

P.PORTO

TECNOLOGIAS WEB

TECNOLOGIAS E SISTEMAS DE INFORMAÇÃO PARA A WEB

POLITÉCNICO
DO PORTO
ESCOLA
SUPERIOR
DE MEDIA ARTES E
DESIGN

TECNOLOGIAS WEB

TECNOLOGIAS E SISTEMAS DE INFORMAÇÃO PARA A WEB

2020-21

Módulo 3 - CSS

Índice

1. Definição
2. Integração com HTML
3. Sintaxe e Selectores
4. Dimensões
5. Unidades de Medida
6. Box model
7. Borders
8. Texto e Fontes
9. Posicionamento de Elementos
10. Flexbox
11. Grid
12. Layout em Colunas
13. Cascading Order
14. Media Queries
15. Transições
16. Backgrounds
17. Validação

Módulo 3 - CSS

1. Definição

1. Definição

- **Cascading Style Sheets (CSS)**
 - Linguagem de **estilização de conteúdos na Web**
 - Mecanismo para adicionar estilo (**cores, fontes, espaçamento, etc.**) a um documento Web
 - Mantida/desenvolvida pela w3c (<https://www.w3.org/Style/CSS/>)
 - Atualmente na sua **versão 3**
 - **Conjunto de regras** que descrevem como os elementos HTML são exibidos no ecrã
 - Exemplo:

```
body {  
    background-color: #d0e4fe;  
}  
h1 {  
    color: orange;  
    text-align: center;  
}  
p {  
    font-family: "Times New Roman";  
    font-size: 20px;  
}
```

1. Definição

- **Cascading Style Sheets (CSS)**
 - Vantagens:
 - **Economiza tempo:** escreve-se o código CSS uma vez e usa-se em vários projetos HTML
 - **Carregamento mais rápido:** não precisamos escrever atributos HTML para cada tag
 - **Fácil manutenção:** mudar o design da página Web pode ser feito somente fazendo determinadas alterações no ficheiro CSS
 - **Independente da plataforma:** CSS é standard e reconhecido por todos os browsers em qualquer plataforma



Módulo 3 - CSS

2. Integração com HTML

2. Integração com HTML

- Existem **3 formas** de definir as regras **CSS**:
 - **Externa** - criação de ficheiro externo **.css**
 - **Interna** - elemento **style** (dentro do elemento **head**) do ficheiro HTML
 - **Inline** - atributo **style** do elemento HTML a estilizar



2. Integração com HTML

- **Externa**

- Criação de um ficheiro externo com extensão .css
- Forma de mudar a aparência de um site inteiro, alterando apenas um ficheiro!
- Cada página deve incluir uma referência para o ficheiro externo no elemento <link>
- O elemento <link> é incluído dentro do elemento <head>

```
<!-- index.html file -->
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="mystyle.css">
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

```
/* mystyle.css file */
body {
  background-color: lightblue;
}
h1 {
  color: navy;
  margin-left: 20px;
}
```

2. Integração com HTML

- **Interna**

- Aplicam estilo numa **única página HTML**
- Definidos dentro do elemento **<style>**, no elemento **<head>**.

```
<!-- index.html file -->
<!DOCTYPE html>
<html>
  <head>
    <style>
      body {
        background-color: linen;
      }
      h1 {
        color: maroon;
        margin-left: 40px;
      }
    </style>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

2. Integração com HTML

- **Inline**

- Um estilo inline pode ser usado para estilizar um **único elemento**.
- Para usar estilos inline, adicione o atributo **style** ao **elemento** relevante.
- O atributo **style** pode conter qualquer propriedade CSS.

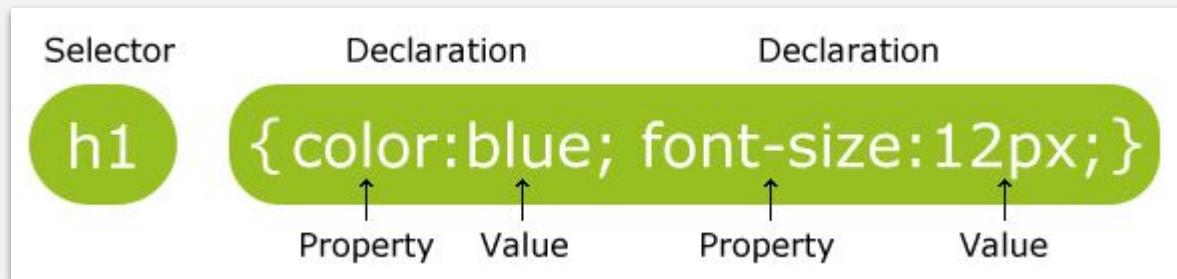
```
<!-- index.html file -->
<!DOCTYPE html>
<html>
  <body>
    <h1 style="color:blue;text-align:center;">This is a heading</h1>
    <p style="color:red;">This is a paragraph.</p>
  </body>
</html>
```

Módulo 3 - CSS

3. Sintaxe e Selectores

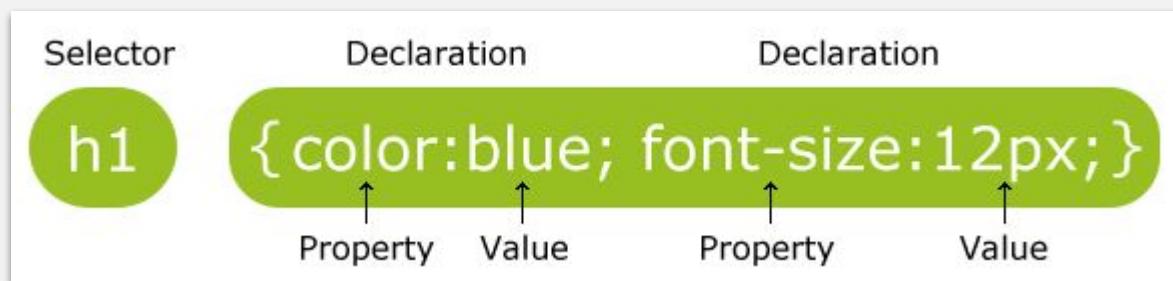
3. Sintaxe e Selectores

- Uma **regra CSS** é composta por:
 - Um **selector** – seleciona os elementos HTML a estilizar
 - Um **bloco de declaração**:
 - Inclui uma ou mais **declarações de estilo** cercadas por chavetas
 - Cada declaração:
 1. Tem uma **propriedade** e um **valor** separados por dois pontos
 2. Termina sempre com um ponto e vírgula



3. Sintaxe e Selectores

- **Selectores CSS** são usados para "encontrar" elementos HTML com base em:
 - Elemento
 - Id
 - Classe
 - Atributo
 - Combinadores
 - Pseudo-classes
 - Pseudo-elementos



3. Sintaxe e Selectores

- **Selector por elemento**
 - Selecciona elementos com base no nome do elemento

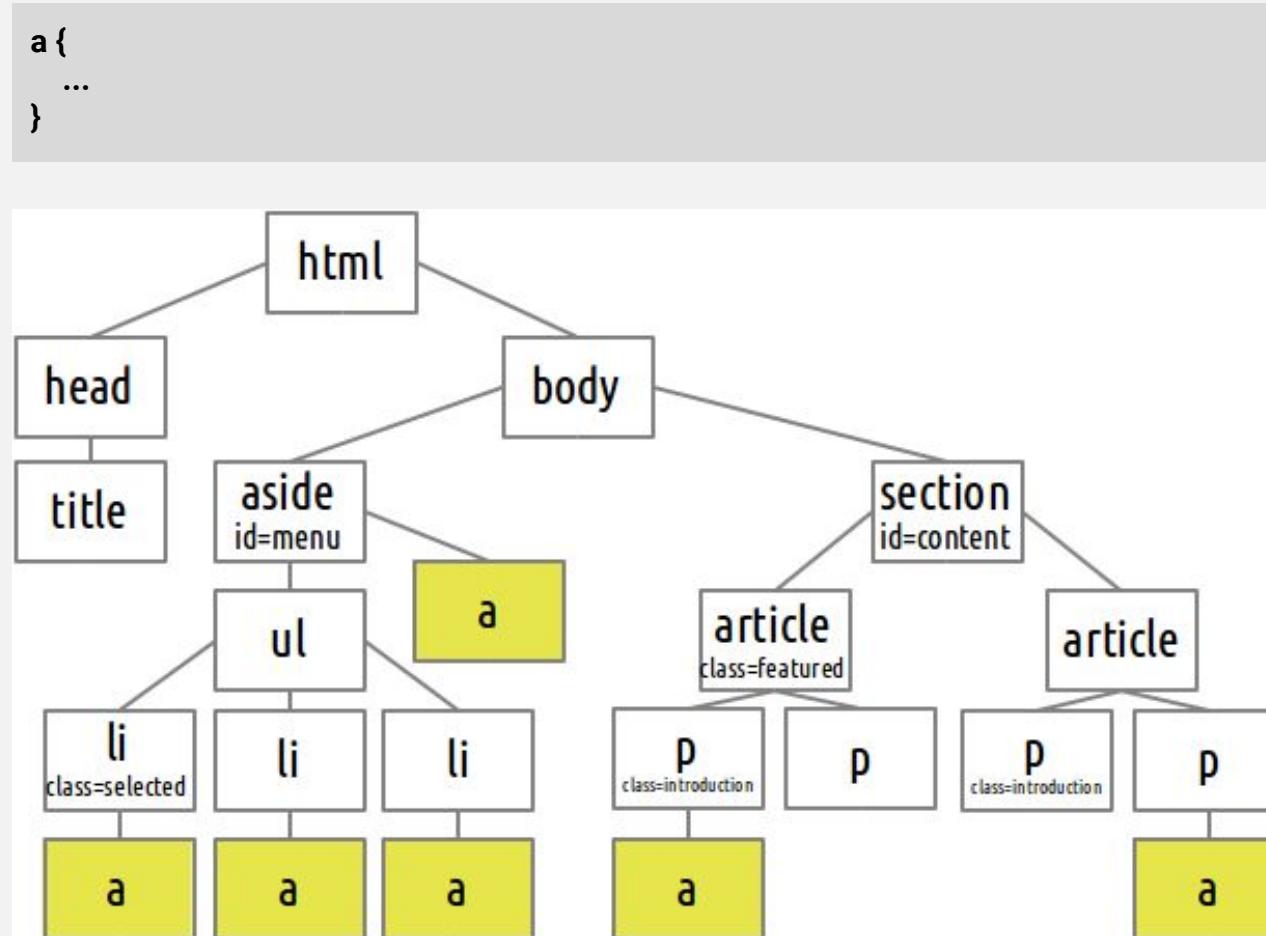
```
/* Seleciona todos os elementos <p> */  
p {  
    text-align: center;  
    color: red;  
}
```

- Pode-se agrupar os selectores, para minimizar o código

```
/* Seleciona todos os elementos <h1> e <h2> e <p> */  
h1, h2, p {  
    text-align: center;  
    color: red;  
}
```

3. Sintaxe e Selectores

- Selector por elemento



3. Sintaxe e Selectores

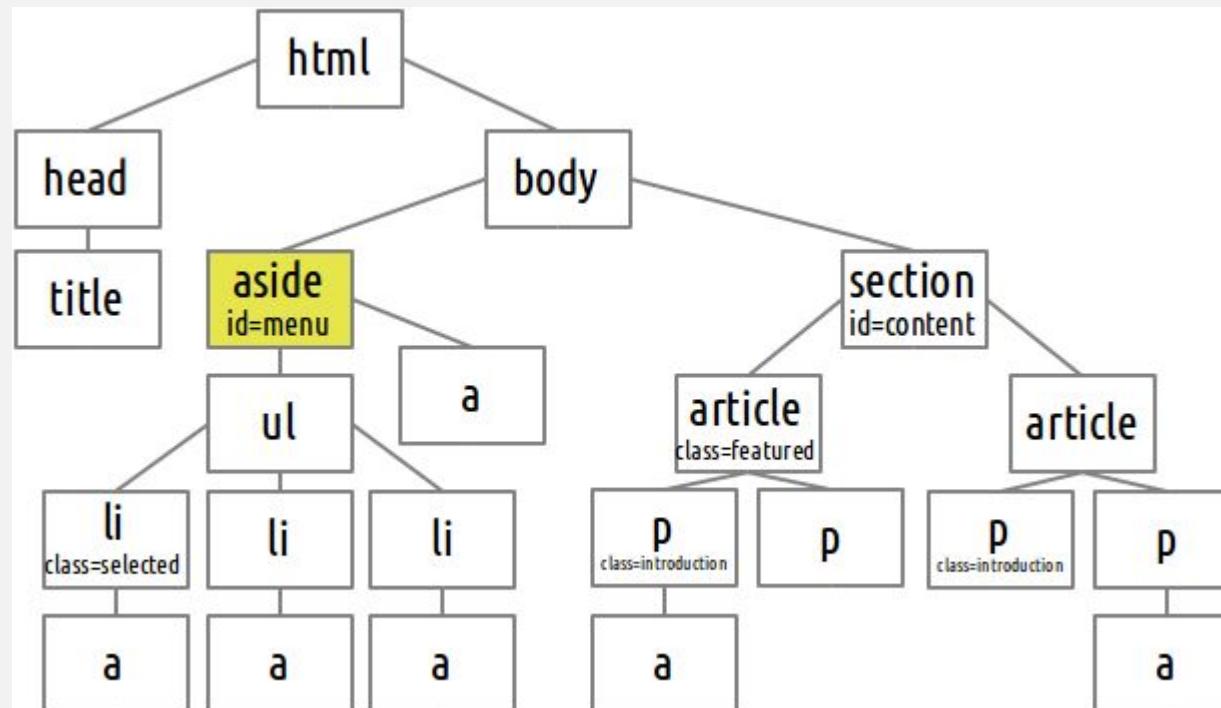
- **Selector por Id**
 - Selecciona elementos com base no atributo **Id** de um elemento HTML
 - A **seleção é unívoca** pois os identificadores de elementos não se repetem

```
/* Seleciona o elemento com o atributo id com o valor "p1" */  
#p1 {  
    background-color: #d0e4fe;  
}  
  
<!-- HTML -->  
<p id="p1">Ola Mundo!!</p>  
<p>Ola Portugal!!</p>
```

3. Sintaxe e Selectores

- Selector por Id

```
#menu {  
  ...  
}
```



3. Sintaxe e Selectores

- **Selector por classe**
 - Seleciona elementos com base no atributo **class** de um elemento HTML
 - Podem ser seleccionados vários elementos diferentes que possuem a mesma classe

```
/* Selecciona todos os elementos com o atributo class com o valor "center" */  
.center {  
    text-align: center;  
    color: red;  
}  
  
<!-- HTML -->  
<h1 class="center">Sou um h1</h1>  
<p class="center">Sou um p</p>
```

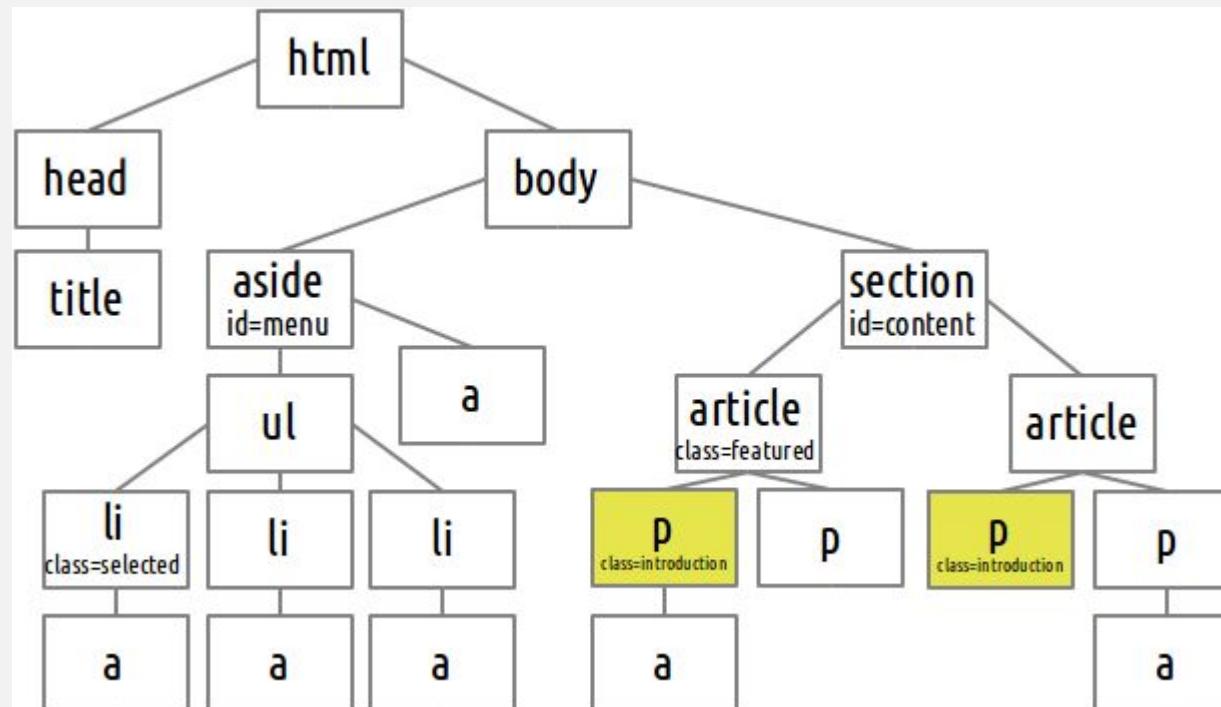
- Também se pode definir que apenas elementos HTML específicos deverão ser afectados por uma classe

```
/* Selecciona todos os elementos <p> com o atributo class com o valor "center" */  
p.center {  
    text-align: center;  
    color: red;  
}
```

3. Sintaxe e Selectores

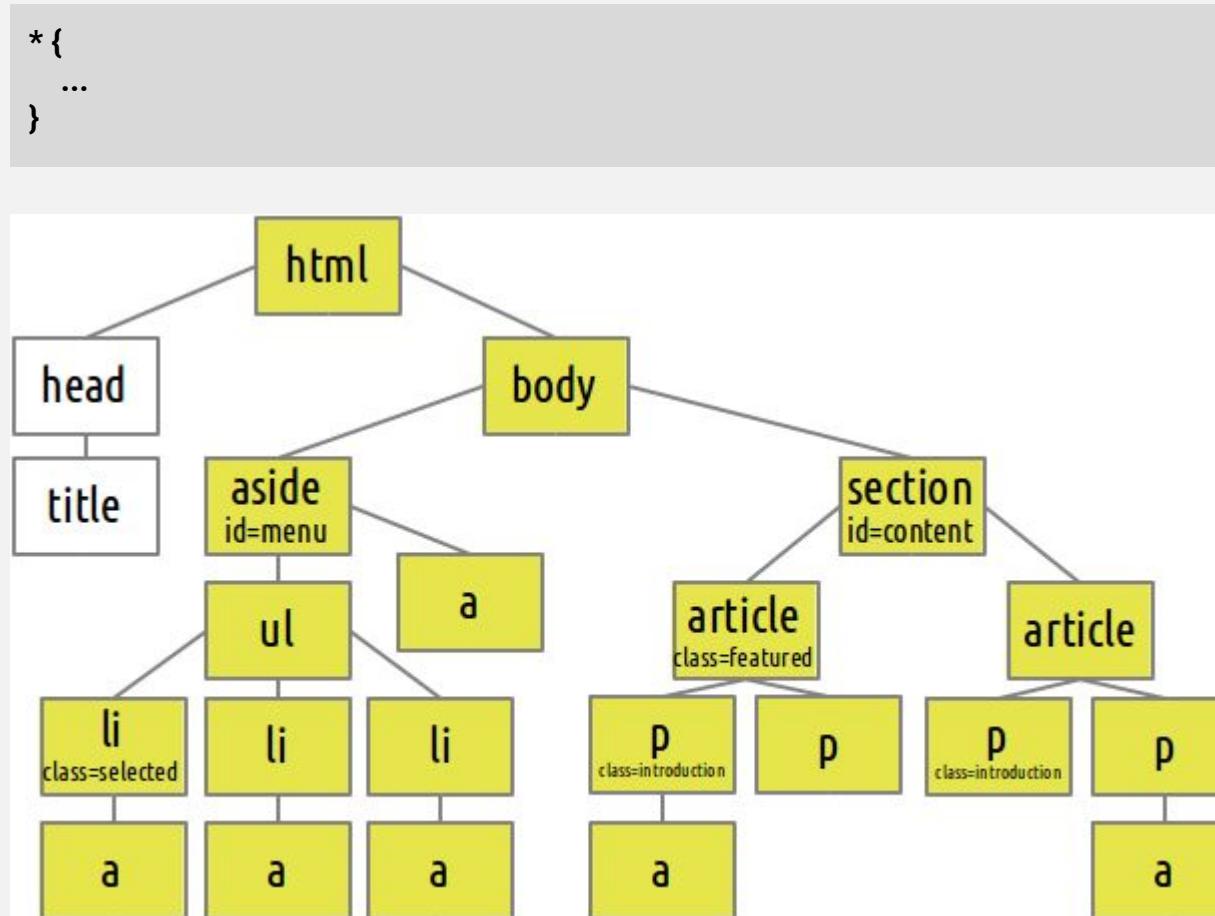
- Selector por classe

```
.introduction {  
    ...  
}
```



3. Sintaxe e Selectores

- Seleccionar todos os elementos



3. Sintaxe e Selectores

- **Selector por atributo**

- Estilizar elementos HTML com atributos específicos ou valores de atributos
- Selector [attribute] usado para selecionar elementos que possuem um atributo específico

```
/* Seleciona todos os <a> que tenham um atributo target */  
a[target] {  
    background-color: yellow;  
}
```

- Selector [attribute="value"] usado para seleccionar elementos com atributos e valores específicos

```
/* Seleciona todos os <a> que tenham um atributo target com o valor “_blank” */  
a[target="_blank"] {  
    background-color: yellow;  
}
```

3. Sintaxe e Selectores

- **Selector por atributo**

- {element}[attribute^="value"]
 - Elemento cujo valor do atributo attribute começa por value;
- {element}[attribute\$="value"]
 - Elemento cujo valor do atributo attribute termina em value;
- {element}[attribute*= "value"]
 - Elemento cujo valor do atributo attribute contém value;

```
/* Seleciona todos os <a> que tenham um atributo target com o valor a começar com "_bl" (_blank é um exemplo disto) */  
a[target^="_bl"] {  
    background-color: yellow;  
}
```

```
/* Seleciona todos os <a> que tenham um atributo target com o valor a terminar com "ank" (_blank é um exemplo disto) */  
a[target$="ank"] {  
    background-color: yellow;  
}
```

```
/* Seleciona todos os <a> que tenham um atributo target com o valor contém com "blank" (_blank é um exemplo disto) */  
a[target*="blank"] {  
    background-color: yellow;  
}
```

3. Sintaxe e Selectores

- **Combinadores**

- Uma regra CSS pode conter mais de que um selector
- Entre os selectores simples podemos incluir um **combinador**
- Existem 4 combinadores diferentes em CSS3:
 - Seletor descendentes (espaço)
 - Seletor filho direto (>)
 - Seletor irmão direto (+)
 - Seletor irmão geral (~)

```
/* Seleciona todos os <p> filhos diretos de <div> */  
div > p {  
    background-color: yellow;  
}
```

