Programação para Internet

Módulo 2

Fundamentos da Linguagem CSS

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

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

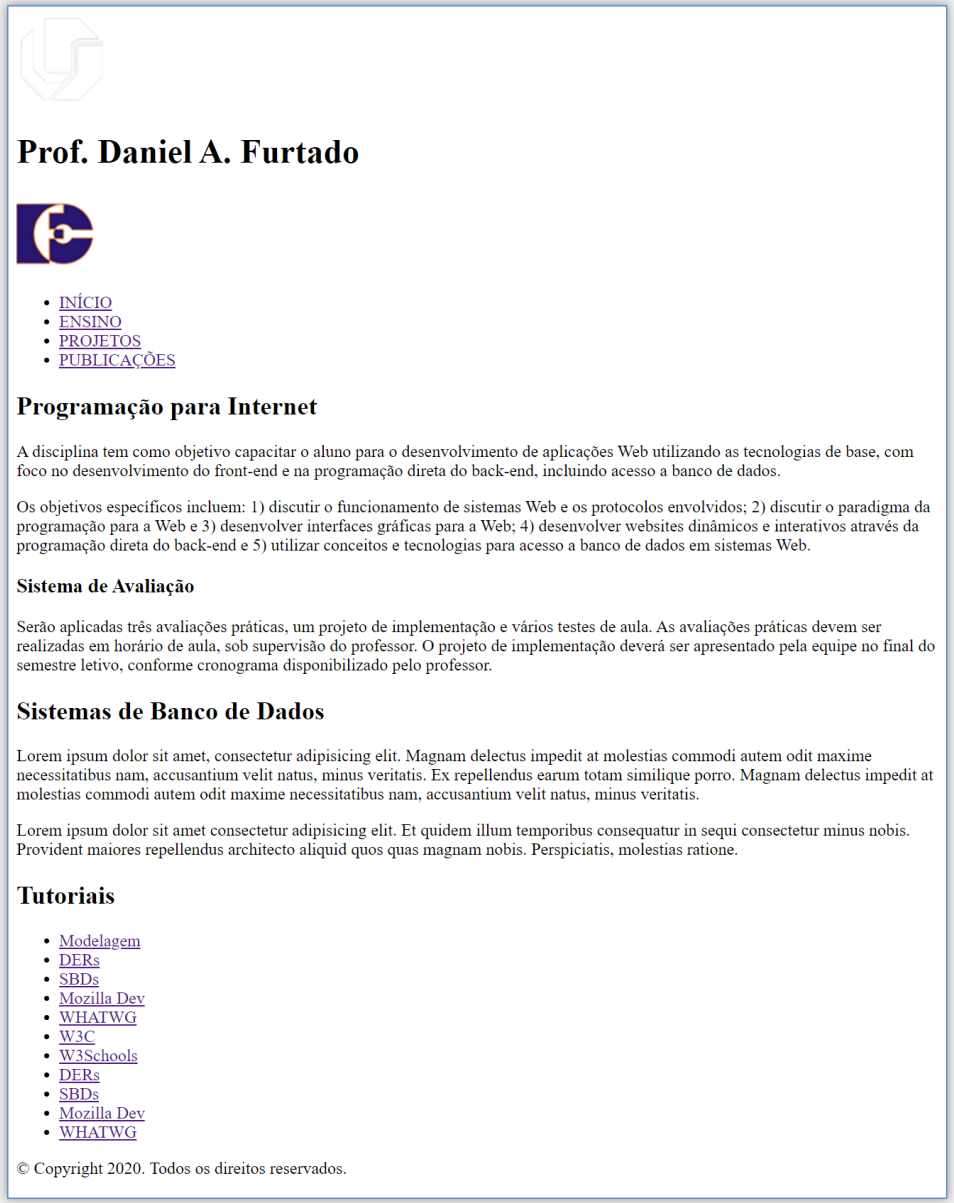
O que é CSS?

- Linguagem de estilos para definir aspectos de apresentação
- **C**ascading **S**tyle **S**heets - 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 adipisicing 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.

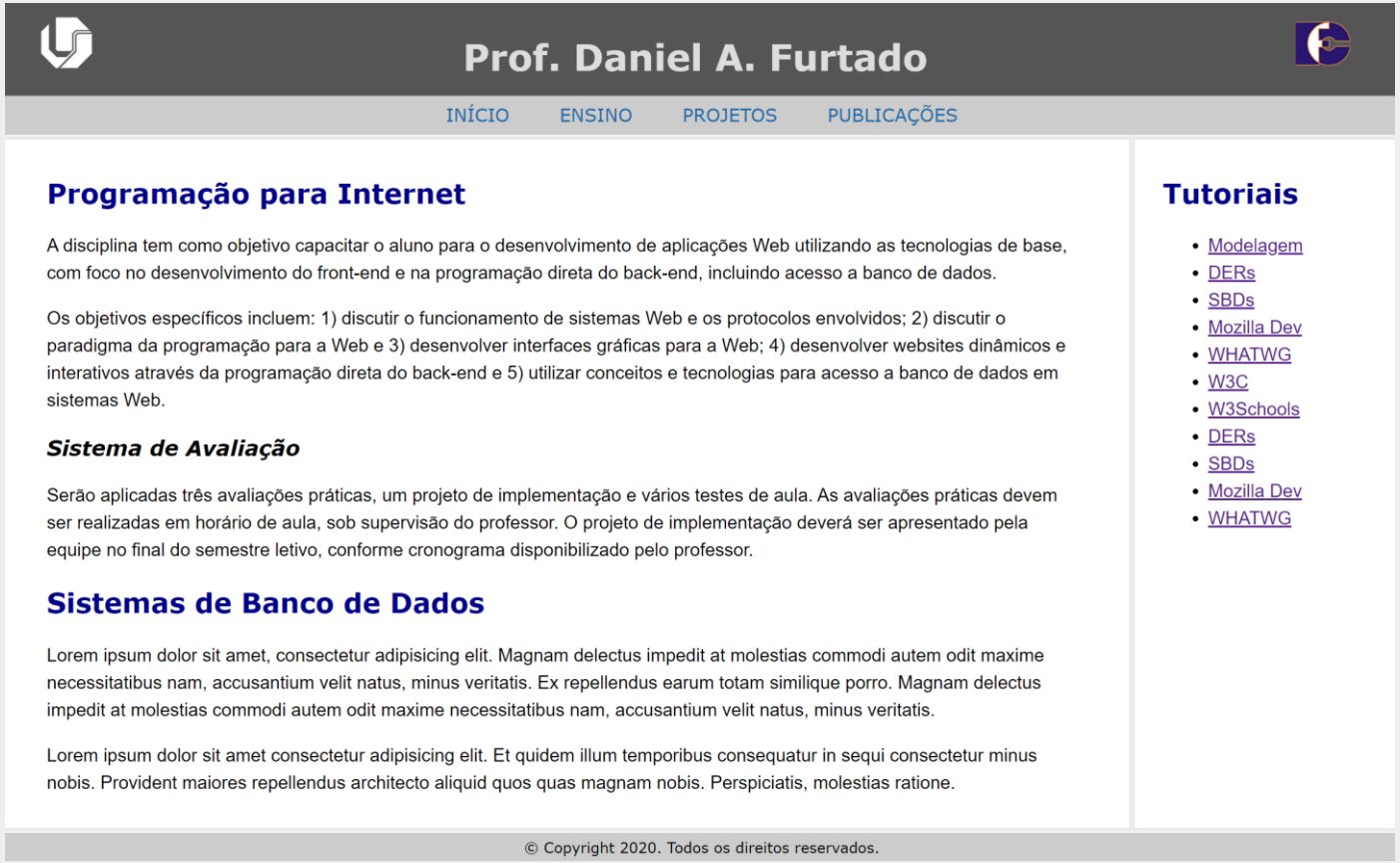
Lorem ipsum dolor sit amet consectetur adipisicing elit. Et quidem illum temporibus consequatur in sequi consectetur minus nobis. Provident maiores repellendus architecto aliquid quos quas magnam nobis. Perspiciatis, molestias ratione.

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 adipisicing 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 adipisicing 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.

Principais 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)*

Atributo ***style***
Inserção de CSS inline

Código CSS Inline

```
<p style="font-size: 14pt; color: blue;">
    Texto em azul com fonte tamanho 14 pontos
</p>
```

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

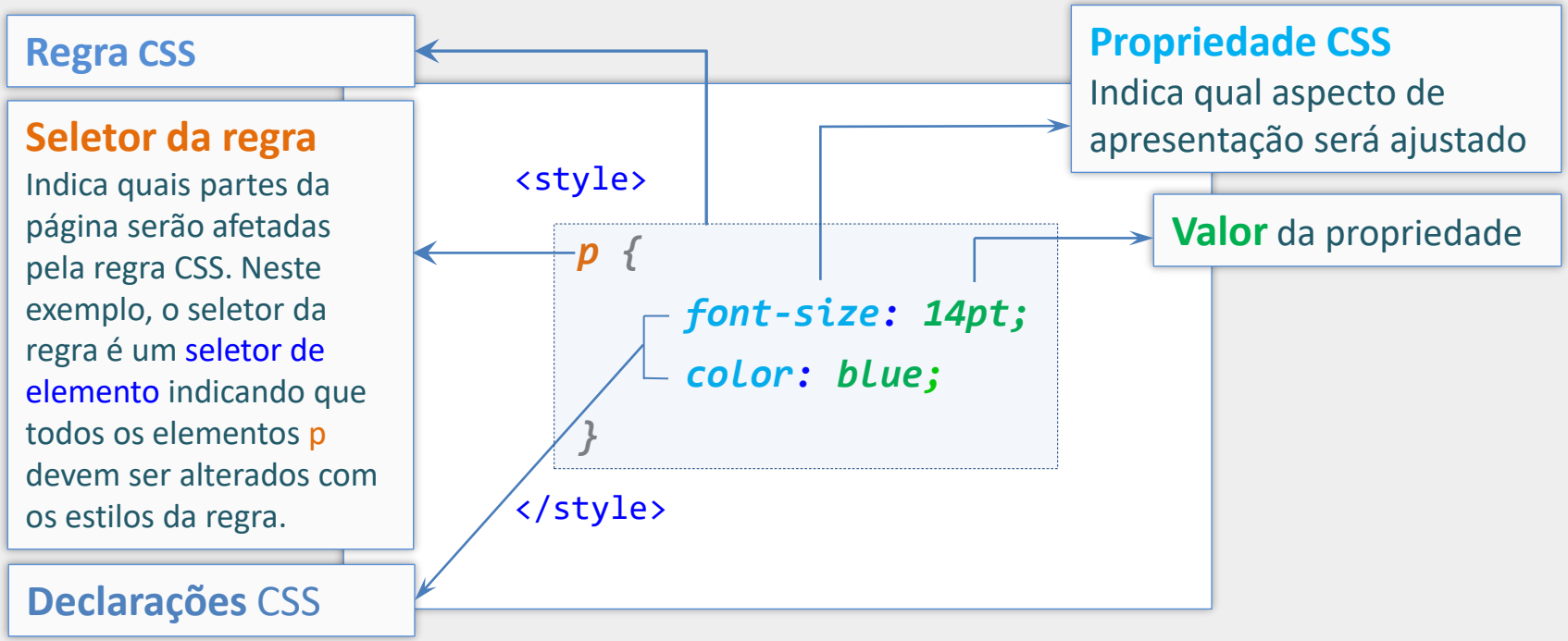
- Exibição adequada no navegador não é garantia de código correto
  - O navegador pode ocultar erros e inconsistências
- Código fora da especificação pode trazer problemas diversos
  - Apresentação inconsistente e imprevisível nos navegadores
- Ferramenta oferecida pelo W3C para validação de código CSS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

Afetará o corpo do documento HTML

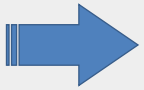
Afetará todos os parágrafos p

Afetará todos os títulos h1

É possível ter múltiplas 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).

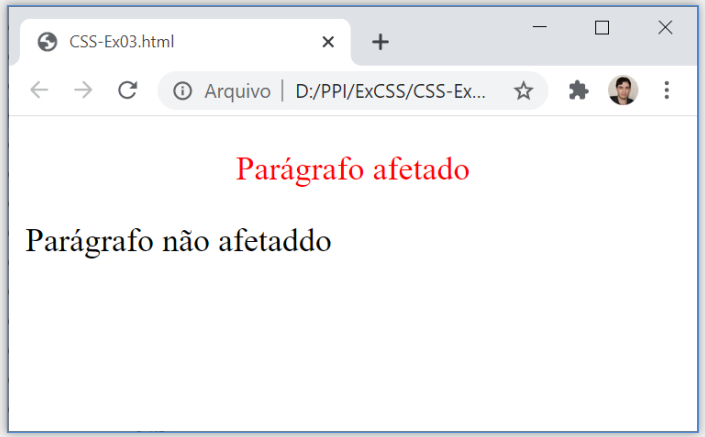
Outros Tipos de Seletores

- 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 descente
  - 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 <li> (Link 2 e Link 3)

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

Seletor de Descendente (*x y*)

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

```
...
  div a {
     text-transform: uppercase;
  }
...
   <body>
     <a href="#">Link 1</a>
     <div>
        <a href="#">Link 2</a>
        <section>
           <a href="#">Link 3</a>
        </section>
     </div>
...
```

Afetará todos os elementos `<a>` que estão 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 não seria afetado porque apesar de ser descendente de um , o Link 3 é filho do elemento <section>, e não de um elemento .

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>
    ...
```

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

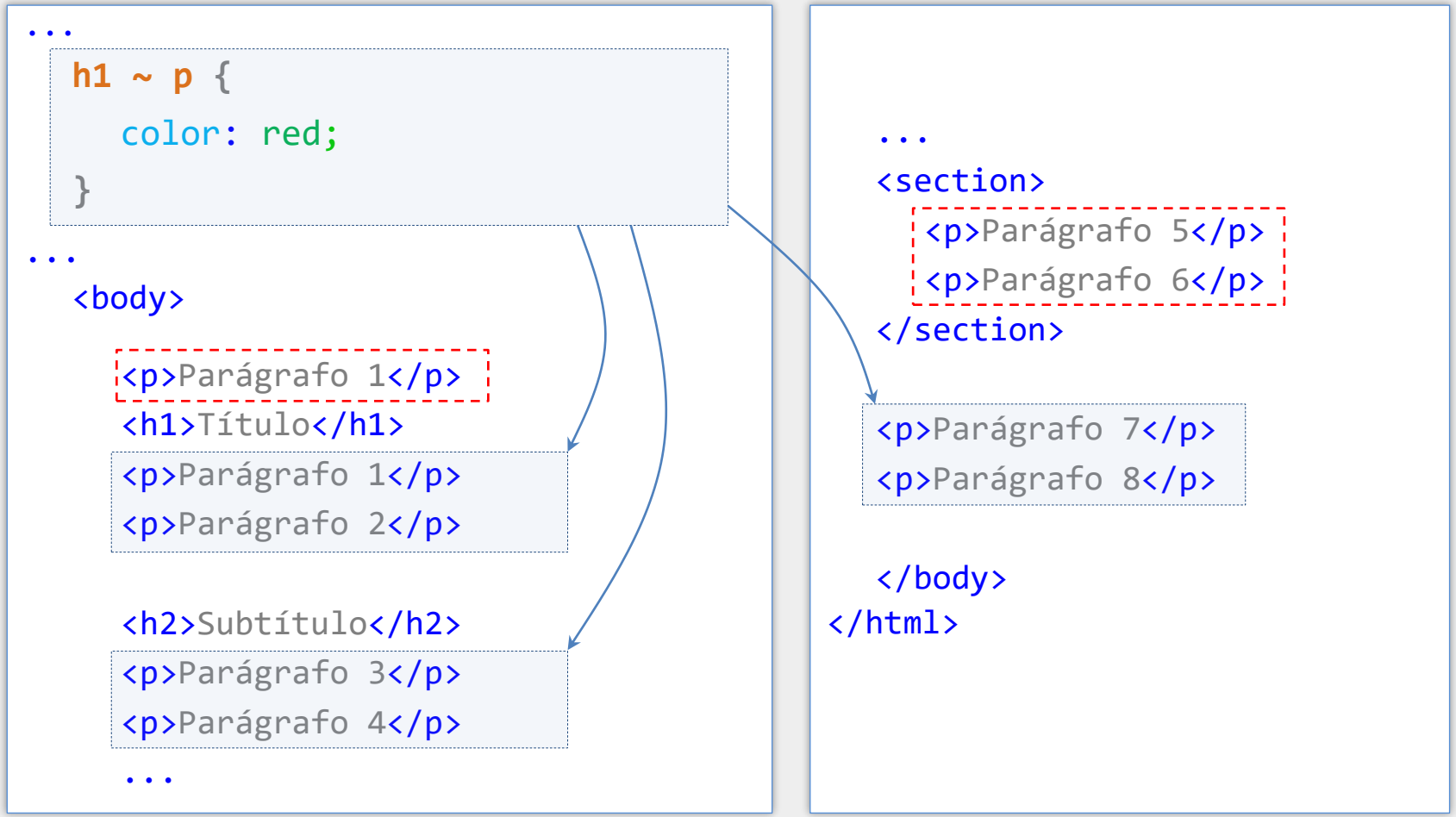
```
   ...
   <h1>Título 3</h1>
   <h2>Subtítulo</h2>
   <p>Parágrafo 5</p>
   <p>Parágrafo 6</p>

   </body>
</html>
```

Parágrafos não afetados devido ao título <h2>

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

Seletor de Irmão Geral (*x ~ 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

```
...
   input[required] {
      background-color: red;
   }
...
<form>
   Nome: <input type="text" required>
   Cidade: <input type="text">
</form>
...
```

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>
...
```

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.

Conteúdo Complementar

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>
...
```

Afetará apenas os `<li>`'s que são filhos de elementos utilizando a classe `.listaHorizontal`

Conteúdo Complementar

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

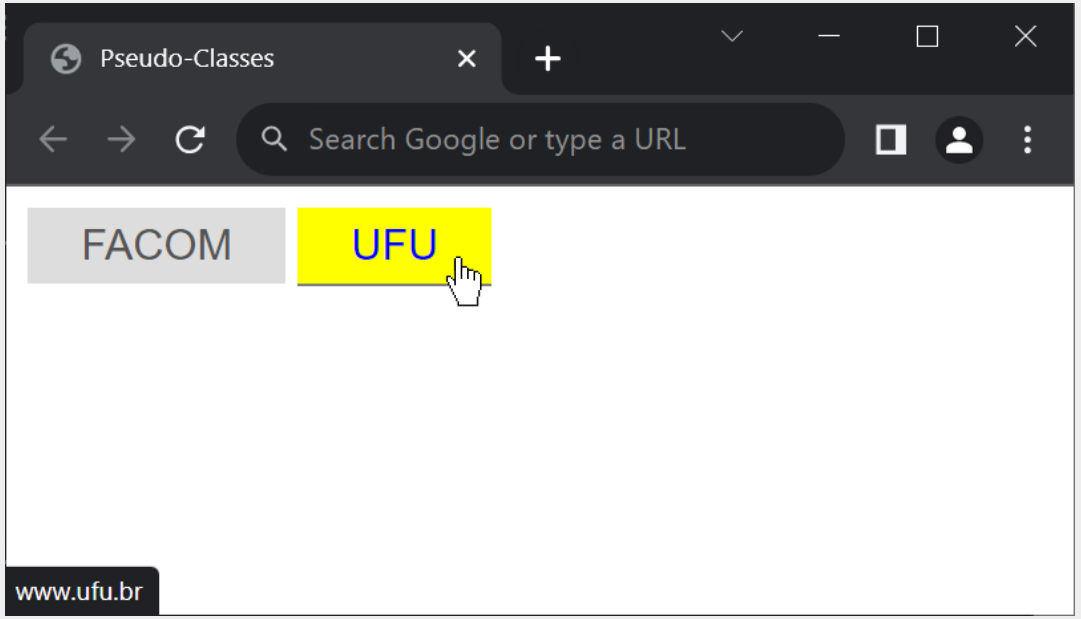
- Uma pseudo-classe permite alterar o estilo de um elemento caso ele se encontra em um estado particular
- Por exemplo, é possível alterar o estilo dos links que já foram visitados ou o estilo dos campos de formulário com conteúdo inválido
- Sintaxe: **seletor** : **pseudo-classe**

Pseudo-Classes Comumente Utilizadas com Links

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

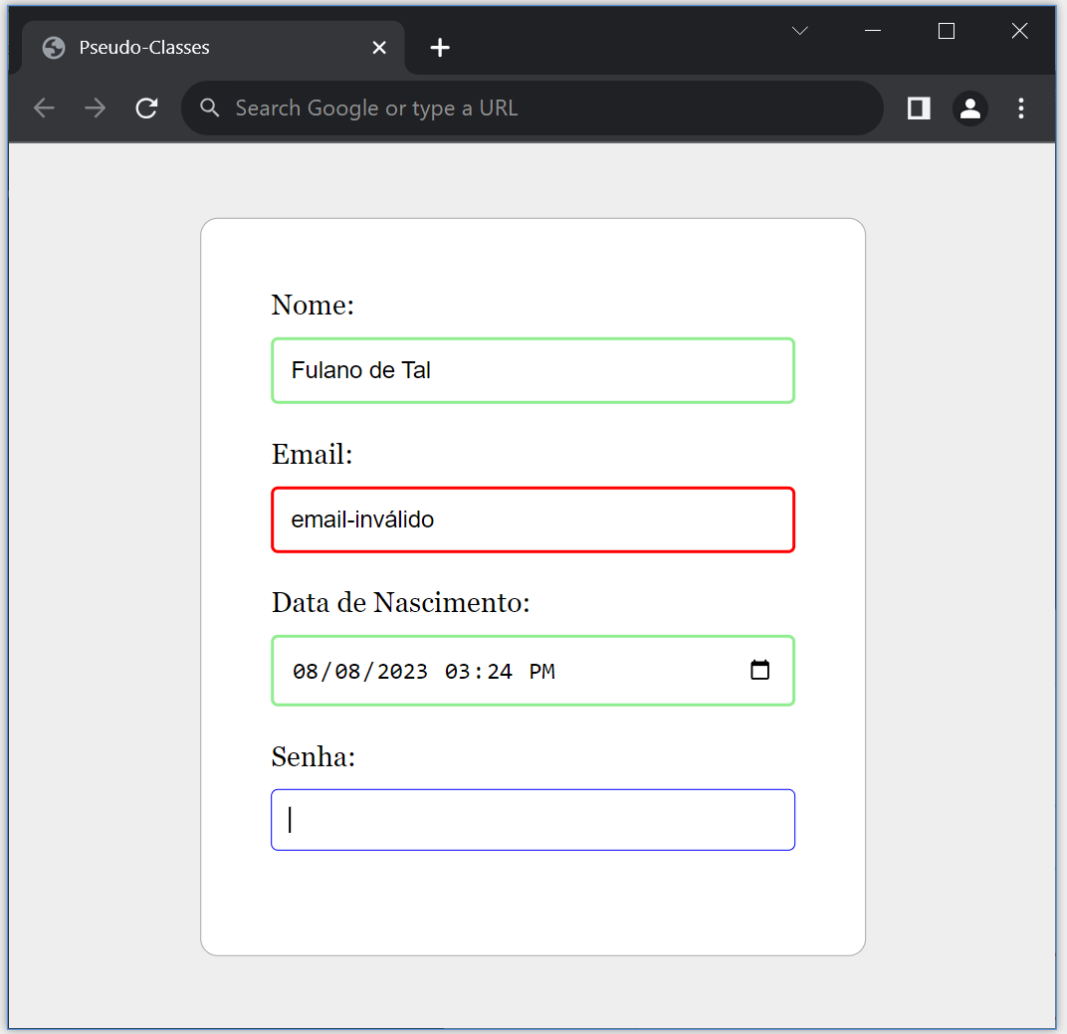


Pseudo-Classes Comuns em Campos de Formulário

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

Usando Pseudo-Classes para Alterar Estilo de Campos

