

# CSS 3

---

# Cosa è il CSS

---

L'acronimo **CSS** sta per **Cascading Style Sheets** (fogli di stile a cascata) e designa un linguaggio di stile per i documenti web. I CSS istruiscono un browser o un altro programma utente su come il documento debba essere presentato all'utente

# albero del DOM

---

## Elementi blocco (block) ed elementi in linea (inline)

Gli elementi blocco sono box che possono contenere altri elementi, sia di tipo blocco che di tipo inline. **Quando un elemento blocco è inserito nel documento viene automaticamente creata una nuova riga nel flusso del documento**

```
<h1>Titolo</h1>
```

```
<p>Paragrafo</p>
```

## Elementi Inline

Gli elementi inline non possono contenere elementi blocco, ma solo altri elementi inline

Tramite i CSS possiamo modificare tale modalità attraverso la proprietà **display**. Grazie a quest'ultima, per fare solo un esempio, possiamo fare in modo che un titolo h1 (elemento blocco) venga mostrato come un elemento in linea

# albero del DOM

---

## Elementi rimpiazzati e non rimpiazzati

Un'altra distinzione da ricordare è quella tra elementi rimpiazzati ed elementi non rimpiazzati,

i **rimpiazzati** sono quelli in cui altezza e larghezza sono definite dall'elemento stesso e non da ciò che lo circonda, ad es:

img

input

text

textarea

select

**Non rimpiazzati** (non hanno dimensioni)

div

section

# Elementi Padri e figli

---

Un elemento si dice **genitore** (**parent**) quando contiene altri elementi. Si dice **figlio** (**child**) quando è racchiuso in un altro elemento

Se si scende di due livelli: diciamo allora che è un **antenato** di e che questo è rispetto al primo un discendente.

gli elementi che sono posti sullo stesso livello, ovvero quelli che hanno lo stesso genitore, si dicono **fratelli** (ingl: siblings)

il capostipite (quello che non ha padri [`<html>`]) si dice che è l'**elemento radice**

# Com'è fatto un CSS

selettore

blocco delle dichiarazioni



proprietà      valore



dichiarazione

```
h1 {  
  color: red;  
  font: 36px Helvetica, Arial, sans-serif;  
}
```

Gli spazi bianchi lasciati all'interno di una regola non influiscono sul risultato

```
p {font: 12px Verdana, arial;}
```

# Proprietà singole e a sintassi abbreviata

---

è possibile fare uso di **proprietà singole** e **proprietà a sintassi abbreviata**

margin-top

margin-right

margin-bottom

margin-left

La regola sarebbe questa:

```
div {  
  margin-top: 10px;  
  margin-right: 5px;  
  margin-bottom: 10px;  
  margin-left: 5px;  
}
```

Lo stesso risultato si può ottenere usando la proprietà a sintassi abbreviata margin:

```
div {margin: 10px 5px 10px 5px;} TOP – RIGHT – BOTTOM - LEFT
```

```
div {margin: 10px 5px;} TOP+BOTTOM – RIGHT+LEFT
```

```
div {margin: 10px;} TOP - RIGHT+LEFT - BOTTOM
```

# Commenti

---

Nello snippet di codice visto ad inizio lezione, le parti racchiuse tra i segni `/*` e `*/`, rappresentano commenti al codice

`/* Stili per i titoli h1 */`

`/* Colore del testo delle liste */`

`/* Colore dei titoli h1 per la stampa */`



# Valori e unità di misura nei CSS

I valori di una proprietà **non vanno mai messi tra virgolette**.

Le uniche eccezioni riguardano i valori espressi da stringhe di testo e i nomi dei font formati da più di una parola.

```
p {font-family: "Times New Roman",  
Georgia, serif;}
```

Nei CSS i valori possono essere espressi da:

numeri possono essere definiti come numeri interi (1, 23, 45, etc.) o in virgola mobile (1.2, 3.45, 4.90, etc.)

unità di misura

percentuali

codici per la definizione dei colori

URI

parole chiave (keywords)

stringhe di testo

```
/* Altezza di linea con un numero */
```

```
p {line-height: 1.2;} 
```

```
/* Larghezza con unità di misura */
```

```
div {width: 300px;} 
```

```
/* Larghezza in percentuale */
```

```
div {width: 60%;} 
```

```
/* Colore con codice esadecimale */
```

```
body {background-color: #2795b6;} 
```

```
/* URL per un'immagine di sfondo */
```

```
body {background-image:
```

```
url('sfondo.jpg'); }
```

```
/* Ripetizione dello sfondo con una
```

```
keyword */
```

```
body {background-repeat: no-repeat;} 
```

```
/* Stringa di testo */
```

```
content: "Viva i CSS";
```

# unità di misura

---

## i più utilizzati:

**px:** unità più utilizzata ed ideale per gli schermi

**em:** è relativa alla dimensione standard (2 em= 2 volte la dimensione dell'attuale dimensione del font)

## Percentuale

Un valore espresso in percentuale è da considerare sempre relativo rispetto ad un altro valore, in genere quello espresso per l'elemento parente. Si esprime con un valore numerico seguito (senza spazi) dal segno di percentuale: **60%** è pertanto corretto, **60 %** no.

```
h1 { line-height: 1.2em }
```

Questa regola significa che la *line-height* dell'elemento *h1* sarà del 20% superiore alla *font-size* dell'elemento *h1*.

# Unità em vs rem

Quando si utilizzano unità **rem**, la dimensione in pixel in cui vengono tradotte dipende dalla dimensione del carattere **dell'elemento principale della pagina**, ovvero l'elemento **html**. Quella dimensione di carattere viene moltiplicata per qualsiasi numero stiate usando con le unità rem.

Ad esempio, con una dimensione di carattere dell'elemento principale di 16px, 10rem sarebbe pari a 160px, cioè  $10 \times 16 = 160$ .

Quando si utilizzano unità em, il valore in pixel finale è una moltiplicazione della dimensione del font sull'elemento cui si applica lo stile.

Ad esempio, se un div ha una dimensione di carattere di 18px, 10em sarebbe pari a 180px, cioè  $10 \times 18 = 180$ .

Se imposto ad una proprietà ad esempio height: 4rem e il valore del font-size dell'elemento html è 12px, l'altezza del contenitore sarà 48px;

```
html{  
  font-size: 12px;  
}  
div{  
  min-height: 4rem; /*=48px*/  
}
```

# Unità dp \_ si utilizza nelle app e web app

L'unità **dp** sta per 'density independent pixel' e indica una unità astratta basata sulla densità fisica dello schermo.

Un dp rappresenta un pixel su uno schermo di 160 dpi (dot per inch - punti per pollice).

Chiaramente il rapporto tra dp e pixel cambia al variare della risoluzione dello schermo.

Ad esempio una immagine con 240 dp se occupa 360px apparirà con una definizione maggiore rispetto alla stessa immagine che occupa 180px

Screen physical width	Screen density	Screen width in pixels	Screen width in dps
1.5 in	120	180 px	240 dp
1.5 in	160	240 px	
1.5 in	240	360 px	

# CSS esterni e interni

---

È esterno un foglio di stile definito in un file separato dal documento. Si tratta di semplici documenti di testo modificabili anche con un editor di testo ai quali si assegna l'estensione .css.

```
<html>
```

```
<head>
```

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

```
</head>
```

```
<body>
```

```
[...]
```

```
</html>
```

# @import

---

Un altro modo per caricare CSS esterni è usare la direttiva @import all'interno dell'elemento <style>:

```
<html>
<head>
  <style>
    @import url(style.css);
  </style>
</head>
<body>
  [...]
</html>
```

La direttiva import deve essere **la prima all'interno dell'attributo style**

# @import

---

Un principio fondamentale è che all'interno del tag <style>, @import deve essere la prima regola definita

@import viene usata innanzitutto per collegare un foglio di stile esterno al documento. La sintassi generica è la seguente:

```
<style type="text/css">  
  @import url(stile.css);  
</style>
```

```
@import url("stile.css");
```

2. Direttiva senza l'indicazione url:

```
@import "stile.css";
```

L'url del foglio di stile può essere relativo, come negli esempi precedenti, o assoluto, come in questo:

```
<style type="text/css">  
  @import url(http://www.miosito.it/stile.css);  
</style>
```

Possibile omettere type="text/css"

# Le @-rules

---

```
@media print {  
  h1 {color: black;}  
}
```

```
@media screen and (min-width:768px){  
  h1 {color: red;}  
}
```

Smartphone: < 768px

Tablet: >=768px;

Notebook: >=992px;

Desktop: >=1200px;

Breakpoint	Class infix	Dimensions
X-Small	<i>None</i>	<576px
Small	<i>sm</i>	≥576px
Medium	<i>md</i>	≥768px
Large	<i>lg</i>	≥992px
Extra large	<i>xl</i>	≥1200px
Extra extra large	<i>xxl</i>	≥1400px



# Come dichiarare una media query

---

attributo media nel link

```
<link rel="stylesheet" media=" screen and (min-width : 576px) "  
href="colore.css" />
```

dentro il foglio di stile

```
@media screen and (min-width: 576px) {  
/* qui vanno le regole CSS */  
}
```

all'interno di un altro foglio di stile

```
@import url(colore.css) screen and (min-width : 576px);
```

# CSS Interni

---

I fogli incorporati sono quelli inseriti direttamente nel documento HTML tramite il tag `<style>`. Anche in questo caso la dichiarazione va posta all'interno della sezione `<head>`

```
<html>
<head>
  <style type="text/css"> (eliminato in HTML5)
    body {background: white;}
    p {color: black;}
    [...]
  </style>
</head>
<body>
  [...]
</html>
```

# CSS in linea

---

L'ultimo modo per formattare un elemento con i CSS consiste nell'uso dell'attributo HTML style. Esso fa parte della collezione di attributi HTML definiti globali: si tratta di quegli attributi applicabili a tutti gli elementi. La dichiarazione avviene a livello dei singoli tag contenuti nella pagina e per questo si parla di fogli di stile in linea. La sintassi generica è la seguente:

```
<h1 style="color: red; background: black;">...</h1>
```

# Ereditarietà

---

le impostazioni di stile applicate ad un elemento **vengono ereditate anche dai suoi discendenti. Almeno fino a quando, per un elemento discendente, non si imposti esplicitamente un valore diverso per quella proprietà.**

```
body {color: #222;}
```

Tutti gli elementi discendenti di body, ereditano questa impostazione. Ma se ad un certo punto definiamo nel codice del CSS un selettore con la proprietà color: white; l'ereditarietà viene spezzata:

```
body {color: #222;}
```

```
li {color: white;}
```

# Peso

---

## 1- User CSS:

Sono i fogli di stile dell'utente: con un CSS in locale l'utente può ridefinire i CSS del browser e quelli dell'autore. **I CSS utente hanno priorità massima, e sono stati pensati soprattutto per l'accessibilità**, ma non solo.

*Strumenti->Opzioni Internet->Generale-> Accesso Facilitato->Fogli di stile utente*

## 2- Author CSS:

Ovvero i fogli di stile specificati dall'autore della pagina. Questi andranno a ridefinire i CSS del browser, e ci sono tre sottotipi, nell'ordine dal meno influente al più influente: **i CSS esterni, quelli incorporati e quelli inlinea**. Possono inoltre essere definiti per diversi *media*, ovvero diversi dispositivi.

## 3- User Agent CSS:

Ovvero **il foglio di stile di default del dispositivo con cui si sta visualizzando la pagina**. In particolare, per quanto riguarda i browser, è il foglio di stile con cui viene visualizzata una pagina senza alcun altro CSS.

# Specificità

---

I fattori del calcolo sono tre e ciascuno di essi rappresenta il valore di una tripletta. Per prima cosa si conta il numero di selettori id presenti nella regola. Si passa quindi a verificare la presenza di classi e pseudo-classi. Infine si conta il numero di elementi definiti nella regola

**gli id pesano più delle classi che pesano più dei singoli elementi**

ID

Classi

Elementi

# Stili in cascata

---

L'ordine, se le dichiarazioni degli stili sono fatte nell'ordine più corretto e logico, è quindi il seguente: gli stili **in linea** prevalgono su quelli **incorporati** che a loro volta prevalgono su quelli **collegati**.

Style=""

<style>

<link href=

# Importanza !important

---

se una dichiarazione viene accompagnata dalla parola chiave !important essa balza al primo posto nell'ordine di applicazione a prescindere da peso, origine, specificità e ordine



# Selettori di Base: Selettore universale

---

Il selettore universale è definito sintatticamente da un asterisco: \*. La sua funzione è quella di selezionare tutti gli elementi presenti in un documento

\* {color: red}

# Selettori di Base: Selettori di tipo

---

rappresentati dal nome di uno specifico elemento HTML

```
h1 {color: red}
```

```
p {color: green}
```

```
li {color: blue}
```

# Selettori di Base: Selettori di classe

---

HTML può essere assegnata una classe usando l'attributo class e assegnando ad esso un valore a nostra scelta

```
<h1 class="titolo">Testo</h1>
```

Nei CSS, per selezionare gli elementi a cui sia stata assegnata una classe, si utilizza questa sintassi:

```
.titolo {color: red}
```

# Selettori di Base: Selettori di ID

---

Anche id è un attributo universale in HTML. Significa che tutti gli elementi presenti nel documento possono avere un loro id. A differenza delle classi, però, uno specifico id può essere assegnato solo ad un elemento

Nei CSS, per selezionare un elemento cui sia stato assegnato un certo id, si usa questa sintassi, facendo precedere il valore dell'id dal simbolo del cancelletto (#):

#titolo

È anche possibile usare prima del cancelletto il nome dell'elemento:

h1#titolo

# Selettori combinatori o di relazione

---

Una categoria fondamentale di selettori CSS è rappresentata dai cosiddetti combinatori (detti anche selettori di relazione). Hanno la funzione di mettere in relazione elementi presenti all'interno dell'albero del documento. Sono quattro:

Selettore di discendenti (spazio)

Selettore di figli (>)

Selettore di fratelli adiacenti (+) e successori

Selettore generale di fratelli (~) e successori [ALT+126]

~ è per il fratello successore generale

+ è per il fratello successore adiacente

# Selettori discendenti

---

## Selettore di discendenti

Il selettore di discendenti è sicuramente quello più utilizzato dei quattro. Non è presente solo nella specifica CSS3 ma anche nelle precedenti versioni ed è utilissimo per evitare l'abuso delle classi per assegnare stili agli elementi

```
div#container p {color: red}
```

serve ad assegnare lo stile solo ai paragrafi contenuti nel div#container

```
<div id="container">
```

```
<p class="titolo">Questo testo è in un paragrafo discendente da un div con id  
<code>container</code> e sarà rosso.</p>
```

```
</div>
```

```
<div id="main">
```

```
<p>Questo testo è in un paragrafo discendente da un div con id  
<code>main</code> e non sarà rosso.</p>
```

```
</div>
```

# Selettore di figli

---

Il selettore di figli (>) consente di selezionare un elemento che è **figlio diretto** dell'elemento padre.

```
body > p {color: red}
```

```
<body>  
<p>Primo paragrafo</p>  
<div>  
<p>Secondo paragrafo</p>  
</div>  
<p>Terzo paragrafo</p>  
</body>
```

solo il primo e il terzo sono **figli diretti di body**. Il secondo è invece figlio diretto di un elemento div

# Selettore di fratelli adiacenti

---

serve a scorrere in orizzontale l'albero del DOM assegnando le regole CSS agli elementi che si trovano allo stesso livello di un altro elemento  
consente di assegnare uno stile all'elemento fratello immediatamente adiacente

```
<div>  
  <h1>1. Questo è il titolo principale.</h1>  
  <h2>1.1 Questo è il primo sottotitolo.</h2>  
  <p>...</p>  
  <h2>1.2 Questo è il secondo sottotitolo.</h2>  
  <p>...</p>  
</div>
```

Applicando la seguente regola

```
h1 + h2 {color: red; text-decoration: underline}
```

verrà selezionato solo il primo <h2> dato che è immediatamente adiacente al tag <h1>.



