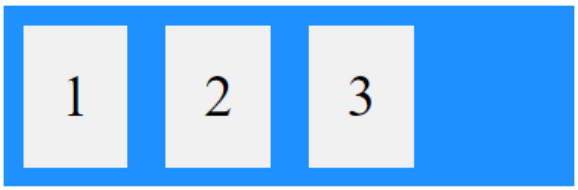
Introduzione a Flexbox

- **Premessa**: flexbox non è stato spiegato. Vi assicuro che può essere estremamente utile nella realizzazione dei progetti, soprattutto se volete creare pagine divise su più colonne e/o siti responsive. Quanto segue è un riassunto di quanto presente su w3schools.

- Introduciamo il layout **flexbox**, che permette di disegnare layout responsive senza utilizzare proprietà `float` e `position`.



- Vediamo un esempio introduttivo

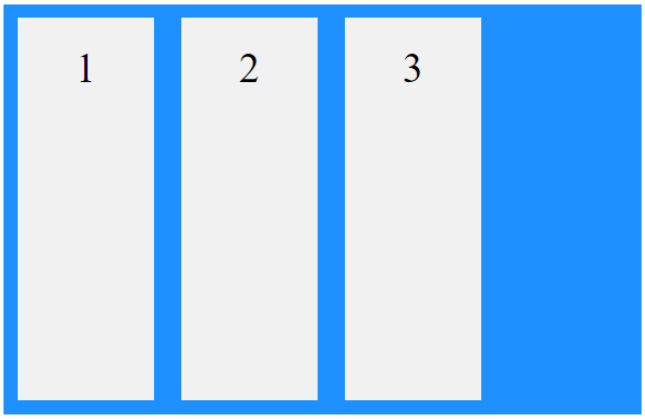
```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
  background-color: DodgerBlue;
  display: flex;
}

.flex-container > div {
  background-color: #f1f1f1;
  margin: 10px;
  padding: 20px;
  font-size: 30px;
}
</style>
</head>
<body>
  <div class="flex-container">
    <div>1</div>
    <div>2</div>
    <div>3</div>
  </div>
</body>
</html>
```

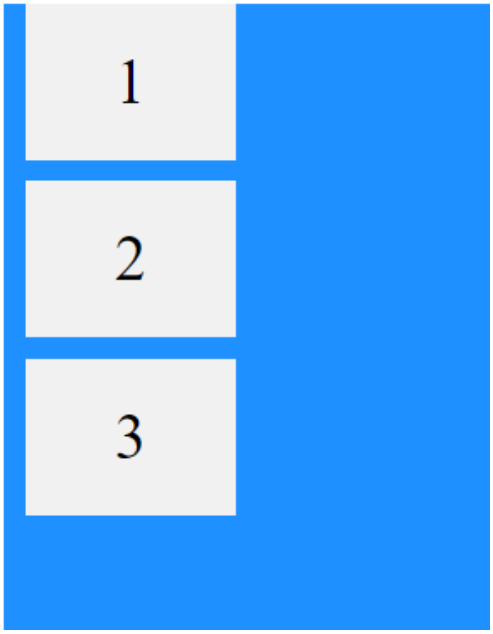
Un layout flessibile deve essere racchiuso in un elemento che presenta la proprietà `display: flex`

Gli elementi figli diretti diventano automaticamente flessibili.

Cosa succede se imposto `.flex-container` con `height: 300px`?



Gli elementi interni al container si sono adeguati alla nuova lunghezza, nonostante non sia definita una lunghezza (le uniche proprietà che determinano la grandezza di questi elementi sono padding e margin, oltre al contenuto)



Anteprima del codice con `height: 300px`, ma rimuovendo `display: flex`. Evidente la diversa disposizione degli elementi!

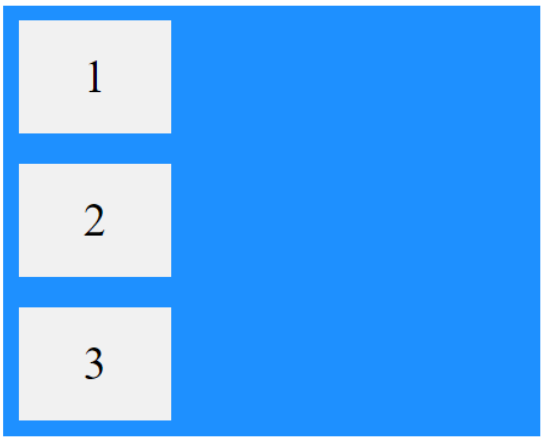
<u>Introdurremo nelle prossime pagine alcune proprietà utili</u>

