Edgemony

CSS

Web Fundamentals

Cascading Style Sheets

Linguaggio per rappresentare l’aspetto grafico di una pagina web.

Usando il CSS possiamo controllare gli aspetti visivi del contenuto di una pagina web come colori, fonts, dimensioni e molto altro.

`style.css`

Syntax

Il CSS è composto da regole di stile che sono interpretate dal browser e quindi applicate all'elemento corrispondente all'interno di una pagina HTML.

Una regola di stile è composta da tre parti.

```
selector { property: value; }
```

Selector, un tag HTML al quale applicare lo stile. `h1`, `p`, `div`, `class`, `id`

Property, `color`, `border`, `text-align`, ...

Value, assegnato alla property, `#FFFFFF`, `1px`, `center`, ...

Type Selectors

```
h1 {
        color: #000000;
}


p {
        text-align: center;
}
```

Universal Selectors

```
* {
  color: #000000;
}
```

Descendant Selectors

```
p a {
  color: #000000;
}
```

Class Selectors

```
.nomeclasse {
  color: #000000;
}


h1.nomeclasse {
  color: #000000;
}
```

ID Selectors

```
#nomeid {
  color: #000000;
}


h1.#nomeid {
  color: #000000;
}
```

Child Selectors

```
p > span {
  color: #000000;
}
```

Attribute Selectors

```
input[type=“text”] {
  color: #000000;
}
```

Multiple Style Rules

```
h1 {
  color: #000000;
  text-align: center;
  text-transform: uppercase;
}
```

Grouping Selector

```
h1, .nomeclasse, #nomeid {
  color: #000000;
  text-align: center;
  text-transform: uppercase;
}
```

Inclusion

Embedded

```
<head>
        <style  type=”text/css”>
                h1 { color: #000000; }
        </style>
</head>
```

Inline

```
<h1 style=”color: #000000;”>Inline CSS</h1>
```

External

```
<head>
        <link rel=“stylesheet” href=”style.css” />
</head>
```

Units

px
```
height: 100px;
```

%
```
width: 50%;
```

em
```
padding: 2em;
```

rem
```
font-size: 2rem;
```

vw, vh

em e px base 18px

…

Colors

HEX

```
#000000
```

rgb & rgba

```
rgb(255, 255, 255);

rgba(255, 255, 255, 0.5);
```

keyword

```
white, black, violet, …
```

…

Background

`background:` molte proprietà in riga

`background-color`: HEX, rgb, …;

`background-image`: url(“image.jpg”);

`background-repeat`: repeat, no-repeat, repeat-x, repeat-y;

`background-position`: left, top, px, %, em

…

Fonts

`font-family`: Arial, Helvetica, sans-serif;

`font-style`: normal, italic, oblique

`font-size`: px, em, rem, %, …

`font-weight`: normal, bold, 900, …

Text

`text-align`: left, center, right, justify;

`text-decoration`: none, overline, underline, line-through, …

`text-transform`: uppercase, lowercase, capitalize

`letter-spacing`: px, %, em, rem, …

`word-spacing`: px, %, em, rem, …

...

