ESERCITAZIONE 2

Corso di
CVeDI (8 CFU)

Elia Guarnieri
elia.guarnieri@unimib.it

RIPASSO HTML E CSS

1. Sintassi abbreviata e priorità dei CSS
2. Box Model e flusso
3. Flexbox

SINTASSI ABBREVIATA E PRIORITÀ DEI CSS

Proprietà e sintassi (estesa e abbreviata) - 1

Nelle definizione delle regole è possibile fare uso di **proprietà singole** e **proprietà a sintassi abbreviata**

```
div {
  margin-top: 10px;
  margin-right: 5px;
  margin-bottom: 10px;
  margin-left: 5px;
}
div {
  /* top | right | bottom | left */
  margin: 10px 5px 10px 5px;
}
```

Proprietà e sintassi (estesa e abbreviata) - 2

Sono molte le proprietà che supportano la sintassi abbreviata:

```
p {
  font-style: italic;
  font-weight: bold;
  font-size: .8em;
  line-height: 1.2;
  font-family: Arial, sans-serif;
}
p {
  font: italic bold .8em/1.2 Arial, sans-serif;
}
```

Proprietà e sintassi (estesa e abbreviata) - 3

Sono molte le proprietà che supportano la sintassi abbreviata:

```
div {
  border-width: 1px;
  border-style: solid;
  border-color: #000;
}
div {
  border: 1px solid #000;
}
```

Proprietà e sintassi (estesa e abbreviata) - 4

Sono molte le proprietà che supportano la sintassi abbreviata:

```
div {
  background-color: #000;
  background-image: url(images/bg.gif);
  background-repeat: no-repeat;
  background-position: left top;
}
div {
  background: #000 url(images/bg.gif) no-repeat left top;
}
```

Attenzione!

[CSS Shorthand Syntax Considered an Anti-Pattern]

Gerarchia della specificità

Per capire quali dichiarazioni applicare, il browser segue una logica di **specificità**

Potete immaginare la specificità come un **punteggio**:

- Il selettore universale * (0)
- I nomi dei tag: p, div, .. (1)
- Classi, pseudoclassi e selettori di attributo (10)
- Id (100)
- Stili inline (1000)

```
ul#primary-nav li.active { ... }
```

```
ul.active > p.fake-class { ... }
```

Gerarchia della specificità

Per capire quali dichiarazioni applicare, il browser segue una logica di **specificità**

Potete immaginare la specificità come un **punteggio**:

- Il selettore universale * (0)
- I nomi dei tag: p, div, .. (1)
- Classi, pseudoclassi e selettori di attributo (10)
- Id (100)
- Stili inline (1000)

```
ul#primary-nav li.active { ... }
```

→ 1 + 100 + 1 + 10 = **112**

```
ul.active > p.fake-class { ... }
```

→ 1 + 10 + 1 + 10 = **22**

Nota

- È importante cercare di mantenere la specificità il più bassa possibile evitando, quindi, selettori troppo complessi
- Anche per evitare conflitti, dovete cercare di creare il minor numero possibile di classi
- Le classi sono fatte apposta per essere assegnate a un gran numero di elementi

Cercate sempre di valutare quale è il modo migliore per evitare di scrivere classi ridondanti

ESERCIZIO 1

Nel codice presente a questo link:

- Sovrascrivete il valore di una proprietà di una classe CSS a vostra scelta sfruttando una dichiarazione più specifica (id o inline)
- Utilizzate l'inspector di Chrome per controllare quale regola ha la priorità (tasto destro → esamina)
- Assegnate un nuovo stile inline che vada a sovrascrivere un'altra dichiarazione a vostra scelta. Conferite priorità alla prima dichiarazione utilizzando `!important`, e osservate il risultato

