

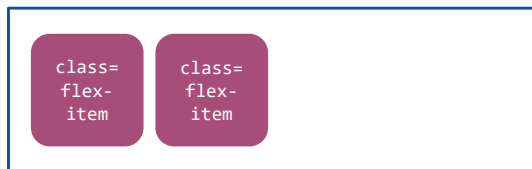
Flexbox basics

Due elementi principali:

- un **Flex container**, che contiene:
- uno o più **Flex item**

Applichiamo proprietà CSS al **flex container** per specificare come devono essere disposti i suoi flex item

id=flex-container



NB: i nomi dell'id e della classe sono arbitrari. Quello che conta sono le proprietà CSS che gli associamo

56

Flexbox basics

Per definire un flex container, impostiamo la proprietà **display**.

Esistono due tipi di container:

- Block container:

`display: flex;`

il prossimo container scorre sotto



- Inline container:

`display: inline-flex;`

il prossimo container scorre a ds



[JsBin](#)

57

Flexbox basics: direzione

In un flex container, i flex item sono disposti su una **direzione**: riga (default) o colonna

- Disposizione in riga:

`display: flex;`



- Disposizione in colonna:

`display: flex;`

`flex-direction: column;`



[JsBin](#)

58

Flexbox basics: justify-content

Possiamo specificare come posizionare gli item nel riquadro del container rispetto alla direzione con la proprietà **justify-content**

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



[JsBin](#)

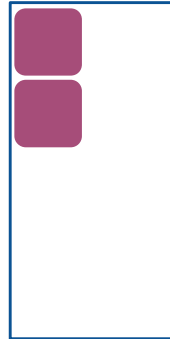
59

Flexbox basics: justify-content

Possiamo specificare come posizionare gli item nel riquadro del container rispetto alla direzione con la proprietà **justify-content**

[JsBin](#)

```
#flex-container {
  display: flex;
  justify-content: flex-start;
  flex-direction: column;
}
```



60

Flexbox basics: justify-content

Possiamo specificare come posizionare gli item nel riquadro del container rispetto alla direzione con la proprietà **justify-content**

[JsBin](#)

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



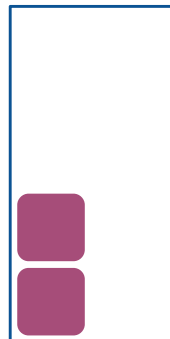
61

Flexbox basics: justify-content

Possiamo specificare come posizionare gli item nel riquadro del container rispetto alla direzione con la proprietà **justify-content**

[JsBin](#)

```
#flex-container {
  display: flex;
  justify-content: flex-end;
  flex-direction: column;
}
```



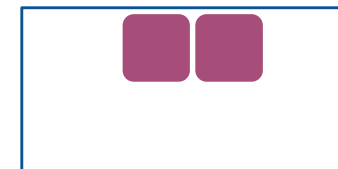
62

Flexbox basics: justify-content

Possiamo impostare come posizionare l'item rispetto alla direzione orizzontale* nel riquadro con **justify-content** nel flex container:

[JsBin](#)

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



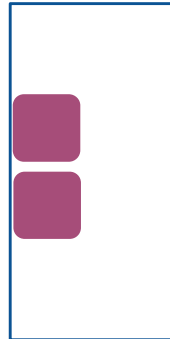
63

Flexbox basics: justify-content

Possiamo specificare come posizionare gli item nel riquadro del container rispetto alla direzione con la proprietà **justify-content**

[JsBin](#)

```
#flex-container {
  display: flex;
  justify-content: flex-center;
  flex-direction: column;
}
```



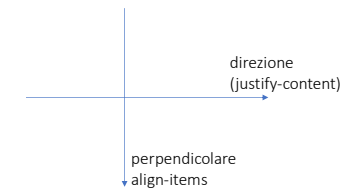
64

Flexbox basics: align-items

Possiamo specificare come posizionare gli item nel riquadro del container **rispetto alla perpendicolare della direzione** con la proprietà **align-items**:

[JsBin](#)

```
#flex-container {
  display: flex;
  align-items: flex-start;
}
```



