CSS 3

Rev. 2.4 del 08/01/2021

CSS 3

CSS 3

Dai CSS 2 ai CSS3 il numero di proprietà passa 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.

Rendering Engine

A settembre 2020 tutte le proprietà CSS3 sembrerebbero essere supportate a livello nativo da tutti i browser. Qualora però qualche proprietà non fosse ancora supportate a livello nativo, occorrerebbe far ricorso ai cosiddetti **Rendering Engine** dei vari browser. I Rendering Engine dei principal browser sono

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

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. IE utilizza per la gestione della grafica non i metodi standard CSS3 ma filtri proprietari Microsoft

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; }
```

Scroll morbido ad una ancora

```
html{  scroll-behavior:smooth; }
```

Questa proprietà fa si che, quando si clicca su un collegamento ipertestuale per raggiungere un'ancora di una pagina, lo scroll verso l'ancora non venga eseguito istantaneamente ma venga eseguito in modo graduale tramite uno scorrimento.

user-select:none applicato ad un **wrapper**, inibisce la selezione del testo negli elementi interni rispetto al wrapper medesimo.

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  #dedede; }
```

- 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.
- il Quinto 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.).

Aggiungendo invece 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 10px  10px  #dedede; }
```

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 **text-transform**:**capitalize** mette la prima lettera del testo in maiuscolo.

La gestione degli sfondi

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`

Consente di visualizzare l'immagine soltanto su una porzione di sfondo del contenitore. Una volta specificate le dimensioni, l'immagine viene visualizzata **per intero** all'interno dell'area indicata.

- Il valore **`cover`** fa sì che l'immagine copra interamente l'area del contenitore. 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`** adatta l'immagine fino a quando una delle due dimensioni non arriva a coprire completamente l'area a disposizione. Sull'altra dimens rimarrà una parte di contenitore 'scoperto'. In pratica il contenitore conterrà l'intera immagine con dei bordi laterali grigi
- Si possono anche specificare direttamente le dimensioni (in pixel o in percentuale rispetto alle dimensioni del box). Es :
  `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 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`

Quando si specificano width e height ed un Bordo per un certo elemento, questo verrà rappresentato a video con una dimensione complessiva pari a **`width + padding + bordo`**. Idem per l'altezza.
La proprietà **`box-sizing`** può assumere solo 2 valori:

- **`content-box`**; (default) l'elemento continua a comportarsi secondo il Box-Model tipico di CSS2.
- **`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 aumentate adeguatamente*). **E' il valore di default 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%);
```

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%,  #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.

Altre Proprietà relative all'interfaccia utente

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 all'elemento selezionato** (: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.

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.

Articolo interessante: http://www.suburban-glory.com/blog?page=111

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. Esempio:

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

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

@import

è una alternativa all'elemento **<LINK>** per richiamare un altro foglio di stile da una sezione di stile. Dopo @import occorre utilizzare la funzione **`url`** a cui può essere passato come parametro:

- un indirizzo relativo `@import url("reset.css");`
- un indirizzo assoluto `@import url("http://www.miosito.it/stili.css");`

- @import **non** deve avere altre righe di stile davanti
- All'interno di @import è possibile anche specificare il supporto cui applicare il CSS, in modo simile a quanto visto a proposito dell'attributo **media**.
  `@import url("stili.css") screen, print;`

@charset

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

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

Se utilizzata in 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 ed 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 oppure l'altro a seconda dei casi, oppure possono anche avere lo stesso nome nel qual caso due appositi attributi

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

indicano se si tratta della versione Bold oppure Italic. Tutte le volte che in un testo sarà impostato **`font-weight= bold,`** il browser utilizzerà automaticamente il file relativo al bold. Nel caso in cui, all'interno di un file di Font siano settati contemporaneamente sia **`font-weight`** che **`font-style`**, questo file verrà automaticamente utilizzato per tutti quei caratteri su cui è impostato **sia** il bold **sia** l'italic o l'oblique

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**

2D - Transforms

La proprietà **TRANSFORM** consente di cambiare **dimensione, forma**, e **posizione** di un elemento della pagina. La modifica di **`width`** e **`height`** (ad esempio in corrispondenza di un effetto **:hover**) provoca il r**iposizionamento** degli elementi circostanti, mentre qualsiasi metodo applicato alla 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

```
div {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.

Il metodo rotate()

Esegue una rotazione **clockwise** dell'oggetto intorno all'asse Z uscente dal foglio, con centro nel baricentro fisico, senza provocare deformazioni all'oggetto. .