3. Sintaxe e Selectores

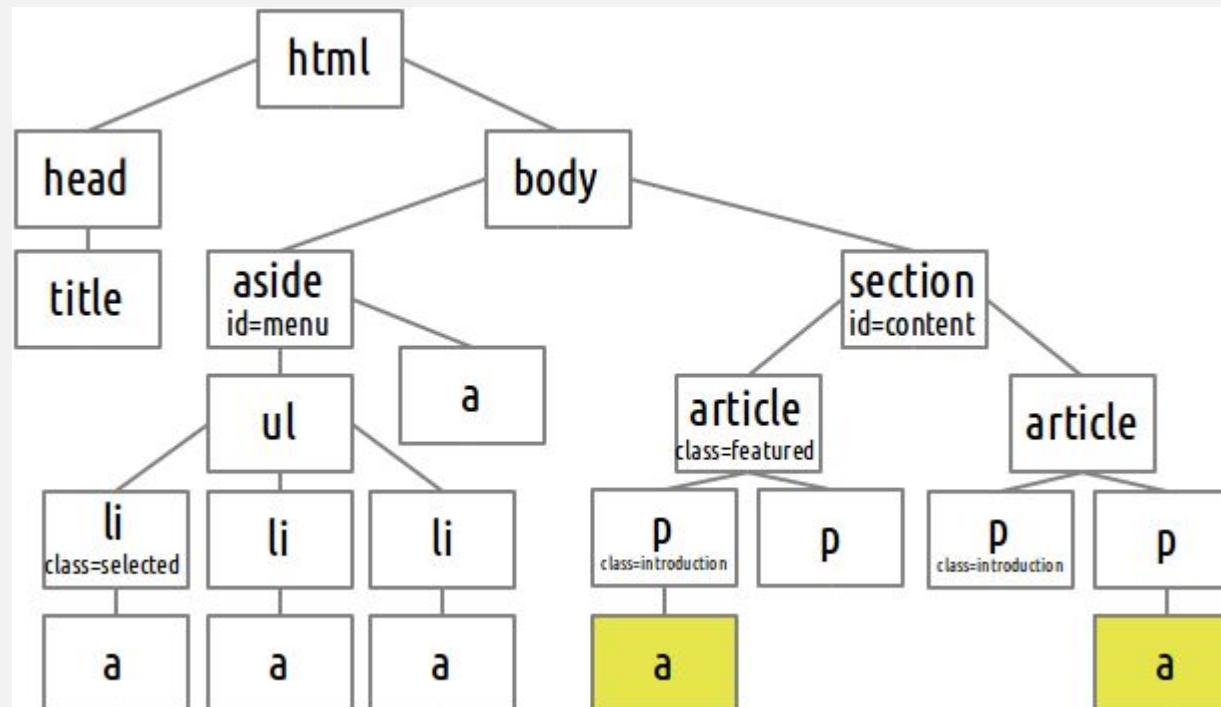
- **Combinadores**

Selector	Exemplo	Descrição
{elemento} {elemento}	div p	Seleciona todos os elementos <p> dentro dos elementos <div>
{elemento}>{elemento}	div>p	Seleciona todos os elementos <p> em que o pai é um elemento <div>
{elemento}+{elemento}	div+p	Seleciona todos os elementos <p> que são colocados imediatamente após os elementos <div>
{elemento}~{elemento}	p~ul	Seleciona todos os elementos precedidos por um elemento <p>

3. Sintaxe e Selectores

- Selector descendentes

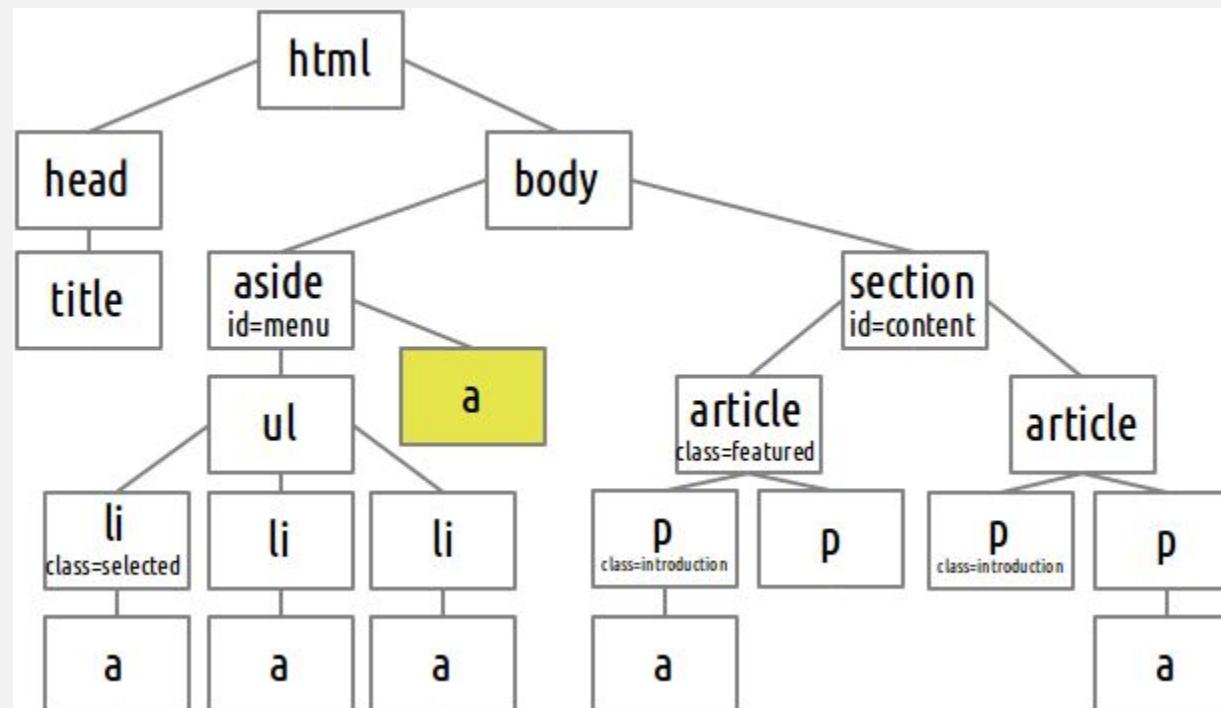
```
article a {  
    ...  
}
```



3. Sintaxe e Selectores

- Seletor filho direto

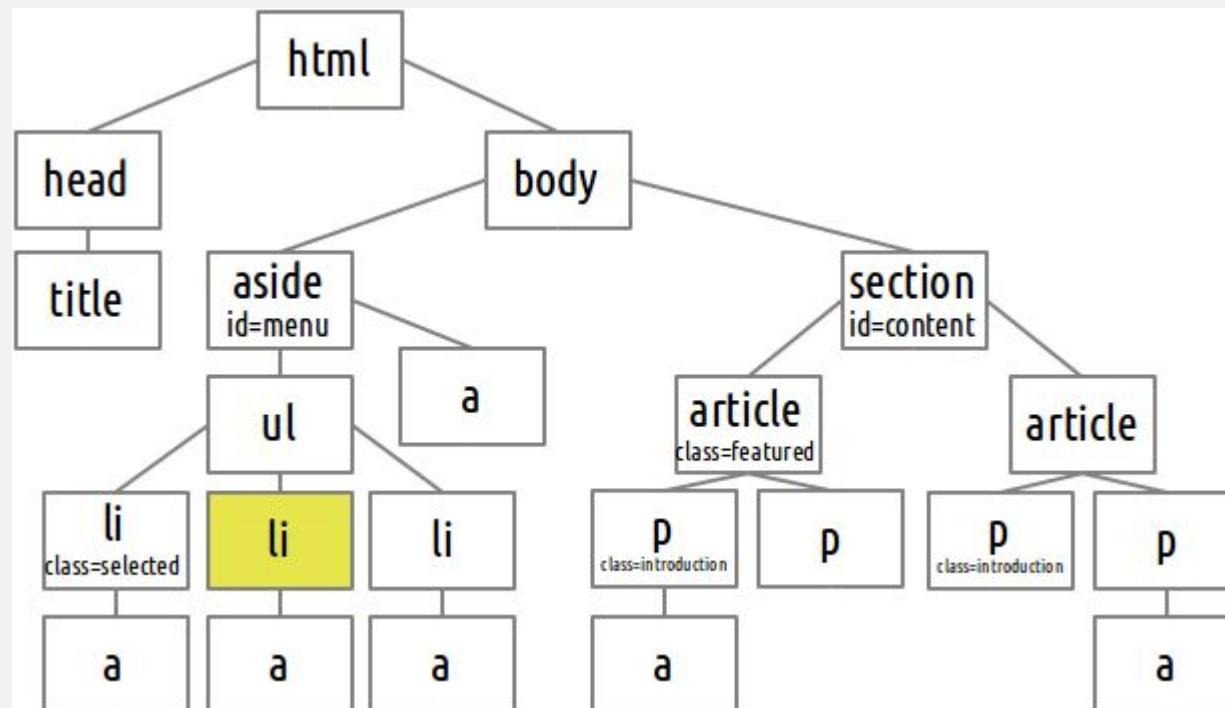
```
aside>a {  
    ...  
}
```



3. Sintaxe e Selectores

- Seletor irmão direto

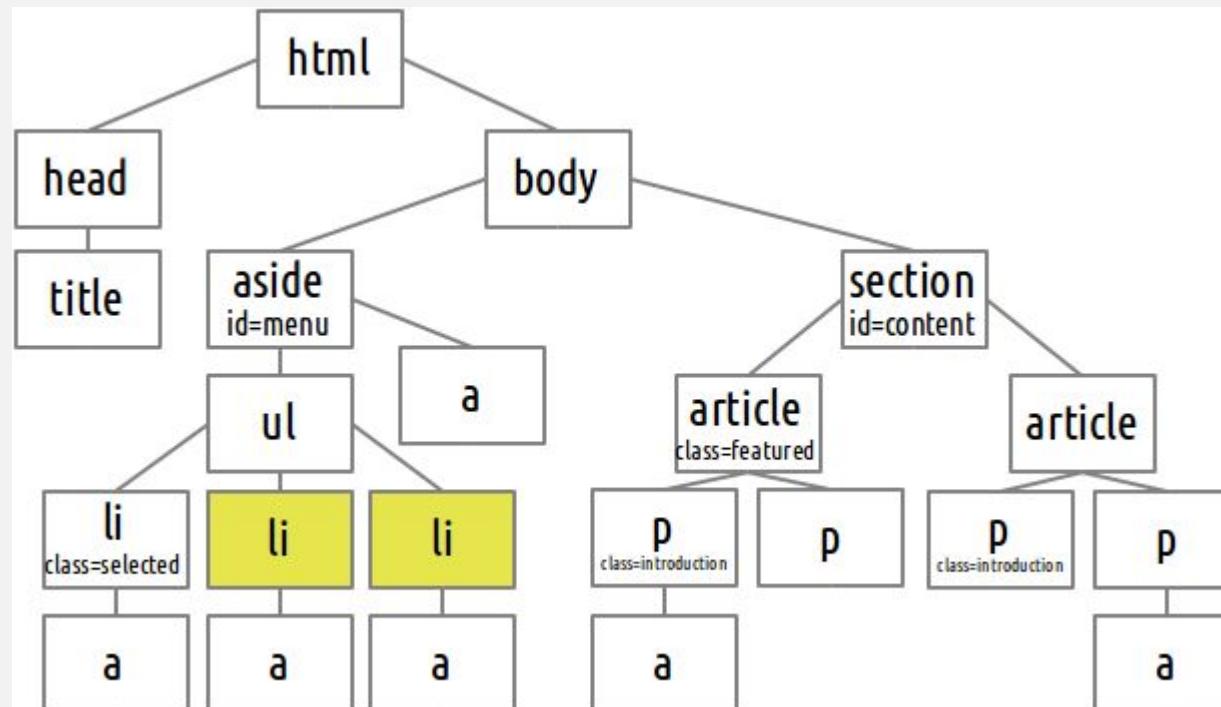
```
.selected+li {  
    ...  
}
```



3. Sintaxe e Selectores

- Seletor irmão geral

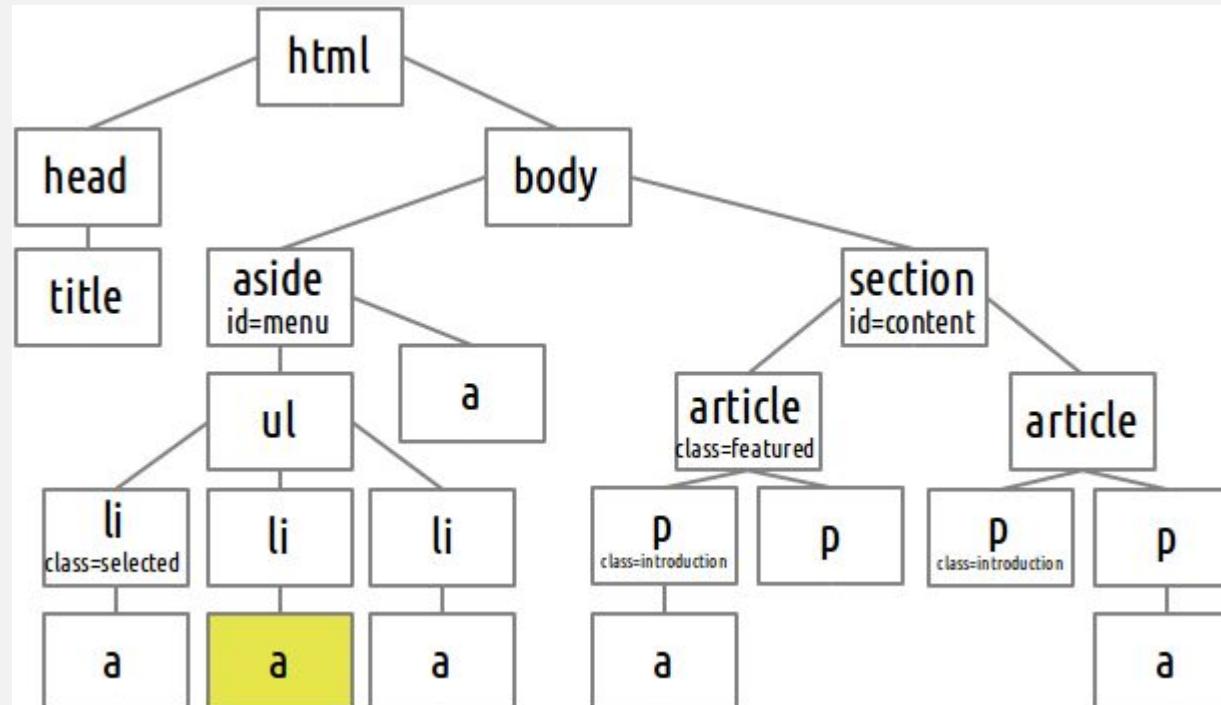
```
.selected~li {  
    ...  
}
```



3. Sintaxe e Selectores

- Os conceitos podem ser combinados para formar selectores complexos

```
aside#menu li.selected+li>a {  
    ...  
}
```



3. Sintaxe e Selectores

- **Pseudo-classes**

- As **pseudo-classes** permitem formatar elementos baseando-se em informação que **não está disponível na árvore do documento (DOM)**, utilizando informação para além das características habituais.
- Por exemplo, pode ser usado para:
 - Estilizar um elemento quando o utilizador passa o rato sobre um elemento
 - Estilizar links visitados e não-visitados de forma diferente

```
/* link nao-visitado */  
a:link {  
    color: #FF0000;  
}
```

```
/* link visitado */  
a:visited {  
    color: #00FF00;  
}
```

```
/* mouse over no link */  
a:hover {  
    color: #FF00FF;  
}
```

```
/* link selecionado */  
a:active {  
    color: #0000FF;  
}
```

3. Sintaxe e Selectores

- **Pseudo-classes [nth-*]**

- {element}:nth-child(n)
 - Elemento “element” filho número n de um “parent”
- {element}:nth-last-child(n)
 - Elemento “element” filho número n de um “parent”, a contar do último
- {element}:nth-of-type(n)
 - Elemento “element”, número n dos seus irmãos
- {element}:nth-last-of-type(n)
 - Elemento “element”, número n dos seus irmãos, contar do último

```
/* Selecciona todos os <p> que são os segundos filhos */  
p:nth-child(2) {  
    background: red;  
}  
/* Selecciona todos os <p> que são os segundos filhos, a contar do último filho */  
p:nth-last-child(2) {  
    background: red;  
}  
/* Selecciona todos os <div> que são os segundos filhos */  
div:nth-of-type(2) {  
    background: red;  
}  
/* Selecciona todos os <p> que são os segundos filhos, a contar do último filho */  
p:nth-last-of-type(2) {  
    background: red;  
}
```

3. Sintaxe e Selectores

- **Pseudo-classes [*-child e empty]**
 - {element}:last-child
 - Elemento “element” que seja o último filho do seu “parent”
 - {element}:only-child
 - Elemento “element” que não tenha irmãos
 - {element}:empty
 - Elemento “element” vazio

```
/* Selecciona todos os <p> que são o último filho */  
p:last-child {  
    background: #ff0000;  
}  
/* Selecciona todos os <p> que são o único filho */  
p:only-child {  
    background: #ff0000;  
}  
/* Selecciona todos os <p> sem conteúdo ou vazios */  
p:empty {  
    background: #ff0000;  
}
```

3. Sintaxe e Selectores

- **Pseudo-classes [*-of-type]**
 - {element}:first-of-type
 - Elemento “element” primeiro do seu tipo
 - {element}:last-of-type
 - Elemento “element” último do seu tipo
 - {element}:only-of-type
 - Elemento “element” único desse tipo dentro do seu parent

```
/* Seleciona todos os <p> que são o primeiro elemento do seu “parent” */  
p:first-of-type {  
    background: red;  
}  
/* Seleciona todos os <p> que são o último elemento do seu “parent” */  
p:last-of-type {  
    background: red;  
}  
/* Seleciona todos os <p> que são o único elemento desse tipo do seu “parent” */  
p:only-of-type {  
    background: #ff0000;  
}
```

3. Sintaxe e Selectores

- **Pseudo-classes [target, enabled, disabled e checked]**
 - {element}:target
 - Elemento “element” que é o target de um fragmento (#)
 - {element}:enabled
 - Elemento “element” que está enabled
 - {element}:disabled
 - Elemento “element” que está disabled
 - {element}:checked
 - Elemento “element” que está checked (radio + checkboxes)

```
/* Destacar a âncora HTML activa */
:target {
    border: 2px solid #D4D4D4; background-color: #e5eecc;
}

/* Definir uma cor de fundo para todos os elementos <input> do tipo="texto" que estão "enabled" ou activos */
input[type="text"]:enabled {
    background: #ffff00;
}

/* Definir uma cor de fundo para todos os elementos <input> do tipo="texto" que estão "disabled" */
input[type="text"]:disabled {
    background: #dddddd;
}

/* Definir a altura e a largura para todos os elementos <input> que estão no estado "checked" */
input:checked {
    height: 50px; width: 50px;
}
```

3. Sintaxe e Selectores

- **Pseudo-classes [not]**
 - {element}:not({selector})
 - Elemento “element” que não tem o selector

```
/* Selecionar todos os elementos que não são do tipo <p> */  
:not(p) {  
    background: #ff0000;  
}  
/* Selecionar todos os elementos que não têm a classe “negative” */  
:not(.negative) {  
    background: #ff0000;  
}
```

3. Sintaxe e Selectores

- **Pseudo-elementos**
 - Usado para estilizar partes de um elemento
 - Por exemplo, ele pode ser usado para:
 - Estilizar a primeira linha ou letra de um elemento
 - Inserir conteúdo antes/depois do conteúdo de um elemento

```
/* primeira letra de um paragrafo */  
p::first-letter {  
    color: #ff0000;  
    font-size: xx-large;  
}  
  
/* insere uma imagem antes de um elemento h1 */  
h1::before {  
    content: url(smiley.gif);  
}
```

3. Sintaxe e Selectores

- **Comentários**

- São delimitados por /* e */
- Devem ser usados para documentação
- Podem ser usados para inativação

```
/* primeira letra de um paragrafo */
p::first-letter {
    color: #ff0000;
    font-size: xx-large;
}

/* insere uma imagem antes de um elemento h1 */
h1::before {
    content: url(smiley.gif);
}
```

Módulo 3 - CSS

4. Dimensões

4. Dimensões

- **Width e Height**

- Definem a largura e altura de um elemento. Os valores podem ser um comprimento, uma percentagem ou auto.

```
div {  
    width: 50%;  
    height: 200px;  
}
```

- Auto é o valor por omissão

4. Dimensões

○ Minimum e Maximum

- Definem a largura e altura mínima e máxima de um elemento. Os valores podem ser um comprimento, uma percentagem ou none.

```
div {  
    max-width: 800px;  
    min-height: none;  
}
```

- None é o valor por omissão

Módulo 3 - CSS

5. Unidades de Medida

5. Unidades de Medida

- **Medidas absolutas**

- As unidades de medida absolutas representam uma medida física. Eles são úteis quando as propriedades físicas do meio de saída são conhecidas, como para o layout de impressão.

mm, cm, in, pt and pc

- **mm:** 1 milímetro
- **cm:** 1 centímetro (10 milímetros)
- **in:** 1 polegada (2.54 centímetros)
- **pt:** 1 ponto (1/72 de uma polegada)
- **pc:** 1 pica (12 pontos)

5. Unidades de Medida

- **Unidades de medida relativa das fontes**
 - São relativas ao tamanho de um determinado caractere ou atributo de fonte na fonte atualmente em vigor no elemento (ou elemento pai em alguns casos).
 - São úteis quando as propriedades físicas do meio de saída são desconhecidas, como para o layout do ecrã.
 - **rem** Representa o tamanho da fonte do elemento raiz. Se usado no elemento raiz, representa o valor inicial (omissão) do navegador (normalmente 16px).
 - **em** Quando usado com font-size, representa o tamanho da fonte do elemento pai. Para comprimentos, representa o tamanho da fonte do elemento atual.

5. Unidades de Medida

- Exemplo (rem)

```
<div>
  <p>Some text</p>
  <div>
    <p>Some more text</p>
  </div>
</div>
```

```
div {
  font-size: 1.2rem;
}
```

Some text

Some more text

5. Unidades de Medida

- Exemplo (rem)

```
<div>
  <p>Some text</p>
  <div>
    <p>Some more text</p>
  </div>
</div>
```

```
div {
  font-size: 1.2em;
}
```

Some text

Some more text

5. Unidades de Medida

- **Pixel**
 - Em ecrans com baixo dpi, o pixel (px) representa um pixel do dispositivo.
 - Em dispositivos com dpi mais altos, um pixel representa um número inteiro de pixels do dispositivo de modo que $1\text{in} \approx 96\text{px}$.

5. Unidades de Medida

- **Percentagem**
 - A percentagem do tipo de dados CSS representa um valor percentual. Uma percentagem consiste num número seguido pelo sinal %. Não há espaço entre o símbolo e o número.

```
div {  
    width: 50%;  
}
```

- Muitas propriedades CSS (width, margin, padding, font-size, ...) podem usar valores percentuais para definir um tamanho em relação ao seu objeto pai.