Proprietà per il container

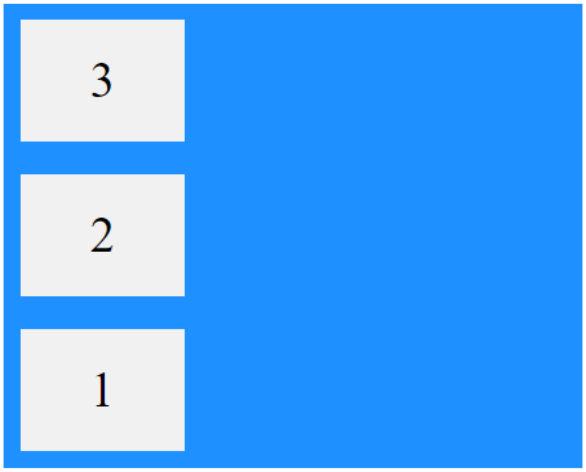
flex-direction

- Questa proprietà permette di definire la direzione in cui gli elementi del contenitore devono essere impilati.

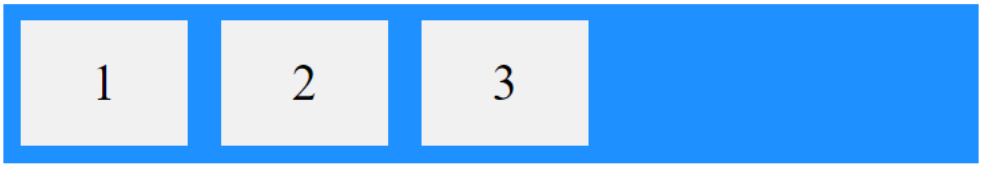
- **Valori possibili**:
  - `column`, impilo gli elementi flessibili verticalmente (dall'alto verso il basso)



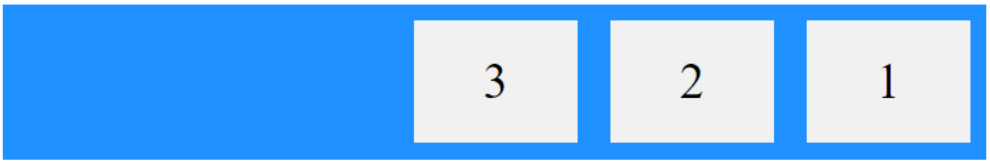
  - `column-reverse`, impilo gli elementi flessibili verticalmente (dal basso verso l'alto)



  - `row`, impilo gli elementi flessibili orizzontalmente (da sinistra a destra) (default)



