



# Cascading Style Sheets

---

**Giuseppe Della Penna**

Università degli Studi di L'Aquila

*giuseppe.dellapenna@univaq.it*

*<http://www.di.univaq.it/gdellape>*





# Compatibilità Cross-Browser

## Modalità Standards e Quirks

- I Browser supportano due modalità di rendering: *Standards mode* and *Quirks mode*.
  - Lo Standards mode lavora seguendo il più possibile le specifiche del W3C, quindi in maniera (quasi) indipendente dal browser.
  - Il Quirks mode segue le regole di formattazione dello specifico browser, con le sue limitazioni ed estensioni.
- La modalità Quirks esiste per rendere i browser compatibili con i vecchi siti web, che erano sviluppati con codice molto *browser-dipendente*. Oggi, è *necessario* sviluppare i nuovi siti in modalità Standards.
- (!) Di default i browser usano la modalità Quirks. Per entrare in modalità Standards occorre inserire all'inizio del documento una dichiarazione doctype come quella che segue
  - Per usare l'**XHTML transitional**:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```
  - Per usare l'**XHTML strict**:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

# I Fogli di Stile

- Un foglio di stile CSS è un documento di testo, costituito da una serie di *regole di stile*.
- Un documento può avere anche più di un foglio di stile associato.
- Un foglio di stile, in generale, può essere *incorporato* in un documento o *collegato* ad esso. In HTML:
  - i fogli di stile CSS incorporati vanno inseriti nella sezione <head> all'interno di tag <style> aventi type="text/css".
  - (i) è anche possibile incorporare una regola di stile inerente uno specifico elemento scrivendola direttamente nell'attributo style dello stesso.
  - i fogli di stile CSS possono essere collegati a una risorsa esterna inserendo nella sezione <head> dei tag <link> aventi type="text/css", rel="stylesheet" e href="uri\_del\_foglio\_di\_stile".



# Fogli di Stile Multipli

- Se si specificano più fogli di stile per lo stesso documento, CSS in generale li combina in ordine di inclusione.
- Tuttavia, in HTML è possibile specificare l'inclusione condizionale di un foglio di stile (o un gruppo di fogli) in un documento in base a vari criteri:
  - Il media di output
  - Le preferenze dell'utente

# Fogli di Stile Multipli

## Media Types

- Quando si incorpora o collega un foglio di stile a un documento HTML, è possibile specificare nell'elemento `<style>` o `<link>` l'attributo *media*.
- Il valore di *media* è una lista separata da virgole di nomi di media descriptors, che identificano il tipo di output per il quale il foglio di stile è adatto:
  - screen (default)
  - tty, tv, projection, handheld: indicano diversi tipi di terminali di tipo visivo
  - print: il foglio di stile selezionato per la stampa (utile ad esempio per eliminare o cambiare colori o elementi opzionali presenti nel rendering su schermo)
  - aural, braille: rendering per sintetizzatori vocali o braille
  - all: tutti i tipi di media



# Fogli di Stile Multipli

## Fogli di Stile Alternativi

- In HTML è possibile dotare di una pagina di tre diversi tipi di stile per lo stesso tipo di media:
  - **Stile persistente:** sono sempre caricati dal browser. Gli stili incorporati nel documento sono sempre persistenti.
  - **Stile preferito:** si tratta dello stile di default che verrà combinato a quello persistente, se presente. Si indica inserendo l'attributo `title="nome_stile"` all'interno del tag `<link>`.
  - **Stili alternativi:** sono stili che possono essere caricati alternativamente a quello preferito, a seconda delle preferenze dell'utente. Si indicano inserendo l'attributo `title="nome_stile"` all'interno del tag `<link>` e modificandone l'attributo `rel` ad `"alternate stylesheet"`.

# Regole

- Una regola CSS definisce uno *stile di formattazione* e una *classe di elementi* a cui deve essere applicato.
- Lo *stile di formattazione* è a sua volta definito da una lista di proprietà valorizzate, con la sintassi **proprietà: valore**, poste all'interno di parentesi graffe e separate da un punto e virgola.
- La *classe di elementi* è invece definita con speciali pattern detti *selettori*.
- Un esempio di regola astratta è  
SEL {P1: V1 [!important]; P2: V2; P3: V3}
- Il modificatore **!important**, opzionale, scritto dopo il valore (ma prima del separatore) di qualsiasi proprietà, serve ad aumentare la *priorità della regola* durante il processo di *cascading*, come vedremo più avanti.



