2017

CSS E FOGLI DI STILE

Guida Css di base

Progetto di Alternanza Scuola-Lavoro – Anno scolastico 2016-2017 Istituto Tecnico Enrico Fermi, Siracusa

SikeWEB

Click different

Eleonora
SikeWEB
01/01/2017

Introduzione ai fogli di stile

Dietro il semplice acronimo CSS (Cascading Style Sheets - Fogli di stile a cascata) si nasconde uno dei fondamentali linguaggi standard del W3C. La sua storia cammina su binari paralleli rispetto a quelli di HTML, di cui vuole essere l'ideale complemento. Da sempre infatti, nelle intenzioni degli uomini del Consortium, l'HTML dovrebbe essere visto semplicemente 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.

Inserire i fogli di stile in un documento

Iniziamo dalle nozioni di base, rimanendo ancora in parte nel territorio dell'HTML. Vari sono i modi per inserire i fogli di stile CSS in un documento HTML. Per capire il meccanismo è necessario chiarire la fondamentale distinzione tra fogli esterni e interni.

CSS esterni e interni

E' **esterno** un foglio di stile definito in un file separato dal documento. Si tratta di semplici documenti di testo editabili anche con il Blocco Note ai quali si assegna l'estensione .css. Un foglio di stile si dice invece **interno** quando il suo codice è compreso in quello del documento HTML. A seconda che si lavori con un CSS esterno o interno variano sintassi e modalità di inserimento. Rispetto a queste diverse modalità si parla di fogli di stile collegati, incorporati o in linea.

Fogli collegati

Per caricare un foglio esterno in un documento si fa uso dell'elemento <link> . La dichiarazione va sempre collocata all'interno della sezione <head> del documento HTML:

```
<html>
<head>
        <title>Inserire i fogli di stile in un documento</title>
        <link rel="stylesheet" type="text/css" href="stile.css">
</head>
<body>
        ……
</body>
</html>
```

L'elemento <link> presenta una serie di attributi di cui è importante spiegare significato e funzione:

- ✓ Rel: descrive il tipo di relazione tra il documento e il file collegato. E' obbligatorio.
- ✓ Href: serve a definire l'URL assoluto o relativo del foglio di stile. E' obbligatorio.
- ✓ Type: identifica il tipo di dati da collegare. Per i CSS l'unico valore possibile è text/css. L'attributo è obbligatorio.

Fogli incorporati

I fogli incorporati sono quelli inseriti direttamente nel documento HTML tramite l'elemento <style>. Anche in questo caso la dichiarazione va posta all'interno della sezione <head>.

```
<html>
<head>
	<title>Inserire i fogli di stile in un documento</title>
	<style type="text/css">
		body {
		background: #FFFFCC;
		}
	</style>
</head>
<body>
	...
</body>
</html>
```

Come si vede il codice inizia con l'apertura del tag <style>, seguono le regole del CSS e la chiusura di </style>.

Fogli in linea

L'ultimo modo per formattare un elemento con un foglio di stile consiste nell'uso dell'attributo style. Esso fa parte della collezione di attributi HTML applicabili a tutti gli elementi. La dichiarazione avviene a livello dei singoli tag contenuti nella pagina e per questo si parla di fogli di stile in linea. La sintassi generica è la seguente:

```
<elemento style="regole_di_stile">
```

Se, ad esempio, si vuole formattare un titolo H1 in modo che abbia il testo di colore rosso scriveremo:

```
<h1 style="color: red;"> Titolo del mio documento</h1>
```

Le cose da osservare nel codice sono due. Come valore di style 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.

