Massimiliano Carnevale

http://www.maniegrafiche.it

Guida pratica ai CSS

Come usare da zero i fogli di stile con stile.

Indice

Introduzione al mondo dei CSS

Fino a poco tempo fa la maggior parte delle pagine web erano un misto di contenuti e presentazione con l'aspetto grafico costituito su tabelle; tutto ciò poteva e può comportare difficoltà quali: **problemi di modifica** di un layout, **problemi di ritrovare parti di codice affogate in tabelle e sottotabelle**, **pesantezza** delle stesse a causa dello stesso motivo.

I **css** permettono di ovviare a tale inconveniente: con essi si può *costruire* il contenuto e successivamente applicarvi lo stile richiesto semplicemente variando la pagina dello stile.

Tutto questo porta poi ad altri vantaggi: è **possibile associare uno stile per ogni dispositivo che richieda la pagina**; uno per i browser classici, uno per le web tv, uno per i telefonini e videofonini. Volendo poi si possono creare molti stili anche per lo stesso dispositivo.

Si potrebbe creare una versione più leggera, con poche immagini o in bassa risoluzione per chi abbia una connessione lenta ed una più pesante, con risoluzioni maggiori per i fortunati possessori di banda larga.

I vantaggi sono molti anche confrontati con (i pochi) svantaggi.

Seguiteci: cercheremo di farmi apprendere questo nuovo mondo con facilità ed in modo progressivo.

Come si dice: è più facile a dirsi che a farsi.

Includere i css

In questo capitolo inizieremo a vedere un po' di codice, in particolare vedremo come inserire i css nelle nostre pagine.

Esistono 4 metodi per utilizzare i css:

1. *richiamarli direttamente nelle pagine come stile dell'elemento usando l'attributo style*
2. *utilizzare l'elemento **<style>** nell'head dei nostri documenti XHTML*
3. *richiamando una pagina di stili esterna attraverso l'elemento **<link>***
4. *utilizzando la direttiva @import in **<style>**.*

Tutti e 4 i metodi permettono di fare la stessa cosa anche se con alcune differenze. Il vantaggio di utilizzare fogli di stili esterni è comunque notevole: nel caso in cui volessimo cambiare look ad un documento ci basterà modificare il foglio di stile.

Molto simile è il comportamento della direttiva @import che in più esclude alcuni vecchi browser che non hanno implementato i css in modo corretto.

Personalmente è il metodo che preferisco.

Gli altri due metodi ci possono esser utili per provare al volo ciò che abbiamo pensato: son molto veloci e pratici per tale motivo ma non mi sento di consigliarveli se non in casi particolarissimi che approfondiremo in seguito.

Altro punto importante che iniziamo ad accennare è l'ereditarietà e la priorità delle regole impostate dai css: se diamo due stesse regole allo stesso elemento, magari una nel foglio di stile esterno ed una nell'elemento **<style>** il motore interpreterà quello a maggior priorità.

In tal caso la regola inserita in <style> andrà a *sovrascrivere* quella del css esterno. Non preoccupatevi comunque anche questo argomento sarà richiamato successivamente.

Ritorniamo all'argomento principale di questa lezione e vediamo come inserire *praticamente* i nostri stili.

---

1. Richiamare i css nelle pagine utilizzando l'attributo style.

L'uso in tal caso è molto semplice.

Supponiamo di voler colorare di rosso un paragrafo. Con questo metodo basta dichiarlo direttamente nell'elemento <p>:

```
<p style='color:red'>Hello World!</p>
```

ed il gioco è fatto.

2. Utilizzare l'elemento <style> nell'head dei nostri documenti XHTML.

Con questo metodo è sufficente inserire le proprietà direttamente nell' *head*:

```
<html>
 <head>
  <style type='text/css' media='all'>
   p {color:red; }
  </style>
 </head>
 <body>
  <p>Hello World!</p>
 </body>
</html>
```

Style può avere due attributi:
type obbligatorio, che ha la funzione di specificare il tipo; in pratica sarà però sempre e solo *text/css* e **media** che serve a specificare a quale piattaforma applicare un foglio di stile (ci torneremo in seguito).

3. richiamare una pagina di stili esterna attraverso l'elemento <link>.

```
<link rel='stylesheet' type='text/css' href='css.css' media='all'>
```

Questo metodo prevede 4 possibili attributi:
type e **media** con la stessa funzione appena vista, **href** che serve a specificare i percorso del foglio di stile (obbligatorio) e **rel** che specifica la relazione tra il foglio di stile ed il documento html. Può assumere due valori: *stylesheet* e *alternate stylesheet*.

4. utilizzare la direttiva @import in <style>

Questo metodo, il mio preferito, prevede di indicare il percorso del foglio di testo entro parentesi tonde nell'elemento style nella forma

```
<style>
  @import url(http://miosito.ext/css.css);
</style>
```

Maggiori approfondimenti potrete trovarli per questo metodo nel prossimo articolo.
Abbiamo imparato come inserire i css nelle nostre pagine: prossimamente vedremo come sfruttare i css nel massimo della loro potenza. Prima però occorrerà iniziare a capire la classificazione degli elementi di un documento. Tutto questo lo vedremo in una delle prossime lezioni.

Metodo @import

L'uso del motodo @import può esser sfruttato anche per permettere l'accesso o l'esclusione di alcuni browser.

Iniziamo con il dire che il W3C ammette una certa possibilità di discrezione nel codice di importazione.

Ammette ad esempio le forme **@import url("style.css"); @import url(style.css);** o anche **@import "style.css";** .
Ci sono però anche altre forme accettate in modo diverso da browser a browser. Questo potrà aiutarci in un modo molto semplice per creare fogli di stile diversi per diversi browser.

Se, ad esempio, volessimo creare due css uno per Internet Explorer versione 4, 5 e 5.5 potremmo richiamarlo con la sintassi **@import\ url("style_ie.css");**, mentre se volessimo inserire un foglio di stile per tutti gli altri tranne i precendenti basterà richiamare il css in questo modo: **@import "style.css"/**/;**. *(Vedremo successivamente che ci sono altri metodi altrettanto efficaci; ma avere a disposizione due tecniche per fare la stessa cosa, non potrà che esserci utile, no?)*

E' da notare che tale metodo può esser applicato in moltissimi casi. In questa pagina son catalogate 33 possibilità diverse. L'autore ha indicato anche se tali possibilità son approvate dal validatore W3C.

Come leggere la tabella?
Semplicemente: per ogni regola di @import son indicate con una + i browser che la supportano. Mescolando adeguatamente le varie regole potremmo costruire un foglio di stile diverso per ogni browser, una comodità che ci farà utile tra un pò quando noteremo come browser diversi interpretino la strssa regola in modi, per così dire, creativi.

Regole dei Css

Siamo arrivati al punto di dover applicare delle regole alle nostre pagine. Come si fa a dire attraverso i css che il nostro paragrafo deve contenere testo di colore rosso? *Come facciamo a dire di applicare uno sfondo giallo ed un margine nullo a tutti i titoli h1? Come diciamo al css che tutti i titoli h1 ed h2 devono avere padding nullo?*

p { color: red;} nel primo caso,
h1 { background-color: yellow; margin: 0; } nel secondo e
h1, h2 { padding: 0; } nell'ultimo. Tutto qua!
Vediamo nel dettaglio la prima regola:

p { color : red; }

selettore proprietà valore
Come avrete notato la regola è divisa in due parti: la prima è il **selettore**, cioè la parte che indica a quale sezione del documento applicare la regola (o le regole) indicata dal **blocco delle dichiarazioni** racchiuso tra parentesi graffe.

Il blocco delle dichiarazioni è un insieme di dichiarazioni formato da una proprietà e da un valore da assegnare alla stessa, divisi tra loro da due punti e terminato da un punto e virgola. Regole successive possono evidentemente applicate come nel caso del titolo h1 del secondo esempio.

E' possibile associare le stesse regole a più selettori come indicato nel terzo esempio ed applicare regole diverse a seconda della disposizione degli elementi nella pagina,

argomento che potrete approfondire nella pagina I selettori.

Un caso particolare di regola è riferita a tutte quelle proprietà, quali ad esempio margin, padding, border, che potrebbero avere valori diversi nei quattro lati.
E' possibile quindi associare valori variabili ai bordi destro, sinistro, superiore ed inferiore con questa sintassi:

```
p {
  margin-top: 0;
  margin-right: 2px;
  margin-bottom: 3px;
  margin-left: 5px;
}
```

La cosa interessante è che in tali situazioni è possibile applicare una regola comune e *risparmiare* codice con la c.d. **sintassi abbreviata** in tal modo:

```
p {
  margin: 0 2px 3px 5px;
}
```

con i valori *per*, rispettivamente, *il lato alto, destro, basso e sinistro* (in senso **orario** partendo dall'alto per capirci :))

E' possibile associare una regola solo per i lati alto e basso ed una per destro e sinistro:

```
p {
   margin: 5px 0;
}
```

con il *primo valore riferito ai lati orizzontali ed il secondo per quelli verticali.*

Se si indicano infine solo tre valori,

```
p {
  margin: 3px 10px 2px;
}
```

*essi saranno riferiti rispettivamente al lato superiore, al lato **destro e sinistro** ed al lato inferiore*; in pratica è equivalente a scrivere: p {margin: 3px 10px 2px 10px;}.

Le classi e gli ID

Classi ed ID sono due attributi del codice xHTML, importantissimi per i fogli di stile. Permettono infatti di associare un'insieme di regole ad un solo selettore (ID) o ad un gruppo (class).

Immaginiamo di voler assegnare a tutti i commenti di un blog il colore grigio ed una grandezza di 10px. Senza l'esistenza di questi due particolari selettori non riusciremmo a farlo con i css, dovremmo usare il tag font (orrore!!!).

Con i CSS ci basta assegnare una classe ai commenti, chiamandola ad esempio *"commenti"* e dichiare tali proprietà nel css in questo semplice modo:

xHTML: [....]

```
<p class="commenti">Primo commento</p>
<p class="commenti">Secondo commento</p>
```

e così via

CSS:

```
commenti {color:gray: font-size:10px;}
```

Notate il puntino (.) prima del nome della classe: serve ad esplicitare che le regole inserite saranno riferite a tutti gli elementi con quella classe esplicita.
E' importante far notare che la classe così definita si applicherà a qualsiasi elemento della pagina con attributo class="commenti".

Scrivendo quindi questo codice xHTML

```
<p class="commenti">Commento</p> Bla bla <div class="commenti">Blocco di testo</div>
```

Anche il contenuto del div sarà stampato con le stesse proprietà stabilite per i commenti. Questo può esser voluto o meno: in tal caso basterebbe assegnare nel css:

```
p.commenti {proprietà:valori}
```

affinché le regole della classe commenti vengano applicate solo ai paragrafi (<p>).

Una sorta di classe particolare è l'ID, che si differenzia di base dalla classe per il fatto che è applicabile SOLO ad un elemento è l'ID. **In pratica può essere applicato UNIVOCAMENTE ad un solo elemento.**

Per definirlo nell'xHTML

```
<p id="id-numero1">Testo </p>
```

e nel css

```
#id-numero1 {proprietà:valore;}
```

Stesse norme per le classi, con la differenza di dover anteporre in questo caso il cancelletto (#) al posto del punto.

I selettori css

Abbiamo già visto nell'articolo dedicato alle regole dei CSS cosa siano i **selettori**: un oggetto a cui è possibile associare valori e proprietà.