# Regole

## Selettori Semplici

- Un selettore semplice è un selettore di base seguito da zero o più *selettori di attributo*, *selettori di classe*, *selettori di ID*, *pseudo classi* e *pseudo elementi*.
- I *selettori di base* sono due:
  - Il *selettore universale* (\*) fa match con ogni elemento.
  - I *selettori di tipo* (stringhe rappresentanti nomi di elementi) fanno match con gli elementi aventi il nome corrispondente.

# Regole

## Selettori di Attributo

- Ad ogni selettore di base possono essere posposti dei selettori di attributo. I possibili selettori di questo tipo sono:
  - **[A]** (l'elemento deve avere un attributo A)
  - **[A=V]** (l'elemento deve avere un attributo A con valore V)
  - **[A~ =V]** (l'elemento deve avere un attributo A il cui valore è una lista separata da spazi contenente V)
  - **[A | =V]** (l'elemento deve avere un attributo A il cui valore è una lista separata da '-' contenente V)

# Regole

## Selettori di Classe

- Quando si lavora con HTML, è disponibile una speciale sintassi abbreviata, detta *selettore di classe*, per lavorare con l'attributo class degli elementi
  - La sintassi “S.C”, applicabile a qualsiasi selettore semplice S, è equivalente a  $S[class\sim=C]$ .
  - Come caso speciale, è possibile scrivere il selettore di classe da solo (mentre in generale i selettori di attributo devono seguire un altro selettore valido), sottintendendo, prima di esso, il selettore universale:  $.C$  è equivalente a  $*.C$ , cioè  $*[class\sim=C]$

# Regole

## Selettori di ID

- In XML (e in HTML) è possibile assegnare ad ogni elemento un ID univoco.
- Questo ID può essere usato nei CSS per applicare formattazione a un elemento specifico.
- Il selettore ID può essere posto dopo ogni selettore di base ed ha la sintassi #ID.
  - Il selettore **S#ID** fa match con l'elemento che corrisponde al selettore S ed ha l'ID specificato
  - (i) E' possibile (e molto comune) usare i selettori di ID da soli, proprio come quelli di classe, sottintendendo un selettore universale: **#ID** equivale a **\*#ID**, cioè all'elemento (di qualunque tipo) avente l'ID specificato.



# Regole

## Pseudo classi

- Le *pseudo classi* permettono di identificare elementi in base ad alcune loro particolari proprietà.
  - **:first-child** (l'elemento è il *primo figlio* del suo genitore)
  - **:link** (*link non visitato*)
  - **:visited** (*link visitato*)
  - **:hover** (elemento attualmente *indicato dall'utente*, ad esempio passandovi sopra il mouse)
  - **:focus** (elemento attualmente *con focus*, cioè che accetta input da tastiera)
  - **:active** (elemento correntemente *attivato*, ad esempio da un click del mouse)

# Regole

## Pseudo elementi

- E' infine possibile applicare formattazione ad elementi fittizi, non propriamente facenti parte del documento e/o delimitati da tag. Questi elementi si chiamano *pseudo elementi*.
  - **:first-line** (la prima riga del blocco di testo contenuto nell'elemento)
  - **:first-letter** (il primo carattere del blocco di testo contenuto nell'elemento)
  - **:before**, **:after** (indicano le posizioni precedente e successiva all'elemento. Si usano un congiunzione con la proprietà CSS content)

# Regole

## Combinazioni di Selettori

- Due selettori  $S$  e  $T$  possono essere combinati in un terzo selettore in vari modi:
- $S T$  (spazio intermedio)  
Questo selettore fa match con gli elementi indicati da  $T$  solo se sono discendenti di un elemento che fa match con  $S$
- $S > T$   
Questo selettore fa match con gli elementi indicati da  $T$  solo se sono figli di un elemento che fa match con  $S$
- $S + T$   
Questo selettore fa match con gli elementi indicati da  $T$  solo se seguono immediatamente un elemento che fa match con  $S$ , e condividono con questo lo stesso genitore.
- $S, T$   
Questo selettore fa match con gli elementi indicati da  $S$  e da  $T$  (or logico o *raggruppamento di selettori*)



# Calcolo dei Valori delle Proprietà di Stile

- Durante il rendering, CSS deve determinare lo stile da assegnare *a ciascun elemento del documento*.
  - Questo implica calcolare il valore *di ogni singola proprietà di stile* che l'elemento può avere.
- Per calcolare il valore di una specifica proprietà P per un elemento E, CSS procede come segue:
  1. **Regole di cascading**  
La proprietà ha uno stile specificato tramite regole CSS?
  2. **Ereditarietà**  
La proprietà può essere ereditata dall'elemento padre?
  3. **Valore di default**  
La proprietà assume il valore dato dallo stylesheet di default.



