~~JavaScript~~ HTML5/CSS3

Introduzione a HTML/CSS3 (2)

Giovanni Iovino: giovanni.iovino@intecs.it

Contenuto:

Seconda parte di introduzione a HTML e CSS per la creazione di interfacce grafiche.

JS

CSS

Cascade Style Sheet

CSS - display

La proprietà **display** indica in che modo deve essere visualizzato un elemento html, permettendoci di modificare lo stile di visualizzazione di default di un elemento. Per i possibili valori della proprietà fare riferimento a https://www.w3schools.com/cssref/pr_class_display.asp. Alcuni esempi:

- **inline:** L'elemento non ha dimensioni proprie (width e height non hanno effetto). Comportamento di default dei tag span.
- **block:** Comportamento di default di div e p. L'elemento occupa tutta la larghezza disponibile e viene visualizzato a capo rispetto all'elemento precedente.
- **none:** L'elemento non è visualizzato e non occupa alcuno spazio
- **inline-block:** Come inline, ma è possibile utilizzare le proprietà width e height
- **flex:** L'elemento è visualizzato come un FlexBox (approfondito successivamente)
- **grid:** L'elemento è visualizzato come un GridBox (approfondito successivamente)

```
<style>
  *{
    box-sizing: border-box;
    margin: 5px;
  }
  .none{
    display: none;
    padding: 20px;
    background-color: yellow;
    width: 400px;
  }
  .inline{
    display: inline;
    padding: 20px;
    background-color: yellow;
    width: 400px;

  }
  .block{
    display: block;
    padding: 10px;
    background-color: red;
    width: 70px;
  }
  .inline-block{
    display: inline-block;
    padding: 20px;
    background-color: green;
    width: 400px;
  }
</style>
```

```
<body>
   <div class="inline">Contenuto inline .....</div>
   <div class="none">Contenuto None ....</div>
   <div class="inline-block">Contenuto inline block</div>
   <div class="block">Block</div>
</body>
```

CSS - box sizing

- Di default la grandezza di un elemento html è calcolata considerando la somma di **L = width|height + padding + border**
- La proprietà **box-sizing** ci permette di cambiare il comportamento di default facendo in modo che la la grandezza dell'elemento corrisponda al valore delle proprietà **width** o **height** assorbendo i valori di padding e border, ovvero **L = width|height:**
  - **box-sizing: content-box** comportamento di default
  - **box-sizing: border-box** comportamento alternativo

```
<style>
  div{
    width: 200px;
    height: 200px;
    padding-top: 100px;
    padding-bottom: 30px;
    padding-left: 50px;
    padding-right: 50px;
    border: 20px solid #cc0000;
    background-color: #ff0000;
    margin: 15px;
  }
  p{
    background-color: #ffff00;
    text-align: center;
  }
  .content{
    box-sizing: content-box;
  }
  .box{
    box-sizing: border-box;
  }
</style>
```

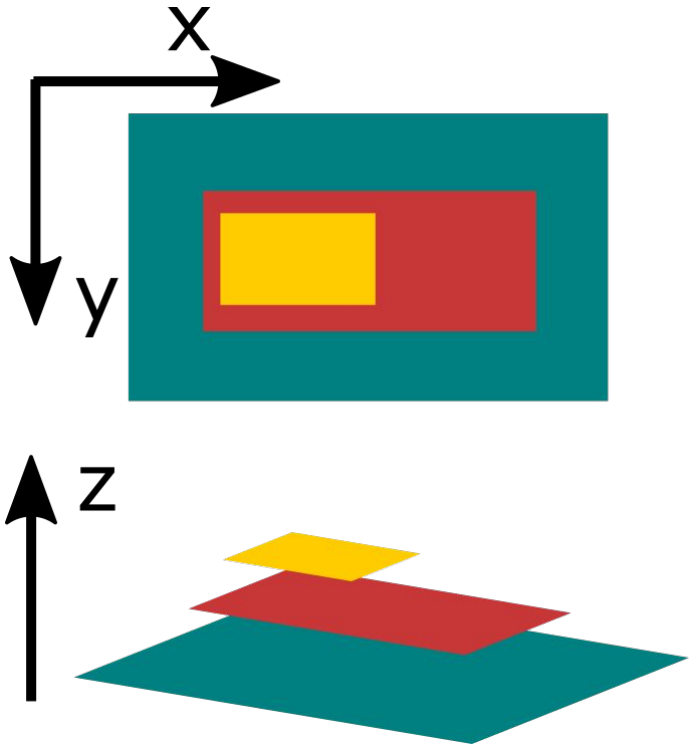
```
<body>
  <div class="content"><p>Ciao</p></div>
  <div class="box"><p>Ciao</p></div>
</body>
```

CSS - positioning

Normalmente gli elementi html sono visualizzati utilizzando un posizionamento che segue il flusso della struttura html, rispettando le regole associate alle proprietà display, width, height, border, padding e margin.

