CSS

Benvenuto nel wikibook:

CSS

Vai ai contenuti >>

Fase di sviluppo:CSS

Indice

Voci

Note

CSS: Cascading Style Sheet

Il **linguaggio CSS** permette di impostare tutti gli aspetti di visualizzazione di una pagina web, quali allineamento, layout (disposizione dei contenuti rispetto allo schermo), formattazione del testo e colori.

Esso è nato dalla necessità di separare il contenuto HTML della pagina dalla grafica e dalla visualizzazione. Le regole per la buona stesura di un codice CSS sono state dettate dal W3C.

Finalit€

Questo wikibook ha come obiettivo di fornire al lettore adeguate conoscenze del linguaggio CSS. Il libro può essere consultato da chi possiede un fondamento di HTML, in quanto il CSS è un linguaggio di applicazione grafica alla struttura preesistente.

Regole e sintassi

Regole e sintassi

CSS esterni e interni

Gli stili di una pagina possono essere definiti sia all'interno del file nel quale devono operare, sia in un file a parte.

- Gli **stili esterni** solitamente vengono usati se si hanno molte pagine a cui applicare il medesimo stile. Questo permette di non fare confusione nel lavoro di modifica e revisione.
- Gli **stili interni** vengono usati se le pagine sui cui operare sono veramente poche o se il sito è semplice (questo sistema è sconsigliato perché nasconde una delle potenzialità dei CSS, ovvero la possibilità di applicarli a più pagine contemporaneamente).

Si noti che i CSS vengono validati dal validatore W3C [1] solamente se sono esterni.

Stili esterni

È il sistema più usato ed è compatibile con tutti i browser. Per inserire un collegamento ad uno stile esterno alla pagina, usiamo il tag `<link />` nell'intestazione della nostra pagina:

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

Il seguente sistema di importazione è invece più adatto a risolvere problemi di compatibilità tra vecchi e nuovi browser:

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

Stili interni

Questo sistema permette di inserire dei CSS direttamente all'interno della pagina:

```
[...]
 <style type="text/css">
   ...codice...
 </style>
[...]
```

Si ricordi che questo sistema non permette la validazione da parte del W3C Validator [1]

Stili *inline*

L'ultimo modo per formattare un elemento con un foglio di stile consiste nell'uso dell'attributo `style` applicabile ad ogni elemento HTML. Questo sistema, chiamato talvolta stile *inline*, è utilizzato all'interno degli elementi del codice HTML della pagina e può risultare utile se c'è bisogno di aggiungere uno stile unico per l'elemento.

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

Se ad esempio si vuole formattare un elemento Fieldset in una pagina HTML, bisogna operare come segue:

```
 <fieldset style="color:red;font-size:14px">
```

In questo caso questo elemento, solo questo nella pagina, avrà il testo al suo interno rosso e grande 14px. Si ricorda che questa modalità è fortemente deprecata dalle più recenti versioni dell'HTML.

Sintassi fondamentale

Un foglio CSS è semplicemente un insieme di definizioni dello stile da applicare a determinati elementi della pagina. La loro sintassi è:

```
selettore { dichiarazioni }
```

Le dichiarazioni tra parentesi graffe racchiudono coppie di *proprietà* : *valore* separate da un punto e virgola. Ad esempio:

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

è un foglio di stile con una sintassi valida. In particolare, il valore per la proprietà *font-size* sarà *12px*. Per separare i valori di una stessa proprietà, vengono utilizzate le virgole:

```
 p { font-family: Arial, Helvetica, Sans Serif; }
```

Commenti

All'interno di un foglio di stile è possibile inserire dei **commenti**, ossia porzioni di testo che verranno ignorate dal browser, delimitati da `/*` e `*/`, che si possono estendere su una o più righe.

Questo può risultare utile soprattutto nei casi di collaborazione a coloro che non hanno scritto di prima mano il codice; può però risultare utile anche all'autore stesso, ad esempio per poter riprendere il proprio codice tra le mani dopo un lungo periodo di tempo; inoltre può risultare comodo per eliminare temporaneamente porzioni di codice durante la fase di test delle proprie pagine web.

```
selettore {


}
```

```
                                                    <-- qui si chiude il commento
   ma questo testo verrà letto
   */ <-- questo genera un errore
```

Note

[1] http://jigsaw.w3.org/css-validator/

Selettori

I CSS possono contenere diverse indicazioni sullo stile dei vari elementi della pagina, i quali devono essere identificati dai **selettori**. I selettori possono riferirsi a:

- elementi HTML
- ID
- classi

Esistono oltre a questi anche particolari elementi o classi, chiamate pseudo-classi e pseudo-elementi.

Elementi HTML

È il caso più semplice di selettore, che definisce la visualizzazione di tutti gli **elementi HTML con un determinato nome**. Ad esempio:

```
p { dichiarazioni }
```

definisce lo stile per tutti gli elementi `p` (i paragrafi) della pagina.