# Selettore generale di fratelli

---

(~) è una generalizzazione di quello visto in precedenza. Esso assegna uno stile a tutti gli elementi che sono fratelli **ALT+126**

```
<div>
  <h1>1. Questo è il titolo principale.</h1>
  <h2>1.1 Questo è il primo sottotitolo.</h2>
  <p>...</p>
  <h2>1.2 Questo è il secondo sottotitolo.</h2>
  <p>...</p>
</div>
```

e applicando il codice CSS seguente

```
h1 ~ h2 {color: red; text-decoration: underline}
```

andremo a selezionare tutti gli elementi <h2> dello stesso livello di <h1> indipendentemente dalla posizione che occupano.

**(DEVONO ESSERE FRATELLI E SUCCESSORI)**

# esempio completo

---

```
div#id1{background-color:#dff70c;} /*p id
id1*/
div#id1 + p{background-color:#0ec0c0
!important;} /*p fratello diretto di id
id1*/
div#id1 > p{background-color:#0b9722;} /*p
figli di id id1*/
div#id1 ~ p{background-color:#29f30e;} /*p
fratello di id id1*/
div#id1 p{background-color:#f30ec1;} /*p
discendenti (figli, nipoti ecc..) di id
id1*/
```

```
<div id="id1">p1
<p>figlio di p1
<p>nipote di p1
<p>pronipote di p1</p>
</p>
<p>2 figlio di p1</p>
</div>
<p>fratello di p1</p>
<p>fratello di p1</p>
```

# esempio completo 2

```
<p>paragrafo</p>
<div id="miodiv">test</div>
<div class="miaclasse">test
<p>figlio e discendente

</p>
<div class="cl1">
<p>discendente</p>
</div>
</div>
<h2>titolo 2</h2>
<div class="cl1">
<h2>titolo figlio cl1</h2>
</div>
<h2>titolo 3</h2>
<h3>fratello adiacente</h3>
<h3>fratello </h3>
```

```
body{margin:10% 10%;}
p{color: #2094b8}
#miodiv{color: #4bb820}
.miaclasse{color:#d32828}
div.miaclasse
p{color:#f33e07}/*discendenti*/
div.miaclasse >
p{color:#4ff027}/*figli*/
h2 ~ h3 {color:#4ff027;}
h2 + h3{color: #2094b8
!important;}
```

# Selettori di Attributo

---

consentono di selezionare gli elementi all'interno di una pagina in base ai loro attributi e assegnare così lo stile desiderato

E[attribute]

Questo selettore individua tutti gli elementi E che possiedono l'attributo attribute, indipendentemente dal contenuto dell'attributo.

a[title] {color: blue; text-decoration: underline}

<a title="Lorem Ipsum" href="#">Lorem Ipsum</a>

E[attribute=value]

Questo selettore individua tutti gli elementi E che possiedono l'attributo attribute che al proprio interno contiene il valore value

a[title="Lorem"] {color: blue; text-decoration: underline}

<a title="Lorem" href="#">Lorem Ipsum</a>

h2[data-indirizzo]{color:#23197c;}  
h2[data-indirizzo="mio"]{color:#c4c5b4;}

<h2 data-indirizzo="mio">titolo</h2>  
<h2 data-indirizzo="no">titolo</h2>

# Pseudo-classi :first-child

---

## Codice HTML:

```
<div id="div1">
<p>Primo paragrafo.....</p>
<p>Secondo paragrafo: nero</p>
<p>Terzo paragrafo: nero</p>
</div>
```

## Codice CSS:

```
#div1 {
color: Black;
font : 12px Verdana, Geneva, Arial,
Helvetica, sans-serif;
}
```

```
#div1 p:first-child {
color: Red;
font-weight: bold;
}
```

**Primo paragrafo: rosso e grassetto  
perchè primo elemento figlio di #div1.**

Secondo paragrafo: nero

Terzo paragrafo: nero

```
div#id1 p:first-child{background-
color:#dff70c;}
```

importante: inserire

**contenitore selettore-figlio:first-child**

# Pseudo-classi :link

---

Usare la pseudo-classe :link. Il [link](#) non visitato è rosso e non sottolineato.

**Codice:**

```
#div2 a:link {  
  color: red;  
  text-decoration: none;  
}
```

# Pseudo-classi :visited

---

## **:visited**

Usare la pseudo classe :visited. Il [link](#) visitato è verde e non sottolineato.

### **Codice:**

```
#div2 a:visited {  
color: Green;  
text-decoration: none;  
}
```

# Pseudo-classi :hover

---

Usare la pseudoclasse :hover. Al passaggio del mouse il link diventa blue e sottolineato.

**Codice:**

```
#div2 a:hover {  
color: Blue;  
text-decoration: underline;  
}
```



# Pseudo-classi :active

---

Usare la pseudoclasse :active. Mentre il tasto sinistro è premuto il link è viola e sottolineato.

## Codice:

```
#div2 a:active {  
color: Purple;  
text-decoration: underline;  
}
```

```
a:link{background-color:#d2d435;}  
a:visited{background-  
color:#3e1a80;}  
a:hover{background-color:#a8121e;}  
a:active{background-  
color:#3daf2d;}
```

# Pseudo-classi :lang

---

## :lang

Ora inseriamo del testo in inglese. Dovrebbe apparire in grassetto.

### Codice CSS:

```
p:lang(en) {  
font-weight: bold;  
}
```

### Codice HTML:

```
<p lang="en">This paragraph is in English!</p>
```

**This paragraph is in English!**

**Importante nel css non vanno gli apici [tr:lang(it)]**

```
input:lang(it){background-color:#3e1a80;}
```

```
<input name="mio" lang="it">
```

# pseudo-classi strutturali

---

Precisazione: Dato un qualsiasi oggetto all'interno del DOM, se l'oggetto contiene degli elementi figli, l'indice dei figli, contrariamente ai linguaggi di programmazione, inizia da **1** e non da **0**.

# :root

---

## **:root**

La pseudo-classe :root identifica l'elemento radice della pagina. Per pagine HTML l'elemento corrispondente è proprio html. le seguenti righe di codice sono quasi equivalenti, anche se la pseudo-classe è più specifica:

```
html {background-color: red; color: white}
```

```
:root {background-color: red; color: white}
```

# E:nth-child()

Sicuramente la pseudo-classe più importante e interessante introdotta in questa versione. E:nth-child(n) identifica l'elemento E che è l'n-esimo figlio del suo elemento padre

<table>

<tr><th>Cognome</th><th>Nome</th><th>Eta</th></tr>

<tr><td>Rossi</td><td>Mario</td><td>23</td></tr>

<tr><td>Verdi</td><td>Luca</td><td>33</td></tr>

<tr><td>Bianchi</td><td>Federica</td><td>24</td></tr>

<tr><td>Ferrari</td><td>Luana</td><td>23</td></tr>

<tr><td>Esposito</td><td>Giovanni</td><td>43</td></tr>

</table>

**tr:nth-child(4)** td {background-color: yellow}

```
<div id="id1">p1
<p>figlio di p1
<p>nipote di p1
<p>pronipote di p1</p>
</p>
<p>2 figlio di p1</p>
</div>
<p>fratello di p1</p>
<p>fratello di p1</p>
```

```
#id1 p:nth-child(1){background-color:#a8121e;}
```

importante: inserire

**contenitore selettore-figlio:nth-child()**

# E:nth-child() even=pari – odd=dispari

---

colorare in maniera alternata le righe di una tabella per migliorarne la leggibilità e l'estetica

```
tr td {background-color: #DBEEF4}
```

```
tr:nth-child(odd) {background-color: #EDF6FB}
```

```
tr:nth-child(even) {background-color: #EDF6FB}
```

```
#id1 p:nth-child(odd){background-color:#a8121e;}
```

**n = parte da 0 e prosegue fino alla fine delle occorrenze**

**tr:nth-child(odd) oppure tr:nth-child(2n+1)**  
Represents the odd rows of an HTML table: 1, 3, 5, etc.

**tr:nth-child(even) oppure tr:nth-child(2n)**  
Represents the even rows of an HTML table: 2, 4, 6, etc.

**:nth-child(7)**  
Represents the seventh element.

**:nth-child(5n)**  
Represents elements **5** [=5×1], **10** [=5×2], **15** [=5×3], **etc.**

The first one to be returned as a result of the formula is **0** [=5×0], resulting in a no-match, since the elements are indexed from 1, whereas n starts from 0. This may seem weird at first, but it makes more sense when the **n** part of the formula is >0, like in the next example.

**:nth-child(n+7)**  
Represents the seventh and all following elements: **7** [=0+7], **8** [=1+7], **9** [=2+7], **etc.**

**:nth-child(3n+4)**  
Represents elements **4** [(3×0)+4], **7** [(3×1)+4], **10** [(3×2)+4], **13** [(3×3)+4], **etc.**

**:nth-child(-n+3)**  
Represents the first three elements. [= -0+3, -1+3, -2+3]

**p:nth-child(n)**  
Represents every <p> element in a group of siblings. This selects the same elements as a simple p selector (although with a higher specificity).

**p:nth-child(1) OR p:nth-child(0n+1)**  
Represents every <p> that is the first element in a group of siblings. This is the same as the [:first-child](#) selector (and has the same specificity).

**p:nth-child(n+8):nth-child(-n+15)**  
Represents the eighth through the fifteenth <p> elements of a group of siblings.



|                                 | Chrome | Edge | Firefox | Internet Explorer | Opera | Safari | WebView Android | Chrome Android | Firefox for Android | Opera Android | Safari on iOS | Samsung Internet |
|---------------------------------|--------|------|---------|-------------------|-------|--------|-----------------|----------------|---------------------|---------------|---------------|------------------|
| :nth-child()                    | 1      | 12   | 3.5     | 9                 | 9.5 ★ | 3.1    | 37              | 18             | 4                   | 10.1 ★        | 2             | 1.0              |
| Matches elements with no parent | 57     | 79   | 52      | No                | 44    | No     | 57              | 57             | 52                  | 43            | No            | 7.0              |
| of <selector> syntax            | No ★   | No ★ | No ★    | No                | No    | 9      | No              | No             | No ★                | No            | 9             | No               |

Full support
  No support

# E:nth-last-child()

---

Funzionamento identico alla pseudo-classe precedente, E:nth-last-child(n) identifica l'elemento E che è l'n-esimo figlio del suo elemento padre partendo dall'ultimo fratello di E

```
tr:nth-last-child(2) { background-color: yellow}
```

evidenzierà il 1 e 2 td riga a partire dall'ultimo elemento fratello.

```
hgroup h1:nth-last-child(2){ /*il penultimo*/  
background-color: brown;  
}
```

1=ultimo elemento

2=penultimo elemento

.....



# E:last-child

---

Questa pseudo-classe seleziona l'elemento E che è l'ultimo figlio del suo elemento genitore.

Il funzionamento è molto semplice. Continuando a utilizzare la tabella precedente, la seguente regola CSS:

```
tr:last-child {background-color: yellow}
```

importante: last-child non ha () finali

```
#id1 p:last-child{background-color:#28c213;}
```

# E:empty

---

Questa pseudo-classe identifica ogni elemento E che non contiene figli. E:empty include anche i nodi di testo. Quindi, un semplice paragrafo che contiene del testo al suo interno non corrisponderà.

Data la regola CSS

```
p:empty {background-color: yellow}
```

dei seguenti paragrafi, solo il primo corrisponderà:

```
<h1></h1>
```

```
<h1><span></span></h1>
```

```
<h1>Lorem ipsum</h1>
```

```
h1:empty{
```

```
background-color: brown;
```

```
}
```

```
h1{width: 200px;border:1px solid black;height: 50px;}
```

# E:not

---

identifica tutti gli elementi di tipo E che non coincidono con il selettore contenuto all'interno del :not.

Cerchiamo di chiarire il concetto con qualche esempio. E partiamo da questo codice HTML:

```
<div class="nero">Lorem ipsum</div>  
<div class="rosso">Lorem ipsum</div>
```

Usando questo codice CSS:

```
div:~not(.nero) {color: red}
```

```
#id1 p:~not(.mia){background-color:#28c213;}
```

# E:enabled

---

E:enabled seleziona tutti gli oggetti di tipo E che sono abilitati all'interno di un'interfaccia utente.

Dati i seguenti campi di input

```
<input type="text" value="Nome" />
```

```
<input type="text" value="Cognome" />
```

```
<input type="text" value="Indirizzo" disabled="disabled" />
```

questo codice CSS:

```
input:enabled {color:red}
```

assegnerà il colore rosso per il testo solo ai campi abilitati, ignorando gli altri.

Ex:

```
input:enabled{background-color: red;}
```

```
<input name="nome" >
```

# E:disabled

---

Il selettore E:disabled è il contrario della pseudo-classe vista in precedenza. Esso infatti seleziona tutti gli elementi di tipo E che sono disabilitati all'interno di un'interfaccia utente.

