Gli stili

Lo strumento degli stili sta piano piano soppiantando tutta una serie di attributi non solo per la formattazione del testo ma anche per l'aspetto estetico di tutti gli elementi di HTML, dalle immagini alle tabelle. HTML è stato pensato alle origini come un linguaggio **strutturale**, alieno da qualunque scopo attinente la **presentazione** di un documento. Per questo obiettivo, ovvero arricchire l'aspetto visuale ed estetico di una pagina, lo strumento designato sono appunto i **CSS** (**Cascading Style Sheets - Fogli di stile a cascata**). Lo scopo è separare il contenuto dalla presentazione: i CSS sono nati per modificare lo stile di elementi strutturali e non vanno intesi come un linguaggio grafico. Per queste cose esistono Flash o Photoshop.

Lo stile si può applicare in quattro modi diversi ad un elemento HTML:

I. fuori tag (*tag-less style*): un qualsiasi elemento di testo può assumere lo stile che vogliamo. Si usa il tag <**span**> ;
II. stile in linea (*inline style*) si cambia lo stile di tutto un tag (ad esempio una tabella o un paragrafo); si usa specificare **style** come attributo del tag;
III. a livello di documento (*document level style*): cambia tutti i tag o una parte di essi in tutto il documento; viene specificato all'interno dell'head e richiamato quando si usano i tag;
IV. come foglio di stile (*CSS: cascading Style Sheet*): si raccolgono tutte le specifiche dello stile in un file esterno che viene collegato al documento HTML con un riferimento di tipo <**link**>.

I. Fuori tag
La sintassi per lo stile fuori tag è la seguente:

```
<span style="font-family:Garamond,Serif; font-size:26pt; font-weight:800;
color:green; "> sto cambiando font, colore  </span> a qualche parola dello
stesso paragrafo!
```

Questo esempio mostra come sia possibile cambiare un font per una frase per poi tornare al font di prima. E' possibile anche cambiare il size oppure il solo colore della scritta oppure la grossezza.

Le cose da osservare nel codice sono due. Come valore di **style** (attributo del tag span) si possono dichiarare più regole di stile. Esse vanno separate dal punto e virgola. I due punti si usano invece per introdurre il valore della proprietà da impostare. Maggiori dettagli sulla sintassi saranno visti in seguito.

II. In linea
Si dichiara lo stile come attributo di un tag:

```
<p style=”font-family:Arial,Helvetica; font-style:italic”> Questo paragrafo
intero è in Arial (o Helvetica) corsivo.</p>
```

III. A livello di documento

I fogli incorporati sono quelli inseriti direttamente nel documento HTML tramite l'elemento **style**. La dichiarazione va posta all'interno della sezione <**head**>. Ad es. per cambiare lo stile relativo ai paragrafi scriveremo:

```
<head>
  <style type=”text/css”>
               p {
                       font-family: Garamond;
                       font-size: 26pt;
                }
  </style>
<head>
```

Se vogliamo cambiare lo stile solo ad alcuni paragrafi useremo il meccanismo delle classi (spiegato in seguito). Definiremo allora una classe “eleganza”:

```
<head>
  <style type=”text/css”>
               p.eleganza {
                       font-family: Garamond;
                       font-size: 26pt;
                }
  </style>
<head>
```

Per applicare tale stile ad un paragrafo si dichiara la sua appartenenza alla classe "eleganza":

```
<p class="eleganza"> paragrafo in Garamond a 26 punti! </p>
```

Si possono definire nuovi stili per tutti i tag standard.

IV. Foglio di stile

Si riporta lo stile in un file esterno, ad es. nel file `stile.css`. Supponiamo di dichiarare lo stile dell'esempio precedente:

```
p       {
                font-family:Garamond;
                font-size:26pt;
        }
```

Come prima cosa dovremo scrivere la dichiarazione dello stile in un file `stile.css`. Poi, all'interno del documento HTML, nella intestazione (ossia nell’head), dichiareremo la relazione con il foglio di stile:

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

A questo punto tutti i paragrafi saranno formattati secondo lo stile da noi definito.

Analogamente a prima possiamo anche definire delle classi (che ci permettono di modificare solo alcuni elementi in un file html). Per cui i) in un file `stile_2.css` scriveremo la dichiarazione della classe

```
p.eleganza      {
                        font-family: Garamond;
                        font-size: 26pt;
                }
```

ii) inseriremo il file `stile_2.css` nell'head del file html; iii) dichiareremo il tag di classe allo stesso modo dello stile a livello di documento:

```
<p class="eleganza">…</p>
```

L'elemento **`<link>`** presenta una serie di attributi:

1. rel: descrive il tipo di relazione tra il documento e il file collegato. E' **obbligatorio**. Per i CSS due sono i valori possibili: **stylesheet** e **alternate stylesheet**.

