Desenvolvimento Web II

Módulo 1

Revisão de HTML e CSS

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

Conteúdo do Módulo

- Revisão de conceitos e elementos comuns da HTML
- Revisão de conceitos e elementos de formulários
- Revisão de conceitos e propriedades comuns da CSS
- Serviço de hospedagem online

Estrutura Básica de um Documento HTML

Indica que é um documento HTML5

Início do documento HTML

Cabeçalho do documento

Corpo do documento

Fim do documento HTML

```
<!DOCTYPE html>
<html>

    <head>
        <title>Titulo da Pagina</title>
    </head>

    <body>

        <h1>Primeira Pagina</h1>
        <p>Ola, mundo!</p>

    </body>


</html>
```

O cabeçalho, <head>, normalmente inclui dados de configuração do documento como título, tipo de codificação dos caracteres, links com outros arquivos etc. Não aparece na página.
O título, <title>, aparecerá na aba do navegador e nos resultados dos buscadores

O corpo, <body>, é a região onde se coloca o conteúdo propriamente dito da página HTML, o qual será apresentado ao usuário.

Estrutura Básica de um Documento HTML

pt = idioma português
BR = país Brasil

Buscadores e configurações de idioma
Navegadores e traduções automáticas

Atributo

Valor do atributo

meta para inserção de metadados
charset para indicar o tipo de codificação dos caracteres
UTF-8 permite caracteres especiais, acentuação, etc.

Salvar o arquivo no editor com a mesma codificação

```
<!DOCTYPE html>
<html Lang="pt-BR">

    <head>
        <meta charset="UTF-8">
        <title>Titulo da Pagina</title>
    </head>

    <body>

        <h1>Primeira Pagina</h1>
        <p>Ola, mundo!</p>

    </body>

</html>
```

Elementos Textuais

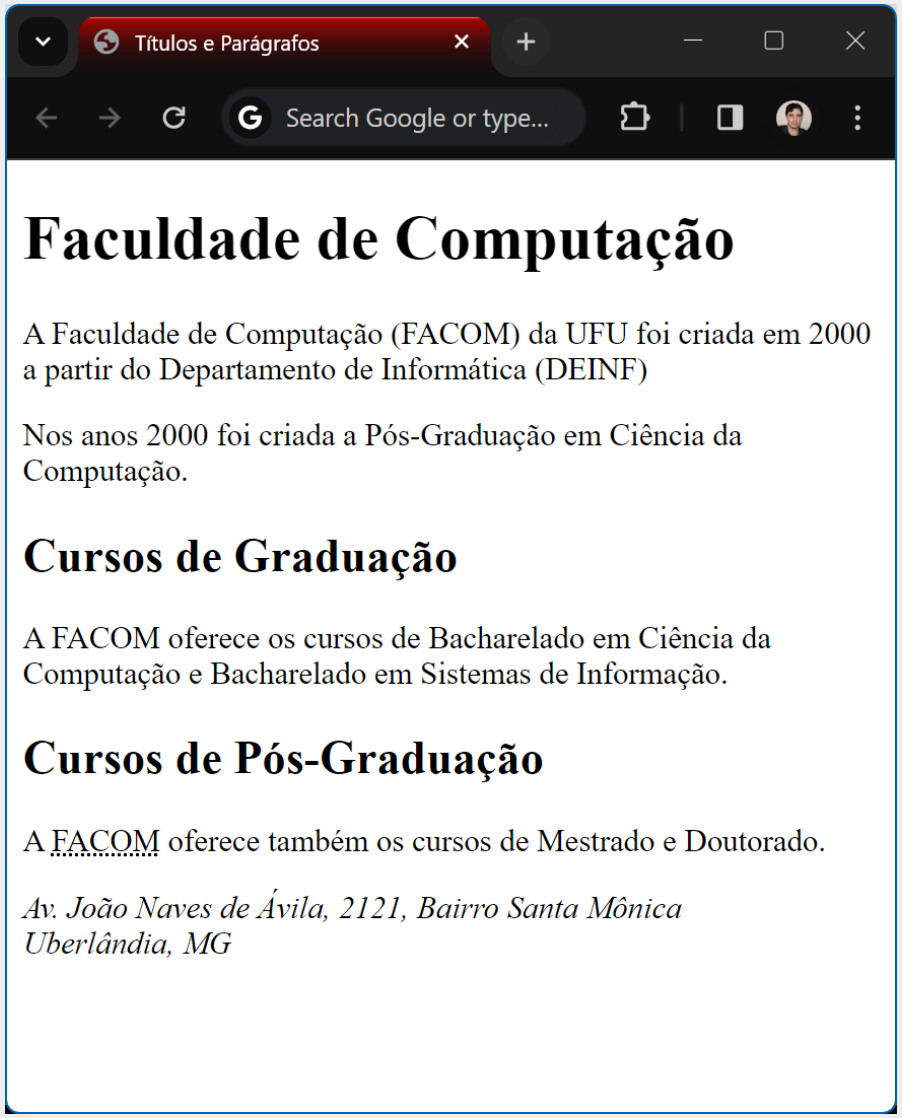
```
...
<body>
  <h1>Faculdade de Computação da UFU</h1>
  <p>A Faculdade de Computação (FACOM) da UFU foi criada em
    2000 a partir do Departamento de Informática (DEINF)</p>
  <p>Nos anos 2000 foi      criada        a Pós-Graduação

    em Ciência da Computação.</p>

  <h2>Cursos de Graduação</h2>
  <p>A FACOM oferece os cursos de Bacharelado em Ciência da
     Computação e Bacharelado em Sistemas de Informação.</p>

  <h2>Cursos de Pós-Graduação</h2>
  <p>A <abbr title="Faculdade de Computação">FACOM</abbr>
    oferece também os cursos de Mestrado e Doutorado.</p>

  <address>
    Av. João Naves de Ávila, 2121, Bairro Santa Mônica <br>
    Uberlândia, MG
  </address>
  <!-- Este é um comentário em HTML e não aparecerá -->
</body>
</html>
```

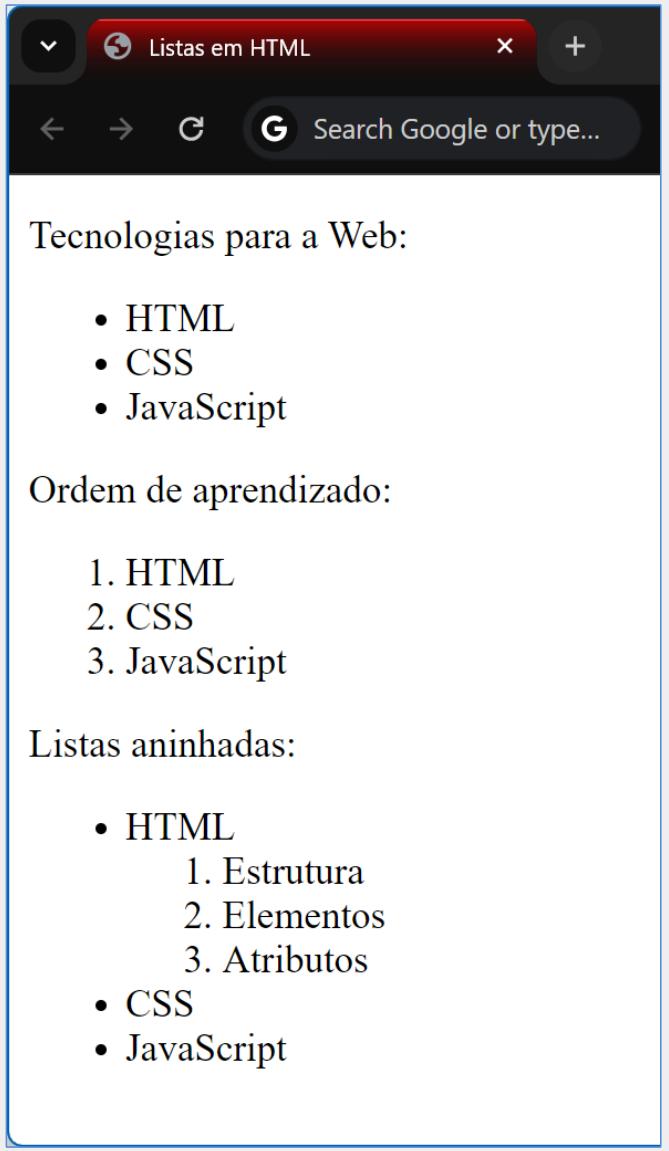


Listas

```
<p>Tecnologias para a Web:</p>
<ul>
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ul>

<p>Ordem de aprendizado:</p>
<ol>
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ol>
```

```
<p>Listas aninhadas:</p>
<ul>
  <li>HTML
    <ol>
      <li>Estrutura</li>
      <li>Elementos</li>
      <li>Atributos</li>
    </ol>
  </li>
  <li>CSS</li>
  <li>JavaScript</li>
</ul>
```



Imagens

- <img src="images/ufu.jpg" alt="UFU campus Santa Mônica">
- Atributo src
  - Permite inserir o caminho do arquivo de imagem (obrigatório)
  - Formatos aceitos: JPEG, PNG, GIF, SVG, BMP, dentre outros.
- Atributo alt
  - Fornece uma descrição da imagem (obrigatório)
  - Historicamente: texto alternativo
  - Atualmente: acessibilidade (leitura de tela), busca por imagem etc.
- Atributos width e height
  - Largura e a altura de exibição da imagem, em pixels (número inteiro, sem unidade)
  - Permitem ao navegador reservar espaço adequado
  - Previnem mudanças no layout durante carregamento
- Atributo title
  - Permite fornecer informação adicional sobre a imagem
  - O texto aparecerá quando o usuário passar o ponteiro do mouse sobre a imagem

Imagens

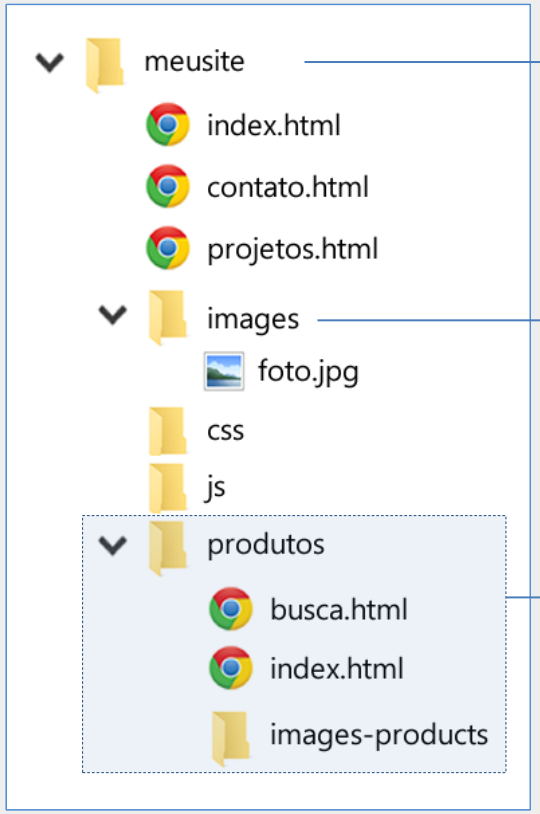
```
<body>

  <p>A UFU é uma fundação pública,
    integrante da Administração
    Federal Indireta, vinculada ao
    Ministério da Educação (MEC).</p>

  <img src="UFU-UDI-01.jpg"
    alt="UFU campus Santa Mônica"
    width="400" height="220"
    title="Foto tirada ao anoitecer">

</body>
```

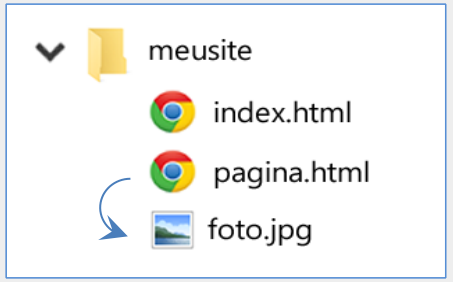
Organização dos Arquivos - Exemplo



- Recomenda-se organizar o website em múltiplas pastas
- E referenciar os arquivos de maneira relativa
- Geralmente há uma opção no editor para abrir a pasta raiz inteira

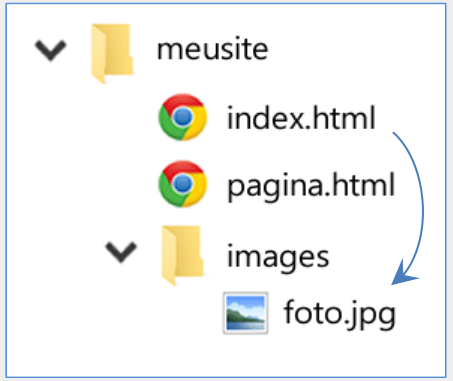
Caminhos Relativos de Arquivos

Acesso a **foto.jpg** na mesma pasta do HTML



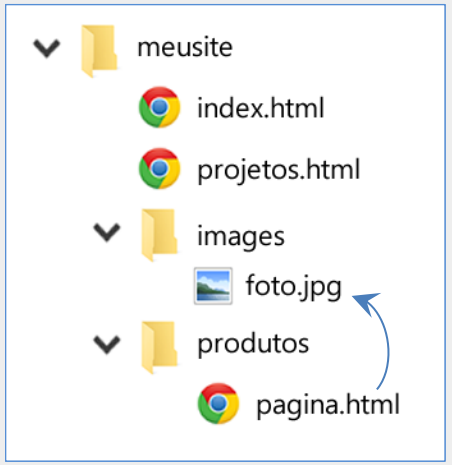

```
<img src="foto.jpg">
```

Acesso a **foto.jpg** na subpasta images




```
<img src="images/foto.jpg">
```

Acesso a **foto.jpg** a partir do arquivo pagina.html da pasta produtos



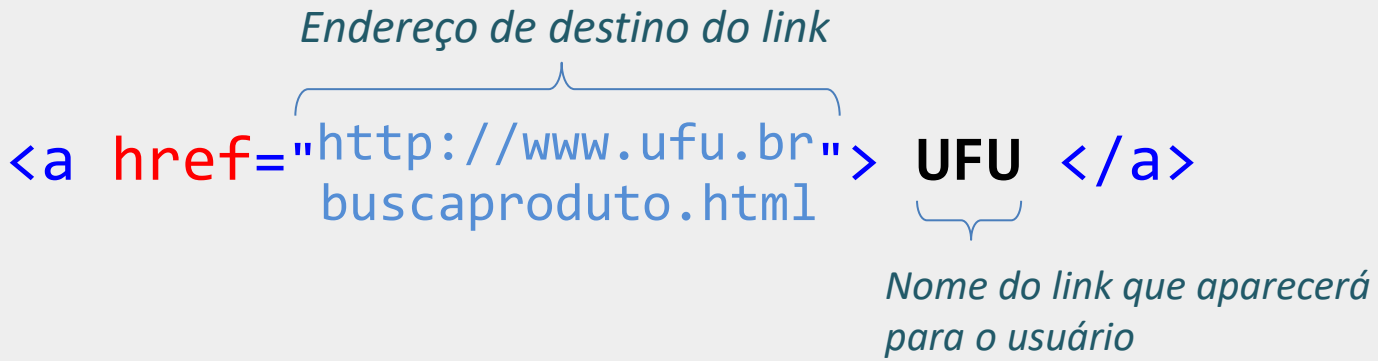
"Ponto-ponto" permite acessar a pasta superior

```
<img src="../images/foto.jpg">
```

```
<img src="/images/foto.jpg">
```

