinity,* detto anche infinito positivo,
- *-Infinity,* detto infinito negativo.

Ogni valore che ha una precisione maggiore di quella rappresentabile viene considerata uguale a zero.
Un altro valore numerico speciale è *NaN*, acronimo di *Not a Number*, che indica un valore numerico non definito.
Esempio, var x = x + 1;
in questo caso dopo l'assegnamento, il valore della variabile x è NaN

Tipo di dato **null**: prevede il solo valore null, che rappresenta un valore che non rientra tra i tipi di dato del linguaggio, cioè non è un valore numerico valido, né una stringa, né un oggetto.
Esempio, var x = null;
Il valore null è diverso dal valore numerico 0 o dalla stringa vuota, infatti questi ultimi appartengono ai tipi numero e stringa, mentre null è un tipo a se stante.

Tipo di dato **undefined**: rappresenta un valore che non esiste ed anche questo tipo di dato contiene un solo valore, undefined: questo è il valore di una variabile non inizializzata, a cui non è stato assegnato nessun valore, nemmeno null.

Tipo di dato **booleano**: prevede due soli valori: true (vero) e false (falso).

Stringhe: una stringa in JavaScript è una sequenza di caratteri delimitata da **doppi o singoli apici**.
Esempi di stringhe: "stringa numero uno", 'stringa numero due'
Non c'è una regola per stabilire quale delimitatore utilizzare. L'unica regola è che il delimitatore finale deve essere uguale al delimitatore iniziale. Un tipo speciale di stringa è la stringa vuota, cioè una stringa senza caratteri, essa può essere rappresentata indifferentemente come "" oppure ''.

Per inserire caratteri speciali all'interno di una stringa si fa ricorso al *carattere di escaping* \ (backslash).
Esempio, per inserire un ritorno a capo si può utilizzare la sequenza \n

```
"Ieri pioveva.\nOggi c'è il sole."
```

la stampa di questa stringa avrà il seguente risultato:

```
Ieri pioveva.
Oggi c'è il sole.
```

È anche possibile *inserire caratteri Unicode in una stringa* utilizzando la sequenza di escaping \u seguita dal codice esadecimale del carattere. Esempio, la sequenza \u00A9 indica il carattere ©:

Le costanti
Sintassi, const nome_costante = valore;
Esempio, const PIGRECO = 3.14;

I vettori
I vettori, o array, consentono di associare più valori ad un unico nome di variabile (o identificatore). In genere i valori contenuti in un array hanno una qualche affinità (ad esempio l'elenco dei giorni della settimana), anche se in JavaScript possono essere eterogenei. L'uso degli array evita di definire più variabili e semplifica lo svolgimento di operazioni cicliche su tutti i valori.
Sintassi dichiarazione,

```
var nome_vettore = [elemento1, elemento2, ..., elementoN ];   // viene dichiarato un vettore ed i suoi elementi
var nome_vettore = new Array(numero_elementi);                // viene istanziato un array generico di numero_elementi
```

Esempio,

```
var giorniDellaSettimana = [ "lunedì", "martedì", "mercoledì", "giovedì", "venerdì", "sabato", "domenica"];
var vettore = new Array(30)
```

Una volta definito un array si può accedere ai singoli elementi facendo riferimento al nome della variabile e all'indice corrispondente all'elemento, ricordando che **la numerazione degli indici parte da zero**.

Esempio, per accedere al primo elemento dell'array dei giorni della settimana si scriverà,

```
var primoGiorno = giorniDellaSettimana[0];
```

naturalmente l'indice scalerà allo stesso modo anche per gli altri giorni,

```
var terzoGiorno = giorniDellaSettimana[2];
var settimoGiorno = giorniDellaSettimana[6];
```

dopo l'assegnamento ciascuna variabile conterrà la stringa corrispondente all'indice.

Espressioni e operatori

- Operatori matematici
  Uguaglianza-Assegnazione ('='), somma ('+'), sottrazione ('-'), moltiplicazione ('*'), divisione ('/'), resto di una divisione (resto = a % b), operatori di incremento '++' (aggiunge 1 al suo operando, es. cont++;) e decremento '--' (sottrae 1 al suo operando, es. i--;), elevamento a potenza (risultato = Math.pow(x,y) equivale a risultato=x^y , oggetto Math), radice quadrata (radice = Math.sqrt(valore), oggetto Math),
  Es. somma = a+b; prodotto = a*b; potenza = Math.pow(base, esponente);

- Operatori Logici e di Confronto

&& , And	| | , Or	! , Not
> , maggiore	< , minore	== , uguale
>= , maggiore o uguale	<= , minore o uguale	

!= , diverso

  Attenzione a non confondere && (AND logico) con & (bitwise AND, AND bit a bit), la stessa cosa vale per || e |.
  Gli operatori di bitwise (che operano sui singoli bit) sono i seguenti:
  "&" AND , "|" OR , "^" XOR , "<<" shift a sinistra , ">>" shift a destra
  "~" NOT, è un operatore unario, cioè opera su un solo argomento indicato a destra dell'operatore

- Operatori su stringhe
  Oltre agli operatori di assegnamento, logici e relazionali, che sono previsti per tutti i tipi, l'unico operatore su stringhe specifico è l'operatore di concatenazione. Esso consente di creare una nuova stringa come risultato della concatenazione di due stringhe ed è rappresentato dal simbolo del "più" (+):
  Esempio,

```
"use strict";
var strumento= "piano";
strumento = strumento + "forte";          // strumento = "pianoforte"
```

  Attenzione: l'operatore + stabilisce che, *se almeno uno dei due operandi è una stringa, allora viene effettuata una concatenazione di stringhe, altrimenti viene eseguita una addizione.*
  Esempio,

```
"use strict";
var x = "12";
var y = x + 13;
```

  dato che la variabile x contiene una stringa, l'operatore + sarà interpretato come una concatenazione di stringhe e pertanto il valore 13 sarà convertito nella stringa "13". Il valore di y sarà quindi la stringa "1213".

Istruzione condizionale

```
if (condizione)    {
                        ...istruzioni allora...
                   }
else
        {
                   ...istruzioni altrimenti...
        }
```

Oss: scrivere x=10 significa assegnare alla x il valore 10, scrivere x==10 significa verificare se x è uguale a 10. Se per errore si scrive if(x=10) la condizione risultante sarà sempre vera in quanto ad x viene assegnato il valore 10 che essendo un valore non nullo restituisce sempre vero.

