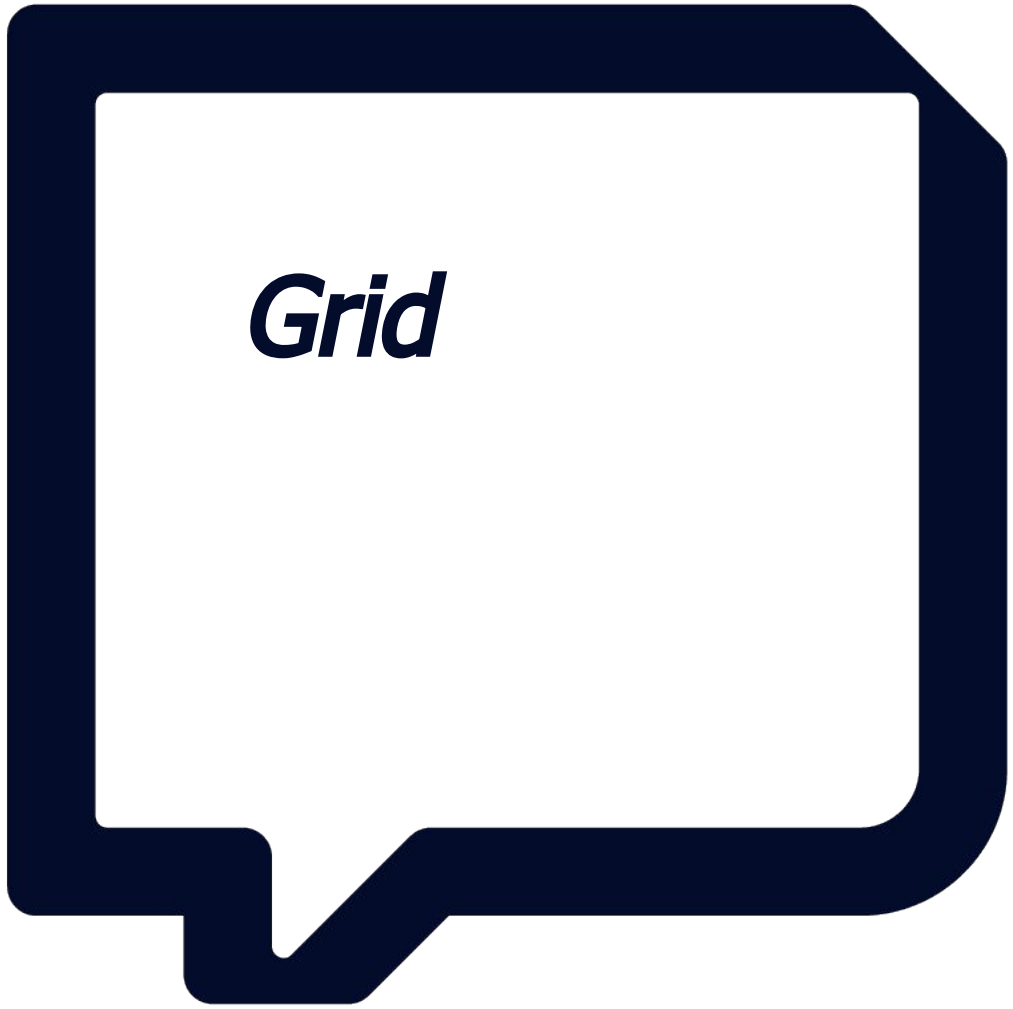
VEM SER

HTML e CSS

Aula 04 - Grid e Animations


Grid

Grid

Grid é um modelo de layout **bidimensional**, ou seja, possui duas direções (linhas **e** colunas).

A vantagem do Grid é proporcionar criar layouts complexos com pouco código, facilitando a legibilidade e trazendo mais agilidade ao desenvolvimento.

Também é possível dividir a página em "grandes regiões", o que torna o código mais organizado e legível.