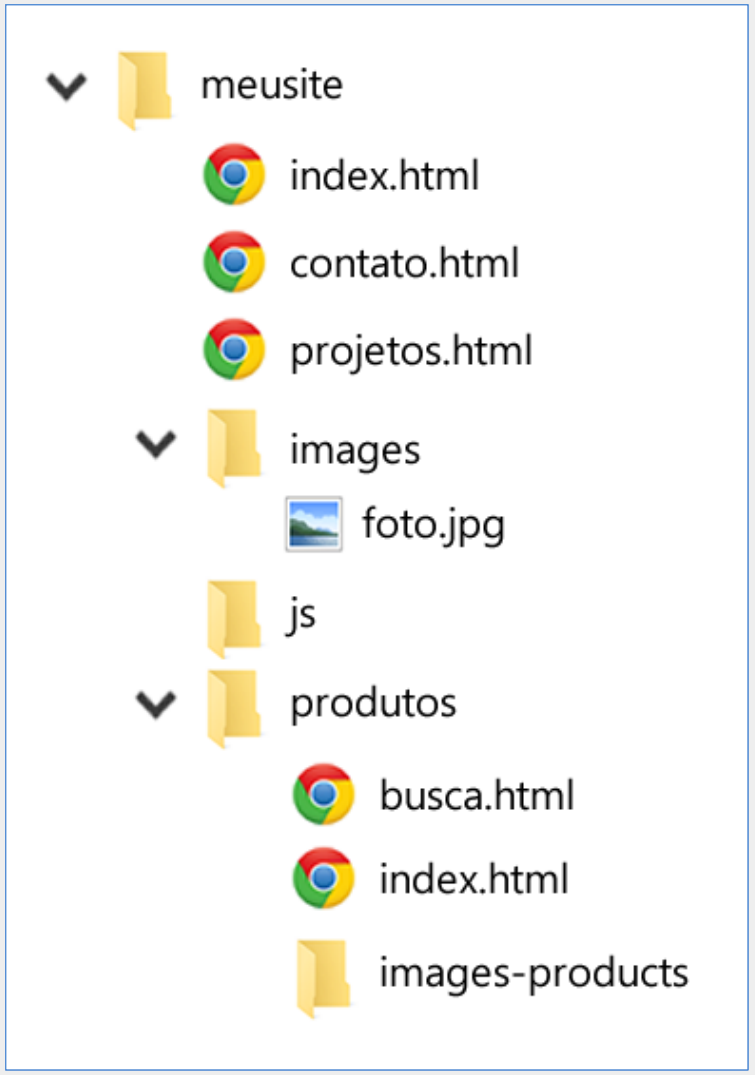
Iniciar o caminho com "/" permite acesso direto à pasta raiz do website, independemente de onde se faz o acesso

Hiperlinks



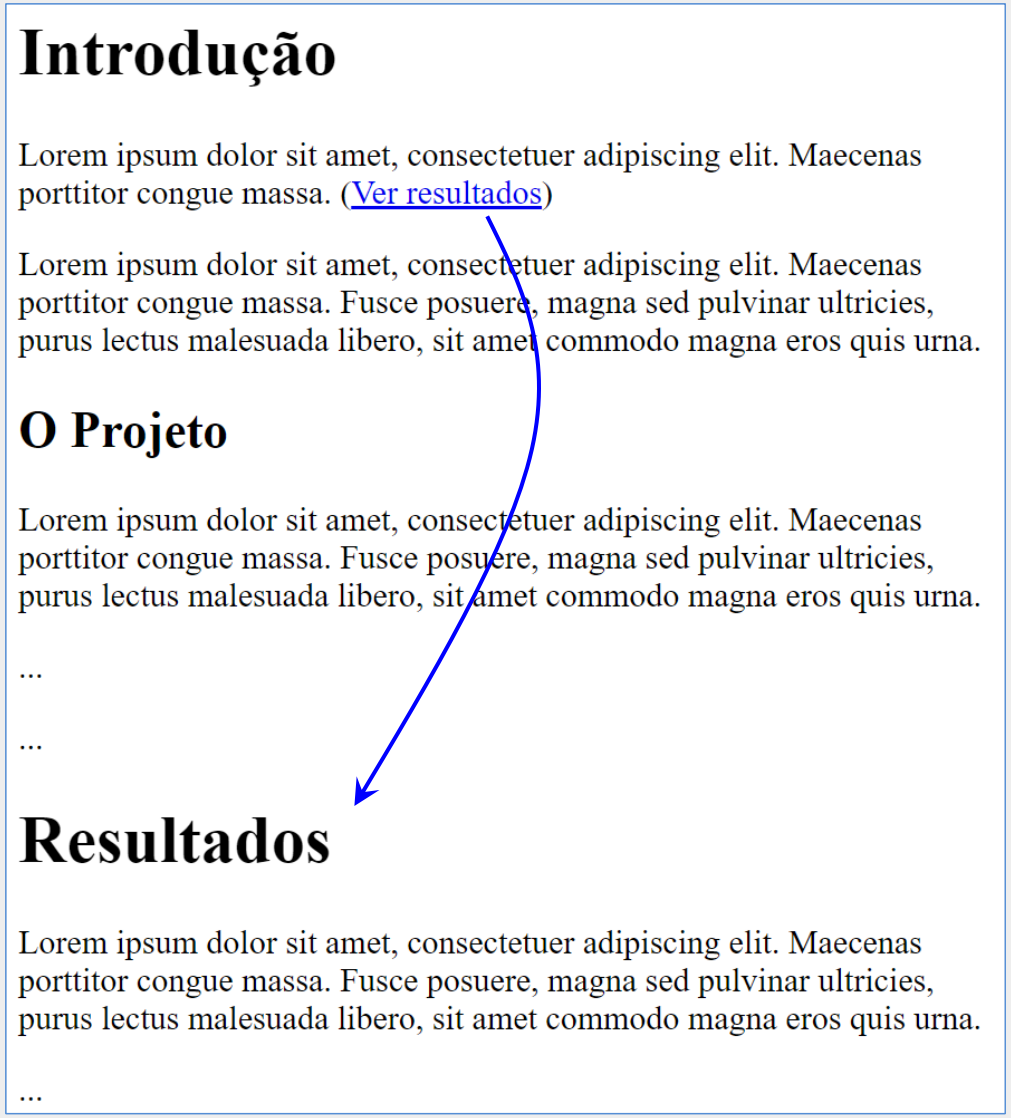
- Endereço absoluto
  - Inclui nome de domínio e protocolo
  - Direcionamento para outro website
- Endereço relativo
  - Sem nome de domínio
  - Direcionamento dentro do próprio site
  - Preferível sempre que possível (não vincula o site a um nome de domínio)

Estrutura de Pastas e Hiperlinks Relativos



- Link de arq. HTML da pasta raiz para arq. HTML da pasta raiz
  - Contato
- Link de arq. HTML da pasta raiz para arq. HTML da pasta produtos
  - Buscar
- Link de arq. HTML da pasta produtos para arq. HTML da pasta produtos
  - Buscar
- Link de arq. HTML da pasta produtos para arq. HTML da pasta raiz
  - Contato ou
  - Contato (apenas no servidor)

Hiperlink para Fragmento do Documento




```
...
<a href="#resultados">
      Ver resultados
</a>
...
<h1 id="resultados">Resultados</h1>
...
```

Outros Hiperlinks

Link para enviar e-mail

```
<a href="mailto:exemplo@exemplo.com">
    Entre em contato por e-mail
</a>
```

Link para número de telefone

```
<a href="tel:034-9999-9999">
    Entre em contato por telefone
</a>
```

Esses dois tipos de hiperlinks são especialmente úteis quando o usuário está visitando a página de um dispositivo móvel como um smartphone. No primeiro exemplo, quando o link for clicado, o navegador abrirá o aplicativo de e-mail do usuário e já preencherá o campo 'enviar para', bastando que o usuário termine o procedimento.
No segundo exemplo, o navegador abrirá o aplicativo de discagem do usuário, com os números prontos para iniciar a chamada.

Tabelas

<table>, <tr>, <td>

- Cria a tabela, a linha e a célula na linha, respectivamente

<thead>, <tbody>, <tfoot>

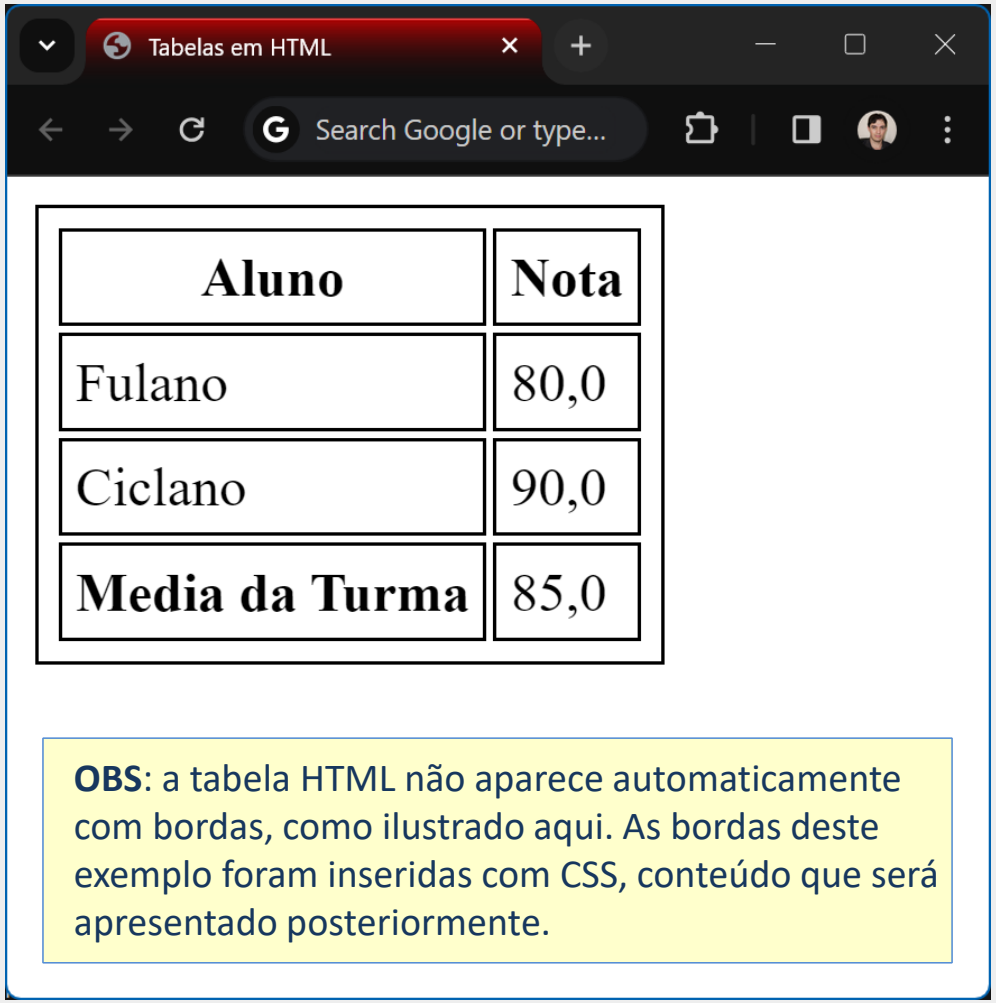
- Definem seções de cabeçalho, corpo e rodapé na tabela, respectivamente
- Comumente utilizados em tabelas longas
- Especialmente úteis quando tabelas longas são enviadas para impressão, pois cabeçalhos e rodapés serão repetidos em cada página, facilitando a leitura.

<th>

- **t**able **h**ead
- Define uma célula de cabeçalho dentro de uma linha
- Por padrão o navegador exibe o conteúdo em negrito e centralizado
- Comumente utilizando dentro de <thead> e <tfoot>

Tabela com Cabeçalho, Corpo e Rodapé

```html
<table>
  <thead> <!-- Cabeçalho da tabela -->
    <tr>
      <th>Aluno</th>
      <th>Nota</th>
    </tr>
  </thead>
  <tbody> <!-- Corpo da tabela -->
    <tr>
      <td>Fulano</td>
      <td>80,0</td>
    </tr>
    <tr>
      <td>Ciclano</td>
      <td>90,0</td>
    </tr>
  </tbody>
  <tfoot> <!-- Rodapé da tabela -->
    <tr>
      <th>Media da Turma</th>
      <td>85,0</td>
    </tr>
  </tfoot>
</table>
```



OBS: a tabela HTML não aparece automaticamente com bordas, como ilustrado aqui. As bordas deste exemplo foram inseridas com CSS, conteúdo que será apresentado posteriormente.

Elementos de Bloco e Elementos de Linha

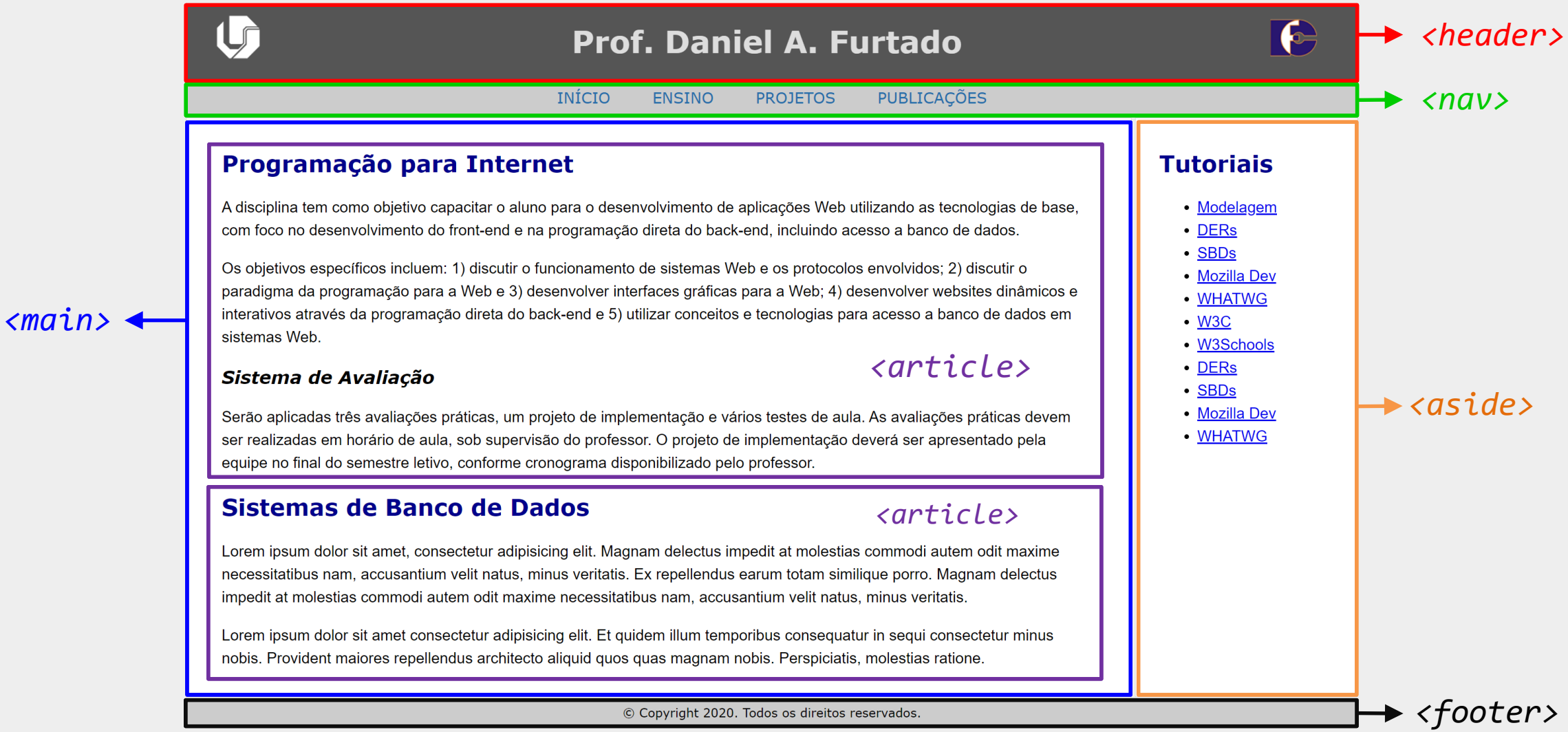
- Elementos de bloco
  - Começam e terminam com uma **quebra de linha**
  - Ocupam, por padrão, **toda a largura disponível**
  - Exemplos: <p>, <h1>, <table>, <ol>, <ul>, <li>,
- Elementos de linha
  - Não começam com quebra de linha
  - Ocupam **somente a largura necessária para sua exibição**
  - Exemplos: <img>, <a>, , <b>,

Elementos e

- **Container genérico de nível de bloco**
- Utilizado para agrupar outros elementos
- Sem semântica (não agrega significado)
- Usar apenas quando não houver elemento semântico mais apropriado (como <header>, <footer>, <main>, <article>, <nav>, <aside> etc.)