ID e classi

Un **id** è un selettore CSS che può essere assegnato ad un solo elemento nella pagina, identificato univocamente dall'attributo `id`. Una classe è un insieme di proprietà che può essere invece assegnato a uno o più elementi HMTL tramite l'attributo `class`. La sintassi è:

```
#nome_id { dichiarazioni }
.nome_classe { dichiarazioni }
```

e, nella pagina HTML, sono richiamate in questo modo:

```
<elemento_html id="nome_id">...</elemento_html>
<elemento_html class="nome_classe">...</elemento_html>
```

Per quanto riguarda le classi, è possibile stabilire dichiarazioni diverse a seconda essa sia attribuita ad un elemento piuttosto che ad un'altra:

```
p.evidenziato { dichiarazioni }
h1.evidenziato { dichiarazioni }
```

imposta uno stile diverso a seconda che la classe evidenziato sia attributa a elementi H1 e elementi P. È possibile inoltre stabilire sottoclassi, la cui sintassi è

```
 elemento.classe.sottoclasse { dichiarazioni }
```

Nell'esempio precedente avremmo potuto quindi aggiungere:

```
 p.evidenziato.grassetto { dichiarazioni }
```

e, nella pagina HTML:

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

Gli pseudo-elementi

Gli **pseudo-elementi** identificano elementi specifici ma che non sono identificati da tag o attributi. La sintassi è:

```
elemento:nome_pseudoelemento { dichiarazioni }
```

I principali pseudo-elementi sono:

- **first-child** identifica il primo elemento figlio contenuto nell'elemento
- **first-letter** identifica la prima lettera del testo contenuto nell'elemento
- **first-line** identifica la prima riga del testo contenuto nell'elemento
- **before** e **after** usati in relazione con la proprietà `content`

Cascade

Per ***cascade*** si intendono le regole con le quali il browser assegna ad un elemento una dichiarazione di stile. Questo è importante nel caso di selettori diversi che indichino lo stesso elemento fornendo dichiarazioni contrastanti.

Nel caso di fogli di stile importati, vengono considerate gli stili del foglio che li importa.

L'origine di un foglio di stile

Prima di analizzare il *cascade*, è bene vedere le possibili origini di un foglio di stile:

- **autore**: è lo stile definito nel sorgente HTML della pagina dal webmaster o dal grafico del sito
- **browser**: i browser conformi agli standard del World Wide Web Consortium devono avere un foglio di stile predefinito da assegnare alle pagine
- **utente**: taluni browser mettono a disposizione all'utente la possibilità di creare un foglio di stile personalizzato, magari tramite un'interfaccia grafica e traducendo poi le impostazioni dell'utente in dichiarazioni CSS

È possibile quindi che questi fogli di stile si sovrappongano e in questo caso vengono applicati in base al cascade.

La specificit€ di un selettore

Altro elemento importante è la **specificit€** di un selettore. Vediamo come calcolarla:

- calcolare il numero di attributi ID nel selettore (= a)
- calcolare il numero degli altri attributi (es. classi) e pseudoclassi del selettore (= b)
- calcolare il numero dei nomi degli elementi HTML (= c)

Concatenare ora i tre numeri nell'ordine a + b + c. Si faccia attenzione a scegliere una base di numerazione opportuna: se, ad esempio, b vale 12, si può scegliere una base esadecimale in modo tale che il numero risulti, ad esempio, 0-c-7 invece che 0-12-7.

Si noti che l'attributo `style` degli elementi HTML possono essere considerati selettori ID, in quanti hanno un riferimento univoco con il tag a cui si riferiscono.

!important

Ultima regola da vedere prima di analizzare le regole del cascade è la dichiarazione ***!important*** la quale indica al browser che una regola è particolarmente importante, dando ad essa priorità rispetto ad eventuali proprietà contrastanti in altre dichiarazioni. Vediamo la sintassi:

```
selettore { [dichiarazioni...]
proprietà: valore !important;
[dichiarazioni...]
}
```

Attribuire uno stile ad un elemento

Il CSS assegna un peso a ciascun blocco di dichiarazioni di stile; nel caso di sovrapposizione di dichiarazioni, vince quella con maggior peso.

Analizziamo finalmente le regole seguite dal browser per attribuire uno stile ad un elemento HTML nella pagina:

- dapprima cerca tutte le dichiarazioni identificate dal **selettore** che corrisponde all'elemento in questione
- la prima analisi riguarda l'**origine** degli stili. Per quanto riguarda le normali dichiarazioni, hanno la precedenza, in ordine, gli stili dell'autore, dell'utente e del browser. Nel caso di dichiarazioni seguite da `!important`, invece, hanno la precedenza, in ordine, gli stili dell'utente, dell'autore e del browser; questo per permettere la piena accessibilità dei contenuti da parte di utenti con handicap visivi, ad esempio. Le dichiarazioni `!important` superano sempre le dichiarazioni normali.
- la seconda distinzione riguarda la **specificit€** del selettore: ha la precedenza il selettore con specificità maggiore
- nel caso due dichiarazioni abbiano stessi peso, specificità e origine vince quella fornita per ultima. Le dichiarazioni dei fogli di stile importati sono considerate come precedenti quasiasi dichiarazione del foglio che importa.

Unit€ di misura, ereditariet€ e box model

Parleremo in questa pagina di alcune delle proprietà più importanti delle pagine di stile, ovvero le unità di misura, la definizione dei valori, il concetto di ereditarietà e i box.

Definire i valori

In alcuni casi, una proprietà accetta solo valori definiti tra una serie di parole chiave; negli altri casi il CSS accetta diversi di valori.

Numeri interi e reali

I **numeri** devono essere indicati senza separatori per le migliaia, usando un punto (.) come separatore decimale. È possibile inoltre specificare il segno del numero (+ o -) anche se la maggior parte delle volte non serve in quanto molte proprietà accettano solo valori positivi.

Grandezze

Le **grandezze** sono usate per definire lunghezze orizzontali e verticali. e sono espresse da un numero (intero o reale) seguito da una unità di misura.

