

CSS

Rev. 5.2 del 09/11/2023

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
Le pseudoclassi	7
Regole base di applicazione dei selettori	7
Elementi Block ed Elementi Inline	8
display:inline-block	9
display:grid	10
La proprietà Overflow	10
Le proprietà ListStyle e Cursor	11
La proprietà Float	12
Le proprietà Position e Z-Index	13
Altre Proprietà CSS.....	14
Considerazioni sulle unità di misura	15
Approfondimenti su selettori e pseudoselettori	16
Esempi di Effetti realizzabili tramite CSS	20

I Fogli di Stile - Cascading Style Sheets

HTML inizialmente affidava la formattazione grafica della pagina ad i vari attributi. Da HTML 3.0 in avanti, anziché incrementare il numero di attributi dei vari TAG, si è preferito seguire un'altra strada, deprecando molti attributi e lasciando all'HTML soltanto la struttura base della pagina e demandando tutta la formattazione grafica ai **Fogli di stile (CSS Cascading Styles Sheets)**.

Le cosiddette **Proprietà di STILE** sono introdotte da un nuovo attributo **STYLE**, che consente di definire delle **proprietà ortogonali rispetto a tutti i TAG HTML**, cioè che possono essere applicati indistintamente a quasi tutti i TAG HTML, 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 vari tag HTML
2. In modo compatto nella sezione di HEAD del file HTML
3. in un file esterno con estensione .CSS

Uno stesso tag può essere ridefinito sia su un file .CSS esterno, sia nell'intestazione della pagina, sia direttamente nel TAG. **Cascading** significa che le proprietà vengono applicate in cascata cioè **le definizioni successive, più vicine al tag, nascondono le definizioni precedenti**. Cioè :

- gli stili inline sovrascrivono sia quelli della head sia quelli del file esterno.
- Le definizioni scritte nella head o in un file esterno sono equivalenti. **Prevalgono quelle scritte dopo**. Cioè se il file esterno è richiamato DOPO gli stili della HEAD, le proprietà del file esterno nascondono quelle della head e viceversa.

(1) stili INLINE definiti all'interno del tag

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

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

(2) stili definiti nella sezione di head

```
<head>
  <STYLE>
    P { color: blue; font-family: verdana, "times new roman" }
    /* "times new roman" ha le virgolette perchè contiene degli spazi */
  </STYLE>
</head>
<body>
  <p> Questo viene scritto in verdana di colore blu </p>
  <p style = "color:red; font-size:30pt;" > Questo invece è rosso </p>
</body>
```

(3) stili definiti in un file esterno .css

Il file esterno consente una maggiore suddivisione fra la struttura della pagina e la sua formattazione grafica.

Inoltre un file esterno potrà essere utilizzato in più pagine HTML di un stesso 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>
```

I tre tipi di selettore

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

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

I Selettori di classe

Consentono di creare più “**classi**” associabili a più istanze dello stesso tag oppure anche di tag differenti. I selettori di classe sono introdotti da un **puntino** e potranno essere associati alle varie istanze dei tag mediante l'utilizzo dell'attributo **CLASS**

```
.mioStile1 { font-size:14pt; color:red; }  
.mioStile2 { font-size:20pt; color:blue; font-weight:bold; }  
  
<p class="mioStile1"> Questo è il mio stile 1 </p>  
<div class="mioStile2"> Questo è il mio stile 2 </div>
```

E' anche possibile definire dei selettori di classe associabili *soltanto* ad un certo tipo di tag, che va espressamente indicato davanti al puntino. Esempio:

```
p.titolo { color:red; font-size:20pt}  
p.sottotitolo { color:blue font-size :12pt}  
  