```css
input:valid {
  border: 2px solid lightgreen;
}

input:invalid {
  border: 2px solid red;
}

input:focus {
  border: 1px solid blue;
  outline: none;
}
```



Outros Exemplos de Pseudo-Classes

Pseudo-Classe	Descrição	Exemplos
:first-child	Permite estilizar o elemento que é o **primeiro** filho do elemento pai	`li:first-child { color: blue; }` altera o 1º item de cada lista
:last-child	Permite estilizar o elemento que é o **último** filho do elemento pai	`li:last-child { color: blue; }` altera o último item de cada lista

:nth-child	Permite estilizar o elemento que é o **n-ésimo** filho do elemento pai	`li:nth-child(2) { … }`
		altera o 2º item de cada lista
		`tr:nth-child(odd) { … }`
		altera as linhas ímpares da tabela
		`tr:nth-child(even) { … }`
		altera as linhas pares da tabela
		`tr:nth-child(3n) { … }`
		altera as linhas de nº 3, 6, 9, 12 etc.

:first-of-type	Permite alterar a primeira ocorrência de um elemento dentro de seu container	`p:first-of-type {color: red;}` altera a cor do 1º parágrafo do container, mesmo que tal parágrafo não seja o 1º filho do container (veja ex. a seguir)

Há muitas outras pseudo-classes: https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes

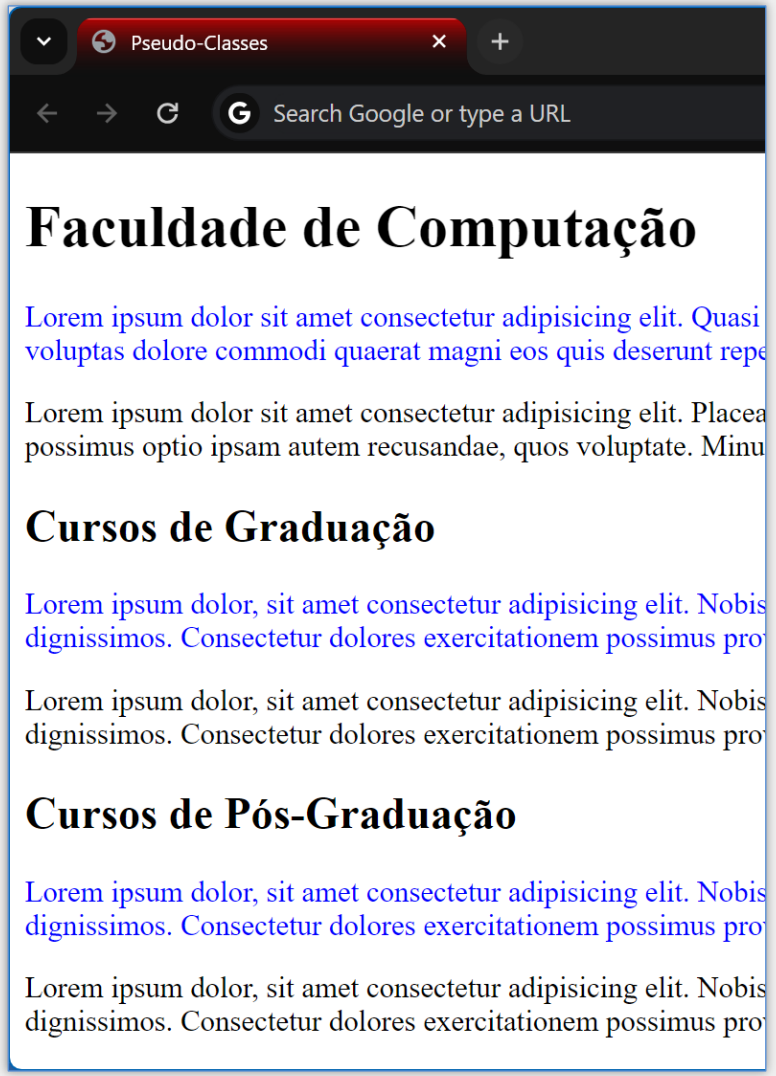
Conteúdo Complementar

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



Conteúdo Complementar

Função de Pseudo-Classe :has()

- A função de pseudo-classe :has() permite selecionar elementos que tenham outros elementos associados conforme 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 têm um **descendente** img, alterando a cor de fundo desses div's para cinza.

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

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

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

Seleciona os títulos h1 que têm 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 do texto dos elementos section, article e div que tenham uma imagem como descendente

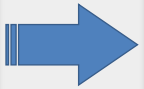
```
:is(section, article, div):has(img) {
  color: gray;
}
```

Conteúdo Complementar

Função de Pseudo-Classe :is()

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;
}
```

Conteúdo Complementar

Pseudo-Elementos

- Permite selecionar 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

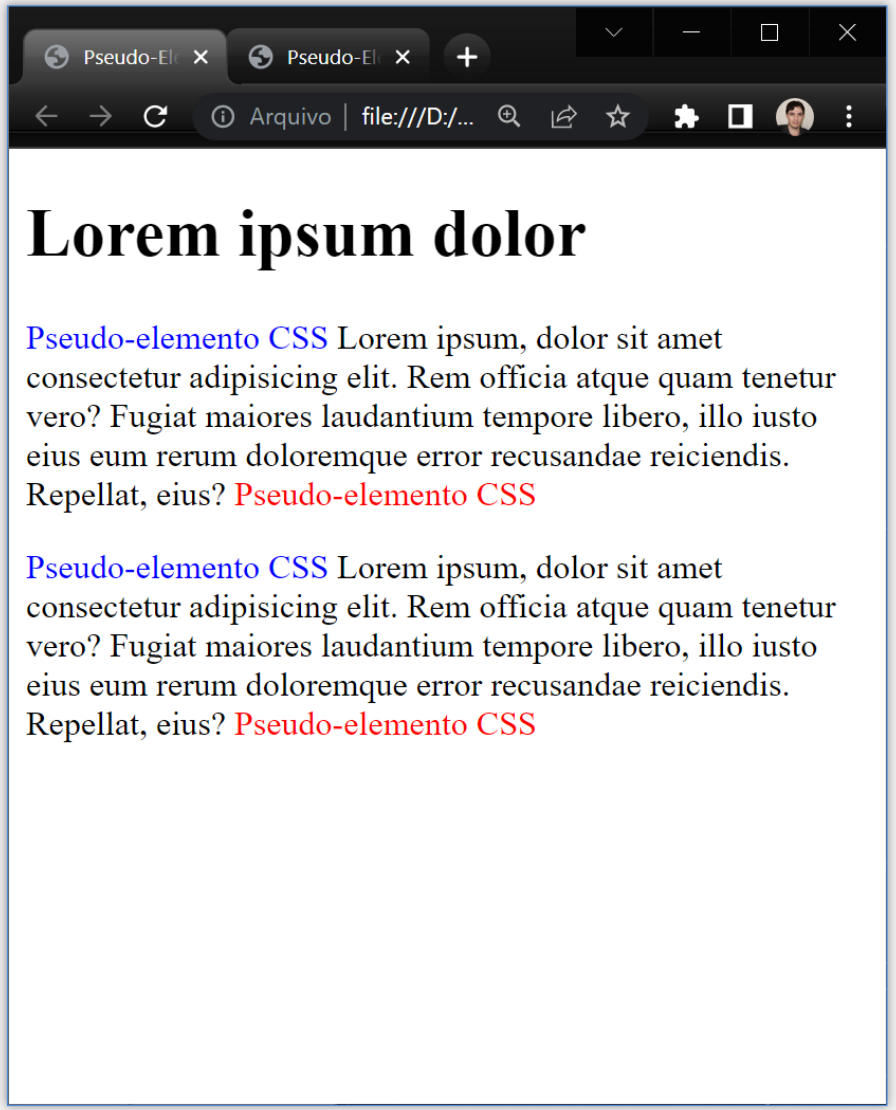
```
p::before {
  content: 'exemplo before';
}
```

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 <img> ou <input>

Pseudo-Elementos - Exemplo

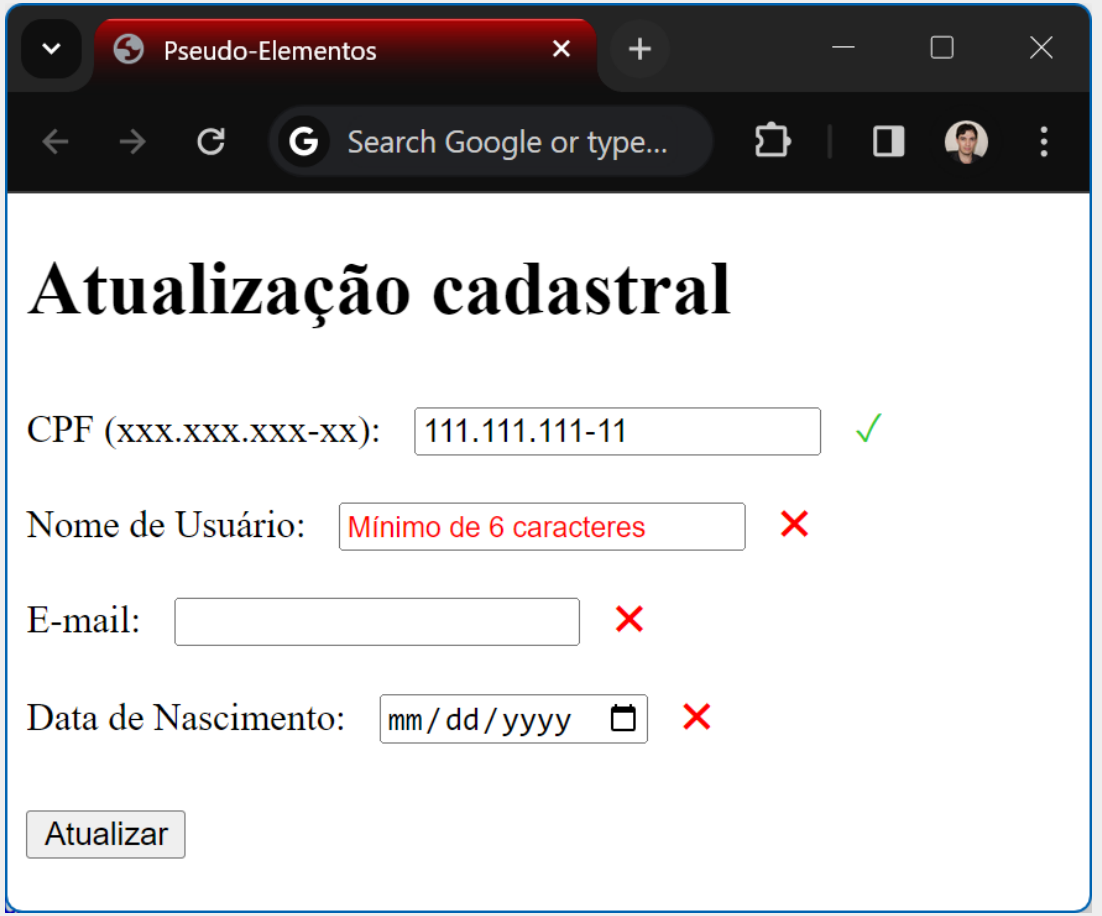
```
  <style>
    p::after {
      content: ' Pseudo-elemento CSS ';
      color: red;
    }

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



Pseudo-Elementos - Exemplo

```css
input:invalid + span::after {
  content: '✗';
  color: red;
}
input:valid + span::after {
  content: '✓';
  color: limegreen;
}
```

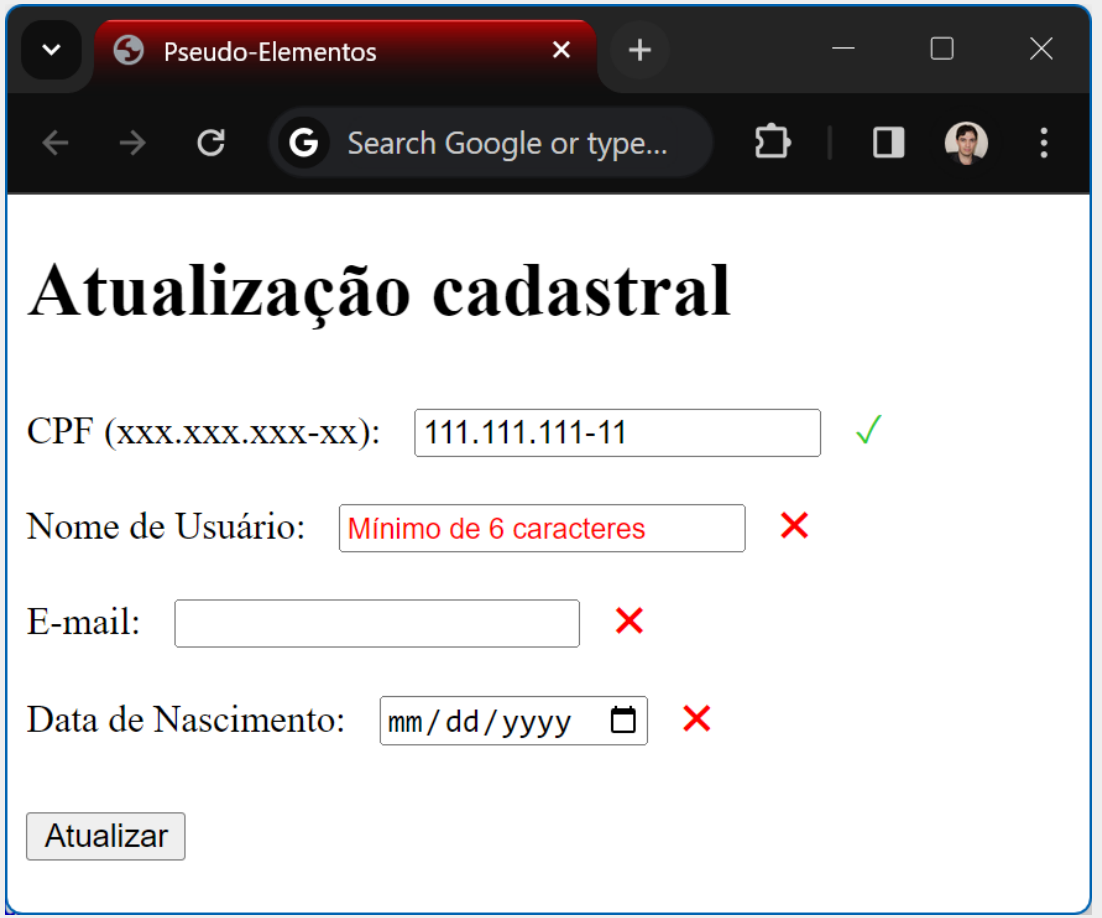
```html
<div>
  <label for="cpf">CPF (xxx.xxx.xxx-xx):</label>
  <input type="text" id="cpf" name="cpf" required
    pattern="\d{3}\.\d{3}\.\d{3}-\d{2}">
  <span></span> <!-- span para exibição de '✗' ou '✓' -->
</div>
<div>
  <label for="user">Nome de Usuário:</label>
  <input type="text" id="nome" name="nome" minlength="6"
    placeholder="Mínimo de 6 caracteres" required>
  <span></span> <!-- span para exibição de '✗' ou '✓' -->
</div>
```



Pseudo-Elementos - Exemplo

```css
div:has(>input:invalid)::after {
  content: '✗';
  color: red;
}
div:has(>input:valid)::after {
  content: '✓';
  color: limegreen;
}
```

```html
<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}">
</div>
<div>
  <label for="user">Nome de Usuário:</label>
  <input type="text" id="nome" name="nome" minlength="6"
    placeholder="Mínimo de 6 caracteres" required>
</div>
```



Solução alternativa utilizando o próprio que agrupa campo e label como container para o símbolo de validação

Observações sobre Pseudo-Elementos

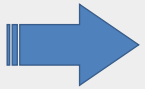
- A notação com 4 pontos (::) foi introduzida em atualizações da CSS
- :: é a forma atualmente recomendada para pseudo-elementos, pois diferencia pseudo-elementos de pseudo-classes, uma vez que há diferença semântica significativa entre eles

Aninhamento de Regras no CSS Nativo (CSS Nesting)

- Permite definir regras dentro de regras
- Deve ser utilizado com cautela, pois é um recurso ainda em desenvolvimento
  - Suportado apenas por versões mais recentes dos principais navegadores
  - Ainda não é reconhecido por ferramentas de validação como jigsaw.w3.org

```css
header {
  color: #555;
  text-align: center;
  width: 100%;
}

header h1 {
  margin: 1rem 0;
}
```

CSS sem aninhamento de regras



```css
header {
  color: #555;
  text-align: center;
  width: 100%;

