

Flex

COME FUNZIONA?

-GLI ELEMENTI CHE VOGLIAMO SIANO DISPOSTI IN MANIERA FLESSIBILE, LI RACCHIUDIAMO IN UN FLEX-CONTAINER. MENTRE TUTTI GLI ELEMENTI FIGLI,SARANNO RACCHIUSI IN UN FLEX-ITEM.

AL GENITORE (FLEX-CONTAINER) SI ANDRA AD UTILIZZARE SU CSS IL CODICE: DISPLAY:FLEX;

Es.

CSS Flexbox

Display: flex è una proprietà del genitore!!!

La proprietà **display: flex** va data al tag genitore (**flex-container**).
I suoi figli diventeranno automaticamente **flex-item**.

The diagram shows a window titled "flex container" containing three "flex item" boxes. The code snippets to the right demonstrate the implementation:

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

```
1 .container-flex {
2   display: flex;
3 }
```

NOTA BENE!!!!:

QUANDO ABBIAMO UN FLEX CONTAINER,POTREMMO ANDARE A DARE COMANDI SOLO AI SUI FIGLI DIRETTI (FLEX ITEM) E NON A VARI ALTRI ANTENATI.

Es.

The screenshot shows a code editor with a dark theme. On the left, there is a snippet of CSS and HTML. The CSS defines a class `.container-flex` with `display: flex;`, a background color of `deepskyblue`, and a child class `.item` with a width of `100px`, aspect ratio of `1`, red background color, and `1rem` margin. The HTML contains a `<div>` element with class `container-flex`, which holds three `<div>` elements with class `item`. Each item contains the text "ciao". To the right of the code editor is a preview window showing three red squares on a blue background, with the word "ciao" repeated above each square.

```
<style>
  .container-flex {
    display: flex;
    background-color: deepskyblue;
  }
  .item {
    width: 100px;
    aspect-ratio: 1;
    background-color: red;
    margin: 1rem;
  }
</style>
</head>
<body>

<div class="container-flex">
  <div class="item">
    <div>ciao</div>
    <div>ciao</div>
    <div>ciao</div>
  </div>
  <div class="item"></div>
  <div class="item"></div>
</div>
```

TUTTAVIA, NULLA CI VIETA DI DARE AL NOSTRO FLEX-ITEM, LA PROPRIETA DISPLAY : FLEX; IN MODO DA FARE DIVENTARE A SUA VOLTA I FIGLI DEI FLEX-ITEM.

Es.

The screenshot shows a code editor with a dark theme. On the left, there is a snippet of CSS and HTML. The CSS defines a class `.d-flex` with `display: flex;` and a child class `.item` with a width of `100px`, aspect ratio of `1`, red background color, and `1rem` margin. The HTML contains a `<div>` element with class `container-flex`, which holds three `<div>` elements with class `item d-flex`. Each item contains the text "ciao". The preview window shows three red squares on a blue background, with the word "ciao" repeated above each square.

```
<style>
  .d-flex {
    display: flex;
  }
  .item {
    width: 100px;
    aspect-ratio: 1;
    background-color: red;
    margin: 1rem;
  }
</style>
</head>
<body>

<div class="container-flex">
  <div class="item d-flex"> <!-- Flex container --> <!-- Flex item --> ...
  </div>
  <div class="item"></div> <!-- | -->
  <div class="item"></div>
</div>
```

I FLEX-ITEM POSSONO ESSERE MESSI ORIZZONTALMENTE O VERTICALMENTE, POSSO SCEGLIERE DOVE METTERE I MIEI CONTENUTI (inizio, centro, alla fine...) E POSSONO AVERE DIMENSIONI FLESSIBILI PER ADATTARSI ALLO SPAZIO DISPONIBILE .