Links

`a`

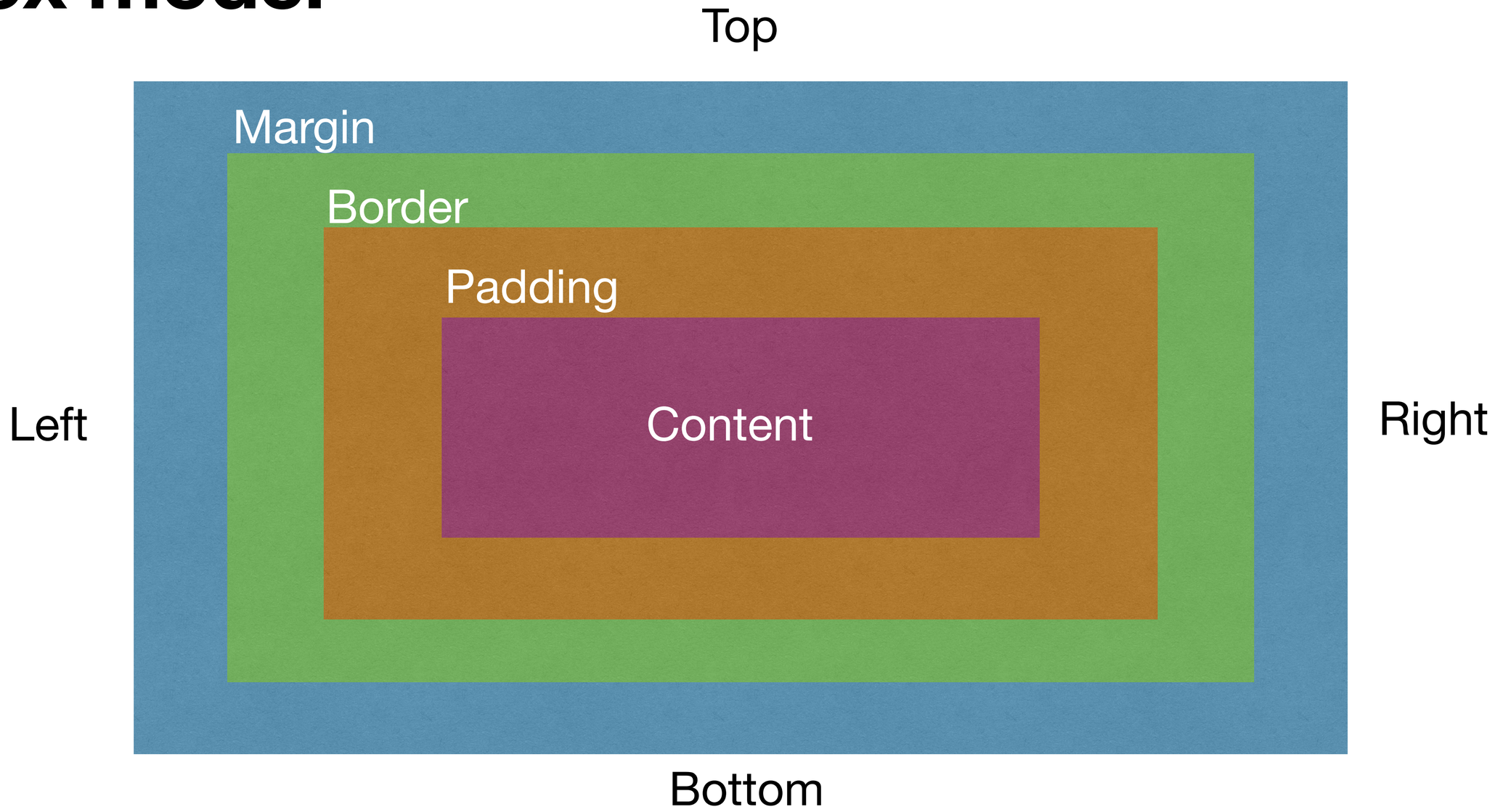
`a:link {color: #000000}` unvisited hyperlinks