Consigli

A questo punto è giusto chiedersi: quando usare l'una o l'altra soluzione? Il punto di partenza nella risposta deve essere questo: i risultati nella formattazione del documento non cambiano. La giusta soluzione sarà

quindi quella richiesta dalla nostra applicazione. Il consiglio è semplice: pianificate, pensate in anticipo a quella che dovrà essere la struttura delle pagine del sito. Costruite per prima cosa un foglio di stile generico ed esterno, da applicare a tutte le pagine del sito. Esso conterrà le regole per formattare gli elementi o le sezioni presenti in tutte queste pagine.
Passate poi ad analizzare sezioni ed elementi presenti solo in certe pagine o che vogliate modificare solo in determinati casi. Supponete, ad esempio, di voler cambiare in rosso il colore di un titolo iniziale solo in una pagina delle 10 del vostro sito. Che fare? Semplice: usare uno stile incorporato solo in quella pagina:

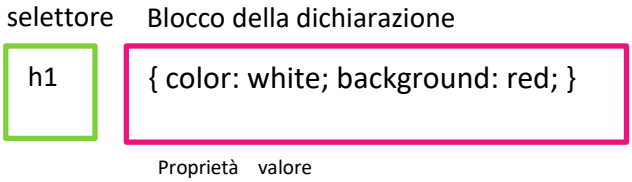
```
<style type="text/css"> h1 {color: red; } </style>
```

Per la legge che regola il meccanismo del cascading questo stile prevarrà su quello del CSS esterno.

Come è fatto un CSS: regole e commenti

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

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. Una limitazione fondamentale da rispettare è questa: per ogni dichiarazione non è possibile indicare più di una proprietà. Questa regola è pertanto errata:

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

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

Commenti

Per inserire parti di commento in un CSS racchiudetelo tra questi segni: /* come segno di apertura e */ come segno di chiusura.

Proprietà singole e a sintassi abbreviata

Nelle definizione delle regole è possibile fare uso di proprietà singole e proprietà a sintassi abbreviata. Le proprietà singole sono la maggior parte: impostano per un dato elemento o selettore un singolo aspetto. Con le shorthand properties, è possibile invece definire con una sola dichiarazione un insieme di proprietà. Chiariamo con un esempio.

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

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

La regola sarebbe questa:

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

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

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

I selettori

Fondamentalmente una regola CSS viene applicata ad un selettore. La parola parla da sé: si tratta di una semplice dichiarazione che serve a selezionare la parte o le parti di un documento soggette ad una specifica regola. Quella che segue è una lista commentata dei vari tipi di selettore. Per verificare i concetti abbiamo preparato per ciascun tipo un documento di esempio con codice e ulteriori spiegazioni.

Selettore di elementi (type selector)

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

Sintassi

```
h1 {color: #000000;}

p {background: white; font: 12px Verdana, arial, sans-serif;}

table {width: 200px;}
```

Raggruppare

È possibile nei CSS raggruppare diversi elementi al fine di semplificare il codice. Gli elementi raggruppati vanno separati da una virgola. Il raggruppamento è un'operazione molto conveniente. Pensate a questo scenario:

```
h1 {background: white;}

h2 {background: white;}

h3 {background: white;}
```

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

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

Selettore del discendente (descendant selector)

Serve a selezionare tutti gli elementi che nella struttura ad albero di un documento siano discendenti di un altro elemento specificato nella regola. Ricordiamo che un elemento è discendente di un altro se è contenuto al suo interno (annidato), a qualsiasi livello.

Sintassi

```
div p {color: black;}

p strong {color: red;}
```

Il selettore va letto per chiarezza da destra a sinistra. Nel primo esempio verranno selezionati tutti i paragrafi (<p>) discendenti di elementi . Nel secondo tutti gli elementi che si trovino all'interno di un paragrafo.

ID e classi: due selettori speciali

In HTML esistono due attributi fondamentali applicabili a tutti gli elementi: sono class e id. Ecco un esempio: qui abbiamo assegnato al paragrafo un attributo class="testorosso":

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

Il valore dell'attributo class deve trovare una corrispondenza in un foglio di stile. Qui, abbiamo definito un CSS incorporato creando un selettore di tipo classe e assegnando ad esso il nome testorosso:

```
<style type="text/css">

 .testorosso { font: 12px arial, Helvetica, sans-serif; color: #FF0000; }

</style>
```

Il testo del nostro paragrafo sarà ora formattato secondo i nostri desideri: testo rosso, carattere arial. dimensione di 12px. Lo stesso meccanismo è valido per i selettori di tipo ID.

