



# CSS

Materiale didattico – CSS: competenze intermedie

Anno scolastico 2024/2025

# Selettori

## Semplici

```
/* selettore universale: tutti i tag */  
  
* {  
    color: red;  
}
```

```
<h1 id="about-us-header" class="red-header">La nostra azienda</h1>
```

```
/* selettore di tag */  
  
h1 {  
    color: red;  
}
```

```
/* selettore di classe */  
  
.red-header {  
    color: red;  
}
```

```
/* selettore di ID */  
  
#about-us-header {  
    color: red;  
}
```

# Selettori

## Combinatori

Si possono usare anche classi e id:  
per esempio `.container > .item`

```
/* combinatore di gruppo */  
h1, h2, h3 {  
  color: red;  
}
```

h1, h2 e h3

```
/* combinatore discendente */  
div h1 {  
  color: red;  
}
```

h1 discendenti di div

```
/* combinatore figlio */  
div > h1 {  
  color: red;  
}
```

h1 figli diretti di div

```
/* combinatore adiacente (fratello) */  
div + h1 {  
  color: red;  
}
```

il primo h1 successivo a div

```
/* combinatore successivo */  
h1 ~ p {  
  color: red;  
}
```

tutti i p successivi a h1

# Selettori

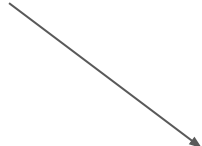
## Pseudo-classi (STATI)

Un link è **attivo** quando viene cliccato e il mouse è ancora premuto su di esso.

```
/* link da visitare */  
a:link {  
    color: #FF0000;  
}
```

```
/* link visitati */  
a:visited {  
    color: #00FF00;  
}
```

```
/* link attivo */  
a:active {  
    color: #0000FF;  
}
```



# Cursore

cursor

```
div {  
  cursor: pointer;  
}
```

I auto	↕ move	👉🚫 no-drop	↕ col-resize
⬮ all-scroll	👉 pointer	🚫 not-allowed	⇅ row-resize
+ crosshair	👉🕒 progress	↔ e-resize	↗ ne-resize
🖱 default	I text	↑ n-resize	↖ nw-resize
🖱? help	⇅ vertical-text	↓ s-resize	↗ se-resize
I inherit	🕒 wait	↔ w-resize	↘ sw-resize

# Selettori

## Pseudo-classi

```
div {  
  background-color: green;  
}
```

Mouse Over Me

```
/* selezionato */  
div:hover {  
  background-color: blue;  
}
```

Mouse Over Me

# Selettori

## Pseudo-classi

```
<ul>
  <li><a>Item 1</a></li>
  <li><a>Item 2</a></li>
  <li><a>Item 3</a></li>
  <li><a>Item 4</a></li>
</ul>
```

Esiste anche  
`nth-last-child()`  
che conta dal basso.

```
/* primo figlio */
li:first-child {
  color: #FF0000;
}
```

- Item 1
- Item 2
- Item 3
- Item 4

```
/* ultimo figlio */
li:last-child {
  color: #FF0000;
}
```

- Item 1
- Item 2
- Item 3
- Item 4

```
/* n-essimo figlio */
li:nth-child(2) {
  color: #FF0000;
}
```

- Item 1
- Item 2
- Item 3
- Item 4

# Selettori

## Specificità

### Priorità

CSS inline

CSS  
incorporato

CSS esterno

### 1 - Proprietà important

```
div {  
    color: red; !important  
}
```

### 2 - CSS inline

```
<div style="color: red"></div>
```

### 3 - Selettori id

```
#my-div {  
    color: red;  
}
```

### 4 - Selettori classe e Selettori pseudo-classe

```
.my-div {  
    color: red;  
}
```

### 5 - Selettori tag

```
div {  
    color: red;  
}
```



# Posizionamento

## Tag di blocco e in linea

- ❑ Un **tag di blocco** (*block-level*) è un elemento che occupa tutta la larghezza disponibile (quella dell'elemento padre) e inizia sempre su una nuova riga. Un elemento di blocco può contenere altri elementi di blocco o in linea.

Esempi: `<p>`, `<h1>`, `<ul>`, `<hr>`, `<header>`, `<footer>`, `<section>`, `<nav>`

- ❑ Un **tag in linea** (*inline-level*) è un elemento che non interrompe il flusso del testo e occupa solo lo spazio necessario. Un elemento in linea può contenere solo altri elementi in linea o del semplice testo.

Esempi: `<a>`, `<img>`, `<strong>`, `<em>`, `<label>`, `<input>`, `<code>`

# Posizionamento

## display

valore di default → `inline`

```
span {  
  display: inline;  
  color: red;  
}
```

Lorem ipsum dolor... **HELLO WORLD!** Vestibulum volutpat tellus diam....

```
span {  
  display: block;  
  color: red;  
}
```

Lorem ipsum dolor...  
**HELLO WORLD!**  
Vestibulum volutpat tellus diam....

```
span {  
  display: none;  
  color: red;  
}
```

Lorem ipsum dolor... Vestibulum volutpat tellus diam....

# Posizionamento

display: inline-block

A eccezione di alcuni elementi come `<img>` o `<video>`, gli elementi in linea generalmente non supportano proprietà del box model come `width`, `height` o `margin`. Per ovviare a questo, è possibile utilizzare il valore **inline-block** per la proprietà `display`, permettendo l'applicazione delle proprietà del box model. Un elemento *inline-block* può contenere anche altri elementi di blocco.

```
span {  
  display: inline-block;  
  width: 150px;  
}
```

Lorem ipsum dolor... **HELLO WORLD!**

Vestibulum volutpat tellus diam...

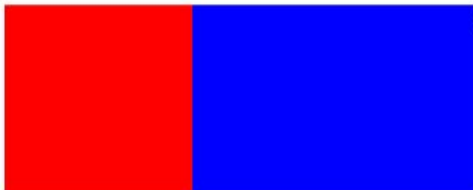
# Posizionamento

## float

Nota che un elemento di blocco non va a capo dopo un elemento flottuante.

La proprietà `float` sposta un elemento a sinistra (`left`) o a destra (`right`) rispetto al suo contenitore, permettendo ad altri elementi di fluirgli attorno.

```
<div style="float: left; width: 100px; height: 100px; background: red;"></div>  
<div style="float: left; width: 150px; height: 100px; background: blue;"></div>  
<p>Questo testo si dispone dopo l'elemento flottuante.</p>
```



Questo testo si dispone dopo l'elemento flottuante.

valore di default → `none`

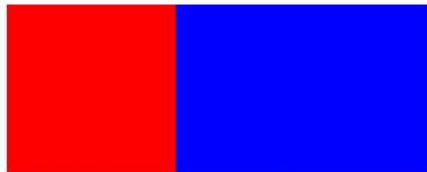
# Posizionamento

valore di default → `none`

## float

La proprietà `clear` interrompe il flusso del contenuto attorno agli elementi flottanti. Specifica se l'elemento non ammette elementi flottanti a sinistra (`left`), destra (`right`) o entrambi i lati (`both`).

```
<div style="float: left; width: 100px; height: 100px; background: red;"></div>  
<div style="float: left; width: 150px; height: 100px; background: blue;"></div>  
<p style="clear: left;">Questo testo si dispone dopo l'elemento flottante.</p>
```



Questo testo si dispone dopo l'elemento flottante.