  & h1 {
    margin: 1rem 0;
  }
}
```

CSS utilizando aninhamento de regras

OBS: Antes de ter suporte nativo, o aninhamento de regras já era utilizado por desenvolvedores utilizando ferramentas pré-processadoras de CSS (SCSS/Sass).

Conteúdo Complementar

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

Principais Propriedades de Ajuste de Fonte

Propriedade	Descrição	Exemplo
font-family	define a fonte em si	font-family: Verdana, Arial, sans-serif
font-style	define o estilo da fonte	font-style: italic
font-size	define o tamanho da fonte	font-size: 20px
font-weight	define a espessura da letra	font-weight: bold
line-height	define o espaçamento entre linhas	line-height: 1.5

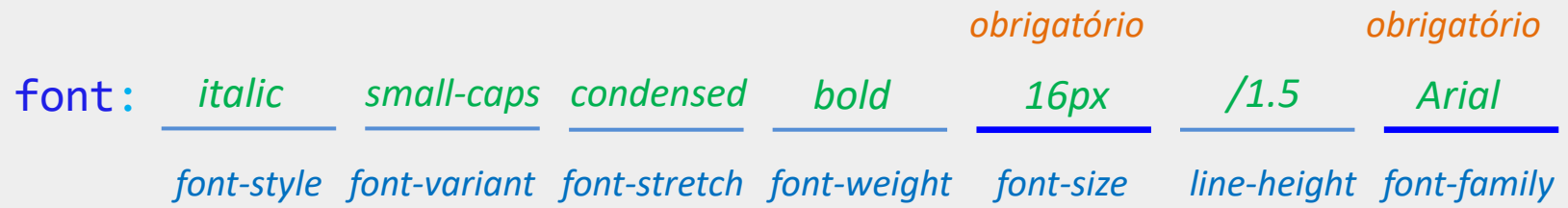
Repare que a propriedade **font-family** permite a indicação de uma lista de nomes de fontes. O navegador utilizará a próxima fonte dessa lista caso eventualmente a fonte anterior não seja encontrada no computador do usuário. Recomenda-se encerrar essa lista de fontes com o nome de uma família genérica 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`

Conteúdo Complementar

Propriedade Abreviada **font** – Sintaxe



- 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;

Conteúdo Complementar

Propriedades de Ajuste de Texto

Propriedade	Descrição	Exemplos
text-align	alinhamento horizontal do texto	text-align: left text-align: justify
vertical-align	alinhamento vertical do texto	vertical-align: top vertical-align: middle
text-decoration	decoração adicional	text-decoration: none text-decoration: underline
text-transform	controle de maiúsculas e minúsculas	text-transform: uppercase text-transform: lowercase
color	define a cor do texto	color: green

Principais Formas de Ajustes de Cor

- Pelo nome da cor
  - color: blue, color: darkblue, color: lightblue, etc.
- Pelo valor RGB em Decimal (red, green, blue)
  - color: rgb(0, 120, 255)
- Pelo valor RGB em Hexadecimal
  - color: #FF0000

Unidades de Tamanho da CSS

Unidades de Tamanho Absoluto

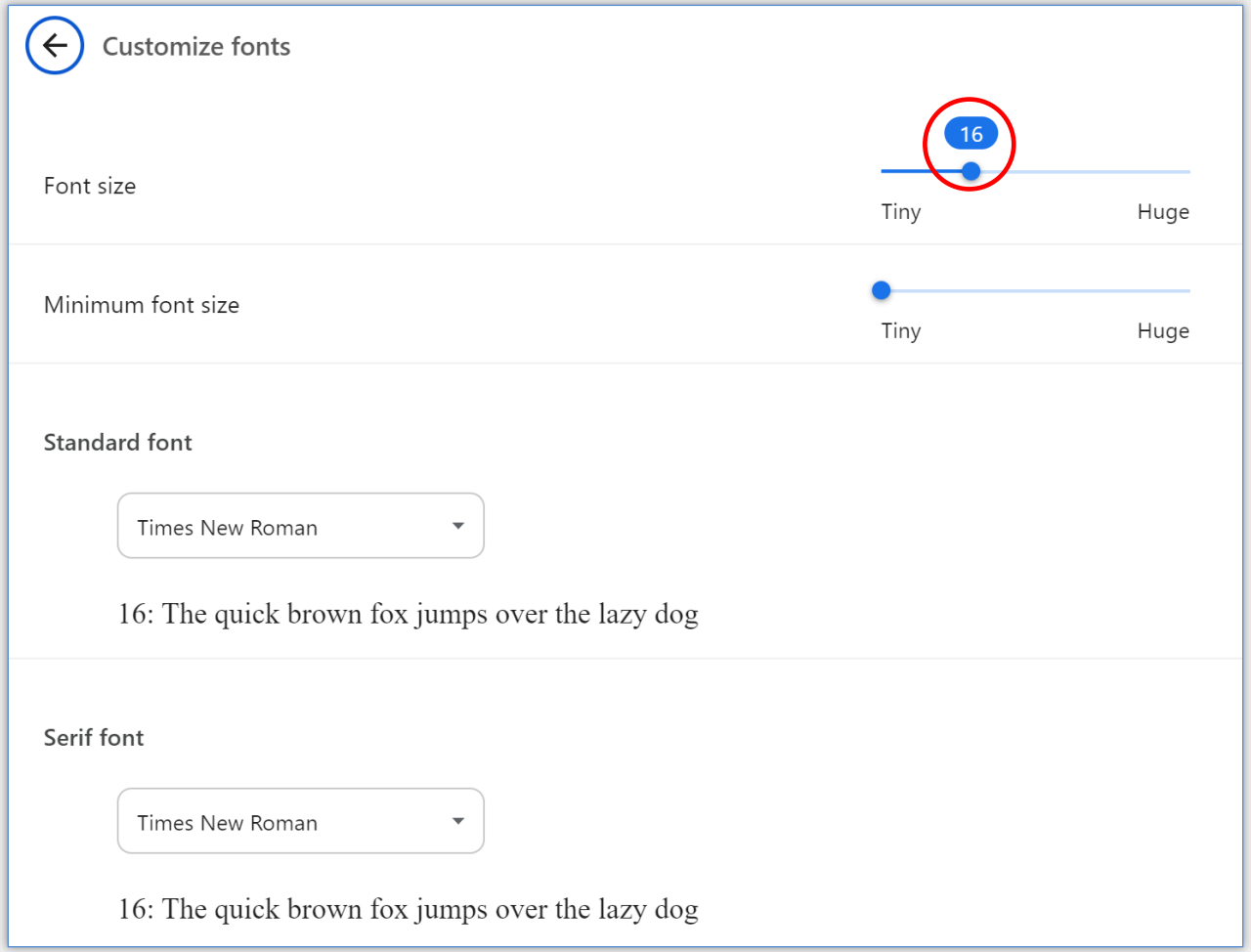
- A unidade **px** (pixels) é a unidade de **tamanho absoluto** mais comum
- Tamanhos absolutos não dependem de tamanhos definidos no elemento pai
- Além disso, ao definir um tamanho utilizando **px**, o tamanho não será afetado por eventual mudança no tamanho de fonte feita pelo usuário nas configurações 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

% – em geral, relativo ao elemento pai

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

Exemplo de Tamanho em

```css
.medio {
   font-size: 16px;
}

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

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

Conteúdo Complementar

Exemplo de Tamanho rem

```css
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

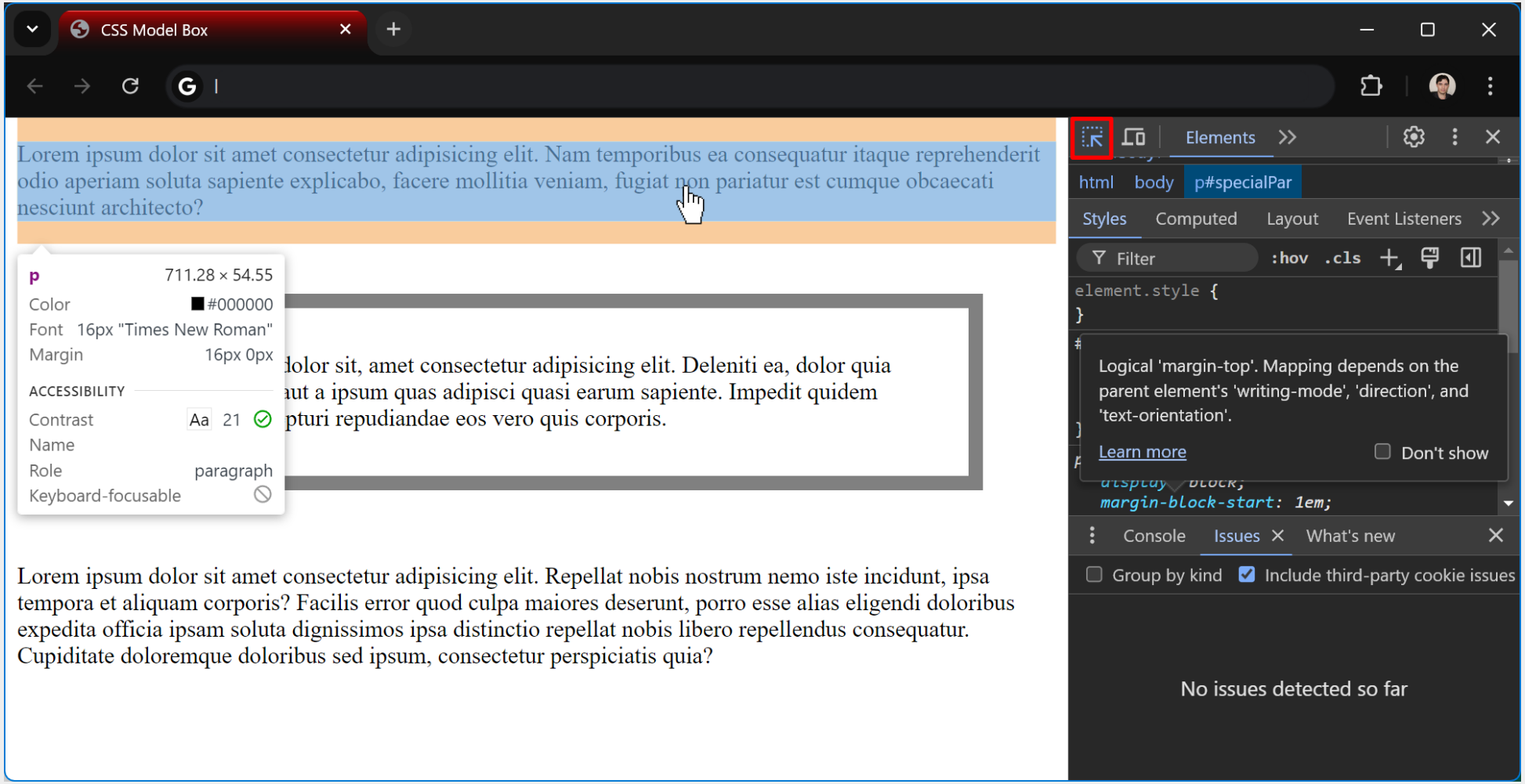
Considere que o tamanho de **fonte padrão** definido no **navegador** seja de 18px

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

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

Neste exemplo, o tamanho de fonte padrão não é alterado no elemento raiz (<html>). Portanto, 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>
```

Verificando a Fonte de um Elemento no Navegador



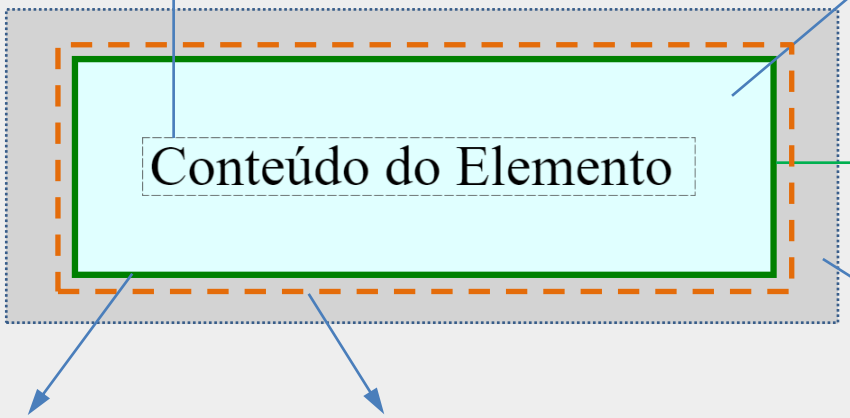
No navegador, é possível obter informações detalhadas sobre a fonte utilizada por um elemento passando o ponteiro do mouse sobre ele. No Google Chrome, tecle F12 e clique no botão de seleção na extremidade superior esquerda do painel direito para ativar a funcionalidade. Em seguida, basta passar o ponteiro do mouse sobre os elementos.

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



Content box
- Região de conteúdo do elemento

Padding do elemento
- Envolve o conteúdo
- Afetado pela cor de fundo
- Dentro da borda

Borda do elemento
- Envolve o *padding*

Margem do elemento
- Externo à borda
- Transparente

Border box
- Região incluindo borda, padding e conteúdo

Outline do elemento
- Externo à borda
- Não toma espaço próprio

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

Ajustes de Margens e Paddings

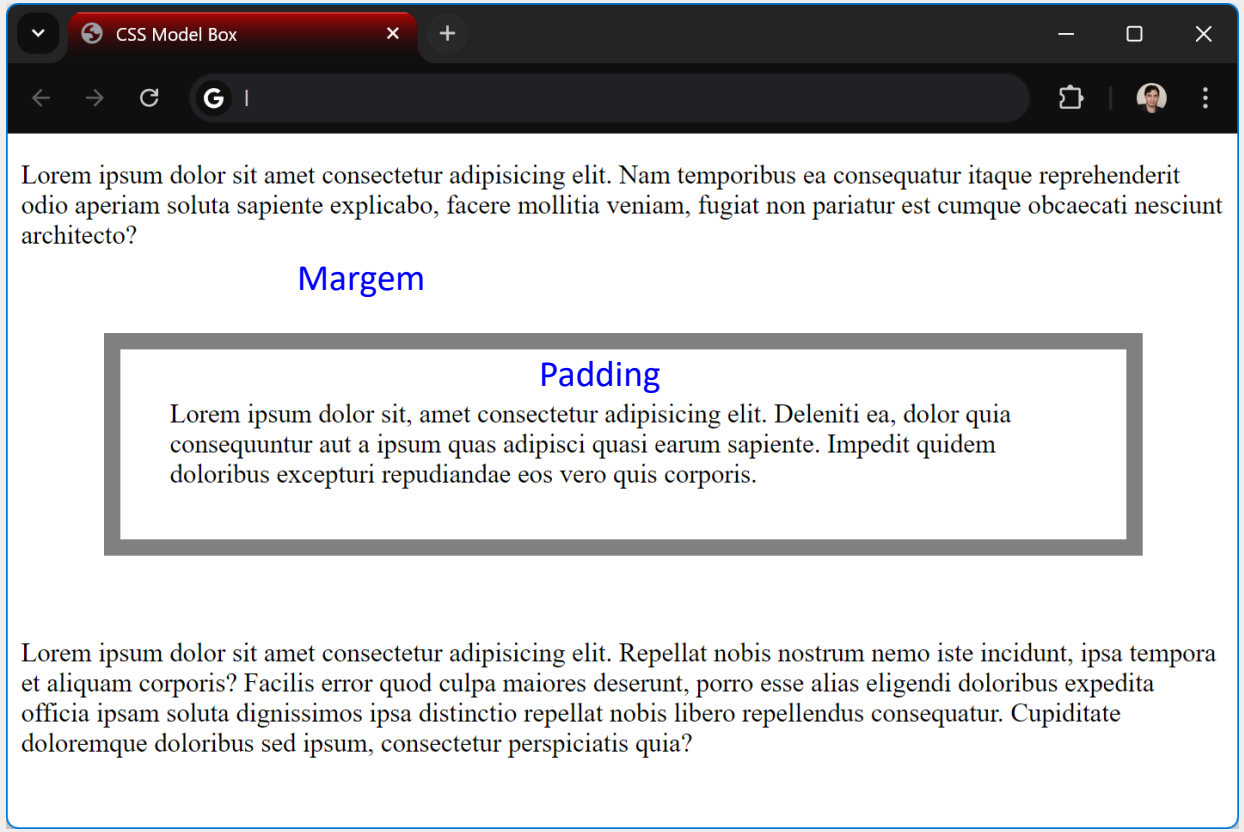
- É possível fornecer até 4 valores para a propriedade margin:
  - margin: 20px; (ajusta todas as margens em 20px)
  - margin: 20px 50px 80px 100px; (superior, direita, inferior e esquerda)
  - margin: 20px 40px 20px; (superior, laterais, inferior)
  - margin: 20px 40px; (superior/inferior e laterais)
- 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

CSS Box Model – Margens, Paddings e Bordas

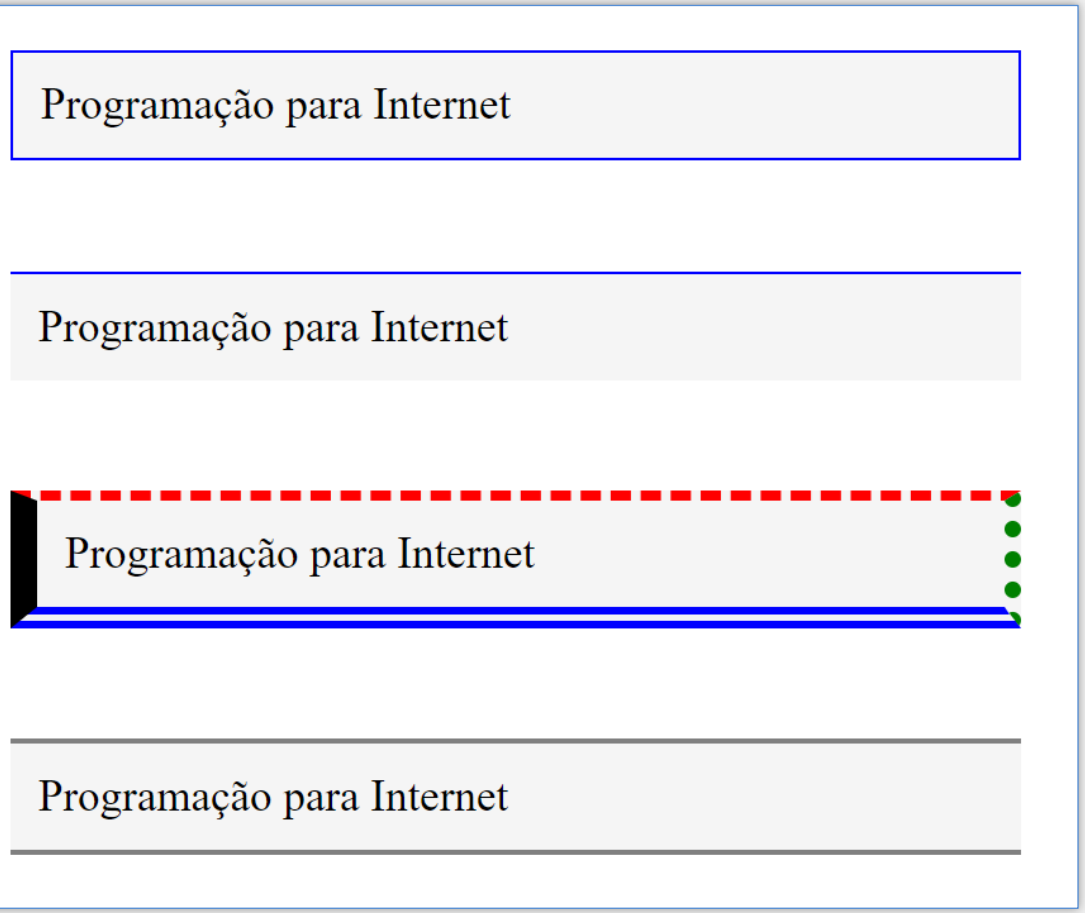
```
  <style>
    #specialPar {
      margin: 50px;
      padding: 30px;
      border: 10px solid gray;
    }
  </style>
</head>
<body>
  <p>Lorem ipsum dolor sit amet consectetur...</p>
  <p id="specialPar">Lorem ipsum dolor sit...</p>
  <p>Lorem ipsum dolor sit amet consectetur...</p>
</body>
```



Ajustes de Borda – Exemplos Adicionais

```html
<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>
```

```css
#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;
}
```