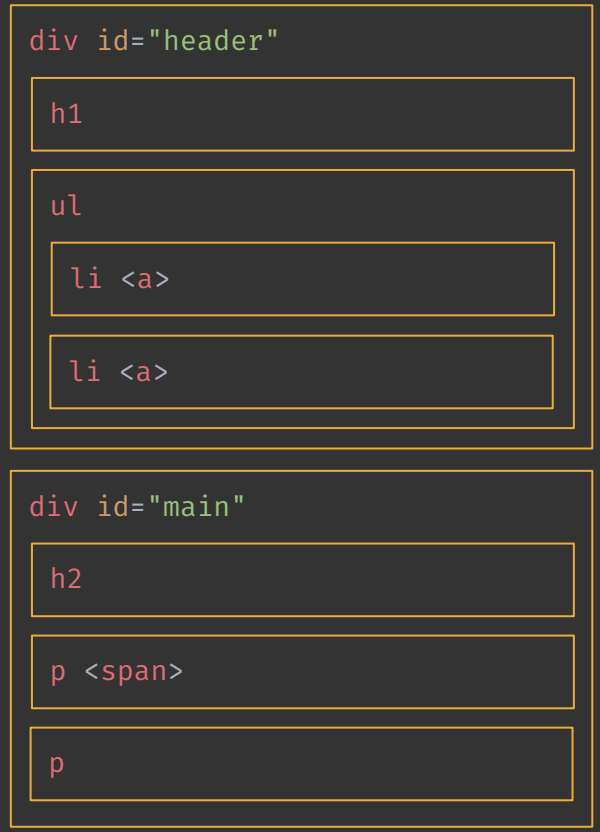
BOX MODEL E FLUSSO

La disposizione degli elementi nell'HTML

Una pagina HTML non è altro che un insieme di elementi (tag) che si comportano come blocchi rettangolari che si dispongono uno sotto l'altro

Il flusso dell'informazione è sempre verticale dall'alto verso il basso

Gli elementi contenuti in altri elementi si disporranno all'interno del contenitore sempre uno sotto l'altro

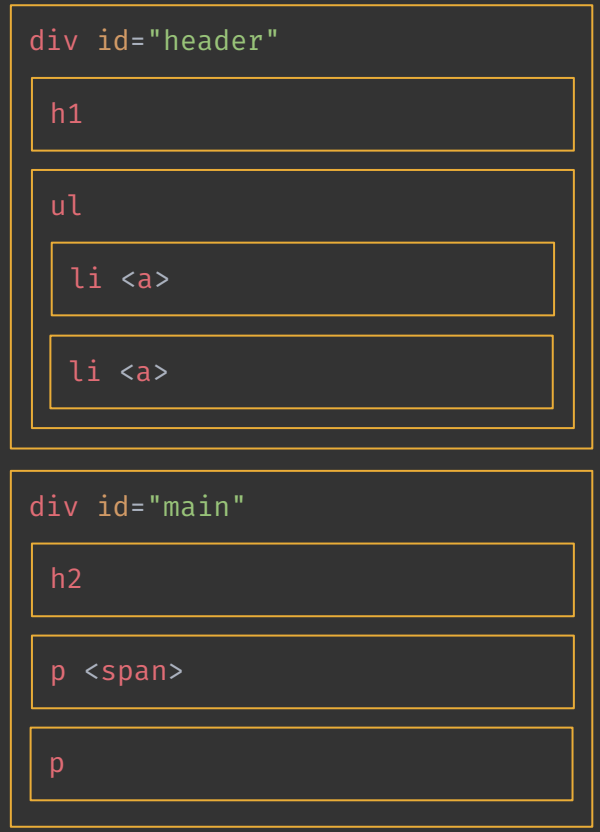


La disposizione degli elementi nell'HTML

Salvo diversamente specificato, ogni elemento di tipo blocco (nel disegno i rettangoli) avrà:

width: 100% del contenitore
height: insieme dei contenuti

Per gli elementi inline (nel disegno tra < >) non si possono specificare width e height e occuperanno la dimensione dell'elemento