Esistono vari tipi di selettori:

- **Selettore Universale** *(Universal Selector)*
- **Selettore di Elementi** *(Type Selectors)*
- **Selettori Discendenti** *(Descendant Selectors)*
- **Selettore Figlio** *(Child Selectors)*
- **Pseudo-classe :first-child** *(The :first-child pseudo-class)*
- **Pseudo-classi dei links** *(The link pseudo-class)*
- **Selettori Adiacenti** *(Adjacent Selectors)*
- **Selettori di Attributi** *(Attribute Selectors)*
- **Selettori di Classi** *(Class Selectors)*
- **Selettori di ID** *(ID Selectors)*
- **Pseudo-classi :first-line e :first:letter** *(The :first-line e :first:letter pseudo-class)*

Selettore Universale *

Permette di associare a tutti gli elementi della pagina delle regole.
Es:

```
* {color:black; margin:0; }
```

Associa a qualsiasi elemento della pagina il colore nero e margine nullo.

Selettore di Elementi

E' il più diffuso. Permette di associare regole ad uno o più elementi:

```
h1 { font-style: Georgia, serif; }
h2 { font-style: Georgia, serif; }
h3 { font-style: Georgia, serif; }
h4 { font-style: Georgia, serif; }
h5 { font-style: Georgia, serif; }
h6 { font-style: Georgia, serif; }
```

C'è la possibilità di associare l'insieme di regole a più elementi separandoli con una virgola:

```
h1, h2, h3, h4, h5, h6 { font-style: Georgia, serif; }
```

scrittura equivalente alla precedente.

Selettori Discendenti

Questa opzione ci permette di associare regole a particolari elementi se e solo se contenuti in altri elementi definiti:

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

Associa il colore rosso solo agli elementi *strong* contenuti in div.

```
<body>
 <strong>Questo non sarà rosso in quanto non contenuto in un un div</strong>
 <div>Il testo successivo, <strong>questo in particolare</strong> sarà mostrato in
rosso</div>
 <div>Anche <p><strong>questo testo</strong></p> sarà mostrato in rosso; non è
importante che i due  elementi siano adiacenti o figli.</div>
</body>
```

Selettore Figlio

Potrebbe essere una buona soluzione a molti problemi, purtroppo non è supportato, così come il selettore adiacente, da Internet Explorer. Secondo le specifiche permette di associare regole ad elementi solo se contenuti da un altro indicato nel selettore e da nessun altro.

```
div > strong { color: red; }
```

Es:

```
<body>
  <div>Il testo successivo, <strong>questo in particolare</strong> sarà mostrato in
rosso</div>
 <div>Anche <p><strong>questo testo</strong></p> NON sarà mostrato in rosso in quanto
tra div e strong  c'è un altro elemento, mentre <strong>questo testo</strong> vicecersa lo
sarà.</div>
</body>
```

Pseudo-classe :first-child

Molto simile al selettore figlio, solo che tale regola vale SOLO per il primo figlio, indipendentemente dalla distanza cui è posto dall'altro.
Es:

```
div > strong:first-child { color: red; }
```

Pseudo-classi dei links

E' probabilmente una delle feature più interessanti dei Css: permette di associare ai links (ed anche agli altri elementi, tranne :link e :visited, in teoria; in pratica visto il non supporto di IE solo all'elemento A) regole diverse a seconda se il link **non è stato visitato (:link), già seguito (:visited), attivo (:active), al passaggio del mouse (:hover) e se selezionato (:focus).**
Es:

```
a:link { color: blue; }
a:visited { color: purple; }
a:active { color: black; }
a:hover { color: red; }
a:focus { color: green; }
```

Selettori Adiacenti

Ha la stessa limitazione dei selettori figli: non è supportato da Internet Explorer. Secondo le specifiche permette di associare regole ad elemento solo se contenuto subito dopo l'altro indicato.

```
div + strong { color: red; }
```

Es:

```
<div>
  <strong>Questo testo sarà mostrato in rosso</strong> mentre
 </div>

<div>
  il prossimo <strong>testo</strong></p> no, in quanto non immediatamente vicino
all'elemento div
</div>
```

Selettori di Attributi

Selettore davvero interessante per i possibili sviluppi verso l'XML ma purtroppo di poca utilità visto che pochissimi browser attualmente non lo supportano. Permette di associare regole diverse all'elemento se dotato di attributi o valori degli stessi.

```
H1[title] { color: red; }
H1[title="pippo"] { color: green; }
```

La prima regola associa il colore rosso a tutti gli elementi h1 dotati di *title,* mentre la seconda regola varrà solo per gli elementi h1 il cui titolo è esattamente pippo. Potrebbe essere molto utile per assegnare valori diversi agli elementi input di un form, per variare lo stile del pulsante submit rispetto ad un campo testo ad esempio; purtroppo ad oggi è conveniente per la stessa operazione assegnare delle classi diverse ai tag.

Selettori di Classi

Permette di assegnare regole a qualunque elemento a cui abbiamo assegnato una classe:

```
*.nomeclasse { color: red;} o, in forma equivalente
.nomeclasse { color: red;}
```

E' possibile ovviamente assegnare regole solo ad uno o più elementi con classe *nomeclasse*, usando la sintassi:

```
strong.nomeclasse  {color: purple; }
```

Selettori di ID

Selettore molto simile al precedente con l'ovvia differenza che le regole saranno assegnate solo ad un particolare ID:

```
#nomeclasse  {color: purple; }
```

Pseudo-classi :first-line e :first:letter

Opzione molto interessante che limita le regole rispettivamente alla prima linea del testo nel primo caso ed alla prima lettera nel secondo per effetti simili a quelli tipografici. Es:

```
p:first-line { text-transform: uppercase; }
p:first-letter { font-size: 300%; color:red; }
```

trasforma la prima riga dei paragrafi (p) in maiuscolo e rende la prima lettera degli stessi 3 volte più grande e di colore rosso.

Le proprietà dei caratteri (font)

In questa lezione vedremo come cambiare la forma estetica dei caratteri tipografici utilizzati nelle vostre pagine web.

Fondamentalmente utilizzeremo e modificheremo queste proprietà:

- ***font-family***
- ***font-style***
- ***font-variant***
- ***font-weight***
- ***font-size***
- ***font***

font:

```
font: <size>  [ / <line-height> ] | <family> | [ <style> | <variant> | <weight> ]
```

es: font: 12px/18px bolder italic arial;

Con font è possibile indicare in forma compatta tutte le proprietà dei caratteri con le stesse regole che trovate successivamente.

font-family:

```
font-family:  <nome carattere> | <font generico>;
```

Es: font-family: Verdana, Arial, sans-serif;

Utilizzando font-family riusciremo a dare un aspetto ai caratteri utilizzati nella pagina. I browser mostreranno il primo carattere, da sinistra, tra quelli utilizzati dal sistema operativo in uso. Nell'esempio precedente stamperà i caratteri in verdana, se non installato, arial altrimenti utilizzerà il font di default per il gruppo sans-serif. Ricordate quindi di provare tutti i caratteri aggiungendone uno la volta a sinistra per controllare l'aspetto estetico avendo cura di indicare una famiglia generica alla fine. I caratteri possono esser raggruppati in due grandi categorie: grazie o serif, che hanno la caratteristica di avere dei *riccioli* di abbellimento e bastoni o sans-serif. In realtà è possibile indicare anche altre 3 famiglie di font: monospace (a spaziatura fissa), cursive e fantasy, famiglie che mi sento di sconsigliarvi ampiamente, tranne alcune eccezioni

particolari.

Se il font ha un nome composto da due o più termini, ad es. Times new roman, occorre metterlo tra ".

Es: font-family: "Times new Roman", serif;

font-size:

font-size: <dimensione> | xx-small | x-small | small | medium | large | x-large | xx-large | larger | smaller

dove dimensione può essere una lunghezza, oppure una percentuale.

Larger e smaller son parametri relativi e servono ovviamente ad aumentare o diminuire la dimensione dei font rispetto all'elemento contetitore.

Approfondimenti: Dimensionamento e misure

font-style:

font-style: normal | italic | oblique;

Es: font-style: italic;

Serve a dare lo stile ai caratteri:. Sostituisce il tag html <i>. Oblique e cursive attualmente nella maggior parte dei browser son equivalenti.

font-variant:

font-variant: normal | small-caps;

Tale soluzione serve a dare l'effetto *"maiuscoletto"* al testo: le lettere maiuscole restano tali, quelle minuscole son riprodotte in maiuscolo ma in un corpo più piccolo.

font-weight:

font-weight: normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900;

Es: font-weight: 700;

Con tale proprietà facciamo stampare i caratteri con peso diverso. In effetti i browser attuali distinguono tra il peso standard e il bold. Pertanto la scelta si riduce a scegliere tra normal e bold, tenendo in mente che bolder e lighter, attributi relativi possono aumentare o diminuire il peso del carattere rispetto all'elemento contenitore. Sostituisce il tag html <b>. Att: font-weight:bold ed il tag non sono equivalenti anche se possa sembrare così; font-weight agisce sull'aspetto del font, sul contenuto semantico del testo racchiuso.

Script visuale per la gestione dei caratteri.

Proprietà dei colori

I colori possono esser definiti nei css attraverso una parola chiave, attraverso la codifica RGB in due metodi: con un codice esadecimale oppure indicando la percentuale di Rosso, verde (Green) e Blue che compongono il colore.

La sintassi da utilizzare è:

- color: **key**
- color: **#rrggbb**
- color: **#rgb** *(forma ridotta)*
- color: **rgb (r, g, b,)**
- color: **rgb (r%,g%,b%)**

Esempi:

```
{color: red;}

{color:#FF0000;}

{color:#F00;}

{color: rgb(255,0,0); }

{color: rgb(100%,0,0);}
```

Tutti i codici riportati indicano il colore rosso.

Praticamente i colori vengono indicati in una scala da 0 a 255, in cui 0 indica nessuna presenza e 255 presenza completa del primario. Tali numeri possono esser indicati in percentuale, in forma decimale o esadecimale (FF=255). Attenzione ad aggiungere il # quando si usa la sintassi esadecimale: anche se IE mostra il colore se si scrive {color:F00;} ciò non vuol dire che sia corretto.

La forma ridotta infine è un modo di scrivere un codice hex alleggerendo il peso quando i codici si ripetono: #FC9 è la forma contratta di #FFCC99, ad esempio.

Tabella nome colori -> codici esadecimali, codici rgb.

Il modello RGB è un modello additivo in cui ogni colore è definito dalla somma dei 3 colori primari. La completa somma di tutti e 3 da il bianco, la somma di due primari da i 3 primari sottrattivi: Giallo, Ciano e Magenta.

Approfontimenti: Wikipedia: RGB

Colore e background

Il colore è applicato alle due proprietà: color e background, che definiscono il colore del testo e lo sfondo.

color:

```
color: <valore>;
```

Dove il valore può esser assegnato utilizzando uno dei metodi sopra illustrati.

background:

```
background: <color> | <url> | <repeat> | <position> | <scroll>
```

Es: background: #F00 url('immaginedisfondo') fixed no-repeat center left fixed;

<color> può assumere un valore come su indicato, può essere transparent per indicare uno sfondo trasparente o none per non dare nessuno sfondo.

<url> è il percorso dell'immagine di sfondo, da usare nella forma url (<valore>), dove valore può esser inserito tra ‘, ", o senza, e può esser un percorso assoluto o **relativo alla posizione del css.**

<repeat> può esser uno dei seguenti valori: **repeat**, opzione di default che fa ripetere l'immagine in orizzontale e verticale, **no-repeat**: nessuna ripetizione; **repeat-x, repeat-y:** permettono la ripetizione lungo l'asse orizzontale (x) o verticale (y).