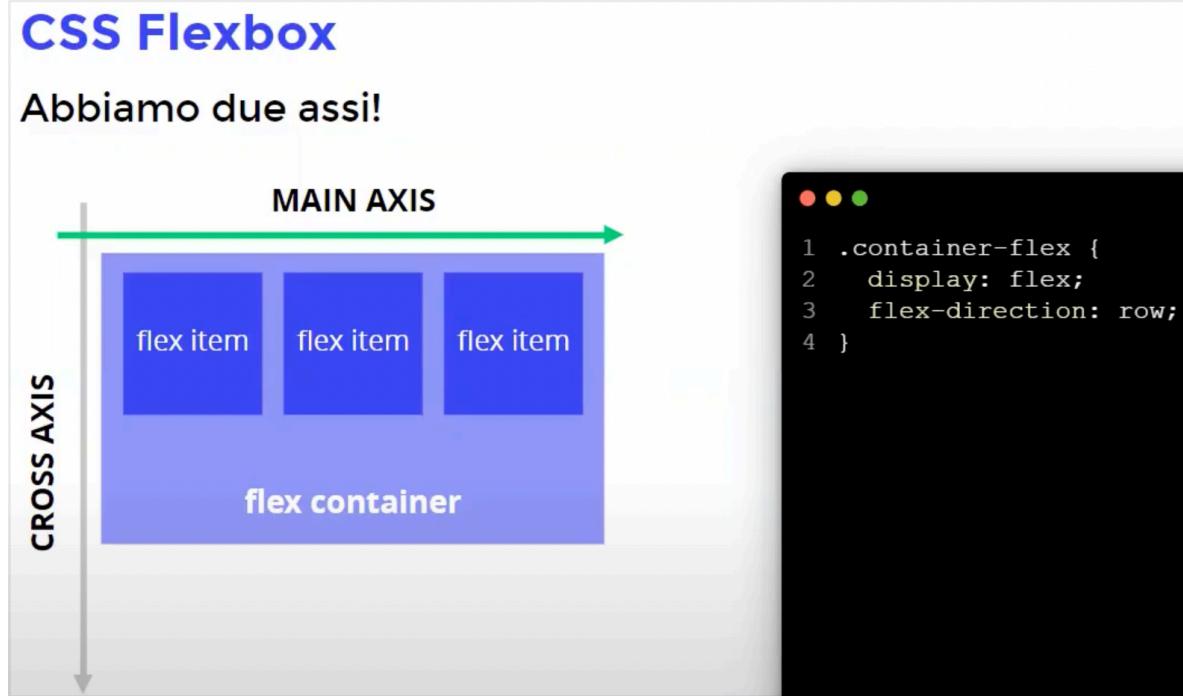
IL CONTENITORE FLEX, HA UN ASSE PRINCIPALE CHIAMATO "MAIN AXIS" CHE E DOVE VENGONO DISPOSTI I NOSTRI FLEX ITEM.
ESISTE ANCHE UN ASSE PERPENDICOLARE, CHIAMATA CROSS AXIS.

LE DIMENSIONI DEI FLEX ITEM, SONO RAPPRESENTATE DA "MAIN-SIZE" E "CROSS-SIZE", CHE SEGUONO MAIN AXIS E CROSS AXIS.

IL MAIN AXIS, COMBACIA CON L'ASSE X, MENTRE IL CROS AXIS, COMBACIA CON L'ASSE Y.

CON "FLEX-DIRECTION : ROW; ANDREMO A DISPORRE L'ASSE SULLA X, IN MODO DA METTERE TUTTI I NOSTRI ITEM AFFIANCO.
(FLEX-DIRECTION:ROW) E LA IMPOSTAZIONE DI DEFAULT.

Es.



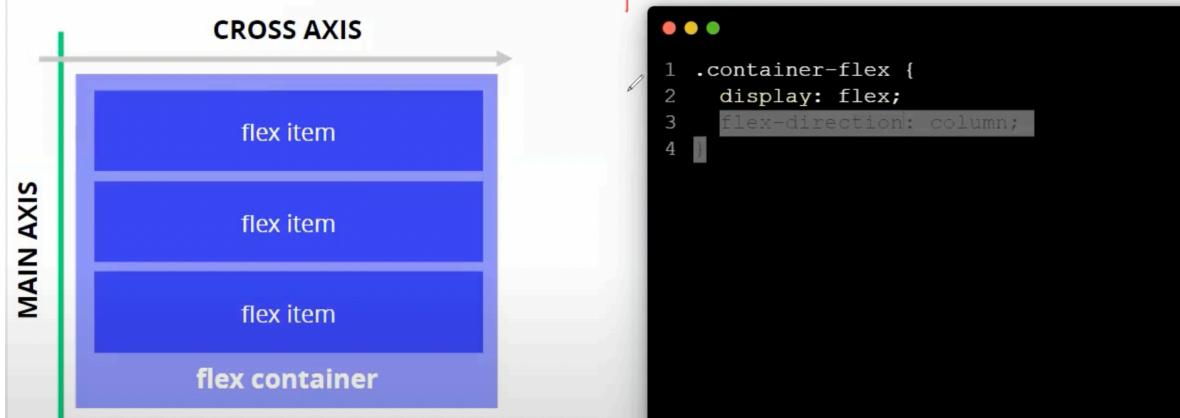
SE INVECE DIAMO "FLEX-DIRECTION : ROW; ANDREMO A RUOTARE L'ASSE COME SE FOSSE UN CUBO.

Es.