Módulo 3 - CSS

6. Box Model

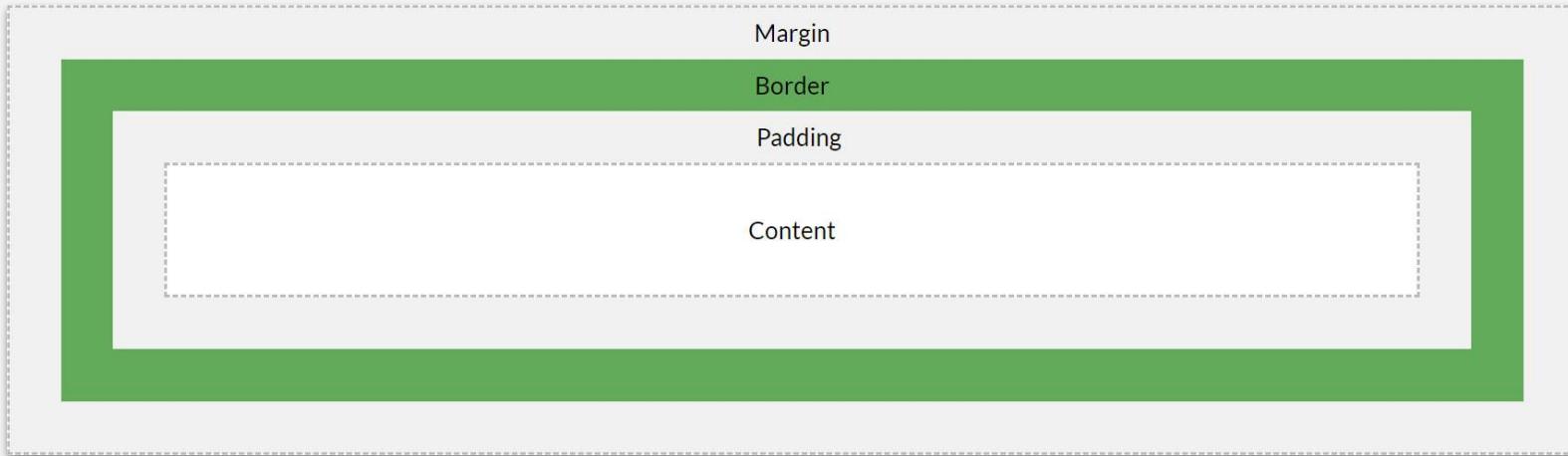
6. Box Model

- Qualquer elemento em web design é **rectangular**
- A largura de um elemento é a soma da largura do seu conteúdo, com os “paddings” laterais e os “borders” laterais
- **Problema:** definir de forma inteligente a largura de um elemento sem fazer demasiados cálculos
- Solução: **Box Model!**

6. Box Model

- O **Box Model** CSS é essencialmente uma **caixa** que envolve todos os **elementos** HTML
- **Consiste em:**
 - **Content:** conteúdo da caixa, onde aparecem texto e imagem.
 - **Padding:** define uma área ao redor do conteúdo, o preenchimento desta área é transparente.
 - **Border:** Um contorno que circunda o padding e o conteúdo.
 - **Margin:** define uma área fora do contorno, esta área é transparente.

6. Box Model



```
div {  
    width: 300px;  
    border: 25px solid green;  
    padding: 25px;  
    margin: 25px;  
}
```

6. Box Model

- Quando se definem as propriedades de “width” e “height” de um elemento, apenas se define a largura e altura da área do conteúdo.
- Para se calcular o tamanho total de um elemento é necessário adicionar padding, borders e margins.

```
div {  
    width: 320px;  
    padding: 10px;  
    border: 5px solid gray;  
    margin: 0;  
}
```

320px (width)
+ 20px (left + right padding)
+ 10px (left + right border)
+ 0px (left + right margin)
= **350px**

6. Box Model

- Cálculo da largura e altura de um elemento:
 - **Largura total** de um elemento:
 - width + left padding + right padding + left border + right border + left margin + right margin
 - **Altura total** de um elemento:
 - height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

Módulo 3 - CSS

7. Borders

7. Borders

- A propriedade **border-style** define o tipo de border:
 - Dotted
 - Dashed
 - Solid
 - Double
 - Groove
 - Ridge
 - Inset
 - Outset
 - None
 - Hidden

7. Borders

A dotted border.

A dashed border.

A solid border.

A double border.

A groove border. The effect depends on the border-color value.

A ridge border. The effect depends on the border-color value.

An inset border. The effect depends on the border-color value.

An outset border. The effect depends on the border-color value.

No border.

A hidden border.

A mixed border.

7. Borders

- **Outros tipos de Border:**
 - **border-image**
 - Permite usar uma imagem como border
 - **border-radius**
 - Permite colocar os cantos redondos
 - **box-shadow**
 - Permite criar sombras nos elementos

7. Borders

- **Border-image**

border-image: url() largura altura tipo

```
#border-img {  
    border-image: url(border.png) 30 round;  
}
```

- **repeat**
 - A imagem cortada e repetida a partir do centro
- **round**
 - A imagem é cortada e ajustada para ficar sempre bem no início e no fim
- **stretch**
 - A imagem é simplesmente esticada

7. Borders

○ Border-radius

```
#example1 {  
    border: 2px solid red;  
    padding: 10px;  
    border-radius: 25px;  
}
```

border-radius: 25px;

The border-radius property defines the radius of the element's corners.

- **border-top-right-radius**
 - radius do canto superior direito
- **border-bottom-right-radius**
 - radius do canto inferior direito
- **border-bottom-left-radius**
 - radius do canto inferior esquerdo
- **border-top-left-radius**
 - radius do canto superior esquerdo

7. Borders

○ Box-shadow

```
box-shadow: offset-x offset-y blur-radius spread-radius color type
```

- Pode-se definir o desvio em xx e yy do centro da sombra.
- **blur-radius**
 - Raio do blur, quanto maior o valor, maior e mais clara fica a sombra. 0 é parecido com border.
- **spread-radius**
 - Raio de expansão da sombra
- **type**
 - outset/inset: sombra para dentro ou para fora

7. Borders

○ Box-shadow

```
#example1 {  
    border: 1px solid;  
    padding: 10px;  
    box-shadow: 5px 10px;  
}  
#example2 {  
    border: 1px solid;  
    padding: 10px;  
    box-shadow: 5px 10px #888888;  
}  
#example3 {  
    border: 1px solid;  
    padding: 10px;  
    box-shadow: 5px 10px red;  
}
```

box-shadow: 5px 10px:

A div element with a shadow. The first value is the horizontal offset and the second value is the vertical offset. The shadow color will be inherited from the text color.

box-shadow: 5px 10px #888888:

You can also define the color of the shadow. Here the shadow color is grey.

box-shadow: 5px 10px red:

A red shadow.

Módulo 3 - CSS

8. Texto e Fontes

8. Texto e Fontes

- A propriedade **color** é usada para definir a cor do texto. A cor pode ser especificada por:
 - nome: “red”
 - hexadecimal: “#ff0000”
 - valor rgb: “rgb(255,0,0)”

```
body {  
    color: blue;  
}  
h1 {  
    color: green;  
}
```

```
body {  
    color: #0000ff;  
}  
h1 {  
    color: #00ff00;  
}
```

```
body {  
    color: rgb(0,0,255);  
}  
h1 {  
    color: rgb(0,255,0);  
}
```

8. Texto e Fontes

- A propriedade **text-align** é usada para definir o alinhamento horizontal do texto
- O alinhamento horizontal do texto pode ser “**left**”, “**right**”, “**center**”, “**justify**”.

```
div.a {  
    text-align: center;  
}  
  
div.b {  
    text-align: left;  
}  
  
div.c {  
    text-align: right;  
}  
  
div.d {  
    text-align: justify;  
}
```

text-align: center:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

text-align: left:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

text-align: right:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

text-align: justify:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

8. Texto e Fontes

- A propriedade **text-decoration** é usada para definir ou remover elementos decorativos do texto
- O valor “**none**” do atributo text-decoration é usado com frequência para remover o efeito sublinhado dos links

```
h1 {  
    text-decoration: none;  
}  
  
h2 {  
    text-decoration: overline;  
}  
  
h3 {  
    text-decoration: line-through;  
}  
  
h4 {  
    text-decoration: underline;  
}
```

This is heading 1

This is heading 2

~~This is heading 3~~

This is heading 4

8. Texto e Fontes

- A propriedade **text-transform** é usada para especificar letras maiúsculas e minúsculas num texto
- Pode ser usado para transformar tudo em letras maiúsculas ou minúsculas, ou capitalizar a primeira letra de cada palavra.

```
div.a {  
    text-transform: uppercase;  
}
```

```
div.b {  
    text-transform: lowercase;  
}
```

```
div.c {  
    text-transform: capitalize;  
}
```

text-transform: uppercase:

LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT.

text-transform: lowercase:

lorem ipsum dolor sit amet, consectetur adipiscing elit.

text-transform: capitalize:

LoREM IpsuM DoloR SiT AmEt, CoNSeCTetuR AdiPiScInG ElIt.

8. Texto e Fontes

- A propriedade **text-indent** é usada para especificar o recuo da primeira linha de um texto

```
div.a {  
    text-indent: 50px;  
}
```

```
div.c {  
    text-indent: 30%;  
}
```

text-indent: 50px:

Lore ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

text-indent: 30%:

Lore ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.

8. Texto e Fontes

- A propriedade **letter-spacing** é usada para especificar o espaço entre os caracteres em um texto

```
h1 {  
    letter-spacing: 3px;  
}  
  
h2 {  
    letter-spacing: 2px;  
}  
  
h3 {  
    letter-spacing: -1px;  
}
```

This is heading 1

This is heading 2

This is heading 3

8. Texto e Fontes

- A propriedade **line-height** é usada para especificar o espaço entre linhas

```
div.a {  
    line-height: normal;  
}  
  
div.b {  
    line-height: 1.6;  
}  
  
div.c {  
    line-height: 80%;  
}  
  
div.d {  
    line-height: 200%;  
}
```

line-height: normal (default):

This is a paragraph with a standard line-height.
The standard line height in most browsers is about 110% to 120%.

line-height: 1.6 (recommended):

This is a paragraph with the recommended line-height.
The line height is here set to 1.6. This is a unitless value;
meaning that the line height will be relative to the font size.

line-height: 80%:

This is a paragraph with a smaller line-height.
The line height is here set to 80%.

line-height: 200%:

This is a paragraph with a bigger line-height.
The line height is here set to 200%.

8. Texto e Fontes

- A propriedade **word-spacing** é usada para especificar o espaço entre as palavras num texto

```
p.a {  
    word-spacing: normal;  
}
```

```
p.b {  
    word-spacing: 30px;  
}
```

```
p.c {  
    word-spacing: 1cm;  
}
```

word-spacing: normal:

This is some text. This is some text.

word-spacing: 30px:

This is some text. This is some text.

word-spacing: 1cm:

This is some text. This is some text.

8. Texto e Fontes

- A propriedade **text-shadow** adiciona sombra a um texto

```
text-shadow: h-shadow v-shadow blur-radius color|none|initial|inherit
```

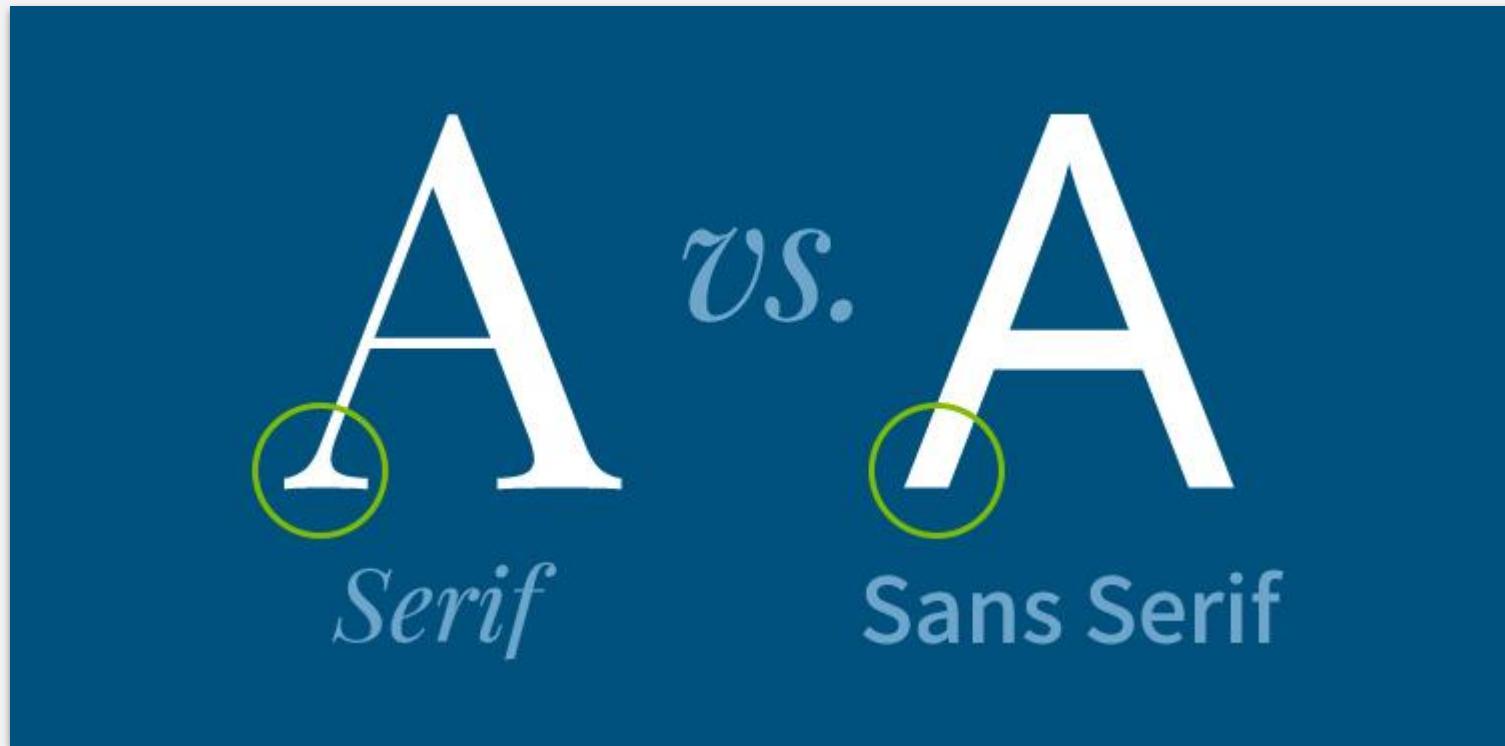
```
h1 {  
    text-shadow: 2px 2px #FF0000;  
}
```

The **text-shadow** Property

Note: Internet Explorer 9 and earlier do not support the text-shadow property.

8. Texto e Fontes

- Fontes: **Serif** e **Sans Serif**



8. Texto e Fontes

○ **font-family**

- A propriedade deve conter vários nomes de fonte como um sistema de "fallback"
- Se o navegador não suportar a primeira fonte, ele tentará a próxima fonte e assim por sucessivamente.
- Comece com a fonte desejada e termine com uma família genérica, para permitir que o navegador escolha uma fonte semelhante na família genérica, se nenhuma outra fonte estiver disponível.

```
p.a {  
    font-family: "Times New Roman", Times, serif;
```

```
}
```

```
p.b {  
    font-family: Arial, Helvetica, sans-serif;
```

```
}
```

This is a paragraph, shown in the Times New Roman font.

This is a paragraph, shown in the Arial font.

8. Texto e Fontes

- A propriedade **font-style** é usada principalmente para especificar texto em itálico
- A propriedade tem 3 valores:
 - **normal**
 - **italic**
 - **oblique** (muito similar a italic mas menos suportado)

```
p.normal {  
    font-style: normal;  
}  
  
p.italic {  
    font-style: italic;  
}  
  
p.oblique {  
    font-style: oblique;  
}
```

This is a paragraph, normal.

This is a paragraph, italic.

This is a paragraph, oblique.

8. Texto e Fontes

- A propriedade **font-size** define o tamanho do texto
- Ser capaz de **gerir o tamanho do texto é importante** no design da web, no entanto, não deverá ser usado para fazer com que os parágrafos pareçam cabeçalhos, ou o contrário.
- **Use as tags HTML adequadas**, como `<h1>` - `<h6>` para títulos e `<p>` para parágrafos.

```
div.a {  
    font-size: 15px;  
}  
  
div.b {  
    font-size: large;  
}  
  
div.c {  
    font-size: 150%;  
}
```

This is some text.
This is some text.
This is some text.

8. Texto e Fontes

- A propriedade **font-weight** define como os caracteres devem ser exibidos em termos de espessura

```
p.normal {  
    font-weight: normal;  
}  
  
p.light {  
    font-weight: lighter;  
}  
  
p.thick {  
    font-weight: bold;  
}  
  
p.thicker {  
    font-weight: 900;  
}
```

This is a paragraph.
This is a paragraph.
This is a paragraph.
This is a paragraph.

8. Texto e Fontes

- **Tipos de Fontes**
 - **TrueType Fonts (TTF)**
 - TrueType é um padrão de fonte desenvolvido no final dos anos 80, pela Apple e pela Microsoft. TrueType é o formato de fonte mais comum para os sistemas operacionais Mac OS e Microsoft Windows.
 - **OpenType Fonts (OTF)**
 - OpenType é um formato para fontes de computador escaláveis. Foi baseado em TrueType e é uma marca registada da Microsoft. Fontes OpenType são usadas hoje em dia nas principais plataformas de computador.
 - **The Web Open Font Format (WOFF)**
 - WOFF é um formato de fonte para uso em páginas da web. Foi desenvolvido em 2009 e agora é uma recomendação do W3C. O WOFF é essencialmente OpenType ou TrueType com compactação e metadados adicionais.

8. Texto e Fontes

- **Tipos de Fontes**
 - **The Web Open Font Format (WOFF 2.0)**
 - Fonte TrueType / OpenType que fornece melhor compressão que WOFF 1.0.
 - **SVG Fonts/Shapes**
 - As fontes SVG permitem que o SVG seja usado como glifos ao exibir texto.
 - **Embedded OpenType Fonts (EOT)**
 - As fontes EOT são uma forma compacta de fontes OpenType projetadas pela Microsoft.

8. Texto e Fontes

- Usar uma fonte
 - A regra **@font-face** define um nome para a fonte (por exemplo, myFirstFont) e, em seguida, aponta para o arquivo da fonte.

```
@font-face {  
    font-family: myFirstFont;  
    src: url(sansation_light.woff);  
}
```

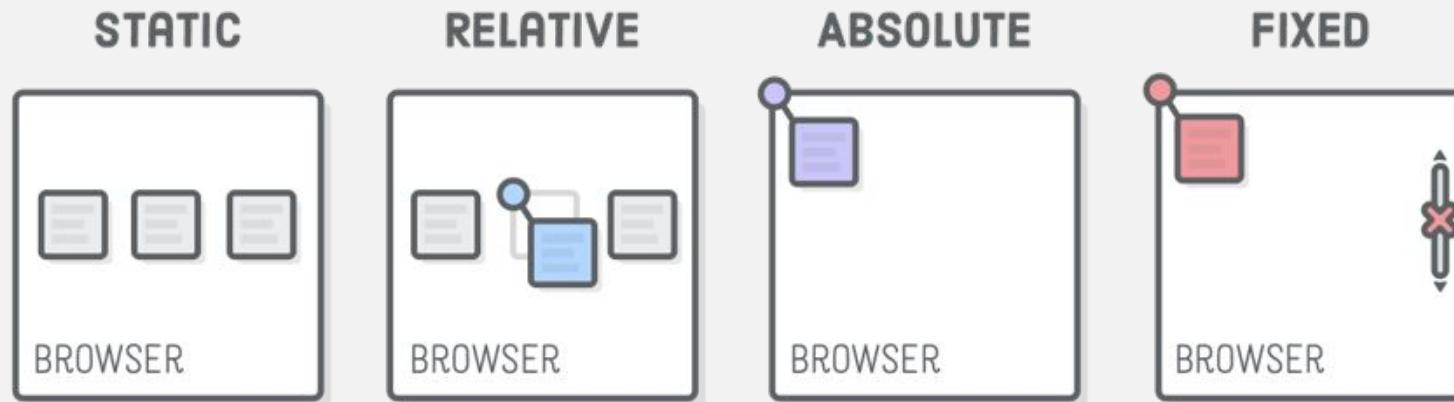
```
div {  
    font-family: myFirstFont;  
}
```

Módulo 3 - CSS

9. Posicionamento de Elementos

9. Posicionamento de Elementos

- **Introdução**
 - “Static positioning” refere-se ao fluxo normal da página
 - Os esquemas de layout CSS Box Model, e flexbox operam nesse fluxo.
 - Porém esse não é o único esquema de posicionamento disponível em CSS



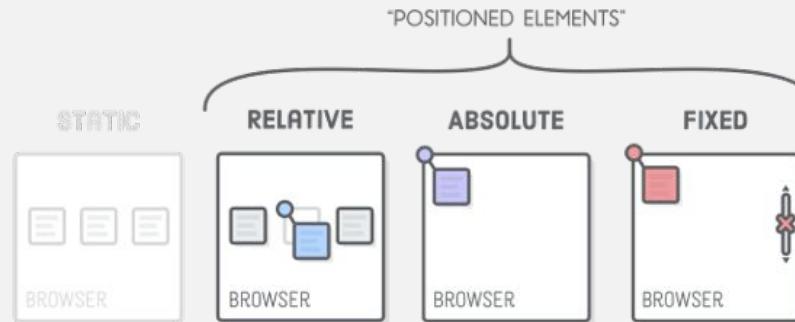
9. Posicionamento de Elementos

- **Introdução**
 - Outros Tipos de posicionamento:
 - **Relative**
 - **Absolute**
 - **Fixed**
 - Este tipo de posicionamento avançado permite fazer coisas como:
 - “Colocar o <div> 20 pixels acima e 50 pixels à direita da origem do seu elemento pai.”
 - A grande maioria dos elementos numa página da Web deve ser organizada de acordo com o fluxo estático da página (**static**).