# Cascading

- In un foglio di stile è possibile (e a volte molto utile) che esistano più regole che fanno match con gli stessi elementi.
- Inoltre CSS prevede, oltre al foglio di stile dato dall'autore (cioè quello associato al documento), altri due fogli di stile da considerare:
  - Foglio di stile utente. L'utente può fornire delle regole di stile, come ad esempio la dimensione di base dei caratteri.
  - Foglio di stile del browser. Ogni browser deve avere un suo foglio di stile di default.
- Quando CSS calcola il valore di *ogni singola proprietà di stile*, prende in considerazione tutte le regole che fanno match con l'elemento da formattare in tutti i fogli di stile, e ne seleziona una sola con il metodo della **cascata**.

# Cascading

## Regole di Selezione

- Quando più regole fanno match con lo stesso elemento, l'effettivo valore di ciascuna proprietà viene selezionato tramite la seguente procedura.
  1. **Priorità di Origine**
    1. Valore !important del foglio di stile utente
    2. Valore !important del foglio di stile autore
    3. Valore del foglio di stile autore
    4. Valore del foglio di stile utente
    5. Valore di default del foglio di stile del browser
  2. **Specificità** (*per proprietà con la stessa priorità di origine*)
    - Un selettore più specifico per un determinato elemento ha la precedenza su uno più generico.
    - In HTML, le regole inserite nell'attributo style dell'elemento sono considerate con la massima specificità.
  3. **Ordine** (*per proprietà con stessa priorità di origine e specificità*)
    - Una regola ha la precedenza su quelle che la *precedono*.

# Ereditarietà

- Molte proprietà (si veda la specifica) vengono automaticamente *ereditate* dai figli di un elemento, se non esistono regole specifiche che ne calcolano un valore diverso.
  - Questo comportamento di default è molto utile quando si creano fogli di stile complessi.
  - Ad esempio, se si specifica un font per i tag P, tutti i tag in esso contenuti (ad esempio B) avranno lo stesso font, a meno che non venga loro assegnato (globalmente o individualmente) uno stile che specifichi un font diverso.
- E' inoltre possibile forzare l'ereditarietà di una proprietà specificando (ove possibile) la parola chiave *inherit* come valore della proprietà stessa.

# Elementi Base del Linguaggio CSS

## Misure ed Unità di Misura

- Le misure vengono espresse nel linguaggio CSS da numeri (anche decimali, e in alcuni casi negativi) seguiti immediatamente dal codice dell'unità di misura.
  - (!) Inserire uno spazio tra il numero e il codice rende di solito illeggibile la proprietà!
  - (i) La misura zero può essere specificata anche senza un codice di unità.
- Esistono due classi di unità di misura: *relative* e *absolute*.
- Le unità *relative* sono **em** (font-size attuale), **ex** (x-height attuale) e **px** (pixel del dispositivo).
  - Sono molto indicate nel caso in cui si desideri far adattare automaticamente le dimensioni al dispositivo e alle sue impostazioni (es. stampante o browser con varie dimensioni di carattere)
- Le unità *absolute* sono **in** (pollici), **cm** (centimetri), **mm** (millimetri), **pt** (punti = 1/72 di pollice), **pc** (pica = 12 punti) .
  - Sono utili solo quando il dispositivo di output è unico e precisamente definito.
- In molti casi, le misure possono anche essere espresse come *percentuali*. La misura di riferimento è di solito quella della stessa proprietà nell'elemento contenitore.



# Elementi Base del Linguaggio CSS

## Colori

- Il colori possono essere definiti nel linguaggio CSS tramite la specifica numerica RGB o attraverso il loro nome proprio .
- Le *specifica RGB* permette di definire qualsiasi colore attraverso il valore delle sue tre componenti *rossa, verde e blu*.
- Le *stringhe RGB esadecimali* hanno la forma **#RRGGBB**, in cui ciascuna coppia di cifre rappresenta il valore esadecimale della rispettiva componente.
  - (i) La forma abbreviata **#RGB** rappresenta il numero in cui ogni componente ha come valore quello della cifra corrispondente ripetuto per due volte.
- Le *stringhe RGB decimali*, invece, si ottengono utilizzando il costrutto **rgb(R,G,B)**, dove R, G e B sono numeri compresi tra 0 e 255 o percentuali (intese come frazioni del valore massimo 255).
- Infine, i colori per cui è definito un *nome proprio* sono **maroon (#800000)**, **red (#ff0000)**, **orange (#ffa500)**, **yellow (#ffff00)**, **olive (#808000)**, **purple (#800080)**, **fuchsia (#ff00ff)**, **white #ffffff**, **lime (#00ff00)**, **green (#008000)**, **navy (#000080)**, **blue (#0000ff)**, **aqua (#00ffff)**, **teal (#008080)**, **black (#000000)**, **silver (#c0c0c0)** e **gray (#808080)**