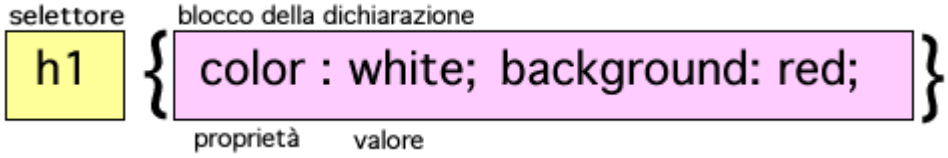
2. href: serve a definire l'URL assoluto o relativo del foglio di stile. E' **obbligatorio**.

3. type: identifica il tipo di dati da collegare. Per i CSS l'unico valore possibile è **text/css**. L'attributo è **obbligatorio**.

4. media: con questo attributo si identifica il supporto (schermo, stampa, etc) cui applicare un particolare foglio di stile. Attributo **opzionale**.

Sintassi degli stili

Un foglio di stile non è altro che un insieme di regole, accompagnate, volendo, da qualche nota di commento. Cos'è e com'è fatta una regola?



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

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

Il selettore serve a definire la parte del documento cui verrà applicata la regola. In questo caso, ad esempio, verranno formattati tutti gli elementi **<h1>**: lo sfondo sarà rosso, il colore del testo bianco. Il blocco delle dichiarazioni è delimitato rispetto al selettore e alle altre regole da **due parentesi graffe**. Al suo interno possono trovare posto più dichiarazioni. Esse sono sempre composte da una coppia:

- **proprietà**
- **valore**

La proprietà definisce un aspetto dell'elemento da modificare (margini, colore di sfondo, etc) secondo il valore espresso. Proprietà e valore devono essere separati dai **due punti**. Se in un blocco si definiscono più dichiarazioni esse vanno separate dal **punto e virgola**.

Abbiamo detto che una regola CSS si applica ad un selettore: parte del documento soggetto ad una specifica regola. Tutti gli elementi di HTML sono dei selettori. È possibile nei CSS raggruppare diversi elementi al fine di semplificare il codice. Gli elementi raggruppati vanno separati da una **virgola**.

Ad es:

```
h1, h2, h3 {background: red;}
```

Id e classi: due selettori speciali

In HTML esistono due attributi fondamentali applicabili a tutti gli elementi: sono **class** e **id**. Tuttavia dichiarare questi attributi a prescindere dai CSS non ha alcun senso e non modifica in alcun modo la presentazione della pagina. Infatti se assegniamo ad un paragrafo un attributo **class="testorosso"**:

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

non accadrà nulla: il problema è che il valore dell'attributo **class** deve trovare una corrispondenza in un foglio di stile. Dobbiamo quindi definire un CSS e creare un selettore di tipo **classe** al quale assegniamo il nome **testorosso**:

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

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

Come si vede dall'esempio per definire una classe si usa far precedere il nome da un semplice punto:

```
.nome_della_classe
```

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

```
<elemento>.nome_della_classe
```

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

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

lo stile verrà applicato solo ai paragrafi che presentino l'attributo **class="testorosso"**.

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

In un documento potrò avere senza problemi questa situazione:

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

La classe **.**testorosso presente nel CSS formatterà allo stesso modo il testo del paragrafo, del div e della tabella. Non sarà invece possibile attribuire a più elementi l'id #testorosso, per cui una situazione del genere non sarà possibile:

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

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

```
#nome_id
```

Con questa regola:

```
#blu {color: blue;}
```

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

```
<h1 id="blu">...</h1>
```

Le pseudo-classi

Una pseudo-classe non definisce un elemento ma un particolare stato di quest'ultimo: imposta uno stile per un elemento al verificarsi di certe condizioni. Ad es. la regola

```
a:link {color: blue;}
```

vuol dire: i collegamenti ipertestuali (<**a**>) che non siano stati visitati (**:link**) avranno il colore blue. Da qui risulta più chiaro il concetto espresso all'inizio: la pseudo-classe **:link** definisce lo stile

(colore blue) solo in una determinata situazione, ovvero quando il link non è stato attivato. Appena ciò dovesse avvenire, il testo non sarà più blu, perchè la condizione originaria è venuta meno.