`a:visited {color: #FFF000}` visited hyperlinks

`a:hover {color: #333333}` mouse hover on hyperlinks

`a:active {color: #FF00CC}` currently

Box model



Borders

`border`

`border-color`: hex, rgb, …

`border-style`: none, solid, dotted, dashed, double, …

`border-width`: px, em, rem, …

`border-radius`: px, em, rem, …

`border-top-style`
`border-right-color`
…

Margins

Space around html element. px, em, rem, …

`margin: 10px;` top, right, bottom, left 10px

`margin: 10px 5px;` top e bottom 10px, right e left 5px

`margin: 10px 5px 20px;` top 10px, bottom 20px, right e left 5px

`margin: 10px 10px 20px 5px;` top 10px, right 10px, bottom 20px, left 5px

`margin-top, margin-right, margin-bottom, margin-left`

Paddings

Space inside html element. px, em, rem, …

`padding: 10px;` top, right, bottom, left 10px

`padding: 10px 5px;` top e bottom 10px, right e left 5px

`padding: 10px 5px 20px;` top 10px, bottom 20px, right e left 5px

`padding: 10px 10px 20px 5px;` top 10px, right 10px, bottom 20px, left 5px

`padding-top, padding-right, padding-bottom, padding-left`

Dimensions

`width: px, %, rem, em, auto, …`

`height: px, %, rem, em, auto, …`

`min-width, max-width, min-height, max-height`

Position

`position:`

`static`: default, posizione normale.

`relative`: relativamente al suo box contenitore.

`absolute`: posizionato in base ai valori forniti (top, right, bottom, left) e rispetto al suo contenitore relative.

`fixed`: posizione fissa rispetto all'intero documento.

`sticky`: inizialmente in relative, allo scroll del documento fissato in base alla sua posizione (top, right, bottom, left)

Level

`z-index`

definisce l’ordine di posizionamento degli elementi.

Gli elementi con `z-index` maggiore vengono disposti davanti ad elementi con `z-index` minore.

```
z-index: 1;

z-index: 2;

...
```

Display

`display`

Questa proprietà serve per controllare il modo in cui un elemento viene. visualizzato.

`block`: elemento blocco

`inline`: elemento inline

`inline-block`: elemento inline ma possiamo specificare margin, padding …

`none`: non viene mostrato

… `flex`, `grid`, …

Visibility & Opacity

`visibility:` definisce se un elemento deve essere visibile o nascosto.

`visible` valore di default, elemento visibile.

`hidden` elemento nascosto ma mantiene il suo posto e le sue dimensioni.

```
Opacity: 0.5;
```
da 0 a 1

Box sizing

Gestire le dimensioni di un box.

Di default, quando si imposta la larghezza (`width`) di un elemento, le dimensioni si riferiscono al suo contenuto, escludendo eventuali spazi occupati da `padding`, `border`, ...

Per riflettere effettivamente le sue dimensioni all'interno della pagina possiamo usare gli attributi `box-sizing`

`border-box` il valore di width fa riferimento al box nella sua interezza, compresi spazi dei `padding` e `border`

`content-box` il valore di width fa riferimento al suo contenuto, comportamento di default

Lists

```
list-style-type:
none, circle, square, upper-roman, lower-alpha, …

list-style-image: url(“image.png”);

list-style-position:
outside, inside
```

Pseudo class

```
selector:pseudo-class {

  property: value;

}
```

```
:active
:checked
:invalid
:disabled
:focus
:hover
:link
```

```
:first-of-type
:last-of-type
:first-child
:last-child:nth-child(n)
…
…
```

Pseudo element

```
selector::pseudo-element {
 property: value;
}
```

```
::after
::before
::first-line
::first-letter
```

```
::after & ::before
Dichiarare content: “”
p::after {
 content: “”
}
```

Math functions

```
calc()

max()

min()


width: calc(100% - 100px);

width: max(50%, 300px)

width: min(50%, 300px)
```

!important

La regola `!important` è usata per dare “molta importanza” ad una proprietà rispetto al “normale”.

aggiungendo `!important` ad una regola css sovrascriverà le regole precedenti assegnate all’elemento.

```
background-color: #000000 !important;
```

Solo se necessario!