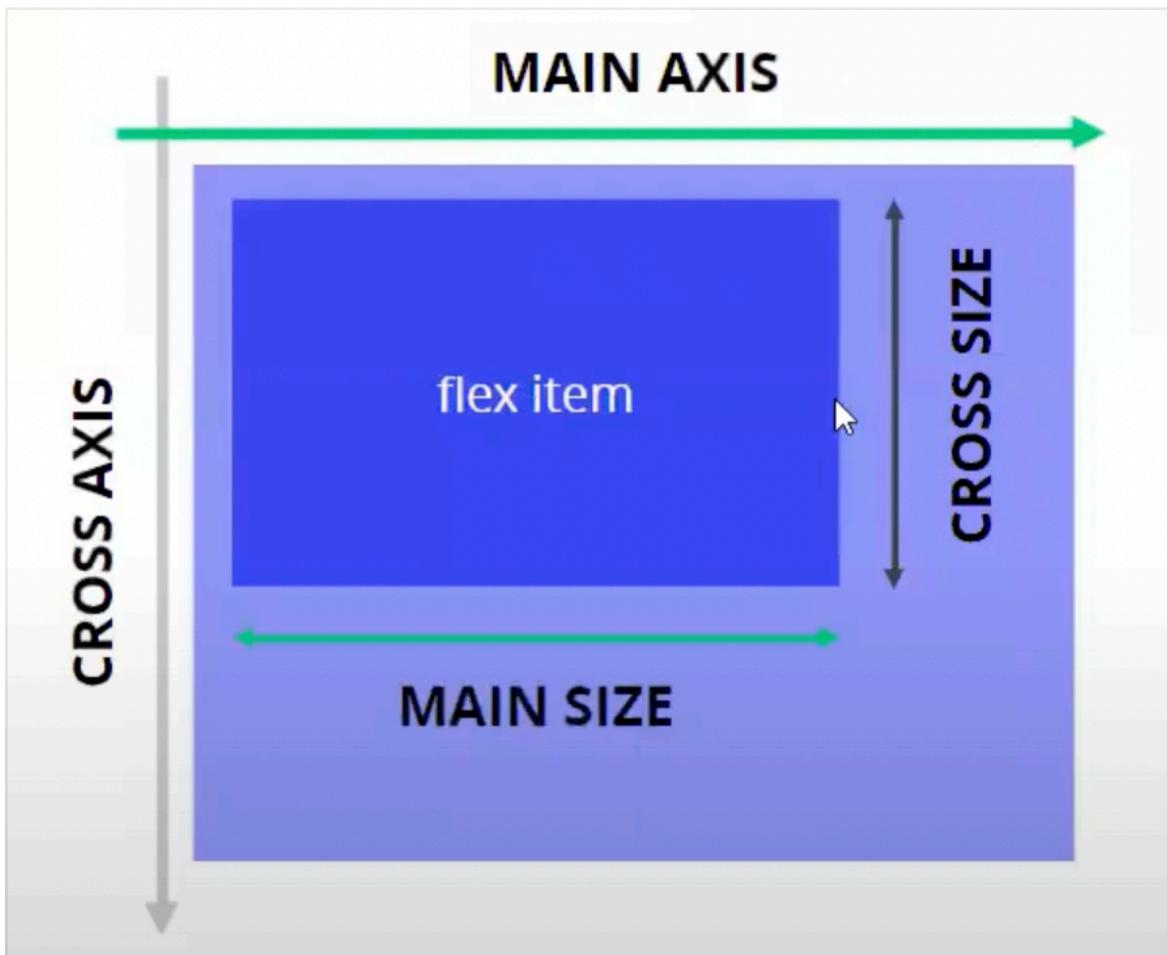
R_{2w}

CSS Flexbox

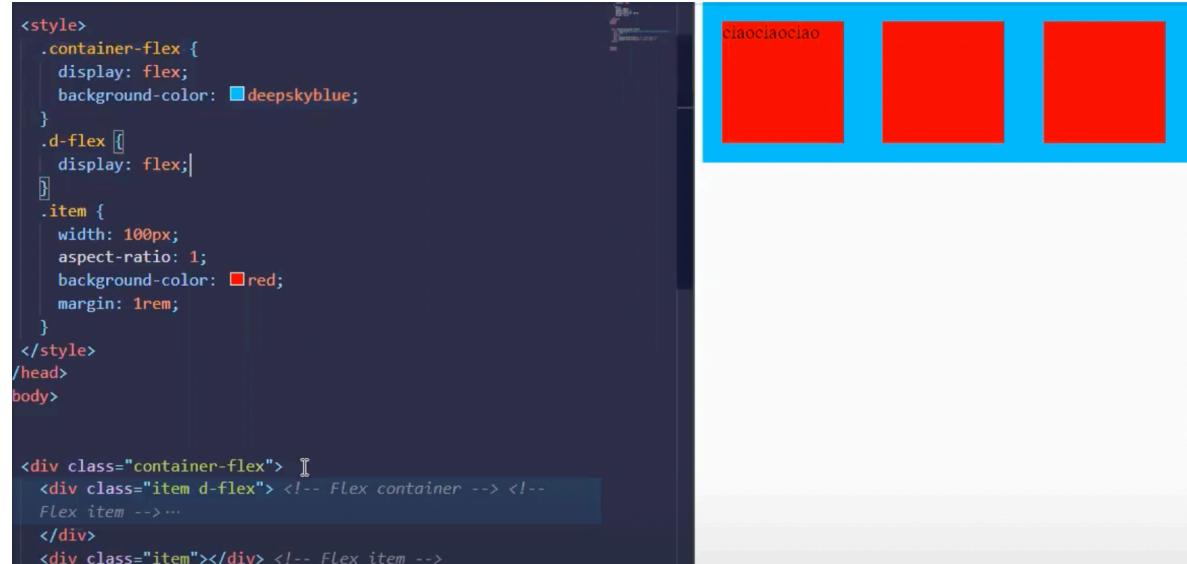
Abbiamo due assi!



