



Programação para Internet

Módulo 2

Fundamentos da Linguagem CSS

Prof. Dr. Daniel A. Furtado - FACOM/UFU

Conteúdo protegido por direito autoral, nos termos da Lei nº 9 610/98

A cópia, reprodução ou apropriação deste material, total ou parcialmente, é proibida pelo autor

Conteúdo do Módulo

- Introdução à CSS
- Seletores, pseudo-classes e pseudo-elementos
- Propriedades de ajuste de fonte e texto
- Unidades absolutas e relativas
- Ajustes de background, margem, padding e bordas
- Exibição e posicionamento
- Cascade, especificidade e herança
- Ícones SVG e Fontes de Ícones

O que é CSS?

- Linguagem de estilos para definir aspectos de apresentação
- **Cascading Style Sheets** - Folhas de Estilo em Cascata
- HTML se preocupa com o **conteúdo** da página
- CSS se preocupa com os **estilos** de apresentação desse conteúdo
- Separação entre conteúdo e apresentação gráfica
- Desenvolvida pelo *CSS Working Group* no W3C

Por que estudar CSS?

- Tecnologia de base
- Solidez e maturidade
- Independência dos frameworks
- Maior controle dos estilos e layouts

Página com código HTML puro, sem CSS



Prof. Daniel A. Furtado



- [INÍCIO](#)
- [ENSINO](#)
- [PROJETOS](#)
- [PUBLICAÇÕES](#)

Programação para Internet

A disciplina tem como objetivo capacitar o aluno para o desenvolvimento de aplicações Web utilizando as tecnologias de base, com foco no desenvolvimento do front-end e na programação direta do back-end, incluindo acesso a banco de dados.

Os objetivos específicos incluem: 1) discutir o funcionamento de sistemas Web e os protocolos envolvidos; 2) discutir o paradigma da programação para a Web e 3) desenvolver interfaces gráficas para a Web; 4) desenvolver websites dinâmicos e interativos através da programação direta do back-end e 5) utilizar conceitos e tecnologias para acesso a banco de dados em sistemas Web.

Sistema de Avaliação

Serão aplicadas três avaliações práticas, um projeto de implementação e vários testes de aula. As avaliações práticas devem ser realizadas em horário de aula, sob supervisão do professor. O projeto de implementação deverá ser apresentado pela equipe no final do semestre letivo, conforme cronograma disponibilizado pelo professor.

Sistemas de Banco de Dados

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Magnam delectus impedit at molestias commodi autem odit maxime necessitatibus nam, accusantium velit natus, minus veritatis. Ex repellendus earum totam similique porro. Magnam delectus impedit at molestias commodi autem odit maxime necessitatibus nam, accusantium velit natus, minus veritatis.

Lorem ipsum dolor sit amet consectetur adipiscing elit. Et quidem illum temporibus consequatur in sequi consectetur minus nobis. Provident maiores repellendus architecto aliquid quos quas magnam nobis. Perspiciatis, molestias ratione.

Tutoriais

- [Modelagem](#)
- [DERs](#)
- [SBDs](#)
- [Mozilla Dev](#)
- [WHATWG](#)
- [W3C](#)
- [W3Schools](#)
- [DERs](#)
- [SBDs](#)
- [Mozilla Dev](#)
- [WHATWG](#)

© Copyright 2020. Todos os direitos reservados.

Página com o mesmo código HTML, porém com acréscimo de CSS



Prof. Daniel A. Furtado



[INÍCIO](#) [ENSINO](#) [PROJETOS](#) [PUBLICAÇÕES](#)

Programação para Internet

A disciplina tem como objetivo capacitar o aluno para o desenvolvimento de aplicações Web utilizando as tecnologias de base, com foco no desenvolvimento do front-end e na programação direta do back-end, incluindo acesso a banco de dados.

Os objetivos específicos incluem: 1) discutir o funcionamento de sistemas Web e os protocolos envolvidos; 2) discutir o paradigma da programação para a Web e 3) desenvolver interfaces gráficas para a Web; 4) desenvolver websites dinâmicos e interativos através da programação direta do back-end e 5) utilizar conceitos e tecnologias para acesso a banco de dados em sistemas Web.

Sistema de Avaliação

Serão aplicadas três avaliações práticas, um projeto de implementação e vários testes de aula. As avaliações práticas devem ser realizadas em horário de aula, sob supervisão do professor. O projeto de implementação deverá ser apresentado pela equipe no final do semestre letivo, conforme cronograma disponibilizado pelo professor.

Sistemas de Banco de Dados

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Magnam delectus impedit at molestias commodi autem odit maxime necessitatibus nam, accusantium velit natus, minus veritatis. Ex repellendus earum totam similique porro. Magnam delectus impedit at molestias commodi autem odit maxime necessitatibus nam, accusantium velit natus, minus veritatis.

Lorem ipsum dolor sit amet consectetur adipiscing elit. Et quidem illum temporibus consequatur in sequi consectetur minus nobis. Provident maiores repellendus architecto aliquid quos quas magnam nobis. Perspiciatis, molestias ratione.

© Copyright 2020. Todos os direitos reservados.

Tutoriais

- [Modelagem](#)
- [DERs](#)
- [SBDs](#)
- [Mozilla Dev](#)
- [WHATWG](#)
- [W3C](#)
- [W3Schools](#)
- [DERs](#)
- [SBDs](#)
- [Mozilla Dev](#)
- [WHATWG](#)

Três Formas de Inserir CSS

1. Embutido na linha (inline)

- Atributo `style`
- O uso deve ser `evitado` (difícil manutenção)

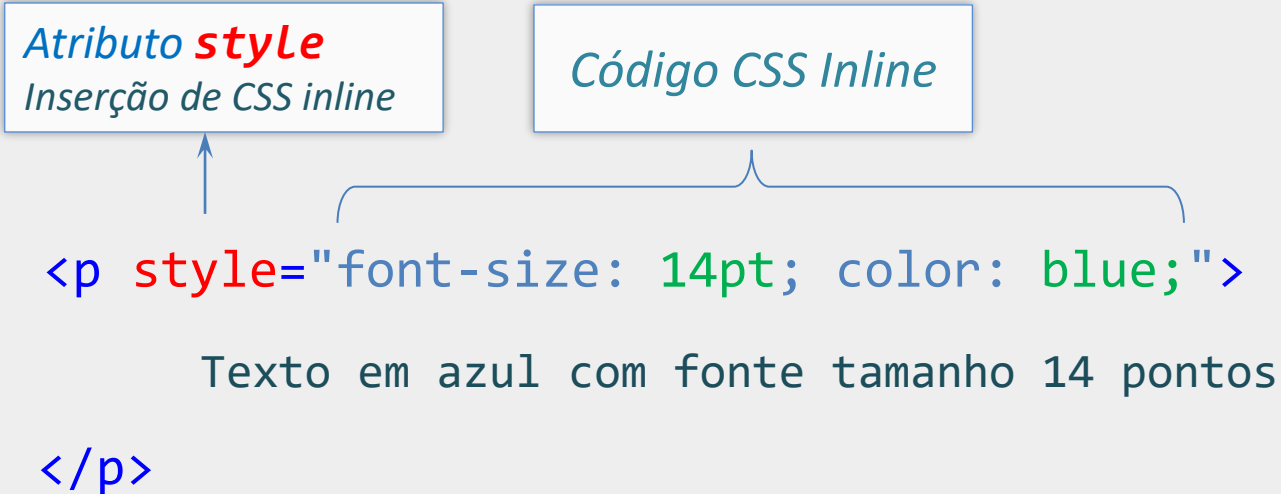
2. Folha de estilos embutida no HTML (interno)

- Utiliza o elemento `<style>` dentro do `<head>`
- Estilos específicos da página, não compartilhados

3. Folha de estilos em arquivo separado (externo)

- Utiliza o elemento `<link>` para referenciar um arquivo com código CSS
- Várias páginas podem utilizar o código CSS do arquivo
- Melhor separação entre conteúdo e estilos

CSS Embutido na Linha (*inline*)



O código CSS afetará apenas o elemento em questão.
Não é uma boa prática.

CSS em Folha de Estilos Embutida

```
<html>

  <head>

    <style>
      p {
        font-size: 14pt;
        color: blue;
      }
    </style>

  </head>

  <body>
    <p> ... </p>
  </body>

</html>
```

Código CSS embutido dentro do elemento `<style>`, no cabeçalho do documento HTML

CSS em Folha de Estilos Separada

Arquivo HTML

```
<html>

  <head>

    <link rel="stylesheet" href="style.css">

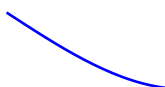
  </head>

  <body>

    <p> ... </p>

  </body>

</html>
```



Arquivo CSS

```
/* Arquivo style.css */

p {
  font-size: 14pt;
  color: blue;
}
```

- ✓ Provê melhor separação de conteúdo (HTML) e estilos (CSS).
- ✓ Permite que várias páginas utilizem o mesmo código CSS.
- ✓ Maior facilidade de manutenção.

Validação do Código CSS

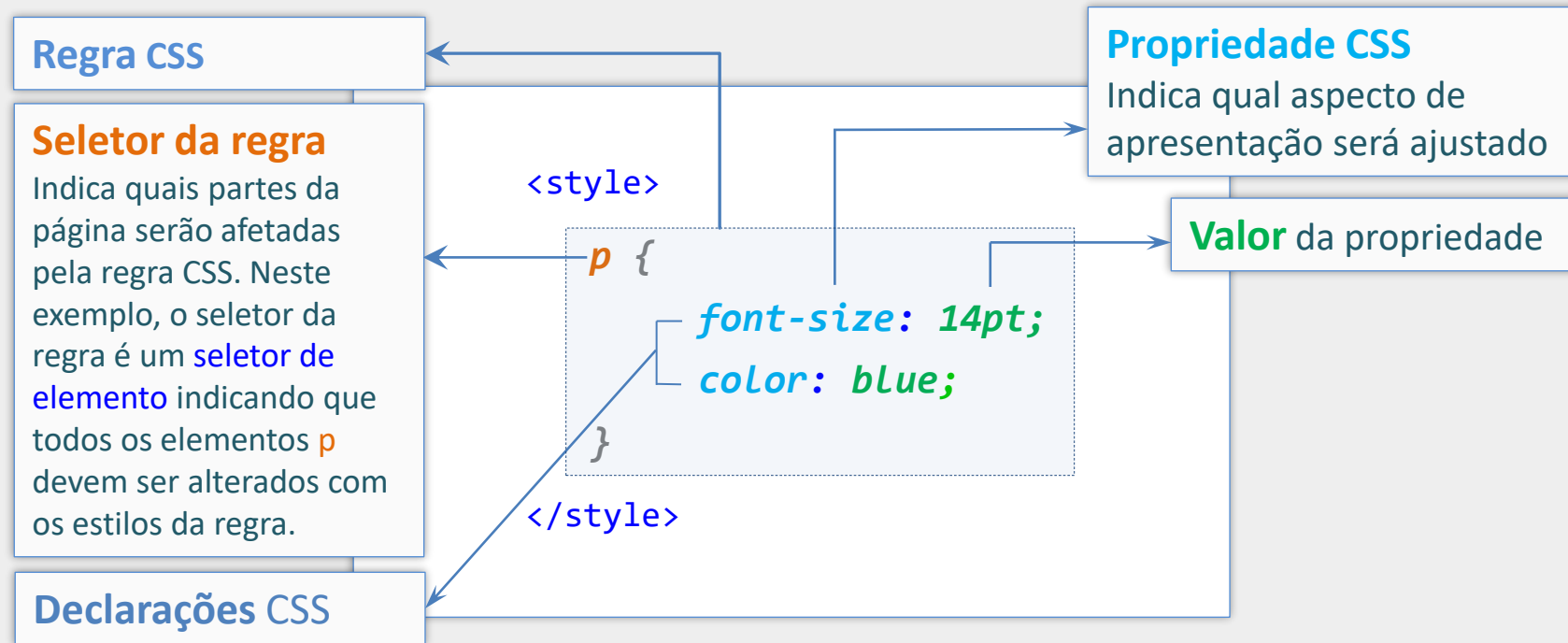
- Exibição adequada no navegador não é garantia de código correto
 - O navegador pode ocultar erros e inconsistências
- Código fora da especificação pode trazer problemas diversos
 - Apresentação inconsistente e imprevisível nos navegadores
- Ferramenta oferecida pelo W3C para validação de código CSS
 - jigsaw.w3.org/css-validator/

CSS e Cache do Navegador

- Eventualmente o navegador pode armazenar estilos CSS em memória
- Neste caso, mudanças nos estilos CSS podem não ter efeito imediato
- Se preciso, tecle **Ctrl-F5** para forçar o navegador a recarregar os estilos
- Outra possibilidade é excluir os dados de navegação (cache) do navegador

Regras, Seletores, Pseudo-Classes e Pseudo-Elementos

Regra, Seletor e Propriedades



Múltiplas Regras e Seletores

```
...  
<style>  
  body {  
    background-color: gray;  
  }  
  
  p {  
    font-size: 20pt;  
    color: blue;  
  }  
  
  h1 {  
    font-family: Verdana;  
  }  
</style>  
...
```

Afetar  o corpo do documento HTML

Afetar  todos os par grafos `p`

Afetar  todos os t tulos `h1`

  poss vel ter m ltiplas regras para estilizar individualmente diferentes partes do documento.

Agrupando Seletores

```
...  
<style>  
  
  h1 {  
    font-family: Verdana;  
    color: darkblue;  
  }  
  
  h2 {  
    font-family: Verdana;  
    color: darkblue;  
  }  
  
  h3 {  
    font-family: Verdana;  
    color: darkblue;  
  }  
  
</style>  
...
```



```
...  
<style>  
  
  h1, h2, h3 {  
    font-family: Verdana;  
    color: darkblue;  
  }  
  
</style>  
...
```

Ao invés de criar várias regras com os **mesmos estilos**, pode-se criar uma única regra agrupando os seletores (tornando o código mais conciso e facilitando a manutenção).

Outros Tipos de Seletores

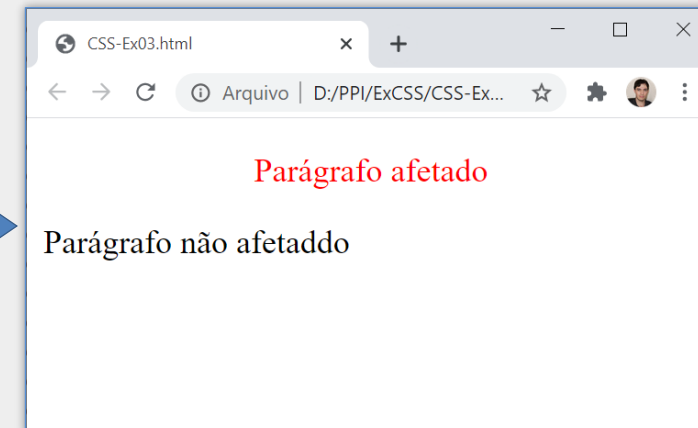
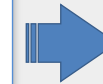
- Além do seletor de elemento apresentado anteriormente, há várias outras formas de indicar quais partes da página devem ser afetadas pela regra CSS
- Algumas dessas formas são:
 - Seletor de ID
 - Seletor de filho
 - Seletor de descende
 - Seletor de irmão adjacente
 - Seletor de irmão geral
 - Seletor de atributo
 - Seletor de classe

Seletor de ID (*#idDoElemento*)

O seletor de ID pode ser utilizado quando se deseja aplicar estilos a apenas **um elemento em particular**. Utiliza-se **#** seguido do **id** do elemento

```
<style>
  #par1 {
    text-align: center;
    color: red;
  }
</style>

<body>
  <p id="par1">Parágrafo afetado</p>
  <p>Parágrafo não afetado</p>
</body>
```



Afeta apenas o elemento que tem o **id** indicado no seletor. Vale lembrar que os **id's** devem ser únicos na página.

Seletor de Filho ($x > y$)

O seletor de filho tem a sintaxe $x > y$ e afeta apenas os elementos y que são filhos de elementos x

```
...
<style>
  li > a {
    text-transform: uppercase;
  }
</style>
...
<body> Link não afetado
  <a href="#">Link 1</a>
  <ul>
    <li> <a href="#">Link 2</a> </li>
    <li> <a href="#">Link 3</a> </li>
  </ul> Links afetados
...
```

Afetará todos os elementos `<a>` que são filhos de elementos `` (Link 2 e Link 3)

O primeiro link não é afetado porque o elemento `<a>` não é filho de nenhum ``, ou seja, não está diretamente dentro de um ``.

Seletor de Descendente (*x y*)

Afeta os elementos **y** que **estão dentro** de elementos **x**, mesmo que tenha outros elementos aninhados entre eles

```
...


Afetar  todos os elementos <a> que est o dentro de elementos <div>, direta ou indiretamente (Link 2 e Link 3)



Se o seletor de filho tivesse sido usado neste exemplo, apenas o Link 2 seria afetado. O Link 3 n o seria afetado porque apesar de ser descendente de um <div>, o Link 3   filho do elemento <section>, e n o de um elemento <div>.



Programa o para Internet



Prof. Dr. Daniel A. Furtado - Proibida c pia, apropria o ou uso sem autoriza o de qualquer parte deste material - Lei n  9 610/98



19


```

Seletor de Irmão Adjacente ($x + y$)

Afeta todo elemento y que aparece **imediatamente depois** de x

```
...
h1 + p {
  color: red;
}
...
<body>
  <h1>Título 1</h1>
  <p>Parágrafo 1 </p>
  <p>Parágrafo 2 </p>

  <h1>Título 2</h1>
  <p>Parágrafo 3</p>
  <p>Parágrafo 4</p>
  ...
```

Afetrá todo parágrafo que está **imediatamente depois** de um `<h1>`

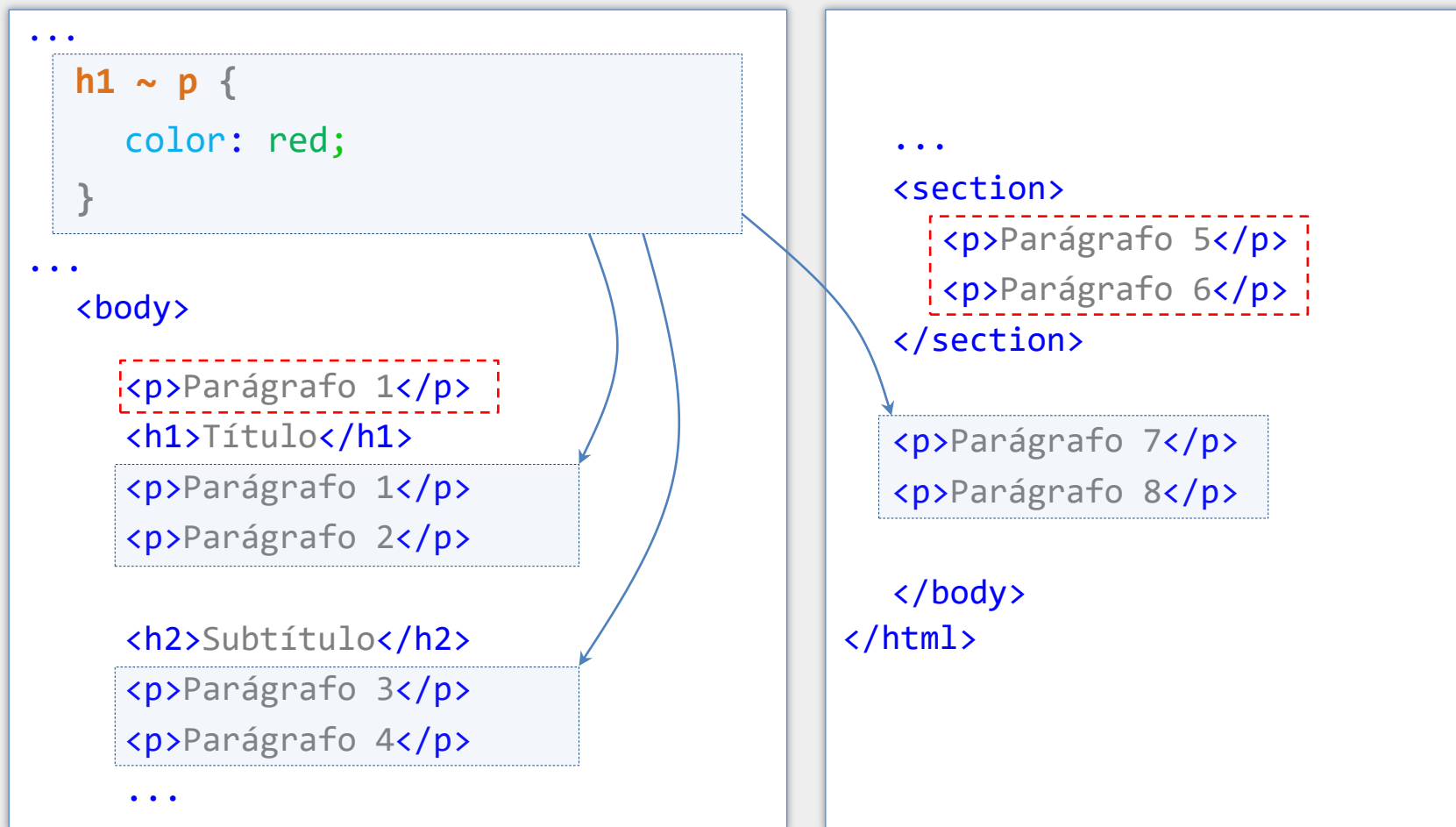
```
...
<h1>Título 3</h1>
<h2>Subtítulo</h2>
<p>Parágrafo 5</p>
<p>Parágrafo 6</p>

</body>
</html>
```

OBS: Eventuais comentários separando os irmãos não interferiria no seletor.

