



Finanziato  
dall'Unione europea  
NextGenerationEU



# CSS

layout&design



Cofinanziato  
dall'Unione europea



# HTML

**HTML**

**Navigazione**

- Home
- Elementi
- Attributi
- HTML5

HTML5  
lo standard per il web



**Indice**

**Indice Sezione**

- [linkElementi HTML](#)
- [linkElementi vuoti](#)
- [linkElementi block vs inline](#)

**Approfondimenti**

- [W3Copen in new](#)
- [MDNopen in new](#)

**Lorem ipsum**

**Fugiat ut voluptatum**

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Provident doloremque doloribus fugiat ut voluptatum a aliquam culpa quas sed dolore magni deleniti, voluptas odio ipsa laborum reprehenderit aliquid. Delectus

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Provident doloremque doloribus fugiat ut voluptatum a aliquam culpa quas sed dolore magni deleniti, voluptas odio ipsa laborum reprehenderit aliquid. Delectus, consectetur!

**Provident doloremque**

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Provident doloremque doloribus fugiat ut voluptatum a aliquam culpa quas sed dolore magni deleniti, voluptas odio ipsa laborum reprehenderit aliquid. Delectus, consectetur!

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Provident doloremque doloribus fugiat ut voluptatum a aliquam culpa quas sed dolore magni deleniti, voluptas odio ipsa laborum reprehenderit aliquid. Delectus, consectetur!

**Adipisicing elit**

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Provident doloremque doloribus fugiat ut voluptatum a aliquam culpa quas sed dolore magni deleniti, voluptas odio ipsa laborum reprehenderit aliquid. Delectus, consectetur!

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Provident doloremque doloribus fugiat ut voluptatum a aliquam culpa quas sed dolore magni deleniti, voluptas odio ipsa laborum reprehenderit aliquid. Delectus, consectetur!

Copyright®

# HTML + CSS

**HTML + CSS**

**Home Elementi Attributi HTML5**



**Indice Sezione**

- Elementi HTML
- Elementi vuoti
- Elementi block vs inline

**Approfondimenti**

- [W3Copen in new](#)
- [MDNopen in new](#)

**Lorem ipsum**

**Fugiat ut voluptatum**

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Provident doloremque doloribus fugiat ut voluptatum a aliquam culpa quas sed dolore magni deleniti, voluptas odio ipsa laborum reprehenderit aliquid. Delectus

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Provident doloremque doloribus fugiat ut voluptatum a aliquam culpa quas sed dolore magni deleniti, voluptas odio ipsa laborum reprehenderit aliquid. Delectus, consectetur!

**Provident doloremque**

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Provident doloremque doloribus fugiat ut voluptatum a aliquam culpa quas sed dolore magni deleniti, voluptas odio ipsa laborum reprehenderit aliquid. Delectus, consectetur!

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Provident doloremque doloribus fugiat ut voluptatum a aliquam culpa quas sed dolore magni deleniti, voluptas odio ipsa laborum reprehenderit aliquid. Delectus, consectetur!

**Adipisicing elit**

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Provident doloremque doloribus fugiat ut voluptatum a aliquam culpa quas sed dolore magni deleniti, voluptas odio ipsa laborum reprehenderit aliquid. Delectus, consectetur!

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Provident doloremque doloribus fugiat ut voluptatum a aliquam culpa quas sed dolore magni deleniti, voluptas odio ipsa laborum reprehenderit aliquid. Delectus, consectetur!

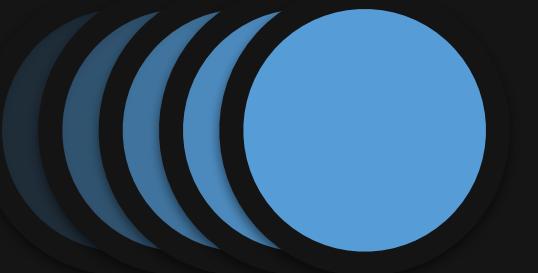
Copyright®

# Cosa puoi fare con i CSS

- Layout e impaginazione;
- Formattare i testi;
- Colori, Bordi e sfondi;
- Transizioni e animazioni;



Aa



# Tecniche di layout CSS

- CSS float;
- CSS flex box;
- CSS Grid Layout Module;
- framework UI;

## CSS Floats

Layout che utilizzano *la proprietà CSS float* per controllare il posizionamento degli elementi.

*Svantaggi:* bisogna prendere accorgimenti per controllare il flusso degli oggetti.

## CSS Flexbox

Flexbox è una modalità relativamente recente di layout introdotta con i CSS3 (2014~).

L'uso di flexbox garantisce che gli elementi si comportino in modo prevedibile quando il layout della pagina deve adattarsi a schermi di dimensioni diverse e a dispositivi di visualizzazione diversi.

Usa la *proprietà CSS display:flex* e altre *proprietà ad essa collegate*.

*Svantaggi:* non funziona in IE10 e precedenti.

## CSS Grid Layout Module

Il modulo *CSS Grid Layout* offre un sistema di layout basato su griglia, con righe e colonne, che semplifica la progettazione di pagine Web senza dover utilizzare float e posizionamento.

Si basa su un insieme di regole css recenti.

Usa la proprietà *CSS display: grid* e altre proprietà ad essa collegate.

*Svantaggi:* livello di supporto solo per browser recenti.

## Frameworks/Library CSS

Bootstrap, Foundation, Bulma, Ulkit, Semantic UI, ...

*Svantaggi:* librerie spesso corpose da caricare non sempre tutte necessarie...

## CSS (Cascading Style Sheets)

Il CSS viene utilizzato per definire gli stili per le tue pagine Web, inclusi il design, il layout e le variazioni di visualizzazione per diversi dispositivi e dimensioni dello schermo.

HTML non ha mai avuto lo scopo di contenere tag per la formattazione di una pagina web!

HTML è stato creato per descrivere il contenuto di una pagina Web:

```
<h1> Questo è un titolo </h1>  
<p> Questo è un paragrafo. </p>
```

Quando tag come `<font>` e attributi di colore sono stati aggiunti alle specifiche HTML 3.2, è iniziato un incubo per gli sviluppatori web. Lo sviluppo e la manutenzione di grandi siti Web è diventato un processo lungo e costoso.

## CSS (Cascading Style Sheets)

Per risolvere questo problema, il World Wide Web Consortium (W3C) ha creato i CSS.

- Il CSS ha rimosso la formattazione dello stile dalla pagina HTML!
- CSS risparmia molto lavoro!
- Le definizioni di stile vengono normalmente salvate in file *.css esterni*.
- Con un foglio di stile esterno, puoi cambiare l'aspetto di un intero sito web cambiando un solo file!

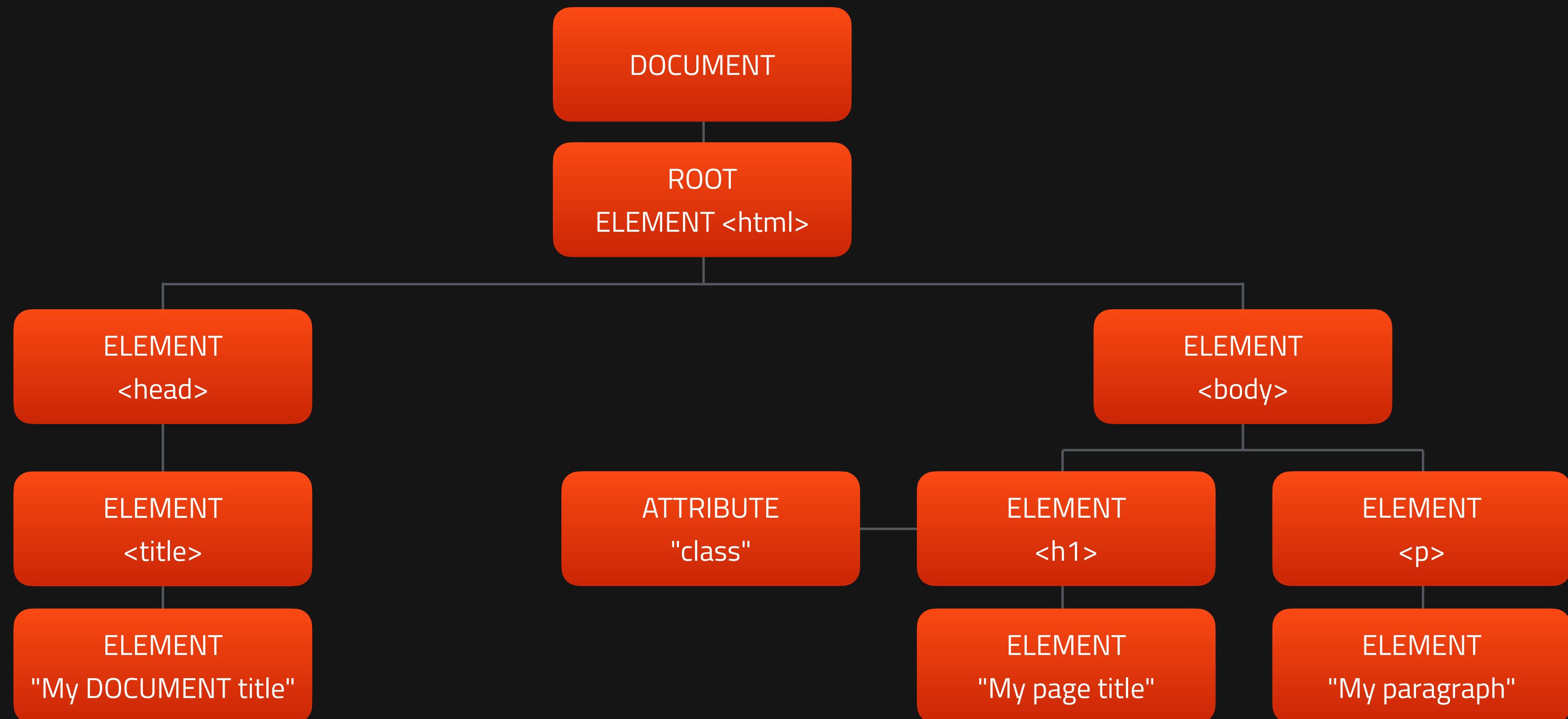
### Versioni CSS

Versione	Anno
CSS 1	Fine 1996
CSS 2	1998
CSS 2.1	2011 (2004)
CSS 3	2014 (1999)

# Il DOM HTML (DOCUMENT OBJECT MODEL)

Quando una pagina Web viene caricata, il browser crea un oggetto "Modello" della pagina.

Il modello DOM HTML è costruito come un albero di oggetti :



## Esempio di creazione del DOM

Esempio per vedere come uno snippet di codice HTML viene convertito in un DOM:

```
<p>
  Usiamo:
  <span>I fogli </span>
  <span>di stile</span>
  <span>a cascata</span>
</p>
```

Nel DOM, il nodo corrispondente al nostro elemento `<p>` è un genitore. I suoi figli sono un nodo di testo e i tre nodi corrispondenti ai nostri elementi `<span>`. I nodi SPAN sono anche genitori, con i nodi di testo come figli:

```
P
└── "Usiamo:"
└── SPAN
  └── "i fogli"
└── SPAN
  └── "di stile"
└── SPAN
  └── "a cascata"
```

## Come funzionano i CSS?

Quando un browser visualizza un documento, deve combinare il contenuto del documento con le sue informazioni sullo stile.

Elabora il documento in una serie di fasi, elencate di seguito in una versione semplificata:

- Il browser carica l'HTML
- Converte l' HTML nel DOM ( Document Object Model ).
- Il browser quindi recupera la maggior parte delle risorse a cui è collegato il documento HTML, come immagini incorporate, video e CSS collegati!
- Il browser analizza il CSS recuperato e ordina le diverse regole in base ai loro tipi di selettore  
Sulla base dei selettori che trova, stabilisce quali regole dovrebbero essere applicate a quali nodi nel DOM e associa loro lo stile come richiesto (questo passaggio intermedio è chiamato *albero di rendering*).
- L'albero di rendering è disposto nella struttura in cui dovrebbe apparire dopo che le regole gli sono state applicate.
- La visualizzazione della pagina viene mostrata sullo schermo (questa fase è chiamata *pittura*).

# Come funzionano i CSS?



## Sintassi CSS

Una serie di regole CSS è composta da un **selettore** e un **blocco di dichiarazioni**:



Il **selettore** intercetta l'elemento HTML a cui applicare lo stile.

I blocchi di **dichiarazione**, racchiusi tra parentesi graffe, sono separati da punto e virgola.

Ogni dichiarazione include un nome di **proprietà** CSS e un **valore**, separati da due punti.

Una dichiarazione CSS termina sempre con un punto e virgola. Per l'ultima istruzione il punto e virgola può essere omesso.

I selettori CSS vengono usati per selezionare gli elementi HTML in base al loro **tag** (nome di elemento), **id**, **classe**, **attributo** e altro (posizione).

*elenco selettori: <https://developer.mozilla.org/en-US/docs/Web/CSS/CSS>Selectors>*

## Cosa succede se il browser incontra una regola CSS che non capisce?

I browser possono non comprendere nuove funzionalità CSS recentemente introdotte nelle specifiche.

Se un browser trova un selettore o una dichiarazione CSS non riconosciuti, semplicemente li ignorerà passando alla parte successiva del codice.

**Questo comportamento avviene sia se c'è un errore di sintassi o sia se la funzionalità è troppo recente per il supporto del browser.**

Analogamente, se un selettore non è compreso, il browser ignorerà l'intera regola e passerà a quella successiva.

## Tre modi per inserire CSS

Esistono tre modi per inserire un foglio di stile nel documento html:

- Stile in linea
- Foglio di stile interno
- Foglio di stile esterno

### Stili in linea

Uno stile in linea può essere usato per applicare uno stile unico per un singolo elemento.

Per usare gli stili in linea, aggiungi l'attributo `style` all'elemento pertinente. L'attributo `style` può contenere qualsiasi proprietà CSS.

L'esempio seguente mostra come cambiare il colore e la dimensione del carattere di un elemento `<h1>` con l'uso dello stile in linea:

```
<h1 style="color:blue; font-size:16px">Questo è un titolo</h1>
```

## Foglio di stile interno

Gli stili interni sono definiti all'interno dell'elemento `<style>` , all'interno della sezione `<head>` di una pagina HTML:

```
<head>
  <style>
    body {
      color: #333;
    }

    h1 {
      color: blue;
    }
  </style>
</head>
```

## Foglio di stile esterno

Ogni pagina deve includere un riferimento al file del foglio di stile esterno all'interno dell'elemento `<link>`.

L'elemento `<link>` deve essere inserito nella sezione `<head>`\*:

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

Un foglio di stile esterno può essere scritto con qualsiasi editor di testi. Il file del foglio di stile deve essere salvato con l'estensione `.css`.

 Con un foglio di stile esterno, puoi cambiare l'aspetto di un intero sito cambiando un solo file!

\*In realtà `<link>` è consentito nel `<body>`. Tuttavia, questa non è una buona pratica: separa i tuoi elementi `<link>` dal contenuto del `<body>`, inserendoli nella sezione `<head>`.

## Fogli di stile multipli

È possibile inserire il riferimento a numerosi fogli di stile.

Fogli di stile con regole globali o specifiche a seconda del media che userà le regole.

Ogni riferimento deve essere specificato con il tag `<link>` nella sezione `<head>`

```
<head>

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

<link rel="stylesheet" type="text/css" href="css/print.css" media="print" >

<link rel="stylesheet" type="text/css" href="css/custom.css" media="screen" >

</head>
```

I fogli di stile vengono letti secondo il loro ordine di posizionamento nel codice, l'ultimo in elenco viene "letto" per... ultimo.

## Selettori

### selettore di elementi

Il selettore di elementi seleziona gli elementi in base al nome del *tag HTML*.

Puoi selezionare tutti gli elementi `<h1>` di un pagina (in questo caso, tutti gli elementi `<h1>` saranno allineati al centro e di colore blu):

```
h1 {  
    text-align: center;  
    color: blue  
}
```

## Elenco di selettori

Se hai elementi con le stesse definizioni di stile, come questo:

```
h1 {text-align: center; color: red}
```

```
h2 {text-align: center; color: red}
```

Puoi raggruppare i selettori per minimizzare il codice.

Per raggruppare i selettori basta separare ciascun selettore con una virgola. Questo ottimizza il css.

Di seguito l'esempio di ottimizzazione del codice precedente:

```
h1, h2 {text-align: center; color: red}
```

## Selettore di attributi: id

Il selettore id utilizza l'attributo **id** di un elemento HTML per selezionare un elemento specifico.

L'id di un elemento *deve essere univoco all'interno di una pagina*, il selettore selezionerà solo quell'elemento.

```
<p id="my-paragraph">Lorem ipsum</p>
```

Per selezionare un elemento con un ID specifico, nel CSS anteponi il simbolo **#**(cancelletto) all'**id** dell'elemento.

```
#my-paragraph {  
    text-align: center;  
    color: blue  
}
```

## Selettore di attributi: classe

Il selettore di classe utilizza l'attributo class degli elementi HTML per selezionare più elementi.

```
<p class="mark">Lorem ipsum</p>  
<p>Lorem ipsum</p>  
<p class="mark">Lorem ipsum</p>
```

Per selezionare elementi con una classe specifica, nel CSS anteponi il carattere punto [ . ] al nome della classe.

```
.mark {  
    color: red;  
    font-weight: bold  
}
```

**ATTENZIONE:** un nome di classe non può iniziare con un numero!

È possibile selezionare elementi HTML con attributi specifici o valori di attributo.

[attributo]

```
a[target] { background-color: yellow }
```

[attributo = "valore"]

```
a[target = "_blank"] { background-color: yellow }
```

```
[ title ~= "casa" ]
```

Seleziona tutti gli elementi con un attributo title *contenente* la parola intera "casa".

```
[ title ~= "casa" ] { border: 2px solid green; }
```

es, seleziona: casa, casa vacanza, non seleziona: villa

```
[ lang |= "en" ]
```

Seleziona tutti gli elementi con un valore di attributo lang che *inizia con prefisso "en"* o *"en-"*.

```
[ lang |= "en" ] { color: #782300; }
```

es, seleziona: "**en-us**", "**en-gb**", non seleziona: "us", "no"

Nota: il valore deve essere una parola intera, sola, come lang = "en", oppure seguita da un trattino (-), come lang = "en-us" (considera la stringa come prefisso).

