

Materiale concesso da HTML.it: <http://www.html.it>

Tutti i capitoli sono a cura di : Cesare Lamanna

Rispetto al corso originale presente sul sito sono stati tolti alcuni paragrafi (sul sito denominati lezione) ed alcuni di quelli qui riportati sono stati ridotti.

Versione originale del corso:

<http://css.html.it/guide/leggi/2/guida-css-di-base/>

Fogli di Stile

1. INTRODUZIONE.....	4
A COSA SERVONO.....	4
GUIDA ALLA CONSULTAZIONE	6
2. CLASSIFICAZIONE DEGLI ELEMENTI.....	7
ELEMENTI BLOCCO E ELEMENTI IN LINEA	7
ELEMENTI RIMPIAZZATI (REPLACED ELEMENTS)	8
STRUTTURA AD ALBERO DI UN DOCUMENTO	8
3. INSERIRE I FOGLI DI STILE IN UN DOCUMENTO	11
CSS ESTERNI E INTERNI	11
FOGLI COLLEGATI	11
USARE @IMPORT	12
FOGLI INCORPORATI.....	13
FOGLI IN LINEA	14
CONSIGLI	14
4. L'ATTRIBUTO MEDIA	16
SINTASSI	16
ESEMPI.....	17
5. COME È FATTO UN CSS: REGOLE E COMMENTI.....	18
COM'È FATTA UNA REGOLA	18
COMMENTI.....	19
PROPRIETÀ SINGOLE E A SINTASSI ABBREVIATA	19
6. I SELETTORI.....	21
SELETTORE DI ELEMENTI (TYPE SELECTOR)	21
RAGGRUPPARE	21
SELETTORE UNIVERSALE (UNIVERSAL SELECTOR).....	22

SELETTORE DEL DISCENDENTE (DESCENDANT SELECTOR)	22
LINKS.....	23
7. ID E CLASSI: DUE SELETTORI SPECIALI	24
CLASSE	25
ID	26
8. VALORI E UNITÀ DI MISURA	27
UNITÀ PER LE DIMENSIONI	27
PERCENTUALE.....	28
COLORI	28
STRINGHE	28
VALORI URI	28
UNITÀ PER GLI ANGOLI.....	28
UNITÀ DI TEMPO.....	29
UNITÀ DI FREQUENZA	29
9. EREDITARIETÀ, CASCADE, CONFLITTI TRA STILI.....	30
EREDITARIETÀ	30
PESO E ORIGINE	31
SPECIFICITÀ	31
IL CONCETTO DI CASCADE	32
IMPORTANZA.....	33
10. IL BOX MODEL.....	34
LARGHEZZA DEL BOX.....	35
LARGHEZZA ED ELEMENTO CONTENITORE	35
USO DEL VALORE AUTO.....	36
MARGINI VERTICALI E ORIZZONTALI TRA GLI ELEMENTI	36
LINKS.....	36
11. GESTIONE DEL COLORE	37
PAROLE CHIAVE	37
#RRGGBB	37
#RGB	37
RGB(RRR%, GGG%, BBB%)	38
RGB(RRR, GGG, BBB)	38
LA PROPRIETÀ COLOR	38
LINKS.....	39
12. GESTIONE DELLO SFONDO	40
BACKGROUND-COLOR	40
BACKGROUND-IMAGE.....	41
BACKGROUND	42
13. GESTIONE DEL TESTO: PROPRIETÀ DI BASE	43
FONT-FAMILY	43
FONT-SIZE	44
FONT-WEIGHT.....	45
FONT-STYLE	46

LINE-HEIGHT	46
FONT	47
USARE I FONT DI SISTEMA	48
TEXT-ALIGN	48
TEXT-DECORATION	49
LINKS.....	49
14. GESTIONE DELLE DIMENSIONI: ALTEZZA.....	50
HEIGHT	50
OVERFLOW	51
MIN-HEIGHT	51
MAX-HEIGHT	52
15. GESTIONE DELLE DIMENSIONI: LARGHEZZA	53
WIDTH	53
MIN-WIDTH.....	54
MAX-WIDTH	55
LINKS.....	55
16. I MARGINI	56
MARGIN-BOTTOM	56
MARGIN-LEFT	57
MARGIN-TOP	57
MARGIN-RIGHT.....	57
MARGIN	58
17. IL PADDING.....	60
PADDING-BOTTOM.....	60
PADDING-LEFT	60
PADDING-TOP	61
PADDING-RIGHT	61
PADDING	61
18. I BORDI.....	63
DEFINIRE LO STILE DI UN SINGOLO BORDO	63
DEFINIRE LO STILE DEI QUATTRO BORDI.....	65
USARE LA PROPRIETÀ BORDER	66
19. AGIRE SULLE TABELLE	67
TABLE-LAYOUT	67
BORDER-COLLAPSE	68
BORDER-SPACING	68
EMPTY-CELLS.....	69
CAPTION-SIDE.....	69
20. RISORSE ON LINE	71
21. BIBLIOGRAFIA.....	73

1. Introduzione

Dietro il semplice acronimo CSS (**Cascading Style Sheets - Fogli di stile a cascata**) si nasconde uno dei fondamentali linguaggi standard del W3C (<http://www.w3c.org>). 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, HTML, così come la sua recente evoluzione, XHTML, 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. L'ideale perseguito da anni si può sintetizzare con una nota espressione: separare il contenuto dalla presentazione.

La prima specifica ufficiale di CSS (**CSS1** (<http://www.w3.org/tr/REC-CSS1>)) risale al dicembre del 1996. Nel maggio 1998 è stata la volta della seconda versione: **CSS2** (<http://www.w3.org/tr/REC-CSS2>). Niente stravolgimenti, ma molte aggiunte rispetto alla prima. **CSS2** non è altro che **CSS1** più alcune nuove proprietà, valori di proprietà e definizioni per stili non canonici come quelli rivolti alla stampa o alla definizione di contenuti audio. E' attualmente allo stato di Working Draft la nuova specifica **CSS3** (<http://www.w3.org/tr/css3-roadmap/>).

A cosa servono

Inizio questa sezione con un'affermazione perentoria: se siete seriamente interessati alle tecnologie web non potete ignorare i CSS. Chi conosce un minimo di HTML è pienamente consapevole dei limiti di questo linguaggio sul lato della pura presentazione. A dire il vero, non è giusto parlare di limiti: HTML non è nato per questo. E' stato però piegato a fare cose che intrinsecamente non era in grado di fare. Bene: quasi tutto quello che con HTML non potete realizzare, riuscirete a farlo con i fogli di stile.

Finalmente, ad esempio, potrete dare al testo delle vostre pagine un aspetto da word-processor: non solo con il **colore** o i **font** che preferite, ma con un sistema di **interlinea** pratico e funzionale, con le **decorazioni** che desiderate, riuscendo a **spaziare** lettere e parole, impostando **stili diversi** per titoli e paragrafi, sfruttando i benefici dell'**indentatura** o della **giustificazione**. Ancora, potrete finalmente distanziare gli elementi della vostra pagina in maniera semplice e intuitiva con un potente meccanismo di gestione dei **margini**. Tonnellate di gif trasparenti usate per spaziare possono essere finalmente gettate nel cestino del vostro computer. E se i margini non bastano sarete in grado ora di aggiungere bellissimi **bordi** non solo alle tabelle, ma a tutti gli elementi di una pagina.

Per non parlare degli **sfondi**: li potrete applicare a quello che volete. E dimenticate quelle brutte micro-textures. Finalmente potrete sbattere su una pagina, magari ben piazzato al centro, quel vostro bellissimo disegno di 600x400 pixel senza doverlo necessariamente vedere trasformato in orribili riquadri ripetuti sulla pagina. Ora siete voi a decidere come usare un'immagine di sfondo: la potete ripetere in una sola direzione, in due o per niente! Facile e comodo.

La cosa più bella è che la **gestione del sito** non sarà mai più un incubo. Se quel bellissimo disegno di prima vi stufa, non dovrete più andare a modificare una per una 300 pagine! I CSS sono separati dal documento. Aprite un foglio di stile, cambiate l'immagine e il gioco è fatto. Il risultato sono pagine più leggere e facili da modificare. Milioni di byte di banda risparmiati per la gioia degli utenti.

Se poi avete a cuore l'accessibilità i CSS sono uno strumento portentoso, anche grazie al fatto di poter essere gestiti con linguaggi di scripting avanzati in grado di modificare con un solo click l'aspetto di una pagina.

Spero e credo che queste semplici considerazioni possano invogliarvi a procedere nella lettura di questa guida di base in cui troverete la spiegazione di tutte le proprietà e dei meccanismi che regolano questo meraviglioso strumento. Quando avrete superato il livello di apprendistato non dimenticate di dare un'occhiata alla sezione PRO di HTML.it. Tanti articoli di approfondimento vi aspettano per farvi sfruttare al meglio le potenzialità del linguaggio.

Guida alla consultazione

Un primo avviso: per una lettura consapevole della guida è richiesta una buona conoscenza di HTML e XHTML. In fin dei conti i CSS agiscono su questi linguaggi e non hanno alcun senso se separati da essi.

La guida è suddivisa in tre parti:

- 1.** Nella prima affronteremo i concetti di base, ovvero i meccanismi e i costrutti che regolano il linguaggio e la sua interazione con (X)HTML.
- 2.** La seconda parte comprende l'elenco ragionato delle proprietà CSS2 suddivise per ambito di applicazione (sfondo, colore, testo, etc)
- 3.** Nella terza forniremo suggerimenti e proposte di approfondimento con bibliografia e webografia.