<p class="titolo"> Questo è un titolo </p>  
<p class="sottotitolo"> Questo è un sottotitolo </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.

```
definisce lo stile associato all'elemento HTML avente ID="BOX"  
#BOX { color:red; font-size:20pt }  
  
<div id="BOX">  
  <a href="Pagina2.html"> Questo link è scritto in rosso 20pt </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** accordati 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> > (<i>oblique</i> era più inclinato dell' <i>italic</i> . 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 > Colore del testo.
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	<normal <length> Necessaria unità di misura Definisce una spaziatura aggiuntiva fra le parole. Sono ammessi valori negativi.
Letter-Spacing	<normal <length> 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 <length> <percentage> Default 0. <percentage> è riferito al valore di Width
Text-Transform	<none capitalize uppercase lowercase> <ul style="list-style-type: none"> • uppercase (Converte in maiuscolo l'intero contenuto) • lowercase (Converte in minuscolo l'intero contenuto) • capitalize (Converte in maiuscolo il primo carattere di ogni 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 al contesto di riga in cui l'oggetto è inserito**. Cioè se ad esempio si inserisce una immagine all'interno di una riga testuale, se si imposta il suo **vertical-align: BOTTOM** (default), il testo circostante sarà allineato in basso (e quindi l'immagine si espanderà verso l'alto).

E' analogo all'attributo html **ALIGN** ed ha come valore di default **BOTTOM**

width e height

Possono esser applicati a qualsiasi tag block o inline-block, compresi **checkbox** e **radio buttons**.

I BORDI di un contenitore

Border-Style	[none dotted dashed solid double groove ridge inset outset] 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)	<color> 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	<thin medium thick <length> > 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 p { border : 1px solid black }

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

Border-Top	<border-top-width> <border-style> <color> 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**

CSS

Background-Color	<color> 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 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; }
```

Le pseudoclassi

Nei CSS 2 sono state definite alcune pseudoclassi introdotte tramite i **due punti** :

:hover stile generico (cioè applicabile a tutti i tag) dal **verbo inglese** hover che significa librarsi, svolazzare. Applicato in corrispondenza del mouse-over ed automaticamente rimosso in corrispondenza del mouse-out.

Es `p:hover { color : yellow; }`

:focus stile applicato solitamente sui controlli indica la presenza del focus sul controllo.

Utile per modificare lo stile di un campo di input nel momento in cui riceve il focus.

:valid stile applicato ai soli textbox nel momento in cui il textbox contiene un valore valido

:invalid stile applicato ai soli textbox nel momento in cui il textbox contiene un valore NON valido

a:link stile applicabile solo al tag <a> rappresenta il colore del collegamento ipertestuale nello stato di riposo

a:visited stile applicabile solo al tag <a> rappresenta il colore del tag dopo essere stato 'visitato'

a:active stile applicabile solo al tag <a> rappresenta il colore del tag nel momento in cui viene cliccato

p:nth-of-type(1) Indica il primo tag p in una sequenza di fratelli

Regole base di applicazione dei selettori

- I vari selettori di stile possono far riferimento a tag posizionati uno dentro l'altro.
A tale scopo si utilizza l'operatore **SPAZIO**
`#box p a` significa **tag a** inserito all'interno di un **tag p** inserito all'interno del tag **#box**
`#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 **class2** definiti all'interno di **tag div** con **class1**
- Tra un selettore e l'altro si può utilizzare la **parentesi angolare >** per indicare un figlio diretto.
`#box > p` significa **tag p** figlio diretto di **#box**
- 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#box 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.
`#liv1 label, #liv2 .class1, #liv2 p a, .riga a { }`
- 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

Livelli di priorità

Gli stili vengono normalmente assegnati in cascata. A parità di importanza il successivo copre il precedente. Esistono però **4 livelli di priorità** che appunto prevalgono rispetto all'ordine di scrittura degli stili

- Liv 4** L'attributo **!important** introduce un livello 4 prioritario rispetto a TUTTI gli altri livelli,
- Liv 3** I selettori di elemento (selettori assoluti) sono prioritari rispetto agli altri due selettori,
- Liv 2** I selettori di classe sono prioritari rispetto al selettore di tag
- Liv 1.5** **p:nth-of-type()** è prioritario rispetto a **p**
- Liv 1** I selettori di tag hanno il livello di priorità più basso

Inoltre **i selettori con path più profondo sono prioritari rispetto a quelli più generici**

```
div div {color:red;}
div {color:green;}           // I tag div interni ad un altro div avranno colore red
div {color:green !important;} // Ora invece avranno colore green
```


Elementi BLOCK ed Elementi INLINE: la proprietà display

Su qualsiasi tag è possibile modificare il valore della proprietà **DISPLAY** da **INLINE** a **BLOCK** e viceversa.