Torniamo alla sintassi. La pseudo-classe (tutte iniziano con i **due punti**) segue senza spazi l'elemento. Subito dopo si crea nel modo consueto il blocco delle dichiarazioni:

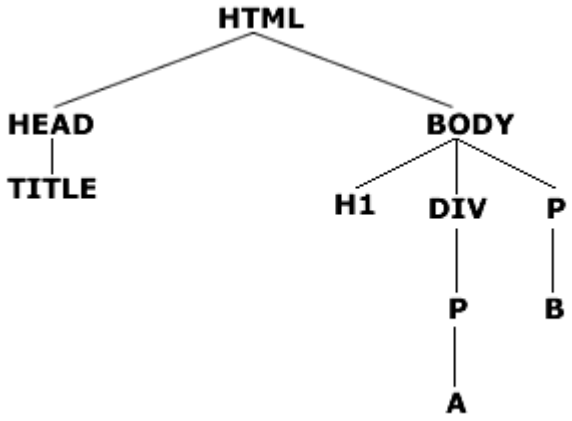
```
selettore:pseudo-classe {proprietà: valore;}
```

Ereditarietà

Secondo questo meccanismo le impostazioni stilistiche applicate ad un elemento ricadono anche sui suoi discendenti. Almeno fino a quando, per un elemento discendente, non si imposti esplicitamente un valore diverso per quella proprietà. Ad es. se impostiamo il colore rosso per il testo (**color: red;**) a livello dell'elemento **body** tutti gli altri elementi suoi discendenti erediteranno questa impostazione. Ma se ad un certo punto definiamo nel codice del CSS un selettore con la proprietà **color: black**; l'ereditarietà viene spezzata.

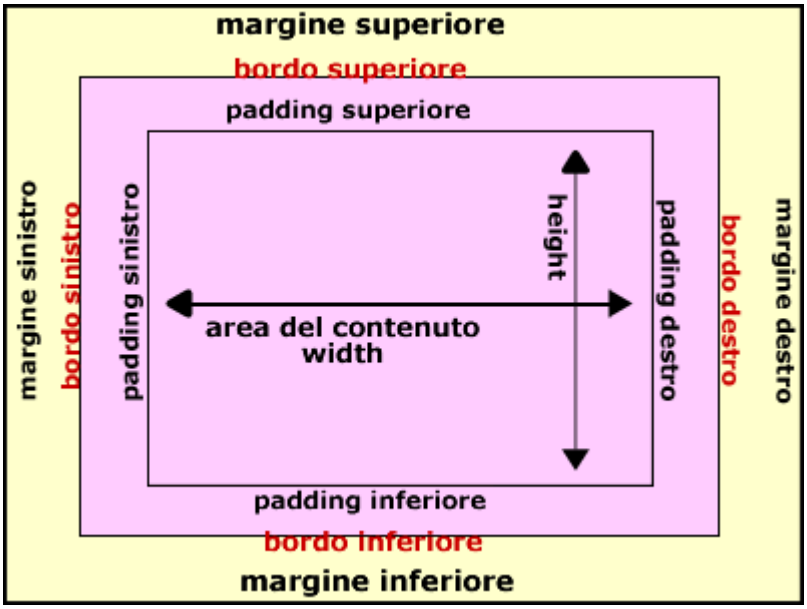
Non tutte le proprietà sono ereditate. In genere non lo sono quelle attinenti la formattazione del box model: margini, bordi, padding, background le più importanti. Il motivo è semplice. Ereditare un bordo è semplicemente senza senso. Se ne imposto uno, diciamo, per un paragrafo sarebbe assurdo che un elemento **<a>** o un testo in grassetto vengano circondati dallo stesso bordo!

Questa è la rappresentazione strutturale di un file HTML secondo il modello ad albero:



Il box model

Una pagina HTML non è altro che un insieme di box rettangolari, che si tratti di elementi **blocco (<p>, <h1>,…)** o di elementi **inline** non fa differenza. Tutto l'insieme di regole che gestisce l'aspetto visuale degli **elementi blocco** viene in genere riferito al cosiddetto **box model**. Ogni box comprende un certo numero di componenti di base, ciascuno modificabile con proprietà dei CSS. La figura qui sotto mostra visivamente tali componenti:



Partendo dall'interno abbiamo:

- **l'area del contenuto**. E' la zona in cui trova spazio il contenuto vero e proprio, testo, immagini, animazioni Flash. Le dimensioni orizzontali dell'area possono essere modificate con la proprietà **width**. Quelle verticali con **height**.
- **il padding**. E' uno spazio vuoto che può essere creato tra l'area del contenuto e il bordo dell'elemento. Come si vede dalla figura, se si imposta un **colore di sfondo** per un elemento questo si estende dall'area del contenuto alla zona di padding.
- **il bordo**. E' una linea di dimensione, stile e colore variabile che circonda la zona del padding e l'area del contenuto.
- **il margine**. E' uno spazio di dimensioni variabili che separa un dato elemento da quelli adiacenti.