Dato lo stesso esempio precedente, un codice come il seguente

```
input:disabled {color:red}
```

corrisponde all'esatto contrario, ovvero assegnerà un colore rosso solo ai campi disabilitati.

# E:checked

---

La pseudo-classe E:checked identifica tutti gli elementi E di un'interfaccia utente che vengono selezionati. Data la scarsa personalizzazione consentita dai browser agli elementi radio e checkbox per cui è pensato tale selettore, per l'esempio corrispondente non modificherò colori o bordi ma utilizzerò un margine per far spostare l'elemento selezionato.

Dato questo codice HTML

```
<input type="checkbox" value="Nome" />Nome  
<input type="checkbox" value="Cognome" />Cognome  
<input type="checkbox" value="Indirizzo" />Indirizzo
```

e questo codice CSS

```
input:checked {margin-left:20px}
```

```
input:checked {margin-left:50px}
```

# Pseudo-elementi ::

---

Elementi che non hanno un tag specifico ma che è possibile modificare e formattare secondo i propri desideri

si utilizzano selettore `::first.letter`

importante: funziona anche `:first-letter` (solo `:` e non `::`) per compatibilità con CSS1 e CSS2

Non funzionano con elementi inline (tipo `span`) ma solo con elementi di blocco (`p`, `h`, `div` ...)

# ::first-letter

---

è possibile formattare la **prima lettera** di qualunque elemento contenente del testo. Le proprietà modificabili sono ovviamente tutte quelle relative al carattere e al testo, ma anche quelle legate al colore, allo sfondo, ai margini, ai bordi e al padding.

```
p::first-letter {color: white; font-weight: bold;}
```

È possibile anche usare classi:

```
p.classe::first-letter {color: white; font-weight: bold;}
```

O selettori di id:

```
p#id::first-letter {color: white; font-weight: bold;}
```

```
td::first-letter{font-variant-caps: small-caps}
```



# ::first-line

---

imposta lo stile della prima riga di un elemento contenente del testo.

Sintassi ed esempi

Valgono le stesse regole generali viste per :first-letter.

```
p::first-line {color: white; }
```

```
p.classe::first-line {color: white; font-weight: bold;}
```

```
p#id::first-line {color: white; font-weight: bold;}
```

# ::before, ::after e il contenuto generato

è possibile inserire nel documento HTML un contenuto non presente nel documento stesso

Pseudo-elemento Descrizione

::before inserisce un altro elemento all'inizio del contenuto dell'elemento individuato dal selettore

::after inserisce un elemento a chiudere il contenuto dell'elemento individuato dal selettore

inserisce un numero prima di un titolo h3

```
h3::before {content: "1 "};
```

inserisce un numero dopo il titolo h4

```
h4::after {content: " 1";}
```

```
table::before {content: "ciao"; }
```

```
div::after {content: "hi";}
```

```
h3.numero::before {  
  content: "1 ";  
  display: inline-block;  
  width: 30px;  
  height: 30px;  
  background: red;  
  padding: 3px;  
  margin-right: 5px;  
  color: white;  
  font-size: 22px;  
}
```

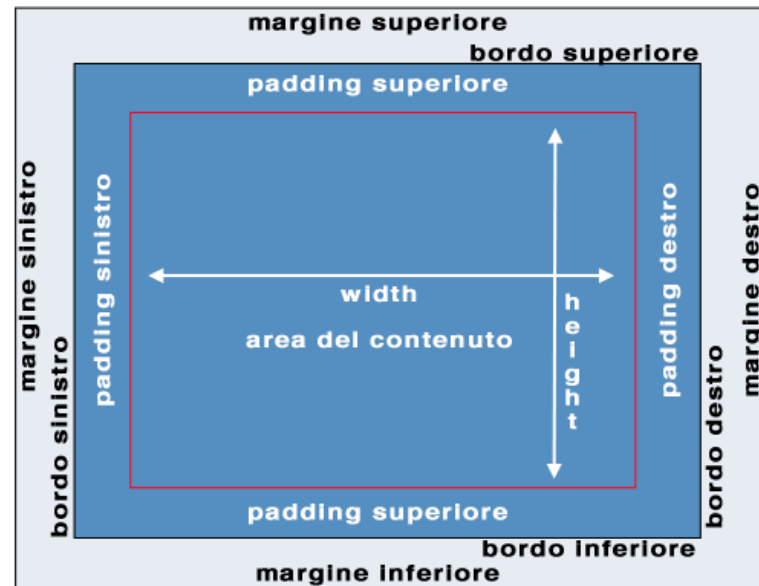
```
input:checked::before {background-color: #c21365; content: "ciao"}
```

# Box Model

## Componenti del box model

Tutto l'insieme di regole che gestisce l'aspetto visuale degli **elementi blocco** viene in genere riferito, appunto, al cosiddetto box model.

Ogni box comprende un certo numero di componenti di base, ciascuno modificabile con proprietà dei CSS. La figura qui sotto mostra visivamente tali componenti:



# Box Model

Larghezza del box

---

Bisogna distinguere tra tre concetti:

la larghezza dell'area del contenuto;

la larghezza complessiva;

la larghezza dell'area visibile.

La **prima** è data dal valore della proprietà **width**.

La **seconda** corrisponde allo spazio occupato sulla pagina compresi i margini ed è data da questa somma:

margine sinistro + bordo sinistro + padding sinistro

+ area del contenuto

+ padding destro + bordo destro + margine destro

La **terza** corrisponde allo spazio occupato sulla pagina esclusi i margini, parliamo insomma della parte del box delimitata dai bordi e a cui si può applicare uno sfondo. È data da questa somma:

+ padding destro + bordo destro

+ area del contenuto

+ padding sinistro + bordo sinistro



# auto

---

Solo per tre proprietà è possibile impostare il valore auto: margini (**margin**), altezza (**height**) e larghezza (**width**). L'effetto dell'uso di auto è quello di lasciar **calcolare al browser l'ammontare del valore** per ciascuna di queste proprietà

Solo i margini possono avere **valori negativi**. Ciò non è consentito per padding, bordi, altezza e larghezza.

# Margini verticali e orizzontali tra gli elementi

---

Per due box adiacenti **in senso verticale** che abbiano impostato un margine inferiore e uno superiore **la distanza NON sarà data dalla somma delle due distanze**. **A prevalere sarà invece la distanza maggiore tra le due**. È il meccanismo del cosiddetto **margin collapsing**.

Le regole sono uguali se i div sono uno sotto l'altro:

```
#my {margin-bottom: 50px}
```

```
#my2 {margin-top: 50px}
```

```
/*#my {margin-bottom: 50px}*/
```

```
#my2 {margin-top: 50px}
```

# La proprietà height

---

definisce la distanza tra il bordo superiore e quello inferiore di un elemento. **Non è ereditata** e si applica a tutti gli elementi tranne:

colonne di tabelle;

elementi inline non rimpiazzati.

---

*Es uno span (elemento inline non rimpiazzato) non prenderà altezza in questo caso*

```
.span {  
    height: 500px;  
    border: 1px solid blue;  
}
```

*Ma se aggiungo display: block alla regola lo prenderà*

---

Il valore può essere espresso da:

un valore numerico con unità di misura;

un valore in percentuale: il valore in percentuale è sempre definito rispetto all'altezza del blocco contenitore, purché esso abbia un'altezza esplicitamente dichiarata; diversamente, la percentuale viene interpretata come auto;

Se nessun contenitore padre avrà una height impostata la height in % verrà calcolata sul viewport  
auto: l'altezza sarà quella determinata dal contenuto.

```
div {height: 250px;}
```

```
ul {height: 50%;}
```

```
p {height: auto;}
```



# La proprietà min-height

---

Imposta un'altezza minima per un elemento. Valgono per questa proprietà le stesse osservazioni fatte per height relativamente al contenuto. Non è ereditata.

## Sintassi ed esempi

```
selettore {min-height: valore;}
```

I valori possono essere:

un valore numerico con unità di misura;

un valore in percentuale.

```
div {min-height: 200px;}
```

```
p {min-height: 30%;}
```

# La proprietà max-height

---

La proprietà max-height serve a impostare l'altezza massima di un elemento. Anche per essa valgono le osservazioni già fatte per il contenuto eccedente. Non è ereditata.

## IMPORTANTE

Sintassi ed esempi

selettore {max-height: valore;}

Per i valori possiamo ricorrere a:

**none:** valore iniziale e di default, l'altezza dell'elemento non è limitata;

un valore numerico con unità di misura;

un valore in percentuale.

div {max-height: 400px;}

p {max-height: 40%;}

form {max-height: none;}

# La proprietà overflow

---

fornisce un modo per **gestire il contenuto che superi i limiti imposti con height**.  
Serve infatti per definire il comportamento di un **elemento blocco nel caso il suo contenuto ecceda dalle sue dimensioni esplicite**

selettore {overflow : valore;}

I valori possono essere espressi con le parole chiave:

visible: valore iniziale, il contenuto eccedente rimane visibile;

hidden: il contenuto eccedente non viene mostrato;

scroll: il browser crea barre di scorrimento che consentono di fruire del contenuto eccedente;

auto: **il browser tratta il contenuto eccedente secondo le sue impostazioni predefinite; di norma dovrebbe mostrare una barra di scorrimento laterale.**

initial: imposta la dimensione del contenitore al contenuto

div {overflow: auto;}

p {overflow: hidden;}

div {overflow: visible;}

p {overflow: scroll;}

# width

---

Con la proprietà width, dunque, impostiamo la larghezza dell'area del contenuto di un box, esclusi padding e bordi.

selettore: {width: valore;}

Il valore per width può corrispondere a:

auto: **valore iniziale e di default**; se non si impostano margini, bordi e padding la larghezza dell'elemento sarà uguale all'area del contenuto dell'elemento contenitore;

un valore numerico con unità di misura;

un valore in percentuale: la larghezza sarà calcolata rispetto a quella dell'elemento contenitore.

La proprietà width non è ereditata.

```
div {width: auto;}
```

```
p {width: 90px;}
```

```
div.box {width: 50%;}
```

# La proprietà min-width

---

Imposta la larghezza minima di un elemento. Si applica a tutti gli elementi, tranne a quelli in linea non rimpiazzati e agli elementi di tabelle. Proprietà non ereditata.

Sintassi ed esempi

```
selettore {min-width: valore;}
```

I valori possono essere:

un valore numerico con unità di misura;

un valore in percentuale: la larghezza sarà come minimo quella espressa dalla percentuale riferita alla larghezza dell'elemento contenitore.

```
div {min-width: 400px;}
```

```
p {min-width: 40%;}
```

# La proprietà max-width

---

Imposta la larghezza massima di un elemento. Non è ereditata.  
l'elemento può assumere una larghezza inferiore rispetto al valore impostato ma non un valore superiore

Sintassi ed esempi

selettore {max-width: valore;}

Per quanto riguarda i valori, essi possono essere rappresentati da:

none: valore di default, non c'è un limite per larghezza dell'elemento;

un valore numerico con unità di misura;

un valore in percentuale.

div {max-width: 400px;}

p {max-width: 40%;}

# Nota max-width

---

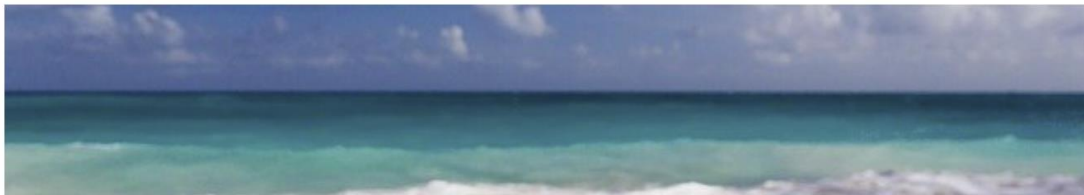
Una immagine per essere responsive andrebbe utilizzata la proprietà max-width: 100% e height: auto

Questo permette di avere una immagine con si adatta al 100% del contenitore e che non perde di qualità se contenitore troppo grande

con max-width



con width 100%



# Overflow

## overflow: visible

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

## overflow: hidden

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea

## overflow: scroll

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud

```
#box2 {  
  background: #4c74be;  
  width: 300px;  
  height: 200px;  
  padding: 30px;  
  margin-bottom: 40px;  
  overflow-x: scroll;  
}
```

## overflow-x: scroll

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquipgrtrytfrewuffrfigreigreigrekgreikeriieigreir ex ea commodo consequat.



# margin-top, left, right, bottom

---

Imposta la distanza tra il lato i un elemento e gli elementi adiacenti. Si applica a tutti gli elementi e non è ereditata.

selettore {margin-top: valore;}

I valori possibili sono:

un valore numerico con unità di misura: il valore è espresso in termini assoluti;

un valore in percentuale: il valore è calcolato come percentuale rispetto alla larghezza (width) del blocco contenitore;

auto: il browser calcola automaticamente la distanza.

div {margin-top: 20px;}

p {margin-top: 10%;}

img {margin-top: auto;}

Oppure margin-left ecc...

# margin

---

È una proprietà a sintassi abbreviata. Con essa è possibile specificare in una sola regola i valori per tutti e quattro i lati di un elemento. Si applica a tutti gli elementi e non è ereditata. I tipi di valori esprimibili sono gli stessi visti per le proprietà singole.

## Sintassi ed esempi

La sintassi di base per questa proprietà è la seguente:

selettore {margin: valore-1, valore-2, valore-3, valore-4;}

L'ordine di lettura va inteso in senso orario. Per cui: il primo valore si riferisce al lato superiore, il secondo a quello destro, il terzo al lato inferiore, il quarto a quello sinistro. In pratica, usare la sintassi vista nell'esempio equivale a scrivere:

```
div {  
  margin-top: 10px;  
  margin-right: 15px;  
  margin-bottom: 10px;  
  margin-left: 20px;  
}
```

# margin

---

Un'ulteriore abbreviazione della sintassi si può ottenere usando tre, due o un solo valore. Queste le regole:

se si usano **tre valori**, il primo si riferisce al margine superiore, il secondo a quelli sinistro e destro, il terzo a quello inferiore;

se si usano **due valori**, il primo si riferisce ai lati superiore e inferiore, il secondo al sinistro e al destro;

se si usa **un solo valore**, un uguale distanza sarà applicata ai quattro lati.