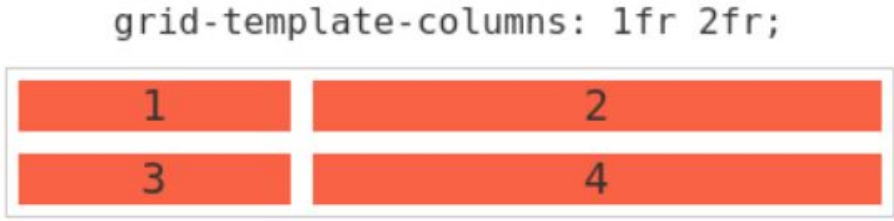
grid-template-columns

```css
.container {
  display: grid;
  grid-template-columns: 100px 100px 100px 100px;
}
```

grid-template-columns: 100px 100px 100px 100px;

fr

```css
.container {
  display: grid;
  grid-template-columns: 1fr 2fr;
}
```



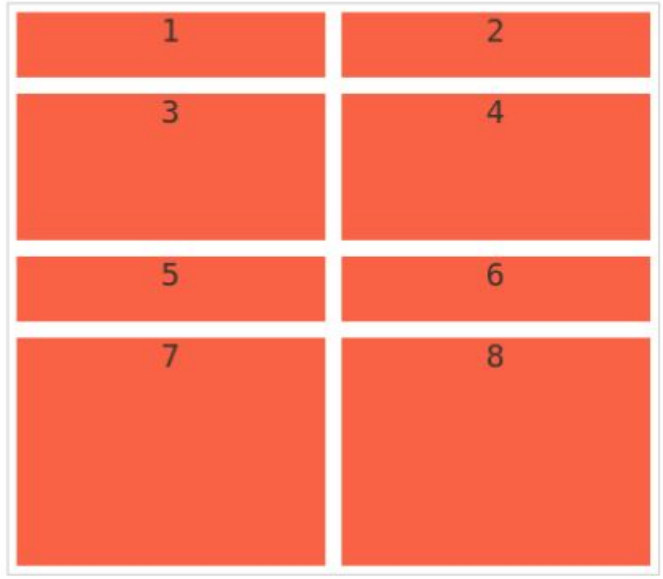
repeat

```css
.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
```

grid-template-rows

```css
.container {
  display: grid;
  grid-template-rows: 50px 100px 50px 150px;
}
```

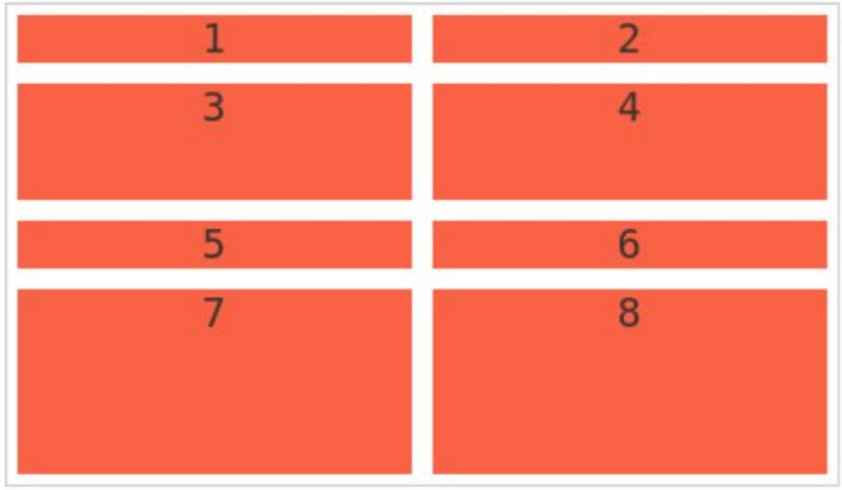
grid-template-rows: 50px 100px 50px 150px;



grid-template-rows

```css
.container {
  display: grid;
  grid-template-rows: 1fr 2fr 1fr 3fr;
}
```

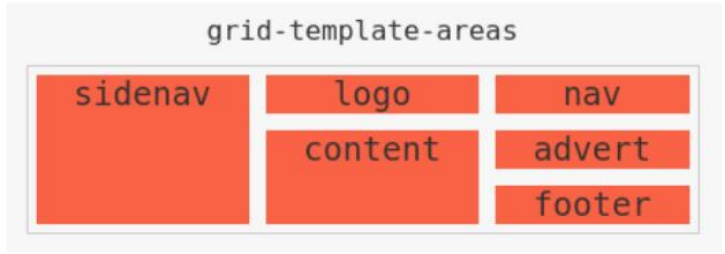
grid-template-rows: 1fr 2fr 1fr 3fr;



