

CSS

Rev. 4.6 del 16/11/2020

CSS

Introduzione ai Fogli di Stile	2
CSS property 2.0 : Color, Font,	4
Le proprietà relative ai Font	4
Le proprietà relative ai Testi	4
Le proprietà relative ai Bordi	5
Le proprietà relative allo Sfondo	5
Le proprietà relative a Margin e Padding	6
Regole base di applicazione dei selettori	7
Personalizzare l'aspetto dei link: le pseudoclassi	7
Elementi Block ed Elementi Inline	8
display:inline-block	9
display:grid	9
La proprietà Overflow	10
Le proprietà ListStyle e Cursor	11
Le proprietà Position e Z-Index	12
La proprietà Float	13
Le proprietà display, visibility, opacity	14
Considerazioni sulle unità di misura	15
Approfondimenti su selettori e pseudoselettori	16
Esempi di Effetti realizzabili tramite CSS	19

I Fogli di Stile - Cascading Style Sheets

HTML non consente approfondite formattazioni sui caratteri e sulla grafica. Ad esempio il tag FONT consente di definire alcuni attributi principali sul tipo di font da utilizzare senza tuttavia consentire caratteri di ampie dimensioni o con uno specifico colore di sfondo. Anziché incrementare il numero di attributi dei vari TAG, da HTML 4.0 in avanti si è preferito seguire un'altra strada, introducendo un attributo **STYLE**, che consente di definire delle **Proprietà di Stile** che sono **ortogonali rispetto a tutti i TAG HTML**, cioè che possono essere applicati indistintamente a qualsiasi TAG HTML, sempre allo stesso modo e con gli stessi valori.

Le **proprietà di STILE** negli ultimi anni sono state molto potenziate rispetto agli attributi HTML e costituiscono una immensa risorsa per la personalizzazione grafica dei siti. Le proprietà di stile vengono di solito scritte in un file esterno rispetto al file HTML, (in un file con estensione .CSS) **consentendo in questo modo una netta separazione fra i contenuti (scritti nel file HTML) e la formattazione grafica (impostata nel file .CSS).**

Le proprietà di stile possono essere scritte in tre posizioni differenti:

(1) direttamente all'interno dei tag HTML (stili INLINE)

In questo caso le proprietà di stile sono introdotte dall'attributo **STYLE** e sono inserite all'interno di una UNICA stringa **nel tipico formato NOME: VALORE**, col punto e virgola come separatore. Il punto e virgola dopo l'ultima voce è facoltativo. Es:

```
<p style = "color:red; font-size:30pt;" > Salve a Tutti </p>
```

(2) Nell'intestazione della pagina (stili INCORPORATI nella pagina)

```
<head>
  <STYLE>
    P { color: green; font-family: verdana,helvetica,sans-serif; }
  </STYLE>
</head>
<body>
  <P> Questo viene scritto in helvetica di colore verde </P>
</body>
```

Molto più compatto e dunque molto più leggibile, ma occorre ripetere gli stili in ogni pagina.

(3) In un apposito File Esterno (Fogli di stile ESTERNI memorizzati in un file .css)

Consente una reale parametrizzazione dello stile, visibile in tutte le pagine HTML dell'intero sito. Le varie pagine HTML, nella sezione HEAD, dovranno richiamare il file CSS mediante il tag **LINK** :

```
<head>
  <link rel="stylesheet" href="mioStile.css" >
</head>
```

All'interno del file .CSS il TAG **STYLE** può essere **OMESSO**, iniziando subito a scrivere gli stili:

```
/* File CSS */
P {color: blue; font-family: "times new roman", times, serif; }
```

Uno stesso tag può essere ridefinito sia su un file .CSS esterno, sia nell'intestazione della pagina, sia direttamente nel TAG (InLine). In tal caso **l'ultima definizione nasconde le precedenti.**

Cioè gli stili inline sovrascrivono sia quelli della head sia quelli del file esterno.

Le definizioni scritte nella head oppure in un file esterno sono equivalenti. Prevalgono quelle scritte dopo.

Cioè se richiamo il file esterno dopo gli stili definiti all'interno di <style>, gli stili del file esterno prevalgono su quelli definiti dentro <style>. In caso contrario prevalgono quelli definiti dentro <style>.

I SELETTORI di stile

Le proprietà di stile possono essere definite mediante tre diversi SELETTORI:

- **SELETTORI di TAG**, cioè associati a tutti i tag di quel tipo. Ad esempio: `p {color:red}`
- **SELETTORI di CLASSE** cioè associati a tutti gli elementi che implementano una certa classe
- **SELETTORI di ELEMENTO** (o selettori assoluti) cioè associati ad un **UNICO** TAG HTML identificato tramite un apposito ID

I Selettori di classe

Consentono di creare tante "**classi**" differenti di uno stesso tag, associabili a istanze diverse dello stesso tag.

I selettori di classe potranno essere associati alle varie istanze del tag mediante l'utilizzo dell'attributo **CLASS**

Esempio:

```
p.titolo { color : red;   font-size : 30pt}
p.sottotitolo { color : RGB(125,125,255);   font-size : 20pt}
```

NE: `p.titolo` **SENZA SPAZI !**

```
<p class="titolo"> Questo è un titolo </p>
<p class="sottotitolo"> Questo è un sottotitolo </p>
```

E' anche possibile definire un **selettore di classe generico**, cioè **senza anteporre davanti il nome di nessun tag**, ma semplicemente cominciando a **scrivere direttamente con il puntino**. Si crea così un selettore generico che **potrà essere associato a qualsiasi tag HTML**. Esempio:

```
.nuovoStile { font-size:24pt; color:red; }
.nuovoStileBold { font-size:24pt; color:red; font-weight:bold;}
```

```
<p class="nuovoStile"> Questo è il nuovo stile </p>
<p class="nuovoStileBold"> Questo è il nuovo stile versione BOLD </p>
```

I Selettori di Elemento (selettori assoluti)

Dato un tag avente un identificativo **ID univoco**, il Selettore di Elemento consente di **associare uno stile ad un singolo elemento della pagina**. Il selettore di Elemento è rappresentato con un **simbolo # (pound)** anteposto al nome del selettore.

`#BOX { }` definisce lo stile dell'elemento HTML avente ID="BOX"

```
<div id="BOX">
  <a href="Pagina2.html"> Questo link è scritto con stile BOX </a>
</div>
```

Annidamento dei selettori

Nella scrittura di un selettore CSS, **lo spazio ha il significato di "all'interno di"**

`#div1 a { color:red }` Tutti i tag `<a>` interni al tag avente `id="div1"` avranno colore rosso

`.class1 a { color:green }` Tutti i tag `<a>` interni ai tag aventi `class="class1"` avranno colore verde

Pseudoselettori

All'interno del selettore di classe, oltre agli stili del tag, è possibile utilizzare anche alcuni **pseudoselettori CSS** accodati a selettore principale tramite i **DUE PUNTI** senza spazi

```
.class1: hover{ color : red; }
```

```
<div class="class1"> Quando il mouse passa su questa scritta, il testo diventa rosso</div>
```

Analisi delle Property CSS 2.0

I campi **COLOR** possono essere espressi mediante i seguenti formati:

16 colori della palette VGA: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white yellow

RGB (r, g, b) Le componenti RGB sono questa volta espresse mediante numeri interi **decimali** compresi tra 0 e 255.

#RRGGBB Colore espresso mediante le componenti esadecimali R G B

RGBa (r, g, b, a) Solo CSS3. alfa indica la trasparenza ed è un numero decimale compreso tra **0=trasparente** e **1=solido**.