9. Posicionamento de Elementos

- **Propriedade Position**

- A propriedade CSS **position** permite alterar o esquema de posicionamento de um elemento específico
- O valor por omissão é **static**



```
.item {  
    position: static;  
}
```

9. Posicionamento de Elementos

- **Position Relative**
 - Position **relative** movimenta elementos relativamente ao local onde eles normalmente apareceriam no fluxo estático da página



RELATIVE POSITIONING

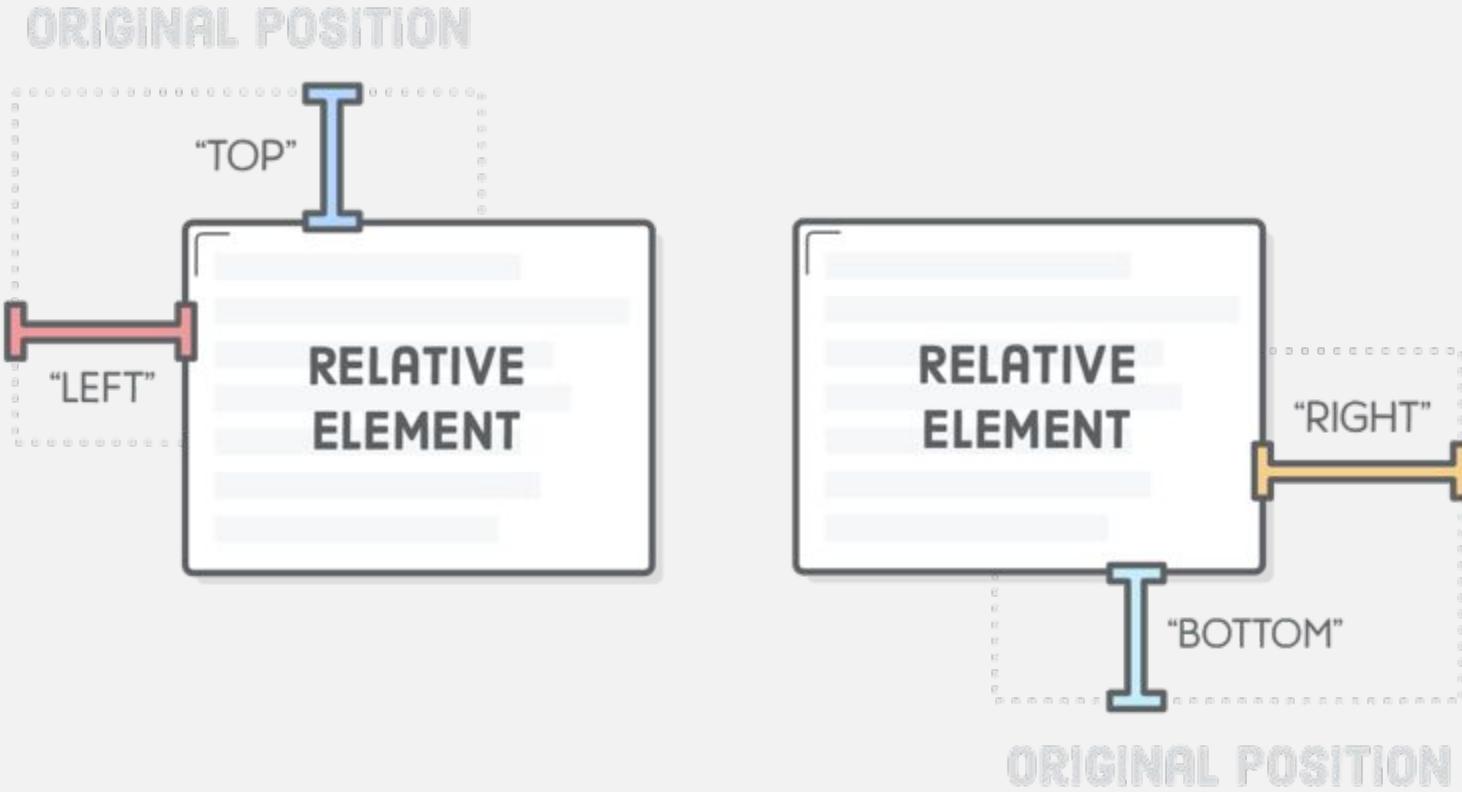
```
.item {  
  position: relative;  
}
```

9. Posicionamento de Elementos

- **Position Relative**
 - O posicionamento relativo funciona de maneira semelhante às margens, com uma diferença muito importante: nem os elementos circundantes nem o elemento pai são afetados pelos valores atribuídos.
 - Todo o resto é apresentado como se o elemento relativo estivesse na sua posição original (static)
 - Pense nos “offsets” como sendo aplicados depois do browser terminar de exibir a página

9. Posicionamento de Elementos

- **Position Relative**
 - Podemos fazer “offset” ao elemento relativo com as propriedades: **top**, **right**, **bottom**, **left**.



9. Posicionamento de Elementos

- **Position Relative**

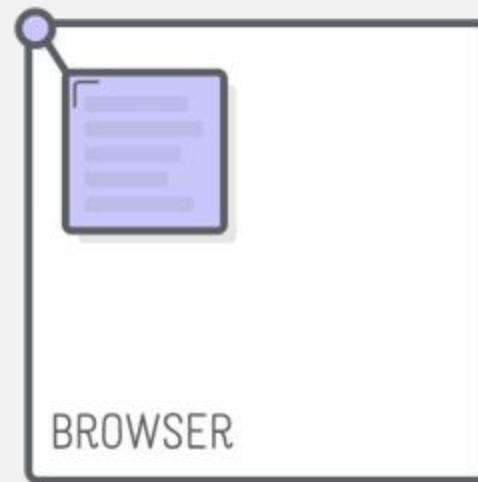
- Estas propriedades aceitam **valores negativos**, o que significa que há duas maneiras de especificar o mesmo deslocamento.
- Poderíamos facilmente usar top: -30px; no lugar do bottom: 30px; na declaração acima.

```
.item-relative {  
    position: relative;  
    right: 30px;  
    bottom: 30px;  
}
```

```
.item-relative {  
    position: relative;  
    left: -30px;  
    top: -30px;  
}
```

9. Posicionamento de Elementos

- **Position Absolute**
 - É semelhante ao posicionamento relativo, mas o deslocamento é relativo a **toda a janela do browser**.
 - Como não há qualquer relação com o fluxo estático da página, esta é a forma mais “manual” de dispor um elemento.



ABSOLUTE POSITIONING

9. Posicionamento de Elementos

- **Position Absolute**

- Remove completamente um elemento do fluxo normal da página
- Com posicionamento absoluto, o espaço onde o elemento iria aparecer de forma estática desaparece.
- É como se o elemento não existisse para o elemento pai e restantes elementos adjacentes

```
.item-absolute {  
    position: absolute;  
    top: 10px;  
    left: 10px;  
}
```

9. Posicionamento de Elementos

- **Position Absolute**
 - Este comportamento não é em geral muito útil, porque isso significaria que tudo na página precisaria de estar posicionado de forma absoluta.
 - Então, porque existe absolute? **(relatively) absolute positioning**
 - O posicionamento absoluto torna-se muito mais prático quando é relativo a algum outro elemento que está no fluxo estático da página
 - As coordenadas para elementos absolutos são sempre relativas ao container mais próximo que é um elemento posicionado

9. Posicionamento de Elementos

- **Position Absolute**

```
.absolute {  
  position: relative;  
}  
  
.item-absolute {  
  position: absolute;  
  top: 10px;  
  left: 10px;  
}
```

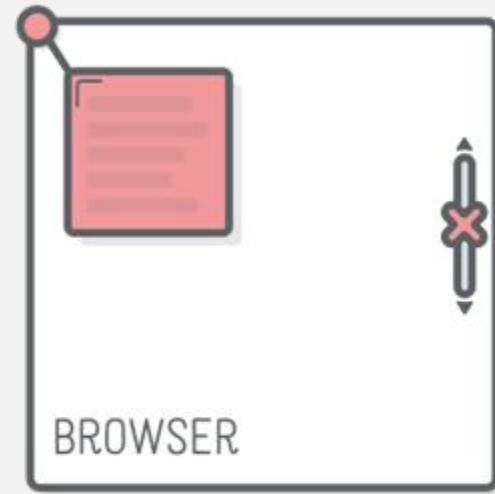


RELATIVELY ABSOLUTE
POSITIONING

9. Posicionamento de Elementos

- **Position Fixed**

- Posicionamento fixo é bastante similar com o posicionamento absoluto
- A principal diferença é que os elementos fixos não fazem scroll com os restantes elementos da página



FIXED POSITIONING

9. Posicionamento de Elementos

- **Position Fixed**

- O elemento é removido do fluxo normal da página
- Permite criar barras de navegação que ficam sempre visíveis no ecrã
- Exemplo: pop-up banners que nunca desaparecem

```
div.fixed {  
    position: fixed;  
    bottom: 0;  
    right: 0;  
    width: 300px;  
    border: 3px solid black;  
}
```

position: fixed;

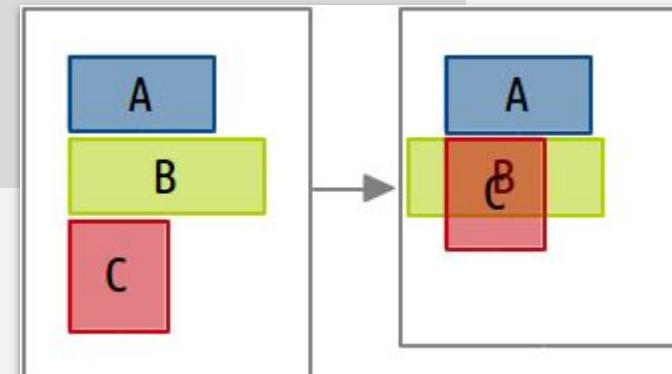
This div element has position: fixed;

9. Posicionamento de Elementos

- **Float**

- A propriedade **float** remove um elemento do fluxo do documento e desloca-o para a esquerda ou para a direita até que toque na borda da caixa que o contém ou outro elemento flutuante.

```
#b {  
    float: left;  
}
```



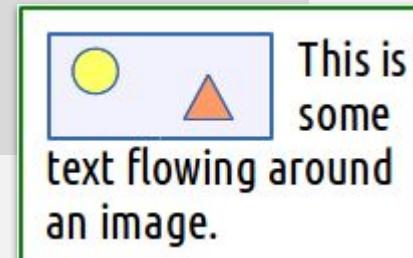
9. Posicionamento de Elementos

-

Floats e texto

- O texto flui em torno de elementos flutuantes. Esta propriedade é útil para colocar texto que flui em torno das imagens.

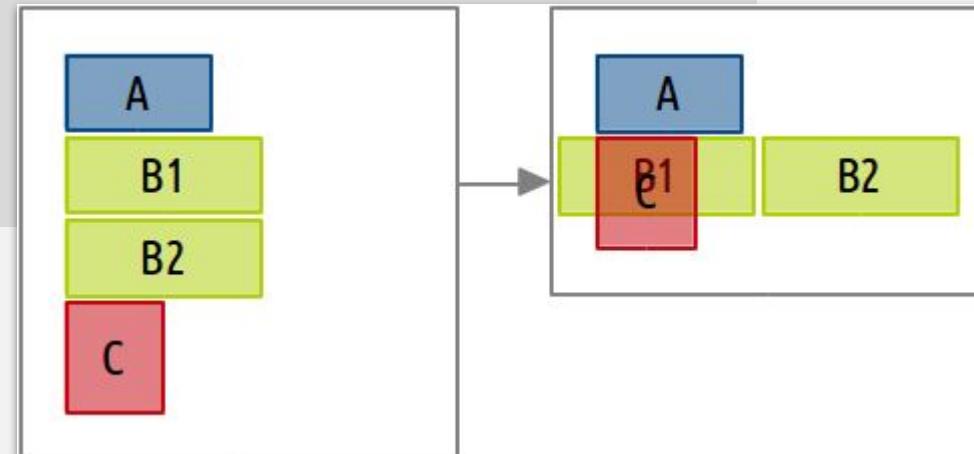
```
.article img {  
    float: left;  
}
```



9. Posicionamento de Elementos

- **Floats múltiplos**
 - Floats vão para a direita ou esquerda até encontrar outro float ou o container pai.

```
#b1, #b2 {  
    float: left;  
}
```

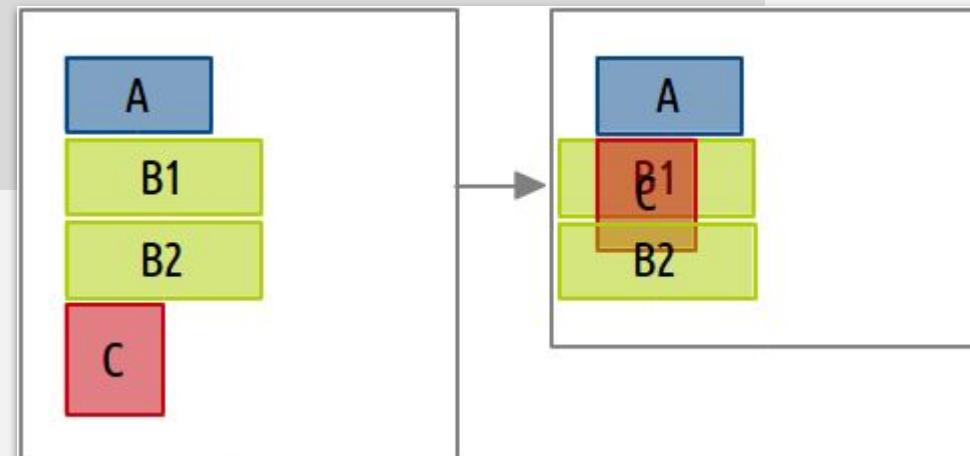


9. Posicionamento de Elementos

- **Clear**

- A propriedade **clear** indica se um elemento pode estar próximo a elementos float que o precedem ou se deve ser movido para baixo.
- Valores podem ser: **left**, **right** ou **both**.

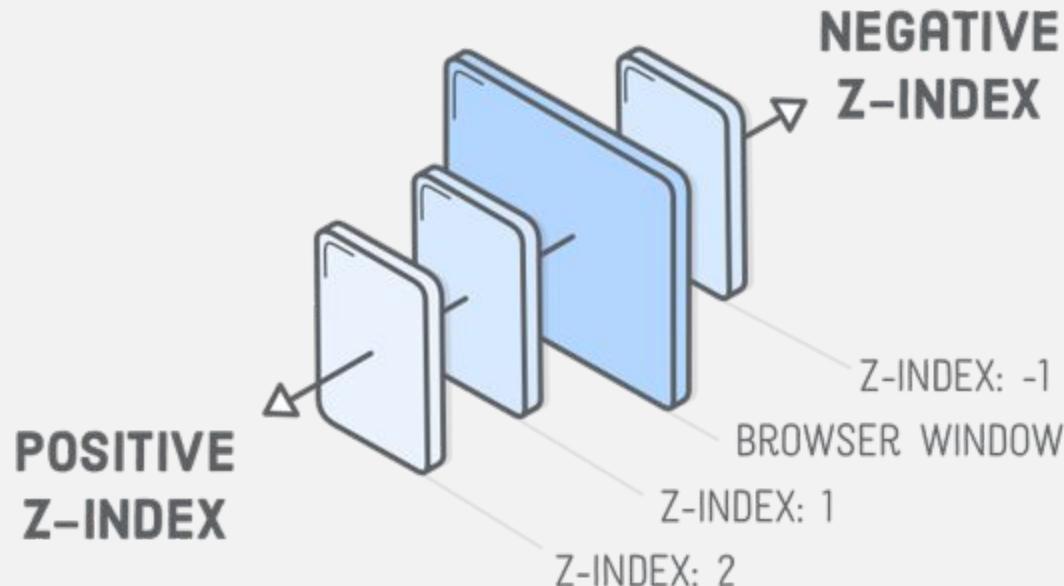
```
#b1, #b2 { float: left; }  
#b1 { clear: both; }
```



9. Posicionamento de Elementos

- **Z-Index**

- A propriedade **z-index** permite controlar a profundidade dos elementos na página
- Analisando como espaço 3D, os valores de z-index negativos irão para trás da página e os positivos encontram-se à frente da página.



9. Posicionamento de Elementos

- **Z-Index**
 - Valor por omissão para z-index é 0 (zero).
 - Apenas **elementos posicionados** podem ter propriedade z-index

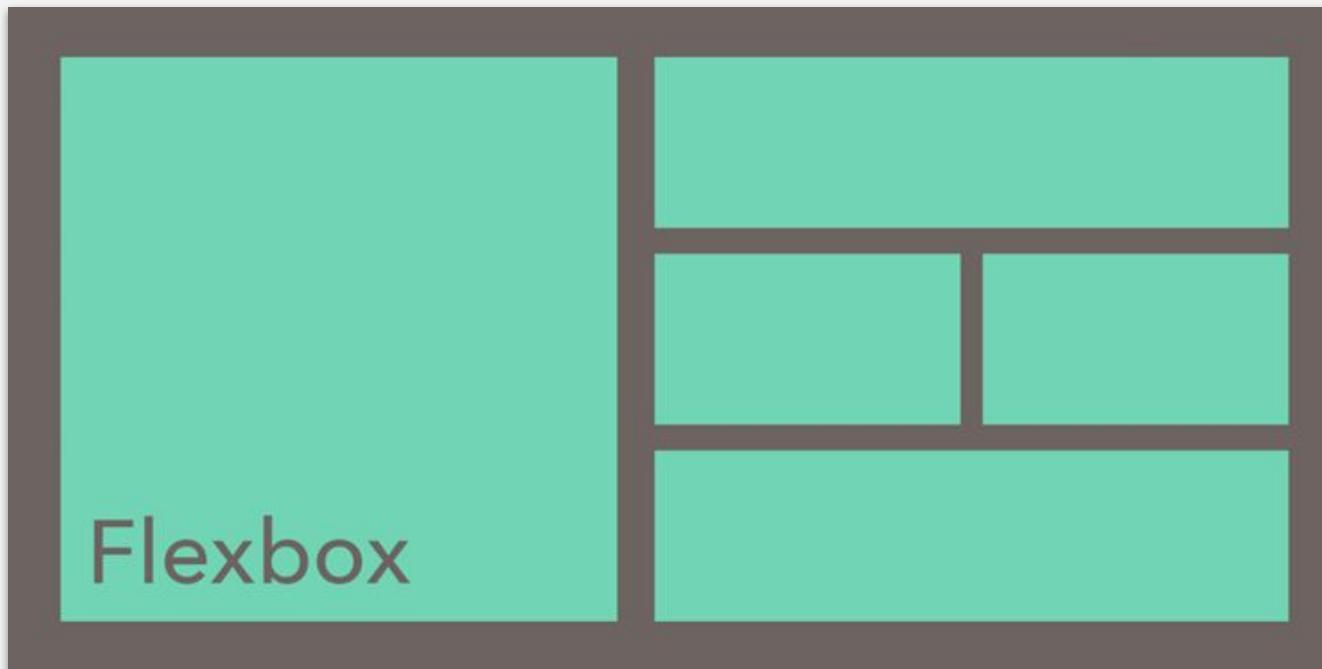
```
.dropdown > span {  
    z-index: 2;  
    position: relative; /* This is important! */  
    cursor: pointer;  
}
```

Módulo 3 - CSS

10. Flexbox

10. Flexbox

- **Flexbox** facilita a criação de uma estrutura de layout flexível e responsiva sem o uso de **float** ou **positioning**



10. Flexbox

- Para começar a usar Flexbox, primeiro é necessário definir um **flex container**.



