

Flexbox

O que aprendemos na aula anterior?

Formatar textos, formulários
HTML, imagens, pseudo
seletores em CSS



Flexbox

display: flex

```
elemento {  
  display: flex;  
}
```

Definição: Torna o elemento um flex container automaticamente transformando todos os seus filhos diretos em flex itens.

Flexbox

flex-direction

```
elemento {  
    display: flex;  
    flex-direction: column;  
}
```

[row, row-reverse, column-reverse]

Definição: Define a direção dos flex itens. Por padrão ele é row (linha), por isso quando o display: flex; é adicionado, os elementos ficam em linha, um do lado do outro.

Flexbox

justify-content

```
elemento {  
    display: flex;  
    justify-content: flex-start;  
}
```

[flex-end, center, space-between, space-around]

Definição: Alinha os itens flex no container de acordo com a direção. A propriedade só funciona se os itens atuais não ocuparem todo o container. Isso significa que ao definir flex: 1; ou algo similar nos itens, a propriedade não terá mais função.

Flexbox

flex-wrap

```
elemento {  
    display: flex;  
    flex-wrap: wrap;  
}
```

[nowrap, wrap-reverse]

Definição: Define se os itens devem quebrar ou não a linha. Por padrão eles não quebram linha, isso faz com que os flex itens sejam compactados além do limite do conteúdo.

Flexbox

align-items

```
elemento {  
    display: flex;  
    align-items: center;  
}
```

[flex-end, flex-start, stretch]

Definição: O align-items alinha os flex itens de acordo com o eixo do container. O alinhamento é diferente para quando os itens estão em colunas ou linhas.

Flexbox

align-content

```
elemento {  
    display: flex;  
    align-content: center;  
}
```

[flex-end, flex-start, stretch]

Definição: Alinha as linhas do container em relação ao eixo vertical. A propriedade só funciona se existir mais de uma linha de flex-itens. Para isso o flex-wrap precisa ser wrap.

Flexbox

align-self

```
elementoPai {  
  display: flex;  
  align-items: flex-start;  
}  
elementoFilho {  
  align-self: flex-end;  
}
```

[flex-end, flex-start, stretch]

Definição: O align-self aceita os mesmos valores que a propriedade align-items mas alinha somente o item selecionado.



ClassDojo

Referências

flexbox

<https://origamid.com/projetos/flexbox-guia-completo/>

<http://www.flexboxdefense.com/>

<https://flexboxfroggy.com/#pt-br>

Na próxima aula

Media Queries



Unidades relativas e media-queries

**O que aprendemos na
aula anterior?**

Flexbox



Mobile first



Responsive Web Design

Mobile First Web Design



Estruturar a página **pensando primeiro**
em um **dispositivo móvel.**

Unidades de medida relativas



Medidas relacionadas ao container pai direto.