Le principali **unit€ di misura** si dividono in:

- **relative**: specificano una grandezza che dipende da un'altra grandezza. Sono:
  - **em**: è relativa all'altezza font in uso. Se, ad esempio, il font è alto 12pt, 1em varrà 12pt, 2em varranno 24pt.
  - **ex**: funziona come em, ma è relativa all'altezza della lettera `x` nel set di caratteri in uso.
  - **px**: pixel, sono relativi al dispositivo di output (solitamente lo schermo) e alle impostazioni del computer dell'utente
- **assolute**: le unità di misura assolute sono equivalenti a quelle usate nella realtà. Sono:
  - **in**: pollici; un pollice equivale a 2,54 centimetri
  - **cm**: centimetri
  - **mm**: millimetri
  - **pt**: punti — un pt per CSS vale 1/72 di pollice
  - **pc**: pica — 1 pica vale 12 punti

Percentuale

I valori in **percentuale** servono per esprimere percentuali del valore che assume la proprietà stessa dell'elemento padre e vanno espresse con un numero seguito dal simbolo di percentuale (%)

URI

I valori **uri** specificano un percorso assoluto (es. http://esempio/dir/file) oppure uno relativo (nell'esempio di prima, dir/file).
I valori URI vanno usati tramite la notazione `url(percorso)`. `percorso` può essere delimitato da apici (').

Stringhe

Le **stringhe**

- teal - verde acqua scuro
- aqua - verde acqua
- tramite un colore associato alle proprietà di sistema. Ad esempio:
  - background - il colore di sfondo del desktop
  - buttonFace - il colore di sfondo dei pulsanti
  - buttonText - testo dei pulsanti
  - captionText - testo delle etichette
  - grayText - testo disabilitato

Ereditariet€

Proprietà di base

Proprietà di base

Nei moduli seguenti verranno descritte le proprietà di base di questo linguaggio. Si parlerà delle proprietà del testo e dello sfondo.

Sfondo

Grazie ai CSS possiamo facilmente assegnare un colore o un'immagine di sfondo, ripetuta o meno, ad un elemento nel web: dal body al paragrafo, dalla tabella all'input form.

Colore

La proprietà **background-color** permette di definire il colore di sfondo di un elemento.

Sintassi:

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

Il valore deve essere un colore. Il colore verrà inoltre usato per riempire gli spazi non coperti da un'eventuale immagine di sfondo.

Esempio:

```
body { background-color: white; }
p { background-color: #FFFFFF; }
.classe1 { background-color: rgb(0, 0, 0) }
```

background-image

La proprietà **background-image** definisce un'eventuale immagine da usare per lo sfondo dell'elemento.

Sintassi:

```
selettore { background-image: <valore> }
```

dove `<valore>` può essere:

- un URL assoluto o relativo che punti ad un'immagine (in questo caso bisognerà usare la notazione `url(percorso)`
- **none** (valore di default): non verrà applicata nessuna immagine allo sfondo

Esempio:

```
body { background-image: url(immaginesfondo.gif); }
.class2 { background-image: url(http://www.wikibooks.it/immagine.gif); }
```

background-repeat

Imposta il modo in cui l'immagine viene ripetuta sullo sfondo.

Sintassi:

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

I valori possibili sono i seguenti:

- **repeat**: l'immagine di sfondo viene ripetuta sia in senso orizzontale sia in senso verticale
- **repeat-x**: l'immagine di sfondo viene ripetuta in senso orizzontale
- **repeat-y**: l'immagine di sfondo viene ripetuta in senso verticale
- **no-repeat**: l'immagine di sfondo viene visualizzata solo una volta, senza ripetizioni

background-attachment

Questa funzione consente di scegliere se l'immagine dovrà essere fissa sullo sfondo, o muoversi con la pagina.

Sintassi:

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

dove `<valore>` può assumere i seguenti valori:

- **fixed**: l'immagine è fissa sullo sfondo
- **scroll**: l'immagine si muove con la pagina

Posizione

Questa funzione decide come deve essere posizionata l'immagine all'interno della pagina.

Sintassi:

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

dove <valore> può assumere questi valori:

- **top left** in alto a sinistra
- **top center** in altro centrata
- **top right** in alto a destra
- **center left** al centro a sinistra
- **center center** al centro centrata
- **center right** al centro a destra
- **bottom left** in basso a sinistra
- **bottom center** in basso centrata
- **bottom right** in basso a destra
- <valori in percentuale>

Per quanto riguarda in valori in percentuale, essi decidono il valore x e y in percentuale rispetto alla pagina.

Esempio:

```
selettore { background-position: 50% 50% }
```

L'immagine è così mostrata al centro.

Riepilogo

Per una maggiore compattezza e semplificazione del codice, possiamo riassumere tutte le proprieta relative allo sfondo scrivendo semplicemente **background** e tutti i 5 valori possibili a seguire.

Esempio:

```
selettore {
     background: blue
                 url(immaginedisfondo.gif)
                 no-repeat
                 scroll
                 center center;
}
```

Alcuni esempi

Per creare sfondi accattivanti e leggeri in modo semplice, è sufficiente usare con un po' di abilità programmi di grafica e la proprietà `background`. Ad esempio è molto facile creare uno sfondo sfumato che riempie l'intera pagina.