# Posizionamento

## position

La proprietà `position` definisce come un elemento è posizionato nel documento specificando la sua distanza tramite le seguenti proprietà:

- `top`: scostamento dal bordo superiore dell'elemento o suo contenitore;
- `bottom`: scostamento dal bordo inferiore dell'elemento o suo contenitore;
- `left`: scostamento dal bordo sinistro dell'elemento o suo contenitore;
- `right`: scostamento dal bordo destro dell'elemento o suo contenitore.

I valori sono specificati in *px*, *%*, *em*, etc. È possibile usare anche valori negativi. Non ha senso usare contemporaneamente `left` e `right`, o `top` e `bottom`.

# Posizionamento

position: relative

L'elemento resta nel flusso normale, ma viene spostato rispetto alla sua posizione originale.

```
div.relative {  
  position: relative;  
  left: 30px;  
  border: 3px solid #73AD21;  
}
```

valore di default → `static`

## Position

An element with position: static; is positioned according to the normal flow:

```
position: static;
```

## Position

An element with position: relative; is positioned relative to its normal position:

```
position: relative;
```

# Posizionamento

position: absolute

valore di default → `static`

```
div.relative {  
  position: relative;  
  width: 400px;  
  height: 200px;  
  border: 3px solid #ffAD21;  
}
```

```
div.absolute {  
  position: absolute;  
  bottom: 20px;  
  right: 0;  
  width: 200px;  
  height: 100px;  
  border: 3px solid #73AD21;  
}
```

L'elemento viene rimosso dal flusso normale e  
posizionato rispetto al primo contenitore `relative`.

## Position

An element with `position: absolute;` is positioned relative to the nearest positioned ancestor:

`position: relative;`

`position: absolute;`



# Posizionamento

## z-index

valore di default → 0

**z-index** controlla l'ordine di sovrapposizione degli elementi lungo l'asse z (la profondità). Funziona solo su elementi `relative`, `absolute`, `fixed` o `sticky`.

```
div.white-box {  
  z-index: 0;  
  ...  
}
```

```
div.grey-box {  
  z-index: 1;  
  ...  
}
```

```
div.green-box {  
  z-index: -1;  
  ...  
}
```

An element with greater stack order is always above an element with a lower stack order.

Green box (z-index: -1)

Black box (z-index: 0)

Gray box (z-index: 1)

# Posizionamento

position: fixed

valore di default → `static`

L'elemento viene posizionato rispetto al viewport della finestra e non si muove durante lo scorrimento della pagina.

## Position

`position: fixed;`

An element with `position: fixed;` is positioned relative to the viewport:

## Position

`position: fixed;`

An element with `position: fixed;` is positioned relative to the viewport:

```
div.fixed {  
  position: fixed;  
  top: 20px;  
  right: 10px;  
  width: 200px;  
  border: 3px solid #73AD21;  
}
```

# Posizionamento

position: sticky

valore di default → `static`

L'elemento si comporta come *relative* fino a raggiungere una posizione specifica durante lo scorrimento, poi diventa *fixed* rispetto al contenitore.

```
div.sticky {  
  position: sticky;  
  top: 0;  
  padding: 5px;  
  background-color: #cae8ca;  
  border: 2px solid #4CAF50;  
}
```

Try to **scroll** inside this frame to understand how sticky positioning works.

I am sticky!

In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.

Scroll back up to remove the stickyness.

Some text to enable scrolling. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisse

I am sticky!

scroll position.

Scroll back up to remove the stickyness.

Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisse concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus ramidiandos nec et. Insciderint efficiantur his ad. Eum no

# Posizionamento

## opacity

valore di default → 1

`opacity` controlla il livello di trasparenza di un elemento, con valori che vanno da 0 (completamente trasparente) a 1 (completamente opaco).

```
div.white-box {  
  z-index: 0;  
  opacity: 0.5;  
  ...  
}
```

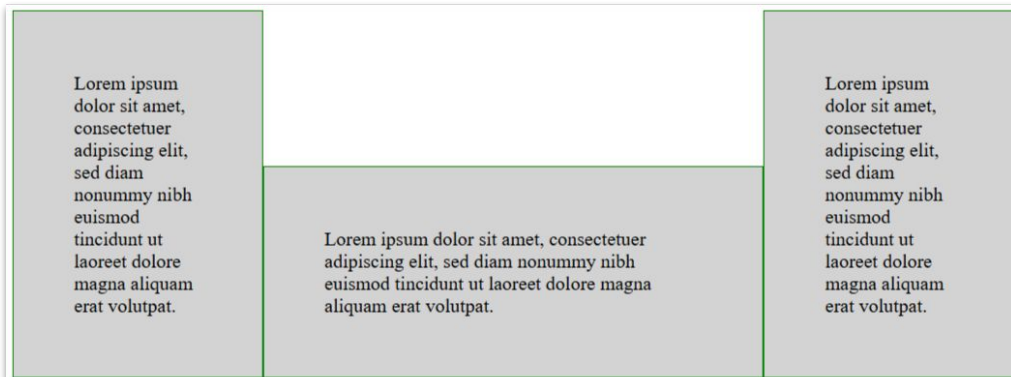
An element with greater stack order is always above an element with a lower stack order.



# Allineamento di blocchi

## Percentuali

NB: tra i `div` non ci devono essere spaziature!



```
.box {  
  box-sizing: border-box;  
  display: inline-block;  
  border: 1px solid green;  
  padding: 50px;  
  background-color: lightgrey;  
}  
.box1, .box3 {  
  width: 25%;  
}  
.box2 {  
  width: 50%;  
}
```

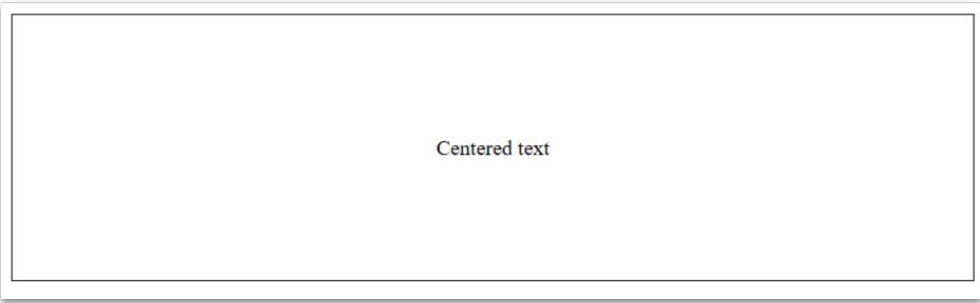
```
<div class="box box1">Lorem..</div><div class="box box2">Lorem..</div><div class="box box3">Lorem..</div>
```

# Centratura

## Testo (1)

Nel caso di contenitore ad altezza variabile..

```
.container {  
  text-align: center;  
  padding: 0 200px;  
  border: 1px solid #000000;  
}
```



Centered text

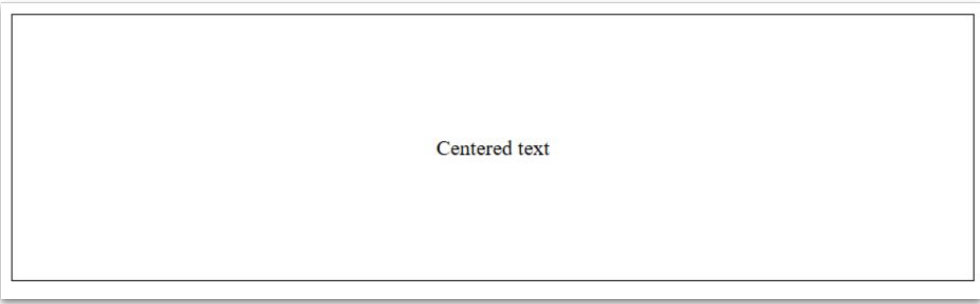
Funziona bene con il  
testo semplice, ma solo  
se non va a capo.