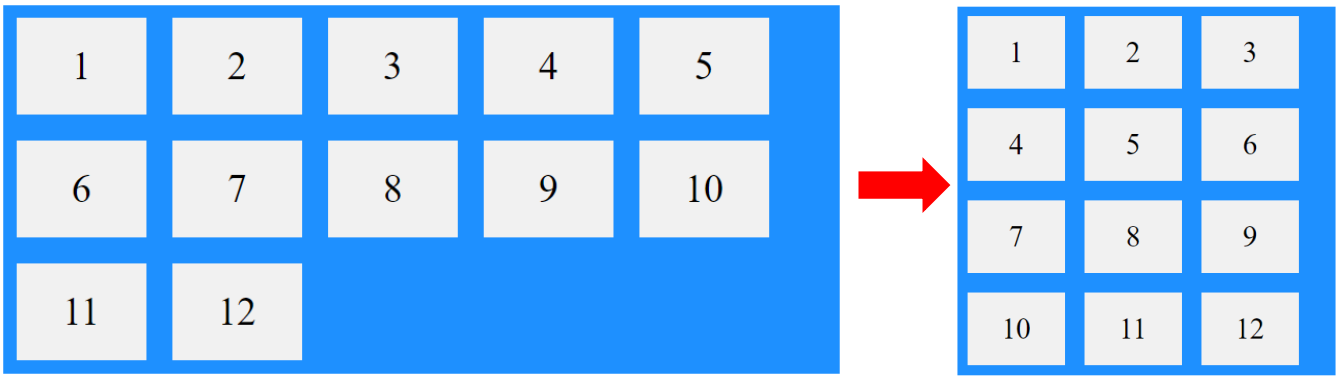
  - `row-reverse`



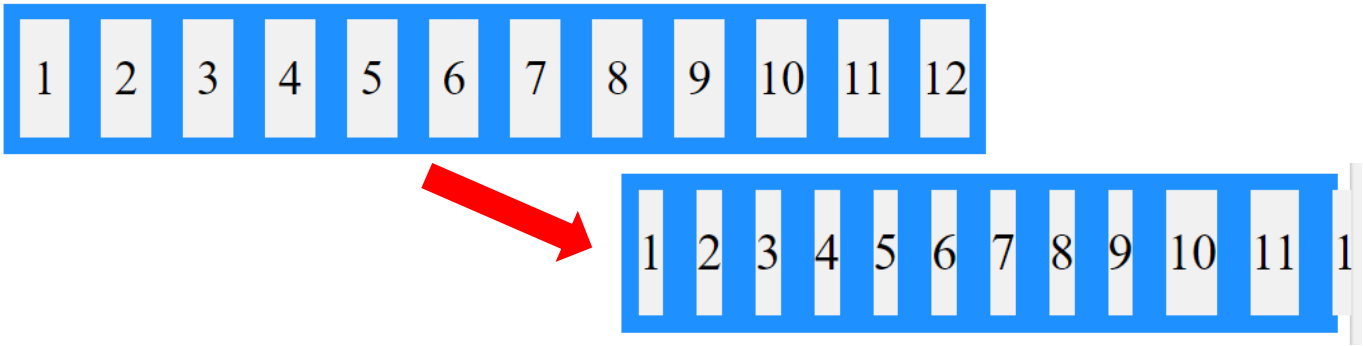
flex-wrap

- Questa proprietà specifica se gli elementi flessibili devono andare a capo o no.

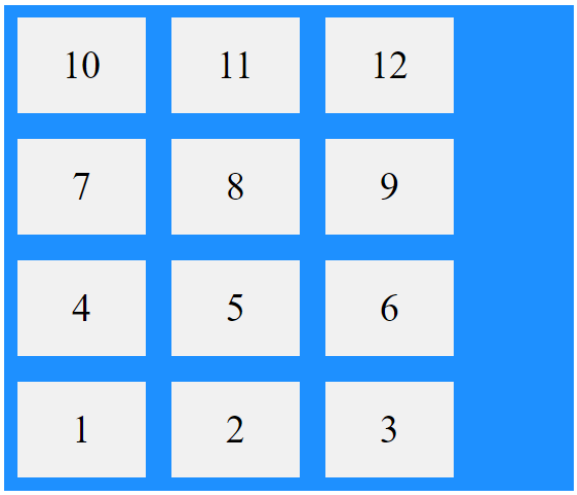
- **Valori possibili**:
  - `wrap`, gli elementi flessibili vanno a capo se necessario



  - `nowrap`, gli elementi flessibili non vanno a capo (default)



  - `wrap-reverse`, stesso effetto della `wrap`, ma con gli elementi posizionati nell'ordine inverso



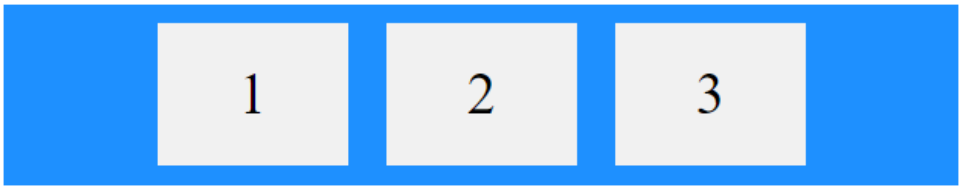
flex-flow

- Proprietà sintesi delle precedenti: possiamo specificare le due proprietà in un colpo solo!
- **Esempio**: `flex-flow: row wrap;`

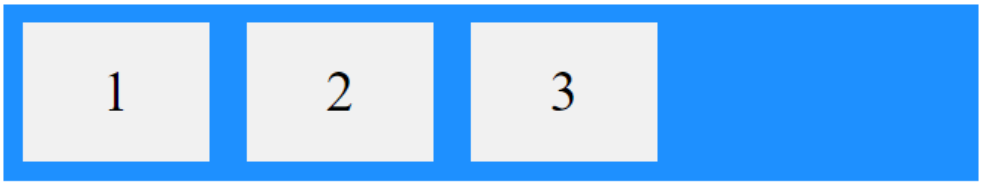
justify-content

- Proprietà che specifica l'allineamento degli elementi flessibili.