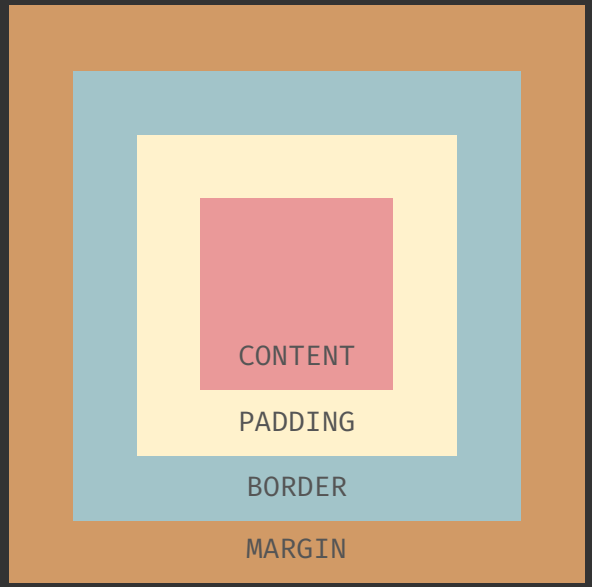


Cambiare la visualizzazione del flusso

BOX MODEL

- **Width** o **height**: per specificare larghezza o altezza del contenuto
- **Padding**: distanzia il contenuto dal suo border
- **Border**: visualizza un bordo attorno al contenuto e lo formatta
- **Margin**: distanzia gli elementi tra di loro e li “sposta” all’interno del flusso

L’ingombro effettivo dell’elemento nella pagina è dato dalla somma delle dimensioni di queste proprietà



Box-sizing

Il **box-sizing** specifica come devono essere calcolati altezza e larghezza totali di un elemento

Di default, nel box model, box-sizing è settato su **content-box**: quando si specifica un certo valore per altezza e larghezza, questo viene applicato al solo contenuto

Se l'elemento ha bordo o padding, l'altezza totale sarà una somma di contenuto, bordo e padding

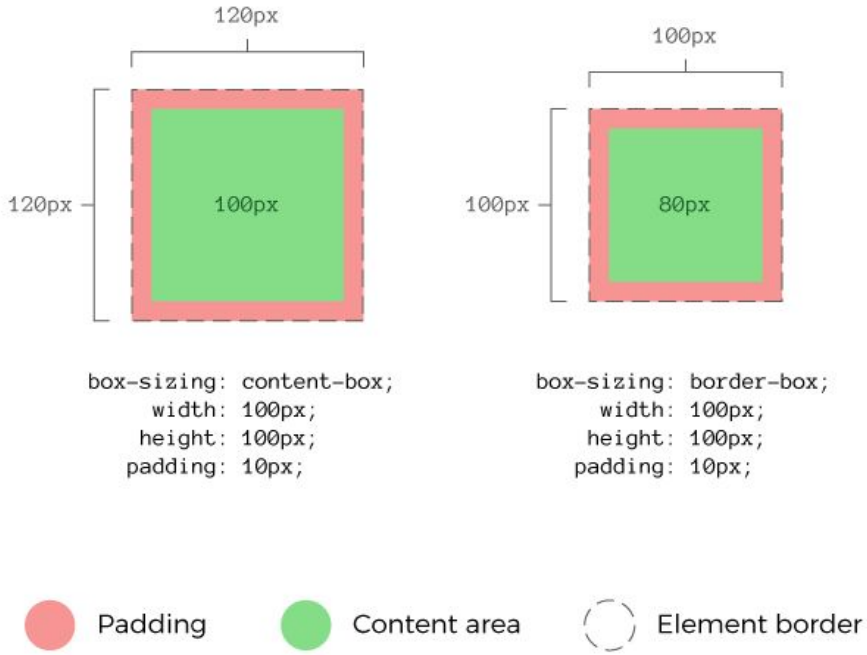
Box-sizing

border-box comunica al browser di includere invece border e padding nel valore specificato: il contenuto si restringerà di conseguenza. Di solito, questo rende più facile controllare la disposizione degli elementi nel flusso