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**
- **width** accetta come valore anche **fit-content** che imposta una larghezza pari al contenuto
- Al posto di **width** si utilizza talvolta **max-width** che presenta un vantaggio: stringendo la finestra, nel caso di **width** il testo eccedente non risulta 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**, **img**) 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**, che equivale a **fit-content**)
- sono normalmente riconosciute le proprietà **background-color** e tutte le proprietà relative al font.
- riconoscono completamente la proprietà **Padding**.
- riconoscono i **margin** laterali ma non riconoscono **Margin-Top** e **Margin-Bottom**
- Le dimensioni complessive del tag vengono calcolate esattamente come per i block tag (come indicato sotto).
- non riconoscono **Text-Align** (che non è significativa). ma riconoscono **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.

Nota: **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) a seconda del parent.

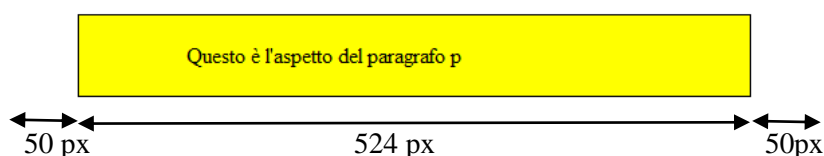
Per avere allineamento perfetto fra due tag inline-block posizionati sulla stessa riga la soluzione migliore è quella di impostare su entrambi la proprietà **vertical-align: middle** tranne che per i **text box** per i quali è consigliato mantenere **SEMPRE** il valore di default che è **vertical-align: text-top**;

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 **Height** complessiva risulta **Height + PaddingTop + PaddingBottom + Bordi = 30 + 20 + 20 + 4 = 74px**.

Se non si indica esplicitamente il valore di **height**, il tag DIV avrà una altezza pari all'altezza del testo + 40px di **padding** (superiore e inferiore) più il **border**, per cui il testo risulterà perfettamente centrato in verticale.

Se invece si specifica il valore di **height**, per avere un perfetto 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 orizzontali (left e right) di elementi consecutivi si sommano sempre.**
- **I margini verticali (top e bottom) di elementi verticalmente consecutivi :**
 - Nel caso di elementi statici di tipo block **si sovrappongono**
 - Nel caso di elementi inline-block, float, position:absolute, display:flex i margini si sommano
- **Se un elemento (block, inline, inline-block) è completamente contenuto all'interno di un elemento block, il suo margin fuoriesce all'esterno del genitore, andando a sovrapporsi a quello del genitore.**

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**.

La stessa cosa si può fare anche per gli **input[type=text]**

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**.

Il valore display:inline-block

Il valore **display:inline-block** rappresenta un'ottima tecnica per visualizzare **una sequenza di elementi affiancati**.

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

- rimanga sostanzialmente un tag **inline** per cui
 - riconosce completamente la proprietà **Padding**.
 - riconosce tutti e 4 i **Margin** ma non il valore **auto**, per cui **non** è possibile impostare **margin: 0 auto**
 - **non** riconosce **Text-Align** (che non è significativa). **ma** riconosce **Vertical-Align** che rappresenta l'allineamento verticale rispetto ad una ipotetica linea di testo.
 - Il valore **width:auto** non corrisponde all'intera riga (come per i tag block), ma al valore del contenuto come per i tag inline (come anche per il tag **float**)
- rispetto ad un normale tag **inline**
 - Riconosce le proprietà **width** e **height**
 - A differenza dei tag inline riconosce **Margin-Top** e **Margin-Bottom**

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

Per questo riconoscono **width**

I tag **input** e **button** inoltre presentano di default il testo centrato verticalmente.