Ma con una sola fondamentale differenza: è ad essa che dovete fare riferimento per scegliere se usare una classe o un ID.

In un documento HTML l'attributo id è usato per identificare in modo univoco un elemento. In pratica, se assegno ad un paragrafo l'id "testorosso", non potrò più usare questo valore nel resto della pagina. Di conseguenza, l'ID #testorosso dichiarato nel CSS trasformerà solo quel paragrafo specifico.

Una singola classe, al contrario, può essere assegnata a più elementi, anche dello stesso tipo.

In un documento potrò avere senza problemi questa situazione:

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

<div class="testorosso">....</div>

<table class="testorosso">...</table>

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

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

Non potrò invece scrivere:

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

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

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

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

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

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

Classe

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

```
.nome_della_classe
```

Questa è la sintassi di base. 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".

ID

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

```
#nome_id
```

Con questa regola (CSS):

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

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

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

Le pseudo classi

Una pseudo-classe non definisce infatti un elemento ma un particolare stato di quest'ultimo. In buona sostanza imposta uno stile per un elemento al verificarsi di certe condizioni. A livello sintattico le pseuso-classi non possono essere mai dichiarate da sole, ma per la loro stessa natura devono sempre appoggiarsi ad un selettore.

Il primo costrutto che esaminiamo è quello con un elemento semplice:

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

La regola 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 blue, 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.

Una pseudo-classe può anche essere associata a selettori di tipo classe. I costrutti possibili sono due.

1. La pseudo-classe segue la dichiarazione della classe:

   ```
   a.collegamento:link {color: green;}
   ```

   Come va letta questa regola? Avranno il testo verde (green) solo i link non visitati che abbiano come attributo class="collegamento". Sarà verde questo collegamento:

   ```
   <a href="pagina.htm" class="collegamento">
   ```

2. è consentita anche questa sintassi:

   ```
   a:link.collegamento
   ```

   in cui la classe segue la pseudo-classe. Significato e risultati sono comunque identici.

:first-child - :last-child

First-child seleziona e formatta un elemento che si trovi ad essere il primo elemento figlio di un altro elemento. Last-child seleziona e formatta un elemento che si trovi ad essere l'ultimo elemento figlio di un altro elemento.

Sintassi

```
<elemento>:first-child {<dichiarazioni>;}

<elemento>:last-child {<dichiarazioni>;}
```

Esempio:

```
p:first-child {color:red;}
```

La regola va così interpretata: assegna il colore rosso solo ai paragrafi che siano il primo elemento figlio di qualsiasi altro elemento.

Esaminando questa porzione di codice:

```
<div>
        <p>blah blah blah</p>
        <p>blah blah blah</p>
        <table>
                <tr>
                        <td>
                                <p>blah blah blah</p>
                                <p>blah blah blah</p>
                        </td>
                </tr>
        </table>
</div>
```

si capisce che solo i paragrafi in grassetto saranno rossi, perchè sono primi figli, rispettivamente, di elemento DIV e di un altro elemento td.

:link - :visited -:hover - :active

1. :link si applica solo all'elemento HTML <a> che abbia anche l'attributo href. Definisce lo stile per questo elemento quando il collegamento punta ad un sito o ad una pagina non ancora visitati.
2. :visited si applica solo all'elemento HTML <a> che abbia anche l'attributo href. Definisce lo stile per questo elemento quando il collegamento punta ad un sito o ad una pagina gia visitati.
3. :hover si applica solo all'elemento HTML <a> che abbia anche l'attributo href. Definisce lo stile per questo elemento il passaggio del mouse sul collegamento.

4. :active si applica solo all'elemento HTML <a> che abbia anche l'attributo href. Definisce lo stile per questo elemento il click sul collegamento.

L'attributo media

Grazie all'attributo Media siamo in grado di impostare un foglio di stile per ogni supporto su cui la nostra pagina verrà distribuita. Dove prima c'era unicamente un browser, domani potranno sempre più esserci palmari, cellulari e altri dispositivi. Per non parlare dei software usati da disabili come i browser vocali. Ciascuno di questi supporti presenta caratteristiche diverse in termini di memoria, ampiezza dello schermo e funzionalità. Riuscire ad adattare uno stile unico a tutti è praticamente impossibile. La soluzione ideale sta quindi nella creazione di fogli di stile ad hoc.