Quasi tutte le lezioni sono corredate di pagine di esempio. Esse non vanno considerate come un semplice materiale accessorio, ma sono **parte integrante e costitutiva** della guida. Contengono sempre note di commento ulteriore ed esempi di codice sugli argomenti trattati. Abbiamo anzi pensato di farvi un regalo. Le abbiamo raccolte, insieme ai CSS che ne definiscono lo stile, in un file zip. Scaricatele. Studiatevi comodamente il codice (X)HTML e quello dei fogli di stile. Giocateci un pò. Modificate le proprietà e fate i vostri esperimenti. Vi assicuro che è il modo migliore per imparare.

Scarica il file "**test.zip** (<http://html.it/guide/esempi/css/test.zip>)".

Nel proporre il riferimento alle proprietà ci siamo limitati all'esposizione della loro sintassi e del funzionamento di base. Non vengono affrontati in questa sede problemi di compatibilità o di bug di singoli browser nel trattamento di tali proprietà (salvo casi importanti). Rimandiamo alle risorse citate nell'ultima parte per approfondimenti sulla questione.

Un aspetto che tengo a sottolineare è quello della traduzione: abbiamo tentato di tradurre in italiano i nomi di molte parole chiave, ma abbiamo sempre dato il corrispondente in inglese. Il motivo è semplice: la bibliografia e le risorse online sono quasi tutte in quella lingua e una non consapevolezza di certi termini potrebbe evidentemente spiazzarvi.

A questo punto non mi rimane che esprimervi il canonico "Buona lettura". Aggiungendo, come sempre, la mia disponibilità e massima apertura per aiuti, suggerimenti, critiche e consigli. Buona lettura!

2. Classificazione degli elementi

La prima lezione di questa guida potrebbe spiazzarvi. Non parleremo di CSS, ma di (X)HTML. O meglio, riprenderemo alcuni aspetti di questo linguaggio che sono propedeutici per una migliore comprensione del meccanismo di funzionamento dei CSS. Sapere bene su che **cosa si interviene con i fogli di stile** è un passo necessario, visto che le cose di cui parleremo, specie con l'avvento e l'abuso degli editor visuali, sono spesso trascurate o misconosciute dai più. Se mi passate la metafora, possiamo dire che faremo come un bravo chirurgo che prima di imparare gli strumenti deve conoscere bene il corpo umano per operare con successo e senza fare danni. Inizieremo con la classificazione degli elementi.

Elementi blocco e elementi in linea

Osservate una pagina (X)HTML tentando di non pensare al contenuto ma solo alla sua struttura. Mettete in atto, insomma, una specie di processo di astrazione. Per aiutarvi ecco un'immagine (<http://html.it/guide/img/css/guidacss002.jpg>). Una pagina (X)HTML, per iniziare, non è altro che un insieme di rettangoli disposti sullo schermo di un monitor. Non importa che si tratti di paragrafi, titoli, tabelle o immagini: sempre di box rettangolari si tratta.

Nell'immagine potete però osservare che non tutti i box sono uguali. Alcuni contengono altri box al loro interno. Altri sono invece contenuti all'interno dei primi e se si trovano (come si trovano) in mezzo a del testo notate che esso scorre intorno senza interrompere il suo flusso e senza andare a capo. Avete nell'immagine la rappresentazione visiva, anche se un pò semplificata, della fondamentale distinzione tra gli elementi (X)HTML, quella tra elementi **blocco** ed elementi **inline**.

Gli elementi **blocco** sono i box che possono contenere altri elementi, sia di tipo blocco che di tipo **inline**. Quando un elemento blocco è inserito nel documento

viene automaticamente creata una nuova riga nel flusso del documento.

Provate a inserire in una pagina (X)HTML queste due righe di codice:

```
<h1>Titolo</h1>
```

```
<p>Paragrafo</p>
```

Le parole "titolo" e "paragrafo" appariranno su due righe diverse, perchè

<H1> e **<P>** sono elementi blocco.

Gli elementi **inline** non possono contenere elementi blocco, ma solo altri elementi inline (oltre che, ovviamente, il loro stesso tipo di contenuto, essenzialmente testo). Nell'immagine sono i rettangoli con il bordo verde.

Come si può notare, quando sono inseriti nel documento non danno origine ad una nuova riga.

Una terza categoria è quella degli **elementi lista**. Comprende in pratica solo l'elemento **LI (list item)**.

Elementi rimpiazzati (replaced elements)

Un'altra distinzione da ricordare è quella tra **elementi rimpiazzati** ed **elementi non rimpiazzati**. I primi sono elementi di cui uno user agent (il "motore" e la mente di un browser) conosce solo le dimensioni intrinseche. Ovvero, quelli in cui altezza e larghezza sono definite dall'elemento stesso e non da ciò che lo circonda. L'esempio più tipico di elemento rimpiazzato è IMG (un'immagine). Altri elementi rimpiazzati sono: INPUT, TEXTAREA, select e OBJECT. Tutti gli altri elementi sono in genere considerati **non rimpiazzati**. La distinzione è importante perchè per alcune proprietà è diverso il trattamento tra l'una e l'altra categoria, mentre per altre il supporto è solo per la prima, ma non per la seconda.

Struttura ad albero di un documento

Un altro concetto fondamentale che dovrete assimilare è quello della struttura ad albero di un documento. Il meccanismo fondamentale dei CSS è infatti **l'ereditarietà**. Esso fa sì che molte proprietà impostate per un elemento siano automaticamente ereditate dai suoi discendenti. Sapersi districare nella struttura ad albero significa padroneggiare bene questo meccanismo e sfruttare al meglio la potenza del linguaggio.

Presentiamo subito un frammento di codice HTML:

```
<html>
```

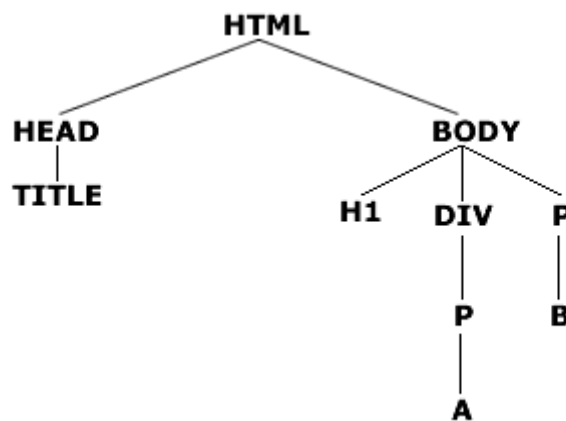
```
<head>
```

```

<title>Struttura del documento</title>
</head>
<body>
  <h1>Titolo</h1>
  <div>
    <p>Primo <a href="pagina.htm">paragrafo</a></p>
  </div>
  <p>Secondo<b>paragrafo</b></p>
</body>
</html>

```

Questa è la sua rappresentazione strutturale secondo il modello ad albero:



Il documento è in buona sostanza una perfetta forma di gerarchia ordinata in cui tutti gli elementi hanno tra di loro una relazione del tipo **genitore-figlio** (**parent-child** in inglese, imparate la dicitura perchè nei linguaggi come DOM o Javascript ci si riferisce agli ordini della gerarchia proprio con questi termini.). Ogni elemento è genitore e/o figlio di un altro.

Un elemento si dice **genitore (parent)** quando contiene altri elementi. Si dice **figlio (child)** quando è racchiuso in un altro elemento. In base a queste semplici indicazioni possiamo analizzare il nostro documento.

BODY, ad esempio è figlio di **HTML**, ma è anche genitore di **H1**, **DIV** e **P**.

Quest'ultimo è a sua volta genitore di un elemento **B**.

Ovviamente, se osservate bene, potreste concludere che anche **BODY** è in qualche modo genitore di **B**. Non è esattamente così. Introduciamo ora un'altra distinzione, mutuata anch'essa dal linguaggio degli alberi genealogici, quella tra **antenato** (ingl: **ancestor**) e **discendente** (ingl: **descendant**). Semplice da capire. La relazione parent-child è valida solo se tra un elemento e l'altro si

scende di un livello. Esattamente come in un albero familiare si indica la relazione tra padre e figlio. Pertanto possiamo dire che **HEAD** è figlio di **HTML**, che **A** è figlio di **P**, etc. tra **DIV** e **A**, invece si scende di due livelli: diciamo allora che **DIV** è un **antenato** di **A** e che questo è rispetto al primo un **discendente**.

C'è un solo elemento che racchiude tutti e non è racchiuso: **HTML**.

Continuando con la metafora familiare potremmo dire che è il capostipite, ma in termini tecnici si dice che esso è **l'elemento radice** (ingl: **root**). E qui spazziamo il campo da un possibile fraintendimento: l'elemento radice di un documento (X)HTML non è **BODY**. E che **HTML** non sia una semplice dichiarazione ma sia trattato alla stregua di qualunque altro elemento lo potete testare aprendo questa pagina

(http://html.it/guide/esempi/css/test/html_body.html) con Explorer 6, Mozilla, Opera 6 o Explorer 5 Mac.

Come si vede abbiamo iniziato una guida dedicata a un linguaggio prettamente visuale tornando alle origini di HTML, ovvero al concetto di **struttura**. Non è un capriccio, è semplicemente dare il giusto inquadramento ai CSS: che sono nati per modificare lo stile di elementi strutturali e non vanno intesi come un linguaggio grafico. Per queste cose esistono Flash o Photoshop.

3. Inserire i fogli di stile in un documento

Iniziamo il nostro percorso dalle fondamentali nozioni di base, rimanendo ancora in parte nel territorio di (X)HTML. Se CSS è un solo linguaggio, vari sono i modi per inserire i fogli di stile CSS in un documento. 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 o TextEdit 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. 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 esistono due possibilità. La prima e più compatibile è quella che fa uso dell'elemento **<LINK>**. La dichiarazione va sempre collocata all'interno della sezione **<HEAD>** del documento (X)HTML:

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

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

- 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**. . Tipicamente si utilizza il valore **stylesheet**, per approfondimenti vedi <http://css.html.it/guide/lezione/18/fogli-di-stile-alternativi/>.
- 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**. L'argomento sarà approfondito nella prossima lezione.