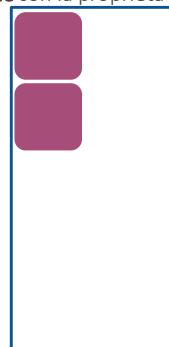
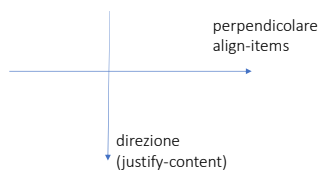
65

Flexbox basics: align-items

Possiamo specificare come posizionare gli item nel riquadro del container **rispetto alla perpendicolare della direzione** con la proprietà **align-items**:

[JsBin](#)

```
#flex-container {
  display: flex;
  align-items: flex-start;
  flex-direction: column;
}
```



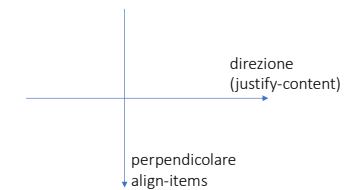
66

Flexbox basics: align-items

Possiamo controllare come posizionare verticalmente l'item nel riquadro con **align-items** nel flex container:

[JsBin](#)

```
#flex-container {
  display: flex;
  align-items: flex-end;
}
```

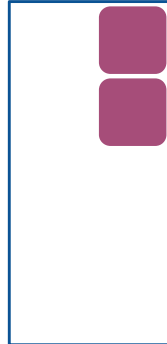
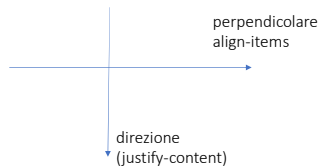


67

Flexbox basics: align-items

Possiamo specificare come posizionare gli item nel riquadro del container **rispetto alla perpendicolare della direzione** con la proprietà **align-items**:

```
#flex-container {
  display: flex;
  align-items: flex-end;
  flex-direction: column;
}
```

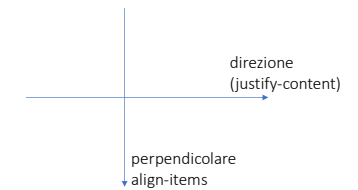
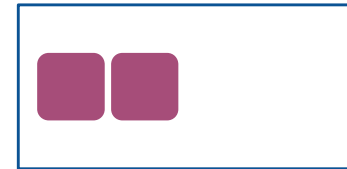


68

Flexbox basics: align-items

Possiamo controllare come posizionare verticalmente (rispetto alla direzione) gli item nel riquadro con **align-items** nel flex container:

```
#flex-container {
  display: flex;
  align-items: center;
}
```

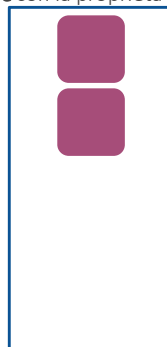
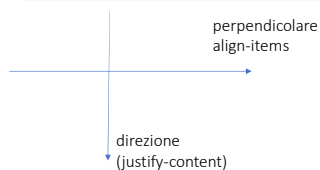
[JsBin](#)


69

Flexbox basics: align-items

Possiamo specificare come posizionare gli item nel riquadro del container **rispetto alla perpendicolare della direzione** con la proprietà **align-items**:

```
#flex-container {
  display: flex;
  align-items: center;
  flex-direction: column;
}
```

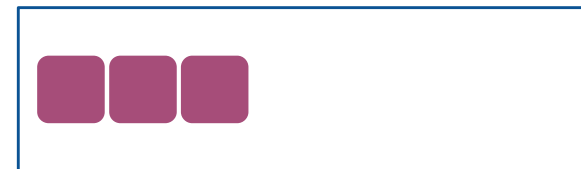
[JsBin](#)


70

Item multipli

Stesse regole se abbiamo più item:

```
#flex-container {
  display: flex;
  justify-content: flex-start;
  align-items: center;
}
```

[JsBin](#)


71

Item multipli

Stesse regole se abbiamo più item:

[JsBin](#)

```
#flex-container {
  display: flex;
  justify-content: flex-end;
  align-items: center;
}
```