# Elementi Base del Linguaggio CSS

## Shortand Properties

- Il linguaggio CSS dispone di molte proprietà che spesso vengono impostate in gruppo, come ad esempio le tre proprietà che definiscono un bordo (colore, spessore, stile) o le proprietà di un font (famiglia, dimensione, peso, ...).
- Per questo motivo, sono disponibili anche le cosiddette proprietà *shorthand*, che permettono, con la loro particolare sintassi, di impostare con una sola operazione i valori di diverse proprietà.
- Nell'impostare una proprietà *shorthand*, i valori di ogni singola proprietà “componente” vengono scritti di seguito, separati da uno spazio. L'ordine non è di solito significativo, in quanto non esistono ambiguità nel linguaggio.
  - (!) Se una o più proprietà vengono omesse nell'impostazione *shorthand*, il loro valore è considerato quello di default.
- (i) Ad esempio, la proprietà CSS font può essere usata per impostare contemporaneamente tutte le proprietà font-style, font-variant, font-weight, font-size, line-height e font-family.



# Bordi

- La maggior parte degli elementi possono essere dotati di un bordo sui quattro lati del loro box. Ogni bordo può possedere differenti caratteristiche (colore, spessore, tratteggio). E' inoltre possibile, nel caso dei bordi delle tabelle, specificare come i bordi adiacenti debbano combinarsi.
- Nota: I bordi delle tabelle specificati tramite CSS sono indipendenti da quelli mostrati tramite l'attributo border di <table>.
- I colori dei bordi possono essere specificati tramite nomi simbolici (ad es. White), o tramite le loro componenti rgb in esadecimale (ad es. #FFFFFF) o decimale (ad es. rgb(255,255,255))
- Lo spessore dei bordi è un valore che può essere specificato in una qualsiasi delle unità di misura accettate dai CSS (ad es. px, pt, mm, ...)
- I principali stili per i bordi sono dotted, dashed, solid, double, groove, ridge, inset, outset. Tuttavia, la maggior parte dei browser supporta solo solid, dotted e dashed.
- Per tutte le proprietà esistono degli shorthand che permettono di impostare gli stessi valori per tutti i lati contemporaneamente e/o di specificare le tre proprietà (colore, spessore, stile) con un'unica istruzione.



# Bordi

## Proprietà CSS

- border (border-top, border-right, border-bottom, border-left)
  - Valori: *{spessore, stile, colore}*
- border-color (border-top-color, border-right-color, border-bottom-color, border-left-color)
  - Valori: *colore* | transparent
- border-style (border-top-style, border-right-style, border-bottom-style, border-left-style)
  - Valori: none | *nome stile bordo* | inherit
- border-width (border-top-width, border-right-width, border-bottom-width, border-left-width)
  - Valori: *misura*
- border-collapse
  - Valori: collapse | separate
  - Elementi: tabelle ed elementi interni inline

# Sfondo

- Tutti gli elementi di tipo blocco possono essere dotati di uno sfondo, costituito da un colore solido o da un'immagine
- Nel caso di sfondi costituiti da immagini, il file da utilizzare deve essere indicato tramite il costrutto url('...') ed è possibile specificare:
  - In che posizione far apparire l'immagine rispetto all'elemento di cui è sfondo.
  - Se l'immagine dovrà essere ripetuta per riempire l'intera superficie messa a disposizione dall'elemento.
  - Se l'immagine debba “scorrere” insieme al contenuto della finestra o rimanere fissa.
- (i) Grazie all'elevata versatilità degli sfondi, questi vengono spesso utilizzati per scopi impropri, ad esempio per creare effetti grafici (bottoni, elementi strutturali della pagina, ecc) che non sarebbero realizzabili importando semplicemente le immagini tramite il costrutto `<img>` di HTML.

# Sfondo

## Proprietà CSS

- background-color (colore di fondo)
  - Valori: *colore* | transparent
- background-attachment (ancoraggio dello sfondo)
  - Valori: scroll | fixed | inherit
- background-image (immagine di fondo)
  - Valori: *url* | none | inherit
- background-position (posizione dell'immagine di fondo)
  - Valori: left top | left center | left bottom | right top | right center | right bottom | center top | center center | center bottom | *x% y%* | *misura misura* | inherit
- background-repeat (estensione dell'immagine di fondo)
  - Valori: repeat | repeat-x | repeat-y | no-repeat | inherit



