CSS - Layout

O que vamos ver hoje?

- Display
- Position
- Flexbox
- Grid
- Responsividade
- Animações

Display

Block

- O elemento block, não aceita elementos na mesma linha que ele, ou seja, quebra a linha após o elemento, e sua área ocupa toda a linha onde ele é inserido.
- Alguns elementos que têm como padrão block: , <h1> até <h6>, <p>, <form>, <header>, <footer>, <section>, <table>.

Inline

- O elemento inline não inicia em uma nova linha nem quebra de linha após o elemento, pois ele ocupa somente o espaço de seu conteúdo.

- Alguns elementos que têm como padrão inline: , <a>, <img>.

Position

Position

- Especifica como um elemento será posicionado na tela.
- Podemos até posicionar em um ponto específico controlando com os parâmetros top, right, bottom e left.

Static

- Este é o valor padrão de todos os elementos HTML, neste posicionamento os elementos não podem ser controlados por top, right, bottom e left, e não tem seu posicionamento afetado pelo posicionamento de outros elementos.

Relative

- Um elemento com relative tem seu posicionamento relacionado com o elemento anterior.

Absolute

- Um elemento com absolute tem seu posicionamento relacionado com o elemento pai e não com o elemento anterior, desta maneira elementos anteriores não irão afetar seu posicionamento.

Fixed

- O elemento fica fixo na tela, isso é, mesmo se acontecer a rolagem ele ficará fixado na página.

Position

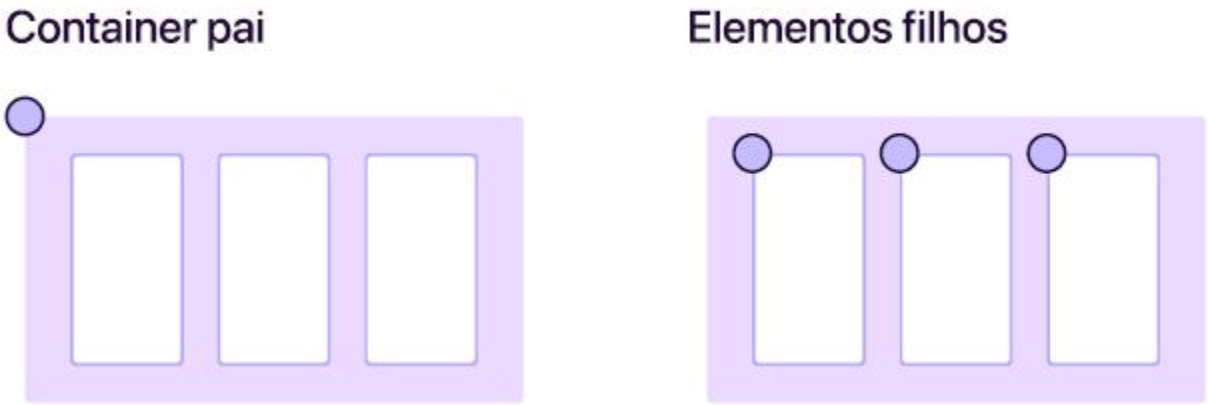
```css
.relative-parent {
  background: #156236;
}

.relative {
  background: #df5d12;
  height: 100px;
  width: 300px;
  position: relative;
  top: 20px;
  left: 45px;
}

.absolute {
  background: #c590f5;
  height: 100px;
  width: 120px;
  position: absolute;
  top: 20px;
  right: 32px;
}

.fixed {
  background: #c4679d;
  height: 50px;
  width: 300px;
  position: fixed;
  bottom: 60px;
  right: 32px;
}
```

```html
<div class="relative-parent">
  <div class="relative">Elemento filho com posicionamento relativo</div>
</div>
<div class="absolute">Elemento com posicionamento absoluto</div>
<div class="fixed">Elemento com posicionamento fixo</div>
```