# padding-top, left, right, bottom

---

Imposta l'ampiezza del padding sul lato superiore di un elemento. Si applica a tutti gli elementi e non è ereditata

selettore {padding-top: valore;}

I valori possono essere:

un valore numerico con unità di misura;

un valore in percentuale calcolato come percentuale rispetto alla larghezza (width) del blocco contenitore.

div {padding-top: 40px;}

p {padding-top: 20%;}

Oppure padding-left ecc...

# padding

---

Proprietà a sintassi abbreviata. Serve a impostare i valori del padding per tutti e quattro i lati di un elemento. Valgono per essa tutte le osservazioni e le regole sintattiche viste per la proprietà margin. **il valore è calcolato come percentuale rispetto alla larghezza (width) del blocco contenitore**

Sintassi ed esempi

La sintassi di base per questa proprietà è la seguente:

selettore {padding: valore-1, valore-2, valore-3, valore-4;}

I valori possono essere:

- un elenco di valori numerici con unità di misura;

- un elenco di valori in percentuale.

Nella definizione dei valori è possibile mischiare percentuali con valori assoluti in unità di misura.

# padding

---

Un'ulteriore abbreviazione della sintassi si può ottenere usando tre, due o un solo valore. Queste le regole:

se si usano **tre valori**, il primo si riferisce al padding superiore, il secondo a quelli sinistro e destro, il terzo a quello inferiore;

se si usano **due valori**, il primo si riferisce ai lati superiore e inferiore, il secondo al sinistro e al destro;

se si usa **un solo valore**, un uguale distanza sarà applicata ai quattro lati

# Definire lo stile di un singolo bordo

---

Iniziamo a vedere come impostare le proprietà per un singolo bordo. Questa la sintassi di base con le proprietà singole:

```
selettore {  
  border-<lato>-color: <valore>;  
  border-<lato>-style: <valore>;  
  border-<lato>-width: <valore>;  
}
```

E questa la sintassi abbreviata:

```
selettore {  
  border-<lato>: <valore width>  
    <valore style>  
    <valore color>;  
}
```

# Border: lato e valori

---

In entrambi gli esempi di sintassi sostituite a **<lato>** uno degli indicatori dei quattro lati: **top, right, bottom o left**

Per quanto concerne i valori, come si vede dall'elenco delle proprietà, di ciascun lato si possono definire per il bordo tre aspetti:

- il colore (color);
- lo stile (style);
- lo spessore (width).



# Border

---

**-color**

**-style**

**-width**

In linea di massima possiamo suddividere le proprietà relative ai bordi in due categorie: **proprietà singole** e **proprietà a sintassi abbreviata**

Sono proprietà singole:

border-top-color, border-top-style, border-top-width, border-bottom-color, border-bottom-style, border-bottom-width, border-right-color, border-right-style,

border-right-width, border-left-color, border-left-style, border-left-width

Sono proprietà a sintassi abbreviata:

border, border-bottom, border-top, border-right, border-left, border-color, border-style, border-width

# border-color, border-width

---

I valori possibili per il **color** sono:

un qualsiasi colore;  
la parola chiave inherit.

il **width**. Esso può essere modificato secondo i seguenti valori:

**un valore numerico con unità di misura;**

**thin:** bordo sottile;

**medium:** bordo di spessore medio;

**thick:** bordo di spessore largo.

# border-style

---

Lo stile di un bordo può invece essere espresso con una delle seguenti parole chiave:

| Stile bordo | Descrizione  |
|-------------|--|
| none        | l'elemento non presenta alcun bordo e lo spessore equivale a 0 |
| hidden      | equivalente a none   |
| dotted      | bordo a puntini  |
| dashed      | bordo a lineette   |
| solid       | bordo solido e continuo  |
| double      | bordo solido, continuo e doppio                                |
| groove      | tipo di bordo in rilievo                                       |
| ridge       | altro tipo di bordo in rilievo                                 |
| inset       | effetto 'incastonato'  |
| outset      | effetto 'sbalzato'   |

# Esempio border

---

Come scrivere, dunque, una regola per impostare uno solo dei bordi? Si può fare così, usando le proprietà singole:

```
div {  
  border-left-color: black;  
  border-left-style: solid;  
  border-left-width: 1px;  
}
```

Ma è molto più comodo scrivere così, facendo ricorso alla proprietà a sintassi abbreviata:

```
div {border-left: 1px solid black;}
```

# Stili per tutti e 4 i bordi

---

Se si vogliono impostare stili per tutti e quattro i bordi del box, si hanno ancora una volta due opzioni. La prima è da usare quando si vogliono impostare insieme i quattro bordi ma si vuole assegnare a ciascuno uno stile diverso in quanto a colore, spessore, stile:

```
selettore {  
  border-width: <valori>;  
  border-style: <valori>;  
  border-color: <valori>;  
}
```

```
div {  
  border-width: 1px 2px 1px 2px;  
  border-style: solid;  
  border-color: black red black red;  
}
```

# Usare la proprietà border

---

L'ultima proprietà a sintassi abbreviata è border. Con essa possiamo definire con una sola regola le impostazioni per i quattro bordi. Il suo uso è però limitato a un solo caso, peraltro molto comune: che i quattro bordi abbiano tutti lo stesso colore, lo stesso stile e lo stesso spessore.

Questa la sintassi:

```
selettore {  
  border: <valore spessore>  
  <valore stile>  
  <valore colore>;  
}
```

Che tradotto in codice reale diventa:

```
div {border: 2px solid black;}
```

# La proprietà outline

---

La cosiddetta 'outline' è una sorta di bordo che è possibile inserire attorno ad oggetti di un documento per evidenziarli. Se impostata, l'outline si colloca esternamente rispetto al bordo definito per l'elemento.

È possibile definire l'aspetto dell'outline secondo le seguenti proprietà:

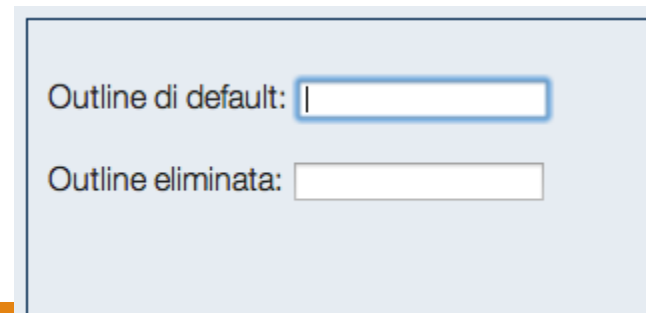
outline-color il colore

outline-style lo stile

outline-width: lo spessore.

Per eliminarla basterà una regola siffatta:

```
input, select,  
textarea, button {  
  outline: none;  
}
```



Outline di default:

Outline eliminata:

```
input {outline: none;}
```

# Border-radius

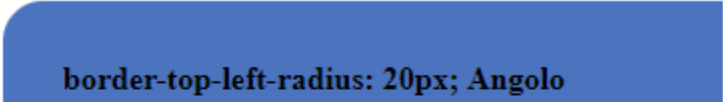
---

si possono realizzare in maniera semplice e intuitiva **angoli arrotondati**  
Vediamo nei dettagli come la specifica definisce questa funzionalità.

Le proprietà coinvolte sono cinque:

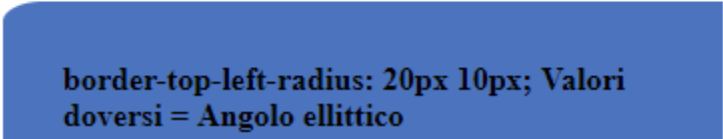
border-top-left-radius  
border-top-right-radius  
border-bottom-right-radius  
border-bottom-left-radius  
border-radius

#box1 {border-top-left-radius: 20px}



**border-top-left-radius: 20px; Angolo**

#box2 {border-top-left-radius: 20px 10px}



**border-top-left-radius: 20px 10px; Valori  
diversi = Angolo ellittico**

Se si definiscono due valori diversi, il primo imposta la misura del raggio orizzontale, il secondo quello del raggio verticale

Per concludere un esempio con quattro valori:

#box {border-radius: 20px 40px 60px 80px (top-left, top-right, bottom-right, bottom-left)}



# Colori

---

## Definizione dei colori

I colori possono essere espressi in vari modi nel contesto di una regola CSS.

## Parole chiave

Si tratta di sedici keyword che definiscono i colori della palette VGA standard di Windows:

Figura 1 – I sedici colori della palette VGA standard



Ecco alcuni

black | navy | blue | maroon | purple | green | red | teal | fuchsia |  
olive | gray | lime | aqua | silver | yellow | white

# Notazione esadecimale: #RRGGBB

---

dunque possibile impostare il colore di un elemento servendosi di codici con notazione esadecimale. In essi, le prime due lettere (o numeri) corrispondono ai valori per il colore rosso (RED), la seconda coppia fa riferimento al verde (GREEN), l'ultima al blue (BLUE). Il codice va preceduto dal simbolo del cancelletto (#).

Un esempio:

color: #CC0000

# RGB

---

## Notazione decimale con RGB

Un altro modo per rappresentare i colori è quello di usare per i tre elementi base del sistema RGB una lista di valori separati da una virgola. I valori possono essere espressi in percentuale (da 0% a 100%) o con una scala che va da 0 (il nero) a 255 (il bianco).

Per indicare il nero useremo, ad esempio:

```
color: rgb(0%, 0%, 0%);
```

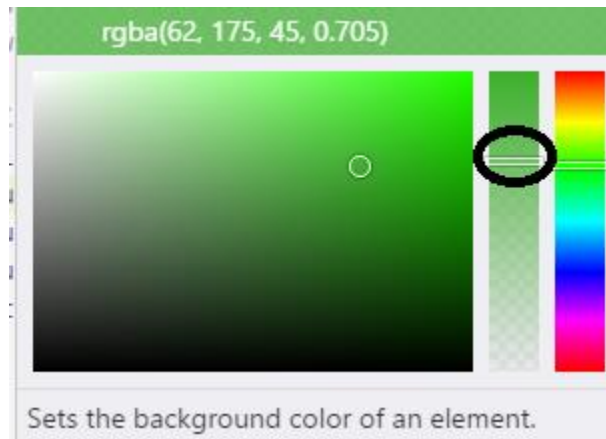
```
color: rgb(0, 0, 0);
```

# Colore RGBa

---

i CSS3 introducono una novità di sicuro interesse. Si tratta della definizione del colore attraverso una notazione **RGBa**: Rosso (Red), Verde (Green), Blue (Blue) e Alfa

body {background-color: **rgba**(255, 255, 255, 0.5)}



# La proprietà color

---

Visti i sistemi per rappresentare i colori, dobbiamo ora chiarire un aspetto importante. Per ogni elemento si possono definire almeno tre colori:

il colore di **primo piano**;

il colore di **sfondo**;

il colore del **bordo**;

La proprietà color definisce esclusivamente:

il colore di primo piano, ovvero quello del testo;

il colore del bordo di un elemento quando non si imposti esplicitamente quest'ultimo con le proprietà border o border-color.

```
p {color: black;}
```

```
div {color: #CC0000;}
```

# Gestione dello sfondo

---

La lista delle proprietà per lo sfondo, applicabili, ed è questa la prima grande innovazione dei CSS, a tutti gli elementi:

background-color

background-image

background-repeat

background-attachment

background-position

# background-color

---

**Definisce il colore di sfondo di un elemento.** Questa proprietà non è ereditata.

## Sintassi

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

## Valori

un qualunque colore

la parola chiave transparent

Usando transparent come valore un elemento avrà come colore quello dell'elemento parente.

```
body { background-color: white; }
```

```
p { background-color: #FFFFFF; }
```

```
.classe1 { background-color: rgb(0, 0, 0)
```

# background-image

---

**Definisce l'URL di un'immagine da usare** come sfondo di un elemento. Questa proprietà non è ereditata.

## Sintassi

```
selettore { background-image: url(valore); }
```

## Valori

un URL assoluto o relativo che punti ad un'immagine

la parola chiave none. Valore di default.

```
body {background-image: url(sfondo.gif); }
```

```
div body {background-image: url(http://www.server.it/images/sfondo.gif); }
```



# background-repeat

---

consente di definire la direzione in cui l'immagine di sfondo viene ripetuta.  
Proprietà non ereditata.

## Sintassi

selettore {background-repeat: valore;}

## Valori

repeat. L'immagine viene ripetuta in orizzontale e verticale. È il comportamento standard.

repeat-x. L'immagine viene ripetuta solo in orizzontale.

repeat-y. L'immagine viene ripetuta solo in verticale.

no-repeat. L'immagine non viene ripetuta.

# background-attachment

---

si imposta il comportamento **dell'immagine di sfondo rispetto all'elemento cui è applicata e all'intera finestra del browser**. Si decide, in pratica, se essa deve **scorrere insieme al contenuto o se deve invece rimanere fissa**. Proprietà non ereditata.

## Sintassi

selettore {background-attachment: valore;}

## Valori

scroll. L'immagine scorre con il resto del documento quando si fa lo scrolling della pagina

fixed. L'immagine rimane fissa mentre il documento scorre (esempio)

```
body { background-image: url(back_400.gif);  
background-repeat: repeat-x;  
background-attachment: fixed; }
```

# background-position

Definisce il punto in cui verrà piazzata un'immagine di sfondo **non ripetuta** o da dove inizierà la ripetizione di una ripetuta. Si applica solo agli elementi blocco o rimpiazzati

selettore {background-position: valoreOriz | valoreVert;}

I valori specificano le coordinate di un punto sull'asse verticale e su quello orizzontale e possono essere espressi con diverse unità di misura e modalità:

*x% y%*

*xpos ypos*

left top

left center

left bottom

right top

right center

right bottom

center top

center center

center bottom

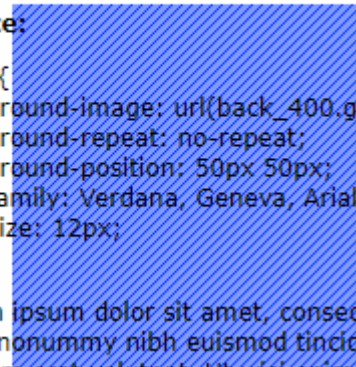
```
body {  
  background-image:  
url(back_400.gif);  
  background-repeat: no-  
repeat;  
  background-position:  
50px 50px;  
}
```

**Background-position:**

Codice:

```
body {  
background-image: url(back_400.gif);  
background-repeat: no-repeat;  
background-position: 50px 50px;  
font-family: Verdana, Geneva, Arial;  
font-size: 12px;  
}
```

Lorem ipsum dolor sit amet, consectetur  
diam nonummy nibh euismod tincidunt



# background-size

Definisce la dimensione dell'immagine di background

I valori possono essere espressi con diverse unità di misura e modalità:

con valori in percentuale

**cover:** scala l'immagine più grande possibile per riempire il contenitore, ne fa vedere anche solo una parte

**contain:** scala l'immagine più grande possibile per riempire il contenitore, ma la mostra sempre intera



# background

---

possiamo definire in un colpo solo tutti gli aspetti dello sfondo. Per essere valida, la dichiarazione non deve contenere necessariamente riferimenti a tutte le proprietà viste finora, ma deve contenere almeno la definizione del colore di sfondo.