Vogliamo ottenere uno sfondo azzurro nella parte superiore della pagina e blu nella parte inferiore; per ridurre il peso della pagina finale, usiamo un'immagine stretta e alta che poi ripeteremo in orizzontale.
Per prima cosa, usando un programma di grafica come Gimp, creiamo un'immagine con altezza a piacere e larghezza 2px e la riempiamo in verticale con il riempimento sfumato che preferiamo.
Una volta salvata l'immagine nella cartella della pagine, ad esempio come "back.png", usiamo qualche proprietà CSS per applicare lo sfondo alla pagina:

```
selettore {
 background-color: lightblue;

 background-image: url(back.png);
 background-repeat: repeat-x;
 background-position: bottom left;
}
```

Testo base

Verranno adesso descritte le proprietà di base del CSS inerenti al testo:

color

La proprietà **color** specifica il colore del testo contenuto in un qualsiasi elemento HTML, ad esempio una pagina intera, o una tabella, o il contenuto di un fieldset, etc.

Sintassi:

```
selettore { color: <colore>; }
```

Dove <colore> è un valore **esadecimale** (ad esempio #FF0000, che indica il rosso) oppure una parola chiave, come negli esempi seguenti:

```
<style>
     body { color: red; }
     table { color: green; }
</style>
```

Con l'esempio di cui sopra, assegneremo il colore rosso al testo di tutta la pagina, tranne quello contenuto nelle tabelle.

Allineare il testo

Per definire l'allineamento del testo rispetto all'interno dell'elemento che lo contiene si usa la proprietà `text-align`.

Sintassi:

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

Dove <pos> può assumere i valori :

- **left**
- **center**
- **right**

che allineano il testo rispettivamente a sinistra, al centro e a destra, e

- **justify**

che giustifica il testo ai margini dell'elemento.

text-decoration

La proprietà `text-decoration` serve per definire eventuali decorazioni in aggiunta al testo.

Sintassi:

```
selettore { text-decoration: <valore> }
```

Dove <valore> può assumere uno dei seguenti valori:

- **none**: al testo non viene applicata nessuna decorazione
- **underline**: il testo viene sottolineato da una riga continua
- **overline**: ogni riga del testo ha una linea continua sopra

- **line-through**: il testo risulta barrato da una linea continua
- **blink**: il testo lampeggia (da colore a trasparente). Non tutti i browser supportano questo valore

Indentare i paragrafi

Per impostare il rientro per la prima riga del testo si usa la proprietà `text-indent`.

Sintassi:

```
selettore { text-indent: <valore> }
```

Dove `<valore>` è un valore percentuale o affiancato da un'unita di misura (in questo caso è preferibile usare em) che definisce lo spazio tra il margine dell'elemento e la prima lettera del testo.
È possibile assegnare a questa proprietà anche valori negativi.

text-transform

Permette di forzare la capitalizzazione del testo.

Sintassi:

```
selettore { text-transform: <cap> }
```

Dove <cap> può assumere uno dei seguenti valori:

- **none**: la capitalizzazione del testo rimane quella definita nel codice HTML
- **capitalize**: forza la prima lettera di ogni parola alla capitalizzazione maiuscola
- **uppercase**: forza il testo ad essere maiuscolo
- **lowercase**: forza il testo alla capitalizzazione minuscola

letter-spacing e word-spacing

Definiscono lo spazio rispettivamente tra i caratteri e le parole del testo

Sintassi:

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

Dove `<valore>` può essere **normal**, che definisce la spaziatura normale tra le lettere o le parole, un valore espresso in percentuale o una grandezza seguita da un'unità di misura valida.

Spazi tra le linee di testo

La proprietà `line-height` serve invece per definire l'altezza della linea testo, che influisce quindi sulle righe successive quando si va a capo. Il suo valore è una grandezza, normalmente espressa in em.

Gestire gli spazi

Come ultima del testo analizziamo la proprietà `white-space`, che vi capiterà di usare abbastanza poco. Questa proprietà serve a definire il comportamento del browser con gli spazi bianchi nel codice HTML, e può assumere i seguenti valori:

- **normal**: il browser si comporta come normalmente (va a capo agli spazi, unisce le sequenze di spazi);
- **pre** funziona come il tag HTML `<pre>`: gli spazi bianchi non vengono "condensati" e gli accapo sono mantenuti rispetto al sorgente HTML;
- **nowrap**: come *normal*, solo che il browser non spezza il testo se c'è uno spazio;

- **pre-line**: come *pre*, ma gli spazi vengono condensati.

Testo avanzato

In questo modulo verranno trattate le proprietà per definire il font di un testo in una pagina, che sono principalmente:

Tipo di carattere

Permette di definire il tipo di carattere usato per la visualizzazione del testo.

Sintassi:

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

dove `<valore>` è una sequenza di uno o più tipi di carattere indicati tra apici nell'ordine in cui verranno usati nel caso non siano installati sul sistema operativo dell'utente. È possibile inoltre definire una famiglia generale di caratteri tramite i seguenti valori:

- **serif**: le serif sono le grazie, ovvero gli abbellimenti delle lettere, che presentano caratteri come il Times New Roman o il FreeSerif. È usato soprattutto per la stampa, in quanto la lettura delle grazie su schermo risulta più faticosa. Esempio: Testo con serif.
- **sans-serif**: sono sans-serif (senza grazie) font come il Verdana, l'Arial, il FreeSans, l'Helvetica, ecc... Sono usati prevalentemente per lo schermo. Esempio: Testo senza serif.
- **cursive**: i font cursive presentano caratteri più simili alla grafia umana piuttosto che a quelli di stampa. Esempio: Cursive.
- **fantasy**: caratteri che uniscono la visualizzazione del testo ad elementi decorativi. Esempio: Testo fantasioso.
- **monospace**: caratteri tipo telescrivente o macchina da scrivere e quindi monospaziati, come il Courier. Esempio: Testo monospaziato.