```
div { ……; transform:rotate(30deg); }
```

Valori negativi provocano una rotazione anticlockwise. La rotazione **NON** sposta il BARIECNTRO, ma **sposta invece gli assi** che seguono la rotazione dell'oggetto. Dopo una rotazione clockwise di 90°, l'asse X punterà verso il basso, mentre l'asse Y punterà verso sinistra. Se dopo la rotazione di 90° si fa una traslazioneX, la traslazione verrà eseguita verso il basso !

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

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

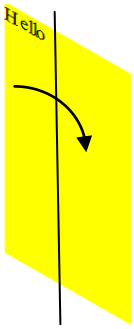
```
div { ……; transform:scale(2, 1.5); }
```

- a differenza di width e height, **scale** NON sposta gli elementi circostanti
- a differenza di width e height, **scale** provoca anche la scalatura del testo.
  Se scaleX e scaleY hanno valori diversi, il testo viene deformato. Abbastanza brutto.

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

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

```
div {transform:skew(0deg, 20deg); }   equivale a
div {transform:skewY(20deg); }
```



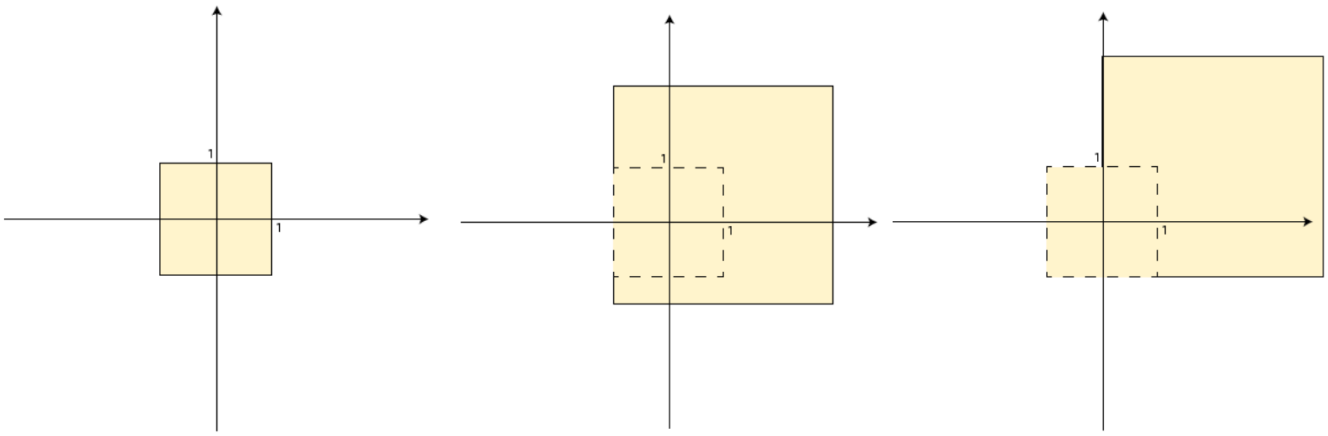
Più metodi possono essere applicati contemporaneamente **sulla stessa riga** nel modo seguente:
(in caso di scrittura su più righe, come sempre l'ultima riga nasconde le precedenti).

```
transform:  translateX(300px)  scale(2.0,1)  rotate(10deg);
```

Nota sulla gestione dell'origine

L'effetto di ingrandimento / riduzione viene applicato proporzionalmente 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 ancora 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), applicando una SCALE(2) il raddoppio sarà VERSO DESTRA e VERSO L'ALTO.

Si supponga ad esempio di avere un quadrato di lato 2 con baricentro nell'origine (*figura 1*)



Applicando un fattore di scala (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 le stesse trasformazioni in ordine invertito si ottiene il risultato di *figura 3* in cui, al momento dell'ingrandimento il quadrato ha baricentro fuori dalll'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 seguente modo:

```
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%

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 si vogliono applicare degli effetti tridimensionali a degli elementi della pagina, 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 potranno essere applicati degli effetti tridimensionali

```
div  { perspective:500px; }
```

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

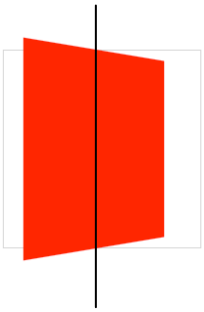
```
div  { transform: perspective(500px) rotateY(45deg);  }
```

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

rotateX

// provoca una rotazione dell'oggetto tridimensionale intorno all'asse X.