Usare @import

Un altro modo per caricare CSS esterni è usare la direttiva **@import** all'interno dell'elemento **<STYLE>**:

```
<style>
  @import url(stile.css);
</style>
```

Questo sistema è uno dei modi più sicuri per risolvere problemi di compatibilità tra vecchi e nuovi browser. Il CSS va collegato definendo un URL assoluto o relativo da racchiudere tra parentesi tonde e che la dichiarazione deve chiudersi con un punto e virgola.

Un principio fondamentale è che all'interno del tag **<STYLE>** **@import** deve essere la prima regola definita. In pratica, se si scrive così la direttiva non funzionerà e il CSS non sarà caricato in quanto **@import** è preceduta da una regola che applica uno stile all'elemento **<H1>**:

```
<style type="text/css">
h1 { color: black; }
@import url(foglio_di_stile.css);
</style>
```

Il modo giusto è dunque questo:

```
<style type="text/css">
@import url(foglio_di_stile.css);
h1 { color: black; }
</style>
```

È possibile invece importare all'interno di un singolo tag **<STYLE>** più fogli di stile:

```
<style type="text/css">
@import url(foglio_di_stile.css);
@import url(foglio_di_stile2.css);
</style>
```

Le regole di tutti i CSS collegati in questo modo saranno applicate al documento secondo l'ordine e i criteri stabiliti dalle norme sull'importanza e la specificità che vedremo tra poco in una prossima lezione.

Un uso interessante di @import è che può essere usata anche all'interno di un foglio di stile per incorporare un altro CSS esterno:

```
@import url(foglio.css)
h1 { color: black; }
```

Supponendo che questo sia un CSS, avremo il risultato di incorporare al suo interno il contenuto del secondo foglio di stile (**foglio.css**). Si veda la pagina di esempio (http://html.it/guide/esempi/css/test/usare_@import.html).

È possibile definire all'interno della direttiva @import anche il supporto cui applicare il CSS, in modo simile a quanto abbiamo visto a proposito dell'attributo **media**. Per fare ciò basta far seguire all'url del CSS l'indicazione di uno dei media previsti nella specifica:

```
<style type="text/css">
@import url(foglio_stampa.css) print;
@import url(foglio_schermo.css) screen;
</style>
```

Fogli incorporati

I fogli incorporati sono quelli inseriti direttamente nel documento (X)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>...

Come si vede il codice inizia con l'apertura del tag **<STYLE>**. Esso può avere due attributi:

1. type (obbligatorio)

2. media (opzionale)

per i quali valgono le osservazioni viste in precedenza. 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 (X)HTML definita **Common**: si tratta di quegli attributi 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 e lo sfondo nero, scriveremo:

```
<h1 style="color: red; background: black;">...</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 sentito è semplice: **pianificate**, pensate in anticipo a quella che dovrà essere la struttura delle pagine del sito. Mettetevi davanti a un monitor o su un pezzo di carta e ragionate: qui metterò un tag <H1>, qui userò una tabella a due colonne, questo box deve avere lo sfondo rosso, etc.

A questo punto potrete costruire 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 150 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.

Se le pagine invece di una fossero 20 il discorso diventerebbe un pò complicato. Bisognerebbe armarsi di pazienza e modificarle una per volta.

Anche qui però la soluzione è dietro l'angolo. Basta fare un nuovo CSS esterno e collegarlo al documento con @import insieme al foglio generico:

```
<link rel="stylesheet" type="text/css" href="stile.css">
<style type="text/css">
  @import url(nuovo_stile.css);
</style>
```

Anche questa volta, le regole dello stile collegato con @import vanno a sovrascrivere quelle ereditate dall'elemento **<LINK>**. Sono solo alcune delle strategie possibili e mi si perdoni la forse eccessiva semplificazione. Basta rendere l'idea.

Un'ultima notazione. L'uso estensivo di fogli in linea rischia di compromettere uno dei principali vantaggi dei CSS, ovvero avere pagine più leggere e facili da gestire. Intervenire nei meandri di una pagina per andare a modificare uno stile e ripetere l'operazione per quante sono le pagine del nostro sito può diventare davvero frustrante. Del resto, il loro uso è ultimamente considerato deprecato anche dal W3C.

4. L'attributo media

Nella lezione precedente abbiamo accennato all'attributo **media**. Grazie ad esso siamo in grado di impostare un foglio di stile per ogni supporto su cui la nostra pagina verrà distribuita. Una possibilità davvero interessante e che andrà sempre più diffondendosi con l'ampliarsi dei mezzi di diffusione delle pagine (X)HTML.

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, oltre che controproducente. 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.
- **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.

Il valore di default è **all**, usato automaticamente in mancanza di una dichiarazione esplicita (ricordiamo infatti che **media** è un attributo opzionale). E' possibile usare più di un valore, ma i nomi della lista vanno separati da una virgola:

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

L'uso più tipico di questa funzionalità consiste nel collegare al documento vari fogli di stile adatti a ciascun supporto. Lo user agent che visualizzerà la pagina sarà in grado, se conforme agli standard, di caricare quello giusto:

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

Esempi

Un'applicazione immediatamente fruibile, a patto di non usare browser di quarta generazione, è quella che prevede l'uso di un foglio di stile per la stampa. Le motivazioni per una simile soluzione possono essere molteplici. L'uso di caratteri o colori particolari, la presenza nella pagina di elementi che vogliamo non far comparire sulla carta, può portare alla necessità di adottare un CSS speciale. Un primo esempio lo potete valutare seguendo questo link (<http://html.it/guide/esempi/css/test/media.html>). Provate a stamparlo. Il testo avrà caratteristiche diverse rispetto a quello che vedete sullo schermo: font, colore, dimensioni in punti e non in pixel. Una controprova: stampate questo documento (http://html.it/guide/esempi/css/test/no_media.html), privo di fogli di stile speciali. Nessun cambiamento rispetto a quanto vedete sullo schermo.

Per un approfondimento trovate l'articolo *Stampe perfette con i CSS* nella sezione PRO di HTML.it.

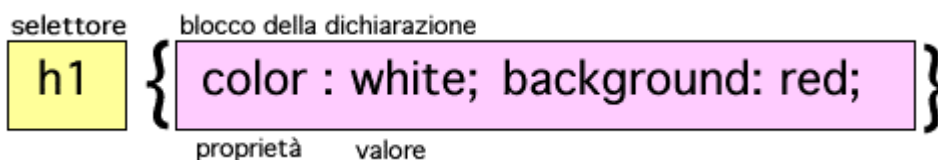
Una risorsa eccellente sull'argomento: Code Style media monitor (<http://www.codestyle.org/css/media/>). Articoli, linee guida, test sull'uso dei fogli di stile basati sui diversi media.

5. Come è fatto un CSS: regole e commenti

Quanto visto finora riguarda essenzialmente il rapporto tra CSS e (X)HTML: tutti gli elementi, gli attributi e le funzionalità analizzate fanno parte della specifica del secondo linguaggio.

Con questa lezione entriamo nel vivo dell'argomento. Iniziamo con l'analisi degli elementi costitutivi di un foglio di stile. Aprite questo file (http://html.it/guide/esempi/css/test/foglio_di_stile.css), dopo averlo scaricato, con il vostro editor preferito. Tutto quello che ci trovate dentro appartiene a due tipologie di dichiarazioni: regole (ingl. **rules**) e commenti. Ecco, un foglio di stile non è altro che questo: 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. I selettori possono essere diversi e a queste varie tipologie dedicheremo una delle prossime lezioni. Per il momento sia chiara la funzione che essi svolgono.

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

- **proprietà**
- **valore**

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

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

Mentre questa è perfettamente valida e plausibile:

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

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

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

Commenti

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

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

Proprietà singole e a sintassi abbreviata

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

Le proprietà singole sono la maggior parte: impostano per un dato elemento o selettore un singolo aspetto. Con le **shorthand properties**, è possibile invece

definire con una sola dichiarazione un insieme di proprietà. Chiariamo con un esempio.

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

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

La regola sarebbe questa:

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

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

margin:

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

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

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

Un ottimo articolo introduttivo: Introduction to shorthand properties
(<http://home.no.net/junjun/html/shorthand.html>).

6. I selettori

La parte preponderante della specifica CSS2 è dedicata all'analisi delle diverse proprietà in grado di definire l'aspetto visuale di elementi e sezioni di una pagina. Prima di tutto, però, è fondamentale capire come e a cosa queste proprietà possono essere assegnate. L'argomento sarà l'oggetto delle prossime quattro lezioni.

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 (X)HTML.

Sintassi

```
h1 {color: #000000;}
```

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

```
table {width: 200px;}
```

Esempio (http://html.it/guide/esempi/css/test/sel_type.html)

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;}
```

Ulteriori esempi e commenti in questa pagina

(http://html.it/guide/esempi/css/test/sel_singgruppi.html).

Selettore universale (universal selector)

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

Sintassi

```
* { color: black; }
```

Esempio (http://html.it/guide/esempi/css/test/sel_universale.html)

Selettore del discendente (descendant selector)

Nella specifica CSS1 questo tipo era definito "selettore contestuale". 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, a qualsiasi livello.

Sintassi

```
div p {color: black;}
```

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

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

Fate attenzione alla struttura del documento ed evitate possibili incongruenze. Esistono regole ben precise sull'annidamento degli elementi che vanno rispettate sia in (X)HTML che nei CSS. Un paragrafo, per esempio, non può contenere un div, così come un elemento inline non può contenere elementi blocco. Utili chiarimenti sull'argomento nella lezione "Il corpo del documento" della Guida a XHTML.