- **Container genérico de nível de linha**
- Sem semântica (não agrega significado)
- Usar apenas quando não houver elemento semântico mais apropriado

Elementos Semânticos de Estrutura e Layout



Elementos Semânticos de Estrutura e Layout - Exemplo

```html
<body>
  <!-- Cabeçalho principal, compartilhado por todas as páginas do website -->
  <header>
    <img src="images/logoUFU.png" alt="Logo da UFU">
    <h1>Prof. Daniel A. Furtado</h1>
  </header>
  <!-- Menu de navegação principal, compartilhado pelas páginas do website -->
  <nav>
    <ul>
      <li><a href="#">ENSINO</a></li>
      <li><a href="#">PROJETOS</a></li>
    </ul>
  </nav>
  <!-- Conteúdo principal da página -->
  <main>
    <h2>Programação para Internet</h2>
    <p>A disciplina tem como objetivo capacitar o aluno...</p>
  </main>
  <!-- Conteúdo indiretamente relacionado ao conteúdo principal -->
  <aside>
    <p>Conteúdo da barra lateral...</p>
  </aside>
  <!-- Rodapé principal, compartilhado por todas as páginas do website -->
  <footer>
    <p>© Copyright 2024. Todos os direitos reservados.</p>
  </footer>
</body>
```

Elemento <section>

- Elemento de seccionamento de conteúdo
- Pode ser usado, por exemplo, para organizar um artigo longo em subseções
- Recomenda-se inserir um título dentro da seção (com <h2>, <h3>, etc.)
- **Não precisa** ser usado necessariamente dentro de <article>

```
...
<article>
    <section>
    ...
    </section>
    <section>
    ...
    </section>
</article>
...
```

Formulário Simples em HTML

Indica o ***script* no servidor** que receberá e processará os dados do formulário

Permite indicar o método HTTP a ser utilizado no envio do form:
GET: dados enviados pela própria URL (*GET request*)
POST: dados enviados no corpo da requisição (*POST request*)

```
<form action="buscaProduto.php" method="get">

    <div>
        <label for="produto">Produto:</label>
        <input type="text" id="produto" name="produto">
    </div>

    <button>Buscar</button>
</form>
```

O **label** é vinculado ao campo pelos atributos **for**/**id**

Clique no rótulo foca o campo
Leitores de tela leem rótulo

Rótulo identificando o campo

Campo textual

Recomenda-se agrupar campo e rótulo em um container genérico (div) ou colocá-los dentro de um parágrafo (p)

Envia o formulário

O dado é submetido com este nome de identificação.
Permite resgatar o dado posteriormente no servidor.
Não precisa ser igual ao id.

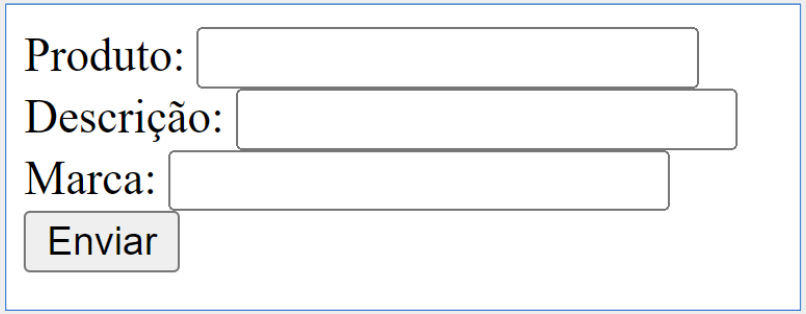
Resultado

OBS: o formulário também será submetido caso o usuário pressione a tecla **Enter**, mesmo que não contenha um botão de submissão. Caso o atributo action esteja vazio, o formulário será submetido ao próprio arquivo HTML no servidor, podendo causar o recarregamento da página.

Formulário Simples em HTML

```
<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>
```



Outra Forma de Vincular os Rótulos aos Campos

- Outra maneira de associar corretamente os rótulos aos campos é inserindo o elemento <input> como conteúdo do elemento <label>
- Neste caso não é necessário utilizar o atributo for juntamente com o id
- Porém o atributo name continua sendo fundamental

```
<form action="cadastra.php" method="post">
   <p><label>Produto: <input type="text" name="prodNome"></label></p>
   <p><label>Descrição: <input type="text" name="prodDesc"></label></p>
   <p><label>Marca: <input type="text" name="prodMarca"></label></p>

   <button>Enviar</button>
</form>
```

Não Utilize
 para Separar os Campos

-
's não devem ser utilizados para separar campos de formulário, conforme detalhado pelo WHATWG na especificação da HTML (https://html.spec.whatwg.org/multipage/text-level-semantics.html#the-br-element);
- Ao invés de quebrar a linha com o
, utilize um container de bloco (div) ou um elemento <p> conforme apresentado anteriormente

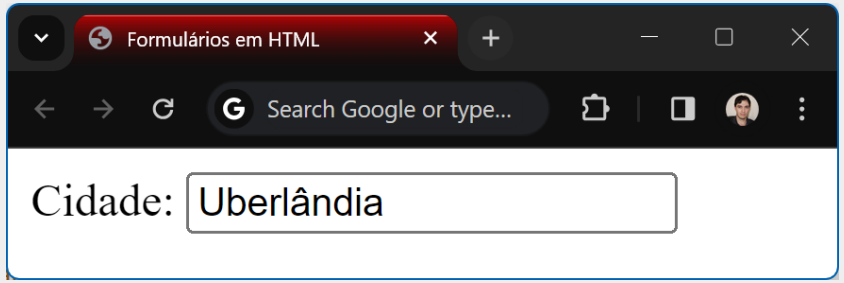
Não faça isso!

```
<form action="cadastra.php" method="post">
  <label>Produto: <input type="text" name="prodNome"></label>
  <br>
  <label>Descrição: <input type="text" name="prodDesc"></label>
  <br>
  <label>Marca: <input type="text" name="prodMarca"></label>
</form>
```

Atributo value em Campos Textuais

- O atributo value em campos textuais pode ser utilizado para fornecer um valor inicial, já preenchido, para o campo;
- O usuário poderá manter o valor sugerido ou alterá-lo;
- Não confundir com o atributo placeholder

```
<div>
  <label for="cidade">Cidade:</label>
  <input type="text" id="cidade" name="cidade" value="Uberlândia">
</div>
```



Envio de Formulário com o Método GET

- Os dados do formulário são codificados pelo navegador na forma de uma string e adicionados no final da URL para envio ao servidor
- Geralmente utilizado em formulários pequenos, sem dados sensíveis (senhas, etc.). Ex.: formulário com palavras de busca

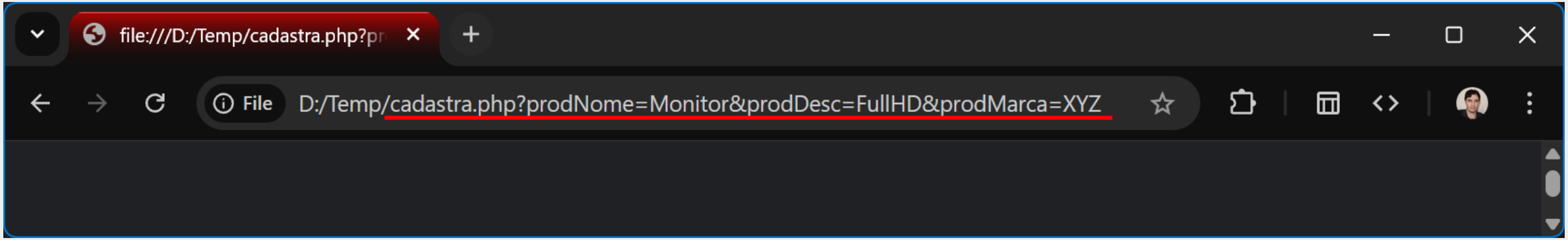
```
<form action="cadastra.php" method="GET">
  <p><label>Produto: <input type="text" name="prodNome"></label></p>
  <p><label>Descrição: <input type="text" name="prodDesc"></label></p>
  <p><label>Marca: <input type="text" name="prodMarca"></label></p>
  <button>Enviar</button>
</form>
```

Produto: Monitor

Descrição: FullHD

Marca: XYZ

Enviar



Envio de Formulário com o Método POST

- Com o método POST, os dados do formulário são enviados no **corpo** da requisição HTTP, **sem aparecer** na barra de endereços como no exemplo anterior
- Permite upload de arquivos
- Mais adequado para formulários longos e/ou com informações sensíveis
- Mais adequado quando os dados serão adicionados a uma base de dados

Observações Sobre Botões

<button> Enviar </button>

- Um elemento button sem o atributo type cria um botão do tipo submit
- Dentro de um <form>, enviará o formulário
- Equivalente à <button type="submit" ...>
- Um elemento button não precisa ter apenas texto como conteúdo
  - Pode-se ter, por exemplo, um <img>

<button type="button"> Nome do botão </button>

- Botão de uso geral
- Não envia um formulário automaticamente. Pode ser útil quando o formulário precisa ser enviado de outras formas (utilizando Ajax, por ex.) ou quando é necessário ter um botão dentro do formulário para desempenhar outra função.

Campo para Envio de Arquivo

- Para enviar um arquivo no formulário, pode-se utilizar o <input> como no exemplo a seguir, o qual criará um botão para seleção de arquivo:
  - <input type="file" name="fotoCadastro">
- É importante mencionar que é necessário, também, utilizar o atributo enctype="multipart/form-data" no elemento <form>
- O atributo multiple permite a seleção de vários arquivos
- O atributo accept permite restringir o tipo de arquivo permitido
  - <input type="file" name="meuArq" accept="image/*">
  - <input type="file" name="meuArq" accept=".png, .jpg, .jpeg">

Atributo enctype

```
<form enctype="multipart/form-data" action="cadastra.php" method="post">
   <input type="file" name="meuArq">
   <button>Enviar</button>
</form>
```

- O atributo enctype pode ser necessário no <form> para alterar o método de codificação dos dados do formulário ao serem enviados
- Caso o formulário contenha arquivos, por exemplo, é necessário utilizar enctype conforme exemplo acima
- Deve ser usado apenas em conjunto com method="post"
- Valores possíveis:
  - "application/x-www-form-urlencoded" valor padrão
  - "multipart/form-data" necessário quando há arquivos no form.
  - "text/plain" dados são enviados como texto simples

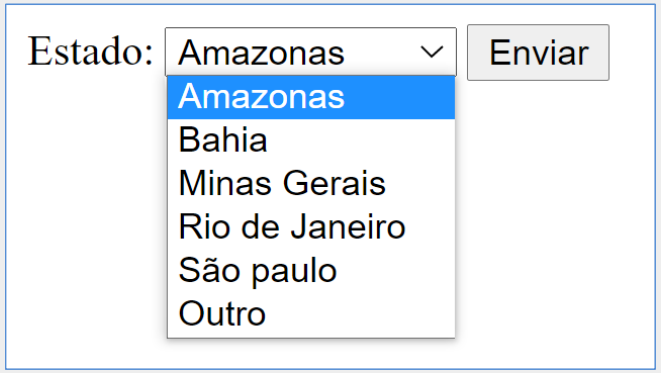
Campo de Preenchimento Obrigatório

- Para indicar que um campo é de preenchimento obrigatório, basta adicionar o atributo required no elemento
- Neste caso, o navegador não permitirá o envio do formulário sem que o campo seja preenchido
- Exemplo:

```
<input type="text" name="produto" required>
```

No caso de campos do tipo radio, o atributo required pode ser colocado em qualquer um dos itens do grupo para que todo o grupo seja tratado como de preenchimento obrigatório. Entretanto, para melhor clareza, recomenda-se o uso em todos os itens.

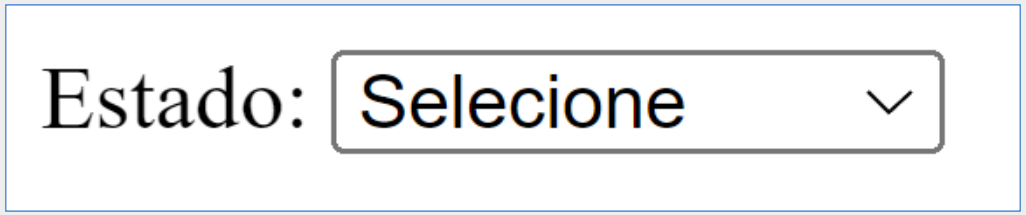
Campo select




```
<label for="estado">Estado: <label>
<select id="estado" name="estado">
    <option value="AM" selected>Amazonas</option>
    <option value="BA">Bahia</option>
    ...
</select>
```

- Permite seleção de um ou vários itens de uma lista suspensa
- Cada opção é inserida com o elemento <option>
- O conteúdo de <option> é apresentado, mas é o seu value que é enviado
- O atributo selected dentro de <option> pré-seleciona a opção
- O atributo multiple possibilita a seleção de vários itens
- Se necessário, utilize um <label> para rotular o campo

Campo select Definido como Obrigatório




```
<p><label>Estado:
   <select name="estado" required>
     <option value="">Selecione</option>
     <option value="AM">Amazonas</option>
     <option value="BA">Bahia</option>
   </select>
</label></p>
```

Quando um <select> de seleção simples é definido como obrigatório (required) e o campo não possui uma opção pré-selecionada (selected), então deve-se utilizar um elemento <option> com value="" para atuar como **placeholder** e forçar o usuário a escolher outra opção (pois se o campo é obrigatório, o navegador não permitirá o envio da opção com value="").

Restringindo valores com o atributo pattern

pattern

- Permite indicar expressão regular para filtrar os valores permitidos
- Ex.: <input type="text" pattern="\d{3}\.\d{3}\.\d{3}-\d{2}">
  - Aceita um texto no formato xxx.xxx.xxx-xx, onde x são números
  - Não é uma máscara, mas uma restrição
- Ex.: <input type="text" pattern="[A-Za-z]{2}">
  - Aceita duas letras com os caracteres alfabéticos A-Z e a-z

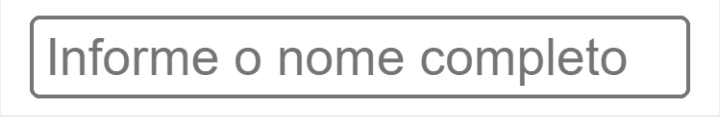
Outros Atributos para Campos Textuais

minlength e maxlength

- Permitem definir o número mínimo e o número máximo de caracteres permitido para o campo, respectivamente
- Ex: <input type="text" maxlength="30">

placeholder

- Permite a inserção de uma dica de preenchimento
- A dica é apagada quando o campo é preenchido
- Ex: <input type="text" placeholder="Informe o nome completo">



Website Online

Colocando o Website Online

- Para colocar o website online normalmente é utilizado um nome de domínio e um servidor web
- Nome de domínio
  - Endereço do website na Web
  - Pode-se registrar um nome de domínio próprio
  - Ou utilizar um nome de domínio existente e criar um subdomínio gratuito
- Servidor Web
  - É possível hospedar o website em servidor terceirizado (gratuito ou pago)
  - Também é possível contratar um serviço de VPS (*virtual private server*)
  - Outra possibilidade é montar um servidor dedicado próprio ou contratar um serviço que o ofereça
    - Necessário instalar servidor HTTP como Apache HTTP, NGINX, Microsoft IIS etc.