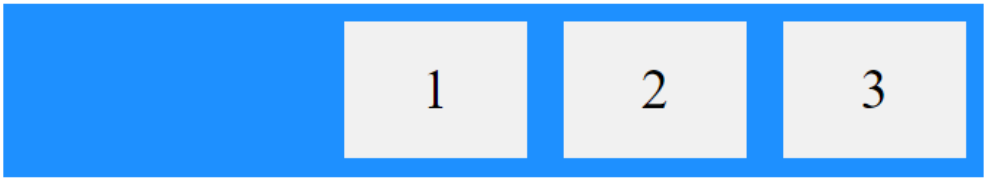
- **Valori possibili**:
  - `center`, elementi flessibili posti al centro del container



  - `flex-start`, elementi flessibili posti a sinistra del container (default)



  - `flex-end`, elementi flessibili posti alla fine del container



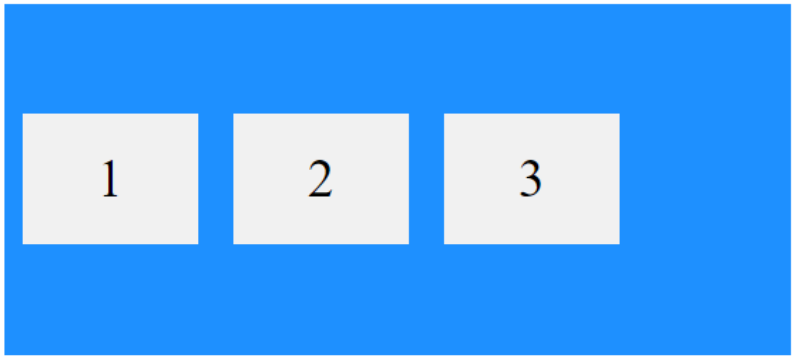
  - `space-around`, spazio tra gli elementi flessibili, ma anche prima e dopo la linea flex

  - `space-between`, spazio tra gli elementi flessibili.

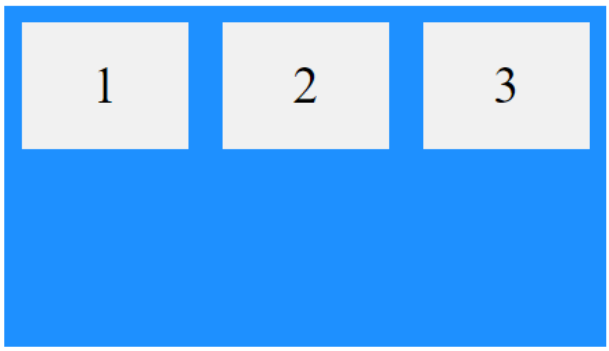
align-items

- Proprietà che specifica il posizionamento verticale degli elementi flessibili all'interno del container.

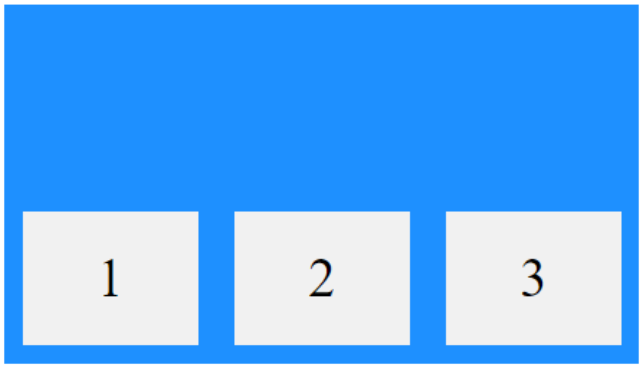
- **Valori possibili** (il container immaginato ha una height fissata):
  - `center`, elementi posti al centro del container



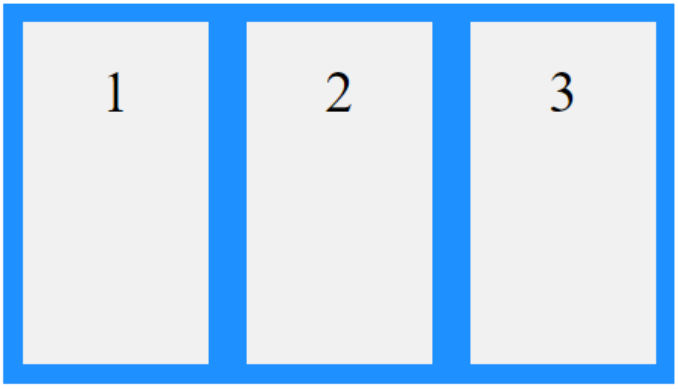
- `flex-start`, elementi posti in cima al container