Object-fit

Specifica come un’immagine o un video deve adattarsi al suo contenitore.

```
object-fit:

fill, contain, cover, scale-down, none
```

Transform

```
transform:

translate(), translateX(), translateY()

rotate()

scale(), scaleX(), scaleY()

skew(), skewX(), skewY()

matrix(scaleX(), skewY(), skewX(), translateX(), translateY())
```

Transition

`transition: width 1s 0.25s ease`

`transition-property`

`transition-duration`

`transition-delay`

`transition-timing-function`

- `ease` default effect, slow start, fast, end slowly
- `linear` same speed from start to end
- `ease-in` slow start
- `ease-out` slow end
- `ease-in-out` slow start and end
- `cubic-bezier(n, n, n, n)` custom

Animation

```
animation: esempio 3s ease-in infinite alternate
```

@keyframes

animation-name

animation-duration

animation-delay

animation-iteration-count

animation-diration

animation-timing-function

```
@keyframe esempio {
  from {background-color: red}
  to {background-color: green}
}
@keyframe esempio2 {
  0% {left: 0px; top: 0px;}
  50% {left: 10px; top:20px}
  100% {left: 40px; top: 60px}
}
```

Media query

Una media query è una funzionalità CSS per adattare il layout a diverse dimensioni dello schermo, che sia mobile, tablet, desktop, tv, …

```
@media screen and (max-width: 768px) {

  …

}
```

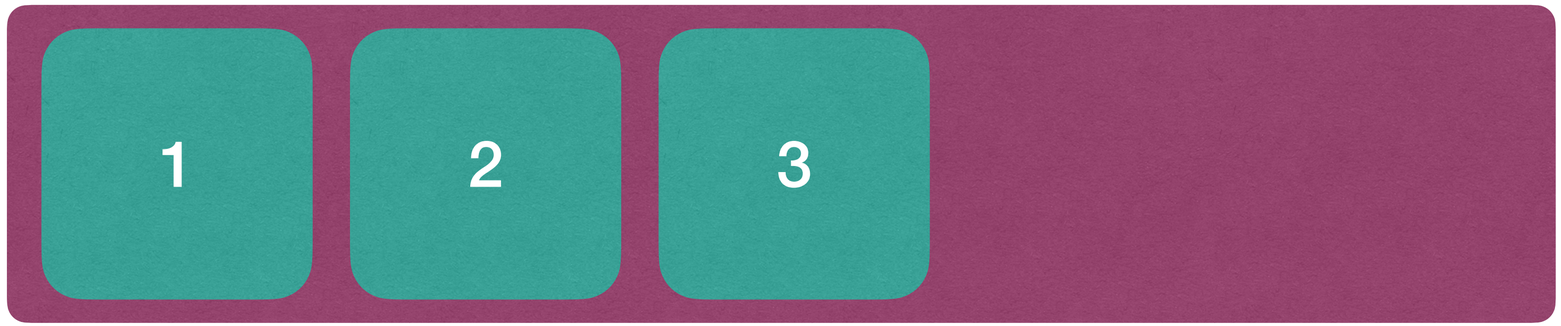
```
@media screen and (min-width: 768px) and (max-width: 1024px) {

  …

}
```

display: flex

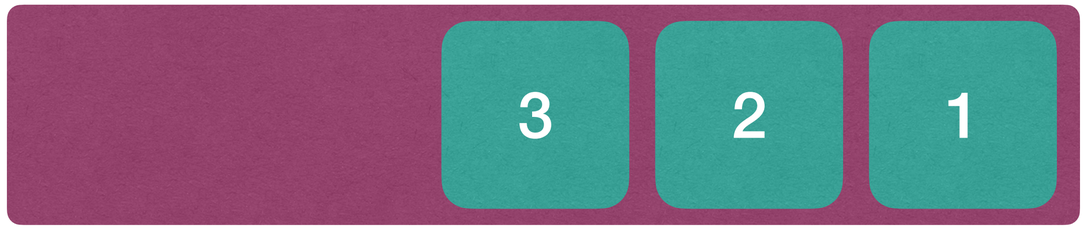
```
.flex-container {
  display: flex;
}
```