[attributo ^ = "valore"] es: `a[ href ^= "https" ]`

Seleziona ogni elemento `<a>` il cui valore dell'attributo href inizia con il valore specificato: "https"

`a[ href ^= "https" ] {color: #782300;}`

es:

```
<a href="http://www.nonsicuro.com">non protetto</a>
<a href="https://www.sicuro.com">sicuro</a> <!-- selezionato -->
```

[attributo \$= "valore"] es: `a[ href $= ".pdf" ]`

Viene utilizzato per selezionare elementi il cui valore di attributo termina con un valore specificato.

`a[ href $= ".pdf" ]`

*Selezione ogni elemento `<a>` il cui valore dell'attributo href termina con ".pdf"*

[attributo \*= "valore"] es: `a[ href *= "min" ]`

Viene utilizzato per selezionare elementi il cui valore di attributo *contiene* una sottocadena specificata.

`a[ href *= "mdn" ]`

*Selezione ogni elemento `<a>` il cui valore dell'attributo href contiene la sotto-cadena "mdn".*

## Selettore di stati: Pseudo-classi

Una *pseudo-classe CSS* è una parola chiave *aggiunta ad un selettore* che specifica *lo stato speciale degli elementi selezionati*.

Ad esempio, `:hover` può essere usato per cambiare il colore dei link quando il puntatore dell'utente vi passa sopra.

```
/* Qualsiasi link sul quale passa il cursore dell'utente */
a:hover {
    color: blue;
}
```

La pseudo-classi permettono di applicare uno stile ad un elemento non solo in relazione al contenuto dell'alberatura del documento, ma anche in relazione a fattori esterni

- come la storia della navigazione ( `:visited` , per esempio),
- lo stato del suo contenuto ( `:checked` su determinati elementi di form),
- oppure la posizione del mouse `:hover` , che permette di sapere se il mouse si trova su un elemento o no.

sintassi delle pseudo-classi:

```
selector:pseudo-class { property : value }
```

Esempi:

```
a:link { color: #000000 }

a:hover { color: #ff0000 }

div:hover { background-color: blue }

input:focus { color: green}

input:read-only { color: darkgray}

li:first-child { color: blue}

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

elenco completo: <https://developer.mozilla.org/it/docs/Web/CSS/Pseudo-classes>

## Pseudo-elementi CSS

Uno **pseudo-elemento** CSS è una parola chiave aggiunta a un selettore che consente di applicare uno stile a una parte specifica degli elementi selezionati

```
p::first-letter{...dichiarazioni...}  
input[type="text"]::placeholder{...dichiarazioni...}
```

Puoi usare solo uno **pseudo-elemento** in un selettore. Deve apparire dopo i selettori semplici nell'istruzione.

Ad esempio, può essere utilizzato per:

- Definire lo stile della prima lettera, o linea, di un paragrafo;
- Generare e inserire del “contenuto” prima (::before) o dopo (::after) un elemento;
- Formattare il placeholder degli input (::placeholder).

## Pseudo-elementi CSS

sintassi degli pseudo-elementi, dopo il selettore si possono usare anche solo ":"

`selector::pseudo-element {property: value}`

`selector:pseudo-element {property: value}`

### Esempio

```
/* prima lettera di ogni elemento <p> */  
p::first-letter {  
    color: blue;  
}
```

elenco completo: <https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements>

## Selettori combinati

Per selezionare gli elementi del documento css possiamo combinare i selettori con alcuni simboli (*combinatori, combinatori*):

```
+ , ~ , > , [spazio come separatore]
```

## Combinatori discendenti

Il combinatore [ spazio come separatore] seleziona i nodi che sono figli (non solo i figli diretti, ma tutti i figli) dell'elemento precedentemente specificato.

```
p span { color: red }
```

Potete inserire una catena di selettori per intercettare l'elemento che vi interessa

```
ul li ul li { color: red }
```

# Combinatori di fratelli adiacenti

Il combinatore '+' seleziona i nodi che seguono immediatamente il precedente elemento specificato.

```
h2 + p { text-align: center }
```

corrisponderà al primo <p> che segue immediatamente un <h2> .

## Combinatori di fratelli generici

Il combinatore ' $\sim$ ' (simbolo *tilde*) seleziona i nodi che seguono (non necessariamente immediatamente) il precedente elemento specificato, se entrambi gli elementi condividono lo stesso genitore.

```
h2 ~ p { text-align: center }
```

## Combinatori di figli

Il combinatore ' $>$ ' seleziona i nodi che sono figli diretti del precedente elemento specificato.

```
p > span { text-align: center }
```

## Selettore universale

Il selettore universale serve a selezionare tutti gli elementi di un documento.

Si esprime con il carattere: \* (*asterisco*).

```
* { color: red }
```

La regola che abbiamo scritto assegna il colore rosso (red) a tutti gli elementi della pagina.

per imparare ad usare i selettori giocando prova...

<https://flukeout.github.io/>

<https://css-speedrun.netlify.app/>

## Ordine a cascata

Quale stile verrà utilizzato se è stato specificato più di uno stile per *lo stesso selettore* (elemento) HTML?

Gli stili, quando sono specificati per lo stesso selettore HTML, vengono applicati secondo un ordine di priorità definito dalla cascata. Gli stili si sovrappongono seguendo una gerarchia precisa, dove il primo stile elencato ha la priorità più alta:

1. Stile in linea (all'interno di un elemento HTML)
2. Fogli di stile interni, fogli di stile esterni
3. Valori predefiniti del browser

Pertanto, gli stili in linea definiti direttamente all'interno di un elemento HTML specifico hanno la priorità più alta e sostituiranno gli stili definiti nei fogli di stile interni, nei fogli di stile esterni o i valori predefiniti del browser, se c'è un conflitto.

## ORDINE a CASCATA

Le regole CSS vengono applicate *a cascata*: le regole dichiarate per ultime sovrascrivono le precedenti

Se alcune proprietà sono state definite per *lo stesso selettore* (elemento), verrà utilizzata l'ultima regola letta:

```
↓  
.my-paragraph {color:red}  
/* altre regole */  
.my-paragraph {color:black} /* sovrascrive la precedente */
```

È una regola frequente quella di inserire come ultimo foglio di stile quello con le proprie regole personalizzate (es: quando si usa un library con stili standard).

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/  
bootstrap.min.css" rel="stylesheet" integrity="sha384-  
giJF6kkoqNQ00vy+HMDP7az0uL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1"  
crossorigin="anonymous">  
<link href="custom.css" rel="stylesheet"> <!-- mio css con regole personalizzate -->
```

## Specificità CSS

Ma se ci sono due o più regole CSS che puntano allo stesso elemento con selettori diversi?

Il browser calcola la **specificità** per determinare quale regola utilizzare.

### Cos'è la specificità?

La specificità è un *punteggio / grado* che determina quali dichiarazioni di stile sono in definitiva applicate a un elemento.

Il selettore universale `*` ha *bassa specificità*, mentre i selettori `id` sono *altamente specifici*!

**Nota:** la specificità è spesso la ragione per cui le tue regole CSS non si applicano agli elementi, sebbene tu pensi che dovrebbero.

## Gerarchia di specificità

Ogni selettore ha il suo posto nella gerarchia di specificità. Esistono quattro categorie che definiscono il livello di specificità di un selettore:

- **Stili in linea** : uno stile in linea è collegato direttamente all'elemento.

Esempio:

```
<h1 style="color: #ffffff">
```

- **ID**
- **Classi, attributi e pseudo-classi**
- **Elementi e pseudo-elementi**

## Come calcolare la specificità?

Per calcolare la specificità di un selettore bisogna attribuire un punteggio secondo lo schema seguente:

- **1000** per l'attributo `style` ;
- **100** per ogni `id` ;
- **10** per ogni `attributo`, `classe` o `pseudo-classe` ,
- **1** per ogni nome di `elemento` o `pseudo-elemento`.

A: `h1 {color: blue}`

B: `#content h1 {color: #666 }`

C: `<div id="content"><h1 style="color: red">Titolo</h1></div>`

La specificità di A è 1: `h1`

La specificità di B è 101: `#content` = 100 + `h1` = 1.

La specificità di C è 1000: `style` = 1000.

Poiché  $1 < 101 < 1000$ , la terza regola (C) ha un livello maggiore di specificità e pertanto verrà applicata.

## La dichiarazione **!important**

La dichiarazione *!important*, fatta seguire alla regola da applicare, consente di forzare l'uso di quella regola indipendentemente dalla specificità

A:     `h1 {color: #000}`

B:     `#content h1 {color:#666 !important }`

C:     `<div id="content">`  
          `<h1 style="color:#ffffff">Titolo</h1>`  
        `</div>`

La dichiarazione *!important* rende altamente specifica quella regola, B sovrascrive C.

Per sovrascrivere B, dovremmo aggiungere ***!important*** alla regola C.

## Ereditarietà nel CSS

Alcuni valori di proprietà CSS impostati sugli elementi padre sono ereditati dai loro elementi figlio, mentre altri no.

Ad esempio, se imposta il *colore* e il *font-family* su un elemento (padre), anche ogni elemento al suo interno (figli) avrà lo stile di quel colore e carattere.

```
body {  
    color: blue;  
}
```

A meno che non si siano applicati valori di colore e carattere diversi direttamente a essi.

```
span {  
    color: black;  
}
```

## Ereditarietà nel CSS

Alcune proprietà non ereditano. Per esempio, se imposta width del 50% su un elemento, tutti i suoi discendenti non ottengono una larghezza del 50% della larghezza del loro genitore.

```
.main {  
    color: rebeccapurple;  
    border: 2px solid #ccc;  
    padding: 1em;  
}
```

Gli elementi all'interno della classe main erediteranno il colore ma non il border e il padding

```
<ul class="main">  
    <li>Item One</li>  
    <li>Item Two  
        <ul>  
            <li>2.1</li>  
            <li>2.2</li>  
        </ul>  
    </li>  
</ul>
```

## Ereditarietà nel CSS

Come faccio a sapere se una proprietà può essere ereditata oppure no?

Leggendo la documentazione.

Prendiamo il caso della proprietà `color`, nella documentazione c'è la sezione "Formal definition" che riporta se la proprietà viene ereditata oppure no:

[https://developer.mozilla.org/en-US/docs/Web/CSS/color#formal\\_definition](https://developer.mozilla.org/en-US/docs/Web/CSS/color#formal_definition)

Inherited: yes

Caso della proprietà `margin`:

[https://developer.mozilla.org/en-US/docs/Web/CSS/margin#formal\\_definition](https://developer.mozilla.org/en-US/docs/Web/CSS/margin#formal_definition)

Inherited: no

## BOX MODEL

Tutti gli elementi HTML possono essere considerati come box.

Nei CSS, "box model" viene usato quando si parla di design e layout.

*Il box model rappresenta essenzialmente una scatola che avvolge ogni elemento HTML.*

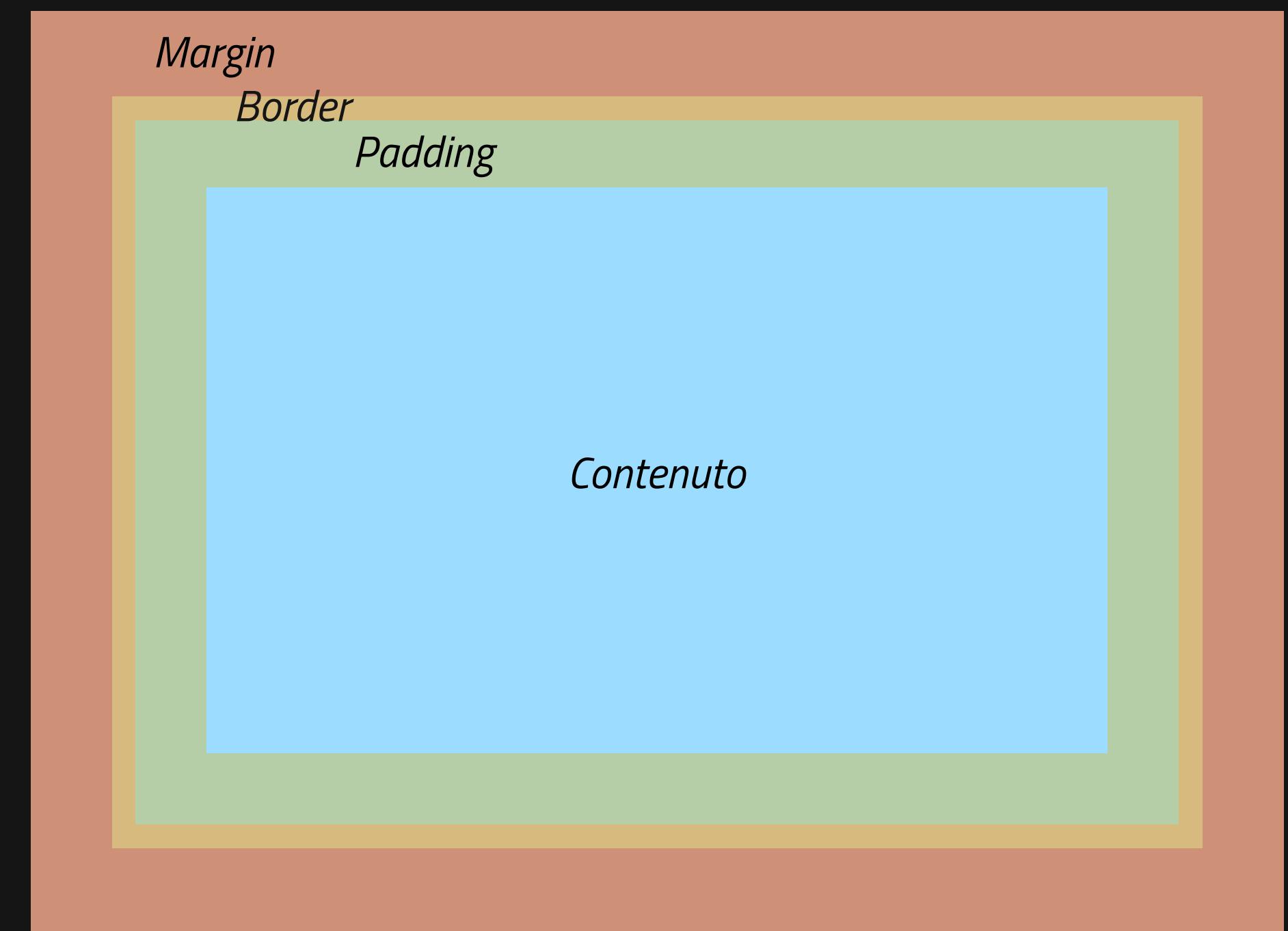
*BOX MODEL*

**Contenuto** : il contenuto del box, dove vengono visualizzati testo e immagini.

**Padding** : area attorno al contenuto. *Spazio tra elemento e contenuto.* L'area (imbottitura) è trasparente, interno al box.

**Border** : un bordo che circonda il riempimento e il contenuto, bordo del box.

**Margine** : area al di fuori del box. *Spazio tra gli elementi.* I margini sono trasparenti, esterno al box.



## Larghezza, altezza e spazio occupato da un elemento

```
div {  
    width: 300px;  
    border: 2px solid green;  
    padding: 24px;  
    margin: 8px  
}
```

Per impostare correttamente la larghezza e l'altezza di un elemento, è necessario ricordare come funziona il box model.

Quando si impostano le proprietà `width` e `height` di un elemento si imposta la larghezza e l'altezza dell'area del contenuto.

Per calcolare la *dimensione effettiva* di un elemento, è necessario aggiungere `padding` e `bordi`.

Per calcolare lo *spazio occupato* da un elemento invece è necessario aggiungere anche i *margini*.

Per disegnare un elemento `<div>` di larghezza totale pari a *350px*:

```
div {  
  
    width: 320px; /* 320px(larghezza contenuto) */  
  
    padding: 10px; /* + 20px(padding sinistro + padding destro) */  
  
    border: 5px solid gray; /* + 10px(bordo sinistro + destro) */  
  
    margin: 0  
}
```

larghezza totale di un elemento:

$$= \text{larghezza} + \text{padding sinistro} + \text{padding destro} + \text{bordo sinistro} + \text{bordo destro}$$

altezza totale di un elemento:

$$= \text{altezza} + \text{padding superiore} + \text{padding inferiore} + \text{bordo superiore} + \text{bordo inferiore}$$

## Uso della funzione calc();

per disegnare un elemento `<div>` di larghezza totale pari a *350px*:

```
div {  
    width: calc(350px - ((10px*2)+(5px*2)); /* 350px(larghezza) - (padding sx e dx +  
    border sx e dx) */  
    padding: 10px;  
    border: 5px solid gray;  
}
```

larghezza totale di un elemento:

$$= \text{larghezza} + \text{padding sinistro} + \text{padding destro} + \text{bordo sinistro} + \text{bordo destro}$$

altezza totale di un elemento:

$$= \text{altezza} + \text{padding superiore} + \text{padding inferiore} + \text{bordo superiore} + \text{bordo inferiore}$$

## Unità di misura nei CSS

I CSS hanno diverse unità di misura per esprimere la dimensione degli elementi.

Molte proprietà CSS usano valori di larghezza, margine, spaziatura, dimensione del carattere, larghezza del bordo, ecc.

La dimensione è un numero seguito da un'unità di misura, ad esempio: **10px** , **2em**

Non si può inserire uno spazio bianco tra il numero e l'unità. Se il valore è 0, l'unità di misura può essere omessa.

```
p { margin-left: 30 px } X  
p { margin-left: 30px }  
p { margin-left: 0 }
```

Per alcune proprietà CSS, sono consentite lunghezze negative.

Esistono due tipi di unità di lunghezza: **assoluta** e **relativa**.

## Lunghezze assolute

Le unità di lunghezza assolute sono fisse: una lunghezza espressa con misure assolute apparirà esattamente della dimensione definita.

La maggior parte di queste unità sono più utili quando vengono utilizzate per la stampa, piuttosto che per l'output su schermo.

Ad esempio, in genere si utilizzano cm(centimetri) sullo schermo. L'unico valore d'uso comune su schermo è il **px** (pixel).

Unit	Description	Equivalent to
cm	Centimeters	$1\text{cm} = 37.8\text{px} = 25.2/64\text{in}$
mm	Millimeters	$1\text{mm} = 1/10\text{th of } 1\text{cm}$
Q	Quarter-millimeters	$1\text{Q} = 1/40\text{th of } 1\text{cm}$
in	Inches	$1\text{in} = 2.54\text{cm} = 96\text{px}$
pc	Picas	$1\text{pc} = 1/6\text{th of } 1\text{in}$
pt	Points	$\text{pt} = 1/72\text{nd of } 1\text{in}$
px	Pixels	$1\text{px} = 1/96\text{th of } 1\text{in}$

## Lunghezze relative

Le unità di lunghezza relative sono *relative* a qualcosa' altro, la dimensione del carattere dell'elemento genitore o la dimensione del viewport.