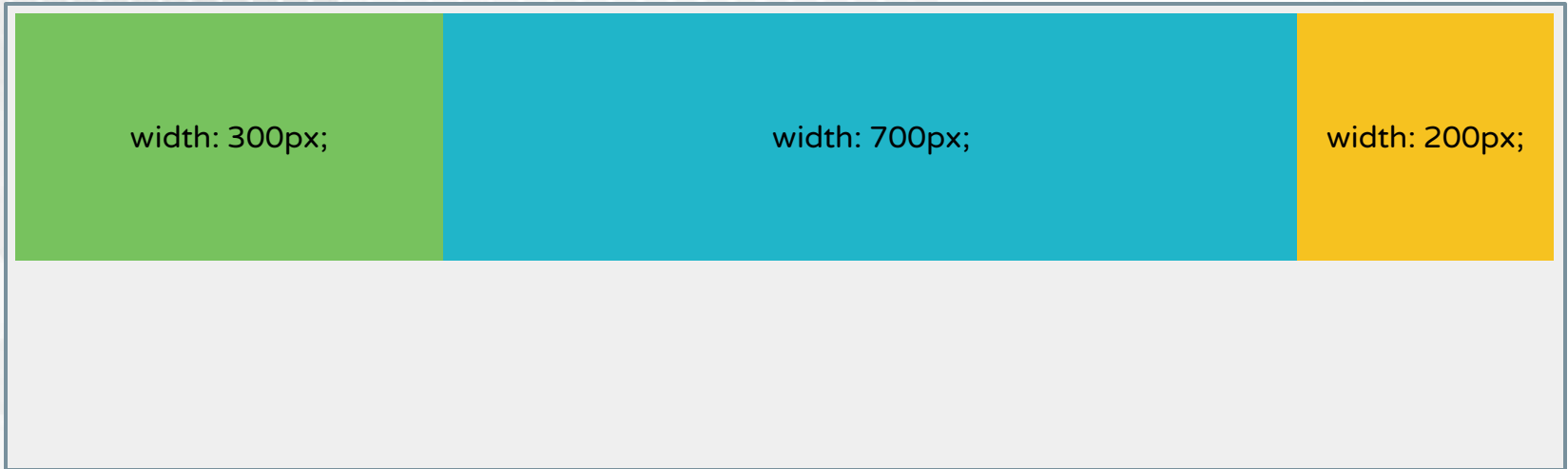
Porcentagens - %

Uma medida expressa em porcentagem **SEMPRE** está relacionada à medida (no mesmo eixo) do **elemento pai** que a contém.

```
.box{  
  width: 25%;  
}
```

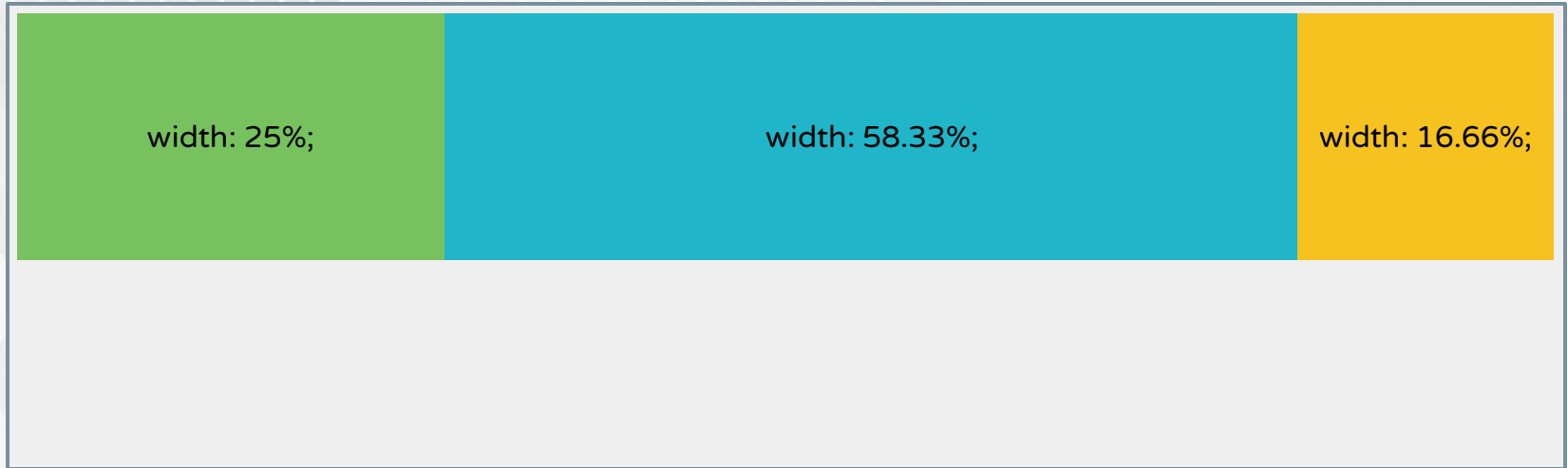
Geralmente, não usamos porcentagens para a altura.

Pixels em percentual



container pai com **1200px** em width

Pixels em percentual



container pai com **1200px** em width

Pixels em percentual

width: 25%;
 $(300/1200)*100$

width: 58.33%;
 $(700/1200)*100$

width: 16.66%;
 $(200/1200)$
 $* 100$

Nesse caso, nossa "base" é **1200px**.

Medida **em**

Os em's são **unidades** de medida que **devem ser usadas** para tudo o que tiver relação com **tipografias**.