Il **box model** è governato da una serie di regole di base concernenti la definizione di un box e il suo rapporto con gli altri elementi.

1. Larghezza del box
Bisogna distinguere tra la larghezza dell'area del contenuto e la larghezza effettiva di un box . La prima è data dal valore della proprietà **width**. La seconda è data da questa somma:

margine sinistro + bordo sinistro + padding sinistro + area del contenuto + padding destro + bordo destro + margine destro

2. Larghezza ed elemento contenitore
Se non si imposta alcun valore per la proprietà **width** o se il valore usato è **auto** la larghezza di un box è uguale a quella dell'area del contenuto dell'elemento contenitore.

3. Uso del valore auto
Solo per tre proprietà è possibile impostare il valore **auto**: margini, altezza e larghezza (width). L'effetto è quello di lasciar calcolare al browser l'ammontare di ciascuna di esse in base alla risoluzione dello schermo e alle dimensioni della finestra.

Solo i margini possono avere valori negativi. Ciò non è consentito per padding, bordi, altezza e larghezza.

Unità di misura

I valori di una proprietà non vanno **mai messi tra virgolette**; uniche eccezioni i valori espressi da stringhe di testo e i nomi dei font formati da più di una parola (esempio: "Times New Roman"). Quando si usano valori numerici con unità di misura, non bisogna lasciare spazio tra numero e sigla dell'unità. E' corretto quindi scrivere **15px** oppure **5em**.

Le unità di misura usate per definire dimensioni, spazi o distanze sono:

- **in (inches - pollici)**: classica misura del sistema metrico americano;
- **cm (centimetri)**;
- **pt (points - punti)**: unità di misura tipografica destinata essenzialmente a definire la dimensione dei font;
- **em (em-height)**: unità di misura spesso usata nei CSS. 1 em equivale all'altezza media di un carattere per un dato font. E' un unità di misura relativa.
- **px (pixels)**: unità di misura ideale su monitor. E' quella più usata e facile da comprendere.

I valori possono essere espressi in percentuale: il valore percentuale è da considerare sempre relativo rispetto ad un altro valore. Si esprime con un valore numerico seguito (senza spazi) dal segno di percentuale: **60%.**

Gestione del colore

Una delle proprietà più importanti è indubbiamente **color**. Quali sono i possibili valori per tale proprietà? Come si definiscono i vari colori? Ecco un elenco dei possibili modi per definire un colore.

- **parole chiave:** Si tratta di 16 parole che definiscono i colori della palette VGA standard di Windows:

  black | navy | blue | maroon | purple | green | red | teal | fuchsia | olive | gray | lime | aqua | silver | yellow | white

- **codici esadecimali**: le prime due lettere (o numeri) corrispondono ai valori per il colore rosso (**RED**), la seconda coppia fa riferimento al verde (**GREEN**), l'ultima al blue (**BLUE**). Ad es. #CC0000.
- Un altro modo per rappresentare i colori è quello di usare per i tre elementi base del sistema RGB una lista di valori in percentuale separati da una virgola. Per indicare il nero useremo, ad esempio rgb(0%, 0%, 0%) mentre per il bianco rgb(100%, 100%, 100%).

- Si definiscono i valori di rosso, verde e blue con tre valori compresi, rispettivamente, tra 0 e 255. Sistema ben noto a chi usa programmi di grafica. Il range 0-255 è l'equivalente decimale di quello esadecimale 00-FF visto in precedenza. Anche qui, i tre valori vanno separati da una virgola. Per il nero: rgb(0, 0, 0,)

La proprietà **color** si applica a tutti gli elementi ed è ereditata. Significa che se si imposta il colore per un elemento esso sarà ereditato da tutti gli elementi discendenti per cui non si definisca esplicitamente un altro colore. La sintassi è semplice:

```
selettore { color: <valore> }
```

Gestione dello sfondo

Ecco la lista delle proprietà riguardanti lo sfondo, applicabili a **tutti gli elementi (**ciascuna di essa definisce un solo, particolare aspetto relativo allo sfondo di un elemento**)**:

1. **background-color**: definisce il colore di sfondo di un elemento. Proprietà non ereditata. La sintassi è:

   ```
   selettore {background-color: <valore>;}
   ```

   Esempio: `p { background-color: #FFFFFF; }`

2. **background-image**: definisce l'URL di un'immagine da usare come sfondo di un elemento. Proprietà non ereditata. La sintassi è:

   ```
   selettore { background-image: url(<valore>); }
   ```

   Esempio: `body {background-image: url(sfondo.gif); }`

3. **background-repeat**: definisce la direzione in cui l'immagine di sfondo viene ripetuta. Proprietà non ereditata. La sintassi è:

   ```
   selettore {background-repeat: <valore>;}
   ```

   Valori: repeat (l'immagine viene ripetuta in orizzontale e verticale. E' il comportamento standard); **repeat-x** (l'immagine viene ripetuta solo in orizzontale); **repeat-y** (l'immagine viene ripetuta solo in verticale); **no-repeat** (l'immagine non viene ripetuta).

   Esempio:
   ```
   body { background-image: url(sfondo.gif);
   background-repeat: repeat; }
   ```

4. **background-attachment** si imposta il comportamento dell'immagine di sfondo rispetto all'elemento cui è applicata e all'intera finestra del browser. Si decide, in pratica, se essa deve scorrere insieme al contenuto o se deve invece rimanere fissa. Proprietà non ereditata. La sintassi è:

   ```
   selettore {background-attachment: <valore>;}
   ```

Valori: **scroll** (l'immagine scorre con il resto del documento quando si fa lo scrolling della pagina); **fixed** (l'immagine rimane fissa mentre il documento scorre)

Esempio: `body { background-image: url(back_400.gif); background-attachment: fixed }`

5. **background-position** Definisce il punto in cui verrà piazzata un'immagine di sfondo non ripetuta o da dove inizierà la ripetizione di una ripetuta. Proprietà non ereditata. La sintassi è:

```
selettore {background-position: <valore> o <valori>;}
```

Valori: i valori possibili sono diversi. Tutti però definiscono le coordinate di un punto sull'asse verticale e su quello orizzontale. Ciò può avvenire nei seguenti modi: i) con valori in **percentuale;** ii) con valori espressi con **unità di misura;** iii) con le parole chiave **top**, **left**, **bottom**, **right**

Esempio:

```
body {
background-image: url(back_400.gif);
background-repeat: no-repeat;
background-position: 50px 50px;
}
```

6. **background:** possiamo definire in un colpo solo tutti gli aspetti dello sfondo. Per essere valida, la dichiarazione non deve contenere necessariamente riferimenti a tutte le proprietà viste finora, ma **deve** contenere almeno la definizione del colore di sfondo. Sintassi

```
selettore {background: color image repeat
attachment position;}
```

Gestione del testo: alcune proprietà

Uno sguardo su alcune proprietà riguardanti il testo.

1. **font-family**: serve a impostare il tipo di carattere di una qualunque porzione di testo. Si applica a tutti gli elementi ed è ereditata. Sintassi:

```
selettore {font-family: <font 1>, <font2>, ....,<famiglia generica>;}
```

E' possibile definire dei font alternativi. Esempio:

```
div {font-family: Georgia, "Times New Roman", serif;}
```

2. **font-size**: definisce le dimensioni. Applicabile a tutti gli elementi ed ereditata. Sintassi:

```
selettore { font-size: <valore>; }
```

I valori delle dimensioni del testo possono essere espressi in vari modi. I diversi sistemi si possono distinguere a seconda che definiscano le dimensioni in senso assoluto o relativo. Sono valori assoluti:

- le sette parole chiave **xx-small**, **x-small**, **small**, **medium**, **large**, **x-large**, **xx-large**
- quelli espressi con le seguenti **unità di misura**: pixel (**px**), centimetri (**cm**), millimetri (**mm**), punti (**pt**).

Sono valori relativi:

- le parole chiave **smaller** e **larger**
- quelli espressi in **em** (em-height)
- quelli espressi in **percentuale**

Esempio:

```
    p {font-size: 12px;}
div.titolo {font-size: 50%;}
```

3. **font-weight**: Serve a definire la consistenza o "peso" visivo del testo. Si applica a tutti gli elementi ed è ereditata.. Sintassi:

```
selettore {font-weight: <valore>;}
```

Valori:

- **valori numerici** 100 - 200 - 300 - 400 - 500 - 600 - 700 - 800 - 900 ordinati in senso crescente (dal leggero al pesante)
- **normal**. Valore di default. E' l'aspetto normale del font ed equivale al valore 400
- **bold**. Il carattere acquista l'aspetto che definiamo in genere grassetto. Equivale a 700
- **bolder.** Misura relativa. Serve a specificare che una determinata porzione di testo dovrà apparire più pesante rispetto al testo dell'elemento parente
- **lighter**. Misura relativa. Il testo sarà più leggero di quello dell'elemento parente