# Centrata

## Testo (2)

Nel caso di contenitore a dimensione fissa..

```
.container {  
  text-align: center;  
  line-height: 100px 0;  
  height: 200px;  
  border: 1px solid #000000;  
}
```



Centered text

# Centratura

## Immagini

```
.container {  
  height: 200px;  
  border: 1px solid #000000;  
  background-image: url('star.jpg');  
  background-repeat: no-repeat;  
  background-position: center;  
  background-size: contain;  
}
```





# Centratura

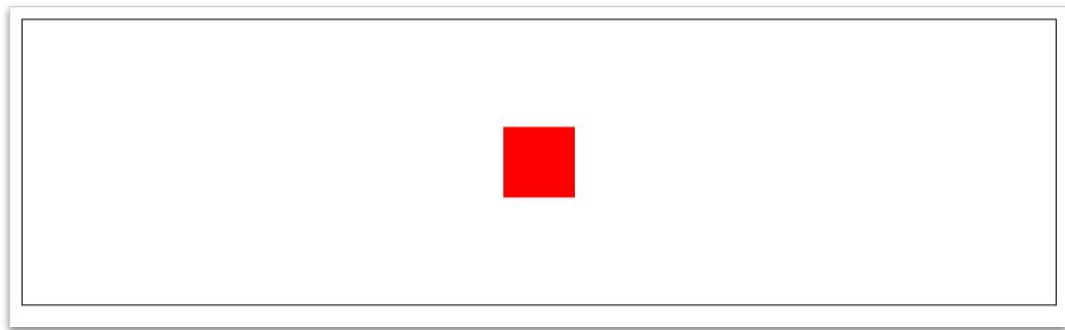
## Elementi di blocco (1)



```
.container {  
  position: relative;  
  height: 200px;  
  border: 1px solid #000000;  
}  
  
.center {  
  width: 50px;  
  height: 50px;  
  background-color: red;  
  position: absolute;  
  top: 50%;  
  left: 50%;  
  margin: -25px 0 0 -25px;  
}
```

# Centratura

## Elementi di blocco (2)



```
.container {  
  height: 200px;  
  border: 1px solid #000000;  
}  
  
.center {  
  width: 50px;  
  height: 50px;  
  background-color: red;  
  margin: 75px auto;  
}
```

# Flexbox

## Layout flessibile

I **flexbox** sono un sistema di layout potente e flessibile che consente di distribuire e allineare gli elementi (*flex items*) all'interno di un contenitore (*flex container*), sia orizzontalmente che verticalmente.

Un *flex container* attiva il modello flexbox con la proprietà `display: flex;` per un contenitore di blocco oppure `display: inline-flex;` per un contenitore inline. Tutti gli elementi figli diretti di un flex container sono automaticamente disposti in base al modello flexbox.

# Flexbox

display: flex

```
.flex-container {  
  display: flex;  
  background-color: dodgerblue;  
}  
  
.flex-container > div {  
  background-color: #f1f1f1;  
  width: 80px;  
  margin: 10px;  
  padding: 20px;  
  font-size: 30px;  
}
```



```
<div class="flex-container">  
  <div>1</div>  
  <div>2</div>  
  <div>3</div>  
  <div>4</div>  
</div>
```

# Flexbox

## flex-direction

valore di default → `row`

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



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



# Flexbox

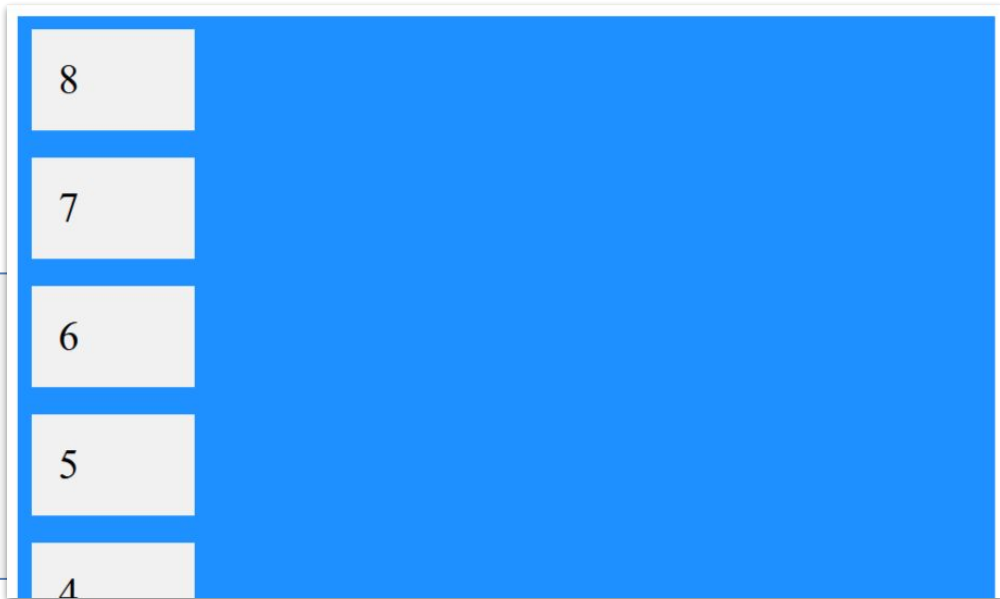
## flex-direction

valore di default → `row`

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



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



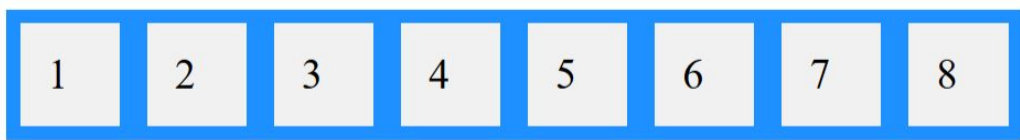
# Flexbox

## flex-wrap

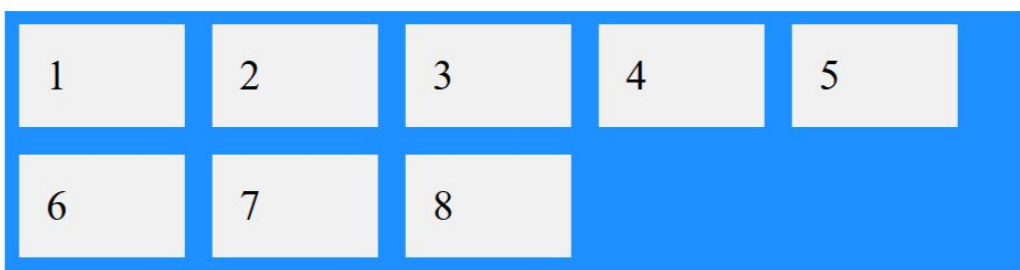
valore di default → nowrap

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

```
<div class="flex-container">  
  <div>1</div> <div>2</div> <div>3</div> <div>4</div>  
  <div>5</div> <div>6</div> <div>7</div> <div>8</div>  
</div>
```



```
.flex-container {  
  display: flex;  
  flex-wrap: wrap;  
  background-color: dodgerblue;  
}
```