Esempio (http://html.it/guide/esempi/css/test/sel_descendant.html)

Links

Se non volete perdervi tra i selettori CSS2 e sapere tutto sul supporto di classi, pseudo-classi e pseudo-elementi, avete a disposizione un'ottima risorsa: CSS2 selectors Support Chart (<http://www.webreview.com/style/css2/charts/selectors.shtml>). Mancano Mozilla ed Explorer 6, ma è un ottimo punto di partenza per sapere quali browser supportano determinati costrutti o selettori.

7. Id e classi: due selettori speciali

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

Partiamo dall'inizio. In (X)HTML esistono due attributi fondamentali applicabili a tutti gli elementi: sono **class** e **id**. Dichiarare questi attributi a prescindere dai CSS non ha alcun senso e non modifica in alcun modo la presentazione della pagina. Ecco un esempio. In questa pagina (http://html.it/guide/esempi/css/test/classe_senza_css.html) abbiamo assegnato al paragrafo un attributo **class="testorosso"**:

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

Come vedete non succede nulla. Il problema è che il valore dell'attributo **class** deve trovare una corrispondenza in un foglio di stile. In questa seconda pagina (http://html.it/guide/esempi/css/test/classe_con_css.html), 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 (X)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 del paragrafo, del div e della tabella.

Ma non questa:

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

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

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

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

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

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

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

Classe

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

```
.nome_della_classe
```

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

Esiste un secondo tipo di sintassi:

```
<elemento>.nome_della_classe
```

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

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

lo stile verrà applicato solo ai paragrafi che presentino l'attributo

class="testorosso". Anche qui è importante stabilire un minimo di strategia.

Il secondo tipo di sintassi va usato solo se pensate di applicare una classe ad

uno specifico tipo di elemento (solo paragrafi o solo div, e così via). Se invece ritenete di doverla applicare a tipi diversi usate la sintassi generica.

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

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

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

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

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

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

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

Esempi di questa e delle altre modalità in questa pagina di esempio (http://html.it/guide/esempi/css/test/sel_classi.html).

ID

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

```
#nome_id
```

Con questa regola:

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

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

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

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

```
p#nome_id
```

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

Pagina di esempio (http://html.it/guide/esempi/css/test/sel_id.html) anche per l'**ID**.

8. Valori e unità di misura

In questa lezione vedremo 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

Questo è la lista delle unità di misura usate per definire dimensioni, spazi o distanze. Nella pratica comune solo alcune di esse sono realmente usate. Le elenchiamo comunque tutte per completezza.

- **in (inches - pollici)**: classica misura del sistema metrico americano. Praticamente nullo il suo valore su un supporto come un browser web viste le variabili relative a risoluzione e ampiezza dei monitor.
- **cm (centimetri)**: stesso discorso visto per i pollici, la difficoltà maggiore sta nella resa su monitor, che è altra cosa rispetto alla carta stampata.
- **mm (millimetri)**: vedi centimetri.
- **pt (points - punti)**: unità di misura tipografica destinata essenzialmente a definire la dimensione dei font.
- **pc (picas)**: unità poco usata. 1 pica equivale a 12 punti.
- **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.

- **ex (ex-height)**: poco usata. 1 ex equivale all'altezza del carattere x minuscolo del font scelto.
- **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

Sui diversi modi per esprimere il valore di un colore si veda la lezione su "colore e CSS".

Stringhe

Alcune proprietà dei CSS possono avere come valore una stringa di testo da inserire come contenuto aggiunto nel documento. I valori espressi da stringhe vanno sempre racchiusi tra virgolette. Le proprietà in questione sono tre: **content**, **quotes**, **text-align** (ma solo per le tabelle definite con i CSS).

Valori URI

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).

Unità per gli angoli

Due proprietà comprese nella sezione dei CSS dedicata ai dispositivi audio possono essere espresse con unità di misura relative agli angoli. Le due proprietà sono **azimuth** e **elevation**. Le unità di misura queste:

- **deg (degree - grado)**: descrive l'ampiezza di un angolo (es. 90deg).

- **grad (gradians)**: descrive l'ampiezza di un angolo su una scala 1-400 (es. 100grad = 90deg)
- **rad (radians)**: descrive l'ampiezza di un angolo su una scala 1-pi greco

Unità di tempo

Anche le unità di tempo trovano spazio solo negli stili audio. Sono usate in genere per impostare la pausa tra le parole lette da un sintetizzatore vocale. Si applicano solo a queste tre proprietà: **pause**, **pause-after**, **pause-before**. Le unità di misura sono:

- **s (secondi)**
- **ms (millisecondi)**

Unità di frequenza

Usate solo negli stili audio, definiscono la frequenza del segnale:

- **hz (Hertz)**
- **khz (Kilohertz)**

9. Ereditarietà, cascade, conflitti tra stili

Se c'è una lezione di questa guida che dovrete stampare e conservare in una cartellina è quella che state per leggere. Stiamo per entrare nel cuore del meccanismo di funzionamento dei CSS. Sono regole un pò complesse, ma basilari. Quindi attenzione massima. Gli esempi che accompagnano la parte teorica vi aiuteranno nel percorso di apprendimento: usateli sempre.

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. L'esempio (<http://html.it/guide/esempi/css/test/ereditarieta.html>) è al riguardo chiarificatore. Se impostiamo il colore rosso per il testo (**color: red;**) a livello dell'elemento **BODY** tutti gli altri elementi suoi discendenti ereditano 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 e lo renderemo esplicito nell'analisi di ciascuna di esse. In genere 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! Fin qui tutto semplice. Ora scendiamo nei dettagli. L'ereditarietà non basta a spiegare le molteplici possibilità di relazione tra le regole di un CSS.

Peso e origine

Da qui in avanti affronteremo un'altra serie di concetti fondamentali tutti riconducibili comunque ad uno stesso ambito: i conflitti possibili 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? Provare per credere: esempio (<http://html.it/guide/esempi/css/test/conflitti.html>).

Il primo concetto da imparare è quello di **peso**. Si riferisce alla maggiore o minore importanza da assegnare a ciascuna regola. Un primo criterio di importanza è dato dall'**origine del foglio di stile**. Quando visualizziamo una pagina (X)HTML possono entrare in gioco nel modificare lo stile degli elementi tre diversi fogli di stile:

- **foglio dell'autore**
- **foglio dell'utente**
- **foglio predefinito del browser**

In ordine di importanza avremo: foglio dell'autore, foglio dell'utente, foglio predefinito del browser.

Tutti i software di navigazione di ultima generazione consentono una gestione di questo aspetto. E' possibile, ad esempio, far sì che il browser ignori i CSS definiti dall'autore delle pagine e formattare queste ultime con un CSS realizzato dall'utente. E ancora, come vedremo, è possibile modificare questa gerarchia con l'uso della parola chiave **!important**. Di base, però, l'ordine è quello definito qui sopra.

Specificità

La specificità, come il nome può suggerire, descrive il peso relativo delle varie regole all'interno di un foglio di stile. Esistono regole ben precise per calcolarla e sono quelle che applica lo user agent di un browser quando si trova davanti ad un CSS.

I fattori del calcolo sono tre e ciascuno di essi rappresenta il valore di una tripletta. Per prima cosa si conta il numero di selettori **ID** presenti nella regola. Si passa quindi a verificare la presenza di **classi**. Infine si conta il numero di **elementi** definiti nella regola. Mai come in questo caso urge l'esempio. Prima regola:

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

Calcolo: un **ID**, 0 classi, 0 elementi. tripletta dei valori: 1-0-0

```
.classe1 {background: #C00;}
```

0 ID, 1 **classe**, 0 elementi. tripletta: 0-1-0

```
h1 {color: red;}
```

0 ID, 0 classi, un **elemento**. tripletta: 0-0-1

Il peso specifico della prima regola è il maggiore. Quello dell'ultima il minore.

In pratica: gli ID pesano più delle classi che pesano più dei singoli elementi.

Non commettete l'errore di valutare il numero più grande a prescindere dalla sua posizione. Questa regola presenta la seguente specificità 1-0-0:

```
#paragrafo {color: green;}
```

ed è più importante di questa che ha i seguenti valori 0-0-2:

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

Subito la prova. Se osservate il terzo esempio di questa pagina

(<http://html.it/guide/esempi/css/test/specificita.html>) vedrete che il paragrafo sarà verde e non rosso.

Il concetto di cascade

Ed ora la summa di tutto quello che abbiamo detto. Il concetto e il meccanismo di **cascade** spiegato con parole semplici. E che sia un elemento chiave lo capite dal nome stesso di quello che state studiando: Cascading Style Sheets. Tenteremo di ricostruire il procedimento di un browser quando incontra un foglio di stile e lo rende sul monitor del nostro computer.

1. Per prima cosa controlla il target stabilito con l'attributo **media** o con dichiarazioni equivalenti. Scarta quindi tutti gli stili riferiti alla stampa o ad altri supporti. Allo stesso tempo scarta tutte le regole che non trovino corrispondenza negli elementi strutturali del documento.

2. Comincia ad ordinare per **peso** e **origine** secondo le regole viste sopra. C'è un CSS definito dall'autore? Userà quello. Altrimenti verificherà la presenza di un foglio di stile utente e in sua assenza applicherà le sue regole stilistiche predefinite.

3. Quindi calcola la specificità dei selettori e in caso di conflitto tra regole usa questo criterio di prevalenza.
4. Se non ci sono conflitti o se peso, origine e specificità coincidono, viene applicata la regola più vicina all'elemento nel codice del documento. L'ordine, se le dichiarazioni degli stili sono fatte nell'ordine più corretto e logico, è quindi il seguente: gli stili **in linea** prevalgono su quelli **incorporati** che a loro volta prevalgono su quelli **collegati**.
- Esempi e commenti in questa pagina
(<http://html.it/guide/esempi/css/test/cascade.html>).

