CSS 3

Rev. 3.1 del 16/12/2024

CSS 3

CSS 3

Dai CSS 2 ai CSS3 (2013) il numero di proprietà è passato da **120** a **245**, e la cifra potrebbe salire con l'introduzione di nuove proprietà. Basta questo dato per capire che i CSS3 coprono le aree più disparate nel contesto della presentazione visuale (e non solo) dei contenuti distribuiti via web.

Flex Layout

```
#wrapper {
    display: flex;     /* esiste anche inline-flex     */
}
#wrapper div {width:100px; height:100px;}
```

Consente di riposizionare gli elementi interni automaticamente in modo **responsive**.
Un contenitore display:flex assume la larghezza del contenitore (stile display:block) a meno che non si specifichi una precisa larghezza.

Per quanto riguarda la larghezza, anche se gli elementi interni hanno una certa **width**, come comportamento di default, la **larghezza** degli elementi interni viene automaticamente ridimensionata in modo da poter visualizzare tutti gli elementi all'interno di una sola riga.

- Se gli elementi interni sono troppo grandi vengono **proporzionalmente rimpiccioliti** in modo da poterli contenere tutti
- Se gli elementi interni sono troppo piccoli viene lasciato uno **spazio libero a destra**, cioè gli elementi **non vengono automaticamente ingranditi**, a meno che non si imposti sui singoli elementi la proprietà **`flex-grow:1`**;
- Se gli elementi interni non hanno una width, assumono la width del testo (cioè sostanzialmente diventano inline-block)

Per quanto riguarda l'altezza, se gli elementi interni non hanno **height**, come comportamento di default, la proprietà **`align-items`** assume il valore **stretch** facendo in modo che gli elementi interni assumano la stessa altezza del contenitore.

Le 4 Proprietà base del contenitore: flex-direction, justify-content e align-content, align-items

`flex-direction` row | row-reverse | column | column-reverse

Indica la disposizione degli elementi interni che possono essere disposti orizzontalmente per righe o verticalmente per colonne, in ordine rispetto a come sono scritti nel file html o in ordine inverso.

- Se ordinati in colonna vengono messi uno sotto l'altro su un'unica colonna
- Riguardo alle dimensioni il comportamento continua ad essere quello di default, nel senso che gli elementi vengono automaticamente rimpiccioliti se necessario, ma non vengono automaticamente ingranditi, a meno che non si imposti **`flex-grow:1`** sugli elementi interni.
- Se rimpiccioliti la riduzione si ferma senza nascondere il testo interno (height:auto)

`flex-wrap` **V**alori possibili: nowrap | **wrap** | wrap-reverse

In caso di flex:wrap, raggiunta la fine riga o la fine colonna (a seconda di flex-direction), automaticamente gli elementi successivi vengono posizionati sulla riga / colonna successiva.

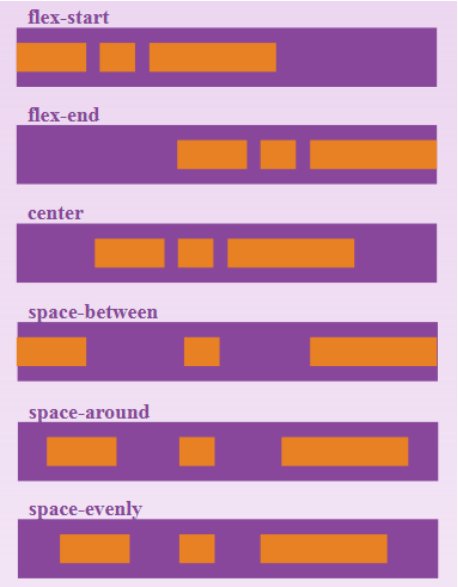
- La disposizione degli elementi parte dall'alto verso il basso.
  Nel caso di wrap-reverse la disposizione parte dal basso verso l'alto.
- flex:wrap impedisce il 'rimpicciolimento' nel senso che gli elementi **non vengono MAI rimpiccioliti**. Al limite deborderanno oltre il contenitore.
  Viene eventualmente ingrandito se si imposta sugli elementi interni la proprietà **`flex-grow:1`**

Centratura manuale degli elementi interni

`justify-content` consente di impostare il tipo di spaziatura orizzontale da utilizzare tra un elemento e l'altro. Valori flex-start | flex-end | center | space-between | space-around | space-evenly
Oltre a *flex-start* e *flex-end* sono disponibili anche gli alias ***start*** e ***end***

In pratica ridistruibuisce tra i vari elementi lo spazio libero di fine riga.
Questa proprietà viene ignorata se:

- non c'è spazio libero a fine riga
- si imposta sugli elementi interni la proprietà **`flex-grow:1`** che provoca l'automatico ingrandimento.

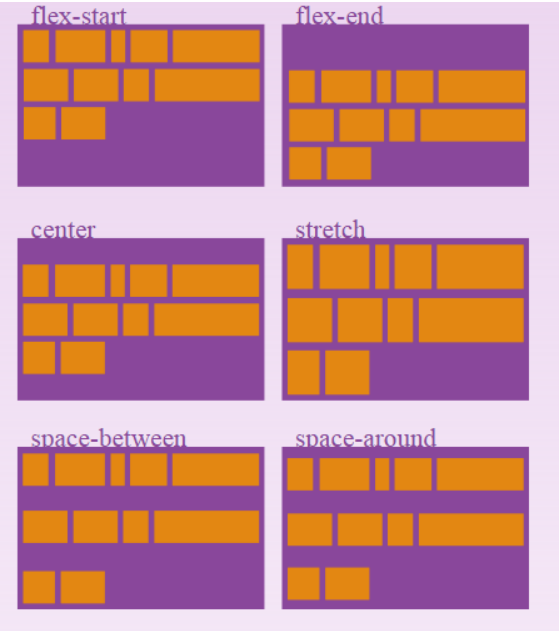


`align-content` proprietà analoga per la spaziatura verticale.
consente di impostare la **spaziatura** degli elementi **sull'asse opposto** rispetto a quello utilizzato per la disposizione.
Valori possibili: normal | flex-start | flex-end | center | space-between | space-around | stretch.