grid-template-areas

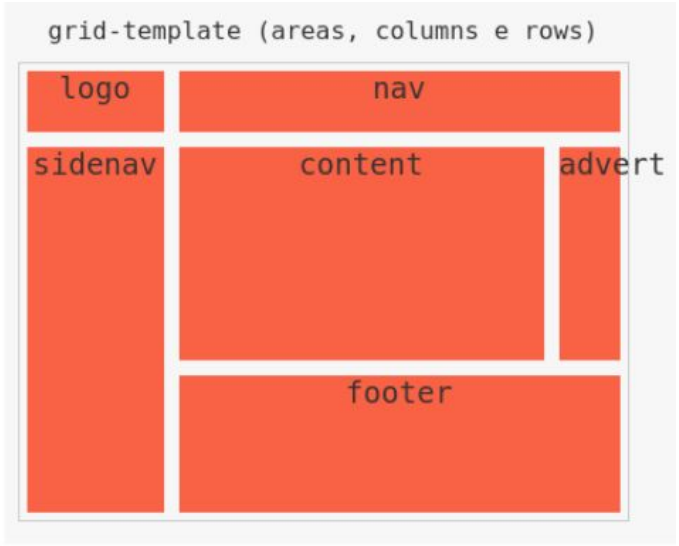
```css
.logo {
  grid-area: logo;
}
.nav {
  grid-area: nav;
}
.content {
  grid-area: content;
}
.sidenav {
  grid-area: sidenav;
}
.advert {
  grid-area: advert;
}
.footer {
  grid-area: footer;
}
```

```css
.container {
  display: grid;
  grid-template-areas:
    "sidenav logo nav"
    "sidenav content advert"
    "sidenav content footer";
}
```



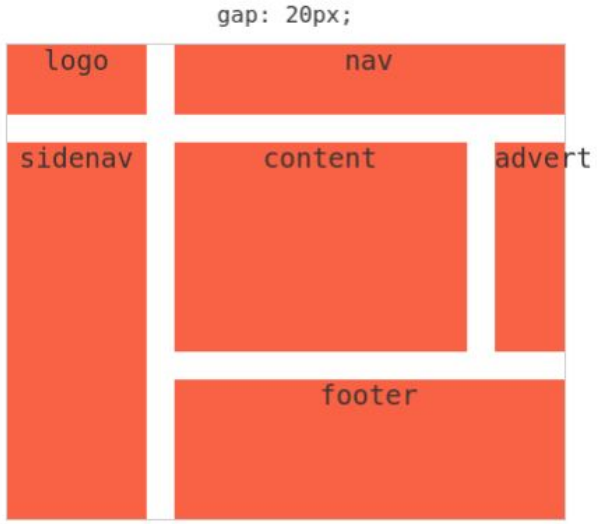
grid-template

```css
.container {
  display: grid;
  grid-template:
    "logo nav nav" 50px
    "sidenav content advert" 150px
    "sidenav footer footer" 100px
    / 100px 1fr 50px;
}
```



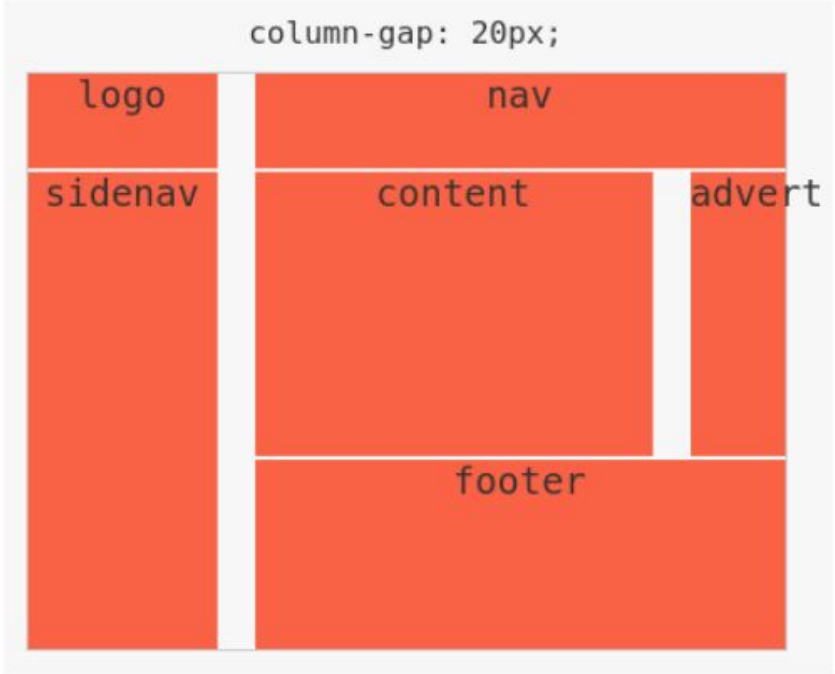
gap

```
1  .gap {
2    gap: 20px;
3  }
```