#RRGGBBAA Colore espresso mediante le componenti esadecimali R G B + canale alfa espresso in esadecimale (**FF=solido**)

Le proprietà relative ai Font

Font-Family	[[<family-name> <generic-family>],] Nomi dei font separati da virgola. Se il browser dispone del primo font, utilizza quello, altrimenti passa al secondo e così via. Si mettono davanti i Font più specifici ed in coda quelli più generici. Non c'è valore di default, che dipende dalle impostazioni del Browser. Esempi tipici windows e macOS: Font-Family: Arial, Helvetica, sans-serif ; con la virgola come separatore Font-Family: "Times New Roman", Times, serif ; Notare le virgolette sui nomi con spazio
Font-Size	font-size: <absolute-size> <relative-size> <length> <percentage> dimensione del font in <i>punti</i> , in pixel, pollici, centimetri, punti web, etc <absolute-size> sono i 7 valori HTML: xx-small x-small small medium large x-large xx-large <relative-size> può essere larger smaller. <length> valore del Font Size in pt o px <percentage> percentuale rispetto al valore di font-size ereditato dal genitore.
Font-weight	<normal> bold 100 200 300 400 500 600 700 800 900 > bold = 700 normal= 400 I valori numerici sono utilizzabili SOLO SE disponibili nel font utilizzato (ad esempio sono disponibili in Arial ma NON in Verdana)
Font-style	<normal> <i>italic</i> <i>oblique</i> > (oblique era più inclinato dell'italic. Oggi sembrano uguali).
Font-variant	<normal> SMALL-CAPS > SMALL CAPS SIGNIFICA CHE ANCHE LE MINUSCOLE SONO SCRITTE IN MAIUSCOLO, MA SONO LEGGERMENTE PIU' PICCOLE.
Font	Consente di specificare tutte le proprietà relative al Font in un'unica dichiarazione Es font: 30pt bold italic Oggi non sembra più essere supportata

Le proprietà relative ai Testi

Color	< color > di primo piano.
Text-Align	< left right center justify >. Applicabile soltanto agli elementi di tipo BLOCK L'impostazione viene applicata ai tag interni, ma SOLO per quelli di tipo INLINE
Vertical-Align	< baseline sub ^{super} top text-top middle bottom text-bottom <percentage> > Applicabile soltanto per i tag INLINE <percentage> è riferito al valore di line-height <ul style="list-style-type: none"> • baseline (align baselines of element and parent) • middle (align vertical midpoint of element with baseline) • top (align top of element with tallest element on the line) • bottom (align bottom of element with lowest element on the line) • sub (subscript) super (superscript) • text-top (align tops of element and parent's font) relative positioning • text-bottom (align bottoms of element and parent's font) relative positioning
Line-Height	<normal> <number> <length> <percentage> > Definisce l'altezza di riga, Normal significa Line-Height = Font-Size , cioè la riga avrà una altezza pari a Font-Size. Eventuali bordi vengono tracciati sul perimetro del testo. Impostando un valore inferiore rispetto al Font-size, eventuali bordi avrebbero interferenza con il testo percentage è riferito al Font-Size, number è un numero puro (senza unità di misura e senza %), che produce una altezza pari a N * Font-Size.
Text-Decoration	< none [underline overline line-through blink] >

Word-Spacing	<code><normal></code> <code><length></code> Necessaria unità di misura Definisce una spaziatura aggiuntiva fra le parole. Sono ammessi valori negativi.
Letter-Spacing	<code><normal></code> <code><length></code> Necessaria unità di misura Definisce una spaziatura aggiuntiva fra i singoli caratteri. Sono ammessi valori negativi.
Text-Indent	Indentazione della prima linea di testo. Applicabile soltanto ai Block Elements <code><length></code> <code><percentage></code> Default 0. <code><percentage></code> è riferito al valore di Width
Text-Transform	<code><none></code> <code>capitalize</code> <code>uppercase</code> <code>lowercase</code> <ul style="list-style-type: none"> • capitalize (Converte in maiuscolo il primo carattere di ogni parola) • <code>uppercase</code> (Converte in maiuscolo l'intera parola) • <code>lowercase</code> (Converte in minuscolo l'intera parola)

Note sulla proprietà **Vertical-Align**

Questa proprietà vale esclusivamente per i **tag inline** (ed inline-block) e definisce l'allineamento verticale dell'elemento rispetto alla riga testuale in cui l'oggetto è inserito. E' analogo all'attributo html **ALIGN** ed ha come valore di default **BOTTOM**. Cioè se si inserisce ad esempio una immagine all'interno di una riga testuale, per default l'immagine sarà allineata BOTTOM e quindi espansa verso l'alto

I BORDI di un contenitore

Border-Style	[<code>none</code> <code>dotted</code> <code>dashed</code> <code>solid</code> <code>double</code> <code>groove</code> <code>ridge</code> <code>inset</code> <code>outset</code>] Stile dei Bordi. Si possono specificare fino a quattro valori, nel qual caso il primo valore è riferito al Top Border, il secondo al Right-Border, Bottom Border e Left Border E' possibile utilizzare le proprietà singole Border-Top –Style, etc su ciascun bordo
Border-Width (solo dopo aver impostato Border-Style)	Consente di specificare tutti quattro i bordi nel seguente ordine: top, right, bottom, left Specificando un solo parametro, il valore viene applicato a tutti quattro i bordi. Specificando 2 parametri, il 1° valore viene applicato a top/bottom, il 2° a left/right
Border-Color (solo dopo aver impostato Border-Style)	<code><color></code> Colore dei bordi. Per default viene assunto il valore della Proprietà COLOR. Può contenere 1 – 2 – 4 valori nel seguente ordine: top, right, bottom, left Specificando un solo colore, il valore viene applicato a tutti quattro i bordi. E' possibile utilizzare le proprietà singole Border-Top –Color, etc su ciascun bordo

Ognuna delle proprietà precedenti può essere applicata a ciascun singolo bordo:

Border-Top-Width	<code><thin></code> <code>medium</code> <code>thick</code> <code><length></code> > Spessore del bordo superiore. Solo valori positivi.
Border-Left-Width	Spessore del bordo sinistro
Border-Right-Width	Spessore del bordo destro
Border-Bottom-Width	Spessore del bordo inferiore

Border	Consente di impostare in un sol colpo width, style e color di tutti quattro i bordi. Es <code>p { border : 1px solid black }</code>
---------------	----------------------------------------------------------------------------------------------------------------------------------------

E' anche possibile esprimere le 3 proprietà iniziali (style width e color) per ogni singolo bordo:

Border-Top	<code><border-top-width></code> <code><border-style></code> <code><color></code> Consente di impostare in un sol colpo width, style e color del Top Border
Border-Left	Consente di impostare in un sol colpo width, style e color del Left Border
Border-Right	Consente di impostare in un sol colpo width, style e color del Right Border
Border-Bottom	Consente di impostare in un sol colpo width, style e color del Bottom Border

Lo SFONDO di un contenitore

La proprietà Background consente di impostare lo **sfondo di un Tag** (cioè il colore e/o un'eventuale immagine di sfondo). Applicabile al body e a tutti i tag di tipo BLOCK. **Tutti i tag hanno per default sfondo trasparente**

Background-Color	<code><color></code> transparent > Colore di sfondo dell'elemento (nel caso l'img non venga caricata)
-------------------------	----------------------------------------------------------------------------------------------------------------------