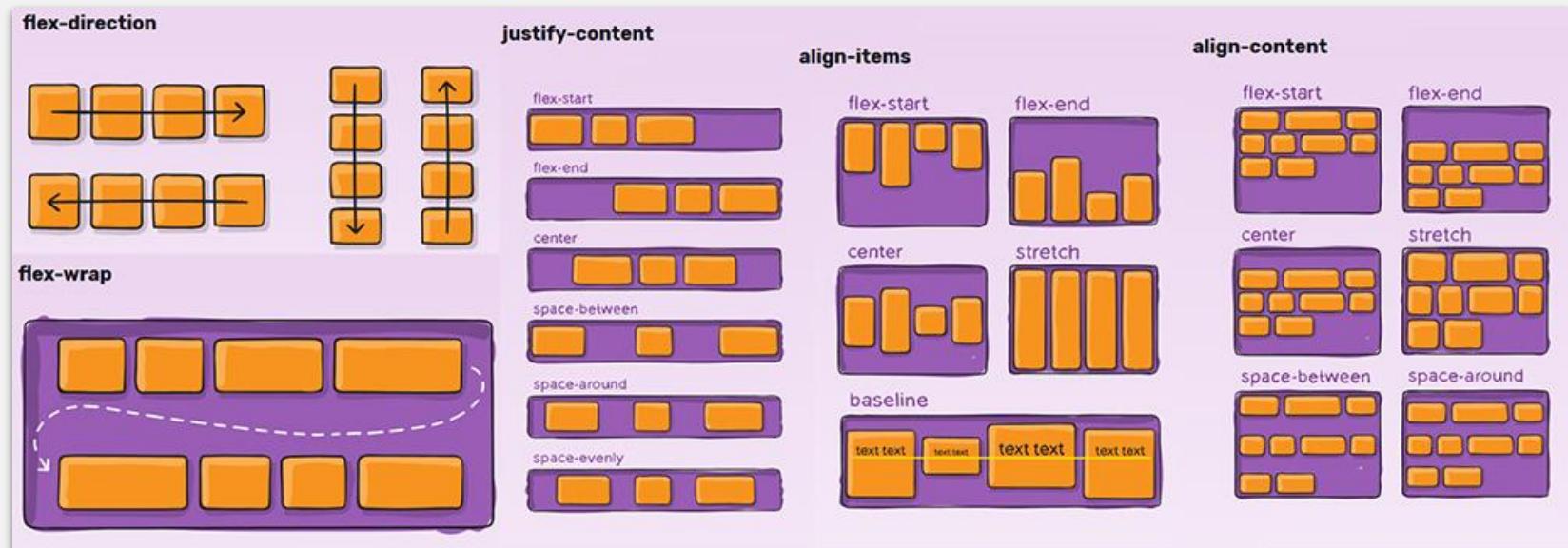
10. Flexbox

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
```

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

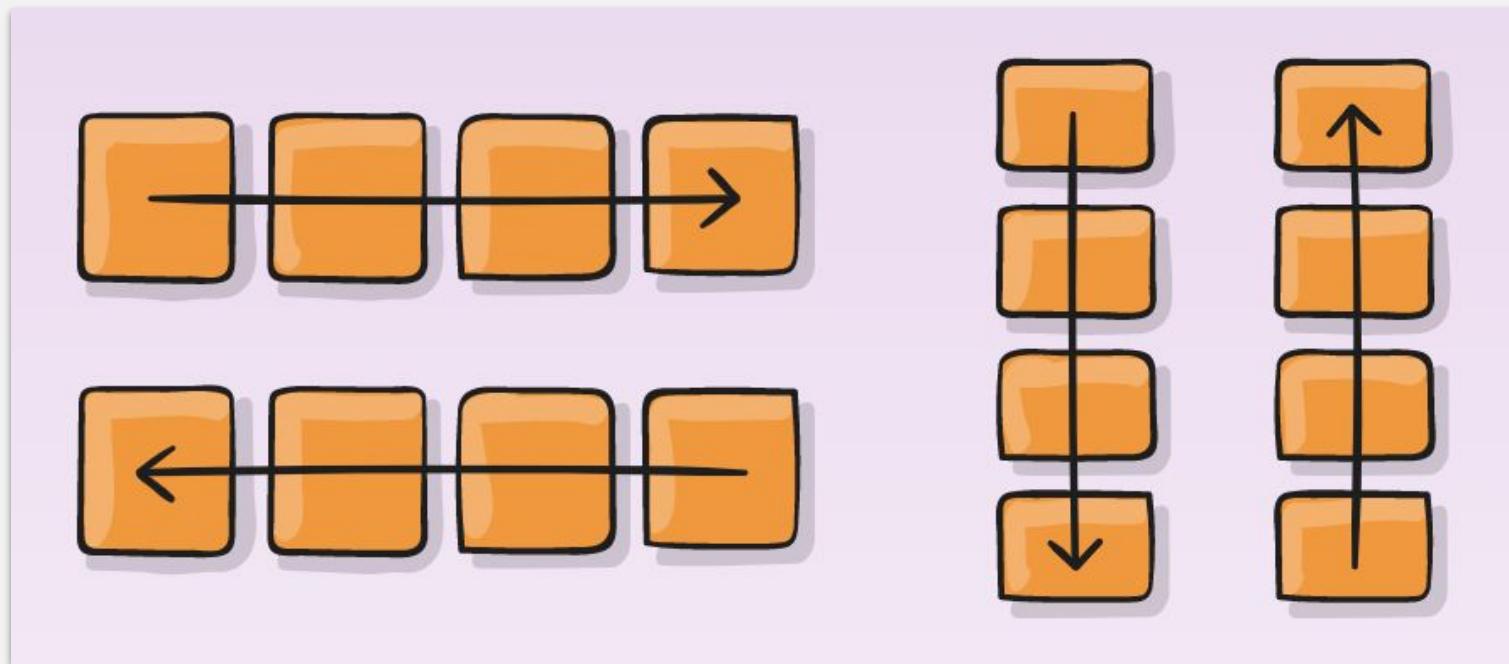
10. Flexbox

- Propriedades do container
 - **flex-direction**
 - **flex-wrap**
 - **flex-flow**
 - **justify-content**
 - **align-items**
 - **align-content**



10. Flexbox

- **flex-direction**
 - Define em que direção e sentido o container vai empilhar os itens flexíveis



10. Flexbox

- **flex-direction**

- O valor **column** empilha os itens flexíveis verticalmente (de cima para baixo)

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

- O valor **column-reverse** empilha os itens flexíveis verticalmente (mas de baixo para cima)

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

10. Flexbox

-

flex-direction

- O valor **row** empilha os itens flexíveis horizontalmente (da esquerda para a direita)

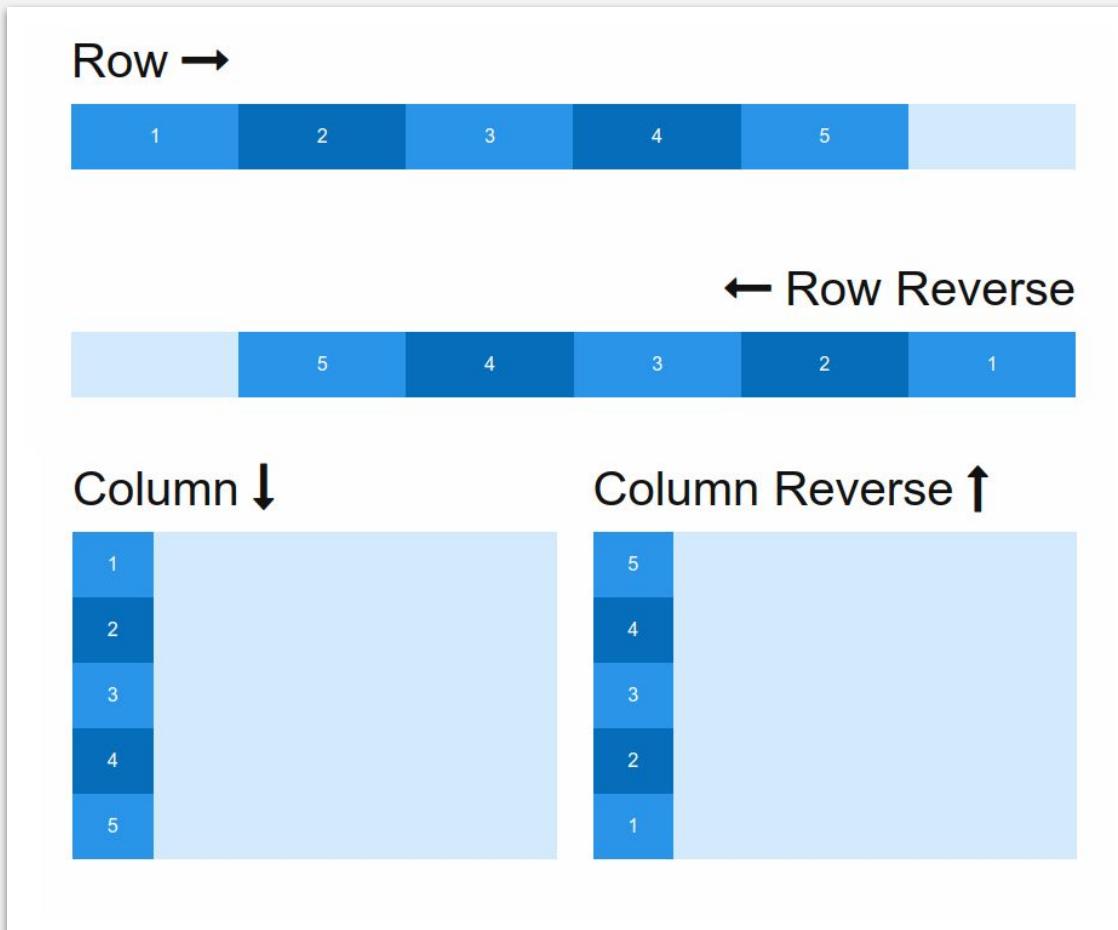
```
.flex-container {  
    display: flex;  
    flex-direction: row;  
}
```

- O valor **row-reverse** empilha os itens flexíveis horizontalmente (mas da direita para a esquerda)

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

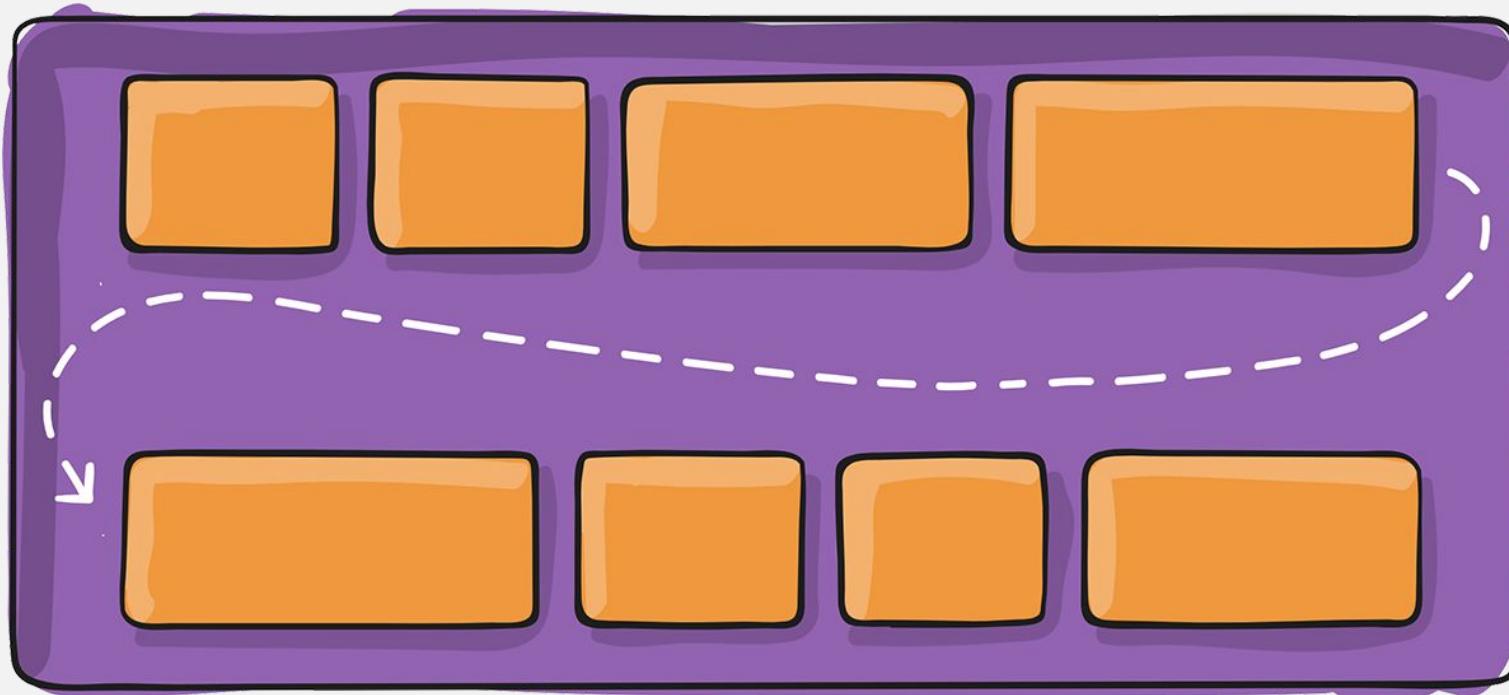
10. Flexbox

- **flex-direction**



10. Flexbox

- **flex-wrap**
 - A propriedade flex-wrap controla se o flex container é single-line ou multi-line e a direção do Cross Axis, que determina a direção em que novas linhas são empilhadas (stacked).



10. Flexbox

- **flex-wrap**
 - O valor **wrap** especifica que os itens flexíveis serão quebrados, se necessário.

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

- O valor **nowrap** especifica que os itens flex não serão quebrados (este é o padrão).

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

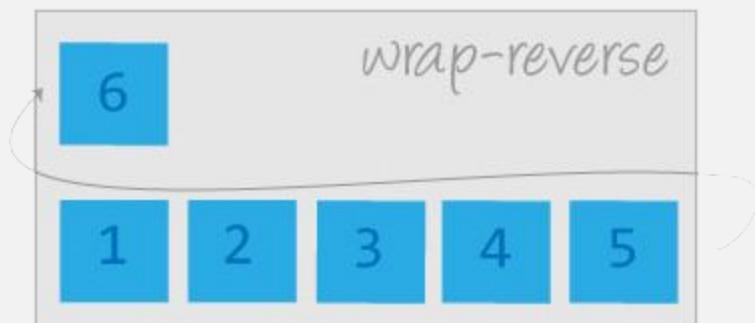
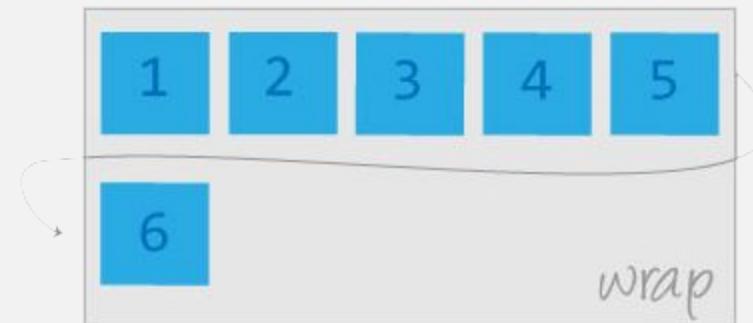
10. Flexbox

- **flex-wrap**
 - O valor **wrap-reverse** especifica que os itens flexíveis serão quebrados, se necessário, na ordem inversa.

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

10. Flexbox

- **flex-wrap**



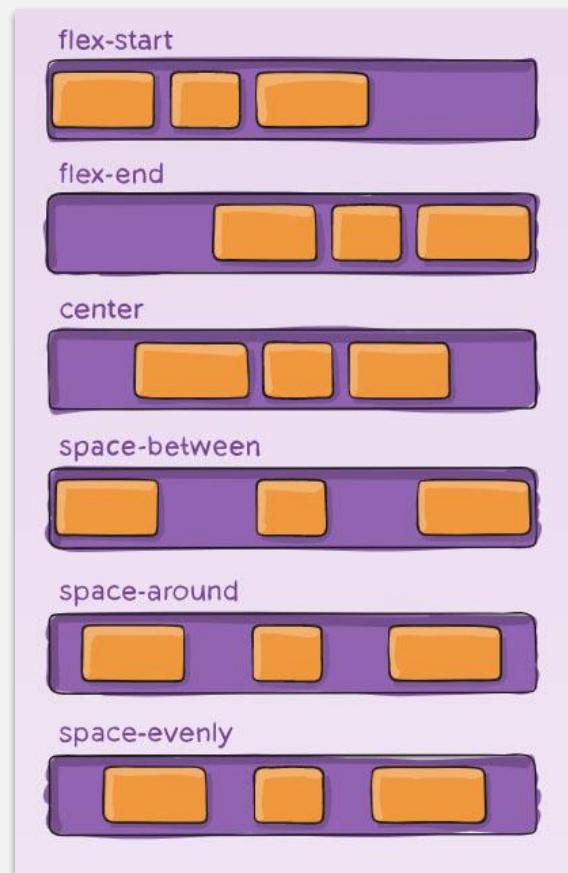
10. Flexbox

- **flex-flow**
 - A propriedade **flex-flow** é uma propriedade abreviada (shorthand) para definir as propriedades **flex-direction** e **flex-wrap**

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

10. Flexbox

- **justify-content**
 - A propriedade **justify-content** é usada para alinhar os itens flexíveis na horizontal



10. Flexbox

- **justify-content**
 - **flex-start** (valor por omissão): os itens são compactados no início da direção flex
 - **flex-end**: os itens são compactados no final da direção flex
 - **center**: os itens são centrados ao longo da linha
 - **space-between**: os itens são distribuídos igualmente na linha; o primeiro item está no início, o último item no fim da linha.
 - **space-around**: os itens são distribuídos igualmente na linha com espaço igual ao seu redor. Visualmente os espaços não são iguais, pois todos os itens têm espaço igual nos dois lados. O primeiro item terá uma unidade de espaço contra o limite container, mas duas unidades de espaço entre o próximo item, o próximo item também possui o seu próprio espaçamento.
 - **space-evenly**: os itens são distribuídos para que o espaçamento entre dois itens (e o espaço para os limites do container) seja igual.

10. Flexbox

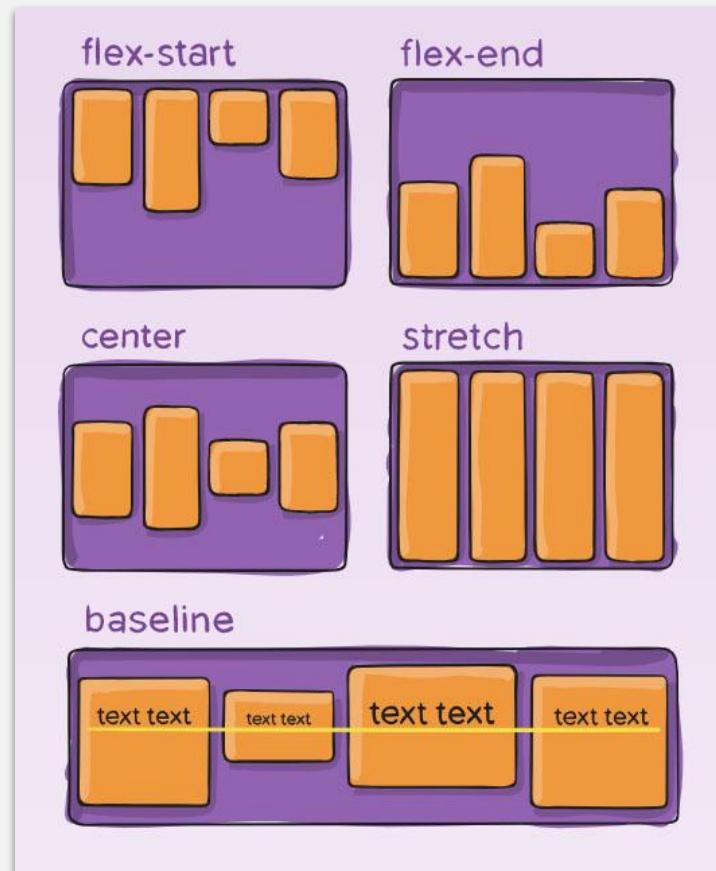
- **justify-content**

```
.flex-container {  
    display: flex;  
    justify-content: center;  
}
```



10. Flexbox

- **align-items**
 - A propriedade **align-items** é usada para alinhar os itens flexíveis na vertical



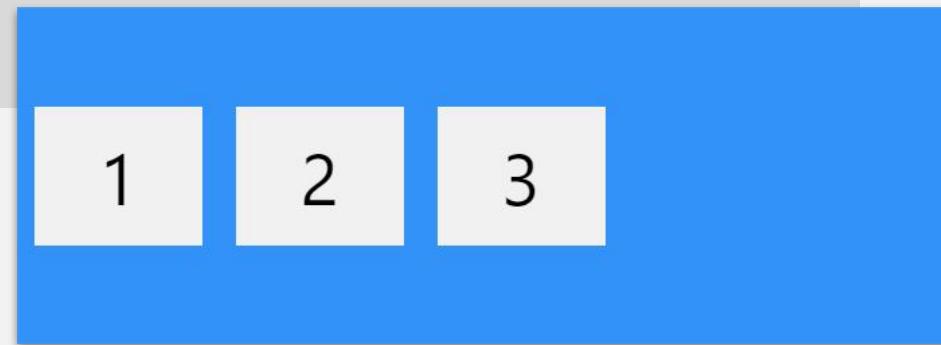
10. Flexbox

- **align-items**
 - **stretch** (valor por omissão): esticar para encher o container (respeita largura mínima/máxima)
 - **flex-start**: os itens são colocados no início
 - **flex-end**: os itens são colocados no final
 - **center**: os itens são centrados
 - **baseline**: os itens são alinhados pelas suas baseline

10. Flexbox

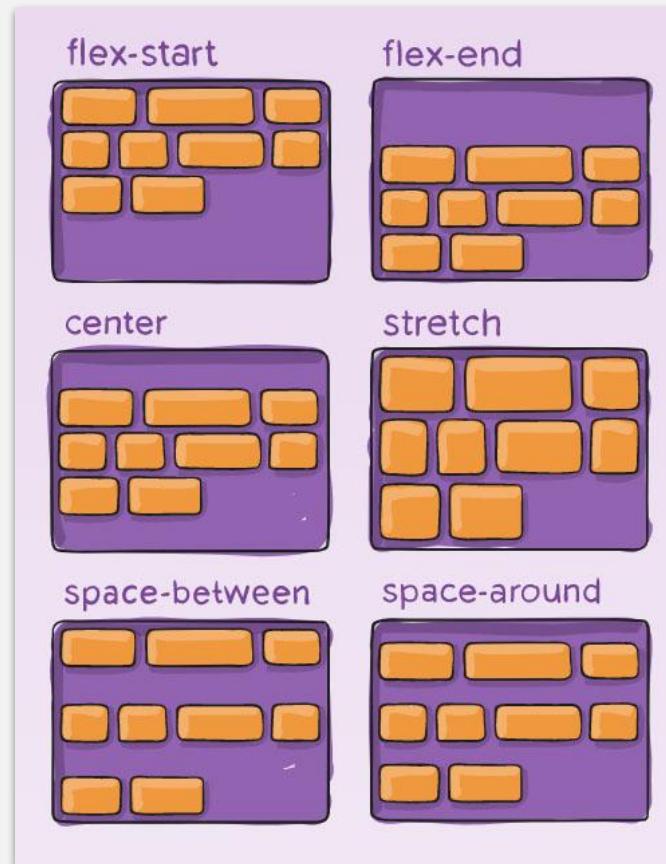
- **align-items**

```
.flex-container {  
    display: flex;  
    height: 200px;  
    align-items: center;  
}
```



10. Flexbox

- **align-content**
 - A propriedade **align-content** é usada para alinhar as linhas de itens flexíveis



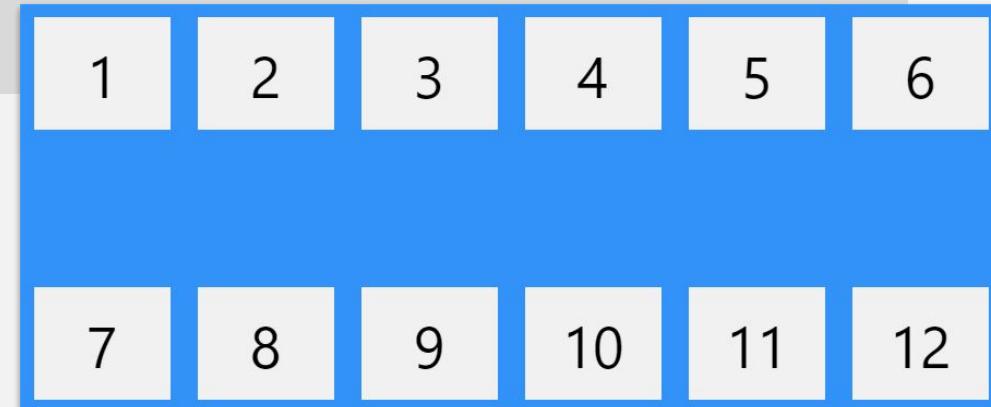
10. Flexbox

- **align-content**
 - **flex-start**: itens colocados no início do container
 - **flex-end**: itens colocados no final do container
 - **center**: itens centrados no container
 - **stretch** (valor por omissão): as linhas esticam para ocupar o espaço
 - **space-between**: itens distribuídos uniformemente; a primeira linha está no início do container enquanto a última está no final
 - **space-around**: itens distribuídos uniformemente com espaço igual ao redor de cada linha

10. Flexbox

- **align-content**

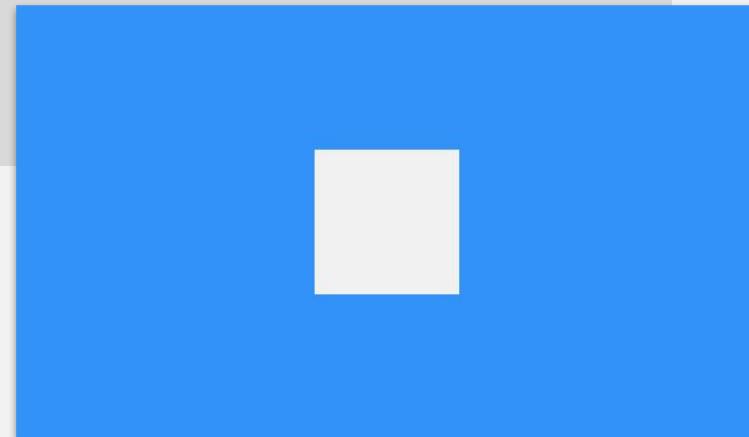
```
.flex-container {  
    display: flex;  
    height: 600px;  
    flex-wrap: wrap;  
    align-content: space-between;  
}
```



10. Flexbox

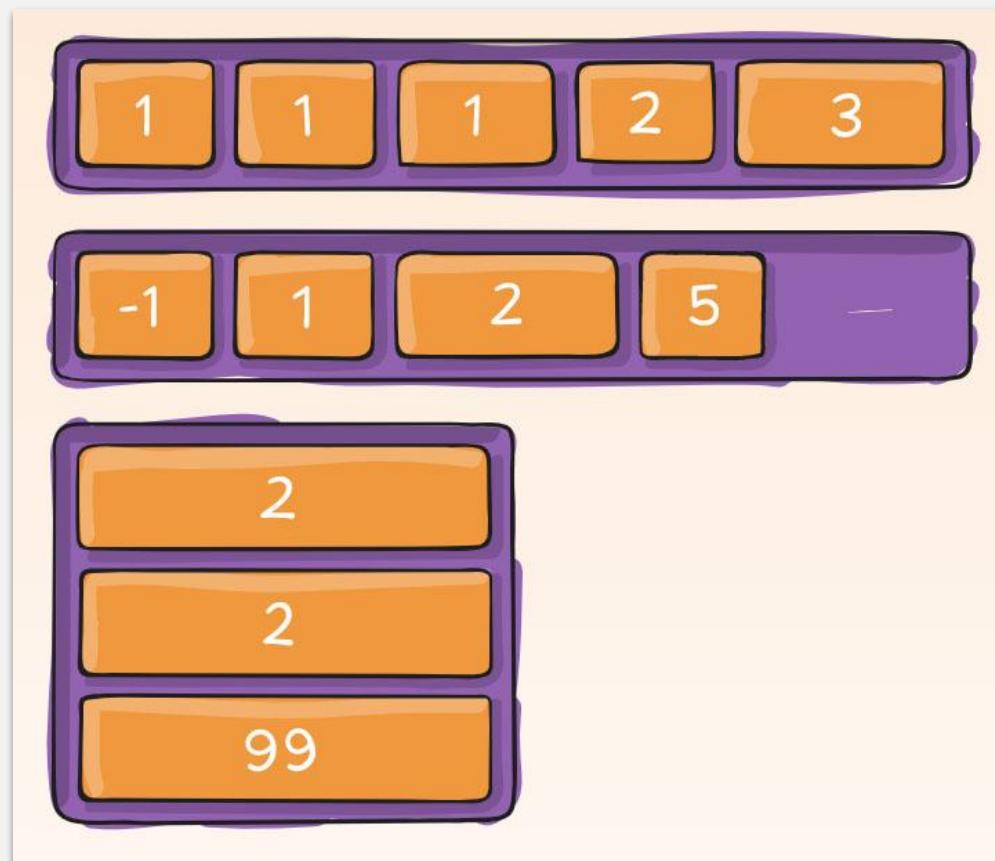
- **Centrar de forma perfeita**
 - Solução: definir **justify-content** e **align-items** como **center**.