Seletor de Irmão Geral ($x \sim y$)

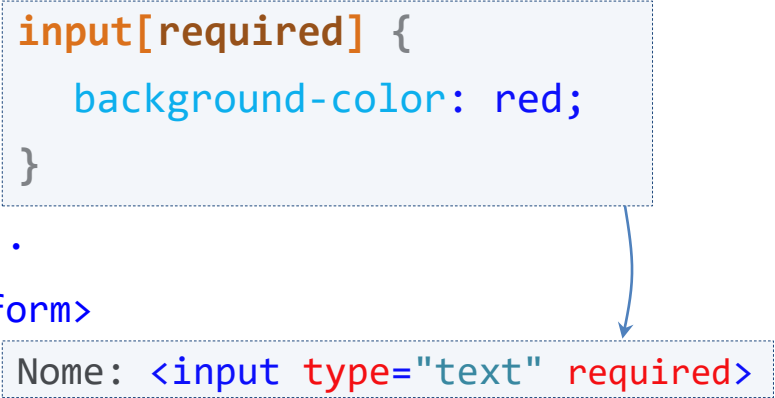
Afeta todos os elementos **y** que **são irmãos** de **x** e aparecem **depois** de **x**



Seletor de Atributo *x[atributo]*

Seleciona elementos de acordo com alguma condição envolvendo seus atributos

Exemplo 1

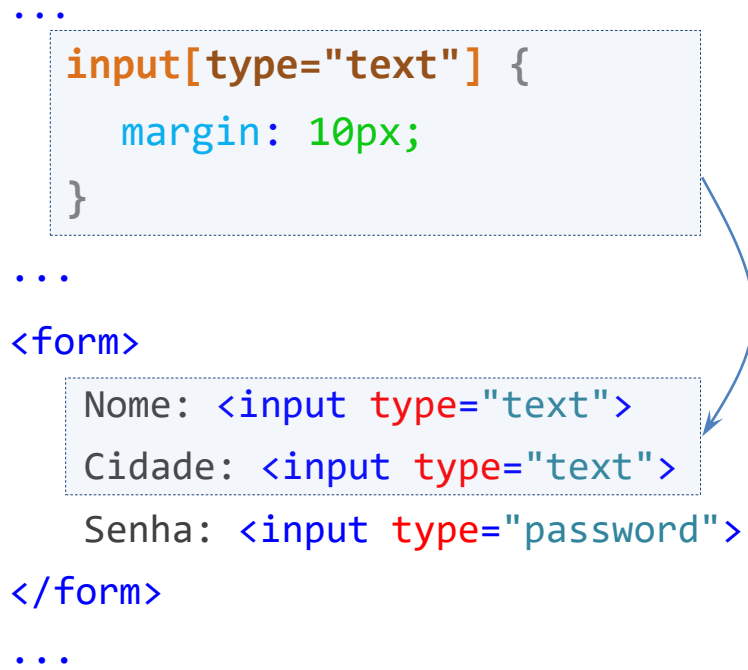
```
...
A diagram illustrating the CSS selector input[required]. A light blue box with a dashed border contains the selector and its rule: input[required] { background-color: red; }. A blue arrow points from the bottom of this box to the required attribute in the <input type="text" required> element within a <form> block in the HTML code below. The required attribute in the HTML is also highlighted with a dashed border.
```

Seleciona os elementos `<input>` que possuem o atributo `required`

Seletor de Atributo *x[atributo]*

Exemplo 2

```
...  
input[type="text"] {  
    margin: 10px;  
}  
...  
<form>  
    Nome: <input type="text">  
    Cidade: <input type="text">  
    Senha: <input type="password">  
</form>  
...
```

A diagram illustrating the CSS selector `input[type="text"]`. A light blue box with a dashed border contains the selector and its rule: `input[type="text"] { margin: 10px; }`. A blue arrow points from this box to the `<input type="text">` elements in the HTML code below. The HTML code shows a `<form>` containing three input fields: "Nome: <input type='text'>", "Cidade: <input type='text'>", and "Senha: <input type='password'>". The `type="text"` attributes in the first two inputs are highlighted in red, matching the `type="text"` in the CSS selector.

Seleciona os elementos `<input>`
cujo atributo `type` tem o valor `text`

Seletor de Classe

- O seletor de classe é um dos tipos de seletores mais utilizados
- Ideal para situações onde se pretende aplicar os estilos mais de uma vez
- Primeiramente, deve-se criar uma **classe** de estilos CSS com os estilos desejados. Utiliza-se o caractere “ponto” seguido do nome da classe:

```
.minhaClasseCSS {  
    /* declarações CSS */  
}
```

- Posteriormente, aplica-se os estilos da classe no elemento desejado utilizando o atributo **class**:

```
<elemento class="minhaClasseCSS">
```


Seletor de Classe – Exemplo

...

```
.destacado {  
  color: blue;  
  text-transform: uppercase;  
}
```

...

```
<h1 class="destacado">Título 1</h1>
```

```
<p>Parágrafo 1 </p>
```

```
<p class="destacado">Parágrafo 2 </p>
```

```
<p>Parágrafo 3 </p>
```

```
<p>Parágrafo 4 </p>
```

```
<p class="destacado">Parágrafo 5 </p>
```

...

Neste exemplo a classe `.destacado` está sendo aplicada no primeiro título e nos parágrafos 2 e 5, fazendo com que eles apareçam em letras maiúsculas e na cor azul

Seletor de Classe – Exemplo

É possível criar uma classe que poderá ser aplicada a apenas um determinado tipo de elemento

```
...  
p.destacado {  
  color: blue;  
  text-transform: uppercase;  
}  
...  
<h1 class="destacado">Título 1</h1>  
<p>Parágrafo 1 </p>  
<p class="destacado">Parágrafo 2 </p>  
...
```

`p.destacado` é uma classe que poderá ser utilizada apenas em elementos `p`.

A classe não terá efeito caso seja utilizada em outros tipos de elementos (neste exemplo, não terá efeito no título `h1`).

Seletor de Classe – Exemplo

```
...  
p.destacado {  
  color: blue;  
  font-style: italic;  
}  
  
h1.destacado {  
  color: orange;  
  text-transform: uppercase;  
}  
...  
  
<h1 class="destacado">Título 1</h1>  
<p class="destacado">Parágrafo 2 </p>  
...
```

Classes específicas de elementos nos dá a possibilidade de definir mais de uma classe utilizando o mesmo nome. Neste exemplo, o título aparecerá em laranja e maiúsculas e o parágrafo aparecerá em azul e itálico.

Seletor de Classe – Exemplo

É possível utilizar outros seletores em conjunto com classes

```
...  
.listaHorizontal > li {  
  background-color: black;  
  margin: 3px;  
  display: inline-block;  
}
```

```
...  
<ul class="listaHorizontal">  
  <li>Home</li>  
  <li>Galeria</li>  
  <li>Sobre</li>  
</ul>  
<ul>  
  <li>A</li>  
  <li>B</li>  
</ul>  
...
```

Afetaré apenas os ``'s
que são filhos de elementos
utilizando a classe
`.listaHorizontal`

Referenciando Múltiplas Classes

É possível aplicar os estilos de **múltiplas classes** em um elemento. Basta utilizar um **espaço** para separar os nomes das classes no atributo **class**

```
...  
.destacado {  
  color: blue;  
  font-style: italic;  
}  
.centralizado {  
  text-align: center;  
}  
...  
<h1 class="destacado centralizado">Título 1</h1>  
...
```

O título aparecerá em azul, itálico e centralizado.
Se houver repetição de propriedades, prevalecerão aquelas referenciadas por último.

Seletores com Pseudo-Classes

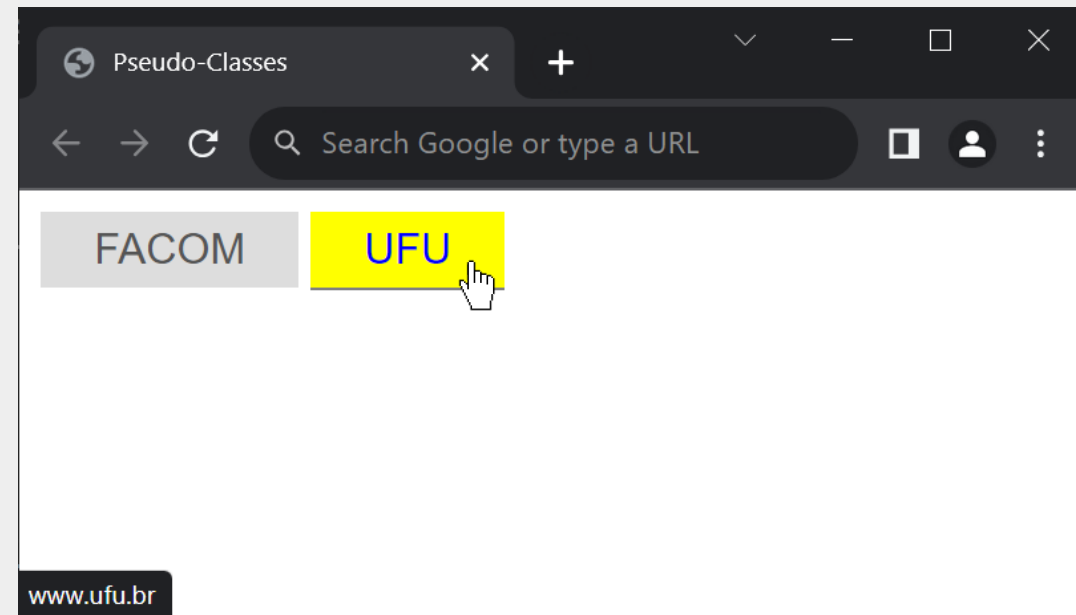
- Uma **pseudo-classe** permite alterar o estilo de um elemento caso ele se encontra em um **estado particular**
- Por exemplo, é possível alterar o estilo dos links **que já foram visitados** ou o estilo dos campos de formulário **com conteúdo inválido**
- Sintaxe: **seletor : pseudo-classe**

Pseudo-Classes Comumente Utilizadas com Links

Pseudo-Classe	Descrição	Exemplos
<code>:link</code>	Define o estilo inicial do link	<code>a:link { color: blue; }</code> links não visitados aparecerão em azul
<code>:visited</code>	Define o estilos de links já visitados	<code>a:visited { color: gray; }</code> links visitados aparecerão em cinza
<code>:hover</code>	Define o estilo de exibição do elemento quando o ponteiro do mouse está sobre ele	<code>a:hover { text-decoration: none; }</code> links aparecerão sem o <i>underline</i> quando o ponteiro do mouse estiver sobre ele
<code>:active</code>	Define o estilo de exibição do elemento quando ativado (botão do mouse pressionado sobre ele)	<code>a:active, p:active { color: gray; }</code> links e parágrafos aparecerão em cinza quando o usuário estiver com o botão do mouse pressionado sobre eles

Usando Pseudo-Classes para Alterar Estilo de Links

```
<style>
  a:link,
  a:visited {
    color: #555;
    background-color: #ddd;
    padding: 5px 20px;
    display: inline-block;
    text-align: center;
    text-decoration: none;
  }
  a:hover,
  a:active {
    background-color: yellow;
    color: blue;
    border-bottom: 1px solid gray;
  }
</style>
</head>
<body>
  <a href="http://www.facom.ufu.br">FACOM</a>
  <a href="http://www.ufu.br">UFU</a>
</body>
```

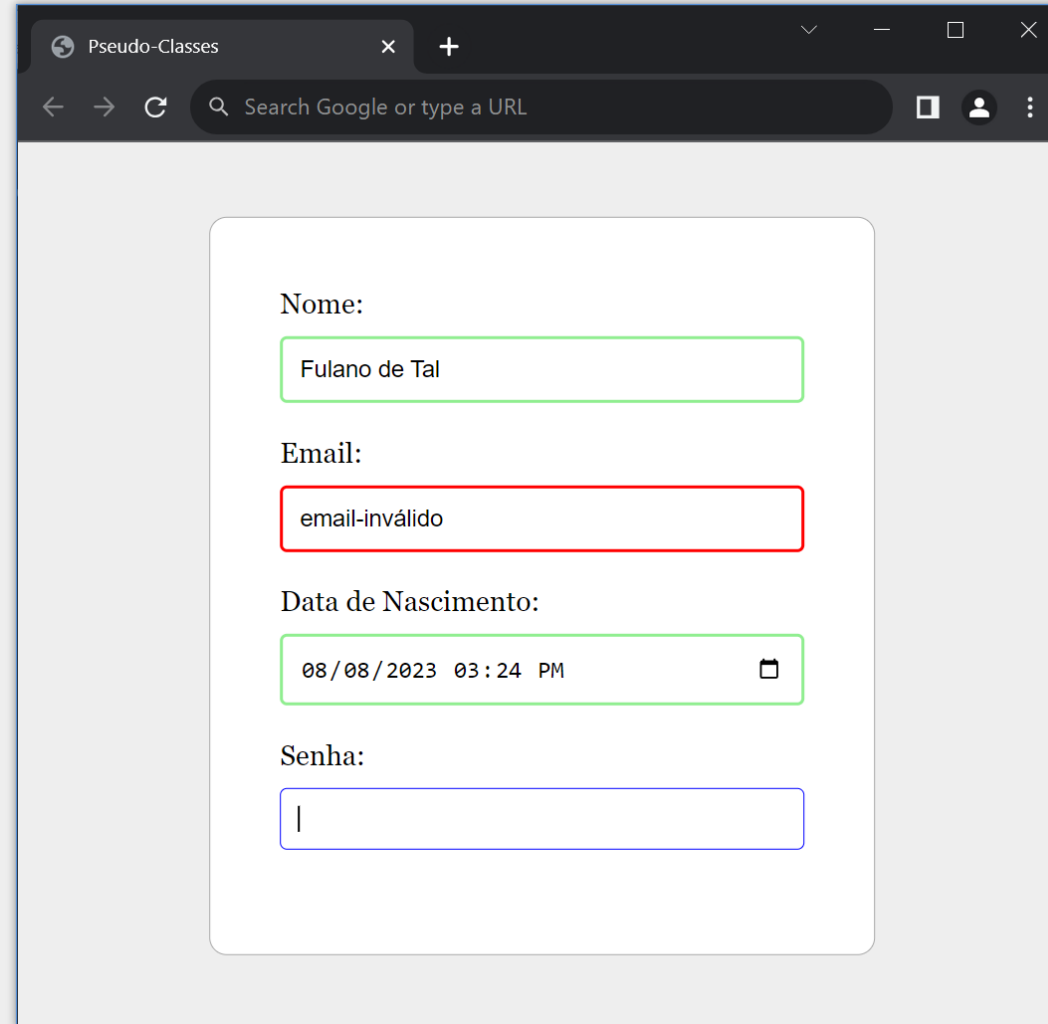


Pseudo-Classes Comuns em Campos de Formulário

Pseudo-Classe	Descrição	Exemplos
:valid	Define o estilo de campos de formulário quando estão no estado válido (conteúdo apropriado)	<code>input:valid { color: green; }</code>
:invalid	Define o estilo de campos quando estão no estado inválido (conteúdo no formato incorreto)	<code>input:invalid { color: red; }</code>
:checked	Define o estilo de campos do tipo <code>radio</code> , <code>checkbox</code> ou <code>option</code> quando selecionado	<code>radio:checked { color: gray; }</code>
:focus	Define o estilo do campo quando estiver em foco (campo em edição)	<code>input:focus {background-color: gray;}</code>

Usando Pseudo-Classes para Alterar Estilo de Campos

```
input:valid {  
  border: 2px solid lightgreen;  
}  
  
input:invalid {  
  border: 2px solid red;  
}  
  
input:focus {  
  border: 1px solid blue;  
  outline: none;  
}
```



Pseudo-Classes

Nome:

Email:

Data de Nascimento:

Senha:

Outros Exemplos de Pseudo-Classes

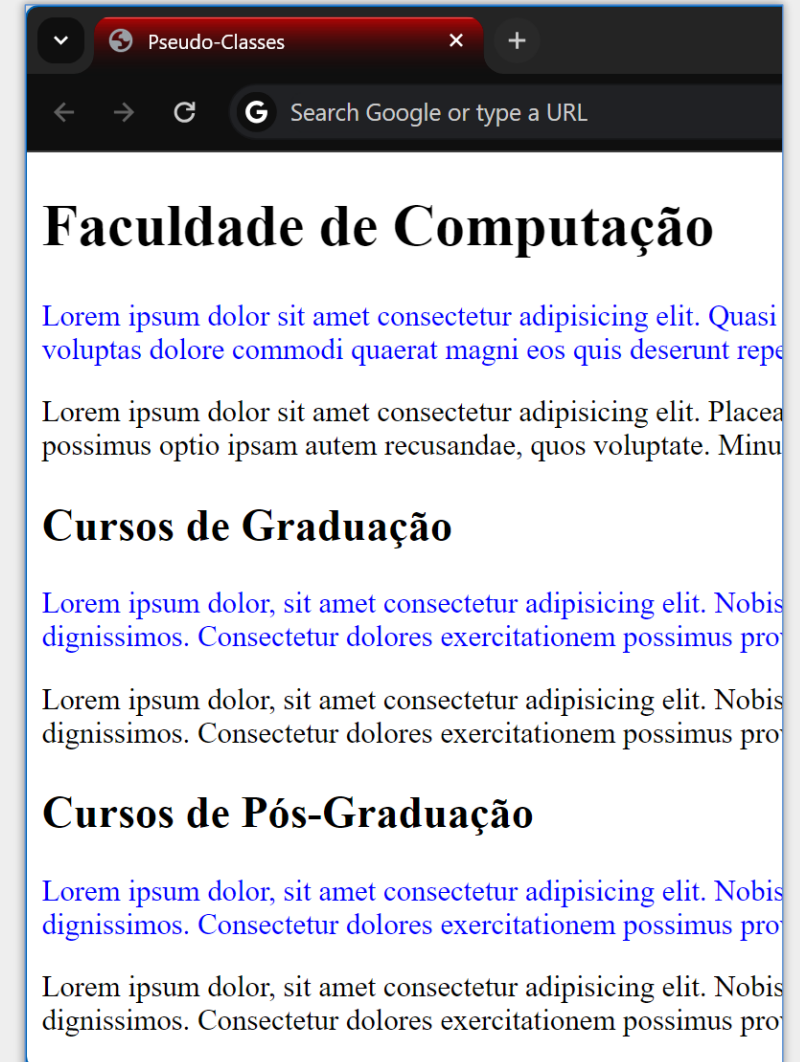
Pseudo-Classe	Descrição	Exemplos
<code>:first-child</code>	Permite estilizar o elemento que é o primeiro filho do elemento pai	<code>li:first-child { color: blue; }</code> altera o 1º item de cada lista
<code>:last-child</code>	Permite estilizar o elemento que é o último filho do elemento pai	<code>li:last-child { color: blue; }</code> altera o último item de cada lista
<code>:nth-child</code>	Permite estilizar o elemento que é o n-ésimo filho do elemento pai	<code>li:nth-child(2) { ... }</code> altera o 2º item de cada lista <code>tr:nth-child(odd) { ... }</code> altera as linhas ímpares da tabela <code>tr:nth-child(even) { ... }</code> altera as linhas pares da tabela <code>tr:nth-child(3n) { ... }</code> altera as linhas de nº 3, 6, 9, 12 etc.
<code>:first-of-type</code>	Permite alterar a primeira ocorrência de um elemento dentro de seu container	<code>p:first-of-type { color: red; }</code> altera a cor do 1º parágrafo do container, mesmo que tal parágrafo não seja o 1º filho do container (veja ex. a seguir)

Exemplo de :first-of-type

```
<style>
  p:first-of-type {
    color: blue
  }
</style>
```

Neste exemplo, `p:first-of-type` seleciona o primeiro parágrafo dentro de cada container. Se fosse utilizado `p:first-child` tais parágrafos não seriam selecionados, pois o 1º filho de cada `<section>` é um `<h2>` (e não um elemento `<p>`) e o 1º filho do `<main>` é um `<h1>`.

```
<body>
  <main>
    <h1>Faculdade de Computação</h1>
    <p>Lorem ipsum dolor sit amet conse</p>
    <p>Lorem ipsum dolor sit amet conse</p>
    <section>
      <h2>Cursos de Graduação</h2>
      <p>Lorem ipsum dolor, sit amet co</p>
      <p>Lorem ipsum dolor, sit amet co</p>
    </section>
    <section>
      <h2>Cursos de Pós-Graduação</h2>
      <p>Lorem ipsum dolor, sit amet co</p>
      <p>Lorem ipsum dolor, sit amet co</p>
    </section>
  </main>
</body>
```



Função de Pseudo-Classe :has()

- A função de pseudo-classe `:has()` permite selecionar elementos com base em outros elementos associados que atendem ao seletor passado para a função `has()`
- Por exemplo, é possível selecionar um **elemento pai** com base em uma condição envolvendo os **descendentes**
- Outra possibilidade é selecionar um elemento qualquer com base em uma condição envolvendo elementos **irmãos**

Função de Pseudo-Classe :has() – Exemplos

```
div:has( img) {  
  background-color: gray;  
}
```

Seleciona os elementos `div` que contêm um **descendente** `img`, alterando a cor de fundo desses `div`'s para cinza

```
div:has(> img) {  
  background-color: gray;  
}
```

Seleciona os elementos `div` que contêm um **filho** `img`, alterando a cor de fundo desses `div`'s para cinza

```
h1:has(+ p) {  
  text-transform: uppercase;  
}
```

Seleciona os títulos `h1` que contêm um parágrafo imediatamente depois (`p` como irmão adjacente)

Função de Pseudo-Classe :is()

- A função de pseudo-classe `:is()` recebe uma lista de seletores como argumento e seleciona os elementos que possam ser selecionados por qualquer seletor da lista
- Em alguns casos, permite criar seletores mais concisos

Altera a cor dos títulos `h1`, `h2` e `h3` que são descendentes de `section`, `article` ou `aside`

```
section h1, section h2, section h3,  
article h1, article h2, article h3,  
aside h1, aside h2, aside h3, {  
  color: gray;  
}
```



Seletor equivalente utilizando `:is`

```
:is(section, article, aside) :is(h1, h2, h3) {  
  color: gray;  
}
```

Pseudo-Elementos

- Permite seleccionar uma **parte específica** de um elemento
- Sintaxe geral: **elemento :: valor**

```
p::first-line {  
  text-transform: uppercase;  
}
```

A primeira linha de cada parágrafo será apresentada com letras maiúsculas

```
p::selected {  
  color: green;  
  background-color: black;  
}
```

O texto que o usuário selecionar nos parágrafos aparecerá na cor verde com fundo preto

Outros Pseudo-Elementos

```
input::placeholder {  
  color: red;  
}
```

Altera a cor dos textos de **placeholder** dos campos **input**

```
p::after {  
  content: 'exemplo after';  
}
```

Insere o texto 'exemplo after' como um pseudo-elemento **depois do conteúdo** do parágrafo

```
p::before {  
  content: 'exemplo before';  
}
```

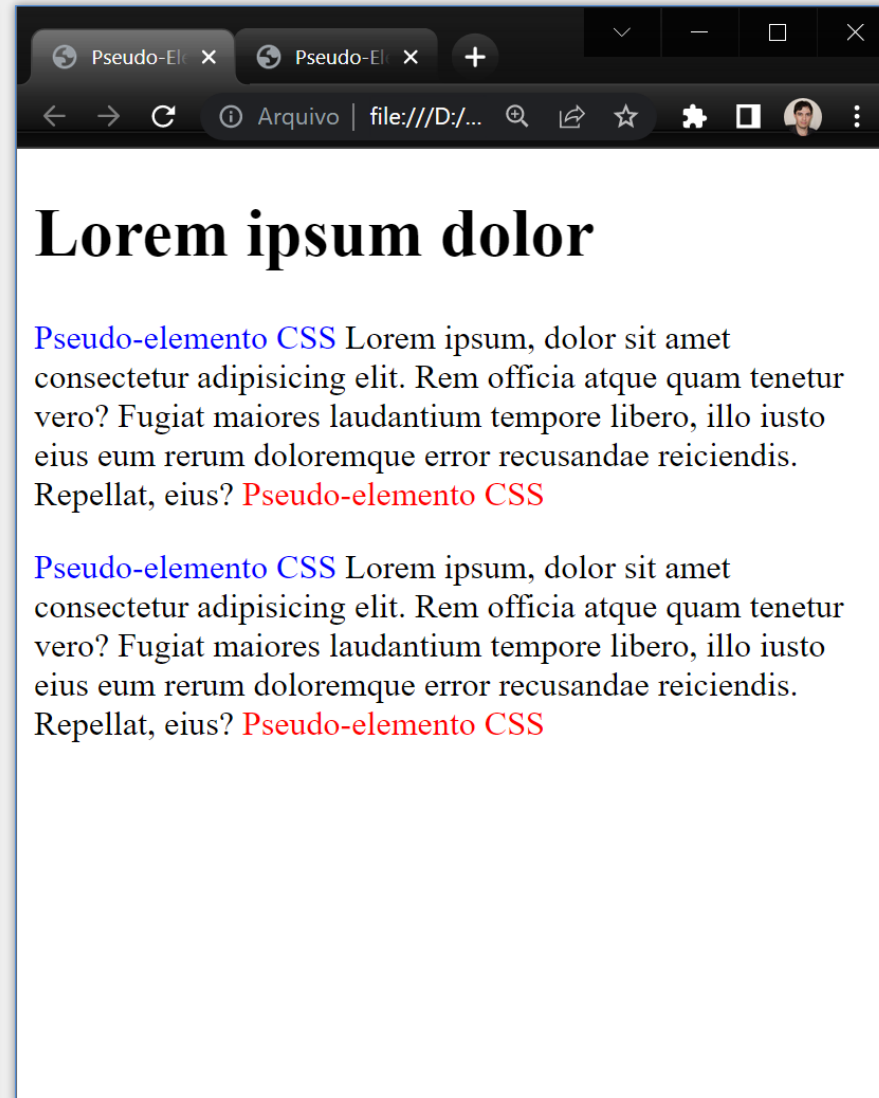
Insere o texto 'exemplo before' como um pseudo-elemento **antes do conteúdo** do parágrafo

OBS: Os pseudo-elementos **::after** e **::before** não podem ser utilizados em elementos sem conteúdo como **** ou **<input>**

Pseudo-Elementos - Exemplo

```
<style>
  p::after {
    content: ' Pseudo-elemento CSS ';
    color: ■ red;
  }

  p::before {
    content: ' Pseudo-elemento CSS ';
    color: ■ blue;
  }
</style>
</head>
<body>
  <h1>Lorem ipsum dolor</h1>
  <p>Lorem ipsum, dolor sit amet consectetur
    laudantium tempore libero, illo iusto
  <p>Lorem ipsum, dolor sit amet consectetur
    laudantium tempore libero, illo iusto
```



Pseudo-Elementos - Exemplo

```
input:invalid + span::after {
  content: 'X';
  color: red;
}

input:valid + span::after {
  content: '✓';
  color: limegreen;
}
```

```
<div>
  <label for="cpf">CPF (xxx.xxx.xxx-xx):</label>
  <input type="text" id="cpf" name="cpf" required
    pattern="\d{3}\.\d{3}\.\d{3}-\d{2}">
  <span></span> <!-- span para exibição de 'X' ou '✓' -->
</div>

<div>
  <label for="user">Nome de Usuário:</label>
  <input type="text" id="nome" name="nome" minlength="6"
    placeholder="Mínimo de 6 caracteres" required>
  <span></span> <!-- span para exibição de 'X' ou '✓' -->
</div>
```

Pseudo-Elementos

Search Google or type...