- Il valore **stretch** ingrandisce l'elemento in modo che venga ricoperto l'intero spazio verticale. Viene riconosciuto **SOLTANTO per gli elementi interni che NON dispongono della proprietà height** (height:auto) e quando **NON** è impostato **`align-items`**

- Il valore di default di `align-content` è:
  - **normal** per gli elementi che dispongono della proprietà height
  - **stretch** per gli elementi che NON dispongono della proprietà height

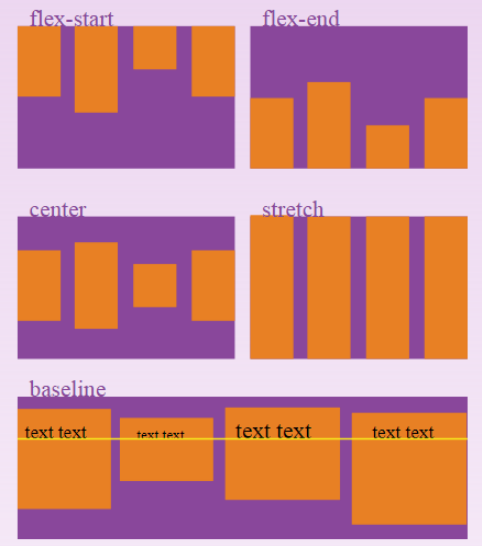
normal:
lo spazio libero viene equamente suddiviso **dopo** ogni riga



Nelle immagini non si vede bene.
Nel caso di **space-between** la prima e l'ultima riga sono 'incollate' al bordo
Nel caso di **space-around** ogni elemento ha lo stesso 'margin' sopra e sotto

`align-items`

consente di impostare l'**allineamento** degli elementi sull'asse opposto rispetto a quello utilizzato per la disposizione sequenziale degli elementi.
Valori possibili: flex-start | flex-end | center | baseline | stretch .



`gap`

La proprietà gap, definita sempre sul contenitore, consente di definire manualmente una spaziatura fra i vari elementi interni nella stessa direzione definita da `flex-direction`

Proprietà degli oggetti interni

Sono in pratica alternative rispetto alla proprietà **`justify-content`** del contenitore. Se impostate "sovrascivono" **`justify-content`**. Mentre **`justify-content`** consente di suddividere in diversi modi la spaziatura rimanente nell'ambito di una riga, queste property consentono invece di suddividere la spaziatura rimanente nell'ambito di una riga in modo proporzionale fra gli elementi, andando ad incrementare / ridurre le width dei vari elementi.

`flex-grow: <number>` Abilita gli elementi interni ad **aumentare** in modo proporzionale le proprie dimensioni in modo da saturare completamente l'eventuale spazio disponibile di fine riga. Se tutti gli elementi hanno lo stesso **`flex-grow,`** lo spazio libero verrà equamente suddiviso fra tutti gli elementi.

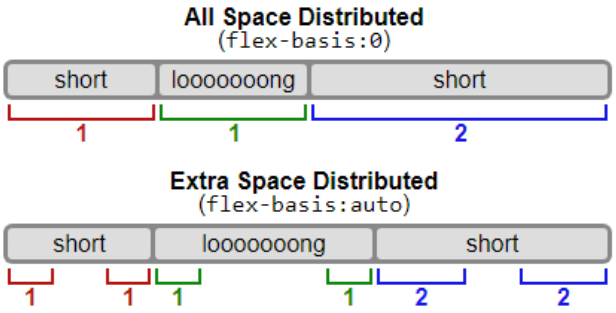
Se un elemento ha ad esempio **`flex-grow:1`** ed un altro **`flex-grow:2,`** il secondo riceverà il doppio dello spazio rispetto all'altro elemento. **`flex-grow:0`** significa che quell'elemento non utilizzerà nessuna percentuale dell'eventuale spazio residuo.

`flex-shrink: <number>` Opposta alla precedente. Abilita gli elementi interni a **contrarre** le proprie dimensioni in modo da poter visualizzare tutti gli elementi, nel caso in cui non possano essere visualizzati tutti nella stessa riga. E' in pratica una alternativa a **`flex-wrap:wrap`**;
Un elemento con **`flex- shrink:2`** verrà contratto il doppio rispetto a quello con **`flex-shrink:1`**

`flex-basis: <number> | auto`

`flex-basis = 0` fa sì che l'extra space venga suddiviso fra gli elementi sulla base della proprietà flex-grow.

`flex-basis` = auto (dafault) definisce invece le dimensioni dell'elemento sulla base del contenuto, spalmando la spaziatura residua sulla base sempre di flex-grow



`flex:` rappresenta una shorthand delle tre proprietà precedenti (grow, shrink, basis)

`margin:auto`
impostato sui div interni, esegue una centratura AUTOMATICA degli elementi interni, sia in orizzontale che in verticale. **Sovrascrive però tutte le proprietà precedenti** del contenitore,
cioè **`justify-content, align-content e align-items`** che consentirebbero invece una centratura più efficace

Nota: `float, clear e vertical-align` non hanno effetto sui flex item

La gestione di Bordi e Ombre

La proprietà `border-radius`

Consente di arrotondare i bordi di un generico elemento. Il

```
.bordiArrotondati {border:2px solid;  border-radius: 10px 10px 10px 10px;}
```

I 4 parametri indicano il raggio di curvatura di ciascun spigolo (superiore sinistro, superiore destro, inferiore sinistro, inferiore destro. Se i 4 valori sono tutti uguali si può utilizzare la notazione breve indicando un solo valore. Sono anche ammesse le voci specifiche come **`border-top-left-radius`**

Notare che i valori espressi non sono da intendersi come raggio di curvatura, ma come estensione della curvatura sull'asse x e sull'asse y. Impostando due valori separati da / come ad esempio

Es
```
.bordiArrotondati { border-radius: 10px/20px; }
```

il primo imposta la larghezza della curvatura sull'asse orizzontale, il secondo imposta la larghezza della curvatura sull'asse verticale, in modo da ottenere angoli ellittici. Impostando il valore 50% si ottiene una curvatura che occupa metà della larghezza e metà dell'altezza. Se queste sono uguali => cerchio.

La proprietà `box-shadow`

Consente di aggiungere una o più ombre ad un qualunque oggetto. E' possibile inserire le ombre non solo all'esterno del box ma anche all'interno. I possibili parametri sono in tutto cinque :

```
div { box-shadow: 5px  5px  10px  2px  #ddd; }
```

- il primo indica la profondità dell'ombra sull'asse x (verso destra). Valori negativi indicano verso sx
- il secondo indica la profondità dell'ombra sull'asse y (verso il basso). Valori negativi verso l'alto.
- il terzo indica il raggio di sfumatura (**blur**) applicato all'ombra. 0 indica senza sfumatura; l'ombra riproduce una copia esatta e nitida del bordo semplicemente traslata di x_{ombra} e y_{ombra}
- Il quarto eventuale valore imposta il livello di diffusione (*spread*) dell'ombra. Più i valori sono alti, più l'ombra tenderà ad espandersi, in tutte le direzioni. Se invece si usano valori negativi, l'ombra tende a contrarsi, fino a scomparire del tutto. Default 0.
- Terzo e Quarto parametro sono FACOLTATIVI.
- L'ultimo parametro indica il colore dell'ombra

Assegnando valori negativi ai primi due parametri, la sfocatura si estenderà verso sinistra e verso l'alto (cioè sempre esterna, ma sui due lati opposti del box rispetto alla situazione precedente.).