flex-direction

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

row



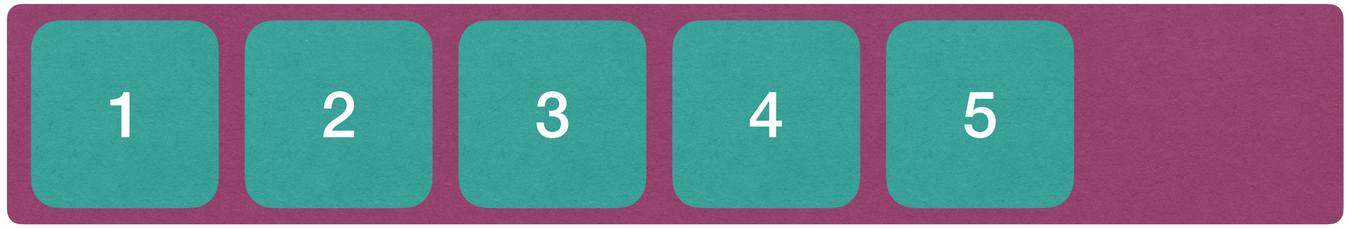
row-reverse

column

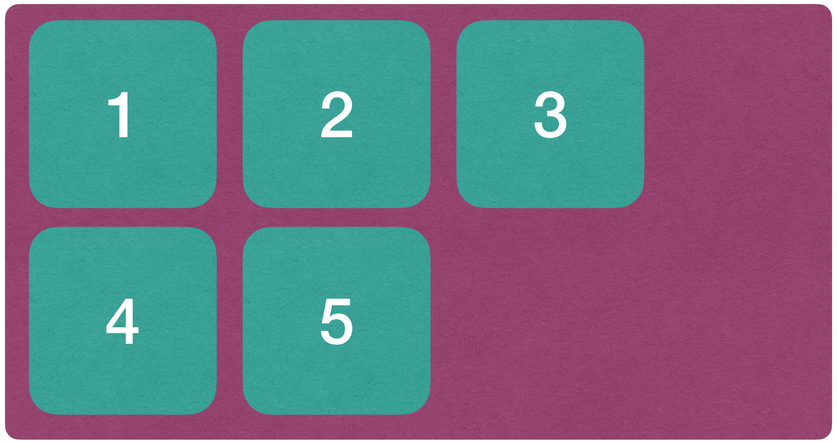
column-reverse

flex-wrap

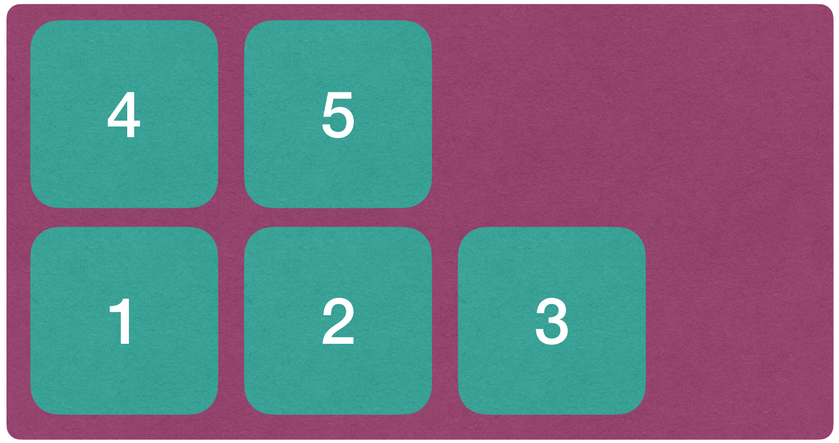
```
.flex-container {
  display: flex;
  flex-wrap: wrap;
}
```