Esempio:

```
p {font-weight: 900;}
```

4. **font-style**: imposta le caratteristiche del testo. Sintassi:

```
selettore {font-style: <valore>;}
```

Valori:

- **normal**: il testo mantiene il suo aspetto normale
- **italic**: formatta il testo in corsivo
- **oblique**: praticamente simile a italic

Esempio:

```
p {font-style: italic;}
```

5. **text-align**: imposta `l'allineamento del testo`. Sintassi:

```
selettore { text-align: <valore>; }
```

Valori:

- **left**. Allinea il testo a sinistra
- **right**. Allinea il testo a destra

- **center**. Centra il testo
- **justify**. Giustifica il testo

Esempio:

```
p {font-style: italic;}
```

Proprietà del box model

Un primo concetto fondamentale: **in genere l'altezza di un elemento è determinata dal suo contenuto**.

1. **height:** definisce la distanza tra il bordo superiore e quello inferiore di un elemento. Non è ereditata. Sintassi:

```
selettore {height: <valore>;}
```

Valori

- **un valore numerico con unità di misura.**
- **un valore in percentuale**. Il valore in percentuale si riferisce sempre all'altezza del blocco contenitore, purché esso abbia un'altezza esplicitamente dichiarata. Diversamente, la percentuale viene interpretata come **auto**.
- **auto**. L'altezza sarà quella determinata dal contenuto.

Esempi

```
div {height: 250px;}
p.blocco {height: 50%;}
```

2. **width:** definisce la larghezza dell'area del contenuto. Le dimensioni orizzontali di un elemento NON sono definite semplicemente dalla proprietà **width**. Si deve distinguere tra la larghezza effettiva di un elemento (i pixel di spazio che occupa sulla pagina) e la larghezza dell'area del contenuto. La prima è data dalla somma di questi valori:

margine sinistro + bordo sinistro + padding sinistro + area del contenuto + padding destro + bordo destro + margine destro

mentre la seconda è impostata tramite la proprietà width. Sintassi:

```
selettore {height: <valore>;}
```

Valori

- **auto**. Valore di default. Se non si impostano margini, bordi e padding la larghezza dell'elemento sarà uguale all'area del contenuto dell'elemento contenitore.
- **un valore numerico con unità di misura.**
- **un valore in percentuale**. La larghezza sarà calcolata rispetto a quella dell'elemento contenitore.

Esempi

```
p {width: 90px;}
div.box {width: 50%;}
```

Possiamo infatti definire il margine come la distanza tra il bordo di un elemento e gli elementi adiacenti. Sono cinque le proprietà con cui è possibile definire un margine. Quattro di esse sono singole e impostano la distanza per ciascun lato del box. L'ultima, **margin**, definisce con una sola dichiarazione tutte le impostazioni per i quattro lati.

3. **margin-bottom:** definisce la distanza tra il lato inferiore di un elemento e gli elementi adiacenti. Si applica a tutti gli elementi e non è ereditata. Sintassi

```
selettore {margin-bottom: <valore>;}
```

Valori

- **un valore numerico con unità di misura**.
- **un valore in percentuale.** Il valore è calcolato come percentuale rispetto alla larghezza (width) del blocco contenitore.
- **auto**. E' quasi sempre corrispondente a 0.

Esempi

```
p {margin-bottom: 10px;}
div.box {margin-bottom: 20%;}
```

4. **margin-left:** definisce la distanza lato sinistro di un elemento e gli elementi adiacenti. Si applica a tutti gli elementi e non è ereditata.. Sintassi

```
selettore {margin-left: <valore>;}
```

Valori

- **un valore numerico con unità di misura**.
- **un valore in percentuale.**
- **auto**.

Esempi

```
h1 {margin-left: 10%;}
img {margin-left: 20px;}
```

5. **margin-top:** definisce la distanza tra il lato superiore di un elemento e gli elementi adiacenti. Si applica a tutti gli elementi e non è ereditata. Sintassi

```
selettore {margin-top: <valore>;}
```

Valori

- **un valore numerico con unità di misura**.
- **un valore in percentuale.**
- **auto**.

Esempi

```
p.box {margin-top: 10%;}
img {margin-top: 20px;}
```

6. **margin-right:** definisce la distanza tra il lato destro di un elemento e gli elementi adiacenti. Si applica a tutti gli elementi e non è ereditata. Sintassi

```
selettore {margin-rigth: <valore>;}
```

Valori

- **un valore numerico con unità di misura**.
- **un valore in percentuale.**
- **auto**.