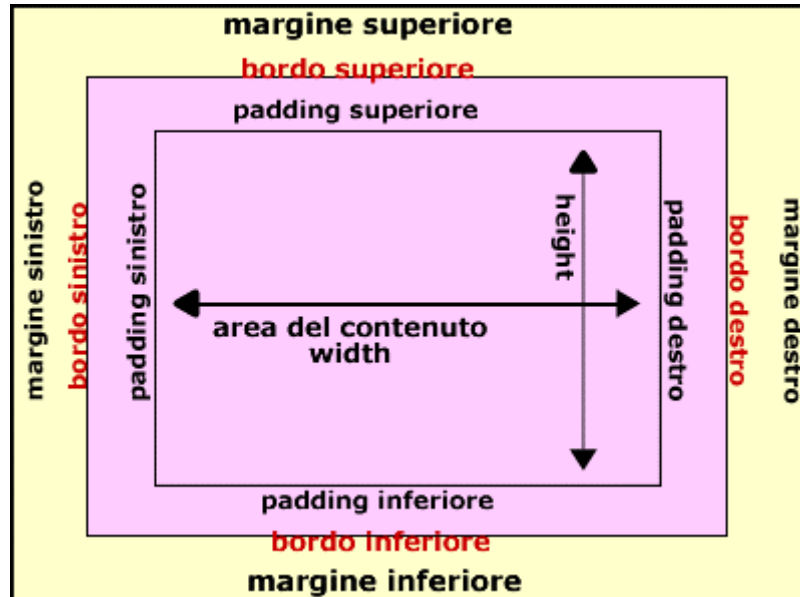
Importanza

Ed ora il concetto di importanza. Semplice e lineare la regola: 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. Ancora un esempio
(<http://html.it/guide/esempi/css/test/importance.html>) per verificare il concetto.

10. Il box model

Se volete usare i CSS per scopi che vadano oltre la semplice gestione di sfondo e testo dovete avere ben chiaro il meccanismo che governa la presentazione dei vari elementi di una pagina. Torniamo per un attimo alla prima lezione. Abbiamo lì mostrato che una pagina (X)HTML non è altro che un insieme di box rettangolari, che si tratti di elementi **blocco** 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.

Attenzione. Queste cose non sono state introdotte con i CSS, ma fanno parte del normale meccanismo di rendering di un documento. Quando realizziamo una pagina (X)HTML senza fogli di stile è il browser ad applicare per alcune di queste proprietà le sue impostazioni predefinite. Per esempio, introdurrà un certo margine tra un titolo e un paragrafo o tra due paragrafi. La novità è che con i CSS possiamo controllare con precisione al pixel tutti questi aspetti. 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.

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

Come si vede infatti nella figura margini, padding e bordi devono considerarsi a tutti gli effetti parte dell'area complessiva dell'elemento. L'esempio (http://html.it/guide/esempi/css/test/box_model.html#es1) chiarisce anche visivamente questo aspetto (attenzione: se visualizzate questa pagina con Explorer 5/5.5 su Windows non vedrete la cosa giusta. Questo browser interpreta male il concetto di larghezza. Per chiarimenti e soluzioni si veda la lezione dedicata alla proprietà **width**).

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. Quest'ultimo è l'elemento che racchiude il box. Esempio (http://html.it/guide/esempi/css/test/box_model.html#es2).

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.

Margini verticali e orizzontali tra gli elementi

Per due box adiacenti in senso verticale che abbiano impostato un margine inferiore e uno superiore la distanza non sarà data dalla somma delle due distanze. A prevalere sarà invece la distanza maggiore tra le due. E' il meccanismo del cosiddetto **margin collapsing**. Tale meccanismo non si applica ai box adiacenti in senso orizzontale. Esempio (http://html.it/guide/esempi/css/test/box_model.html#es4).

Ulteriori approfondimenti su questi argomenti li troverete nelle prossime lezioni, quando affronteremo l'analisi delle singole proprietà. Qui aggiungiamo che regole particolari su alcuni degli aspetti trattati si applicano agli elementi posizionati. Ne parleremo in una lezione specifica.

Links

Il box-model secondo il W3C (<http://www.w3.org/tr/REC-CSS2/box.html>): complesso e un pò intricato.

Il box-model, teoria e pratica (<http://www.webreference.com/html/tutorial9/>): ottimo tutorial sul sito WebReference.com. Lettura vivamente consigliata.

11. Gestione del colore

Iniziamo la rassegna delle proprietà di CSS2 analizzando una di quelle che può a buon diritto dirsi fondamentale. Esamineremo prima le diverse possibilità di definire i valori dei vari colori, per poi analizzare usi e sintassi della proprietà **color**.

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

L'ordine è quello dei colori visualizzati nell'immagine.

#RRGGBB

Le semplici 16 parole chiave non esauriscono ovviamente la gamma dei colori visualizzabili su un monitor moderno. Già in HTML era possibile impostare il colore di un elemento servendosi di **codici esadecimali**. In essi, 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**).

Esempio

#CC0000

Il codice sopra rappresenta una tonalità di rosso.

#RGB

Molti dei codici esadecimali sono rappresentati da valori duplicati. E' possibile usare per essi una sintassi abbreviata in cui i valori per il rosso, il verde e il

blue sono definiti solo dalla prima lettera o numero. Il colore dell'esempio precedente può essere definito anche così:

#C00

Nell'uso di questa sintassi vanno valutati i risultati con colori che non presentino coppie di valori duplicati. Il risultato può essere leggermente diverso a livello di tonalità a seconda dei casi.

rgb(rrr%, ggg%, bbb%)

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%)

Per il bianco:

rgb(100%, 100%, 100%)

rgb(rrr, ggg, bbb)

Ultima possibilità. 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.

Questo è il codice del nero:

rgb(0, 0, 0,)

La proprietà color

Visti i sistemi per rappresentare i colori, dobbiamo ora chiarire un aspetto importante. Per ogni elemento si possono definire almeno tre colori:

1. **il colore di primo piano** (foreground color)
2. **il colore di sfondo** (background color)
3. **il colore del bordo** (border color)

La proprietà **color** definisce esclusivamente:

- il colore di primo piano, ovvero quello del testo

- il colore del bordo di un elemento quando non si imposti esplicitamente quest'ultimo con le proprietà **border** o **border-color** (vedi l'esempio allegato (<http://html.it/guide/esempi/css/test/color.html#p6>)).

Per gli altri due casi esistono proprietà ad hoc che esamineremo nelle prossime lezioni. Una buona norma dei CSS vuole comunque che per un elemento di cui si sia definito il colore di primo piano si definisca anche e sempre un colore di sfondo.

Sintassi

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> }
```

Valori

I valori possibili sono:

- qualunque valore di un colore definito con i metodi visti sopra
- la parola chiave **inherit**. Con essa si dice esplicitamente al browser di ereditare le impostazioni definite per l'elemento parente. E' stata introdotta con CSS2 e da qui in avanti sarà citata nella guida senza ulteriori spiegazioni.

Esempi

```
p {color: black; background-color: white; }
```

Ulteriori esempi in questa pagina

(<http://html.it/guide/esempi/css/test/color.html>).

Links

CSS color names (http://www.w3schools.com/css/css_colornames.asp): ottima risorsa dal sito W3Schools. Tabella Websafe con valori esadecimali e nomi dei colori (ricordiamo però che nei CSS gli unici nomi accettati sono i 16 visti nella lezione). L'utilità più grande la notate se cliccate sul codice esadecimale del colore. Si aprirà una pagina in cui quel colore viene usato come sfondo e tutti gli altri vengono usati per il testo. Eccellente quando si deve valutare l'impatto visivo della coppia colore testo-sfondo.

"I colori e il loro uso nel web": dalla sezione PRO di HTML.it.

12. Gestione dello sfondo

Esamineremo in questa lezione le proprietà relative alla gestione dello sfondo. Sin dalle origini di HTML è stato possibile definire un colore o un'immagine di sfondo per le nostre pagine web. La scelta si restringe comunque a due elementi: il corpo principale della pagina (**<BODY>**) o le tabelle. Un'altra limitazione riguarda il comportamento delle immagini di sfondo: esse vengono infatti ripetute in senso orizzontale e verticale fino a riempire l'intera area della finestra, del frame o della tabella. Proprio questo comportamento ha fatto sempre propendere per la scelta di piccole textures in grado di dare la sensazione visiva dell'uniformità. Grazie ai CSS queste limitazioni vengono spazzate via e le possibilità creative, compatibilità permettendo, sono davvero infinite.

Ecco la lista delle proprietà, applicabili, ed è questa la prima grande innovazione dei CSS, a **tutti gli elementi**:

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

Ciascuna di essa definisce un solo, particolare aspetto relativo allo sfondo di un elemento. 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. La analizzeremo alla fine. Vediamo di seguito la definizione delle prime due proprietà, la descrizione di tutte le proprietà si trova all'indirizzo indicato all'inizio di questa pagina.

background-color

Definisce il colore di sfondo di un elemento. Questa proprietà non è ereditata.

Sintassi

selettore {background-color: <valore>;}

Valori

- **un qualunque colore**
- la parola chiave **transparent**

Usando **transparent** come valore un elemento avrà come colore quello dell'elemento parente. Nell'esempio (http://html.it/guide/esempi/css/test/back_color.html): il colore di #div2 è impostato su **transparent**, per cui esso sarà, per l'appunto, trasparente, lasciando emergere il colore dell'elemento parente (#div1).

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. Questa proprietà non è ereditata.

Sintassi

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

Valori

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

Usare **none** equivale a non impostare la proprietà: nessuna immagine verrà applicata come sfondo.

Esempi