no-wrap



wrap

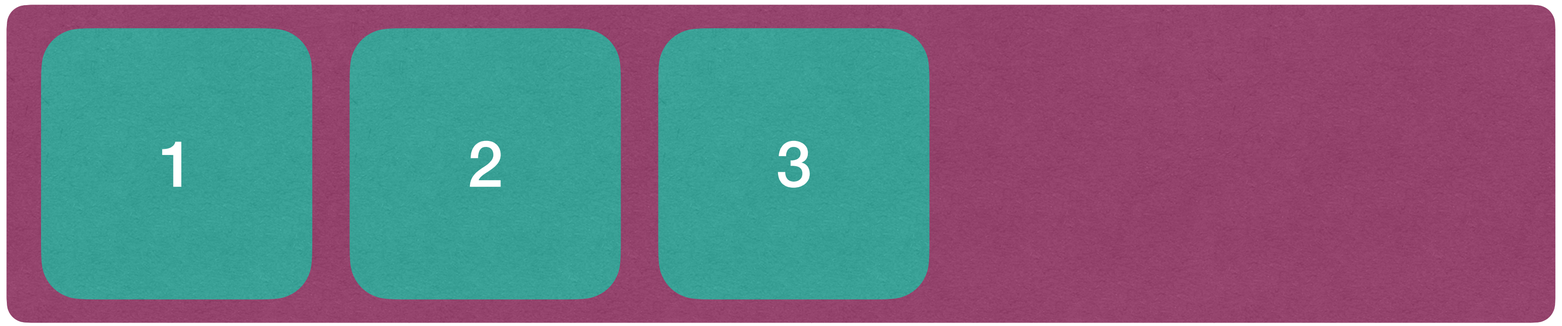


wrap-reverse

flex-flow

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

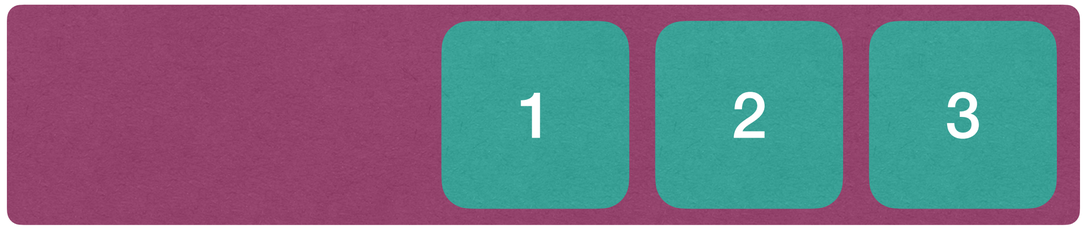
flex-direction + flex-wrap



justify-content

```
.flex-container {
  display: flex;
  Justify-content: flex-start;
}
```