Le unità di lunghezza relativa vengono ridimensionate meglio tra diversi mezzi di rendering.

Unit	Relative to
em	Dimensione del carattere del genitore, nel caso delle proprietà tipografiche come font-size, e dimensione del carattere dell'elemento stesso, nel caso di altre proprietà come width.
rem	Relativo alla dimensione del carattere dell'elemento radice (root element).
vw	1% della larghezza della finestra di visualizzazione (viewport). *
vh	1% dell'altezza della finestra di visualizzazione (viewport). *
vmin	1% della dimensione più piccola della finestra di visualizzazione (viewport).*
vmax	1% della dimensione più grande della finestra di visualizzazione (viewport).*
%	Relativo all'elemento genitore
...**	...**

\* Viewport = la dimensione della finestra del browser. Se il *viewport* è largo 10 cm, 1vw = 0,1 cm.

\*\* Elenco completo: [https://developer.mozilla.org/en-US/docs/Learn/CSS/Building\\_blocks/Values\\_and\\_units](https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Values_and_units)

## Imposta dimensione carattere con em

L'unità di misura **em** è raccomandata dal W3C, per consentire agli utenti di ridimensionare il testo (nel menu del browser).

**1em** è uguale alla dimensione del carattere corrente. La dimensione predefinita del testo nei browser è 16px.

Quindi,  $1\text{em} = 16\text{px}$ .

La dimensione può essere trasformata da *pixel* ad *em* utilizzando questa formula:  $\text{pixel} / 16 = \text{em}$

```
h1 {font-size: 2.5em} /* 40px/16=2.5em */
h2 {font-size: 1.875em} /* 30px/16=1.875em */
p {font-size: 0.875em} /* 14px/16=0.875em */
```

Nell'esempio sopra, la dimensione del testo in em consente di regolare la dimensione del testo in tutti i browser.

Sfortunatamente, c'è ancora un problema con le versioni precedenti di IE. Il testo diventa più grande di quando dovrebbe essere ingrandito e più piccolo di quando dovrebbe essere ridotto.

Soluzione: usare una combinazione di % ed em

La soluzione che funziona in tutti i browser è quella di impostare una dimensione del carattere predefinita in percentuale per l'elemento `<body>` e definire le dimensioni del font per gli altri elementi in em:

```
body {font-size: 100%}  
h1 {font-size: 2.5em}  
h2 {font-size: 1.875em}  
p {font-size: 0.875em}
```

Questo codice dovrebbe mostrare le stesse dimensioni del testo in tutti i browser e consentire a tutti i browser di ingrandire o ridimensionare il testo.

## Inserire commenti nel CSS

I commenti vengono utilizzati per spiegare il codice, tornano utili in fase di modifica.

I commenti vengono ignorati dai browser.

Un commento CSS inizia con `/*` e termina con `*/`

I commenti possono comprendere più righe:

```
p {  
    color: red;  
    /* Questo commento occupa una linea */  
    text-align: center;  
}  
/* Questo commento  
occupa più  
linee */
```

## margin

Le proprietà margin viene utilizzata per creare spazio attorno agli elementi.

Si possono specificare i margini per ciascun lato di un elemento:

- **margin-top**
- **margin-right**
- **margin-bottom**
- **margin-left**

Possono avere i seguenti valori:

- **auto**: il browser calcola il margine
- length - specifica un margine in **px**, **pt**, **cm**, ecc (es. margin-left: 14px;).
- **%** : specifica un margine in% della larghezza dell'elemento contenitore (es: margin-left: 5%;)
- **inherit**: specifica che il margine deve essere ereditato dall'elemento padre

Sono consentiti i valori negativi.

margin - shorthand\* (codice abbreviato)

Per abbreviare il codice, è possibile specificare tutte le proprietà del margine in una sola dichiarazione.

Se la proprietà margin ha quattro valori:

```
margin: 25px 50px 75px 100px;
```

T R B L

Top

Left

Right

Bottom



Se la proprietà margin ha tre valori:

```
margin: 25px 50px 75px;
```

T R/L B

Se la proprietà margin ha due valori:

```
margin: 25px 50px;
```

T/B R/L

Se la proprietà margin ha un valore:

```
margin: 25px;
```

All

**margin: 0 auto;** il browser calcola il margine, utilizzato per centrare i contenitori (div) orizzontalmente. L'elemento *dove* deve avere una larghezza definita.

## Crollo del margine

I margini superiore e inferiore degli elementi collassano in un unico margine che è uguale al più grande dei due margini.

\*nota: [https://developer.mozilla.org/en-US/docs/Web/CSS/Shorthand\\_properties](https://developer.mozilla.org/en-US/docs/Web/CSS/Shorthand_properties)

Le proprietà *border* consentono di specificare lo *stile*, la *larghezza* e il *colore del bordo* di un elemento.

## border-style

Specifica il tipo di bordo da visualizzare.

```
p.solid { border-style: solid}
```

Sono consentiti i seguenti valori:

**dotted** Definisce un bordo con pallini

**dashed** Definisce un bordo tratteggiato

**solid** Definisce un bordo solido

**double** Definisce un doppio bordo

**groove** Definisce un bordo scanalato 3D. L'effetto  
dipende dal valore del colore del bordo

**ridge** Definisce un bordo increspato 3D. L'effetto  
dipende dal valore del colore del bordo

**inset** Definisce un bordo interno 3D. L'effetto dipende  
dal valore del colore del bordo

**outset** Definisce un bordo iniziale 3D. L'effetto dipende  
dal valore del colore del bordo

**none** Non definisce alcun bordo

**hidden** Definisce un bordo nascosto

## border-width

Specifica la larghezza dei quattro bordi.

La larghezza può essere impostata come una dimensione specifica (in px, pt, cm, em, ecc.).

O utilizzando uno dei tre valori predefiniti:

**thin**, **medium**, o **thick**.

```
p.solid { border-style: solid; border-width: 5px}
p.solid { border-style: solid; border-width: medium}
p.dotted { border-style: dotted; border-width: 5px}
p.dotted { border-style: dotted; border-width: thin}
```

## Larghezze laterali specifiche

`border-width` può avere da uno a quattro valori (per il bordo superiore, il bordo destro, il bordo inferiore e il bordo sinistro):

```
p.solid { border-style: solid; border-width: 20px 5px}
```

## border-color

Imposta il colore dei quattro bordi.

```
p.solid{border-style: solid; border-color: red}
```

Colori laterali specifici

border-color può avere da uno a quattro valori (per il bordo superiore, il bordo destro, il bordo inferiore e il bordo sinistro).

```
p.solid{border-style: solid; border-color: red green blue yellow}
```

border-style deve essere impostato per visualizzare i bordi

## Bordi, lati individuali

È possibile specificare un bordo diverso per ogni lato.

Di seguito le proprietà per specificare ciascuno dei bordi (superiore, destro, inferiore e sinistro):

```
p {  
    border-top-style: dotted;  
    border-right-style: solid;  
    border-bottom-style: dotted;  
    border-left-style: solid;  
}
```

Funziona anche con `border-width` e `border-color`.

## Bordi (shorthand)

È possibile specificare le diverse proprietà in un'unica regola.

La proprietà border è una scorciatoia per le seguenti proprietà di bordi:

border-width

border-style [necessario]

border-color

```
p { border: 2px dotted #222 }
```

```
p { border: dotted }
```

Esempio: bordo sinistro

```
p { border-left: 6px solid red }
```

## border-radius

Definisce il raggio degli angoli dell'elemento, consente cioè di aggiungere angoli arrotondati agli elementi.

Questa proprietà può avere da uno a quattro valori.

```
p.radius{border-style: solid; border-radius: 25px; padding: 10px}
```

```
p.radius{border-style: solid; border-radius: 25px 10px; padding: 10px}
```

```
p.radius{border-style: solid; border-radius: 25px 10px 8px; padding: 10px}
```

Valgono le regole viste con `margin` e `padding` per tutte le proprietà `border`.

## border-radius

Questa proprietà è una scocciatoia per le proprietà CSS seguenti:

```
border-top-left-radius: solid;
```

```
border-top-right-radius: solid;
```

```
border-bottom-right-radius: solid;
```

```
border-bottom-left-radius: solid;
```

## padding

La proprietà padding viene utilizzata per generare spazio attorno al contenuto di un elemento, all'interno di qualsiasi bordo definito.

I CSS hanno proprietà per specificare il riempimento per ciascun lato di un elemento:

- padding-top
- padding-right
- padding-bottom
- padding-left

Tutte le proprietà di padding possono avere i seguenti valori:

- *length* - specifica un padding in **px**, **pt**, **cm**, ecc.
- **%**: specifica un riempimento in % della larghezza dell'elemento contenitore
- **inherit**: specifica che il padding deve essere ereditato dall'elemento padre
- **initial**: specifica che il padding deve essere quello iniziale

Non sono consentiti i valori negativi.

## padding - shorthand (codice abbreviato)

Per abbreviare il codice, è possibile specificare tutte le proprietà del padding in una sola dichiarazione.

Se la proprietà padding ha quattro valori:

```
padding: 25px 50px 75px 100px;
```

T R B L

Se la proprietà padding ha tre valori:

```
padding: 25px 50px 75px;
```

T R/L B

Se la proprietà padding ha due valori:

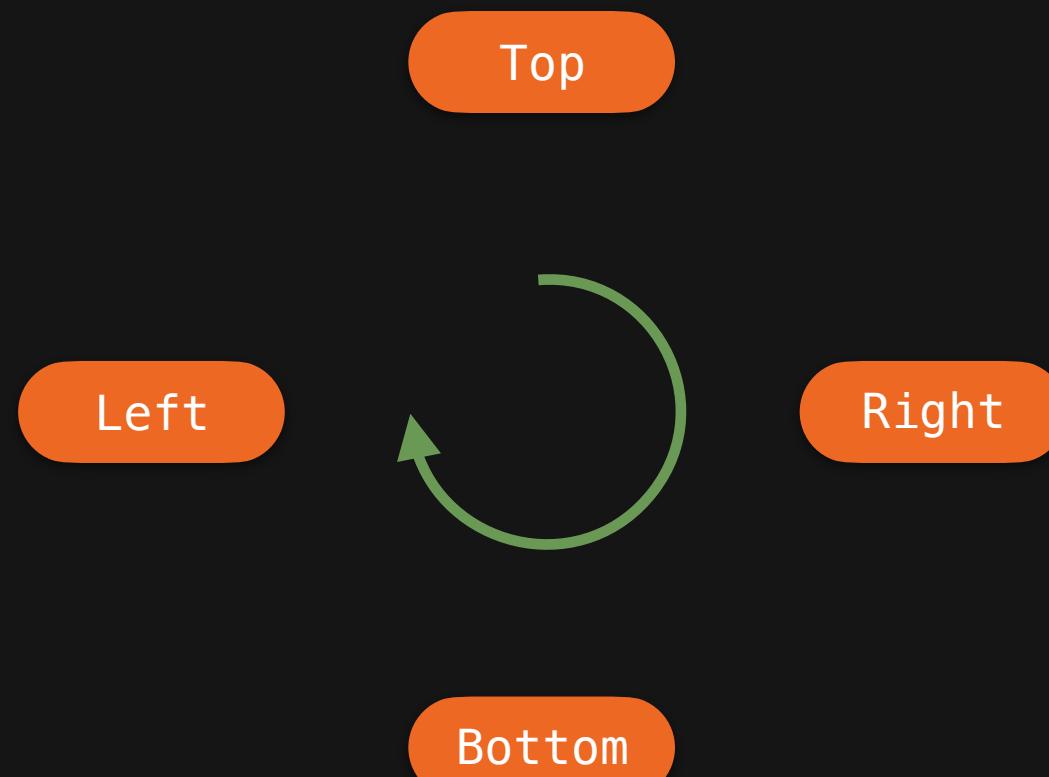
```
padding: 25px 50px;
```

T/B R/L

Se la proprietà padding ha un valore:

```
padding: 25px;
```

All



## height: e width:

Le proprietà height e width vengono utilizzate per impostare l'altezza e la larghezza di un elemento.

height e width possono essere impostati su:

- **auto** (Valore di default. Il browser calcola l'altezza e la larghezza)
- px, cm, ecc.
- valore percentuale (%) rispetto al blocco contenitore.
- inherit, initial

```
div {  
    height: 100px;  
    width: 500px  
}
```

## max-width:

La proprietà max-width viene utilizzata per impostare la larghezza massima di un elemento.

- px, cm, ecc,
- percentuale (%) del blocco contenitore
- none (impostazione da evitare).

Usando `width` (slide precedente) quando la finestra del browser è più piccola della larghezza dell'elemento (500px) il browser aggiungerà una barra di scorrimento orizzontale alla pagina per consentirci di visualizzarlo tutto.

```
div {  
    height: 100px;  
    max-width: 500px;  
}
```

Usando `max-width`, la dimensione del div sarà *al massimo 500px*; quando la dimensione della finestra è minore il div diminuirà la sua dimensione evitando le barre scorrimento.

Larghezza minima: `min-width: ;`

Altezza massima: `max-height: ;`

Altezza minima: `min-height: ;`

## overflow:

La proprietà *overflow* controlla cosa succede ai contenuti che sono troppo grandi rispetto al contenitore.

In questo caso consente di specificare se *ritagliare il contenuto* o *aggiungere barre di scorrimento*.

ha i seguenti valori:

- **visible** - Predefinito. L'overflow non è troncato. Il contenuto "esce" dal contenitore
- **hidden** - L'overflow è troncato e il resto del contenuto non sarà visibile
- **scroll** - L'overflow è troncato, ma viene aggiunta una barra di scorrimento per vedere il resto del contenuto
- **auto** - Se l'overflow è troncato, aggiunge una barra di scorrimento per vedere il resto del contenuto

Nota: overflow funziona solo per elementi di blocco con un'altezza e/o larghezza specificata.

## overflow-x e overflow-y

Le proprietà *overflow-x* e *overflow-y* specificano se modificare l'overflow del contenuto solo orizzontalmente o solo verticalmente.

color:

La proprietà `color` viene utilizzata per impostare il colore del testo.

Il colore del testo predefinito per una pagina viene definito nel `<body>`.

```
body {color: #333}
```

Come visto nella sezione HTML, relativamente all'uso dei colori, il colore si può esprimere attraverso:

Nomi di colore

Valore HEX

Valore RGB, RGBA

Valore HSL, HSLA

Altre proprietà CSS utilizzano questi tipi di valori per esprimere il colore:

`border-color` , `background-color` ...

## Backgrounds

`background-color` specifica il colore di sfondo di un elemento.

```
div {background-color: blue}
```

`background-image` specifica un'immagine da utilizzare come sfondo di un elemento.

Per impostazione predefinita, l'immagine viene ripetuta in modo da coprire l'intero elemento.

L'immagine di sfondo per una pagina può essere impostata in questo modo:

```
body {  
    background-image: url('../img/wave-background.jpg');  
    background-repeat: no-repeat  
}
```

## background-repeat

specifica come e se l'immagine si ripete

```
div {background-repeat: no-repeat}  
[repeat, no-repeat, repeat-x, repeat-y, space, round}
```

## background-position

specifica la posizione.

```
div {background-position: right top}  
[left top, left center, left bottom, ... ], xpos ypos, x% y%
```

## background-attachment

specifica se l'immagine scrolla o rimane fissa.

```
div {background-attachment: fixed}
```

## background - shorthand (codice abbreviato)

La proprietà abbreviata per lo sfondo è **background**:

```
body {background: #ffffff url('../img/smile.png') no-repeat center/80%}
```

che imposta le seguenti proprietà in un'unica dichiarazione:

background-color

background-image

background-repeat repeat, no-repeat, repeat-x, repeat-y, space, round

background-attachment scroll [default], fixed, local

background-position [left top, left center, left bottom, ... ], xpos ypos, x% y%

background-size %, px, cover, contain

background-origin padding-box [default], border-box, content-box

background-clip border-box [default], padding-box, content-box

Il valore background-size può essere incluso solo immediatamente dopo background-position, separato dal carattere '/':

center/80%

## Backgrounds multipli

CSS consente di aggiungere più immagini di sfondo per un elemento, tramite la proprietà `background-image`.

Le diverse immagini di sfondo sono separate da virgole e le immagini sono sovrapposte l'una all'altra.

*La prima immagine in elenco è al livello superiore.*

L'esempio seguente ha due immagini di sfondo, la prima immagine è allineata in alto a destra, la seconda immagine è allineata in alto a sinistra):

```
#box {  
  background-image: url('../img/HTML5_logo.svg'), url('../img/emoji-angry-fill.png');  
  background-position: right top, left top;  
  background-repeat: no-repeat, no-repeat  
}
```

È possibile specificare più immagini di sfondo utilizzando le singole proprietà di sfondo (come slide precedente) o utilizzando la proprietà background abbreviata (shorthand). *Solo l'ultimo sfondo può includere un colore di sfondo.*

```
#box {  
  
background:  
  
url('../img/HTML5_logo.svg') right top no-repeat,  
url('../img/emoji-angry-fill.png') left top no-repeat #000;  
}
```

```
<div id="box">  
  
<h1>Lorem Ipsum Dolor</h1>  
  
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod  
tincidunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis  
nibh euismod consequat.</p>  
  
</div>
```

## Font

Le proprietà dei font definiscono *la famiglia di caratteri, l'enfasi, le dimensioni e lo stile* di un testo.

In CSS, ci sono *due tipi di nomi di famiglia di font*:

**famiglia generica** : un gruppo di famiglie di caratteri (come *Serif, Sans-Serif, Monospace, Cursive* e *Fantasy*)

F

sans-serif

**famiglia di caratteri** - una specifica famiglia di font (come "*Times New Roman*" o *Arial*)

F

serif

sullo schermo del computer, i caratteri sans-serif sono considerati più facili da leggere rispetto ai caratteri serif (con *grazie*).

## font-family:

La famiglia di caratteri di un testo è impostata con la proprietà **font-family**.

Specifica un elenco prioritario di uno o più nomi di famiglie di caratteri e / o nomi di famiglie generici per l'elemento selezionato.

**font-family** di solito contiene diversi nomi di font come sistema di "fallback".

Se il browser non supporta il primo font, prova con il font successivo e così via\*.

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

Nota : se il nome di una famiglia di font è composto da più di una parola, deve essere racchiuso tra virgolette, ad esempio: "Times New Roman".

\* Vedi: [https://www.w3schools.com/cssref/css\\_websafe\\_fonts.asp](https://www.w3schools.com/cssref/css_websafe_fonts.asp)

## Font non standard: @font-face

Specifica un carattere personalizzato con cui visualizzare il testo; consente di non usare i caratteri *websafe*, implica il caricamento della risorsa sul server remoto.