1em começará sendo igual a **16px** (a menos que configuremos outra coisa)

```
p {  
  font-size: 1em;  
  line-height: 1.5em;  
}
```



```
<body>  
  <div>  
    <p>  
      <strong><strong>  
    </p>  
  </div>  
</body>
```

Diagram illustrating font-size inheritance in HTML:

- `<body>` is annotated with `font-size: 16px;`.
- `<div>` is annotated with `font-size: 1em;`.
- `<p>` is annotated with `font-size: 1em;`.
- `` is annotated with `font-size: 1em;`.

`<body>` tem 16px em font-size, os outros elementos, por padrão, vêm com 1em em font-size.

Portanto, os demais elementos têm 16px de font-size.



```
<body>  
  <div>  
    <p>  
      <strong><strong>  
    </p>  
  </div>  
</body>
```

Diagram illustrating font-size inheritance in HTML:

- `<body>` has a font-size of 16px (indicated by a green arrow).
- `<div>` has a font-size of 2em (indicated by a red arrow).
- `<p>` has a font-size of 1em (indicated by a green arrow).
- `` has a font-size of 1em (indicated by a green arrow).

Se alteramos o font-size de `<div>` para 2em, ele terá 2 vezes o tamanho de font-size do pai.

`<div>` agora tem 32px de font-size.

Portanto, agora, os filhos de `<div>` mudaram porque alteramos o pai.

Os filhos de `<div>` ficaram com 32px de font-size.



```
<body>  
  <div>  
    <p>  
      <strong><strong>  
    </p>  
  </div>  
</body>
```

Diagram illustrating font-size inheritance in CSS:

- `<body>` has a font-size of 16px (indicated by a green arrow).
- `<div>` inherits the font-size from `<body>` and has a font-size of 2em (indicated by a red arrow).
- `<p>` inherits the font-size from `<div>` and has a font-size of 1.5em (indicated by a red arrow).
- `` inherits the font-size from `<p>` and has a font-size of 1em (indicated by a green arrow).

`<div>` tem 32px de font-size.

Se definimos para `<p>` o font-size: 1.5em, `<p>` terá 1,5 vezes o tamanho do font-size do pai.

`<p>` agora tem 48px em font-size.

`` tem 1 vez o font-size do pai.

`<p>` tem 48px em font-size.



```
<body>  
  <div>  
    <p>  
      <strong><strong>  
    </p>  
  </div>  
</body>
```

Diagram illustrating font-size inheritance in HTML:

- `<body>` is assigned `font-size: 10px;` (indicated by a red arrow).
- `<div>` inherits from `<body>` and is assigned `font-size: 2em;` (indicated by a red arrow).
- `<p>` inherits from `<div>` and is assigned `font-size: 1.5em;` (indicated by a red arrow).
- `` inherits from `<p>` and is assigned `font-size: 1em;` (indicated by a green arrow).

Agora, se a base do documento (`<body>`) for alterada, com 10px de font-size, todos os elementos mudarão de tamanho, mesmo sem que o font-size deles seja alterado.