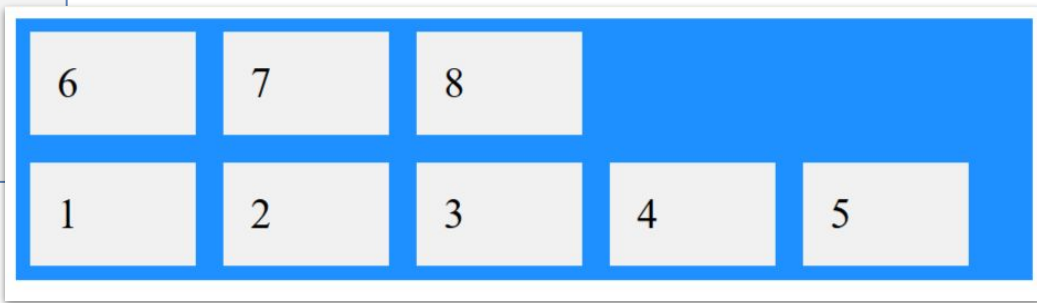
# Flexbox

## flex-wrap

valore di default → nowrap

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

```
<div class="flex-container">  
  <div>1</div> <div>2</div> <div>3</div> <div>4</div>  
  <div>5</div> <div>6</div> <div>7</div> <div>8</div>  
</div>
```



NB: con `wrap` e `wrap-reverse` la proprietà `width` dei figli viene rispettata. Invece con `nowrap` la proprietà `width` dei figli non viene rispettata!



# Flexbox

## flex-flow

Tramite la shortcut `flex-flow` è possibile specificare tutte le due proprietà precedenti (`flex-direction` e `flex-wrap`).

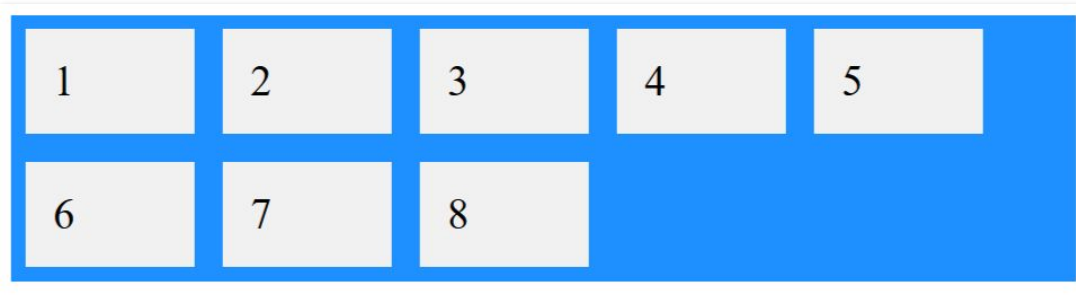
```
.flex-container {  
  flex-flow: row-reverse wrap;  
}
```

# Flexbox

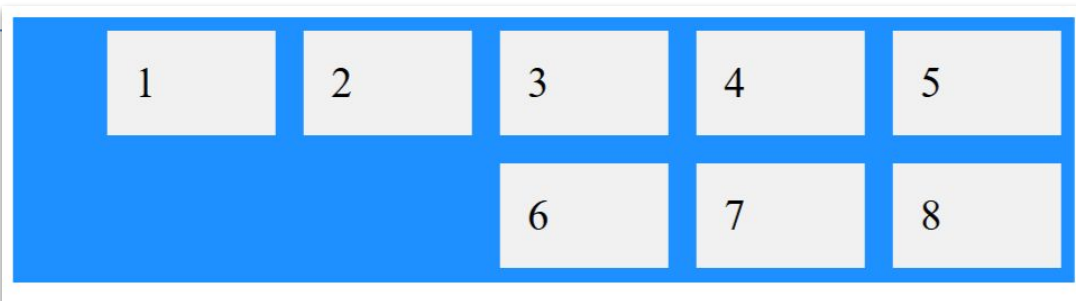
## Justify-content (inizio/fine)

valore di default → `flex-start`

```
.flex-container {  
  display: flex;  
  flex-wrap: wrap;  
  justify-content: flex-start;  
  background-color: dodgerblue;  
}
```



```
.flex-container {  
  display: flex;  
  flex-wrap: wrap;  
  justify-content: flex-end;  
  background-color: dodgerblue;  
}
```

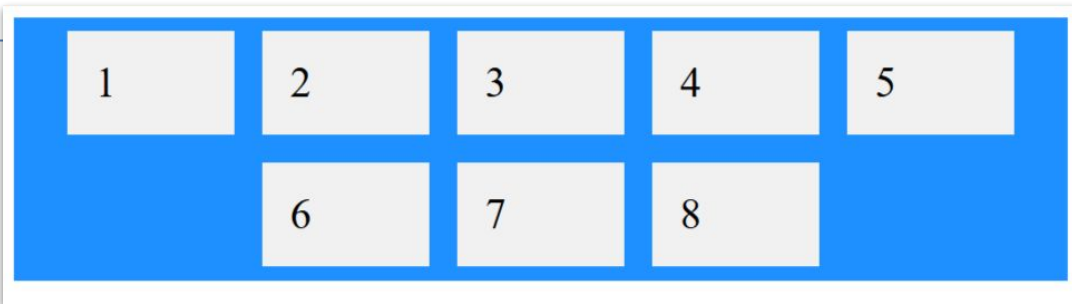


# Flexbox

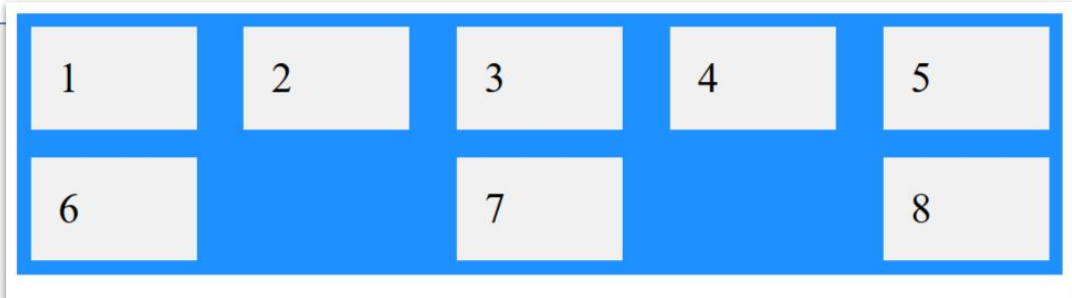
## Justify-content (centrato/tra)

valore di default → `flex-start`

```
.flex-container {  
  display: flex;  
  flex-wrap: wrap;  
  justify-content: center;  
  background-color: dodgerblue;  
}
```



```
.flex-container {  
  display: flex;  
  flex-wrap: wrap;  
  justify-content: space-between;  
  background-color: dodgerblue;  
}
```