### Formati di font

- TrueType Fonts (TTF)
- OpenType Fonts (OTF)
- Web Open Font (WOFF)
- Web Open Font (WOFF 2.0)
- Font TrueType / OpenType
- SVG
- Embedded OpenType Fonts (EOT)

I font EOT sono una forma compatta di font OpenType progettati da Microsoft per essere utilizzati come caratteri incorporati nelle pagine Web.

**generatore web-font:** <https://www.fontsquirrel.com/tools/webfont-generator>

approfondimento: <https://dev.to/lambdatest/typography-and-cross-browser-compatibility-testing-56j5>

Download versioni multiple: <https://www.axllent.org/code/google-font-downloader/>

I caratteri vengono definiti all'interno della regola CSS `@font-face`.

```
@font-face {  
    font-family: 'Open Sans';  
    src: url('../font/OpenSans-Regular.woff') format('woff'),  
        url('../font/OpenSans-Regular.ttf') format('truetype')  
}  
  
body {  
    font-family: 'Open Sans', sans-serif;  
}
```

Bisogna indicare tutte le famiglie di font che verranno usate nel sito: quindi per usare la versione bold del vostro carattere dovete aggiungere la dichiarazione

```
@font-face {  
    font-family: 'Open Sans bold';  
    src: url('../font/OpenSans-Bold.woff') format('woff'),  
        url('../font/OpenSans-Bold.ttf') format('truetype')  
}  
  
b, strong {  
    font-family: 'Open Sans bold'  
}
```

## Font non standard: **Google Fonts**

Se non desideri utilizzare nessuno dei caratteri standard in HTML, puoi utilizzare i caratteri gratuiti di Google:

<https://fonts.googleapis.com/>

Basta aggiungere un collegamento al foglio di stile speciale nella sezione `<head>` e quindi fare riferimento al carattere nel CSS.\*

```
<head>
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link href="https://fonts.googleapis.com/css2?family=Open+Sans:wght@300;400;700&display=swap"
        rel="stylesheet">
</head>
```

```
body {
  font-family: "Open Sans", sans-serif;
}
```

\* Attraverso lo strumento di google potete scegliere più caratteri della stessa famiglia (thin, light, regular, bold, black...) o più font.

font-size:

La proprietà **font-size** imposta la dimensione del carattere.

Il valore della dimensione del carattere può essere una dimensione *assoluta* o *relativa*.

*Dimensione assoluta (px, keywords):*

px xx-small x-small small medium large x-large xx-large xxx-large

*Dimensione relativa (em, %, keywords):*

Imposta la dimensione relativa all'elemento padre  
em % larger smaller

**Nota:** se non si specifica una dimensione per il carattere, *la dimensione predefinita per il testo normale*, come i paragrafi, è 16px (16px = 1em).

font-weight:

La proprietà **font-weight** specifica il peso di un font:

```
p.normal { font-weight: normal}  
p.highlights { font-weight: bold}
```

il valore può essere anche espresso con: **100, 200, 300, 400, 500, 600, 700, 800, 900**

font-variant:

La proprietà **font-variant** specifica se un testo deve essere visualizzato o meno in carattere maiuscoletto.

In un carattere maiuscoletto, tutte le lettere minuscole vengono convertite in lettere maiuscole. Tuttavia, le lettere maiuscole convertite vengono visualizzate con caratteri più piccoli rispetto alle lettere maiuscole originali nel testo.

```
p.normal { font-variant: normal}  
p.smallcaps { font-variant: small-caps}
```

font-style:

La proprietà **font-style** specifica lo stile di un font, normale o corsivo:

```
p.normal { font-style: normal}  
p.highlights { font-style: italic}
```

font: - shorthand (codice abbreviato)

La proprietà font è una scorciatoia per le seguenti proprietà:

font-style

font-variant

font-weight

font-size / line-height

font-family

Sono richiesti i valori della *dimensione del carattere* e della *famiglia di caratteri*. Se manca uno degli altri valori, viene utilizzato il loro valore predefinito.

**Nota:** la proprietà line-height imposta lo spazio tra le righe.

```
body {font: italic small-caps bold 100%/1.47 "Helvetica  
Neue", Helvetica, Arial, "Lucida Grande", sans-serif;}
```

## Formattazione del testo

text-align:

La proprietà `text-align` viene utilizzata per impostare l'allineamento orizzontale di un testo.

Un testo può essere allineato a sinistra o a destra, centrato o giustificato.

```
p {text-align: center}
```

```
p {text-align: left}
```

```
p {text-align: right}
```

```
p {text-align: justify}
```

## line-height:

La proprietà `line-height` viene utilizzata per specificare lo spazio tra le righe (altezza della linea):

```
p {line-height: 1.47}
```

## text-indent:

La proprietà `text-indent` viene utilizzata per specificare il rientro della prima riga di un testo:

```
p { text-indent: 50px}
```

## vertical-align:

La proprietà `vertical-align` viene utilizzata per impostare l'allineamento verticale di un elemento.

Imposta l'allineamento verticale di un box `inline` , `inline-block` o `table-cell` .

```
span {vertical-align: baseline}  
sub, super, text-top, text-bottom, middle, top, bottom
```

La proprietà `vertical-align` può essere utilizzata in due contesti:

Per **allineare verticalmente il riquadro di un elemento in linea** all'interno del riquadro della riga che lo contiene. Ad esempio, potrebbe essere utilizzato per posizionare verticalmente un'immagine in una riga di testo (effetti più evidenti con un valore di `line-height` alto).

Per allineare verticalmente **il contenuto di una cella in una tabella** .

text-decoration:

`text-decoration` viene utilizzato per impostare o rimuovere particolari “decorazioni” dal testo.

Il valore `text-decoration: none;` viene spesso utilizzato per rimuovere la sottolineatura di default dai collegamenti:

```
a {text-decoration: none}  
a {text-decoration: overline}  
a {text-decoration: line-through}  
a {text-decoration: underline}
```

Nota: non è consigliabile sottolineare il testo che non è un collegamento, poiché questo spesso confonde il lettore.

## text-transform:

**text-transform** viene utilizzato per specificare lettere maiuscole e minuscole in un testo.

Può essere utilizzato per trasformare tutto in lettere maiuscole o minuscole o in maiuscolo la prima lettera di ogni parola:

```
span {text-transform: uppercase}  
span {text-transform: lowercase}  
span {text-transform: capitalize}
```

## letter-spacing:

La proprietà `letter-spacing` viene utilizzata per specificare lo spazio tra i caratteri in un testo.

```
h1 {letter-spacing: 3px}  
h2 {letter-spacing: -3px}
```

## word-spacing:

La proprietà `word-spacing` viene utilizzata per specificare lo spazio tra le parole in un testo:

```
h1 {word-spacing: 3px}  
h2 {word-spacing: -3px}
```

## liste

La proprietà `list-style-type` definisce lo stile del marcatore dell'elemento della lista:

imposta un simbolo diverso dal default per gli elementi della lista, oppure può eliminarlo:

```
ul.list {list-style-type: square}
```

```
ul.list {list-style-type: none}
```

```
ul.list {list-style-type: square}
```

```
li.list-item {list-style-type: circle}
```

La proprietà `list-style-image` definisce una immagine come marcatore dell'elemento della lista:

```
ul.list {list-style-image: url('arrow_down.svg')}
```

La proprietà `list-style-position` specifica la posizione degli indicatori degli elementi dell'elenco.

```
ul.list {list-style-position: outside}  
ul.list {list-style-position: inside}
```

`list-style` è la proprietà shorthand: `type` , `image` , `position`

```
ul.list {list-style: square inside url('arrow_down.svg')}
```

La proprietà `list-style` viene specificata con uno, due o tre valori in qualsiasi ordine.

Se `list-style-type` e `list-style-image` sono entrambi impostati, `list-style-type` viene utilizzato come *fallback* se l'immagine non è disponibile.

## Formattazione liste con pseudo-elementi

::before, creo uno pseudo elemento prima del contenuto del <li>

```
ul.list li::before {  
    content: "";  
    width: 12px;  
    height: 12px;  
    background-color: red;  
    border-radius: 50%;  
    display: inline-block;  
    margin-right: 8px;  
}
```

## Formattazione Tabelle

La proprietà `border-collapse` imposta se i bordi della tabella devono essere compressi in un unico bordo:

```
table {border-collapse: collapse}
```

Per applicare dimensioni, bordi, spaziature e allineamenti si usano le proprietà CSS:

`width`, `height`, `text-align`, `border`, `vertical-align`,  
`padding`...

## display:

La proprietà display specifica se / come viene visualizzato un elemento.

Ogni elemento HTML ha un valore di visualizzazione predefinito a seconda del tipo di elemento. Il valore di visualizzazione predefinito per la maggior parte degli elementi è **block** o **inline**.

```
p.d-none {display: none}  
p.d-inline {display: inline}  
p.d-block {display: block}  
p.d-inline-block {display: inline-block}
```

**display: none** è comunemente usato con JavaScript per nascondere e mostrare elementi senza eliminarli e ricrearli.

Elenco completo di tutti i valori che può assumere la proprietà display, approfondimento:

[https://www.w3schools.com/cssref/pr\\_class\\_display.asp](https://www.w3schools.com/cssref/pr_class_display.asp)

<https://developer.mozilla.org/en-US/docs/Web/CSS/display>

display:

modificare il valore di visualizzazione predefinito

La modifica di un elemento in linea in un elemento del blocco o viceversa può essere utile per rendere la pagina un aspetto specifico e seguire comunque gli standard Web.

Un esempio è la disposizione di elementi `<li>` in linea per il menu orizzontali:

```
li {display: inline-block}
```

Nota: l' impostazione della proprietà di visualizzazione di un elemento modifica solo la modalità di visualizzazione dell'elemento, NON il tipo di elemento.

Quindi, un elemento in linea dichiarato con `display: block;` non può contenere altri elementi di blocco al suo interno.

Il tag `<a>` non può contenere elementi `<li>` anche se dichiarato con: `display: block`

## display: inline-block

La differenza principale rispetto a `display: block` è che `display: inline-block` non aggiunge un'interruzione di riga dopo l'elemento, quindi l'elemento può posizionarsi accanto ad altri elementi.

`display: inline-block` consente di impostare una larghezza e un'altezza sull'elemento, margini superiore e inferiore e il padding superiore e inferiore sono rispettati

`display: inline` non si possono impostare i margini superiori e inferiori, né il padding superiore e inferiore, né l'altezza né la larghezza.

## Pseudo-classi

I collegamenti possono essere visualizzati in diversi modi:

```
/* unvisited link */  
a:link {color: #FF0000}  
  
/* visited link */  
a:visited {color: #00FF00}  
  
/* mouse over link */  
a:hover {color: #FF00FF}  
  
/* selected link */  
a:active {color: #0000FF}
```

Nella definizione CSS `a:hover` DEVE venire dopo `a:link` e `a:visited` per essere efficace! `a:active` DEVE venire dopo `a:hover`.

`a {color: #FF00FF}`  
sovrascrive i comportamenti di `a:link` e `a:visited`

I nomi delle pseudo-classi sono *case-insensitive*.

Attraverso le pseudo-classi si possono personalizzare i diversi comportamenti dei link e molto altro... [nth = ennesimo elemento]

```
:first-child  
:last-child  
:nth-child(n)  
:nth-of-type(n)  
:nth-last-child(n)  
:nth-last-of-type(n)  
:not(selector)  
:focus  
:checked  
:empty
```

*elenco completo: <https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes>*

*altri esempi: <https://developer.mozilla.org/en-US/docs/Web/CSS/:3nth-last-child>*

## Pseudo-elementi

Gli pseudo-elementi sono elementi generati "al volo" tramite il css; la proprietà "content" può essere definita con testo, immagini o codice dei simboli\*:

```
::after; ::before; ::selection; ::first-line; ::first-letter
```

```
h1::after{content:"\2605"; }
```

```
h1::before{content:"Sez."; color: red; font-size: 0.8rem}
```

```
h1::after{content:url(../img/emoji–angry–fill.svg)}
```

\* <https://www.toptal.com/designers/htmlarrows/symbols/black-star/>

elenco completo: <https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements>

## position:

La proprietà *position* specifica il tipo di metodo di posizionamento utilizzato per un elemento.

Gli elementi vengono quindi posizionati usando le proprietà **top**, **bottom**, **left** e **right**.

- **static**: (default)
- **relative**: l'elemento viene posizionato in base al flusso normale del documento e quindi sfalsato rispetto a se stesso in base ai valori di **top**, **right**, **bottom** e **left**.
- **absolute**: l'elemento viene rimosso dal normale flusso del documento e non viene creato spazio per l'elemento nel layout di pagina. È posizionato rispetto al suo antenato posizionato più vicino , se presente; in caso contrario, viene posizionato rispetto al blocco contenitore iniziale . La sua posizione finale è determinata dai valori di **top**, **right**, **bottom** e **left**.
- **fixed**: l'elemento viene rimosso dal normale flusso del documento e non viene creato spazio per l'elemento nel layout di pagina. È posizionato rispetto al blocco contenitore iniziale stabilito dal viewport.
- **sticky**: Il posizionamento appiccicoso è un ibrido di posizionamento *relative* e *fixed*.  
L'elemento viene trattato come posizionato relativo fino a quando non attraversa una soglia specificata, a quel punto viene trattato come posizionato fisso.

```
#box1 {  
    position: relative;  
    width: 100%;  
    background: url('../img/wave-background.jpg') no-repeat;  
    height: 400px;  
    background-size: cover  
}  
#box1 .caption {  
    position: absolute;  
    background: violet;  
    padding: 2rem 1rem;  
    bottom: 1rem; left: 1em  
}  
h1, h2 { margin: 0 }
```

```
<div id="box1">  
    <div class="caption">  
        <h1>Lorem Ipsum Dolor</h1>  
        <h2>Ut enim ad minim veniam</h2>  
    </div>  
</div>
```

## z-index: elementi sovrapposti

Quando gli elementi sono posizionati, possono sovrapporsi ad altri elementi.

La proprietà `z-index` specifica l'ordine di stack di un elemento (quale elemento deve essere posizionato sopra, o sotto, gli altri).

Un elemento può avere un ordine positivo o negativo:

```
#box2 {  
  position: absolute;  
  left: 20px;  
  top: 20px;  
  z-index: 1  
}
```

Poiché `#box2` ha uno `z-index` di `1`, verrà posizionato davanti agli altri box (*esempio completo slide successiva*).

Un elemento con un maggiore ordine di stack è sempre davanti a un elemento con un ordine di stack inferiore.

Nota: se due elementi posizionati si sovrappongono senza uno `z-index` specificato, l'elemento posizionato per ultimo nel codice HTML verrà mostrato in alto.

```
.container {  
    width: 1200px;  
    margin: 0 auto;  
    position: relative;  
    height: 300px  
}  
.box {  
    width: 300px;  
    height: 200px;  
    margin: 1rem;  
    padding: 1rem;  
    display: inline-block;  
    background-color: orange;  
    position: absolute;  
    top: 0  
}
```

```
.box p {  
    text-align: center;  
    font-size: 3rem;  
}  
#box1{  
    background-color: orange  
}  
#box2{  
    background-color: green;  
    top: 20px; left: 20px;  
    z-index: 1;  
}  
#box3{  
    background-color: blue;  
    top: 40px; left: 40px  
}
```

```
<div class="container">  
    <div id="box1" class="box"><p>1</p></div>  
    <div id="box2" class="box"><p>2</p></div>  
    <div id="box3" class="box"><p>3</p></div>  
</div>
```

## Position: sticky

Il posizionamento appiccicoso è un ibrido di posizionamento relativo (*relative*) e fisso (*fixed*).

L'elemento viene trattato come posizionato relativo fino a quando non attraversa una soglia specificata, a quel punto viene trattato come posizionato fisso.

```
.sticky {  
  position: sticky;  
  top: 0px;  
}
```

Per l'esempio vedi slide successiva.

Nota: chiarimento sul comportamento della posizione sticky.

```
.container {  
    width: 400px;  
    margin: 0 auto;  
    height: 250px;  
    overflow-y: scroll  
}  
  
.box {  
    width: 60px;  
    height: 60px;  
    margin: 1rem;  
    background-color: orange;  
    display: inline-block;  
}  
  
.sticky {  
    position: sticky;  
    top: 0;  
}
```

```
<div class="container">  
    <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit.  
        Deleniti sunt odit ad impedit voluptatum vero doloremque aliquam  
        dolor, incidunt alias voluptates eveniet,  
        totam blanditiis fugit, harum rem expedita deserunt a?</p>  
    <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit.  
        Deleniti sunt odit ad impedit voluptatum vero doloremque aliquam  
        dolor, incidunt alias voluptates eveniet,  
        totam blanditiis fugit, harum rem expedita deserunt a?</p>  
    <div class="box">  
        <p>1</p>  
    </div>  
    <div class="box sticky">  
        <p>2</p>  
    </div>  
    <div class="box">  
        <p>3</p>  
    </div>  
    <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit.  
        Deleniti sunt odit ad impedit voluptatum vero doloremque aliquam  
        dolor, incidunt alias voluptates eveniet,  
        totam blanditiis fugit, harum rem expedita deserunt a?</p>  
    <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit.  
        Deleniti sunt odit ad impedit voluptatum vero doloremque aliquam  
        dolor, incidunt alias voluptates eveniet,  
        totam blanditiis fugit, harum rem expedita deserunt a?</p>  
    <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit.  
        Deleniti sunt odit ad impedit voluptatum vero doloremque aliquam  
        dolor, incidunt alias voluptates eveniet,  
        totam blanditiis fugit, harum rem expedita deserunt a?</p>  
</div>
```

## float

La proprietà float specifica come deve posizionarsi un elemento rispetto al suo contenitore.

L'elemento viene rimosso dal normale flusso della pagina, sebbene rimanga comunque una parte del flusso.

*float* viene anche utilizzato per il posizionamento e il layout nelle pagine Web (tecnica sostituita dal modello *flexbox* e più recentemente dal modello *grid*).