Colocando o Website Online com o infinityfree

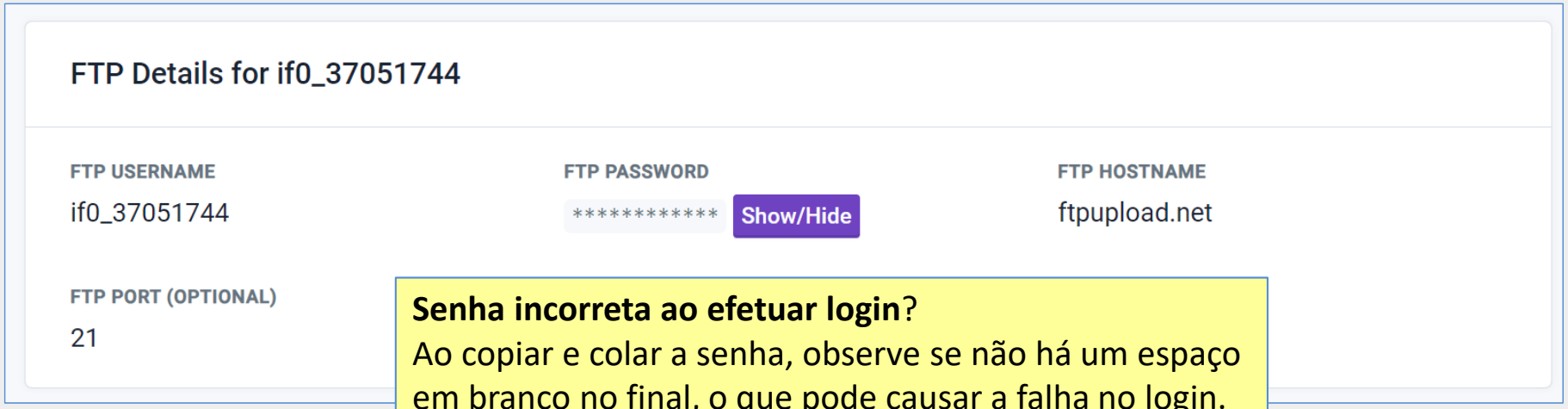
1. Crie uma conta gratuita em www.infinityfree.net e faça login
2. Crie uma nova conta de hospedagem e registre um subdomínio (slide a seguir)
3. Acesse os dados para conexão por FTP
4. Envie os arquivos do website utilizando um software cliente FTP
5. Acesse o website no navegador utilizando o nome de subdomínio criado

infinityfree – Criação de conta e subdomínio gratuito

1. Depois de logar, acesse **Accounts** → **Create Account**
2. Escolha o plano **gratuito**
3. Digite um nome para o **subdomínio** (por ex. trabalhosppi, fulanoppi etc.)
4. Escolha uma das opções para o **domínio** (por ex. infinityfreeapp.com)
5. Verifique a **disponibilidade** e vá para o próximo passo
6. Na etapa de **informações adicionais**, escolha uma opção para **Email Consent.** Não é necessário alterar os demais campos.
7. Finalize a criação clicando em **Create Account**.

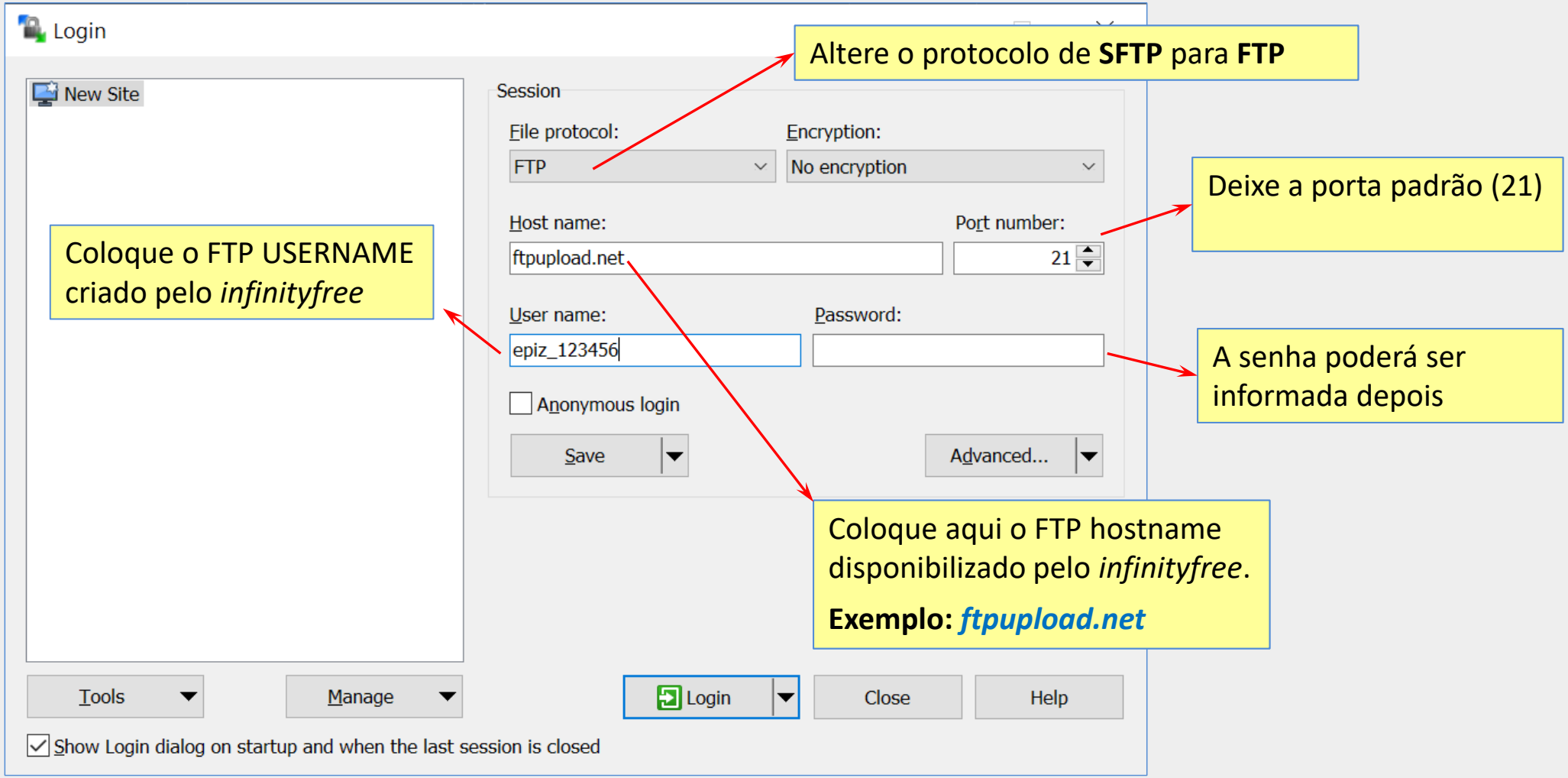
infinityfree – Acessando dados para conexão por FTP

1. Depois de criar a conta, clique novamente em **Accounts**
2. Clique no link relativo à conta criada
3. No painel à esquerda, clique em **FTP Details**
4. Observe as informações apresentadas (FTP Username, FTP Password e FTP Hostname). Elas serão necessárias para se conectar ao servidor do infinityfree e enviar os arquivos das páginas web.



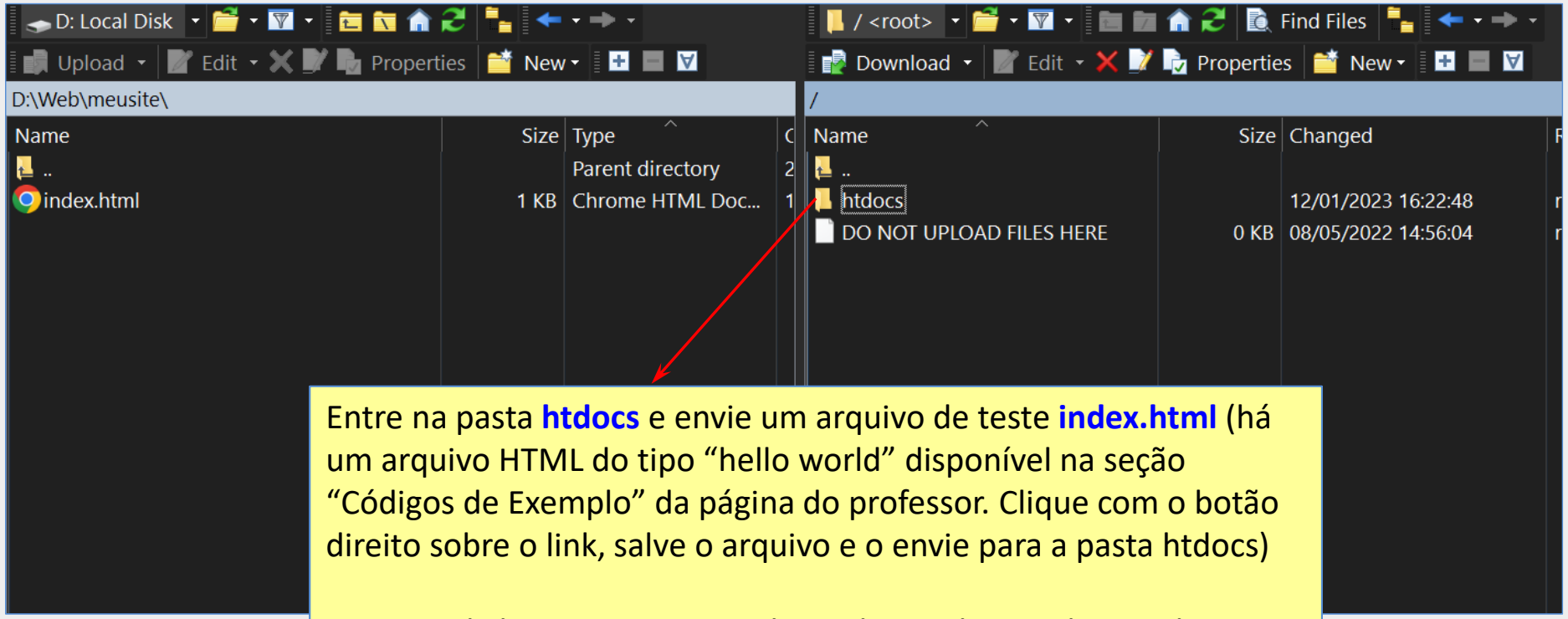
Senha incorreta ao efetuar login?
Ao copiar e colar a senha, observe se não há um espaço em branco no final, o que pode causar a falha no login.

Envio dos Arquivos por FTP com o WinSCP



Para colocar o website online, precisamos de um programa cliente FTP. Usuários do Windows podem baixar e instalar gratuitamente o **WinSCP**. Para usuários de outros sistemas, há alternativas como o **FileZilla**.

Envio dos Arquivos com o WinSCP



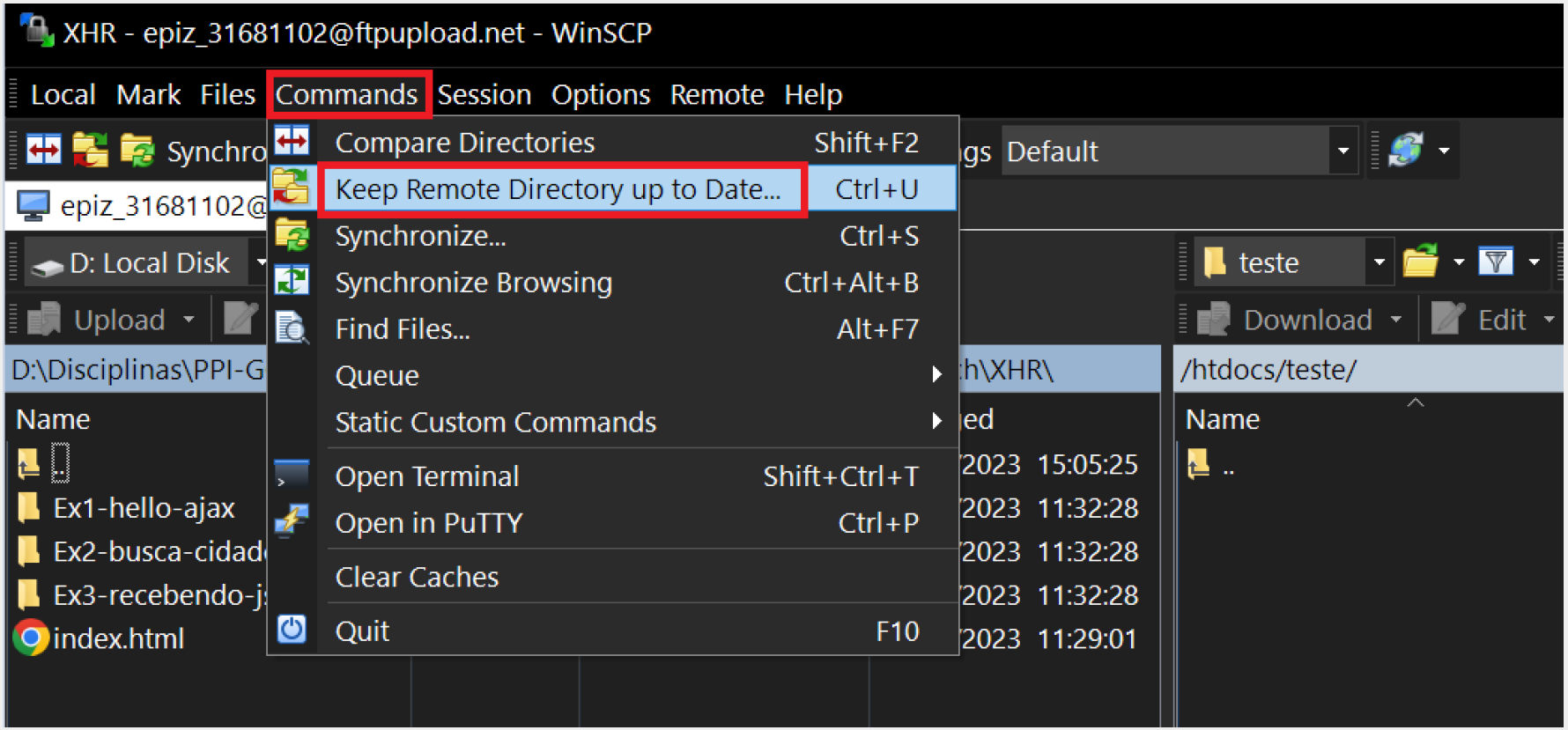
Entre na pasta **htdocs** e envie um arquivo de teste **index.html** (há um arquivo HTML do tipo "hello world" disponível na seção "Códigos de Exemplo" da página do professor. Clique com o botão direito sobre o link, salve o arquivo e o envie para a pasta htdocs)

Em seguida basta acessar o website digitando o endereço de cadastro no navegador (seusite.infinityfreeapp.com)

Enviando Arquivos Automaticamente

- Durante o desenvolvimento do website é possível abrir a pasta raiz localmente na ferramenta de desenvolvimento (Visual Studio Code) e configurar o WinSCP para enviar os arquivos automaticamente quando eles forem salvos localmente
- Passos:
  1. Depois de se conectar com o WinSCP, escolha a pasta do website no servidor;
  2. Escolha a pasta do website localmente (lado esquerdo da janela). Essa pasta será monitorada pelo WinSCP;
  3. Acesse o menu Commands → Keep Remote Directory up to Date → Start
  4. Minimize a janela do WinSCP. A partir de agora, quando salvar um arquivo localmente no Visual Studio Code, ele será enviado automaticamente para o servidor