Esempio,

```
"use strict";

x = y % 2;
if (x == 0) {
        messaggio = x + " è pari";
}

else {
        messaggio = x + " è dispari";
}
```

Switch-case
Spesso in un programma c'è necessità di confrontare il valore di una variabile con una serie di valori fino a trovare quello corrispondente. L'struzione switch confronta il valore della variabile al valore specificato dalla clausola case.
Sintassi:

```
switch(nome_var){
        case label1:
                // istruzioni
                break;
        case label2:
                // istruzioni
                break;
        default:
                // istruzioni
}
```

dove nome_var è una variabile che va a confrontarsi con i diversi casi e label sono i valori di confronto con nome_var

Ciclo while

```
while(condizione) {
        istruzioni_ciclo_while;
}
```

Ciclo do-while

```
do{
        istruzioni_ciclo_do_while;
} while(condizione);
```

Ciclo for

```
for (inizializzazione ; condizione ; incremento)
    {
        istruzioni_ciclo_for;
    }
```

Funzioni
Sintassi dichiarazione,

```
function nome_funzione(parametro1, parametro2, ..., parametroN)
    {
        // istruzioni
    }
```

Sintassi richiamo,

```
nome_funzione(parametro1, parametro2, ..., parametroN)
```

Come si può notare, rispetto ai linguaggi tradizionali, si omette sia il tipo di parametri, sia il tipo restituito

L'istruzione return: nel corpo della funzione può essere presente l'istruzione return che consente di terminare e restituire un valore al codice che l'ha chiamata. Questo consente di assegnare ad una variabile il valore restituito da una funzione o utilizzare una funzione all'interno di una espressione.

Esempio,

```
"use strict";
function somma(x, y) {
        var z;
        z = x + y;
        return z;
}
```

In questo caso i valori da sommare verranno passati alla funzione somma() al momento dell'invocazione:

```
var risultato = somma(11, 5);
```

Passaggio parametri per valore e per riferimento: il passaggio di valori relativi a tipi di dato primitivi avviene sempre per valore mentre il passaggio di oggetti avviene sempre per riferimento.

Le variabili dichiarate all'interno del corpo di una funzione, sono accessibili soltanto all'interno della funzione e non vengono viste fuori di essa o, in termini tecnici, hanno uno scope locale. Lo **scope** o *ambito di visibilità* di una variabile è la parte di uno script all'interno del quale si può fare riferimento alla variabile stessa. Le variabili dichiarate all'interno di una funzione sono dette *locali* alla funzione dal momento che sono accessibili soltanto all'interno del suo corpo. Le variabili dichiarate fuori da qualsiasi funzione sono dette *globali* e sono accessibili da qualsiasi punto dello script, anche all'interno di funzioni.

Oggetti JavaScript: proprietà e metodi

Nella programmazione ad oggetti vi è un'ulteriore astrazione dei tipi di dato, che vengono chiamate ***proprietà o attributi***, e che uniti alle funzioni che vengono chiamate ***metodi***, formano le cosiddette ***classi***. Le classi costituiscono dei modelli astratti e quando vengono utilizzate per 'dichiarare variabili', si parla di **'*istanziare un oggetto*'**: quindi una 'classe' sta ad un 'tipo di variabile', come un 'oggetto' sta ad una 'variabile', mentre 'istanziare' sta a 'dichiarare'.
Esempio,

classe, animale	classe, veicolo
proprietà, numero di zampe	proprietà, numero di ruote, numero di marce
metodi, alimentazione	metodi, accensione
oggetti istanziabili, cane, gatto	oggetti istanziabili, bicicletta, motocicletta, automobile

altri esempi di classi, persona, un ordine, fattura, una prenotazione, eccetera...

La parte del programma Javascript che fa uso di un oggetto si chiama client.
Un linguaggio di programmazione è definito ad oggetti quando permette di implementare tre meccanismi usando la sintassi nativa del linguaggio:

- incapsulamento, che consiste nella separazione della cosiddetta interfaccia di una classe dalla corrispondente implementazione, in modo che i client di un oggetto di quella classe possano utilizzare la prima (interfaccia), ma non la seconda (implementazione)
- ereditarietà, permette essenzialmente di definire delle classi a partire da altre già definite
- polimorfismo, permette di scrivere un client che può servirsi di oggetti di classi diverse, ma dotati di una stessa interfaccia comune; a tempo di esecuzione, quel client attiverà comportamenti diversi senza conoscere a priori il tipo specifico dell'oggetto che gli viene passato.

Come detto, un oggetto possiede:

- *Dati*, detti **proprietà** e rappresentati da coppie di 'nome:valore'
- *Funzionalità*, sono dette **metodi** e rappresentate da funzioni.

Esempio, istanziare un oggetto

```
var persona = {
        nome: "Mario",
        cognome: "Rossi",
        indirizzo: {
                via: "Via Garibaldi",
                numero: 15,
                CAP: "00100",
                citta: "Roma"
        }
};
```

La proprietà indirizzo è a sua volta un oggetto composto da specifiche proprietà.

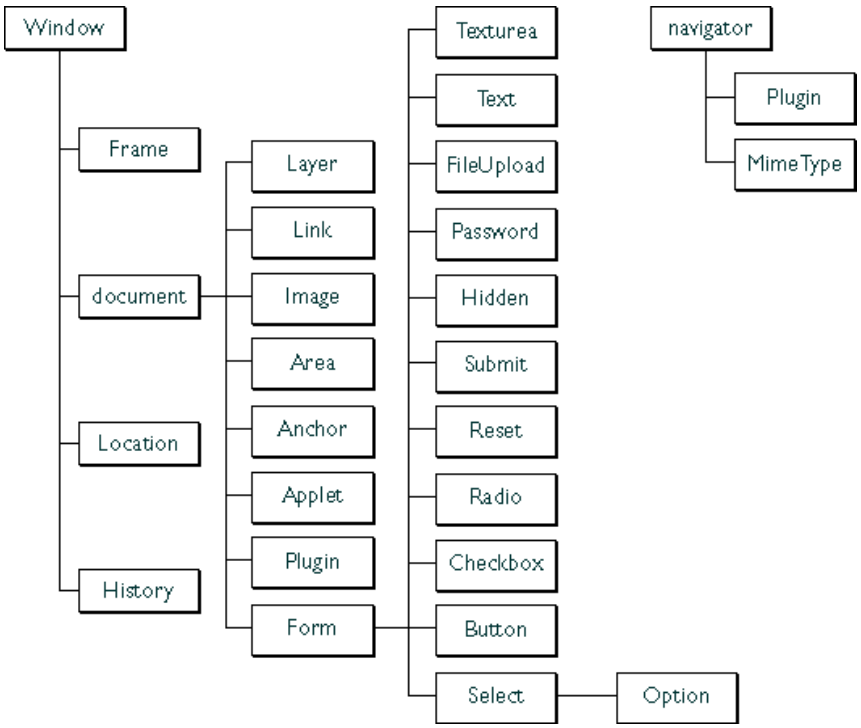
Accedere alle proprietà
Si usa la cosiddetta **dot-notation** in base alla quale si indica un oggetto e la proprietà a cui si è interessati separandoli con un punto:
Esempio, `var nome_esempio = persona.nome;`
Il tentativo di accedere ad una proprietà non definita in un oggetto non genera un errore, ma restituisce il valore undefined: nel seguente esempio, quindi, la variabile 'eta' avrà valore undefined: `var eta = persona.eta;`

Metodi
A differenza delle proprietà di un oggetto che rappresentano dati, i metodi rappresentano attività che un oggetto può compiere. Da un punto di vista sintattico, la definizione di un metodo per un oggetto è abbastanza simile alla definizione di una proprietà.
Vediamo un esempio:

```
function visualizzaNomeCognome() { return "Mario Rossi"; }
var stringa = visualizzaNomeCognome();
```

In questo caso si è definita una funzione che si limita a restituire una stringa e si è assegnato il nome della funzione ad una variabile stringa.