Può avere uno dei seguenti valori:

**left** - L'elemento fluttua a sinistra del suo contenitore

**right**: l'elemento fluttua a destra del suo contenitore

**none** - valore predefinito

**inherit**: l'elemento eredita il valore float del suo genitore

Nel suo uso più semplice, la proprietà float può essere utilizzata per far scorrere il testo intorno alle immagini.

```
.container{  
    background-color: #eee;  
    padding: 1em;  
    width: 400px  
}  
.float-left-img{float: left; width: 60px; padding-right: .75rem}
```

```
<div class="container">  
  
    <p class="float-left-txt">  
          
        Lorem ipsum dolor sit amet consectetur, adipisicing elit. Exercitationem  
        est mollitia repellat at nemo facere excepturi quasi dolor, hic nam eligendi,  
        eveniet voluptatum ullam harum quas ipsam consequatur, quae architecto.  
    </p>  
</div>
```

```
.box {  
    width: 300px;  
    padding: 1rem 2rem;  
    margin: 1rem;  
    background-color: orange  
}
```

```
<div class="container">  
  
    <div class="box" style="float: left">1</div>  
  
    <div class="box" style="float: left">2</div>  
  
    <div class="box" style="float: left">3</div>  
  
</div>
```

```
.box {  
    width: 300px;  
    padding: 1rem 2rem;  
    margin: 1rem;  
    background-color: orange  
}
```

```
<div class="container">  
  
    <div class="box" style="float: left">1</div>  
  
    <div class="box" style="float: right">2</div>  
  
    <div class="box" style="float: none">3</div>  
  
</div>
```

## clear

La proprietà `clear` specifica quali elementi possono fluttuare accanto all'elemento su cui applichiamo il `clear` e su quale lato.

Può avere uno dei seguenti valori:

`none` - valore predefinito

`left` - non sono ammessi elementi mobili sul lato sinistro

`right`: non sono ammessi elementi mobili sul lato destro

`both` - non sono ammessi elementi mobili sul lato sinistro o destro

`inherit`: l'elemento eredita il valore `clear` del suo genitore

Il modo più comune di utilizzare la proprietà `clear` è dopo aver utilizzato la proprietà `float` su un elemento.

Se un elemento ha il `float` a sinistra, per eliminare l'effetto del `float` bisogna utilizzare il `clear: left;` (sinistra).

L'elemento con il `float` si sposterà, gli altri elementi si posizioneranno sotto di esso seguendo il normale flusso del documento.

```
.box {  
    width: 300px;  
    padding: 1rem 2rem;  
    margin: 1rem;  
    background-color: orange  
}
```

```
<div class="container">  
  
    <div class="box" style="float: left">1</div>  
  
    <div class="box" style="float: right">2</div>  
  
    <div class="box" style="clear: right">3</div>  
  
</div>
```

```
.container{background-color: #eee; padding:1em; width:1200px; margin: 0 auto}
.box {
    width:300px;
    padding:1rem 2rem;
    margin:1rem;
    background-color:orange
}

.footer{
    clear: both;
}
```

```
<div class="container">

    <div class="box" style="float: right">1. Lorem ipsum... </div>
    <div class="box" style="float: left">2. Lorem ipsum... </div>

    <div class="footer">
        Lorem ipsum...
    </div>
</div>
```

## Il clearfix

Per eliminare i vari problemi generati dal float nella creazione del layout è stato scritto il codice seguente da applicare al contenitore degli oggetti a cui applichiamo il float:

```
.clearfix::after {  
    content: "";  
    clear: both;  
    display: block;  
}
```

riferimento alla soluzione del clearfix:

<https://css-tricks.com/snippets/css/clear-fix/>

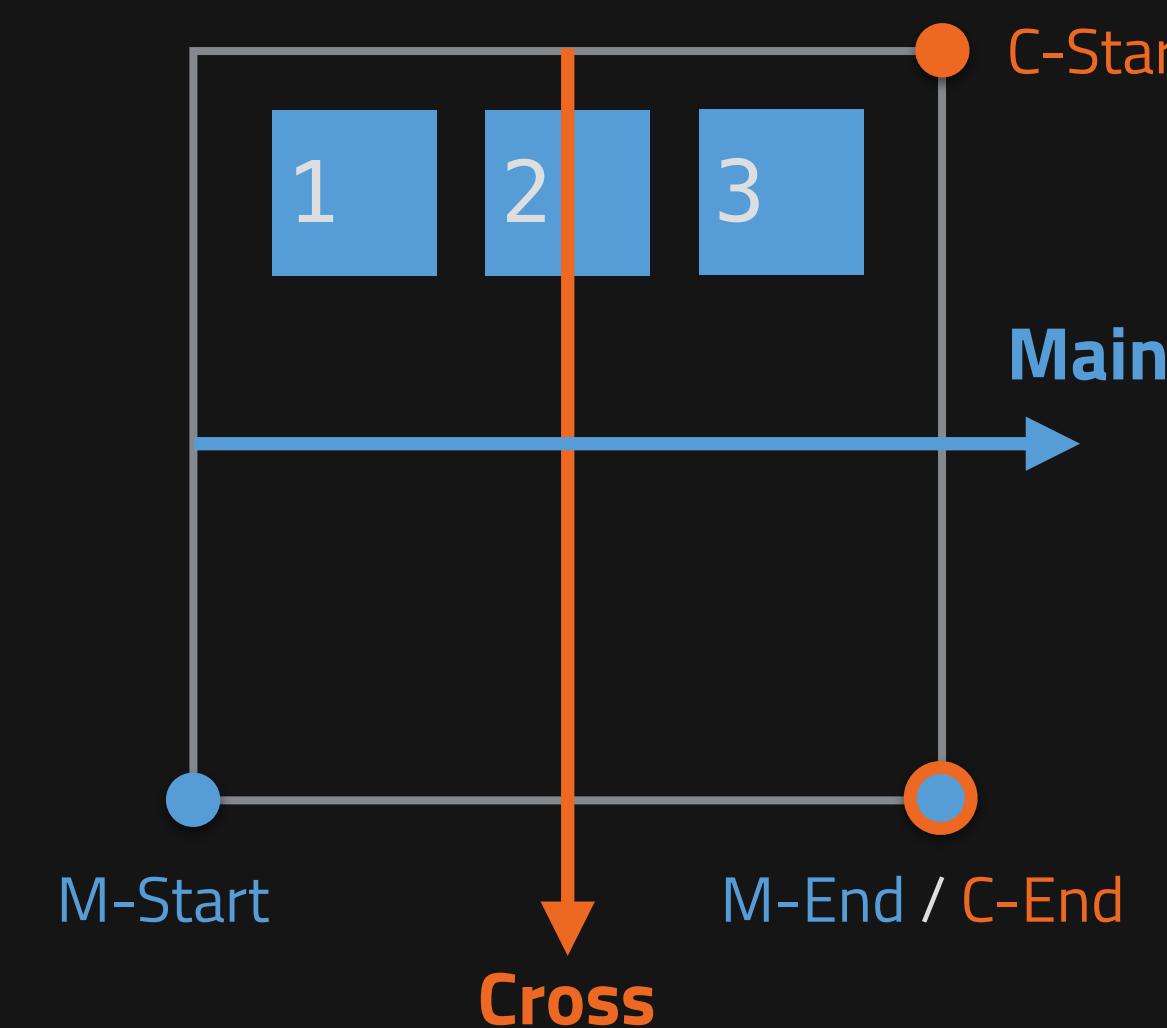
display: flex;

## Proprietà del contenitore flex

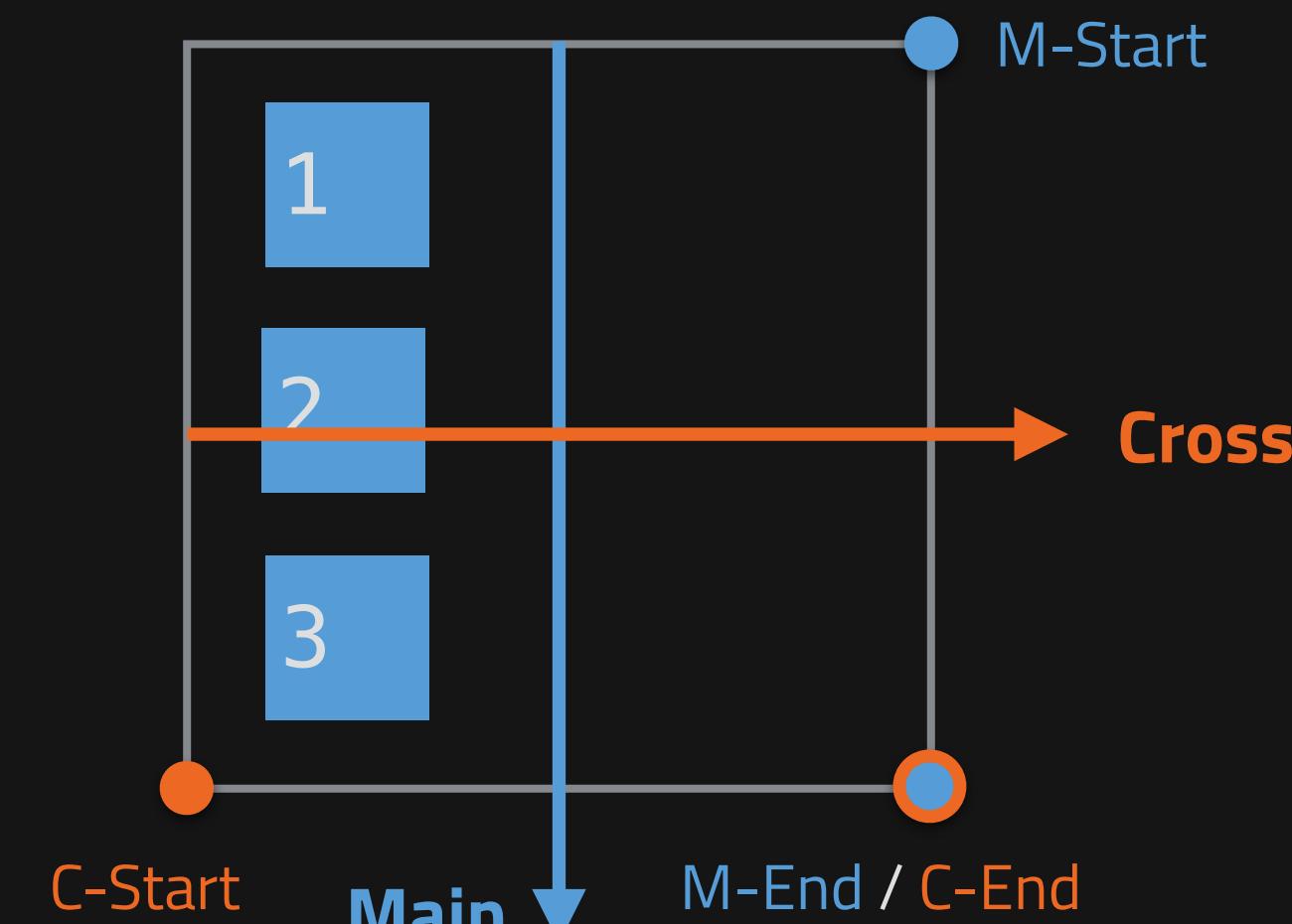
- flex-direction
  - flex-wrap
  - justify-content
  - align-items
  - align-content
- flex-flow [shorthand: flex-direction flex-wrap]*



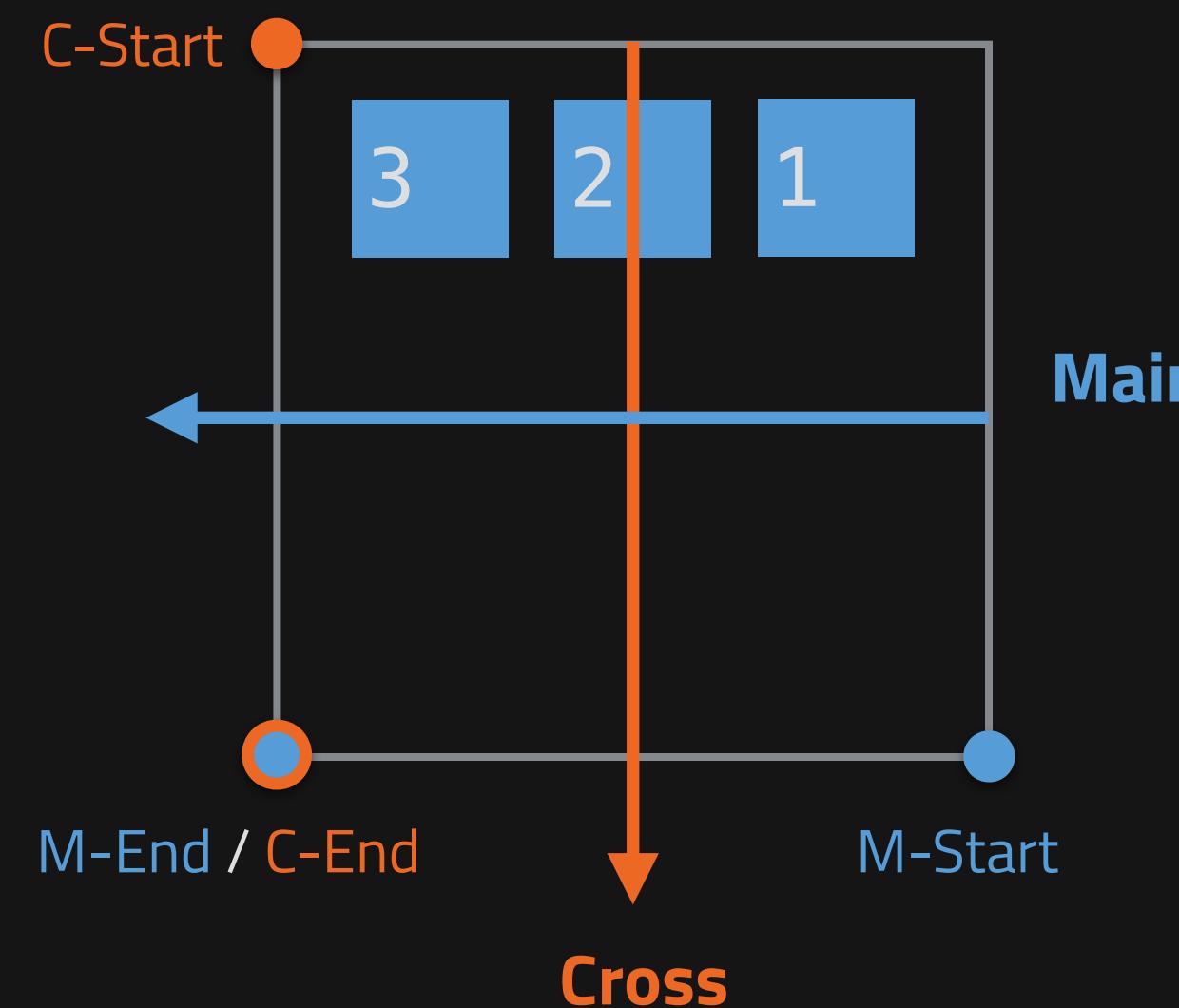
flex-direction: row



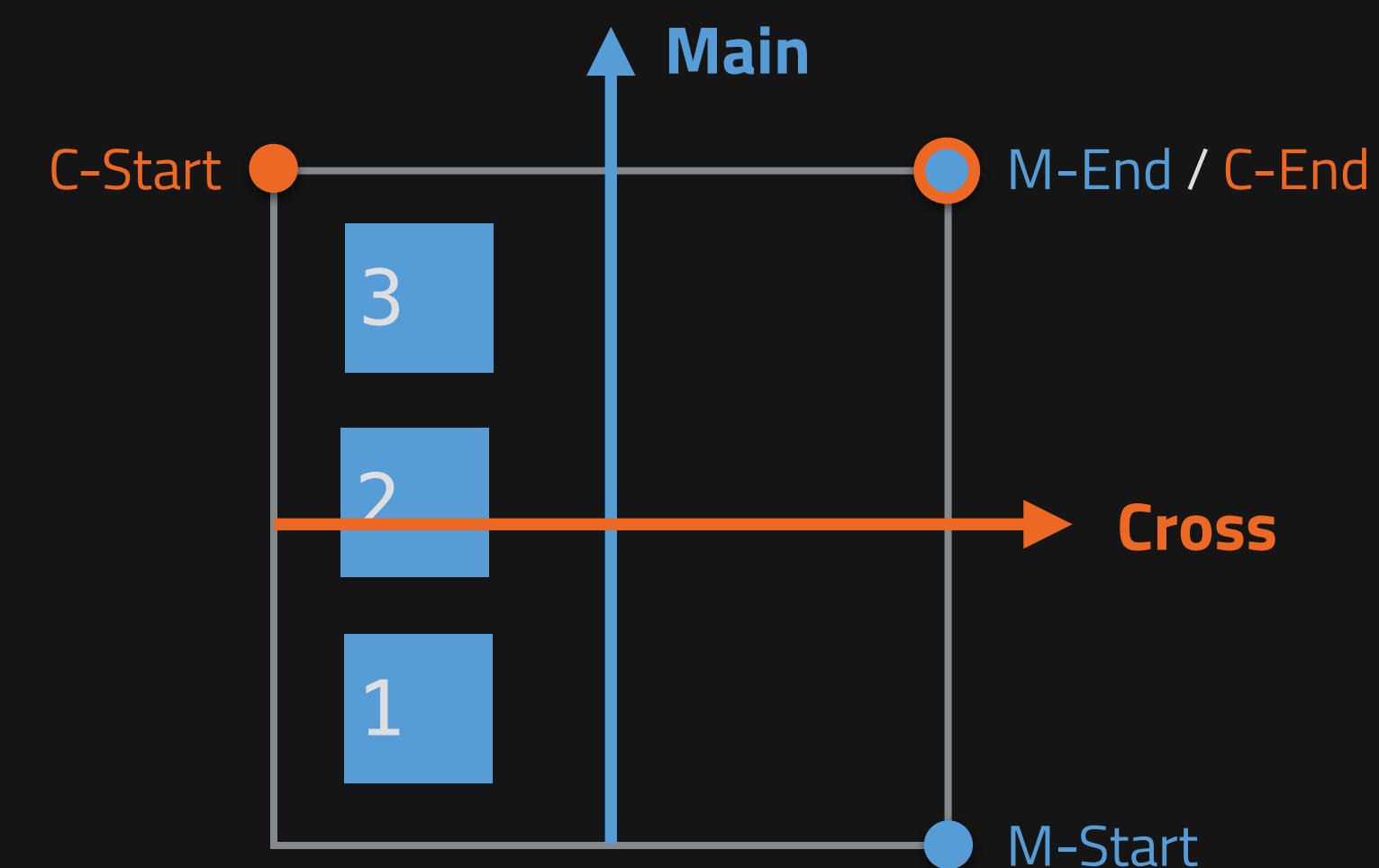
flex-direction: column



flex-direction: row-reverse



flex-direction: column-reverse



## flex-direction:

La proprietà `flex-direction` stabilisce l'asse principale, definendo in questo modo la direzione degli elementi flessibili posizionati nel contenitore `flex`.