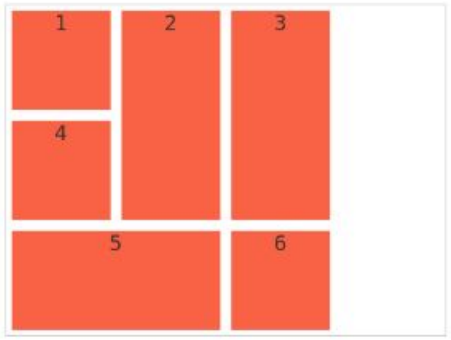
column-gap

```
1  .column-gap {
2    gap: 2px;
3    column-gap: 20px;
4  }
```

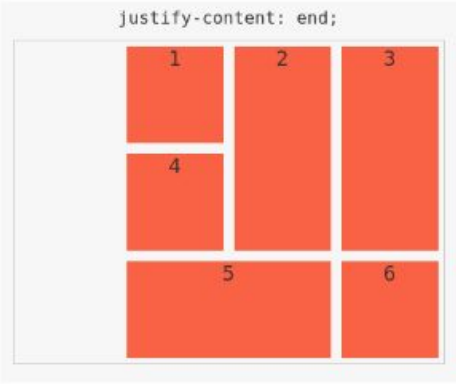


justify-content

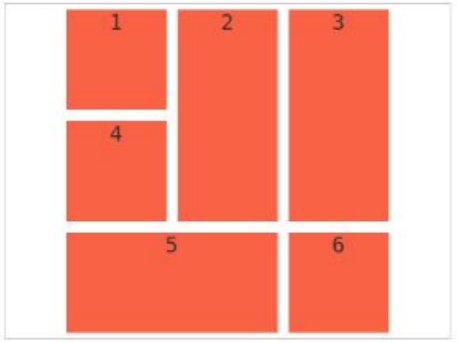
justify-content: start;



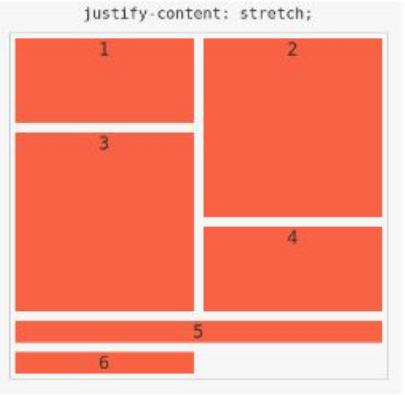
justify-content: end;



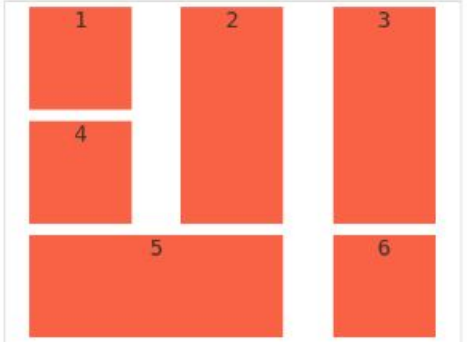
justify-content: center;



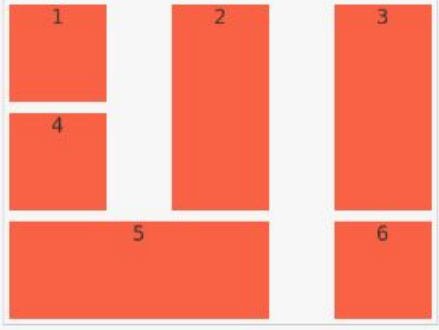
justify-content: stretch;