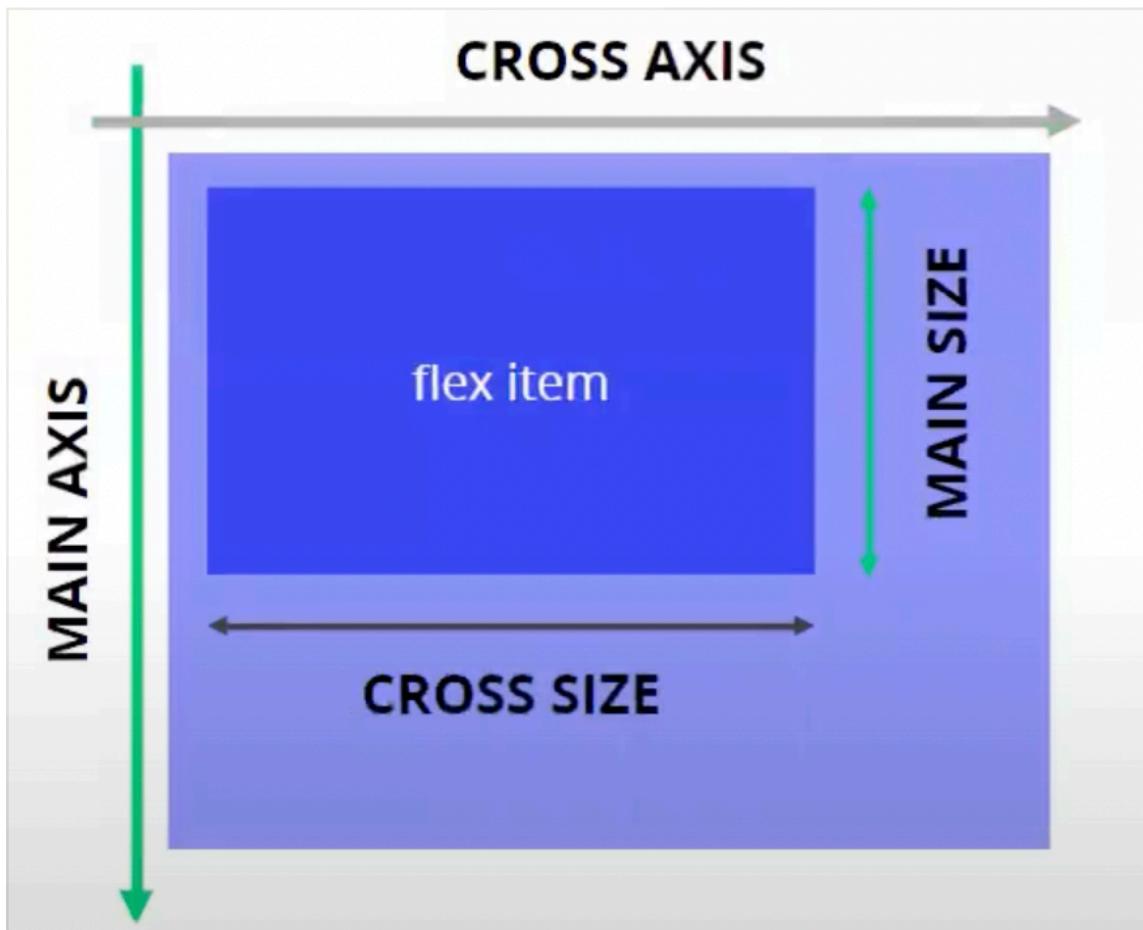
FLEX-DIRECTION:ROW;



```
<style>
  .container-flex {
    display: flex;
    background-color: #deepskyblue;
  }
  .d-flex {
    display: flex;
  }
  .item {
    width: 100px;
    aspect-ratio: 1;
    background-color: red;
    margin: 1rem;
  }
</style>
</head>
<body>

<div class="container-flex">
  <div class="item d-flex"> <!-- Flex container --&gt; &lt;!--
  Flex item --&gt; ...
  &lt;/div&gt;
  &lt;div class="item"&gt;&lt;/div&gt; &lt;!-- Flex item --&gt;
&lt;/div&gt;</pre>
```