Atualização cadastral

CPF (xxx.xxx.xxx-xx): ✓

Nome de Usuário: ✗

E-mail: ✗

Data de Nascimento: ✗

Observações sobre Pseudo-Elementos

- A notação com 4 pontos (:::) foi introduzida em atualizações da CSS
- :: é a forma atualmente recomendada para pseudo-elementos, pois diferencia pseudo-elementos de pseudo-classes, uma vez que há diferença semântica significativa entre eles

Ajustes Textuais

Tipos de Fontes



Fonte Sans-serif
Sem prolongamentos
Ex.: Arial, Verdana



Fonte Serif
Com prolongamentos
Ex.: Times New Roman



Fonte Monospace
Letras com mesma largura de exibição
Ex.: Courier New, Consolas

Propriedades de Ajuste de Fonte

Propriedade	Descrição	Exemplo
<code>font-family</code>	define a fonte em si	<code>font-family: Verdana, Arial, sans-serif</code>
<code>font-style</code>	define o estilo da fonte	<code>font-style: italic</code>
<code>font-size</code>	define o tamanho da fonte	<code>font-size: 20px</code>
<code>font-weight</code>	define a espessura da letra	<code>font-weight: bold</code>
<code>font-variant</code>	define variantes da fonte	<code>font-variant: small-caps</code> TÍTULO
<code>font-stretch</code>	estica ou comprime a fonte	<code>font-stretch: expanded</code>
<code>line-height</code>	define o espaçamento entre linhas	<code>line-height: 1.5</code>

Repare que a propriedade `font-family` permite a indicação de uma lista de nomes de fontes. O navegador utilizará a próxima fonte dessa lista caso eventualmente a fonte anterior não seja encontrada no computador do usuário. Recomenda-se encerrar essa lista de fontes com o nome de uma família genérica como `sans-serif` ou `serif`, pois se nenhuma das fontes for encontrada, será utilizada a fonte padrão daquela família.

Propriedade Abreviada **font**

- Em CSS, uma **propriedade abreviada** é um tipo de propriedade que possibilita definir, de uma só vez, vários aspectos de apresentação (propriedades constituintes)
- **font** é uma propriedade abreviada que permite definir, em uma única linha, todos os aspectos relacionados à fonte, ou seja:
 - `font-family`
 - `font-size`
 - `font-style`
 - `font-weight`
 - `font-stretch`
 - `font-variant`
 - `line-height`

Propriedade Abreviada **font** – Sintaxe

`font:` *italic* *small-caps* *condensed* *bold* *16px* */1.5* *Arial*
font-style font-variant font-stretch font-weight font-size line-height font-family

obrigatório *obrigatório*

- Apenas os valores de `font-size` e `font-family` são obrigatórios
- `font-family` deve ser o último valor
- `font-style`, `font-variant` e `font-weight` devem vir antes de `font-size`
- `line-height` deve vir logo depois de `font-size`, acompanhado de `/`
- A omissão de uma propriedade constituinte retorna seu valor ao valor inicial
- Outros exemplos:
 - `font: 18px Verdana;`
 - `font: bold 16px Georgia;`
 - `font: italic 14px /1.5 Consolas;`

Propriedades de Ajuste de Texto

Propriedade	Descrição	Exemplos
<code>text-align</code>	alinhamento horizontal do texto	<code>text-align: left</code> <code>text-align: justify</code>
<code>vertical-align</code>	alinhamento vertical do texto	<code>vertical-align: top</code> <code>vertical-align: middle</code>
<code>text-decoration</code>	decoreção adicional	<code>text-decoration: none</code> <code>text-decoration: underline</code>
<code>text-indent</code>	recoo de 1ª linha	<code>text-indent: 30px</code>
<code>text-transform</code>	controle de maiúsculas e minúsculas	<code>text-transform: uppercase</code> <code>text-transform: lowercase</code> <code>text-transform: capitalize</code>
<code>color</code>	define a cor do texto	<code>color: green</code>

É possível ajustar a cor de várias formas

- Pelo nome da cor
 - color: `blue`, color: `darkblue`, color: `lightblue`, etc.
- Pelo valor RGB em Decimal (red, green, blue)
 - color: `rgb(0, 120, 255)`
 - color: `rgba(0, 120, 255, 0.5)` (50% translúcido; 0 = totalmente transp., 1 = opaco)
- Pelo valor RGB em Hexadecimal
 - Notação com 6 dígitos. Exemplo: `#FF0000`
 - Notação com 3 dígitos. Exemplo: `#AF5` (equivalente a `#AAFF55`)
 - Com transparência. Exemplo: `#FF000080` (50% translúcido)
- Pelo código HSL (matiz, saturação, luminosidade)
 - `hsl(m, s, l)`
 - `m`: 0-360; `s`: 0-100%, `l`: 0-100%;



Ajuste de Cor – Exemplos

```
p { color: blue; }  
p { color: #0000FF; }  
p { color: #00F; }  
p { color: rgb(0, 0, 255); }  
p { color: hsl(240, 100%, 50%); }
```

Formas equivalentes de se definir a cor azul para os parágrafos

Unidades de Tamanho da CSS

Unidades de Tamanho Absoluto

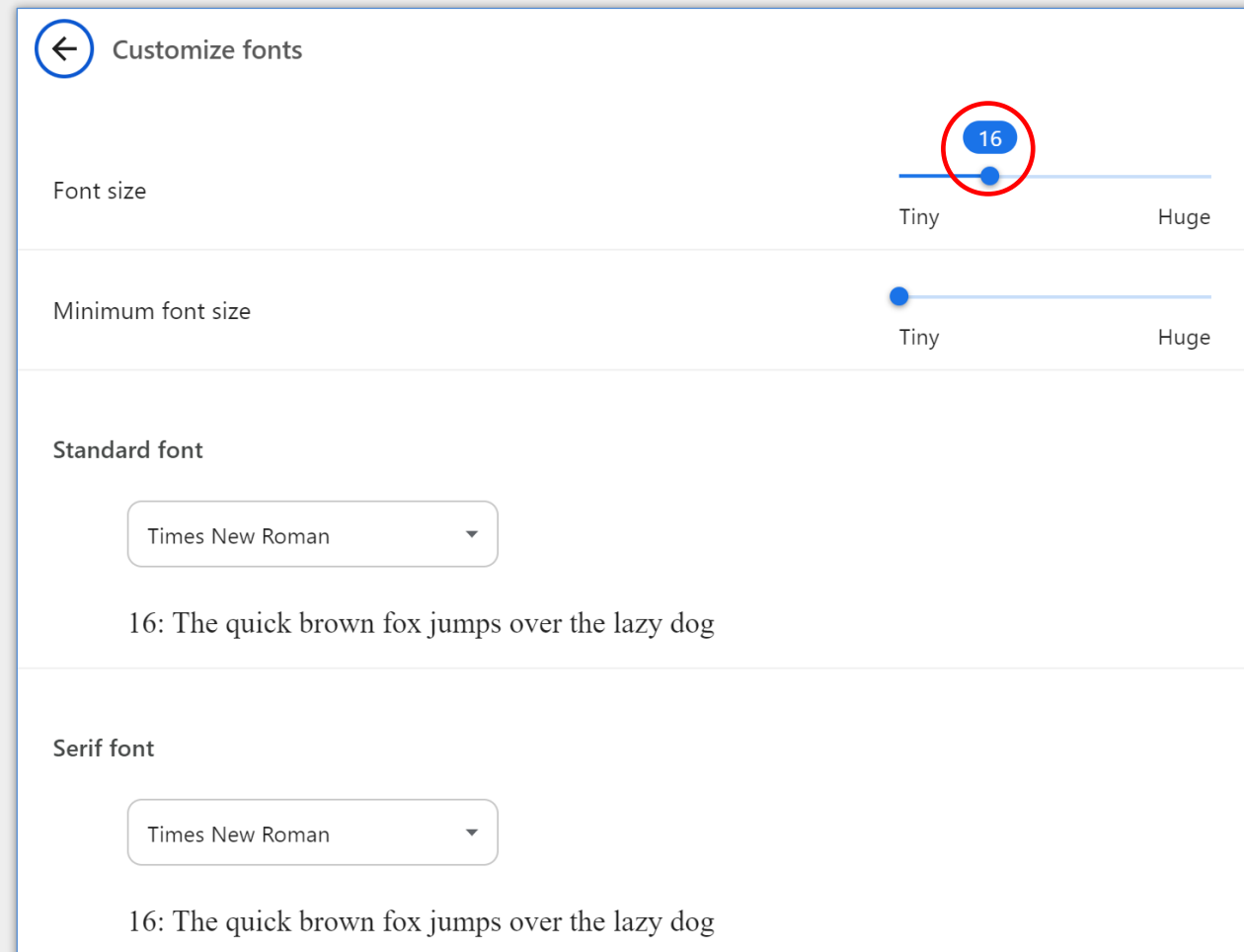
- A unidade **px** (pixels) é a unidade de **tamanho absoluto** mais comum
- Tamanhos absolutos não dependem de tamanhos definidos no elemento pai
- Além disso, ao definir um tamanho utilizando **px**, o tamanho não será afetado por eventual mudança no tamanho de fonte feita pelo usuário nas configurações do navegador (portanto, os tamanhos definidos pelo usuário **serão desprezados**)
- Devem ser utilizados com cautela

Unidades de Tamanho Relativo

- Podem depender de outros tamanhos e configurações como aquelas definidas pelo usuário no navegador, do tamanho definido no elemento pai, do tamanho da viewport (região visível da página no navegador) etc.

Definição do Tamanho Padrão no Navegador

No Google Chrome, acesse *Configurações* → *Aparência* → *Personalizar fontes*



Unidades de Tamanho Relativo mais Comuns

em – relativo ao tamanho da fonte corrente (herdado do elemento pai)

- **2em** = dobro da fonte corrente

rem – relativo ao tamanho da fonte do elemento raiz (<html>)

- **2rem** = dobro do tamanho da fonte do elemento raiz

Exemplo de Tamanho em

```
.medio {  
  font-size: 16px;  
}  
  
.grande {  
  font-size: 2em;  
}
```

```
<main class="medio">  
  Texto 16px  
  <section class="grande">  
    Texto 32px  
    <article class="grande">  
      Texto 64px  
    </article>  
  </section>  
</main>
```


Exemplo de Tamanho rem

```
html {  
  font-size: 14px;  
}  
  
.medio {  
  font-size: 16px;  
}  
  
.grande {  
  font-size: 2rem;  
}
```

```
<main>  
  Texto 14px  
  <section class="medio">  
    Texto 16px  
    <article class="grande">  
      Texto 28px  
      <div class="grande">  
        Texto 28px  
      </div>  
    </article>  
  </section>  
</main>
```

Exemplo de Tamanho rem

Considere que o tamanho de **fonte padrão** definido no **navegador** seja de 18px

```
.medio {  
  font-size: 16px;  
}  
  
.grande {  
  font-size: 2rem;  
}
```

Neste exemplo, o tamanho de fonte padrão não é alterado no elemento raiz (`<html>`). Neste caso, o tamanho padrão é aquele definido nas configurações do navegador (18px).

```
<main>  
  Texto 18px  
  <section class="medio">  
    Texto 16px  
    <article class="grande">  
      Texto 36px  
      <div class="grande">  
        Texto 36px  
      </div>  
    </article>  
  </section>  
</main>
```

Outras Unidades de Tamanho Relativo

% – em geral, relativo ao elemento pai

- `width: 50%` – define a largura em 50% da largura do container

vh – relativo à altura da viewport (**viewport height**)

- `30vh` corresponde a 30% da altura da viewport

vw – relativo à largura da viewport (**viewport width**)

- `100vw` corresponde a 100% da largura da viewport

ch – relativo à largura de um caractere utilizando a fonte do elemento

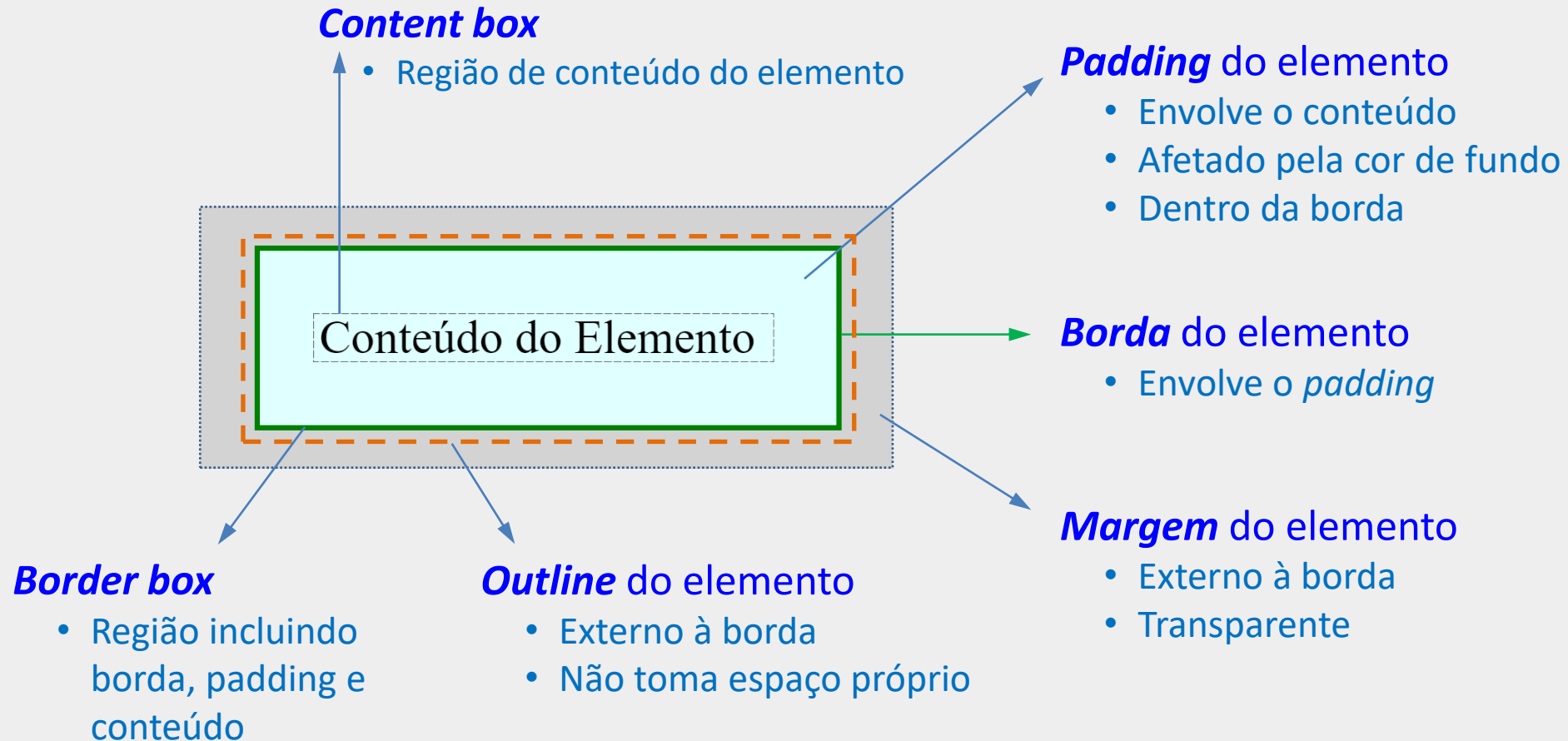
- `width: 10ch` – define a largura para comportar até 10 caracteres

Ajustes de Tamanho, Margem, Padding, Borda e Background

CSS Box Model

- Na HTML, a maioria dos elementos tem a apresentação estruturada em um formato de **caixa** (box), onde há uma região para exibição do **conteúdo**, uma região para **bordas**, outra para **margens** etc.
- Todas essas partes podem ser ajustadas com CSS utilizando um modelo que é conhecido como **CSS Box Model**

CSS Box Model



A CSS disponibiliza as propriedades **padding**, **margin**, **border** e **outline** para ajuste dos tamanhos dessas regiões

Ajustes de Margens e Paddings

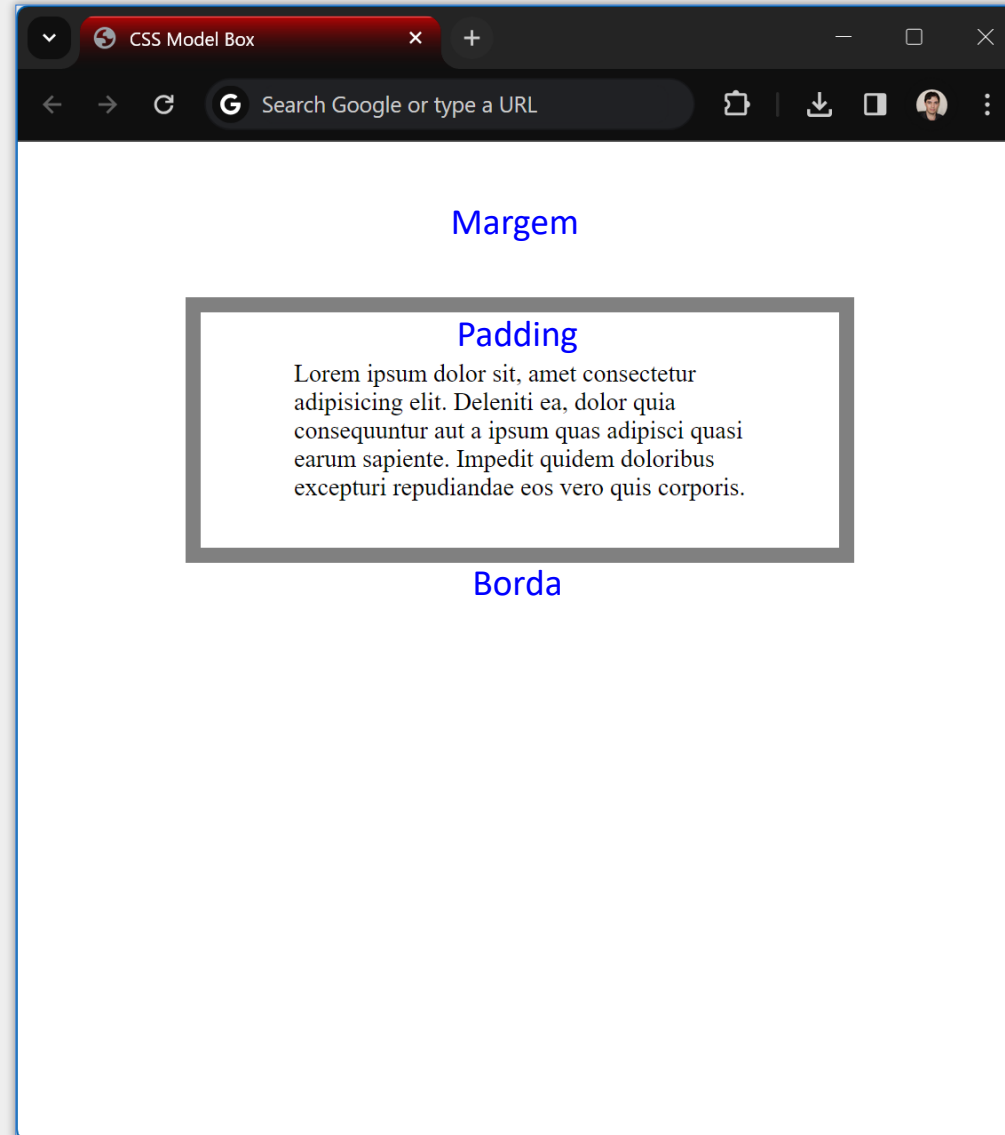
- É possível fornecer até 4 valores para a propriedade **margin**:
 - `margin: 20px;` (ajusta todas as margens em 20px)
 - `margin: 20px 50px 80px 100px;` (superior, direita, inferior e esquerda)
 - `margin: 20px 40px 20px;` (superior, laterais, inferior)
 - `margin: 20px 40px;` (superior/inferior e laterais)
- A mesma sintaxe se aplica à propriedade **padding**
- Há também propriedades para ajuste individual:
 - `margin-left`, `margin-right`, `margin-top` e `margin-bottom`
 - `padding-left`, `padding-right`, `padding-top` e `padding-bottom`

Ajustes de Borda

- A propriedade `border` permite definir a borda por completo (4 lados)
- Sintaxe mais comum:
 - `border: espessura estilo cor`
 - Estilos possíveis: `solid`, `dotted`, `dashed`, `double` ou `none`
 - A ordem dos valores não importa
- Exemplos:
 - `border: 1px solid blue;`
 - `border: 2px double red;`
 - `border: none;`
- Há também propriedades para ajuste de cada borda individualmente
 - `border-top`, `border-bottom`, `border-left`, `border-right`
- E propriedades para ajustar apenas um aspecto específico das bordas
 - `border-color`, `border-style`, `border-width`

CSS Box Model – Margens, Paddings e Bordas


```
<style>
  html {
    margin: 0;
    padding: 0;
  }
  .box {
    margin: 100px;
    padding: 30px 60px;
    border: 10px solid gray;
  }
</style>
</head>
<body>
  <div class="box">
    Lorem ipsum dolor sit, amet
    adipisci quasi earum sapiente
  </div>
</body>
```