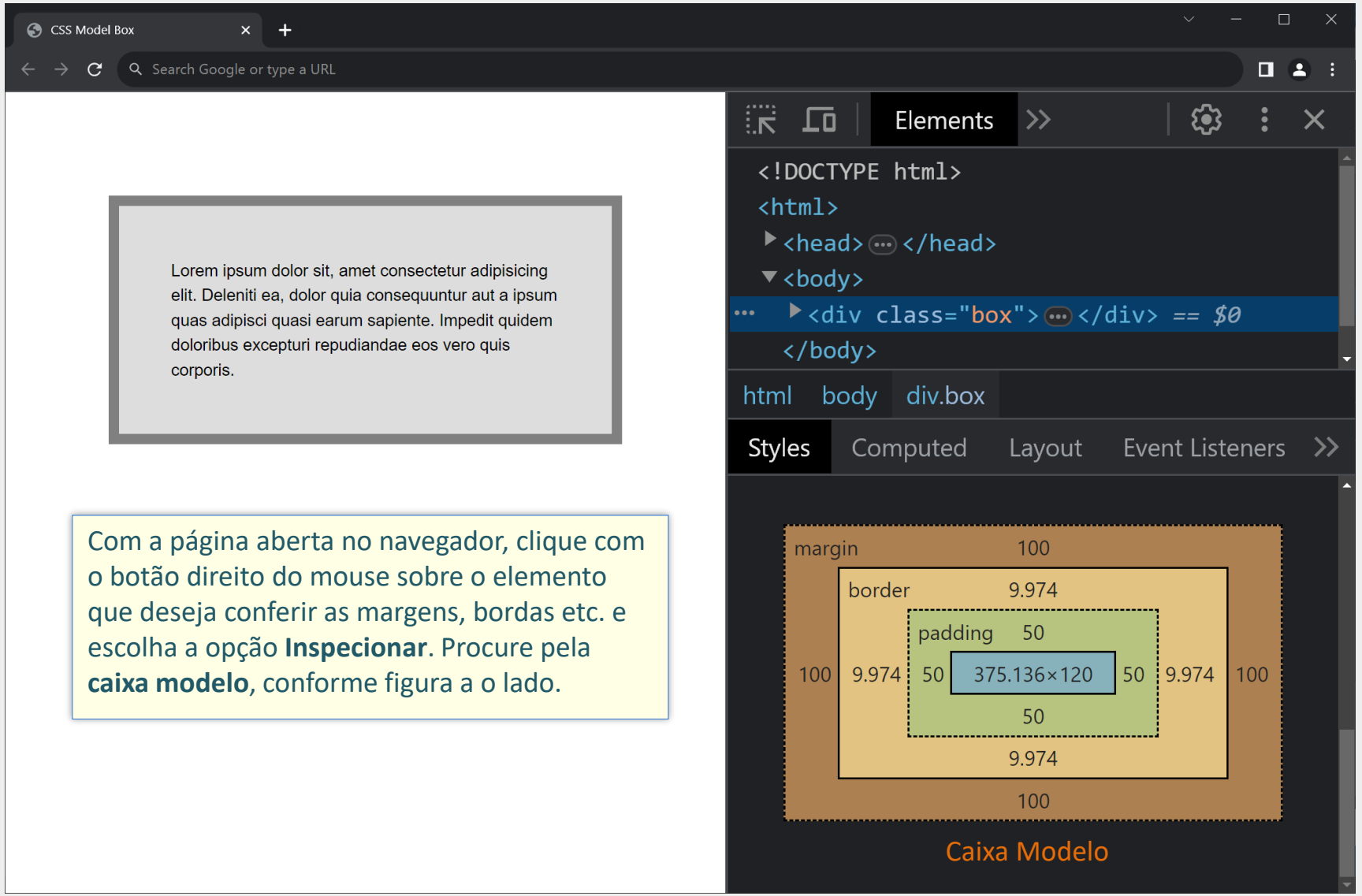


Conteúdo Complementar

CSS Box Model – Visualização no Navegador

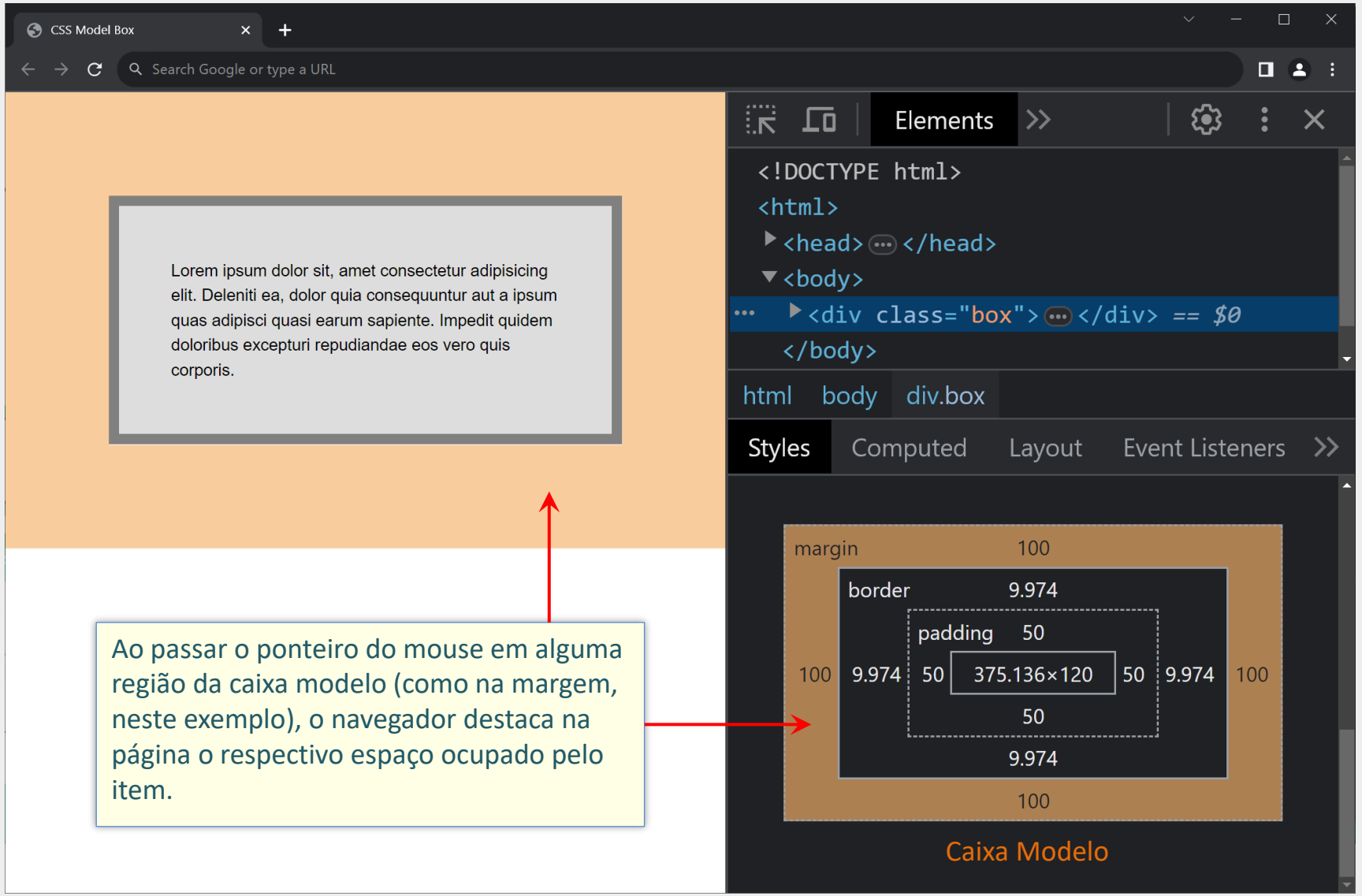
- Durante o desenvolvimento do layout da página, pode ser necessário verificar os tamanhos da região de conteúdo, margens, paddings e bordas
- Esses tamanhos podem ser conferidos no próprio navegador
- No Google Chrome, basta clicar sobre o elemento com o botão direito do mouse e escolher **Inspecionar**
- 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.

CSS Box Model – Visualização no Navegador



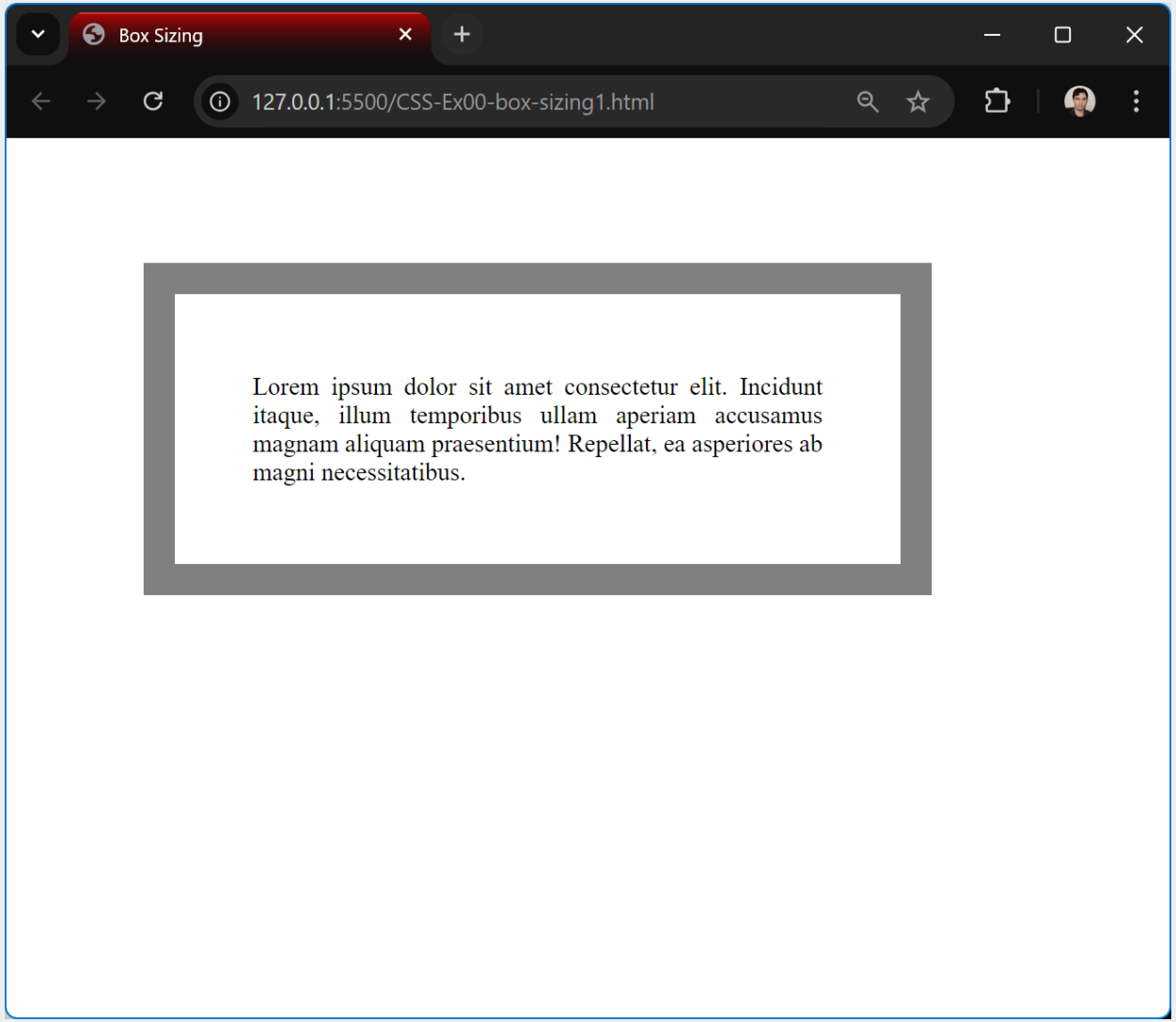
Ao passar o ponteiro do mouse em alguma região da caixa modelo (como na margem, neste exemplo), o navegador destaca na página o respectivo espaço ocupado pelo item.

Ajustes de Largura e Altura

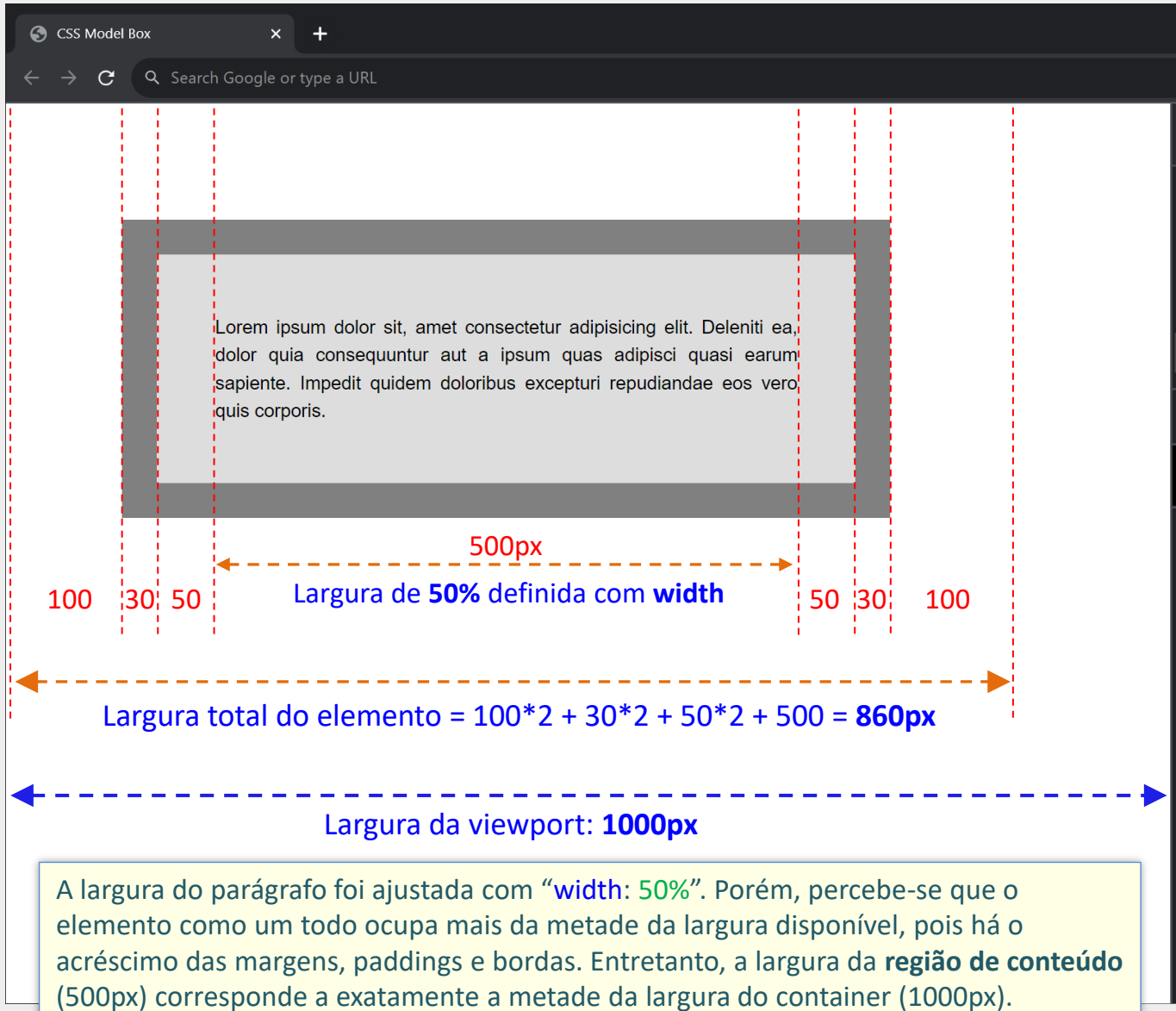
- Para definir a largura e a altura de um elemento pode-se utilizar as propriedades width e height
- Na maioria dos casos, width e height definem tamanhos para a região de conteúdo do elemento (abordagem padrão)
  - Largura total = larg. conteúdo + margens + bordas + paddings
  - Esse comportamento pode 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

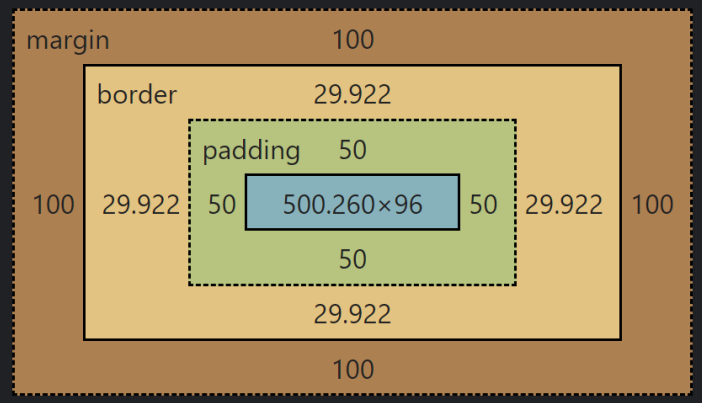
```
  <style>
    p {
      width: 50%;
      text-align: justify;
      margin: 100px;
      padding: 50px;
      border: 30px solid gray;
    }
  </style>
</head>
<body>
  <p>Lorem ipsum dolor sit amet consectetur
    elit. Incidunt itaque, illum temporibus
    ullam aperiam accusamus magnam aliquam
    praesentium! Repellat, ea asperiores
    ab magni necessitatibus.</p>
</body>
```



Largura Total do Elemento – Cálculo Padrão




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



A largura do parágrafo foi ajustada com "width: 50%". Porém, percebe-se que o elemento como um todo ocupa 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

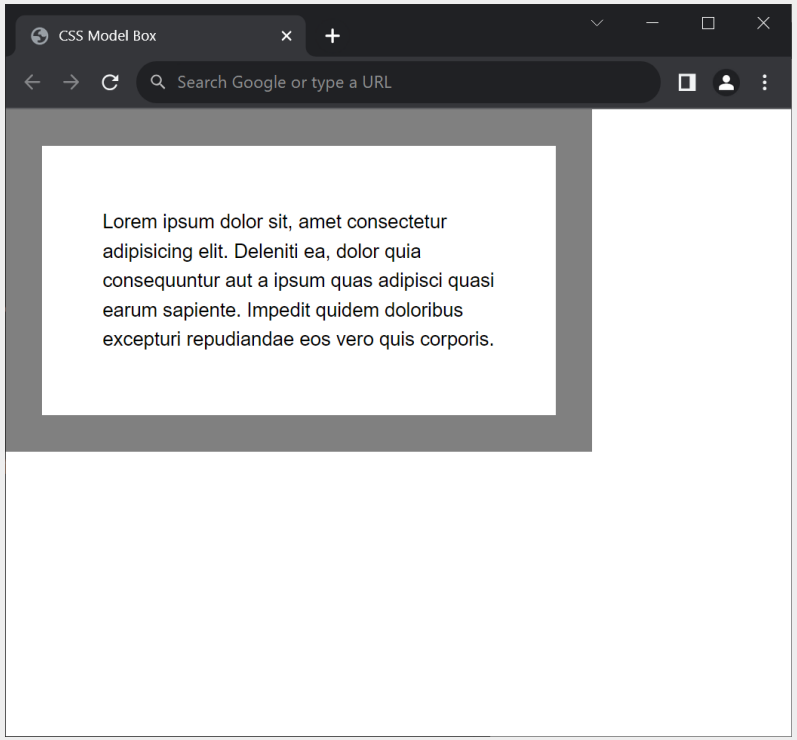
- Altera o modo em que a largura e altura do elemento é calculada
- box-sizing: content-box
  - Valor padrão para a maioria dos elementos (conforme exemplo anterior)
  - Ao definir width, estamos definindo a largura da região de conteúdo (content box)
  - A largura total do elemento será: width + bordas + 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

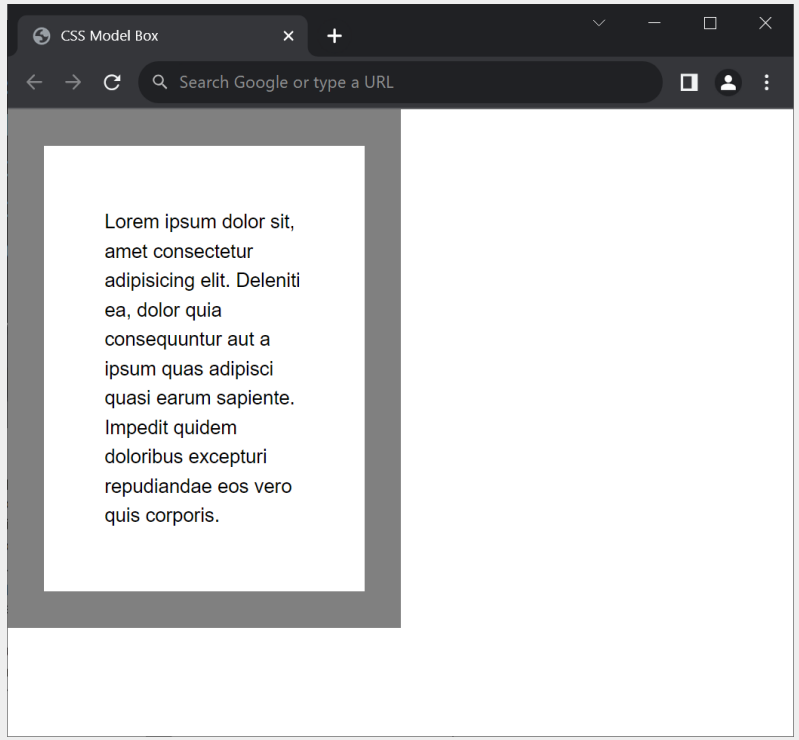
Com box-sizing: content-box, "width: 50%" faz com que a região de conteúdo (texto) ocupe 50% da largura do container, mas a largura total do elemento (incluindo paddings, bordas e margens) ultrapassa a metade do container.

Este é o mecanismo padrão utilizado pela maioria dos elementos.