Background-Image	< url(img/sfondo.gif) none initial (valore originario) > Immagine di sfondo. Se il nome del file contiene degli spazi occorrono gli apici : url ('mio file.jpg') ; E' buona abitudine usare SEMPRE gli apici all'interno della URL
Background-Repeat	< repeat repeat-x repeat-y no-repeat > repeat = ripete l'immagine di sfondo sia orizzontale sia verticale riempiendo la pagina. Default repeat-x = ripete l'immagine di sfondo soltanto in orizzontale (tile orizzontale) repeat-y = ripete l'immagine di sfondo soltanto in verticale (tile verticale)
Background-Position	Specifica l'allineamento dell'immagine di sfondo rispetto al contenitore. Il 1° valore indica il posizionamento orizzontale, il 2° indica il posizionamento verticale Valori possibili: LEFT/CENTER/RIGHT, TOP/CENTER/BOTTOM Ad esempio background-position: RIGHT BOTTOM, Se si specifica un solo valore si intende orizzontale, ed il verticale è assunto = center Per default l'allineamento è LEFT TOP (0%, 0%)
Background-Attachment	< scroll fixed > Con scroll l'immagine di sfondo scorre insieme al contenuto della pagina Con fixed l'immagine di sfondo rimane fissa sul video anche quando si fa scorrere il contenuto
Background	Consente di specificare tutte le proprietà di Background in un'unica dichiarazione

NB Il path dell'immagine deve essere sempre definito a partire dalla cartella nella quale si trova il file .css

NB: L'immagine viene SEMPRE visualizzata a partire dallo spigolo in alto a sinistra. Se il contenitore è molto più piccolo dell'immagine verrà visualizzata solo una piccola porzione di immagine (appunto la porzione in alto a sinistra). Volendo visualizzare una porzione diversa di immagine, è possibile applicare a background-position dei valori negativi che sostanzialmente traslano l'immagine verso sinistra e verso l'alto.

Ad esempio background-position: -50px -50px visualizza l'immagine a partire dalle coordinate (50,50)

Background-size	auto (default) se l'immagine è più grande del contenitore viene visualizzata solo una porzione in alto a sinistra cover: l'immagine viene ridotta in modo da ricoprire interamente il contenitore contain: l'immagine viene ridotta in modo da essere completamente visualizzata cover e contain differiscono solo se il contenitore ha un aspect ratio differente rispetto all'immagine
------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

MARGIN e PADDING. Area di occupazione degli oggetti

Margin rappresenta il margine **esterno** di un elemento. Consente di distanziare / avvicinare un elemento rispetto agli elementi vicini. Applicabile soltanto ai tag di tipo BLOCK, cioè ai contenitori.

Margin-Top Margin-Left / Right Margin-Bottom	<length> <percentage> < auto > Sono ammessi valori negativi. <percentage> è riferito alle dimensioni del genitore. Margini di elementi consecutivi si sovrappongono
Margin	Consente di specificare tutti quattro i margini nel seguente ordine: top, right, bottom, left Specificando un solo parametro, il valore viene applicato a tutti quattro i margini. Specificando 2 parametri, il primo valore viene applicato a top/bottom, il secondo a left/right Specificando tre parametri il significato è il seguente: top, right/left, bottom E' possibile utilizzare su left/right il valore AUTO per eseguire la centratura orizzontale

Padding rappresenta invece lo "spazio interno" tra il bordo e il contenuto del tag.

Padding-Top Padding-Left / Right Padding-Bottom	Distanza tra il contenuto dell'elemento ed il bordo superiore. Accetta soltanto valori positivi. <percentage> è riferito alle dimensioni del genitore. NON è CONSENTITO usare AUTO con PADDING
Padding	Consente di specificare tutti quattro i padding esattamente come per margin.

Il padding può anche essere utilizzato per impostare l'indentazione del tag `blockquote`.

```
blockquote { padding-left: 10px; }
```

Regole base di applicazione dei selettori

- I vari selettori di stile possono far riferimento a tag istanziati uno dentro l'altro.
A tale scopo si utilizza l'operatore **SPAZIO**
#liv1 p a significa **tag a** inserito all'interno di un **tag p** inserito all'interno del **tag #liv1**
#box .red significa elementi interni al tag **#box** che implementano la classe **red**
#box p.red significa elementi di tipo **p** interni al tag **#box** che implementano la classe **red**
div.class1 div.class2 significa **tag div** con stile **class2** definiti all'interno di un **tag div** con stile **class1**
- Anche davanti al selettore assoluto (come per le classi) si può facoltativamente inserire il tipo del tag in cui il selettore assoluto è stato definito: **div#liv1 label**
- Le regole di stile possono essere assegnate contemporaneamente a più tag / classi.
A tal fine occorre separare i tag mediante l'operatore **VIRGOLA**
div, p, .riga { } significa applicare lo stile a tutti i tag **div**, **p** e alla classe **.riga**
- Le varie regole possono essere arbitrariamente combinate insieme.
div#liv1 label, div#liv2 label, div#liv2 p a, .riga { }
- Uh qualsiasi elemento può implementare contemporaneamente più classi**
semplicemente separandole mediante uno spazio. Es **<div class="box graphics">**
In tal caso NON conta l'ordina con cui sono richiamate le classi, ma l'ORDINE con cui le classi sono scritte all'interno del file.CSS
- Non è consentito assegnare agli ID dei nomi numerici

Livelli di priorità

Gli stili vengono normalmente assegnati in cascata. A parità di importanza il successivo copre il precedente

- Il selettori di elemento (assoluto) è prioritario rispetto agli altri due selettori,**
cioè prevale anche se si viene scritto prima
- Il selettori di classe è prioritario rispetto al selettore di tag**
cioè prevale anche se si viene scritto prima
- I selettori con path più profondo sono prioritari rispetto a quelli più generici**
#box1 div {color:red}
div {color:green}
I tag **div** interni a **box1** avranno colore **red**

Personalizzare l'aspetto dei Link: Le pseudoclassi

Oltre che ai tag, le proprietà di stile possono essere applicate anche ad alcuni **attributi** dei tag.

Nei CSS 2 sono state infatti definite alcune pseudoclassi (principalmente relative al tag **a**) introdotte tramite i due punti :

a:hover stile automaticamente applicato in corrispondenza del mouse-over ed automaticamente rimosso in corrispondenza del mouse-out. Es **a:hover { color : yellow; }**

a:link stile del collegamento ipertestuale nello stato di riposo

a:visited stile del collegamento ipertestuale dopo essere stato 'visitato'

a:active stile del collegamento ipertestuale nel momento in cui viene cliccato

Poiché HTML non ha memoria, non esiste un **a:current**, che può essere gestito solo attraverso JavaScript.

:focus generico (cioè applicabile a più tag) indica la presenza del focus. Utile per modificare lo stile di un campo di input nel momento in cui riceve il focus. Per quanto concerne il **tag a**, l'attributo **focus** in condizioni normali non è quasi mai attivo in quanto, quando l'elemento riceve il focus dopo essere stato cliccato, di solito viene richiamata un'altra pagina o un'altra sezione della stessa pagina, per cui l'elemento perde immediatamente il focus.

L'unico caso in cui l'elemento mantiene il focus è il quando il collegamento contiene un link alla **default route** **href=""**, link che in realtà non viene eseguito, per cui il fuoco rimane sull'elemento.