# Formattazione

## caratteri

- color (colore carattere)
  - Valori: *colore*
- font-family (tipo di carattere)
  - Valori: uno o più nomi di font, separati da virgole, in ordine di priorità
- font-size (dimensione carattere)
  - Valori: *misura*
- font-style (corsivo)
  - Valori: normal | italic | oblique
- font-variant (maiuscoletto)
  - Valori: normal | small-caps
- font-weight (grassetto)
  - Valori: normal | bold | bolder | lighter
- text-decoration (sottolineatura)
  - Valori: none | underline | overline | line-through
- text-transform (maiuscole/minuscole)
  - Valori: capitalize | uppercase | lowercase | none



# Formattazione

## paragrafi

- line-height (altezza delle righe)
  - Valori: *misura* | normal
- text-align (allineamento orizzontale paragrafi)
  - Valori: left | right | center | justify
- vertical-align (allineamento verticale paragrafi)
  - Valori: top | middle | bottom
  - Elementi: celle di tabelle
- text-indent (rietro sinistro paragrafi)
  - Valori: *misura*
- word-spacing (spazio tra parole)
  - Valori: *misura* | normal
- letter-spacing (spazio tra lettere)
  - Valori: *misura* | normal

# Liste

- Tramite CSS è possibile creare liste puntate e numerate di tipo semplice, con immagini standard o numeri (arabi, romani, ecc.) come punti elenco, usando l'attributo `list-style-type`.
- La funzionalità più avanzata resa disponibile tramite l'attributo `list-style-image` prevede l'uso di immagini come punti elenco. In questo caso, l'attributo `list-style-type` viene ignorato.
- Quando si creano liste di tipo personalizzato con immagini, bisogna sempre regolare i rientri, il padding e i margini degli elementi in modo da ottenere l'effetto visivo desiderato.
  - (!) A seconda del browser, il margine e/o il padding dell'elemento `list-item` determinano il rientro dell'elemento stesso e/o lo spazio tra il punto elenco e il testo associato.
- (i) Gli attributi di tipo `list` possono essere applicati a tutti gli elementi per la cui proprietà `display` sia impostata al valore `list-item`.

# Liste

## Proprietà CSS

- **list-style-type** (tipi standard di punto elenco)
  - **Valori:** disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-latin | upper-latin | armenian | georgian | lower-alpha | upper-alpha | none |
- **list-style-image** (punto elenco immagine)
  - **Valori:** *uri* | none
- **list-style-position** (posizione del punto elenco rispetto al testo)
  - **Valori:** inside | outside

# Box Model

## Controllare la Generazione dei Box

- E' possibile specificare in che modo debba essere generato il box associato a un elemento.
- display
  - **Valori:** inline | block | list-item | none
  - **Block** genera un box di tipo blocco (che occupa uno spazio orizzontale e verticale separato dagli altri box, come p o div)
  - **Inline** genera un box inline, che entra a far parte del flusso esterno senza interromperlo (come b o span)
  - **List-item** visualizza il box come un elemento di lista (come li)
  - **None** disattiva la generazione del box, rimuovendo l'elemento associato dal documento a tutti gli effetti.

# Box Model

## Mostrare e Nascondere Elementi

- Dopo aver generato il box relativo a un elemento, si può specificare se il contenuto del box debba essere o meno renderizzato.
- **visibility**
  - **Valori:** **visible** | hidden
  - **Visible** (default) mostra l'elemento.
  - **Hidden** nasconde l'elemento.
- (!) Impostando la proprietà ad hidden, il box dell'elemento non viene rimosso dal flusso del documento, per cui il suo ingombro viene comunque considerato nel calcolo del layout.

# Box Model

## Gestione del Contenuto

- Generalmente il contenuto di un blocco è limitato alle dimensioni del blocco stesso. Il contenuto può però essere più grande del suo contenitore.
- In questi casi, si può specificare come trattare la parte che fuoriesce dal contenitore.
- overflow
  - **Valori:** **visible** | hidden | scroll | auto
  - **Visible** (default) lascia che il contenuto extra sia renderizzato anche fuori dal contenitore.
  - **Hidden** nasconde la parte di contenuto che fuoriesce dal contenitore.
  - **Scroll** fa apparire delle barre di scorrimento all'interno del contenitore, in modo che il contenuto possa essere gestito tramite scrolling. (!) Le barre appariranno in ogni caso, anche se il contenuto non fuoriesce.
  - **Auto** fa apparire delle barre di scorrimento all'interno del contenitore, in modo che il contenuto possa essere gestito tramite scrolling, solo se questo fuoriesce dal contenitore.