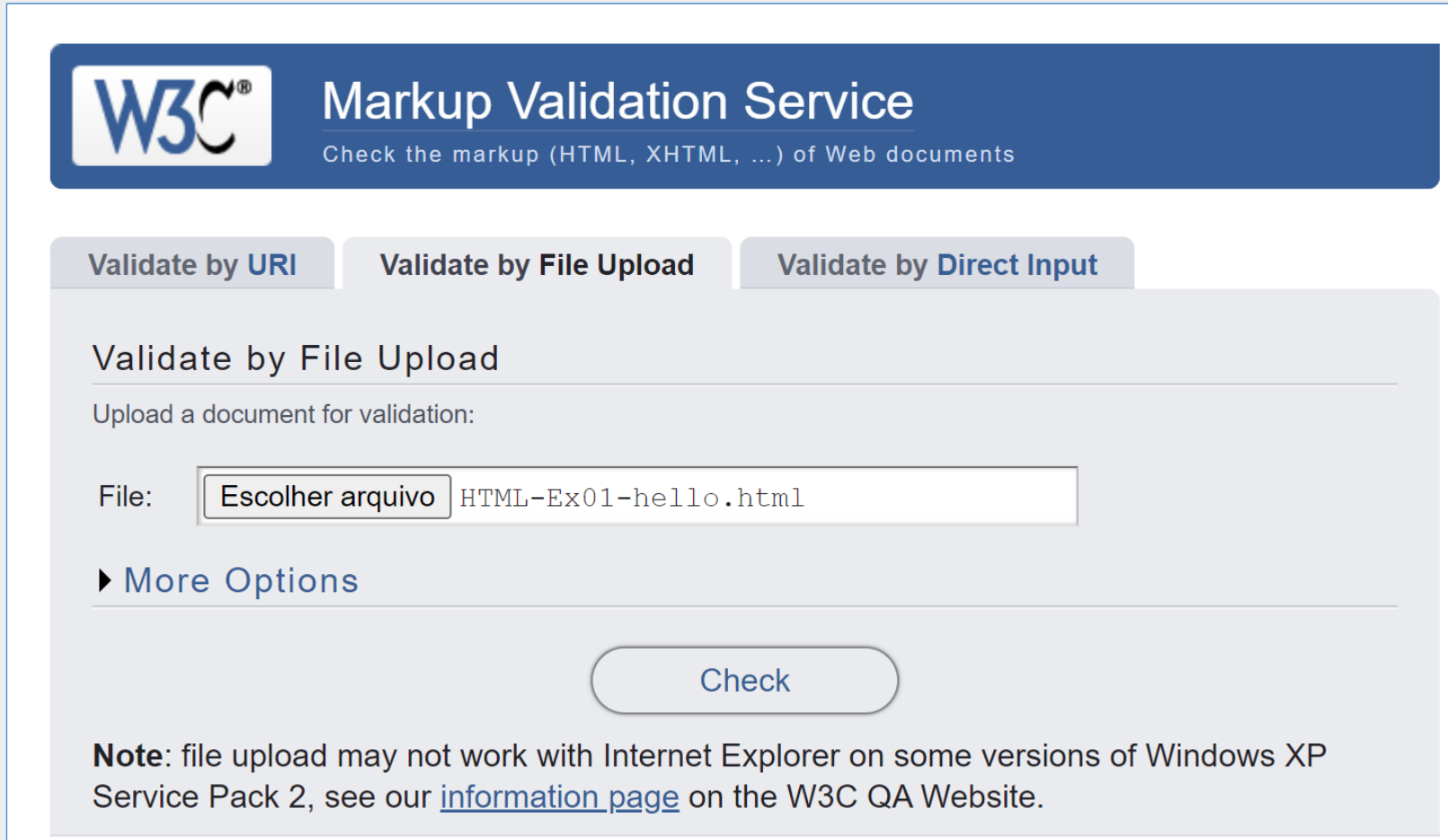
Enviando Arquivos Automaticamente



Validação da Página HTML

- Exibição adequada no navegador não é garantia de código correto
  - O navegador pode ocultar erros e inconsistências
- Documento fora da especificação
  - Apresentação inconsistente e imprevisível nos navegadores
- Serviço oferecido pelo W3C para validação de código HTML5
  - validator.w3.org

Validação da Página HTML



Atenção: Ao validar uma página hospedada no **infinityfree**, prefira a validação por **upload de arquivo** ao invés de validar diretamente pelo endereço online, pois o serviço de hospedagem do infinityfree pode alterar o conteúdo HTML que é enviado ao navegador, inserindo erros que não existem de fato no arquivo HTML original.

Visual Studio Code – Atalhos Essenciais

- Aplicar a formatação e indentação do código
  - SHIFT + ALT + F (Windows) ou CTRL + SHIFT + i (Linux)
- Mover um bloco de código para cima ou para baixo
  - Depois de selecionar, utilize ALT + ↑ ou ALT + ↓
- Copiar um bloco de código para cima ou para baixo
  - Depois de selecionar, utilize ALT + SHIFT + ↑ ou ALT + SHIFT + ↓
- Edição de múltiplas linhas simultaneamente
  - Segure a tecla ALT e clique nos pontos a serem editados
  - Ou segure CTRL + ALT e utilize as setas
- Comentar e descomentar múltiplas linhas
  - Depois de selecionar, utilize CTRL + ;
- Abrir a Command Pallete e buscar por arquivos e comandos
  - Depois de selecionar, utilize CTRL + P

Visual Studio Code – Extensões

- Visualizar página HTML em tempo real
  - Extensão Live Preview da Microsoft
- Autocomplete para JavaScript
  - Extensão IntelliCode da Microsoft
- Renomear tag de fechamento junto com tag de abertura
  - Extensão Auto Rename Tag
- Autocomplete e **indentação** para código PHP e outros recursos
  - Extensão PHP Intelephense

CSS

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 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

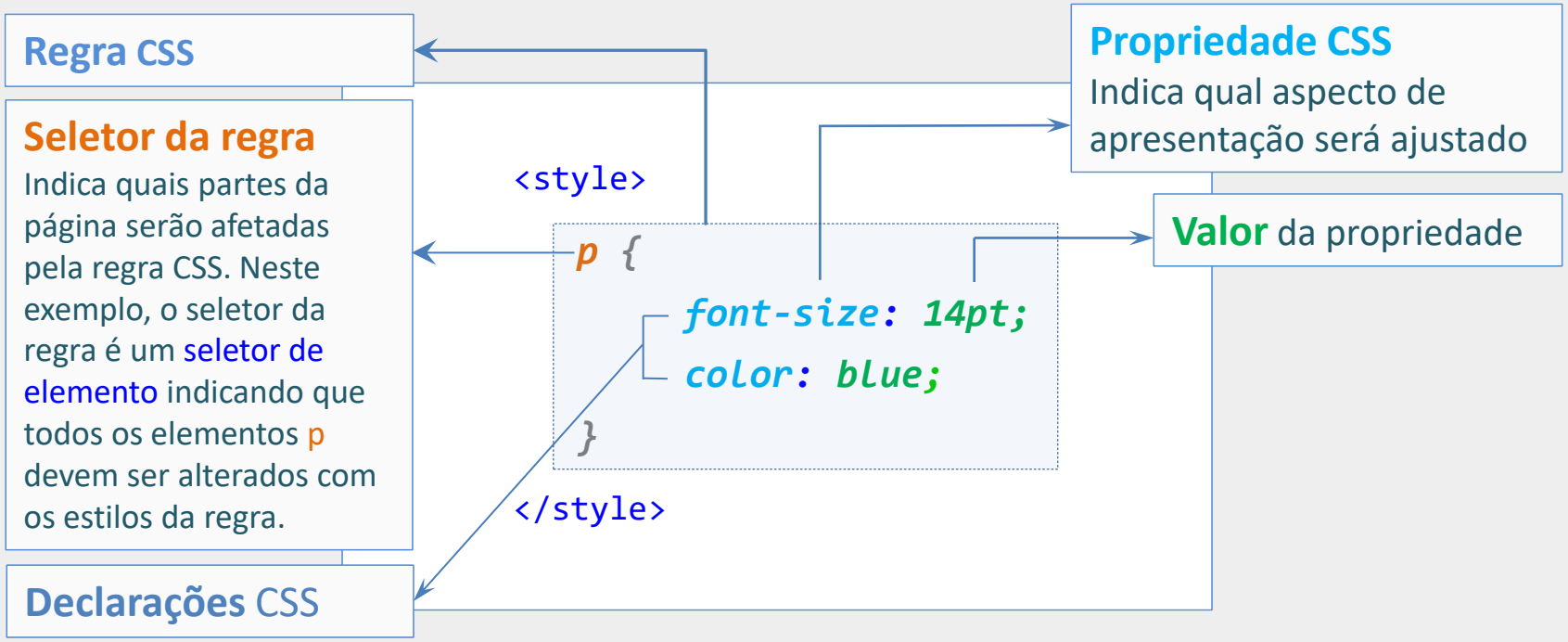
```
/* Arquivo style.css */

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.

Dentro de um arquivo CSS é possível importar o código CSS de outro arquivo utilizando a diretiva: @import url("outro-arquivo.css")

Regra, Seletor e Propriedades

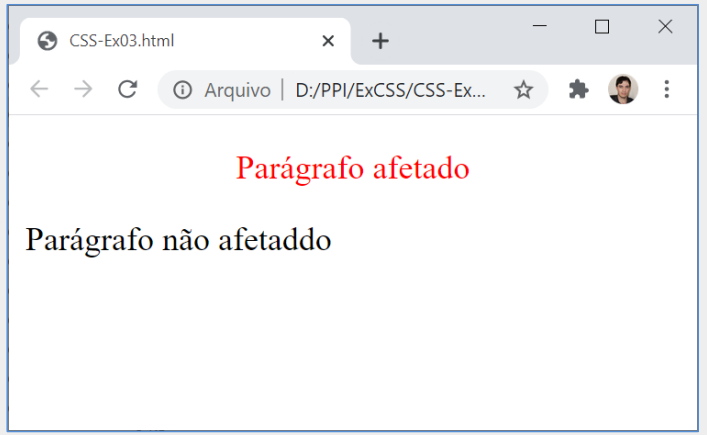


Seletor de ID (*#idDoElemento*)

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 , **direta ou indiretamente** (Link 2 e Link 3)

Se o seletor de filho tivesse sido usado neste exemplo (div > a), 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 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 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 com Pseudo-Classe

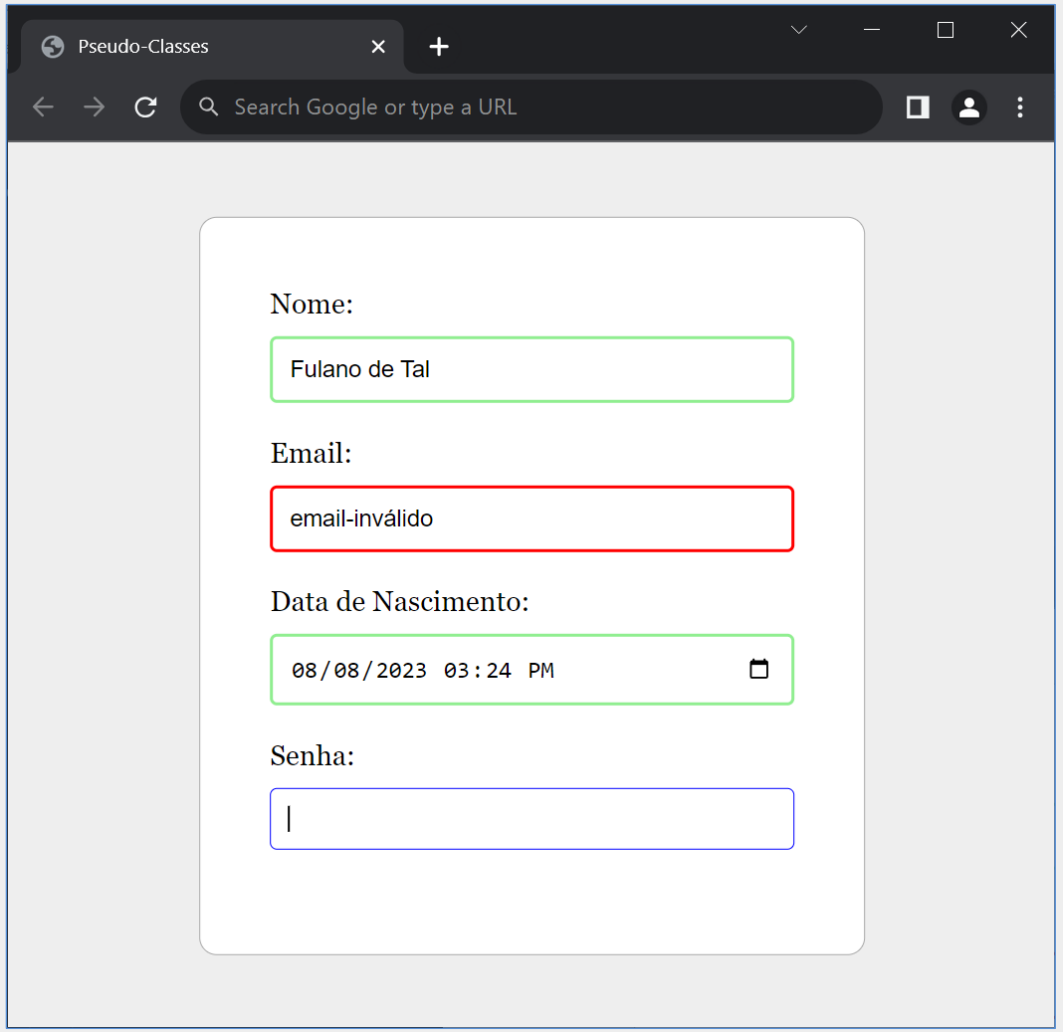
- Uma pseudo-classe permite alterar o estilo de um elemento **caso ele esteja em um estado particular**
- Por exemplo, é possível alterar o estilo dos links que já foram visitados ou o estilo dos campos de formulário com conteúdo inválido
- Sintaxe: **seletor : pseudo-classe**

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;
}
```



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)

Qual é a diferença entre div:has(> img) com div > img?
Resposta: O primeiro afeta os **div's** que possuem um **img** como filho. O segundo afeta as **imagens** que são filhas de div's.