`<div>` = 20px de font-size

`<p>` = 30px de font-size

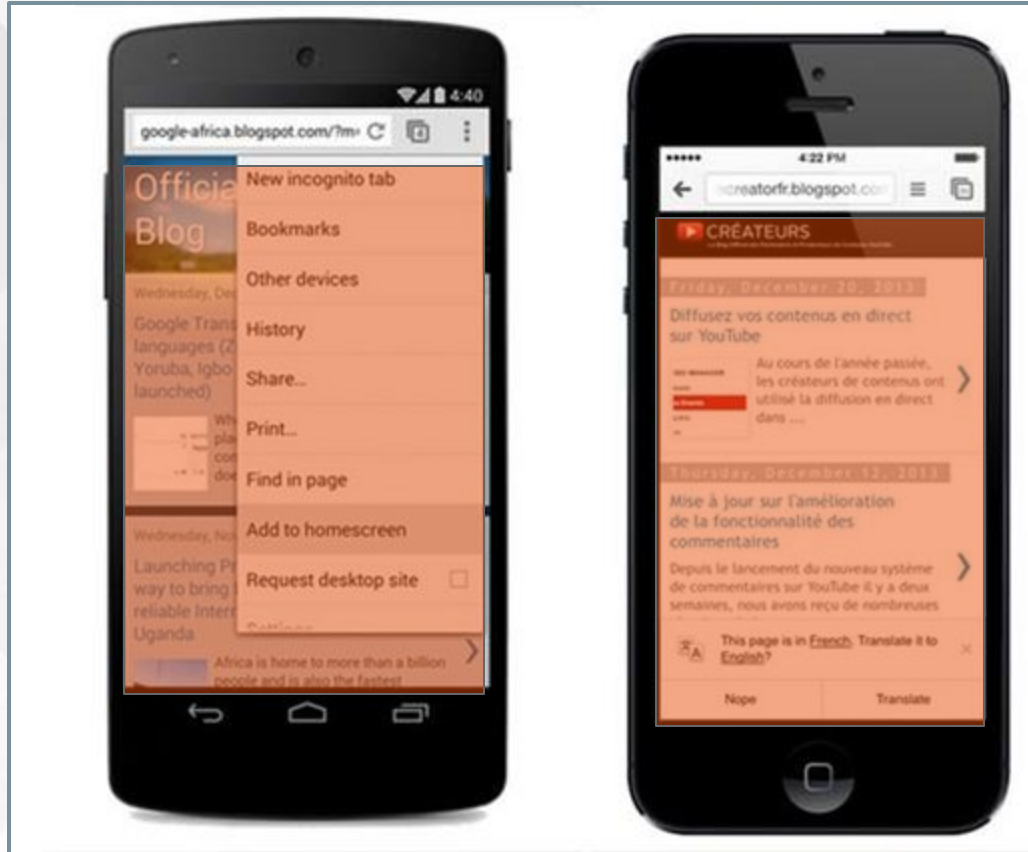
`` = 30px de font-size

Responsividade

A habilidade da aplicação de se adaptar de forma **satisfatória** em qualquer display e dispositivo.



O Viewport



Tag para RWD

A tag **<meta> viewport** dá instruções ao navegador sobre **como dimensionar** e escalar a página web durante o carregamento.

```
<meta name="viewport"  
      content="width=device-width, initial-scale=1">
```


Esta é a estrutura **básica** desta tag, às vezes ela pode ter mais informações.
Saiba mais: [Responsive Web Design](#)

Viewport Measures - vw / vh

Toda medida expressa em **vw/vh** terá **SEMPRE** como eixo de referência o **viewport** do documento.

```
.box {  
  width: 25vw;  
  height: 50vh;  
}
```

Viewport faz referência à “caixa visível” de conteúdo dentro de um navegador.



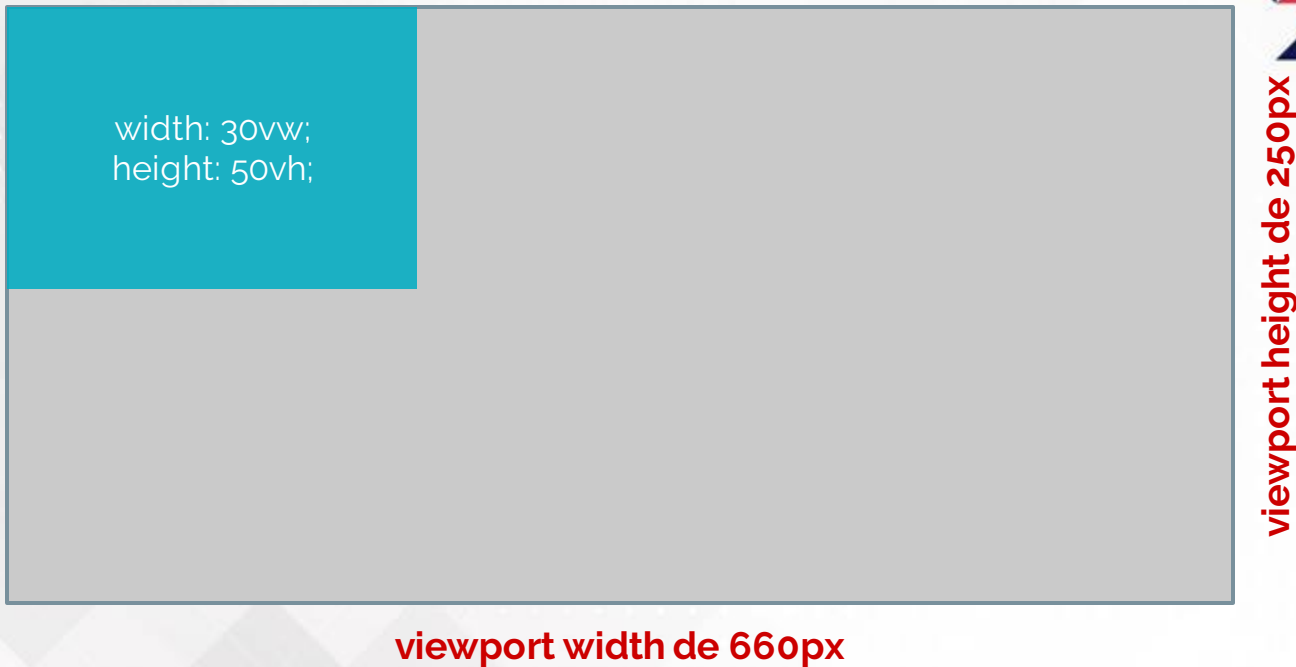
A diagram showing a large gray rectangle representing a viewport. Inside the top-left corner of this rectangle is a smaller teal rectangle. The teal rectangle is labeled with its dimensions in viewport units. The gray rectangle is labeled with its dimensions in pixels. The text is positioned as follows: the teal box label is inside the box; the gray box label is to the right of the box; the viewport width label is below the box; and the viewport height label is to the right of the box.