Sintassi

L'attributo media può accompagnare sia l'elemento <LINK> che l'elemento <STYLE>. Ecco due esempi di sintassi:

```
<link rel="stylesheet" type="text/css" media="print" href="print.css" />

<style type="text/css" media="screen">...</style>
```

Per sfruttare a fondo questa opzione è fondamentale conoscere i diversi valori possibili per l'attributo:

- ✓ all. Il CSS si applica a tutti i dispositivi di visualizzazione. Valore di default.
- ✓ screen. Valore usato per la resa sui normali browser web.
- ✓ print. Il CSS viene applicato in fase di stampa del documento.
- ✓ projection. Usato per presentazioni e proiezioni a tutto schermo.
- ✓ aural. Da usare per dispositivi come browser a sintesi vocale.
- ✓ braille. Il CSS viene usato per supporti basati sull'uso del braille.
- ✓ embossed. Per stampanti braille.
- ✓ handheld. Palmari e simili.
- ✓ tty. Dispositivi a carattere fisso.
- ✓ tv.Web-tv

@media

Lo stesso risultato (un CSS per ogni dispositivo) si può ottenere con la direttiva @media. La sintassi generica è questa:

```
@media <valore> {regole CSS}
```

@media va, quindi, seguito dal nome di uno dei supporti scelti come target specifico e dalle regole di stile racchiuse tra parentesi graffe.

Esempio:

```
<style type="text/css">
@media screen {
        h1 {color: black;}
        p {color: red;}
}
@media print {
        h1 {color: red;}
        p {color: black;}
}
 </style>
```

Anche questo costrutto va inserito, come si vede, all'interno del tag <STYLE>. Un uso più potente, però, è quello di inserire la direttiva nel codice di un foglio esterno. Immaginate di avere un foglio di stile per formattare le vostre pagine e che vogliate impostare colori diversi per un elemento a seconda che si visualizzi il testo sullo schermo o su carta stampata.

Invece di costruire due fogli di stile potete creare in un solo CSS esterno queste regole e diversificare l'aspetto dell'elemento:

```
@media print {
        h1 { color: black; }
}
@media screen {
        h1 { color: red; }
}

@media (max-width: 780px){
        .logo{
                display:none !important;
        }
}
@media (max-width: 780px) and (min-width : 551px){
        .logo{
                display:none !important;
        }
}
```

Come valore è possibile anche specificare max-width o min-width per fare in modo che la regola CSS venga eseguita soltanto per risoluzioni dello schermo pari a quelle specificate tra le parentesi tonde.

Valori ed unità di misura

Quali sono i tipi di valore e le unità di misura con cui è possibile impostare le tante proprietà dei CSS?

Prima di tutto, però, è opportuno spiegare due fondamentali regole di base.

1. 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").
2. 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. E' invece, sbagliato usare 15 px o 5 em. In questi casi la regola sarà ignorata o mal interpretata.

Unità per le dimensioni

Questa è la lista delle unità di misura usate per definire dimensioni, spazi o distanze.

- pt (points - punti): unità di misura tipografica destinata essenzialmente a definire la dimensione dei font.
- em (em-height): unità di misura spesso usata dagli autori 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.

Percentuale

Un valore espresso in percentuale è da considerare sempre relativo rispetto ad un altro valore, in genere quello espresso per l'elemento parente. Si esprime con un valore numerico seguito (senza spazi) dal segno di percentuale: 60% è pertanto corretto, 60 % no.

Colori

Ci sono diversi modi per esprimere il valore di un colore:

- valori esadecimali: #RRGGBB. Ad esempio #FFFFFF
- parole chiave. Si tratta di sedici keyword 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
- notazione RGB: Ad esempio rgb(0, 0, 0);

Valori URL

Si tratta di URL che puntano a documenti esterni (in genere immagini, come negli sfondi). Possono essere URL assoluti o relativi. In questo caso il path fa sempre riferimento alla posizione del foglio di stile e non del documento HTML. La definizione dell'indirizzo è sempre introdotta dalla parola chiave url e va inserita tra parentesi tonde senza virgolette. Esempio: url(immagini/sfondo.gif).