Inoltre il posizionamento degli elementi html tiene conto della gerarchia degli elementi, applicando delle regole di visualizzazione **basate sui layer**:

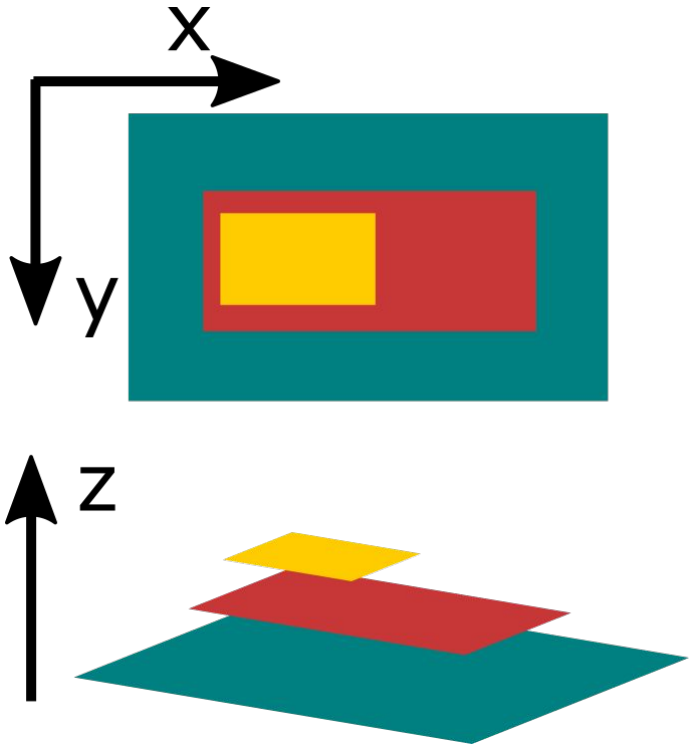
"gli elementi child sono visualizzati "sopra" i relativi elementi parent (ordinamento su un asse immaginario z)"



CSS - positioning

Relativamente alle regole di posizionamente gli elementi html possono essere di due tipi:

- **non posizionati:** Si seguono le regole di visualizzazione di default basate sul flow del documento html
- **posizionati:** Gli elementi sono visualizzati seguendo delle regole di posizionamento orizzontale (x,y) e verticale (z) definite da alcune proprietà CSS:
  - **position:** tipo di posizionamento da applicare
  - **top, left ,right, bottom:** definiscono l'offset per il posizionamento dell'elemento
  - **z-index:** definisce l'ordinamento dell'elemento relativamente all'asse immaginario z.



CSS - positioning

La proprietà **position** può assumere i seguenti valori:

- **static:** Valore di default, per elementi **non posizionati** che seguono il normale flow della struttura html. Le proprietà **top, left, right, bottom** non hanno effetto.
- **relative:** L'Elemento segue ancora il flow della pagina, ma è possibile posizionarlo relativamente a se stesso usando le proprietà **top, left, right, bottom.**
- **absolute:** L'elemento è rimosso dal flow di struttura e posizionato vicino il primo elemento padre **di tipo posizionato** o se non esiste vicino il root. Le proprietà **top, left, right, bottom** definiscono l'offset dell'elemento.
- **fixed:** Come absolute, ma l'elemento è posizionato sempre vicino all'elemento root
- **sticky:** Come relative, segue il flow della pagina, ma se è posizionato all'interno di un elemento scrollabile, in caso di scroll l'elemento è posizionato con offset relativo all'elemento padre scrollabile.

```
<style>
  *{
    box-sizing: border-box;
    margin: 5px;
  }
  .container{
    position: static;
    display: block;
    background-color: green;
    padding:0;
    width: 400px;
    height: 200px;
    margin-top: 100px;
  }
  /*provare a cambiare la proprietà position */
  .child{
    position: static;
    display: block;
    background-color: cyan;
    padding:0;
    left: 50px;
    top:80px;

  }
</style>
```

```
<body>
  <div class="container">
    <div class="child">Child position</div>
  </div>
</body>
```