Elementi BLOCK ed Elementi INLINE: la proprietà display

I tag **block** (**DIV**, **P**, **H**, **LI**, **TABLE**, etc) sono in genere **contenitori preposti a contenere testo o altri oggetti**

- riconoscono le proprietà **width height**. Hanno come valore di default di **width** l'intera larghezza della pagina (o del genitore) e come valore di default di **height** il valor **auto**, cioè l'altezza necessaria a contenere il testo o gli elementi interni. Il valore percentuale (**Width:50%**) è riferito rispetto alle dimensioni del genitore
- Vengono posizionati per default **uno sotto l'altro** (uno per riga: position = static).
- Anche modificando **width** continuano comunque ad 'impegnare' l'intera riga e non accettano altri oggetti sulla stessa riga (salvo utilizzo della proprietà **float**)
- riconoscono **Text-Align**, ma non **Vertical-Align**
- Come valore di **width** si può utilizzare **fit-content** che imposta una larghezza pari al contenuto
- Al posto di **width** si utilizza talvolta **max-width** che presenta un grande vantaggio: stringendo la finestra, nel caso di **width** il testo eccedente non risulta più visibile, mentre nel caso di **max-width** la larghezza dell'oggetto viene automaticamente ridotta ed il testo riscalato di conseguenza. Maggiore responsività

I tag **inline** (**a**, **span**, **label**, **b**, **i**, **u**) **vengono visualizzati in linea con il testo circostante.**

- possono essere inseriti all'interno di una riga come il normale testo
- non riconoscono le proprietà **Width Height** (che assumono il valore **auto**, pari al contenuto dell'oggetto)
- Sono normalmente riconosciute le proprietà **background-color** e tutte le proprietà relative al font.
- non riconoscono **Text-Align** (che non è significativa). ma riconoscono **Vertical-Align**.
- Tra un tag e l'altro c'è sempre un carattere ** ** derivante dalla pagina html (a meno di tag "attaccati")
- riconoscono i **margini** laterali ma non riconoscono **Margin-Top** e **Margin-Bottom**
- Viene riconosciuta la proprietà **Padding**. La larghezza complessiva del tag viene calcolata come Lunghezza del testo + **PaddingLeft** + **PaddingRight** + Bordi, mentre l'altezza complessiva viene anch'essa calcolata come Altezza del testo + **PaddingTop** + **PaddingBottom** + Bordi. **Padding-Top** e **Padding-Bottom** sono riconosciuti, però NON consentono di spaziare l'elemento inline rispetto al testo antecedente o successivo, che deborda all'interno dell'eventuale padding dell'elemento corrente.

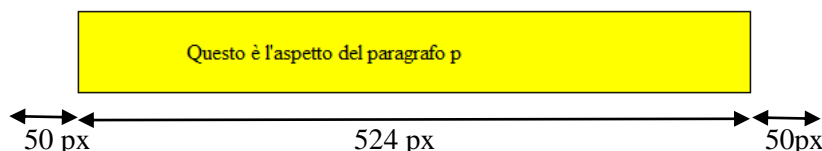
Su qualsiasi tag è possibile modificare il valore di DISPLAY da :INLINE a :BLOCK e viceversa.

Occupazione complessiva di un BLOCK TAG

I valori di **width** e **height** **NON rappresentano la reale dimensione del box, ma l'area utile interna.** L'occupazione complessiva è ottenuta sommando l'eventuale dimensione del **padding** e del **bordo**.

Cioè il **padding** contribuisce ad aumentare le dimensioni dell'oggetto. Esempio:

```
p {
    width:400px;
    height:30px;
    padding: 20px 40px 20px 80px;
    border: 2px solid black;
    margin: 50px; }
```



La **Width** complessiva risulta **400 + 40 (right padding) + 80 (left padding) + 4 (bordi) = 524 px.**

La stessa cosa vale per **Height** che rappresenta l'altezza interna del contenitore. Al valore di Height, occorre poi sommare **PaddingTop**, **PaddingBottom** e Bordi. Nell'esempio precedente, l'altezza complessiva sarà **PaddingTop + Height + PaddingBottom + Bordi = 20 + 30 + 20 + 4 = 74px.**

Se non si specifica il valore di height, il tag **DIV** lascia esattamente 20px di padding sia superiore che inferiore, per cui il testo risulterà perfettamente centrato in verticale. Se invece si specifica il valore di height, per avere allineamento verticale height dovrebbe avere lo stesso valore dell'occupazione verticale del testo.

Margin definisce invece il margine esterno all'oggetto al di là del bordo. Non interviene nel computo delle dimensioni dell'oggetto. **I margini di elementi consecutivi si sovrappongono.**

- Negli elementi floating i margini si sommano regolarmente
- Se un elemento si trova all'interno di un altro elemento, il margin dell'elemento interno si **somma** al padding dell'elemento esterno.

Centrata verticale del testo all'interno di un Block Tag

La proprietà **Line Height** indica l'interlinea, cioè l'occupazione verticale della singola riga. Il default è **Line-Height = Font-Size**, cioè la riga di default ha una altezza pari a Font-Size. Eventuali bordi vengono tracciati sul perimetro del testo. Impostando un valore inferiore rispetto al Font-size, eventuali bordi andrebbero ad interferire con il testo.

Se il contenitore contiene una sola riga, il modo più semplice per eseguire la centratura verticale è quella di utilizzare la proprietà **line-height**, impostando il valore di **line-height** allo stesso valore di **height**.

Nel caso invece di righe multiple, non si può ovviamente agire su **Line Height**.

In questo caso la soluzione migliore è quella di **omettere height** (**height: auto**) e agire sul Padding.

Nota

- Se Height è inferiore rispetto all'occupazione verticale del testo, il testo deborderà al di sotto dell'area occupata dal paragrafo, andando eventualmente ad interferire con gli elementi successivi.
- Se Height è superiore rispetto all'occupazione verticale del testo la scritta risulterà allineata verso l'alto, con una eccedenza di pixel prima del Padding Bottom.

Il valore display:inline-block

Il valore **display:inline-block** fa sì che il tag :

- pur essendo inline, riconosca le proprietà **width** e **height**, per cui rappresenta una ottima alternativa a **float**
- Nel caso di un tag contenitore, il valore **width:auto** non corrisponde all'intera riga, ma al valore del contenuto (come per i tag inline e come anche nel caso di **float**)
- Come tutti i tag inline riconosce **vertical-align** che rappresenta l'allineamento verticale rispetto ad una ipotetica linea di testo, con l'oggetto che
 - nel caso di align = bottom, si espande al di sopra della linea del testo,
 - nel caso di align = top, si espande al di sotto della linea del testo.

vertical-align sembra assumere default differenti da tag a tag e addirittura a parità di tag assume valori specifici differenti (di solito espressi in em) tra quello antecedente e quello susseguente. Per avere allineamento perfetto fra due tag inline-block posizionati sulla stessa riga la soluzione migliore è quella di impostare su entrambi la proprietà **vertical-align:middle**

- riconoscono la proprietà **margin** ma non riconoscono il valore auto, per cui non è possibile impostare il valore margin: 0 auto

I tag **img**, **input**, **button**, **textArea**, **select** sono già di default inline-block.

I tag **input** e **button** inoltre presentano di default il testo allineato verticalmente senza bisogno di line-height

Il valore display:grid

Il valore **display:grid** è molto comodo per definire ad esempio delle maschere di inserimento/visualizzazione.

Si supponga di avere un tag DIV centrato nella pagina all'interno del quale si vuole inserire un elenco ordinato di TEXTBOX (uno sotto l'altro) ognuno preceduto da una propria LABEL descrittiva

Sia per le LABEL che per i TEXTBOX si desidera una larghezza di 160px.

Per ogni riga si desidera una altezza di 25px.

CSS

category:	cooking
title:	Everyday Italian
lang:	en
author:	Giada De Laurentis
year:	2005
price:	30.00

```
#wrapper {
  display: grid;
  width: fit-content;           // in alternativa si può impostare width=350px
  width: -moz-fit-content;      // in firefox fit-content deve essere preceduto da -moz
  // se si assegna un valore a width, come 2° colonna si può impostare auto
  grid-template-columns: 160px fit-content;
  grid-template-rows: 25px 25px 25px 25px 25px 25px;
  grid-column-gap: 8px;
  grid-row-gap: 5px;
}
```

Il valore **grid-template-columns**, su una o più colonne, può essere impostato al valore **auto**, nel qual caso la larghezza della colonna si adegua sulla base del contenuto.

Il valore **grid-template-row** deve essere settato per ogni singola riga. Se omesso le righe assumono l'altezza necessaria per visualizzare il contenuto.