Ereditarietà, cascade, conflitti fra stili

Ereditarietà

Il primo concetto è quello di 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.

Se impostiamo, ad esempio, 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.

Peso e origine

Analizziamo i possibili conflitti tra gli stili e le regole. Tenteremo, in pratica, di rispondere a quesiti come questo. Se definisco queste regole in un CSS:

```
p {color: black;}

.testo {color: red;}
```

e in una pagina HTML scrivo questo codice:

```
<p class="testo">Testo del paragrafo</p>
```

perchè il testo del paragrafo sarà rosso e non nero? Perchè il selettore classe prevale su quello semplice con elemento? Il primo concetto da imparare è quello di peso. Si riferisce alla maggiore o minore importanza da assegnare a ciascuna regola.

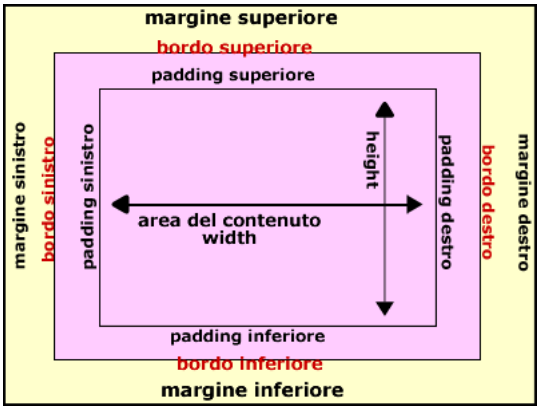
Le regole che verranno seguite sono le seguenti:

1. **Cascade**: gli stili in linea prevalgono su quelli incorporati che a loro volta prevalgono su quelli collegati.
2. **Specificità**: gli ID pesano più delle classi che pesano più dei singoli elementi.
3. **Importanza:** se una dichiarazione viene accompagnata dalla parola chiave !important essa balza al primo posto nell'ordine di applicazione a prescindere da peso, origine, specificità e ordine.

Il box model

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. 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.
- 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' lo spazio tra il bordo dell'elemento ed il suo esterno.

Gestione dello sfondo

Sin dalle origini di HTML è stato possibile definire un colore o un'immagine di sfondo per le nostre pagine web. Ecco la lista delle proprietà, applicabili a tutti gli elementi:

1. background-color
2. background-image
3. background-repeat
4. background-attachment
5. background-position

La proprietà background, invece, è una proprietà a sintassi abbreviata con cui possiamo definire sinteticamente e con una sola dichiarazione tutti i valori per lo sfondo.

background-color

Definisce il colore di sfondo di un elemento.

Sintassi

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

Valori

- un qualunque colore
- la parola chiave transparent

Esempi

```
body { background-color: white; }

p { background-color: #FFFFFF; }

.classe1 { background-color: rgb(0, 0, 0);}
```

background-image

Definisce l'URL di un'immagine da usare come sfondo di un elemento.

Sintassi

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

Valori

- un URL assoluto o relativo che punti ad un'immagine
- la parola chiave none. Valore di default.

Esempi

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

div body {background-image: url(http://www.server.it/images/sfondo.gif); }
```

background-repeat

Essa consente di definire la direzione in cui l'immagine di sfondo viene ripetuta.

Sintassi

```
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.

Esempi

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

div { background-image: url(sfondo.gif); background-repeat: repeat-x; }
```

background-attachment

Con questa proprietà 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.

Sintassi

```
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

Esempi

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

background-position

Proprietà un po' complessa. Definisce il punto in cui verrà piazzata un'immagine di sfondo non ripetuta o da dove inizierà la ripetizione di una ripetuta.