Oggetti predefiniti - DOM di una pagina Web
Rappresenta la gerarchia degli oggetti che compongono la pagina HTML. Fornisce un'interfaccia di programmazione per accedere, navigare e manipolare gli oggetti contenuti nella pagina HTML.
Le interfacce che consentono a JavaScript di rapportarsi con un browser sono chiamate DOM (*Document Object Model* in italiano *Modello a Oggetti del Documento*).



Oggetto window
E' l'oggetto base della struttura e rappresenta la scheda di navigazione corrente. Quando viene aperta una nuova scheda di navigazione, il browser genera un nuovo oggetto window che rappresenta appunto la scheda in cui verrà caricato e visualizzato il documento HTML. L'oggetto window rimane allocato finchè la finestra non viene chiusa.

I principali oggetti figli dell'oggetto window sono:
- document, documento html caricato nella finestra
- location, informazioni sulla url in corso
- history, informazioni relative alle url visitate prima e dopo della attuale
- navigator, oggetto di navigazione, utile ad esempio per eseguire dei redirect

window è l'oggetto predefinito del DOM, per cui i suoi metodi sono utilizzabili anche omettendo window stesso.
Esempio,
invece di scrivere il comando window.alert("Questa e' una prova");
si può scrivere semplicemente alert("Questa e' una prova");

Metodi dell'oggetto window

alert("Messaggio");	visualizza una finestra di messaggio con un unico pulsante OK non modificabile
confirm("Messaggio");	Finestra di conferma contenente i due pulsanti OK e ANNULLA, restituisce true se l'utente clicca su OK oppure false se l'utente clicca ANNULLA.
prompt("Messaggio");	rappresenta un InputBox con i pulsanti OK e ANNULLA, un secondo parametro opzionale indica un valore iniziale da assegnare al campo di immissione; cliccando su OK restituisce come stringa il valore inserito, cliccando su ANNULLA restituisce null.
open("file.html", "target");	valori di target, _self apre la nuova pagina nella stessa scheda _blank apre la nuova pagina in una nuova scheda (sempre schede diverse, DEFAULT) _blanck apre la nuova pagina in una nuova scheda (sempre la stessa)
parseInt(stringa);	converta una stringa in un numero intero, si ferma automaticamente al primo carattere non numerico
parseFloat(stringa);	converte una stringa in float
Nan	Not a Number, quando si esegue un'operazione matematica su una variabile non inizializzata (o non contenente numeri) JavaScript restituisce come risultato dell'espressione il valore Nan.
isNan(nome_var);	consente di testare se la variabile nome_var contiene il valore Nan, nel qual caso restituisce true

Eventi dell'oggetto window

window.onload = function() { ... }	utilizzabile direttamente all'interno del file.js

Oggetto window.document
Rappresenta il documento HTML che si sta visualizzando all'interno dell'oggetto window.

Proprietà

title	è il titolo della pagina impostato nella head dal tag title
lastModified	data e ora dell'ultima modifica della pagina

Eventi

onLoad	viene attivato al termine del caricamento dell'intera pagina
onUnload	richiamato quando si abbandona un documento (sostituito da un nuovo documento)
onResize	richiamato quando la finestra attiva viene ridimensionata

Metodo write(stringa)
Consente di scrivere dinamicamente il contenuto di una nuova pagina. La stringa può essere qualunque tag html. Se il metodo viene eseguito verso una pagina già caricata, write troverà il documento chiuso e provvederà a rimuoverlo sostituendolo con un documento vuoto in cui andrà a scrivere il contenuto di 'stringa'.
Esempio,

```
window.document.write("<title> Questo e' un TITOLO di prova </title> <p> Questo e' un PARAGRAFO di prova </p>");
```

Oggetti figli di document
Sono figli di document tutti i tag della pagina html.
Esempi,

```
document.documentElement        tag <html>
document.body                   tag <body>
document.nome_form.nome_elemento_form.value
    es,  var nome = document.modulo.txt_nome.value;
         assegna a nome il valore della casella di testo "txt_nome", del form di nome "modulo"
```

Proprietà, Metodi ed Eventi comuni a tutti i controlli

disabled	true / false
tabIndex	indice di tabulazione del controllo
form	rappresenta un puntatore alla form di appartenenza del controllo
focus()	assegna il fuoco al campo
onFocus	evento generato dopo che il campo ha ricevuto il fuoco
blur()	toglie il fuoco al campo passandolo al controllo con tabIndex successivo
onBlur	evento generato dopo che il campo ha perso il fuoco
onClick	indica il click del mouse
onChange	indica che il contenuto del controllo è cambiato
onMouseOver	ingresso del mouse sull'elemento
onMouseOut	uscita del mouse dall'elemento
onMouseMove	spostamento del mouse all'interno dell'elemento
onKeyPress	indica tasto premuto
onKeyDown	viene passato il codice fisico del tasto
onKeyUp	viene passato il codice fisico del tasto

TextBox (validi anche per TextBox multiline - TextArea)

value	oggetto di tipo string che rappresenta il contenuto del campo (anche nel caso delle TextArea)
onChange()	generato dopo che è cambiato il contenuto del campo; in realtà viene richiamato soltanto quando si abbandona il campo. Per intercettare i singoli caratteri occorre utilizzare onKeyPress
select()	metodo che seleziona il contenuto del campo
onSelect()	generato dopo che il contenuto del campo è stato (anche solo parzialmente) selezionato

Button

value	testo del pulsante
onClick	click sul pulsante, il metodo ".click()" consente di forzare un click sul pulsante

Vettori di controlli
in JavaScript tutti i controlli (dello stesso tipo) che hanno lo stesso "name" sono considerati come facenti parte di un vettore in cui il primo controllo del gruppo ha indice 0, il successivo ha indice 1 e così via... E' dunque possibile scandirli tramite un ciclo for cha va da 0 a N-1. La proprietà ".length" restituisce il numero di elementi appartenenti al gruppo.

Radio Button
Sono mutuamente esclusivi soltanto gli option button con lo stesso "name", che possono pertanto essere gestiti come un vettore di controlli.

.length	numero di pulsanti appartenenti al gruppo
[i].checked	true / false, indica se il radio button è selezionato; l'indice parte da 0
[i].value	valore nascosto restituito in corrispondenza del submit se il radio button è selezionato
onClick()	generato in corrispondenza di un click sulla casella (dopo che il cambio di stato è avvenuto)
[i].click()	forza un click sul pulsante che cambia pertanto di stato; occorre cambiare anche tutti gli altri

Per vedere qual è la voce selezionata occorre fare un ciclo sui singoli attributi checked e vedere per ognuno se è selezionato oppure no. Il checked può essere forzato da codice, in tal caso occorre però anche riportare a false il checked di tutti gli altri.