Esempi

```
h1 {margin-right: 10%;}
img {margin-right: 20px;}
```

7. **margin:** è possibile specificare i valori per tutti e quattro i lati di un elemento. Si applica a tutti gli elementi e non è ereditata. Sintassi

```
selettore {margin: top right bottom left}
```

Valori

- **un valore numerico con unità di misura**.
- **un valore in percentuale.**
- **auto:** la distanza sarà automaticamente calcolata rispetto alla larghezza dell'elemento contenitore.

Un uso interessante del valore **auto** per i lati sinistro e destro è quello che consente di centrare in tal modo un elemento rispetto alla pagina o al box contenitore.

Esempi

```
div {margin: 10px 15px 10px 20px;}
```

Un altro modo per creare spazio intorno ad un elemento è quello di usare il padding. Quando si usa il padding, lo spazio di distanza viene inserito al di qua dei bordi dell'elemento e non all'esterno. La cosa risulta evidente se usate un colore di sfondo per l'elemento diverso da quello della pagina. Nel caso del padding, lo spazio inserito avrà proprio il colore di sfondo dell'elemento, a differenza dei margini. Un'analogia rispetto ai margini sta nella sintassi. Anche qui quattro proprietà singole per i lati e una a sintassi abbreviata (**padding**).

8. **padding-bottom:** imposta l'ampiezza del padding sul lato inferiore di un elemento. Si applica a tutti gli elementi e non è ereditata. Sintassi

```
selettore {padding-bottom: <valore>;}
```

Valori

- **un valore numerico con unità di misura**. Il valore è espresso in termini assoluti.
- **un valore in percentuale.** Il valore è calcolato come percentuale rispetto alla larghezza (width) del blocco contenitore.

Esempi

```
p.box {padding-bottom: 10px;}
```

9. **padding-left:** imposta l'ampiezza del padding sul lato sinistro di un elemento. Si applica a tutti gli elementi e non è ereditata Sintassi

```
selettore {padding-left: <valore>;}
```

Valori

- **un valore numerico con unità di misura**.
- **un valore in percentuale.**

Esempi

```
p {padding-left: 10%;}
```

10. **padding-top:** imposta l'ampiezza del padding sul lato superiore di un elemento. Si applica a tutti gli elementi e non è ereditata Sintassi

```
selettore {padding-top: <valore>;}
```

Valori

- **un valore numerico con unità di misura**.
- **un valore in percentuale.**

Esempi

```
h1 {padding-top: 10px;}
```

11. **padding-right:** imposta l'ampiezza del padding sul lato destro di un elemento. Si applica a tutti gli elementi e non è ereditata. Sintassi

```
selettore {padding-rigth: <valore>;}
```

Valori

- **un valore numerico con unità di misura**.

- **un valore in percentuale.**

Esempi

```
p.box {padding-right: 10px;}
```

12. **padding:** proprietà a sintassi abbreviata. Serve a impostare i valori del padding per tutti e quattro i lati di un elemento. Sintassi

```
selettore {padding: top right bottom left}
```

Valori

- **un elenco di valori numerici con unità di misura**.
- **un elenco di valori in percentuale.**

Esempi

```
p {padding: 10px 20px;}
```

Per definire le proprietà dei **bordi** si possono usare proprietà a sintassi singola o a sintassi abbreviata. Le prime definiscono singoli aspetti di ciascuno dei quattro bordi. Le seconde consentono di riunire in una sola regola le diverse impostazioni. Ad es. per definire lo stile di un singolo bordo

Sintassi (con proprietà singole)

```
selettore {
        border-<lato>-color: <valore>;
        border-<lato>-style: <valore>;
        border-<lato>-width: <valore>;
}
```

Sintassi (abbreviata)

```
selettore { border-<lato>: <valore width> <valore style> <valore color>; }
```

In entrambi gli esempi di sintassi sostituite a **<lato>** uno degli indicatori dei quattro lati: **top, right, bottom o left.**

Si possono definire per il bordo tre aspetti:

1. **il colore (color)**
2. **lo stile (style)**
3. **lo spessore (width)**

I valori possibili per il **colore** sono:

- **un qualsiasi colore**
- **la parola chiave inherit**

Il colore può essere espresso in uno qualunque dei modi visti precedentemente. Se non si imposta un valore specifico, il colore sarà quello di primo piano impostato con la proprietà **color**.