FLEX-DIRECTION:COLUMN;



The screenshot shows a code editor on the left and a browser preview on the right. The code defines a class `.container-flex` with `flex-direction: column;`, a class `.d-flex` with `display: flex;`, and a class `.item` with `width: 100px;`, `aspect-ratio: 1;`, `background-color: red;`, and `margin: 1rem;`. The browser preview shows three red squares stacked vertically, with the text "ciaociaociao" above them.

```
<style>
  .container-flex {
    display: flex;
    background-color: deepskyblue;
    flex-direction: column;
  }
  .d-flex {
    display: flex;
  }
  .item {
    width: 100px;
    aspect-ratio: 1;
    background-color: red;
    margin: 1rem;
  }
</style>
</head>
<body>

  <div class="container-flex">
    <div class="item d-flex"> <!-- Flex container --> <!-- Flex item --> ...
  </div>

```

CON ROW-REVERSE,CAMBIEREMO IL PUNTO DI PARTENZA DEL NOSTRO ASSE.

Es.

(I quadrati sono 3) QUI SI E INVERTITO ANCHE L ORDINE NUMERICO DEI QUADRATI, PERCHE IL NUMERO 1 ANZI CHE PARTIRE DA IN ALTO A SX, E PARTITO DA IN ALTO A DX.

The screenshot shows a code editor on the left and a browser preview on the right. The code defines a class `.container-flex` with `flex-direction: row-reverse;`. The browser preview shows two red squares positioned at the top right of the blue container, with the number "2" above the top one and "1" above the bottom one.

```
initial-scale=1.0">
<title>Flexbox</title>
<style>
  .container-flex {
    display: flex;
    background-color: deepskyblue;
    flex-direction: row-reverse;
  }

```

CON COLUMN-REVERSE, OTTERREMO LA STESSA COSA MA AL CONTRARIO.

Es.

```

<style>
  .container-flex {
    display: flex;
    background-color: #deepskyblue;
    flex-direction: column-reverse;
    height: 600px;
  }

  .d-flex {
    display: flex;
  }

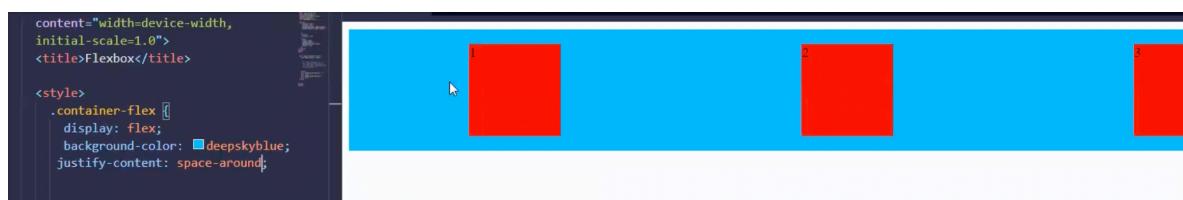
  .item {
    width: 100px;
    aspect-ratio: 1;
    background-color: #red;
    margin: 1rem;
  }
</style>
</head>
<body>

<div class="container-flex">
  <div class="item d-flex">
    1
    <!-- Flex container --> <!-- Flex item <div>ciao</div>-->
    2
    <!-- Flex item -->
    3
    <!-- Flex item -->
  </div>
</div>

```

CON LA PROPRIETA JUSTIFY-CONTENT: ANDREMO AD ORGANIZZARE I NOSTRI CONTENUTI, METTENDOLI COSÌ AL CENTRO, ALLA FINE... QUESTA PROPRIETA PUO ESSERE APPLICATA SOLO AL FLEX CONTAINER ED ANDRA A MODIFICARE IL MAIN AXIS.

Es. (Justify-content:space-around;)