Un esempio:
https://codepen.io/apierotti1/pen/MWvWNOg?editors=1100

CSS Box Sizing

content-box VS border-box

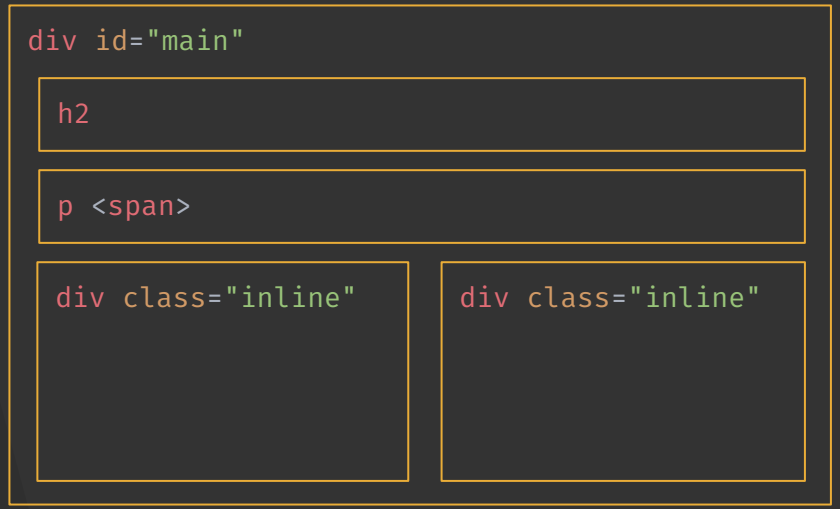


crediti: https://www.webriderz.com/

Cambiare la visualizzazione del flusso

DISPLAY

- **display: none**
  Nasconde gli elementi e li rimuove dal flusso
- **display: inline | block | inline-block**
  Cambia il tipo di comportamento del blocco rispetto al suo normale comportamento



Esempio interattivo

Inline vs inline-block vs block

Comportamento	inline	inline-block	block
Rispetta padding e margin left/right	SI	SI	SI
Rispetta padding top/bottom	SI	SI	SI
Rispetta margin top/bottom	NO	SI	SI
Prende di default la larghezza del proprio contenitore (non del proprio contenuto)	NO	NO	SI
Obbliga a cambiare riga (non permette ad altri elementi di stare accanto)	NO	NO	SI
Rispetta altezza e larghezza quando specificate	NO	SI	SI

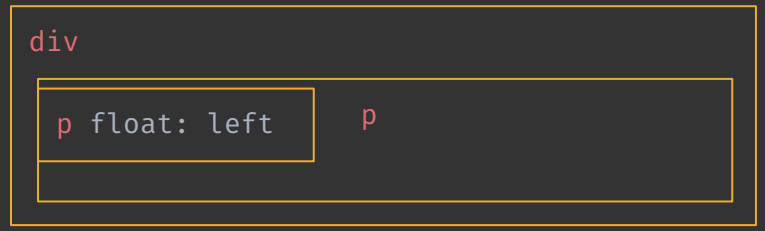
Cambiare il flusso

FLOATING

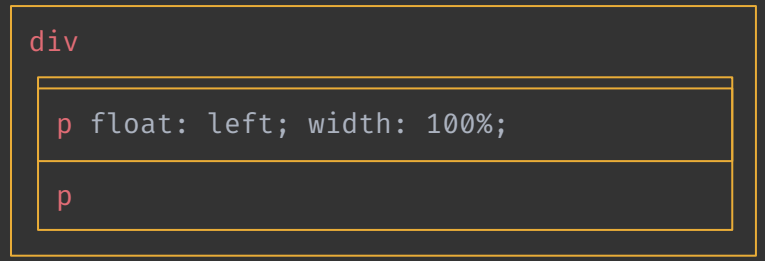
float: left | right - Sposta un elemento a sinistra o destra (1) e lascia scorrere i contenuti successivi attorno

N.B. La sua larghezza diventerà la larghezza del suo contenuto (1), a meno che non sia specificata (2-3)