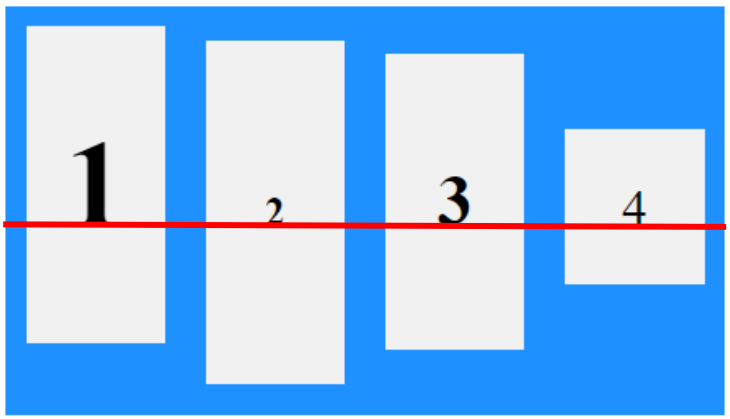
- `flex-end`, elementi posti in fondo al container



- `stretch`, l'altezza degli elementi flessibili dipende dalla lunghezza del container (default)



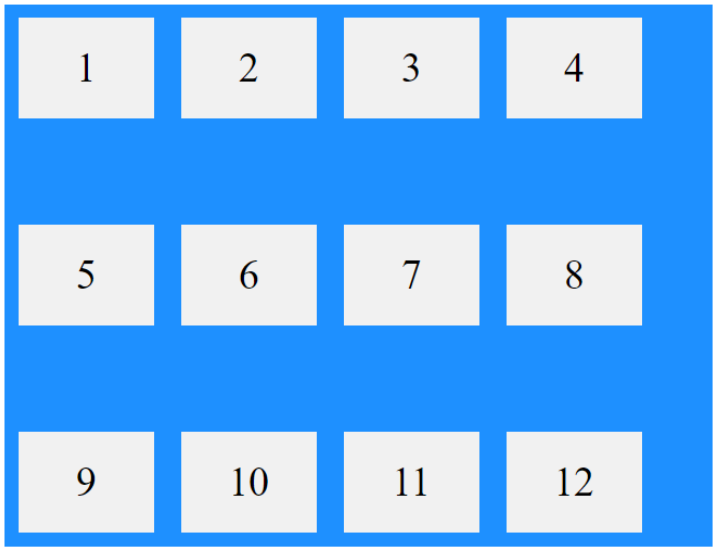
- `baseline`, gli elementi flessibili si allineano in base al baseline (attenzione ai numeri con font-size diverso)



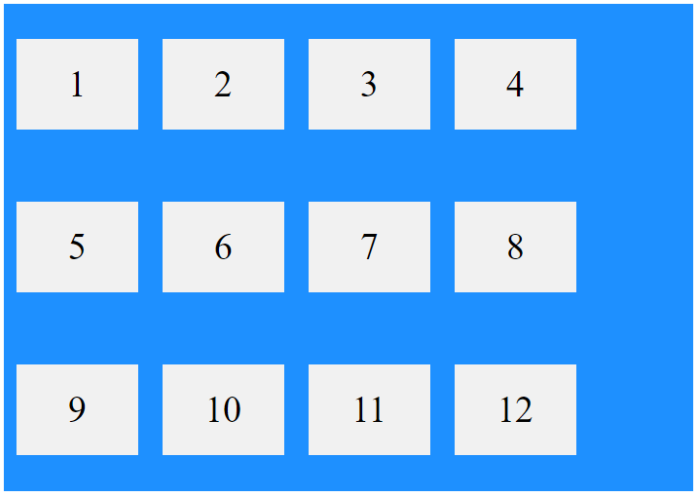
align-content

- Proprietà che specifica come allineare le linee flex.