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

rotateY

// provoca una rotazione dell'oggetto tridimensionale intorno all'asse Y

rotateZ

// provoca una rotazione dell'oggetto tridimensionale intorno all'asse Z.
// Nel caso di oggetti privi di perspective coincide con il metodo 2D **rotate()**

rotate3d(x,y,z)

// provoca una rotazione dell'oggetto tridimensionale intorno a tutti e tre gli assi contemporaneamente.

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.

scale3d(x,y,z)

// Riscalatura contemporanea su tutte e 3 le dimensioni

translate3d(x,y,z)

// Traslazione contemporanea lungo i 3 assi.

transform-style: preserve-3d

Abbiamo visto che la proprietà **perspective** deve essere applicata al genitore in modo da poter applicare delle trasformazioni 3D ai tag figli. E se ci sono altri figli interni (nipoti) che anche loro devono eseguire delle trasformazioni 3D? In tal caso sui figli di primo livello occorre applicare la proprietà **transform-style: preserve-3d** la quale 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; }
```

La variazione di valore di una proprietà avviene tipicamente all'interno dell'evento :hover

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

Notare che, in questo codice, in corrispondenza dell'hover, height passa istantaneamente da 100 a 300, mentre width si allargherà gradualmente nell'arco di 2 secondi

Come sempre nel caso di hover, l'effetto avrà inizio nel momento in cui la proprietà cambia di valore, mentre verrà eseguita la transizione inversa nel momento in cui il mouse abbandona l'elemento

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, elencando le property separate da virgola:

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

Il valore **`all`** (che può anche essere omesso) consente di animare **tutte** le proprietà dell'elemento nel momento in cui queste dovessero cambiare di valore. Abbastanza eccessivo.

Nota: 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;  }

      // oppure
       transition: width 1s linear 2s;
```

Nota: La prop transition **NON** è utilizzabile su background-image ma si può usare un trucco sovrapponendo 2 img

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.

Si supponga di partire dal seguente blocco:

```
<div id="myDiv">    <p> Lorem ipsum </p>   </div>
```

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;   }
```

Sequenza delle azioni

Per definire la sequenza delle azioni da svolgere all'interno dell'animazione si una la Rule **@keyframes**

```
@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 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.

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.

Attributi relativi alle animazioni

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 può essere 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"**

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.

Flex Layout

```
#wrapper {
    display: flex;     /* esiste anche inline-flex     */
}
```

Consente di riposizionare gli elementi interni automaticamente in modo **responsive**.
Se gli elementi interni hanno una larghezza che **complessivamente** supera la larghezza del contenitore, allora la loro larghezza viene automaticamente ridotta in modo da visualizzarli tutti (a meno che non venga impostata la proprietà **flex-wrap:wrap**).

```
#wrapper div{
    width: 100px;
    height: 100px;
    margin: auto;
}
```

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

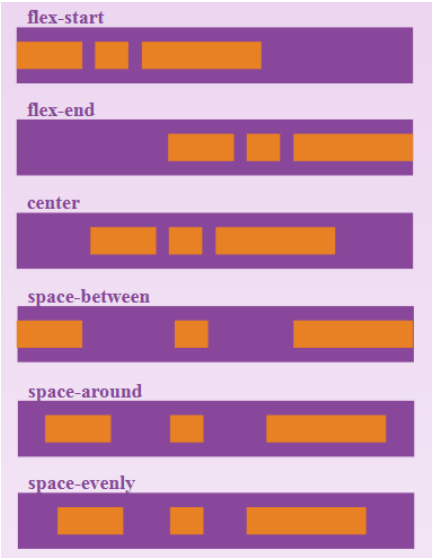
Proprietà del contenitore

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 oppure in ordine inverso. Se gli elementi presenti non ci stanno tutti sulla singola riga / colonna, vengono automaticamente ridotti di dimensioni.

flex-wrap valori possibili: nowrap | **wrap** | wrap-reverse In caso di wrap, raggiunta la fine della riga, automaticamente gli elementi successivi vengono posizionati sulla riga successiva senza essere ridimensionati. La disposizione degli elementi parte dall'alto verso il basso. Nel caso di wrap-reverse la disposizione parte dal basso verso l'alto. La proprietà margin:auto viene ignorata.