Ajustes de Borda – Exemplos Adicionais

```
<p id="p1"> Programação para Internet </p>
<p id="p2"> Programação para Internet </p>
<p id="p3"> Programação para Internet </p>
<p id="p4"> Programação para Internet </p>
```


```
#p1 {
  border: 1px solid ■blue;
}
#p2 {
  border-top: 1px solid ■blue;
}
#p3 {
  border-color: ■red ■green ■blue ■black;
  border-style: dashed dotted double solid;
  border-width: 4px 6px 8px 10px;
}
#p4 {
  border-color: ■gray;
  border-style: solid none;
  border-width: 2px;
}
```




Programação para Internet



Programação para Internet



Programação para Internet



Programação para Internet

CSS Box Model – Visualização no Navegador

- Durante o desenvolvimento do layout da página, pode ser necessário verificar os tamanhos da região de conteúdo, margens, paddings e bordas
- Esses tamanhos podem ser conferidos no próprio navegador
- No Google Chrome, basta clicar sobre o elemento com o botão direito do mouse e escolher **Inspecionar**
- Em seguida, pode ser necessário rolar a interface para visualizar a **caixa modelo** com os tamanhos em pixels (veja próximo slide)

CSS Box Model – Visualização no Navegador

Com a página aberta no navegador, clique com o botão direito do mouse sobre o elemento que deseja conferir as margens, bordas etc. e escolha a opção **Inspecionar**. Procure pela **caixa modelo**, conforme figura a o lado.

```
<!DOCTYPE html>
<html>
  <head> ... </head>
  <body>
    <div class="box"> ... </div> == $0
  </body>
</html>
```

margin	100
border	9.974
padding	50
content	375.136x120

Caixa Modelo

CSS Box Model – Visualização no Navegador

Loem ipsum dolor sit, amet consectetur adipisicing elit. Deleniti ea, dolor quia consequuntur aut a ipsum quas adipisci quasi earum sapiente. Impedit quidem doloribus excepturi repudiandae eos vero quis corporis.

```
<!DOCTYPE html>
<html>
  <head> ... </head>
  <body>
    <div class="box"> ... </div> == $0
  </body>
</html>
```

margin 100
border 9.974
padding 50
100 9.974 50 375.136x120 50 9.974 100
50
9.974
100

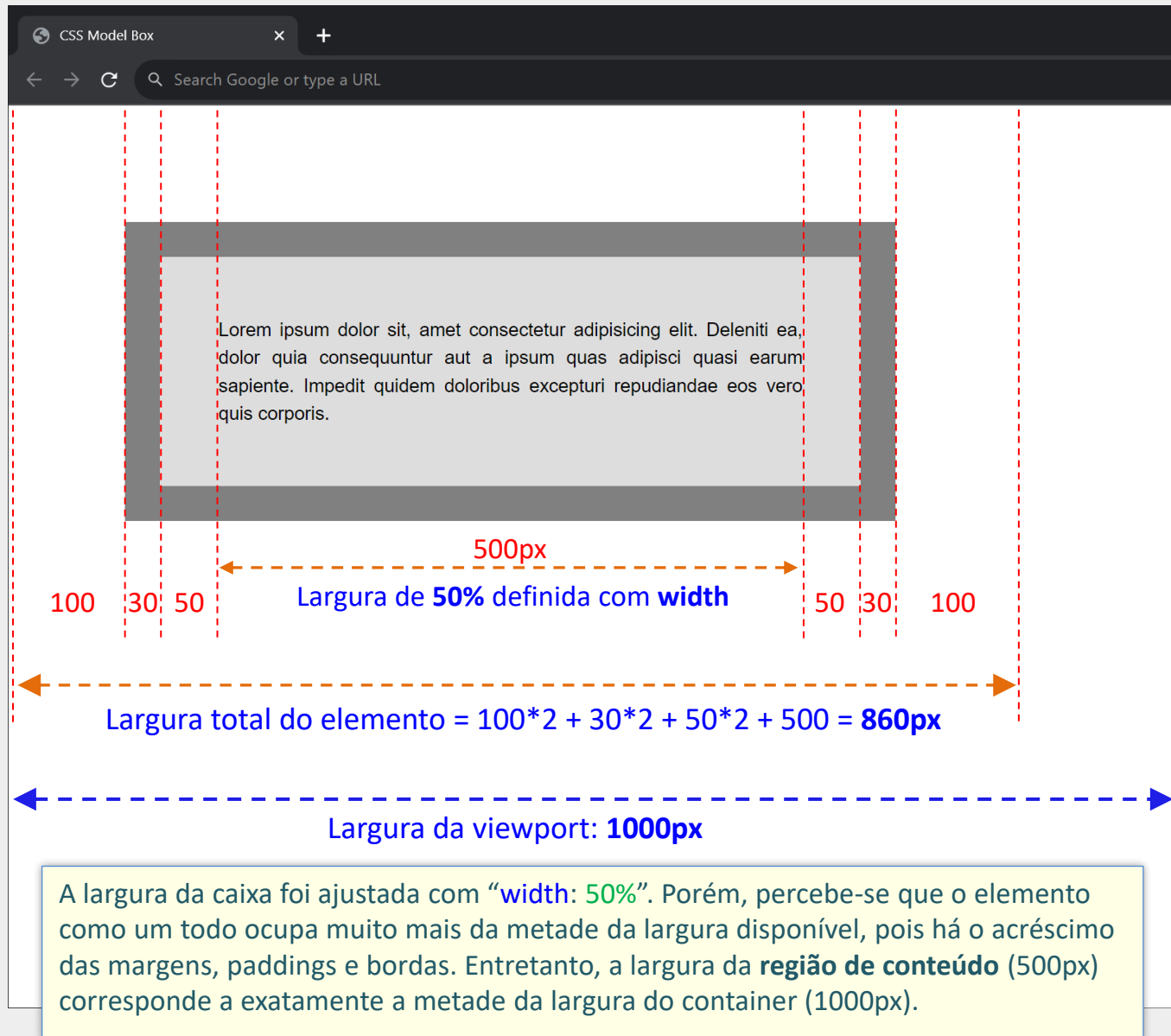
Caixa Modelo

Ao passar o ponteiro do mouse em alguma região da caixa modelo (como na margem, neste exemplo), o navegador destaca na página o respectivo espaço ocupado pelo item.

Ajustes de Largura e Altura

- Para definir a largura e a altura de um elemento pode-se utilizar as propriedades `width` e `height`
- Na maioria dos casos, `width` e `height` definem tamanhos para a **região de conteúdo** do elemento (abordagem padrão)
 - Largura total = larg. conteúdo + margens + bordas + paddings
 - Esse comportamento pode ser alterado com a propriedade `box-sizing`
- `height` não altera a **altura** de alguns elementos de linha (``, `<a>` etc)

Largura Total do Elemento – Cálculo Padrão



```
.box {  
  width: 50%;  
  text-align: justify;  
  margin: 100px;  
  padding: 50px;  
  border: 30px solid gray;  
  background-color: #ddd;  
}
```

margin 100
border 29.922
padding 50
500.260x96
padding 50
border 29.922
margin 100

A largura da caixa foi ajustada com “width: 50%”. Porém, percebe-se que o elemento como um todo ocupa muito mais da metade da largura disponível, pois há o acréscimo das margens, paddings e bordas. Entretanto, a largura da **região de conteúdo** (500px) corresponde a exatamente a metade da largura do container (1000px).

Propriedade box-sizing

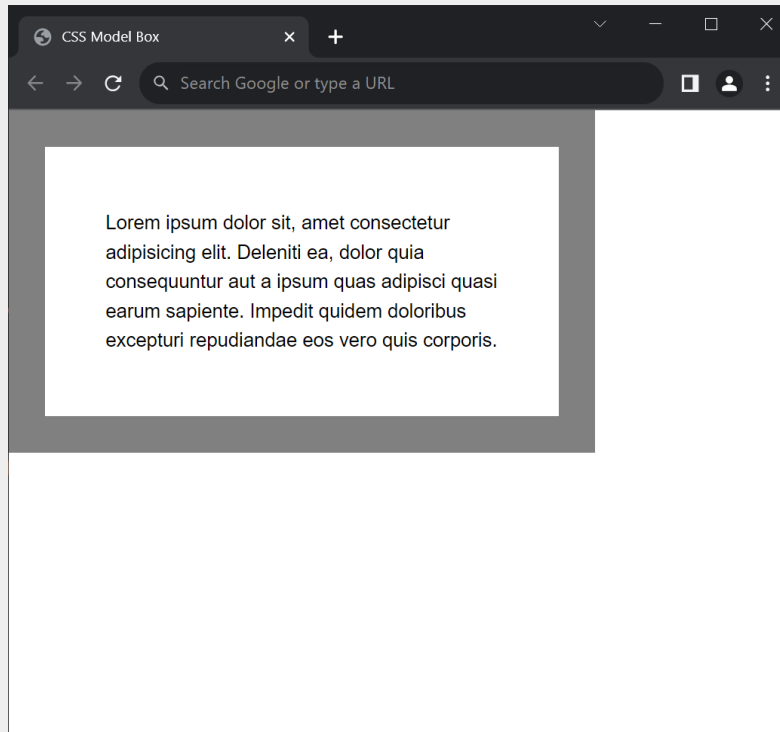
- Altera o modo em que a largura e altura do elemento é calculada
- **box-sizing: content-box**
 - Valor padrão para a maioria dos elementos
 - Ao definir **width**, estamos definindo a largura da **região de conteúdo** (content box)
 - A largura total do elemento será: **width** + bordas + paddings + margens
- **box-sizing: border-box**
 - O cálculo da largura/altura inclui os tamanhos das bordas e paddings (**mas não as margens**)
 - Ao definir a largura com **width**, estamos definindo, na verdade, a largura da caixa de conteúdo **em conjunto** com a borda e o padding (largura da **border box**)
 - A largura total será: **width** + margens

box-sizing: content-box vs box-sizing: border-box

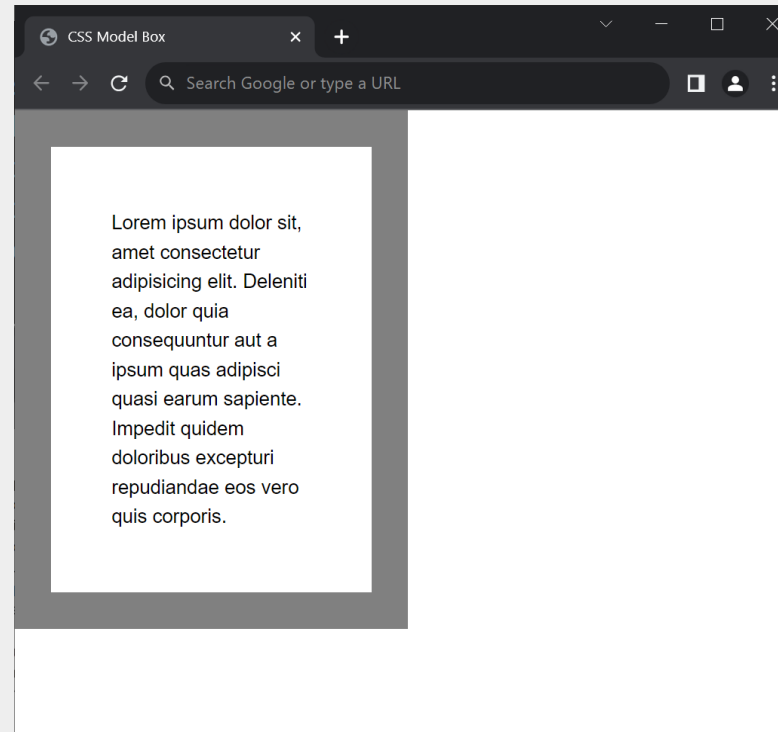
Com **box-sizing: content-box**, “width: 50%” faz com que a região de conteúdo (texto) ocupe 50% da largura do container, mas a largura total do elemento (incluindo paddings, bordas e margens) ultrapassa a metade do container.

Este é o mecanismo padrão utilizado pela maioria dos elementos.

```
.box {  
  box-sizing: content-box;  
  width: 50%;  
  margin: 0px;  
  padding: 50px;  
  border: 30px solid gray;  
}
```



```
.box {  
  box-sizing: border-box;  
  width: 50%;  
  margin: 0px;  
  padding: 50px;  
  border: 30px solid gray;  
}
```



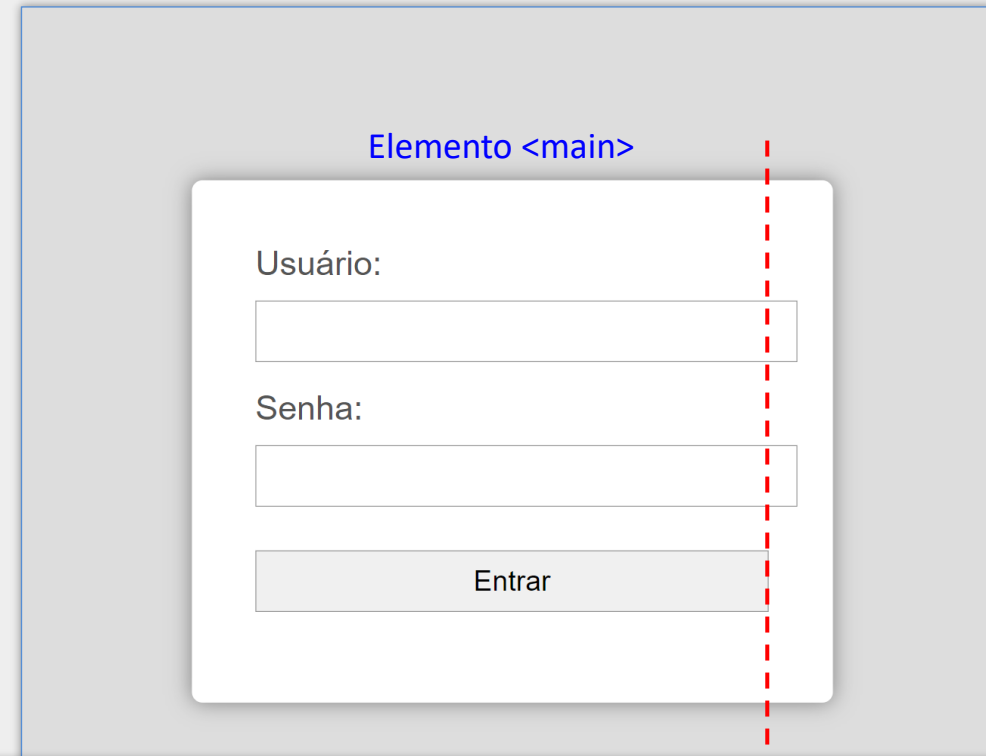
Alterando a propr. **box-sizing** para o valor **border-box**, “width: 50%” faz com que toda a caixa da borda ocupe 50% da largura do container

Elementos com `box-sizing` padrão diferentes

- A maioria dos elementos utiliza, por padrão, `box-sizing: content-box`
- Porém há elementos que utilizam, por padrão, `box-sizing: border-box`, como é o caso do elemento `<button>`
- Utilizar elementos com `box-sizing` diferentes no layout pode resultar em inconsistências visuais (veja exemplo no próximo slide)

Elementos com box-sizing padrão diferentes

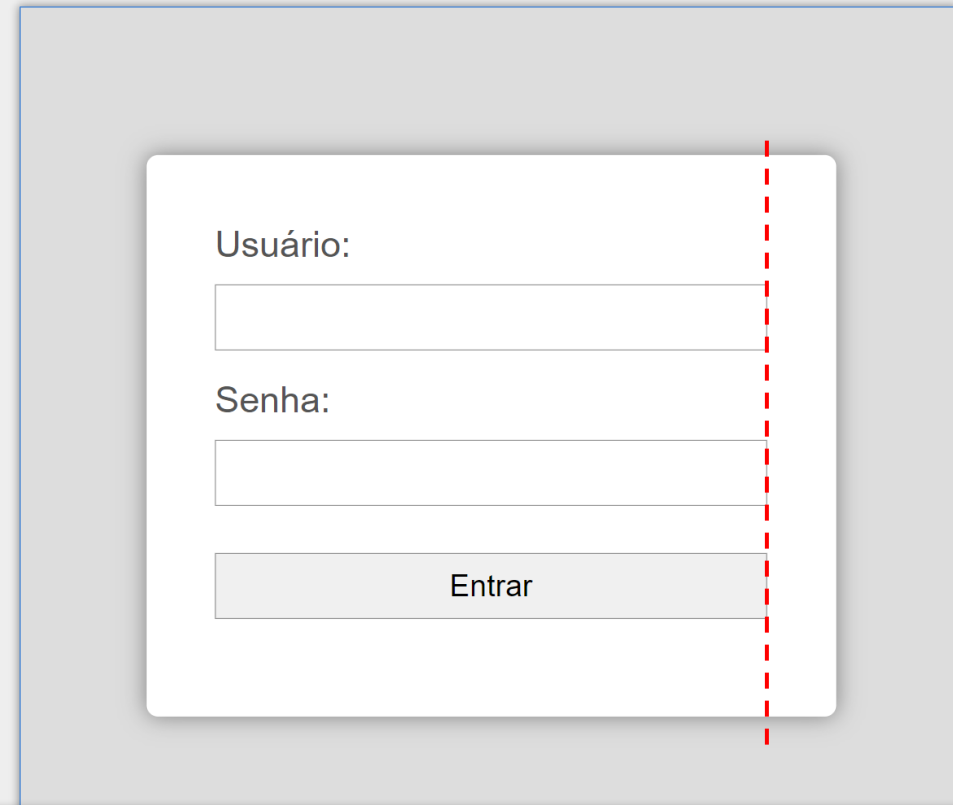
```
main {  
  width: 300px;  
  padding: 30px;  
  background-color: white;  
}  
  
input,  
button {  
  width: 100%;  
  padding: 0.4rem;  
  margin: 0.5rem 0;  
  outline: none;  
  border: 0.5px solid gray;  
}
```



Neste exemplo, os campos textuais e o botão foram definidos para ocupar 100% da largura do container (região branca). Entretanto não temos o efeito desejado, pois os campos textuais ultrapassam o alinhamento do botão (linha pontilhada). Isso acontece porque os elementos `<input>` utilizam, por padrão, `box-sizing: content-box`, enquanto o elemento `<button>` utiliza `box-sizing: border-box`.

Propriedade box-sizing - Exemplo

```
main {  
  width: 300px;  
  padding: 30px;  
  background-color: □white;  
}  
input,  
button {  
  width: 100%;  
  padding: 0.4rem;  
  margin: 0.5rem 0;  
  outline: none;  
  border: 0.5px solid ■gray;  
  box-sizing: border-box;  
}
```



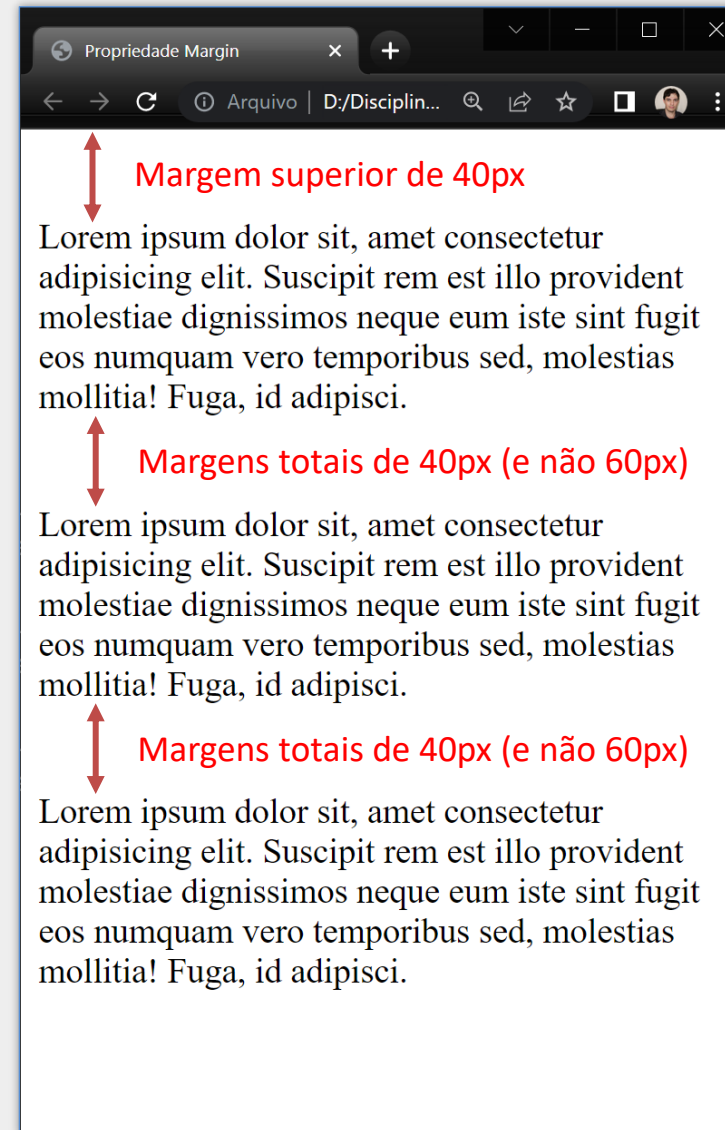
Uma solução simples é definir `box-sizing: border-box` para os elementos `<input>`, possibilitando que tanto os campos quanto o botão utilizem o mesmo mecanismo de cálculo de tamanho.

Sobreposição de Margens

- Margens entre dois elementos de bloco vizinhos **não se somam**
- Em vez disso, prevalece a **maior** delas
- Esse efeito é conhecido como **sobreposição de margens** (margin collapsing)

Sobreposição de Margens – Exemplo

```
<style>
  p {
    margin-top: 40px;
    margin-bottom: 20px;
  }
</style>
</head>
<body>
  <p>Lorem ipsum dolor sit, amet c
  molestiae dignissimos neque eu
  mollitia! Fuga, id adipisci.</p>
  <p>Lorem ipsum dolor sit, amet c
  molestiae dignissimos neque eu
  mollitia! Fuga, id adipisci.</p>
  <p>Lorem ipsum dolor sit, amet c
  molestiae dignissimos neque eu
  mollitia! Fuga, id adipisci.</p>
  <!-- vertical margin collapsing
</body>
```



Outros Ajustes de Tamanho

- Além das propriedades `width` e `height`, há também propriedades para definir a largura mínima, a largura máxima, a altura mínima e a altura máxima de um elemento:
 - `min-width`, `max-width`,
 - `min-height`, `max-height`
- Tais propriedades são especialmente importantes em layouts responsivos
- Outros valores possíveis para propriedades de ajuste de tamanho incluem:
 - `max-content`, `min-content` e `fit-content`

Borda e Background Arredondados

- `border-radius` permite arredondar os cantos da **borda** e do **background** dos elementos
- O valor da propriedade indica o raio da curvatura

```
#p1 {  
  border-radius: 10px;  
}  
  
#p2 {  
  border-radius: 10px 20px 40px 60px  
}  
  
#p3 {  
  border-radius: 10px 40px;  
  border: none;  
}
```

Programação para Internet (P1)

Programação para Internet (P2)

Programação para Internet (P3)

Caixa de Sombreamento com box-shadow

`box-shadow: 20px 10px;`



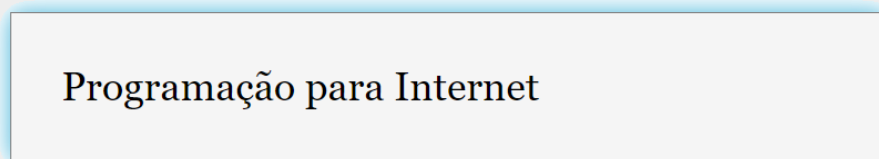
`box-shadow: 5px 5px gray;`



`box-shadow: 5px 5px 5px gray;`



`box-shadow: 0 0 10px cyan;`



OBS: O `box-shadow` não ocupa espaço próprio (como acontece com a borda) e não interfere no layout.