Il colore può anche essere scritto come primo valore

```
div { box-shadow: #ddd 10px  10px; }
```

Aggiungendo in testa un parametro **`inset`**, l'effetto ombra viene renderizzato verso l'interno, cioè a destra del bordo sinistro e sotto il bordo superiore, come se l'elemento fosse incassato.

```
div { box-shadow: inset #ddd 10px  10px  #ddd; }
```

E' possibile anche assegnare più di un'ombra all'interno della stessa regola separandole con virgola.

```
div { box-shadow:  3px 3px 2px #333333,
                   6px 7px 4px #30F744;}
```

1 Ombra grigia + 1 ombra verde

Le ombre si estendono nei margini, ammesso che questi esistano e abbiano dimensione sufficiente. Le ombre non intervengono nel conteggio delle dimensioni.

La gestione dei testi

La proprietà **word-wrap**

Se una singola parola era troppo lunga rispetto alla larghezza del contenitore, questa debordava verso destra fuori dal contenitore. Con il word-wrap la parola viene spezzata esattamente in corrispondenza del bordo.

```
p { word-wrap:break-word; }
```

La proprietà **text-shadow**

Consente di **aggiungere un'ombreggiatura al testo**.

```
h1 { text-shadow: 2px  2px  0px  black; }
```

I quattro parametri indicano rispettivamente:

- il primo indica la profondità dell'ombra sull'asse x (verso destra). Valori negativi indicano verso sx
- il secondo indica la profondità dell'ombra sull'asse y (verso il basso). Valori negativi verso l'alto
- il terzo indica il raggio di sfumatura (**blur**) applicato al testo dell'ombra. Senza sfumatura l'ombra riproduce una copia esatta e nitida del testo semplicemente traslata di x_{ombra} e $y_{ombra.}$ Poco Leggibile
- Valori maggiori di 4px provocano un effetto alone in genere abbinato a traslazioni X e Y pari a 0.
- il Quarto ed Ultimo Parametro indica il color dell'ombra
- Come per Box-Shadow si possono applicare più ombre contemporaneamente, separate da virgola.

La proprietà **resize**

Consente di far diventare un elemento **resizable** per l'utente. Per trascinare occorre selezionare lo spigolo in basso a destra. Sui tag DIV è utilizzabile solo in abbinamento a **overflow:auto**.

```
div { resize:both; overflow:auto; }
```

none	The user cannot resize the element
horizontal	The user can adjust the width of the element
vertical	The user can adjust the height of the element
both	The user can adjust both the height and the width of the element

Le proprietà **outline** e **outline-offset**

L'**outline** è il bordino blu che viene aggiunto per default **intorno ai controlli quando sono selezionati** (cioè quando hanno il :focus)

```
div {
   border:2px solid black;
   outline:2px solid red;
   outline-offset:15px; }
```

Crea un bordo rosso spesso 2 px esternamente rispetto al bordo del contenitore, con una distanza di 15px. Questo bordo **NON** viene contemplato nel calcolo degli spazi. In caso di Offset negativi il bordo viene tracciato all'interno del Box.

div:focus {outline:none} elimina il bordino intorno al controllo quando questo ha il focus

La gestione degli sfondi : background-size

Sfondi con immagini multiple

E' possibile caricare parallelamente più immagini di sfondo che vengono caricate una sopra l'altra (la prima indicata sarà in primo piano, l'ultima sarà sotto di tutte). Va bene per comporre lo sfondo con diverse immagini posizionate in aree diverse oppure quando l'immagine di primo piano (img1) ha uno sfondo trasparente che non copre completamente l'immagine sottostante

```
background-image: url(img1.gif),  url(img2.gif);
```

Per ciascuna immagine caricata è anche possibile definire dei posizionamenti diversi:

```
background-image: url(img1.png), url(img2.png);
background-position: top left,  bottom left
background-repeat:  no-repeat;
```

img1 viene allineata in alto a sinistra, img2 in basso sempre a sinistra

La proprietà `background-size`

Il tag `<img>` visualizza sempre l'immagine per intero, eventualmente aumentando le dimensioni del tag `<img>` medesimo. Viceversa, nel caso di un tag `<div>`, se l'immagine è più grande rispetto alle dimensioni del tag `<div>`, viene visualizzata SOLOTANTO una PORZIONE di immagine, cioè la sezione in alto a sinistra. `background-size` consente invece di visualizzare l'immagine per intero, cioè in modo da ricoprire quasi interamente l'area del contenitore (compreso il **padding**)

- Il valore **`cover`** : Se immagine e contenitore non hanno le stesse proporzioni, una dimensione dell'immagine eccederà le dimensioni del contenitore e non verrà visualizzata.
- Il valore **`contain`** : Se immagine e contenitore non hanno le stesse proporzioni, su una delle due dimensioni rimarrà una parte di contenitore 'scoperto' in cui si vderà il colore di sfondo del contenitore stesso.
- Si possono anche specificare direttamente le dimensioni (in pixel o in percentuale rispetto alle dimensioni del box).

  Ad esempio: **`background-size: 140px 220px;`** l'immagine viene ridimensionata in modo da coprire esattamente le dimensioni indicate con conseguente **stretch**
  `background-size:50% 50%;` l'immagine viene ridimensionata in modo da coprire il 50% del contenitore sia in orizzontale che in verticale con conseguente **stretch**

  Impostando invece un solo valore, questo viene interpretato come **dimensione X**.
  Ad es: **`background-size:50%`** l'immagine copre il 50% in orizzontale del contenitore, mentre l'altezza Y verrà calcolata **in automatico** senza provocare deformazione. Se la dimensione Y risultante va sforare l'altezza a sua disposizione, la parte eccedente non viene visualizzata.

La proprietà `box-sizing`

La proprietà **`box-sizing`** può assumere solo 2 valori:

- **`content-box`**; (default) l'elemento continua a comportarsi secondo il Box-Model tipico di CSS2, cioè se si specificano width e height per un certo elemento, questo verrà rappresentato a video con una dimensione complessiva pari a **`width + padding + bordo`**. Idem per l'altezza.
- **`border-box`**; l'elemento occuperà esattamente uno spazio pari a **width** e **height**. Padding e Bordo verranno "scalati" all'interno del contenitore (*ammesso che ci stiano, altrimenti le dimensioni vengono comunque aumentate adeguatamente*). **E' il valore di default soltanto per i button.**

Le proprietà `background-clip` e `background-origin`

`background-clip` Estensione del colore di sfondo di un contenitore
`background-origin` Estensione dell'immagine di sfondo di un contenitore

I possibili valori assegnabili a queste proprietà sono:

`border-box` Lo sfondo si estende anche "sotto" i bordi. Default per **background-clip**
`padding-box` Lo sfondo ricopre il Padding ma non i Bordi. Default per **background-origin**
`content-box` Lo sfondo ricopre solo la cosiddetta 'Area del Contenuto' (padding escluso)

Impostazione di un gradiente di sfondo

A livello sintattico, si tratta di un valore speciale della proprietà **`background-image`** che consente di "creare" una immagine sulla base di precise sfumature di colore.