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




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



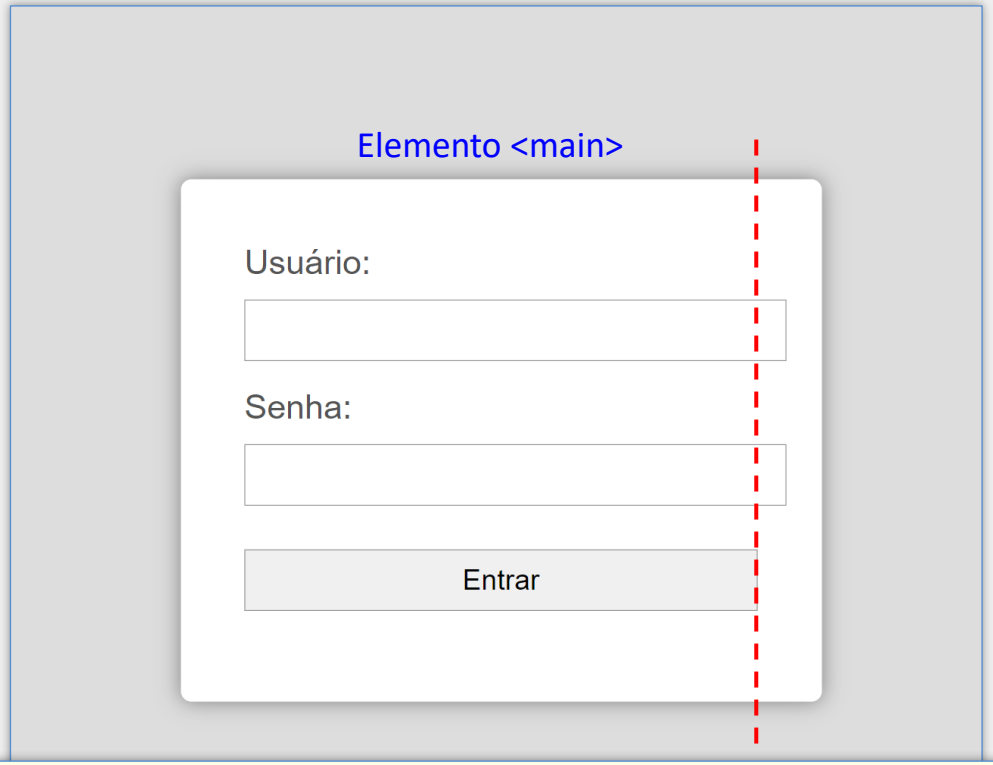
Alterando a propr. box-sizing para o valor border-box, "width: 50%" faz com que toda a caixa da borda ocupe 50% da largura do container

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

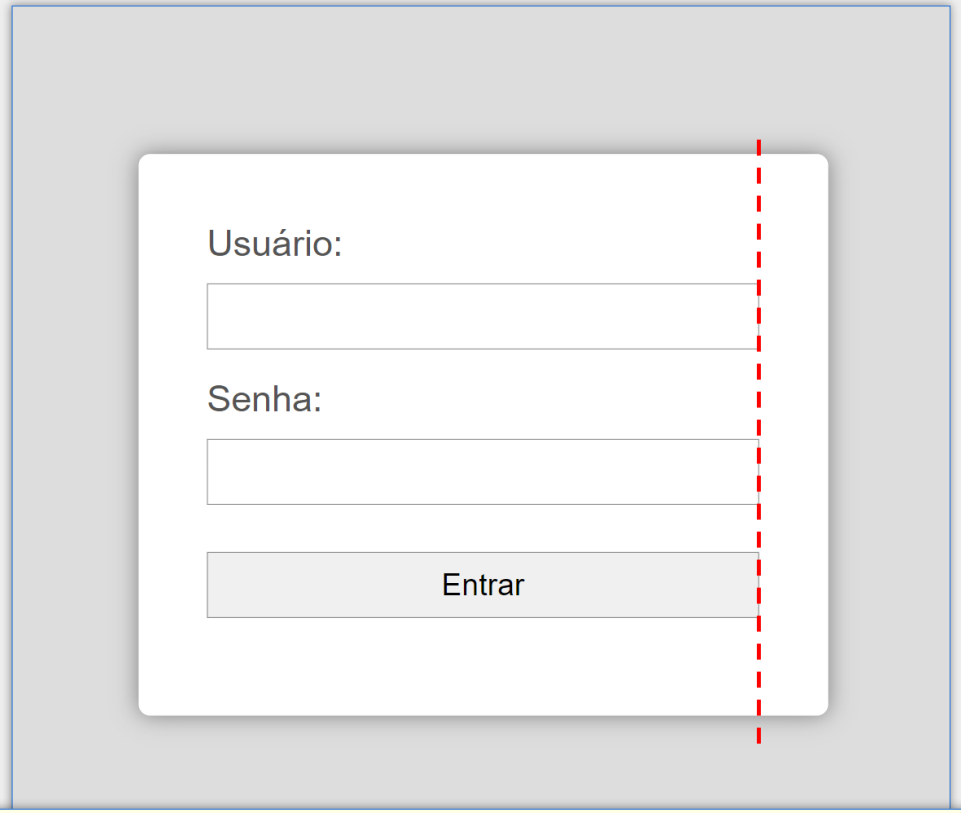
```css
main {
  width: 300px;
  padding: 30px;
  background-color: white;
}

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 campos textuais utilizam, por padrão, box-sizing: content-box, enquanto o elemento <button> utiliza box-sizing: border-box.

Propriedade box-sizing - Exemplo

```css
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.

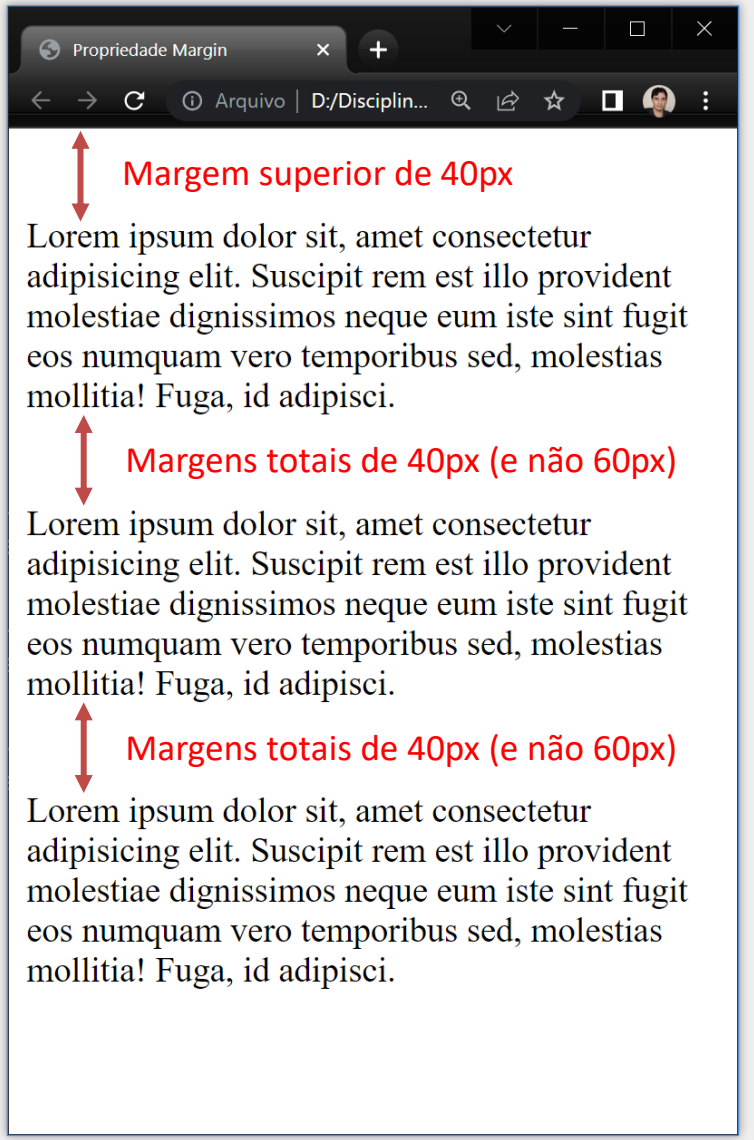
Sobreposição de Margens

- 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)

Conteúdo Complementar

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>Lorem ipsum dolor sit, amet c
    molestiae dignissimos neque eu
    mollitia! Fuga, id adipisci.</
  <p>Lorem ipsum dolor sit, amet c
    molestiae dignissimos neque eu
    mollitia! Fuga, id adipisci.</
  <!-- vertical margin collapsing
</body>
```



Conteúdo Complementar

Outros Ajustes de Tamanho

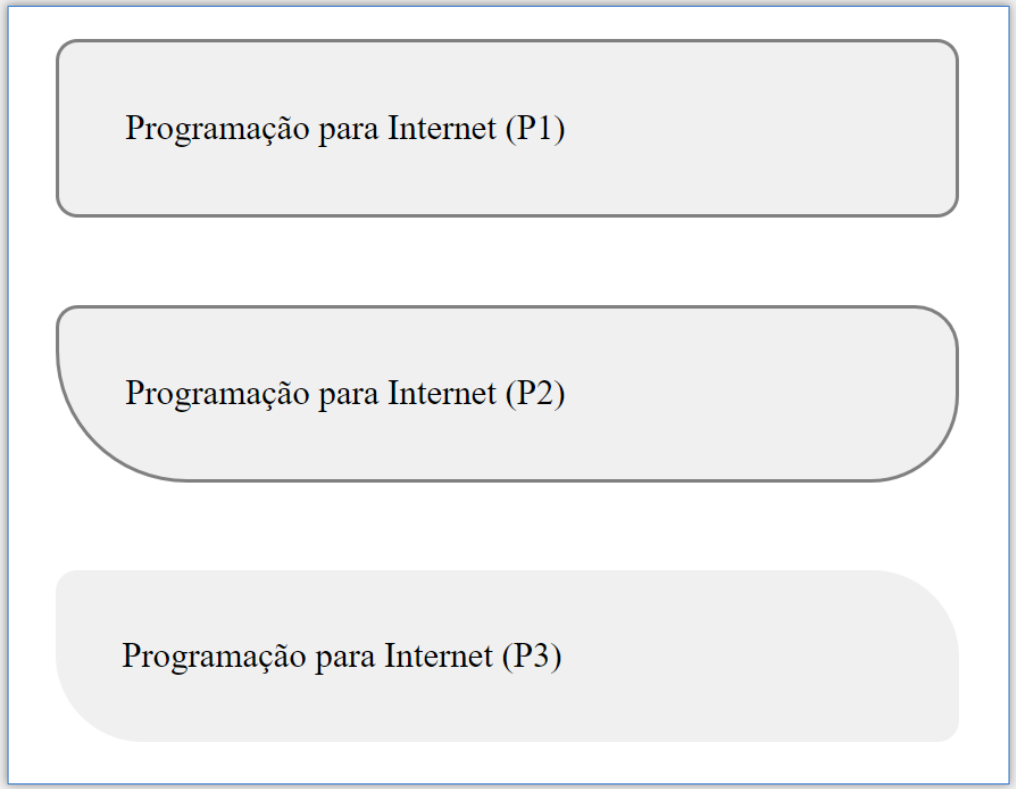
- Além das propriedades width e height, há também propriedades para definir a largura mínima, a largura máxima, a altura mínima e a altura máxima de um elemento:
  - min-width, max-width,
  - min-height, max-height
- Tais propriedades são especialmente importantes em layouts responsivos
- Outros valores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

Conteúdo Complementar

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

```css
#p1 {
  border-radius: 10px;
}

#p2 {
  border-radius: 10px 20px 40px 60px
}

#p3 {
  border-radius: 10px 40px;
  border: none;
}
```



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 (mantem proporção)

Ajustes de Fundo – Exemplo

```
<style>
  body {
    background-image: url("images/bg2.jpg");
    background-repeat: no-repeat;
    background-size: cover;
  }
</style>
```

Exibição e Posicionamento

Propriedade display

- Altera o modo de apresentação do elemento
- Um elemento de linha pode ser exibido como bloco e vice-versa
- Possibilita criar layouts flexíveis (próximo módulo)
- Também é possível ocultar um elemento, removendo-o do layout
- Alguns valores possíveis:
  - none, block, inline, inline-block
  - flex, grid, inline-flex, inline-grid

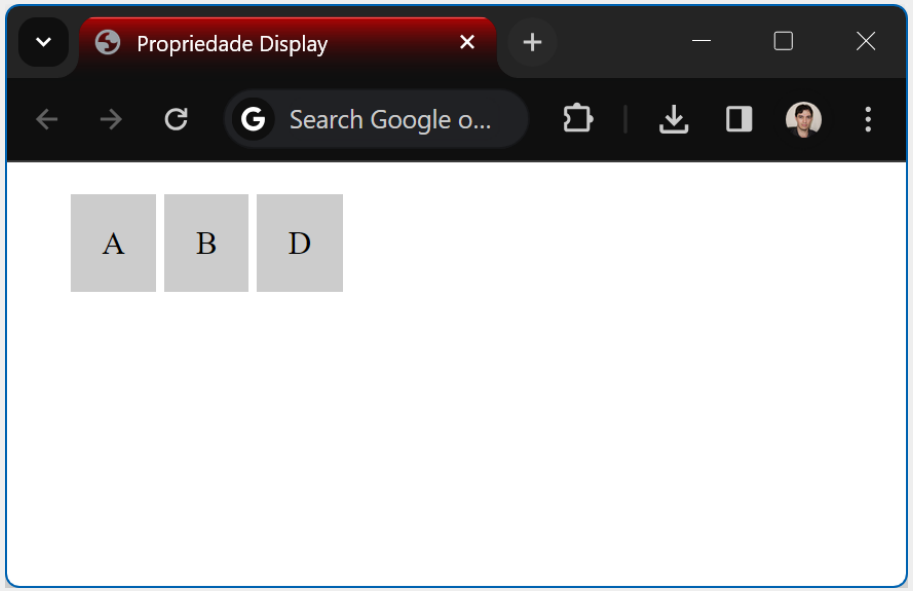
Propriedade display

display: none

- oculta completamente o elemento, removendo-o 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)
- 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:

Enviar

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

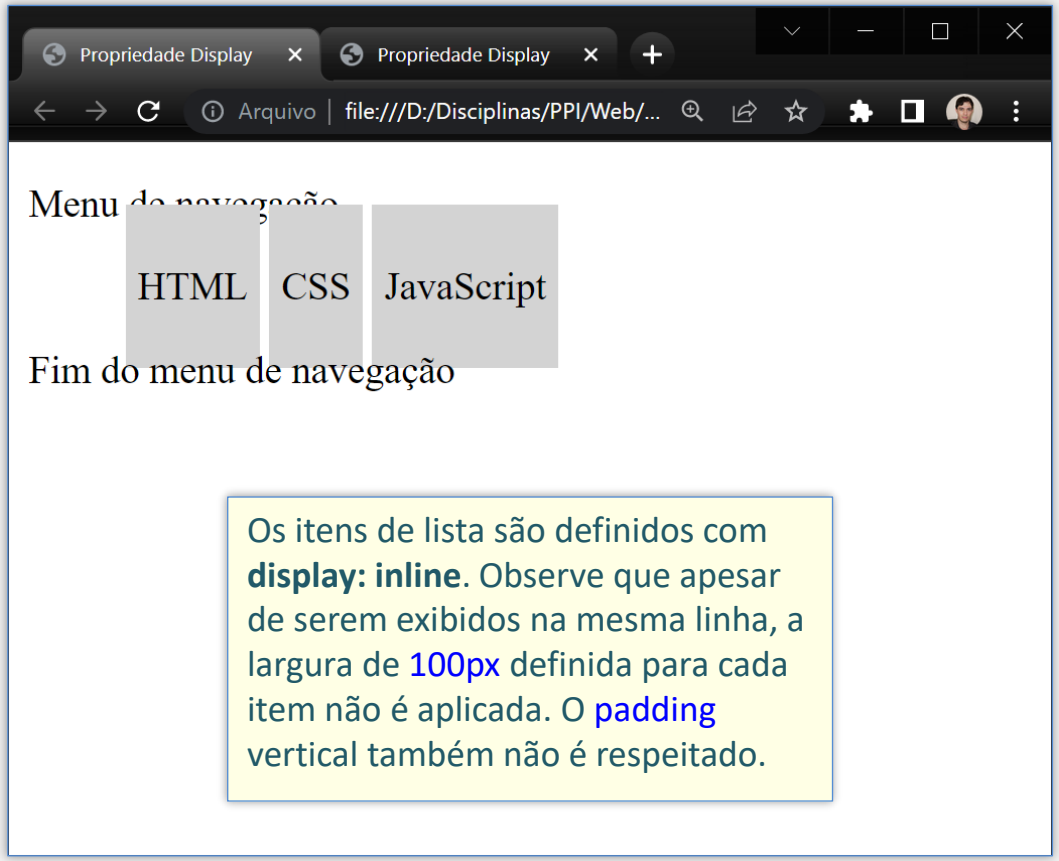
- faz com que o elemento seja exibido em nível de linha (como o)
- elementos de linha não começam obrigatoriamente com quebra de linha e só ocupam o espaço necessário para exibição de seu conteúdo
- width e height não terão efeito em alguns elementos
- margens e paddings verticais de alguns elementos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

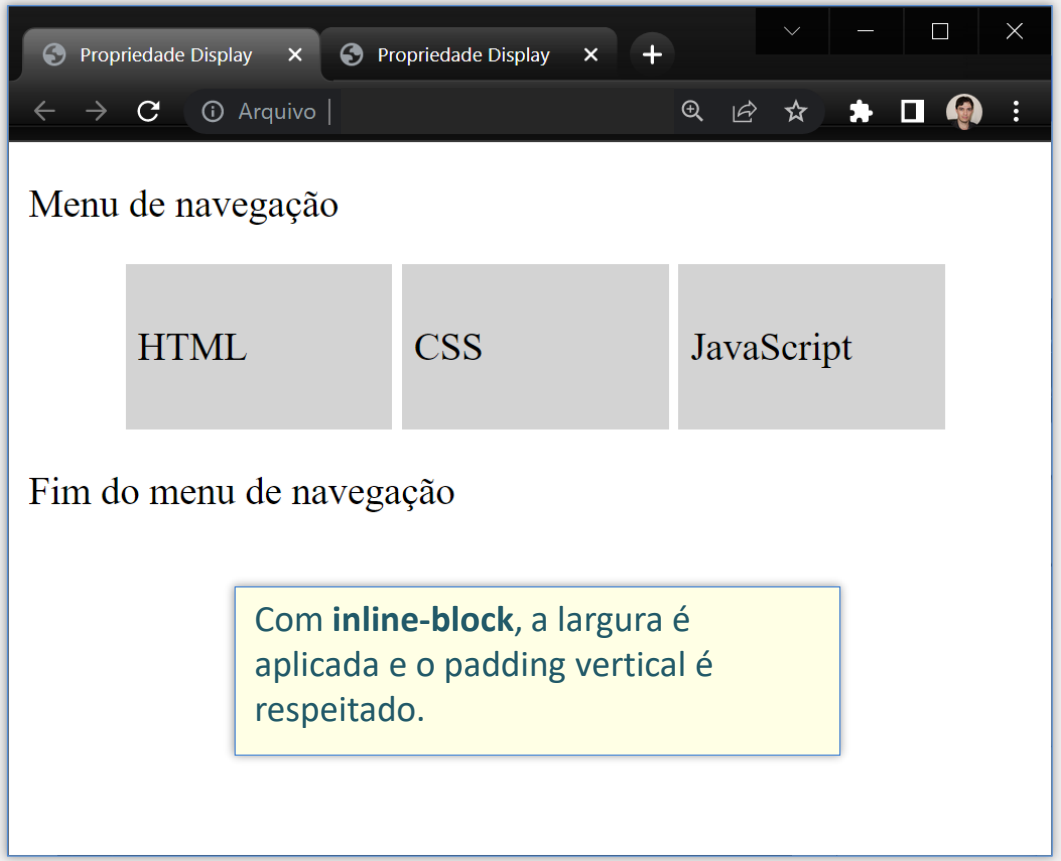
```
<style>
  nav li {
    background-color: lightgray;
    padding: 25px 5px;
    width: 100px;
    display: inline;
  }
</style>
</head>
<body>
  <nav>
    <p>Menu de navegação</p>
    <ul>
      <li>HTML</li>
      <li>CSS</li>
      <li>JavaScript</li>
    </ul>
    <p>Fim do menu de navegação</p>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



Os itens de lista são definidos com **display: inline**. Observe que apesar de serem exibidos na mesma linha, a largura de 100px definida para cada item não é aplicada. O padding vertical também não é respeitado.