## Sintassi

background:

*bg-color*

*bg-image position/bg-size*

*bg-repeat*

*bg-origin*

*bg-clip* (<https://developer.mozilla.org/en-US/docs/Web/CSS/background-clip>

*border-box, padding-box, content-box, text*)

*bg-attachment* ;

```
body {
```

```
    background: #00ff00 url("smiley.gif") no-repeat fixed center;
```

```
}
```

# object-fit

---

La proprietà CSS object-fit imposta come il contenuto di un elemento sostituito, come un `<img>` o un `<video>`, dovrebbe essere ridimensionato per adattarsi al suo contenitore.

Possibili valori

object-fit: contain;

object-fit: cover;

object-fit: fill;

object-fit: none;

object-fit: scale-down;



contain



fill



cover



none



scale-down

# object-position

La proprietà CSS object-position specifica l'allineamento del contenuto dell'elemento sostituito selezionato all'interno della casella dell'elemento. **Le aree del riquadro che non sono coperte dall'oggetto dell'elemento sostituito mostreranno lo sfondo dell'elemento.**

```
#banner{  
  height: 400px;  
}  
#banner img{  
  object-fit: cover;  
  object-position: 100% 100%; /*se img height 800px*  
  width: 100%;  
  height: 100%;  
}
```



object-fit: cover;



object-fit: cover;

+

object-position: 100% 100%;

object-position: 50% 50%;



object-position: left bottom;



object-position: right top;



object-position: 250px 125px;

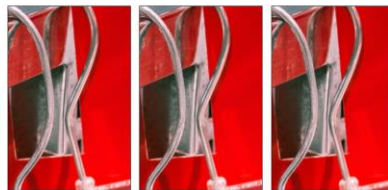


(fuori dal box)

# object position

---

object-position: 30% 10%;



object-position: 50% 50%;



```
<div id="gallery">
  <div>
    
  </div>
  <div>
    
  </div>
  <div>
    
  </div>
</div>

#gallery {
  margin: 10px auto;
  display: block;
  width: 80%;
}
#gallery div {
  width: 200px;
  height: 300px;
  display: inline-block;
  margin-right: 10px;
  border: 1px solid black;
}
#gallery img {
  width: 100%;
  height: 100%;
  object-fit: none;
  object-position: 30% 10%;
}
```



# Sprite e CSS

Generatore: <http://css.spritegen.com/>

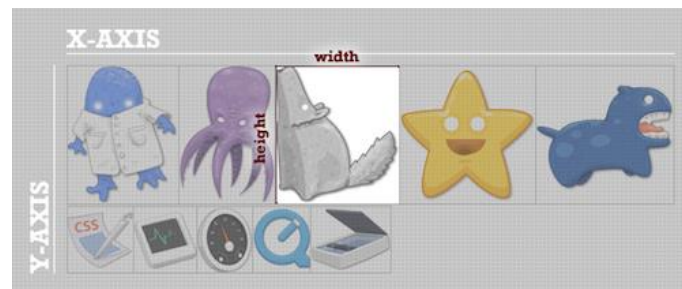
Gli sprites CSS sono un modo per combinare insieme delle immagini al fine di ottimizzare il caricamento della pagina riducendo il numero di richieste HTTP al server

1 l'immagine di base



2 creare il rivelatore del nostro sprite

```
.sprite {background:url(..images/mySprite.png);}
```



# Sprite e CSS

---

```
.sprite
{background:url(..images/mySprite
.png);}
    .monster {height:128px;}
    .application {height:61px;}
        /* Monsters */
        .doctor {width:103px;
background-position:-2px -2px;}
        .octopus {width:89px;
background-position:-106px -2px;}
        .wolf {width:115px;
background-position:-196px -2px;}
        .star {width:126px; background-
position:-312px -2px;}
        .dog {width:128px; background-
position:-439px -2px;}
```

```
.css {width:61px; background-
position:-2px -133px;}
    .activityMonitor {width:58px;
background-position:-64px -
133px;}
    .dashboard {width:51px;
background-position:-123px -
133px;}
    .quicktime {width:53px;
background-position:-175px -
133px;}
    .scanner {width:74px;
background-position:-229px -
133px;}
```

# Sprite: il codice HTML

---

```
  

```

```
  

```

```
  

```

```
  

```

```
  

```

# La proprietà display

---

Il valore può essere rappresentato unicamente da una parola chiave. Nella pratica comune,

**block** l'elemento viene reso come un elemento blocco

**inline** l'elemento a cui viene applicata assume le caratteristiche degli elementi inline

**inline-block** l'elemento può assumere, come gli elementi blocco, dimensioni esplicite (larghezza e altezza), margini e padding, ma come tutti gli elementi inline, si disporrà orizzontalmente e non verticalmente, potendo essere circondato dal testo ed essendo sensibile all'allineamento verticale

**none** l'elemento non viene mostrato; o meglio: è come se non fosse nemmeno presente nel documento, in quanto non genera alcun box; l'uso del valore none è uno dei mezzi con cui, nei CSS, si può nascondere un elemento

**display: inline-block**

Abbiamo un [link](#) dichiarato inline-block.

Posso assegnare larghezza, margini, bordi, padding.

# Float

---

float

Con questa proprietà è possibile **rimuovere un elemento dal normale flusso del documento** e **spostarlo su uno dei lati (destro o sinistro) del suo elemento contenitore**

selettore {float: valore;}

float può assumere questo valori:

| Valore | Descrizione |
|--------|-------------|
|--------|-------------|

|      |   |
|------|---|
| left | l'elemento viene spostato sul lato sinistro del box contenitore, il contenuto scorre a destra |
|------|---|

|       |   |
|-------|---|
| right | l'elemento viene spostato sul lato destro, il contenuto scorre a sinistra |
|-------|---|

|      |   |
|------|---|
| none | valore iniziale e di default in mancanza di una dichiarazione esplicita; l'elemento mantiene la sua posizione normale |
|------|---|

```
img {  
  float: left;  
}
```

[https://www.w3schools.com/cssref/tryit.asp?filename=trycss3\\_align-content](https://www.w3schools.com/cssref/tryit.asp?filename=trycss3_align-content)

# float

---

elementi float vengono resi automaticamente **block-level**: questo significa che si può attribuire loro una larghezza e/o un'altezza via CSS.

gli elementi float vengono **traslati dal flusso degli elementi di pagina verso uno dei due estremi del loro contenitore**

, e che elementi adiacenti nel codice “sentono” la loro presenza, regolandosi di conseguenza. Ciò nonostante, gli *antenati* degli elementi float, ossia i loro contenitori, li ignorano.

un elemento float assumerà la larghezza massima per il suo contenuto

# Distanziare i float

---

Nella maggior parte dei casi, soprattutto per i layout a due o tre colonne, è importante evitare che un elemento si mantenga distanziato da un lato del suo contenitore anche quando termina l'effetto di uno dei float adiacenti. Ci sono tre modi ottenere un effetto simile:

- Usare i margini.
  - Specificare la larghezza dell'elemento adiacente al float.
  - Usare i float opposti.
- 
- tratto da:
  - <https://www.html.it/articoli/float-teoria-e-pratica/>

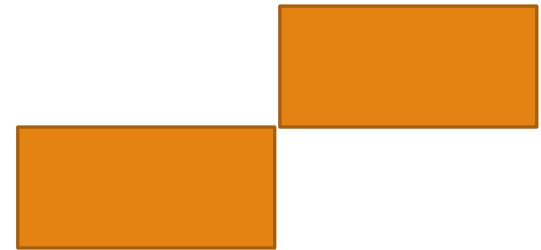
# clear

---

La proprietà clear serve a impedire che al fianco di un elemento compaiano altri elementi con il float. Si applica solo agli elementi blocco e non è ereditata

selettore {clear: valore;}

I valori possibili sono:



| Valore | Descrizione |
|--------|-------------|
|--------|-------------|

|       |   |
|-------|---|
| none  | gli elementi con float possono stare a destra e sinistra dell'elemento; |
| left  | si impedisce il posizionamento a sinistra di altri elementi;            |
| right | si impedisce il posizionamento a destra di altri elementi;              |
| both  | si impedisce il posizionamento su entrambi i lati.                      |



# position

---

position è la proprietà fondamentale per la gestione della posizione degli elementi: determina la modalità di presentazione di un elemento sulla pagina. Si applica a tutti gli elementi e non è ereditata.

selettore {position: valore;}

I valori con cui è possibile definire la modalità di posizionamento sono quattro:

static;

relative;

absolute;

fixed

sticky

# position: static

---

È il valore di default, quello predefinito per tutti gli elementi non posizionati secondo un altro metodo. **Rappresenta la posizione normale che ciascuno di essi occupa nel flusso del documento.**

[https://www.w3schools.com/cssref/playit.asp?filename=playcss\\_position&preval=relative](https://www.w3schools.com/cssref/playit.asp?filename=playcss_position&preval=relative)

# position: relative

---

L'elemento viene posizionato **relativamente al suo box contenitore**. La posizione viene impostata con le proprietà top, left, bottom o right

# position: absolute

---

L'elemento, o meglio, il box dell'elemento, viene rimosso dal flusso del documento ed è posizionato in base ai valori forniti con le proprietà top, left, bottom o right.

Il posizionamento assoluto (position: absolute;) avviene sempre rispetto **al box contenitore dell'elemento**. Questo è rappresentato dal **primo elemento antenato (ancestor) che abbia un posizionamento diverso da static**

Se tale elemento non esiste il posizionamento assoluto avviene in base all'elemento radice html

```
#box-1 {position: relative;}  
#box-2 {  
  position: absolute;  
  top: 0;  
  left: 20px  
}
```

# position: fixed

---

Usando questo valore, il box dell'elemento viene, come per absolute, sottratto al normale flusso del documento. La differenza sta nel fatto che per fixed il box contenitore è sempre la cosiddetta **viewport**. Con questo termine si intende **la finestra principale del browser**, ovvero l'area del contenuto. Altra differenza fondamentale: un box posizionato con fixed non scorre con il resto del documento. Rimane, appunto, fisso al suo posto

# position: sticky

Usando questo valore, il box dell'elemento vi segue come per relative il flusso del documento, ma si blocca come se fosse fixed quando arriva alla posizione dello user scroll

```
<h1 style="text-align: center;">Questo il Titolo</h1>
<div id="menu-top">
  <nav id="top">
    <ul>
      <li>
        <a href="#top">Top</a>
      </li>
      <li>
        <a href="#offerte">Offerte</a>
      </li>
    </ul>
  </nav>
</div>
#menu-top{
  background-color: green;
  width: 100%;
  height: 50px;
  position: sticky;
  position: -webkit-sticky;
  top: 0px;
  z-index: 4;
}
```

Questo il Titolo

[Top](#) [Offerte](#)

[Top](#) [Offerte](#)

Messaggio Principale

le tue vacanze o il tuo week end arrivano all'ultimo minuto, prenotare un hotel a Rimini non

Scopri di più

# Impostare la posizione: top

---

Come si accennava nella lezione precedente, il significato di top cambia secondo la modalità di posizionamento.

Per gli elementi posizionati con absolute o fixed definisce la distanza verticale rispetto al bordo superiore dell'elemento contenitore. Per gli elementi posizionati con relative stabilisce invece l'ammontare dello spostamento rispetto al lato superiore della posizione originari

I valori possono essere:

un valore numerico con unità di misura;

un valore in percentuale: la percentuale è relativa all'altezza dell'elemento contenitore;

auto.

```
div {top: 10px;}
```

```
p {top: 10%;}
```

# left

---

## left

Per gli elementi con posizione assoluta o fissa definisce la distanza dal bordo sinistro del box contenitore

## Sintassi ed esempi

selettore {left: valore;}

I valori possono essere:

un valore numerico con unità di misura;

un valore in percentuale: la percentuale è relativa alla larghezza dell'elemento contenitore;

auto.

div {left: 10px;}

p {left: 10%;}



# bottom

---

Per i box con posizione assoluta o fissa definisce la distanza dal bordo inferiore dell'elemento contenitore. Per quelli posizionati relativamente lo spostamento rispetto al lato inferiore della posizione originaria.

## Sintassi ed esempi

selettore {bottom: valore;}

I valori possono essere:

un valore numerico con unità di misura;

un valore in percentuale: la percentuale è relativa all'altezza dell'elemento contenitore;

auto.

div {bottom: 10px;}

p {bottom: 10%;}

# right

---

Per i box con posizione assoluta o fissa definisce la distanza dal bordo destro dell'elemento contenitore. Per quelli posizionati relativamente lo spostamento rispetto al lato destro della posizione originaria.

## Sintassi ed esempi

selettore {right: valore;}

I valori possono essere:

un valore numerico con unità di misura;

un valore in percentuale: la percentuale è relativa alla larghezza dell'elemento contenitore;

auto.

div {right: 10px;}

p {right: 10%;}

# visibility

---

Una nota fondamentale riguarda la differenza tra l'uso di `visibility` e della dichiarazione `display: none` per nascondere un elemento. Usando `visibility`, l'elemento non viene rimosso dal flusso del documento. Significa che, pur essendo invisibile, il box che genera occupa comunque lo spazio dettato dalle sue dimensioni. Potremmo dire, semplificando, **c'è ma non si vede**. Con `display: none`, invece, come abbiamo visto, l'elemento viene rimosso dal flusso del documento e non occupa alcuno spazio sulla pagina.