CheckBox
Presentano lo stesso funzionamento dei Radio Button, con l'unica differenza che non sono esclusivi

List Box

selectIndex	indice della (prima) voce selezionata
options	vettore delle opzioni appartenenti alla lista partendo da 0
length	numero di options presenti nella lista, identico a options.length
onChange	evento consigliato richiamato in corrispondenza della selezione di una nuova opzione
onClick	evento richiamato ad ogni singolo click sulla lista

options.length	numero di opzioni
options[i].selected	true / false a seconda che l'opzione sia selezionata o meno
options[i].text	testo visualizzato all'interno dell'opzione
options[i].value	valore nascosto interno all'opzione

L'oggetto <Select> dispone inoltre di un attributo "value" che, in corrispondenza della selezione di una voce, contiene il value del tag <options> attualmente selezionato. Viceversa gli option button, essendo costituiti da un vettore di oggetti privo di un contenitore esterno come <select>, non dispongono di una proprietà "value" riferita all'intero vettore, per cui per vedere quale option button è selezionato occorre fare un ciclo sui singoli attributi checked. Non è disponibile nemmeno selectIndex.

Esempio, somma di due numeri dati in input

Codice del file HTML

```
<!doctype html>
<html>
	<head>
		<title> Somma di due numeri </title>
		<script type="text/javascript" src="js_somma_due_numeri/index.js"> </script>
	</head>
	<body>
		<form method="POST">
			<p> Sito che calcola la somma di due numeri </p>	<br>
			<p>
				Inserire il primo numero
				<input type="text" name="input_primo_numero" size="20">
			</p>
			<p>
				Inserire il secondo numero
				<input type="text" name="input_secondo_numero" size="20">
			</p>
			<p>
				<input type="button" value="Calcola somma" onClick="calcola_somma(this)">
				<input type="text" name="visualizza_somma" size="20">
			</p>
		</form>
	</body>
</html>
```

Codice del file index.js

```
"use strict";
function calcola_somma(oggetto)
{
	var primo_numero, secondo_numero, somma;
	// l'istruzione parseFloat converte il contenuto fra parentesi da stringa a variabile di tipo float
	primo_numero = parseFloat(oggetto.form.input_primo_numero.value);
	secondo_numero = parseFloat(oggetto.form.input_secondo_numero.value);
	somma = primo_numero + secondo_numero;
	oggetto.form.visualizza_somma.value = somma;
}
```

Esempio, Validare il form di pagina 10 e inviarlo solo se tutti i campi sono stati compilati correttamente

Codice del file index.js

```
function Controlla_form()
{       // funzione che controlla gli elementi di un form 'modulo' presente nella pagina HTML
        // assegna a delle variabili i valori degli elementi del form 'modulo'
        var nome, cognome, paese, sesso, hobby0, hobby1, hobby2, hobby3, commento;
        nome = document.modulo.txt_nome.value;
        cognome = document.modulo.txt_cognome.value;
        paese = document.modulo.paese.options[document.modulo.paese.selectedIndex].value;
        sesso = document.modulo.sesso.value;

        // in questo caso invece di dichiarare più variabili è possibile dichiarare un vettore (array)
        //var hobby = document.modulo.hobby[0].value;                   // esempio per estrarre il valore
        hobby0 = document.modulo.hobby[0].checked;
        hobby1 = document.modulo.hobby[1].checked;
        hobby2 = document.modulo.hobby[2].checked;
        hobby3 = document.modulo.hobby[3].checked;

        commento = document.modulo.commento.value;

        // visualizzazione di prova del contenuto delle variaibili
        window.alert(nome+"\n"+cognome+"\n"+paese+"\n"+sesso+"\n"+hobby0+"\n"+hobby1+"\n"+hobby2+"\n"+hobby3+"\n"+commento);

        // controlla se i campi da compilare sono stati compilati, in caso negativo visualizza un messaggio di errore
        if ((nome=="")||(cognome=="")||(sesso=="")||((hobby0==false)&&(hobby1==false)&&(hobby2==false)&&(hobby3==false)))
                {
                        window.alert("Uno o piu' campi da compilare");
                        return false
                }
        else
                {
                        window.alert("Campi compilati con successo. Il modulo verra' inviato.");
                        return true;
                }
}
```

Codice del file HTML

all'inizio del form modificare la linea di codice come segue,

```
<form name="modulo" onSubmit="return Controlla_form();" method="post" action="esegui.php">
```

La parola chiave "this"

il puntatore "this" utilizzato nella pagina html all'interno di un gestore di evento, rappresenta un riferimento (puntatore) all'oggetto HTML in cui si trova.

Esempio,

```
<input type="radio" name="optElenco" value="xx" onClick="visualizza(this)">
.....
function visualizza (obj) {
                alert(obj.value);
}
```

this può essere applicato a qualsiasi controllo / tag. Ad esempio più TextBox sull'evento onChange possono richiamare una stessa funzione "Elabora()" passandogli this come parametro: onChange="Elabora(this)". Con la stessa tecnica si può anche passare "this.id" oppure l'intera form: onChange = "Elabora(this.form)".

Esempio, sempre in riferimento al form di pagina 10
Utilizzando la parola chiave "this", creare una funzione che permetta di visualizzare il valore di un oggetto e passargli come oggetti i checkbox dell'elemento hobby

file index.js

```
function visualizza_checkbox(oggetto)
{
        window.alert (oggetto.value);
        return 0;
}
```

file HTML

```
<!-- CHECKBOX -->
Hobby<br>
<input type="checkbox" name="hobby" value="S" onClick="visualizza_checkbox(this)"> Sport<br>
<input type="checkbox" name="hobby" value="L" onClick="visualizza_checkbox(this)"> Lettura<br>
<input type="checkbox" name="hobby" value="C" onClick="visualizza_checkbox(this)"> Cinema<br>
<input type="checkbox" name="hobby" value="I" onClick="visualizza_checkbox(this)"> Internet<br>
```

In questo esempio cliccando su un checkbox verrà visualizzato il valore corripondente utilizzando una sola funzione, confrontare il codice con l'esempio precedente e valutarne le differenze (soprattutto con le varie dichiarazioni delle variabili hobby0, hobby1, hobb2, hobby3).