<position> indica la posizione dell'immagine di sfondo. può assumere valori numerici, percentuali o uno dei seguenti: **top, center, bottom** per il posizionamento verticale, **left, center, right** per quello orizzontale.

<scroll> può assumere i valori **scroll** (opz. di default) e **fixed** per indicare se l'immagine debba seguire o no lo scrolling della pagina.

Le stesse proprietà possono esser indicate singolarmente con:

- background-color: <color>;
- background-image: <url>;
- background-position: <position>;
- background-repeat: <repeat>;
- background-attachment: <scroll>;

Proprietà dei testi

I vantaggi dell'uso dei CSS viene resa evidente dalle proprietà dei testi, proprietà che permettono il controllo tipografico accurato, semplice per tutti ma allo stesso tempo potente. Con i css infatti è possibile controllare qualunque aspetto tipografico permettendo una resa avanzata dei testi usando le seguenti proprietà:

text-align:

```
text-align: left | center | right | justify
```

Esempi:

```
{text-align:justify;}
```

Permette di allineare il testo rispettivamente a sinistra, centrato, a destra e giustificato.

text-decoration:

```
text-decoration: none | underline | overline | line-through | blink
```

Esempi:

```
a {text-decoration:underline;}
a:visited {text-decoration:line-through;}
a:hover {text-decoration:none;}
```

per stampare il testo: senza nessuna decorazione, sottolineato, sopralineato, barrato e lampeggiante. L'uso di blink è da evitare poiché potrebbe creare problemi fisici ad alcuni e perché, obiettivamente, è fastidioso non poco.

text-transform:

```
text-transform: none | uppercase | lowercase | capitalize
```

permette di trasformare il testo in: tutto maiuscolo (uppercase), tutto minuscolo

(lowercase), ogni prima lettera in maiuscolo (capitalize) o lasciare il testo formattato come da xHTML.

line-height:

line-height: normal | <valore> | <valore percentuale>

Esempi:

{line-height: 20px;}
{line-height: 200%;}

per impostare l'interlinea del testo.

word-spacing:

word-spacing: normal | <lunghezza>

permette di aumentare o diminuire lo spazio tra le parole. Di default ha valore 0. Impostando un valore positivo tale lunghezza si aggiunge alla distanza solita, impostandone uno negativo si dimuisce. **Attenzione**:

{word-spacing:10px;}

imposta una distanza tra le parole di 10px PIU' il valore standard, non 10px in assoluto.

letter-spacing:

letter-spacing: normal | <lunghezza>

Come sopra. Di default ha valore 0 che sta ad indicare lo spazio tra le lettere standard. Anche in questo caso la lunghezza indicata va a sottrarsi o ad aggiungersi al valore di default.

text-indent:

text-indent: <lunghezza> | <percentuale>

Permette di stabilire il rientro del capoverso di un paragrafo o di un qualunque blocco di testo.

vertical-align:

vertical-align: baseline | bottom | middle | sub | super | text-bottom | text-top | top | <valore> | <percentuale>

Inziamo subito a dire cosa NON è: non corrisponde al valign delle tabelle **SE non usato nelle tabelle**. Non allinea al centro di un blocco un'immagine, tanto per esser chiari, ma indica l'allineamento verticale di un elemento online rispetto ai contigui. Per esser chiari, fa ciò che si fa con gli apici o pedici.

I possibili valori accettati sono:

- **baseline** (allinea l'elemento alla linea base dell'elemento contiguo)
- **bottom** (l'elemento viene allineato con il più basso degli elementi della linea)
- **middle** (l'elemento viene piazzato al centro dell'elemento contenitore)
- **sub** (pedice)
- **super** (apice)

- **text-bottom** (l'elemento viene allineato con la parte bassa della linea di testo)
- **text-top** (l'elemento viene allineato con la parte alta della linea di testo)
- **top** (l'elemento viene allineato con il più alto degli elementi della linea)
- ***valore*** (il tal caso viene allineato secondo il valore indicato. accetta anche valori negativi)
- ***percentuale*** (allinea l'elemento di un x% rispetto al valore del line-height dichiarato.)

Esempi:

```
{vertical-align:middle;}
```

Approfondimento: Allineamento verticale con i CSS.

white-space:

```
{white-space: normal | pre | nowrap}
```

Se nel testo xhtml si inseriscono più spazi successivi, o anche ritorni a capo, questi saranno ignorati e verrà mostrato a schermo un solo spazio. Con tale proprietà si modifica tale comporamento. Con **normal** due o più spazi o ritorni capo vengono mostrati in un unico carattere di spazio, **no-wrap** fa sì che il ritorno a capo del testo si ripercuota nella pagina mostrata, mentre più spazi successivi vengono stampati come un unico spazio, con **pre** si imposta il ritorno a capo e gli spazi così come scritti nel codice.

Proprietà dei bordi

I bordi possono esser definiti da tre proprietà:

- border-width: <dimensione>;
- border-color: <colore>;
- border-style: <stile>;

che vengono applicati a tutti i bordi del box.

<Dimensione> può essere una lunghezza come definitita in questo articolo oppure i valori

- **thin** (sottile)
- **medium** (medio)
- **thick** (spesso)

<Stile> può essere:

- **none** (nessuno)
- **solid** (continuo)
- **double** (doppio)
- **dashed** (tratteggiato)
- **dotted** (punteggiato)
- **inset** (incassato)
- **outset** (in rilievo)
- **groove** (scanalato [in basso])
- **ridge** (scanalato [in alto])

<Colore> infine può definito come indicato in questo articolo.

Esiste anche una forma compatta:

```
border: <dimensione> <colore> <style>.
```

Utilizzando tali proprietà assegnamo le caratteristiche a tutti e 4 i lati:

Es: border: 4px red solid;

crea un bordo uniforme di 4 pixel di colore rosso. Se volessimo assegnare lo stesso bordo **solo** a quello inferiore potremmo usare la notazione:

border-bottom: 4px red solid;

E' possibile inolte assegnare proprietà diverse nel seguente modo:

border-width: 1px 2px 3px 4px;

che son riferiti, rispettivamente al bordo superiore, destro, inferiore e sinistro.

Es. pratici:

border:1px solid rgb(255, 204, 0);

border-width:3px; border-style:dotted; border-color:red blue green purple;

border-bottom:1em double rgb(153, 153, 153);

Script che consente di creare in forma visuale i bordi voluti.

Liste

Le liste in html sono distinte in liste ordinate ( ) e liste non ordinate ( ).

Esempi possono essere:

1. item 1
2. item 2
3. item 3

- item 1
- item 2
- item 3

che corrispondono rispettivamente al codice xhtm:

```
<ol>
  <li>item 1</li>
  <li>item 2</li>
  <li>item 3</li>
</ol>
```

e

```
<ul>
  <li>item 1</li>
  <li>item 2</li>
  <li>item 3</li>
</ul>
```

Con i css è possibile definire fondamentalmente il tipo di rappresentazione della lista con la proprietà **list-style** e le categorie specifiche

- **list-style-position**: [inside | outside]
- **list-style-type**: [none | disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-alpha | upper-alpha | lower-greek | lower-latin | upper-latin | hebrew | armenian | georgian | cjk-ideographic | hiragana | katakana | hiragana-iroha | katakana-iroha]
- **list-style-image**: <url>

List-style-position ci da la possibilità di inserire il marcatore all'interno o all'esterno del blocco <li>.

List-style-type è probabilmente la proprietà più sfruttata nei css, nonché la più interessante che ci permette di stabilire se:

- non utilizzare nessuno stile (none);
- utilizzare un disco (disc), opzione di default;
- utilizzare un cerchio (circle);
- utilizzare un quadrato (square)

per le liste non ordinate e

- non utilizzare nessuno stile (none);
- utilizzare i numeri decimali (decimal), opzione di default;
- utilizzare i numeri decimali partendo da zero (decimal-leading-zero);
- utilizzare i numeri romani in minuscolo (lower-roman)
- o in maiuscolo (upper-roman);
- utilizzare le lettere dell'alfabeto in minuscolo (lower-alpha);
- o ovviamente in maiuscolo (upper-alpha);
- utilizzare le lettere latine, greche, ebree, armene, georgiane, cirilliche, gli ideogrammi giapponesi e così via.

List-style-image infine ci da l'occasione di utilizzare immagini al posto dei classici marcatori, opzione molto carina ma da sfruttare con attenzione per evitare pastrocchi grafici inguardabili.

Uno dei problemi più diffusi che si incontra quando si inizia a lavorare con i css sulle liste è la loro apparente difficoltà di mostrarle simili su vari browser a causa del diverso valore di default degli stessi riguardo i margini ed il padding delle liste e dei vari elementi della lista.
In seguito pubblicheremo un esempio pratico, per ora il consiglio è di impostare il padding ed il margin sia ad <ul> che <li> (ed ovviamente anche a <ol> nel caso si usi una lista ordinata) e ridefinirli di volta in volta in base alle necessità.

Esiste infine un tipo di lista usata pochissimo che è la cosiddetta "**lista di definizione**", definita dal tag <dl> con due tag caratteristici:

<dt> titolo della definizione
<dd> definizione vera e propria

Es.

```
<dl>
  <dt>Lista</dt>
  <dd>elenco di cose o di persone, cifre, dati e simboli</dd>
  <dt>Definizione</dt>
  <dd>specificazione esatta, determinazione</dd>
</dl>
```

a cui corrisponderà

Lista

elenco di cose o di persone, cifre, dati e simboli

Definizione

specificazione esatta, determinazione

Come si nota può esser utilmente impiegata per la creazione di glossari, per separare i partecipanti ad una conversazione e così via.

Senza css questi tag si comporteranno come elementi di blocco, come d'altronde succede per le liste ordinate e non-ordinate ed è pertanto possibile utilizzare le stesse proprietà che utilizzeremo per qualsiasi elemento blocco, cosa che le rende particolarmente interessanti.

Tabelle

I css permettono design tableless, cioè senza tabelle, che ritornano finalmente ad assumere il valore per le quali son state progettate: mostrare dati tabellari, statistici. Anche per le tabelle esistono specifiche proprietà CSS:

table-layout:

table-layout: auto | fixed

che permette alle tabelle, alle righe o alle colonne delle stesse, di adattarsi al contenuto o di restare fisse in base, in altezza o larghezza a ciò che si è indicato.

empty-cells:

empty-cells: show | hide

per scegliere se mostrare o meno le celle vuote.

border-collapse:

border-collapse: collapse | separate

indica se far *collassare* i bordi in un unico bordo o separarli in più livelli.

border-spacing:

border-spacing: <lunghezza> <lunghezza>

da usare ovviamente con *border-collapse:separate* per indicare lo spazio tra i bordi. Si possono specificare due valori: 1 per la spaziatura destra e sinistra ed il secondo per quella in alto e basso; se si indica un solo valore vale per tutti e 4 i bordi.

caption-side:

caption-size: top | right | bottom | left

per indicare in che lato mostrare il valore di caption della tabella.

Box-model

Avvicinandoci ai css questo sarà uno dei concetti più di tutti ci assilleranno. Cosa si intende per box-model?