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

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

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

Pseudo-Elementos ::after e ::before

- O pseudo-elemento `::after` representa um espaço para conteúdo localizado depois do conteúdo normal do elemento (mas ainda dentro do elemento)
- O pseudo-elemento `::before` representa um espaço antes do conteúdo
- Normalmente eles são utilizados em conjunto com a propriedade **content** para inserir conteúdo nesses espaços com CSS

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

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

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

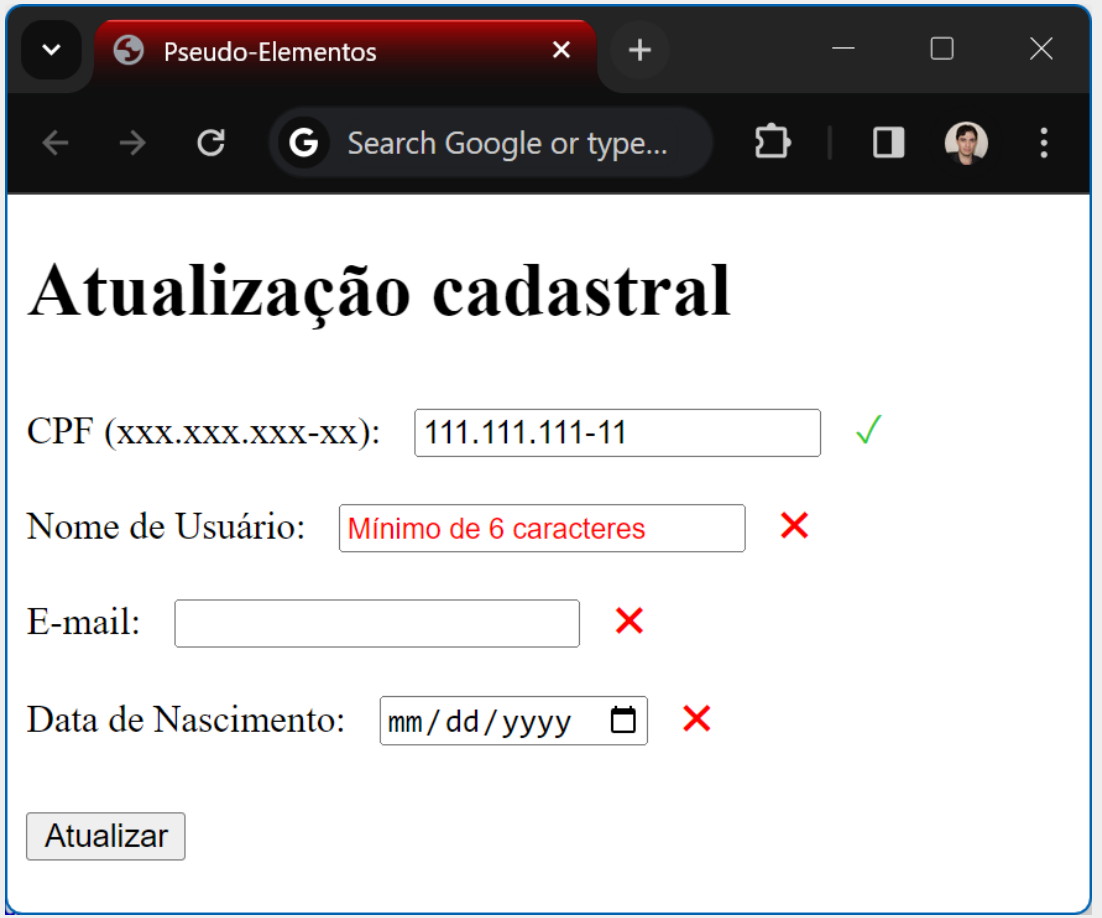
Insere o texto 'exemplo before' como um pseudo-elemento **antes do conteúdo** do parágrafo

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

Pseudo-Elementos – Exemplo Prático

```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

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 realizada 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.

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

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 (**v**iewport **h**eight)

- 30vh corresponde a 30% da altura da viewport

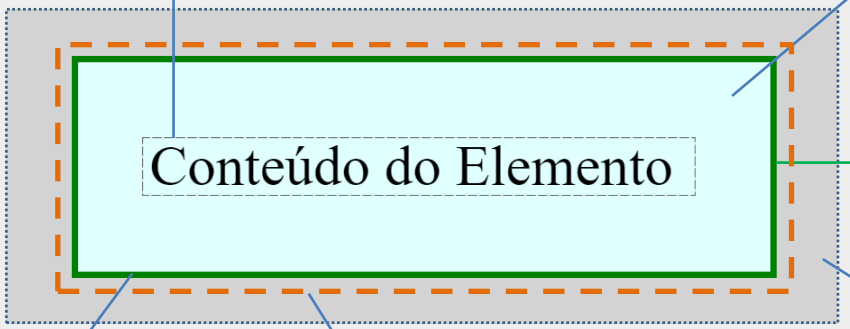
vw – relativo à largura da viewport (**v**iewport **w**idth)

- 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

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 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.)

Propriedade box-sizing

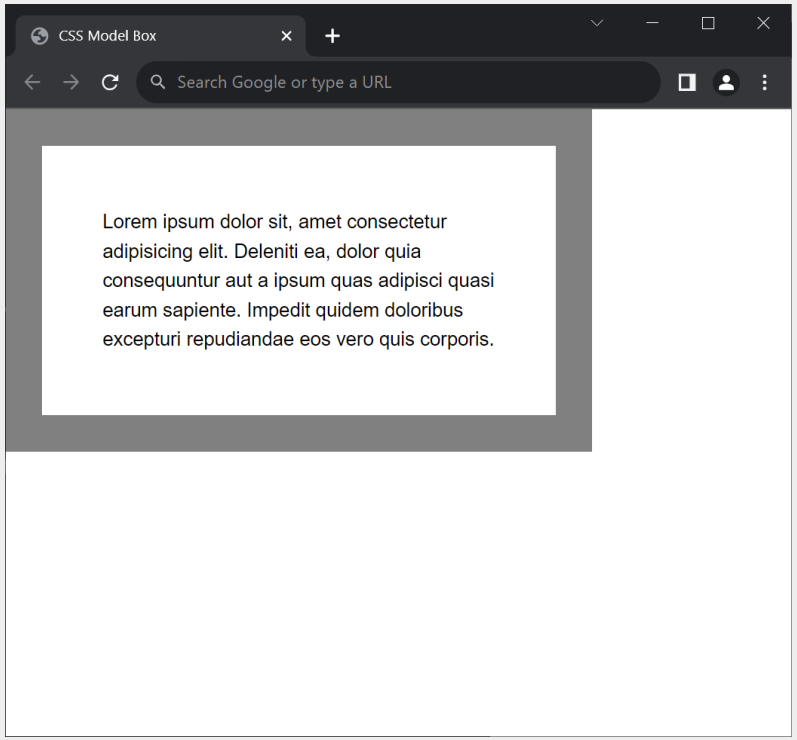
- Altera o modo em que a largura e altura do elemento é calculada
- box-sizing: content-box
  - Valor padrão para a maioria dos elementos (conforme exemplo anterior)
  - Ao definir width, estamos definindo a largura da região de conteúdo (content box)
  - A largura total do elemento será: width + bordas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

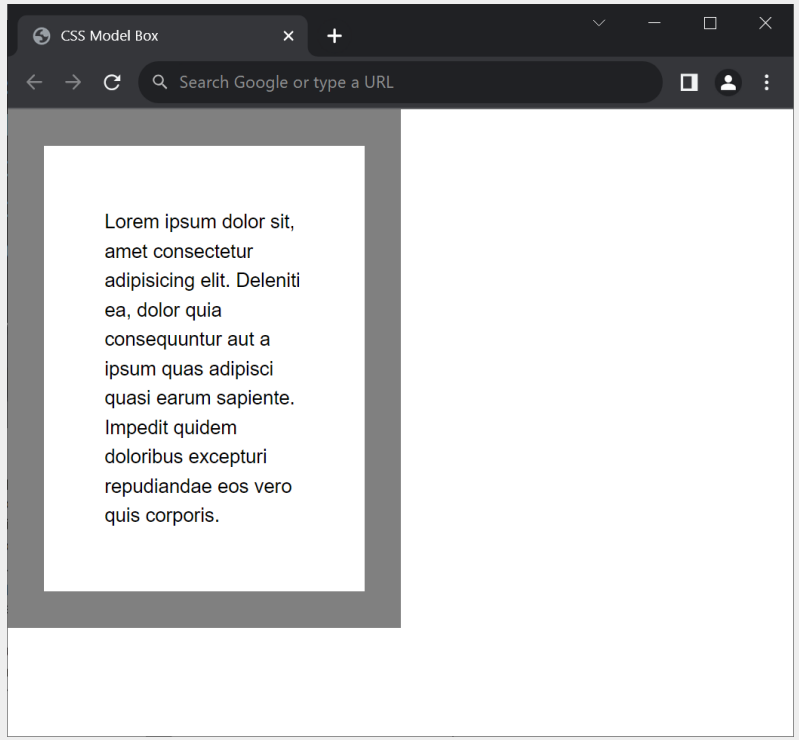
Com box-sizing: content-box, "width: 50%" faz com que a região de conteúdo (texto) ocupe 50% da largura do container, mas a largura total do elemento (incluindo paddings, bordas e margens) ultrapassa a metade do container.

Este é o mecanismo padrão utilizado pela maioria dos elementos.

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



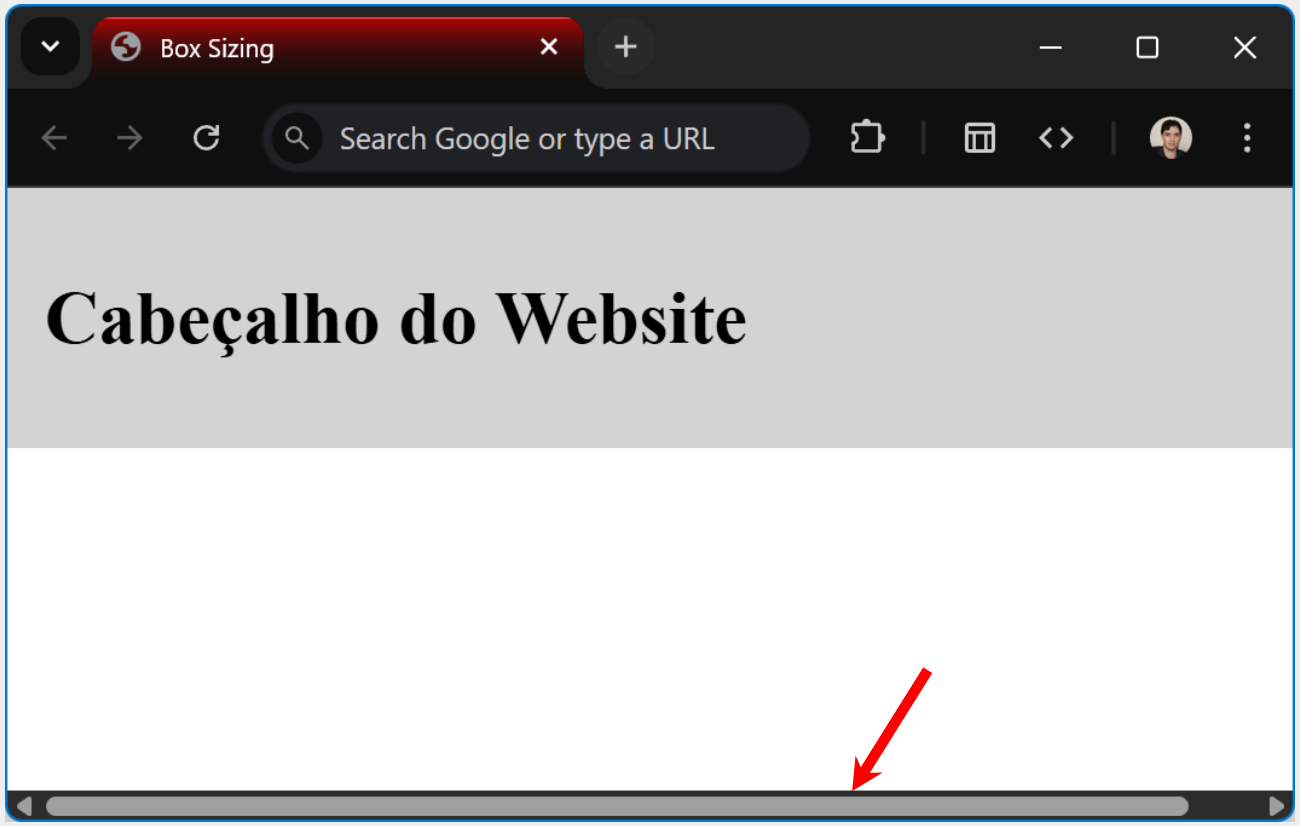

```
.box {
  box-sizing: border-box;
  width: 50%;
  margin: 0px;
  padding: 50px;
  border: 30px solid gray;
}
```



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

Por que aparece a barra de rolagem horizontal neste exemplo?

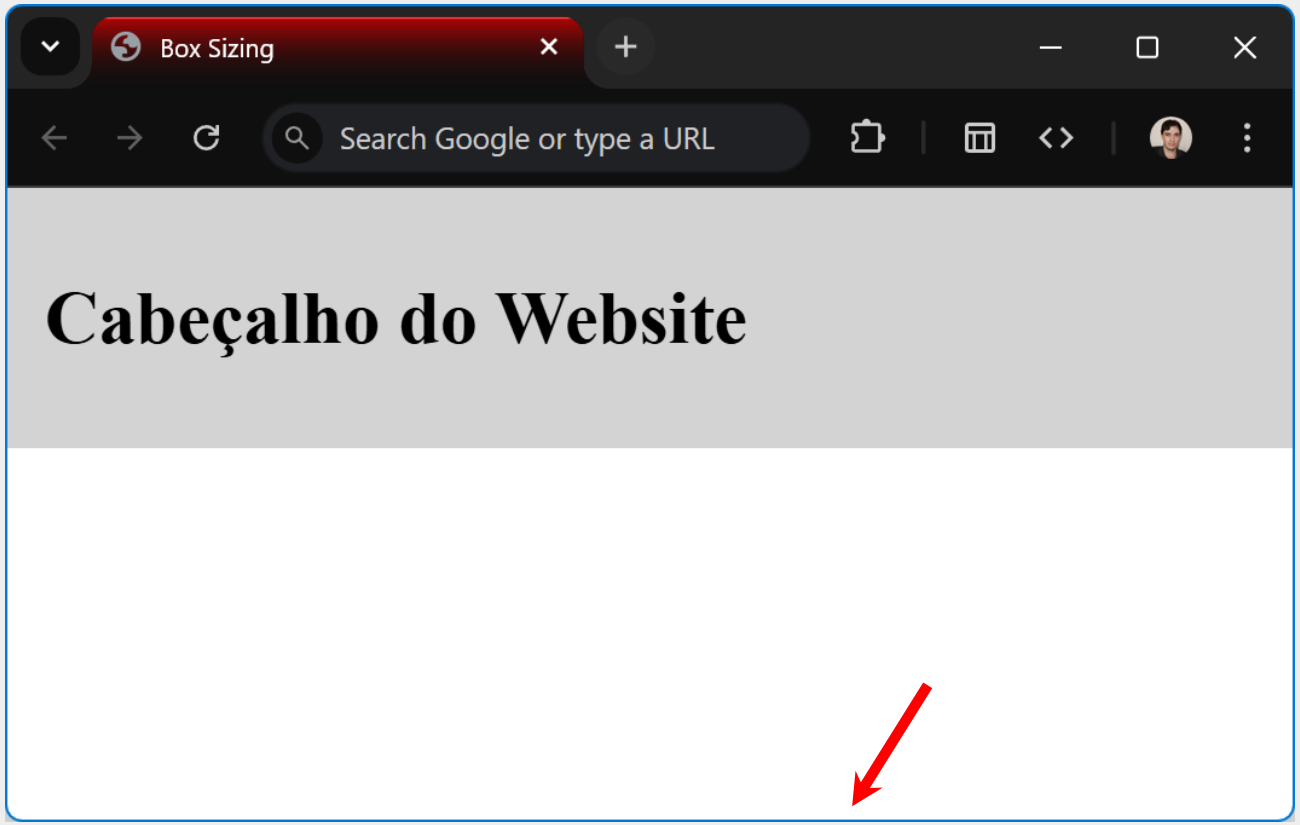
```html
<html>
<head>
  <title>Box Sizing</title>
  <meta charset="UTF-8">
  <style>
    body {
      margin: 0;
    }
    header {
      width: 100%;
      padding: 1rem;
      background-color: lightgrey;
    }
  </style>
</head>
<body>
  <header>
    <h1>Cabeçalho do Website</h1>
  </header>
</body>
</html>
```



Observe que foi definido um padding para o elemento header, assim como uma largura de 100%. Como o elemento utiliza box-sizing: content-box, sua largura total irá ultrapassar a largura da viewport quando for acrescida a largura do padding. Por esse motivo aparece a indesejada barra de rolagem horizontal.

Solução: basta alterar o box-sizing do header