Oggetto Math, funzioni matematiche in JavaScript

Proprietà

Le proprietà previste dall'oggetto rappresentano alcune costanti matematiche:

Math.E	costante di Eulero, base dei logaritmi naturali, 2.718
Math.LN2	logarimtonaturale di 2, 0,693
Math.LN10	logaritmo naturale di 10, 2,303
Math.PI	Il valore di pi greco, approssimativamente 3,14

Esempi,

```
if (x > Math.PI) {
        window.alert("Valore maggiore di pi greco");
}

window.alert("Il doppio di pi greco è " + Math.PI * 2)
```

I metodi

Math.abs(nome_var);	valore assoluto di nome_var
Math.max(Num1, Num2) ;	restituisce il maggiore fra due numeri
Math.min(Num1, Num2);	restituisce il minore fra due numeri
Math.pow(base, esponente);	elevamento a potenza, x^y
Math.sqrt(nome_var);	radice quadrata
Math.ceil(nome_var);	arrotondamento all'intero superiore
Math.floor(nome_var);	tronca all'intero inferiore
Math.round(nome_var)	arrotondamento all'intero più vicino, in base al valore dell'ultima cifra decimale
Math.random()	resistuisce un numero compreso fra 0 e 1, il randomize è automatico per generare un numero compreso fra min e max Math.floor(Math.random() * (max - min + 1) + min);

Esempio,

```
Math.ceil(3.4);          // 4
Math.floor(3.4)          // 3
Math.round(3.4)          // 3
Math.round(3.6)          // 4
```

SEGUONO ORA GLI ELENCHI DI UNA SERIE DI OGGETTI ED EVENTI PREDEFINITI

Oggetti predefiniti - Elenco

JavaScript ha diversi generi di oggetti predefiniti, in particolare Array, Boolean (booleani), Date (oggetti contenenti una data e un'ora), Function (funzioni), Math (oggetto contenente funzioni di uso nel calcolo matematico), Number (numeri), Object (oggetti) e String (stringhe). Altri oggetti sono gli "oggetti ospiti", definiti non dal linguaggio ma dall'ambiente di esecuzione. In un browser, i tipici oggetti ospite appartengono al DOM: window (finestra), form (maschera), link (collegamento) ecc.

Oltre a permettere la definizione di oggetti, il JavaScript mette a disposizione tutta una serie di oggetti che possono essere utilizzati per i propri script:

- Anchor
- Applet
- Area
- Array
- Base
- Basefont
- Body
- Button
- Checkbox
- Date
- Document
- Event
- File
- FileUpload
- Form
- Frame
- Frameset
- Function
- Hidden
- History
- Iframe
- Image
- Layer
- Link
- Location
- Math
- Meta
- Navigator
- Number
- Object
- Option
- Password
- Radio
- RegExp
- Reset
- Screen
- Select
- Style
- String
- Submit
- Table
- TableData
- TableHeader
- TableRow
- Text
- Textarea
- Window

Eventi - Elenco

Esistono varie categorie di eventi:

1. Eventi attivabili dai tasti del mouse
2. Eventi attivabili dai movimenti del mouse
3. Eventi attivabili dal trascinamento del mouse (drag and drop)
4. Eventi attivabili dall'utente con la tastiera
5. Eventi attivabili dalle modifiche dell'utente
6. Eventi legati al "fuoco"
7. Eventi attivabili dal caricamento degli oggetti
8. Eventi attivabili dai movimenti delle finestre
9. Eventi legati a particolari bottoni
10. Altri e nuovi tipi di eventi

Eventi attivabili dai tasti del mouse

Lista eventi:

1. *onClick*: attivato quando si clicca su un oggetto;
2. *onDblClick*: attivato con un doppio click;
3. *onMouseDown*: attivato quando si schiaccia il tasto sinistro del mouse;
4. *onMouseUp*: attivato quando si alza il tasto sinistro del mouse precedentemente schiacciato;
5. *onContextMenu*: attivato quando si clicca il tasto destro del mouse aprendo il ContextMenu.

Il doppio click è un evento che ingloba gli altri e, per la precisione, attiva in successione onmousedown, onmouseup, onclick.

Tag di applicazione

A, ADDRESS, APPLET, AREA, B, BDO, BIG, BLOCKQUOTE, BODY, BUTTON, CAPTION, CENTER, CITE, CODE, DD, DFN, DIR, DIV, DL, DT, EM, EMBED, FIELDSET, FONT, FORM, HR, I, IMG, INPUT type=button, INPUT type=checkbox, INPUT type=file, INPUT type=image, INPUT type=password, INPUT type=radio, INPUT type=reset, INPUT type=submit, INPUT type=text, KBD, LABEL, LEGEND, LI, LISTING, MAP, MARQUEE, MENU, NEXTID, NOBR, OBJECT, OL, P, PLAINTEXT, PRE, RT, RUBY, S, SAMP, SELECT, SMALL, SPAN, STRIKE, STRONG, SUB, SUP, TABLE, TBODY, TD, TEXTAREA, TFOOT, TH, THEAD, TR, TT, U, UL, VAR, XMP

Eventi attivabili dai movimenti del mouse

Lista eventi:

1. *onMouseOver*: attivato quando il mouse si muove su un oggetto;
2. *onMouseOut*: attivato quando il mouse si sposta da un oggetto;
3. *onMouseMove*: si muove il puntatore del mouse, ma poiché questo evento ricorre spesso (l'utilizzo del mouse è frequente), non è disponibile per default, ma solo abbinato con la cattura degli eventi, che si spiegherà in seguito.

Tag di applicazione

A, ADDRESS, APPLET, AREA, B, BDO, BIG, BLOCKQUOTE, BODY, BUTTON, CAPTION, CENTER, CITE, CODE, DD, DFN, DIR, DIV, DL, DT, EM, EMBED, FIELDSET, FONT, FORM, HR, I, IMG, INPUT type=button, INPUT type=checkbox, INPUT type=file, INPUT type=image, INPUT type=password, INPUT type=radio, INPUT type=reset, INPUT type=submit, INPUT type=text, KBD, LABEL, LEGEND, LI, LISTING, MAP, MARQUEE, MENU, NEXTID, NOBR, OBJECT, OL, P, PLAINTEXT, PRE, RT, RUBY, S, SAMP, SELECT, SMALL, SPAN, STRIKE, STRONG, SUB, SUP, TABLE, TBODY, TD, TEXTAREA, TFOOT, TH, THEAD, TR, TT, U, UL, VAR, XMP

Eventi attivabili dal trascinamento del mouse (drag and drop)
Lista eventi:

1. *onDragDrop*: evento attivato quando un utente trascina un oggetto sulla finestra del browser o quando rilascia un file sulla stessa;
2. *onMove*: attivato quando un oggetto muove una finestra o un frame;
3. *onDragStart*: evento attivato appena l'utente inizia a trascinare un oggetto;
4. *onDrag*: attivato quando il mouse trascina un oggetto o una selezione di testo nella finestra dello stesso browser o anche di un altro o anche sul Desktop;
5. *onDragEnter*: attivato appena l'utente trascina un oggetto su un obiettivo valido dello stesso o di un altro browser;
6. *onDragOver*: attivato quando l'utente trascina un oggetto su un obiettivo valido ad ospitarlo, ed è simile all'evento precedente, ma viene attivato dopo quello;
7. *onDragLeave*: attivato quando l'utente trascina un oggetto su un obiettivo adatto per ospitarlo, ma non vi viene rilasciato;
8. *onDragEnd*: attivato quando l'utente rilascia l'oggetto al termine del trascinamento.
9. *onDrop*: attivato quando il mouse si alza il tasto del mouse in seguito ad un'operazione di trascinamento;