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>
```

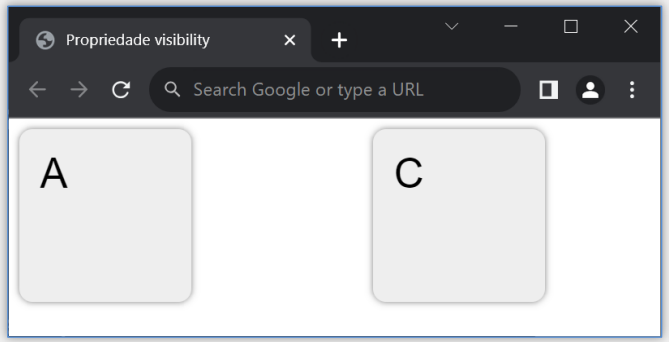


Propriedade visibility

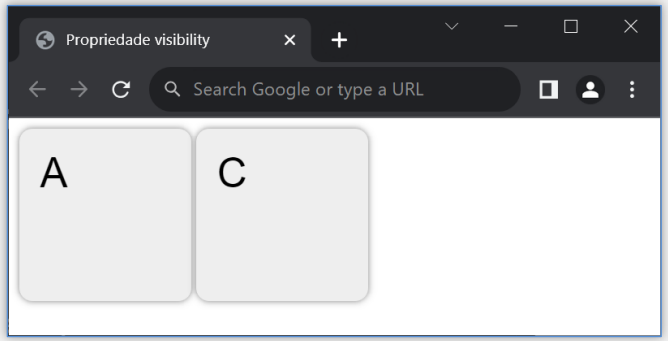
- Mostra ou oculta um elemento sem alterar o layout
- visibility: visible
  - elemento visível, aparecendo normalmente
- visibility: hidden
  - o elemento será ocultado, mas continuará ocupando espaço no layout

visibility: 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>
```



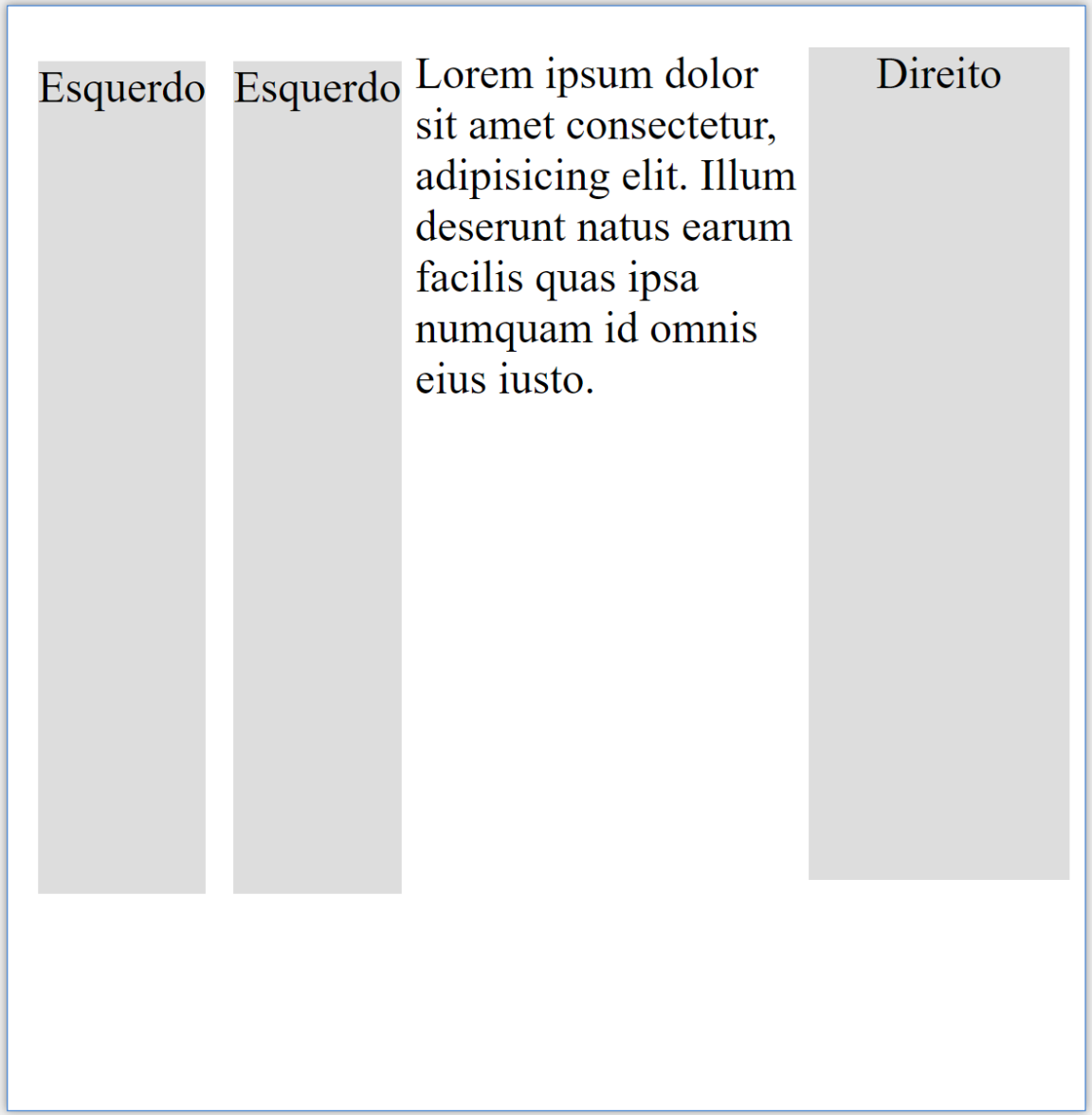
Propriedade float

- 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"

Conteúdo Complementar

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 consecte
      omnis eius iusto.</p>
  </main>
```



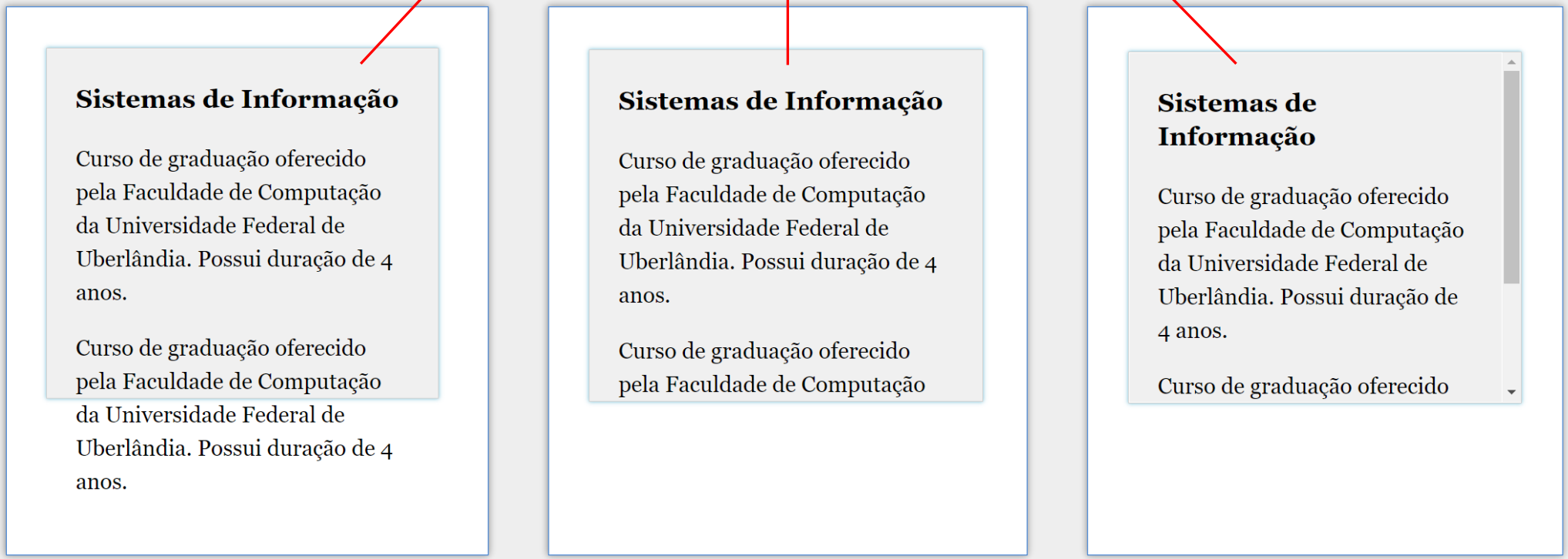
Conteúdo Complementar

Propriedade overflow

- Elementos com tamanhos fixados (width, height) podem ter o conteúdo extrapolando os limites da borda
- A propriedade overflow permite definir como esse conteúdo deve ser apresentado quando ele 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)



overflow: visible;

overflow: hidden;

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

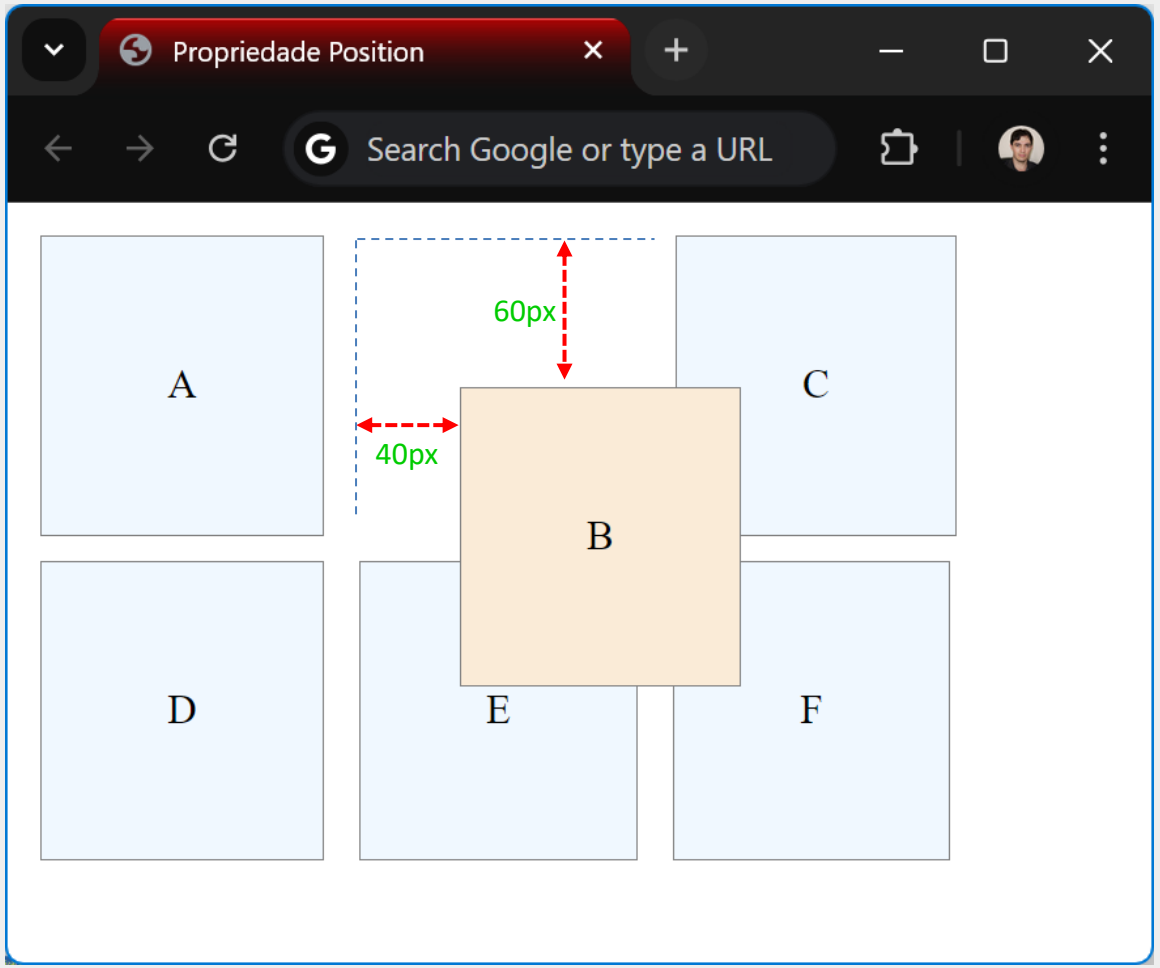
- Valor padrão
- Elemento posicionado de acordo com fluxo normal do documento

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: relative

```
    span {
      background-color: aliceblue;
      margin: 5px;
      padding: 50px;
      border: 1px solid grey;
      display: inline-block;
    }
    span:nth-child(2) {
      background-color: antiquewhite;
      position: relative;
      top: 60px;
      left: 40px;
    }
  </style>
</head>
<body>
  <span>A</span>
  <span>B</span>
  <span>C</span>
  <span>D</span>
  <span>E</span>
  <span>F</span>
</body>
```



Propriedade position - continuação

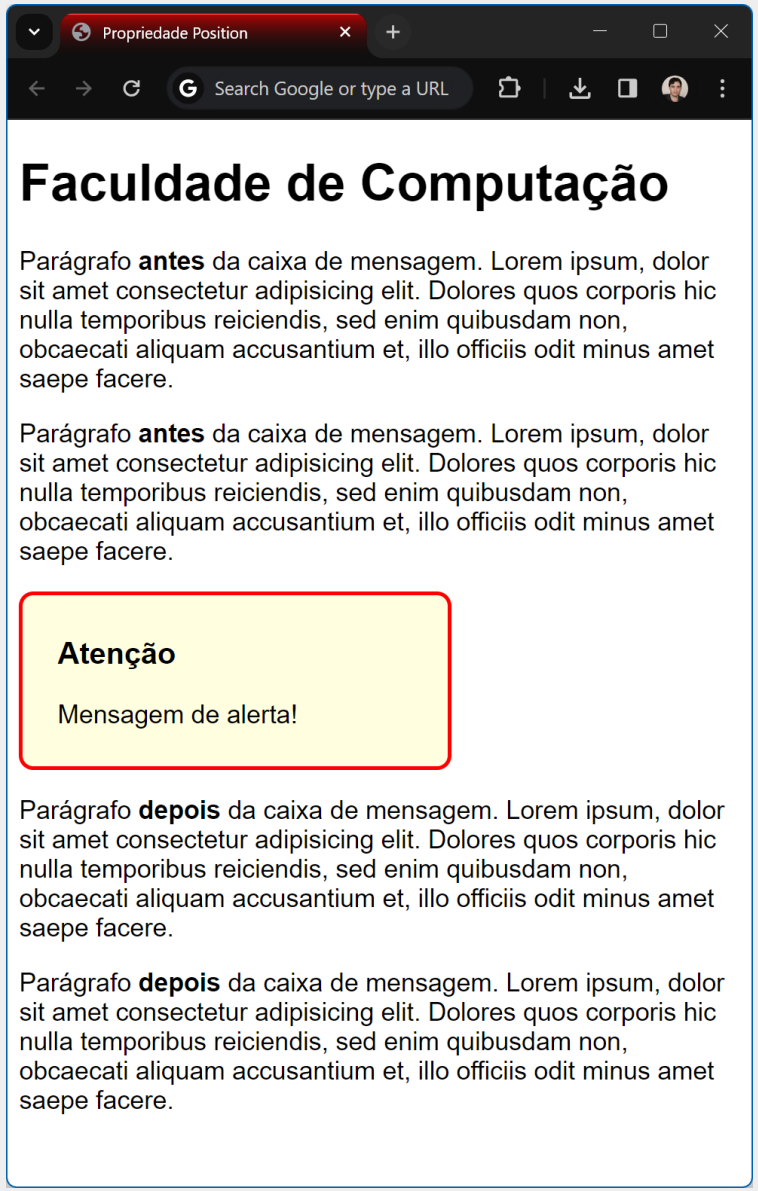
position: absolute

- 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

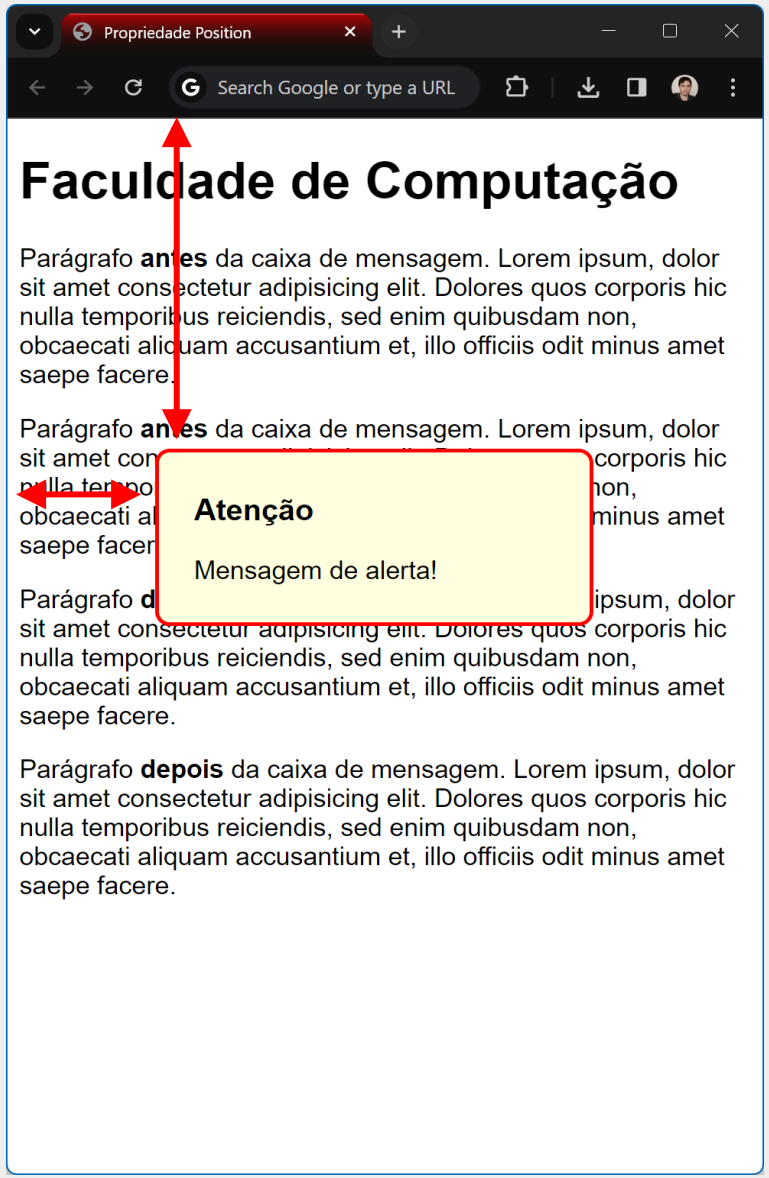
```
    .messageBox {
      width: 25ch;
      border: solid 3px red;
      border-radius: 10px;
      background-color: lightyellow;
      padding: 0.5rem 1.5rem;
    }
  </style>
</head>
<body>
  <main>
    <h1>Faculdade de Computação</h1>
    <p>Parágrafo <b>antes</b> da caixa
    <p>Parágrafo <b>antes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