72

Item multipli

Stesse regole se abbiamo più item:

[JsBin](#)

```
#flex-container {
  display: flex;
  justify-content: center;
  align-items: center;
}
```



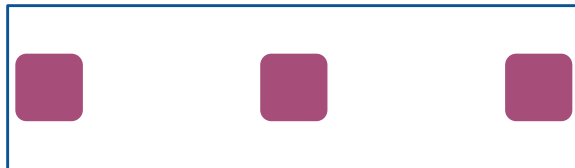
73

Spaziatura di item multipli

space-between e **space-around** permettono di gestire lo spazio tra e intorno agli item all'interno del contenitore:

[JsBin](#)

```
#flex-container {
  display: flex;
  justify-content: space-between;
  align-items: center;
}
```



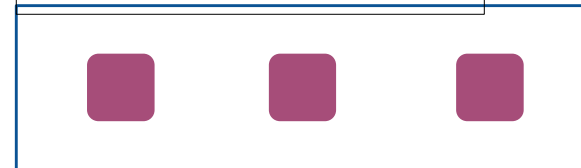
74

Item multipli

space-between e **space-around** permettono di gestire lo spazio tra e intorno agli item all'interno del contenitore:

[JsBin](#)

```
#flex-container {
  display: flex;
  justify-content: space-around;
  align-items: center;
}
```



75

Prima di continuare...

76

Cosa succede se il flex item è un elemento inline?

```

HTML
<html>
<head>
  <meta charset="utf-8">
  <title>Flexbox example</title>
</head>
<body>

  <div id="flex-container">
    <span class="flex-item"></span>
    <span class="flex-item"></span>
    <span class="flex-item"></span>
  </div>

</body>

CSS
#flex-container {
  display: flex;
  border: 2px solid black;
  height: 150px;
}

.flex-item {
  border-radius: 10px;
  background-color: purple;
  height: 50px;
  width: 50px;
  margin: 5px;
}

```

77

???

[JsBin](#)

```

HTML
<html>
<head>
  <meta charset="utf-8">
  <title>Flexbox example</title>
</head>
<body>


  <div id="flex-container">
    <span class="flex-item"></span>
    <span class="flex-item"></span>
    <span class="flex-item"></span>
  </div>

</body>

CSS
#flex-container {
  display: flex;
  border: 2px solid black;
  height: 150px;
}

.flex-item {
  border-radius: 10px;
  background-color: purple;
  height: 50px;
  width: 50px;
  margin: 5px;
}

```



78

Ricordiamo block vs inline

Se #flex-container **non è impostato** display: flex:

[JsBin](#)

```

HTML
<html>
<head>
  <meta charset="utf-8">
  <title>Flexbox example</title>
</head>
<body>

  <div id="flex-container">
    <span class="flex-item"></span>
    <span class="flex-item"></span>
    <span class="flex-item"></span>
  </div>

</body>

CSS
#flex-container {
  border: 2px solid black;
  height: 150px;
}

.flex-item {
  border-radius: 10px;
  background-color: purple;
  height: 50px;
  width: 50px;
  margin: 5px;
}

```



Gli elementi **span.flex-items** non sarebbero visualizzati: siccome sono vuoti, essendo inline, non avrebbero altezza e larghezza

79

Flexbox layout

[JsBin](#)



Perchè cambia se **display: flex**?

Sono ancora elemento inline, eppure hanno altezza e larghezza

80

Flexbox layout

- Quando impostiamo un elemento container **display: flex**, i figli diretti sono **flex items** e seguono un nuovo insieme di regole.
- **Flex items non sono né block né inline**; hanno un insieme di regole diverse per altezza, larghezza, disposizione.
 - I contenuti di un flex item seguono le regole block/inline, relativamente al bordo.
- Le dimensioni (**height** e **width**) dei flex items seguono regole articolate

Vediamo su [JsBin](#)

81

Dimensionare i flex item

Dimensioni flex item

I flex items hanno una larghezza* iniziale, che per default è:

- La larghezza del contenuto
- Il valore impostato esplicitamente con **width**
- Il valore impostato con **flex-basis**

Il valore iniziale della larghezza* di un flex item viene chiamato **flex basis**.