Sintassi

```
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:

- con valori in percentuale
- con valori espressi con unità di misura
- con le parole chiave top, left, bottom, right

L'esempio renderà tutto più chiaro.

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

Significa che l'immagine apparirà a 50px dal bordo superiore e a 50px da quello sinistro della finestra. Potevo usare invece dei pixel altre unità di misura o percentuali. Come al solito, la scelta delle percentuali mi assicura una maggiore affidabilità a risoluzioni diverse. Allo stesso modo potevo utilizzare le parole chiave.

Gestione del testo: proprietà di base

La proprietà che definiscono il modo in cui il testo appare sullo schermo sono tante Le proprietà di base sono quelle che definiscono i seguenti aspetti:

- il font da usare
- la sua dimensione
- la sua consistenza
- l'interlinea tra i paragrafi
- l'allineamento del testo
- la sua decorazione (sottolineature e simili)

font-family

La proprietà font-family serve a impostare il tipo di carattere di una qualunque porzione di testo.

```
p {font-family: arial, Verdana, sans-serif;}
```

Quando la pagina verrà caricata, il browser tenterà di usare il primo font della lista. Se questo non è disponibile userà il secondo. In mancanza anche di questo, verrà utilizzato il font principale della famiglia sans-serif presente sul sistema.

font-size

Insieme a font-family è la proprietà considerata essenziale nella definizione dell'aspetto del testo, di cui definisce le dimensioni.

Sintassi

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

Valori

I valori delle dimensioni del testo possono essere espressi in vari modi:

- con le sette parole chiave xx-small, x-small, small, medium, large, x-large, xx-large
- con le seguenti unità di misura: pixel (px), percentuale (%)o in em-height (em)

Esempi

```
p {font-size: 12px;}

 div.titolo {font-size: 50%;}

#div1 {font-size: large;}

h2 {font-size: 1.2em;}
```

font-weight

Serve a definire la consistenza o "peso" visivo del testo.

Sintassi

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

Valori

Il "peso" visivo di un carattere può essere espresso con una scala numerica o con parole chiave:

- 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

Esempi

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

div {font-weight: bold;}
```

font-style

Imposta le caratteristiche del testo in base ad uno di questi tre valori:

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

Sintassi

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

Esempi

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

Line-height

La proprietà, in realtà, serve a definire l'altezza di una riga di testo all'interno di un elemento blocco.

Sintassi

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

Valori

- normal. Il browser separerà le righe con uno spazio ritenuto "ragionevole". Dovrebbe corrispondere a un valore numerico compreso tra 1 e 1.2
- valore numerico. Usando valori numerici tipo 1.2, 1.3, 1.5 si ottiene questo risultato: l'altezza della riga sarà uguale alla dimensione del font moltiplicata per questo valore.
- un valore numerico con unità di misura. L'altezza della riga sarà uguale alla dimensione specificata.
- percentuale. L'altezza della riga viene calcolata come una percentuale della dimensione del font.
- Il consiglio è di non usare mai la percentuale, di valutare attentamente l'utilizzo di unità esplicite e di affidarsi senza problemi al valore numerico.

Esempi

```
p {line-height: 1.5;}
body {line-height: 15px;}
```

Text-align

La proprietà serve a impostare 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

Text-decoration

Imposta particolari decorazioni e stili per il testo

Sintassi

```
p {text-decoration: <valore> o <valori>;}
```

Valori

- none. Il testo non avrà alcuna decorazione particolare
- underline. Il testo sarà sottolineato
- overline Il testo avrà una linea superiore
- line-through. Il testo sarà attraversato da una linea orizzontale al centro
- blink. Testo lampeggiante

Esempi

```
p {text-decoration: none;}
a {text-decoration: underline;}
```

text-transform

La proprietà serve a cambiare gli attributi del testo relativamente a tre aspetti: maiuscolo, minuscolo, prima lettera maiuscola.

Sintassi

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

Valori

- none. Valore di default. Nessuna trasformazine viene applicata.
- capitalize. La prima lettera di ogni parola viene trasformata in maiuscolo.
- uppercase. Tutto il testo è maiuscolo.
- lowercase. Tutto il testo è minuscolo.

letter-spacing

Aumenta lo spazio tra le lettere di una parola.