# Box Model

## Margini e Spazi

- margin (margin-right, margin-left, margin-top, margin-bottom)
  - Il margine è lo spazio vuoto lasciato tra il bordo del box di un oggetto e quello degli oggetti circostanti.
  - **Valori:** *misura*
- padding (padding-top, padding-right, padding-bottom, padding-left)
  - Il padding è lo spazio vuoto lasciato tra il bordo del box di un oggetto e il contenuto del box stesso.
  - **Valori:** *misura*
- (!) In molti casi, margin e padding producono lo stesso effetto visivo, ma bisogna sempre usare l'attributo logicamente più adatto allo scopo.

# Box Model

## Dimensionamento

- La dimensione di ogni elemento può essere impostata in vari modi, sovrascrivendo completamente o vincolando la dimensione naturale calcolata dal browser.
- Nel primo caso, è possibile specificare misure assolute o percentuali, che si intendono applicate alle corrispondenti dimensioni del contenitore.
- Nel secondo caso, è possibile specificare dimensioni minime o massime, anch'esse espresse come misure assolute o percentuali.

# Box Model

## Proprietà di Dimensionamento

- width, height
  - Valori: *misura* | **auto** | inherit
  - Impostano rispettivamente l'ampiezza e l'altezza dell'elemento. Il valore auto corrisponde alla dimensione naturale, calcolata attraverso le altre proprietà dell'elemento.
- max-height, max-width, min-height, min-width
  - Valori: *misura* | inherit
  - Impostano le dimensioni minime o massime dell'elemento.

# Box Model

## Posizionamento

- Gli elementi possono essere posizionati in quattro modi diversi, specificabili con l'attributo CSS `position`:
  - **Static**: l'oggetto viene posto nella sua posizione naturale, data dal flusso del testo (default).
  - **Relative**: l'oggetto viene posizionato alle coordinate impostate tramite gli attributi `left`, `right`, `top` e `bottom` e relative alla sua posizione naturale.
  - **Absolute**: l'oggetto viene posizionato alle coordinate impostate tramite gli attributi `left`, `right`, `top` e `bottom` e relative all'angolo superiore sinistro del suo più diretto contenitore *con posizionamento non statico*.
  - **Fixed**: l'oggetto viene posizionato alle coordinate impostate tramite gli attributi `left`, `right`, `top` e `bottom` e relative all'angolo superiore sinistro del viewport.
  - Gli elementi posizionati in maniera *absolute* o *fixed* si dicono *rimossi dal flusso del testo*. La loro posizione non dipende più dagli elementi che li circondano, anche se, nel caso *absolute*, continuano a scorrere insieme al resto della pagina.

# Box Model

## Proprietà di Posizionamento

- **position**
  - **Valori:** static | relative | absolute | fixed
  - Determina il tipo di posizionamento dell'elemento
- **top, left, right, bottom**
  - **Valori:** *misura*
  - Determina la posizione dell'elemento, in base alle regole definite dal valore della proprietà position
- **z-index**
  - **Valori:** *numero* | auto | inherit
  - Determina la posizione dell'elemento (posizionato) sull'asse Z. Valori maggiori spostano l'elemento verso l'utente.

# Box Model

## Floats

- Tramite la tecnica di floating è possibile rimuovere degli elementi dal flusso del testo e posizionarli in maniera dinamica sul bordo sinistro o destro del loro contenitore.
- Gli elementi posizionati tramite floating si distribuiscono sempre al meglio in relazione allo spazio a loro disposizione.
- Il testo all'esterno degli elementi floating scorre intorno al loro margine in maniera automatica.
- Questo tipo di effetto è spesso usato per creare menu, layout a colonne, ecc.



# Box Model

## Proprietà per i Floats

### ■ float

- **Valori:** left | right | none
- Imposta l'oggetto come floating sul lato sinistro o destro del contenitore. Il valore none disabilita il floating.

### ■ clear

- **Valori:** left | right | both
- La proprietà clear impostata su un elemento fa sì che tutti i floats del tipo specificato (sinistri, destri o entrambi) vengano disposti prima dell'elemento stesso.

# CSS Media Queries

- Tra le caratteristiche introdotte da CSS3 che sono supportate ormai da tutti i browser moderni ci sono le *media queries*.
- E' possibile usare le media queries
  - All'interno dell'attributo *media* del tag link, per importare un certo foglio di stile solo se la query è soddisfatta, oppure
  - Direttamente nei fogli di stile racchiudendo insieme di regole all'interno di parentesi graffe, precedute dalla *at-rule @media*
- Una media query è composta da un *media type* (*screen, print, ecc.*) messa in AND con una serie di *media features*, il cui supporto può variare nei browser.
  - orientation: [portrait | landscape]
  - width: <misura>, min-width: <misura>, max-width: <misura>
- E' possibile mettere in OR più media queries separandole con una virgola, ad esempio

```
@media screen and (min-width: 300px),
screen and (orientation: landscape) {...}
```
- Un'interessante libreria di supporto (da provare!) utilizza javascript per rendere disponibili le media queries anche nei browser più datati:  
<http://code.google.com/p/css3-mediaqueries-js/>