*larghezza nel caso di righe;
altezza nel caso di colonne

82

83

Dimensioni flex item: larghezza iniziale

I flex items hanno una larghezza* iniziale, definita **flex basis**, che vale:

- Larghezza del contenuto
- Il valore impostato esplicitamente con **width**

La larghezza* esplicita di un flex item è rispettata *per tutti i flex items*, siano questi inline, block, o inline-block.

*larghezza nel caso di righe;
altezza nel caso di colonne

84

Dimensioni flex item

Se togliamo **height** e **width**, i nostri flex items spariscono, perchè **flex basis** ora è la dimensione del contenuto, che è vuoto:

[JsBin](#)

```
HTML
<title>Flexbox example</title>
</head>
<body>

  <div id="flex-container">
    <span class="flex-item"></span>
    <div class="flex-item"></div>
    <span class="flex-item"></span>
  </div>

</body>
</html>

CSS
#flex-container {
  display: flex;
  border: 2px solid black;
  height: 150px;
}

.flex-item {
  border-radius: 10px;
  background-color: purple;
  margin: 5px;
}
```

Vediamo su [JsBIN](#)



85

flex-shrink

La larghezza* di un flex item può compattarsi (**riducendola rispetto a flex basis**) per adattare la disposizione degli item alle dimensioni del container attraverso la proprietà **flex-shrink**:

flex-shrink:

- 1, il flex item si compatta per stare dentro il container.
- 0, mantiene le sue dimensioni.

flex-shrink: 1 per default.

*larghezza nel caso di righe;
altezza nel caso di colonne

86

```
#flex-container {
  display: flex;
  align-items: flex-start;
  border: 2px solid black;
  height: 150px;
}

.flex-item {
  width: 500px;
  height: 100px;
  border-radius: 10px;
  background-color: purple;
  margin: 5px;
}
```

[JsBin](#)



Gli elementi si comprimono per aderire alle dimensioni del container.

87


```
#flex-container {
  display: flex;
  align-items: flex-start;
  border: 2px solid black;
  height: 150px;
}

.flex-item {
  width: 500px;
  height: 100px;
  flex-shrink: 0;

  border-radius: 10px;
  background-color: purple;
  margin: 5px;
}
```

[JsBin](#)

Impostando `flex-shrink: 0;`



88

flex-grow esempio

Togliamo height e width dai nostri flex item:

```
HTML
<title>Flexbox example</title>
</head>
<body>
  <div id="flex-container">
    <span class="flex-item"></span>
    <div class="flex-item"></div>
    <span class="flex-item"></span>
  </div>
</body>
</html>

CSS
#flex-container {
  display: flex;
  border: 2px solid black;
  height: 150px;
}

.flex-item {
  border-radius: 10px;
  background-color: purple;
  margin: 5px;
}
```

[JsBin](#)

90

flex-grow

E' il complementare di shrink: attraverso la proprietà `flex-grow`, la larghezza* del flex item può adattarsi alle dimensioni del container **crescendo**:

flex-grow:

- 1, il flex item cresce per occupare tutto lo spazio possibile del contenitore
- 0, non cresce

flex-grow: 0 per default.

*width in the case of rows; height in the case of columns

89

flex-grow esempio

Se impostiamo `flex-grow: 1`, i flex items riempiono lo spazio:

```
HTML
<title>Flexbox example</title>
</head>
<body>
  <div id="flex-container">
    <span class="flex-item"></span>
    <div class="flex-item"></div>
    <span class="flex-item"></span>
  </div>
</body>
</html>

CSS
#flex-container {
  display: flex;
  border: 2px solid black;
  height: 150px;
}

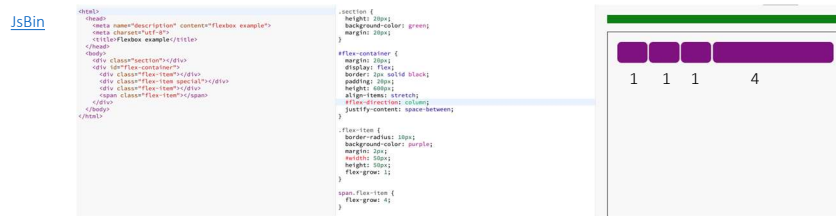
.flex-item {
  border-radius: 10px;
  flex-grow: 1;
  background-color: purple;
  margin: 5px;
}
```

[JsBin](#)

91

flex-grow distribuzione non uniforme

Impostando valori diversi di **flex-grow** è possibile ottenere distribuzioni diverse della larghezza degli elementi



92

Flex item height**?!?

flex-grow controlla solo la larghezza* degli item.