## Sintassi ed esempi

selettore {visibility: valore}

I valori possibili sono:

`visible`: valore iniziale e di default, l'elemento è visibile;

`hidden`: l'elemento è nascosto, ma mantiene il suo posto nel layout dove apparirà come una zona vuota;

`collapse`: usato solo per elementi di tabella (righe, colonne, celle).

`div {visibility: visible;}`

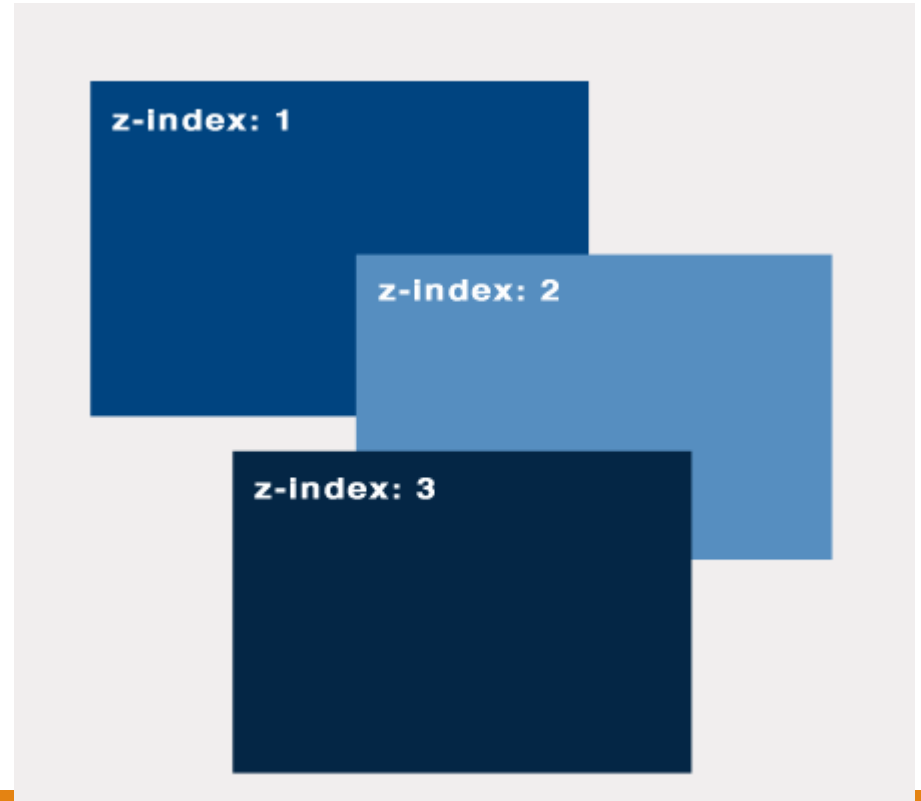
`p {visibility: hidden;}`

`td {visibility: collapse;}`

# La proprietà z-index

---

Con essa si imposta l'ordine di posizionamento dei vari elementi sulla base di una scala di livelli.



# Gestione del testo

---

La gestione del testo e della tipografia è un aspetto essenziale dei CSS. Le proprietà che definiscono il modo in cui il testo appare sullo schermo sono tante e abbiamo deciso di suddividere l'argomento in due lezioni. Iniziamo quindi dalle proprietà di base

- il font da usare;
- la sua dimensione;
- la sua consistenza
- l'interlinea tra le righe;
- l'allineamento del testo:
- la sua decorazione (sottolineature, etc.).

# Font-family

---

## font-family

La proprietà font-family serve a impostare il tipo di carattere tipografico per una qualunque porzione di testo. Si applica a tutti gli elementi ed è ereditata.

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

Quando la pagina viene caricata, il browser tenterà di usare il primo font della lista. Se questo non è disponibile sul dispositivo dell'utente userà il secondo. In mancanza anche di questo, verrà utilizzato il font principale della famiglia sans-serif presente sul sistema

serif (Times New Roman);

sans-serif (Arial);

cursive (Comic Sans);

fantasy (Allegro BT);

monospace (Courier).

# usare google font

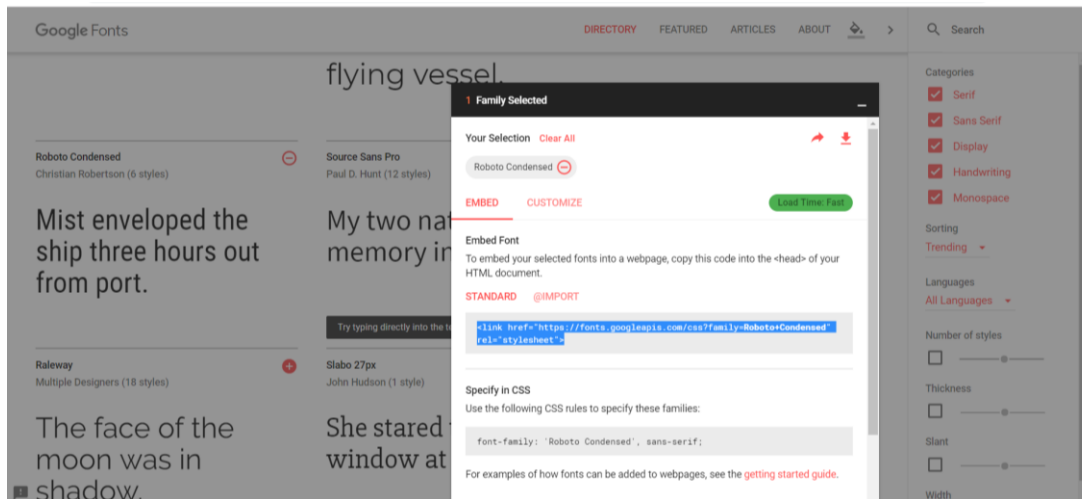
andare su google font

inserire il link

```
<link href="https://fonts.googleapis.com/css?family=Roboto+Condensed" rel="stylesheet">
```

usare la regola:

```
font-family: 'Roboto Condensed', sans-serif;
```



# inserire un font dal ttf

---

scaricare il font ad esempio fontsquirrel o google fonts

<https://www.fontsquirrel.com/>

scaricare font e metterlo nella cartella (nell'esempio /fonts)

includerlo nel css

```
@font-face{  
src: url("font/SinkinSans-100Thin.otf");  
font-family: SinkinSans;  
}  
body {  
font-family: SinkinSans;
```



# font-size

---

è la proprietà considerata essenziale nella definizione dell'aspetto del testo, di cui definisce le dimensioni. È applicabile a tutti gli elementi ed ereditata

**Dimensione assoluta** significa che essa non dipende da nessun altro elemento ed è quella **definita dall'unità di misura usata**.

**Dimensione relativa** significa che essa viene calcolata in base alla dimensione del testo **dell'elemento parente**.

le sette parole chiave xx-small, x-small, small, medium, large, x-large, xx-large; quelli espressi con le seguenti unità di misura: pixel (px), centimetri (cm), millimetri (mm), punti (pt), picas (pc), pollici (in), x-height (ex).

Di tutte queste unità, le uniche proponibili per il testo sono punti e pixel. Si consiglia di usare la prima solo per CSS destinati alla stampa.

# font-size

---

Sono valori relativi:

le parole chiave `smaller` e `larger`;

quelli espressi in `em`;

quelli espressi in percentuale.

Nelle pratiche più comuni, la scelta del dimensionamento dei font viene fatta tra `pixel`, `em`, `rem` e percentuale.

\* Se la dimensione di un font dovrebbe essere `12px`, mettere `2em` lo fa diventare `24px`, mentre metterlo `3em` significa metterlo `36px`

# font-weight

---

Serve a definire la consistenza o "peso" visivo del testo. Si applica a tutti gli elementi ed è ereditata

Il "peso" visivo di un carattere può essere espresso con una scala numerica o con parole chiave:

valori numerici: 100 – 200 – 300 – 400 – 500 – 600 – 700 – 800 – 900 ordinati in senso crescente (dal più leggero al più pesante);

**normal**: valore di default, è l'aspetto normale del font ed equivale al valore **400**;

**bold**: il carattere acquista l'aspetto che definiamo in genere 'grassetto'; equivale a 700;

**bolder**: misura relativa; serve a specificare che una determinata porzione di testo dovrà apparire più pesante a livello visuale rispetto al testo dell'elemento parente;

**lighter**: misura relativa; il testo sarà più leggero di quello dell'elemento parente.

p {font-weight: 900;}

div {font-weight: bold;}

# font-style

---

Imposta le caratteristiche del testo in base ad uno di questi tre valori:

normal: il testo mantiene il suo aspetto normale;

italic: formatta il testo in corsivo;

oblique: praticamente simile a italic.

La proprietà si applica a tutti gli elementi ed è ereditata.

Sintassi ed esempi

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

```
p {font-style: italic;}
```

# line-height

---

serve a definire l'altezza di una riga di testo all'interno di un elemento blocco. Ma l'effetto ottenuto è appunto quello di impostare uno spazio tra le righe

normal: il browser separerà le righe con uno spazio ritenuto "ragionevole"; dovrebbe corrispondere a un valore numerico compreso tra 1 e 1.2;

un valore numerico: usando valori numerici tipo 1.2, 1.3, 1.5 si ottiene questo risultato: l'altezza della riga sarà uguale alla dimensione del font moltiplicata per questo valore;

un valore numerico con unità di misura: l'altezza della riga sarà uguale alla dimensione specificata;

percentuale: l'altezza della riga viene calcolata come una percentuale della dimensione del font.

```
p {line-height: 1.5;}
```

```
body {line-height: 15px;}
```

# font

---

La proprietà font è una proprietà a **sintassi abbreviata** che serve a impostare con una sola dichiarazione tutte le principali caratteristiche del testo. Le proprietà definibili in forma abbreviata con font sono:

la sequenza:

**font: font-style font-variant font-weight font-size/line-height font-family**

p {font: bold 12px/1.5 Georgia, "Times New Roman", serif;}

# text-decoration

---

I valori che è possibile usare sono:

none: il testo non avrà alcuna decorazione particolare;

underline: il testo sarà sottolineato;

overline: il testo avrà una linea superiore;

line-through: il testo sarà attraversato da una linea orizzontale al centro;

```
p {text-decoration: none;}
```

```
a { text-decoration: underline;}
```

[text-decoration-style](#) ( solid, wavy, dotted, dashed, double)

[text-decoration-color](#)

[text-decoration-line](#) (underline, overline, line-through))

# font-variant

---

Consente di trasformare il testo in maiuscoletto (lettere in maiuscolo rese con dimensioni uguali ai caratteri minuscoli ). Proprietà ereditata.

selettore {font-variant: valore;}

I valori possibili sono solo due:

**normal**: il testo ha il suo aspetto normale; valore iniziale e di default;

**small-caps**: trasforma il testo in maiuscoletto.

h2 {font-variant: small-caps;}



# text-indent

---

Definisce **l'indentazione della prima riga in ogni elemento contenente del testo**.  
Proprietà ereditata.

selettore {text-indent: valore;}

Si può esprimere il valore con:

un valore numerico con unità di misura;

un valore in percentuale.

Come al solito, il valore con unità di misura è assoluto, quello in percentuale è relativo. In questo caso il valore è relativo alla larghezza dell'area del contenuto. In pratica, se per un paragrafo largo 200px imposto un'indentazione uguale al 10%, essa sarà uguale a 20px.

p {text-indent: 10px;}

div {text-indent: 10%;}

# text-transform

---

Questa proprietà serve a cambiare gli attributi del testo relativamente a tre aspetti: maiuscolo, minuscolo, prima lettera maiuscola. È una proprietà ereditata.

selettore {text-transform: valore;}

la keyword **none**: valore di default; nessuna trasformazione viene applicata;

**capitalize**: la prima lettera di ogni parola viene trasformata in maiuscolo;

**uppercase**: tutto il testo diventa maiuscolo;

**lowercase**: tutto il testo è minuscolo.

p {text-transform: capitalize;}

h1 {text-transform: uppercase;}

# letter-spacing

---

Aumenta lo spazio tra le lettere di una parola. Proprietà ereditata.

selettore {letter-spacing: valore;}

Per i valori si può scegliere tra:

normal: valore di default; le lettere mantengono il loro spazio normale;

un valore numerico con unità di misura: le lettere saranno spaziate secondo la distanza impostata.

È possibile anche impostare valori negativi. Ciò farà sì che le lettere appaiano sempre più compresse.

h1 {letter-spacing: 5px;}

# word-spacing

---

Proprietà complementare a letter-spacing. Serve ad aumentare lo spazio tra le parole comprese in un elemento. Proprietà ereditata.

selettore {word-spacing: valore;}

Per i valori possiamo usare:

normal: valore di default; le parole mantengono il loro spazio normale;

un valore numerico con unità di misura: le parole saranno spaziate secondo la distanza impostata.

p {word-spacing: 1.2em;}

div { word-spacing: 15px;}

# @font-face

---

Tramite **@font-face** possiamo utilizzare font in questi formati: TrueType (TTF), OpenType (OTF), WOFF, SVG, Embedded OpenType (EOT).

```
@font-face {  
  font-family: myFirstFont;  
  src: url(sansation_light.woff);  
}
```

```
div {  
  font-family: myFirstFont;  
}
```

# Text-shadow

---

consente di creare un testo ombreggiato grazie alla proprietà **text-shadow**

il primo (2px) definisce lo spostamento dell'ombra sull'asse orizzontale (x);  
il secondo (2px) definisce lo spostamento dell'ombra sull'asse verticale (y);  
il terzo valore (3px) imposta il livello di sfocatura (blur) dell'ombra: più alto è questo valore, più sfocata apparirà l'ombra; se si usa 0 otterremo un'ombra netta e senza sfocatura;  
il quarto valore (#333) definisce il colore dell'ombra.