Ad esempio possiamo definire così il font di una pagina web:

```
  body { font-family: Verdana, Arial, Helvetica, sans-serif; }
```

Dimensione del testo

Definisce l'altezza del testo che dipenderà anche dal tipo di carattere usato. Normalmente questa proprietà è definita in em oppure in pt.

Sintassi:

```
 selettore {
```

- bolder
- lighter

che definiscono un testo normale, grassetto, grassetto pesante e più "leggero". Normalmente il valore normal corrisponde al peso numerico 500.

Testo corsivo

La proprietà `font-style` permette di definire un font in corsivo italico o obliquo.

Sintassi:

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

dove `valore` può essere **normal**, **italic** o **oblique**.

Maiuscoletto

Per impostare un testo maiuscoletto si usa la proprietà `font-variant`.

Sintassi:

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

dove `valore` può assumere il valore **normal** o **small-caps** (maiuscoletto). Normalmente il testo maiuscoletto presenta i caratteri minuscoli come delle riduzioni dei corrispettivi caratteri maiuscoli.

La proprietà riassuntiva font

La proprietà **font** permette di riassumere le proprietà viste in questo modulo.

Sintassi:

```
selettore { font: <font-style> <font-variant> <font-weight> <font-size>/<line-height> <font-family>;}
```

Notare come line-height vada impostato subito dopo font-size, separato da quest'ultimo dalla barra "/".

È possibile inoltre indicare al posto dei diversi valori font, una delle seguenti famiglie usate dal sistema in uso:

- **caption**: il font usato per le etichette ad esempio dei pulsanti
- **icon**: il font usato per le etichette delle icone
- **menu**: il font usato nei menu
- **message-box**: il font usato per le finestre di dialogo
- **small-caption**: il font usato per le etichette dei piccoli controlli
- **status-bar**: il font usato per la barra di stato delle finestre

font-stretch e font-size-adjust

Nelle specifiche CSS 2 sono definite anche le proprietà *font-stretch* e *font-size-adjust*, sfortunatamente poco usate a causa della loro scarsa compatibilità con i vari browser; pertanto non ci soffermeremo su di esse in questa sede.

Liste

In questo modulo verrà trattata la formattazione delle liste che rispetto a quella fornita dall'HTML è sicuramente più gestibile.

list-style-type

Questa proprietà permette di stabilire, analogamente all'attributo `type` delle liste HTML, il simbolo usato come punto elenco, sia di una lista ordinata sia di una non ordinata. Può assumere i seguenti valori:

- liste ordinate:
  - **decimal**: numeri decimali, partendo da 1
  - **decimal-leading-zero**: numeri decimali con zeri aggiuntivi (01...99)
  - **lower-roman** e **upper-roman**: numeri romani minuscoli (i, ii, iii, iv, v, ...) e maiuscoli (I, II, III, IV, V, ...)
  - **lower-latin/lower-alpha** e **upper-latin/upper-alpha**: lettere dell'alfabeto latino minuscole (a, b, c, ... z) e maiuscole (A, B, C, ... Z)
  - **lower-greek**: lettere dell'alfabeto greco minuscole (α, β, γ, ...)
  - **georgian**: numerazione Georgiana (an, ban, gan, ...)
  - **armenian**: numerazione Armena
- liste non ordinate: sono disponibili i normali valori **disc**, **square** e **circle**, che funzionano come l'attributo `type` delle liste HTML

Impostando la proprietà al valore **none** la lista apparrà, se non viene specificata un'immagine, senza punto elenco.

list-style-image

CSS permette di definire come punto elenco un'immagine; nel caso sia stato impostato un valore sia per `list-style-type` sia per questa proprietà, viene mostrata solo l'immagine (se è valida).

Sintassi:

```
list-style-image: url(<percorso>);
```

`<percorso>` è quindi un percorso che identifica un'immagine.

list-style-position

Questa proprietà permette di impostare la posizione del punto elenco di ogni elemento `li`. Può assumere i seguenti valori:

-

list-style

Esiste inoltre la proprietà riassuntiva `list-style` che riassume la formattazione delle liste. La sintassi è:

```
list-style: <list-style-type> <list-style-position> <list-style-image>;
```

Disegnare la struttura della pagina

Disegnare la struttura della pagina

In questo modulo verranno trattate le diverse proprietà legate al box model, già analizzato precedentemente nel corso del libro.

Gestione delle dimensioni e dell'overflow

Le proprietà **width** e **height** permettono di impostare rispettivamente la larghezza e l'altezza del contenuto del box in questione. Il valore di queste proprietà può essere una grandezza oppure un valore in percentuale riferito all'elemento genitore; è possibile inoltre usare un valore **auto**, che funziona in modo diverso per le due proprietà:

- per quanto riguarda l'altezza, con un valore auto verrà determinata dal contenuto dell'elemento stesso
- per quanto riguarda la larghezza, con un valore auto assume le dimensioni dell'elemento genitore

La gestione delle dimensioni del contenuto di un elemento era tuttavia già possibile in HTML attraverso gli attributi `width` e `height`, anche se non era applicabile a tutti i tag.

Possono risultare utili inoltre le proprietà **min-height**, **max-height**, **min-width** e **max-width** che servono per impostare le dimensioni minime o massime che può assumere l'elemento in questione.