Per centrare verticalmente una sequenza di elementi occorre impostare su ciascuno di essi lo stesso valore di **vertical-align (top oppure middle)**. *Nota:* Tra un elemento e l'altro permane comunque sempre uno spazio HTML dovuto alla spaziatura fra i tag. (non con float:left).

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 all'interno del quale si vuole inserire un elenco ordinato di righe ognuna costituita da una LABEL seguita dal corrispondente TEXTBOX

category:	<input type="text" value="cooking"/>
title:	<input type="text" value="Everyday Italian"/>
lang:	<input type="text" value="en"/>
author:	<input type="text" value="Giada De Laurentis"/>
year:	<input type="text" value="2005"/>
price:	<input type="text" value="30.00"/>

```
#wrapper {
  display: grid;
  width: fit-content;           // larghezza del wrapper. Va bene anche un valore fisso: width=350px
  grid-template-columns: 160px auto;           // larghezza di ogni colonna
  grid-template-rows: 25px 25px 25px 25px 25px 25px; // altezza di ogni riga
  grid-column-gap: 8px;
  grid-row-gap: 5px;
}
```

Il valore `grid-template-columns` può essere impostato al valore `fit-content` su una o più colonne

Il valore `grid-template-row` deve essere settato per ogni singola riga.

Se omissa le righe assumono l'altezza necessaria per visualizzare il contenuto.

Se presente, tutti gli elementi interni assumono quell'altezza di riga, compresi **checkbox** e **radiobutton**

Per gestire gli allineamenti, si possono impostare sugli elementi interni la seguente proprietà specifiche per il `display:grid`, proprietà che possono assumere i valori `start/end/center`:

```
#wrapper div{
  justify-self:end;           /* equivale a text-align:right */
  align-self:center;         /* equivale a vertical-align:middle */
}

div { grid-column: span 2 }  fa sì che div si espanda su 2 colonne.
div { grid-row: span 2 }    fa sì che div si espanda su 2 righe
```

La proprietà Overflow

Ha senso quando il testo interno **eccede** le dimensioni di un contenitore che ha **dimensioni fisse**.



















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

I possibili valori che può assumere sono:

- **visible** (default) La porzione eccedente le dimensioni del box viene mostrata eccedendo le dimensioni
- **hidden** La porzione eccedente le dimensioni del box non viene visualizzata
- **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” gli elementi **FLOAT** interni, nel qual caso occorre **NON** impostare height: il contenitore si estende automaticamente
overflow:auto impostato su un contenitore, fa sì che l'elemento vada **AUTOMATICAMENTE** a colmare un eventuale spazio libero lasciato alla sua destra da un elemento float:left antecedente avente una sua width.

- **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.







La proprietà cursor

		
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

CSS Reset

Quando si definiscono gli stili, occorre tener conto anche dei valori di default utilizzati dai vari tag, valori che possono cambiare da browser a browser.

La soluzione a questo problema è quella di definire un file di reset che resettati tutti i valori di default in modo da equalizzare la visualizzazione su tutti i browser. A tale scopo sono stati realizzati diversi fogli di stile chiamati **CSS Reset** che vanno inseriti prima di tutte le altre regole CSS del progetto. Tra i più noti Eric Meyer CSS Reset e Yahoo CSS Reset. Tra i valori più importanti da resettare margin e padding :

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

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

Inoltre, per far sì che un wrapper a sfondo colorato si estenda in verticale su tutta la pagina, occorre settare all'inizio del file le seguenti regole:

```
html, body, #wrapper {height:100%}
```

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 puntino/numerazione da utilizzare Esempio: <code>OL { list-style-type: upper-alpha }</code>
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.

Ci sono alcune combinazioni per le quali **List-Style** viene automaticamente settato a **none**. Ad esempio :

- 1) quando gli LI sono di tipo inline-block
- 2) quando UL ha padding=0 e si trova ancorato sul lato sinistro del genitore.

La proprietà FLOAT

La proprietà float consente di **ancorare un elemento su uno dei lati (destro o sinistro) del contenitore** in modo simile agli attributi HTML ALIGN=left e ALIGN=right del tag IMG.