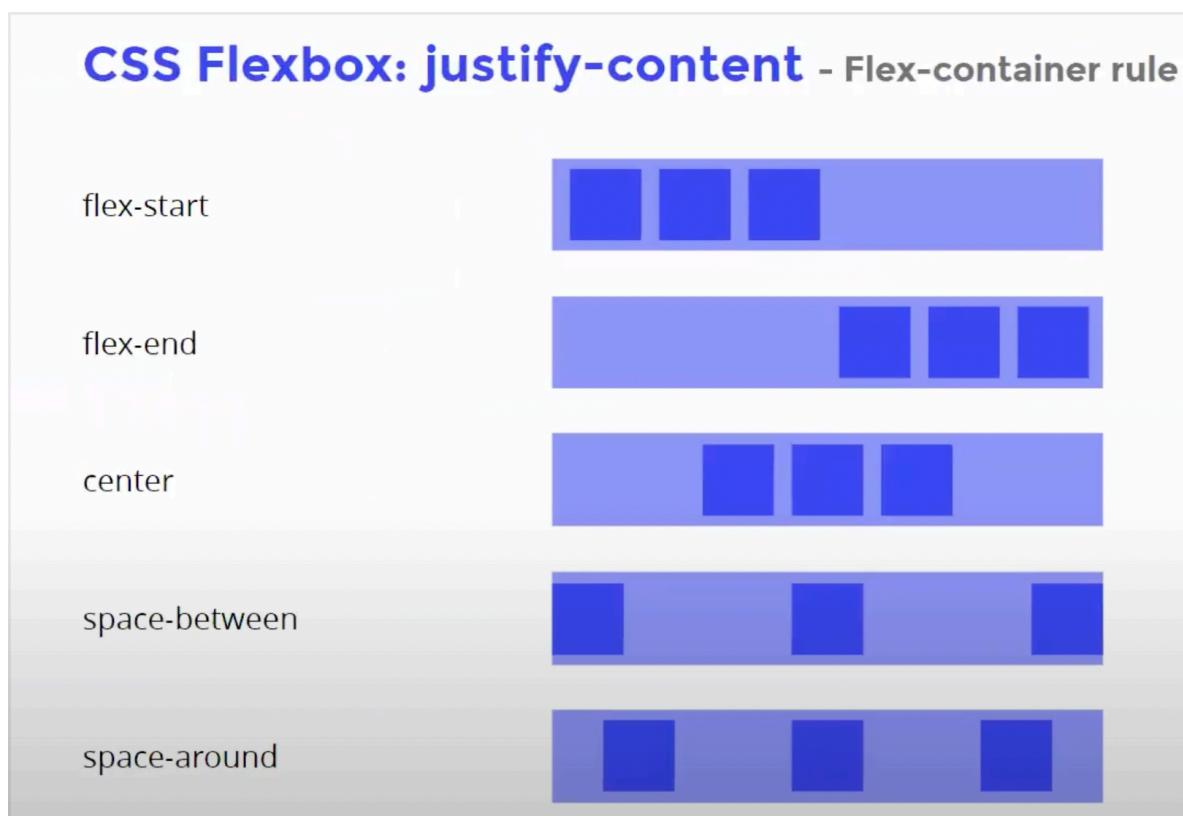


JUSTIFY-CONTENT PUO AVERE LE SEGUENTI PROPRIETA:

- FLEX-START
- FLEX-END

- CENTER
- SPACE BETWEEN: mantiene lo stesso spazio tra un elemento e l altro
- SPACE-AROUND: crea una spaziatura omogenea sulla sx degli elementi
- SPACE-EVENLY: mantiene lo spazio in modo omogeneo tra gli elementi

Es.



CON LA PROPRIETA ALIGN-ITEMS POTREMMO SCEGLIERE COME SI MUOVERANNO GLI ELEMENTI LUNGO IL CROSS AXIS E ANCHE QUESTA E UNA PROPRIETA DA APPLICARE AL FLEX CONTAINER.

Es. (Align-items:center;)

```

.container-flex {
  display: flex;
  background-color: #deepskyblue;
  height: 600px;
  align-items: center;
}

.d-flex {
  display: flex;
}
.item {
  width: 100px;
  height: 100px;
  background-color: red;
  margin: 1rem;
}
</style>
</head>
<body>

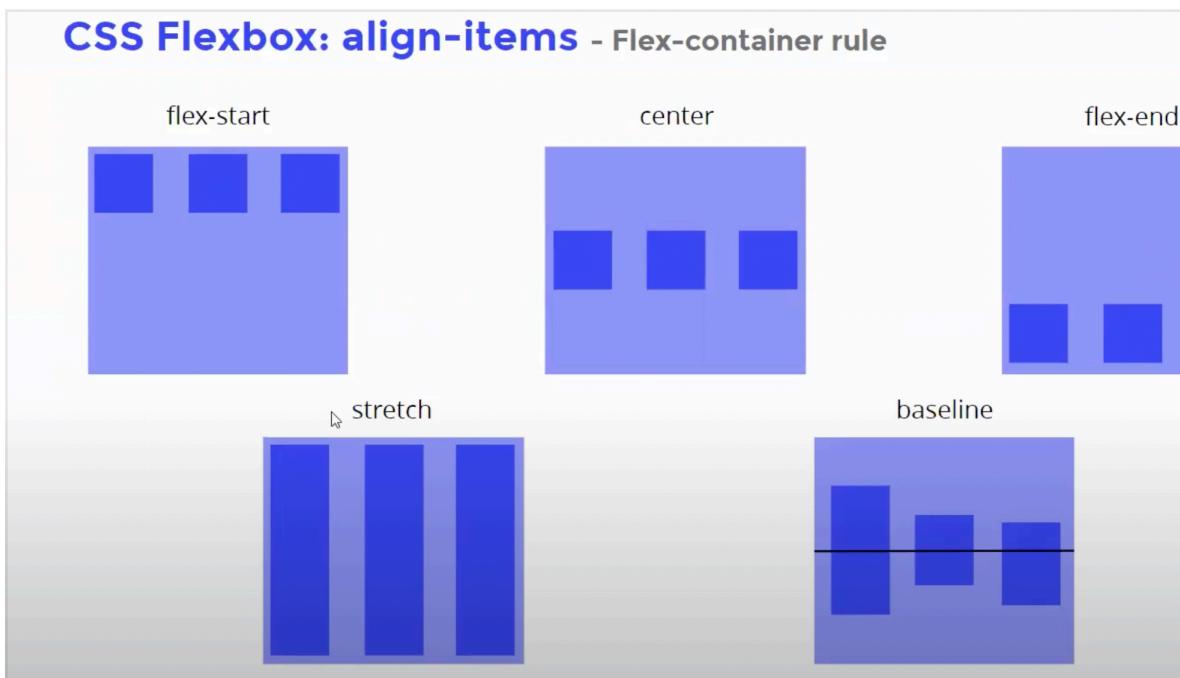
<div class="container-flex">
  <div class="item d-flex">