Esempio Sticky

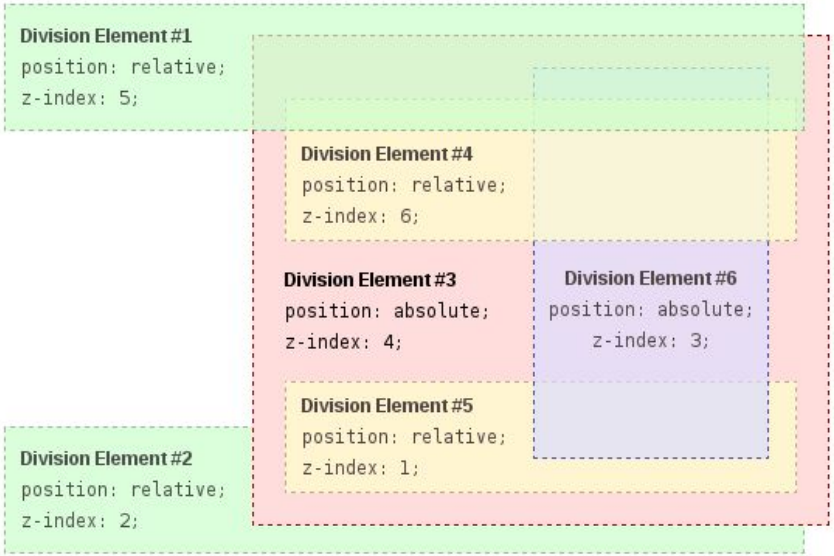
```
<body>
  <div class="container">
 <div class="sticky-container">

     <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
     <div class="child">Sticky</div>
     <p>
      Praesent nec tincidunt leo, eget tincidunt erat. Praesent interdum sollicitudin nibh, at dictum erat iaculis et. Sed ultrices imperdiet enim, at vestibulum arcu blandit ut. Quisque eget blandit velit. Cras vitae lacus risus. Nunc aliquam lacinia felis ut lacinia. Duis at quam scelerisque orci fringilla vehicula ut et est. Mauris pretium fringilla ipsum, eget iaculis libero ullamcorper vitae. Fusce maximus varius dolor et sagittis. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas massa ex, hendrerit eu interdum eget, maximus ac turpis. Integer ac scelerisque quam. Vivamus id ullamcorper velit. Nunc congue diam nec risus aliquam, in dapibus turpis pharetra. Donec a erat sed sem maximus sodales. Integer eu tortor nec enim fringilla sodales.
 </div>
</div>
</body>
```

```
<style>
  *{
    box-sizing: border-box;
    margin: 5px;
  }
  .container{
    position: static;
    display: block;
    background-color: #ccffcc;
    padding:0;
    width: 400px;
    height: 200px;
    margin-top: 100px;

  }
  .sticky-container {
      padding: 10px;
      height: 100px;
      overflow-y: scroll;
      background-color: #ccccff;
  }
  /*provare a cambiare la proprietà position */
  .child{
    position: sticky;
    display: block;
    background-color: #ffcccc;
    padding:0;
    left: 0px;
    top:0px;
    width:100px;
  }
</style>
```

CSS - stacking-context & z-index

Abbiamo detto che gli elementi sono ordinati su un asse z immaginario che permette la visualizzazione di elementi sovrapposti. L'ordinamento sull'asse z non è assoluto ma relativo, nello specifico gli elementi di tipo **posizionato** creano quello che viene definito **stacking-context**. All'interno di ciascun stacking-context il valore della proprietà **z-index** determina l'ordinamento dell'elemento.