Tag di applicazione
A, ADDRESS, APPLET, AREA, B, BDO, BIG, BLOCKQUOTE, BODY, BUTTON, CAPTION, CENTER, CITE, CODE, DD, DFN, DIR, DIV, DL, DT, EM, EMBED, FIELDSET, FONT, FORM, HR, I, IMG, INPUT type=button, INPUT type=checkbox, INPUT type=file, INPUT type=image, INPUT type=password, INPUT type=radio, INPUT type=reset, INPUT type=submit, INPUT type=text, KBD, LABEL, LEGEND, LI, LISTING, MAP, MARQUEE, MENU, NEXTID, NOBR, OBJECT, OL, P, PLAINTEXT, PRE, RT, RUBY, S, SAMP, SELECT, SMALL, SPAN, STRIKE, STRONG, SUB, SUP, TABLE, TBODY, TD, TEXTAREA, TFOOT, TH, THEAD, TR, TT, U, UL, VAR, XMP

Eventi attivabili dall'utente con la tastiera
Lista Eventi:

1. *onKeyPress*: evento attivato quando un utente preme e rilascia un tasto o anche quando lo tiene premuto;
2. *onKeyDown*: attivato quando viene premuto il tasto;
3. *onKeyUp*: evento attivato quando un tasto, che era stato premuto, viene rilasciato;
4. *onHelp*: attivato quando un utente preme il tasto F1;

Eventi attivabili dalle modifiche dell'utente
onChange
Attivato quando il contenuto di un campo di un form o modulo è modificato o non è più selezionato. Viene utilizzato anche con gli oggetti FileUpload, Select, Text, TextArea.
Esempio:

```
<input type="text" value="Enter email address" name="userEmail" onChange=validateInput(this.value) />
<script type="text/javascript">
  this.myForm.userEmail.focus();
  this.myForm.userEmail.select();

  function validateInput()
  {
    userInput = new String();
    userInput = this.myForm.userEmail.value;
    if (userInput.match("@"))
      alert("Thanks for your interest.");
    else
      alert("Please check your email details are correct before submitting");
  }
</script>
```

onCellChange
Attivato quando si modifica un elemento in un Database, per questa sua caratteristica ha un uso non propriamente legato a JavaScript;
onPropertyChange
Evento attivato quando cambia la proprietà di un elemento;
onReadyStateChange
Evento attivato quando lo stato del caricamento di un elemento cambia, l'evento è utile, ad esempio, per verificare che un elemento sia stato caricato.
Tag di applicazione
A, ADDRESS, APPLET, AREA, B, BDO, BIG, BLOCKQUOTE, BODY, BUTTON, CAPTION, CENTER, CITE, CODE, DD, DFN, DIR, DIV, DL, DT, EM, EMBED, FIELDSET, FONT, FORM, HR, I, IMG, INPUT type=button, INPUT type=checkbox, INPUT type=file, INPUT type=image, INPUT type=password, INPUT type=radio, INPUT type=reset, INPUT type=submit, INPUT type=text, KBD, LABEL, LEGEND, LI, LISTING, MAP, MARQUEE, MENU, NEXTID, NOBR, OBJECT, OL, P, PLAINTEXT, PRE, RT, RUBY, S, SAMP, SELECT, SMALL, SPAN, STRIKE, STRONG, SUB, SUP, TABLE, TBODY, TD, TEXTAREA, TFOOT, TH, THEAD, TR, TT, U, UL, VAR, XMP

Eventi legati al "fuoco"

onFocus

Questo handler è l'opposto di onBlur per cui si attiva quando l'utente entra in un campo;

onBlur

Viene attivato quando il puntatore del mouse o il cursore esce dalla finestra corrente utilizzando il mouse o il carattere TAB. Applicato ai moduli, invece, tale handler si avvia se si esce dal campo il cui tag contiene il controllo;

Esempio

```
Enter email address <input type="text" value="" name="userEmail" onBlur=addCheck()>
<script type="text/javascript">
function addCheck()
{
  alert("Please check your email details are correct before submitting")
}
</script>
```

onSelect

Attivabile quando si seleziona del testo all'interno di una casella di testo sia col mouse sia tenendo premuto SHIFT e selezionando con i tasti Freccia;

onSelectStart

Si attiva quando si inizia a selezionare un evento;

onbeforeEditFocus

Si attiva con un doppio click o con un click su un oggetto che ha già la selezione, quando questo è in DesignMode;

onLoseCapture

Si attiva quando un oggetto perde la cattura del mouse.

Tag di applicazione

A, ADDRESS, APPLET, AREA, B, BDO, BIG, BLOCKQUOTE, BODY, BUTTON, CAPTION, CENTER, CITE, CODE, DD, DFN, DIR, DIV, DL, DT, EM, EMBED, FIELDSET, FONT, FORM, HR, I, IMG, INPUT type=button, INPUT type=checkbox, INPUT type=file, INPUT type=image, INPUT type=password, INPUT type=radio, INPUT type=reset, INPUT type=submit, INPUT type=text, KBD, LABEL, LEGEND, LI, LISTING, MAP, MARQUEE, MENU, NEXTID, NOBR, OBJECT, OL, P, PLAINTEXT, PRE, RT, RUBY, S, SAMP, SELECT, SMALL, SPAN, STRIKE, STRONG, SUB, SUP, TABLE, TBODY, TD, TEXTAREA, TFOOT, TH, THEAD, TR, TT, U, UL, VAR, XMP

Eventi attivabili dal caricamento degli oggetti

onLoad

Questo handler funziona nel caricamento di oggetti, per lo più finestre e immagini;

onUnload

È l'opposto del precedente e funziona quando si lascia una finestra per caricarne un'altra o anche per ricaricare la stessa (col tasto refresh);

onAbort

L'Handler dell'evento onAbort permette di specificare del codice nel caso in cui l'utente blocchi il caricamento di un oggetto, o che si blocchi il caricamento di un'immagine.

Questo handler usa le seguenti proprietà dell'evento.

Esempio

```
<IMG NAME = "myPic" SRC = "images/myPic.gif" onAbort = "alert('Loading of image aborted!')">
```

onError

Si attiva quando il caricamento di un oggetto causa un errore, ma solo se questo è dovuto ad un errore di sintassi del codice e non del browser così funziona su un link errato di un'immagine della pagina, ma non su un link errato di caricamento di una pagina intera. Opera non gestisce questo evento, ormai obsoleto: per una corretta gestione degli errori si utilizza il costrutto try... catch;

onBeforeUnload

Questo handler funziona allo stesso modo di onUnload ma si carica in un momento prima;

onStop

Questo handler funziona quando si ferma il caricamento della pagina con il tasto stop del browser e dovrebbe funzionare anche allo stesso modo di onUnload caricandosi prima di questo ma dopo onBeforeUnload.