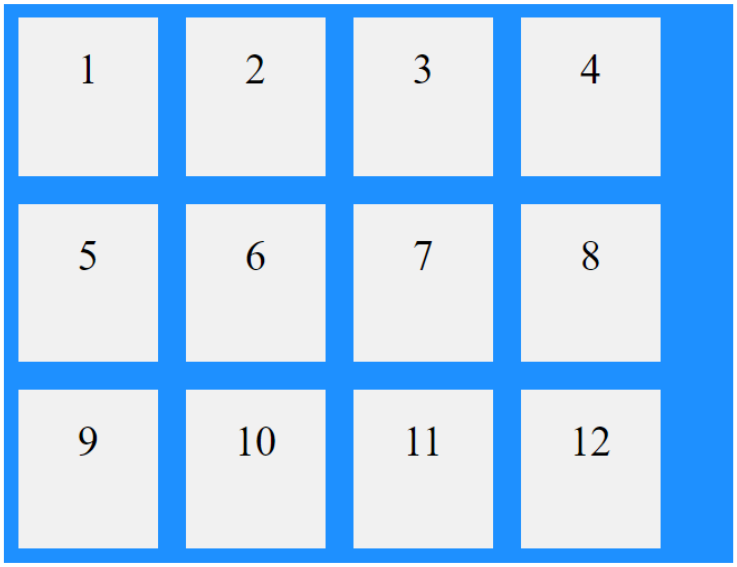
- **Valori possibili** (il container immaginato ha una height fissata):
  - `space-between`, spazio tra le linee



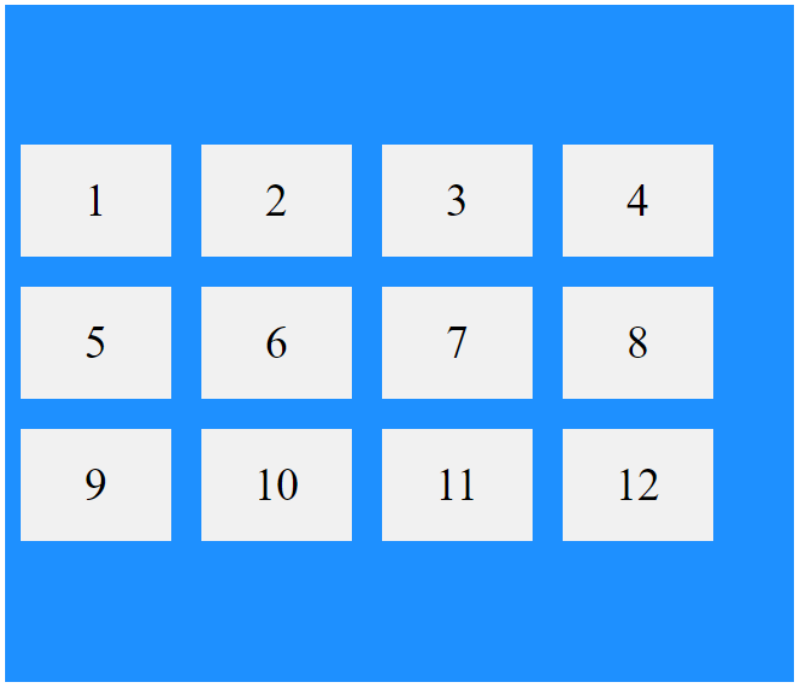
  - `space-around`, spazio tra linee (prima, tra loro, e dopo)



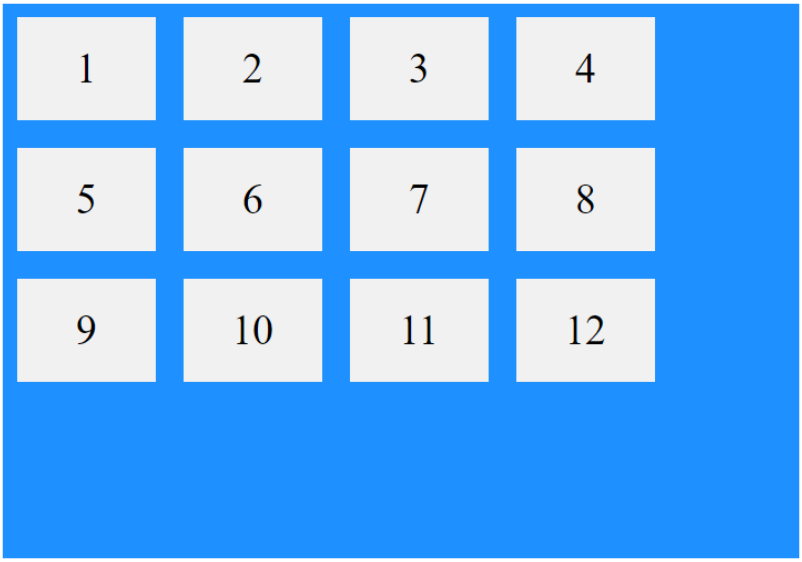
  - `stretch`, gli elementi flessibili sono allungati in modo tale da riempire tutta la lunghezza del container (default)