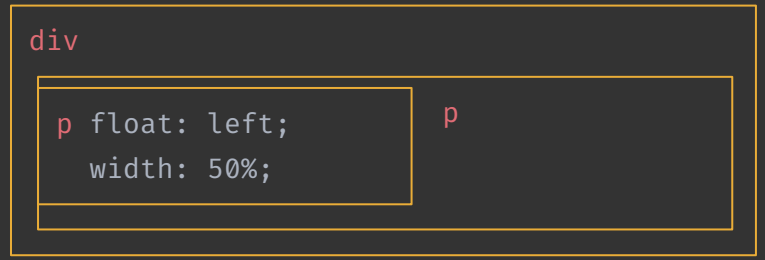
1



2



3



Cambiare il flusso

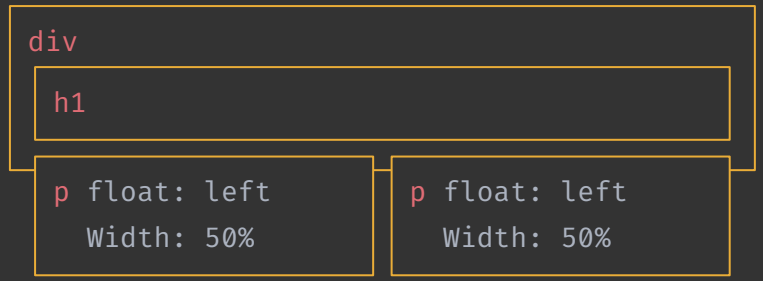
FLOATING

N.B. L'elemento flottante viene rimosso dal flusso, quindi non causa la crescita del genitore! (4)

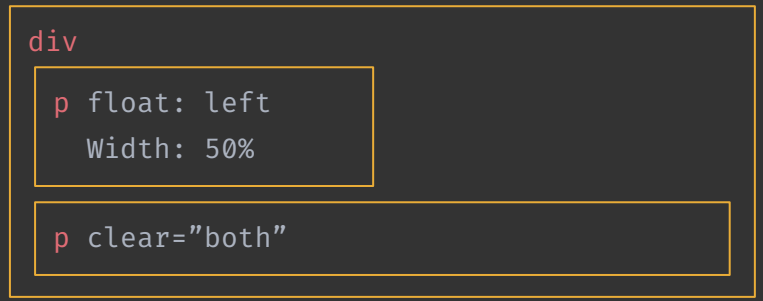
L'elemento flottante costringe i fratelli a posizionarsi accanto a lui, se c'è spazio

clear: left | right | both - Spinge l'elemento sotto a un box floattante (5) (si applica solo agli elementi blocco)

4



5



Cambiare il flusso

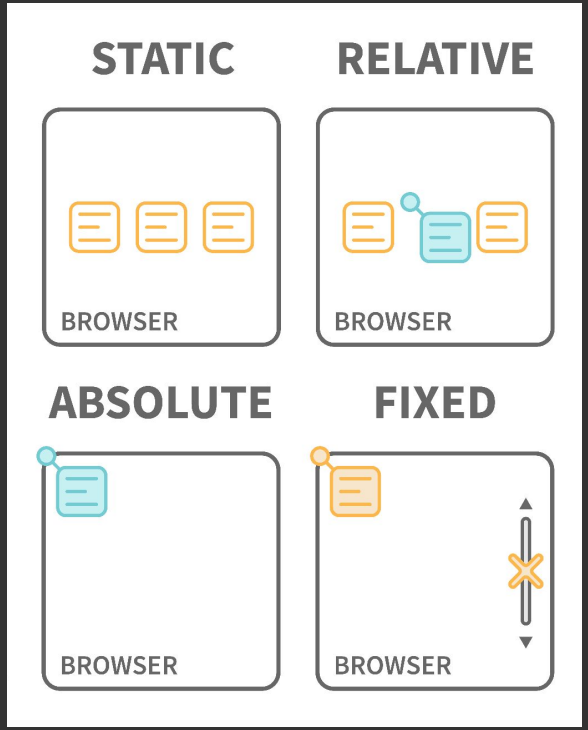
POSITION

position: static - valore di default, l'elemento è posizionato seguendo il flusso del documento

position: relative - l'elemento è posizionato seguendo il flusso del documento e accetta top, right, bottom e left

position: absolute - l'elemento è rimosso dal flusso e la sua posizione è relativa al suo antenato più prossimo con position dichiarata o al body

Position: fixed - l'elemento è rimosso dal flusso e la sua posizione è relativa al body



Esempio interattivo

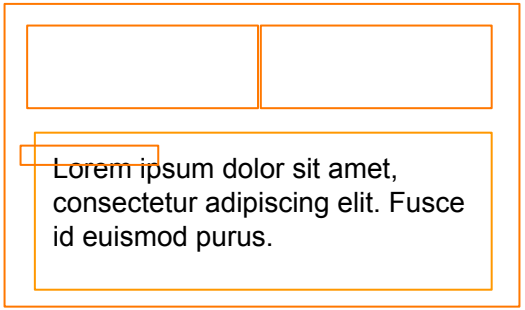
Posizionare gli elementi al centro della pagina