```

ALIGN-ITEMS PUO AVERE LE SEGUENTI PROPRIETA:

- FLEX-START
- FLEX-END
- CENTER
- STRETCH: allunga l elemento per fare occupare tutto lo spazio a disposizione
- BASELINE: allinea i flex-item sulla base della riga inferiore al testo che abbiamo messo

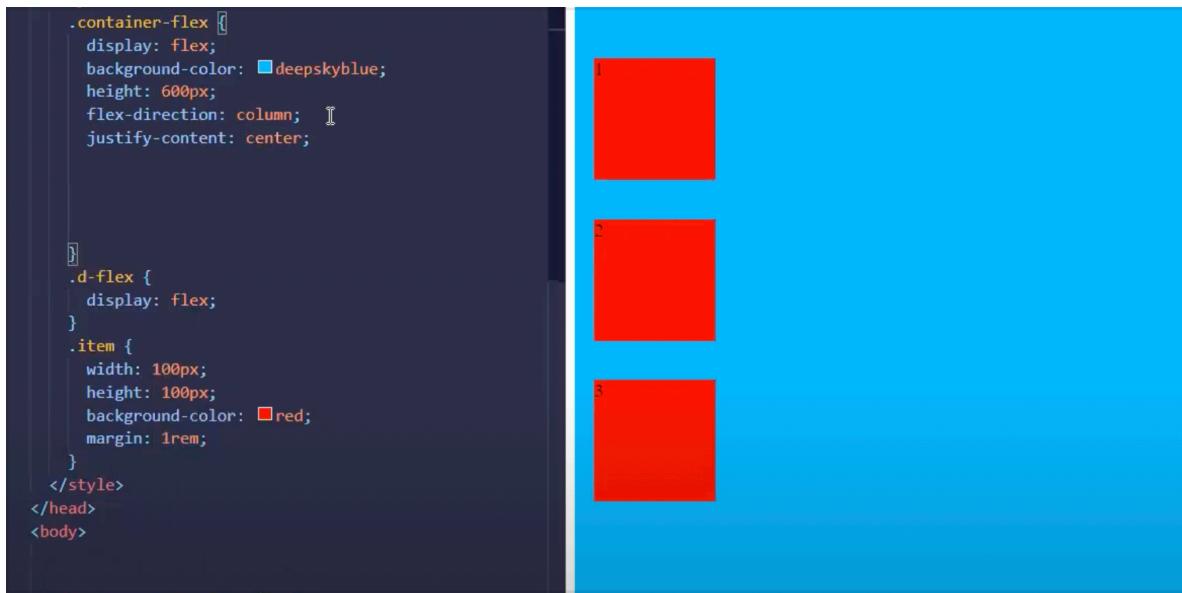
Es.



ATTENZIONE!!!

SE CAMBIO IL VALORE DEI MIEI ASSI (DA ROW A COLUMN), SI SPOSTA TUTTO!!!

Es. (in questo caso abbiamo spostato la FLEX-DIRECTION su COLUMN, facendo così ruotare il MAIN-AXIS e il CROS-AXIS, utilizzando poi JUSTIFY-CONTENT: CENTER; andremo a modificare sempre il MAIN-AXIS, che però essendo ruotato, andrà a portare i quadratini al centro, lasciandoli in colonna.



```
.container-flex {
  display: flex;
  background-color: #deepskyblue;
  height: 600px;
  flex-direction: column;
  justify-content: center;
}

.d-flex {
  display: flex;
}
.item {
  width: 100px;
  height: 100px;
  background-color: #red;
  margin: 1rem;
}
```

CHIARAMENTE LA STESSA COSA VALE ANCHE PER ALIGN-ITEMS.