```
body {background-image: url(sfondo.gif); }  
div body {background-image: url(http://www.server.it/images/sfondo.gif); }  
Usando questa proprietà da sola si ottiene lo stesso risultato che in HTML si  
aveva con l'attributo background (vedi esempio  
(http://html.it/guide/esempi/css/test/back\_image.html)).
```

Piccolo consiglio. Una buona norma e il buon senso vogliono che quando si definisce un'immagine come sfondo si usi sempre anche un colore di sfondo e che questo colore consenta una lettura agevole del testo. Se le immagini sono

disabilitate, ad esempio, il browser mostrerebbe il suo colore di sfondo predefinito: se questo è bianco e il nostro testo pure sarebbe evidentemente un disastro. Attenzione, dunque!

background

E veniamo alla proprietà **background**. Con essa, ricordiamolo, 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: background-color background-image background-repeat background-attachment background-position;}

I valori non vanno separati da virgole. L'ordine non è importante, ma quello dato è il più logico e andrebbe rispettato: si va in ordine di importanza.

Esempi

In questa pagina (<http://html.it/guide/esempi/css/test/background.html>) di esempio lo sfondo della pagina è stato così settato con l'uso della proprietà

background:

```
body {  
background: white url(sfondo.gif) repeat-x fixed;  
}
```

13. Gestione del testo: proprietà di base

Se c'è un aspetto essenziale dei CSS questo è il nuovo approccio che hanno introdotto per la gestione del testo. Prima c'era il tag ****: pagine pesanti e difficili da gestire. Oggi qualcosa che può trasformare una pagina web in un perfetto esempio di stile tipografico e che finalmente consente, almeno in parte, la potenza e la flessibilità di un word-processor.

La proprietà che definiscono il modo in cui il testo appare sullo schermo sono tante e abbiamo deciso di suddividere l'argomento in due lezioni. Iniziamo quindi dalle 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)

La lezione è corredata di un documento di esempio (http://html.it/guide/esempi/css/test/testo_base.html) che mostra l'uso delle proprietà esaminate e corredata di codice.

font-family

La proprietà font-family serve a impostare il tipo di carattere di una qualunque porzione di testo. Si applica a tutti gli elementi ed è ereditata.

Gli uomini del W3C hanno creato un meccanismo che consente all'autore di impostare nei CSS non un font singolo e unico, ma un elenco di caratteri alternativi. Il meccanismo funziona così:

```
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. La spiegazione di tutto ciò è semplice: ovviare al problema dei diversi font presenti sulle piattaforme software.

Dunque: quando si imposta la proprietà font-family si possono usare tutti i font che si vuole, ma non dimenticate mai di inserire alla fine l'indicazione di una famiglia generica. Esse sono cinque (tra parentesi riportiamo i caratteri predefiniti su ciascuna sui sistemi Windows):

- **serif** (Times New Roman)
- **sans-serif** (arial)
- **cursive** (Comic Sans)
- **fantasy** (Allegro BT)
- **monospace** (Courier)

I nomi dei font della lista vanno separati dalla virgola. I caratteri con nomi composti da più parole vanno inseriti tra virgolette. Se usate famiglie strane e poco comuni come fantasy o cursive usate più di una famiglia generica. Questa andrebbe sempre messa alla fine, altrimenti risulta praticamente inutile la definizione di font specifici.

Sintassi

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

Esempi

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

font-size

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

Sintassi

selettore { font-size: <valore>; }

Valori

I valori delle dimensioni del testo possono essere espressi in vari modi. Per un approfondimento sull'uso dei diversi metodi e sulle implicazioni in termini di accessibilità e compatibilità cross-browser si consiglia la lettura dell'articolo "Stabilire le dimensioni dei font con i CSS: vantaggi, problemi e soluzioni". I diversi sistemi si possono distinguere a seconda che definiscano le dimensioni in senso assoluto o relativo. Dimensione assoluta significa che essa non

dipende da nessun altro elemento ed è quella definita dall'unità di misura usata. Dimensione relativa significa che essa viene calcolata in base alla dimensione del testo dell'elemento parente.

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**), picas (**pc**), pollici (**in**), x-height(**ex**). Di tutte queste unità le uniche proponibili per il testo sono punti e pixel. Si consiglia di usare la prima solo per CSS destinati alla stampa.

Sono valori relativi:

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

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. Si applica a tutti gli elementi ed è ereditata.

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
- **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

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

La proprietà si applica a tutti gli elementi ed è ereditata.

Sintassi

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

Esempi

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

Line-height

Grazie a line-height è possibile finalmente usare nelle nostre pagine un sistema di interlinea degno di questo nome. La proprietà, in realtà, serve a definire l'altezza di una riga di testo all'interno di un elemento blocco. Ma l'effetto ottenuto è appunto quello tanto agognato da tutti i web designer: un modo per impostare uno spazio tra le righe. La proprietà si applica a tutti gli elementi ed è ereditata.

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;}
```

font

La proprietà **font** può essere paragonata a **background**. Si tratta di una proprietà a sintassi abbreviata che serve a impostare con una sola dichiarazione tutte le principali caratteristiche del testo. Le proprietà definibili in forma abbreviata con **font** sono:

- font-family
- font-size
- line-height
- font-weight
- font-style
- font-variant
- font di sistema

Sintassi

```
selettore {font: <valori>;}
```

Alcune indicazioni sull'uso. I valori relativi alle singole proprietà non vanno separati da virgole. Virgola che deve invece separare i valori definiti per la font-family. Anche in questo caso i nomi dei font costituiti da più parole vanno racchiusi tra virgolette. Per quanto riguarda l'ordine, la dichiarazione dovrebbe sempre finire con la coppia font-size > font-family.

Il valore della proprietà line-height si imposta invece così, facendo seguire il suo valore a quello di font-size e separando i due con una slash:

font-size/line-height

Esempi

p {font: bold 12px/1.5 Georgia, "Times New Roman", serif;}

Nell'ordine abbiamo definito: font-weight, dimensione/ interlinea, font-family.

Usare i font di sistema

All'interno della proprietà font è possibile impostare, invece della normale lista di caratteri, un valore corrispondente ai cosiddetti **font di sistema**. Si tratta del tipo di carattere che usiamo sul nostro sistema operativo per definire l'aspetto di elementi dell'interfaccia come barre dei menu, icone o barra di stato. In particolare, le parole chiave relative ai font di sistema sono 6 e riguardano:

1. **caption**: carattere usato per pulsanti e menu a tendina
2. **icon**: carattere usato per la descrizione delle icone
3. **menu**: carattere usato nei menu delle varie finestre
4. **message-box**: usato per i gli alert e gli altri box con messaggi di sistema
5. **small-caption**: carattere per i controlli più piccoli
6. **status-bar**: il font usato per la barra di stato

Se si intende usare uno dei font di sistema non è necessario specificare altri valori oltre alla keyword. Il nostro testo, in pratica, erediterà le impostazioni definite nel sistema operativo relative a carattere e dimensione.

Esempi

p {font: menu;}

div {font: status-bar;}

Text-align

La proprietà serve a impostare l'allineamento del testo. E' ereditata e si applica a tutti gli elementi.

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. Ereditabile e applicabile a tutti gli elementi.

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;}

Links

Negli articoli che seguono troverete risposte a importanti questioni legate alla gestione del testo con i CSS e che esulano dall'impianto di una guida come questa. La lettura è pertanto fortemente consigliata:

L'arte del testo perfetto: come utilizzare i CSS per formattare il testo di una pagina web

Stabilire le dimensioni dei font in un CSS: vantaggi, problemi e soluzioni
I font e la tipografia del testo

14. Gestione delle dimensioni: altezza

Con le prossime cinque lezioni vedremo come sia possibile definire e controllare le diverse proprietà del box-model (guida_css_15.htm). In questa vedremo in che modo nei CSS si impostino le dimensioni verticali di un elemento. Un primo concetto fondamentale: **in genere l'altezza di un elemento è determinata dal suo contenuto**. Più testo inserisco in box, in un paragrafo o in una cella di tabella più essi saranno estesi in senso verticale. Semplice. Le proprietà che analizzeremo servono in pratica a fornire un meccanismo in grado di controllare o superare in qualche modo questo comportamento standard.

height

Questa proprietà definisce la distanza tra il bordo superiore e quello inferiore di un elemento. Non è ereditata e si applica a tutti gli elementi tranne:

- colonne di tabelle
- elementi inline non rimpiazzati

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.

L'uso di **height** va sempre valutato con attenzione e non pensando di farne una scorciatoia per avere layout precisi al pixel. Il caso più importante da valutare è quando il contenuto dovesse superare i limiti imposti tramite la proprietà. Il comportamento dei vari browser è al riguardo notevolmente diverso, come si può osservare nell'esempio (<http://html.it/guide/esempi/css/test/height.html>). Explorer non fa altro che ignorare l'altezza impostata estendendola. Mozilla e Opera rispettano la regola, ma il contenuto eccedente va a sovrapporsi agli elementi vicini o sottostanti. Ricordatelo: molte volte si può ottenere lo stesso risultato visivo desiderato usando proprietà come padding e margin. E considerate sempre che l'altezza è indirettamente influenzata anche dalla larghezza di un elemento.

Esempi

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

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 e non è ereditata.

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.

Potete verificare i risultati delle diverse impostazioni nella pagina di esempio (<http://html.it/guide/esempi/css/test/height.html>).

Esempi

```
div {overflow: auto;}
```

min-height

Imposta un'altezza minima per un elemento. Valgono per questa proprietà le stesse osservazioni fatte per **height** relativamente al contenuto. Non è ereditata e non è supportata da Explorer Win (test (<http://html.it/guide/esempi/css/test/min-height.html>)).

Sintassi

selettore {min-height: <valore>;}

Valori

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

Esempi

```
div {min-height: 200px;}
```

max-height

La proprietà **max-height** serve a impostare l'altezza massima di un elemento (test (<http://html.it/guide/esempi/css/test/max-height.html>)). Anche per essa valgono le osservazioni già fatte per il contenuto eccedente. Non è ereditata.

Sintassi

selettore {max-height: <valore>;}

Valori

- **none.** Valore iniziale. L'altezza dell'elemento non è limitata.
- **un valore numerico con unità di misura.**
- **un valore in percentuale**

Esempi