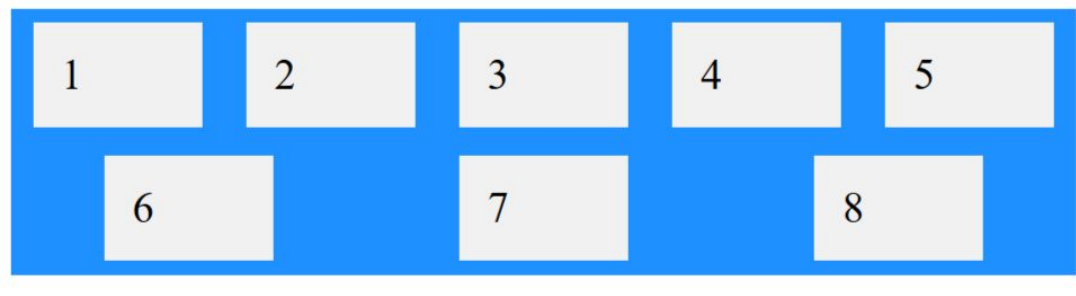


# Flexbox

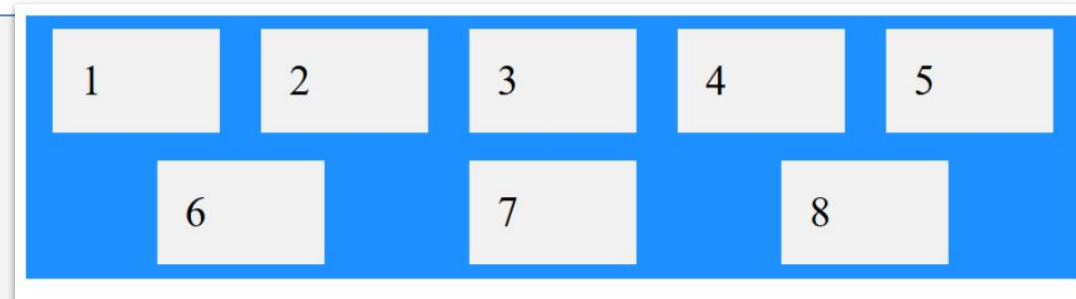
## Justify-content (intorno/uniforme)

valore di default → `flex-start`

```
.flex-container {  
  display: flex;  
  flex-wrap: wrap;  
  justify-content: space-around;  
  background-color: dodgerblue;  
}
```



```
.flex-container {  
  display: flex;  
  flex-wrap: wrap;  
  justify-content: space-evenly;  
  background-color: dodgerblue;  
}
```



# Flexbox

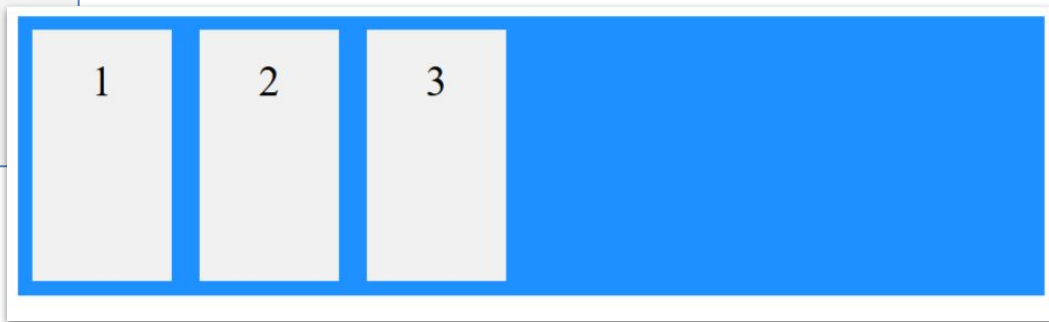
## Align-items (allineamento verticale)

valore di default → `normal`

I valori `normal` e `stretch` hanno lo stesso effetto.

```
.flex-container {  
  display: flex;  
  height: 200px;  
  align-items: stretch;  
  background-color: dodgerblue;  
}
```

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

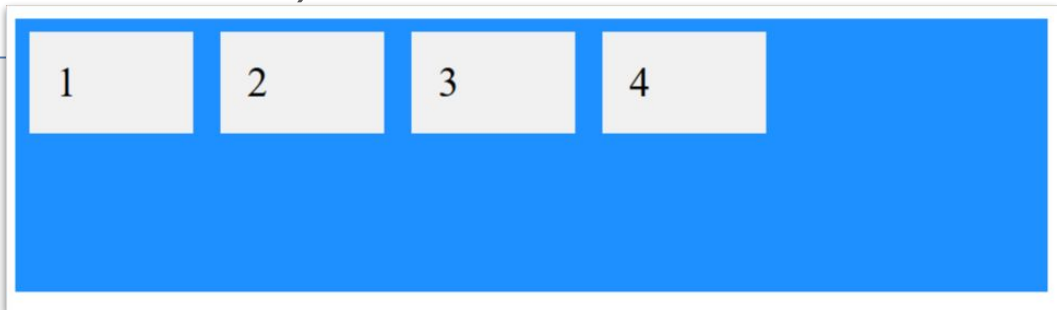


# Flexbox

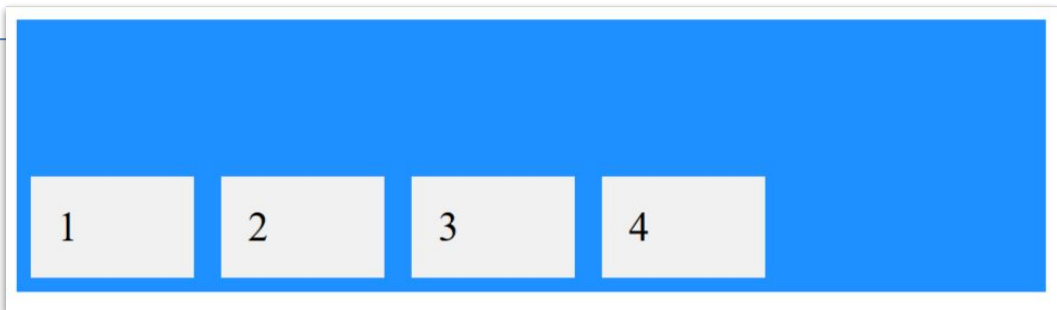
valore di default → `normal`

## Align-items (allineamento orizzontale)

```
.flex-container {  
  display: flex;  
  height: 200px;  
  align-items: flex-start;  
  background-color: dodgerblue;  
}
```



```
.flex-container {  
  display: flex;  
  height: 200px;  
  align-items: flex-end;  
  background-color: dodgerblue;  
}
```