Sintassi

```
selettore { letter-spacing: <valore>; }
```

Valori

- normal. Valore di default. Le lettere mantengono il loro spazio normale.
- un valore numerico con unità di misura. Le lettere saranno spaziate secondo la distanza impostata.

word-spacing

Proprietà complementare a letter-spacing. Serve ad aumentare lo spazio tra le parole comprese in un elemento.

Sintassi

```
selettore { word-spacing: <valore>; }
```

Valori

- normal. Valore di default. Le parole mantengono il loro spazio normale.
- un valore numerico con unità di misura. Le parole saranno spaziate secondo la distanza impostata.

Gestione delle dimensioni: altezza

In che modo nei CSS si impostano le dimensioni verticali di un elemento?

height

Questa proprietà definisce la distanza tra il bordo superiore e quello inferiore di un elemento.

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.

overflow

Esiste un modo per gestire il contenuto che superi i limiti imposti con height. Usare la proprietà overflow. Si applica a tutti gli elementi blocco.

Sintassi

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

Valori

- visible. Valore iniziale. Il contenuto eccedente rimane visibile (con i problemi visti sopra).
- hidden. Il contenuto eccedente non viene mostrato.
- scroll. Il browser crea barre di scorrimento che consentono di fruire del contenuto eccedente.
- auto. Il browser tratta il contenuto eccedente secondo le sue impostazioni predefinite. Di norma dovrebbe mostrare una barra di scorrimento laterale.

Gestione delle dimensioni: larghezza

La definizione della larghezza e più in generale la formattazione orizzontale degli elementi sono più problematiche rispetto all'altezza e alla formattazione verticale.

width

Le dimensioni orizzontali di un elemento sono date dalla combinazione di varie proprietà. Un errore macroscopico è ritenere che esse siano definite semplicemente dalla proprietà width. In pratica, dovete sempre 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
- La seconda è impostata tramite la proprietà width.

E' possibile ovviamente che i due valori coincidano. Accade quando di un particolare elemento non si impostino margini, paddding e bordi; o quando il valore di tali proprietà sia uguale a 0.

Sintassi

```
selettore {width: <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.

I margini

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, è una proprietà a sintassi abbreviata utile per definire con una sola dichiarazione tutte le impostazioni per i quattro lati.

- margin-bottom: Definisce la distanza tra il lato inferiore di un elemento e gli elementi adiacenti.
- margin-left: Definisce la distanza tra il lato sinistro di un elemento e gli elementi adiacenti. Si applica a tutti gli elementi
- margin-top:Definisce la distanza tra il lato superiore di un elemento e gli elementi adiacenti. Si applica a tutti gli elementi
- margin-right: Definisce la distanza tra il lato destro di un elemento e gli elementi adiacenti. Si applica a tutti gli elementi

Valori

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

Esempi

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

div { margin-top: 10px; margin-right: 15px; margin-bottom: 10px; margin-left: 20px; }
```

L'ordine di lettura va inteso in senso orario. Per cui: il primo valore si riferisce al lato superiore, il secondo a quello destro, il terzo al lato inferiore, il quarto a quello sinistro

Il padding

Un altro modo per creare spazio intorno ad un elemento è quello di usare il padding. Un'analogia rispetto ai margini sta nella sintassi. Anche qui quattro proprietà singole per i lati e una a sintassi abbreviata (padding).

- padding-bottom: Imposta l'ampiezza del padding sul lato inferiore di un elemento.
- padding-left: Imposta l'ampiezza del padding sul lato sinistro di un elemento.
- padding-top: Imposta l'ampiezza del padding sul lato superiore di un elemento.
- padding-right: Imposta l'ampiezza del padding sul lato destro di un elemento

Valori

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

Esempi

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

I bordi

In linea di massima possiamo suddividere le proprietà in due categorie: singole e 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.

- Sono proprietà singole:border-top-color, border-top-style, border-top-width, border-bottom-color, border-bottom-style, border-bottom-width, border-right-color, border-right-style, border-right-width, border-left-color, border-left-style, border-left-width.
- Sono proprietà a sintassi abbreviata:border, border-bottom, border-top, border-right, border-left, border-color, border-style, border-width

Come si vede dall'elenco delle proprietà di ciascun lato si possono definire per il bordo tre aspetti:

- il colore (color)
- lo stile (style)
- lo spessore (width)

Lo stile di un bordo può 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

Il colore può essere espresso in uno qualunque dei modi visti (esadecimale, rgb, parole chiave).