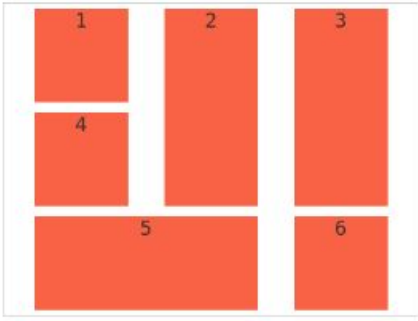
justify-content: space-around;



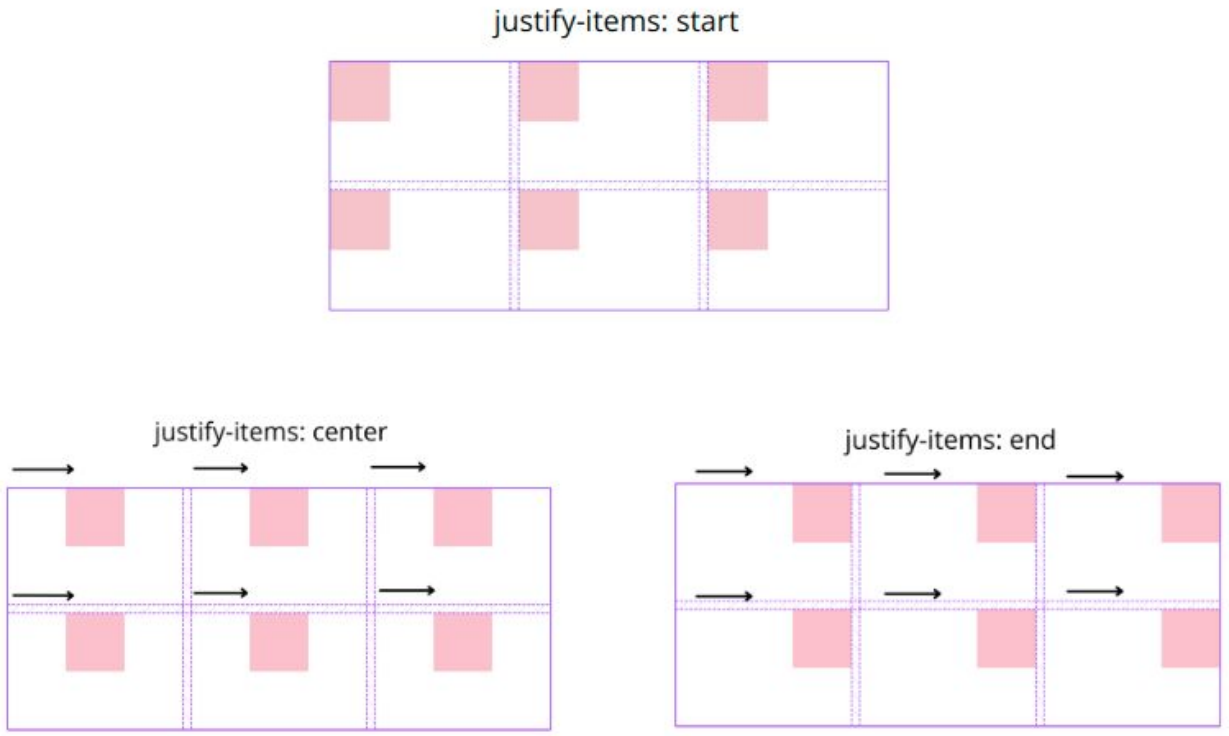
justify-content: space-between;



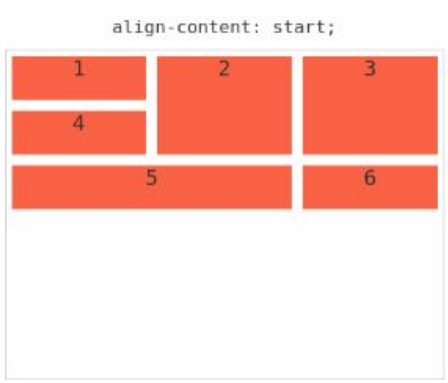
justify-content: space-evenly;