- Per posizionare orizzontalmente al centro un elemento blocco (a cui avete assegnato una larghezza fissa) potete usare **margin: auto**. Lo spazio a destra e sinistra dell'elemento sarà distribuito equamente (Try)
- Per centrare orizzontalmente il testo dentro a un elemento, usate semplicemente **text-align: center**.
- Per centrare verticalmente il testo potete invece usare **padding** (Try)
- In alternativa, si può impostare un valore line-height uguale all'altezza dell'elemento.
- Si può, infine, usare flex (lo vediamo tra poco)

ESERCIZIO 2

- Replicate gli elementi che vedete nella slide successiva
- Non è necessario che il layout sia responsive
- Cercate di sfruttare tutte le caratteristiche del box model viste fino ad ora
- Potete usare flex se lo ritenete più comodo

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce id euismod purus. Proin id egestas turpis, quis semper est. Suspendisse fringilla dui eu egestas dignissim. Suspendisse commodo magna in enim consequat elementum. Aliquam vehicula diam vitae justo porta dapibus. Morbi ligula dui, sodales id mauris ac, congue convallis mauris.



FLEXBOX

Flexbox layout

Flexbox ha l'obiettivo di fornire un modo più efficiente di disporre, allineare, e distribuire lo spazio tra gli oggetti all'interno dei loro contenitori, anche quando le loro dimensioni sono sconosciute e/o dinamiche

Proprietà per il genitore

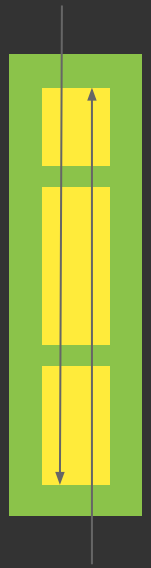
- flex-direction
- flex-wrap
- flex-flow
- justify-content
- align-items
- align-content

Proprietà per il genitore

flex-direction

Stabilisce l’asse principale, ovvero la direzione degli oggetti nel container flex. Gli oggetti si dispongono principalmente su righe orizzontali o colonne verticali

```
.container {
  flex-direction: row | row-reverse |
                  column | column-reverse;
}
```

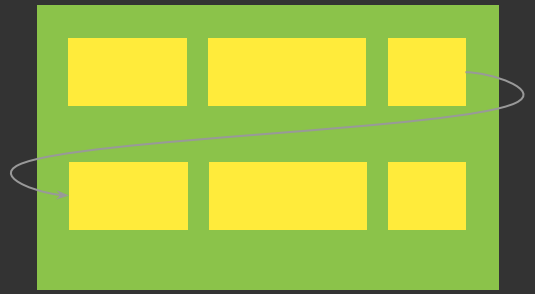


Proprietà per il genitore

flex-wrap

Gli oggetti flex tenteranno di incastrarsi in una sola fila

Si può cambiare questo comportamento con **flex-wrap: wrap**



```
.container {
  flex-wrap: nowrap | wrap | wrap-reverse;
}
```

Proprietà per il genitore

flex-flow

Si tratta di una forma contratta che permette di definire direction e wrap in una sola dichiarazione

```
.container {
  flex-flow: column wrap;
}
```