flex-start



flex-end

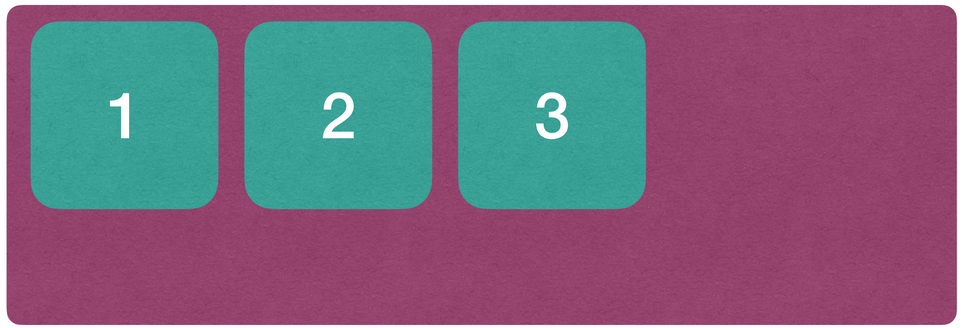
space-between

space-around

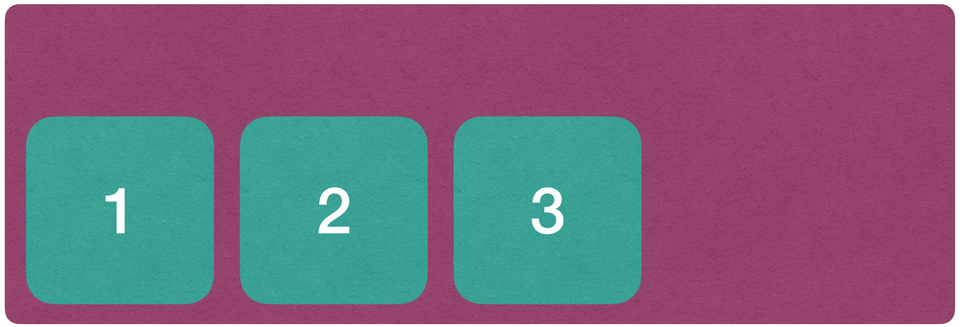
Center

align-items

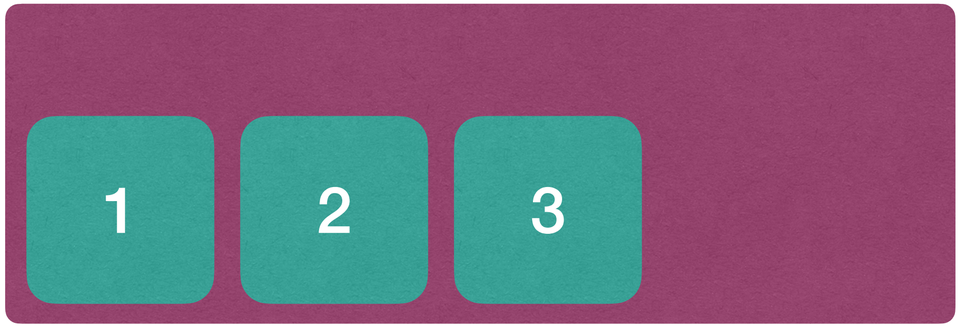
```
.flex-container {
  display: flex;
  height: 300px;
  align-items: flex-start;
}
```