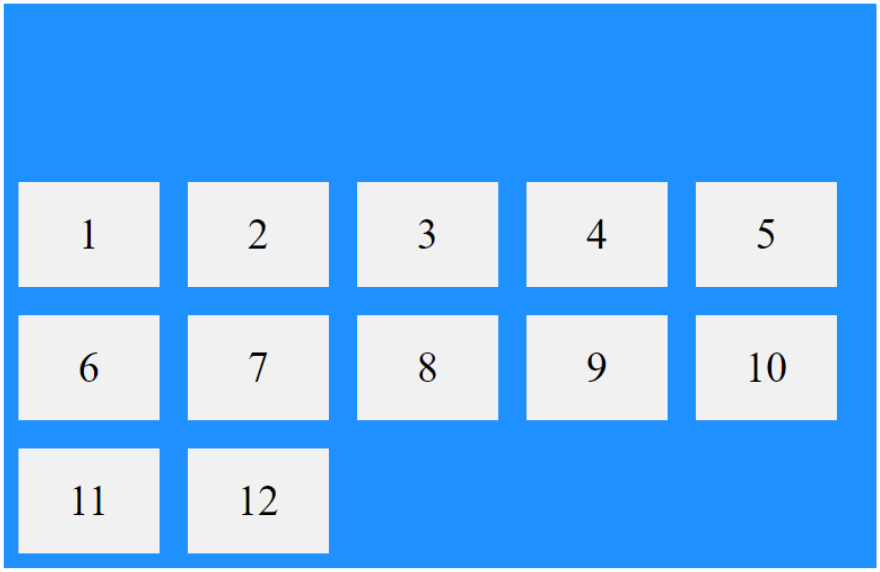
- `center`, linee poste nel mezzo del container



- `flex-start`, linee poste in cima al container



- `flex-end`, linee poste in fondo al container



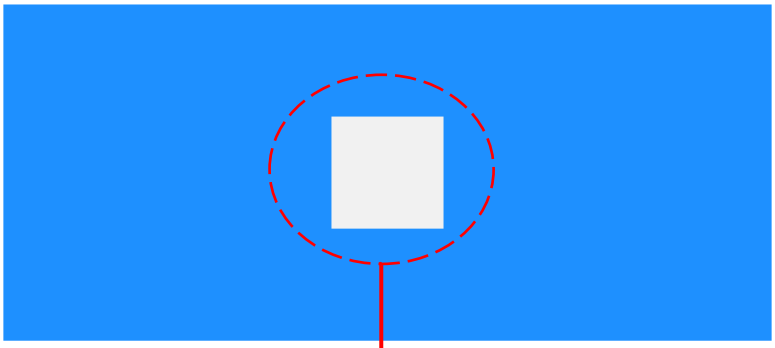
Elemento perfettamente centrato

```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 300px;
  background-color: DodgerBlue;
}

.flex-container>div {
  background-color: #f1f1f1;
  color: white;
  width: 100px;
  height: 100px;
}
</style>
</head>
<body>
  <h1>Perfect Centering</h1>

  <p>A container with both the justify-content and the align-items properties set to <em>center</em> will align the item(s) in the center (in both axis).</p>

  <div class="flex-container">
    <div></div>
  </div>
</body>
</html>
```



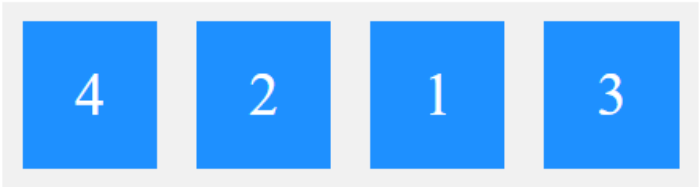
Anteprima

Proprietà per gli elementi

order

- Proprietà con cui possiamo stabilire l'ordine degli elementi flessibili.