Flexbox

Flexbox

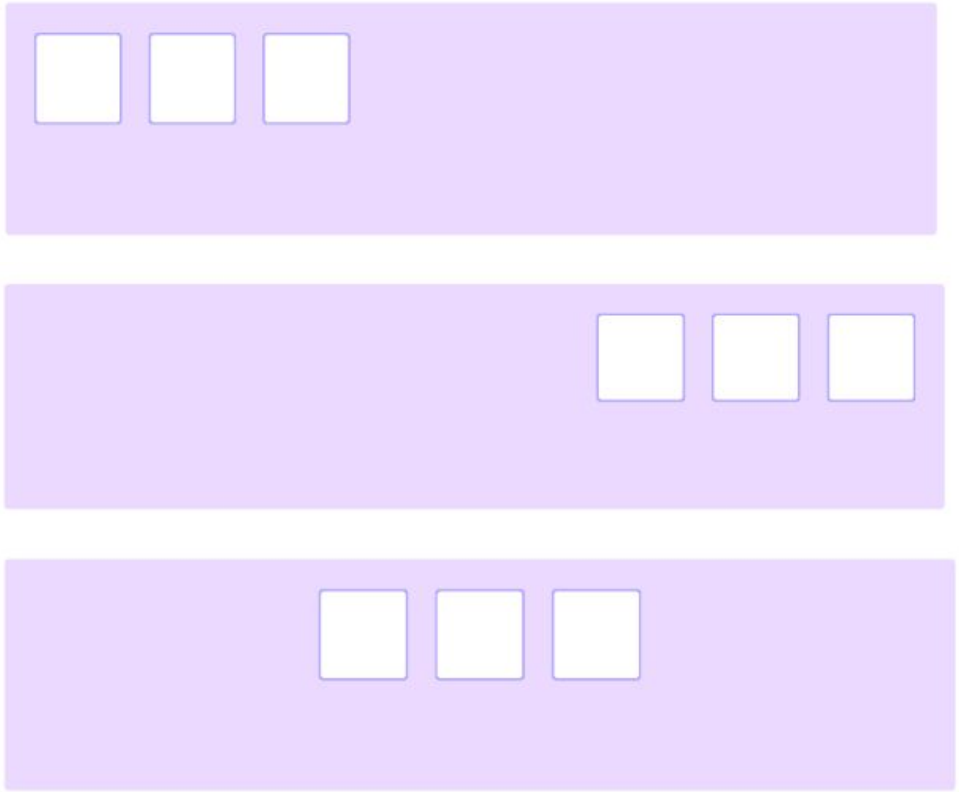
- São regras que configuram o alinhamento, posicionamento e distribuição dos elementos em uma caixa pai através de regras.



justify-content

- justify-content: define como o navegador distribui o espaço entre e ao redor dos itens ao longo do eixo principal.
    - flex-start, flex-end, flex-center.

justify-content



justify-content

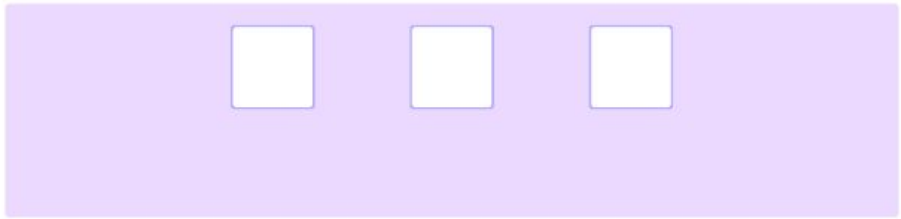
- space-between: primeiro item fica no começo e o último no fim, os restantes ficam alinhados entre si no meio.

justify-content

- space-around: Pega todo o espaço que sobrar na linha e distribui igualmente.

justify-content

- space-evenly: Espaço distribuído igualmente, com exceção do primeiro e do último.



Flexbox

- align-items: define como o navegador distribui o espaço vertical.

align-items

flex-start

Os itens são postos no começo do eixo transversal.

align-items

flex-end

Os itens são alocados no final do eixo transversal.

align-items

stretch

Os itens se estenderão por todo o eixo.

align-items

center

Os itens são alinhados no meio do eixo.

CSS Grid

Grid

- Estrutura do layout em linhas e colunas.
- A estrutura principal chamamos de grid container e é utilizada declarando a propriedade display: grid.

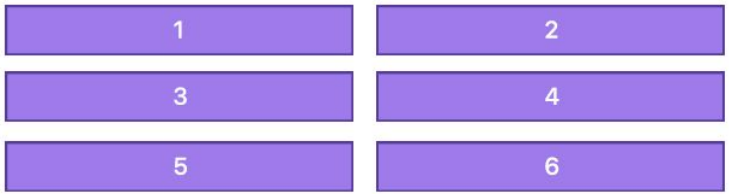
```css
.container {
 display: grid;
}
```

```html
<div class="container">
 <div class="item">1</div>
 <div class="item">2</div>
 <div class="item">3</div>
 <div class="item">4</div>
</div>
```

grid-template-columns

- Para criar colunas usamos a propriedade grid-template-colums que recebe os valores de cada coluna separados por espaço:

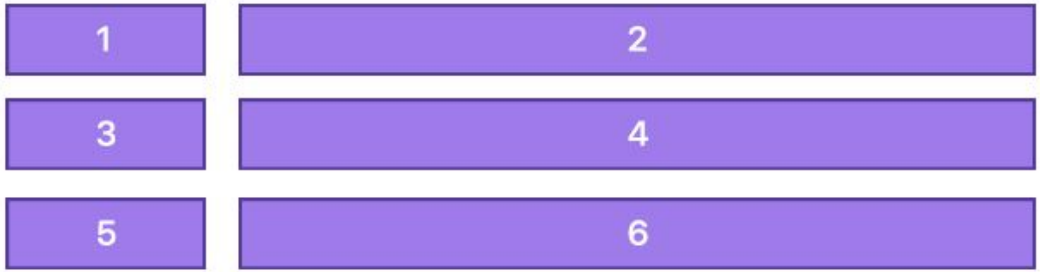
```css
.container {
 display: grid;
 grid-template-columns: 200px 200px;
}
```



grid-template-columns

- Podemos criar um layout em que a primeira coluna ocupa um espaço fixo e a segunda ocupa todo o resto:

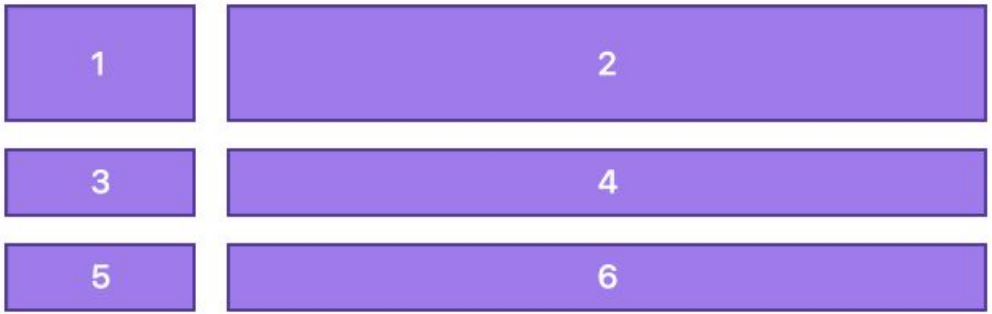
```css
.container {
 display: grid;
 grid-template-columns: 200px 1fr;
}
```



grid-template-rows

- Usado para criar linhas:

```css
.container {
  display: grid;
  grid-template-columns: 200px 1fr;
  grid-template-rows: 200px 100px;
}
```



grid-template-areas

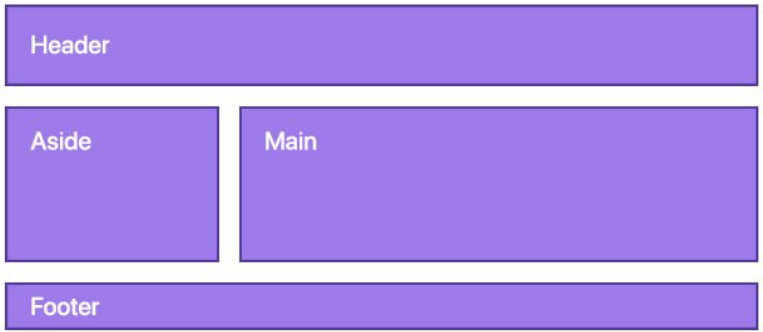
- Geralmente, os itens vão se ajeitando conforme a ordem que estão no html. No entanto podemos dar nomes a cada uma das áreas da nossa grade e depois indicar onde cada elemento deve ir.

```html
<div class="container">
 <div class="item header">Header</div>
 <div class="item sidenav">Sidenav</div>
 <div class="item main">Main</div>
 <div class="item footer">Footer</div>
</div>
```

grid-template-areas

```css
.header {
 grid-area: header;
}
.sidenav {
 grid-area: sidenav;
}
.content {
 grid-area: main;
}
.footer {
 grid-area: footer;
}
```

```css
.container {
 display: grid;
 grid-template-columns: 30% 60%;
 grid-template-rows: 30px 80px 20px;
 grid-template-areas: "header header"
                      "sidenav main"
                      "footer footer";
}
```



Responsividade

Responsividade

- Existem diversas formas de aplicar a responsividade em um site. a mais utilizada é o uso de media queries.
- A responsividade desta forma é aplicada via CSS com propriedades de media types.
- Os types mais utilizados são:
  - all: todos os dispositivos.
  - print: impressoras.
  - screen: telas em geral.
  - speech: leitores de tela.

Media Queries

- O exemplo a seguir altera a cor de fundo da página para roxo se a janela tiver 480 pixels de largura ou menos:

```
1 @media screen and (max-width: 480px) {
2   body {
3     background-color: #00ff00;
4   }
5 }
```

Media Queries

- Padrões de resolução:

```
@media (min-width:320px) { /* smartphones* / }
@media (min-width:480px) { /* smartphones landscape */ }
@media (min-width:600px) { /* tablets */ }
@media (min-width:801px) { /* tablets ands desktops */ }
@media (min-width:1025px) { /* big landscape tablets, laptops, and desktops */ }
@media (min-width:1281px) { /* laptops and desktops */ }
```

Animações

Animações

- Com o uso de keyframes podemos realizar vários tipos de animações:

```css
p {
  background-color: rgb(226, 162, 43);
  animation-duration: 3s;
  animation-name: animacao;
  animation-iteration-count: infinite;
}

@keyframes animacao {
  from {
    background-color: blueviolet;
    color: #0905ff;
  }
  to {
    background-color: blue;
    color: #fff;
  }
}
```

Material Complementar

Material Complementar

- Gerar animações:
  - https://animista.net/play/

Resumo

Resumo 

- Podemos criar layouts usando Flexbox ou Grid.
- Deixamos o site responsivo através de media queries.

Material Complementar

Media Queries

- Guia Flexbox:
  - https://www.alura.com.br/artigos/css-guia-do-flexbox
- Guia Grid:
  - https://developer.mozilla.org/pt-BR/docs/Web/CSS/CSS_grid_layout/Basic_concepts_of_grid_layout

Dúvidas? 

Programa

3000 TALENTOS TI

Obrigado(a)!