Per le regole di creazione degli stacking-context fare riferimento a:
https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Positioning/Understanding_z_index/The_stacking_context



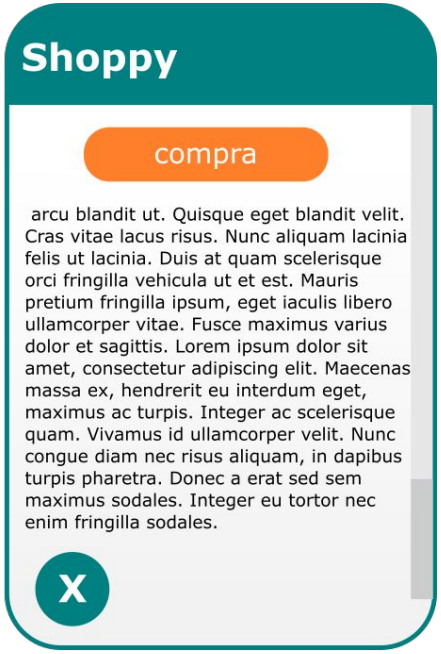
Esempio stacking-context

```html
<body>
  <div class="box1">
    box1
  </div>
  <div class="box2">
    box2
  </div>

  <div class="box3">
    <div class="box4">
      box4
    </div>
    box3
  </div>
</body>
```

```css
<style>
  *{
    box-sizing: border-box;
    margin: 0;
    padding:0;
  }
  .box1{
    position: absolute;
    display: block;
    background-color: #ffffcc;
    width: 80%;
    height: 200px;
    top: 80px;
    left:50px;
    z-index: 2;
  }
  .box2{
    position: absolute;
    display: block;
    background-color: #ffcccc;
    width: 400px;
    height: 200px;
    top: 20px;
    left:40px;
    z-index: 1;
  }
  .box3{
    position: absolute;
    display: block;
    background-color: #ccffcc;
    width: 200px;
    height: 200px;
    top: 150px;
    left:80px;
    z-index: 3;
  }
  .box4{
    position: absolute;
    display: block;
    background-color: #ffccff;
    width: 100px;
    height: 100px;
    top: 50px;
    left:50px;
    z-index: 1;
  }
</style>
```

Esercizio

Provare a creare un componente che rispecchi il mockup dell'immagine a lato



CSS - FlexBox

FlexBox offre un modello di layout che permette di creare componenti organizzati per righe o per colonne, offrendo la possibilità di centrare gli elementi e il loro contenuto sia verticalmente che orizzontalmente.

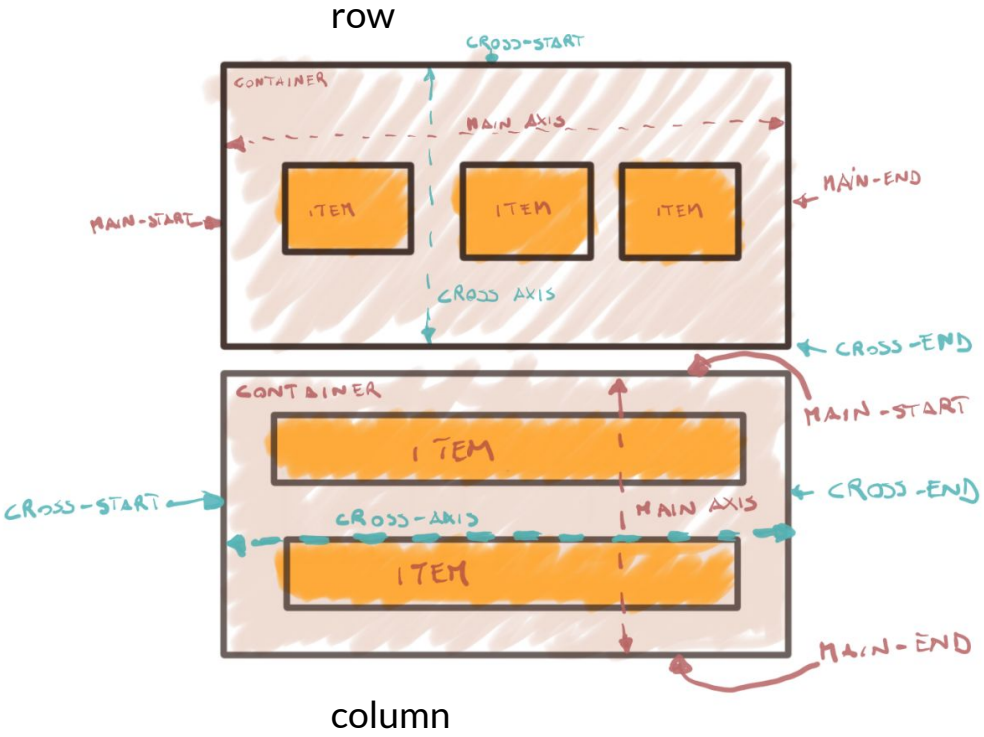
Ad un elemento viene applicato il layout FlexBox quando ha la proprietà **display: flex.** L'elemento prende il nome di **flex-container.**