```
.flex-container {  
    display: flex;  
    height: 300px;  
    justify-content: center;  
    align-items: center;  
}
```



10. Flexbox

- **order**
 - A propriedade **order** define a ordem dos elementos



10. Flexbox

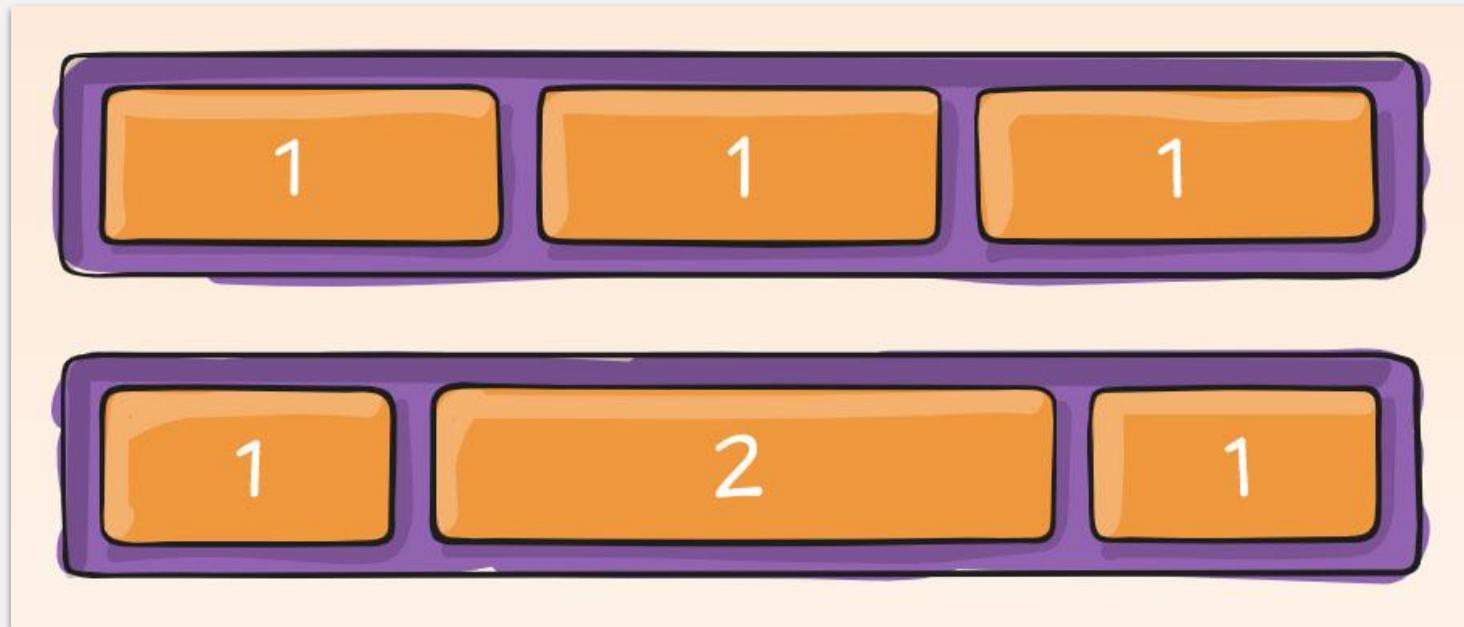
- o **order**

```
<div class="flex-container">  
  <div style="order: 3">1</div>  
  <div style="order: 2">2</div>  
  <div style="order: 4">3</div>  
  <div style="order: 1">4</div>  
</div>
```



10. Flexbox

- **flex-grow**
 - A propriedade **flex-grow** especifica quanto um item flexível crescerá em relação ao restantes



10. Flexbox

- **flex-grow**

```
<div class="flex-container">  
  <div style="flex-grow: 1">1</div>  
  <div style="flex-grow: 1">2</div>  
  <div style="flex-grow: 8">3</div>  
</div>
```

1

2

3

10. Flexbox

- **flex-shrink**

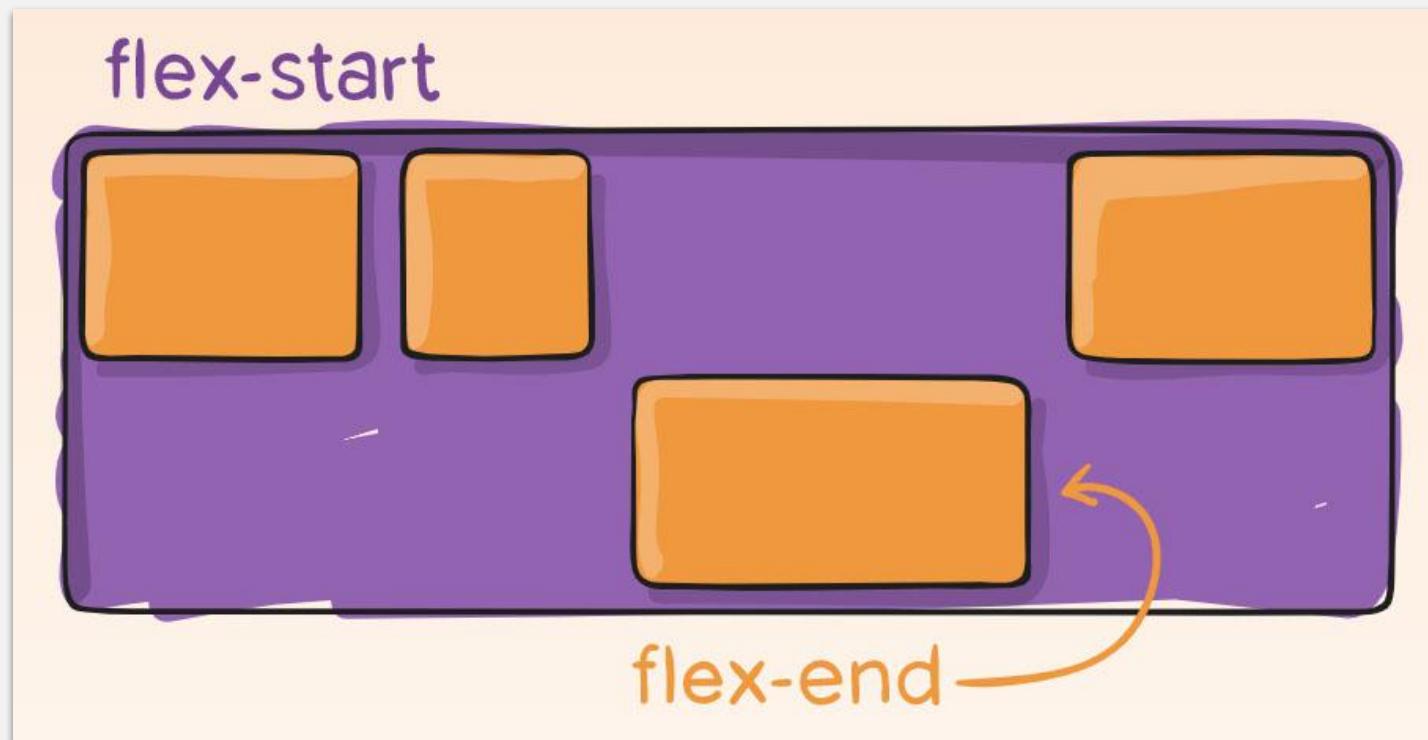
- A propriedade **flex-shrink** especifica quanto um item flexível diminuirá em relação ao restantes
- O valor deve ser um número, por omissão é 1.

```
<!-- Do not let the third flex item shrink as much as the other flex items -->
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex-shrink: 0">3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
  <div>7</div>
  <div>8</div>
  <div>9</div>
  <div>10</div>
</div>
```



10. Flexbox

- **align-self**
 - Permite fazer “override” do alinhamento “default” (ou aquele especificado pelo **align-items**) de forma individual em flex items
 - Consulte **align-items** para os valores disponíveis



10. Flexbox

- **align-self**

```
<div class="flex-container">  
  <div>1</div>  
  <div>2</div>  
  <div style="align-self: center">3</div>  
  <div>4</div>  
</div>
```

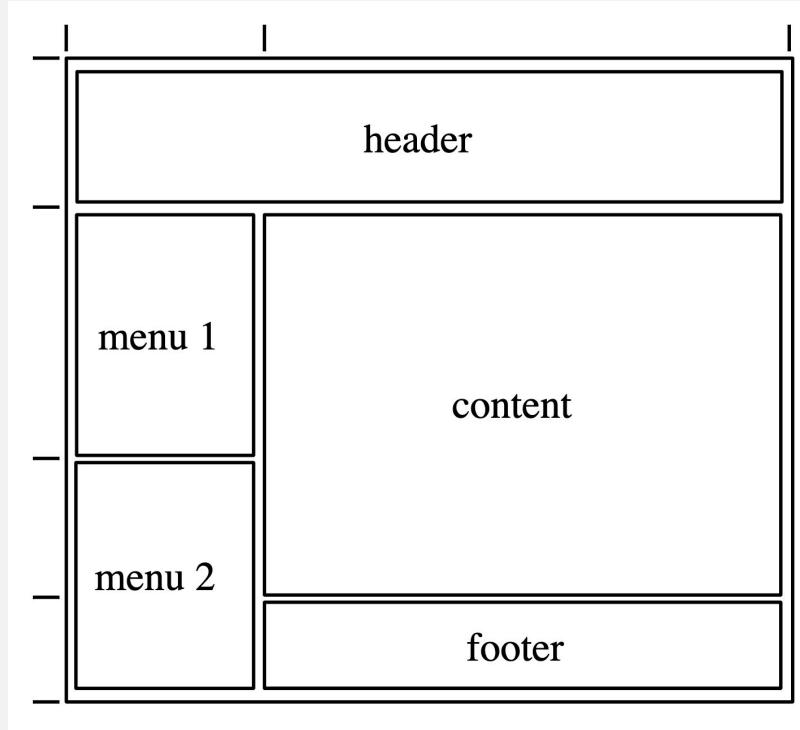


Módulo 3 - CSS

11. Grid

11. Grid

- Um layout grid permite alinhar elementos em colunas e linhas.



- <https://css-tricks.com/snippets/css/complete-guide-grid/>

11. Grid

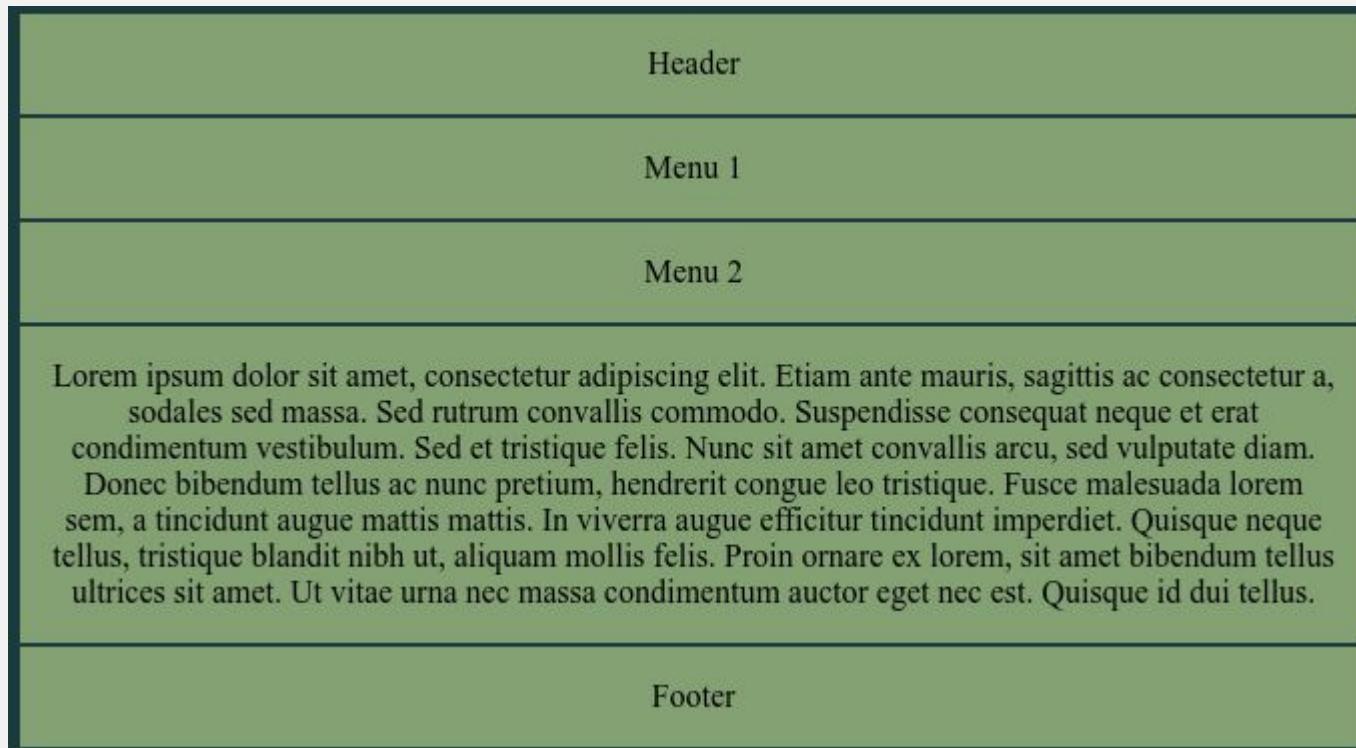
- Exemplo

```
<div class="container">
  <div class="item header">Header</div>
  <div class="item menu1">Menu 1</div>
  <div class="item menu2">Menu 2</div>
  <div class="item content">Lorem ipsum...</div>
  <div class="item footer">Footer</div>
</div>
```

```
.container {
  background-color: #1A3C3D;
  padding: 5px;
}
.item {
  color: black;
  text-align: center;
  margin: 2px;
  padding: 1em;
  background-color: #84A174;
}
```

11. Grid

- ## ○ Exemplo



11. Grid

- Alterando a propriedade **display** do container para “**grid**” transforma o container num **grid layout**.

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

- Por omissão existe apenas uma coluna.

11. Grid

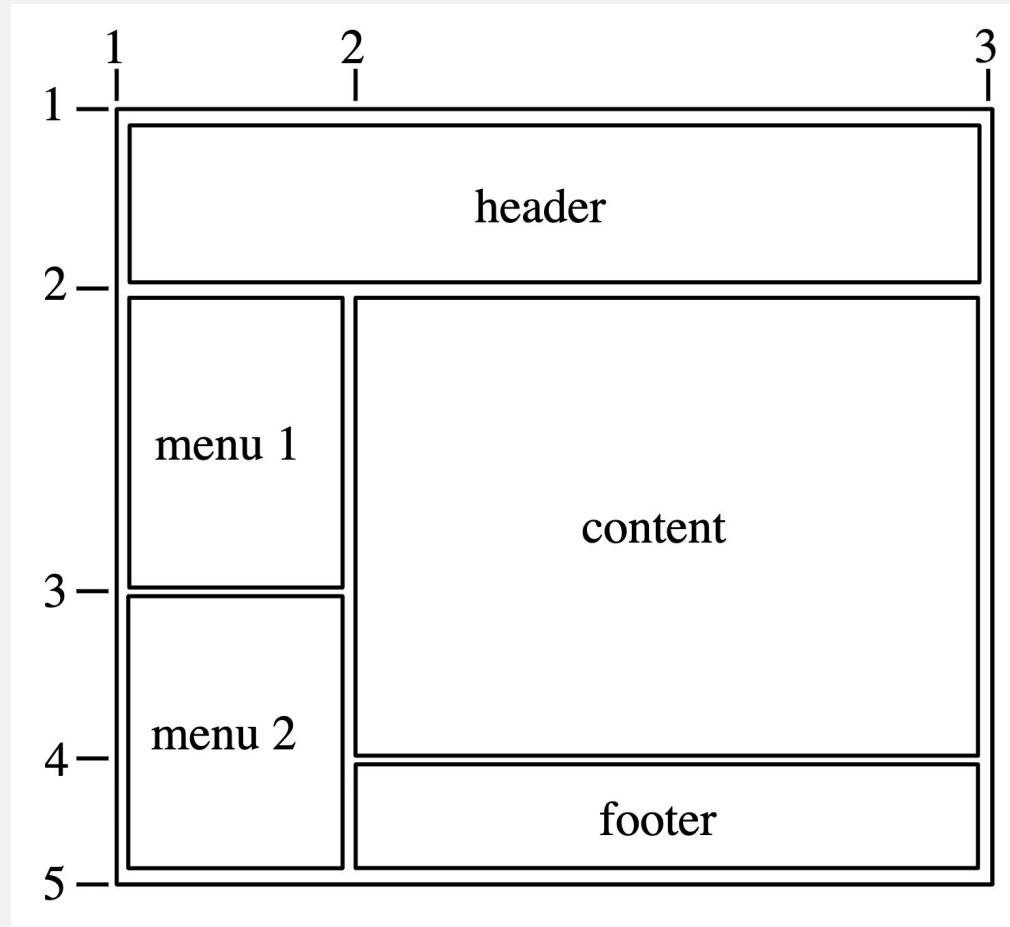
- Grid Templates
 - As propriedades **grid-template-columns** e **grid-template-rows** permitem definir o número e tamanho de colunas e linhas
 - Dimensões podem ser definidas como **auto**, um **comprimento**, uma **percentagem**, ou uma **fracção** do espaço livre (usando a unidade **fr**)

```
.container {  
    grid-template-columns: auto 1fr;  
    grid-template-rows: auto auto 1fr auto;  
}
```

Header	Menu 1
Menu 2	<p>Lore ipsum dolor sit amet, consectetur adipiscing elit. Etiam ante mauris, sagittis ac consectetur a, sodales sed massa. Sed rutrum convallis commodo. Suspendisse consequat neque et erat condimentum vestibulum. Sed et tristique felis. Nunc sit amet convallis arcu, sed vulputate diam. Donec bibendum tellus ac nunc pretium, hendrerit congue leo tristique. Fusce malesuada lorem sem, a tincidunt augue mattis mattis. In viverra augue efficitur tincidunt imperdiet. Quisque neque tellus, tristique blandit nibh ut, aliquam mollis felis. Proin ornare ex lorem, sit amet bibendum tellus ultrices sit amet. Ut vitae urna nec massa condimentum auctor eget nec est. Quisque id dui tellus.</p>
Footer	

11. Grid

- Por omissão, as linhas da grelha recebem valores numéricos.



11. Grid

- Podemos atribuir uma localização a um item dentro da grelha, referindo-nos a linhas específicas usando as propriedades: **grid-column-start**, **grid-column-end**, **grid-row-start** e **grid-row-end**.

```
.header {  
    grid-column-start: 1;  
    grid-column-end: 3;  
    grid-row-start: 1;  
    grid-row-end: 2;  
}
```

- Os valores podem ser os nomes numéricos por omissão das linhas da grelha ou um nome atribuído por nós.
- Os “end values” também podem ser o número de linhas ou colunas a serem estendidas. Por omissão, esses valores são um intervalo de um.

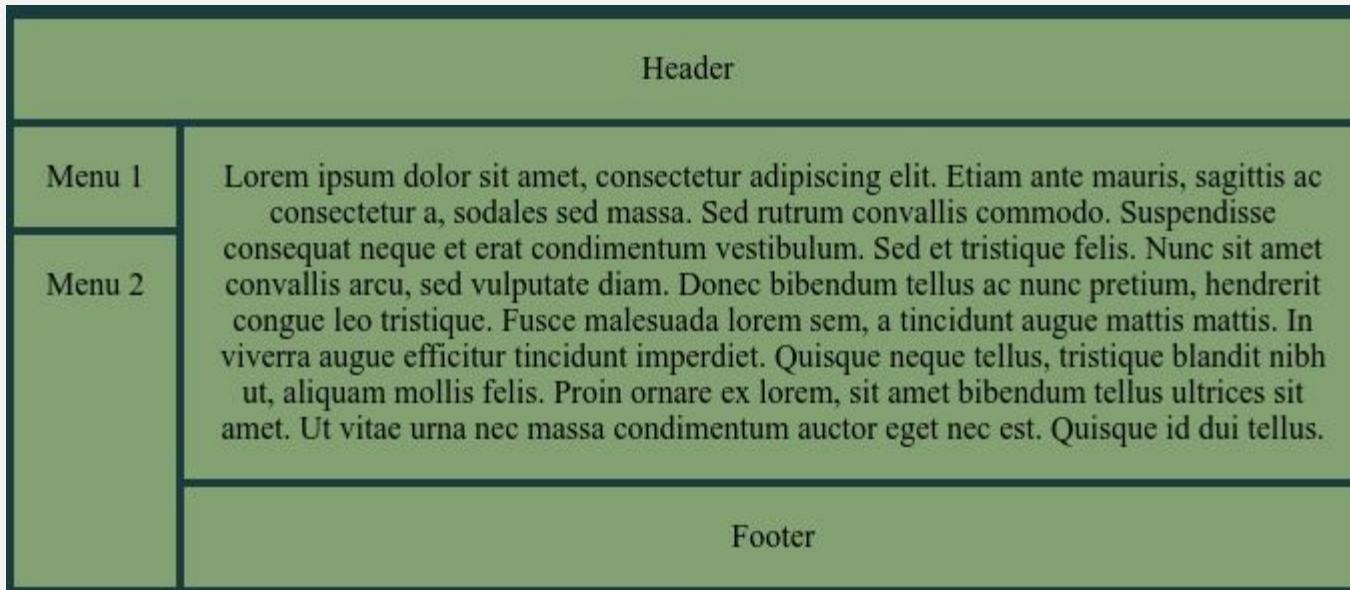
```
.header {  
    grid-column-end: span 2;  
    grid-row-end: span 1; /* not needed - default value */  
}
```

11. Grid

- As propriedades **grid-column** e **grid-row** podem ser usadas como uma abreviação (**shorthand**) para atribuir a localização de um item. Cada um deles recebe dois valores separados por uma barra (início/fim).
- A propriedade **grid-area** pode ser usada como uma abreviação para os quatro valores numa só vez: início da linha / início da coluna / final da linha / final da coluna.