Flexbox è un concetto di layout a direzione singola. Pensa agli elementi flessibili come principalmente disposti in file orizzontali o verticali.

Il valore `row` (default) dispone gli elementi flessibili orizzontalmente (da sinistra a destra uno di fianco all'altro):

```
.flex-container {  
    display: flex;  
    flex-direction: row  
}
```

Il valore `row-reverse` dispone gli elementi flessibili orizzontalmente (da destra a sinistra):

```
.flex-container {  
    display: flex;  
    flex-direction: row-reverse  
}
```

Il valore `column` impila gli elementi flessibili verticalmente (dall'alto verso il basso):

```
.flex-container {  
  display: flex;  
  flex-direction: column  
}
```

Il valore `column-reverse` inverte l'ordine (dal basso verso l'alto):

```
.flex-container {  
  display: flex;  
  flex-direction: column-reverse  
}
```

## flex-wrap:

La proprietà flex-wrap specifica se gli elementi flessibili debbano essere “contenuti” dal contenitore e quindi se possono scorrere verso il basso o meno.

Il valore **wrap** specifica che gli elementi flessibili si muoveranno verso il basso se necessario (se non ci stanno più in larghezza):

```
.flex-container {  
    display: flex;  
    flex-wrap: wrap  
}
```

Il valore **nowrap** specifica che gli elementi flessibili non si muoveranno verso il basso (questo è il valore predefinito):

```
.flex-container {  
    display: flex;  
    flex-wrap: nowrap  
}
```

nota: per mantenere la dimensioni width dei div il flex-wrap deve essere impostato su wrap;

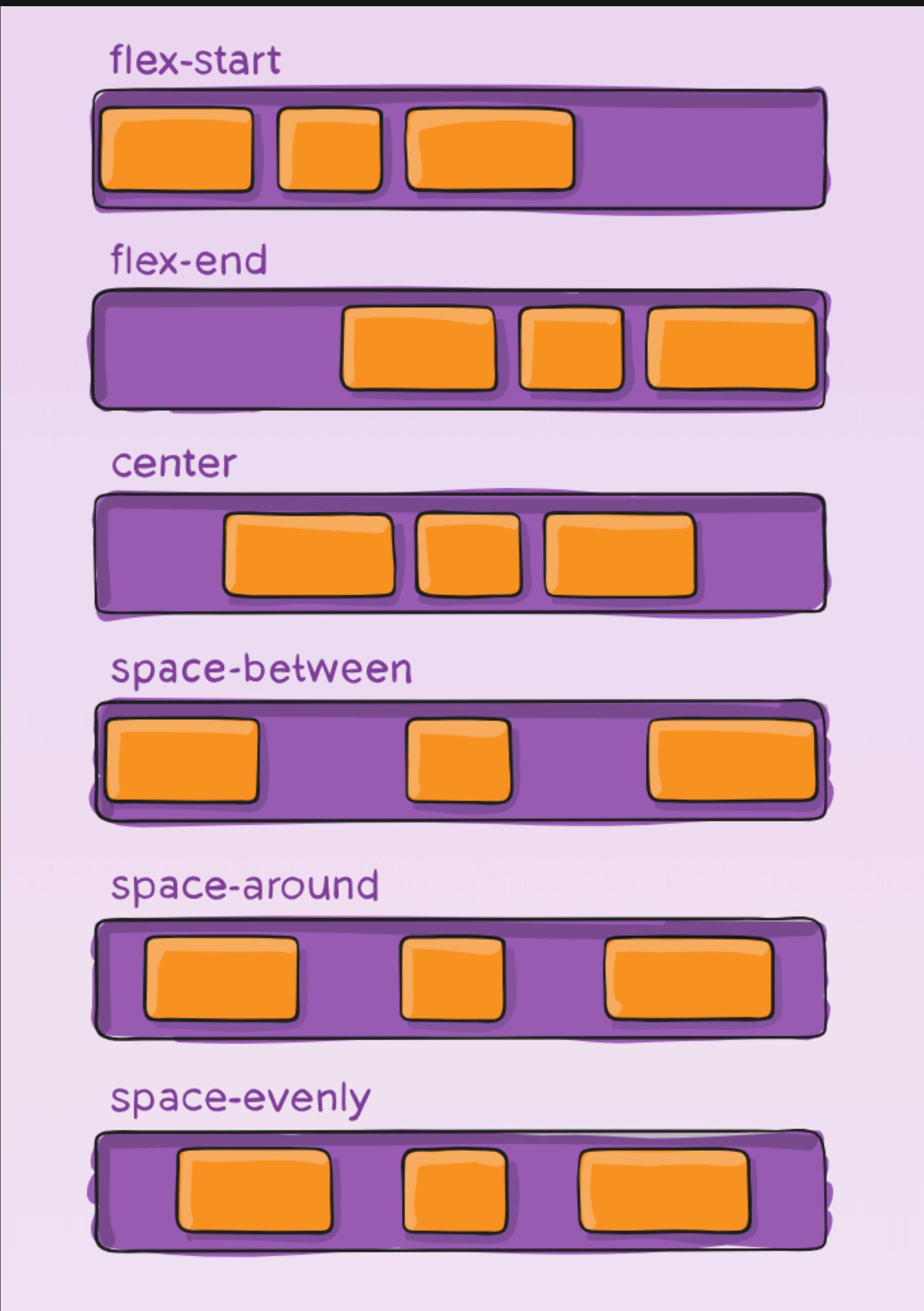
Il valore `wrap-reverse` specifica che gli elementi flessibili si muoveranno verso il basso se necessario, nell'ordine inverso:

```
.flex-container {  
    display: flex;  
    flex-wrap: wrap-reverse  
}
```

La proprietà `flex-flow` è una abbreviazione (shorthand) per impostare `flex-direction` e `flex-wrap`.

```
.flex-container {  
    display: flex;  
    flex-flow: row-reverse wrap  
}
```

# justify-content



## justify-content:

La proprietà justify-content viene utilizzata per allineare **orizzontalmente** gli elementi flessibili, definisce l'allineamento lungo l'asse principale orizzontale.

Aiuta a distribuire lo spazio libero extra rimasto quando tutti gli elementi flessibili su una linea sono rigidi o flessibili, ma hanno raggiunto la loro dimensione massima:

center allinea gli elementi flessibili al centro del contenitore:

```
.flex-container {display: flex; justify-content: center}
```

flex-start allinea gli elementi flessibili all'inizio del contenitore (default):

```
.flex-container {display: flex; justify-content: flex-start}
```

flex-end allinea gli elementi flessibili alla fine del contenitore:

```
.flex-container {display: flex; justify-content: flex-end}
```

space-between: visualizza gli elementi flessibili con lo spazio tra gli elementi:

```
.flex-container {display: flex; justify-content: space-between}
```

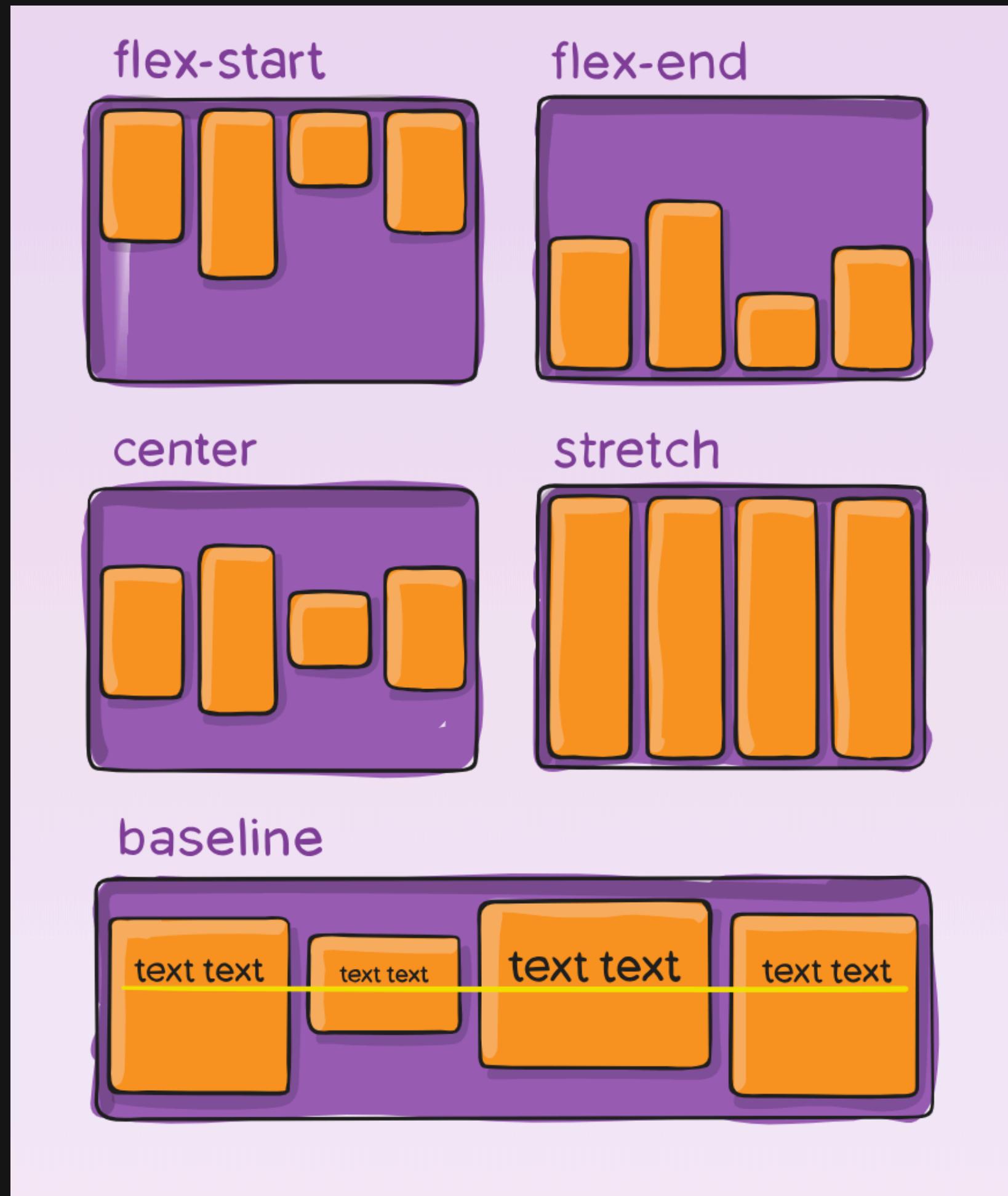
space-around: visualizza gli elementi flessibili con spazio uguale tra gli elementi:

```
.flex-container {display: flex; justify-content: space-around}
```

space-evenly: gli oggetti sono distribuiti in modo che la spaziatura tra due elementi (e lo spazio sui bordi) sia uguale.

```
.flex-container {display: flex; justify-content: space-evenly}
```

# align-items



## align-items:

La proprietà align-items specifica l'allineamento predefinito per gli elementi all'interno del contenitore flessibile.

Definisce il comportamento predefinito per la disposizione degli elementi flessibili lungo l'asse trasversale sulla linea corrente:

center allinea gli elementi flessibili al centro del contenitore:

```
.flex-container {display: flex; height: 900px; align-items:center}
```

flex-start allinea gli elementi flessibili nella parte superiore del contenitore:

```
.flex-container {display:flex; height: 900px; align-items:flex-start}
```

flex-end: allinea gli elementi flessibili nella parte inferiore del contenitore:

```
.flex-container {display: flex; height: 900px;  
    align-items: flex-end}
```

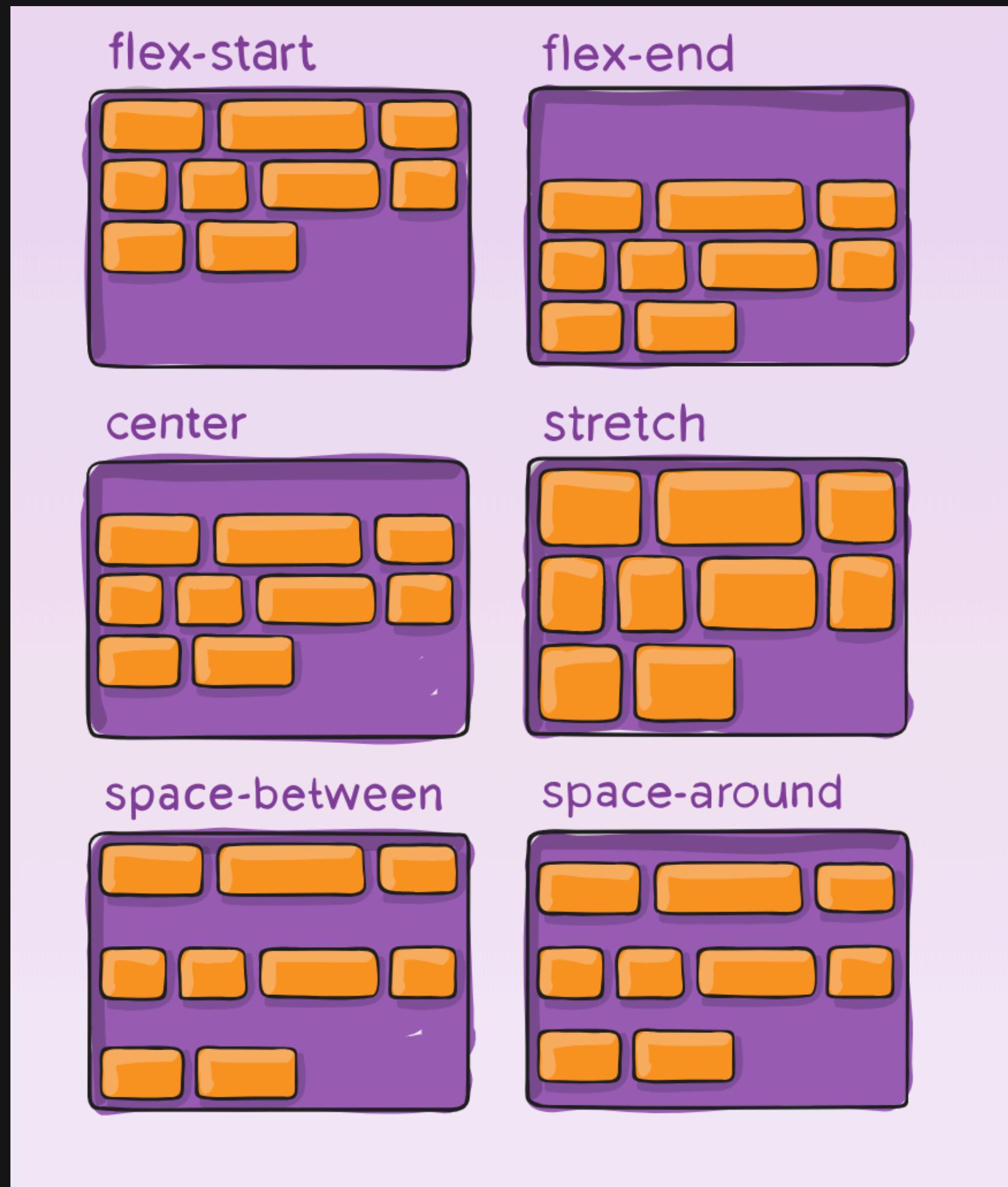
stretch: allunga gli elementi flessibili per riempire il contenitore (default, gli elementi non devono avere altezza definita):

```
.flex-container {display: flex; height:900px;  
    align-items: stretch}
```

baseline: allinea gli elementi flessibili con le linee di base del testo:

```
.flex-container {display: flex; height: 900px;  
    align-items: baseline}
```

# align-content



## align-content:

La proprietà align-content viene utilizzata per allineare le linee degli elementi flessibili.

Questo allinea le linee di un contenitore flessibile all'interno quando c'è spazio extra nell'asse trasversale, in modo simile a come il contenuto di giustificazione allinea i singoli elementi all'interno dell'asse principale.

**Nota:** questa proprietà *non ha effetto* quando c'è *una sola riga di elementi flessibili*.

space-between: items equamente distribuiti; la prima riga è all'inizio del contenitore mentre l'ultima è alla fine:

```
.flex-container {display: flex; height: 900px;  
flex-wrap: wrap; align-content: space-between}
```

space-around: items distribuiti uniformemente con uguale spazio attorno a ciascuna riga:

```
.flex-container {display: flex; height: 900px;  
flex-wrap: wrap; align-content: space-around}
```

space-evenly: gli items sono distribuiti uniformemente con uguale spazio attorno a loro:

```
.flex-container {display: flex; height: 900px;  
flex-wrap: wrap; align-content: space-evenly}
```

stretch: le linee si allungano per occupare lo spazio rimanente:

```
.flex-container {display: flex; height: 900px;  
flex-wrap: wrap; align-content: stretch}
```

center: items centrati nel contenitore:

```
.flex-container {display: flex; height: 900px;  
flex-wrap: wrap; align-content: center}
```

flex-start: items all'inizio del contenitore:

```
.flex-container {display: flex; height: 900px;  
flex-wrap: wrap; align-content: flex-start}
```

flex-end: items alla fine del contenitore:

```
.flex-container {display: flex; height: 900px;  
flex-wrap: wrap; align-content: flex-end}
```

## Proprietà degli elementi flex

I figli di un contenitore flessibile diventano automaticamente elementi flessibili (flex).

Le proprietà definibili per gli elementi flessibili sono:

order: specifica l'ordine degli elementi flessibili.

```
#box1 {order: 1} /* può avere valori negativi */
```

```
#box2 {order: 2}
```

```
#box3 {order: 3}
```

flex-basis: specifica la lunghezza iniziale di un elemento flessibile.

```
.box {flex-basis: 100px} /* puoi usare valori % */
```

**flex-grow:** specifica quanto un oggetto flessibile crescerà rispetto al resto degli elementi flessibili se il contenitore è da riempire.

```
.box {flex-grow: 0} /* default, non si allarga, no valori negativi */
```

```
.box {flex-grow: 1}
```

**flex-shrink:** specifica quanto un elemento flessibile si restringerà rispetto al resto degli elementi flessibili quando non c'è abbastanza spazio sulla riga.

```
.box {flex-shrink: 1} /* default, si stringe, no valori negativi */
```

```
.box {flex-shrink: 0}
```

*flex-shrink* e *flex-grow* vengono utilizzate insieme a *flex-basis*

flex: è una proprietà scorciatoia per:

- flex-grow,
- flex-shrink
- flex-basis.

```
.box {flex: 0 1 auto} /* valori di default */
```

align-self: specifica l'allineamento per l'elemento selezionato all'interno del contenitore flessibile.