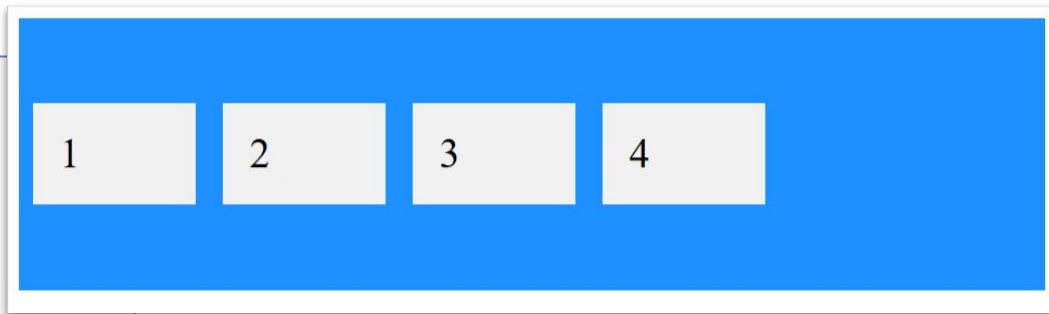


# Flexbox

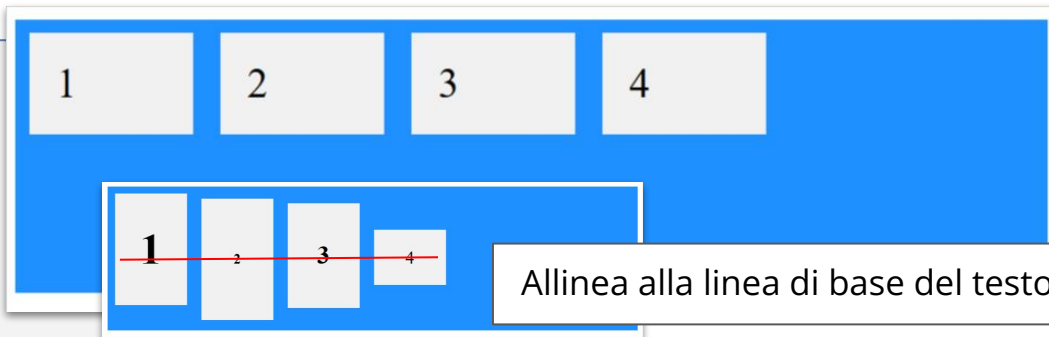
## align-items

valore di default → `normal`

```
.flex-container {  
  display: flex;  
  height: 200px;  
  align-items: center;  
  background-color: dodgerblue;  
}
```



```
.flex-container {  
  display: flex;  
  height: 200px;  
  align-items: baseline;  
  background-color: dodgerblue;  
}
```



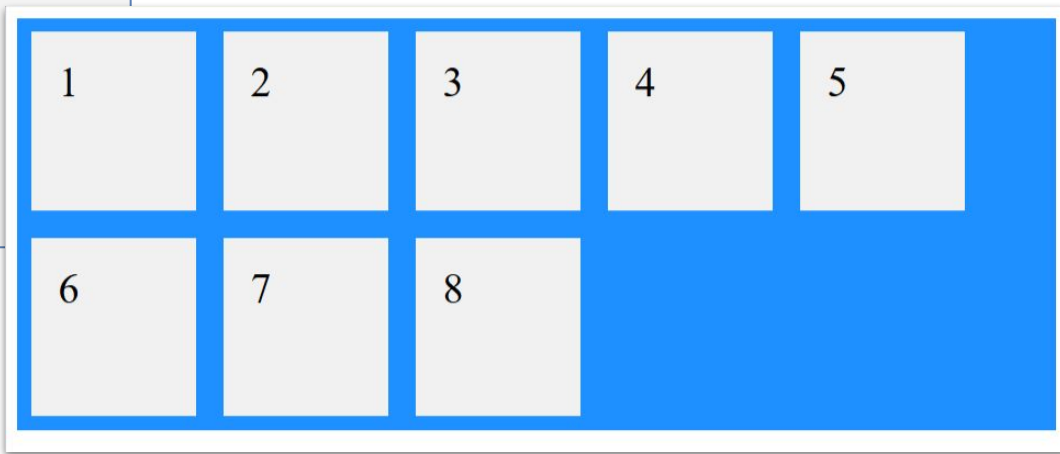
# Flexbox

## align-content

valore di default → `stretch`

```
.flex-container {  
  display: flex;  
  height: 300px;  
  align-content: stretch;  
  background-color: dodgerblue;  
}
```

```
<div class="flex-container">  
  <div>1</div> <div>2</div> <div>3</div> <div>4</div>  
  <div>5</div> <div>6</div> <div>7</div> <div>8</div>  
</div>
```



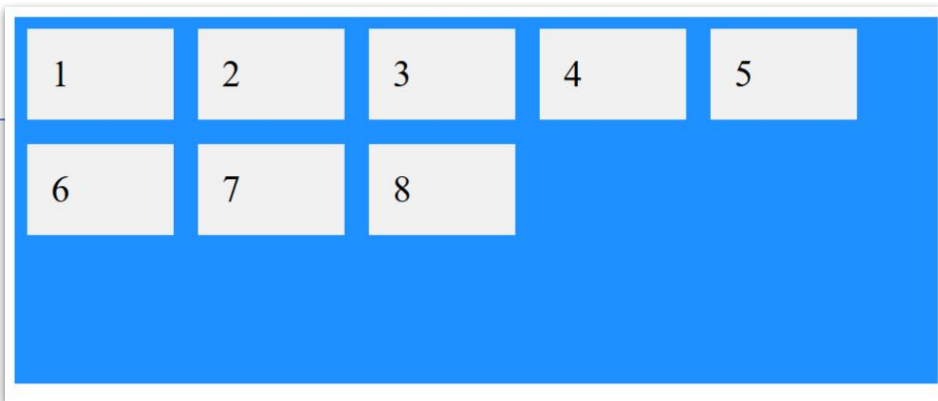


# Flexbox

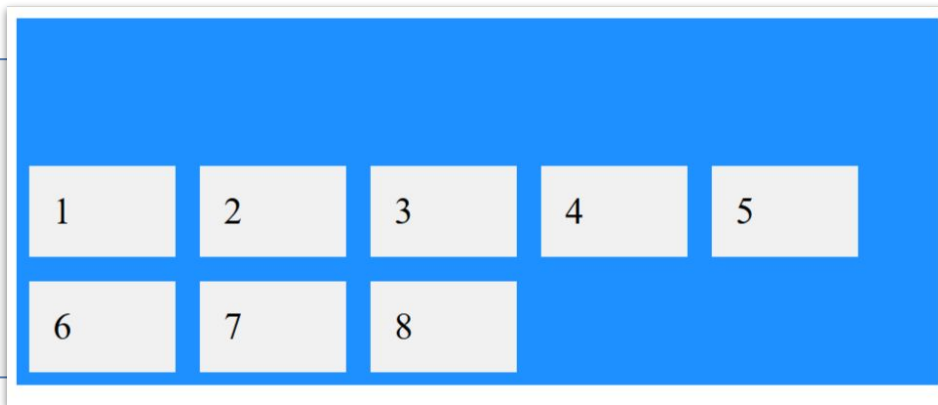
## align-content

valore di default → [stretch](#)

```
.flex-container {  
  display: flex;  
  height: 300px;  
  align-content: flex-start;  
  background-color: dodgerblue;  
}
```



```
.flex-container {  
  display: flex;  
  height: 300px;  
  align-content: flex-end;  
  background-color: dodgerblue;  
}
```