Immaginiamo la pagina come una pagina tipografica per iniziare (esempio: un quotidiano). In esso ogni elemento presente può esser raffigurato come una scatola, ognuna delle quali può contenerne altre. Con i css dobbiamo ragionare nello stesso modo. Ogni pagina html è formata da un box principale, dentro il quale son presenti il box <body> riempito a sua volta di altre **elementi.**

Questi possono esser di due tipi: **block-level** (blocchi) o **inline**.

Qual è la differenza? Semplice:

- *un elemento block level si posiziona sotto il blocco precedente, quello inline a fianco;*
- *un blocco occupa in larghezza tutto lo spazio dell'elemento che lo contiene, l'inline solo quello effettivo del suo contenuto;*
- *un blocco può contenere altri blocchi ed elementi inline, l'inline solo altri inline;*
- *un blocco possiede delle dimensioni configurabili tramite proprietà quali height e width, un inline no.*

Elementi tipicamente block-level sono: ul, ol, form, p (anche se questo fa eccezione in quanto no può contenere altri blocchi al suo interno) e div, il blocco generico.
Elementi inline sono: a, em, strong, b e span, elemento inline generico.
Elenco approfondito degli elementi [in inglese].
C'è da notare che è possibile modificare tale proprietà con il comando display; **display: block**, rende esplicito un elemento come block-level, **display:inline** fa il contrario.

Una particolare proprietà dei CSS è **float**: cioè, letteralmente *galleggiamento*. Tale proprietà permette di disporre del testo attorno all'elemento.
L'esempio successivo spiegherà meglio di mille parole il concetto

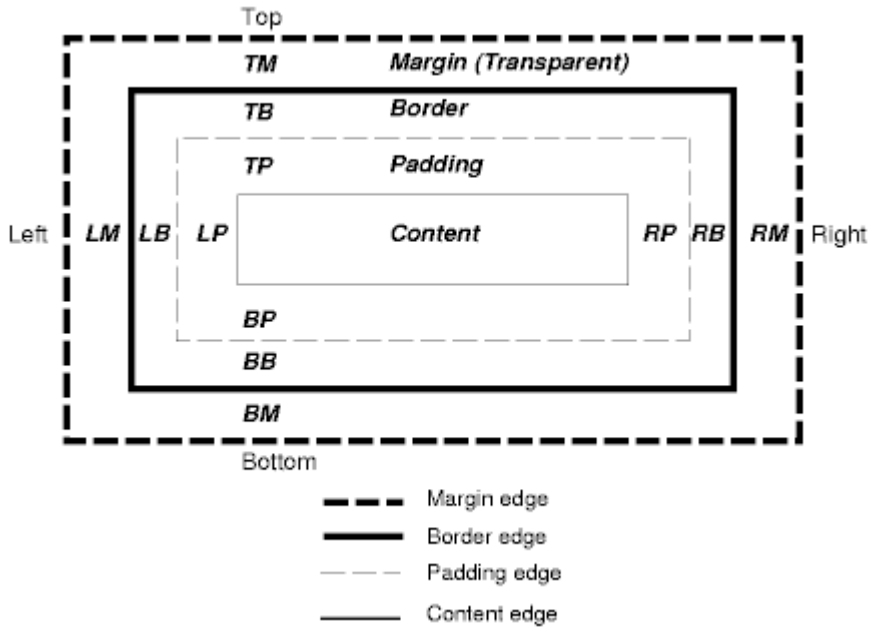
```
1  [....]
2  <style type="text/css">
3      #content {width:300px;}
4      #id1 {float:left;}
5  </style>
6  [....]
7  <div id="content">
8      <p><img src="./immaginerossa.jpg" alt="" />Lorem ipsum dolo
9      sit amet, consectetuer adipiscing elit, sed diam nonummy
10     nibh euismod tincidunt ut laoreet dolore magna
11     aliquam erat volutpat.</p>
12     <p><img src="./immagineblu.jpg" id="id1" alt="" />Lorem
13     ipsum dolor sit amet, consectetuer adipiscing elit,
14     sed diam nonummy nibh euismod tincidunt
15     ut laoreet dolore magna aliquam erat volutpat.</p>
16 </div>
```


Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.


Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Notate che mentre l'immagine rossa sarà seguita da una linea di testo, con la proprietà "*float:left;*" *ordiniamo* al testo di circondare l'immagine.
La proprietà **clear** invece permette di stabilire se un blocco debba esser circondato da testo. Praticamente permette di ripristinare l'allineamento a seguito dell'uso di float.
Float: può assumere valori (right | left | none) mentre clear può avere valori (right | left | both | none).

Un aspetto fondamentale da capire il prima possibile è la conformazione di un blocco. Approfitteremo dell'immagine presente sul sito del W3C:



Notiamo che ogni blocco è formato da 4 sezioni concatenate:

- Contenuto
- Padding
- Bordo
- Margin

Il padding è ciò che c'è tra il bordo del blocco ed il contenuto.
Il margine è ciò che c'è tra i vari blocchi.

Impostando la larghezza (widht) attraverso i css si imposta la dimensione AL SOLO contenuto.

Quindi un blocco con width: 400px, padding: 50px; border: 10px; margin: 30px occuperà uno spazio totale di 490px, non 400 come si potrebbe esser portati a pensare le prime volte. Ovviamente lo stesso discorso vale per l'altezza.

Tutto facile? No. Internet explorer fino alla versione 5 (e se non impostato il doctype anche nella versione 6) considerava ERRONEAMENTE width come la larghezza di contenuto+padding+bordo. Questo ha portato molti errori di interpretazione ed ha costretto molti a trovare una soluzione inventando hack per risolvere il baco.
Uno dei più famosi fu la famosa **Regola di Tantek**.

Come funziona? Si sfrutta uno dei tanti altri bachi di explorer per fargli fare ciò che vogliamo.
Vediamola in pratica con l'esempio precedente:

```
div.blocco {
width:460px;  #dimensione per IE5
padding:50px;
border:10px;
margin:30px
voice-family: "\"}\"";
voice-family: inherit;
width:400px;
}
body>div.blocco {
width: 400px;
}
```

IE 5 interpreterà la dimensione di 460px ed ignorerà tutto ciò che c'è dopo *voice-family:*

"\"}\""; a differenza degli altri browser che leggeranno anche il resto e trovando width:400px riassegneranno la larghezza corretta. Per completezza di informazione, ci son anche browser, quali alcune versioni di Opera, che nonostante interpritino correttamente il box-model non riescono a leggere oltre il comando voice-family: "\"}\""; : a ciò serve la regola successiva body>div.blocco

Ci son altre tecniche da utilizzare: quella che io preferisco è la cosiddetta Simplified Box Model Hack (SBMH):

```
div.blocco {
  width: 400px; \width: 460px; w\idth: 400px;
}
```

Qual è il principio? La prima dichiarazione, senza escape (width:400px;) verrà letta dai browser quali Opera 5, ad esempio, che non leggono ciò che segue lo slash(\); \width: 460px; sarà letto da tutti, tranne Opera5 appunto e pochi altri e ridifinerà la larghezza del box, w\idth:400px sarà seguita da tutti gli altri browser tranne, appunto, IE 5.
Semplificando: il valore corretto sarà inserito nella prima e terza proprietà; quello per IE 5 nella seconda.

Tale tecnica vale per qualsiasi proprietà, ma, e c'è un ma, secondo le regole w3c uno slash non può precedere le lettere: a, b, c, d, e, f. Quindi potrà esser sfruttata per l'altra proprietà che ci interessa per la definizione del layout, height e non, ad esempio, per font-family, cosa che ci dispiacerà poco in effetti. :)

Dimensionamento e misure

Le misure e le dimensioni nei css, utilizzate in molti ambiti tra i quali e non solo: bordi, altezza e larghezza dei blocchi, grandezza dei caratteri, margini, padding, possono essere espresse in forma relativa oppure assoluta nella forma: numero seguito da due lettere che indicano l'unità di misura. Attenzione agli errori frequenti che possono creare problemi di visualizzazione, in particolare: tra i numeri e la misura non possono esserci spazi. Se la misura è zero non deve esser seguito da unità di misure.

```
Es.: corrette: 12px; 1.45em; 0;
errate: 12 px, 1. 45em, 0px;
```

Dimensioni assolute:

- **cm:** centimetri;
- **mm:** millimetri;
- **in:** inches, pollici, dove un pollice = 25,4 mm
- **pt:** punti, cioè la 72a parte di un inch, circa 0.35mm; in pratica 3pt ~ 1mm
- **pc:** picas, un sesto di inch, circa 4,2 mm

Tali dimensioni son ottime se usate nella stampa e nei dispositivi a dimensione fissa, son sconsigliabili viceversa a video dato che essi son per loro natura diversi uno dall'altro. Dimensioni assolute comportano a schermo visioni diverse delle proprie pagine da computer a computer. Tenete sempre in mente che il web non è la tipografia classica e che è praticamente impossibile creare siti che siano perfettamente simili su qualsiasi computer.

Dimensioni relative:

- **em:** dimensione di una lettera m del carattere utilizzato
- **px:** pixel

- **%:** percentuale

La dimensione dei pixel dipende dalla risoluzione dei dispositivi utilizzati. Un pixel è la più piccola particella *unica* dello stesso. A video son i vari elementini luminosi che messi assieme formano un'immagine. Un em è la dimensione di una lettera m del set di caratteri utilizzato. Se impostiamo una misura in em essa è riferita alla dimensione dei font del livello contenitore. Praticamente:

```
body {font-size:20px;}
 div#div1 {font-size:1.5em;}
 div#div2 {font-size:0.8em;}
<body>
 <div id="div1">Bla bla<div id="div2">Bla bla</div></div>
</body>
```

Il body avrà dimensioni **20px**. Div1 avrà dimensione di **una volta e mez**za quella di body, cioè **30px**; Div2 **0.8volte** quella del suo contenitore quindi 30*0.8=**24px**. Le dimensioni percentuali seguono lo stesso ragionamento su indicato. Le misure relative son quelle più utilizzate nelle pagine. Il mio consiglio è quello di impostare una dimensione per il body in pixel ed usare em per tutte le altre sezioni. Le misure in pixel dovrebbero esser usate solo per le dimensioni dei bordi dei blocchi.

Css per la stampa

La stampa di pagine web presenta spesso delle difficoltà. Sovente il web master si preoccupa dell'impaginazione a schermo, magari utilizzando layout liquidi dimenticando che la stampa ha caratteristiche diverse. Come ovviare al problema? Tempo fa era d'uso creare pagine apposite per la stampa, generandole lato server e togliendo gli elementi di navigazione (menu, header e footer, tipicamente).

Lo stesso, e molto di più può esser fatto via CSS: vediamo passo passo come fare:

Nascondere blocchi ed elementi alla stampa.

Studiamo la pagina e segnamoci quali sono i blocchi e gli elementi grafici da nascondere alla stampante. Creiamo un nuovo file css ed impostiamo ad essi la proprietà **display:none;**

Impostare le dimensioni in misure assolute

Diamo larghezza automatica al blocco da stampare, impostiamo la grandezza dei caratteri possibilmente in punti, 12 potrebbe essere una buona soluzione, ricordiamoci che molti stampano in bianco e nero, quindi se necessario cambiamo i colori ai blocchi.

Prestare attenzione agli elementi flottanti ed agli sfondi

Le stampe son spesso impostate automaticamente, e giustamente, sul non stampare le immagini di sfondo. Se usiamo elementi che vorremmo mostrare con immagini in sottofondo o se usiamo tecniche di image replacement, teniamolo presente e riscriviamo le regole nel css di stampa. Attenzione inoltre agli elementi flottanti o con posizione assoluta che potrebbero dar problemi di posizionamento. Sovrascriviamo tali regole con altre che permettano la loro collocazione nel normale flusso di informazioni

Aggiungere l'url dei link (opz.)