Ajustes de Fundo – Background

background-color

- Permite ajustar a cor de fundo do elemento (conteúdo e paddings)
- Ex.: `div { background-color: gray; }`

background-image

- Permite inserir uma imagem de fundo para o elemento
- Ex.: `body { background-image: url("images/bgImage.png"); }`

background-repeat

- Define o modo de repetição da imagem de fundo (caso ela seja menor)
- Valores: `no-repeat`, `repeat-x` (rep. hor.), `repeat-y` (rep. vert.), `repeat` (rep. hor. e vert.)
- Valor padrão: `repeat`

background-size

- Permite ajustar o tamanho de exibição da imagem de fundo
- Ex: `background-size: cover` - estica a imagem para ocupar todo o fundo

background

- Propriedade abreviada que permite definir todos os aspectos de uma vez (exercício)

Ajustes de Fundo – Exemplo

```
<style>
  body {
    background-image: url("images/bg.jpg");
    background-repeat: no-repeat;
    background-size: cover;
  }
</style>
```

Testando imagem de fundo

Lorem ipsum dolor sit amet consectetur adipisicing elit. Libero numquam at atque ipsam voluptas magni in nihil laborum est, magnam quia repellat quisquam recusandae dolor nisi possimus quas? Exercitationem, assumenda!



Exibição e Posicionamento

Propriedade display

- Altera o modo de apresentação do elemento
- Um elemento de linha pode ser exibido como bloco e vice-versa
- Permite ocultar um elemento, removendo do layout
- Alguns valores possíveis:
 - none, block, inline, inline-block
 - flex, grid, inline-flex, inline-grid

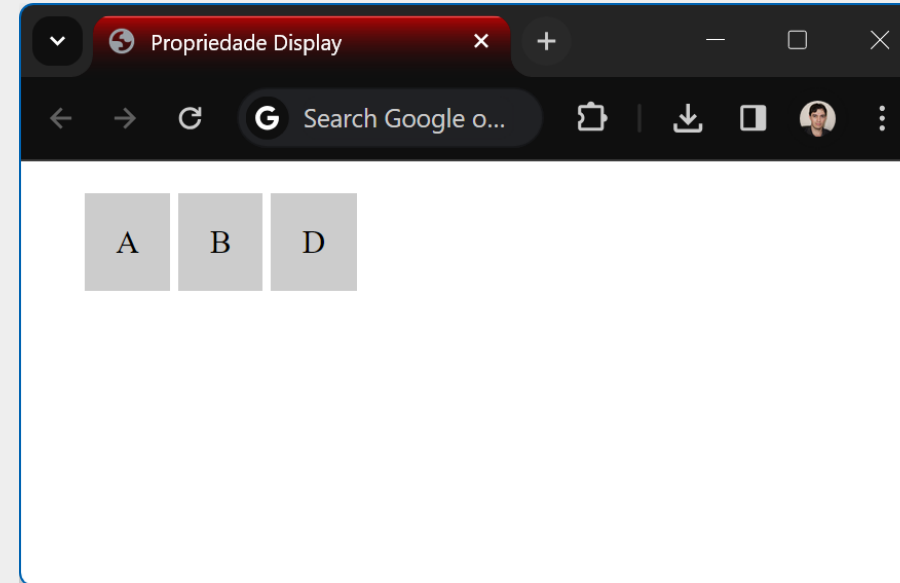
Propriedade display

`display: none`

- oculta completamente o elemento, **removendo o mesmo do layout**
- libera o espaço para outros elementos próximos

Ocultando elemento do layout com `display: none`

```
<style>
  span {
    background-color: #CCC;
    padding: 1rem;
  }
  .oculto {
    display: none;
  }
</style>
</head>
<body>
  <span>A</span>
  <span>B</span>
  <span class="oculto">C</span>
  <span>D</span>
</body>
```



Observe que o item C foi removido do layout, dando lugar ao item D.

Propriedade display

display: block

- faz com que o elemento tenha exibição em nível de bloco (como o `<div>`)
- elementos de bloco são exibidos em nova linha e ocupam toda a largura disponível

Exibindo campos em nova linha com `display: block`

```
<form action="cadastra.php" method="post">
  <div>
    <label for="produto">Produto:</label>
    <input type="text" id="produto" name="prodNome">
  </div>
  <div>
    <label for="descricao">Descrição:</label>
    <input type="text" id="descricao" name="prodDesc">
  </div>
  <div>
    <label for="marca">Marca:</label>
    <input type="text" id="marca" name="prodMarca">
  </div>
  <button>Enviar</button>
</form>
```

```
input {
  display: block;
  margin-bottom: 1rem;
}
```

Produto:

Descrição:

Marca:

Os campos `input` tem exibição padrão em nível de linha, e aparecem normalmente na frente dos rótulos. Porém, neste exemplo eles são exibidos embaixo do rótulos, pois sua exibição foi alterada com `display: block`. Repare que não foi necessário inserir nenhum `
`.

Propriedade display

display: inline

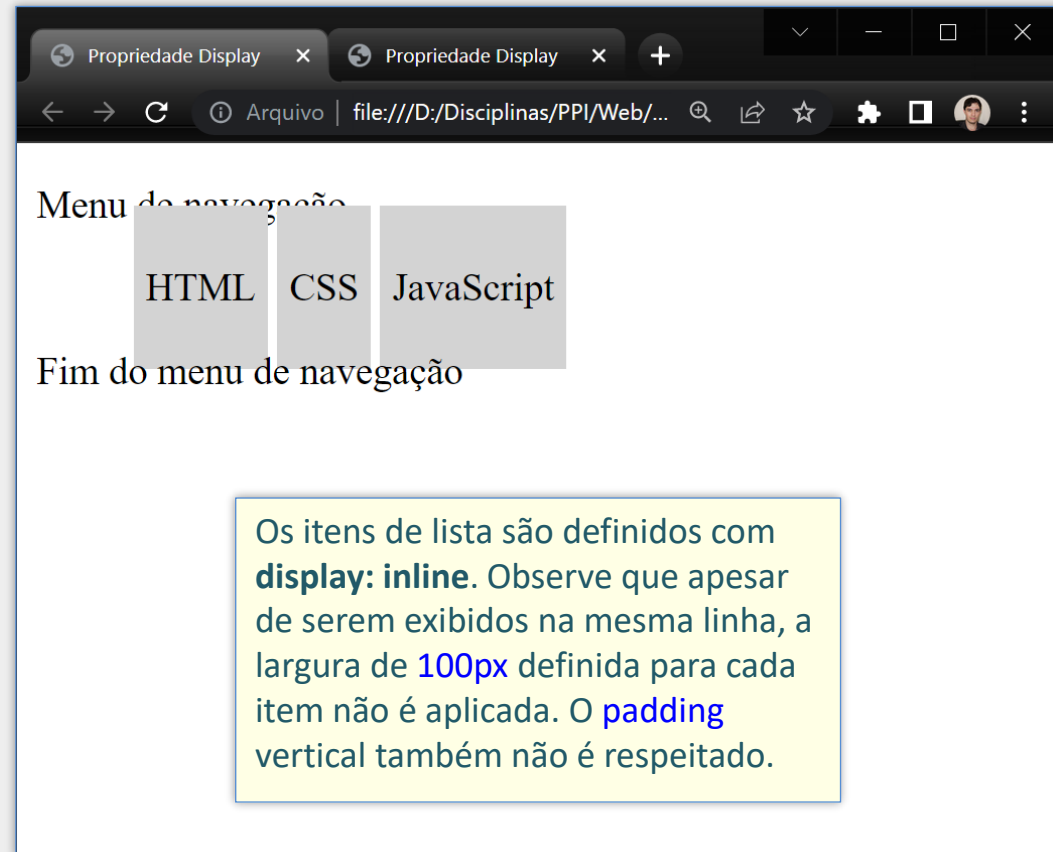
- faz com que o elemento seja exibido em nível de linha (como o ``)
- elementos de linha não começam obrigatoriamente com quebra de linha e só ocupam o espaço necessário para exibição de seu conteúdo
- `width` e `height` não terão efeito em alguns elementos
- margens e paddings verticais de alguns elementos **não são respeitados**

display: inline-block

- exibição em nível de linha, mas com a possibilidade de usar `width` e `height`
- margens e paddings superiores e inferiores **respeitados**

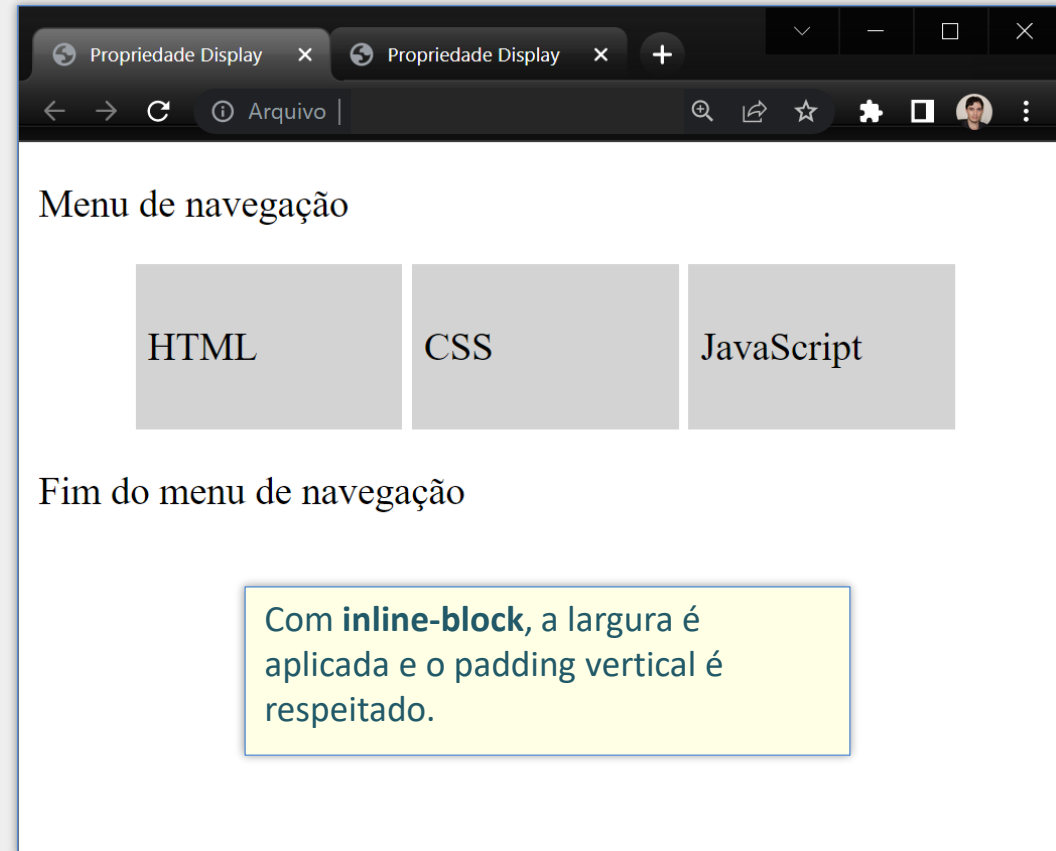
Item de Lista com display: inline

```
<style>
  nav li {
    background-color: lightgray;
    padding: 25px 5px;
    width: 100px;
    display: inline;
  }
</style>
</head>
<body>
  <nav>
    <p>Menu de navegação</p>
    <ul>
      <li>HTML</li>
      <li>CSS</li>
      <li>JavaScript</li>
    </ul>
    <p>Fim do menu de navegação</p>
  </nav>
</body>
```



Item de Lista com display: inline-block

```
<style>
  nav li {
    background-color: lightgray;
    padding: 25px 5px;
    width: 100px;
    display: inline-block;
  }
</style>
</head>
<body>
  <nav>
    <p>Menu de navegação</p>
    <ul>
      <li>HTML</li>
      <li>CSS</li>
      <li>JavaScript</li>
    </ul>
    <p>Fim do menu de navegação</p>
  </nav>
</body>
```

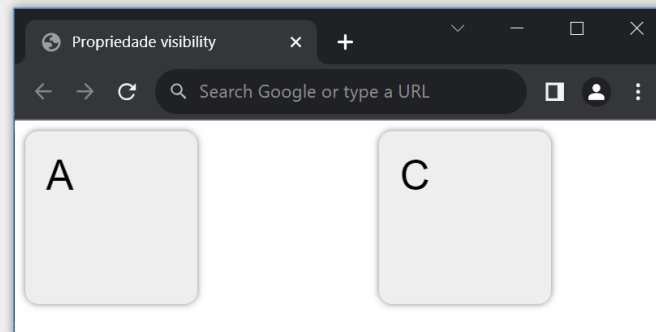


Propriedade visibility

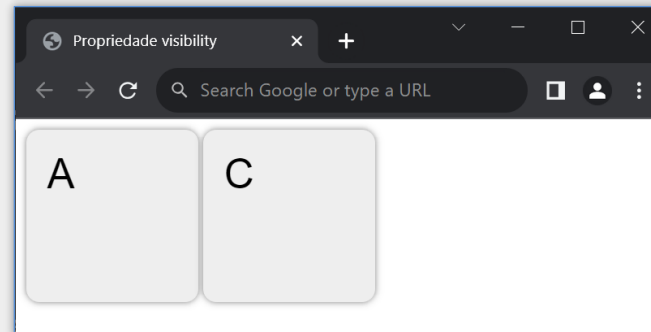
- Mostra ou oculta um elemento **sem alterar o layout**
- **visibility: visible**
 - elemento visível, aparecendo normalmente
- **visibility: hidden**
 - o elemento será ocultado, mas **continuará ocupando espaço no layout**
- **visibility: collapse**
 - permite ocultar linhas e colunas de tabelas sem alterar o layout geral
 - as demais linhas e colunas permanecem com seus tamanhos originais

visibility: hidden vs display: none

```
<style>
  div {
    display: inline-block;
    width: 100px;
    height: 100px;
    background-color: #eee;
  }
  .oculto {
    visibility: hidden;
  }
</style>
</head>
<body>
  <div>A</div>
  <div class="oculto">B</div>
  <div>C</div>
</body>
```



```
<style>
  div {
    display: inline-block;
    width: 100px;
    height: 100px;
    background-color: #eee;
  }
  .oculto {
    display: none;
  }
</style>
</head>
<body>
  <div>A</div>
  <div class="oculto">B</div>
  <div>C</div>
</body>
```



visibility: collapse

Aluno	Prova 1	Prova 2
Fulano	7,0	8,0
Ciclano	6,0	9,0
Media da Turma	6,5	8,5

Tabela original com todas as linhas e colunas sendo exibidas

Aluno	Prova 1	Prova 2
Fulano	7,0	8,0
Ciclano	6,0	9,0

Última linha ocultada com `visibility: collapse`
Repare que as larguras das colunas permanecem inalteradas.

Aluno	Prova 1	Prova 2
Fulano	7,0	8,0
Ciclano	6,0	9,0

Última linha ocultada com `display: none`
Repare que as larguras das colunas são reajustadas conforme conteúdo (col. **Aluno**)

Propriedade float

- Posiciona o elemento no lado esquerdo ou direito de seu container
- Permite que texto e outros elementos se posicionem à sua volta
- Alguns valores
 - `left` elemento "flutua" no lado esquerdo do container
 - `right` elemento "flutua" no lado direito do container
 - `none` elemento não "flutua"

Propriedade float - Exemplo

```
<style>
  .esquerdo {
    background-color: #ddd;
    width: 15%;
    height: 300px;
    margin: 5px;
    float: left;
  }
  .direito {
    background-color: #ddd;
    text-align: center;
    width: 25%;
    height: 300px;
    float: right;
  }
</style>
</head>
<body>
  <aside class="esquerdo">Esquerdo</aside>
  <aside class="esquerdo">Esquerdo</aside>
  <aside class="direito">Direito</aside>
  <main>
    <p>Lorem ipsum dolor sit amet consectetur
    | omnis eius iusto.</p>
  </main>
```

Esquerdo

Esquerdo

Lorem ipsum dolor
sit amet consectetur,
adipiscing elit. Illum
deserunt natus earum
facilis quas ipsa
numquam id omnis
eius iusto.

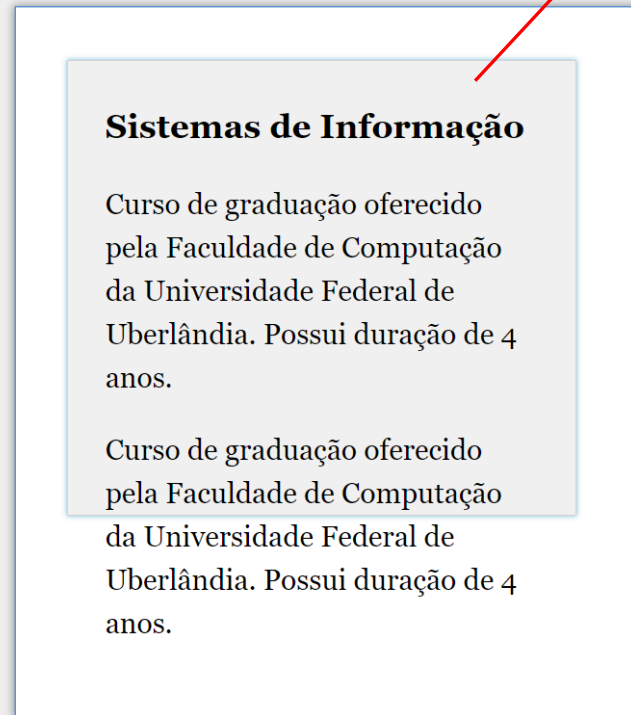
Direito

Propriedade overflow

- Define como o conteúdo do elemento deve ser exibido ao extrapolar a borda
- Propriedade abreviada de `overflow-x` e `overflow-y`
- Alguns valores
 - `visible` conteúdo sempre visível, ainda que fora dos limites (default)
 - `hidden` conteúdo cortado, se necessário, para caber no espaço
 - `scroll` barras de rolagens são sempre apresentadas
 - `auto` barras de rolagens apresentadas apenas quando necessário

Propriedade overflow - Exemplo

Painéis com tamanho fixo (dimensões definidas com `width` e `height`)



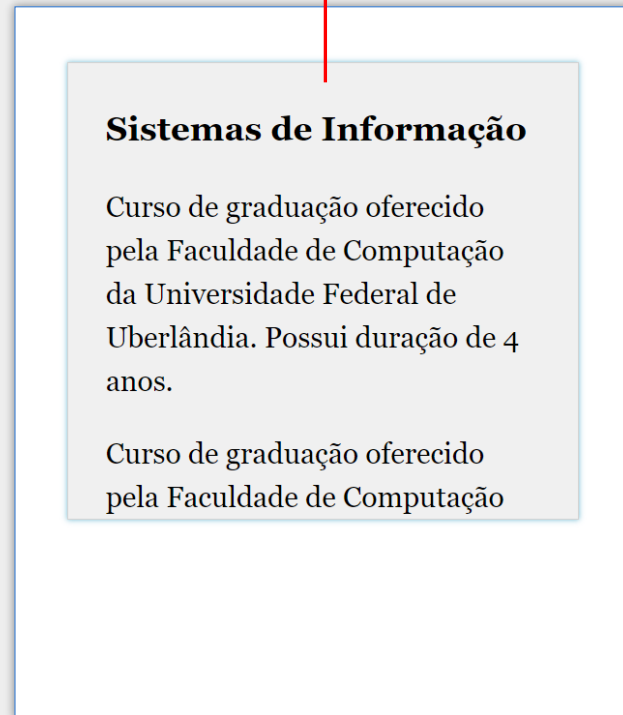
Sistemas de Informação

Curso de graduação oferecido pela Faculdade de Computação da Universidade Federal de Uberlândia. Possui duração de 4 anos.

Curso de graduação oferecido pela Faculdade de Computação da Universidade Federal de Uberlândia. Possui duração de 4 anos.

This panel illustrates the `overflow: visible` property. It contains two paragraphs of text. The second paragraph is partially cut off at the bottom of the panel, and the text is visible outside the panel's boundaries.

`overflow: visible;`



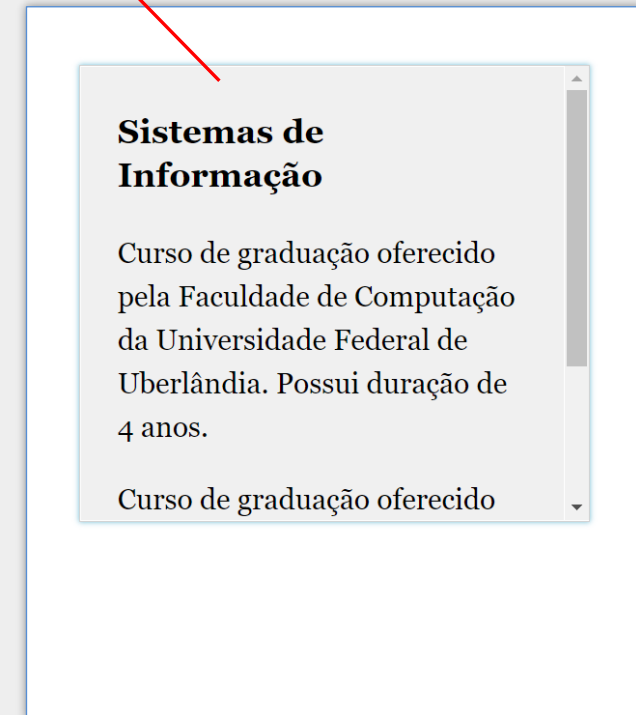
Sistemas de Informação

Curso de graduação oferecido pela Faculdade de Computação da Universidade Federal de Uberlândia. Possui duração de 4 anos.

Curso de graduação oferecido pela Faculdade de Computação

This panel illustrates the `overflow: hidden` property. It contains two paragraphs of text. The second paragraph is partially cut off at the bottom of the panel, and the text is not visible outside the panel's boundaries.

`overflow: hidden;`



Sistemas de Informação

Curso de graduação oferecido pela Faculdade de Computação da Universidade Federal de Uberlândia. Possui duração de 4 anos.

Curso de graduação oferecido

This panel illustrates the `overflow: auto` property. It contains two paragraphs of text. The second paragraph is partially cut off at the bottom of the panel, and the text is not visible outside the panel's boundaries. A vertical scrollbar is visible on the right side of the panel, indicating that the content can be scrolled.