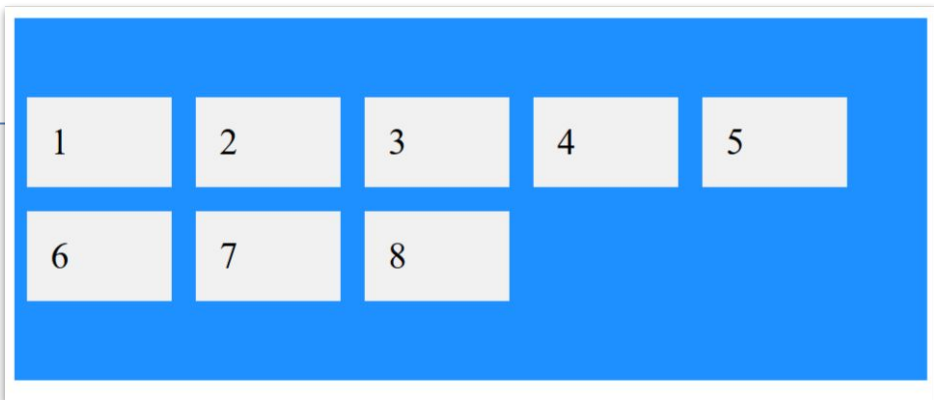


# Flexbox

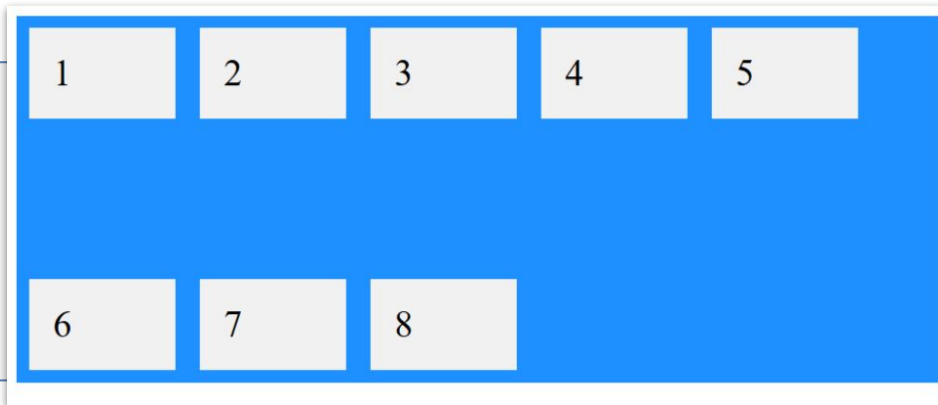
## align-content

valore di default → [stretch](#)

```
.flex-container {  
  display: flex;  
  height: 300px;  
  align-content: center;  
  background-color: dodgerblue;  
}
```



```
.flex-container {  
  display: flex;  
  height: 300px;  
  align-content: space-between;  
  background-color: dodgerblue;  
}
```

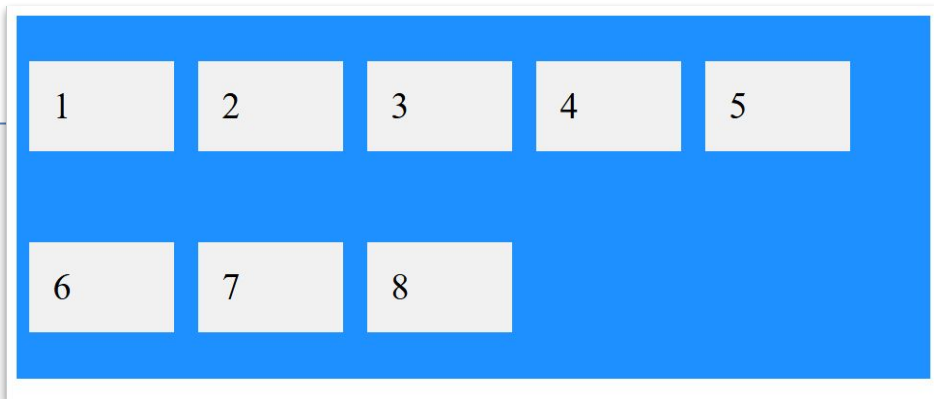


# Flexbox

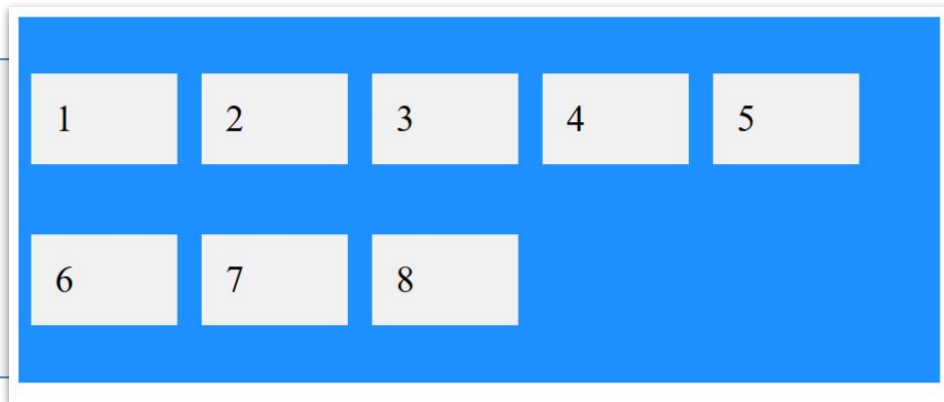
## align-content

valore di default → [stretch](#)

```
.flex-container {  
  display: flex;  
  height: 300px;  
  align-content: space-around;  
  background-color: dodgerblue;  
}
```



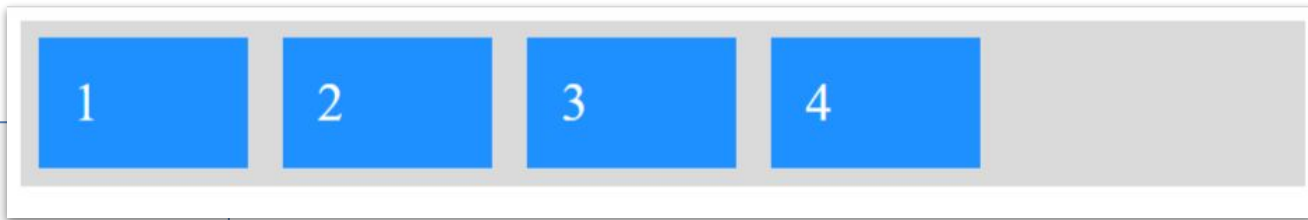
```
.flex-container {  
  display: flex;  
  height: 300px;  
  align-content: space-evenly;  
  background-color: dodgerblue;  
}
```



# Flexbox

## Flex items

```
.flex-container {  
  display: flex;  
  background-color: #dadada;  
  color: white;  
}  
  
.flex-container > div {  
  background-color: dodgerblue;  
  width: 80px;  
  margin: 10px;  
  padding: 20px;  
  font-size: 30px;  
}
```



```
<div class="flex-container">  
  <div>1</div>  
  <div>2</div>  
  <div>3</div>  
  <div>4</div>  
</div>
```