- Gli elementi **FOAT** non vengono conteggiati nel normale rendering della pagina (esattamente come i position:absolute), per cui possono sovrapporsi agli elementi statici.
- La proprietà FLOAT è applicabile sia ai tag BLOCK sia ai tag INLINE che diventano **INLINE-BLOCK**, per cui **non si estendono più per tutta la riga, ma solo per la larghezza necessaria per visualizzare il contenuto**, a meno che non si specifichino le proprietà WIDTH e HEIGHT
- negli elementi **float** e **position absolute** i margini verticali non si sovrappongono ma si sommano

Se il tag **FLOAT** è inserito all'interno di una porzione di testo, **il testo semplicemente scorrerà intorno ad esso sul lato opposto rispetto all'elemento float**.

Se invece dopo un tag **FLOAT:LEFT** ci sono degli elementi **BLOCK** il comportamento è il seguente:

- Se si desidera che gli elementi BLOCK vengano affiancati all'elemento FLOAT, , **a livello di html** l'elemento float deve essere dichiarato **"prima"** degli elementi BLOCK .
- In ogni caso gli elementi **block** andranno a **"sovrapporsi"** all'elemento float non conteggiato nel rendering
- L'elemento block successivo non si sovrappone se :
 1. se anch'esso viene dichiarato **float**
 2. se ha un margin-left superiore alla larghezza dell'elemento float, cioè tale da spostarlo alla sua destra
 3. se viene dichiarato **overflow:auto**, nel qual caso andrà ad occupare lo spazio orizzontale libero a destra dell'elemento fluttuante.
- Una 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 adeguato. L'elemento BLOCK deve essere dichiarato DOPO rispetto ad entrambi gli elementi FLOAT.

Terminazione della fluttuazione

- Se l'elemento successivo è un elemento **inline** oppure **inline-block**, questo **"sente"** agli elementi fluttuanti e dunque viene automaticamente posizionato *dopo* di loro.
- Se l'elemento successivo è un elemento **block**, si può utilizzare la proprietà **CLEAR:BOTH** sul primo elemento block successivo alla sequenza float. **CLEAR:BOTH interrompe la fluttuazione posizionando l'elemento sulla prima riga libera DOPO gli elementi fluttuanti**. Il valore CLEAR:LEFT interrompe soltanto il float:left e non il float:right e viceversa

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 o 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 stesso **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 bene specificare manualmente la larghezza desiderata.
- Gli elementi **float** non **sentono margin:0 auto** per cui un contenitore float non può essere centrato nella pagina. Viceversa nel caso di contenitore **overflow:auto** si ha un problema nel caso del DRAG & DROP
- Se il contenitore non ha **height** né **float** né **overflow auto** ma ha **width**, sarà in pratica 'inesistente' ai fini del layout. Se però l'elemento successivo ha overflow:auto, 'sente' comunque il contenitore di sinistra e verrà posizionato alla sua destra.

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. Impostando **top** e **left** al valore 0 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

Gli elementi con position != static non vengono conteggiati nel normale rendering della pagina, per cui più elementi possono sovrapporsi tra loro oppure possono sovrapporsi ad eventuali elementi statici presenti in quella posizione. La sovrapposizione dovrà essere gestita tramite z-index

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) Impostando **position!=static** su un qualsiasi elemento, la proprietà **display** assume automaticamente il valore **inline-block**, a meno che non venga esplicitamente ridefinita, per cui **width** non sarà pari alla larghezza del contenitore ma sarà pari alla larghezza del contenuto.
- 2) Impostando **position: absolute** su un elemento senza impostare top e left (o right/bottom), queste property assumono come valore di default **auto**, che sta ad indicare l'angolo in alto a sinistra del componente in cui si trovano (cioè nell'angolo in alto a sinistra del genitore diretto)
- 3) Negli elementi **float** e **position != static** i margini verticali non si sovrappongono ma si sommano
- 4) Le **position: absolute** e **float** sono in genere alternativa (o si imposta una oppure l'altra). Nel caso si impostassero entrambe, la proprietà **position: absolute** prevale sulla proprietà **float**.