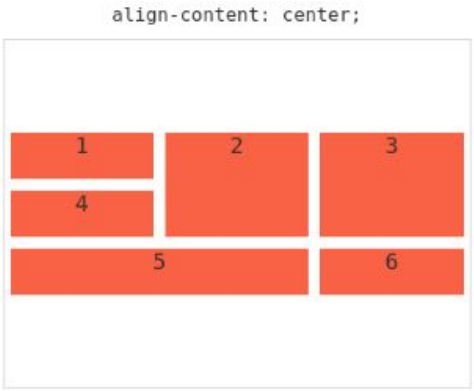


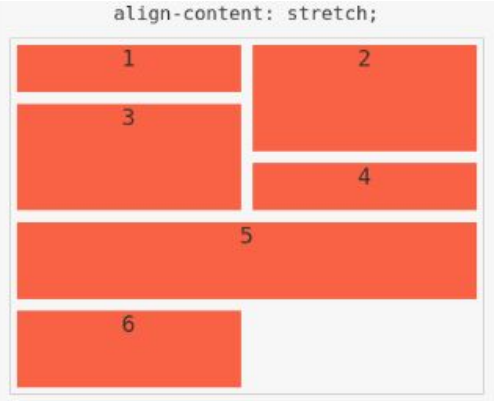
justify-items

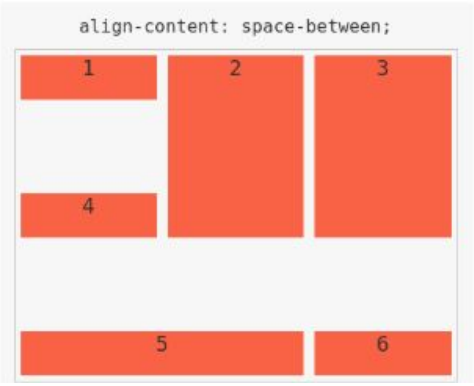


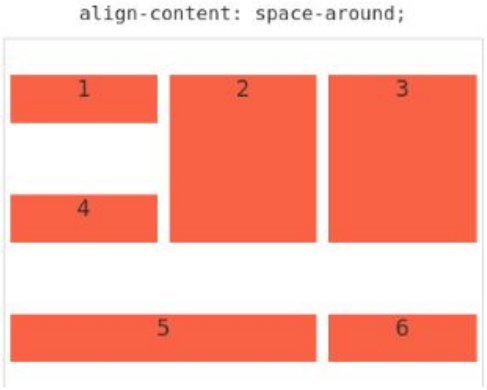
align-content

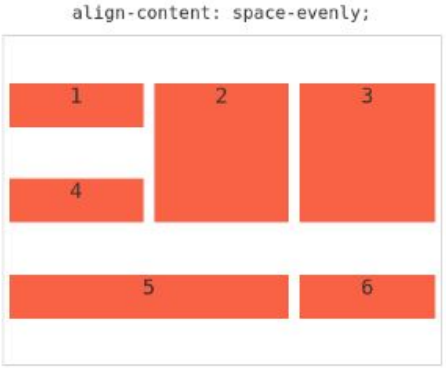




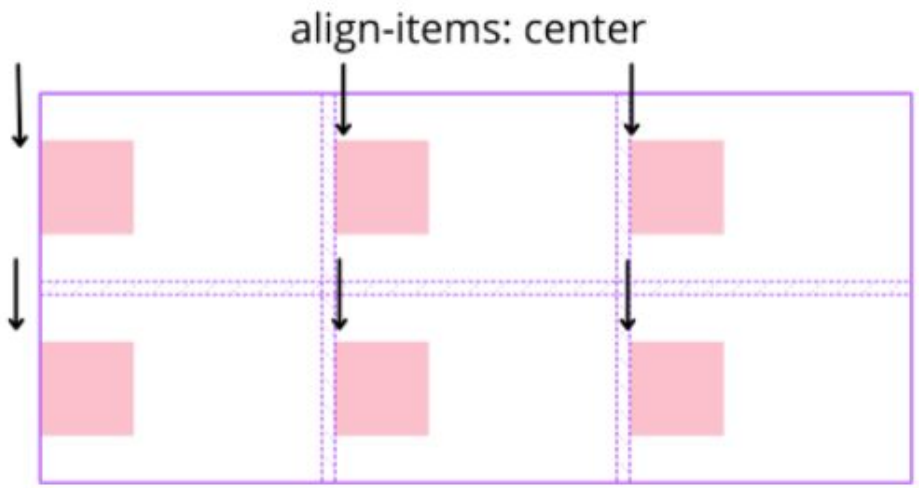


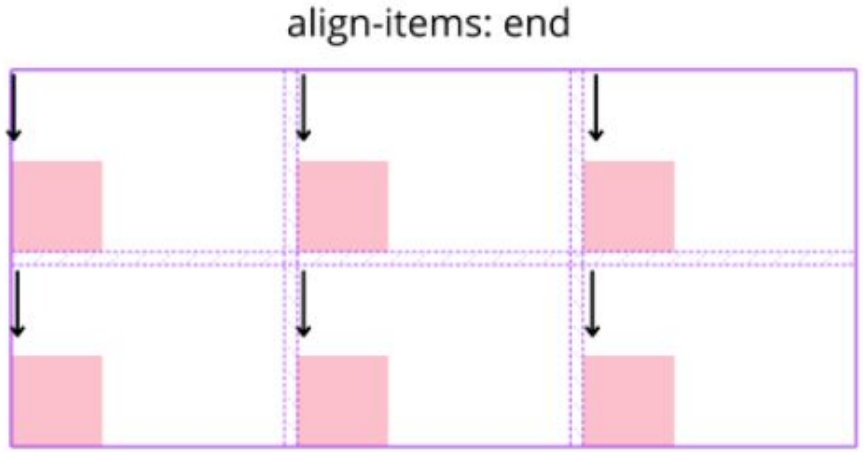






align-items






Variáveis

Variáveis

No design são chamadas de ***tokens*** ou ***design tokens***.
São muito importantes para aumentar a legibilidade, escalabilidade e organização do código.

```css
:root {
  --cor-principal: #c5c;
}
```

```css
.container-um {
  background-color: var(--cor-principal);
}
```

Medidas:
Absolutas
vs
Relativas

px: A medida absoluta

Uma medida absoluta nada mais é do que aquela **que não depende de um outro referencial** e que **não se altera** independentemente da circunstância.

A medida absoluta mais conhecida é **px** ou *pixel.*
Quando definimos uma altura de 10px, por exemplo, independentemente se o dispositivo que está acessando for celular, tablet ou desktop, a altura será, obrigatoriamente, 10px.

em, rem: Medidas relativas

Uma medida relativa é aquela que **depende de algum referencial** para ter seu valor determinado.
Usamos em e rem, geralmente, para fontes.

em: *emphemeral unit*, em tradução livre: "unidade efêmera". No caso da unidade em, o referencial é o **elemento pai**.
Se o elemento pai tiver um *font-size* de 20px e o filho 1em, então o filho terá 20px de *font-size.*

rem: *root emphemeral unit.* No caso da unidade rem, o referencial dela é o elemento **raiz (*:root*)** do CSS.
Se o root tiver um *font-size* de 10px e os elementos da página tiverem 1rem, então eles terão 10px de *font-size.*