Tag di applicazione

1. onLoad Questo gestore è usato con i tag <BODY> e <FRAMESET> e da JavaScript 1.1anche con <IMG> mentre in Explorer occorre aggiungere anche i tag <SCRIPT>, <LINK>, <EMBED>, <APPLET>. In JavaScript 1.2 in Netscape si aggiunge anche il tag <LAYER>.
2. onUnload Questo gestore è usato con i tag <BODY> e <FRAMESET> anche in Internet Explorer.
3. onAbort Questo gestore è usato solo con il tag<IMG> anche in Internet Explorer.
4. onError Questo gestore è usato solo con il tag<IMG> e con Window mentre in Internet Explorer anche con <OBJECT> e <STYLE>.
5. onBeforeUnload Questo gestore è usato con i tag <BODY> anche in Internet Explorer.
6. onStop Questo gestore è usato con i tag <BODY> anche in Internet Explorer.

Eventi attivabili dai movimenti delle finestre

Lista Eventi:

1. onResize: Questo handler si attiva quando l'utente rimpicciolisce o ingrandisce una finestra o un frame o, in caso particolare per Explorer, un oggetto a cui siano stati fissati l'altezza e la larghezza o anche la posizione, come ad esempio un layer;
2. onScroll: attivato quando si effettua lo scrolling della pagina sia col mouse con i tasti PGUP e PGDOWN o anche con il metodo doScroll.

Tag di applicazione

A, ADDRESS, APPLET, B, BIG, BLOCKQUOTE, BUTTON, CENTER, CITE, CODE, custom, DD, DFN, DIR, DIV, DL, DT, EM, EMBED, FIELDSET, FORM, FRAME, Hn, HR, I, IMG, INPUT type=button, INPUT type=file, INPUT type=image, INPUT type=password, INPUT type=reset, INPUT type=submit, INPUT type=text, ISINDEX, KBD, LABEL, LEGEND, LI, LISTING, MARQUEE, MENU, OBJECT, OL, P, PRE, S, SAMP, SELECT, SMALL, SPAN, STRIKE, STRONG, SUB, SUP, TABLE, TEXTAREA, TT, U, UL, VAR, window, XMP

Eventi legati a particolari bottoni

1. onSubmit: Questo handler è attivato dal click su tasto di Invio di un form;
2. onReset: questo handler è attivato dal click su tasto di Annulla di un form.

Tag di applicazione

Handler applicabile solamente all'oggetto Form.

Caricare un file ".css" diverso per ogni browser

Creare il proprio sito scrivendo il relativo file ".css" e provarlo sul proprio browser di riferimento, dopodichè provare il sito anche su altri browser e valutare le differenze. Se vi sono delle differenze significative, allora può essere utile creare file ".css" diversi per ogni browser e richiamarli seguendo il successivo esempio.

Creare una copia del proprio file ".css" per ogni browser, le copie verranno rinominate nei seguenti modi: "chrome.css", "mozilla.css", "safari.css", "ie11.css" ("ie10.css", "ie9.css", ...).

Aggiungere il seguente script, nella sezione <head> del file HTML.

```
<script type="text/javascript">
	var chrome;
	var safari;
	var firefox;
	var ie;

	chrome = (/Chrome/.test(navigator.userAgent))?1:0;		// test(navigator.userAgent)), contiene la stringa /Chrome/?
								// SE SI assegna 1 alla variabile 'chrome', altrimenti assegna 0
	safari = (/Safari/.test(navigator.userAgent))?1:0;
	firefox = (/Firefox/.test(navigator.userAgent))?1:0;
	ie=(/MSIE (\d+\.\d+);/.test(navigator.userAgent))?1:0;

	if ((chrome == 1) || ((chrome == 1) && (safari == 1))) {
		document.write("<link rel='stylesheet' href='css/chrome.css' type='text/css'>");
	}

	if(firefox == 1)	document.write("<link rel='stylesheet' href='css/mozilla.css' type='text/css'>");
	if(safari == 1)		document.write("<link rel='stylesheet' href='css/safari.css' type='text/css'>");

	if (ie == 1){
		var str = navigator.userAgent;
		var index = str.search(/MSIE/);
		var vers = str.substring(index+5,index+6);

		if (vers == 11 )	document.write("<link rel='stylesheet' href='css/ie11.css' type='text/css'>");
		if (vers == 10 )	document.write("<link rel='stylesheet' href='css/ie8.css' type='text/css'>");
		if (vers == 9)		document.write("<link rel='stylesheet' href='css/ie7.css' type='text/css'>");

		// qui si possono aggiungere controlli per ulteriori versioni di Internet Explorer
	}

	// qui si possono aggiungere controlli per ulteriori browser
</script>
```

Bibliografia essenziale

- HTML.it
- it.wikipedia.org
- Tecnologie Multimediali, prof. Mana Roberto
- Fonti varie