```
div {max-height: 400px;}
```

15. 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. A questo punto è utile richiamare alla mente i concetti visti per il box model e ribadire a cosa si riferiscono le proprietà che stiamo per esaminare.

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** (si vedano gli esempi (http://html.it/guide/esempi/css/test/box_model.html) già visti per il box model).

E' possibile ovviamente che i due valori coincidano. Accade quando di un particolare elemento non si impostino margini, padding e bordi; o quando il valore di tali proprietà sia uguale a 0. L'esempio (http://html.it/guide/esempi/css/test/width.html#largh_effettiva) mostra chiaramente questa situazione. I due box dovrebbero avere la stessa larghezza. Ma se usate Explorer 5 o 5.5 su Windows noterete che non è così. E qui introduciamo una questione di rilevanza assoluta per il semplice fatto che coinvolge il browser attualmente più usato. Explorer per Windows fino alla versione 5.5 interpreta male il concetto di **width**. Nel senso che con questa proprietà intende la larghezza globale dell'elemento, compresi margini, padding e bordi. Proprio l'errore che prima abbiamo definito macroscopico! Per

chi ha Explorer 6 abbiamo preparato questo documento di esempio (http://html.it/guide/esempi/css/test/width_nodotype.html) che ricalca e mostra con evidenza l'errore. Se nel documento, infatti, non si usa nessun DOCTYPE il browser di Microsoft funziona in modalità retro-compatibile, ripetendo bug ed errori dei suoi predecessori. Un comportamento utile per gli sviluppatori che possono così simulare nella versione 6 i comportamenti delle precedenti versioni.

Nell'articolo Un sito ridimensionabile con i CSS: parte II viene spiegato un trucchetto utile per correggere l'anomala gestione di **width** da parte di Explorer 5.

Fatta questa fondamentale premessa, possiamo analizzare nei dettagli la proprietà.

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.

La proprietà width non è ereditata.

Esempi

```
p {width: 90px;}
```

```
div.box {width: 50%;}
```

Per verificare il funzionamento delle varie impostazioni avete a disposizione la consueta pagina (<http://html.it/guide/esempi/css/test/width.html>) con codice e commenti. L'avvertenza speciale è per i possessori di Explorer 5/5.5 su Windows: testate il documento con Mozilla o Opera per apprezzare la resa corretta degli esempi. Quello che vedrete voi non è sempre la cosa giusta.

min-width

Imposta la larghezza minima di un elemento. Si applica a tutti gli elementi tranne che a quelli in linea non rimpiazzati e agli elementi di tabelle. Proprietà non supportata da Internet Explorer e non ereditata.

Sintassi

selettore { min-width: <valore>; }

Valori

- **un valore numerico con unità di misura.**
- **un valore in percentuale.** La larghezza sarà come minimo quella espressa dalla percentuale riferita alla larghezza dell'elemento contenitore.

Esempi

```
div { min-width: 400px; }
```

Per testare la proprietà usate questa pagina

(<http://html.it/guide/esempi/css/test/min-width.html>).

max-width

Imposta la larghezza massima di un elemento. Non è ereditata.

Sintassi

selettore { max-width: <valore>; }

Valori

- **none.** Valore di default. Non c'è un limite per larghezza dell'elemento.
- **un valore numerico con unità di misura.**
- **un valore in percentuale**

Links

Utili riferimenti agli argomenti trattati nella lezione e casi di applicazione li trovate negli articoli della sezione PRO dedicati alla realizzazione di layout con i CSS. Gli articoli sono corredati di codice, esempi e link a risorse esterne.

Un sito ridimensionabile con i CSS: parte I.

Un sito ridimensionabile con i CSS: parte II.

Un sito ridimensionabile con i CSS: parte III.

16. I Margini

Una delle maggiori limitazioni di HTML è la mancanza di un meccanismo interno per spaziare gli elementi di un documento. Per ottenere questo risultato si è ricorso negli anni ad una serie di trucchetti, magari efficaci, ma fuori dalla logica originaria del linguaggio. L'unica eccezione consisteva nella possibilità di definire una distanza tra le celle di una tabella con l'attributo **cellspacing**. I CSS risolvono il problema con l'uso di una serie di proprietà in grado di definire con precisione la distanza tra gli elementi. Possiamo infatti definire il margine come la distanza tra il bordo di un elemento e gli elementi adiacenti. Per ulteriori chiarimenti e per la relazione tra i margini e gli altri elementi del box-model si veda la lezione "Il box model".

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. Si applica a tutti gli elementi e non è ereditata.

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.
- **auto.** E' quasi sempre corrispondente a 0.

Ricordiamo che se per un elemento si imposta un margine inferiore ed esso si trova sopra ad un altro elemento che abbia impostato un margine superiore, la

distanza tra i due sarà quella maggiore e non data dalla somma delle due proprietà (meccanismo del margin collapsing).

Esempi

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

margin-left

Definisce la distanza tra il lato sinistro di un elemento e gli elementi adiacenti. Si applica a tutti gli elementi e non è ereditata.

Valori

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

Esempi

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

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.

Valori

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

Esempi

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

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.

Valori

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

Esempi

```
h1 {margin-right: 10%;}
```

```
img {margin-right: 20px;}
```

margin

E' una proprietà a sintassi abbreviata. Con essa è possibile specificare i valori per tutti e quattro i lati di un elemento. Si applica a tutti gli elementi e non è ereditata.

Valori

- **un valore numerico con unità di misura.**
- **un valore in percentuale.**
- **auto.** Per la proprietà margin il valore auto significa che la distanza sarà automaticamente calcolata rispetto alla larghezza dell'elemento contenitore.

Per usare al meglio questa proprietà è fondamentale conoscere le regole che ne gestiscono il funzionamento.

In prima istanza è ovvio usare per essa quattro valori, uno per ciascun lato:

```
div {margin: 10px 15px 10px 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. In pratica usare la sintassi vista nell'esempio equivale a scrivere:

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

Nella definizione dei valori, inoltre, è possibile mischiare percentuali con valori assoluti in unità di misura.

Un ulteriore abbreviazione della sintassi si può ottenere usando tre, due o un solo valore. Queste le regole:

- se si usano tre valori, il primo si riferisce al margine superiore, il secondo a quelli sinistro e destro, il terzo a quello inferiore
- se si usano due valori, il primo si riferisce ai lati superiore e inferiore, il secondo al sinistro e al destro
- se si usa un solo valore, un uguale distanza sarà ai quattro lati

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. Per un esempio di questa applicazione e di tutte le altre proprietà si veda il documento di esempio

(<http://html.it/guide/esempi/css/test/margini.html>).

Esempi

```
p {margin: 20px 10px;}  
div {margin: 20px;}  
h1 {margin: 10px auto;}
```

17. Il Padding

Un altro modo per creare spazio intorno ad un elemento è quello di usare il padding. Come per i margini, HTML incorpora un meccanismo simile solo per le celle di tabella, tramite il cosiddetto **cellpadding**. Se avete presente il funzionamento di questo attributo, vi risulterà immediatamente chiara la differenza tra margini e 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. Pertanto, valutate in base alle vostre esigenze se usare la prima o la seconda via.

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. Si applica a tutti gli elementi e non è ereditata.

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;}
```

padding-left

Imposta l'ampiezza del padding sul lato sinistro di un elemento. Si applica a tutti gli elementi e non è ereditata.

Valori

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

Esempi

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

padding-top

Imposta l'ampiezza del padding sul lato superiore di un elemento. Si applica a tutti gli elementi e non è ereditata.

Valori

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

Esempi

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

padding-right

Imposta l'ampiezza del padding sul lato destro di un elemento. Si applica a tutti gli elementi e non è ereditata.

Valori

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

Esempi

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

padding

Proprietà a sintassi abbreviata. Serve a impostare i valori del padding per tutti e quattro i lati di un elemento. Valgono per essa tutte le osservazioni e le regole sintattiche viste per **margin**.

Valori

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

Esempi

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

```
div {padding: 10%;}
```

Anche per il padding avete a disposizione un documento di esempio (<http://html.it/guide/esempi/css/test/padding.html>).

Ricordiamo un'ultima interessante applicazione di margini e padding. Ogni browser inserisce un margine più o meno ampio tra il bordo della finestra e l'area del contenuto. In HTML era possibile eliminarlo tramite una serie di attributi (e qui Explorer e Netscape avevano sintassi diverse) dell'elemento body:

```
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
```

La stessa cosa si può ottenere con i CSS (tra l'altro quegli attributi sono da tempo considerati svalutati dal W3C):

```
body {
margin: 0px;
padding: 0px; }
```

18. I Bordi

Riuscire a definire l'aspetto dei bordi è un'altra delle cose per cui vale la pena imparare ad usare i CSS. L'impatto visivo, se guidati dal buon gusto, è di sicuro effetto: la pagina acquista rigore e una suddivisione più logica, estetica e razionalità strutturale possono trovare un felice connubio.

Dal punto di vista del linguaggio la definizione dei bordi può risultare un pò intricata per il numero di proprietà coinvolte, che se consentono all'autore la massima flessibilità, rendono complicata la gestione del codice.

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`

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.**

Valori

Come si vede dall'elenco delle proprietà di ciascun lato 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 nella lezione " Il box model". 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**

Ciascuna definisce un particolare aspetto del bordo. Per verificare il quale potete osservare gli esempi contenuti in questa pagina (http://html.it/guide/esempi/css/test/stile_bordi.html).

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.

Nel caso delle parole chiave la dimensione esatta non è specificata dal linguaggio.

Esempi

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

Molto più comodo scrivere così:

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

Definire lo stile dei quattro bordi

Sintassi

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

Usando questa sintassi e queste proprietà si imposta lo stile per tutti e quattro i bordi del box. I valori possibili sono quelli visti prima a proposito del colore, dello stile e dello spessore.

Per ciascuna di queste proprietà è possibile definire da uno a quattro valori, uno per lato. Se ne usiamo quattro l'ordine di lettura è questo: top, right, bottom, left. Se invece ne impostiamo uno, due o tre valgono le stesse regole viste per i margini nella lezione "Gestione delle dimensioni: larghezza"

Esempi

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

Il risultato sarà questo (<http://html.it/guide/esempi/css/test/bordi.html#es5>).

Usare la proprietà border

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;}
```

Otterremo un box con questo aspetto

(<http://html.it/guide/esempi/css/test/bordi.html#es4>).