width: 30vw;
height: 50vh;

viewport height de 450px

Neste caso, a caixa terá 0,3 vezes a largura do viewport (**96px**) e 0,5 vezes a altura do viewport (**225px**)

viewport width de 320px



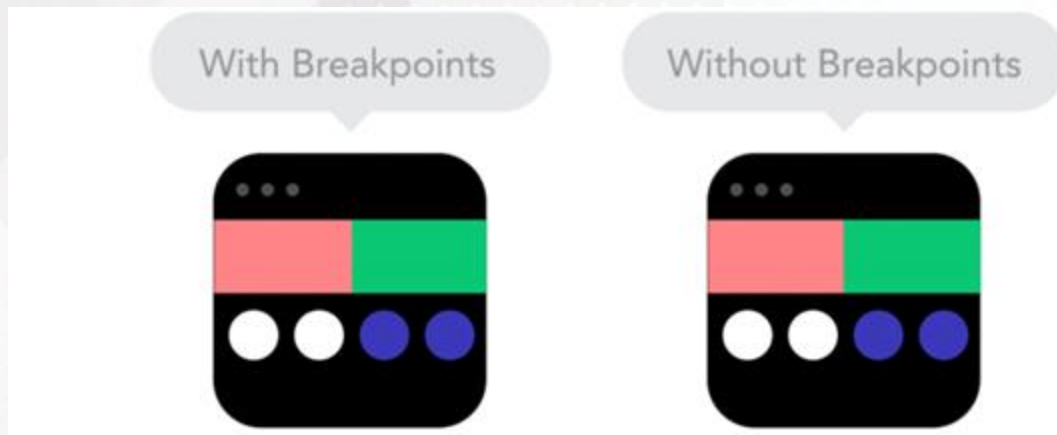
Neste caso, a caixa terá 0,3 vezes a largura do viewport (**198px**) e 0,5 vezes a altura do viewport (**125px**)

Media-queries



Conjunto de regras CSS que permitem reorganizar o conteúdo de acordo com as condições de visualização do documento.

Sempre devem ser escritos no final da folha de CSS.



Media-queries

para Mobile **First**

```
@media (min-width: 460px) {  
    /* regras de CSS */  
}
```

Especificar a **min-width** é como dizer: “se o mínimo é **Npx** de largura, aplicar isto” ou “deste ponto para cima”.

Media-queries

para Mobile **Last**

```
@media (max-width: 960px) {  
    /* regras de CSS */  
}
```

Especificar a **max-width** é como dizer: “se o máximo é Npx de largura, aplicar isto” ou “deste ponto para baixo”.

Media-queries

definindo a **orientação**

```
@media (max-width: 768px) and (orientation: portrait){  
    /* regras de CSS */  
}
```

Especificar **a orientação (portrait ou landscape)**, é como dizer “se a largura máxima é Npx e o dispositivo está em posição vertical/horizontal, aplicar isto”.