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 (cioè più vicini all'utente) rispetto ad elementi con z-index minore.

z-index è definito SOLTANTO per oggetti che hanno position: absolute/relative/fixed.

Tutti gli elementi statici hanno per default z-index=0 non modificabile.

Per visualizzare un elemento absolute al di sotto degli elementi statici occorre assegnargli il valore **z-index = -1**

Altre Proprietà CSS 2

max-width: larghezza massima dell'elemento

max-height: altezza massima dell'elemento

min-width: larghezza minima dell'elemento

min-height: altezza minima dell'elemento

max-width e **min-width** rappresentano un primo esempio di responsive design

- **max-width** serve per limitare la larghezza di un oggetto su schermi di grandi dimensioni.
- **min-width** serve per evitare un eccessivo 'rimpicciolimento' su schermi di piccole dimensioni

Entrambe queste impostazioni equivalgono sostanzialmente ad impostare **width:auto**, cioè l'elemento si estende per l'intera larghezza del contenitore.

- Nel caso di **max-width** nel momento in cui la larghezza dell'elemento supera la soglia indicata, **width** assumerà il valore indicato, impedendo che l'elemento continui ad allargarsi oltre tale soglia
- Nel caso di **min-width** nel momento in cui la larghezza dell'elemento scende sotto la soglia indicata, **width** assumerà il valore indicato, impedendo che l'elemento continui a restringersi al di sotto di tale soglia

max-height risulta molto comodo nel caso in cui si abbia una sequenza di **tag img** disposti orizzontalmente, 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 doverle ingrandire con il rischio di perdere di qualità (ad esempio Esercizio Ricette Angular)

display Impostando il valore **none** l'elemento viene completamente '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 meno forte del precedente. Può assumere i valori **visibility:hidden** **visibility:visibile** **visibility:collapse**. 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. Sia **display:none** che **visibilità:hidden** **disabilitano gli eventi** !

opacity Ancora meno forte dei due precedenti. Numero con la virgola tra **0=Transparent** e **1=solido** (default). Utile per rendere semitrasparente un oggetto rispetto al contenitore sottostante.

- A differenza dei due precedenti continua a sentire gli eventi (compreso **:hover**).
- Rende trasparente sia il testo dell'oggetto, sia il suo sfondo.
- Applica inderogabilmente la trasparenza anche a TUTTI gli oggetti figli. Se il box ha opacity:50% e l'elemento interno opacity:100%, l'elemento interno avrà opacità risultante 75%. Difficile da gestire.
- A differenza di RGBa è applicabile anche alle immagini di sfondo.
- Avendo un valore numerico, consente l'utilizzo della proprietà **transition**.

rgba() Il meno forte di tutti. Può essere applicato separatamente al testo dell'oggetto oppure al suo sfondo. Esattamente come opacity è un numero decimale compreso tra **0=trasparente** e **1=solido**

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

- A differenza di opacity, il valore di RGBa NON viene ereditato dagli elementi interni. MEGLIO, perché spesso gli elementi interni NON devono essere trasparenti.
- Riguardo alla funzione **rgb()** sono anche disponibili le seguenti sintassi:
color: rgb(0 0 0); // senza virgole
color: rgb(0 0 0 / 0.5); // con trasparenza, senza ricorrere a rgba

user-select:none Applicata ad un contenitore, inibisce la selezione del testo negli elementi interni al contenitore.

white-space Può assumere i valori **normal** (default, il testo interno va a capo in corrispondenza del margine destro)
- **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 ma vanno a capo).

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** indica le dimensioni dell'oggetto rispetto alla **height** del **wrapper contenitore**.

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

% 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:150%** equivale a **font-size:1.5em** 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.

em applicato a **font-size** è equivalente a % (fattore di moltiplicazione del font-size **rispetto al font-size del genitore**)

font-size:0.8em equivale a **font-size:80%**

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.