I collegamenti a stampa ovviamente non possono esser seguiti però potrebbe esser necessario indicare dove puntano tali collegamenti. I css permettono di mostrare il

contenuto dell'attributo href usando tale codice:

```
a:link:after {
  content:  " [" attr(href) "]";
}
```

che ha però due controindicazioni: alcuni browser la ignorano, e se vi viene in mente IE6 non sbagliate di certo, e soprattutto se l'href è un collegamento relativo sarà stampato in questa forma e non in forma assoluta. Detto più semplicemente se linkate pagine con la forma "/index.html" troverete stampato tale valore. Come ovviare? Beh, non usare collegamenti relativi sarebbe la soluzione ideale oppure usando un trick

```
a[href=^="/"]:after {
   content: " [http://www.sito.com" attr(href) "] ";
}
```

che sta per: se l'href inizia per / stampa il http://www.sito.com prima del valore dell'attributo.

Collegare il css per la stampa alle pagine

Una volta scritte tutte le regole è necessario associare il file css creato per la stampa alle pagine xHTML:

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

attenzione alle altre regole scritte per gli altri media: potreste avere delle sorprese: se associamo un css senza specificare il media o impostandolo a tutti queste regole saranno applicate **anche** alla stampa. Per non incorrere in rappresentazioni non volute una buona soluzione è associare le regole per il video utilizzando l'@import. Cioè:

```
<style type="text/css" media="all">@import "generale.css";</style>
<link rel="stylesheet" type="text/css" media="print" href="print.css" />
```

Bene. Ora dovreste avere delle pagine perfette per la stampa. Ma i css ci permettono di più: ci danno la possibilità di controllare il flusso di stampa e, volendo, forzare la stampa di un blocco all'inizio della pagina successiva. A tale scopo ci vengono incontro le regole:

```
page-break-after: auto | always | avoid | left | right | inherit;

page-break-before: auto | always | avoid | left | right | inherit;

page-break-inside: always | auto | inherit;
```

che forzano o impediscono l'interruzione della stampa nella pagin del blocco considerato, rispettivament dopo (after) prima (before) o dentro (inside).

I valori indicano:

- auto: non forza né impedisce l'interruzione
- always: forza sempre l'interruzione di pagina
- avoid: evita sempre l'interruzione
- left: forza una (o due) interruzione di pagina in modo che la pagina successiva sia stampata come una pagina sinistra
- right: forza una (o due) interruzione di pagina in modo che la pagina successiva sia stampata come una pagina destra
- inherit: eredita la stessa proprietà dal blocco genitore.

Consigli di scrittura dei file .css

Ok. Fin qui nulla di che. Come avrete notato, i css non hanno nulla di trascendentale ed il loro uso è sufficentemente semplice. A questo punto potrebbero sorgerci domande quali:

- Che programma usare per scrivere il codice dei CSS?
- Come scriverli e possibilimente ottimizzarli?
- Come verificare la corretta visualizzazione su browser diversi?

Alla prima domanda non posso che dare una risposta molto soggettiva. Io personalmente utilizzo notepad++ scrivendo tutto il codice a mano, cosa che tra l'altro faccio anche con l'html ed il php, non perché sia un talebano del codice ma per una questione di **velocità di scrittura e di controllo accurato dello stesso**. Ovviamente sfrutto alcune caratteristiche del programma imprenscindibili: colorazione del codice, funzioni di undo e redo a più livelli, sostituzione su tutto il documento anche attraverso l'uso di espressioni regolari per dirne alcune.

Cosa consiglio? Qualunque text-editor: pico, nano, vim, dreamweaver, notepad++, quanta, phpedit e così via. Provateli, cercate quello che vi sembra più comodo, ergonomico, usabile e poi continuate ad usare solo quello. Per la visualizzazione basta un semplice F5 sul browser. Non è indispensabile che il vostro editore abbia preview integrate.
Esistono infine programmi che vi permettono di creare i css in forma visuale senza conoscere tutte le proprietà: tra i tanti a me piace SimpleCSS. Può esser comodo per chi è alle prime armi per imparare a memoria i valori e le proprietà, a condizione che una volta superata la prima fase, si passi ad un vero editore di testi.

Il tutto ovviamente IMHO, ovvero, per me, secondo me.

Anche il secondo punto sfiora le preferenze personali. Posso in generale dare alcuni consigli su come opero io. Non è detto che sia il metodo migliore, anzi.
Secondo alcune scuole di pensiero i css dovrebbero occupare il minor spazio possibile, cercando quindi di accumulare le regole per quanto possibile. Io, soprattutto per siti con CSS abbastanza impegnativi tento di solito a dividerli in più parti, scrivendone uno *generale*, uno *solo per la stampa*, se ci sono sezioni sufficentemente distanti come stile grafico, *uno per sezione* ed a volte *uno per alcuni browser* (NN4 e IE5 nello specifico) quando richiesto dal committente o dal target del sito.
Cerco poi di NON scrivere le regole selettore per selettore, una cosa tipo `h1 { margin:1em 1em; font-size:1.7em; border-bottom:1px solid gray; }` ma di suddiverli nel css, o se molto corposi in più css, a seconda di ciò che dichiarano: **colori, bordi, box-model, testo.** Può sembrare un lavoro inutile ma vi assicuro che **è molto comodo quando poi torniamo a manutere il sito avere tutte le proprietà sotto mano e sott'occhio** e non andare di Ctrl-F a cercarlo nel testo di un documento di 500 righe.

Infine, e ve ne accorgerete presto..., avvertirete la necessità di controllare la resa di quanto scritto su molti browser diversi, quali, ma non solo: IE5.01 e successivi, Opera 7 e successivi (se non dalla 5), Firefox, Netscape, Safari, Konqueror, Galeon e chi ne ha ne metta. Molto dipende dal Sistema Operativo che utlizzate: l'ideale sarebbe di avere varie postazioni con sistemi operativi diversi e più computer. Ok ok, spesso è fantascienza. Come fare allora?
Su piattaforma Windows non dovreste aver problemi con Firefox, qualsiasi versione, Opera e Safari che possono coesistere anche in più versioni e più piattaforme.
Qualche problema in più ce lo da Internet Explorer che sovrascrive la precedente installazione. Fortunatamente esistono hack per avere più versioni di IE sullo stesso PC. Approfondimenti.

Per Mac, Intel in particolare, esiste Parallels che permette di avere due Windows e Mac

sulla stessa sessione di lavoro. Con lo stesso principio lavora WmWare: permette di avere più macchine virtuali sullo stesso dispositivo, con la possibilità quindi, di avere windows e linux sulla stessa macchina. Per i Linari puri e duri esiste infine la possibilità di utilizzare IEs4Linux.

Infine esistono servizi web che ci danno l'anteprima su vari browser, semplicemente, indicandogli la pagina da controllare ed i browser che ci interessano; addirittura alcuni ci permettono di scegliere se mostrare il rendering con flash, javascript e java disabilitati o meno. Ottimo in tal senso è Browsershots.

Segnalo infine evolt un sito che raccoglie moltissime versioni di browser, utile ad esempio, a chi non ha sottomano NN4 e non sa dove andarlo a cercare.

Come Creare un sito a 2 colonne tableless

In questo articolo vedremo un esempio pratico e di difficoltà medio bassa di creazione di un layout tableless a due colonne, con header in alto e footer in basso. Chiaramente potrete riutilizzare il codice mostrato per qualsiasi vostro progetto.

Iniziamo a definire la struttura della pagina xhtml, inserendo il giusto doctype per xhtml 1.1, definendo la lingua italiana ed il charset come UTF-8 ed aggiungendo il link al css, inserito nella stessa directory del file e 4 blocchi per header, footer e le due colonne, messi in un blocco contenitore:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="it">
  <head>
   <title>Prova layout 2 colonne</title>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
   <link type="text/css" rel="stylesheet" href="style.css" title="Style" media="all" />
 </head>
 <body>
   <div id="contenitore">
     <div id="header"> </div>
     <div id="colsx"> </div>
     <div id="coldx"> </div>
     <div id="footer"> </div>
   </div>
 </body>
</html>
```

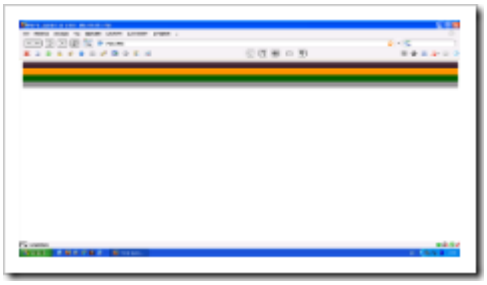
Modifichiamo il css dando i colori di sfondo ai vari blocchi:

- bianco al body (non è ridondante, visto che molti, me compreso non assegnano colori al css predefinito dal browser)
- #4A2C3E all'header
- #F90 alla colonna sinistra
- #060 a quella di destra
- #999 al footer
- al contenitore non diamo alcun valore lasciando quindi visibile il colore dello sfondo del body.

Il css a questo punto sarà questo:

```
body {background:white;}
 div#contenitore {}
  div#header {background:#4A2C3E;}
  div#colsx {background:#F90;}
  div#coldx {background:#060;}
  div#footer {background:#999;}
```

Ed il risultato sarà questo (*clicca sulle immagini per vederle a grandezza maggiore*):



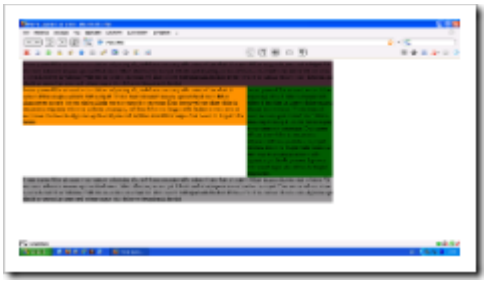
Definiamo ora le larghezze dei vari blocchi. Ipotizzo che tutto il sito avrà larghezza totale di 900px (ovviamente potremmo anche definire la dimensione in percentuale o valori diversi):

Il css sarà:

```
body {background:white;}
 div#contenitore {width:900px;}
  div#header {background:#4A2C3E;}
  div#colsx {background:#F90; width:650px; float:left;}
  div#coldx {background:#060; width:250px; float:left;}
  div#footer {background:#999; clear:left;}
```

In rosso le parti aggiunte. Noterete che utilizziamo le proprietà **float e clear** per far sì che le colonne si affianchino una all'altra ed il footer si riposizioni correttamente al di sotto di entrambe.

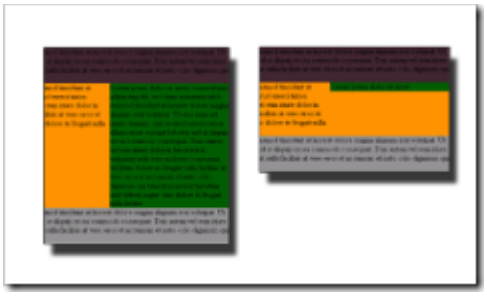
Il risultato finora:



Salta subito all'occhio che sotto la colonna sinistra abbiamo dello spazio bianco: ciò è dovuto al fatto che non abbiamo definito un'altezza e quindi il blocco, una volta riempito dal testo fittizio, finirà lasciando intravedere ciò che vi è sotto. Lo stesso problema, inverso avremo nel caso in cui l'altezza della colonna destra sia minore dell'altra.

Soluzioni possibili possono essere:

1. definire un'altezza comune ed aggiungere le scrollbar simulando i frame
2. definire un'altezza minima ai blocchi con min-height, utilizzando l'hack per i browser di casa Microsoft, tenendo in mente che comunque avremmo problemi con altri browser come safari
3. dare un colore di sfondo al contenitore uguale al blocco di minore altezza, avendo ben presente che se l'altro ha altezza maggiore avremo dei problemi.



Seguiremo la strada numero 3 ipotizzando che il blocco di destra abbia altezza maggiore e

comunque non preoccupandoci troppo del caso opposto.

Altra cosa che notiamo è che il tutto è allineato a sinistra. Volendo allinearlo a destra dovremmo usare la proprità **margin: 0 auto;** al blocco contenitore, proprietà che dice allo stesso di posizionarsi a distanza 0 da sopra e sotto ed automaticamente da destra e sinistra dal contenitore in modo centrato. IE non segue questa proprietà correttamente e per supplire al problema inseriamo nel body: **text-align:center**, che posiziona tutto ciò che c'è nel body al centro. Successivamente reimpostiamo **text-align:left;** in contenitore per dare allineamento a sinistra agli oggetti presenti.

Css con le modifiche al momento:

```
body {background:white; text-align:center;}
 div#contenitore {background:#F90; width:900px; margin:0 auto; text-align:left; }
  div#header {background:#4A2C3E;}
  div#colsx {background:#F90; width:650px; float:left;}
  div#coldx {background:#060; width:250px; float:left;}
  div#footer {background:#999; clear:left;}
```

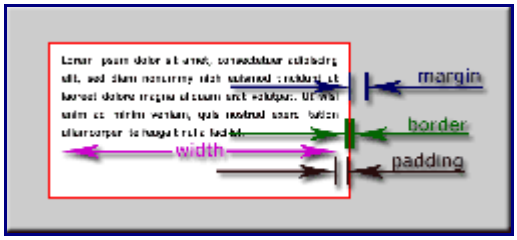
Anteprima:

Ci siamo quasi: impostiamo ora un famiglia ai caratteri, inseriamo il titolo della pagina ed il sottotitolo, il padding ai blocchi e ci fermiamo qui.

```
body {background:white; text-align:center; font-family:Helvetica, arial, sans-serif;}
 div#contenitore {background:#F90; width:900px; margin:0 auto; text-align:left; }
  div#header {background:#4A2C3E; height:5em;}
   h1 {color:white; text-align:right; padding:1em 1em 0 0; font-family:Georgia, serif;}
  div#colsx {width:630px; \width:650px; w\idth:630px; float:left; padding:10px;}
  div#coldx {background:#060; width:228px; \width: 250px; w\idth:228px; float:left;
padding:10px; border-left:2px solid white;}
  div#footer {background:#999; clear:left; border-top:3px solid #4A2C3E; text-align:center; }
```

In **rosso** vediamo le ultime modifiche; abbiamo

- impostato la famiglia di caratteri come helvetica, in mancanza del quale vedremo l'arial o il generico bastoni impostato dal browser;
- assegnato un'altezza di 5 volte l'altezza media di una riga all'header;
- creato una regola per l'h1 del titolo del sito: colore bianco, allineato a destra, con padding di 1em da sopra e da destra e con font Georgia o il bastoni di default;
- dato un padding alle due colonne di 10px per lato ed assegnato un bordo sinistro bianco di due pixel alla colonna destra;
- assegnato un bordo di 3 px in alto al footer dello stesso colore dell'header ed impostato un'allineamento al centro.

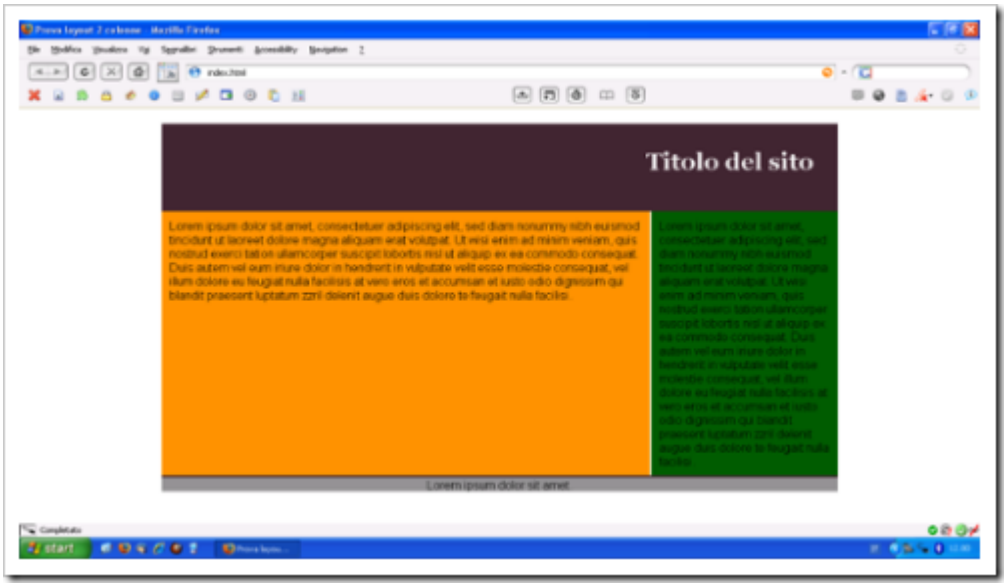


In **azzurro** sono presenti le modifiche fatte per evitare un errore di enterpretazione del box model di IE 5 ed in modo limitato del 6 (si verifica l'errore se non si è specificato un doctype). Qual è la logica? Il width, da specifiche del W3C, dovrebbe indicare la larghezza del blocco SENZA padding, bordo e margine, come indicato nell'immagine accanto.

Le vecchie versioni di IE recepiscono come width il width effettivo più il bordo ed il padding.

Noi siamo partiti da una larghezza di 650px per la prima colonna: aggiungendo il padding di 10px a destra ed a sinistra abbiamo dato come larghezza 630px (650-10-10); la seconda colonna era inizialmente di 250px; tolti 20px di padding e 2 di bordo resta una larghezza di 228px. Per le vecchie versioni di IE la larghezza dovrebbe esser immutata a 250. Dato che IE5 non riesce a leggere questa istruzione w\idth a differenza di altri browser, possiamo impostare la larghezza prima e ridefinirla successivamente con un'istruzione che non sarà ripresa dal browser Microsoft.

Questo sarà il risultato definitivo:



Esempio e codici.

Esempio di creazione di un form

Con questo tutorial illustreremo come creare un form in html+css da utilizzare successivamente come base per un modulo contatti senza l'uso di tabelle.

Il primo passo sarà quello di *costruire* il codice html:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head profile="http://gmpg.org/xfn/11">
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>Form di esempio</title>
 </head>
 <body>
  <form method="post" action="" onsubmit="alert('Disabilitato'); return false;"
id="form1">
    <fieldset>
     <legend>Anagrafica:</legend>
     <label for="nome">Nome: </label>
      <input type="text" id="nome" name="nome" tabindex="1" value="" title="Nome"
/><br />
     <label for="cognome">Cognome: </label>
      <input type="text" id="cognome" name="cognome" tabindex="2" title="Cognome"
/><br />
     <label for="email" class="required" >Email: </label>
      <input type="text" id="email" name="email" tabindex="3" title="Email" />
     </fieldset>
     <fieldset>
```

```
      <legend>Altri campi:</legend>
      <label for="chb_1" tabindex="4" >Seleziona:</label>
       <input type="checkbox" id="chb_1" name="chb_1" /><br />
      <label for="indirizzo">Indirizzo: </label>
       <textarea name="commenti" rows="3" cols="16" id="indirizzo" tabindex="5"
title="indirizzo"></textarea><br />
      <button type="submit" id="submit" tabindex="u">Invia</button>
     </fieldset>
     <p class="info"><em>* Campo obbligatorio.</em></p>
   </form>
  </body>
</html>
```

Noterete come abbia utilizzato **button** *type="submit"* al posto di **input** *type="submit"* per la maggior flessibilità data da quest'ultimo, la presenza dei **tabindex** per guidare il visitatore nel percorso attraverso il tasto di tabulazione e i tag **fieldset**, per raggruppare i campi simili, **legend** per "marcarli" e **label** per definire il contenuto dei campi.

Questo il risultato finora ottenuto, che potrete vedere anche in questa pagina.

Applichiamo lo stile, iniziando a definire le regole base per il body, a cui daremo sfondo bianco, font verdana e grandezza base di 12pixel e per il form che ci supponiamo debba avere una larghezza di 400 pixel.

```
body {background:#FFF; font-family:Verdana, sans-serif; font-size:12px;}
form {width:400px;}
```

Definiamo successivamente lo stile di fieldset e di legend.

```
fieldset {background:#FAEAAB url("bg_form1.jpg") repeat-y right; margin:2em 0;
padding:1.5em 0 0.5em 0; border-width:0.2em 0 0.2em 1em; border-style:solid; border-
color:#FC0; position:relative;}
legend {position:absolute; top: -11px; margin-left:10px; font-size:0.9em; font-weight:bold;
text-align:center; background:#EFE8FC; padding:0.1em 1em; color:#666; border-
width:0.2em 0.5em; border-style:solid; border-color:#AAA;}
```

Fieldset avrà un colore di sfondo arancio pastello e dei bordi arancio sottili in alto e spessi ai lati, con quello sinistro definito da un'immagine che avremo definito come sfondo, allineata a destra e ripetuta in verticale. Gli assegniamo una posizione esplicita relativa per avere un punto di riferimento con il tag legend che si posizionerà da esso e terminiamo aggiungendo opportuni margini e padding. Legend avrà ovviamente positione assoluta a partire da fieldset, gli daremo un margine sinistro per *staccarlo* dai bordi arancio, ed un top negativo per *allinearlo* al margine superiore. Il resto credo sia abbastanza chiaro a questo punto: allineamento centrale, grassettato, margini e padding per staccare il testo

dal box e bordi grigi.

Definiamo le etichette:

```
label {width:140px; display: -moz-inline-box; display: inline-block; padding:2px 6px;}
label:first-letter {font-size:1.3em;}
```

che abbiamo supposto larghe 140px. Notate la proprietà display: inline-block che firefox interpreta con -moz-inline-box. Serve a definire un blocco come inline: una via di mezzo tra un blocco vero e proprio senza che il campo successivo torni a capo.