Per ottenere l'allineamento delle label a destra si può utilizzare la seguente proprietà che può assumere il valori start/end/center:

```
#wrapper label{
  align-self:center;    /* allineamento verticale */
  justify-self:end;     /* allineamento orizzontale */
}
```

Su una singola cella è possibile impostare la proprietà **grid-column: span 3** per fare in modo che la cella si espanda sulle 3 colonne successive. Oppure **grid-row: span 3**

La proprietà Overflow

Ha senso quando il testo interno **eccede** le dimensioni di un contenitore che deve avere dimensioni fisse e non può estendersi in altezza per contenere tutto il testo.

```
<div style="width:160px; height:80px;
```

I possibili valori che può assumere sono:

- **visible** La porzione eccedente le dimensioni del box viene mostrata eccedendo le dimensioni (default).
- **hidden** La porzione eccedente le dimensioni del box non viene mostrato
- **auto** Se si impostano **Width e Height**, viene visualizzata la barra di scorrimento necessaria (verticale o orizzontale o entrambe) **solo** se il testo eccede le dimensioni del contenitore.
overflow:auto è **anche** utilizzato, sempre su un contenitore, per "sentire" anche gli elementi FLOAT interni (nel qual caso occorre NON impostare width e height).
- **scroll** Indipendentemente dal contenuto, sul contenitore vengono applicate sia la barra di scorrimento verticale che quella orizzontale. La barra di scorrimento orizzontale ha senso se all'interno del contenitore ci sono degli elementi (es immagini) la cui larghezza eccede le dimensioni del box, oppure degli elementi con position:absolute in posizioni che eccedono la larghezza del box.

- Sul **body** è buona regola impostare SEMPRE width:100%, height 100%

Utilizzo della slide bar su un contenitore interno

Impostando su un contenitore una altezza fissa con **overflow:auto** c'è l'inconveniente che il testo, durante il trascinamento della slide bar, tenderebbe a invadere il padding inferiore. Per evitare questo inconveniente una soluzione spesso adottata è quella di utilizzare un box interno impostando sul contenitore interno **overflow:auto**, **margin:0**, **padding:0**, width e height pari a quelle definite per il contenitore esterno (area client). In questo modo il testo non può uscire dal contenitore interno e non potrà quindi invadere il padding del contenitore esterno.

Centrata di un box all'interno di un altro box


















Un'altra soluzione per centrare un box all'interno di un altro box è quella di eseguire un posizionamento assoluto del contenitore interno nell'area client del contenitore esterno:

- sul contenitore esterno la proprietà **POSITION:RELATIVE**.
- sul contenitore interno si posiziona il **punto TopLeft al centro del contenitore esterno** (valore=50%) e poi si imposta un margine superiore pari alla metà della altezza del contenitore interno ed un margine sinistro pari alla metà della larghezza del contenitore interno.

```
#container{
    position:relative;
    width:600px;
    height:200px;
    padding:0px; }

#inner {
    position:absolute;
    width:200px; height:200px;
    top:50%; left:50%;
    margin-top:-100px;
    margin-left:-100px; }
```








La proprietà cursor per cambiare forma al cursore del mouse

		
pointer	default	crosshair
		
text	help	move
		
n-resize	ne-resize	nw-resize
		
s-resize	se-resize	sw-resize
		
e-resize	w-resize	wait
		
progress	auto	url("url")

Esempio:

```
div:hover { cursor: pointer; }
```

Nei CSS3 sono stati introdotti i seguenti cursori:

		
not-supported	not-supported	not-supported
		
copy	alias	cell
		
all-scroll	no-drop	not-allowed
		
col-resize	row-resize	vertical-text

LIST Properties (Elenchi puntati e numerati)

List-Style-Type	disc circle square decimal lower-roman upper-roman lower-alpha upper-alpha none Applicabile a UL e OL . Indica il tipo di marker da utilizzare Esempio: OL { list-style-type: upper-alpha } Impostando margin:0 il List-Style-Type viene automaticamente impostato a none
List-Style-Image	<url> none Indica l'immagine da utilizzare come marker al posto dei puntini. Prioritaria rispetto a List-Style-Type
List-Style-Position	inside outside Inside fa sì che la seconda riga continui sotto il puntino
List-Style	Consente di impostare in un sol colpo le tre proprietà precedenti.

CSS Reset

Quando si progetta un foglio di stile, bisogna tener conto anche degli **stili di default utilizzati in ogni browser**. Purtroppo in questo non esiste uno standard, così i valori iniziali possono variare da browser a browser per cui, in fase di realizzazione del layout, si possono ottenere risultati abbastanza diversi nei vari browser. La soluzione più semplice a questo problema è quella di impostare inizialmente un file di reset che resettì tutti i valori di default. A tale scopo sono stati realizzati diversi fogli di stile chiamati **CSS Reset** e vanno inseriti prima delle altre regole CSS del progetto. Tra i più noti Eric Meyer CSS Reset e Yahoo CSS Reset. Esempio:

```
* { margin: 0; padding: 0; ..... }
```

* rappresenta il cosiddetto **selettore universale** che consente di applicare lo stile a tutti i tag.

La proprietà Position

Può assumere i seguenti valori:

- **static** [default]. L'oggetto segue il normale rendering della pagina, cioè il normale posizionamento HTML
- **absolute** : Impostando **position: absolute** su un elemento, mediante le proprietà **top**, **left**, **right** e **bottom** si può posizionare l'elemento in modo assoluto rispetto alla pagina. In riferimento a queste 4 proprietà, in genere se ne impostano soltanto due (top/left oppure bottom/right) e le altre due vengono desunte automaticamente sulla base del valore di width e height.
Per default **top** e **left** valgono 0, per cui se si imposta **position: absolute** senza impostare top e left, l'elemento verrà posizionato ad inizio pagina nell'angolo in alto a sinistra
- **relative** Impostando **position: relative** su un elemento, i suoi valori di **top**, **left**, **right** e **bottom** non saranno più riferiti alla pagina ma alla **posizione iniziale dell'elemento** (cioè la posizione che avrebbe nel normale rendering della pagina con **position: static**). Ad esempio **top: 10px** sposta l'elemento di 10px in basso rispetto alla posizione che avrebbe con static. Viceversa **bottom: 10px** consente di spostare l'elemento in alto di 10px. Sotto questo aspetto **top: 10px** è equivalente a **bottom: -10px**
- **fixed** Identico ad absolute, ma in più insensibile rispetto allo scroll della pagina

Impostando **position: absolute** sugli elementi interni ad un certo contenitore, se il contenitore presenta un qualunque valore di **Position** diverso da static, (**absolute**, **relative**, **fixed**), il posizionamento degli elementi interni diventa ASSOLUTO rispetto al **contenitore** e non più rispetto alla pagina. In questi casi in genere si imposta sul contenitore il valore **position: relative** senza impostare **top** e **left** in modo che il contenitore segua il normale rendering della pagina consentendo però il posizionamento assoluto degli elementi interni

Note:

- 1) Gli elementi **position: relative** **position: absolute** **position: fixed** non vengono conteggiati nel normale rendering della pagina, per cui più elementi possono anche sovrapporsi.
Se in un punto ci sono già altri oggetti si avrà una sovrapposizione che dovrà essere gestita tramite z-index.
- 2) Impostando **position: relative** **position: absolute** **position: fixed** su un qualsiasi elemento, la proprietà **display** assume automaticamente il valore **inline-block**, a meno che non venga esplicitamente ridefinita
- 3) La proprietà **position: absolute** prevale anche sulla proprietà **float**.
Comunque in genere o si usa uno oppure l'altro.

La proprietà z-index

Accetta un valore numerico compreso tra -2 Miliardi e +2 Miliardi. **Il default è zero**
Introduce, oltre a x e y, un terzo asse relativo alla profondità. Elementi con z-index maggiore vengono visualizzati davanti (**sopra**) rispetto ad elementi con z-index minore. Definito SOLTANTO per oggetti che hanno **position: absolute/relative/fixed**. **Tutti gli oggetti statici hanno per default z-index=0 non modificabile**. Assegnando ad un elemento absolute **z-index=-1** viene visualizzato al di sotto degli elementi statici.