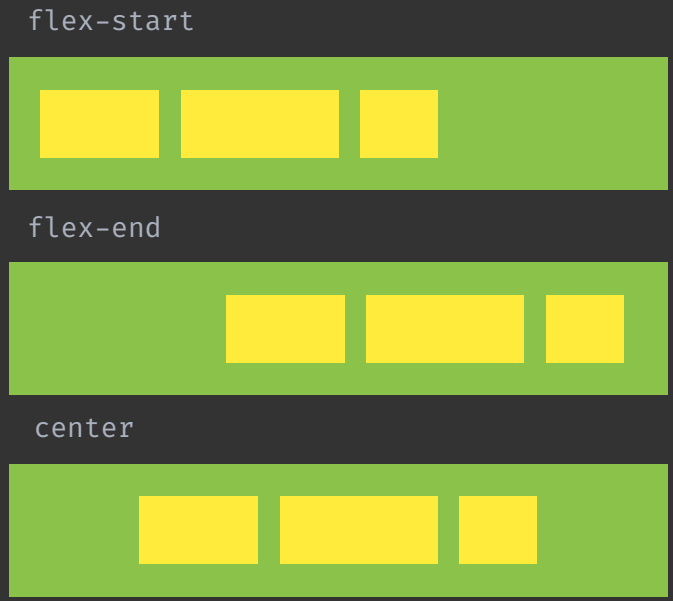
Proprietà per il genitore

justify-content

Definisce l'allineamento lungo l'asse principale (quella orizzontale se la direzione è row). Aiuta a gestire lo spazio libero

Di default è impostato su **flex-start**

```
.container {
  justify-content: flex-start | flex-end |
                   center | space-between |
                   space-around |
                   space-evenly | start |
                   end | left;
}
```



Proprietà per il genitore

justify-content

space-between dispone il primo e l'ultimo elemento all'inizio e alla fine della linea

space-around inserisce un'unità di spazio a destra e a sinistra di ogni elemento (es . il primo avrà 1 unità di spazio di distanza dai margini della linea, ma tra il primo e il secondo ci saranno 2 unità di spazio)

space-evenly assegna lo stesso spazio a destra e sinistra, anche agli elementi accanto ai margini

https://codepen.io/team/css-tricks/pen/zzJMGJ

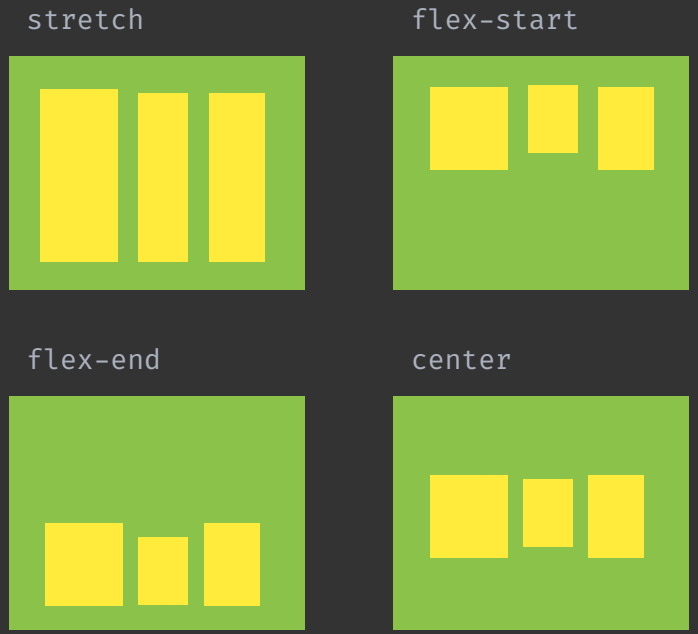
Proprietà per il genitore

align-items

Definisce il modo in cui gli oggetti sono disposti lungo l'asse perpendicolare (si tratta dell'asse verticale se la direzione è row). Di default è impostato su **stretch**

baseline fa in modo che le linee di base del testo siano allineate

```
.container {
  align-items: stretch | flex-start |
               flex-end | center |
               baseline;
}
```