```
<head>
  <title>Box Sizing</title>
  <meta charset="UTF-8">
  <style>
    body {
      margin: 0;
    }
    header {
      box-sizing: border-box;
      width: 100%;
      padding: 1rem;
      background-color: lightgrey;
    }
  </style>
</head>
<body>
  <header>
    <h1>Cabeçalho do Website</h1>
  </header>
</body>
</html>
```



Após alterar o box-sizing para border-box, a largura de 100% para o elemento header passa a ser aplicada considerando toda a **caixa da borda** e não mais ultrapassa a largura horizontal da viewport.

Propriedade display

display: none

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

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

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

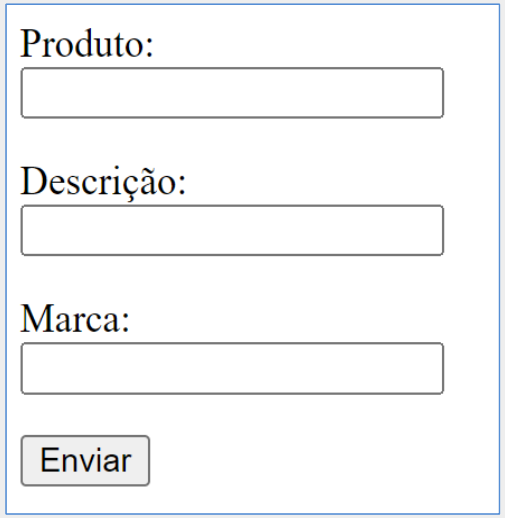
display: inline-block

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

Exibindo campos em nova linha com display: block

```html
<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>
```

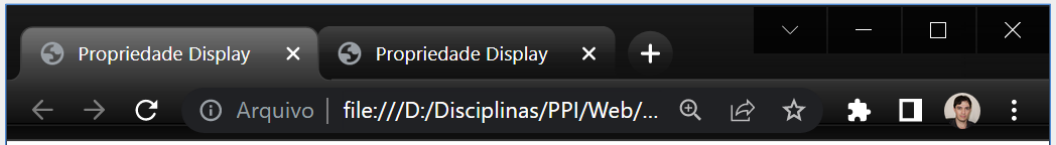
```css
input {
  display: block;
  margin-bottom: 1rem;
}
```



Os campos input tem exibição padrão em nível de linha, aparecendo 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
.

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>
```



Menu de navegação

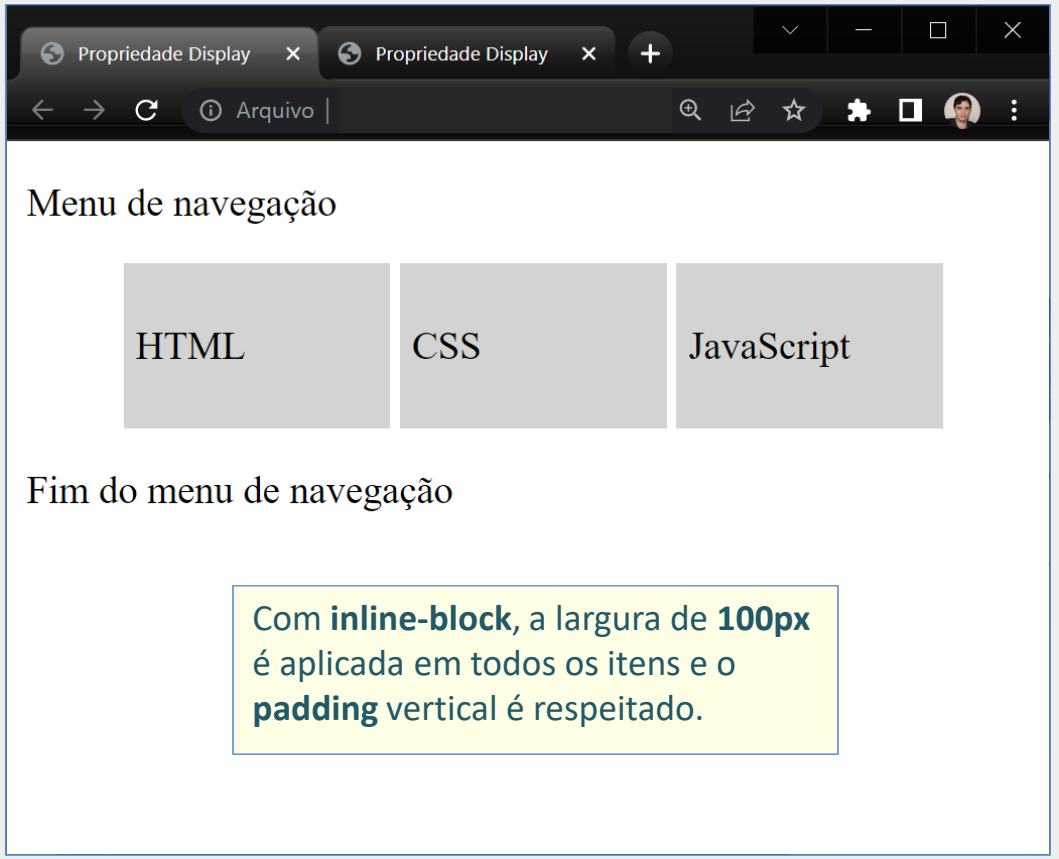
HTML CSS JavaScript

Fim do menu de navegação

Os itens de lista (<li>) são exibidos normalmente um abaixo do outro, pois são elementos de **bloco** por padrão. Neste exemplo, a exibição é alterada com **display: inline**. Observe que os itens são exibidos na mesma linha, porém a largura de 100px definida para cada item não é respeitada. O padding vertical também não é respeitado, pois o fundo cinza aparece em cima do parágrafo anterior e do posterior.

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 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 - continuação

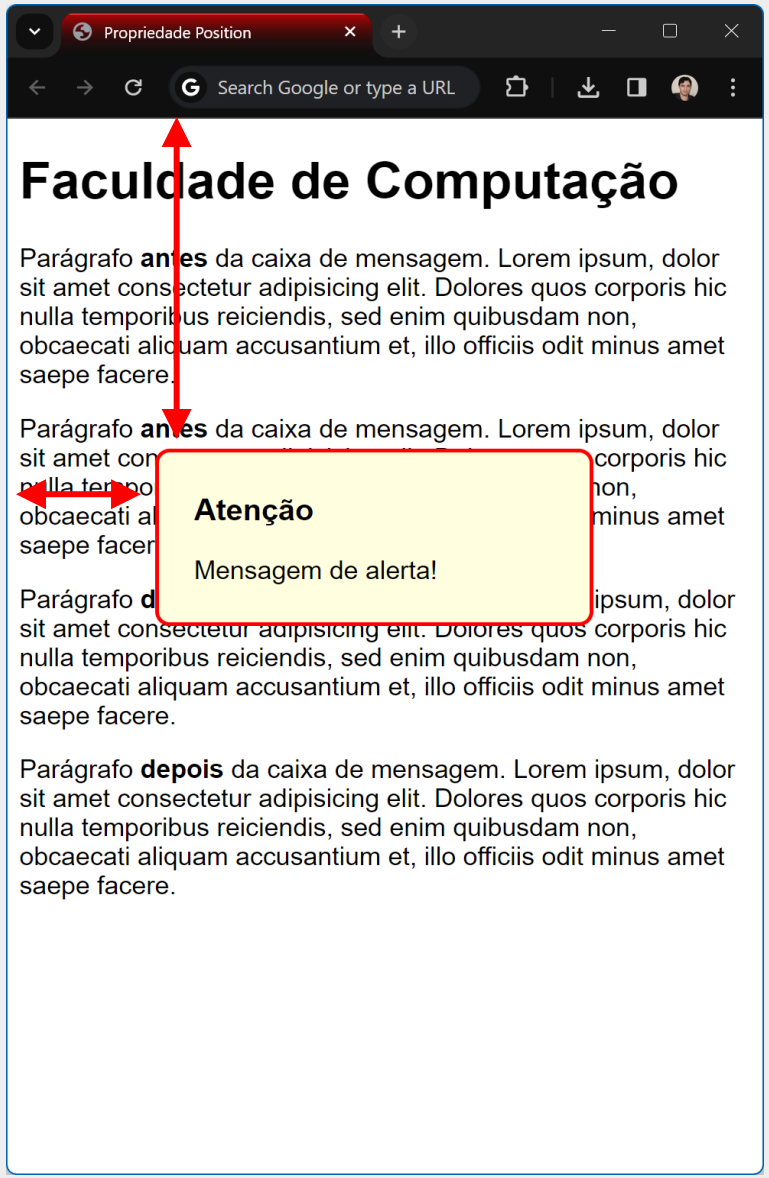
position: absolute

- O elemento é removido do fluxo normal - portanto, **não ocupa espaço no layout**
- O elemento é posicionado **sobre** o restante do conteúdo utilizando as propriedades top, left, right e/ou 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: 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.

Propriedade position - continuação

position: fixed

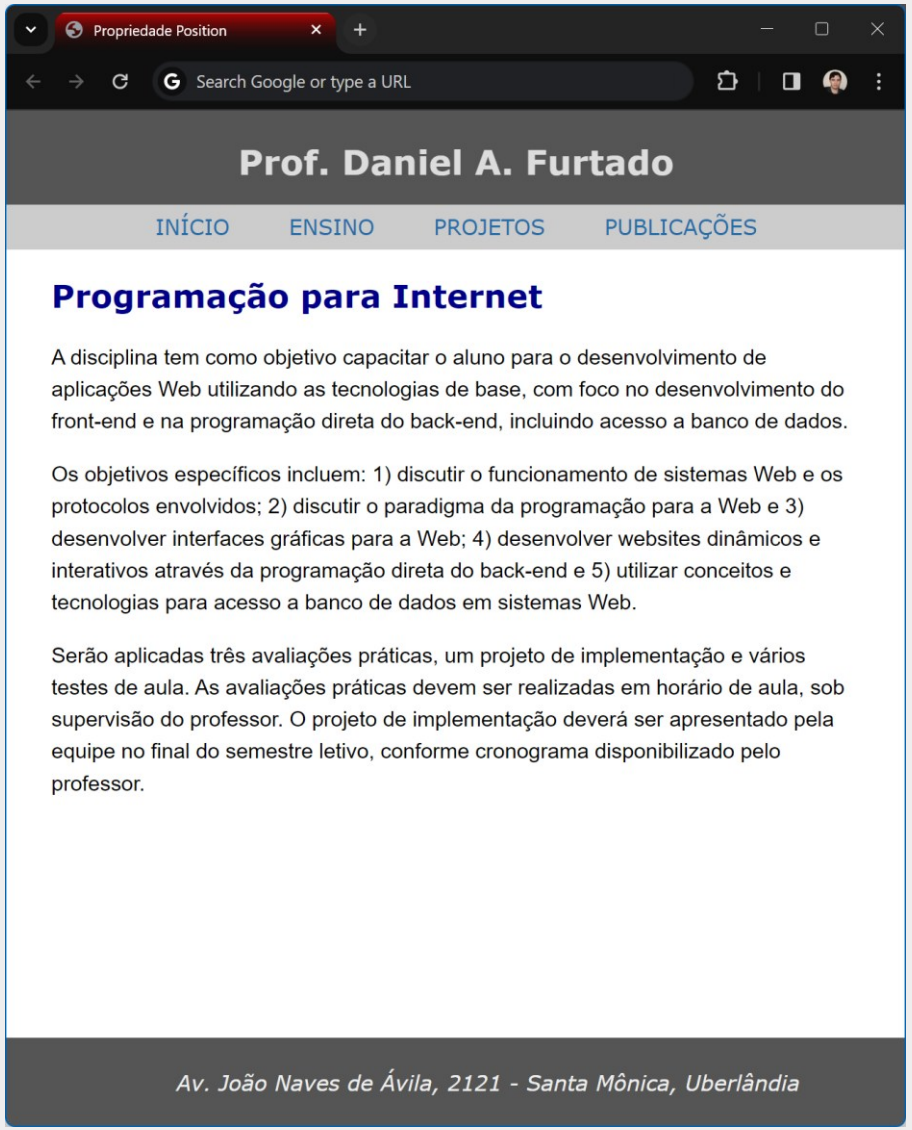
- Elemento **removido do layout** e posicionado com relação à **viewport** utilizando top, left, right e bottom
- O elemento é posicionamento **sobre** os demais
- A posição do elemento **não se altera com a rolagem** da página
- Quando impresso, aparecerá na mesma posição em todas as páginas
- Geralmente 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.

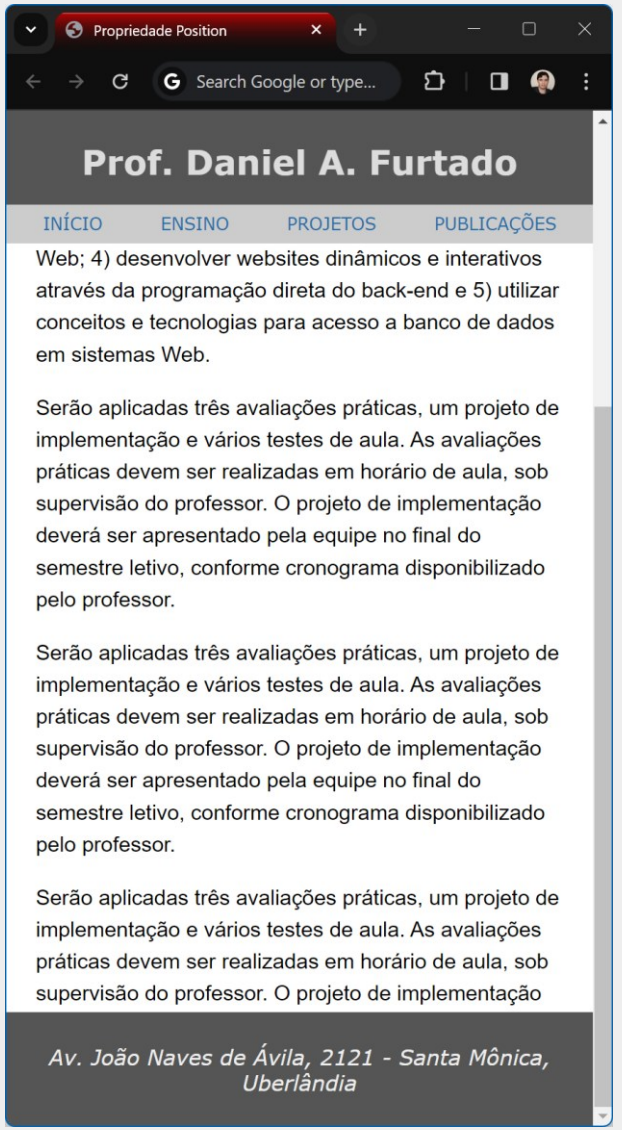
Exemplos disponíveis em https://youtu.be/caJ7Q65aiLE?t=2200

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 o rodapé permanece fixado na base da janela mesmo após rolagem da página, podendo ocultar o restante do conteúdo propriamente dito (segunda imagem).