Gradiente-Lineare

```
background-image: linear-gradient(to bottom, #FFF 30%,  #00F 90%);
background-image: linear-gradient(180deg,    #FFF 30%,  #00F 90%);
```

Parametri:

1. **direzione del gradiente**. Può assumere i seguenti valori: **`to bottom`** (default), **`to right`**, **`to bottom right`**, etc oppure il valore di un angolo espresso in gradi, ad esempio **`0deg`** che equivale a to top, 9**`0deg`** che equivale a to right, etc.
2. **colori di stop** espressi nella sequenza **`colore posizione`**. Nell'esempio precedente:
   dalla posizione 0 alla posizione 30 ci sarà un bianco saturo senza sfumature
   dalla posizione 30 alla posizione 90 ci sarà una sfumatura dal bianco al blu
   dalla posizione 90 alla posizione 100 ci sarà un blu saturo senza sfumature
   La posizione può essere espressa in forma percentuale o come decimale compreso tra 0.0 e 1.0

```
linear-gradient(to bottom right, #FFFFFF 0%, #DF15AA 35%, #AACFEF 100%)
```

La posizione può anche essere omessa, nel qual caso l'asse viene suddiviso proporzionalmente fra i colori impostati. Ad esempi arcobaleno:

```
linear-gradient(to right, red, orange, yellow, green, blue, indigo, violet);
```

Gradiente-Radiale

```
background-image: radial-gradient(circle at 50% 25%, #FFF 0%, #00F 50%, #0AE 100%);
background-image:radial-gradient(ellipse at center,  #FFF 0%, #00F 50%, #0AE 100%);
```

Parametri:

- **Tipo** di gradiente. Può essere **`circle`** oppure **`ellipse`**, che è il valore di default.
  L'**aspect ratio** dell'ellisse dipende esclusivamente dalla dimensioni del contenitore.
  In caso di contenitore quadrato l'ellisse degenera in un cerchio
- Dopo il **Tipo**, tramite la clausola **`at`**, si può specificare il **punto di origine** del gradiente, che può essere una combinazione delle parole chiave: center, top, bottom, left, right oppure può essere espresso tramite coordinate percentuali. Il valore di default è : **`ellipse at center`**
- Seguono i colori di stop gestiti esattamente come nel caso precedente.

La proprietà `border-image`

Consente di utilizzare una immagine come bordo. Per avere un buon effetto l'immagine deve avere la forma di una cornice come la seguente che ha dimensioni 99 x 99 px *(dimensione comunque priva di importanza in quanto la larghezza effettiva della cornice potrà essere impostata tramite image-width).*

Il parametro più importante per la configurazione è **`border-image-slice.`**
Impostandolo al 100% l'immagine non viene tagliata e sarà rappresentata per intera ai 4 spigoli del box per una larghezza data dalla proprietà `border-image-width` espressa in percentuale rispetto alle dimensioni del box (typ 10%)
Impostandolo ad una valore = 50% l'immagine viene sostanzialmente tagliata in 4 parti uguali che verranno ciascuna rappresentata ai 4 spigoli del box
Impostandolo ad una valore < 50% l'immagine viene divisa in 9 parti: i 4 angoli (rombi rossi), i 4 lati (rombi chiari) e la parte centrale che viene resa trasparente. La parte centrale di ogni lato viene ripetuta per l'intera larghezza/altezza del box.
Impostandolo ad una valore = 33% il taglio divide l'immagine esattamente in 9 parti uguali.
La parte centrale di ogni lato viene ripetuta per l'intera larghezza/altezza del box, come nell'esempio:
Impostandolo ad una valore compreso tra 50% e 100% una porzione di immagine compare solo ai 4 spigoli (non sui lati). Con un valore del 90% su ognuno dei 4 spigoli verrà tagliato un 10% di immagine.

Esempio con image-slice:33%


```
div {
    width:300px; height:300px;
    padding:15px;
    border:30px black solid;
    border-image:url(border.png);
    border-image-slice:33%;
    border-image-width:30px;//30px per rombo
    border-image-repeat:round round; }
```

`border-image-width` rappresenta la larghezza di visualizzazione del bordo, ed è del tutto indipendente dalle reali dimensioni dell'immagine costituente il bordo.

`border-image-repeat` rappresenta il modo con cui deve essere ripetuto il bordo. **round** significa che l'immagine viene ripetuta. Il primo si riferisce al lato superiore e inferiore. Il secondo a quelli laterali. Impostando **stretch** invece che round, la parte centrale del lato viene deformata fino a coprire l'intero lato

Insieme a BORDER-IMAGE è raccomandato di utilizzale la proprietà **BORDER** assegnandogli uno spessore pari al Width del Border-Image. Anche se questo bordo non è visibile in quanto 'coperto' da Border-Image, agisce però a livello di dimensioni complessive, allargando il Box di 30 px compressivi.

Le @rules

Le cosiddette **@-rules (at-rules)** sono costrutti particolari **utilizzabili all'interno di una qualunque sezione di stile** ed introdotti dal simbolo della chiocciola. Possono essere utilizzate sia all'interno dell'attributo **<STYLE>**del file html sia all'interno di un foglio di stile.

Le @rules devono sempre presentare un punto e virgola di chiusura (come le CSS Property).

@charset

E' l'equivalente del meta tag charset di HTML. La sintassi è semplicissima:

```
@charset "UTF-8";
```

All'interno di un file CSS, deve essere la prima dichiarazione del file.

@media

Consente di impostare una **Media Query**. Analogo all'attributo **media** di HTML5.

```
@media (min-width:576px){
   h1 {color: red;}
}
```

@font-face

La direttiva **@font-face** consente di utilizzare nella pagina web font non standard salvati fisicamente all'interno della cartella del sito. Il font verrà scaricato insieme alla pagina che quindi verrà visualizzata correttamente anche quando il font non è installato all'interno del PC dell'utente.