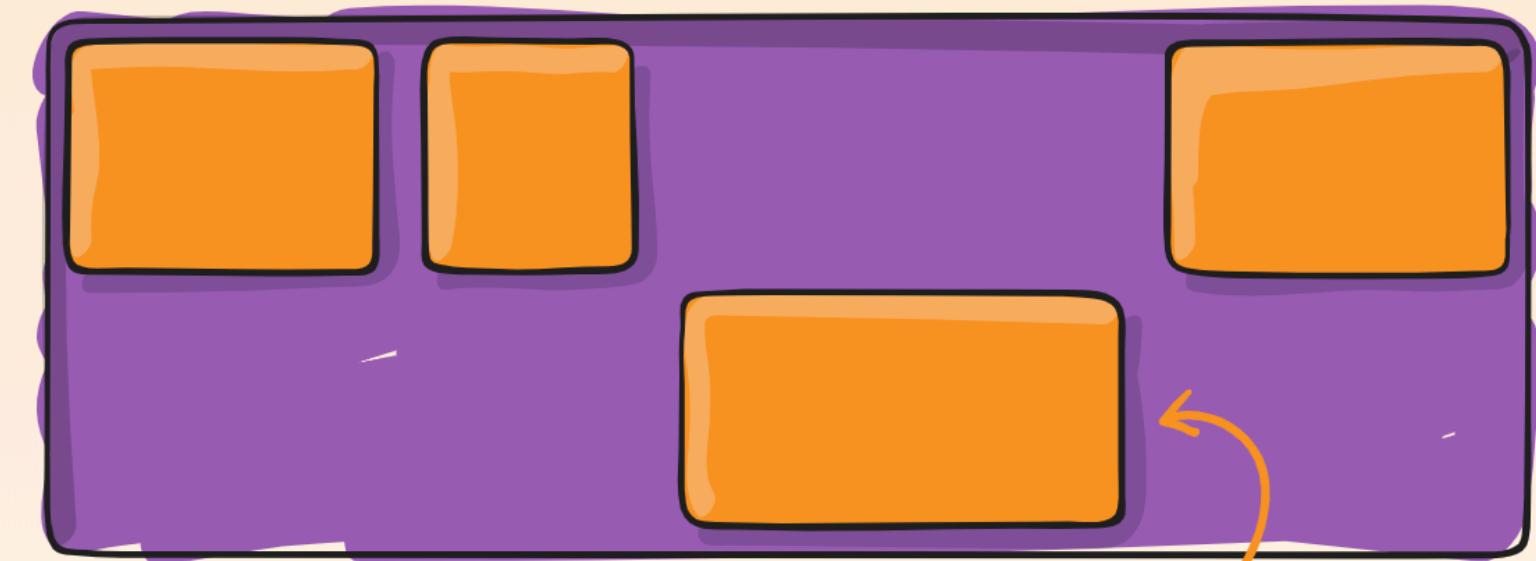
```
#box3 {align-self: flex-end}
```

align-self sostituisce l'allineamento predefinito impostato dalla proprietà **align-items** del contenitore.

nota: *float*, *clear* e *vertical-align* non hanno alcun effetto su un oggetto flessibile.

**align-self**

**flex-start**



**flex-end**

# layout&design

Grid system

## box-sizing: content-box vs border-box

La proprietà `box-sizing`, introdotta con i CSS3, ci consente di includere o meno il padding, il bordo e la larghezza (e altezza) nel calcolo delle dimensioni dell'elemento stesso (vedi `box-model`).

Per impostazione predefinita, la larghezza e l'altezza di un elemento sono calcolate in questo modo:

$width + padding + border = \text{larghezza effettiva di un elemento}$

```
div { width:100px; padding:20px; border:1px solid #333 } <!-- 100 + 20 + 20 + 2 =  
142 -->
```

$height + padding + border = \text{altezza effettiva di un elemento}$

Quando si imposta la larghezza / altezza di un elemento, l'elemento appare spesso più grande di quanto definito (poiché il bordo e il padding dell'elemento vengono aggiunti alla larghezza / altezza specificata dell'elemento).

box-sizing: border-box;

Impostando *box-sizing: border-box*; l'elemento, il riempimento e il bordo sono inclusi in larghezza e altezza.

Dal momento che il risultato dell'uso *box-sizing: border-box*; è più pratico, molti sviluppatori vogliono che tutti gli elementi delle loro pagine funzionino in questo modo.

Il codice sottostante assicura che tutti gli elementi siano dimensionati in questo modo più intuitivo.

```
* , ::after, ::before {box-sizing: border-box}
```

## Costruire una grid

Una grid-view ha spesso 12 colonne (offre più combinazioni di layout<sup>1</sup>) e ha una larghezza totale del 100% e si restringerà e si espanderà man mano che ridimensionate la finestra del browser.

<sup>1</sup> Combinazioni layout

**10 colonne:**

10 è divisibile per:

10, 5, 2, 1 => 4

**12 colonne:**

12 è divisibile per:

12, 6, 4, 3, 2, 1 => 6

**16 colonne:**

16 è divisibile per:

16, 8, 4, 2, 1 => 5

Vogliamo utilizzare una **visualizzazione a griglia con 12 colonne** per avere maggiore controllo sul layout.

Per prima cosa dobbiamo calcolare la percentuale per ogni colonna:  $100\% / 12 = 8,33\%$ .

Quindi creiamo una classe per ciascuna delle 12 colonne "col-" e un numero che definisce il numero di colonne occupate:

```
.col-1 {width: 8.33%}
.col-2 {width: 16.66%}
.col-3 {width: 25%}
.col-4 {width: 33.33%}
.col-5 {width: 41.66%}
.col-6 {width: 50%}

.col-7 {width: 58.33%}
.col-8 {width: 66.66%}
.col-9 {width: 75%}
.col-10 {width: 83.33%}
.col-11 {width: 91.66%}
.col-12 {width: 100%}
```

Impostiamo il box-sizing in border-box per contenere il padding nelle dimensioni delle colonne;

```
* , ::after, ::before {box-sizing: border-box}
```

Ora dobbiamo decidere come affiancare le colonne: usiamo float? O meglio flex box?

## Layout con il modello Flexbox

Il layout Flexbox semplifica la progettazione di una struttura di layout flessibile e reattiva senza utilizzare il float.

### Flexbox

Per iniziare a utilizzare il modello Flexbox, è necessario prima definire il contenitore come flex.

```
.flex-container {display: flex; width: 900px}  
.box {background-color: orange; margin: .6rem; padding: .6rem; font-size: 5rem }
```

Un contenitore flessibile con tre elementi flessibili:

```
<div class="flex-container">  
  <div class="box">1</div>  
  <div class="box">2</div>  
  <div class="box">3</div>  
</div>
```

rif: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

Vogliamo realizzare la **griglia a 12 colonne** con flex:

Come in precedenza creiamo una classe per ciascuna delle 12 colonne class="col-" e un numero che definisce il numero di colonne occupate, impostiamo la dimensione:

```
.col-1 { flex-basis: 8.33% }

.col-2 { flex-basis: 16.66% }

.col-3 { flex-basis: 25% }

.col-4 { flex-basis: 33.33% }

.col-5 { flex-basis: 41.66% }

.col-6 { flex-basis: 50% }
```

```
.col-7 { flex-basis: 58.33% }

.col-8 { flex-basis: 66.66% }

.col-9 { flex-basis: 75% }

.col-10 { flex-basis: 83.33% }

.col-11 { flex-basis: 91.66% }

.col-12 { flex-basis: 100% }
```

Impostiamo tutte le colonne con le proprietà flex necessarie;

```
[class^="col-"] {

  flex-shrink: 0; /* impediamo al box di restringersi */
  flex-grow: 0; /* valore di default, il box non si espande */
  padding-right: 15px;
  padding-left: 15px
}
```

Ogni riga deve essere racchiusa in un <div> a cui applichiamo una la classe "row".

codice HTML:

```
<div class="row">  
  <div class="col-3">...</div> <!-- 25% -->  
  <div class="col-9">...</div> <!-- 75% -->  
</div>
```

Definiamo il contenitore flessibile dichiarando la proprietà display con il valore *flex*:

```
.row {  
  display: flex;  
  flex-wrap: wrap;  
  margin-right: -15px;  
  margin-left: -15px  
}
```

## Flexbox bug

A questo link potete vedere l'elenco completo:

<https://github.com/philipwalton/flexbugs>

## Layout con il modello Grid Layout

Il modulo CSS Grid Layout offre un sistema di layout basato su griglia, con righe e colonne, che semplifica la progettazione di pagine Web senza dover utilizzare float e posizionamento.

Supporto (versione del browser minima).



57.0



16.0



52.0



10



44

## Layout con il modello Grid Layout

Per definire una griglia usiamo il valore `grid` della proprietà `display`.

Come con Flexbox, questo abilita il Grid Layout; tutti i figli diretti del contenitore diventano elementi della griglia.

```
<style>.grid-container {display: grid}</style>

<div class="grid-container">
  <div>Uno</div>
  <div>Due</div>
  <div>Tre</div>
  <div>Quattro</div>
  <div>Cinque</div>
  <div>Sei</div>
  <div>Sette</div>
</div>
```

La dichiarazione `display: grid` crea una griglia di una colonna, quindi gli elementi continueranno a essere visualizzati uno sotto l'altro.

Per vedere qualcosa che assomigli di più alla griglia, dovremo aggiungere alcune colonne con la proprietà: `grid-template-columns`.

Aggiungiamo tre colonne da 300 pixel. Si può utilizzare qualsiasi unità di lunghezza o percentuale.

```
.grid-container {display: grid; grid-template-columns: 300px 300px 300px;}
```

## L' unità *fr*

Oltre a creare griglie utilizzando lunghezze e percentuali, possiamo utilizzare l' unità *fr* per ridimensionare in modo flessibile righe e colonne della griglia.

Questa unità rappresenta *una frazione dello spazio disponibile* nel contenitore della griglia.

```
.grid-container {  
    display: grid;  
    grid-template-columns: 2fr 1fr 1fr;  
}
```

## Spazi tra le righe e colonne

Per creare spazi tra le righe e le colonne usiamo le proprietà `column-gap` per gli spazi tra le colonne `row-gap` per gli spazi tra le righe. `gap` come scorciatoia per `row-gap` e `column-gap`.

```
.grid-container {  
  display: grid;  
  grid-template-columns: 2fr 1fr 1fr;  
  gap: 16px;  
}
```

# Ripetizione della definizione di colonne

Puoi ripetere tutto o solo una sezione del tuo elenco di colonne utilizzando la funzione `repeat()` CSS.

```
.grid-container {  
    display: grid;  
    grid-template-columns: repeat(3, 1fr);  
    gap: 16px;  
}
```

# Puoi dare un'altezza alle righe.

```
.grid-container {  
    display: grid;  
    grid-template-columns: repeat(3, 1fr);  
    gap: 16px;  
    grid-auto-rows: 100px;  
}
```

Approfondimento sul modello grid

[https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS\\_layout/Grids](https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Grids)

# Puoi definire le altezze delle righe.

```
.grid-container {  
    display: grid;  
    grid-template-columns: repeat(3, 1fr);  
    gap: 16px;  
    grid-template-rows: 100px 300px 150px;  
}
```

Approfondimento sul modello grid

[https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS\\_layout/Grids](https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Grids)

## Griglia con il modello grid layout.

Con CSS Grid, non hai bisogno di creare una griglia con classi css: la griglia è già presente nelle specifiche.

```
.container {  
    display: grid;  
    grid-template-columns: repeat(12, minmax(0, 1fr));  
    gap: 20px;  
}
```

La funzione CSS `minmax()` definisce un intervallo di dimensioni maggiore o uguale a min e minore o uguale a max.

Ora possiamo utilizzare il posizionamento basato su linee per posizionare il nostro contenuto sulla griglia a 12 colonne.

La griglia ha sempre delle linee. Queste righe sono numerate, a partire da 1. Si riferiscono alla modalità di scrittura del documento.

La linea 1 della colonna si trova sul lato sinistro della griglia e la linea 1 della riga, in alto.

Possiamo organizzare le cose secondo queste linee specificando la linea di inizio e di fine. Lo facciamo utilizzando le seguenti proprietà:

grid-column-start, grid-column-end,  
grid-row-start, grid-row-end

Queste proprietà possono avere tutte un numero di riga come valore. Puoi anche usare le proprietà abbreviate:

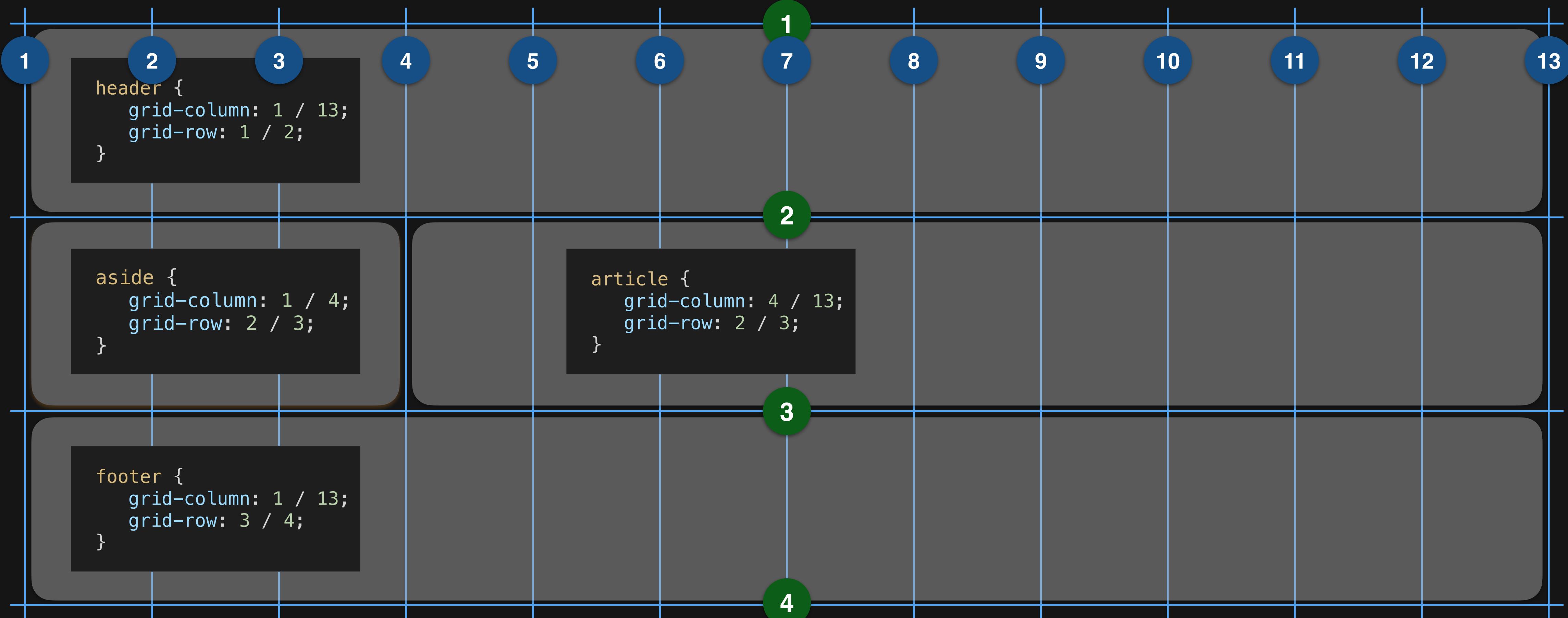
grid-column  
grid-row

Questi ti consentono di specificare le righe di inizio e fine contemporaneamente, separate da una barra /

```
header {  
    grid-column: 1 / 13;  
    grid-row: 1 / 2;  
}  
  
aside {  
    grid-column: 1 / 4;  
    grid-row: 2 / 3;  
}  
  
article {  
    grid-column: 4 / 13;  
    grid-row: 2 / 3;  
}  
  
footer {  
    grid-column: 1 / 13;  
    grid-row: 3 / 4;  
}
```

## Griglia con le indicazioni sulle linee

Griglia 12 colonne e 3 righe : posizionamento delle linee (*vedi esempio codice css slide precedente*)



# CSS

RWD: layout responsive

## RWD: Responsive Web Design

Il responsive Web Design rende la tua pagina web leggibile su tutti i dispositivi.

Il responsive Web Design utilizza solo HTML e CSS.

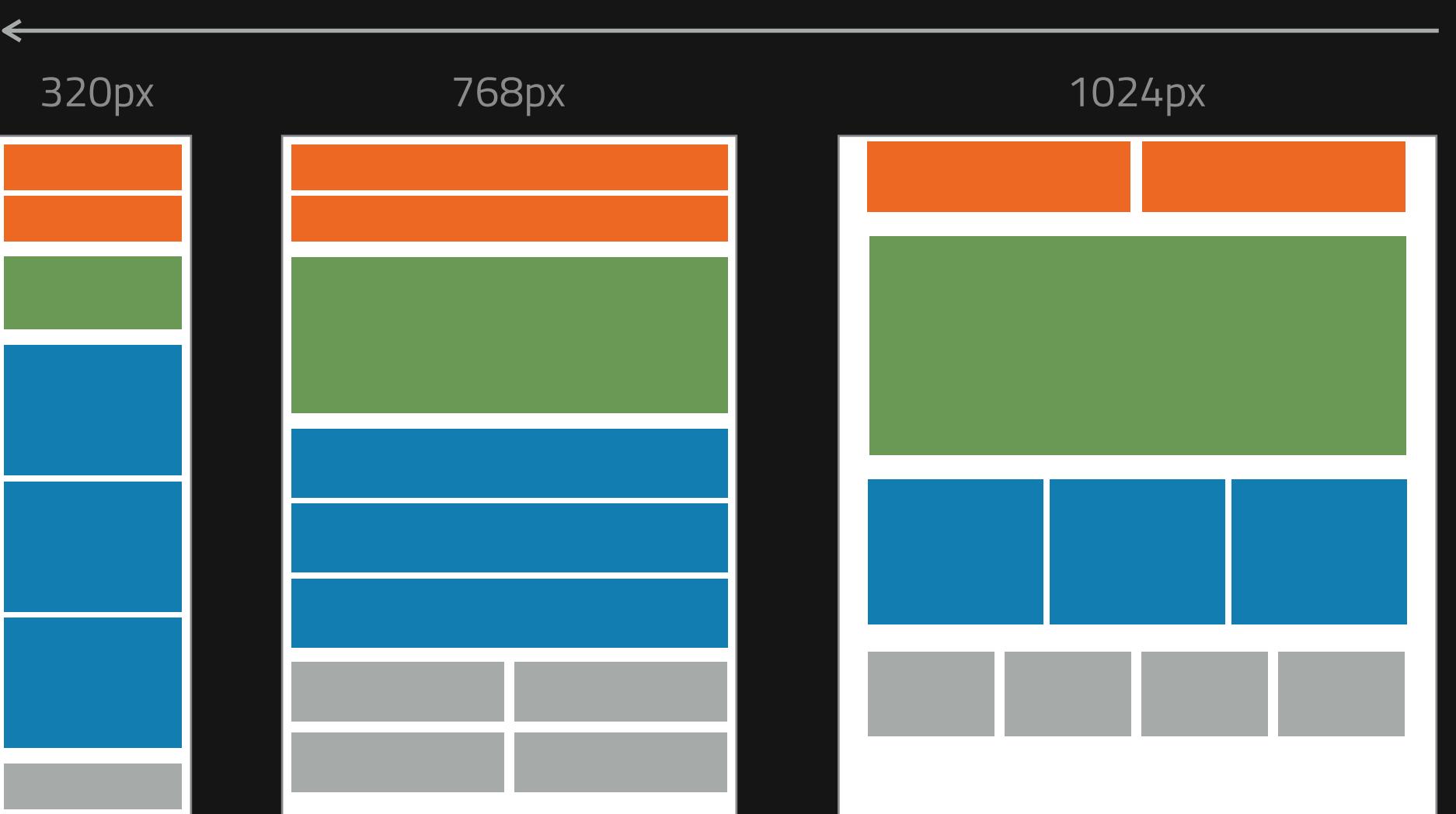
### Progettare per una migliore esperienza per tutti gli utenti

Le pagine Web possono essere visualizzate utilizzando diversi dispositivi: desktop, tablet e telefoni.