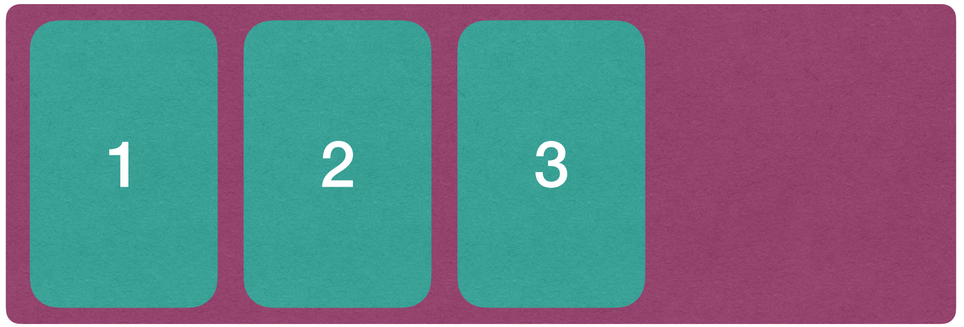
flex-start



flex-end



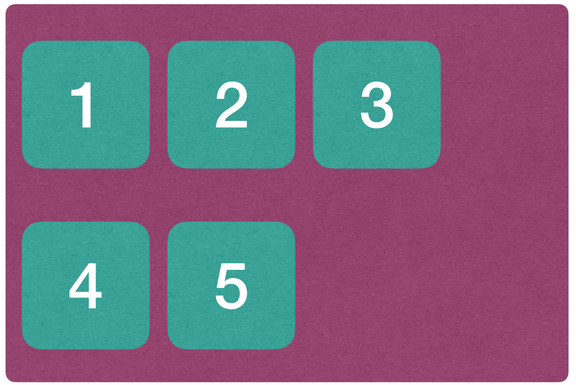
center



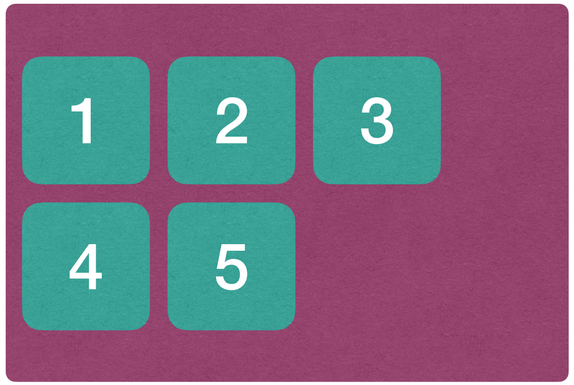
sketch

align-content

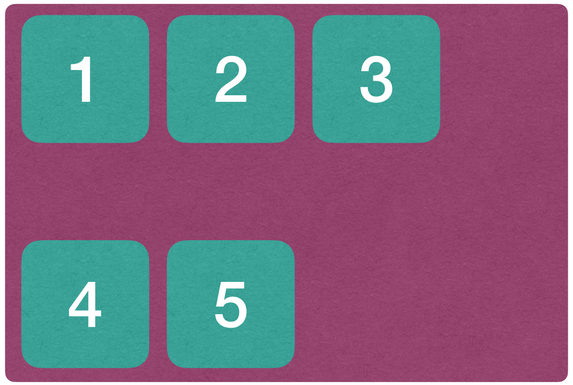
```
.flex-container {
  display: flex;
  height: 600px;
  flex-wrap: wrap
  align-content: space-between;
}
```



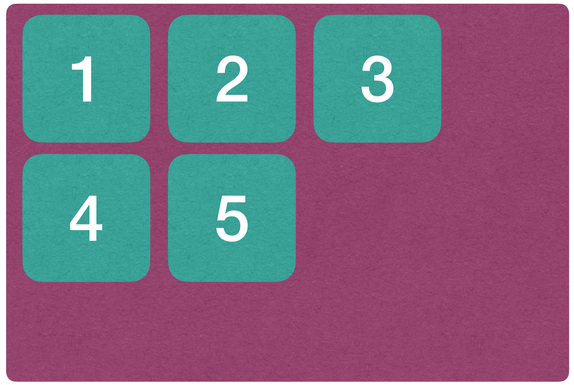
space-around



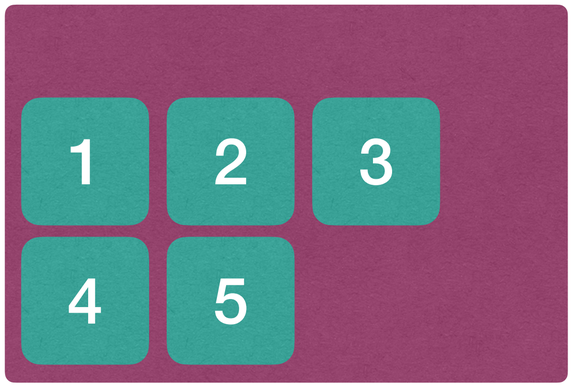
center



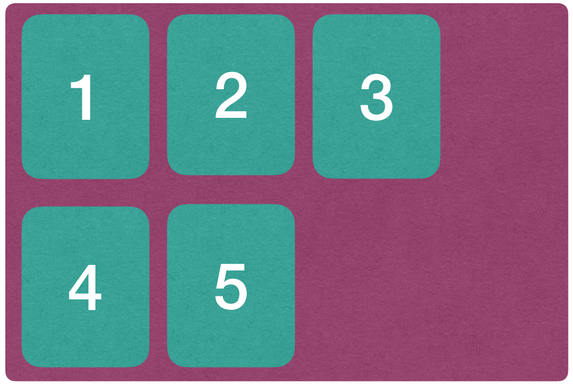
space-between



flex-start



flex-end



sketch

Variables

```
:root {
  —green: #00FF00;
  —black: #000000;
  —white: #FFFFFF;
}


color: var(—green);
Background-color: var(—black);
```

Split css

```
@import “file.css”
```