`overflow: auto;`

Propriedade position

- Define como o elemento é **posicionado** na página
- Normalmente é utilizada em conjunto com **top**, **left**, **right** e **bottom**
- Valores possíveis
 - **static**
 - **relative**
 - **absolute**
 - **fixed**
 - **sticky**
- O elemento é dito **posicionado** quando **position** tem valor diferente de **static**

Propriedade position

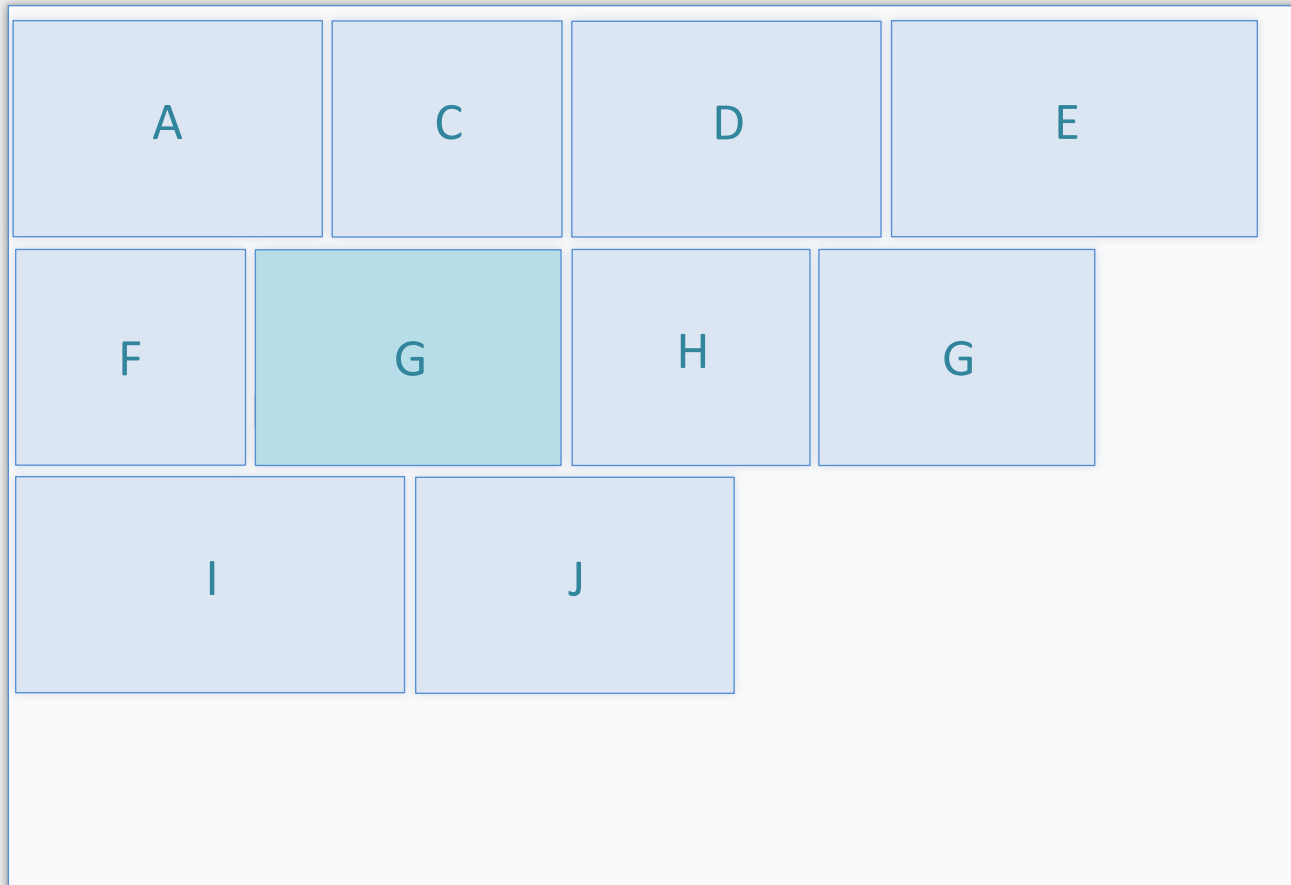
position: static

- Valor padrão
- Elemento posicionado de acordo com fluxo normal do documento

position: relative

- Permite posicionar o elemento de maneira relativa à sua posição normal
- O elemento é primeiramente posicionado de acordo com o fluxo normal. Em seguida, é deslocado da sua posição com **top**, **left**, **right** e **bottom**
- O deslocamento **não afeta a posição** dos elementos à volta

Exemplo de position: static

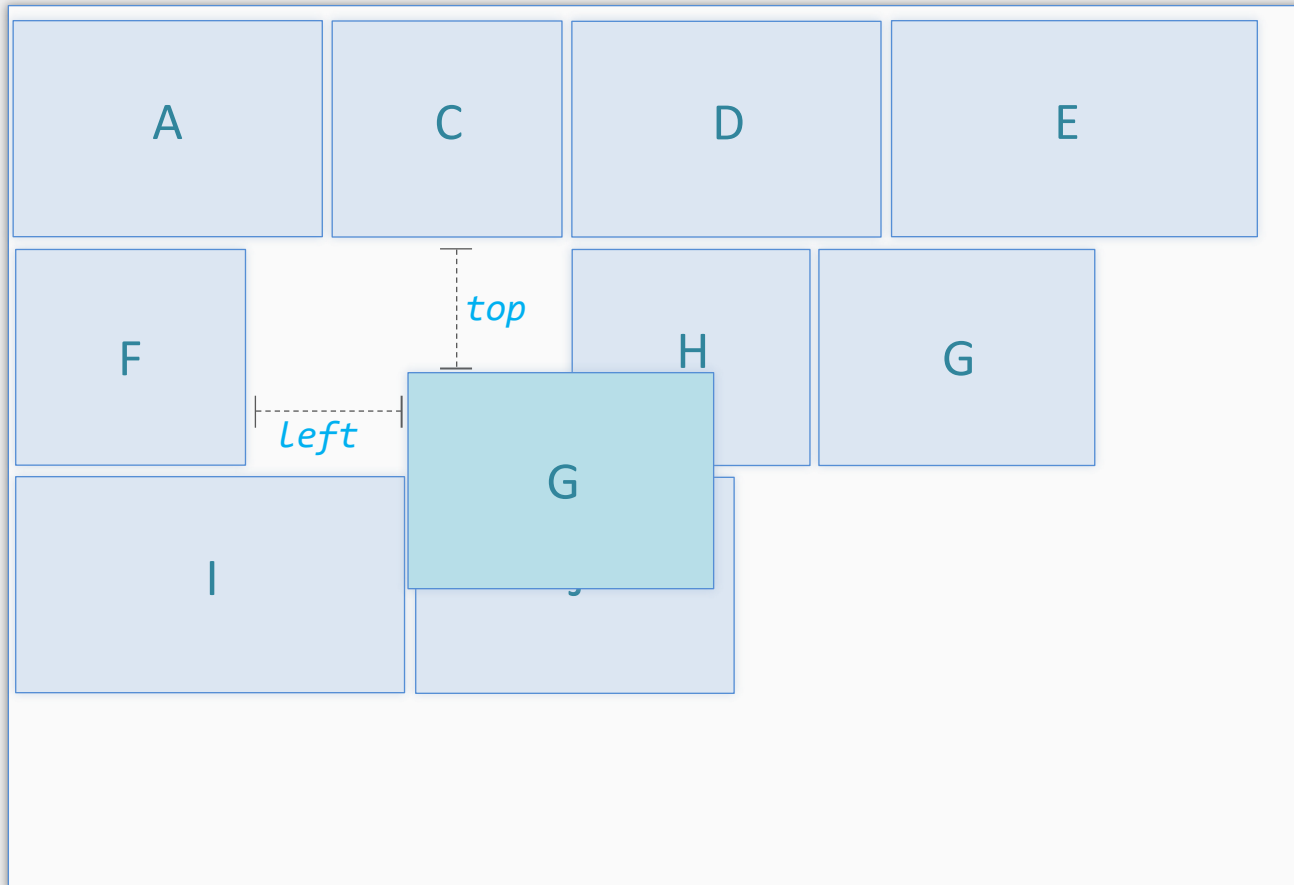


```
.box {  
  display: inlineblock;  
  background-color: skyblue;  
}
```

```
#box_G {  
  position: static;  
  background-color: aqua;  
}
```

O posicionamento estático é o posicionamento padrão do elemento (sua posição segue o fluxo normal do layout).

Exemplo de position: relative



```
#box_G {  
  position: relative;  
  left: 50px;  
  top: 50px;  
  background-color: aqua;  
}
```

Propriedade `position` - continuação

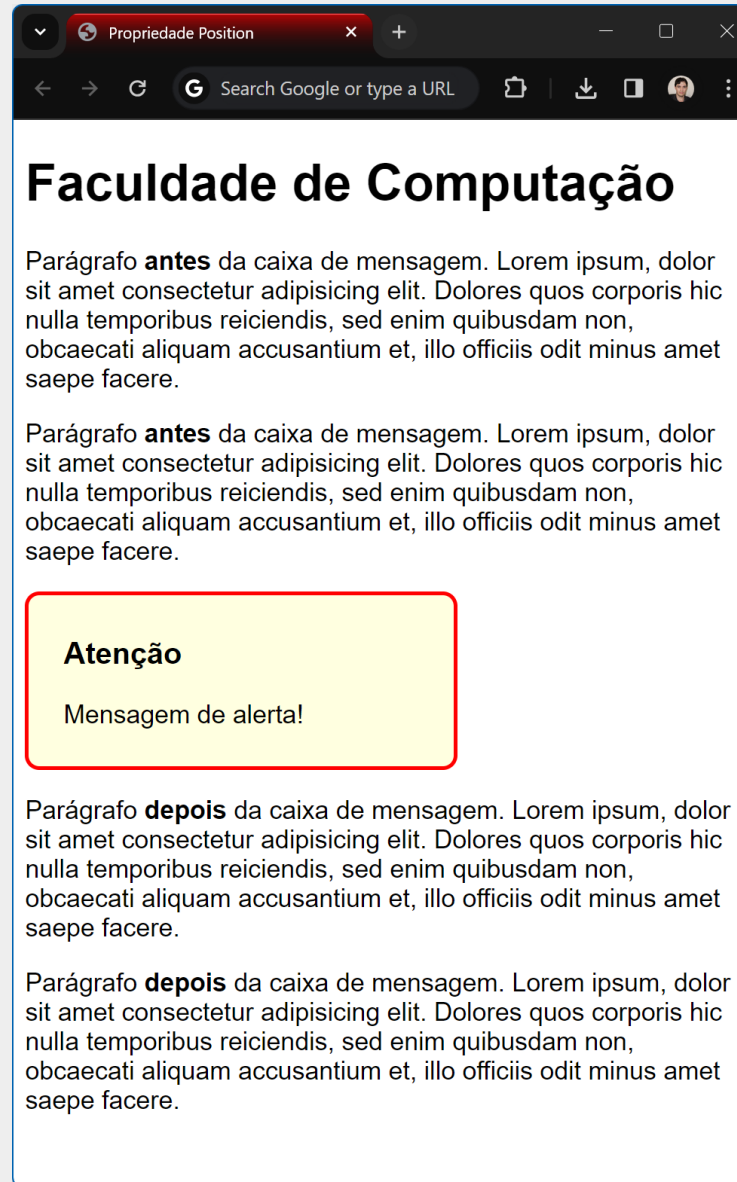
`position: absolute`

- O elemento é **removido** do fluxo normal
- Portanto, **não ocupa espaço no layout**
- Deve ser posicionado com `top`, `left`, `right` e `bottom`
- O posicionamento é relativo **ao ancestral mais próximo posicionado***, se houver
 - Caso contrário, o posicionamento é **relativo ao elemento raiz** (`<html>`)
- Pode ser utilizado para centralizar um elemento de bloco

*O ancestral mais próximo posicionado é o primeiro elemento acima na hierarquia (pai, avô etc.) **que tem** a propriedade `position` com valor diferente de `static`.

Exemplo de position: static

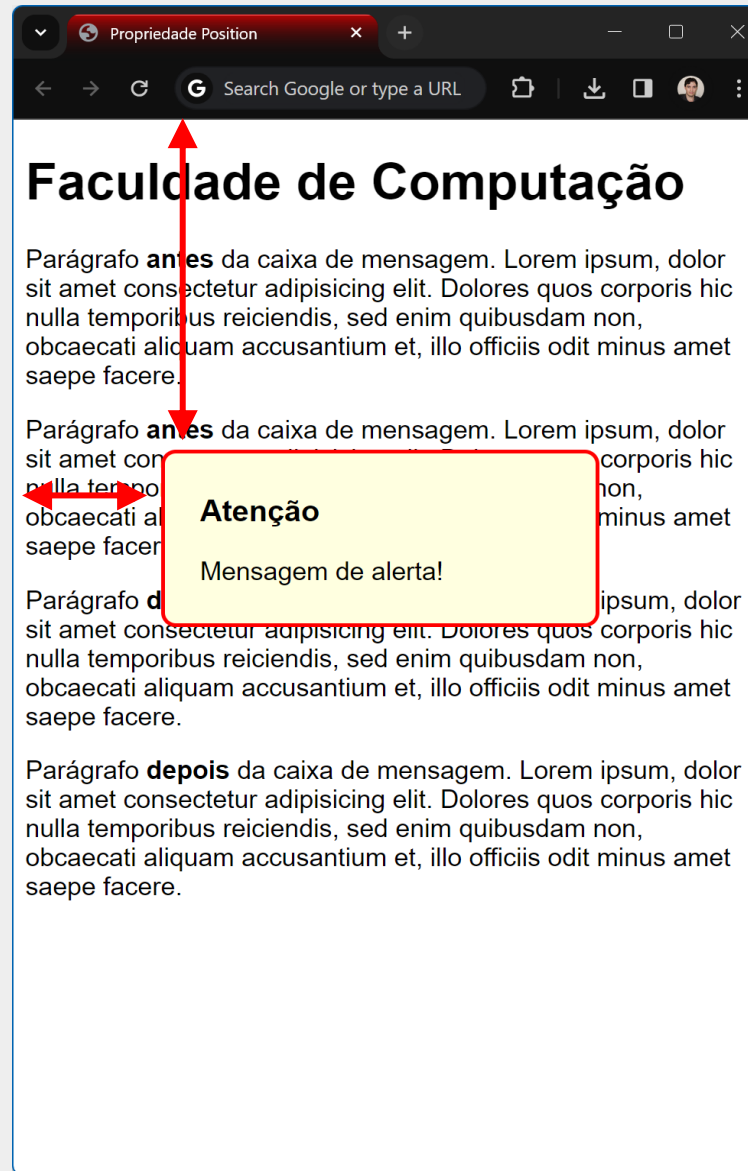
```
.messageBox {  
  width: 25ch;  
  border: solid 3px red;  
  border-radius: 10px;  
  background-color: lightyellow;  
  padding: 0.5rem 1.5rem;  
}  
</style>  
</head>  
<body>  
  <main>  
    <h1>Faculdade de Computação</h1>  
    <p>Parágrafo <b>antes</b> da caixa</p>  
    <p>Parágrafo <b>antes</b> da caixa</p>  
    <div class="messageBox">  
      <h3>Atenção</h3>  
      <p>Mensagem de alerta!</p>  
    </div>  
    <p>Parágrafo <b>depois</b> da caixa</p>  
    <p>Parágrafo <b>depois</b> da caixa</p>  
  </main>  
</body>
```



A caixa de mensagem com borda vermelha deste exemplo é um elemento `div` com posicionamento padrão (`static`). Repare que há dois parágrafos antes e dois parágrafos depois.

Exemplo de position: absolute

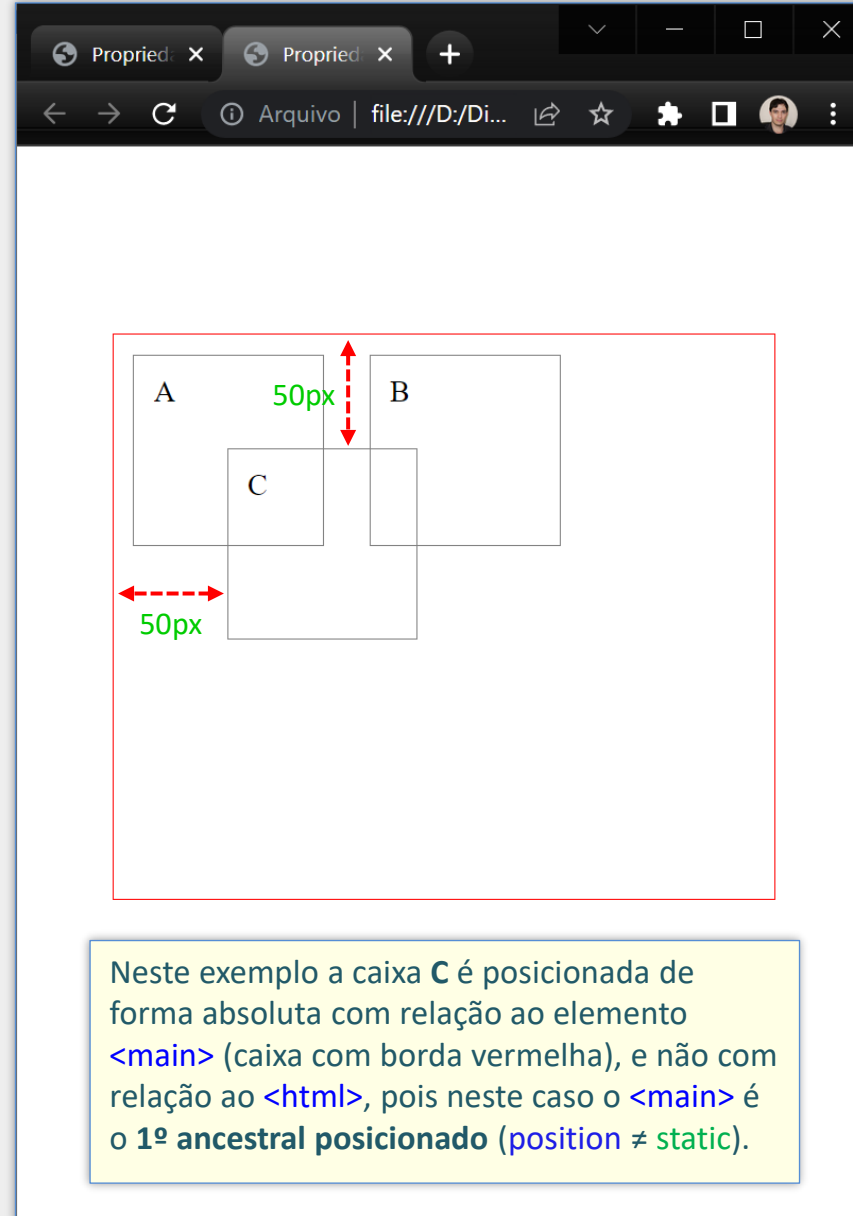
```
.messageBox {  
  width: 25ch;  
  border: solid 3px red;  
  border-radius: 10px;  
  background-color: lightyellow;  
  padding: 0.5rem 1.5rem;  
  position: absolute;  
  top: 230px;  
  left: 100px;  
}  
</style>  
</head>  
<body>  
  <main>  
    <h1>Faculdade de Computação</h1>  
    <p>Parágrafo antes da caixa</p>  
    <p>Parágrafo antes da caixa</p>  
    <div class="messageBox">  
      <h3>Atenção</h3>  
      <p>Mensagem de alerta!</p>  
    </div>  
    <p>Parágrafo depois da caixa</p>  
    <p>Parágrafo depois da caixa</p>  
  </main>  
</body>
```



Neste exemplo o posicionamento da caixa de mensagem é alterado para **absolute** e sua posição é ajustada com relação ao elemento raiz (<html>), pois nenhum outro ancestral tem **position ≠ static**. Observe que a caixa saiu do layout e aparece sobre o restante do conteúdo e os dois parágrafos seguintes ocuparam o seu lugar original.

Exemplo de position: absolute

```
<style>
  main {
    width: 80%; height: 300px;
    margin: 100px auto;
    border: 1px solid red;
    position: relative;
  }
  article {
    margin: 10px; padding: 10px;
    border: 1px solid #555;
    width: 80px; height: 80px;
    display: inline-block;
  }
  .posAbs {
    position: absolute;
    left: 50px;
    top: 50px;
  }
</style>
</head>
<body>
  <main>
    <article>A</article>
    <article>B</article>
    <article class="posAbs">C</article>
  </main>
</body>
```



Propriedade `position` - continuação

`position: sticky`

- Elemento posicionado de acordo com fluxo normal do documento
- Elemento continua **ocupando espaço no layout**
- Elemento "gruda" no primeiro ancestral com mecanismo de rolagem
- Normalmente é utilizado em conjunto com `top: 0`

Exemplos disponíveis em <https://youtu.be/caJ7Q65aiLE?t=2200>

Exemplo 1 de position: sticky

```
<header>
  <h1>Prof. Daniel A. Furtado</h1>
  <nav>
    <ul>
      <li><a href="#">INÍCIO</a></li>
      <li><a href="#">ENSINO</a></li>
      <li><a href="#">PROJETOS</a></li>
      <li><a href="#">PUBLICAÇÕES</a></li>
    </ul>
  </nav>
</header>
```

```
header {
  width: 100%;
  background-color: #555;
  color: #ddd;
  padding-top: 5px;
  margin: 0;
  position: sticky;
  top: 0;
}
```

Prof. Daniel A. Furtado

INÍCIO ENSINO PROJETOS PUBLICAÇÕES

Programação para Internet

A disciplina tem como objetivo capacitar o aluno para o desenvolvimento de aplicações Web utilizando as tecnologias de base, com foco no desenvolvimento do front-end e na programação direta do back-end, incluindo acesso a banco de dados.

Os objetivos específicos incluem: 1) discutir o funcionamento de sistemas Web e os protocolos envolvidos; 2) discutir o paradigma da programação para a Web e 3) desenvolver interfaces gráficas para a Web e 4) desenvolver websites dinâmicos e interativos através da programação direta do back-end e 5) utilizar conceitos e tecnologias para acesso a banco de dados em sistemas Web.

Serão aplicadas três avaliações práticas, um projeto de implementação e vários testes de aula. As avaliações práticas devem ser realizadas em horário de aula, sob supervisão do

Página com cabeçalho (que inclui barra de navegação) fixado com `position: sticky`. Exibição sem rolagem vertical.

Prof. Daniel A. Furtado

INÍCIO ENSINO PROJETOS PUBLICAÇÕES

devem ser realizadas em horário de aula, sob supervisão do professor. O projeto de implementação deverá ser apresentado pela equipe no final do semestre letivo, conforme cronograma disponibilizado pelo professor.