La proprietà **overflow** permette di impostare il comportamento che deve avere un elemento nel caso il suo contenuto superi le dimensioni previste dalle proprietà width e height (questo fenomeno è appunto chiamato con il termine tecnico *overflow*). Può assumere uno tra i seguenti valori:

- **hidden**: il contenuto in eccedenza viene ignorato e non viene quindi visualizzato
- **visible**: le dimensioni del box vengono riadattate in modo tale da mostrare il contenuto per intero
- **scroll**: vengono aggiunte al box delle barre di scorrimento orizzontali e/o verticali che permettono all'utente di vedere per intero il contenuto semplicemente scorrendolo
- **auto**: il browser si comporta secondo le sue impostazioni

I margini

La gestione dei margini prevede l'uso di quattro proprietà `margin-top`, `margin-bottom`, `margin-left`, `margin-right` che specificano rispettivamente il margine superiore, inferiore, sinistro e destro dell'elemento. Possono assumere una grandezza, un valore in percentuale o **auto** (che normalmente imposta margini nulli). Si noti che i valori per i margini possono essere anche negativi; ad esempio:

```
margin-left: -10px;
```

mostra l'elemento spostato di 10px a sinistra rispetto alla posizione che dovrebbe assumere normalmente.

Esiste inoltre una proprietà riassuntiva `margin` che permette di impostare insieme le quattro proprietà relative ai marigni. In base a quanti valori vengano forniti, si comporta in maniere differenti:

- se ne viene indicato solo uno, questo si riferirà a tutti e quattro i margini
- se ne vengono indicati due, il primo si riferisce ai margini orizzontali e il secondo a quelli verticali
- se ne vengono indicati tre, il primo e il terzo sono riferiti rispettivamente ai margini superiore e inferiore, il secondo a quelli verticali
- se ne vengono indicati quattro, il primo si riferirà al margine superiore, il secondo a quello destro, il terzo a quello inferiore e il quarto a quello sinistro

Il padding

Il padding, ovvero lo spazio tra il bordo di un contenitore ed il suo contenuto, è gestito tramite quattro proprietà `padding-top`, `padding-bottom`, `padding-left`, `padding-right` che specificano rispettivamente il padding superiore, inferiore, sinistro e destro dell'elemento. Possono assumere una grandezza o un valore in percentuale.

Esiste inoltre una proprietà riassuntiva `padding` che permette di impostare insieme le quattro proprietà relative al padding; si comporta allo stesso modo di `margin`.

I bordi

I bordi in CSS sono gestibili tramite le proprietà:

```
border-<lato>-style
border-<lato>-width
border-<lato>-color
```

che specificano rispettivamente lo stile, lo spessore e il colore del lato (*top*, *bottom*, *left*, *right*) indicato.

Per quanto riguarda lo stile dei bordi, è possibile specificare uno dei seguenti valori:

- **none**: il bordo non viene visualizzato (es:)
- **dotted**: una successione di punti (es:)
- **dashed**: il bordo è visualizzato con dei trattini (es:)
- **solid**: il bordo è una linea continua (es:)
- **double**: il bordo è costituito da due linee separate da uno spazio bianco (es:)
- **groove** e **ridge**: il bordo risulta incassato o in rilievo tramite un gioco di colori (es: e)
- **inset** e **outset**: l'intero box risulta incassato o in rilievo tramite un gioco di colori (es: e)

Per quanto riguarda il colore, è possibile specificare un valore di colore valido oppure il valore **transparent** (bordo trasparente che influisce però sulla visualizzazione della struttura).

Per quanto riguarda lo spessore dei bordi, la proprietà `border-<lato>-width` può assumere i valori:

- **thin** (un bordo sottile)
- **medium** (un bordo normale)
- **thick** (un bordo spesso)
- un valore di grandezza, normalmente espressa in px, che deve essere positiva

Proprietà riassuntive dei bordi

Anche per la definizione dei bordi esistono delle proprietà riassuntive. Sono:

```
border-style
border-width
border-color
```

Queste tre proprietà specificano lo stile, lo spessore e il colore dei quattro bordi assieme. Possono avere da uno a quattro valori e il loro funzionamento è simile a quello di `margin`.

Esistono inoltre le proprietà:

```
border-<lato>: <border-<lato>-width> <border-<lato>-style> <border-<lato>-color>
```

che definiscono i diversi aspetti dei bordi per un singolo lato.

Infine possiamo utilizzare anche una proprietà la cui sintassi è:

```
border: <border-width> <border-style> <border-color>
```

che specifica insieme i valori delle tre proprietà dei bordi per tutti e quattro i lati.

Posizionamento

I CSS permettono di posizionare all'interno di una pagina oggetti che varino o restino fissi indipendentemente dalla dimensione in cui viene visualizzata la pagina e/o allo scorrere di questa.