La tua pagina web dovrebbe avere un bell'aspetto ed essere facile da usare, indipendentemente dal dispositivo.

Di conseguenza le pagine Web devono adattare il contenuto per qualsiasi dispositivo.

RWD è la tecnica usata per ridimensionare, nascondere, ridurre, ingrandire o spostare il contenuto della pagina web per renderlo usabile su qualsiasi schermo.



## Viewport

Il viewport (area di visualizzazione) varia a seconda del dispositivo e sarà più piccolo su un telefono cellulare che sullo schermo di un computer.

Prima di tablet e telefoni cellulari, le pagine Web erano progettate solo per schermi di computer ed era comune che le pagine Web avessero un design statico e una dimensione fissa.

Le pagine web di dimensioni fisse erano troppo grandi per adattarsi al viewport di tablet e telefoni cellulari.

Per risolvere questo problema, i browser su quei dispositivi, ridimensionavano l'intera pagina Web per adattarsi allo schermo.

## Impostazione del Viewport

HTML5 ha introdotto un metodo per consentire ai web designer di assumere il controllo sulla vista, attraverso il tag <meta>.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

L'elemento meta con name="viewport" fornisce al browser le istruzioni su come controllare le dimensioni e il ridimensionamento della pagina.

content="width=device-width, ... imposta la larghezza della pagina in modo che segua la larghezza dello schermo del dispositivo (che varierà a seconda del dispositivo).

...initial-scale=1.0" imposta il livello di zoom iniziale quando la pagina viene caricata per la prima volta dal browser.

## @media

La regola @media, introdotta con i CSS2, ha reso possibile definire stili diversi per diversi tipi di media: computer, stampanti, dispositivi palmari, tv e così via. In precedenza si poteva creare uno stile ad hoc per media particolari:

```
<link rel="stylesheet" type="text/css" media="print, handheld" href="foo.css">
```

Sfortunatamente questi tipi di media non hanno mai ricevuto molto supporto dai dispositivi, a parte il tipo di supporto per la stampa.

```
@media screen { h1 { background-color: red; } }  
 @media print { h1 { color: red} }
```

CSS3 ha introdotto le media queries\*

Le *media queries* nei CSS3 estendono l'idea dei tipi di media CSS2: invece di cercare un tipo di dispositivo, guardano alla capacità del dispositivo o alla dimensione dello schermo. Le query multimediali possono essere utilizzate per controllare molti parametri, come ad esempio:

- larghezza e altezza del viewport
- orientamento (il tablet / telefono è in modalità orizzontale o verticale?)
- risoluzione
- ...

L'utilizzo di query multimediali è una tecnica diffusa per la distribuzione di un foglio di stile su misura per desktop, portatili, tablet e telefoni cellulari.

\*[https://developer.mozilla.org/en-US/docs/Web/CSS/Media\\_Queries/Using\\_media\\_queries](https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries)

## Sintassi

Una media queries è costituita da un tipo di supporto e può contenere una o più espressioni, che possono essere risolte in true o false.

```
@media not|only mediatype and (expressions) {  
    /* istruzioni CSS */  
}
```

Il risultato della query è true se il tipo di supporto specificato corrisponde al tipo di dispositivo su cui viene visualizzato il documento e tutte le espressioni nella query multimediale sono vere.

Quando una query multimediale è vera, vengono applicate le regole del foglio di stile o dello stile corrispondenti, seguendo le normali regole a cascata.

*not*: inverte il risultato dell'intera query. Se la query restituisce vero col *not* restituirà falso e viceversa. Ad esempio:

```
@media not screen and (color) /* significa 'tutto, escluso schermi a colori' */
```

*only*: impedisce ai browser più vecchi che non supportano le query multimediali di applicare gli stili specificati. Non ha alcun effetto sui browser moderni.

*and*: combina un tipo di supporto con una funzione multimediale o altre funzionalità multimediali.

Il tipo di supporto è facoltativo e il tipo **all** sarà il valore di default. Sono tutti opzionali.

Tuttavia, se si utilizza **not** o **only**, è necessario specificare anche un tipo di supporto.

## breakpoint tipici

Ci sono moltissimi schermi e dispositivi con altezze e larghezze diverse, quindi è difficile creare un breakpoint esatto per ogni dispositivo. Per semplificare le cose, potresti scegliere come target cinque gruppi:

```
/* Extra small devices (phones, 480px and down) */
@media only screen and (max-width: 480px) { ... }

/* Small devices (portrait tablets and large phones, 600px and up) */
@media only screen and (min-width: 600px) { ... }

/* Medium devices (landscape tablets, 768px and up) */
@media only screen and (min-width: 768px) { ... }

/* Large devices (laptops/desktops, 992px and up) */
@media only screen and (min-width: 992px) { ... }

/* Extra large devices (large laptops and desktops, 1200px and up) */
@media only screen and (min-width: 1200px) { ... }
```

Le query multimediali possono anche essere utilizzate per modificare il layout di una pagina in base all'orientamento del dispositivo.

```
@media only screen and (orientation: landscape) { ... }
```

L'esempio seguente cambia il colore di sfondo in funzione della dimensione del display.

Il colore del body è grigio:

```
body {  
    background-color: #dedede /* imposta il colore di sfondo su: grigio */  
}
```

Quando la finestra è minore di 992 pixel il colore del body sarà blu:

```
/* per gli schermi minori di 992px, imposta il colore di sfondo su: blu */  
@media only screen and (max-width: 992px) {  
    body {  
        background-color: rgb(0,0,255)  
    }  
}
```

Quando la finestra è meno di 768px il body sarà rosso

```
/* per gli schermi minori di 768px, imposta il colore di sfondo su: red */  
@media only screen and (max-width: 768px) {  
    body {  
        background-color: rgb(255,0,0)  
    }  
}
```

*nota:* 992px e 768px sono breakpoint "tipici" per i dispositivi.

## MOBILE FIRST

Mobile First significa progettare per dispositivi mobili prima di progettare per desktop o qualsiasi altro dispositivo (questo renderà la visualizzazione della pagina più veloce su dispositivi più piccoli).

*vedi statistiche:*

<https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet>

<https://www.statista.com/statistics/277125/share-of-website-traffic-coming-from-mobile-devices/>

Ciò significa che dobbiamo modificare il nostro CSS.

```
body {  
    background-color: rgb(255,0,0); /* imposta il background: red */  
}  
/* per gli schermi più grandi di 768px, imposta il colore di sfondo su: blu */  
@media only screen and (min-width: 768px) {  
    body {  
        background-color: rgb(0,0,255)  
    }  
}  
/* per gli schermi più grandi di 992px imposta il colore di sfondo su: grigio */  
@media only screen and (min-width: 992px) {  
    body {  
        background-color: #dedede  
    }  
}
```

Approccio Mobile First nella definizione della griglia.

```
/* For mobile phones: */
.col-1, .col-2, .col-3, .col-4, .col-5, .col-6, .col-7, .col-8, .col-9, .col-10, .col-11, .col-12{
    width: 100.00%; padding: 0 15px
}

@media only screen and (min-width: 600px){
    /* For desktop: */
    .col-1 {flex: 0 0 8.33%}
    .col-2 {flex: 0 0 16.66%}
    .col-3 {flex: 0 0 25%}
    .col-4 {flex: 0 0 33.33%}
    .col-5 {flex: 0 0 41.66%}
    .col-6 {flex: 0 0 50%}
    .col-7 {flex: 0 0 58.33%}
    .col-8 {flex: 0 0 66.66%}
    .col-9 {flex: 0 0 75%}
    .col-10 {flex: 0 0 83.33%}
    .col-11 {flex: 0 0 91.66%}
    .col-12 {flex: 0 0 100%}
}
```

## MOBILE FIRST: immagini

width: 100%;

Se la proprietà width è impostata su 100%, l'immagine sarà reattiva e scalerà:

```
img {  
    width: 100%  
}
```

Si noti che nell'esempio sopra, l'immagine può essere ingrandita più della sua dimensione originale.

Una soluzione migliore, in molti casi, sarà quella di utilizzare la proprietà *max-width*.

max-width: 100%;

Se la proprietà max-width è impostata su 100%, l'immagine verrà ridimensionata se necessario, ma mai più grande della sua dimensione originale:

```
img {  
    max-width: 100%; height: auto; /* usiamo valore auto per sovrascrivere l'attributo height */  
}
```

## <picture>

L'elemento `<picture>` contiene un numero di elementi `<source>`, ciascuno riferito a diverse sorgenti di immagini. Così il browser può scegliere l'immagine che meglio si adatta alla vista corrente e / o al dispositivo.

L'attributo media dell'elemento `<source>` definisce gli schermi che visualizzeranno l'immagine.

```
<picture>
  <source media="(max-width:992px)" srcset="assets/img/nomusicnolife800x450.jpg" width="800" height="450">
  <source media="(max-width:768px)" srcset="assets/img/nomusicnolife720x405.jpg" width="720" height="405">
  <source media="(max-width:480px)" srcset="assets/img/nomusicnolife382x215.jpg" width="382" height="215">
  
</picture>
```

È necessario fornire un elemento `<img>`, con `src` e `alt`, subito prima di `</picture>`, altrimenti non verranno visualizzate immagini.

L'elemento `<img>` viene utilizzato per fornire la compatibilità con le versioni precedenti per i browser che non supportano l'elemento `<picture>` o se nessuno dei tag `<source>` corrisponde.

L'attributo `srcset` è obbligatorio e definisce la fonte dell'immagine.

L'attributo `media` è facoltativo e accetta le query multimediali

# NORMALIZE vs RESET CSS

## Reset

L'obiettivo del foglio di stile *reset* è ridurre le incoerenze del browser relativamente a: interlinea, margini, dimensioni dei caratteri, delle intestazioni, e così via.

<https://meyerweb.com/eric/tools/css/reset/>

L'approccio di questo foglio di stile è di "azzerare" gli stili del browser: questo significa che bisogna ridefinire gli stili di tutti gli aspetti tipografici, per esempio.

Non è un approccio vantaggioso perché bisogna riprogettare tutto da capo.

Il foglio di stile *normalize.css* usa un approccio più conservativo (vedi slide successiva).

## Normalize

Gli scopi di normalize.css sono i seguenti:

- *Mantenere le impostazioni predefinite del browser utili anziché cancellarle.*
- Normalizzare gli stili per un'ampia gamma di elementi HTML.
- Correggere bug e inconsistenze comuni del browser.
- Migliorare l'usabilità con sottili miglioramenti.
- Documentare il codice chiaramente usando commenti e documentazione dettagliata.
- Supportare una vasta gamma di browser (compresi i browser per dispositivi mobili) e includere CSS che normalizzano elementi HTML5, tipografia, elenchi, contenuto incorporato, moduli e tabelle.

<https://necolas.github.io/normalize.css/latest/normalize.css>

CDN: <https://cdn.jsdelivr.net/npm/normalize.css@8.0.1/normalize.css>

Bootstrap reboot (reset di bootstrap)

<https://getbootstrap.com/docs/4.1/content/reboot/>

I produttori di browser talvolta aggiungono prefissi a proprietà CSS sperimentali o non standard, in modo che gli sviluppatori possano sperimentare le nuove funzionalità.

I principali browser utilizzano i seguenti prefissi:

**-webkit-** (Chrome, Safari, versioni più recenti di Opera, quasi tutti i browser iOS (incluso Firefox per iOS), in pratica qualsiasi browser basato su WebKit)

**-moz-** (Firefox)

**-o-** (versioni di Opera)

**-ms-** (Internet Explorer e Microsoft Edge)

Nella maggior parte dei casi, per utilizzare una nuova proprietà di stile CSS, si prende la proprietà CSS standard e si aggiunge il prefisso per ciascun browser. Le versioni con prefisso vanno elencate sempre prima (nell'ordine che preferisci) mentre la normale proprietà CSS sarà elencata per ultima\*.

```
{display: -webkit-box}  
{display: -moz-box}  
{display: -ms-flexbox}  
{display: -webkit-flex}  
{display: flex}
```

\* verificare su <http://shouldiprefix.com/>

## Ottimizzazione css

- **Usare gli shorthands** (scorciatoie):

*background, border, font...*

- **Usare proprietà a valori multipli:**

*margin, padding*

- **Ottimizzare i colori:** usare la notazione a tre cifre per i colori: es: #333333 può essere scritto #333 \*

- **Evitare le proprietà ininfluenti:**

```
p { width: 500px; height: auto}  
span { position: absolute; display: block}
```

*height* è ininfluente, lo fa già il browser

*display: block* è inutile; *position: absolute* rende automaticamente l'elemento in *display: block*;

- **Non creare catene troppo specifiche** (usa le classi):

```
header nav ul li a {...} => .primary-link {...}
```

- **Riutilizzare il codice:** usate dei pattern da applicare agli elementi comuni (es. ombre e angoli arrotondati...).

- **Minificare e comprimere i file:** rimozione dai file sorgente di tutti i caratteri non necessari (ritorni a capo, spazi e commenti)

<https://cssminifier.com/>

<https://www.minifier.org/>

\*: ovviamente se il colore è definito per coppie, es: #aabbcc; #a8a8a8, #007733..., invece #d61524 non è abbreviabile.

## favicon

Cosa sono le favicon?

La *favicon* è un'icona che aiuta a identificare una pagina web.

Viene utilizzata per migliorare l'esperienza di navigazione in alcuni contesti chiave.

A seconda del browser e del contesto, potrebbero essere visualizzate:

- accanto al nome del sito in una scheda del browser
- in un elenco di segnalibri
- come icona di avvio su dispositivi mobile e desktop
- nella stessa barra degli indirizzi o nella scheda del browser
- nei "tiles" di windows 10
- ...

Il formato ICO è un formato di immagine per la visualizzazione delle icone e ha le sue radici nella prima versione del sistema operativo Windows — Microsoft Windows 1.0 — rilasciata nel 1985.

È fondamentalmente un *formato contenitore* di immagini per l'archiviazione di *uno o più file di immagine bitmap* (BMP, successivamente è stato aggiunto il PNG).

Nel 1999, Microsoft ha portato il formato ICO su Internet Explorer 5, come modo per aggiungere un segnalibro e per identificare un sito: nasce la *favicon*.

Inizialmente, le *favicon* sul Web avevano una dimensione di 16 pixel per lato, ma nel corso degli anni è stato aggiunto il supporto per l'inclusione di più dimensioni diverse:

32 × 32  
24 × 24  
48 × 48  
64 × 64  
128 × 128  
256 × 256

creazione file .ico: <https://redketchup.io/icon-converter>

approfondimento: <https://mobiforge.com/design-development/adding-favicons-in-a-multi-browser-multi-platform-world>

## Esempio di codice per inserire i vari tipi di favicon

```
<link rel="shortcut icon" href="./assets/img/favicon.png">

<link rel="shortcut icon" sizes="16x16 24x24 32x32 48x48 64x64" href="./assets/img/favicon/
favicon.ico">
<!-- Mobile (Android, iOS & others) -->
<link rel="apple-touch-icon" sizes="57x57" href="./assets/img/favicon/favicon-57.png">
<link rel="apple-touch-icon-precomposed" sizes="57x57" href="./assets/img/favicon/
favicon-57.png">
<link rel="apple-touch-icon" sizes="72x72" href="./assets/img/favicon/favicon-72.png">
<link rel="apple-touch-icon" sizes="114x114" href="./assets/img/favicon/favicon-114.png">
<link rel="apple-touch-icon" sizes="120x120" href="./assets/img/favicon/favicon-120.png">
<link rel="apple-touch-icon" sizes="144x144" href="./assets/img/favicon/favicon-144.png">
<link rel="apple-touch-icon" sizes="152x152" href="./assets/img/favicon/favicon-152.png">
```

## Esempio di codice per inserire i vari tipi di favicon

```
<!-- Windows 8 Tiles -->  
<meta name="application-name" content="Scotch Scotch scotch">  
<meta name="msapplication-TileImage" content="./assets/img/favicon/  
favicon-144.png">  
<meta name="msapplication-TileColor" content="#2A2A2A">  
<!-- iOS Settings -->  
<meta content="yes" name="apple-mobile-web-app-capable">  
<meta name="apple-mobile-web-app-status-bar-style" content="black-translucent">
```

## Favicon Generators:

<http://realfavicongenerator.net>

<https://favicomatic.com/>

<https://www.guru99.com/best-favicon-generators.html> [elenco dei migliori 25]

## Altre Risorse CSS (... oltre a quelle citate nelle slide precedenti )

### Generatori di codice

<https://bggenerator.com/> generatore di background

<https://cssgradient.io/> generatore di gradienti

<https://border-radius.com/> generatore di bordi arrotondati

<https://toolset.mrw.it/css/css3-box-shadow.html> generatore di ombra

<https://bennettfeely.com/clippy/> generatore di clip per background

<https://animista.net/> generatore di animazioni

### Font

<https://fonts.google.com/>

### Giochi

per imparare ad usare i selettori, per cimentarsi con flexbox, grid e altre divertenti sfide!

<https://nikitahl.com/learn-css-by-playing-games> elenco di 12 giochi CSS