font-family: indica il nome da assegnare al font

src: indica dove andare a cercare il file. Può assumerei valori **local()** che indica al browser di effettuare la ricerca all'interno del PC client, oppure **url()** che indica invece l'indirizzo web dove reperire il file. Questo indirizzo può essere assoluto (http://) oppure relativo a partire dalla cartella corrente.
All'interno di **src** è possibile specificare più sorgenti differenti separate da virgola.
Il browser proverà ad utilizzare il primo, se non trova il file passa al successivo e così via.

```
@font-face {
      font-family: 'Sansation';
      src: local("Sansation"),
            url("http://www.webserver.it/ Sansation.ttf");
            url('font/Sansation_Regular.ttf') format('truetype'),
            url('font/Sansation_Regular.woff') format('woff'),
            url('font/Sansation_Regular.svg') format('svg'),
            url('font/Sansation_Regular.eot');
}

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

Per quanto riguarda il **Bold** e l'**Italic**, per avere una migliore definizione è in genere preferibile creare

- un secondo Font-File contenente la versione **Bold** di tutti i vari caratteri
- un terzo Font-File contenente la versione **Italic** di tutti i vari caratteri.

Questi Font possono avere un nome diverso (ad esempio `SansationRegular` e `SansationBold`) invocando un file diverso a seconda dei casi oppure, meglio, possono avere lo stesso nome nel qual caso occorre gestire due appositi attributi che indicano se si tratta della versione Normal / Bold / Italic:

font-weight (può assumere i valori Normal / Bold)
font-style (può assumere i valori Normal/Italic/Oblique)

- Se questi due attributi non vengono impostati, il browser utilizzerà il font per tutte le tipologie di carattere (normal, bold e italic), interpolando automaticamente le versioni bold e italic
- Se si imposta **font-weight= normal,** il browser utilizzerà quel font ***soltanto*** per il carattere normal. Per il bold verifica se esiste un formato **font-weight=bold**.
  Se esiste lo usa, altrimenti esegue una interpolazione della versione normal.
- Idem per l'italic
- Nel caso in cui siano settati contemporaneamente sia **font-weight** che **font-style**, questo file verrà utilizzato per tutti quei caratteri su cui è impostato **sia** il bold **sia** l'italic

Nota: I formati **SVG** e **WOFF** NON sono installabili in windows. Provando a trascinare il font in Pannello di Controllo / Caratteri compare un errore di font non valido. Sono accettati solo **TTF** e **OTF**

@import

E' una alternativa all'elemento **<LINK>** per richiamare un foglio di stile dall'interno di una sezione di stile invece che direttamente dall'HTML. Dopo @import occorre utilizzare la funzione **url** del foglio di stile da utilizzare, espresso in fora relativa a partire dalla cartella corrente oppura in forma assoluta :

```
@import url("reset.css");
@import url("http://www.miosito.it/stili.css");
```

- **@import non** deve avere altre righe di stile davanti
- Dopo il path è possibile specificare il supporto cui applicare il CSS, in modo simile all'attributo **media**. `@import url("stili.css") screen, print;`

@import può essere utilizzato comodamente anche per accedere ad un font tramite CDN.

https:// fonts.google.com mette a disposizione moltissimi font alcuni gratuiti altri a pagamento. Cliccando sul font si apre una **pagina di dettaglio** in cui, scendendo nella pagina, sono visualizzati tutti i vari formati disponibili (es thin 100, thin 100 bold, etc). E' possibile :

- selezionare un singolo formato, cliccando sul +, in modo da caricarlo nel 'carrello'
- svuotare il carrello dei formati selezionati
- fare il download d tutti i formati presenti nel carrello
- copiare il link HTML (<link>) oppure CSS (@import) per utilizzare il CDN con riferimento ai soli formati presenti nel carrello (es thin 100, thin 100 bold, etc).
- scaricare l'intera famiglia utilizzando il pulsante in alto **Download family**.

Copiando la URL all'interno del browser si vedono bene i formati linkati e soprattutto il nome del `font-family` da utilizzare poi all'interno del codice CSS

Esempio

Il seguente link

```
@import url('https://fonts.googleapis.com/css2?family=Imperial+Script');
```

punta al seguente file

```
@font-face {
  font-family: 'Imperial Script';
  font-style: normal;
  font-weight: 400;
  src:
url(https://fonts.gstatic.com/s/imperialscript/v3/5DCPAKrpzy_H98IV2ISnZBbGrVNfNePkjts.woff2) format('woff2');
  unicode-range: U+0102-0103, U+0110-0111, U+0128-0129, U+0168-0169, U+01A0-01A1, U+01AF-01B0, U+1EA0-1EF9, U+20AB; }
```

2D – Transform

La proprietà **TRANSFORM** consente di cambiare **posizione, dimensione e forma** di un elemento della pagina. A differenza di **`width`** e **`height`** (modificati ad es in corrispondenza di un effetto **:hover**) che provocano il r**iposizionamento** degli elementi circostanti,
la proprietà TRANSFORM non ha alcun effetto sugli elementi circostanti.

La proprietà TRANSFORM non è applicabile agli elementi inline (sì per i float e gli inline-block)

I metodi translate(), translateX(), translateY()

Trasla l'oggetto lungo il SUO asse X e/o lungo il SUO asse Y

```
transform:translateX(50px);
transform:translateY(100px);
transform:translate(50px, 100px); // 50px verso destra, 100px verso il basso
```

Per default l'**ORIGINE** degli assi di un oggetto corrisponde al suo **Baricentro** fisico.
La traslazione sposta il Baricentro ma NON sposta l'ORIGINE degli assi per cui, dopo una traslazione, l'oggetto si troverà con il baricentro fuori dall'ORIGINE.

I metodi rotateZ() rotateX() rotateY()

rotateZ esegue una rotazione **clockwise** intorno all'asse Z uscente dal foglio, con centro nel baricentro fisico. Valori negativi provocano una rotazione anticlockwise.

```
div { transform:rotateZ(30deg); }
```

La rotazione **NON** sposta il BARICENTRO, ma **sposta invece gli assi** che seguono la rotazione dell'oggetto. Dopo una rotazione clockwise di 90°, l'asse X punterà verso il basso, Y verso sinistra.

rotateX provoca una rotazione dell'oggetto intorno all'asse X.
rotateY provoca una rotazione dell'oggetto intorno all'asse Y.

La proprietà **`backface-visibility:hidden`** fa si che, dopo una rotazione di 180° sull'asse X oppure Y, il contenuto presente all'interno dell'oggetto non risulti più visibile.

I metodi scale(), scaleX(), scaleY()

Eseguono un ingrandimento / riduzione dell'oggetto nella direzione del suo asse X e/o del suo asse Y. Il baricentro dell'oggetto rimane FERMO.

```
transform:scaleX(2);
transform:scaleY(1.5);
transform:scale(2, 1.5);
```

Nota: a differenza di width e height, **scale** provoca anche la scalatura del testo.
Per cui se scaleX e scaleY hanno valori diversi, il testo viene deformato.

Applicazione contemporanea di più metodi

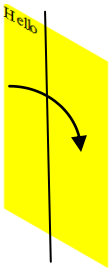
Più metodi possono essere applicati contemporaneamente ma **SOLO sulla stessa riga**
In caso di scrittura su più righe, come sempre l'ultima riga nasconde le riga precedenti.