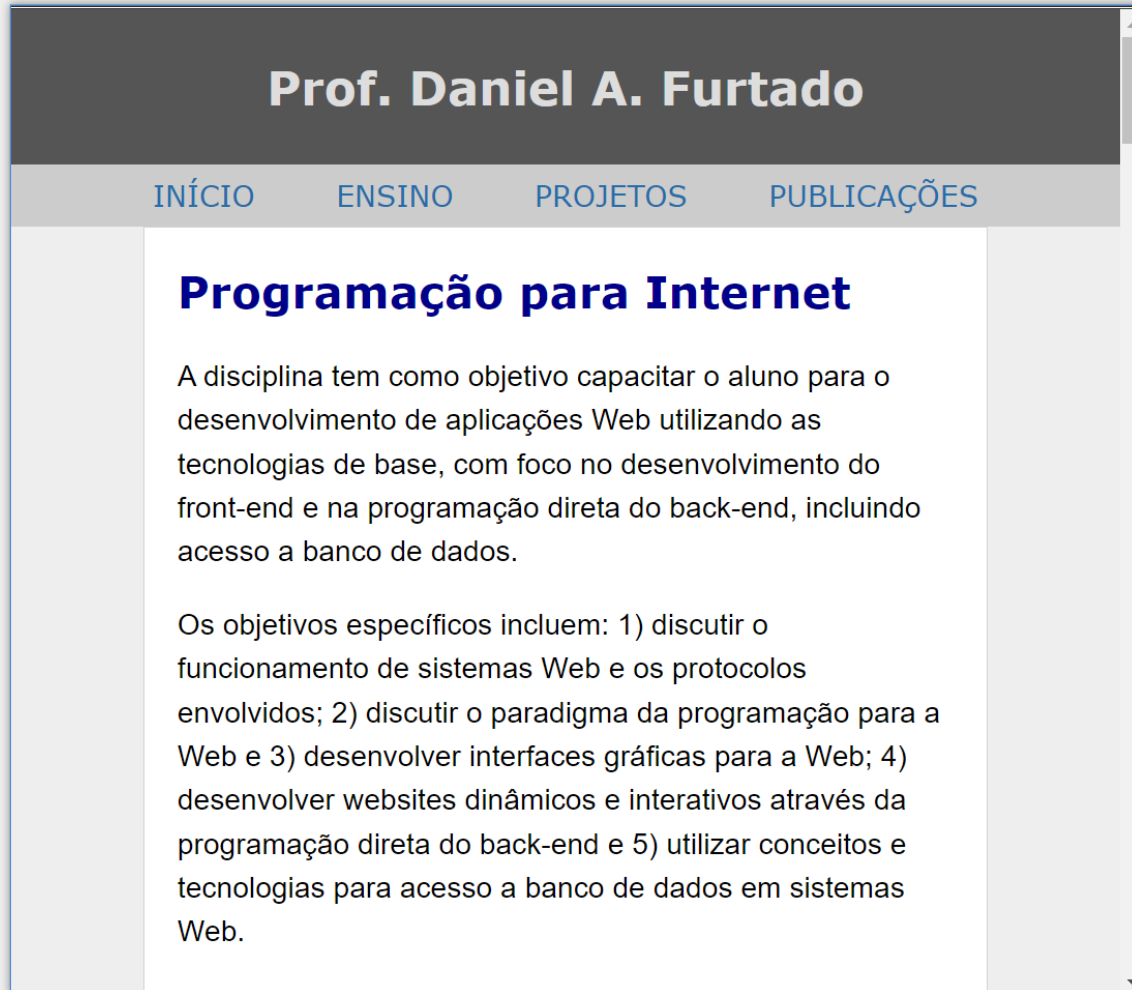
A disciplina tem como objetivo capacitar o aluno para o desenvolvimento de aplicações Web utilizando as tecnologias de base, com foco no desenvolvimento do front-end e na programação direta do back-end, incluindo acesso a banco de dados.

Os objetivos específicos incluem: 1) discutir o funcionamento de sistemas Web e os protocolos envolvidos; 2) discutir o paradigma da programação para a Web e 3) desenvolver interfaces gráficas para a Web; 4) desenvolver websites dinâmicos e interativos através da programação direta do back-end e 5) utilizar conceitos e tecnologias para acesso a banco de dados em sistemas Web.

Serão aplicadas três avaliações práticas, um projeto de

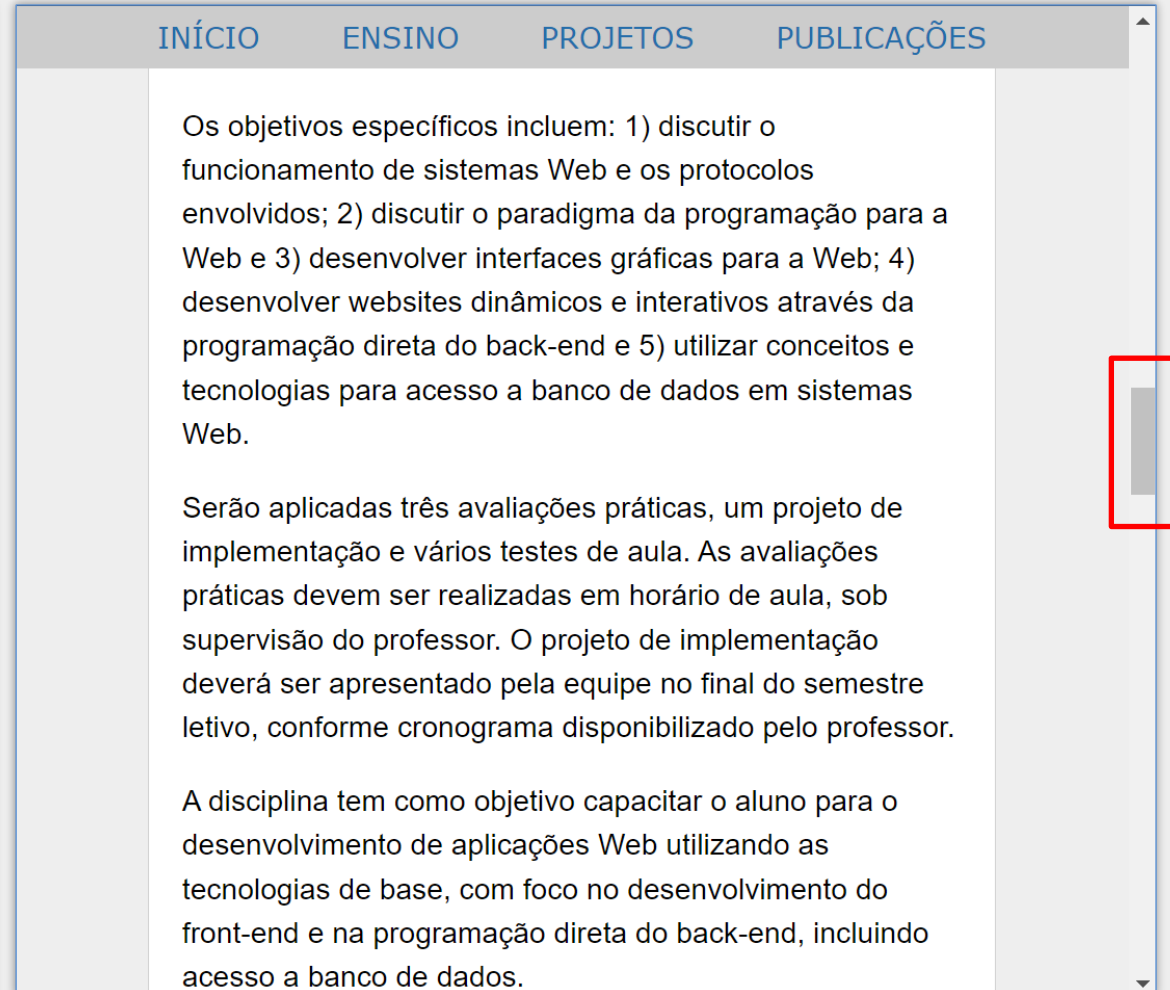
Mesma página sendo exibida após rolagem da tela. Repare que o cabeçalho continua sendo exibido no topo da página.

Exemplo 2 de position: sticky



The screenshot shows a website header for Prof. Daniel A. Furtado. The header has a dark grey background with the name 'Prof. Daniel A. Furtado' in white. Below the name is a navigation bar with four items: 'INÍCIO', 'ENSINO', 'PROJETOS', and 'PUBLICAÇÕES'. The 'INÍCIO' item is highlighted in blue. Below the navigation bar is a white area with the title 'Programação para Internet' in bold blue text. Below the title is a paragraph of text: 'A disciplina tem como objetivo capacitar o aluno para o desenvolvimento de aplicações Web utilizando as tecnologias de base, com foco no desenvolvimento do front-end e na programação direta do back-end, incluindo acesso a banco de dados.' Below this paragraph is another paragraph: 'Os objetivos específicos incluem: 1) discutir o funcionamento de sistemas Web e os protocolos envolvidos; 2) discutir o paradigma da programação para a Web e 3) desenvolver interfaces gráficas para a Web; 4) desenvolver websites dinâmicos e interativos através da programação direta do back-end e 5) utilizar conceitos e tecnologias para acesso a banco de dados em sistemas Web.'

Neste exemplo a barra de navegação (<nav>) foi colocada fora do <header> e seu posicionamento foi feito com **position: sticky**.



The screenshot shows a website page with a navigation bar at the top. The navigation bar has four items: 'INÍCIO', 'ENSINO', 'PROJETOS', and 'PUBLICAÇÕES'. The 'INÍCIO' item is highlighted in blue. Below the navigation bar is a white area with a paragraph of text: 'Os objetivos específicos incluem: 1) discutir o funcionamento de sistemas Web e os protocolos envolvidos; 2) discutir o paradigma da programação para a Web e 3) desenvolver interfaces gráficas para a Web; 4) desenvolver websites dinâmicos e interativos através da programação direta do back-end e 5) utilizar conceitos e tecnologias para acesso a banco de dados em sistemas Web.' Below this paragraph is another paragraph: 'Serão aplicadas três avaliações práticas, um projeto de implementação e vários testes de aula. As avaliações práticas devem ser realizadas em horário de aula, sob supervisão do professor. O projeto de implementação deverá ser apresentado pela equipe no final do semestre letivo, conforme cronograma disponibilizado pelo professor.' Below this paragraph is a third paragraph: 'A disciplina tem como objetivo capacitar o aluno para o desenvolvimento de aplicações Web utilizando as tecnologias de base, com foco no desenvolvimento do front-end e na programação direta do back-end, incluindo acesso a banco de dados.'

Após rolagem vertical da página, a barra de navegação sobe e 'gruda' no topo, permanecendo nesse local com o incremento da rolagem.

Propriedade `position` - continuação

`position: fixed`

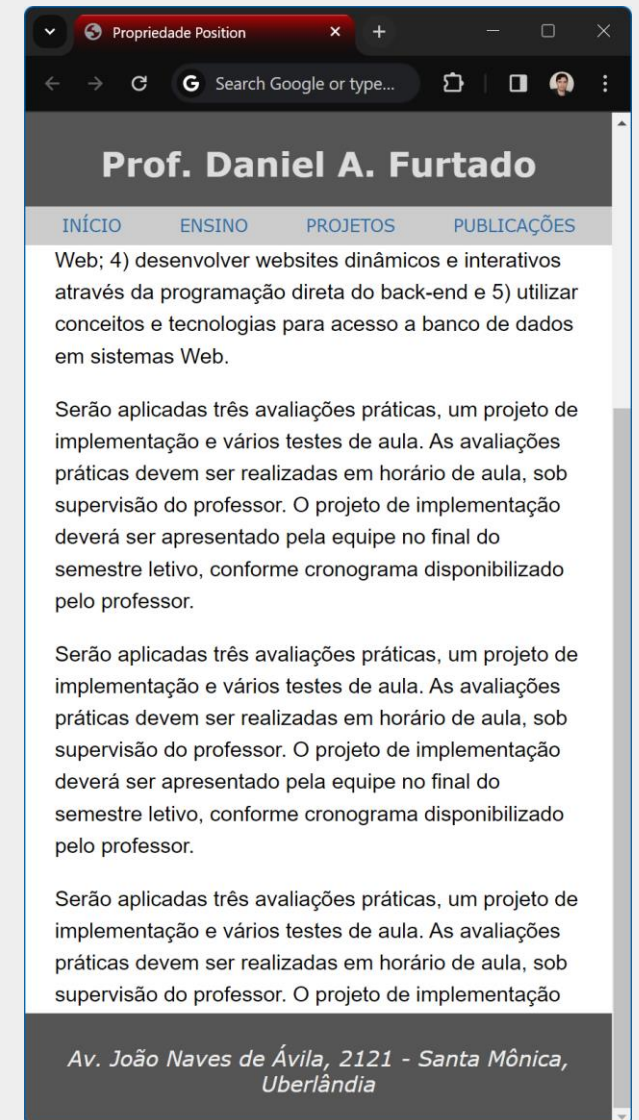
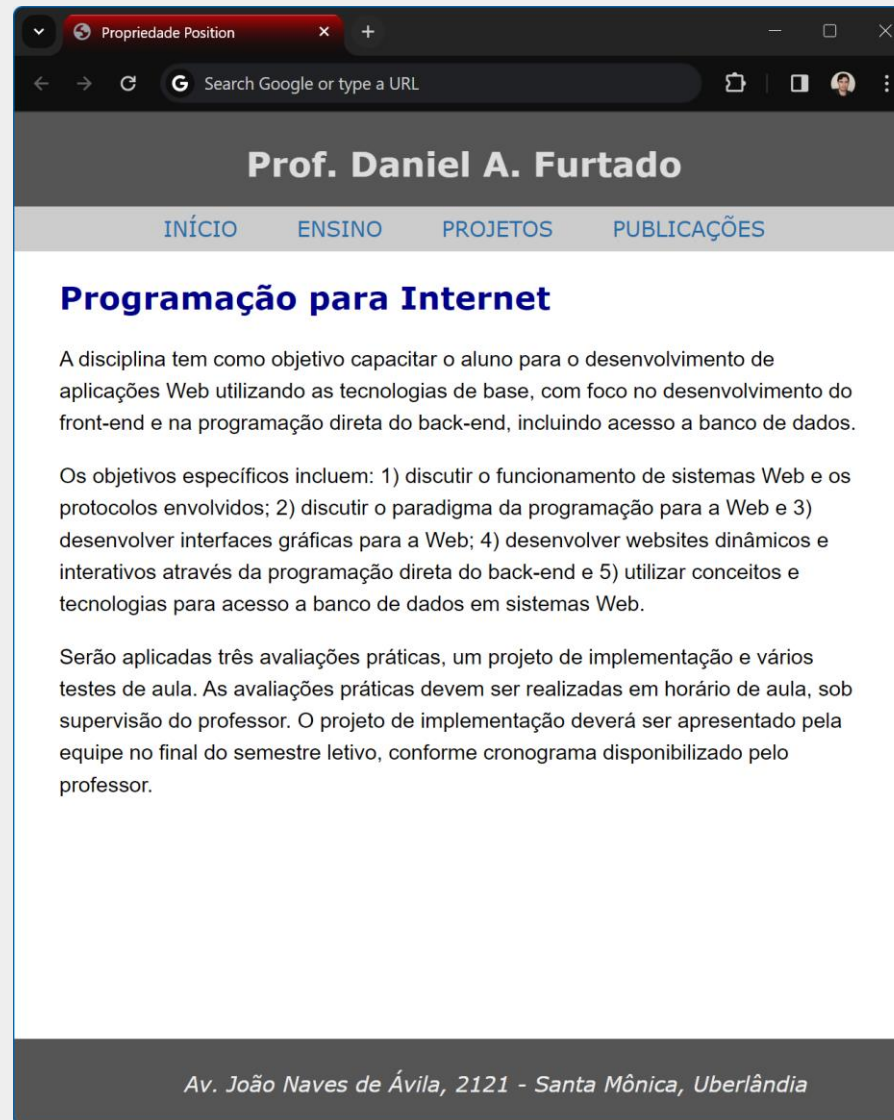
- Elemento **removido do layout** e posicionado com relação à **viewport** utilizando `top`, `left`, `right` e `bottom`
- A posição do elemento **não se altera com a rolagem** da página
- Quando impresso, aparecerá na mesma posição em todas as páginas
- Geralmente utilizado quando se deseja posicionar algo de maneira fixa na tela, sobre o restante do conteúdo, como em painéis de avisos sobre cookies, janelas modais etc.

Exemplos disponíveis em <https://youtu.be/caJ7Q65aiLE?t=2200>

Exemplo de position: fixed

```
footer {  
  box-sizing: border-box;  
  width: 100%;  
  padding: 1.5rem;  
  color: #eee;  
  background-color: #555;  
  text-align: center;  
  position: fixed;  
  bottom: 0;  
}
```

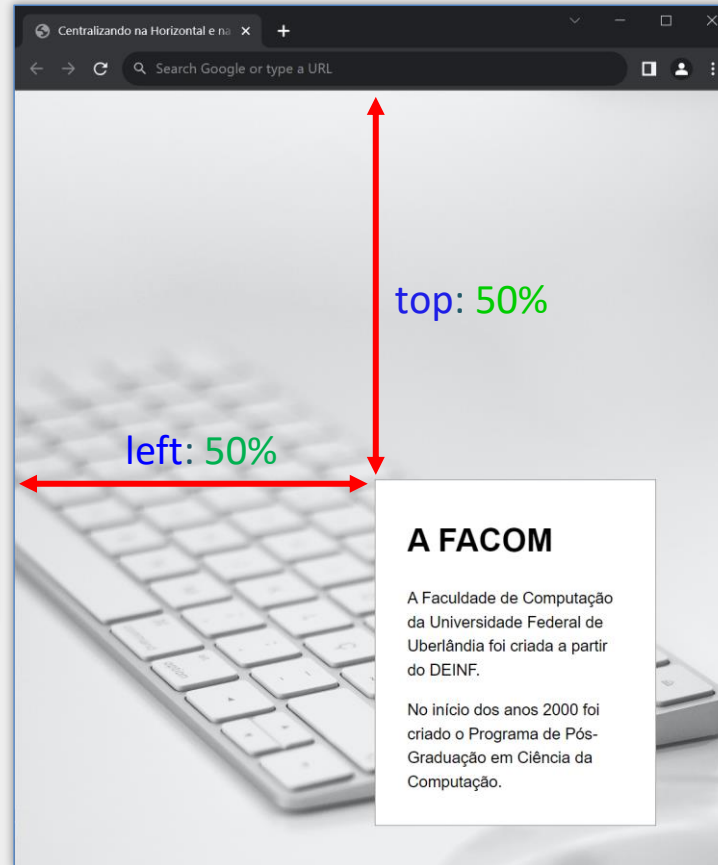
Neste exemplo a faixa de rodapé é fixada na base da viewport utilizando `position: fixed` em conjunto com `bottom: 0`. Observe que a faixa permanece na tela mesmo após rolagem da página. Portanto, ela poderá cobrir o conteúdo propriamente dito (segunda imagem ao lado).



Propriedade transform

- Permite mover, rotacionar, torcer e escalonar um elemento
- Exemplos:
 - `transform: translateX(50px);` move o elemento 50px na horizontal (p/ direita)
 - `transform: translateY(50px);` move o elemento 50px na vertical (p/ baixo)
 - `transform: translate(30px,10px);` move 30px na horizontal e 10px na vertical
 - `transform: rotate(45deg);` rotaciona o elemento em 45 graus
 - `transform: scale(2);` dobra o tamanho nas duas direções

Centralizando com Posicionamento Absoluto e transform



A caixa "FACOM" tem posicionamento absoluto com relação ao elemento raiz. Seu **canto superior esquerdo** é posicionado ao centro utilizando: `top: 50%; left: 50%`

```
.caixaFacom {  
  width: 30%;  
  position: absolute;  
  top: 50%;  
  left: 50%;  
}
```



Acrescentando a propriedade `transform` com o valor `translate(-50%, -50%)`, aplicamos uma translação na horizontal para a esquerda equivalente a 50% de sua **largura**, e uma translação na vertical para cima equivalente a 50% de sua **altura**. Como resultado, o **centro** da caixa é colocado no centro do container.

OBS: vale lembrar que esta forma de centralização remove o elemento do layout, deixando ele sobre os demais.

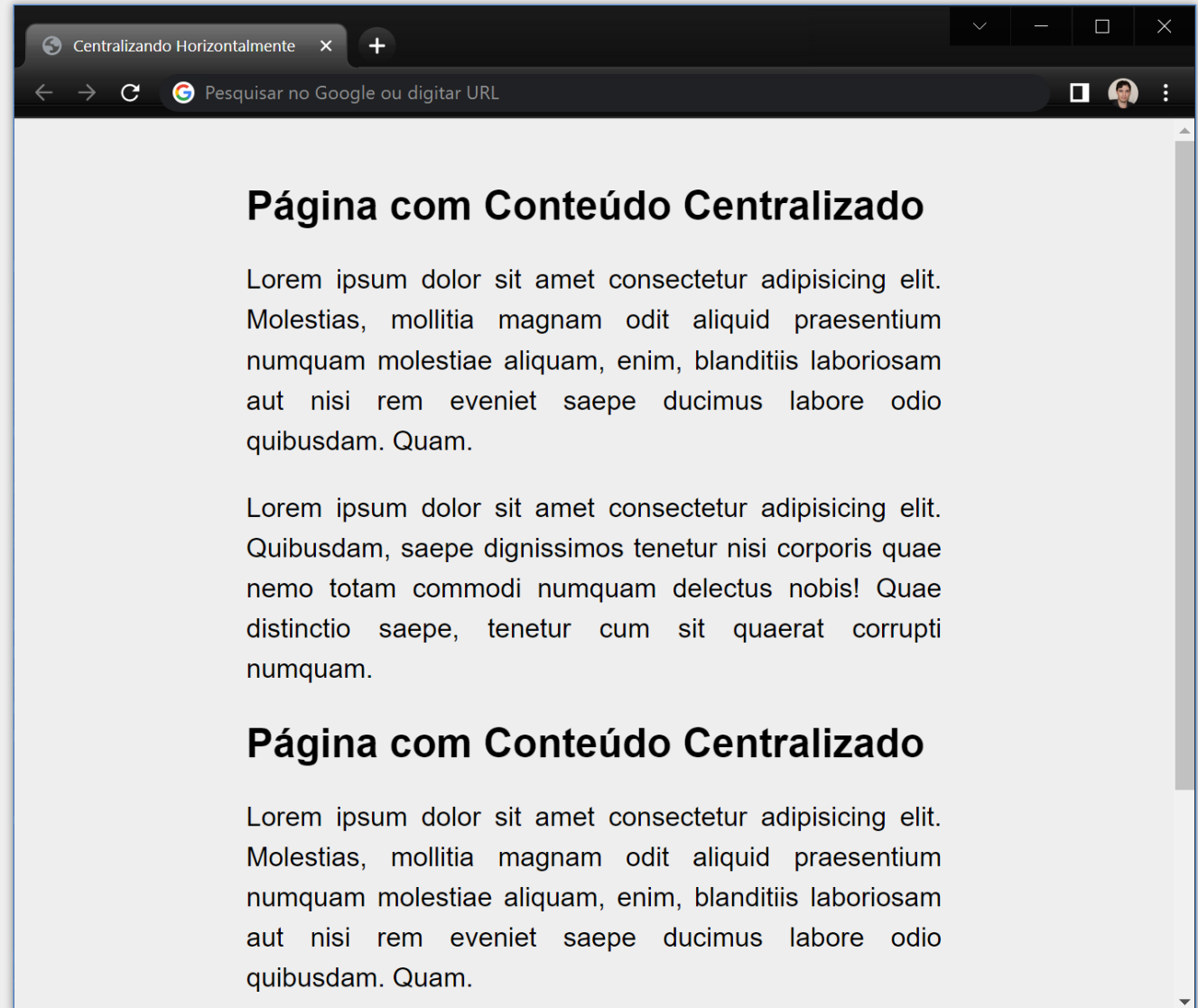
```
.caixaFacom {  
  width: 30%;  
  position: absolute;  
  top: 50%;  
  left: 50%;  
  transform: translate(-50%, -50%);  
}
```

Centralizando na Horizontal com `width` e `margin`

- Uma forma de centralizar **horizontalmente** um elemento de bloco, sem removê-lo do layout, é utilizando as propriedades `width` e `margin`
- Define-se uma largura com `width` e coloca-se a **margens laterais** em `auto`
- Com as margens laterais em `auto`, o navegador ajustará os valores igualmente para preencher o espaço, resultando na centralização
- Para centralizar apenas o texto dentro de um elemento, pode-se utilizar apenas `text-align: center`