Nella pagina, un elemento può essere posizionato secondo tre modi differenti:

- seguendo il normale andamento della pagina previsto dall'HTML
- senza alcun legame con il resto del flusso degli elementi
- allineandosi in modo tale che gli elementi si dispongano al lato (questa tecnica, chiamata in gergo tecnico *floating*, è simile al funzionamento dell'attributo `align` delle immagini)

Posizionarsi relativamente agli altri elementi della pagina

Per posizionare un elemento rispetto agli elementi della pagina secondo quanto previsto dal puro HTML, usiamo la proprietà **position** con questi valori:

- **static** (valore di default per quasi tutti gli oggetti HTML)
- **relative**: l'elemento è posizionato seguendo il normale flusso degli elementi nella pagina, tramite l'uso di quattro proprietà (left, right, top e bottom) che assumono come valore una grandezza e specificano la distanza (positiva o negativa) dell'elemento rispetto alla posizione che questo dovrebbe occupare normalmente nella pagina. Ad esempio:

```
b {position: relative; top: -5px;}
```

Con questo codice, tutti gli elementi bold appariranno spostati di 5 pixel verso l'alto rispetto alla linea del paragrafo. In particolare:

- **top** indica la distanza del bordo superiore dell'oggetto dal bordo superiore dell'elemento in cui è contenuto
- **bottom** indica la distanza del bordo inferiore dell'oggetto dal bordo inferiore dell'elemento in cui è contenuto
- **left** indica la distanza del bordo sinistro dell'oggetto dal bordo sinistro dell'elemento in cui è contenuto
- **right** indica la distanza tra il bordo destro dell'oggetto dal bordo destro dell'elemento in cui è contenuto

Posizionamento assoluto e fisso

Per posizionamento assoluto (dal latino *ab solutum*, sciolto dal resto) o fisso si intende un posizionamento che non segue il flusso degli elementi. È fissato tramite la proprietà `position` con l'uso dei valori:

- **absolute**: l'elemento è posizionato rispetto al suo blocco contenitore (la pagina o un altro elemento block-level a sua volta posizionato con la medesima proprietà) tramite le proprita left, right, top e bottom. L'elemento risulterà quindi legato allo scorrere della pagina
- **fixed**: l'elemento è posizionato rispetto alla finestra sempre tramite le quattro proprietà left, right, top e bottom e non è legato allo scorrimento della pagina. Ad esempio:

```
  div.bann {
  position: fixed;
  bottom: 0px;
  right: 0px;
}
```

mostra l'elemento fisso nell'angolo in basso a destra dello schermo (può servire, ad esempio, per la creazione di banner o simili)

Usare il floating

Il *floating*

rincomincia, per ripigliar poi nome di lago dove le rive, allontanandosi di nuovo, lascian l'acqua distendersi e rallentarsi in nuovi golfi e in nuovi seni... (da *I promessi sposi, capitolo primo*, A. Manzoni)

In questo caso il paragrafo qui a fianco, invece di disporsi a destra del blocco, si dispone allineandosi con il suo lato inferiore: ciò è causato dall'uso della dichiarazione CSS `clear: left`. L'effetto è simile a quello del blocco senza floating ma è stato ottenuto con due modalità differenti. Ovviamente la proprietà `clear` risulta comodo con molti elementi flottanti nella pagina.
La proprietà `clear` è usata spesso in relazione all'elemento `br` in questo modo:

```
<br style="clear: both" />
```

che interrompe il testo effettuando il clear in maniera semplice e pulita.

Appendici

Contenuto generato

Una delle grandi funzionalità del CSS consiste nella possibilità di generare automaticamente il contenuto degli elementi della pagina.
Per fare ciò si usa la proprietà **content**, che purtroppo non è supportata sempre correttamente da tutti i browser.

content

La proprietà **content** imposta il contenuto dell'elemento. Può assumere come valore:

- **none**: non viene generato alcun contenuto
- una combinazione di:
  - una **stringa**
  - un **percorso** che indica una risorsa da inserire (es. un'immagine)
  - un **contatore** (vedi nel prossimo paragrafo)
  - **open-quote** o **close-quote**: virgolette di apertura o di chiusua aumentando di uno il livello di indentazione
  - **no-open-quote** o **no-close-quote**: non mostra virgolette ma aumenta il livello di indentazione

Questa proprietà è usata soprattutto con le pseudoclassi :before e :after, ad esempio in questo modo:

```
body:after {
 content: 'The end';
}
```

imposta il contenuto finale del corpo della pagina. Le pseudoclassi :before e :after non sono però supportate correttamente da IE.

Uso delle virgolette

Per specificare i diversi tipi di virgolette da usare per i diversi livelli di indentazione usiamo la proprietà **quotes** la cui sintassi è:

```
 quotes: [apertura1 chiusura1] [apertura2 chiusura2] [apertura''n''
chiusura''n'']
```

Per spiegare l'uso facciamo un esempio:

```
p { quotes: '<<' '>>' '"' '"'; }
blockquote:before { content: open-quote; }
blockquote:after { content: close-quote; }
```

```
<!-- nell'HTML -->
<p>Luigi disse:
 <blockquote>
 Lorem ipsum dolor sit amet, ...<br/>
 Infine disse:
```

```
   <blockquote>
    Ma questa è un'altra storia.
   </blockquote>
  E finì.
  </blockquote>
  Questo è quello che disse Luigi.
</p>
```

che fornisce come output:

Luigi disse:

> << Lorem ipsum dolor sit amet, ...
> Infine disse:
>
> "Ma questa è un'altra storia."
>
> E finì. >>

Questo è quello che disse Luigi.

Lavorare con i contatori

CSS permette inoltre di generare automaticamente contenuti numerati. Per fare ciò è possibile usare due proprietà:

```
counter-reset: <nome1> <valore1> <nome2> <valore2> <nome''n''> <valore''n''>
counter-increment: <nome1> <valore1> <nome2> <valore2> <nome''n''> <valore''n''>
```

La prima proprietà resetta il contatore chiamato `<nome''n''>` al valore `<valore''n''>`. Se omesso il valore, il contatore viene impostato a 0. La seconda proprietà incrementa il valore del contatore chiamato `<nome''n''>` di `<valore''n''>` posizioni. Se omesso il valore, il contatore viene incrementato di 1.