Lo **stile** di un bordo può invece essere espresso con una delle seguenti parole chiave

- **none** (l'elemento non presenta alcun bordo e lo spessore equivale a 0); **hidden** (equivalente a none); **dotted**; **dashed**; **solid; double;**. **groove**; **ridge**; **inset**; **outset**

Infine abbiamo lo **spessore**. Esso può essere modificato secondo i seguenti valori:

- **un valore numerico con unità di misura**
- **thin**. Bordo sottile.
- **medium**. Bordo di medio spessore.
- **thick**. Bordo di largo spessore.

Esempi

```
div {
border-left-color: black;
border-left-style: solid;
border-left-width: 1px;
}
```

MA molto più comodo scrivere così:

```
div {border-left: 1px solid black;}
```

L'ultima proprietà a sintassi abbreviata è **border**. Con essa possiamo definire con una sola regola le impostazioni per i quattro bordi. Il suo uso è però limitato a un solo caso, peraltro molto comune: che i quattro bordi abbiano tutti lo stesso colore, lo stesso stile e lo stesso spessore.

Sintassi

```
selettore {border: <valore spessore> <valore stile> <valore colore>;}
```

Esempi

```
div {border: 2px solid black;}
```

Posizionamento degli elementi

position è la proprietà fondamentale per la gestione della posizione degli elementi, di cui determina la modalità di presentazione sulla pagina. Si applica a tutti gli elementi e non è ereditata.

Sintassi `<selettore> {position: <valore>;}`

Valori

- **static:** e’ il valore di default

- **absolute:** il box dell'elemento viene rimosso dal flusso del documento ed è posizionato in base alle coordinate fornite con le proprietà **top, left, right o bottom**. Il posizionamento avviene sempre rispetto al **box contenitore** dell'elemento.
- **fixed:** usando questo valore il box dell'elemento viene, come per **absolute**, sottratto al normale flusso del documento. La differenza sta nel fatto che per **fixed** il box contenitore è la finestra principale del borwser.
- **relative:** l'elemento viene posizionato relativamente al suo box contenitore. La posizione viene anche qui impostata con le proprietà **top, left, bottom, right**. Ma qui esse non indicano un punto preciso, ma l'ammontare dello spostamento in senso orizzontale e verticale rispetto al box contenitore.

Passiamo ora all'analisi delle proprietà che concretamente definiscono **dove** un elemento posizionato va a collocarsi, dal momento che che con **position** stabiliamo solo il **come**.

1. Per gli elementi posizionati con **absolute** o **fixed** definisce la distanza verticale rispetto al bordo superiore dell'elemento contenitore. Per gli elementi posizionati con **relative** stabilisce invece l'ammontare dello spostamento rispetto al lato superiore della posizione originaria. In questo caso, usando valori positivi il box viene spostato sotto, mentre con valori negativi lo spostamento avviene verso l'alto.

   Sintassi `<selettore> {top: <valore>;}`

   Valori

   - **un valore numerico con unità di misura**
   - **un valore in percentuale** La percentuale è relativa all'altezza dell'elemento contenitore.
   - **auto**

   Esempi

   ```
   div {top: 10px;}
     p {top: 10%;}
   ```

2. **left** Per gli elementi con posizione assoluta o fissa definisce la distanza dal bordo sinistro del box contenitore. Per quelli con posizione relativa stabilisce lo spostamento rispetto al lato sinistro della posizione originaria. Valori positivi spostano l'elemento verso destra, valori negativi verso sinistra.

   Sintassi

   <selettore> {left: <valore>;}

   Valori

   - **un valore numerico con unità di misura**
   - **un valore in percentuale** La percentuale è relativa alla larghezza dell'elemento contenitore.
   - **auto**

Esempi

div {left: 30px;}

3. bottom: Per i box con posizione assoluta o fissa definisce la distanza dal bordo inferiore dell'elemento contenitore. Per quelli posizionati relativamente lo spostamento rispetto al lato inferiore della posizione originaria.

   Sintassi

   <selettore> {bottom: valore>}

   Valori

   - **un valore numerico con unità di misura**
   - **un valore in percentuale**
   - **auto**

   Esempi

   div {bottom: 50px;}

4. Per i box con posizione assoluta o fissa definisce la distanza dal bordo destro dell'elemento contenitore. Per quelli posizionati relativamente lo spostamento rispetto al lato destro della posizione originaria.

   Sintassi

   <selettore> {right: valore;}

   Valori

   - **un valore numerico con unità di misura**
   - **un valore in percentuale**
   - **auto**

   Esempi

   div {right: 50px;}

   Per quantpo riguarda la definizione della posizione, va detto che essa è in ogni caso definita da due coordinate. In genere si usano le proprietà **top** e **left**.