Es.

```
<style>
  .container-flex {
    display: flex;
    background-color: #deepskyblue;
    height: 600px;
    flex-direction: column;
    justify-content: center;
    align-items: center;
  }

  .d-flex {
    display: flex;
  }

  .item {
    width: 100px;
    height: 100px;
    background-color: #red;
    margin: 1rem;
  }
</style>
</head>
<body>

<div class="container-flex">
```

CON LA PROPRIETA FLEX-WRAP, SI SPECIFICA SE I FLEX ITEMS SI DEBBANO DISPORRE SU UNA UNICA FILA O SE SI DEVONO METTERE SU PIU FILE E A SECONDA DI QUALE VERSO.

E SEMPRE UNA PROPRIETA DA DARE AL FLEX CONTAINER E PUO ASSUMERE I SEGUENTI VALORI:

- WRAP: gli elementi non si strattano come farebbero di default con flex, ma vanno a capo (così da simulare un comportamento responsive)
- WRAP-REVERSE: come wrap, ma cambia l'ordine di wrappatura

Es. FLEX-WRAP:WRAP-REVERSE;

```
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Flexbox</title>

<style>
  .container-flex {
    display: flex;
    background-color: #deepskyblue;
    justify-content: center;
    align-items: center;
    flex-wrap: wrap-reverse;
  }

  .d-flex {
    display: flex;
  }

  .item {
    width: 100px;
    height: 100px;
    background-color: #red;
    margin: 1rem;
  }
</style>
</head>
<body>

<div class="container-flex">
```

CON LA PROPRIETA FLEX-BASIS POTRO IMPOSTARE LA DIMENSIONE DI BASE DI UN FLEX ITEMS, E UNA PROPRIETA DA DARE AL FLEX ITEMS ED E LEGATO AL MAIN-AXIS.

CSS Flexbox

flex-basis - Flex-item rule

Questa proprietà imposta le "dimensioni base" di un Flex-Item e lo gestisce in modo dinamico. Può assumere gli stessi valori accettati da width o height: valori assoluti, valori percentuali ed il valore auto (valore di default).

La differenza sostanziale con width o height è che setta il main-size, quindi è legato al main-axis.

flex-basis insieme a flex-grow e flex-shrink, agiscono sul «Main Size» di un Flex-Item



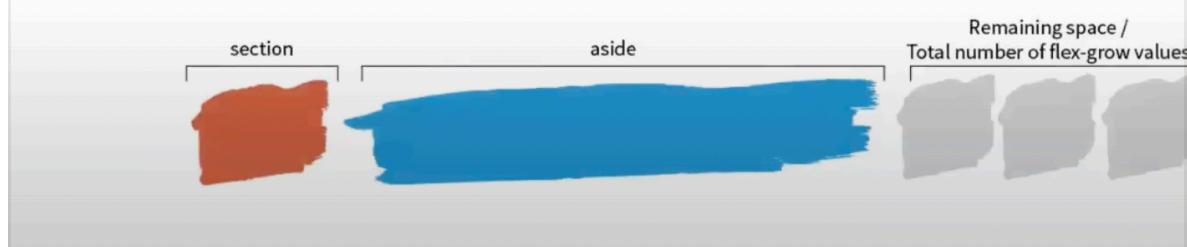
CON LA PROPRIETA FLEX-GROW, ANDREMO A IMPOSTARE IL FATTORE DI CRESCITA DI UN FLEX-ITEM E LO GESTICE IN MODO DINAMICO, E UNA PROPRIETA DA DARE AL FLEX ITEM

CSS Flexbox

flex-grow - Flex-item rule

Questa proprietà imposta il "**fattore di crescita**" di un Flex-Item e lo gestisce in modo dinamico. Può assumere i seguenti valori numerici interi:

- 0 il Flex-Item non può "crescere" (valore di default)
- 1 il Flex-Item può crescere
- >1 il Flex-Item può crescere fino al fattore indicato



CON LA PROPRIETA FLEX-SHRINK ANDREMO A IMPOSTARE IL FATTORE DI RIDUZIONE DI UN FLEX-ITEM E LO GESTIREMO IN MODO DINAMICO, E UNA PROPRIETA DA DARE AL FLEX ITEM

CON LA PROPRIETA ORDE, ANDREMO ANDREMO A IMPOSTARE L'ORDINE IN CUI I FLEX-ITEM VENGONO DISPOSTI IN PAGINA, E UNA PROPRIETA DA DARE AL FLEX ITEM

STRUTTURA FLEX