```
transform: translateX(300px) rotate(10deg) scale(1.5);
```

I metodi skew(), skewX(), skewY()

Esegue una rotazione **clockwise** intorno all'asse X e/o intorno all'asse Y

```
transform:skewY(20deg)
transform:skewX(10deg)
transform:skew(10deg, 20deg);
```



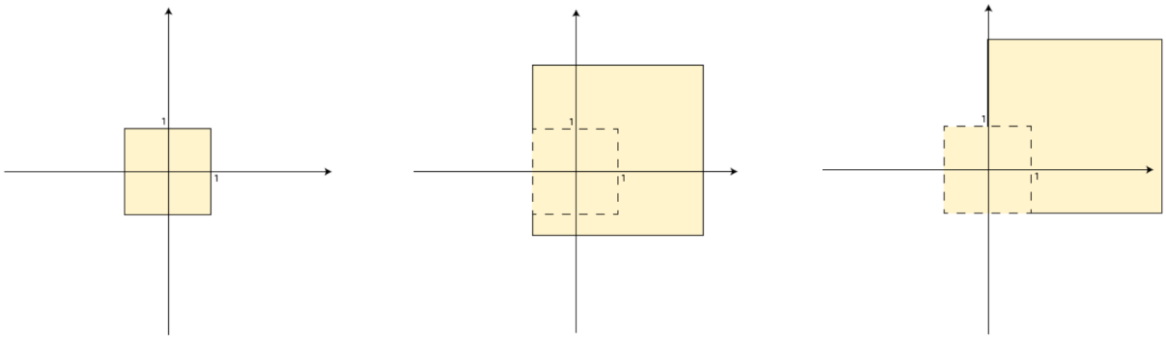
Notare che, mentre il metodo **translate** gode della proprietà **commutativa** (cioè facendo prima translateX o translateY il risultato non cambia), per skew non è così. Non solo ma `skew(15, 15)` produce un risultato intermedio tra skewX(15) skewY(15) e skewY(15) skewX(15) perché nel caso skew(15, 15) le due animazioni vengono fatte sostanzialmente in parallelo e non in sequenza.

Nota sulla gestione dell'origine

Tutte le TRANSFORM sono applicate rispetto all'ORIGINE degli assi. Se il Baricentro coincide con l'ORIGINE l'oggetto (ad es un quadrato di lato 2 con baricentro nell'origine), applicando una SCALE(2) si ottiene un quadrato di lato 4 sempre con baricentro nell'origine.

Ma se il baricentro è fuori dall'origine (ad es il quadrato si trova nel primo quadrante in alto a destra), quindi con origine BOTTOM LEFT, lo SCALE(2) raddoppia VERSO DESTRA e VERSO L'ALTO.

Esempio:Si supponga di avere un quadrato di lato 2 con baricentro nell'origine (*figura 1*)
Applicando uno scale(2) e poi una traslazione (1, 0.5) si ottiene il risultato di *figura 2*, cioè un quadrato di lato 4 con baricentro (1, 0.5).
Applicando invece le stesse trasformazioni in ordine invertito si ottiene il risultato di *figura 3* in cui, al momento dell'ingrandimento il quadrato ha baricentro fuori dall'ORIGINE. L'ingrandimento X avverrà completamente verso destra, mentre l'ingrandimento Y avverrà per ¼ verso il basso e per ¾ verso l'alto, ed il nuovo baricentro sarà (2, 1)



La proprietà `transform-origin`

In qualunque momento è possibile modificare l'ORIGINE di un oggetto nel modo seguente:

```
transform-origin: center center;
```

il 1° parametro rappresenta il valore **x** e può assumere i valori left, center, right, val, val%
il 2° parametro rappresenta il valore **y** può assumere i valori top, center, bottom, val, val%

`transform-origin` deve essere impostato sull'elemento sul quale viene applicato il transform.
Tutte le `transform` su quell'elemento verranno eseguite con riferimento alla nuova origine.

Il metodo `matrix()`

Il metodo matrix applica una trasformazione matriciale all'oggetto per eseguire contemporaneamente le azioni di **`translate, scale e skew`**. Al metodo matrix() occorre passare i seguenti 6 valori :

```
transform:matrix(a, b, c, d, e, f);
```

a = fattore di scala sull'asse X (esegue un transform.scale)
d = fattore di scala sull'asse Y (esegue un transform.scale)

e = traslazione sull'asse X (esegue un transform.translate)
f = traslazione sull'asse Y (esegue un transform. translate)

b = numero puro che indica un coefficiente di rotazione intorno all'asse Y (typ tra 0 e 1)
c = numero puro che indica un coefficiente di rotazione intorno all'asse X (typ tra 0 e 1)

3D - Transforms

La proprietà `perspective`

Per poter utilizzare i metodi 3D su un oggetto, occorre prima definire una **PROSPETTIVA** sull'oggetto.
Si può definire **oggetto tridimensionale** un qualunque oggetto dotato di **prospettiva**.
La proprietà **`perspective`** definisce la **profondità** dell'oggetto cioè la sua distanza in px dallo schermo

- Maggiore è questa distanza, più lontano risulterà l'oggetto e minore sarà l'effetto 3D.
- Se questa distanza è troppo piccola (rispetto alle dimensioni dell'oggetto), l'oggetto NON viene rappresentato. La perspective deve essere almeno DOPPIA o TRIPLA rispetto alle dimensioni dell'oggetto. Se

Se si vogliono applicare degli effetti tridimensionali a certi elementi, la proprietà **`perspective`** deve essere applicata al contenitore di quegli elementi. Cioè la proprietà **`perspective`** non agisce sull' elemento a cui viene applicata ma sugli elementi figli, ai quali si potranno applicare effetti tridimensionali

```
div  { perspective:400px; }
```

Se non si imposta la proprietà **`perspective`**, è come se avesse un valore infinito, per cui **si perde** l'effetto 3D

Esiste anche un metodo **`perspective()`** applicabile alla proprietà **`transform`** analogo al precedente che però agisce sull'elemento medesimo al quale viene applicato. Può quindi essere utilizzato direttamente sull'elemento al quale si vogliono applicare gli effetti tridimensionali

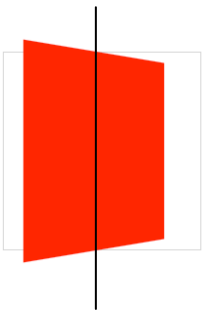
```
div  { transform: perspective(400px) rotateY(45deg);  }
```

Esempio:

```
<section class="container">
      <div class="box"></div>
</section>
```

```
.container {
    width:100px; height: 100px;
    perspective: 400px;
}

.box {
    width:100%;  height: 100%;
    background: red;
    transform: rotateY(45deg);
}
```



Per default, il punto di fuga (vanishing point) della prospettiva è posizionato al centro dell'oggetto . E' possibile cambiare questa posizione mediante la proprietà **perspective-origin** che ha gli stessi parametri della proprietà **transform-origin**.

Metodi di trasformazione 3D cioè applicati a oggetti dotati di perspective

`translateZ(z)`
// provoca una traslazione dell'oggetto tridimensionale lungo l'asse Z. In pratica viene variato il valor di prospettiva. Valori positivi avvicinano l'oggetto all'utente, valori negativi lo allontanano.

`scaleZ(z)`
// Riscalatura nella dimensione Z.

`rotateX rotateY rotateZ`
Nel caso di oggetti privi di perspective coincidono con i metodi 2D

`translate3d(x,y,z)`
// Traslazione contemporanea lungo i 3 assi.

`rotate3d(x,y,z)`
// provoca una rotazione dell'oggetto tridimensionale intorno a tutti e tre gli assi contemporaneamente.

`scale3d(x,y,z)`
// Riscalatura contemporanea su tutte e 3 le dimensioni

Nota: Quando si imposta la perspective su un genitore, vengono definiti sul contenitore stesso gli assi X Y Z che passano per il centro del contenitore. Se all'interno del contenitore ci sono più figli di primo livello, un eventuale **rotateY** su di essi non verrà eseguito rispetto all'asse Y dell'elemento medesimo, ma rispetto all'asse Y unico che passa al centro del contenitore. Idem per **rotateX**

`transform-style: preserve-3d`

La proprietà **`perspective`** deve essere applicata al genitore in modo da poter applicare delle trasformazioni 3D ai tag figli. La proprietà **`transform-style: preserve-3d`** applicata sui figli fa sì che anche i nipoti possano implementare delle trasformazioni 3D indipendenti. In caso contrario (il default è **`transform-style: flat`**) i nipoti saranno "flat" cioè piatti, come 'incollati' sui figli.

La proprietà `transition`

La proprietà **`transition`** fa si che le variazioni del valore di una qualsiasi proprietà avvengano **gradualmente** nel tempo specificato. (default 0, quindi effetto **non** applicato).

La proprietà transition di solito applicata all'interno del tag a cui si riferisce. In questo modo la transizione verrà applicata tutte le volte che la proprietà indicata cambierà di valore. Esempio:

```
div {width:100px; height:100px; transition: width 2s;}
```

Le variazioni su width *(ad es all'interno di :hover)* verranno eseguite con una transizione di 2 secondi.

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

In questo esempio in corrispondenza dell'hover, height passa istantaneamente da 100 a 300, mentre width si allargherà gradualmente nell'arco di 2 secondi. Sul mouseout ci sarà la transizione inversa.

Animazioni su più proprietà

E' possibile animare contemporaneamente più property di uno stesso oggetto. In questo caso il nome **transiction** deve essere impostato una sola volta, passando come valore un elenco di property separate da virgola. Dopo ogni property si può assegnare un tempo specifico:

```
div { transition: width 2s,  height 2s,  transform 3s; }
div:hover {width:200px; height:200px;  transform:rotate(180deg); }
```

E' anche possibile assegnare soltanto un singolo tempo, nel qual caso **TUTTE** le property dell'oggetto, in caso di cambiamento di valore, eseguiranno una transiction nel tempo indicato:

```
transition: 2s;
```

Sono anche disponibili le seguenti sintassi, però piuttosto inutili in quanto

- La prima è una semplice ridondanza della precedente
- La seconda anima comunque TUTTE le proprietà, quindi è addirittura fuorviante

```
transition: all 2s;
transition: width,  height,  transform,  2s;
```

Note:

- Se la transition viene scritta dentro hover, verrà fatta **solo** sul mouseOver e **NON** sul mouseOUT
- La prop transition **NON** è utilizzabile su background-image ma si può usare un trucco sovrapponendo 2 img
- Le transition alle volte si verificano anche al page load creando effetti abbastanza indesiderati. Questo problema potrebbe essere risolto spostando la proprietà **transition** direttamente all'interno di :hover. In questo modo si risolve il problema del page-load, perché la transizione verrà eseguita SOLTANTO in corrispondenza dell'hover, però nasce un problema ancora più grave, cioè che non viene più eseguita l'animazione di ritorno: al termine dell'hover l'elemento rimane nello stato di over senza più ritornare nello stato di riposo.

  ```
  div:hover { width:300px;  height:300px;  transition: width 2s,  height 2s; }
  ```

  La soluzione al problema del page-load è rappresentata da un piccolo script che, al termine del caricamento della pagina, rimuove una classe .preload applicata al tag body. Questa classe avrà il seguente aspetto: `.preload * { transition:none !important; }`

Forma completa della proprietà transiction

In realtà i parametri accettati dalla proprietà **transiction** sono 4:

transition-property	Nome della/delle proprietà a cui su vuole applicare l'effetto
transition-duration	Durata della transizione. Default 0
transition-timing-function	**EASE significa rallentamento** (si legge IS) Indica la curva di velocità della transizione e può assumere i valori: **ease-in** (rallenta in avvio) **ease-out** (rallenta nel finale); **ease-in-out**; (rallenta sia in avvio che nel finale) **linear**; velocità lineare. In molti casi è la soluzione migliore **ease** (**default**, rallenta nella parte centrale)
transition-delay	Ritardo nell'avvio della transizione (sia in apertura che in chiusura). 0s

I quattro valori possono essere espressi separatamente oppure sulla stessa riga:

```
div {  transition-property: width;
       transition-duration: 1s;
       transition-timing-function: linear;
       transition-delay: 2s;     }

       transition: width 1s linear 2s, height 2s ease 3s;
```

Animazioni

Le animazioni nascono come alternativa a Flash nel contesto del mobile, su dispositivi che non hanno la potenza di calcolo di un computer e per i quali il consumo di risorse deve ridursi al minimo.

Definizione dell'Animazione.

```
#myDiv {
       animation-name: animazione1;
       animation-duration: 4s;
       animation-timing-function: linear;
       animation-delay: 2s
       animation-iteration-count: infinite;
}
```

oppure

```
#myDiv {
       animation: animazione1 4s linear 2s infinite;
}
```

Significato dei parametri

animation-name Con la proprietà animation-name si definisce il nome dell'animazione da associare ad un elemento. Il nome deve corrispondere a quello impostato nella regola @-keyframes.
animation-duration **I**mposta la durata dell'animazione. Il valore è espresso in secondi.
animation-timing-function analoga alla proprietà transition-timing-function vista nelle transizioni; descrive l'EASE, cioè accelerazioni e rallentamenti.
animation-delay Consente di impostare un ritardo espresso in sec sull'avvio dell'animazione **Default = 0**
animation-iteration-count La proprietà animation-iteration-count è usata per impostare il numero di volte che un'animazione sarà ripetuta. Il valore è un numero intero >= 1 oppure la parola **infinite**, con cui si crea un loop infinito**. Il valore di default è 1.**
animation-direction Consente che l'animazione venga eseguita in ordine inverso. A tal fine occorre assegnare alla proprietà il valore Reverse. **Default ="normal"**. **Alternate** esegue un movimento in una direzione e il movimento successivo nella direzione opposta.
animation-play-state Specifies whether the animation is running or paused. **Default "running"**

Definizione dei keyframes

Per definire la sequenza delle azioni da svolgere si usa **@keyframes** in cui occorre definire le property css da modificare in corrispondenza di ogni keystop