```
.header {  
    grid-area: 1 / 1 / span 1 / span 2;  
}  
.menu1 {  
    grid-column: 1;  
    grid-row: 2;  
}  
.menu2 {  
    grid-column: 1;  
    grid-row: 3 / 5;  
}  
.content {  
    grid-column: 2;  
    grid-row: 2 / span 2;  
}  
.footer {  
    grid-column: 2;  
    grid-row: 4;  
}
```

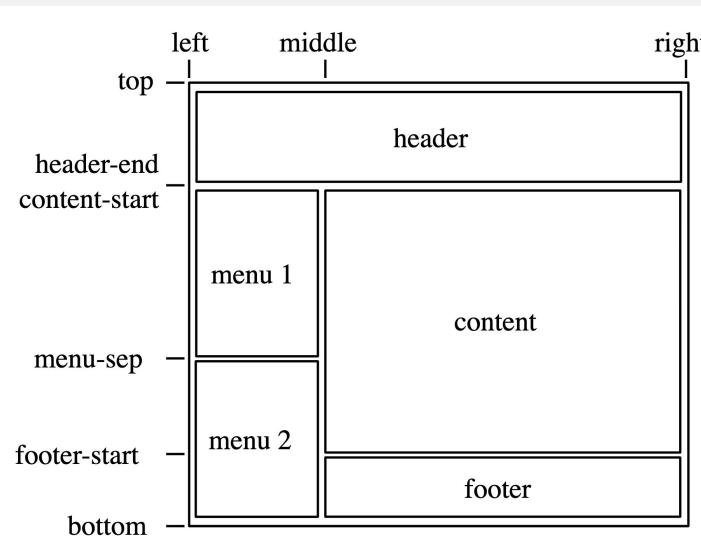
11. Grid



11. Grid

- Ao definir o modelo de grelha, podemos atribuir nomes às linhas da grelha. Uma linha pode ter mais que um nome.

```
.container {  
    grid-template-columns: [left] auto [middle] 1fr [right];  
    grid-template-rows: [top] auto [header-end content-start] auto  
                      [menu-sep] 1fr [footer-start] auto [bottom];  
}  
.content {  
    grid-area: content-start / middle / footer-start / right;  
}
```



11. Grid

- Ao dar nomes aos itens usando a propriedade **grid-area**, podemos definir um modelo de grelha de uma forma mais visual.
- Qualquer número de pontos adjacentes pode ser usado para declarar uma única célula vazia.

```
.container {  
    grid-template-columns: auto 1fr;  
    grid-template-rows: auto auto 1fr auto;  
    grid-template-areas: "header header"  
                        "menu1 content"  
                        "menu2 content"  
                        "menu2 footer";  
}  
  
.header { grid-area: header; }  
  
.menu1 { grid-area: menu1; }  
  
.menu2 { grid-area: menu2; }  
  
.content { grid-area: content; }  
  
.footer { grid-area: footer; }
```

Módulo 3 - CSS

12. Layout em Colunas

12. Layout em Colunas

○ Introdução

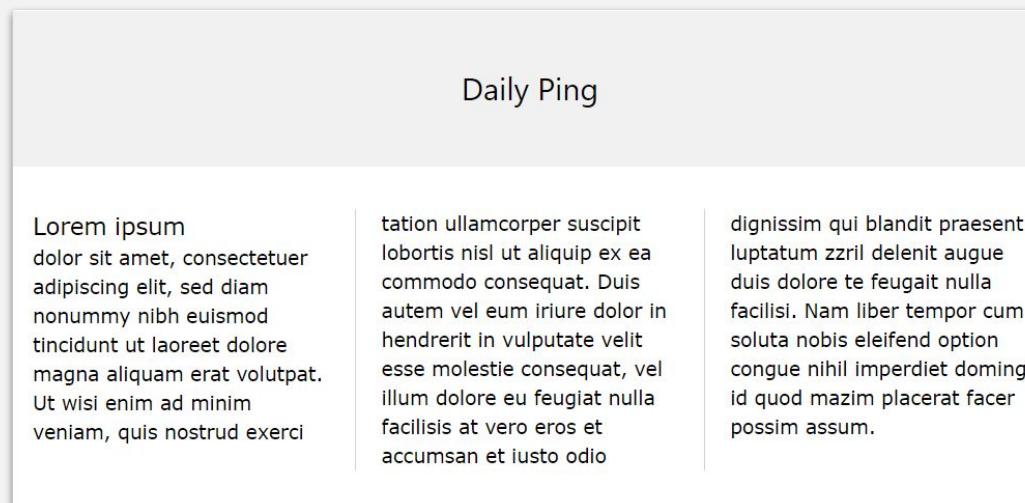
- Na Web, as tabelas também foram usadas para descrever layouts com várias colunas.
- O principal benefício de usar colunas baseadas em CSS é a flexibilidade
- O conteúdo pode fluir de uma coluna para outra e o número de colunas pode variar dependendo do tamanho da viewport
- A remoção da marcação da tabela de apresentação dos documentos permite que eles sejam apresentados com mais facilidade em vários dispositivos



12. Layout em Colunas

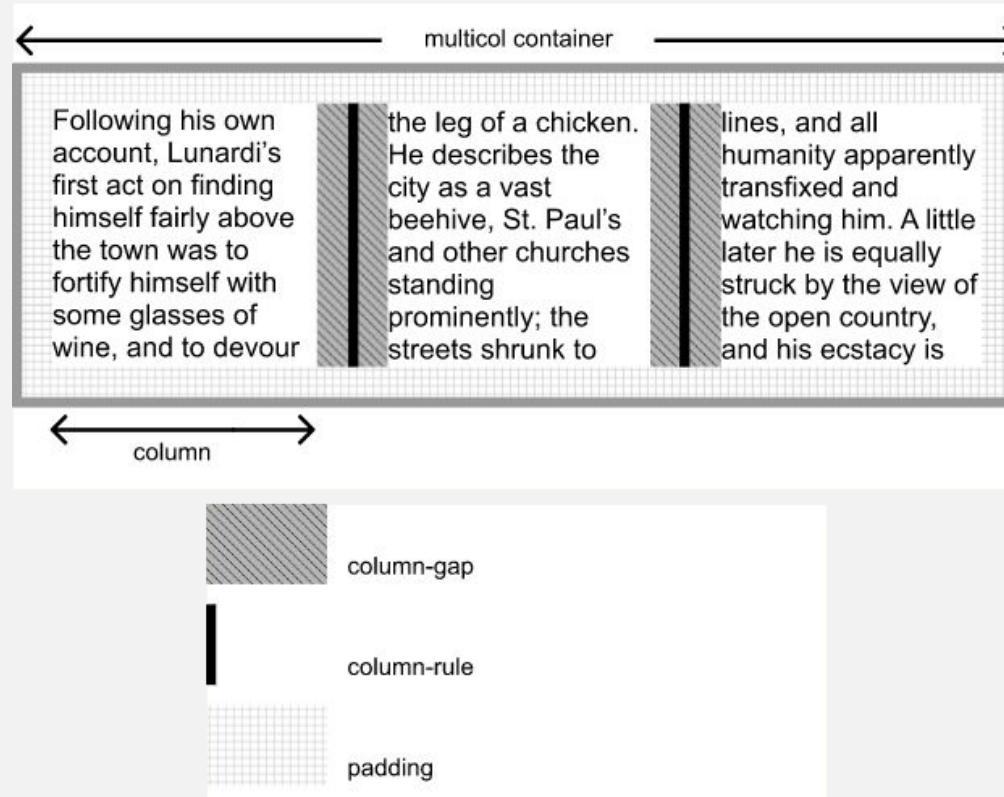
○ Multi-column Layout

- Um container de várias colunas é um elemento cuja propriedade “**column-width**” ou “**column-count**” não tem o valor “**auto**”
- No modelo box-model, o conteúdo de um elemento é enviado para a caixa de conteúdo do elemento correspondente.
- O multi-column layout introduz um **novo tipo de container** entre a caixa de conteúdo e o conteúdo



12. Layout em Colunas

- **Multi-column Layout**
 - Column boxes num container multi-column são colocadas em linhas



12. Layout em Colunas

- **Column-count**

- A propriedade **column-count** especifica o número de colunas em que um elemento deve ser dividido

```
<div id="col">
  <p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit,
    sed do eiusmod tempor incididunt ut labore et dolore magna
    aliqua.
  </p>
  <p>
    Ut enim ad minim veniam, quis nostrud exercitation ullamco
    laboris nisi ut aliquip ex ea commodo consequat.
  </p>
  <p>
    Duis aute irure dolor in reprehenderit in voluptate velit
    esse cillum dolore eu fugiat nulla pariatur.
  </p>
  <p>
    Excepteur sint occaecat cupidatat non proident, sunt in
    culpa qui officia deserunt mollit anim id est laborum.
  </p>
</div>
```

```
#col {
  column-count: 2;
}
```

Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex
ea commodo consequat.

Duis aute irure dolor in reprehenderit in
voluptate velit esse cillum dolore eu fugiat
nulla pariatur.

Excepteur sint occaecat cupidatat non
proident, sunt in culpa qui officia deserunt
mollit anim id est laborum.

12. Layout em Colunas

○ Column-width

- A propriedade **column-width** define a largura mínima desejada da coluna
- Se column-count não estiver definido, o navegador criará automaticamente o número de colunas na largura disponível

```
<div id="wid">  
    Lorem ipsum dolor sit amet, consectetur adipisicing elit,  
    sed do eiusmod tempor incididunt ut labore et dolore magna  
    aliqua. Ut enim ad minim veniam, quis nostrud exercitation  
    ullamco laboris nisi ut aliquip ex ea commodo consequat.  
    Duis aute irure dolor in reprehenderit in voluptate velit  
    esse cillum dolore eu fugiat nulla pariatur. Excepteur sint  
    occaecat cupidatat non proident, sunt in culpa qui officia  
    deserunt mollit anim id est laborum  
</div>
```

```
#wid {  
    column-width: 100px;  
}
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut	labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris	nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum	dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia	deserunt mollit anim id est laborum
---	---	--	--	---

12. Layout em Colunas

- **Columns**
 - A propriedade **columns** é um “shorthand” para as propriedades **column-count** e **column-width**
 - Especifica a largura mínima de cada coluna, e o número máximo de colunas.

```
div {  
    columns: 100px 3;  
}
```

12. Layout em Colunas

- **Column-gap**
 - A propriedade **column-gap** especifica o espaço entre colunas

```
<div id="column_gap">  
  Lorem ipsum dolor sit amet, consectetur adipisicing elit,  
  sed do eiusmod tempor incididunt ut labore et dolore magna  
  aliqua. Ut enim ad minim veniam, quis nostrud exercitation  
  ullamco laboris nisi ut aliquip ex ea commodo consequat.  
  Duis aute irure dolor in reprehenderit in voluptate velit  
  esse cillum dolore eu fugiat nulla pariatur. Excepteur sint  
  occaecat cupidatat non proident, sunt in culpa qui officia  
  deserunt mollit anim id est laborum  
</div>
```

```
#column_gap {  
  column-count: 5;  
  column-gap: 2em;  
}
```

Lore dol con adip elit, eius tempo incidi	labo dolo aliqu enim minim veniam, nostru exercit	ullam laboris nisi ut aliquip ex ea commo consequat. Duis aute irure dolor in reprehenderit	in volunta velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non	proident, sunt in culpa qui officia deserunt mollit anim id est laborum
--	--	--	--	--

12. Layout em Colunas

- **Column-rule-style**
 - A propriedade **column-rule-style** especifica o estilo da régua entre colunas

```
<p>This is a bunch of text split into three columns.  
The `column-rule-color` property is used to change  
the color of the line that is drawn between columns.  
Don't you think that's wonderful?</p>
```

```
p {  
    column-count: 3;  
    column-rule-style: dashed;  
}
```

This is a bunch of text split into three columns. The **'column-rule-style'** property

is used to change the style of the line that is drawn between

columns. Don't you think that's wonderful?

12. Layout em Colunas

- **Column-rule-color**
 - A propriedade **column-rule-color** especifica a cor da régua entre colunas

```
<p>This is a bunch of text split into three columns.  
The `column-rule-color` property is used to change  
the color of the line that is drawn between columns.  
Don't you think that's wonderful?</p>
```

```
p {  
    column-count: 3;  
    column-rule-style: solid;  
column-rule-color: blue;  
}
```

This is a bunch of text split into three columns. The 'column-rule-color' property

is used to change the color of the line that is drawn between

columns. Don't you think that's wonderful?

12. Layout em Colunas

- **Column-rule-width**
 - A propriedade column-rule-width especifica a espessura da régua entre colunas

```
<p>This is a bunch of text split into three columns.  
The `column-rule-color` property is used to change  
the color of the line that is drawn between columns.  
Don't you think that's wonderful?</p>
```

```
p {  
    column-count: 3;  
    column-rule-style: solid;  
    column-rule-width: thick;  
}
```

This is a bunch of text split into three columns. The 'column-rule-width' property is used to change the width of the line that is drawn between columns. Don't you think that's wonderful?

12. Layout em Colunas

- **Column-rule**
 - A propriedade **column-rule** é um shorthand para:
 - **column-rule-with**
 - **column-rule-style** (obrigatório)
 - **column-rule-color**

```
<p class="content-box">  
  This is a bunch of text split into three columns.  
  Take note of how the `column-rule` property is used  
  to adjust the style, width, and color of the rule  
  that appears between the columns.  
</p>
```

```
.content-box {  
  padding: 0.3em;  
  background: #ff7;  
  column-count: 3;  
  column-rule: inset 2px #33f;  
}
```

This is a bunch of text split into three columns. Take note of how the 'column-rule' property is used to adjust the style, width, and color of the rule that appears between the columns.

12. Layout em Colunas

- **Column-span**

- A propriedade **column-span** define o número de colunas que um elemento deve abranger

```
<article>
  <h2>Header spanning all of the columns</h2>
  <p>The h2 should span all the columns. The rest of the text should be distributed among the columns.
  </p>
  <p>This is a bunch of text split into three columns using the CSS `columns` property. The text is equally
  distributed over the columns.
  </p>
  <p>This is a bunch of text split into three columns using the CSS `columns` property. The text is equally
  distributed over the columns.
  </p>
  <p>This is a bunch of text split into three columns using the CSS `columns` property. The text is equally
  distributed over the columns.</p>
  <p>This is a bunch of text split into three columns using the CSS `columns` property. The text is equally
  distributed over the columns.</p>
</article>
```

Header spanning all of the columns

```
article {
  columns: 3;
}
h2 {
  column-span: all;
}
```

The h2 should span all the columns. The rest of the text should be distributed among the columns.

This is a bunch of text split into three columns using the CSS `columns` property. The text is equally distributed over the columns.

This is a bunch of text split into three columns using the CSS `columns` property. The text is equally distributed over the columns.

This is a bunch of text split into three columns using the CSS `columns` property. The text is equally distributed over the columns.

This is a bunch of text split into three columns using the CSS `columns` property. The text is equally distributed over the columns.

Módulo 3 - CSS

13. Cascading Order

13. Cascading Order

○ Cascata

- Caso não sejam definidas regras css, são aplicadas as do navegador (Browser default)
- Se forem definidas regras css elas podem estar dispersas por:
 - Inline (no elemento ao qual queremos aplicar o estilo)
 - Interna/Externa (no elemento <head>)
- Se 2 regras seleccionarem o mesmo elemento, qual é aplicada?
 - Se os nomes das propriedades das regras:
 - forem diferentes, estas são agregadas (herança)
 - forem iguais, são aplicadas as regras em cascata:
 1. Cálculo de especificidade
 2. Ordem de especificação
 3. Anotação !important



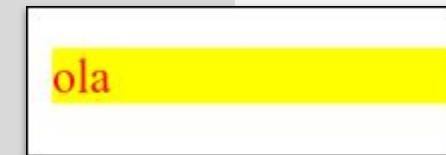
13. Cascading Order

- **Cascata**

- **Herança**

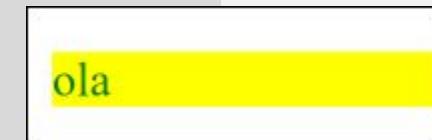
- As propriedades (se diferentes) vão sendo herdadas para o mesmo elemento

```
<style>
  p {
    color:red;
  }
</style>
...
<p style="background-color: yellow">ola</p>
```



- As propriedades se iguais, têm prioridade as mais específicas (cálculo de especificidade).

```
<style>
  p {
    color:red;
  }
</style>
...
<p style="color:green; background-color: yellow">ola</p>
```

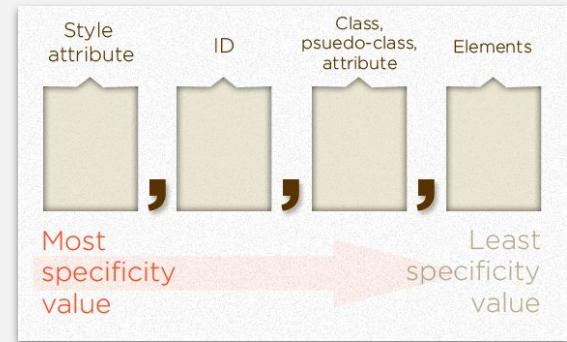


13. Cascading Order

○ Cascata

- Cálculo da especificidade

- Mais específicas prevalecem sobre as mais gerais
- Regras que se aplicam:
 - a vários tipos são mais gerais
 - apenas a um tipo são mais específicas
- Como se calcula?



```
#sidebar ul li a.myclass:hover{}
```

inline

0

IDs

1

classes

2

elements

3

& pseudo classes
& attributes

& pseudo elements

13. Cascading Order

- **Cascata**
 - Cálculo da especificidade
 - Exemplo:

```
...
<style>
  body h1 {
    color: green;
  }
  h1 {
    color: purple;
  }
</style>
...
<html>
  <body>
    <h1>ESMAD</h1>
  </body>
</html>
```

- Cálculo da especificidade:
 - 1^ªregra: 0002
 - 2^ªregra: 0001
- Apesar da 2^ª regra ser definida depois, tem menos especificidade que a 1^ª regra
- **Logo é a 1^ª regra que é aplicada!**

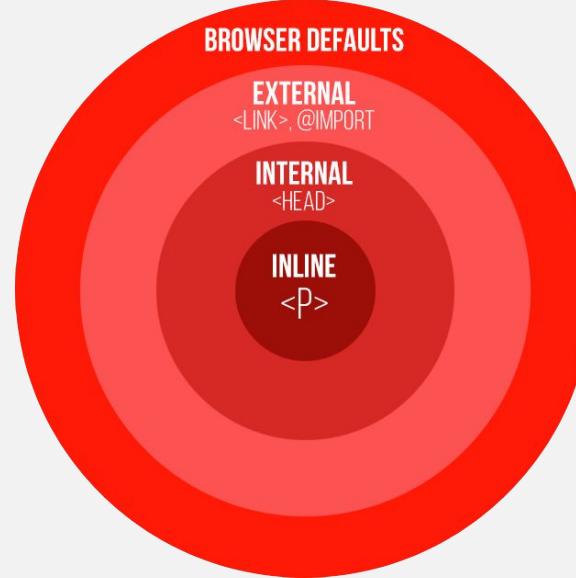
ESMAD

13. Cascading Order

- **Cascata**

- **Ordem de especificação**

- Se duas regras têm a mesma especificidade e selecionam o mesmo elemento, qual a regra a aplicar?
 - Segue-se a lógica da prioridade e tem precedência a regra definida em último lugar:
 1. Estilo inline (**máxima prioridade**).
 2. Estilos externo e interno (<head>).
 3. Caso não exista nenhum estilo, são aplicados os padrões do browser.



- Aplicação do estilo interno/externo depende de qual está escrito em último lugar!