Centratura manuale degli elementi interni (alternativi a margin:auto)

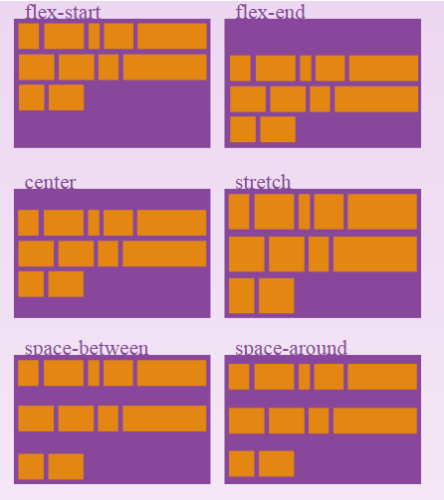
justify-content consente di impostare MANUALMENTE il tipo di spaziatura orizzontale da utilizzare per i vari elementi. Nel caso di margin:auto Viene ignorata
Valori possibili: flex-start | flex-end | center | space-between | space-around | space-evenly.
In pratica ridistruibuisce spazio libero di fine riga. Se a fine riga non c'è spazio libero, questa proprietà viene ignorata.



align-content consente di impostare MANUALMENTE il tipo di spaziatura verticale da utilizzare per i vari elementi. Nel caso di margin:auto Viene ignorata
Valori possibili: normal | flex-start | flex-end | center | space-between | space-around | stretch.
Analogo al precedente sull'asse opposto rispetto a quello utilizzato per la disposizione sequenziale degli elementi. Il valore **stretch** deforma l'elemento in modo che venga ricoperto l'intero spazio verticale. Viene riconosciuto SOLTANTO per gli elementi interni che NON dispongono della proprietà height. Il valore di default di `align-content` è:

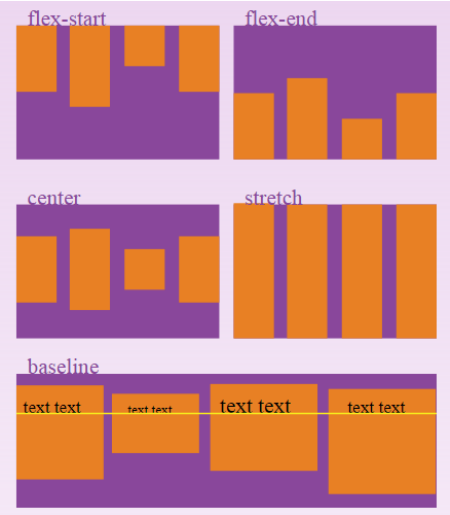
- **normal** per gli elementi che dispongono della proprietà height
- **stretch** per gli elementi che NON dispongono della proprietà height

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



Nell'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 rispetto alla riga. Nel caso di margin:auto Viene ignorata. Valori possibili: flex-start | flex-end | center | baseline | stretch . Consente di definire l'allineamento degli elementi sull'asse opposto rispetto a quello utilizzato per la disposizione sequenziale degli elementi.



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.

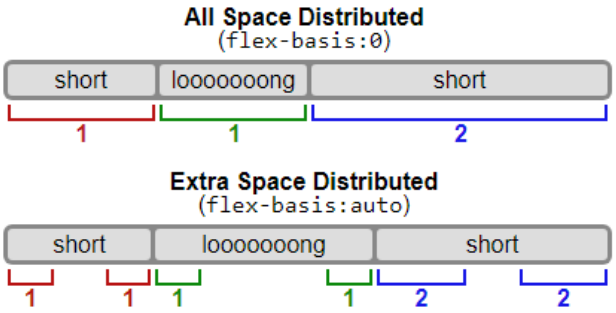
`flex-grow`: **`<number>`** Abilita gli elementi interni ad **aumentare** in modo proporzionale le proprie dimensioni in modo da saturare completamente l'eventuale spazio disponibile di fine riga. Se tutti gli elementi hanno lo stesso **`flex-grow,`** lo spazio libero verrà equamente suddiviso fra tutti 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 (default) fa sì che l'extra space venga suddiviso fra gli elementi sulla base della proprietà flex-grow.

flex-basis = 0 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)

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