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
- Quando utilizada em conjunto com posicionamento absoluto/fixo, a propriedade transform permite centralizar totalmente um elemento na página/viewport
  - Porém o elemento centralizado é **removido do layout** e aparecerá **sobre** os demais

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%
Neste caso, os percentuais são relativos à altura e largura da viewport, respectivamente.

```
...
<section class="caixaFacom">
  ...
</section>
...
```


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

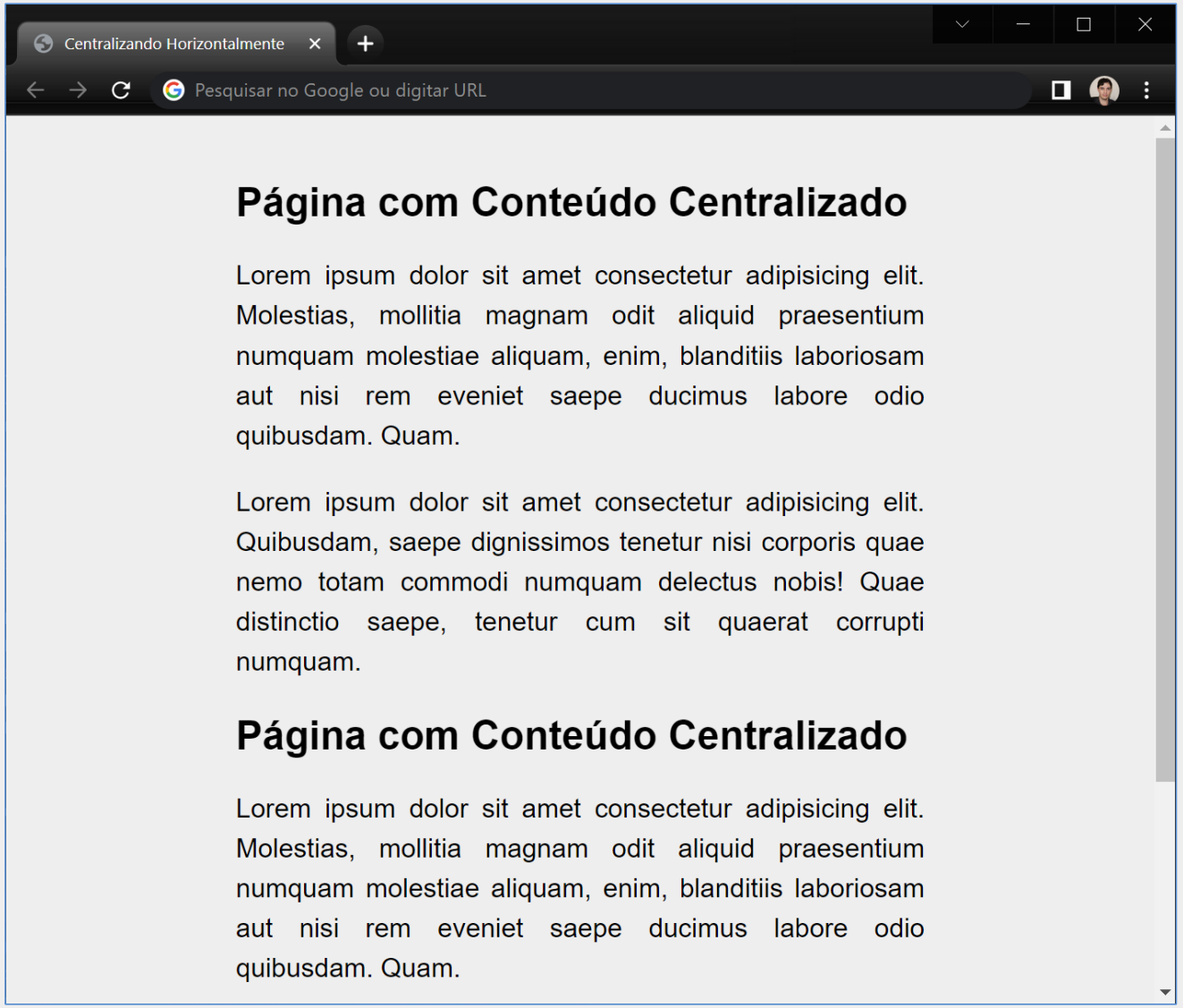
OBS 2: Com position: absolute a centralização vertical será perdida com a rolagem da página. Porém ela permanecerá caso seja utilizado position: fixed.

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

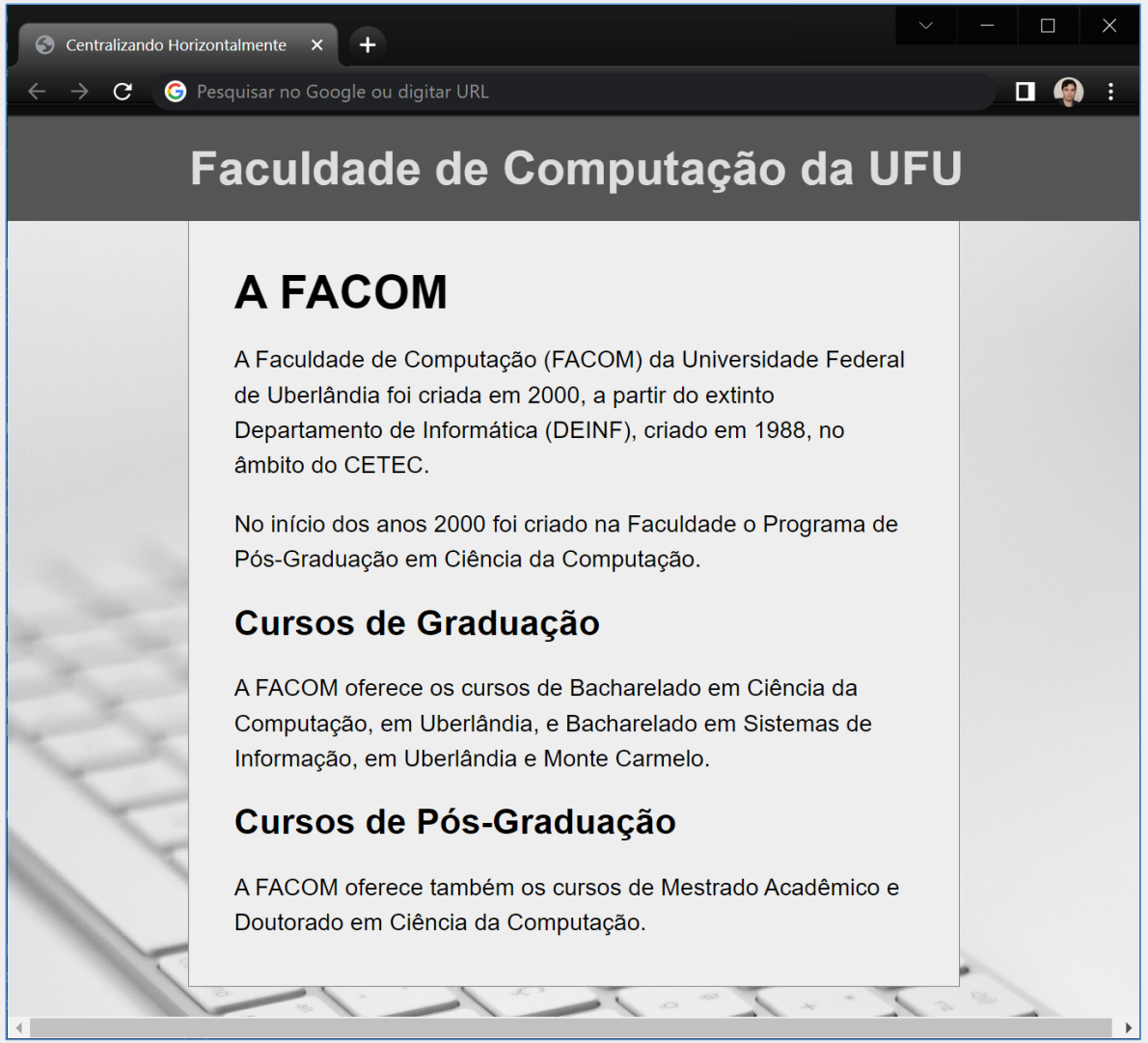
Centralizando na Horizontal com width e margin

```
  <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
```

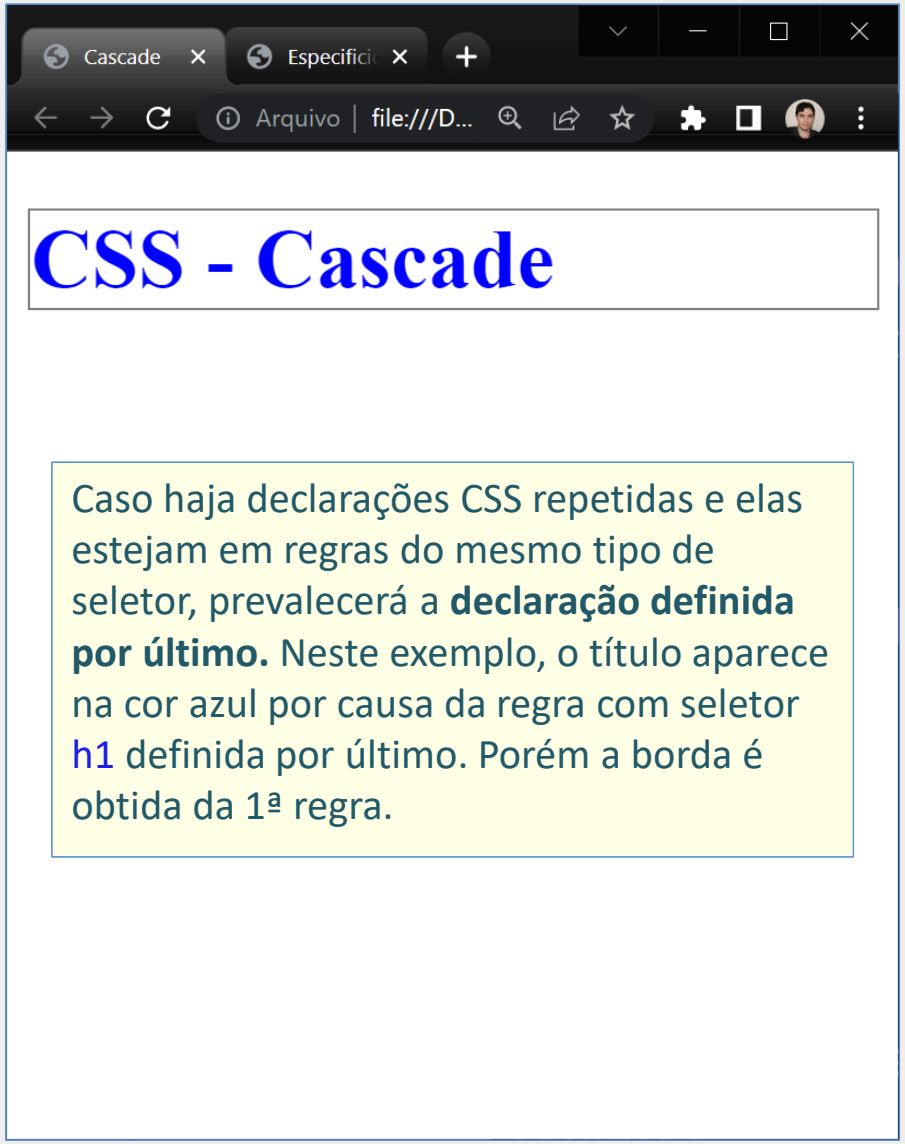


Cascade, Especificidade e Herança

- 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
- Vários fatores são considerados:
  - 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

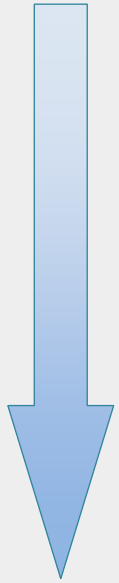
Mais geral

Seletor de Elemento
(e pseudo-elemento)

Seletor de Classe
(e pseudo-classe)

Seletor de ID

Código Inline



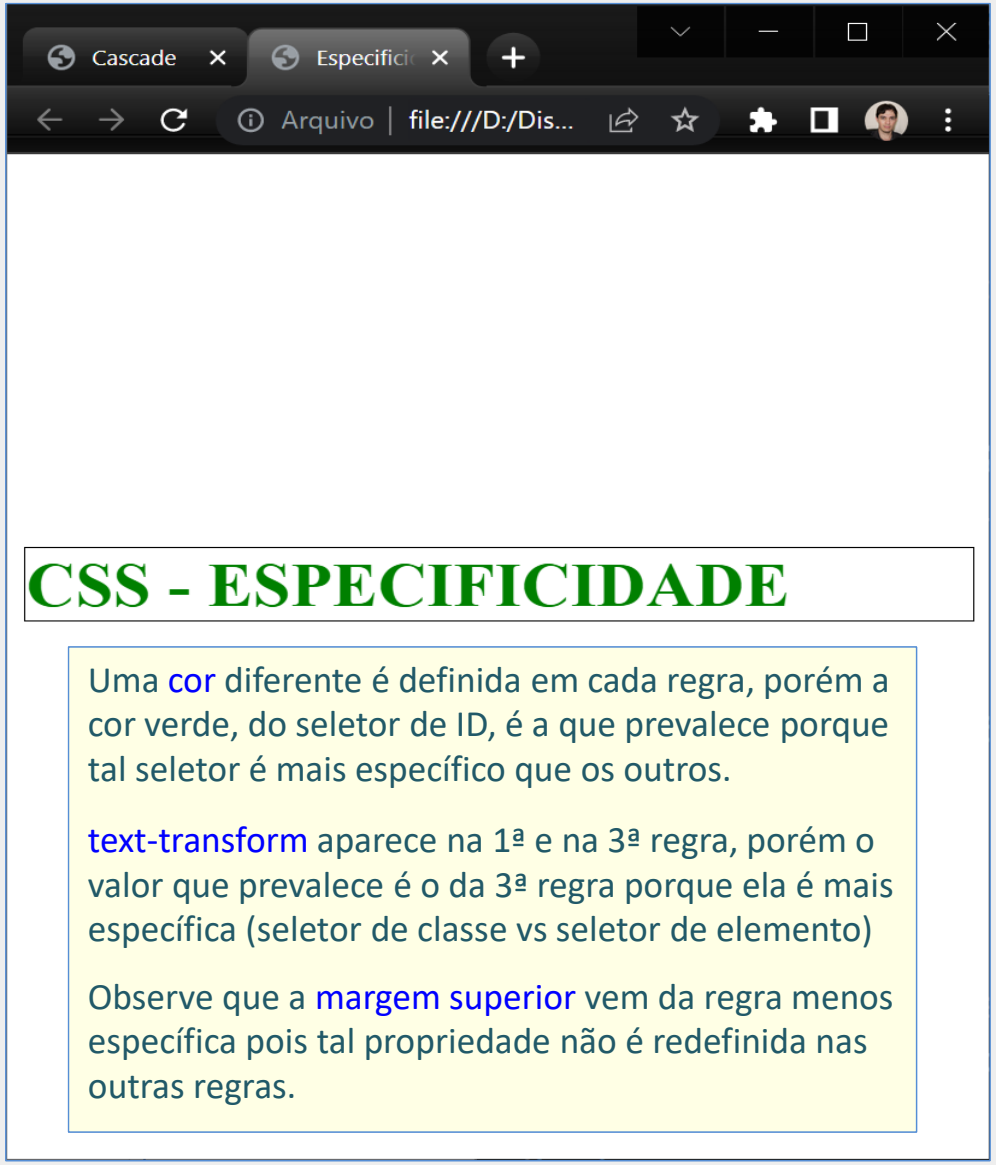
Maior especificidade implica em possível sobreposição de outras declarações.
Código inline **>** seletor de ID **>** seletor classe **>** seletor de elemento

Mais específico

Propriedades dentro de regras mais específicas tem maior **peso**, e portanto poderão sobrepor as propriedades de regras menos específicas. Um seletor de ID, por exemplo, é mais específico que um seletor de classe.

Especificidade das Regras - Exemplo

```
  <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>
```



Referências

- https://developer.cdn.mozilla.net/en-US/docs/Web/HTML
- https://html.spec.whatwg.org/multipage/
- https://www.w3.org/TR/html52/
- https://www.w3schools.com/html/
- **HTML and CSS: Design and Build Websites**, Jon Duckett.