13. Cascading Order

- **Cascata**

- **Ordem de especificação**
 - Exemplo:

```
...
<style>
  body h1 {
    color: green;
  }
  html h1 {
    color: purple;
  }
</style>
...
<html>
  <body>
    <h1>ESMAD</h1>
  </body>
</html>
```

- Cálculo da especificidade:
 - 1^ªregra: 0002
 - 2^ªregra: 0002
- Como ambas as regras têm a mesma especificidade, aplica-se a regra definida em último lugar.
- **Logo é a 2^ª regra que é aplicada!**

ESMAD

13. Cascading Order

- **Cascata**

- **Anotação !important**

- Precedência das regras pode ser forçada colocando **!important** depois do valor
 - Deve ser usada com sobriedade

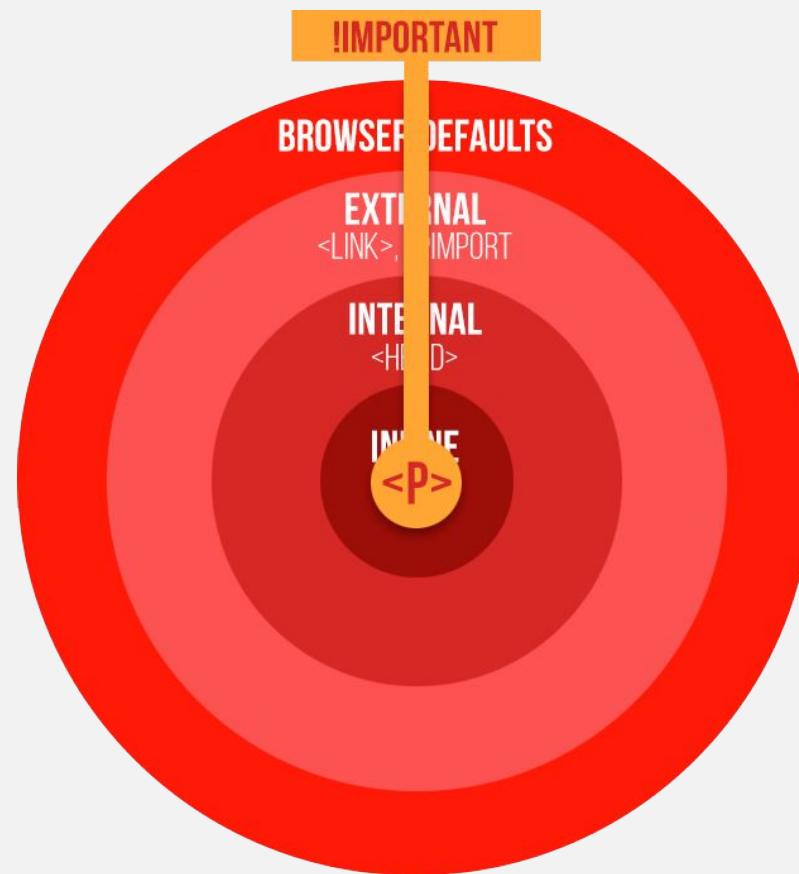
```
...
<style>
  h1 {
    color: green !important;
  }
</style>
...
<html>
  <body>
    <h1 style="color: red">ESMAD</h1>
  </body>
</html>
```

- Cálculo da especificidade:
 - 1^aregra: 0001
 - 2^aregra: 1000
- A segunda regra (inline) tem uma especificidade maior ($1000 > 1$), mas a anotação **!importante** definida na 1^a regra força o uso da regra.
- **Logo é a 1^a regra que é aplicada!**

ESMAD

13. Cascading Order

- **Cascata**
 - Anotação **!important**



Módulo 3 - CSS

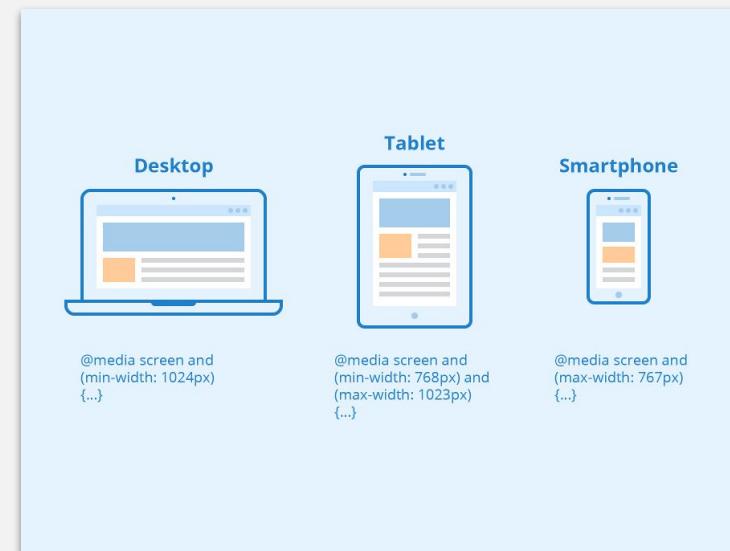
14. Media Queries

14. Media Queries



Contexto

- A regra **@media** introduzida em CSS2, permite aplicar regras CSS diferentes mediante diferentes contextos:
 - **Largura/altura do navegador**
 - **Largura/altura do dispositivo**
 - **Densidade do ecrã**
 - **Orientação do ecrã**



14. Media Queries

○ Sintaxe

- Num ficheiro CSS incluído de forma normal:

```
@media (validação) {  
    /* CSS a aplicar */  
}
```

```
@media only screen and (max-width: 600px) {  
    body {  
        background-color: lightblue;  
    }  
}
```

- Validando a inclusão do ficheiro:

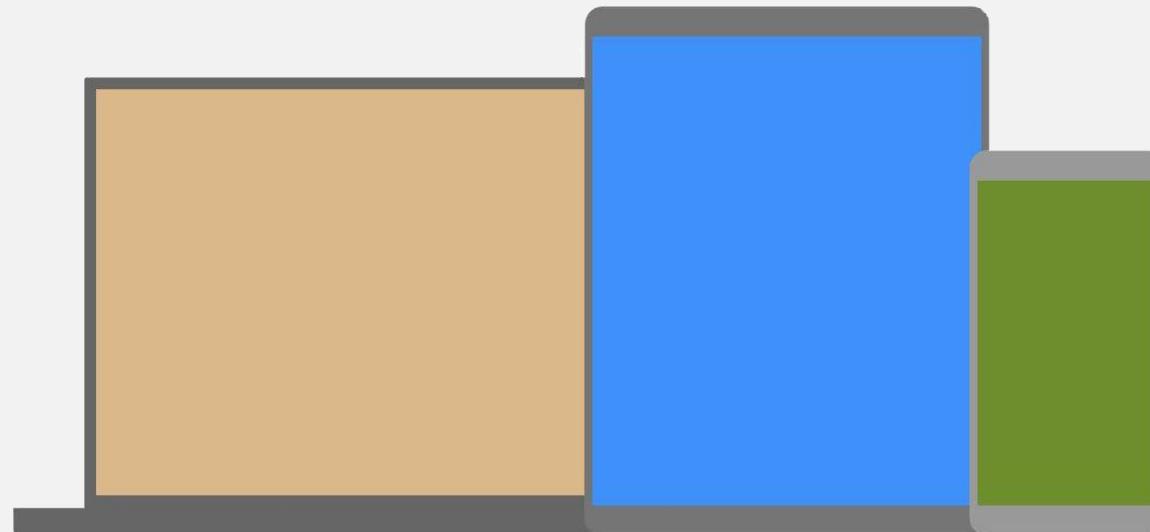
```
<link href="path.css" media="(validação)" />
```

```
<head>  
    <link rel="stylesheet" type="text/css" href="theme.css">  
    <link rel="stylesheet" type="text/css" href="print.css" media="print">  
</head>
```

14. Media Queries

- **CSS3 Media Types**

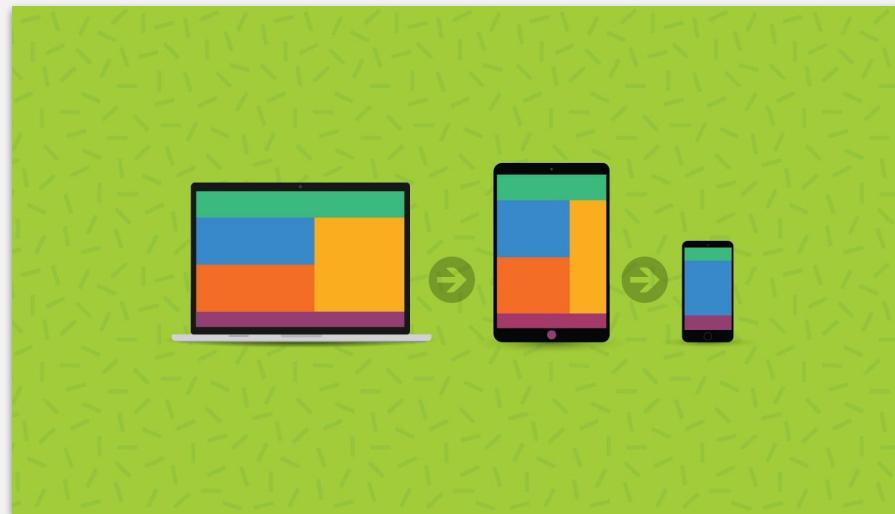
- **all**: usado para todo o tipo de dispositivos
- **print**: usado para impressoras
- **screen**: usado para ecrans de computadores, tablets, smart-phones etc.
- **speech**: usado para *screen readers* que leem as páginas através de som



14. Media Queries

- **CSS3 Media Features**

- **max-width:** largura máxima do browser
- **min-width:** largura mínima do browser
- **max-device-width:** largura máxima do dispositivo
- **min-device-width:** largura mínima do dispositivo
- **orientation:** orientação do dispositivo
- **device-pixel-ratio:** rácio de resolução



14. Media Queries

- Exemplos

```
@media all and (max-width: 960px) { ... }
```

```
@media all and (orientation: landscape) { ... }
```

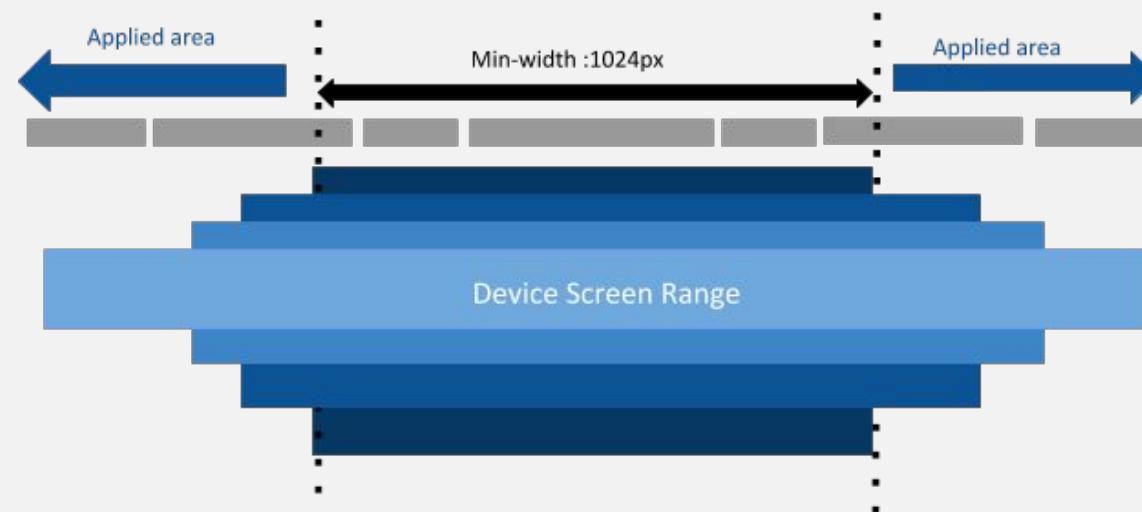
```
@media all and (device-pixel-ratio: 2) { ... }
```

```
@media all and (max-width: 320px) and (orientation:portrait) { ... }
```

14. Media Queries

- **min-width**

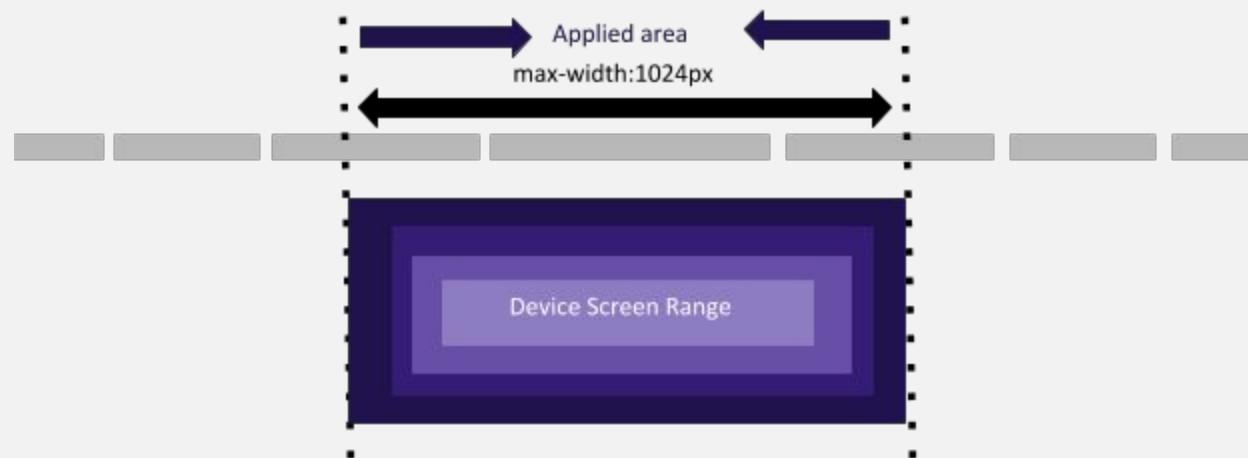
```
@media (min-width: 1024px) {  
    #ButtonWrapper {  
        width: 100%;  
    }  
}
```



14. Media Queries

- **max-width**

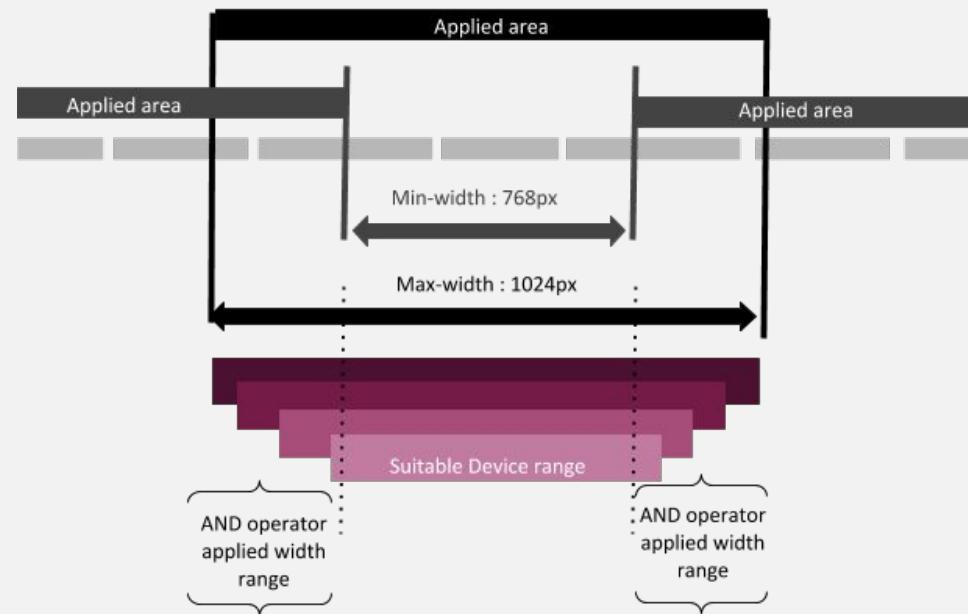
```
@media (max-width: 1024px) {  
    #ButtonWrapper {  
        width: 70%;  
    }  
}
```



14. Media Queries

- Operador lógico “and”

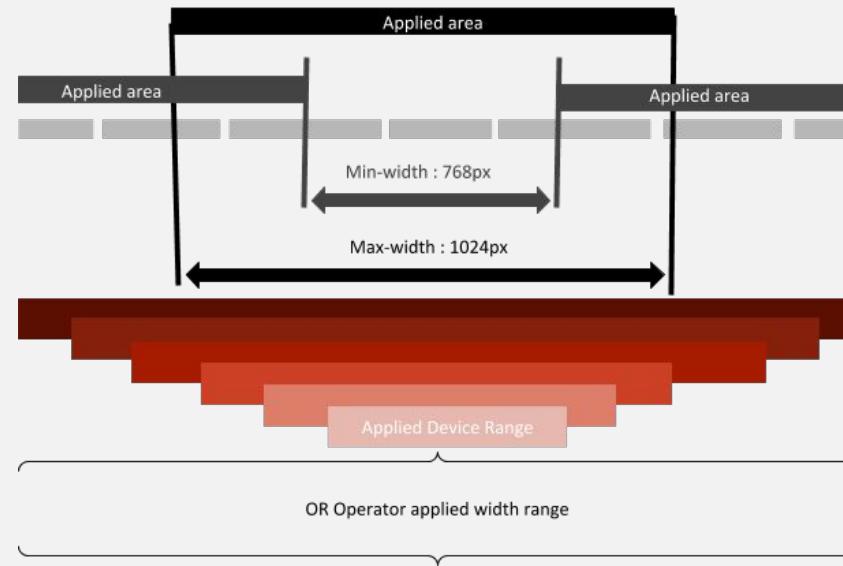
```
@media screen and (min-width: 768px) and (max-width: 1024px) {  
    #ButtonWrapper {  
        width: 70%;  
    }  
}
```



14. Media Queries

- Operador lógico “or”
 - O operador lógico “or” é conseguido através do uso de vírgulas (,)

```
@media screen and (min-width: 768px), (max-width: 1024px) {  
    #ButtonWrapper {  
        width: 80%;  
    }  
}
```



14. Media Queries

- **Operador lógico “not”**
 - é possível aplicar lógica inversa com o operador **not**

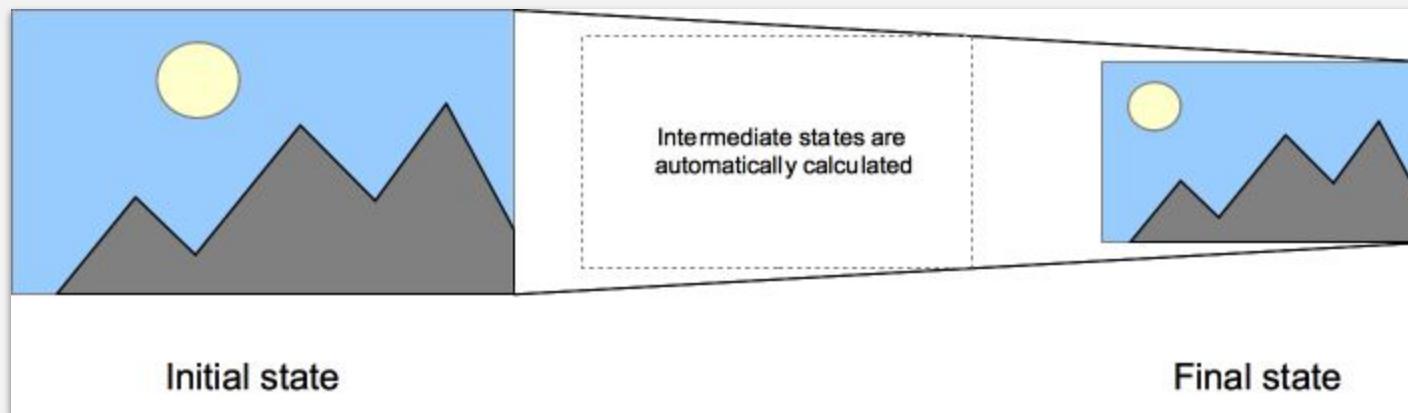
```
@media not all and (max-width: 600px) {  
    html { background: red; }  
}
```

Módulo 3 - CSS

15. Transições

15. Transições

- **Permitem animar propriedades de CSS**
 - Em vez de estarem num determinado estado e mudarem imediatamente de valor, a alteração de estado é feita de forma gradual, por um determinado período de tempo.



15. Transições

- Permitem definir:
 - As propriedades a animar: **transition-property**
 - Quanto tempo devem demorar: **transition-duration**
 - Tipo de aceleração da transição: **transition-timing-function**
 - Quando devem animar: **transition-delay**
- Mais comum usar o shorthand **transition**

```
#box {  
    transition-property: opacity;  
    transition-duration: 3s;  
    transition-timing-function: linear;  
    transition-delay: 2s;  
}
```

15. Transições

transition: propriedade duração tipo-de-animação atraso

```
div.with-transition {  
    transition: background-color 2s linear 0s;  
}  
span.with-transition {  
    transition: color 3s ease 0.5s;  
}  
div#with-transition {  
    transition: width 0.5s ease 2s;  
}
```

- **Lista de propriedades animáveis:**
 - https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_animated_properties

15. Transições

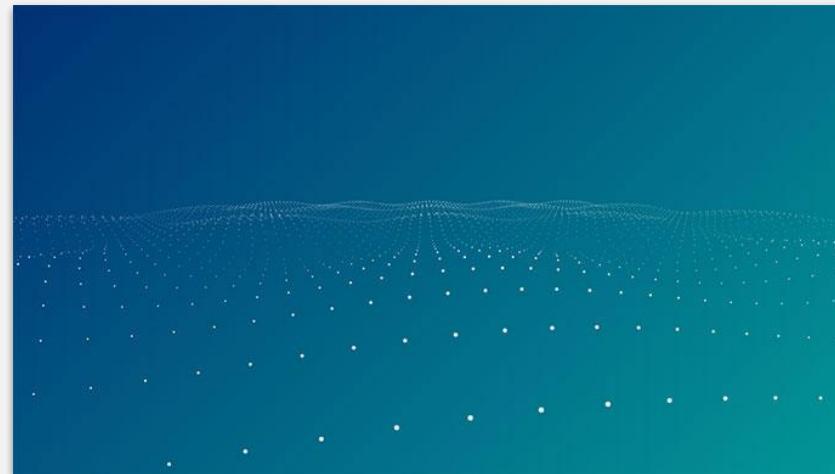
- **Tipo de animação:**
 - **ease**: início lento, fase intermédia rápida, final lento.
 - **linear**: velocidade constante
 - **ease in**: início lento, final rápido.
 - **ease out**: início rápido, final lento.
 - **ease in out**: identico a ease, mas com curvas de aceleração/desaceleração mais pronunciadas.
- Exemplos:
 - https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Transitions/Using_CSS_transitions

Módulo 3 - CSS

16. Backgrounds

16. Backgrounds

- As propriedades **background** de CSS são usadas para definir os efeitos em plano de fundo dos elementos
- Propriedades existentes:
 - **background-color**
 - **background-image**
 - **background-repeat**
 - **background-position**
 - **background-attachment**



16. Backgrounds

- **Background-color**

```
body {  
    background-color: lightblue;  
}
```

Hello World!

This page has a light blue background color!

16. Backgrounds

- **Background-image**

```
body {  
    background-image: url("paper.gif");  
}
```

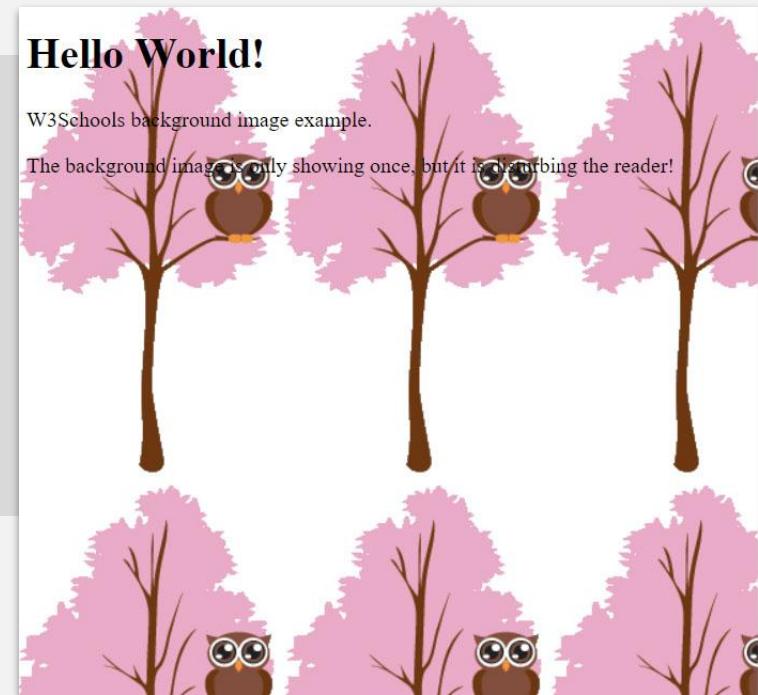
Hello World!

This page has an image as the background!

16. Backgrounds

- **Background-repeat**
 - Definição das repetições:
 - repeat
 - no-repeat
 - repeat-x
 - repeat-y

```
body {  
    background-image: url("img_tree.png");  
    background-repeat: repeat;  
}
```



16. Backgrounds

- **Background-position**

```
body {  
    background-image: url("img_tree.png");  
    background-repeat: no-repeat;  
background-position: right top;  
    margin-right: 200px;  
}
```

Hello World!

W3Schools background no-repeat, set position example.

Now the background image is only shown once, and positioned away from the text.

In this example we have also added a margin on the right side, so the background image will never disturb the text.



16. Backgrounds

- **Background-attachment**
 - Define se a imagem de background deve ser **fixa** na página (caso seja fixa não faz scroll com o resto da página).

```
body {  
background-image: url("img_tree.png");  
background-repeat: no-repeat;  
background-position: right top;  
margin-right: 200px;  
background-attachment: fixed;  
}
```

The background-attachment Property

The background-attachment property specifies whether the background image should scroll or be fixed (will not scroll with the rest of the page).

Tip: If you do not see any scrollbars, try to resize the browser window.

The background-image is fixed. Try to scroll down the page.

The background-image is fixed. Try to scroll down the page.

The background-image is fixed. Try to scroll down the page.



16. Backgrounds

- Shorthand **background**
 - Permite definir todas as propriedades background numa só instrução.

```
body {  
    background: #ffffff url("img_tree.png") no-repeat right top;  
}
```

Módulo 3 - CSS

17. Validação

17. Validação

<http://jigsaw.w3.org/css-validator/>