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.

Esempio

```
div {border-left: 1px solid black;}
div {
border-width: 1px 2px 1px 2px;
border-style: solid;
border-color: red;
}
```

Le liste

list-style-image

Definisce l'URL di un'immagine da usare come marcatore di un list-item. Si applica agli elementi LI e a quelli per i quali si imposti la proprietà display sul valore list-item.

Sintassi

```
<selettore> {list-style-image: url(<url_immagine>);}
```

Valori

- un URL assoluto o relativo che punti ad un'immagine.
- none. Non viene usata nessuna immagine.

list-style-type

Definisce l'aspetto del marcatore. Si applica agli elementi LI e a quelli per i quali si imposti la proprietà display sul valore list-item.

Sintassi

```
<selettore> {list-style-type: <valore>;}
```

Valori

La scelta dei valori possibili è lunghissima.

- disc: un cerchietto pieno colorato. Il colore può essere modificato per tutti i tipi con la proprietà color.
- circle: un cerchietto vuoto.
- square: un quadratino
- decimal: sistema di conteggio decimale 1, 2, 3, ...
- decimal-leading-zero: sistema di conteggio decimale ma con la prima cifra preceduta da 0. 01, 02, 03, ...
- lower-roman: cifre romane in minuscolo. i, ii, iii, iv, ...
- upper-roman: cifre romane in maiuscolo. I, II, III, IV, ...
- lower-alpha: lettere ASCII minuscole. a, b, c, ...
- upper-alpha: lettere ASCII maiuscole. A, B, C, ...
- lower-latin: simile a lower-alpha
- upper-latin: simile a upper-alpha
- lower-greek: lettere minuscole dell'alfabeto greco antico.

Due proprietà speciali: display, float

display

definisce il trattamento e la presentazione di un elemento. Fin quando non la si usa ognuno segue la sua natura e il suo comportamento standard, ma con display possiamo intervenire su di esso e in certi casi ribaltarlo.

Sintassi

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

Valori

La lista dei possibili valori è lunghissima. Solo alcuni di essi sono però di normale utilizzo e supportati:

- inline. Valore di default. L'elemento assume le caratteristiche degli elementi inline.
- block. L'elemento viene trattato come un elemento blocco.
- none. L'elemento non viene mostrato. O meglio: è come se non fosse nemmeno presente nel documento, in quanto non genera alcun box. Diversa la proprietà visibility:hidden, che invece si limita a nascondere l'elemento.

float

Con questa proprietà è possibile rimuovere un elemento dal normale flusso del documento e spostarlo 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 di float.

Sintassi

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

Valori

- left. L'elemento viene spostato sul lato sinistro del box contenitore, il contenuto scorre a destra.
- right. L'elemento viene spostato sul lato destro, il contenuto scorre a sinistra.
- none. Valore iniziale e di default in mancanza di una dichiarazione esplicita. L'elemento mantiene la sua posizione normale.

Una nota importantissima. Se usate il float con le immagini non avete problemi perchè esse hanno una dimensione intrinseca che il browser riconosce. Ma se lo applicate ad altri elementi dovete esplicitamente impostare una dimensione orizzontale con la proprietà width.

Posizionamento degli elementi

position

position è la proprietà fondamentale per la gestione della posizione degli elementi, di cui determina la modalità di presentazione sulla pagina.

Sintassi

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

Valori

- static: E' il valore di default, quello predefinito per tutti gli elementi non posizionati secondo un altro metodo. Rappresenta la posizione normale che ciascuno di essi occupa nel flusso del documento.
- Absolute: L'elemento, o meglio, 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
- Fixed: un box posizionato con fixed non scorre con il resto del documento. Rimane, appunto, fisso al suo posto
- Relative: L'elemento viene posizionato relativamente al suo box contenitore.