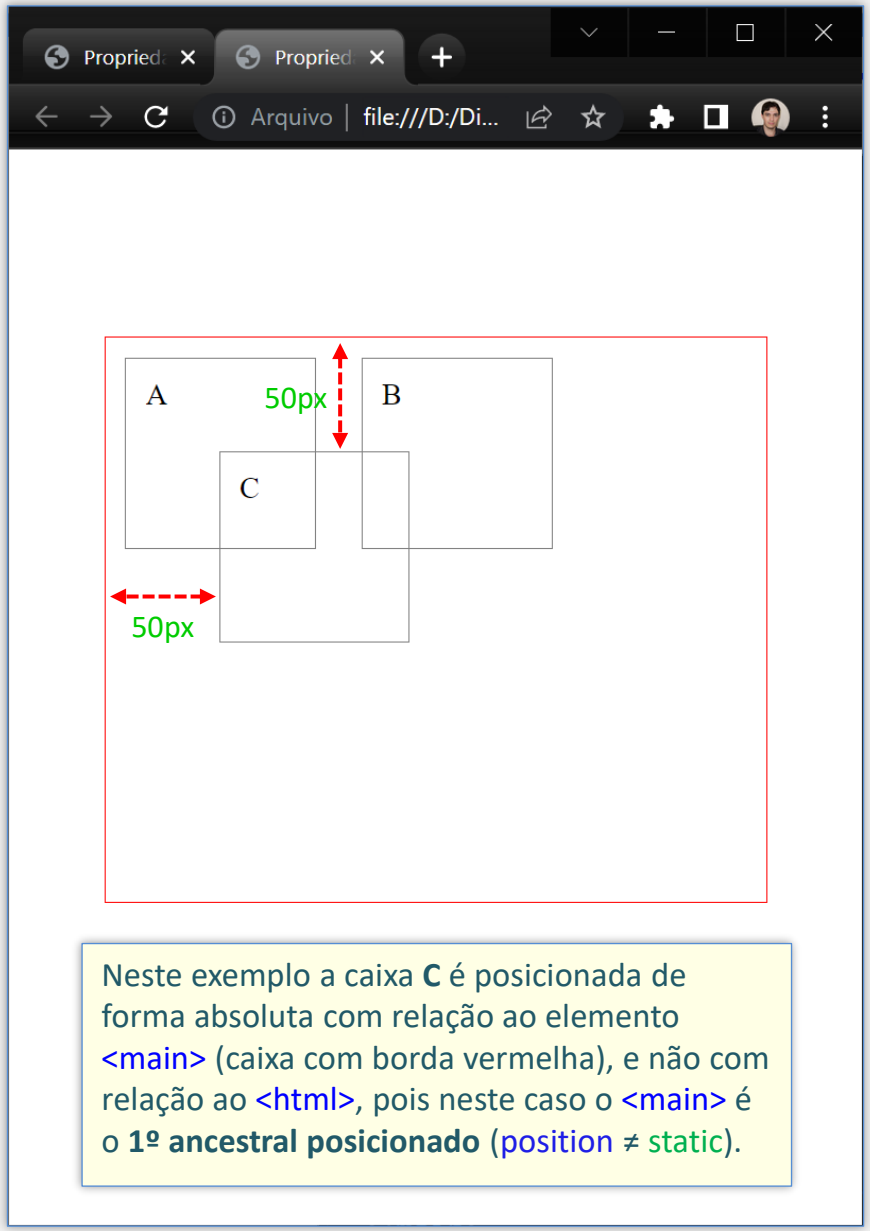
```
    .messageBox {
      width: 25ch;
      border: solid 3px red;
      border-radius: 10px;
      background-color: lightyellow;
      padding: 0.5rem 1.5rem;
      position: absolute;
      top: 230px;
      left: 100px;
    }
  </style>
</head>
<body>
  <main>
    <h1>Faculdade de Computação</h1>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



Neste exemplo a caixa **C** é posicionada de forma absoluta com relação ao elemento <main> (caixa com borda vermelha), e não com relação ao <html>, pois neste caso o <main> é o **1º ancestral posicionado** (position ≠ static).

Conteúdo Complementar

Propriedade position - continuação

position: sticky

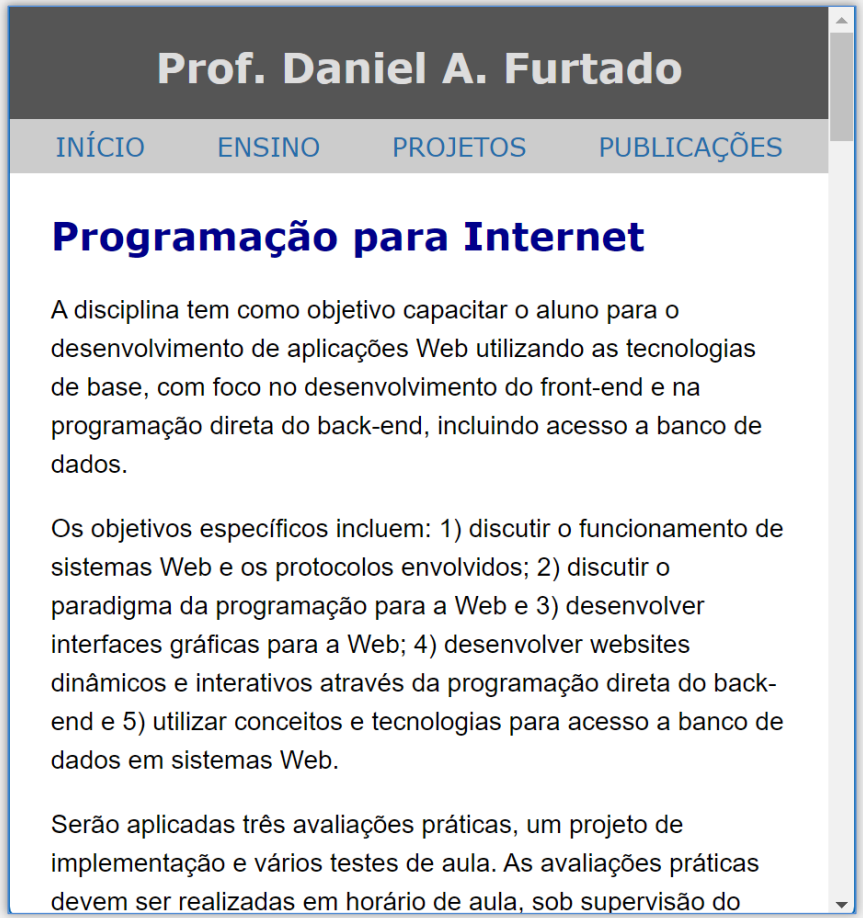
- Elemento posicionado de acordo com fluxo normal do documento
- Elemento continua **ocupando espaço no layout**
- Elemento "gruda" no primeiro ancestral com mecanismo de rolagem
- Frequentemente utilizado em conjunto com top: 0

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

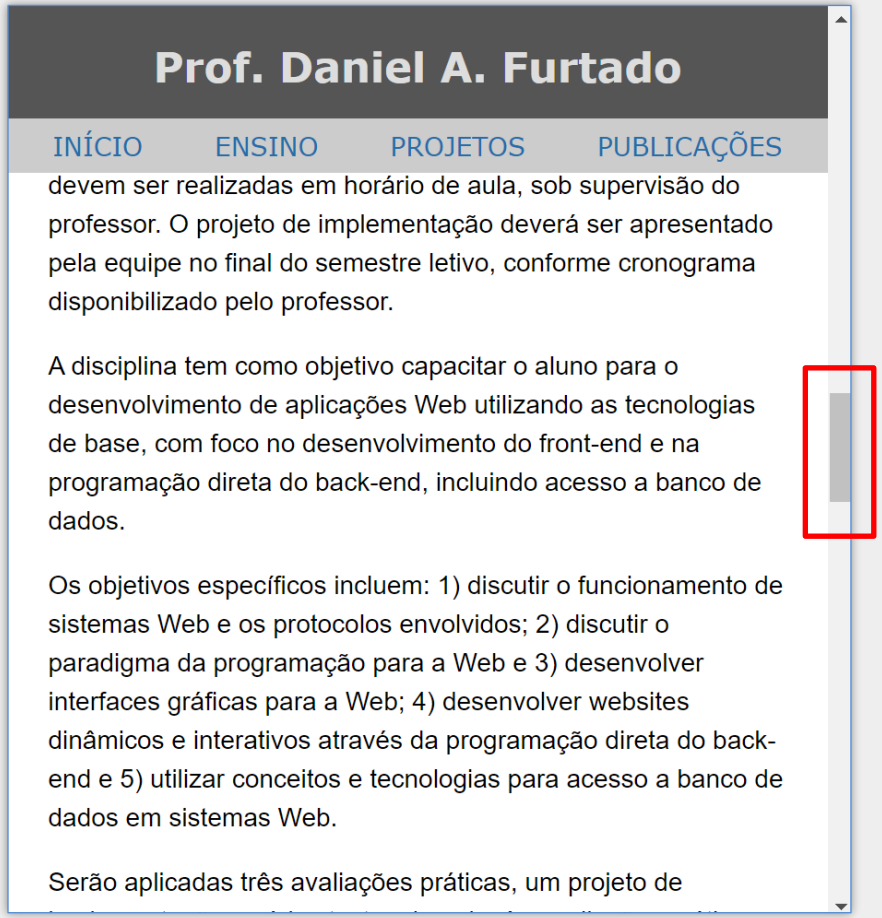
Exemplo 1 de position: sticky

```html
<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>
```

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

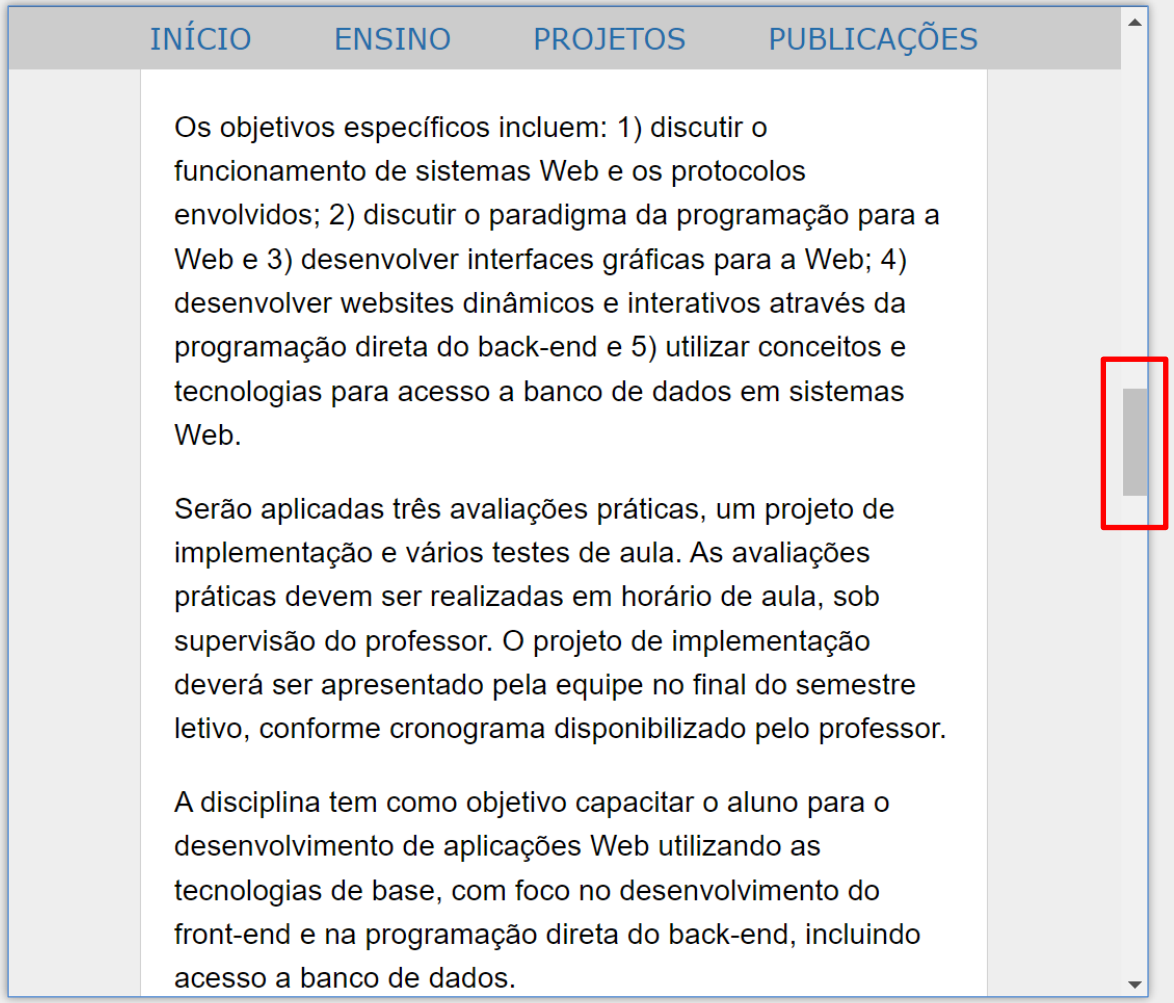


Página com cabeçalho (que inclui barra de navegação) fixado com position: sticky. Exibição sem rolagem vertical.



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

Exemplo 2 de position: sticky



Neste exemplo a barra de navegação (<nav>) foi colocada fora do <header> e seu posicionamento foi feito com position: sticky e top: 0.

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 desejar posicionar algo de maneira fixa na tela, sobre o restante do conteúdo, como em painéis de avisos sobre cookies, janelas modais etc.

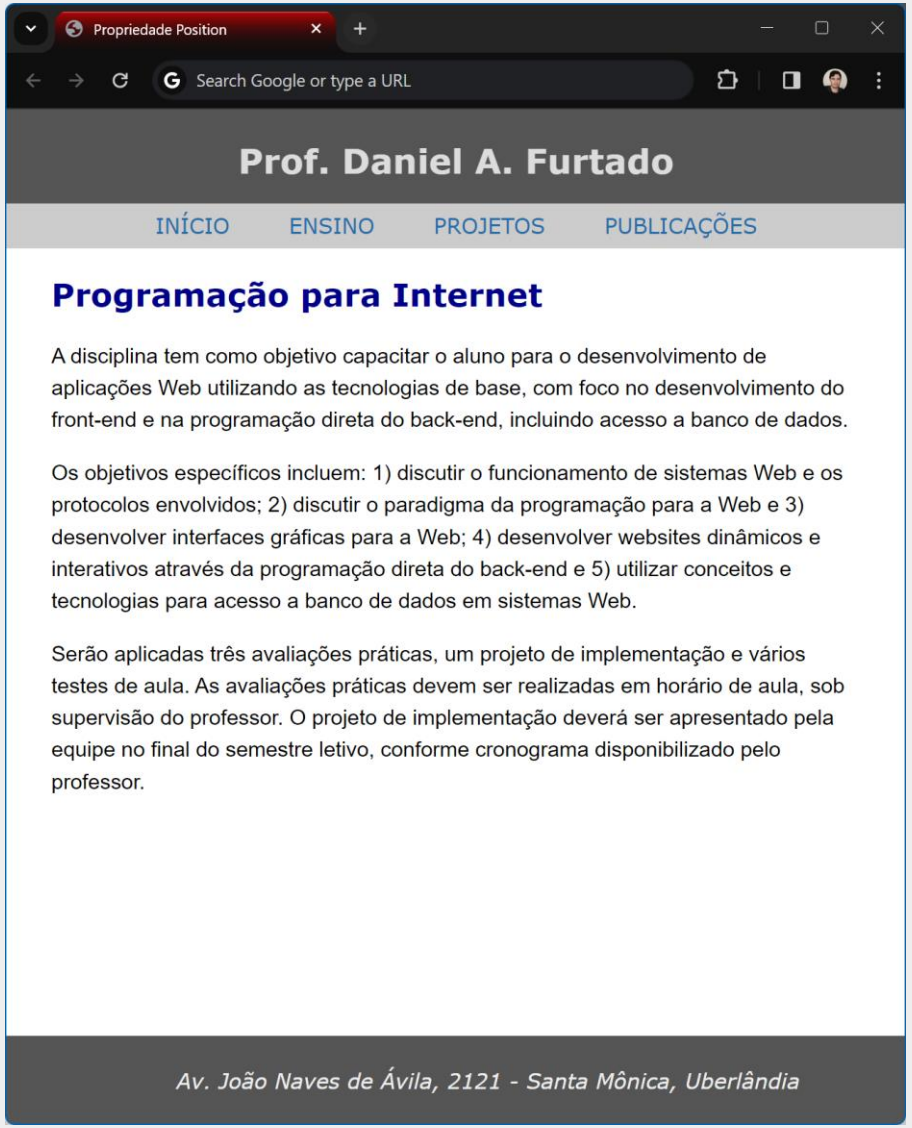
Exemplos disponíveis em https://youtu.be/caJ7Q65aiLE?t=2200

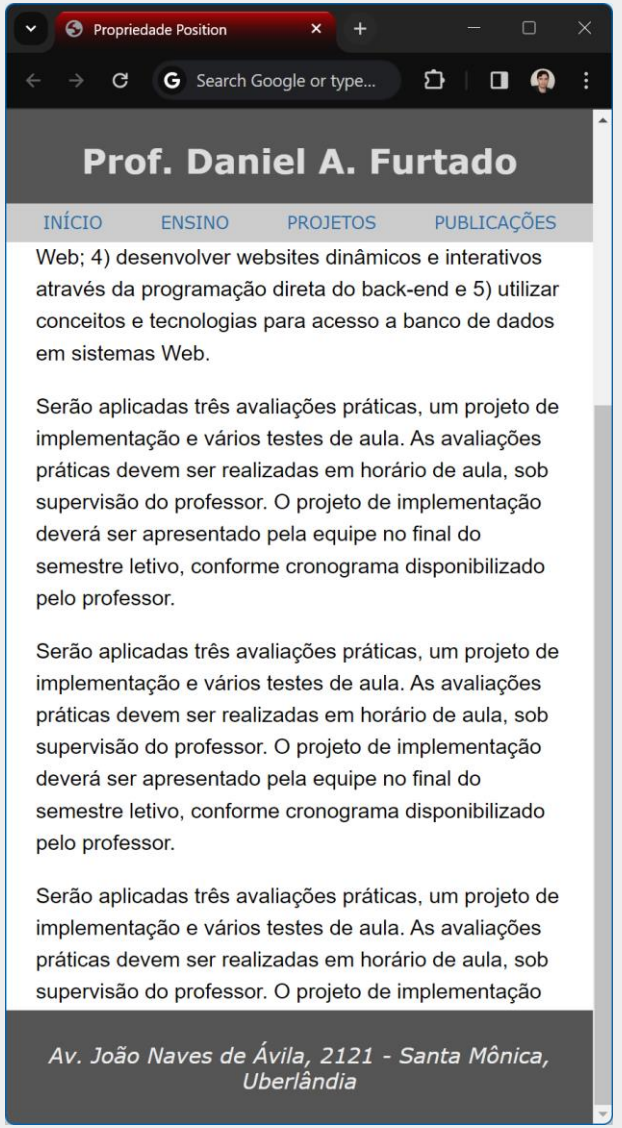
Conteúdo Complementar

Exemplo de position: fixed

```css
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).