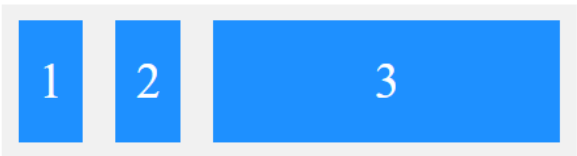
```
<div class="flex-container">
  <div style="order: 3">1</div>
  <div style="order: 2">2</div>
  <div style="order: 4">3</div>
  <div style="order: 1">4</div>
</div>
```



flex-grow

- Proprietà con cui specifichiamo quanto debba crescere un certo elemento flessibile rispetto agli altri.

```
<div class="flex-container">
  <div style="flex-grow: 1">1</div>
  <div style="flex-grow: 1">2</div>
  <div style="flex-grow: 8">3</div>
</div>
```



flex-shrink

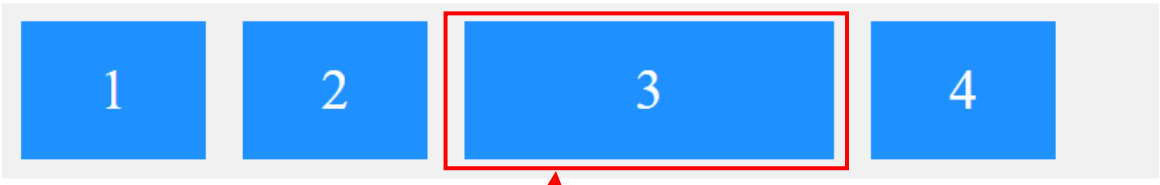
- Proprietà con cui specifichiamo debba restringersi un certo elemento flessibile rispetto agli altri.


```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex-shrink: 0">3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
  <div>7</div>
  <div>8</div>
  <div>9</div>
  <div>10</div>
</div>
```

Abbiamo stabilito che questo elemento non debba restringersi quanto gli altri elementi

flex-basis

- Imposto la larghezza iniziale di un certo elemento flessibile (possiamo esprimere la cosa anche in %)



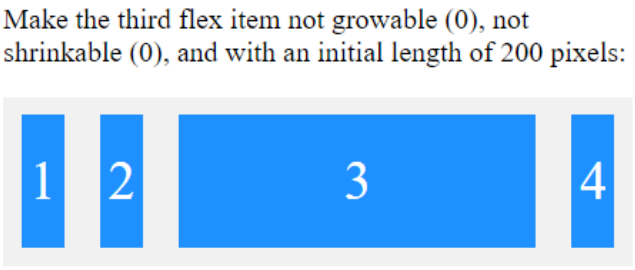

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex-basis:200px">3</div>
  <div>4</div>
</div>
```

Se allargo la schermata l'elemento rimane a 200px, se la riduco l'elemento avrà larghezza minore

flex

- Proprietà sintesi delle precedenti (`flex-grow`, `flex-shrink` e `flex-basis`)

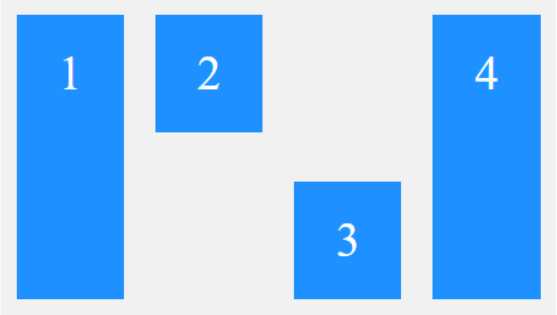
```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex: 0 0 200px">3</div>
  <div>4</div>
</div>
```



align-self

- Proprietà con cui specificare l'allineamento per un singolo elemento all'interno del container.

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



Responsive Flexbox

Laptop and Desktops:

1	2	3

Mobile phones and Tablets:

1
2
3

Possiamo realizzare siti responsive utilizzando la regola `@rule` assieme alle proprietà *flexbox*:

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

/* Responsive layout - makes a one column layout instead of a two-column layout */
@media (max-width: 800px) {
  .flex-container {
    flex-direction: column;
  }
}
```

Le strategie sono infinite, come le vie del signore. Possiamo fare la cosa anche così, indicando width specifici

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

.flex-item-left {
  flex: 50%;
}

.flex-item-right {
  flex: 50%;
}

@media (max-width: 800px) {
  .flex-item-right, .flex-item-left {
    flex: 100%;
  }
}
```