vw, vh: Medidas relativas

Usamos vw e vh para tamanhos em geral (alturas e larguras).
O referencial é o tamanho da tela que está acessando.

vw: *view width*, sendo que 100vw é 100% da largura da tela que está acessando.

vh: *view height*, sendo que 100vh é 100% da altura da tela que está acessando

%: Medida relativa

O referencial do % é o tamanho do elemento pai. Ou seja, se o pai tem uma *width* de 100px e o filho tem 50% de *width*, o tamanho do filho será 50px.


Transitions
e Animations

Transition

Uma transição irá intermediar dois estados de um elemento.

A propriedade **transition** é *shorthand* para quatro propriedades:

- transition-property;
- transition-duration;
- transition-timing-function;
- transition-delay.

Animation

Com o ***CSS Animation*** conseguimos fazer animações relativamente complexas sem o uso de JavaScript.

Animation

Cada animação tem ***keyframes,*** que nos ajudam a definir como deve ocorrer a sequência da animação. Ou seja, como a animação deve ser renderizada.

Eles nos ajudam a estabelecer onde a animação deve começar e onde deve terminar (from/to, 50%/100%).

Também podem existir mais de um estado intermediário, por exemplo: 25%/50%/75%/100%.

Animation

A propriedade **animation** é um *shorthand* para as seguintes propriedades:

- **animation-name;**
- **animation-duration;**
- **animation-delay;**
- **animation-timing-function;**
- **animation-iteration-count;**
- animation-direction;
- animation-fill-mode;
- animation-play-state.

Dicas de material

https://css-tricks.com/snippets/css/complete-guide-grid/
https://www.youtube.com/watch?v=LCEgHntqBps


Let's Tech Up Together