```
@keyframes animazione1 {
    0% {background-color: red;   opacity: 1.0; }
  100% {background-color: blue;  opacity: 0.75; }
}
```

Subito dopo @-keyframes va inserito il **nome dell'animazione** (nell'esempio 'animazione1').
Tutto quello che segue nella dichiarazione va racchiuso tra parentesi graffe.

All'interno vanno definiti i comportamenti dell'animazione. Ciascuna dichiarazione corrisponde a un **keyframe** dell'animazione in cui si definiscono l'aspetto e/o la posizione dell'oggetto.

In qualunque animazione vanno definiti almeno 2 stati, quello iniziale e quello finale:

- Lo stato iniziale va dichiarato con il valore **0%** oppure con la parola chiave **from**.
- Lo stato finale con il valore **100%** oppure con la parola chiave **to**.

Tra i due keyframe necessari, 0% e 100%, si possono inserire altri fotogrammi chiave, sempre definiti attraverso valori percentuale.

- Se non si definisce lo stato iniziale, l'animazione parte dal valore di riposo (se esiste)
- Se non si definisce lo stato finale, l'animazione mantiene in modo statico il valore dell'ultimo keystop

Supponendo di creare un'animazione di 10 sec, impostando il primo fotogramma chiave al 33%, esso entrerà in azione al 33% di 10 sec, ovvero dopo 3,3 secondi. E così via.

NB: se si desidera creare una animazione fluida che si ripeta all'infinito, deve necessariamente esistere il keystop 100% in cui devono essere impostati gli stessi valori dei keystop 0%

La Gestione di Colonne Multiple

A partire da un certo contenitore si possono assegnare le seguenti proprietà:

- **column-count** Numero di colonne
- **column-width** Larghezza di ogni colonna
- **column-gap** Spaziatura fra le colonne
- **column-rule** Crea un bordo nello spazio (gap) tra le colonne. La sintassi è la stessa di border e consente di definire larghezza, stile e colore del bordo.
- **columns** short hand delle prime 2 proprietà (column-count e column-width)**.**

Es:

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

`column-count` e **`column-width`** in genere sono alternativi. Fissato uno si lascia al browser il compito di calcolare automaticamente l'altro.

Assegnando **column-width** e **column-gap** calcola il numero di colonne da disegnare.
Il valore column-width indica la larghezza ottimale delle colonne, nel senso che il browser potrà allargare o restringere le dimensioni di quel tanto che è necessario ad adattare le colonne al layout. Nell'esempio, essendo la larghezza del div contenitore pari a 750px e avendo usato per column-gap un valore di 25px, le colonne saranno due con larghezza rispettivamente pari a 362px. Impostando column-gap = 80px, le colonne saranno una sola in quanto due sforerebbero la soglia di 750px.

Esistono altre 2 proprietà minori:
`column-span` consente di estendere un elemento (ad esempio un titolo) su più colonne. Accetta un valore numerico che indica il numero di colonne su cui espandersi, oppure **`all`** (tutte le colonne).
`column-fill` : `balance | auto` serve a bilanciare nel modo adeguato il contenuto tra le varie colonne. Nel caso di balance lo spazio eccedente viene equamente suddiviso fra le colonne, mentre nel caso di auto le colonne verranno valutate in sequenza ed il contenuto potrà influenzare la loro larghezza.

CSS Custom Properties

Una delle più interessanti novità proposte nei CSS3 sono le **custom properties** (dette anche CSS variables) che consentono in pratica di definire delle variabili in cui memorizzare dei valori che potranno poi essere riutilizzati più volte all'interno della pagina CSS, senza doverli ripetere ogni volta.

Una variabile può essere definita all'interno di un qualsiasi selettore CSS ed occorre anteporre al nome della variabile due trattini `--`. Normalmente vengono definite all'interno della pseudo-classe CSS **:root** che corrisponde sostanzialmente al tag **<html>**
La variabile può poi essere acceduta utilizzando la parola chiave **var**

```
:root { --backgroundColor: #000000; }
div { background-color: var(--backgroundColor); }
```

Se la variabile viene dichiarata all'interno di un contenitore CSS, la variabile sarà visibile per quell'elemento e per tutti gli elementi interni.

```
#wrapper { --myColor: #000000; }
div { background-color: var(--myColor); }
// assegnato a wrapper e a tutti i div interni a wrapper
```

Notare che si sarebbe potuto scrivere semplicemente:

```
#wrapper {
   --myColor: #000000;
   background-color: var(--myColor); }
```

Tutti gli elementi interni avrebbero ereditato il colore di sfondo indicato

Notare che se ad una classe **.inner** interna a **#wrapper** si assegna un valore diverso, tutti gli elementi interni a **.inner** (**.inner** compreso) assumeranno il nuovo valore, mentre gli elementi esterni assumeranno il valore definito da #wrapper.

```
#wrapper .inner{ --myColor: #777777; }
```

Impostazione di un tema dark / light

```
:root {
    --background-light: #f4f4f9;
    --background-dark: #333333;
    --text-light: #ffffff;
    --text-dark: #333333;
}

.dark-mode {
    --background-light: #333333;
    --background-dark:#f4f4f9;
    --text-dark: #ffffff;
    --text-light: #333333;
}

body {
    background-color: var(--background-light);
    color: var(--text-dark);
}

button.addEventListener('click', function(){
    body.classList.toggle('dark-mode');
```

@property

@property è una delle diverse **@rule** (cioè precedute dal carattere @) introdotte nei CSS3.

Tramite **@property** si possono gestire le variabili CSS con più precisione. È possibile specificare il tipo della proprietà, il valore di default ed eseguire un controllo sull'ereditarietà che nel caso precedente veniva sempre eseguita senza possibilità di scelta.

```
@property --myColor {
  syntax: "<color>";           /* variable  type */
  initial-value: #000000;
  inherits: false;
}
```

`@property` definisce soltanto la variabile, senza però inizializzarla. `initial-value` è una specie di valore statico che non è ben chiaro a che cosa serva. Dopo di che l'utilizzo rimane lo stesso di prima. L'utente dovrà inizializzare la variabile poi utilizzarla:

```
#wrapper {
  --myColor: #000000;
  background-color: var(--myColor);
}
```

Questa volta avendo impostato `inherits:false;` il valore non verrà ereditato dagli elementi interni

Rendering Engine

Inizialmente molte delle proprietà CSS3 proposte dallo standard non erano supportate dai browser che, per renderle utilizzabili facevano ricorso a dei **Rendering Engine** esterni i principali dei quali erano:

- Safari/Chrome → WebKit (prefisso **–webkit**)
- Firefox/Seamonkey → Gecko (prefisso **–moz**)
- Opera → Presto (prefisso **–o**)
- IE→ Filtri Microsoft (prefisso **–ms**)

A settembre 2020 tutte le proprietà CSS3 sembrerebbero essere supportate a livello nativo da tutti i browser. Qualora però qualche proprietà non dovesse ancora essere supportata occorre continuare a far ricorso ai **Rendering Engine.** Quando non si sa se una certa proprietà è supportata dai browser a livello nativo oppure no, occorre gestire la proprietà mediante 5 righe: la riga nativa e le righe relative a ciascuno dei motori precedenti.

Ad es, per la gestione dei bordi arrotondati, si potrebbe scrivere:

```
div { border-radius:25px;
      -webkit-border-radius:25px;
      -moz-border-radius:25px;
      -o-border-radius:25px; }
```