19. Agire sulle tabelle

Alcune proprietà dei CSS sono destinate alla modifica stilistica delle tabelle. Il supporto, chiariamolo subito, non è uniforme per tutti i browser. Mozilla brilla anche qui per la sua fedeltà agli standard, mentre Explorer Windows cade su parecchi aspetti. Nell'analisi delle proprietà ci limiteremo all'enunciazione dei concetti di base, tralasciando la miriade di regole particolari e spesso complesse che ne governano il comportamento. Del resto, l'uso di queste proprietà è davvero molto ridotto, dal momento che gli stessi risultati possono essere ottenuti in diversi modi, anche con i tradizionali sistemi incorporati in (X)HTML.

Ad accompagnare la spiegazione teorica questa pagina (<http://html.it/guide/esempi/css/test/tabelle.html>) di esempi e test.

table-layout

Questa proprietà imposta il metodo di layout di una tabella. Non è ereditata. Si applica solo alle tabelle.

Sintassi

```
<selettore> {table-layout: <valore>;}
```

Valori

- **auto**. Il layout della tabella viene definito automaticamente dal browser.
- **fixed**. Le regole di presentazione sono quelle impostate dall'autore nel CSS.

Nel caso del valore **auto** tutto è affidato al meccanismo di rendering dello user agent. Usando invece **fixed** possiamo innanzitutto definire la larghezza della tabella tramite la proprietà **width**. Volendo creare una tabella di 400px, quindi, scriveremo questa regola:

```
table {  
table-layout: fixed;
```

```
width: 400px;  
}
```

Capite bene che la stessa cosa si può ottenere in (X)HTML usando sempre l'attributo **width**:

```
<table width="400px">
```

Nella pagina di esempio (<http://html.it/guide/esempi/css/test/tabelle.html>) abbiamo riportato un paio di casi speciali per spiegare meglio il meccanismo di funzionamento. In particolare abbiamo verificato cosa accade se si imposta **auto** come valore della proprietà **width** e il comportamento dei vari browser nei casi in cui si stabilisca una larghezza esplicita anche per le celle della tabella.

Esempi

```
table.tabella1 {table-layout: fixed;}  
table {table-layout: auto;}
```

border-collapse

Attraverso questa proprietà possiamo stabilire in che modo trattare i bordi e gli spazi tra le celle di una tabella. Si applica solo alle tabelle ed è ereditata.

Sintassi

```
<selettore> {border-collapse: <valore>;}
```

Valori

- **collapse**. Se viene impostato un bordo, le celle della tabella lo condividono.
- **separate**. Se viene impostato un bordo, ogni cella ha il suo, separato dalle altre. Lo spazio tra le celle e tra i bordi si imposta con la proprietà **border-spacing**.

Esempi

```
table {  
border: 2px solid black;  
border-collapse: separate;  
border-spacing: 5px;  
}
```

border-spacing

Imposta lo spazio tra le celle di una tabella. Va usata solo in presenza di un modello di border-collapse settato su **separate**. Proprietà ereditata e applicabile solo alle tabelle.

Sintassi

<selettore> {border-spacing: <valore>;}

Valori

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

Esempi

Vedi sopra

empty-cells

Gestisce il trattamento delle celle di tabella senza contenuto. Agisce solo su quelle che non presentino al loro interno alcun tipo di markup, nemmeno il classico ** **; inserito in genere proprio per simulare la presenza di contenuto. Proprietà ereditata.

Sintassi

<selettore> {empty-cells: <valore>;}

Valori

- **show**. Mostra i bordi della cella.
- **hide**. I bordi non vengono mostrati e apparirà solo uno spazio vuoto.

caption-side

Le buone norme dell'accessibilità vogliono che una tabella sia sempre preceduta da una sorta di titolo/riassunto. In (X)HTML questa funzione è demandata al tag **<CAPTION>** di cui riportiamo sotto un esempio di sintassi:

```
<table>
<caption>Titolo della tabella</caption>
<tr>
<td>...
</table>
```

La proprietà **caption-side** definisce il lato su cui vogliamo far comparire tale titolo. E' ereditata. La supporta solo Mozilla.

Sintassi

<selettore> {caption-side: <valore>;}

Valori

- **top**. Caption sul lato superiore.
- **right**. Caption sul lato destro.
- **bottom**. Caption sul lato inferiore.
- **left**. Caption sul lato sinistro.

Esempi

```
table.tabella9 {  
table-layout : fixed;  
width : 300px;  
background : Silver;  
caption-side : bottom;  
}
```

20. Risorse on line

Iniziamo la nostra rassegna di risorse online da due ricchissime raccolte di links sull'argomento:

la sezione sui CSS di Website Tips

(<http://www.websitetips.com/css/index.shtml>)

Potremmo finire qui, perchè i due siti offrono davvero una miriade di risorse su qualunque aspetto dei CSS. Segnaliamo comunque i punti di riferimento essenziali.

Il W3C (<http://www.w3c.org>) è ovviamente il primo segnalato. Sul sito del Consorzio vanno evidenziate innanzitutto le specifiche complete per CSS1 (<http://www.w3.org/tr/REC-CSS1>) e CSS2 (<http://www.w3.org/tr/REC-CSS2/>). Vi segnalo la pagina principale (<http://www.w3.org/tr/xml-stylesheet/>) della sezione dedicata ai fogli di stile: contiene un elenco di risorse sempre aggiornato, con news sui lavori in corso, su editor e browser. Se volete poi seguire l'evoluzione dell'imminente specifica su CSS3 (<http://www.w3.org/tr/2001/WD-css3-roadmap-20010119/>) questo è l'indirizzo giusto.

Un altro sito immancabile è meyerweb (<http://www.meyerweb.com/>), la casa sul web di Eric Meyer. Poche persone hanno dato e fatto tanto per spiegare a tutti come usare questo meraviglioso strumento.

Un'altra risorsa imprescindibile sono le pagine del CSS Pointers Group (<http://css.nu/pointers/>): saggi, articoli, news, test, libri. Tutto fatto con competenza e spirito da pionieri.

Il sito Mako4Css (<http://www.mako4css.com/index.htm>) contiene interessanti tutorial su diversi aspetti dei CSS ed è una specie di luogo di ritrovo per tutti quelli che cercano soluzioni al problema compatibilità con Netscape 4.

Ottimi articoli li trovate anche su A list apart

(<http://www.alistapart.com/index.html>). La webzine curata da Jeffrey Zeldman

ha svolto un ruolo essenziale nell'adozione dei CSS e degli standard in genere. Alcuni pezzi presenti sul sito sono davvero dei punti fermi in materia. Zeldman è tra l'altro sempre attento a tutte le novità sull'argomento CSS e una visita periodica al suo Daily Report (<http://www.zeldman.com/>) vi terrà aggiornati su tutto.

Tutorial e reference di valore su CSS2 li trovate su RichInStyle

(<http://www.richinstyle.com/>), House of Style

(http://www.westciv.com/style_master/house/index.html), Zvon.org

(<http://www.zvon.org>) e W3Schools

(<http://www.w3schools.com/css/default.asp>).

21. Bibliografia

In italiano l'unico libro di un certo valore è:

CSS Pocket Reference, di *Eric A. Meyer*, Economici , Hops Libri, 2001

Si tratta dell'edizione ridotta (molto ridotta) del manuale sui CSS1 uscito in America nel 2000.

Per il resto dobbiamo affidarci alle nostre conoscenze di inglese e ad uno di questi libri, tutti ottimi e realizzati dai migliori esperti internazionali sull'argomento. Su Amazon costano meno, trovate utili recensioni degli utenti e vi arrivano in dieci giorni. Niente male, direi.

Cominciamo con la triade di opere di Eric Meyer. Il primo è un manualone completo su CSS1 con una piccola introduzione su CSS2. Il secondo è una classica reference, con l'elenco completo e commentato delle proprietà CSS2. Il terzo è invece eminentemente pratico: 13 progetti spiegati e realizzati passo per passo. Ricchissimo di indicazioni, trucchetti e soluzioni concrete.

Cascading Style Sheets - The definitive guide, di *Eric Meyer*, O'Reilly, 2000

Cascading Style Sheets 2 - Programmer's Reference, di *Eric Meyer*, Osborne/McGraw Hill, 2001

Eric Meyer on CSS, di *Eric Meyer*, New Riders, 2002

Un libro recente, dall'approccio molto pratico e centrato soprattutto sulle tecniche avanzate (layout e posizionamento dinamico) è questo:

Cascading Style Sheets: Separating Content from Presentation, di *Owen Briggs e altri*, Glasshaus, 2002

Gli autori sono tutti "evangelisti" del nuovo modo di fare layout con i CSS, la gente che sta dietro siti di riferimento come Glish o theNoodleIncident.

Altri due buoni manuali:

Cascading Style Sheets (CSS) by Example, di *Steve Callihan* , Que, 2001

Cascading Style Sheets: Designing for the Web, di *Hakon Wium Lie, Bert Bos, Robert Cailliau*, Addison-Wesley, 1999

L'ultima citazione è per un libro che esplora con un approccio tutto pratico la relazione tra CSS, Javascript e DOM. Il lato dinamico e interattivo dei fogli di stile:

JavaScript + Css + DOM Magic, by *Makiko Itoh* Paperback , New Riders Publishing, 2002