Referenze ed elenco dei comandi del linguaggio CSS

Attributo	Valori	Descrizione
Elementi di testo		
text-align	left, right, center, justify	Allineamento orizzontale del testo o di un oggetto
vertical-align	top, middle, bottom	Allineamento verticale del testo o di un oggetto
text-decoration	none, underline, overline, line-through	Sottolinea in varie modalità un testo
text-transform	uppercase, lowercase, capitalize	Imposta un testo in maiuscolo o in minuscolo
text-indent	numerico	Testo rientrato espresso in px, pt, %, em, es. 12px
letter-spacing	numerico	Spazio tra una lettera e l'altra espresso... (come sopra)
line-height	numerico	Spazio tra una riga e l'altra espresso... (come sopra)
Elementi di formattazione del testo		
font-family	nome carattere	Carattere da utilizzare, es. Verdana
font-size	numerico	Grandezza testo espresso in px, pt, %, em, es. 12px
font-weight	normal, light, demi-light, extra-light, medium, bold, demi-bold, extra-bold	Imposta il carattere in vari spessori di grassetto
font-style	normal, italic, oblique	Imposta il carattere in corsivo
font-variant	normal, small-caps	Imposta maiuscole e minuscole alla stessa grandezza
first-letter	tutti i possibili valori di stile testuali	Proprietà della prima lettera di una riga
first-line	tutti i possibili valori di stile testuali	Proprietà della prima riga di un paragrafo
Link		
a:link	colore, sottolineatura, grandezza carattere ...	Impostazione di default dei link
a:hover	colore, sottolineatura, grandezza carattere ...	Azione al passaggio del mouse
a:active	colore, sottolineatura, grandezza carattere ...	Azione sul link attivo (al click)
a:visited	colore, sottolineatura, grandezza carattere ...	Azione del link visitato
Colori e sfondi		
color	colore (White - #FFFFFF - 250,250,250)	Imposta il colore degli elementi selezionati (testo, bordi ...)
background	transparent, colore..., url (file), repeat, scroll, position	Proprietà varie dello sfondo di una pagina o di una tabella
background-color	transparent, colore...	Imposta il colore dello sfondo di una pagina o di una tabella
background-image	none, url (file)	Seleziona un'immagine di sfondo per una pagina o una tabella
background-repeat	repeat, repeat-x, repeat-y, no-repeat	Imposta la ripetizione dell'immagine di sfondo...
background-attachment	scroll, fixed	Proprietà di scrolling dello sfondo
background-position	relative, absolute, allineamento orizzontale, allineamento verticale	Posizionamento dinamico dell'immagine di sfondo
Margini di una pagina o di un box		
margin	auto, numerico	Effetto rientrato espresso in px, pt, %, em, es. 12px (tutti i lati)
margin-top	auto, numerico	Effetto rientrato espresso in px, pt, %, em, es. 12px (superiore)
margin-left	auto, numerico	Effetto rientrato espresso in px, pt, %, em, es. 12px (sinistro)
margin-right	auto, numerico	Effetto rientrato espresso in px, pt, %, em, es. 12px (destro)
margin-bottom	auto, numerico	Effetto rientrato espresso in px, pt, %, em, es. 12px (inferiore)
Bordi di una pagina o di un box		
border	stile, dimensione, colore	Combina tutti i valori per tutti i bordi
border-top	stile, dimensione, colore	Combina tutti i valori per il bordo superiore
border-left	stile, dimensione, colore	Combina tutti i valori per il bordo sinistro
border-right	stile, dimensione, colore	Combina tutti i valori per il bordo destro
border-bottom	stile, dimensione, colore	Combina tutti i valori per il bordo inferiore
border-style	none, solid, double, groove, ridge, inset, outset	Imposta lo stile di tutti i bordi del box
border-top-style	none, solid, double, groove, ridge, inset, outset	Imposta lo stile del borde superiore del box
border-left-style	none, solid, double, groove, ridge, inset, outset	Imposta lo stile del borde sinistro del box
border-right-style	none, solid, double, groove, ridge, inset, outset	Imposta lo stile del borde destro del box
border-bottom-style	none, solid, double, groove, ridge, inset, outset	Imposta lo stile del borde inferiore del box
border-color	colore (White - #FFFFFF - 250,250,250)	Imposta il colore di tutti i bordi del box
border-top-color	colore (White - #FFFFFF - 250,250,250)	Imposta il colore del borde superiore del box
border-left-color	colore (White - #FFFFFF - 250,250,250)	Imposta il colore del borde sinistro del box
border-right-color	colore (White - #FFFFFF - 250,250,250)	Imposta il colore del borde destro del box
border-bottom-color	colore (White - #FFFFFF - 250,250,250)	Imposta il colore del borde inferiore del box
border-width	dimensione	Dimensione espressa in px, pt, %, em, es. 12px
border-top-width	dimensione	Dimensione espressa in px, pt, %, em, es. 12px
border-left-width	dimensione	Dimensione espressa in px, pt, %, em, es. 12px
border-right-width	dimensione	Dimensione espressa in px, pt, %, em, es. 12px
border-bottom-width	dimensione	Dimensione espressa in px, pt, %, em, es. 12px
float	none, left, right	Appiattisce i bordi
clear	none, left, right, both	Cancella i bordi
Spaziatura interna di una pagina o di un box		
padding	numerico	Espresso in px, pt, %, em, es. 12px
padding-top	numerico	Espresso in px, pt, %, em, es. 12px

padding-left padding-right padding-bottom	numerico numerico numerico	Espresso in px, pt, %, em, es. 12px Espresso in px, pt, %, em, es. 12px Espresso in px, pt, %, em, es. 12px
Visualizzazione e visibilità di tutti gli elementi		
display visibility overflow	none, block, inline, list-item visible, hidden, inherit auto, visible, hidden, scroll	Elemento NON visibile: NON considera lo spazio che occuperebbe Elemento NON visibile: considera lo spazio che occuperebbe Visualizzazione dinamica delle barre di scorrimento
Liste		
list-style-type list-style-image list-style-position list-style	none disk circle square decimal lower-roman upper-roman lower-alpha upper-alpha none, file immagine inside, outside tutti	Nessun puntatore Cerchio pieno Cerchio vuoto Quadrato pieno Numeri arabi (1. 2. 3. ...) Numeri romani in minuscolo (i. ii. iii. iv. ...) Numeri romani in maiuscolo (I. II. III. IV. ...) Lettere minuscole (a. b. c. ...) Lettere maiuscole (A. B. C. ...) Inserisce un'immagine al posto dei puntatori elencati Stabilisce la posizione dell'elenco sullo schermo Tutte le proprietà utilizzate, es. list-style: disk, inside, url(file.gif)
Dimensionamento degli elementi		
width height clip	numerico numerico rect, circle, polygon	Larghezza di un elemento espressa in px, pt, %, em, es. 12px Altezza di un elemento espressa in px, pt, %, em, es. 12px Forma di un elemento, quadrata, circolare, astratta
Posizionemanto degli elementi su di un documento		
position top left z-index	absolute, relative, static numerico numerico numerico, auto	Posizionamento di un elemento sulla pagina Posizionamento dall'alto espresso in px, pt, %, em, es. 12px Posizionamento dal basso espresso in px, pt, %, em, es. 12px Consente di sovrapporre elementi, i valori sono 1 e -1
Cursore		
cursor	auto default crosshair hand move e-resize ne-resize nw-resize n-resize se-resize sw-resize s-resize w-resize text wait help	Forma automatica, stabilita dallo sviluppatore per default Freccia normale Croce tipo mirino mano, classica di quando si passa su di un link Croce con frecce direzionali come quando si sposta una popup Freccia verso destra Freccia normale verso destra Freccia normale verso sinistra (come per default) Freccia verso l'alto Freccia normale in basso verso destra Freccia normale in basso verso sinistra Freccia verso il basso Freccia verso sinistra Linea orizzontale, classica di quando si passa su di un testo Clessidra Punto interrogativo
Proprietà di stampa		
page-break-before page-break-after	auto, always :: left, right auto, always :: left, right	Interruzione di riga prima della fine del testo da stampare Interruzione di riga dopo la fine del testo da stampare
Barre di scorrimento (stile & colore)		
scrollbar-base-color scrollbar-face-color scrollbar-face-color scrollbar-arrow-color scrollbar-highlight-color scrollbar-dark-shadow-color scrollbar-3d-light-color scrollbar-track-color	colore colore colore colore colore colore colore colore	Colora l'intermezzo tra le sfumature di tutte le sue componenti Colora la barra vera e propria, cioè la parte in movimento Colora l'ombra della barra di scorrimento vera e propria Colora le frecce della barra di scorrimento Colora il separatore tra la barra e la parte con le frecce Colora la parte esterna all'ombra della barra di scorrimento Colora la parte tridimensionale della parte contenete le frecce Colora la parte sottostante della barra, quella su cui si muove