# Responsive Design

- Il **responsive design** è una tecnica di progettazione dei layout resa famosa dall'articolo del 2010 di Ethan Marcotte su *A List Apart*:  
<http://alistapart.com/article/responsive-web-design>
- Con questa tecnica il layout di un sito web *si adatta dinamicamente alle dimensioni e alle caratteristiche del dispositivo di output*.
- I tre componenti di un responsive design sono
  - i layout fluidi (tipicamente a griglia),
  - le CSS3 media queries
  - le immagini flessibili
- Prima delle CSS media queries, un design responsive era ottenibile solo con l'ausilio di design fluidi supportati da script, mentre ora è possibile ottenere effetti molto più avanzati utilizzando i soli fogli di stile.
- Applicando variazioni specifiche ai fogli di stile tramite le media queries, è possibile ad esempio nascondere elementi, riposizionarne altri, diminuire bordi e spaziature, ecc.

# Responsive Design

## Esempio

- Per i browser “normali”
  - Design liquido, con ampiezze percentuali (possibilmente anche per le immagini). Un design liquido rende il layout robusto anche per i dispositivi che non supportano ancora le media queries.
- Per i tablet:
  - Ad esempio: *@media only screen and (min-width: 768px) and (max-width: 959px)*
  - Eliminare padding e spaziature “creative”, diminuire leggermente la dimensione del carattere, ecc.
- Per i cellulari:
  - Ad esempio: *@media only screen and (max-width: 767px)*
  - Design “fixed”, ad esempio di 320 pixel, oppure inserire una min-width sugli elementi principali per evitare che si riducano troppo.
  - Eliminare le colonne linearizzandone il contenuto, nascondere gli elementi secondari (parte di header e footer, ecc.), mostrare menu più compatti.

# Un Layout Tabulare con i Floats

- Vedremo ora come realizzare un **layout liquido a griglia**, utilizzabile in tutti i casi si desideri un allineare elementi in righe e colonne, usando solo i *floats*.
- Questo tipo di costruzione è alla base dei layout **responsive**, perché si adatta alle dimensioni del browser. E' anche possibile **fluidificare** questo tipo di griglie, per ottenere effetti molto complessi.
- Sorgenti:
  - La base di questo tipo di layout è il 960 grid system (<http://960.gs/>) che tuttavia è fixed (larghezza 960 pixel) e non usa media queries.
  - La parte con media queries è tratta da Skeleton (<http://www.getskeleton.com/>), che tuttavia rimane fixed per ampiezze superiori a 960 pixel.
  - Infine, l'ispirazione per la versione fluida della griglia è tratta da <http://www.designinfluences.com/fluid960gs/>.
  - Si veda anche l'articolo introduttivo sui *fluid grids* di Ethan Marcotte: <http://alistapart.com/article/fluidgrids>.

# Un Layout Tabulare con i Floats

## Le righe

- Il design permette di disporre un numero massimo di **16 colonne su ogni riga**.
- Le celle di ciascuna riga hanno tra loro una **spaziatura dell'1%**.
- Le classi *container* e *row* sono, rispettivamente, i contenitori dell'intero layout e delle singole righe:
  - `.container { position: relative; width: 98%; padding: 0; }`
    - Il contenitore per la tabella ha un piccolo margine extra. Non è necessario ed è possibile inserire direttamente le righe
  - `.row { margin-bottom: 10px; }`
    - Le righe hanno una leggera spaziatura, che può essere rimossa

# Un Layout Tabulare con i Floats

## Le colonne

- La classe *column* definisce le proprietà comuni delle colonne:
  - `.container .column, .container .columns { float: left; display: inline; margin-left: 1%; margin-right: 1%; }`
- Le colonne *alpha* e *omega* (prima e ultima) non hanno spaziatura sinistra e destra:
  - `.column.alpha, .columns.alpha { margin-left: 0; }`  
`.column.omega, .columns.omega { margin-right: 0; }`
- Le classi *one*, *two*, ecc. permettono di definire l'ampiezza di ciascuna cella come numero di colonne occupate (1-16):
  - `.container .one.column { width: 4.25%; }`  
`.container .two.columns { width: 10.5%; }`
  - Le dimensioni per le celle da tre a sedici colonne sono rispettivamente: 16.75%, 23%, 29.25%, 35.5%, 41.75%, 48%, 54.25%, 60.5%, 66.75%, 73%, 79.25%, 85.5%, 91.75%, 98%