## Messaggio

```
text-shadow: 5px 5px 6px black;
```

```
text-shadow: 2px 2px 3px #333;
```

**Lorem ipsum dolor sit amet, consectetur**

# list-style-image

---

Definisce l'URL di un'immagine da usare come marcatore di un list-item. Proprietà ereditata. Si applica agli elementi `li` e a quelli per i quali si imposti la proprietà `display` sul valore `list-item`.

selettore {list-style-image: url(<url\_immagine>);}

Nella definizione della sintassi per questa e per le altre proprietà che vedremo nella lezione, possiamo impostare la regola a partire dall'elemento/selettore `li`:

```
li {list-style-image: url(liststyle.png);}
```

# list-style-position

Imposta la posizione del marcatore rispetto al testo del list-item. Proprietà ereditata. Si applica agli elementi li e a quelli per i quali si imposti la proprietà display sul valore list-item.

selettore {list-style-position: valore;}

Il valore può corrispondere ad una di queste due parole chiave:

**outside:** valore di default; è il comportamento standard, il marcatore è collocato all'esterno del testo;

**inside:** il marcatore diventa parte integrante del testo e ne rappresenta in un certo senso il primo carattere; se il testo va a capo il marcatore apparirà all'interno del box.

li {list-style-position: inside;}

#lista li {list-style-position: outside;}

| <code>list-style-position: outside;</code>  | <code>list-style-position: inside;</code>   |
|---|---|
| <ul style="list-style-type: none"><li>• Coffee - A brewed drink prepared from roasted coffee beans...</li><li>• Tea</li><li>• Coca-cola</li></ul> | <ul style="list-style-type: none"><li>• Coffee - A brewed drink prepared from roasted coffee beans...</li><li>• Tea</li><li>• Coca-cola</li></ul> |



# Tabelle con i CSS: table-layout

imposta il metodo di layout di una tabella. Non è ereditata. Si applica solo alle tabelle.

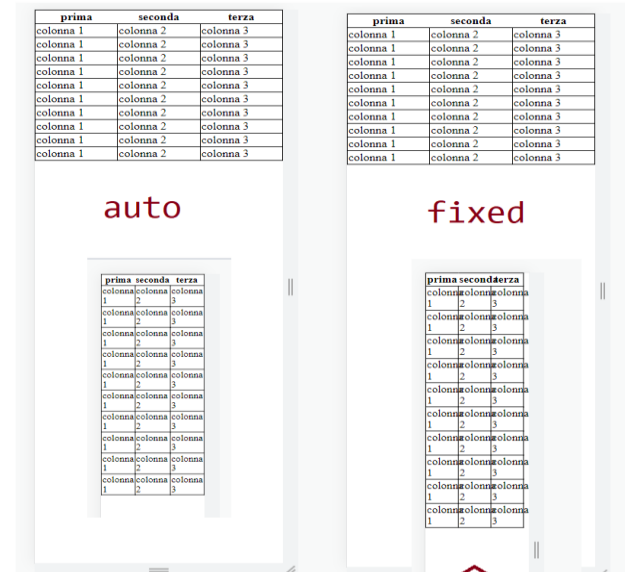
**auto:** il layout della tabella viene definito automaticamente dal browser;

**fixed:** le regole di presentazione sono quelle impostate dall'autore nel CSS.

Nel caso del valore auto, tutto è affidato al meccanismo di rendering del browser. Usando invece fixed possiamo innanzitutto definire la larghezza della tabella tramite la proprietà width.

```
table.ex1 {table-layout: auto; width: 150px}
```

```
table.ex2 {table-layout: fixed; width: 150px}
```



il contenuto sborda dalla cella

## border-collapse

possiamo stabilire in che modo trattare i bordi e gli spazi tra le celle di una tabella. Si applica solo alle tabelle ed è ereditata

collapse: se viene impostato un bordo, le celle della tabella lo condividono;  
separate: se viene impostato un bordo, ogni cella ha il suo, separato dalle altre;  
lo spazio tra le celle e tra i bordi si imposta con la proprietà border-spacing.

```
table.ex2 {table-layout: fixed; border: 2px
solid black; border-collapse: separate; border-
spacing: 5px; }
```

# collapse separate

[illegible]

# empty-cells

---

Gestisce il trattamento delle celle di tabella senza contenuto. Agisce solo su quelle che non presentino al loro interno alcun tipo di markup. Proprietà ereditata.

Sintassi ed esempi

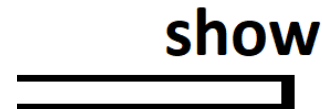
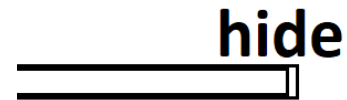
Anche in questo caso due i valori possibili:

```
td {empty-cells: show;}
```

show: mostra i bordi della cella;

hide: i bordi non vengono mostrati e apparirà solo uno spazio vuoto.

**IMPORTANTE: hide funziona con border-collapse:separate**














# Modificare l'aspetto del cursore con i CSS

I CSS offrono un meccanismo per modificare l'aspetto del cursore quando si passa con il mouse sopra un elemento.

Tale meccanismo è gestito con la proprietà `cursor`. È una proprietà ereditata e si applica a tutti gli elementi.

selettore {`cursor`: valore;}

`cursor` può assumere uno di questi valori. È possibile praticamente usare tutti i tipi di cursore abituali dell'interfaccia utente che usate. Ecco la lista:

|   |   |   |   |
|---|---|---|---|
|  all-scroll  |  col-resize  |  crosshair     |  e-resize        |
|  hand       |  help       |  move         |  n-resize       |
|  ne-resize |  no-drop   |  not-allowed |  nw-resize     |
|  pointer   |  progress  |  row-resize  |  s-resize      |
|  se-resize |  sw-resize |  text        |  vertical-text |
|  w-resize  |  wait      |   |   |

# Opacity

---

Con opacity è possibile definire il livello di trasparenza di qualunque elemento. Di fatto, come recita la specifica, con opacity regoliamo il modo in cui un oggetto presente sulla pagina si fonde nella resa a schermo con lo sfondo

```
#id1{background-color: #0cfa2c;position: absolute;z-index:1;width: 200px;height: 200px;}  
#id2{background-color: #000000; opacity: 0.5;position: absolute;z-index:2;width: 200px;height: 200px;}
```

# Box-shadow

---

la proprietà box-shadow introdotta nei CSS3 è forse, insieme a border-radius, quella più utile: con poche righe di codice CSS consente di ottenere un effetto di indubbia efficacia senza dover ricorrere a immagini, div aggiuntivi, hack di vario genere.

box-shadow può essere applicata a tutti gli elementi.

Vediamo subito uno snippet di codice con quella che è la sintassi di base:

(orizz – vert – sfumatura – raggio - colore)

```
box-shadow: 5px 5px 10px 2px #333333;
```

```
box-shadow: 3px 3px 50px 2px #3c3c3c;
```






## Box 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

# -webkit- e -moz-

---

per alcune vecchie versioni dei browser alcune proprietà non sono supportate,  
è il caso di box-shadow che nelle versioni precedenti  
al 10.0 di chrome occorre usare: -webkit-box-shadow  
al 4.0 di firefox -moz-box-shadow  
al 5.1 di safari -webkit-box-shadow

| Property   |  |  |  |  |  |
|------------|---|---|---|---|---|
| box-shadow | 10.0<br>4.0 -webkit-  | 9.0   | 4.0<br>3.5 -moz-  | 5.1<br>3.1 -webkit-   | 10.5  |





# Layout multicolonna 1

---

gruppo di proprietà che rendono possibile la dislocazione del contenuto di un box su più **colonne**

Sono le proprietà essenziali, quelle con cui si imposta il **layout multi-colonna** nei suoi fondamenti:

**column-width**

**column-count**

**column-gap**

**column-rule**

Proprietà abbreviata: **columns**:

**columns**: 100px 3;

(minimum width for each column, and the maximum number of columns)

# column-count

---

divide un blocco in colonne

```
#esterno{  
width: 500px;  
column-count: 3;  
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in

hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod

mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius.

# column-width e column-gap

---

se impostata sia column-width che column-count verrà applicata la column-count

```
.newspaper{  
width: 750px;  
margin: 20px auto;  
column-width: 350px;  
column-gap: 25px;  
}
```

## Test multicolonna

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

## Titolo

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

## Titolo

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

# column-count

---

Per impostare il **numero di colonne** si può ricorrere anche alla proprietà `column-count`. La differenza rispetto a `column-width` è che in questo caso lasciamo al browser il compito di calcolare automaticamente la larghezza delle colonne in base al numero che usiamo

```
.newspaper{  
width: 750px;  
margin: 20px auto;  
column-count: 3;  
column-gap: 25px;  
}
```

## Test multicolonna

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

## Titolo

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

## Titolo

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

# column-rule

---

L'ultima delle proprietà di base che andiamo ad analizzare è column-rule. Serve a creare un bordo nello spazio (gap) tra le colonne

```
#container {  
width: 750px;  
margin: 20px auto;  
column-width: 350px;  
column-gap: 25px;  
column-rule: 1px solid black;  
}
```

## Test multicolonna lorem ipsum dolor sit amet, consectetur adipisicing

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

### Titolo

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

### Titolo

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

# column-span

---

il titolo si espande fino a comprendere tutte le colonne. Al posto avremmo potuto usare un valore numerico. Usando 2, per esempio, il testo si sarebbe esteso solo sulle prime due colonne, non è supportato da tutti i browser

valori possibili: none | all | initial | inherit

```
h1 {font-size: 18px;  
-webkit-column-span:all;  
-moz-column-span:all;  
column-span:all;  
}
```

**Test multicolonnaTest multicolonnaTest multicolonnaTest multicolonnaTest multicolonnaTest multicolonna**

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

# Flexible box layout (flexbox)

---

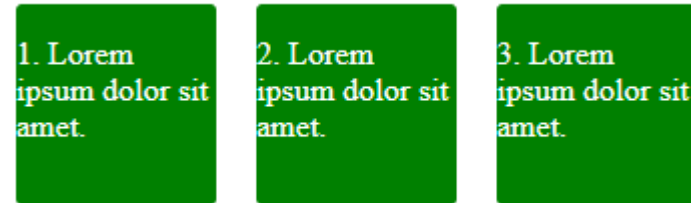
la struttura minima per agire con il flexbox prevede un elemento contenitore e degli elementi figli

```
<div id="container" >
  <div id="box1" >
    <p>1. Lorem ipsum dolor sit amet.</p>
  </div>
  <div id="box2" >
    <p>2. Lorem ipsum dolor sit amet.</p>
  </div>
  <div id="box3" >
    <p>3. Lorem ipsum dolor sit amet.</p>
  </div>
</div>
```

# Flexbox – orientamento verticale

---

```
#container{  
width: 500px;  
display:flex;  
flex-direction: row;  
flex-wrap: nowrap;  
}  
#box1,#box2,#box3{  
width: 100px;  
height: 100px;  
background-color: green;  
color:white;  
border-radius: 3px;  
margin: 10px;  
}
```

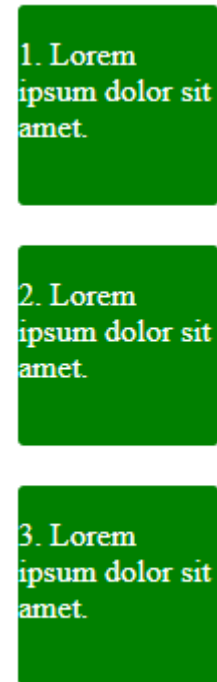




# Flexbox – orientamento orizzontale

---

```
#container{  
width: 500px;  
display:flex;  
flex-direction: column;  
flex-wrap: nowrap;  
}  
#box1,#box2,#box3{  
width: 100px;  
height: 100px;  
background-color: green;  
color:white;  
border-radius: 3px;  
margin: 10px;  
}
```



# Flexbox justify-content

determina l'allineamento  
orizzontale

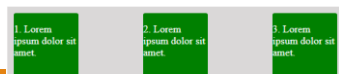
**flex-start** - Default value. Items  
are positioned at the beginning of  
the container

**flex-end** - Items are positioned at  
the end of the container

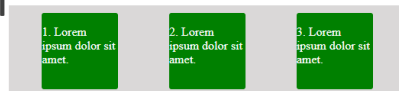
**center**: items are centered in  
container

**space-between** - Items are  
positioned with space between  
the lines

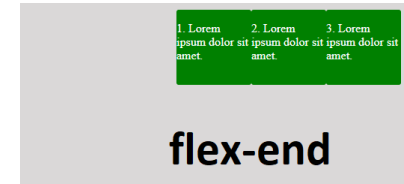
**space-around** - Items are  
positioned with space before,  
between, and after the lines



space between



space around



flex-end

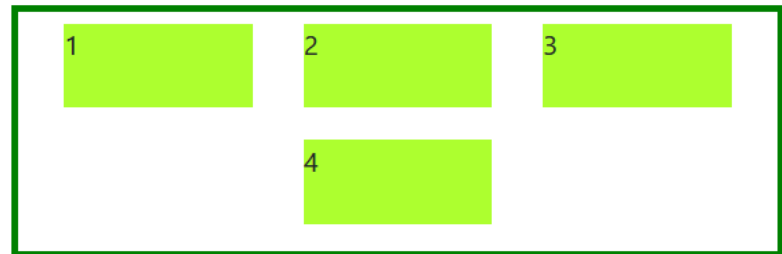
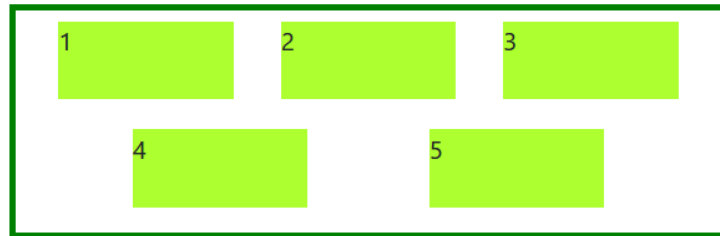
```
#container{  
width: 500px;  
height: 500px;  
display: flex;  
flex-direction: row;  
flex-wrap: nowrap;  
justify-content: space-around;  
background-color: #dad8d8;  
padding: 10px;  
}  
#box1, #box2, #box3{  
width: 100px;  
height: 100px;  
background-color: green;  
color: white;  
border-radius: 3px;  
}
```

# Flexbox justify-content

---

justify-content: **space-evenly**

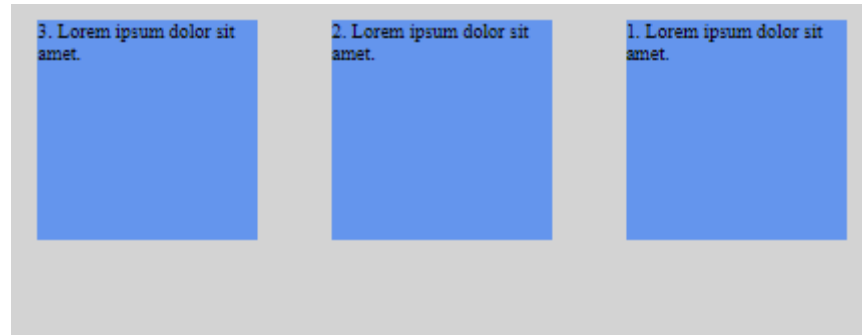
gli elementi sono ugualmente spaziati nella riga



# row-reverse, column-reverse

---

inverte l'ordine degli item (ex row reverse)



`flex-direction: row-reverse;`

per le colonne: column-reverse

# align-items: allineamento verticale

stretch - Default value. Items are stretched to fit the container

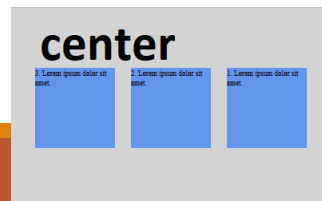
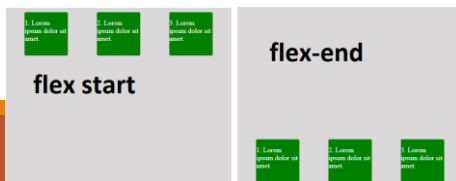
flex-start - Items are positioned at the top of the container

flex-end - Items are positioned at the bottom of the container

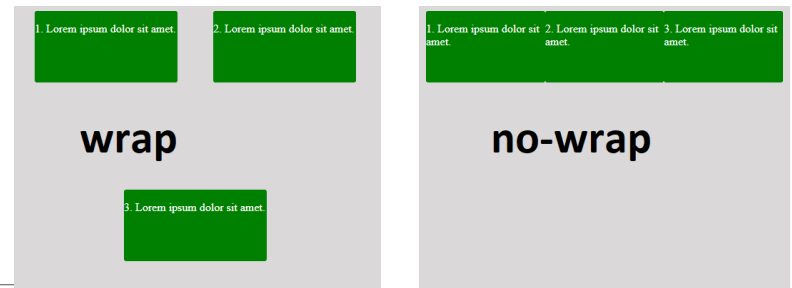
center - Items are positioned at the center of the container (vertically)

baseline - Items are positioned at the baseline of the container