La proprietà FLOAT

La proprietà float consente di rimuovere un elemento dal normale flusso del documento e **ancorarlo su uno dei lati (destra o sinistra) del suo elemento contenitore** in modo simile agli attributi HTML ALIGN=left e ALIGN=right del tag **IMG**. Applicabile sia tag BLOCK sia a che INLINE che in entrambi i casi diventano inline-block. Il contenuto che circonda l'elemento scorrerà intorno ad esso sul lato opposto rispetto a quello indicato come valore di **float**. Gli elementi float non vengono conteggiati nel normale rendering della pagina,

- Un elemento definito FLOAT **non si estende più per tutta la riga, ma solo per la larghezza necessaria per visualizzare il suo contenuto**. L'elemento in sostanza si comporta come un elemento **INLINE-BLOCK** sensibile alle proprietà WIDTH e HEIGHT
- Gli elementi successivi all'elemento float, **se non sono float**, si comportano nel seguente modo:
 - Se l'elemento successivo è un elemento **block**, questo viene **“sovrapposto”** agli elementi float (come se questi non fossero conteggiati nel rendering)
 - L'elemento block non si sovrappone se viene dichiarato **overflow:auto**, nel qual caso andrà ad occupare l'intero spazio orizzontale a sua disposizione, oppure se utilizza la proprietà **CLEAR:BOTH**, che interrompe la fluttuazione posizionando l'elemento sulla prima riga libera dopo gli elementi fluttuanti. La proprietà CLEAR:BOTH sembra essere riconosciuta soltanto dai tag BR. Il valore CLEAR:LEFT interrompe soltanto il float:left e non il float:right
 - Se l'elemento successivo è un elemento **inline** oppure **inline-block**, questo **“sente”** agli elementi fluttuanti e dunque viene posizionato *dopo* di loro.
- Se si desidera affiancare degli elementi, gli elementi float devono essere dichiarati **“prima”** degli elementi non float, in modo che l'elemento finale possa essere dichiarato float:none e overflow:auto ed andare ad occupare tutto lo spazio rimanente fino a fine riga.
- Un'altra tecnica talvolta utilizzata. è quella di inserire tre livelli, **il primo float:left**, il **secondo float:right** ed **il terzo non fluttuante** (con overflow:auto) in modo che vada ad occupare interamente tutta l'area centrale. Questo terzo elemento può non essere dichiarato overflow:auto, ma è sufficiente che abbia un margin-left o un padding-left adeguati in modo da non sovrapporsi al primo elemento float di sinistra. Notare che, **a livello di html**, l'elemento float:right deve essere dichiarato **prima** rispetto all'elemento centrale

Gestione del wrapper esterno

Il contenitore esterno non **“sente”** gli elementi float contenuti al suo interno, a meno che dopo gli elementi float ci siano altri elementi di tipo block con **clear:both**, oppure di tipo inline oppure di tipo inline-block.

Nel caso di un contenitore contenente soltanto elementi fluttuanti occorre implementare una delle seguenti soluzioni:

- Specificare manualmente **width** e **height** del contenitore in modo che riesca a “contenere” gli elementi float
- Dichiarare il contenitore **float** oppure **overflow:auto**. In entrambi i casi l'**altezza** del contenitore si adatta in modo da contenere tutti gli elementi float interni. Per quanto concerne la **larghezza**, nel caso di overflow:auto occupa tutta la riga, mentre nel caso float occupa soltanto lo spazio minimo necessario a contenere gli elementi interni. In entrambi i casi è consigliato specificare manualmente la larghezza desiderata

Limiti nell'utilizzo del valore float su un contenitore:

- Nel caso di contenitore **float** :
 - Non **sente margin:0 auto** per cui il contenitore non può essere centrato nella pagina
 - Occorre impostare float su tutta la catena degli antenati (a meno che non abbiano la loro altezza)
- Nel caso di contenitore **overflow:auto** si ha un problema nel caso del **DRAG & DROP** dove risulta impossibile trascinare gli elementi al di fuori del contenitore, in quanto il contenitore si **“allarga”** introducendo la barra di scorrimento.

Altre Proprietà CSS 2.0

display Impostando il valore **none** l'elemento viene 'rimosso' dalla pagina e gli elementi successivi scorreranno verso l'alto occupando il posto dell'elemento con **display:none**.

Gli altri valori maggiormente utilizzati sono **block**, **inline**, **inline-block**, **grid**

visibility Consente di nascondere / mostrare un elemento. Può assumere i valori **hidden** **visibile** **collapse**. **visibility:hidden**, a differenza di **display:none**, nasconde l'elemento senza rimuovere la sua occupazione spaziale. Rimane un spazio bianco all'interno della pagina. E' come se rendesse l'elemento totalmente trasparente

opacity Numero con la virgola tra **0=Transparent** e **1=solido (default)**. Utile per rendere semitrasparente un oggetto rispetto allo sfondo del contenitore. Rende trasparente **sia** il testo dell'oggetto, **sia** il suo sfondo. A differenza di **visibility:hidden** e **display:block**, un oggetto con **opacity:0** continua a sentire gli eventi (compreso lo pseudo selettore **:hover**). Inoltre, avendo un valore numerico, consente l'utilizzo della proprietà **transition**. E' sostanzialmente equivalente alla funzione **RGBa** applicata sia al testo sia allo sfondo.

rgba() La trasparenza è un numero decimale compreso tra **0=trasparente** e **1=solido** (esattamente come opacity)

```
background-color: rgba(0, 0, 0, 0.50); // oppure
background-color: rgba(0, 0, 0, 50%);
```

Notare che, a differenza di **opacity**, il valore di **RGBa** **NON** viene **ereditato** dagli elementi interni. MEGLIO, perché spesso gli elementi interni **NON** devono essere trasparenti. Inoltre, se il box ha **opacity:50%** e l'elemento interno **opacity:50%**, in realtà l'elemento interno avrà **opacity:25%**. Difficile da gestire.

white-space Può assumere i valori **normal** (default, gli spazi multipli sono compattati in un unico spazio), **nowrap** viene eliminato l'a capo automatico e le righe lunghe debordano oltre il bordo destro del contenitore. **pre** gli spazi multipli rimangono inalterati senza essere compattati, come il tag **PRE** dell'HTML. Le righe lunghe debordano. **pre-wrap** come **pre** però con il **wrap on** (le righe lunghe non debordano).

min-width: larghezza minima dell'elemento, **min-height**: altezza minima dell'elemento;
max-width: larghezza massima dell'elemento, **max-height**: altezza massima dell'elemento.

Queste proprietà sono significative specie in caso di elementi che possono cambiare dimensione come le **textarea**

max-height è molto comodo nel caso in cui si abbia una sequenza di **tag img**, i quali adattano le loro dimensioni in base alle dimensioni delle immagini contenute. Invece di impostare una **height** fissa per tutte le **img**, si può utilizzare **max-height** in modo che le immagini più grandi vengano riscalate mantenendo le proporzioni, mentre le immagini più piccole verranno visualizzate così come sono senza perdere di qualità.

Clip clip: top right bottom left. Esempio: clip:25px 125px 125px 25px
 Crea delle aree di visibilità. Solo ciò che sta dentro l'area viene visualizzato.

Utile per visualizzare porzioni di immagini o per creare aree bianche intorno agli elementi

Il significato del valore inherit

Assegnare ad una Property CSS di un certo elemento il valore **inherit**, significa che quella property in quell elemento deve ereditare il valore della stessa property nell'elemento genitore (contenitore esterno), che peraltro rappresenta il comportamento di default di molti tag (tutti quelli relativi al testo, compreso color). Questo non vale invece, ad esempio, per **padding**, **margin**, **background-color**, **border** che, indipendentemente dal genitore, utilizzano il valore di default per quel tag.