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 font-size ereditato applicando un fattore moltiplicativo o di riduzione** come indicato nella seguente tabella:

H1 raddoppia entrambi i valori (font **24pt** e padding **32px**)

H2 applica un aumento del 50% (font **18pt** e padding **24px**)

H3 applica un aumento del 16,5% (metà di 33) (font 14pt e padding 19px)

H4 applica al font lo stesso font-size e lo stesso padding del genitore

H5 applica una riduzione di 16,5% (font 10pt padding 13px)

H6 applica una riduzione di 33% (font 8pt padding 10px)

Nota: Se il tag H1 anziché trovarsi dentro un tag DIV si trova dentro 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.

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; }
```

Approfondimenti su selettori e pseudoselettori

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.

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

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

Questa regola diventa prioritaria rispetto a qualunque altra regola anche più specifica.

Utilizzo dell'ID in modo non univoco

L'ID in realtà può anche essere assegnato in modo non univoco purché su tag differenti (approccio non proprio ortodosso). Ad esempio se un tag DIV ed un tag P hanno entrambi ID="titolo", da CSS si può accedere separatamente ai due tag nel modo seguente:

```
div#titolo { color="blue" }
p#titolo { color="green" }
```

Accesso agli elementi che implementano più classi

Qualunque tag html può implementare più classi:

```
<p class="class1 class2"> </p>
```

Se si vogliono definire delle proprietà CSS per gli elementi che implementano sia classe1 che classe2, nel file CSS le due classi devono essere scritte “attaccate” senza spazi intermedi

```
.class1.class2 {  
    background-color:green;  
}
```

Selettori di filtro

- 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.
div + p
Individua tutti i tag <p> scritti immediatamente dopo un tag <div> (*directly following*).
- Il selettore ~ indica tutti i fratelli successivo ad un certo tag.
div ~ p
Individua tutti i tag <p> che seguono genericamente un tag <div> all'interno di un medesimo livello

Non esistono purtroppo selettori che consentano di accedere al/ai fratello/i antecedenti.

Selettori di attributo []

Il selettore di attributo [] consente di individuare tutti gli elementi della pagina che implementano un certo attributo html. E' utilizzabile **per qualsiasi attributo html**, compresi quelli definiti dal programmatore, però SOLO per quelli statici definiti mediante **setAttribute** e non per quelli definiti dinamicamente tramite puntino.

Attenzione a **NON lasciare spazi** tra il nome del selettore e le parentesi quadre.

```
<input type="text" name="txtNome" classe="3B" required >  
input[type=text]  
input[name=txtNome]  
input[classe=3B]  
input[type=radio]  
input[type=checkbox]
```

Selettori di attributo per attributi booleani

Per gli attributi booleani invece delle [] si utilizza il simbolo **:**.

Attenzione anche in questo caso a **NON lasciare spazi** tra il nome del selettore e i due punti.

```
:checked    (radiobuttons o checkbox attualmente selezionati)  
:selected  (la option di un listBox attualmente selezionata)  
:disabled  (tutti i tag aventi l'attributo html disabled=true)  
:enabled   (tutti i tag aventi l'attributo html disabled=false)  
:required  (tutti i tag aventi l'attributo required)
```

I vari pseudoselettori possono anche essere applicati in forma combinata:

```
input[type=radio][name=opt1]:checked
```

Esempio: applicare uno sfondo grigio su tutti i textbox disabilitati

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

Lo pseudoselettore `nth-of-type(i)`

Quando, all'interno di un contenitore, si ha una sequenza anche NON consecutiva di elementi dello stesso tipo (ad esempio una sequenza di DIV oppure una sequenza di P oppure una sequenza di LI), è possibile accedere ad ogni singolo elemento della collezione utilizzando lo pseudo selettore **`nth-of-type()`** seguito dall'indice dell'elemento a cui si intende accedere:

```
#wrapper > div:nth-of-type(3) // Terzo tag DIV figlio diretto di wrapper
#wrapper > p:nth-of-type(3)   // Terzo tag P figlio diretto di wrapper