```
#container{  
width: 500px;  
height: 500px;  
display:flex;  
flex-direction: row;  
flex-wrap: nowrap;  
justify-content: space-around;  
align-items: flex-end;  
background-color: #dad8d8;  
padding: 10px;  
}  
#box1,#box2,#box3{  
width: 100px;  
height: 100px;  
background-color: green;  
color:white;  
border-radius: 3px;  
}
```



# flex-wrap



nowrap - Default value. The flexible items will not wrap

wrap - The flexible items will wrap if necessary

wrap-reverse - The flexible items will wrap, if necessary, in reverse order



```
#container{  
width: 500px;  
height: 500px;  
display: flex;  
flex-direction: row;  
flex-wrap: nowrap;  
justify-content: space-around;  
background-color: #dad8d8;  
padding: 10px;  
flex-wrap: no-wrap;  
}  
#box1, #box2, #box3{  
width: 200px;  
height: 100px;  
background-color: green;  
color: white;  
border-radius: 3px;  
}
```

# flex-wrap

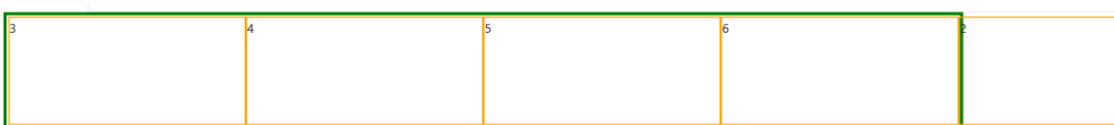
---

il default è no-wrap: quindi nel caso gli elementi non possano essere contenuti tutti nel contenitore a causa della width

wrap



no-wrap



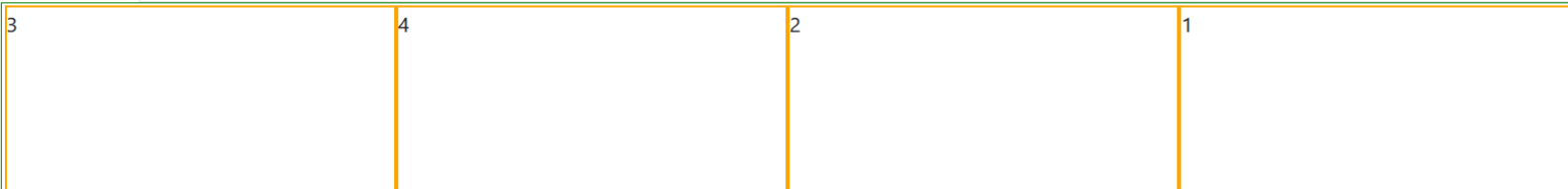
esce dal contenitore

# Order

---

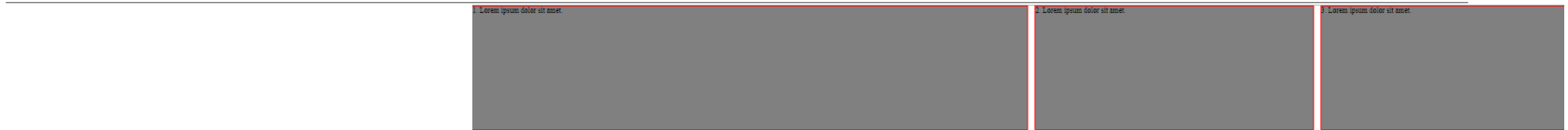
Oltre a invertire l'ordine in cui gli articoli flessibili vengono visualizzati visivamente, puoi scegliere come target singoli articoli e modificare la posizione in cui appaiono nell'ordine visivo con la proprietà `order`.

```
.flex{
  display: flex;
  width: 80%;
  min-height: 150px;
  border: 1px solid green;
  padding: 2px;
}
.flex div {
  min-width: 25%;
  border: 2px solid orange;
}
.flex div:nth-child(1){
  order: 2;
}
.flex div:nth-child(2){
  order: 1;
}
```





# Datagrid – 3 colonne



```
.grid-container{  
display: grid;  
grid-template-columns: 50% 25%  
25%;  
grid-gap:10px;  
}  
#container div{border:1px solid  
#ff0000; background-color:  
grey;width:100%;height: 150px; }
```

# Datagrid – 2 colonne e span



```
.grid-container{  
display: grid;  
grid-template-columns: auto auto;  
grid-gap:10px;  
}  
#container div{border:1px solid  
#ff0000; background-color:  
grey;width:100%;height: 150px; }  
#box3{grid-column: span 2;}
```

# Datagrid – 2 colonne, 2 row e span

---

```
.grid-container {  
display: grid;  
grid-template-columns: auto auto;  
grid-template-rows: auto auto auto;  
grid-gap: 10px;  
width: 50%;  
}
```

```
#container div {  
border: 1px solid #ff0000;  
background-color: grey;  
width: 100%;  
min-height: 150px;  
}
```

```
#box4 {  
grid-column: span 2;  
}
```

```
#box1 {  
grid-row: span 2;  
}
```



# Grid: esempio su responsive codice <=576px

---

```
@media screen and (max-width:576px){  
#contenitore{  
display: grid;  
grid-template-columns: 50% 50%;  
}  
#contenitore div:nth-child(1){  
grid-column: span 2;  
}
```

```
#contenitore div{  
min-height: 100px;  
min-width: 100px;  
}  
#contenitore div:nth-child(1){  
background-color: red;  
}  
#contenitore div:nth-child(2){  
background-color: blue;  
}  
#contenitore div:nth-child(3){  
background-color: violet;  
}  
#contenitore{  
grid-column-gap: 5px;  
grid-row-gap: 5px;  
}
```

```
<body>  
<div id="contenitore">  
<div ></div>  
<div></div>  
<div></div>  
  
</div>  
</body>
```



# Grid: esempio su responsive codice

## >576px

---

```
@media screen and (min-width:577px){  
#contenitore{  
display: grid;  
grid-template-columns: 50% 25% 25%;  
max-height: 300px;  
grid-column-gap: 5px;  
}  
}
```

```
#contenitore div{  
min-height: 100px;  
min-width: 100px;  
}  
#contenitore div:nth-child(1){  
background-color: red;  
}  
#contenitore div:nth-child(2){  
background-color: blue;  
}  
#contenitore div:nth-child(3){  
background-color: violet;  
}  
#contenitore{  
grid-column-gap: 5px;  
grid-row-gap: 5px;  
}
```

```
<body>  
<div id="contenitore">  
<div ></div>  
<div></div>  
<div></div>
```

```
</div>  
</body>
```



# backdrop-filter

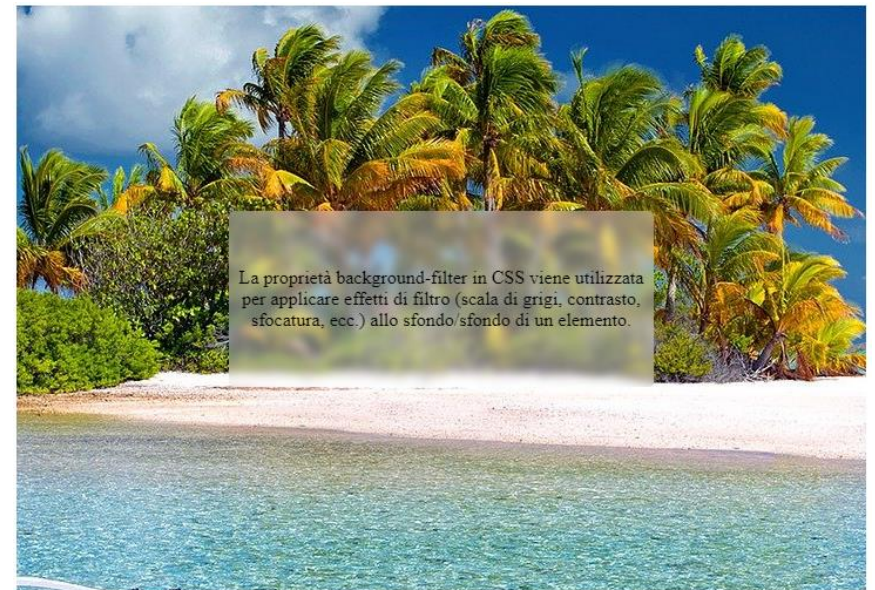
La proprietà backdrop-filter in CSS viene utilizzata per applicare effetti di filtro (scala di grigi, contrasto, sfocatura, ecc.) allo sfondo/sfondo di un elemento.

Va applicata all'elemento contenuto che deve avere un background con trasparenza

## Browser compatibility

[Report problems with this compatibility data on GitHub](#)

|                 | Desktop |      |         |                   |       |        | Mobile          |                |                     |               |               |                  |
|-----------------|---------|------|---------|-------------------|-------|--------|-----------------|----------------|---------------------|---------------|---------------|------------------|
|                 | Chrome  | Edge | Firefox | Internet Explorer | Opera | Safari | WebView Android | Chrome Android | Firefox for Android | Opera Android | Safari on iOS | Samsung Internet |
| backdrop-filter | 76      | 17   | 70<br>★ | No                | 63    | 9 -X-  | 76              | 76             | No ★                | 54            | 9 -X-         | 12.0             |



```
backdrop-filter: blur(4px);
```

# Animazioni in CSS

Creano delle animazioni tramite il css

le proprietà fondamentali:

@keyframes //definisce la regola, può essere separata in più momenti

```
@keyframes lampeggia{  
  from{opacity: 1.0;}  
  50%{opacity: 0.2;}  
  100%{opacity: 1.0;}  
}
```

```
#esterno{  
  width: 100px;  
  height: 100px;  
  background-image:  
  url(img1.jpg);  
  border-radius: 5px;  
  animation-name:  
  lampeggia;  
  animation-duration:3s;  
  background-color:  
  #37c54a;  
  animation-iteration-  
  count: infinite;  
}
```



# proprietà delle animazioni

---

animation-name: definisce il nome del keyframe

animation-duration: definisce la durata

animation-iteration: definisce il numero di volte (oppure infinite)

animation-delay: definisce i secondi di ritardo (ad es. 2s)

animation-direction: normal | reverse | alternate | alternate-reverse | initial | inherit;

animation-play-state: paused | running : mette in pausa una animazione

animation-timing-function: animation-timing-function:

linear | ease | ease-in | ease-out | ease-in-out | step-start | step-end | steps(int,start | end) | cubic-bezier( $n,n,n,n$ )



# animazione movimento

---

```
#esterno{  
width: 100px;  
height: 100px;  
background-image:  
url(img1.jpg);  
border-radius: 5px;  
animation-name: muovi;  
animation-duration: 3s;  
background-color: #37c54a;  
animation-iteration-count:  
infinite;  
position: relative;  
animation-timing-function: ease-in;  
}
```

```
@keyframes muovi{  
from{left: 5px;}  
50%{left: 200px;}  
100%{left: 5px;}  
}
```

esempi:

[https://webplatform.github.io/docs/tutorials/css\\_animation\\_s/](https://webplatform.github.io/docs/tutorials/css_animation_s/)

# altro esempio

---

```
<div id="canvas">  
  <div class="muovi">muovi</div>  
</div>
```

```
.muovi{  
  position: relative;  
  background-color: blueviolet;  
  padding: 10px;  
  border-radius: 5px;  
  text-align: center;  
  width: 100px;  
  margin: 20px;  
  animation: ritorna 2s infinite alternate, muovi  
2s infinite alternate;  
}  
#canvas {  
  width: 100%;  
  padding: 50px;  
}  
@keyframes muovi {  
  from {left: 0px;}  
  100% {left: calc(100% - 100px);}  
}  
@keyframes ritorna {  
  from {left: calc(100% - 100px)}  
  100% {left:0;}  
}
```

# css3 transition

---

Effettua una transizione, normalmente utilizzata sui cambi di stato ad esempio hover, active ecc..

```
#esterno{
width: 100px;
height: 100px;
background-image:
url(img1.jpg);
border-radius: 5px;
background-color:
#37c54a;
position: relative;
transition: 2s;
}
#esterno:hover{
background-color:
blueviolet;
}
```

# proprietà delle transition

---

transition-delay

transition-duration (es: 1s)

transition-property

transition-timing-function: ease |  
linear | ease-in | ease-out | ease-  
in-out

# Riferimenti bibliografici

---

I contenuti sono tratti dal sito [html.it](http://html.it), [w3schools.com](http://w3schools.com), [developer.mozilla.org](http://developer.mozilla.org) e rielaborati dal docente