L'impostazione del valore **inherit** ha senso per quelle proprietà che **non** ereditano automaticamente oppure per quelle proprietà il cui valore è stato modificato tramite CSS.

```
body { color:black; font-family:Georgia; }
h2 { color:violet; font-family:Arial; }
#sidebar h2 { color: inherit; font-family: inherit; }
```

h2 all'interno di **sidebar** non sarà violet ma eredita il valore di **sidebar** il quale a sua volta sarà ereditato da **body**, quindi il suo colore sarà **nero**. Idem per **font-family**.

Le seguenti assegnazioni sono invece inutili in quanto, salvo diverse indicazioni, h3 eredita automaticamente da body senza bisogno di scriverlo esplicitamente.

```
h3 { color: inherit; font-family: inherit; }
```

Considerazioni sulle Unità di Misura

I campi Dimensionali (width, height, etc) possono essere espressi mediante le seguenti Unità di Misura:

Absolute

- **px** (pixel) (**1px = 3/4 pt**)
 - **pt** (points; utilizzato solo per il font-size. **1pt=1/72 pollice**)
 - **in** (inches; 1in=2.54cm)
 - **cm, mm, pc** (picas; 1pc=12pt)
- (Attenzione che tra il valore numerico e l'unità di misura **non devono essere lasciati degli spazi**)

Relative

- **%** rappresenta una percentuale **rispetto alla corrispondente proprietà del contenitore**
- **em** simile alla precedente, però **riferita al font-size dell'elemento corrente**

Riepilogo sulle Unità di Misura del font:

punti web	em	px	pt	
1	0,625 em	10 px	7,5 pt	xx-small
2	0,82 em	13 px	10 pt	x-small
3	1 em	16 px	12 pt (default)	small
4	1.13 em	18 px	13,5 pt (sono ammessi i decimali)	medium
5	1.5 em	24 px	18 pt	large
6	2 em	32 px	24 pt	x-large
7	3 em	48 px	36 pt	xx-large

Significato di em e %

% applicato su **width**, indica le dimensioni dell'oggetto rispetto alla **width** del wrapper **contenitore**.

% applicato su **height** NON viene riconosciuto

% applicato su **padding** e **margin**, indica le dimensioni rispetto alla **width** del wrapper **contenitore**.

em applicato a **width height padding** e **margin**, indica le dimensioni **rispetto al font-size** dell'elemento stesso. Impostare **padding: 1em** significa assegnare al padding lo stesso valore del font-size.

Se l'elemento utilizza un font-size maggiore, il padding aumenterà di conseguenza. Idem se il font-size si riduce.

em applicato a **font-size** indica un fattore di moltiplicazione del font-size **rispetto al font-size del contenitore**.

La property **font-size** viene di solito ereditata dal genitore (a differenza di padding, margin, e background-color che assumono il valore di default del tag in cui si trovano). Per cui **font-size: 1.5em** equivale a **font-size: 150%** e significa incrementare il font-size di un 50 % rispetto al font del genitore (arrivando eventualmente fino al valore del tag html che rappresenta la radice del DOM ed ha un font-size di default pari a 16px = 12 pt (3 punti web).

Nel caso dei tag input, hanno un font-size di default pari a 10pt, indipendente dal font-size del genitore.

rem applicato a **font-size** indica un fattore di moltiplicazione del font-size **rispetto al font-size del tag <html>**.

rem applicato a **width height padding** e **margin**, indica le dimensioni **rispetto al font-size del tag <html>**.

Note su tag H e font-size

I tag H anziché ereditare normalmente il font-size del genitore in cui si trovano, “modificano” il valore ereditato applicando un fattore moltiplicativo o di riduzione (2 nel caso di h1, 1,5 nel caso di h2, +1/6 nel caso di h3 come anticipato nel documento HTML quando si sono introdotti i tag H).

Se però il tag H1 anziché trovarsi dentro un normalissimo tag DIV si trova dentro ad uno dei nuovi tag HTML5 (**section article, nav, aside**), il fattore moltiplicativo di H1 diventa solo più 1,5.

Per fare in modo che H1 si comporti sempre allo stesso modo indipendentemente dal tag genitore, si può impostare all'interno del file reset.css la seguente property: **h1 {font-size:2em}**

cioè h1 deve *sempre* raddoppiare il font-size del genitore, indipendentemente dal contenitore in cui si trova.

Approfondimenti su selettori e pseudo selettori

- Il selettore **>** indica Figlio diretto. Il seguente indica i tag p direttamente scritti all'interno dei tag div
div > p
- Il selettore **+** indica il primo Fratello successivo rispetto al tag indicato.
Il seguente esempio indica il primo tag p scritto successivamente al tag div#id1
div#id1 + p
- Il selettore **~** indica un generico fratello successivo ad un certo tag. Il seguente esempio indica tutti i tag p che seguono un tag img all'interno di un medesimo livello
img ~ p
- I selettori scritti in modo più specifico prevalgono su quelli scritti in modo più generico, anche se questi ultimi sono scritti dopo rispetto ai precedenti.

```
section section p {color:red}  
section p {color:blue}
```

tutti gli elementi **p** contenuti all'interno di una **section** a sua volta contenuta all'interno di un'altra **section** avranno colore **rosso** e non **blu**, in quanto il primo CSS è più specifico.

```
<section> <section> <p> lorem ipsum </p> </section> </section>
```

In alternativa si potrebbe scrivere all'interno del secondo CSS:

```
section p {  
    color:blue !important;  
}
```

In questo modo la nuova regola diventa prioritaria rispetto a quelle più specifiche, indipendentemente dalla posizione in cui è scritta.

Accesso agli elementi che implementano più classi

```
.class1.class2 {  
  
}
```

Le due classi vanno scritte senza spazi intermedi

Pseudo Selettori

Sono introdotti dal simbolo **:**
e possono essere utilizzati all'interno di una qualunque espressione CSS.

```
:checked
:enabled
:disabled
:selected (applicato alle singole option di un tag select)
:button
:radio
:submit
:visible
:hidden
```

La seguente sintassi con parentesi quadra vale per **molte** attributi html ma non per tutti:

```
input[type=text]      // Attenzione a NON lasciare spazi davanti alle parentesi quadre !
input[type=radio]
input[type=checkbox]
input[type=button]
input[type=submit]
input[name=txtNome]
input[name=opt1]

input[name=optGenere]:checked
p[style="font-style:italic"]
```

In questo caso il confronto viene fatto con l'**INTERO** attributo **style** impostato dinamicamente

Esempio: applicare uno sfondo grigio su tutti gli elementi disabilitati

```
input[type="text"]:disabled {
    background-color: #dddddd;
}
```

CSS Functions

```
:not(selettore_Secondario) // Es input[type=radio]:not(:checked)
:contains(testo);         // TRUE se l'elemento contiene il testo indicato (anche annidato)
#menu li:has(ul)           // Le voci di menu che contengono un tag UL (anche annidato)
```

PseudoClassi di filtro su un gruppo di elementi

```
:first-child              // Primo figlio generico (indipendentemente dal tipo)
:first-of-type           // Primo elemento del suo tipo
```

:first-child restituisce true se l'elemento corrente gode della proprietà di essere primo figlio (primogenito) del proprio genitore, qualunque esso sia, e indipendentemente dal proprio tipo. Esempio:

```
<div id="wrapper">
  <p> ..... </p>
  <p> <input .....> </p>
  <input .....>
  <p> ..... </p>
</div>
```

Il primo `<input>` presenta `:first-child` uguale a true perché è in assoluto il primo figlio di `<p>`
 Il secondo `<input>` presenta `:first-child` uguale a false perché dentro wrapper, prima di lui, ci sono altri figli.