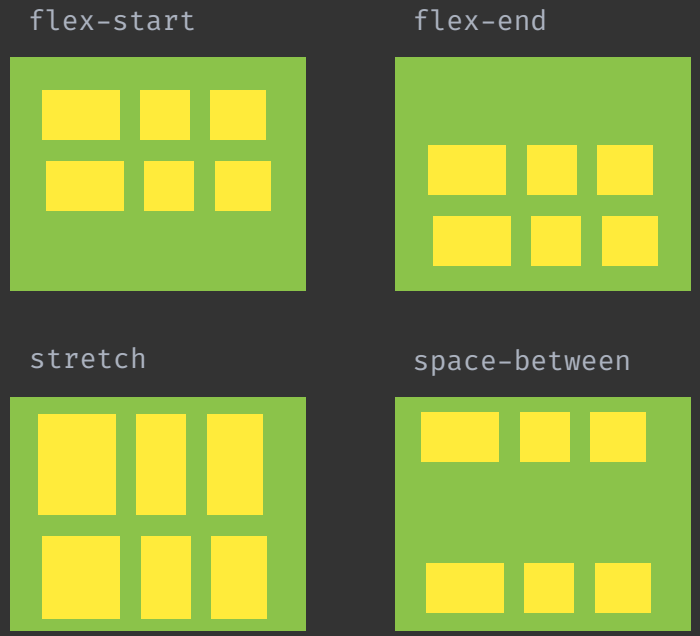
Proprietà per il genitore

align-content

Organizza le righe di elementi sull'asse perpendicolare

Non ha effetto se c'è una sola riga

```
.container {
  align-content: flex-start | flex-end |
                 center | space-between |
                 space-around |
                 space-evenly | stretch |
                 start | end | baseline;
}
```



align-items vs align-content

La differenza si percepisce quando ci sono più righe

- **align-items** dispone gli elementi rispetto alla riga (se viene impostato su flex-end, ad esempio, gli elementi si posizionano attaccandosi al fondo della riga)
- **align-content** gestisce lo spazio tra una riga e le altre

Provate a modificare le proprietà qua:
https://codepen.io/asim-coder/pen/WrMNWR

Proprietà per i figli

- order
- flex-grow
- flex-shrink
- flex-basis
- align-self

Proprietà per i figli

order

Può modificare l'ordine in cui compaiono gli elementi.

Di base, questi si dispongono seguendo l'ordine in cui compaiono nel codice.

Se un elemento ha, ad esempio, **order: 4**, questo tenterà di disporsi per quarto

```
.item {
  order: 4; /* default 0 */
}
```

Proprietà per i figli

flex-grow

Permette ad un oggetto di crescere, se necessario. Accetta un valore che funge da proporzione

Se tutti gli oggetti hanno **flex-grow: 1**, lo spazio sarà distribuito in modo equo. Un oggetto con valore 2 prenderà il doppio dello spazio

```
.item {
  flex-grow: 4; /* default 0 */
}
```

flex-grow: 4

Proprietà per i figli

flex-shrink

Imposta il fattore di restringimento di un elemento

Se la dimensione di tutti gli elementi è maggiore del loro contenitore, gli elementi si restringono per adattarsi secondo il valore impostato di flex-shrink

Se shrink è 0, l'elemento non si restringe

```
.item {
  flex-shrink: 4; /* default 1 */
}
```

flex-shrink: 0

Proprietà per i figli

flex-basis

Definisce la dimensione di default di un elemento prima che lo spazio rimanente del contenitore venga distribuito

Può essere una dimensione (20%, 5rem, etc.) o “auto”. In quest’ultimo caso vengono considerate le proprietà di larghezza e altezza dell’elemento

```
.item {
  flex-basis: 20%; /* default auto */
}
```

Proprietà per i figli

flex-grow, flex-shrink, flex-basis

flex-basis si utilizza molto spesso insieme alle altre due proprietà.
La forma contratta flex vi permette di definire rispettivamente
flex-grow, flex-shrink, flex-basis

```
.item {
  /* sarà minimo 200px, ma crescerà se ne avrà la possibilità */
  flex: 1 0 200px;
  /* sarà 200px al massimo e potrà contrarsi se necessario */
  flex: 0 1 200px;
  /* assumerà la dimensione dei contenuti e potrà crescere se avrà spazio */
  flex: 1 0 auto;
  /*default*/
  flex: 0 1 auto;
}
```

Proprietà per i figli

align-self

Permette di sovrascrivere, per un singolo elemento, la proprietà align-items impostata dal genitore

Ricordatevi che float, clear e vertical-align non avranno effetto su elementi flex

```
.item {
  align-self: auto | flex-start | flex-end | center | baseline | stretch;
}
```

🔗 PER ESERCITARSI E APPROFONDIRE

- Test your skills: Flexbox
- Basic concepts of flexbox
- What Happens When You Create A Flexbox Flex Container?
- A Complete Guide to Flexbox

RIFERIMENTI UTILI

- HTML elements reference
- CSS selectors reference
- CSS selector tester
- FreeCodeCamp
- Google

ESERCITAZIONE 2

FINE