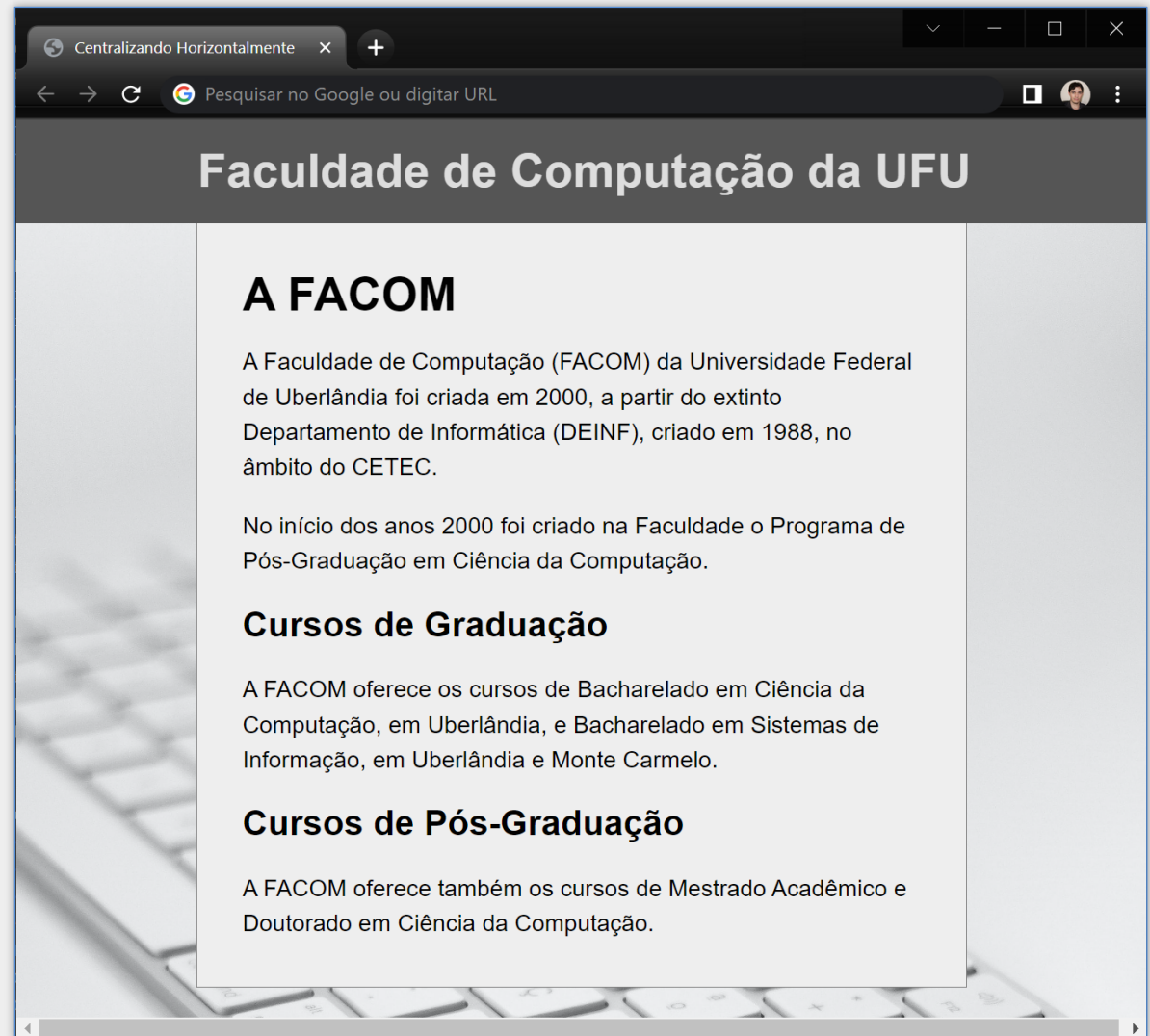
Centralizando na Horizontal com width e margin

```
<style>
  body {
    text-align: justify;
    width: 60%;
    margin: 40px auto;
  }
</style>
</head>
<body>
  <h2>Página com Conteúdo Centr
  <p>Lorem ipsum dolor sit amet
  <p>Lorem ipsum dolor sit amet
  <h2>Página com Conteúdo Centr
  <p>Lorem ipsum dolor sit amet
  <p>Lorem ipsum dolor sit amet
</body>
```



Centralizando na Horizontal com width e margin

```
header {
  background-color: #555;
  width: 100%;
}
main {
  background-color: #eee;
  width: 60%;
  margin: 0 auto;
}
</style>
</head>
<body>
  <header>
    <h1>Faculdade de Computação da UFU</h1>
  </header>
  <main>
    <h1>A FACOM</h1>
    <p>A Faculdade de Computação (FACOM) da
      Universidade Federal de Uberlândia foi
      a partir do extinto Departamento de In
```



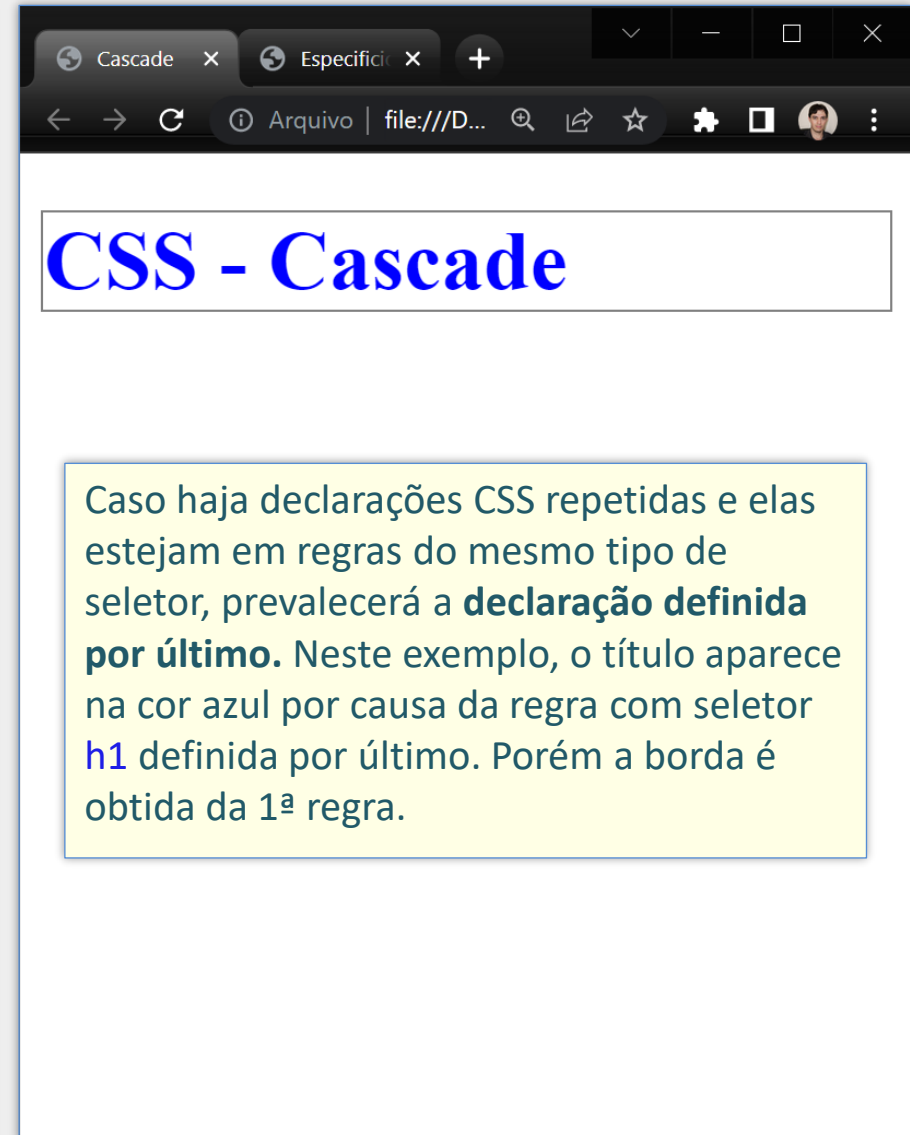
Cascade, Especificidade e Herança

Cascade, Especificidade e Herança

- Em caso de regras conflitantes, qual regra será aplicada?
- Problemas comuns:
 - Regras CSS sem efeito
 - Resultado muito diferente do esperado
- Nessas situações é fundamental conhecer como o navegador decide pelas regras
 - Ordem no código
 - Especificidade
 - Herança

Ordem das Regras – Cascade

```
<style>
  h1 {
    color: red;
    border: 1px solid gray;
  }
  h1 {
    color: blue;
  }
</style>
</head>
<body>
  <h1>
    CSS - Cascade
  </h1>
</body>
```



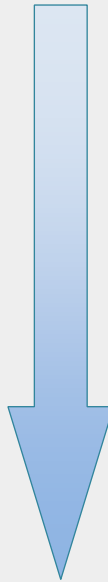
Especificidade

Seletor de Elemento

Seletor de Classe

Seletor de ID

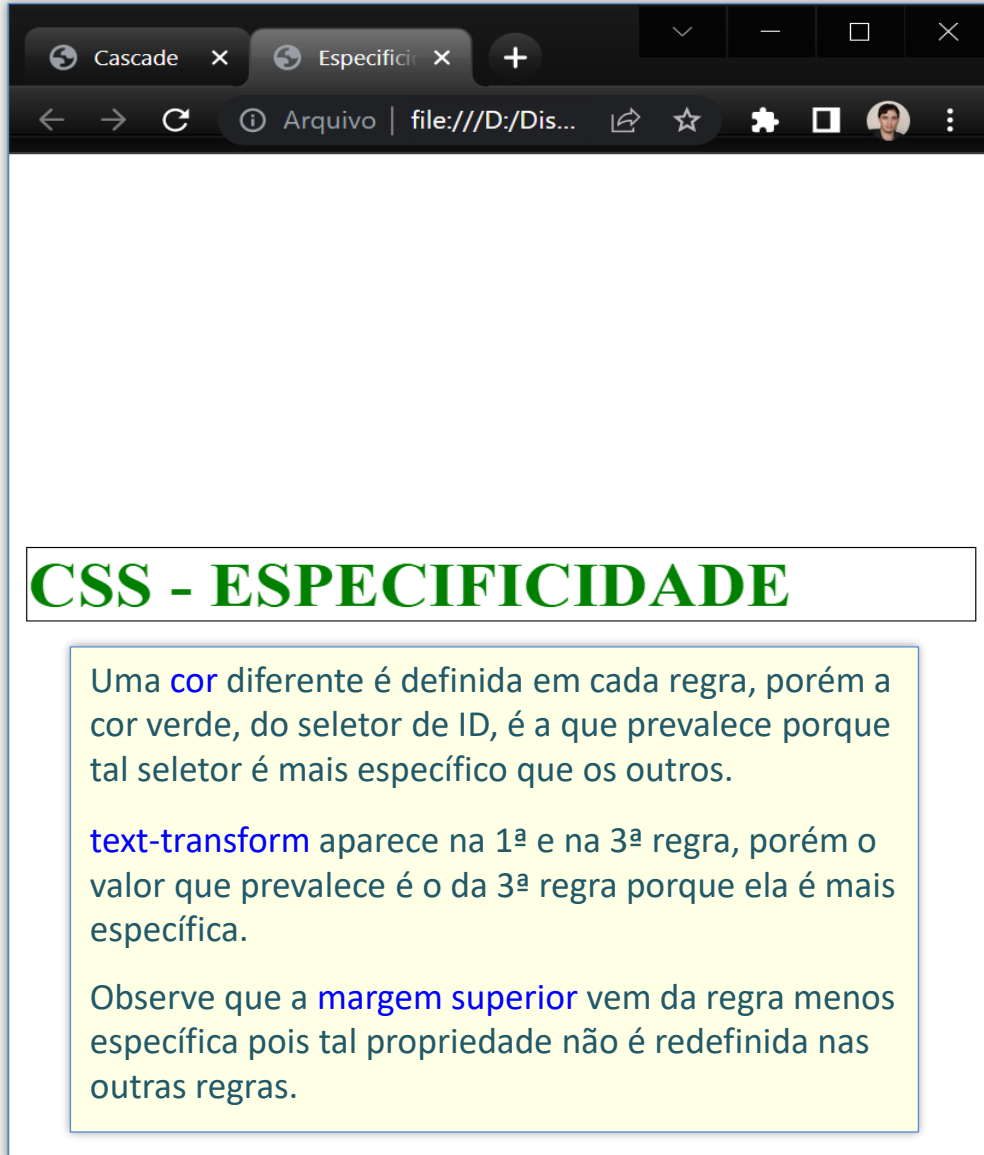
Código Inline



Maior especificidade
Sobreposição de declarações anteriores

Especificidade das Regras

```
<style>
  h1 {
    color: red;
    text-transform: lowercase;
    margin-top: 200px;
  }
  #tituloVerde {
    color: green;
    border: 1px solid black;
  }
  .tituloAzul {
    color: blue;
    text-transform: uppercase;
  }
</style>
</head>
<body>
  <h1 id="tituloVerde" class="tituloAzul">
    CSS - Especificidade
  </h1>
</body>
```



CSS - ESPECIFICIDADE

Uma cor diferente é definida em cada regra, porém a cor verde, do seletor de ID, é a que prevalece porque tal seletor é mais específico que os outros.

`text-transform` aparece na 1ª e na 3ª regra, porém o valor que prevalece é o da 3ª regra porque ela é mais específica.

Observe que a **margem superior** vem da regra menos específica pois tal propriedade não é redefinida nas outras regras.

Herança

- Algumas propriedades herdam os valores do elemento pai
 - Ex.: `font-family`, `color` e `text-align`
 - Ex.: um `<p>` dentro de um `<div>` herda a fonte definida para o `<div>`
- Outras propriedades não herdam
 - Ex.: `width`, `margin` e `padding`
 - Ex.: um `<p>` dentro de um `<div>` não herda as margens definidas para o `<div>`
- Definir uma propriedade para o valor `inherit` ativa a herança

Ícones

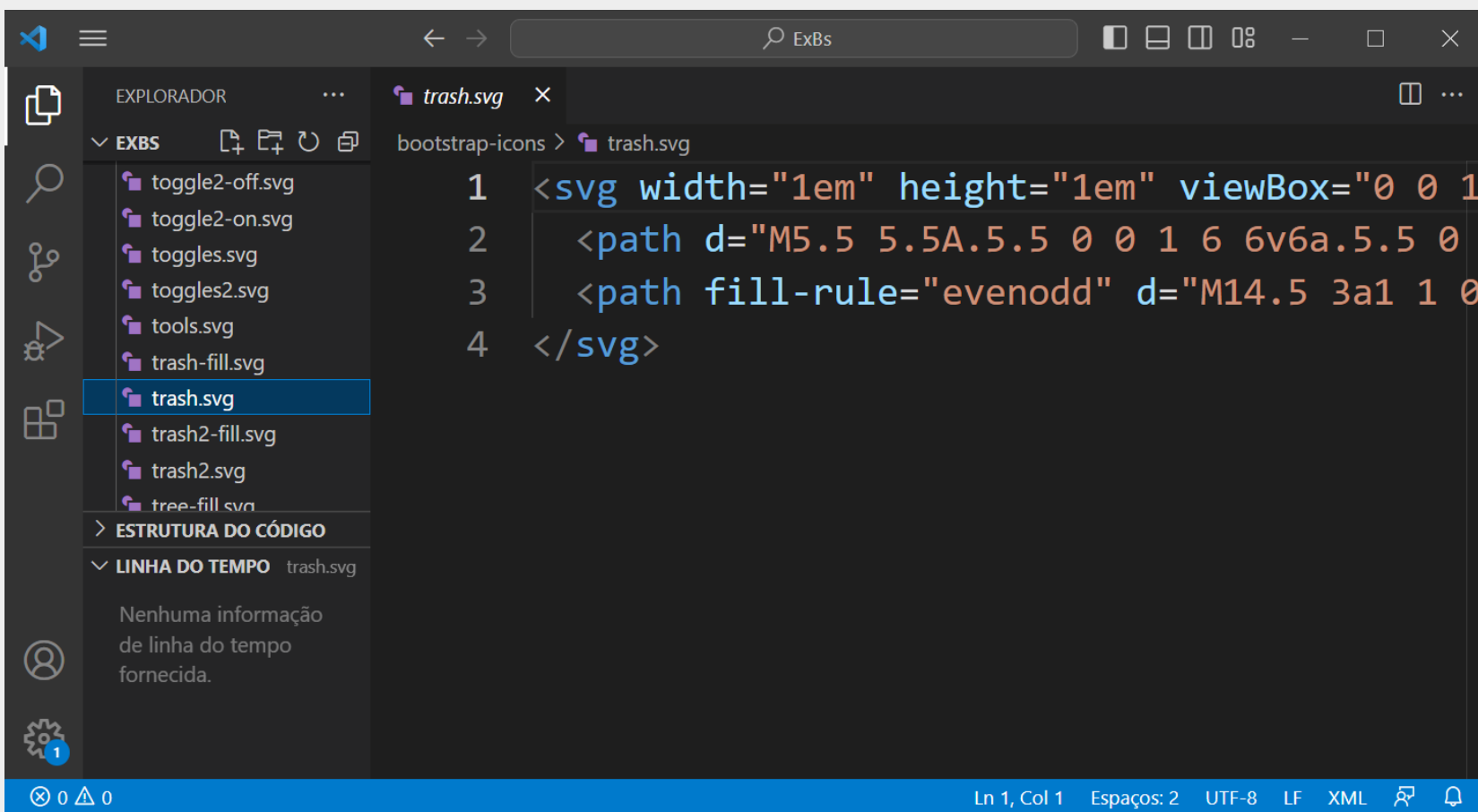
Ícones

- Em páginas web é comum que botões e outros elementos contenham ícones
- Tais ícones podem ser inseridos de várias formas, como:
 - Ícones SVG
 - Ícones de Fonte

Ícones SVG

- SVG → **Scalable Vector Graphics**
- Formato gráfico vetorial em XML
- Melhor dimensionamento mantendo qualidade (independente da resolução)
- Cada ícone é um bloco de código XML (tamanho reduzido)
- Podem ser usados como imagens (tag ``)
- Flexibilidade com ícones multicoloridos e animações

Exemplo de Ícone SVG do Framework Bootstrap



```
1 <svg width="1em" height="1em" viewBox="0 0 16 16">
2   <path d="M5.5 5.5A.5.5 0 0 1 6 6v6a.5.5 0 0 1 6 6" data-bbox="5.5 5.5 6 6" />
3   <path fill-rule="evenodd" data-bbox="14.5 3a1 1 0" />
4 </svg>
```



Conteúdo XML relativo ao ícone `trash.svg` (ícone da lixeira) que faz parte da biblioteca de ícones do Bootstrap (baixado de icons.getbootstrap.com)

Formas de Uso de Ícones SVG

Como imagem

- ``
- Não é considerada boa prática (por eficiência, semântica etc.)

Embutido

- Código XML do ícone embutido no próprio HTML
- Não requer requisições HTTP adicionais

Sprite

- Combina o XML de múltiplos ícones em um único arquivo
- É necessário referenciar um ícone em particular no arquivo utilizando tag `<use>` da XML

Formas de Uso de Ícones SVG - Exemplo

```
<!-- ícone SVG inserido como imagem -->
<button class="btn btn-danger">
  
  Excluir
</button>

<!-- código XML do ícone embutido no próprio HTML -->
<button class="btn btn-danger">
  <svg width="1em" height="1em" viewBox="0 0 16 16" class="bi bi-trash" fill="currentColor"
    xmlns="http://www.w3.org/2000/svg">
    <path d="M5.5 5.5A.5.5 0 0 1 6 6v6a.5.5 0 0 1-1 1 0 0 0 1 1 0 0 1 1 1H13v9a2 2 0 0 1-2 2H5a2 2 0 0 1-2-2V4h" />
  </svg>
  Excluir
</button>

<!-- ícone inserido como SVG sprite -->
<!-- o arquivo 'bootstrap-icons.svg' vem junto com a biblioteca de ícones -->
<button class="btn btn-danger">
  <svg class="bi" width="16" height="16" fill="currentColor">
    <use xlink:href="bootstrap-icons.svg#trash" />
  </svg>
  Excluir
</button>
```

O arquivo [bootstrap-icons.svg](#) contém o código XML de todos os ícones, mas cada ícone tem um nome de identificação (id).

Ícones de Fonte

- Ícones de fonte não utilizam XML ou arquivos de imagem
- Utilizam caracteres de uma **font-family** para renderizar os ícones
- Cada caractere da fonte corresponde a um ícone/símbolo específico
- Os ícones são carregados pelo navegador como um arquivo de fonte, geralmente utilizando links de redes CDN
 - Possui tamanho reduzido comparado aos arquivos de imagem
- Há várias bibliotecas de ícones disponíveis:
 - Font Awesome, Material Icons, Bootstrap Icons etc.
- Os ícones são inseridos utilizando **nomes de classe CSS** em conjunto com o nome do ícone propriamente dito
- Os ícones podem ser estilizados com propriedades CSS

Utilizando os Ícones de Fonte da Coleção Material Icons

1. Escolha o ícone e a família de ícones em:

```
https://fonts.google.com/icons
```

2. inclua uma referência CSS para a fonte de ícones apropriada

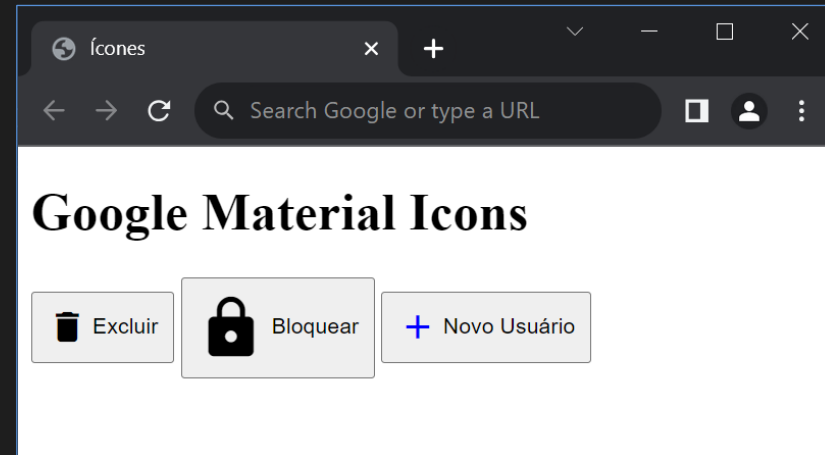
```
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
```

3. Utilize a respectiva **classe** e o nome do ícone como **conteúdo**

```
<span class="material-icons">delete</span>
```

Google Material Icons - Exemplo

```
<!-- Google Material Icons CSS -->
<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
<style>
  button {
    padding: 0.5rem;
  }
  .material-icons {
    vertical-align: middle;
  }
</style>
</head>
<body>
  <h1>Google Material Icons</h1>
  <button>
    <span class="material-icons">delete</span> Excluir
  </button>
  <button>
    <span class="material-icons" style="font-size: 42px">lock</span> Bloquear
  </button>
  <button>
    <span class="material-icons" style="color: blue">add</span> Novo Usuário
  </button>
</body>
```



Referências

- www.w3.org/Style/CSS/
- developer.mozilla.org/en-US/docs/Web/CSS
- www.w3.org/Style/CSS/learning
- <https://getbootstrap.com/docs>
- **HTML and CSS: Design and Build Websites**, Jon Duckett.