# Flexbox

## order

valore di default → 0



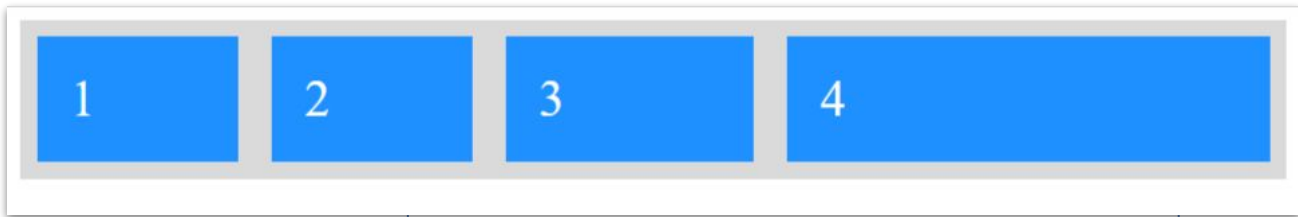
```
.flex-container > div {  
  background-color: dodgerblue;  
  width: 80px;  
  margin: 10px;  
  padding: 20px;  
  font-size: 30px;  
}
```

```
<div class="flex-container">  
  <div style="order: 3">1</div>  
  <div style="order: 2">2</div>  
  <div style="order: 4">3</div>  
  <div style="order: 1">4</div>  
</div>
```

# Flexbox

## flex-grow

valore di default → 0



```
.flex-container > div {  
  background-color: dodgerblue;  
  width: 80px;  
  margin: 10px;  
  padding: 20px;  
  font-size: 30px;  
}
```

```
<div class="flex-container">  
  <div style="flex-grow: 0">1</div>  
  <div style="flex-grow: 0">2</div>  
  <div style="flex-grow: 1">3</div>  
  <div style="flex-grow: 6">4</div>  
</div>
```

flex-grow: 1 → l'item può crescere fino al 100%  
flex-grow: 6 → l'item può crescere fino al 600%

# Flexbox

## flex-shrink

valore di default → 1



```
.flex-container > div {  
  background-color: dodgerblue;  
  width: 80px;  
  margin: 10px;  
  padding: 20px;  
  font-size: 30px;  
}
```

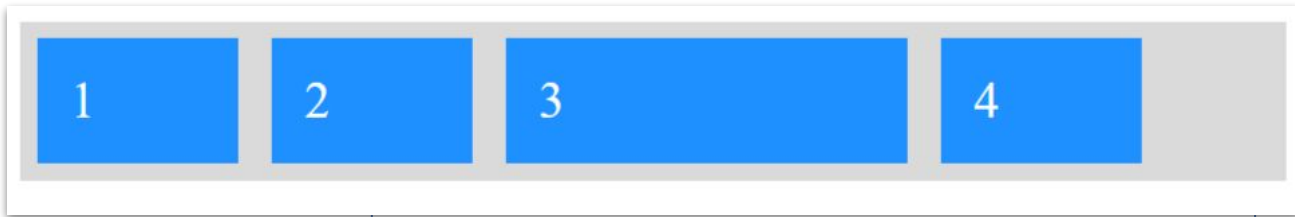
```
<div class="flex-container">  
  <div>1</div><div>2</div>  
  <div style="flex-shrink: 0">3</div>  
  <div style="flex-shrink: 2">4</div>  
  <div>5</div><div>6</div>  
  <div>7</div><div>8</div>  
</div>
```

`flex-shrink: 2` → l'item può rimpicciolirsi fino al 200% (rispettando lo spazio di contenuto e padding).

# Flexbox

## flex-basis

valore di default → 0



```
.flex-container > div {  
  background-color: dodgerblue;  
  width: 80px;  
  margin: 10px;  
  padding: 20px;  
  font-size: 30px;  
}
```

```
<div class="flex-container">  
  <div>1</div>  
  <div>2</div>  
  <div style="flex-basis: 200px">3</div>  
  <div>4</div>  
</div>
```

`flex-basis` → dimensione iniziale dell'item



# Flexbox

## flex

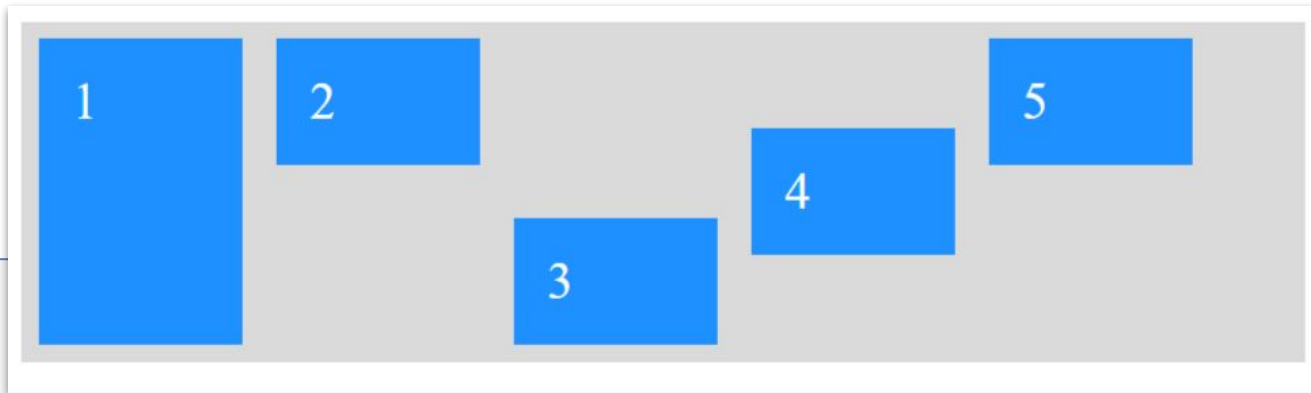
Tramite la shortcut `flex` è possibile specificare tutte le proprietà precedenti (`flex-grow`, `flex-shrink` e `flex-basis`).

```
.flex-container > div:first-child {  
  flex: 0 0 200px;  
}
```

# Flexbox

## align-self

```
.flex-container {  
  display: flex;  
  height: 200px;  
  background-color: #dadada;  
  color: white;  
}  
  
.flex-container > div {  
  background-color: dodgerblue;  
  width: 80px;  
  margin: 10px;  
  padding: 20px;  
  font-size: 30px;  
}
```



```
<div class="flex-container">  
  <div style="align-self: stretch">1</div>  
  <div style="align-self: flex-start">2</div>  
  <div style="align-self: flex-end">3</div>  
  <div style="align-self: center">4</div>  
  <div style="align-self: baseline">5</div>  
</div>
```

valore di default → `auto`

# Flexbox

## Centratura

```
.container {  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  height: 200px;  
  border: 1px solid #000000;  
}
```



# Overflow

## overflow

Esiste anche il valore `clip` ma è poco supportato.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

valore di default → `visible`

```
<div>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat</div>
```

```
div {  
  background: lightblue;  
  width: 110px;  
  height: 110px;  
  overflow: hidden;  
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh

```
div {  
  background: lightblue;  
  width: 110px;  
  height: 110px;  
  overflow: scroll;  
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam

```
div {  
  background: lightblue;  
  width: 110px;  
  height: 110px;  
  overflow: visible;  
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh

# Overflow

## text-overflow

valore di default → `clip`

```
<div>Lorem ipsum dolor sit amet.</div>
```

```
div {  
  white-space: nowrap;  
  width: 100px;  
  overflow: hidden;  
  text-overflow: clip;  
  border: 1px solid #000000;  
}
```

Lorem ipsum d

```
div {  
  white-space: nowrap;  
  width: 100px;  
  overflow: hidden;  
  text-overflow: ellipsis;  
  border: 1px solid #000000;  
}
```

Lorem ipsu...