E' possibile agire anche sulla altezza**?



*larghezza nel caso di righe; altezza nel caso di colonne

** altezza nel caso di righe; larghezza nel caso di colonne

93

align-items: stretch;

Il valore di default value di **align-items** è **stretch**, che significa che ogni item cresce verticalmente* per riempire il contenitore.

(Nel caso in cui la proprietà **height** dell'item non sia impostata)

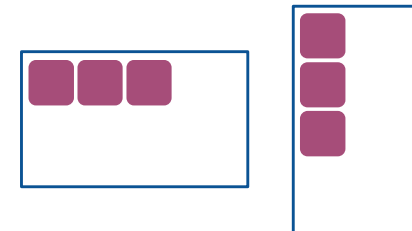


*verticalmente nel caso di righe;
orizzontalmente nel caso di colonne

94

Flex layout recap

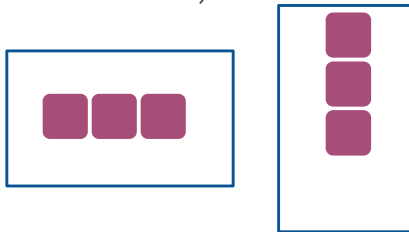
- Con **display: flex**, l'elemento diventa un **flex container** e i suoi figli **flex items**.
- Gli item di un flex si dispongono in una riga o in una colonna, impostando la proprietà **flex-direction** del container.



95

Flex layout recap

- **justify-content** distribuisce gli items orizzontalmente (se **flex-direction: row**), verticalmente (se **flex-direction: column**)
- **align-items** distribuisce gli items verticalmente (se **flex-direction: row**), orizzontalmente (se **flex-direction: column**)



96

Flex layout recap

Con **flex-direction: row**:

- **flex basis** è la larghezza iniziale di un flex item
 - Esplicitamente imposta con **width**, **flex-basis**, o il contenuto dell'elemento
- La larghezza di un item si riduce per far rientrare tutti gli item nel contenitore se **flex-shrink** è impostato a 1
- La larghezza di un item cresce per riempire il contenitore se **flex-grow** è impostato a 1



97

Flex layout recap

Con **flex-direction: column**:

- L'altezza di un flex item è:
 - Quella impostata esplicitamente con **height** sull'item, oppure
 - L'altezza del contenuto dell'item, oppure
 - L'altezza del contenitore se nel contenitore è impostata la proprietà **align-items: stretch**;

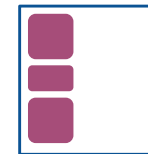


98

Flex layout recap

Per **flex-direction: column**:

- **flex basis** è l'altezza iniziale di un flex item
 - Quella impostata esplicitamente con **height**, **flex-basis**, or l'altezza del contenuto
- L'altezza di una flex item si riduce per riempire il contenitore se **flex-shrink** è impostato a 1
- L'altezza di una flex item cresce per riempire lo spazio del contenitore se **flex-grow** è impostato a 1



99

Flex layout recap

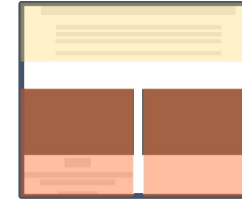
Con `flex-direction: column`:

- La larghezza di un flex item è:
 - Quella impostata con `width` nell'item, oppure
 - La larghezza del contenuto dell'item, oppure
 - La larghezza del contenitore se `align-items: stretch`;



100

Ora possiamo concludere il nostro esercizio!



Follow along on [Codepen](#)

101