Un elemento con layout FlexBox si sviluppa su due assi che dipendono dalla direzione (row/column):

- main-axis
- cross-axis

Gli elementi contenuti all'interno del **flex-container** prendono il nome di **flex-items.**

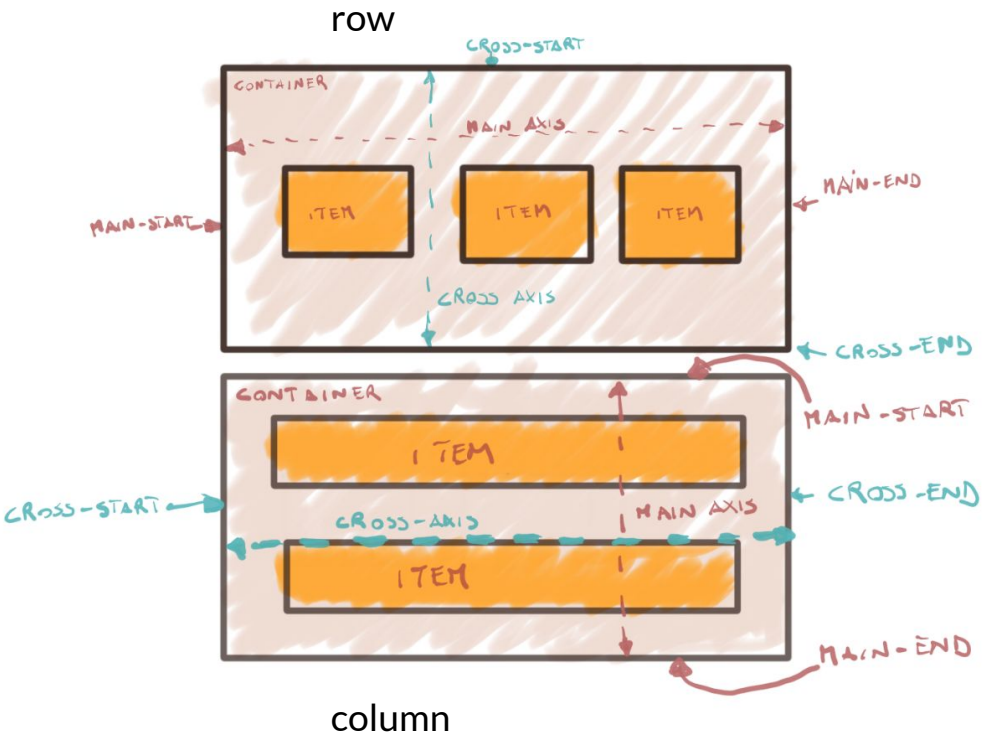
https://developer.mozilla.org/it/docs/Learn/CSS/CSS_layout/Flexbox



CSS - Flex-Container

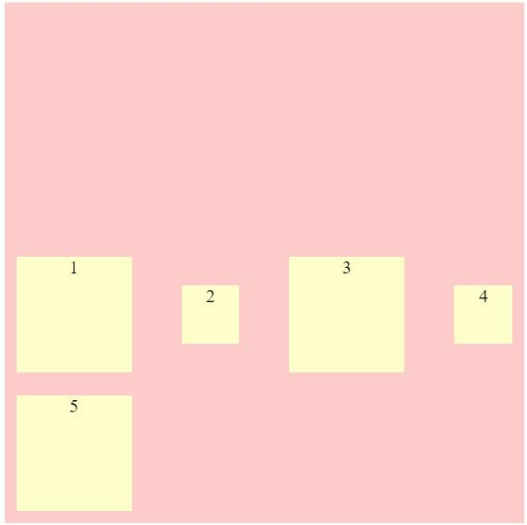
Il flex-container può avere le seguenti proprietà che hanno effetti sul layout flex:

- **display: flex**
- **flex-direction:** Determina se l'asse principale del container è quello orizzontale o quello orizzontale. In base a questa proprietà i flex-items saranno organizzati in righe o in colonne.
- **flex-wrap:** Indica se gli elementi all'interno del container possono andare a capo(o su una colonna separata) in mancanza di spazio nel container.
- **justify-content:** Allinea i flex-items lungo l'asse principale.
- **align-items:** Allinea gli elementi lungo l'asse secondario
- **align-content:** Allinea le linee (righe o colonne) rispetto al flex-container quando c'è spazio non occupato dai flex-items