# Un Layout Tabulare con i Floats

## Incapsulamento dei floats

- Per concludere, qualche trucco per assicurarsi che le colonne float rimangano all'interno della propria riga, su ogni browser:
  - `.row:before, .row:after { content: '\0020'; display: block; overflow: hidden; visibility: hidden; width: 0; height: 0; }`
  - `.row:after { clear: both; }`
  - `.row { zoom: 1; }`



# Un Layout Tabulare con i Floats

## Esempio

```
<div class="row">
  <div class="three columns">
    A
  </div>
  <div class="thirteen columns">
    B
  </div>
</div>
<div class="row">
  <div class="three columns">
    C
  </div>
  <div class="thirteen columns">
    <div class="row">
      <div class="eight columns">
        D
      </div>
      <div class="eight columns">
        E
      </div>
    </div>
  </div>
</div>
```

A	B	
C	D	E

# Un Layout Tabulare con i Floats

## Media queries

- Ecco come il layout fluido può *diventare lineare* al di sotto di una certa ampiezza. Disabilitando il *floating* e non costringendo l'ampiezza, le colonne “andranno a capo” ponendosi una sotto l'altra!
- In più, in quest'esempio “blocchiamo” la dimensione dell'intero layout per impedire che scenda al di sotto di una certa soglia minima.

```
@media only screen and (max-width: 767px) {  
  .container { width: 300px; }  
  .container .columns, .container .column { margin: 0; }  
  .container .one.column, .container .one.columns, ... {  
    float: none; width: auto; }  
}
```



# Contenuto Dinamico

## Prefissi, Suffissi, Virgolette e Contatori

- **quotes**
  - **Valori:** none | (stringa stringa)+
  - Indica le virgolette (aperta e chiusa) da usare per questo elemento, se necessario. E' possibile specificare più coppie, una per ciascun livello di nidificazione delle virgolette stesse.
- **counter-reset**
  - **Valori:** (stringa [intero])+
  - Azzera (o imposta al numero dato) il valore dei contatori indicati, relativi all'elemento.
- **counter-increment**
  - **Valori:** (stringa [intero])+
  - Incrementa di uno (o del numero dato) il valore dei contatori indicati, relativi all'elemento.
- **content**
  - **Valori:** none | (stringa | counter(C,S) | open-quote | close-quote)+
  - Si applica ai soli pseudo elementi :before e :after, e specifica il testo che andrà inserito rispettivamente prima o dopo un elemento. I valori di open-quote e close-quote sono quelli impostati dall'attributo quotes. C può essere il nome di un qualunque contatore visibile dall'elemento. S (opzionale) è uno dei valori definiti per la proprietà list-style-type.

# Altre Proprietà

## ■ cursor

- **Valori:** [*uri*, ]\* (auto | crosshair | default | pointer | move | e-resize | ne-resize | nw-resize | n-resize | se-resize | sw-resize | s-resize | w-resize | text | wait | help | progress)
- Imposta la forma del mouse quando è sopra l'elemento.
- La lista di uri, opzionale, indica una o più risorse esterne da usare come cursore. Il browser utilizzerà la prima di cui riconosce il formato.
- In ogni caso, è necessario fornire anche uno dei cursori standard, come unico valore o come ultima scelta nella lista.



# Compatibilità Cross-Browser

## Stylesheet di Reset

- Molti browser applicano alle proprietà dei vari elementi **default diversi**, corrispondenti al loro *stylesheet di default*.
- Per realizzare CSS crossbrowser, può essere utile **azzerare queste differenze** e creare i fogli di stile a partire da una **base minima comune**. Questo si può ottenere inserendo all'inizio del foglio di stile una specifica come la seguente.

(<http://meyerweb.com/eric/tools/css/reset/>)

```
html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre, a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp, small, strike, strong, sub, sup, tt, var, b, u, i, center, dl, dt, dd, ol, ul, li, fieldset, form,
label, legend, table, caption, tbody, tfoot, thead, tr, th, td, article, aside, canvas, details, embed, figure, figcaption, footer,
header, hgroup, menu, nav, output, ruby, section, summary, time, mark, audio, video
{ margin: 0; padding: 0; border: 0; font-size: 100%; font: inherit; vertical-align: baseline; }
article, aside, details, figcaption, figure, footer, header, hgroup, menu, nav, section /* HTML5 */
{ display: block; }
body
{ line-height: 1; }
ol, ul
{ list-style: none; }
blockquote, q
{ quotes: none; }
blockquote:before, blockquote:after, q:before, q:after
{ content: ""; content: none; }
table
{ border-collapse: collapse; border-spacing: 0; }
```