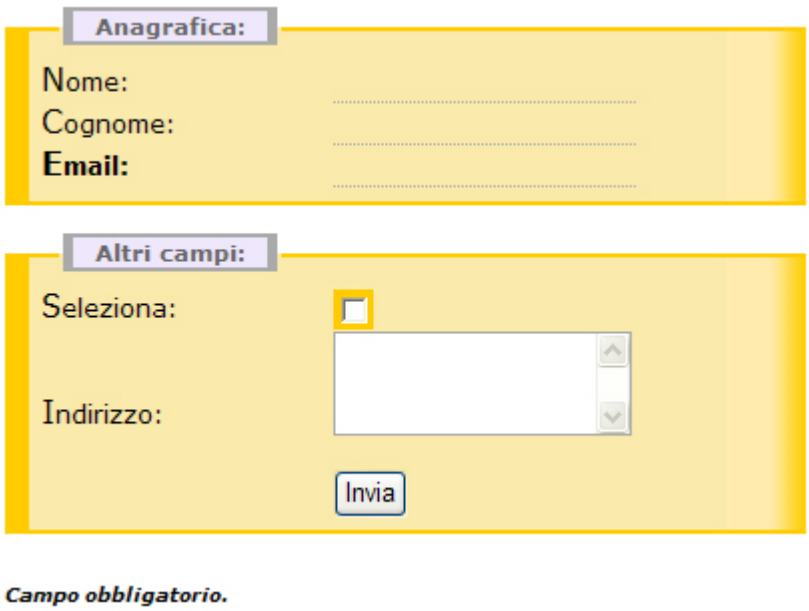
Prossimo passo sarà quello di definire i vari campi:

```
input {background:none; border-width:0 0 1px 0; border-style:dotted; border-color:#AAA;}
textarea {border:1px solid #AAA;}
#chb_1 {background:#FC0; margin-left:-4px; border:0;}
#submit {margin:1em 0 0 152px; text-align:center;}
```

con proprietà abbastanza intuitive da capire. Notate che ho preferito assegnare valori attraverso id specifici. In teoria i browser dovrebbero tutti supportare il comando **input[type=”checkbox”]** ma si sa come la teoria e la pratica siano due mondi abbastanzi distanti.

Il risultato finale sarà:

```
body {background:#FFF; font-family:Verdana, sans-serif; font-size:12px;}
form {width:400px;}
fieldset {background:#FAEAAB url(”bg_form1.jpg”) repeat-y right; margin:2em 0;
padding:1.5em 0 0.5em 0; border-width:0.2em 0 0.2em 1em; border-style:solid; border-
color:#FC0; position:relative;}
legend {position:absolute; top: -10px; margin-left:10px; font-size:0.9em; font-weight:bold;
text-align:center; background:#EFE8FC; padding:0.1em 1em; color:#666; border-
width:0.2em 0.5em; border-style:solid; border-color:#AAA;}
label {width:140px; display: -moz-inline-box; display: inline-block; padding:2px 6px;}
label:first-letter {font-size:1.3em;}
input {background:none; border-width:0 0 1px 0; border-style:dotted; border-color:#AAA;}
textarea {border:1px solid #AAA;}
#chb_1 {background:#FC0; margin-left:-4px; border:0;}
#submit {margin:1em 0 0 152px; text-align:center;}
label.required {font-weight:bold;}
p.info {font-size:0.8em; font-weight:bold;}
```



Qui troverete la pagina completa.

Approfondimenti:

- Galleria di form sullo stile di css-zen-garden;
- Esempio appronfondito di un form table-less.

Allineamento verticale con i css

Uno dei problemi a cui vanno incontri i neofiti che provano i css è l'allineamento verticale di un blocco, un'immagine, rispetto ad un blocco contenitore. Un qualcosa tipo valign="middle" delle tabelle.

Molti iniziano provando la proprietà vertical-align:middle.

Ovviamente non ottengono il risultato sperato. Perché? Poiché, come spiegato nel capitolo riguardante le proprietà dei testi, tale proprietà serve ad allineare più elementi **inline** successivi. Qui troverete degli esempi che valgono più di mille parole (o almeno spero :P).

Come fare allora per poter centrare un blocco di testo, un'immagine dentro un blocco? Abbiamo in effetti varie possibilità.

Consideriamo il primo caso: un menu con un'immagine di sfondo ed il testo su un'unica riga.

Qui vedete l'esempio che riportiamo per semplicità:


```
ul {list-style-type:none;}
  ul li {float:left; background:transparent url('bg-vert-align.gif') top center;
   width:100px; height:30px; text-align:center;
  }
   ul li a {font-size:12px; color:#111; text-decoration:none; }
    ul li a:hover {font-size:12px; color:#FFF; font-weight:bold;}
```


```
ul {list-style-type:none;}
  ul li {float:left; line-height:30px; background:transparent url('bg-vert-align.gif') top center;
   width:100px; height:30px; text-align:center;
  }
   ul li a {font-size:12px; color:#111; text-decoration:none; }
    ul li a:hover {font-size:12px; color:#FFF; font-weight:bold;}
```

noterete nell'ultimo caso l'inserimento della proprietà "line-height:30px;", cioè l'altezza dell'immagine di sfondo. In tal modo si forza l'interlinea alla grandezza dello sfondo, centrando automaticamente il testo all'etichetta.
Attenzione: tale tecnica ha una controindicazione. Se il testo occupa più linee, lo stesso scenderà in basso, con effetti assolutamente non voluti. Noterete tale problema ingrandendo il testo dell'esempio (con firefox » control [+]).

Se volessimo invece **centrare verticalmente un box dentro un altro box o alla pagina** si può ricorrere al posizionamento assoluto degli stessi.
Un ottimo articolo per tale scopo è presente su constile.org.

Come creare un sito a 3 colonne con i css

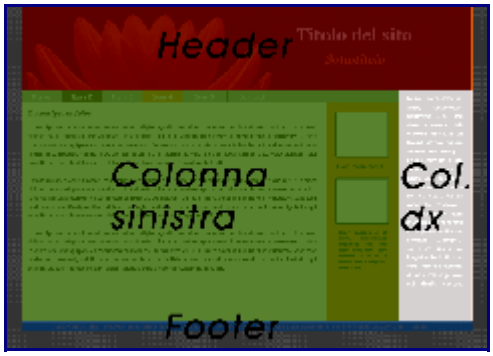
Con questo tutorial vedremo come costruire un layout a 3 colonne più header e footer completo.

Diamo per scontato che abbiate già letto il tutorial su come costruire un semplice layout a due colonne e quindi che abbiate compreso cos'è il box model, come aggirare le limitazioni di IE, cosa significhi elemento inline e blocco e come utilizzare le proprietà float e clear.

Supponiamo che il nostro grafico ci abbia inviato questo layout (spero per voi che abbia fatto anche di meglio :))

[clicca sulle immagini per ingrandirle]

Per prima cosa dovremmo decidere come dividere gli spazi *tipografici*: nell'immagine successiva noterete una mia personalissima suddivisione. Ovviamente voi potrete sceglierne un'altra se vi fosse più comoda o se non vi trovaste a vostro agio con quella da me scelta.



Iniziamo dunque a dividere la pagina con lo stesso metodo già utilizzato in questo tutorial:

Avremo a questo punto tale codice html:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="it">
 <head>
  <title>Prova layout 3 colonne</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <link type="text/css" rel="stylesheet" href="style.css" title="Style" media="all" />
 </head>
 <body>
  <div id="contenitore">
   <div id="header"> </div>
   <div id="colsx"> </div>
   <div id="coldx">Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam
nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi
enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip
ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit
esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan
et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait
```

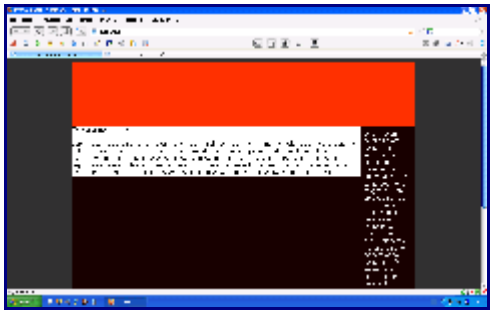
```
nulla facilisi.</div>
  <div id="footer"> </div>
 </div>
</body>
</html>
```

Ed il css sarà invece:

```
body {background:#333; text-align:center; font-family:Helvetica, arial, sans-serif;  font-
size:12px;}
 div#contenitore {background:#180000; width:915px; margin:0 auto; text-align:left; }
  div#header {background:#F30; height:170px;}
  div#colsx {background:#FFF; width:766px; \width:767px; w\idth:766px; float:left; border-
right:1px solid #CCC;}
  div#coldx {background:#180000; color:#FFF; width:127px; \width: 148px; w\idth:127px;
float:left; padding:10px; border-top:1px solid #333;}
  div#footer {background:#333; clear:left; border-top:6px solid #CCC; text-align:center; }
```

Noterete come assomigli terribilmente al codice del tutorial a 2 colonne già linkato sopra. Le uniche differenze pratiche sono l'aggiunta di un paio di bordi, in particolare il bordo di 6 pixel del footer.

Siamo arrivati a questo punto: (non preoccupatevi dello sfondo completamente marrone; quella parte sarà occupata dalla colonna sinistra. L'ho messo poiché suppongo che la parte sinistra sarà più alta della colonna destra e per tal motivo lo sfondo marrone servirà a compensare la mancanza di altezza di quest'ultima, come noterete tra poco).



Ci occuperemo ora della colonna sinistra. Dall'immagine noterete sicuramente che la parte in verde potrebbe esser divisa in 3 blocchi principali:

- il menu in alto alla stregua di un header classico;
- una colonna sinistra;
- ed una destra.

Seguendo la solita procedura aggiungeremo:

***xhtml** (solo body):*

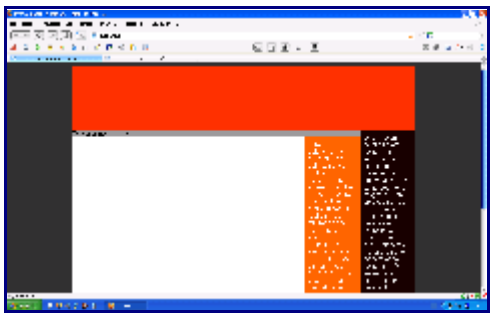
```
<body>
 <div id="contenitore">
  <div id="header"> </div>
  <div id="colsx">
     <div id="menu">Qui verr&agrave; inserito il menu</div>
```

```
        <div id="subcolsx"> </div>
        <div id="subcoldx">Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam
nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi
enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip
ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit
esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan
et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait
nulla facilisi.</div>
      </div>
      <div id="coldx">Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam
nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi
enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip
ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit
esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan
et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait
nulla facilisi.</div>
    <div class="clearlft"></div>
    <div id="footer" class="clearlft"> </div>
  </div>
</body>
```

css:

```
body {background:#333; text-align:center; font-family:Helvetica, arial, sans-serif;  font-
size:12px;}
  div#contenitore {background:#180000; width:915px; margin:0 auto; text-align:left; }
    div#header {background:#F30; height:170px;}
    div#colsx {background:#FFF; width:766px; \width:767px; w\idth:766px; float:left; border-
right:1px solid #CCC;}
        div#menu {background:#999;}
        div#subcolsx { background:#FFF; float:left; width:598px; \width: 619px; w\idth:598px;
padding:10px; border-right:1px solid #000;}
        div#subcoldx {background:#F60; color:#FFF; width:127px; \width: 147px;
w\idth:127px; float:left; padding:10px;}
    div#coldx {background:#180000; color:#FFF; width:127px; \width: 148px; w\idth:127px;
float:left; padding:10px; border-top:1px solid #333;}
    div#footer {background:#333; border-top:6px solid #CCC; text-align:center; }
  .clearlft {clear:left; }
```

Le parti in rosso son quelle appena aggiunte; ciò è quello che abbiamo ottenuto finora:



Iniziamo a "riempire" i blocchi dal menu. In giro son presenti moltissimi menu già pronti; in questo tutorial proveremo a farli da soli.

Sostituiamo il div del menu con una lista non ordinata, che riempiremo di elementi ai quali daremo la proprietà di essere elementi inline i quali conterranno i veri collegamenti.

Il css che scriveremo sarà:

```
ul#menu {background:#999; padding:0; margin:0; list-style:none; height:2.75em; border-
width:0.1em 0; border-color:#333; border-style:solid;}
 ul#menu li {float:left;}
  ul#menu li a {display:block; color:#FFF; background:#999; font-size:1.1em; line-
height:2.5em; padding:0 1.5em; text-decoration:none; border-right:0.08em solid #333; font-
family:Verdana, sans-serif; font-weight:bold;}
  ul#menu li a.current {background:#300000;}
  ul#menu li a:hover {background:#F60;}
```

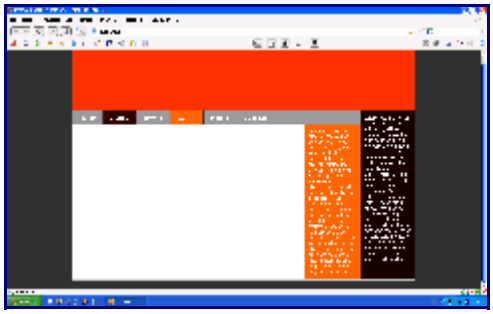
a cui corrisponderà questo codice xhtml:

```
<ul id="menu">
  <li><a href="#">Home</a></li>
  <li><a href="#" class="current">Item 2</a></li>
  <li><a href="#">Item 3</a></li>
  <li><a href="#">Item 4</a></li>
  <li><a href="#">Item 5</a></li>
  <li><a href="#">Contatti</a></li>
</ul>
```

Provo a spiegare cosa abbiamo fatto; abbiamo:

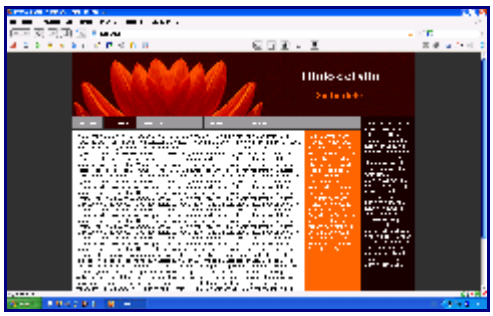
- indicato i vari elementi della lista (**li**) come inline, per affiancarli uno all'altro;
- indicato gli elementi **a** come blocchi per poter sfruttare la possibilità di cliccare su tutto il blocco e non solo sulla scritta;
- assegnato un colore alla barra del menu (#999) ed allo sfondo dei vari collegamenti;
- dato ai blocchi **a** un'altezza di 1.1 em cioè il 110% della grandezza standard dei caratteri e un'altezza delle linee di 2.5em (250%) del testo;
- assegnato al blocco menu un'altezza di 2.75em (cioè 2.5 di line-height*1.1);
- messo il bordino a tutto il blocco menu in alto ed in basso, di dimensioni 0.1em, colore #333, solido;
- dato il bordo ai blocchi a solo **a** destra per evitare sovrapposizioni. (0.08, cioè 0.1/1.2); *come sopra*
- tolto il padding ed il margine al menu per evitare spazi bianchi e tolto lo stile delle liste (il classico puntino tipo quello di questa lista);
- tolto la sottolineatura ai link ed assegnato il colore bianco;
- assegnato il colore per l'effetto hover e per la pagina corrente a cui abbiamo dato una classe apposita

Questo è il risultato:



Prossimo step è quello di lavorare sull'header. Ciò che ci serve è assegnare l'immagine di sfondo e posizionare il titolo con h1 ed il sottotitolo con h2. Vista la dimensione fissa della testata potremmo semplicemente ritagliare l'immagine ed assegnarla come sfondo. *Se fosse stato un layout liquido avremmo dovuto preoccuparci di creare un'immagine di sfondo di una larghezza molto ampia per coprire tutte le possibili risoluzioni; avremmo dovuto creare due immagini: una del fiore che avremo posizionato come immagine fissa a sinistra ed un pattern dello sfondo da ripetere lungo l'asse X.* Assegnamo poi le proprietà ai blocchi **h1** ed **h2**, assegnando ad entrambi le stesse proprietà: famiglia di caratteri Georgia, margine sinistro di 510 px, dato che vogliamo centrare gli stessi rispetto al fiore, largo appunto 510 pixels. Non ci resta che definire i colori ed il gioco è quasi fatto. Attenzione: dato che in h1 inseriremo un collegamento, dobbiamo dare il colore all'elemento a e non solo ad h1.
Le modifiche fatte per questo step saranno evidenziate nel codice con il colore azzurro.

Screenshot a questo punto:



Ci restano le ultime modifiche da fare:

- assegnare la grandezza e lo stile ai vari elementi **h1, h2, h3… hx;**
- cambiare il colore del testo al box arancio;
- dare l'effetto con le linee dello sfondo (prendendo un'immagine 4×4 e ripetendola);
- dare il colore al testo del footer ;
- togliere il padding al body;
- impostare la grandezza iniziale ai caratteri;

ed abbiamo finito. *Le modifiche fatte per questo step saranno evidenziate nel codice con il colore verde.*

xhtml finale:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="it">
 <head>
  <title>Prova layout 3 colonne</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <link type="text/css" rel="stylesheet" href="style.css" title="Style" media="all" />
 </head>
 <body>
  <div id="contenitore">
   <div id="header">
    <h1><a href="./index.html">Titolo del sito</a></h1>
    <h2>Sottotitolo</h2>
   </div>
   <div id="colsx">
    <ul id="menu">
     <li><a href="#">Home</a></li>
     <li><a href="#" class="current">Item 2</a></li>
     <li><a href="#">Item 3</a></li>
     <li><a href="#">Item 4</a></li>
     <li><a href="#">Item 5</a></li>
     <li><a href="#">Contatti</a></li>
    </ul>
   <div id="subcolsx">
    <h3>Titolo articolo</h3>
    <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh
euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim
veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo
consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie
consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio
dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla
facilisi.</p>
    <h3>Altro titolo</h3>
    <h4>Titolo sezione a</h4>
    <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh
euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim
veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo
consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie
consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio
dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla
```

```
facilisi.</p>
      <h4>Titolo sezione b</h4>
      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh
euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim
veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo
consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie
consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio
dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla
facilisi.</p>
      <h4>Titolo sezione c</h4>
      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh
euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim
veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo
consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie
consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio
dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla
facilisi.</p>
    </div>
    <div id="subcoldx">
      <h5>Titolo sezione</h5>
      <img src="./img1.gif" alt="immagine 1" />
      <p>Descrizione dell'immagine 1</p>
      <img src="./img2.gif" alt="immagine 2" />
      <p>Descrizione dell'immagine 2</p>
      <div>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh
euismod tincidunt ut laoreet</div>
    </div>
  </div>
  <div id="coldx">Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam
nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi
enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip
ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit
esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan
et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait
nulla facilisi.</div>
    <div class="clearlft"></div>
    <div id="footer" class="clearlft">Lorut laoreet dolore magna aliquam erat volutpat. Ut
wisi enim ad minim veniam, quis nostrud exerci tation</div>
  </div>
 </body>
</html>
```

css finale:

```
body {background:#333 url("./linee.gif"); text-align:center; font-family:Helvetica, arial, sans-
serif; font-size:12px; margin:0;}
 a img {border:0;}
 div#contenitore {background:#180000; width:915px; margin:0 auto; text-align:justify;}
  div#header {background:#F30 url("./header.jpg") no-repeat; height:170px;}
   div#header h1, div#header h2 {text-align:center; margin-left:510px;}
   div#header h1 { margin-top:0; padding-top:1em; color:#FFF;}
    div#header h1 a {color:#FFF;}
   div#header h2 {color:#F60;}
  div#colsx {background:#F60; width:766px; \width:767px; w\idth:766px; float:left; border-
right:1px solid #CCC; }
   ul#menu {background:#999; padding:0; margin:0; list-style:none; height:2.75em; border-
width:0.1em 0; border-color:#333; border-style:solid;}
    ul#menu li {float:left;}
     ul#menu li a {display:block; color:#FFF; background:#999; font-size:1.1em; line-
height:2.5em; padding:0 1.5em; text-decoration:none; border-right:0.08em solid #333; font-
family:Verdana, sans-serif; font-weight:bold;}
    ul#menu li a.current {background:#300000;}
    ul#menu li a:hover {background:#F60;}
  div#subcolsx { background:#FFF; float:left; width:598px; \width: 619px; w\idth:598px;
padding:10px; border-right:1px solid #000;}
  div#subcoldx {background:#F60; color:#180000; width:127px; \width: 147px;
w\idth:127px; float:left; padding:10px; text-align:center; margin-top:1em;}
  div#subcoldx {margin-top:0;}
```

```
  div#subcoldx img {border:2px outset #180000;}
 div#coldx {background:#180000; color:#FFF; width:127px; \width: 148px; w\idth:127px;
float:left; padding:10px; border-top:1px solid #333; }
 div#footer {background:#333; text-align:center; border-top:6px solid #CCC; color:#F60;
font-style:italic;}

 h1, h2, h3, h4, h5, h6 {font-family:Georgia, serif; }
 h1 {font-size:2.5em;}
 h2 {font-size:1.8em;}
 h3 {font-size:1.6em;}
 h4 {font-size:1.4em; font-style:italic;}
 h5 {font-size:1.2em; font-style:italic;}
 h6 {font-size:1.1em; font-style:italic;}
 .clearlft {clear:left; }
```

Ed ecco il risultato finale:

Esempio e codici.

Risorse

In questo capitolo indicheremo risorse e link utili per chi voglia approfondire l'argomento.

Innanzitutto una visita al sito del W3C dove son indicate le specifiche è d'obbligo per tutti. Per chi avessi problemi con l'inglese è disponibile una traduzione in italiano.

Un altro indirizzo da avere sottomano è w3schools.com, in particolare la sezione css dove troverete notizie, consigli ed esempi per ogni regola.

Per controllare la bontà del nostro lavoro è utile utilizzare un validatore. Il migliore è probabilmente ed ovviamente quello del W3C.

Un'ottima risorsa in italiano è sicuramente constile.org dove son presenti tutorial, esempi, consigli e suggerimenti sui fogli di stile.

Nei segnalibri di un coder css non dovrebbero mancar inoltre: http://www.alistapart.com/ e http://glish.com/css/ due ottime risorse in inglese.

Esistono poi tool semi-automatici per creare liste e form ed una raccolta di layout divisi per numero di colonne e tipo layout (fisso o liquido).

Infine son sicuramente da visitare CSS Zen Garden e CSS Zen Sentiero che mostrano come i fogli di stile non siano un limite per gli artisti del web design, tutt'altro!

Note sull'autore

Massimiliano Carnevale, conosciuto in rete con il nick "**Massy**", è responsabile di ***Manie Grafiche***: una web agency che offre servizi web che spaziano dalla **realizzazione di siti internet** (e-commerce, portali, sistemi di prenotazione online, corporate blog), al **posizionamento sui motori di ricerca**, alle varie forme di **visibilità online**.
Con una forte attenzione al lato commerciale: ovvero a **come far convertire un visitatore in un cliente**.

Grazie alle competenze in vari settori (*programmazione, seo, accessibilità, copywriting, grafica*) questa agenzia offre la possibilità di fare un sito internet da zero, oppure di intervenire su progetti già esistenti, integrandoli con nuove funzionalità (o effettuando un restyling grafico-strutturale).

Oltre alla parte online, *Manie Grafiche* realizza servizi di **grafica "tradizionale"**: produzione di materiale pubblicitario di vario tipo come ad esempio **depliant, brochure, biglietti da visita**..

Nel tempo libero Massy è editore in Dmoz, nella categoria, World/Italiano, legge fumetti, Rat-Man in particolare, è fotoamatore e, quando il tempo lo permette, suona il sax.