Esempio flex-container

```html
<body>
  <div class="flex-container">
    <div>1</div>
    <div class="small">2</div>
    <div >3</div>
    <div class="small">4</div>
    <div>5</div>
  </div>
</body>
```



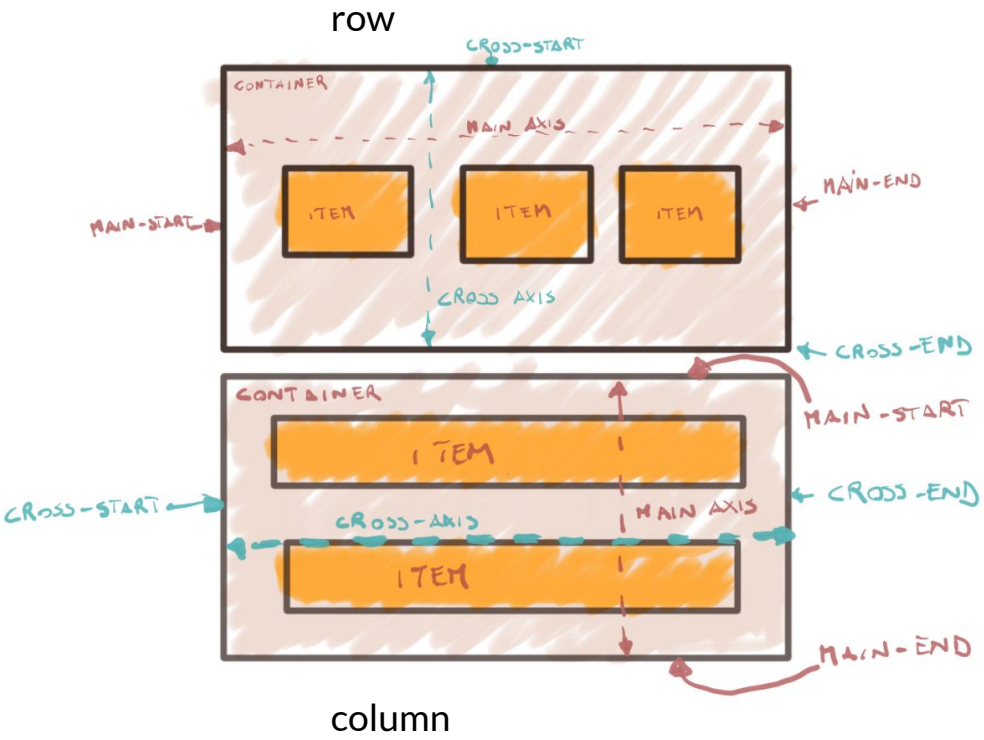

```css
<style>
  .flex-container {
    background-color: #ffcccc;
    width: 450px;
    height:450px;
    display: flex;
    flex-wrap: wrap;
    flex-direction: row; /*provare a cambiare in column*/
    justify-content: space-between;
    align-content: flex-end;
    align-items: center;
  }

  .flex-container div {
    background-color: #ffffcc;
    width: 100px;
    height: 100px;
    text-align: center;
    margin: 10px;
  }
  .flex-container div.small {
    width: 50px;
    height: 50px;
  }
</style>
```

CSS - Flex-Items

I flex-items possono avere le seguenti proprietà che hanno effetti sul layout flex:

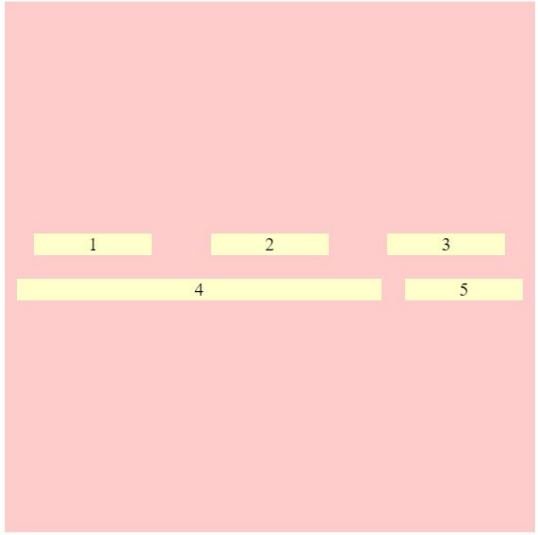
- **order:** Specifica l'ordine dell'item sull'asse principale. Se non specificato si usa l'ordine derivato dall'inserimento nella struttura html.
- **flex-grow:** Numero che specifica quanto può "crescere" l'item rispetto agli altri flex-item
- **flex-shrink:** Numero che specifica quanto può "restringersi" un item rispetto agli altri flex-items
- **flex-basis:** Specifica la larghezza/altezza iniziale dell'item.
- **align-self:** Sovrascrive solo per l'item la regola align-items del flex-container

Nota: Un flex-item può essere a sua volta un flex-container, se ha la proprietà display settata su flex. Si possono creare layout flex annidati!



Esempio flex-items

```html
<body>
    <div class="flex-container">
        <div>1</div>
        <div class="small">2</div>
        <div >3</div>
        <div class="big">4</div>
        <div class="stretch">5</div>
    </div>
</body>
```

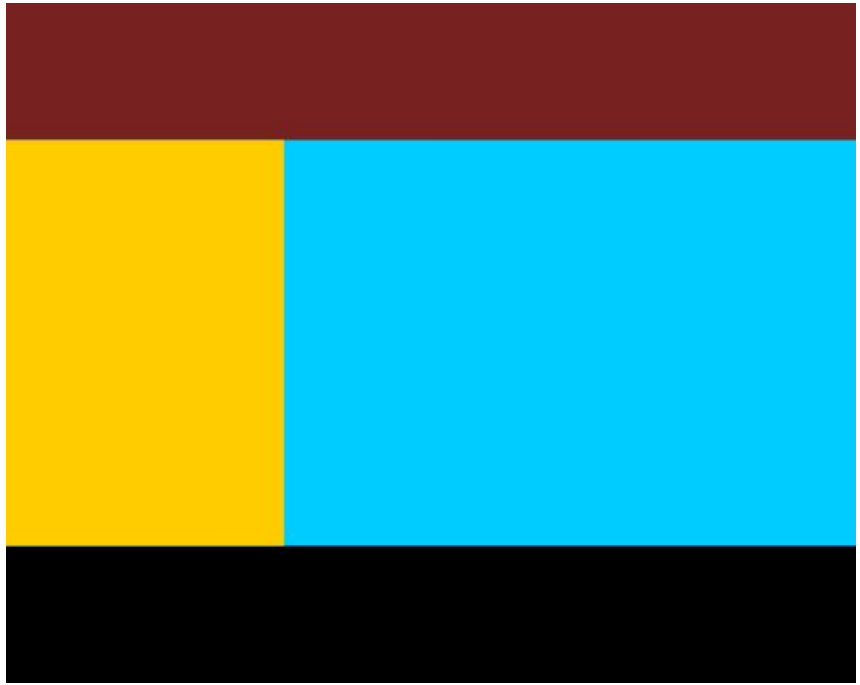



```css
<style>
  .flex-container {
    background-color: #ffcccc;
    width: 450px;
    height:450px;
    display: flex;
    flex-wrap: wrap; /*provare a cambiare in no-wrap*/
    flex-direction: row; /*provare a cambiare in
column*/
    justify-content: space-around;
    align-content: center;
    align-items: center;
  }

  .flex-container div {
    background-color: #ffffcc;
    text-align: center;
    margin: 10px;
    flex-basis: 100px;
    flex-shrink:1; /* provare a mettere 0*/
    flex-grow:0;
  }
  .flex-container div.small {
    flex-shrink: 2;
  }
  .flex-container div.big {
    flex-grow: 2;
  }
  .flex-container div.stretch {
    align-self: stretch;
  }
</style>
```

JS

Esercizio

Creare il layout a fianco usando solo elementi FlexBox. Il layout deve occupare l'intera pagina e deve essere fluido, ovvero deve adattarsi quando la pagina cambia dimensione



Grid-Box layout

E’ un altro tipo di layout CSS che permette di utlizzare delle griglie per la visualizzazione e l’allineamento degli elementi. Per saperne di più fare riferimento a https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout

Materiale della lezione

https://plnkr.co/edit/n0tCAIrYLt2U3BPZ