Per accedere al valore corrente del contatore si usa la funzione

```
counter(<nome>, <stile>)
```

dove `<nome>` è il nome del contatore e `<stile>` è lo stile del contatore (funziona in maniera identica alla proprietà `list-style-type`

```
   <p>...</p>
   <h2>Il corpo</h2>
   <p>...</p>
  <h1>La formattazione del testo</h1>
   <p>...</p>
   <h2>I tag di formattazione</h2>
   <p>...</p>
   <h2>Il tag font</h2>
   <p>...</p>
```

produrrà come output

```
Capitolo 1. La pagina
 ...
 Sezione 1.1. L'intestazione
 ...
 Sezione 1.2. Il corpo
 ...
Capitolo 2. La formattazione del testo
 ...
 Sezione 2.1. I tag di formattazione
 ...
 Sezione 2.2. Il tag font
 ...
```

Altre proprietà

In questo modulo analizzeremo alcune proprietà CSS che non sono state presentate nel corso del libro ma sono comunque rilevanti.

cursor

Questa proprietà serve a modificare l'aspetto del cursore quando si posiziona sull'elemento in questione. Può assumere come valore una combinazione dei seguenti valori (posizionarsi sopra il testo per vederne l'effetto):

- **auto**: il cursore automatico per il tipo di elemento
- **default**: il cursore predefinito di sistema (normalmente una freccetta)
- **pointer**: il puntatore usato ad esempio per i link (normalmente la manina con l'indice puntato)
- **text**: il cursore usato per il testo selezionabile (solitamente il caret I)
- **wait**: indica di aspettare (solitamente è una clessidra)
- **progress**: l'applicazione sta lavorando, ma può comunque reagire ai comandi (solitamente una freccia con accanto una clessidra)
- **crosshair: la croce usata per puntare con precisione (a forma di "+")**
- **help: usato per indicare la possibilità di visualizzare un aiuto relativo all'elemento in questione (spesso un punto interrogativo)**
- **move: il cursore visualizzato su elementi spostabili (normalmente la manina chiusa)**
- **e-resize**, **ne-resize**, **nw-resize**, **n-resize**, **se-resize**, **sw-resize**, **s-resize**, **w-resize**: le frecce usate per i ridimensionamenti. Le lettere che precedono `-resize` indicano il verso — N = nord (north), S = sud (south), E = est (east), W = ovest (west).
- un **percorso** indicante un file di cursore

Il browser scorrerà la lista di valori fino a quando non incontrerà un cursore valido o rappresentabile. Altrimenti userà il cursore automatico.

display

La proprietà **display** serve a definire la visualizzazione e il *rendering* dell'elemento da parte del browser. Può assumere, tra gli altri, i valori:

- **inline**: l'elemento viene visualizzato come in linea
- **block**: l'elemento è visualizzato come blocco
- **list-item**: l'elemento viene visualizzato come un oggetto di una lista (formattabile tramite le proprietà `list-style`)
- **none**: l'elemento non viene visualizzato e non influisce sulla visualizzazione della pagina

Questa proprietà può servire:

- nell'ambito di pagine XML, i quali elementi non hanno associato un *rendering*
- ad esempio per poter posizionare e gestire come un blocco (quindi impostare dimensioni, posizioni, ecc...) elementi in-line come i link

È invece molto interessante l'uso della dichiarazione `display: none;` che, soprattutto se utilizzata insieme a linguaggi dinamici come JavaScript, permette di nascondere o visualizzare elementi della pagina con un semplice clic dell'utente.

visible

La proprietà **visible** determina se un elemento è visibile o meno all'interno della pagina. Può assumere come valori:

- **visible**: l'elemento risulta visibile
- **hidden**: l'elemento risulta nascosto ma occupa comunque spazio nella pagina

Differenze tra display e visible

Abbiamo visto che esistono due modi per rendere invisibile un elemento HTML: il primo è quello di usare la proprietà `display: none;` e il secondo prevede l'uso di `visible: hidden;`.
La differenza tra i due metodi è che, mentre con la prima dichiarazione l'elemento viene considerato come se non esistesse del tutto, con il secondo l'elemento influisce ancora sulla visualizzazione della pagina e nel flusso degli elementi che lo circondano.

Compatibilit€

Come pure molte altre tecnologie usate per la programmazione web, anche il linguaggio CSS soffre di **problemi di compatibilit€** tra i diversi browser disponibili.
Infatti, sebbene ormai sia possibile adottare soluzioni accettabili e compatibili, persistono ancora nei diversi browser differenze con le specifiche ufficiali del WWW Consortium.

I problemi sono legati soprattutto:

- alla compatibilità tra versioni precedenti dei browser (**retro-compatibilit€**)
- alle **differenze di *rendering*** (visualizzazione) tra i diversi browser

L'affidabilit€ dei diversi browser

Attualmente nessuno dei browser in commercio offre il supporto completo agli standard W3C, tuttavia molti vi si avvicinano:

- **Internet Explorer**

```
<!--[if IE]>
 <link rel="stylesheet" type="text/css" href="stileIE.css">
 <![endif]-->
```

L'ultimo suggerimento è quello di testare sempre le proprie pagine su diversi motori di rendering; è infatti oramai possibile sviluppare, come già detto, soluzioni accettabili e compatibili con più broswer.

Fonti, licenze e autori delle immagini

Licenza