Na próxima aula

Visualização e pseudo-
elementos



Visualização e pseudo-elementos

O que aprendemos na aula anterior?

Mobile first, viewport, unidades relativas, simulação de dispositivos, media queries,



Barras de navegação

As barras de navegação devem ser feitas de forma semânticas para SEO



Barras de navegação

Semanticamente, as barras de navegação devem ser feitas com listas, de preferência desordenadas, e links:

```
<ul>
  <li><a href="inicio.html"> Início </a></li>
  <li><a href="quem.html"> Quem somos </a></li>
  <li><a href="servicos.html"> Serviços </a></li>
  <li><a href="lojas.html"> Lojas </a></li>
  <li><a href="contato.html"> Contato </a></li>
</ul>
```


Barras de navegação

Depois de criar a estrutura, vamos precisar:

- Retirar o estilo padrão de lista
- Posicionar `` um ao lado do outro
- Fazer com que os `<a>` se comportem como blocos

Tabelas



Tabelas

```
<table>
  <tr>
    <td>linha 1, célula 1</td>
    <td>linha 1, célula 2</td>
  </tr>
  <tr>
    <td>linha 2, célula 1</td>
    <td>linha 2, célula 2</td>
  </tr>
</table>
```

As tabelas são definidas com a tag `<table>`.

Tabelas

```
<table>
  <tr>
    <td>linha 1, célula 1</td>
    <td>linha 1, célula 2</td>
  </tr>
  <tr>
    <td>linha 2, célula 1</td>
    <td>linha 2, célula 2</td>
  </tr>
</table>
```

Uma tabela é dividida em linhas com a tag `<tr>` (table row),

Tabelas

```
<table>
  <tr>
    <td>linha 1, célula 1</td>
    <td>linha 1, célula 2</td>
  </tr>
  <tr>
    <td>linha 2, célula 1</td>
    <td>linha 2, célula 2</td>
  </tr>
</table>
```

Em cada linha os espaços são divididos em células de dados com a tag `<td>` (table data).

Tabelas

Cabeçalhos

```
<table>
  <tr>
    <th>Cabeçalho coluna 1</th>
    <th>Cabeçalho coluna 2</th>
  </tr>
  <tr>
    <td>linha 2, célula 1</td>
    <td>linha 2, célula 2</td>
  </tr>
</table>
```

Os cabeçalhos em uma tabela são definidos com a tag `<th>` (table heading).

Tabelas

Estilos

```
table, th, td {  
    border: 1px solid black;  
    border-collapse: collapse;  
}
```

Podemos modificar os estilos da tabela via CSS:

border: Define uma borda

border-collapse: Entra em colapso bordas de célula

border-spacing: Define o espaçamento entre as células

Tabelas

colspan, rowspan

```
<table>
  <tr>
    <td rowspan="2">linha 1, célula 1</td>
    <td colspan="2">linha 1, célula 2</td>
  </tr>
  <tr>
    <td>linha 2, célula 1</td>
    <td>linha 2, célula 2</td>
  </tr>
</table>
```

Definição: atributos dos elementos <td> e <th> para mesclar colunas ou linhas.

Pesquisa



Pontos positivos (Boa, mantém assim!)

Pontos negativos (Indicar pontos negativos e sugestão de solução)

Pseudo-elementos

Permitem inserir conteúdo com CSS.

São usados sobre um seletor pré-definido.

Podemos inserir conteúdo de texto, imagens ou elementos de bloco.

Saiba mais:

https://www.w3schools.com/css/css_pseudo_elements.asp

https://www.w3schools.com/css/css_pseudo_classes.asp

https://www.w3schools.com/cssref/selector_nth-child.asp

Pseudo-elementos

::before & ::after

```
h1::before {  
  content: "Capítulo - ";  
}  
  
h1::after {  
  content: " - de 10";  
}
```

Capítulo - 1 - de 10

Pseudo-elementos

::before & ::after

```
.caixa::before {  
  content: url(../images/icone-1.gif);  
}  
  
.caixa::after {  
  content: url(../images/icone-2.gif);  
}
```

Na próxima aula

Bootstrap