:first-of-type è equivalente a nth-of-type(1)
:last-of-type è equivalente a nth-of-type(ultimo)
```

even e odd

Gli pseudoselettori **`: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.

even e **odd** si scrivono **SENZA** i due punti davanti e sono utilizzabili soltanto all'interno di **`nth-child(even)`** e **`nth-of-type(odd)`**

Come si calcola `nth-of-type`

La scritta **`#wrapper p:nth-of-type(3)`** indica tutti quei tag **`<p>`**, figli o nipoti di wrapper, che godono della proprietà di essere **il terzo tag `<p>`** rispetto al proprio genitore.

La scritta **`#wrapper :nth-of-type(3)`** indica tutti quei tag, figli o nipoti di wrapper, che godono della proprietà di essere **terzogeniti del loro tipo** rispetto al proprio genitore.

Si tenga presente però che `nth-of-type` è SEMPRE calcolato esclusivamente sulla base dei tag, e non sulla base di classi o pseudoSelettori aggiuntivi

- La scritta **`#wrapper p.myclass:nth-of-type(3)`** non rappresenta il terzo tag `<p>` che implementa la classe `myClass`, ma indica tutti quei tag `<p>`, figli o nipoti di wrapper, che godono in generale della proprietà di essere **terzogeniti del loro tipo**. Se questo **terzogenito** implementa la classe `myClass`, allora sarà restituito, altrimenti se il **terzogenito** non implementa la classe `myClass`, verrà restituito un insieme vuoto. **Cioè il 3 è calcolato sui tag `<p>`, non sulle classi `myClass`**
- La scritta **`#wrapper .myclass:nth-of-type(3)`** indica **tutti** i tag interni a wrapper che godono della proprietà di essere **terzogeniti del loro tipo**, ma **solo se** implementano la classe `myClass`.
- Allo stesso modo, anche nel caso dei vari pseudo selettori, **`nth-of-type` è SEMPRE calcolato rispetto al tag iniziale a cui si riferisce e NON sullo pseudo selettore.**

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

va a prendere il **5° tag input** della pagina (solo se questo tag è di tipo radio), **indipendentemente dal type dei tag input precedenti**. Se ci fossero quattro `input[type=text]` e dopo un tag `input[type=radio]`, le proprietà verrebbero applicate a questo **5° tag input (di tipo radio)**

Cioè ancora i vari pseudo selettori, anche se usati in forma multipla, NON vengono applicati in cascata, ma **qualunque pseudoselettore viene applicato sempre sul tag iniziale**, e **NON** su una ipotetica collezione individuata dallo pseudoselettore precedente:

Lo pseudoselettore `:nth-child(i)`

E' simile a `:nth-of-type(i)` ma indica l'i-esimo elemento indipendentemente dal tipo.

```
#wrapper :nth-child(3)
```

indica il 3 elemento all'interno di wrapper indipendentemente dal suo tipo

Pseudoselettori di tipo functions

```
:not(selettore_Secondario) // Es input[type=radio]:not(:checked)
#menu li:has(ul)           // Le voci di menu che contengono il tag indicato (anche annidato)
:contains(testo);          // deprecato
```

Gli pseudoselettori `::after` e `::before`

`::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

Personalizzazione della grafica di checkbox e radio buttons

Per poter modificare la grafica di un checkbox o di un radio button, occorre necessariamente rimuovere gli stili applicati di default che non possono essere sovrascritti nemmeno con il **!important**

```
input[type=checkbox] {
  -webkit-appearance: none;
  display: inline-block;
  width:16px; height:16px;
  padding:0; border-radius: 3px;
  background-color:#eee;
  border: 1px solid #ccc;
  box-shadow: 1px 1px 2px rgba(0,0,0,0.05);
  position: relative; }

input[type=checkbox]:checked {
  background-color: #ffa;
  color: #999; }

input[type=checkbox]:checked::after {
  content: '\2714'; /* codice unicode della spunta */
  font-size: 12px;
  position: absolute;
  top: -2px; left: 2px; }
```

Esempi di Effetti

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; }
```