Il primo `<input>` presenta `:first-of-type` uguale a true perché, rispetto al proprio padre, è il primo elemento di tipo input. Il secondo `<input>` presenta `:first-of-type` uguale a true perché, all'interno di wrapper, è il primo elemento di tipo input

```
:nth-child(i)           // i-esimo figlio generico (indipendentemente dal tipo) (a base 1)
:nth-of-type(i)         // i-esimo elemento del suo tipo (a base 1)

:last-child            // Ultimo figlio generico (indipendentemente dal tipo)
:last-of-type          // Ultimo elemento del suo tipo

:only-child            // Figlio unico
div :nth-child(i)      // i-esimo elemento generico contenuto in un tag div
div :nth-of-type(i)    // i-esimo elemento del suo tipo contenuto in un tag div

p:nth-child(i)         // i-esimo figlio generico, ma solo se di tipo p
p:nth-of-type(i)       // i-esimo elemento del suo tipo, ma solo se di tipo p
Se nth-child non è di tipo p, il selettore restituisce false
```

`:nth-child()` e `:nth-of-type()` **operano SOLO sui tag html e non sulle classi**. Se vengono applicati ad una classe, in realtà è come se venissero applicate al tag che implementa quella classe.

`:nth-child()` e `:nth-of-type()` oltre all'indice i-esimo accettano come parametro anche i valori: **even**: tutti gli elementi pari - **odd**: tutti gli elementi dispari

Gli pseudoselettori `:first` e `:last` accodati a qualunque altro selettore restituiscono rispettivamente il primo e l'ultimo elemento della collezione. Sono analoghi ai metodi jQuery `.first()` e `.last()`

Lo pseudoselettore `::after` consente di aggiungere un testo in coda al contenuto dell'elemento corrente. `::before` è simile ma aggiunge il testo PRIMA del contenuto del tag corrente.

Quello che in CSS2 era `:after`, in CSS3 è diventato `::after` perché, da un punto di vista logico, a differenza degli altri pseudoselettori, non rappresenta un elemento del DOM a cui accedere, ma consente di aggiungere un nuovo contenuto ad un certo elemento. Per cui è stata creata una sintassi differente `::` che però da punto di vista sintattico è equivalente ai :

In pratica è come se aggiungesse un tag `span` fittizio in coda o in cima al contenuto del tag.

Un apposito attributo `content` specifica il testo da aggiungere:

```
p::after {
  content: " - Remember this";
  background-color: yellow;
  color: red;
  font-weight: bold;
}
```

Il testo - **Remember this** sarà aggiunto in coda al contenuto di ogni tag P ed avrà sfondo giallo e testo rosso:

I live in Ducksburg - **Remember this**

NOTA Tutti i vari pseudoselettori anche se usati in forma multipla NON vengono applicati in cascata, cioè **qualunque pseudoselettore viene applicato sempre sul tag iniziale**, e NON su una ipotetica collezione individuata dallo pseudoselettore precedente:

```
input[type=radio]:nth-of-type(5) { }
```

va a prendere il 5° tag `input` della pagina, (indipendentemente dal type dei tag input precedenti) e ne restituisce il puntatore, però solo se questo tag è di tipo radio, altrimenti restituisce `null`.

Esempi di Effetti realizzabili tramite CSS

Allineamento di una immagine rispetto al testo circostante

Quando si ha la necessità di inserire delle immagini all'interno del normale flusso di testo di una pagina web, spesso si hanno alcuni problemi con l'allineamento. Questo accade perché, di default, le immagini hanno la proprietà **float** impostata su **none** e l'immagine viene normalmente inserita **inline** rispetto al testo.

Per allineare un'immagine ai lati del testo è sufficiente agire sulla proprietà **float** impostando per l'immagine una classe di allineamento:

```
.alignleft { float: left; padding: 15px 15px 15px 0; } /* 0 a sinistra */
```

Dopo di che il tag **** può tranquillamente essere inserito in qualunque punto all'interno del flusso di testo e verrà visualizzata lateralmente senza provocare interruzione del flusso.

Trasparenza e Bordo "spaziato"

Per ottenere un bordo spaziato è sufficiente assegnare all'immagine una apposita Proprietà **PADDING**:

```
img.imgStyle1 {  
    opacity: 0.7; /* 1=solido */  
    border: 1px solid #000;  
    padding: 5px;  
}
```

Il padding viene però applicato SOLO se l'immagine NON eccede le dimensioni del contenitore.

Eliminazione del bordo blu intorno alle immagini-link

Una immagine utilizzata come collegamento ipertestuale viene di solito visualizzata con un bordo blu intorno. Questo perché i browser impostano di default il bordo blu sui link. Per eliminare il bordo occorre impostare :

```
img { border: none; }
```

Inserimento di un testo all'interno di una immagine

Per ottenere l'effetto di un testo sovrapposto ad una immagine si può sfruttare la proprietà **background** inserendo l'immagine come sfondo del blocco testuale;

```
<div class="imgStyle2"> Lorem Ipsum</div>
```

```
div.imgStyle2 {  
    width: 400px;  
    height: 300px;  
    background-image: url('img.jpg');  
    background-repeat: no-repeat;  
    text-align: center;  
    line-height: 450px; /*225px superiori rispetto al centro  
scritta*/  
}
```

All'interno del block **DIV** verrà caricata l'immagine di sfondo che deve avere le stesse dimensioni del contenitore oppure verrà visualizzata soltanto la porzioni di immagine in alto a sinistra con dimensioni pari alle dimensioni del box . Impostando **line-height > width** la scritta verrà visualizzata nella parte inferiore dell'immagine.

Inserimento di un testo a comparsa all'interno di un'immagine

```
<div class="comparsa"> <p> Lorem Ipsum </p> </div>

div.comparsa {
    width: 400px;
    height:300px;
    background-image: url('img.jpg');
    background-repeat: no-repeat;
    position:relative;
    padding: 0;
}

div.comparsa p {
    background: rgba(0,0,0,0.5); // 0=trasparente, 1=solido
    color:#EEEEEE; /* colore del testo */
    display:none; /* nasconde la scritta */
    position:absolute;
    left:0px;
    bottom:0px;
    margin: 0;
    height:40px;
    line-height:40px;
    padding-left:10px; /* piccolo spaziatura del testo a sinistra */
    width:390px;
}
```

Invece di impostare **width:390px** si potrebbe impostando **width: 100%** in modo che il tag `<p>` occupi la stessa larghezza del genitore. Avendo però impostato un padding di 10, (ed essendo width espresso al netto del padding) questo padding andrà a sommarsi alle dimensioni dell'area del tag p, eccedendo le dimensioni del contenitore. Il problema può essere risolto impostando sul contenitore esterno **overflow: hidden**. In questo modo, se le dimensioni del tag p interno eccedono le dimensioni del contenitore (come effettivamente è), la parte eccedente non viene mostrata.

Per far comparire il testo in corrispondenza del mouse over è sufficiente impostare;

```
div.comparsa:hover p {
    display: block; }
```

Posizionamento di una Barra a fondo pagina insensibile allo scroll

Si utilizza un primo DIV container largo quanto la pagina, avente **position: fixed** (che lo rende insensibile allo scroll) e **bottom: 0** che lo posiziona allineato al bordo inferiore del genitore (la pagina HTML).

Si utilizza quindi un innerObject contenente gli oggetti desiderati.

```
#footer_bar {
    position: fixed;
    bottom: 0;
    width: 100%;
}
```

```
#bar_innerObject {
    background: #FAA53A;
    width: 900px;
    height: 35px;
    margin: 0 auto;
    overflow: hidden; }
```