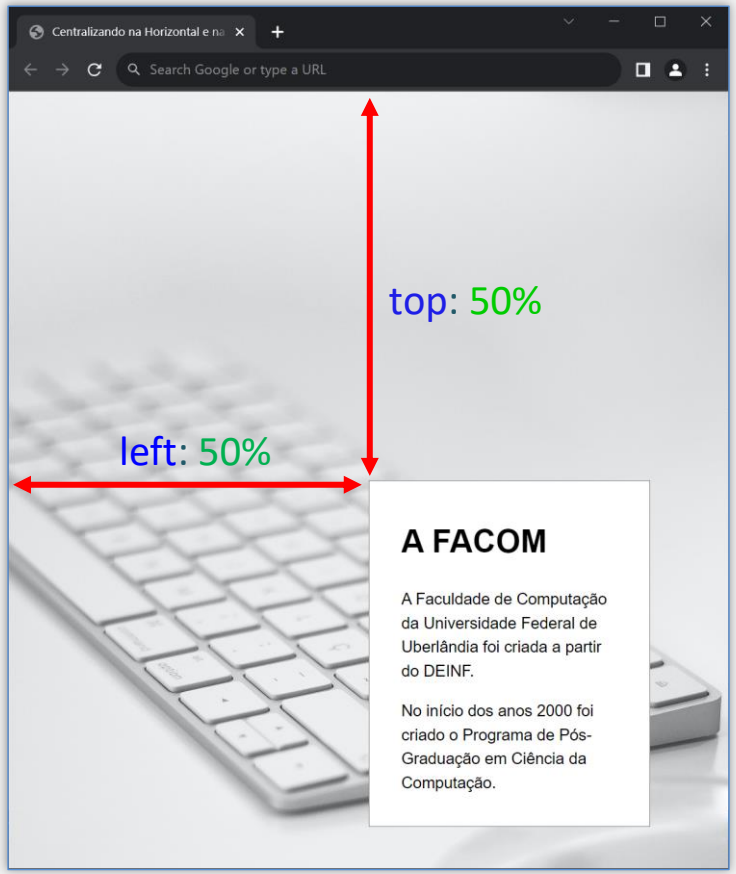
Conteúdo Complementar

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%;
}
```


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

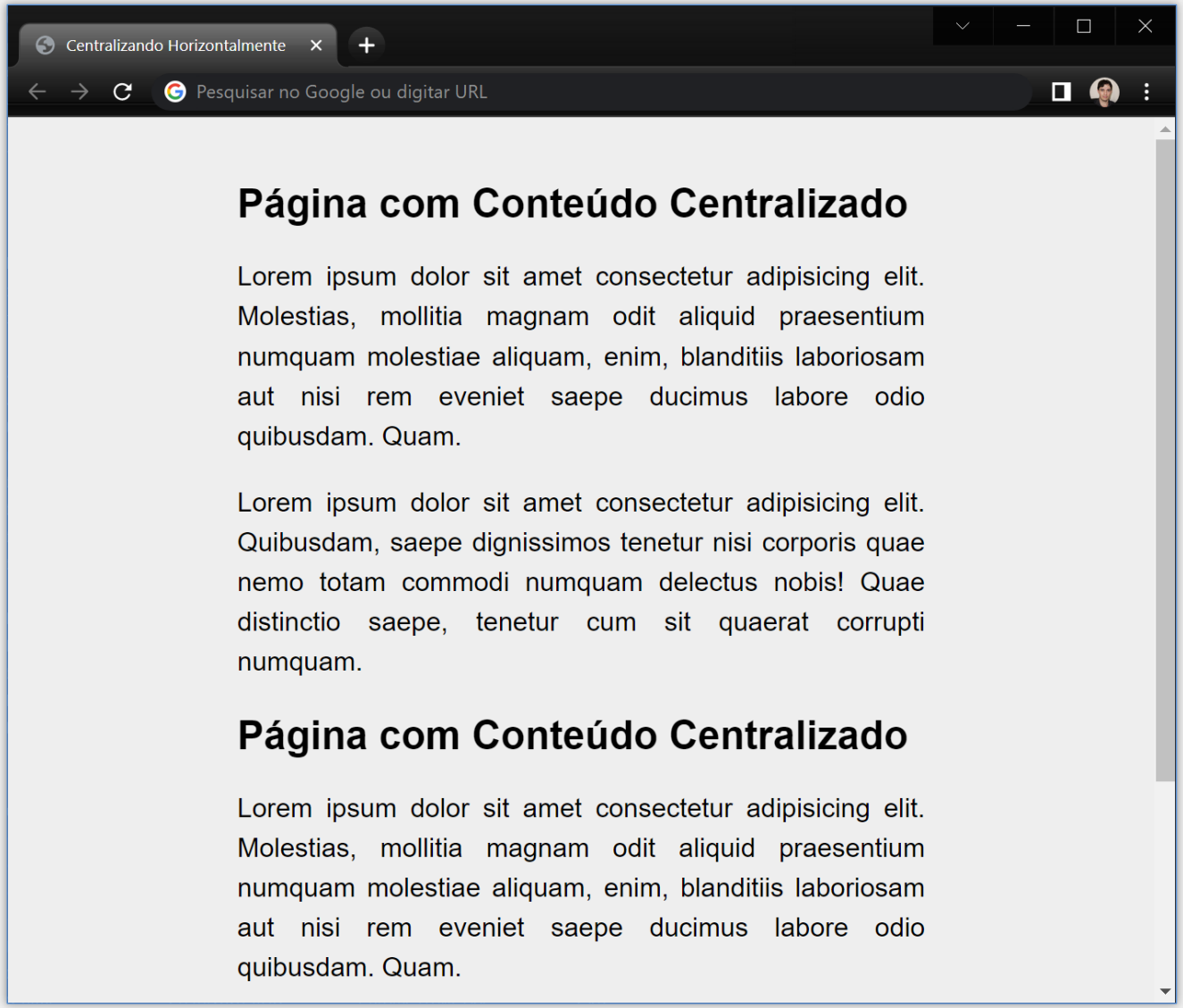
OBS: vale lembrar que esta forma de centralização remove o elemento do layout, deixando ele sobre os demais.

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 restante, resultando na centralização
- Para centralizar apenas o texto dentro de um elemento, pode-se utilizar apenas text-align: center

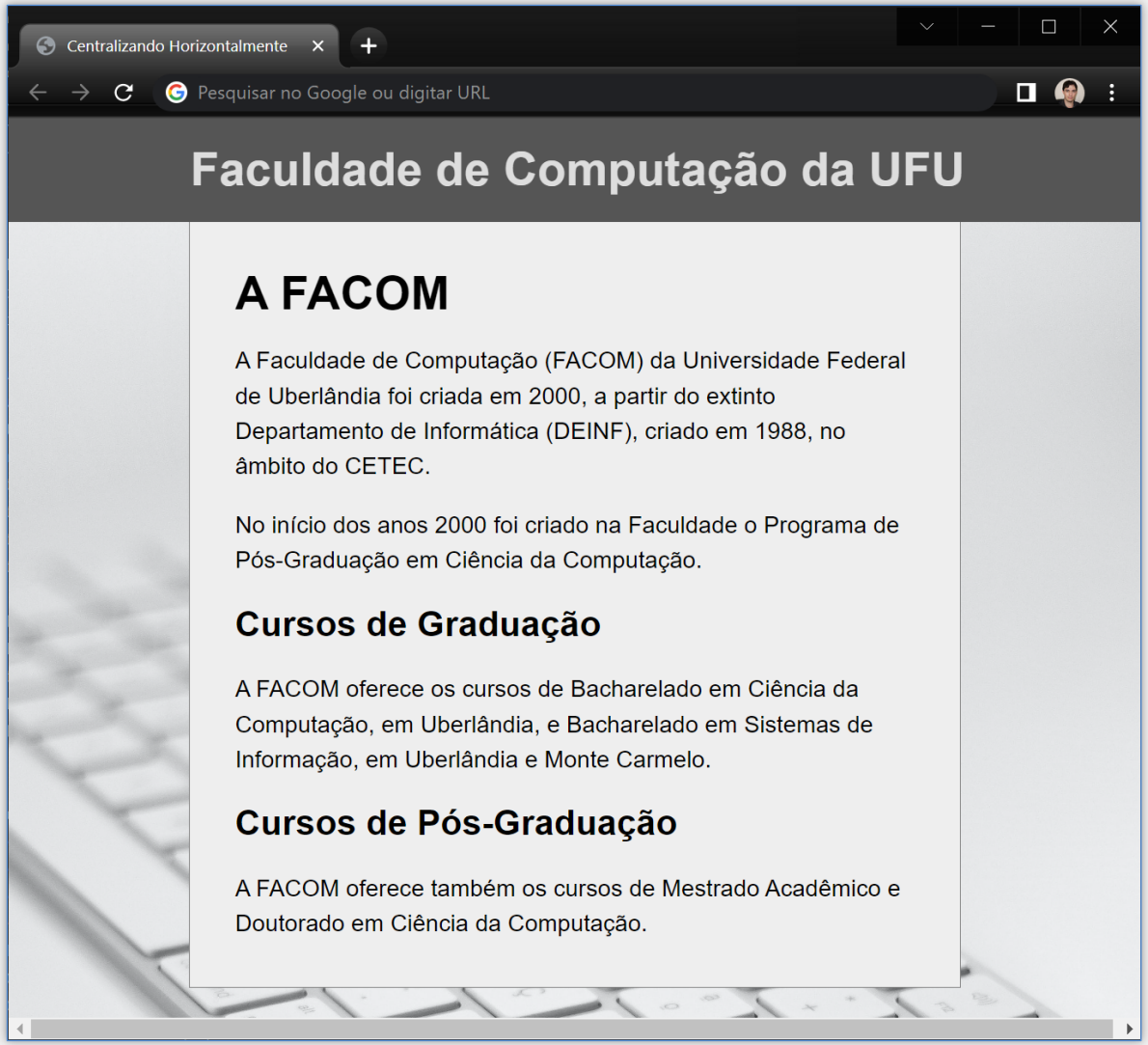
Centralizando na Horizontal com width e margin

```
  <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
```



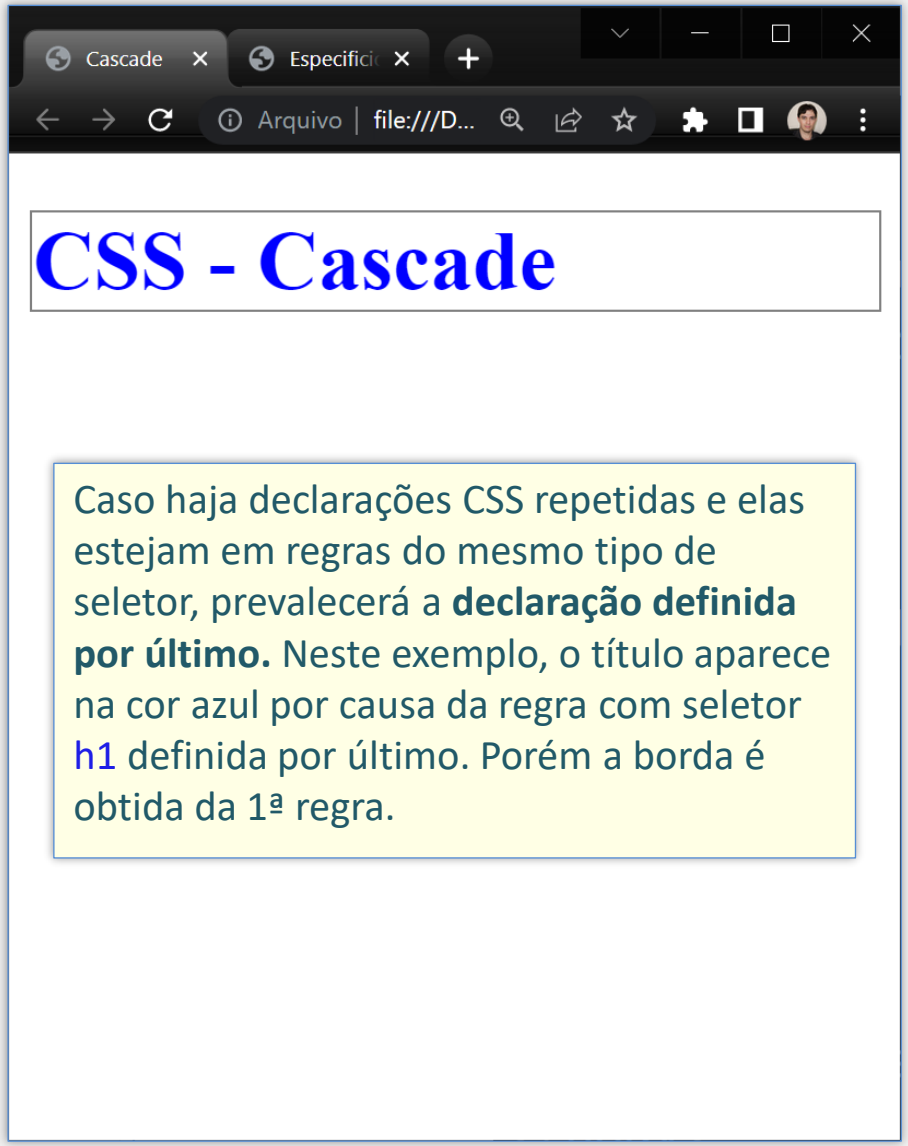
Cascade, Especificidade e Herança

Cascade, Especificidade e Herança

- Em caso de regras CSS 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 pela aplicação das regras CSS
  - Ordem no código
  - Especificidade
  - Herança

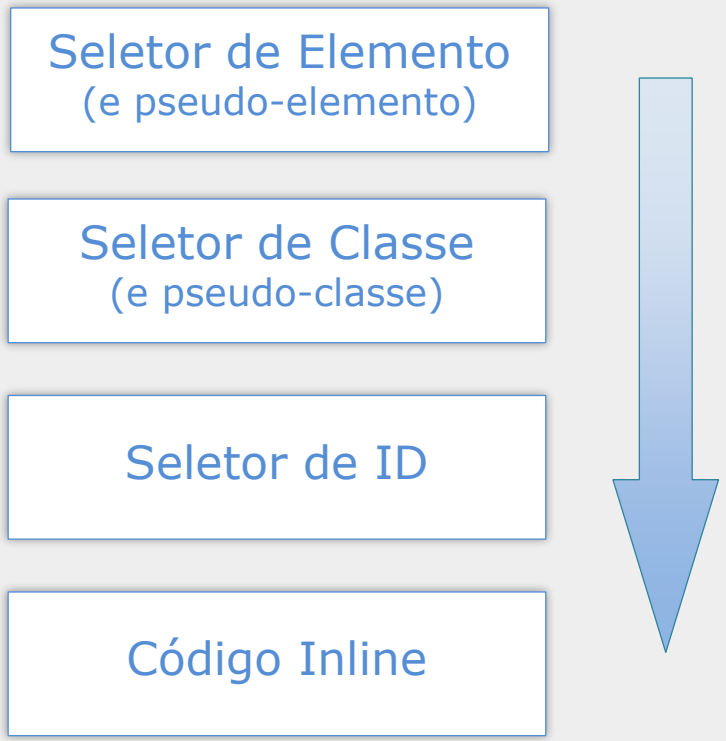
Ordem das Regras – Cascade

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



Caso haja declarações CSS repetidas e elas estejam em regras do mesmo tipo de seletor, prevalecerá a **declaração definida por último.** Neste exemplo, o título aparece na cor azul por causa da regra com seletor h1 definida por último. Porém a borda é obtida da 1ª regra.

Especificidade

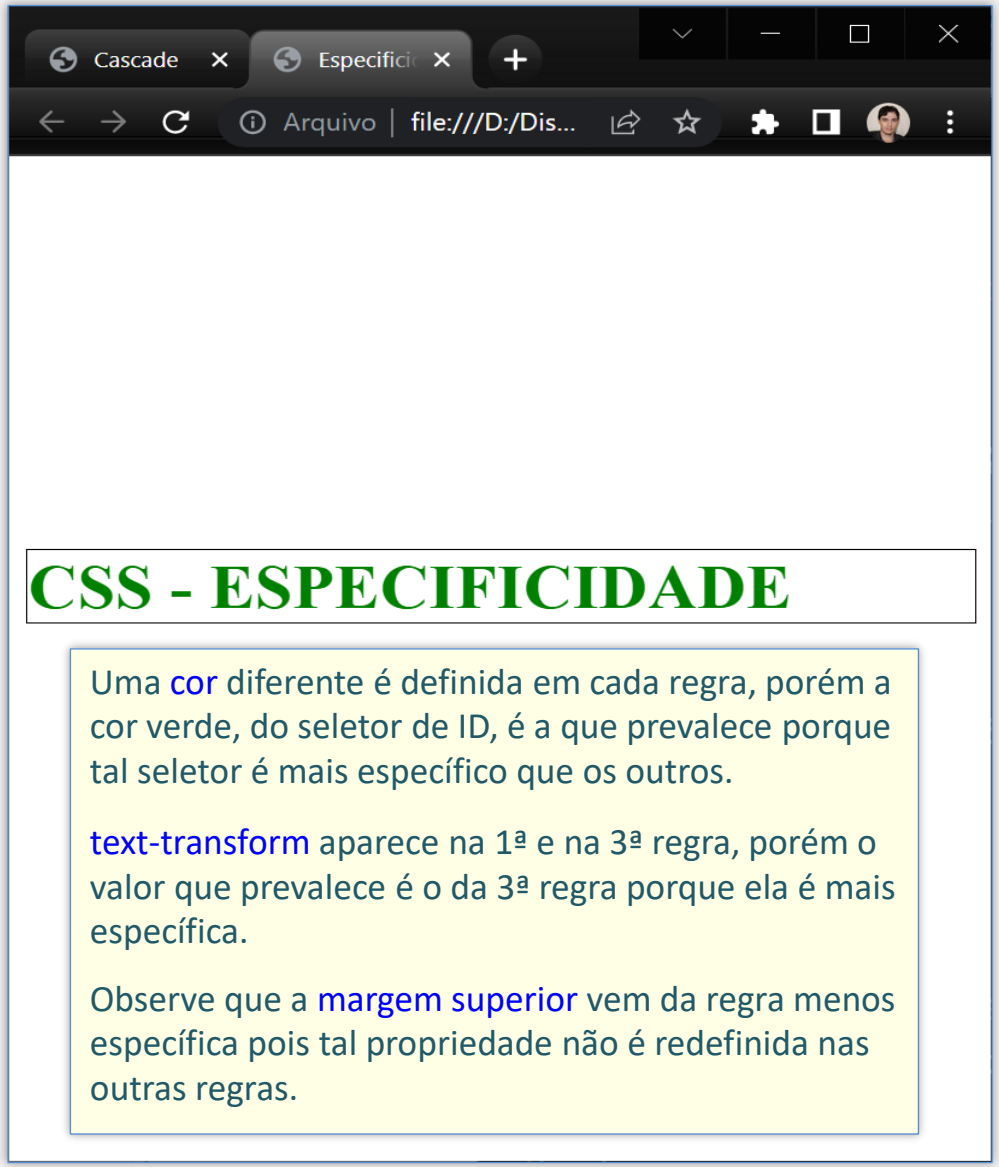


Maior especificidade

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>
```



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 herda a fonte definida para o
- Outras propriedades não herdam
  - Ex.: width, margin e padding
  - Ex.: um <p> dentro de um não herda as margens definidas para o
- Definir uma propriedade para o valor inherit ativa a herança

Conteúdo Complementar

Ícones

Conteúdo Complementar

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

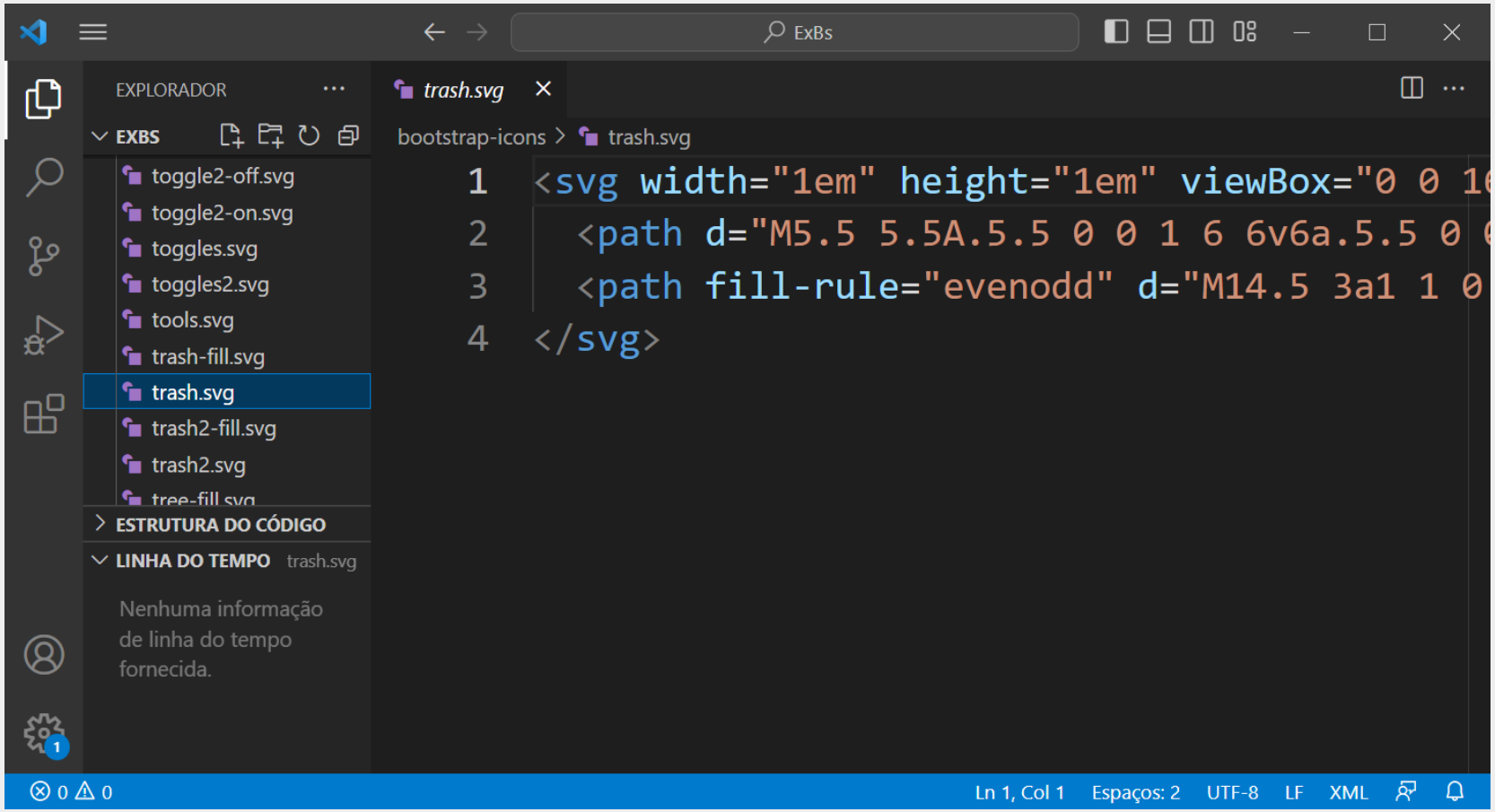
Conteúdo Complementar

Ícones SVG

- SVG ➔ **S**calable **V**ector **G**raphics
- 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 `<img>`)
- Flexibilidade com ícones multicoloridos e animações

Conteúdo Complementar

Exemplo de Ícone SVG do Framework Bootstrap



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

Conteúdo Complementar

Formas de Uso de Ícones SVG

Como imagem

- <img src="arquivoSvgDoIcone.svg">
- 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

Conteúdo Complementar

Formas de Uso de Ícones SVG - Exemplo

```
<!-- ícone SVG inserido como imagem -->
<button class="btn btn-danger">
  <img src="bootstrap-icons/trash.svg" alt="" width="16" height="16" title="Excluir">
  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 0V6a.5.5 0 0 1 .5-.5zm2.5 0a.5.5 0 0 1 .
    <path fill-rule="evenodd" d="M14.5 3a1 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).

Conteúdo Complementar

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, Google 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">
```

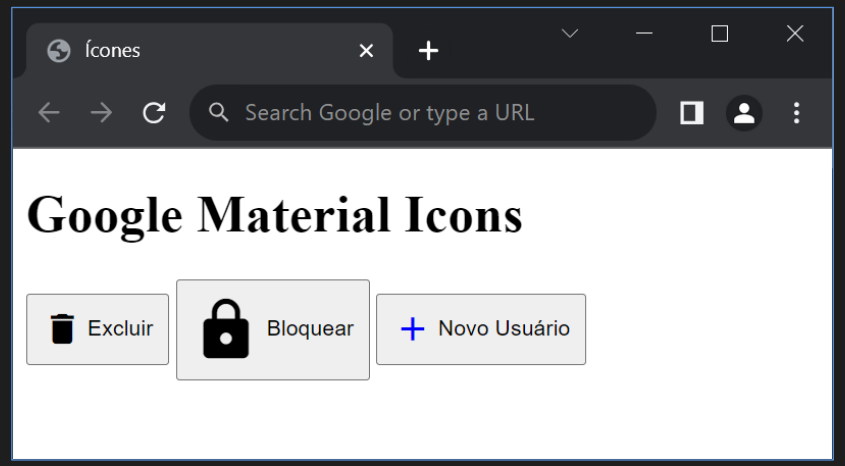
3. Utilize a respectiva classe e o nome do ícone como conteúdo

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

Conteúdo Complementar

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>
```



Conteúdo Complementar

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.