



<> Estudo de HTML

O que é HTML?

HTML (HyperText Markup Language) é a linguagem usada para definir a estrutura de páginas web. Ela utiliza **tags** para organizar títulos, parágrafos, links, imagens, listas e muito mais.

Define a **estrutura** e o **conteúdo** da página.

Estrutura básica de um documento HTML

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Meu Primeiro Site</title>
</head>
<body>
  <h1>Olá, mundo!</h1>
  <p>Este é meu primeiro site em HTML.</p>
</body>
</html>
```

Page Title (<title>)

A tag **<title>** define o título da aba do navegador. Ela fica dentro da seção **<head>** do documento HTML. Esse título também aparece ao compartilhar links e é importante para SEO (motores de busca).

✅ Exemplo:

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8" />
  <title>Meu Primeiro Site</title>
</head>
<body>
  <h1>Bem-vindo!</h1>
</body>
</html>
```

💡 **Dica:** Escolha um título descritivo e curto! Ele ajuda na usabilidade, nos resultados do Google e na identificação da aba aberta no navegador.

📁 Estrutura Semântica do HTML5

- 📁 **<header>** — Cabeçalho da página ou seção
- 📁 **<main>** — Conteúdo principal do site (apenas um por página)
- 📁 **<footer>** — Rodapé da página (contatos, direitos autorais)
- 📁 **<section>** — Seção temática de conteúdo (capítulos, áreas do site)
- 📁 **<article>** — Conteúdo independente e reutilizável (post de blog, card, notícia)
- 📁 **<picture>** — Agrupa imagens responsivas (diferentes resoluções ou formatos)
- 📁 **<div>** — Bloco genérico, sem significado semântico (usado para agrupar conteúdo)

📁 Principais tags HTML

- 📁 **<h1> a <h6>** – Títulos

Exemplo:

```
<h1>Título Principal</h1>
      <h2>Subtítulo</h2>
      <h3>Seção menor</h3>
```

- 📁 **<p>** – Parágrafos

Exemplo:

```
<p>Lorem ipsum dolor sit amet.</p>
```

<a> – Links

Exemplo:

```
<a href="https://www.w3schools.com">Visitar W3Schools</a>
```

 – Imagens

Exemplo:

```

```

Trabalhando com Imagens no HTML ()

A tag **** é usada para exibir imagens em páginas web. Ela é auto-fechável (não usa ``) e possui atributos essenciais como:

- ✓ **src** — Caminho da imagem (local ou URL externa);
- ✓ **alt** — Texto alternativo para acessibilidade e SEO;
- ✓ **width** e **height** — Largura e altura da imagem (px ou %);

Exemplo básico:

```

```

Por que o atributo **alt** é importante?

- ✓ Melhora a acessibilidade (leitores de tela conseguem descrever a imagem);
- ✓ Ajuda no SEO (Google entende o conteúdo da imagem);
- ✓ Aparece caso a imagem não seja carregada;

HTML x CSS — Como definir tamanho de imagens?

Você pode definir **width** e **height** no HTML, mas a melhor prática é usar CSS.






```

```

```
<style>  
.foto {  
    width: 300px;  
    height: auto;  
}
```

```
border-radius: 10px;  
}  
</style>
```

Formatos de imagens mais usados

-  **JPG / JPEG** — Fotos, mais leve, perde qualidade (compressão com perdas);
-  **PNG** — Imagens com transparência e qualidade alta (sem perdas);
-  **SVG** — Imagens vetoriais (logos, ícones, escaláveis sem perder qualidade);
-  **GIF** — Imagens animadas simples;
-  **WEBP** — Formato moderno do Google, mais leve, suporta transparência e animação;

Exemplo com imagens em diferentes formatos:

```
  
    
    
    
  
```




<figure> e <figcaption> — Imagens com legenda

Para adicionar uma legenda a uma imagem, usamos as tags **<figure>** e **<figcaption>**. Isso torna o HTML mais semântico e melhora acessibilidade.

```
<figure>  
    
  <figcaption>Figura 1: Logo oficial do HTML5</figcaption>  
</figure>
```

Listas no HTML

Listas são usadas para organizar conteúdos em itens. No HTML existem três tipos principais de listas:

-  **** — Lista não ordenada (com marcadores como ●, ○, ■)
-  **** — Lista ordenada (numerada: 1, 2, 3...)
-  **<dl>** — Lista de descrição (termo + definição)

📌 Exemplo de lista não ordenada ():

```
<ul>

    <li>Maçã</li>
    <li>Banana</li>
    <li>Laranja</li>
</ul>
```

📌 Exemplo de lista ordenada ():

```
<ol>

    <li>Primeiro passo</li>
    <li>Segundo passo</li>
    <li>Terceiro passo</li>
</ol>
```

📌 Exemplo de lista de descrição (<dl>):

```
<dl>

    <dt>HTML</dt>
    <dd>Linguagem de marcação para estruturar páginas web.</dd>

    <dt>CSS</dt>
    <dd>Usado para estilizar o conteúdo do HTML.</dd>
</dl>
```

💡 **Dica:** Todas as listas (ul, ol, dl) são compostas por itens. Para **ul** e **ol**, os itens são definidos com ****. Já nas listas de descrição **<dl>**, usamos **<dt>** (termo) e **<dd>** (descrição).

📊 Tabelas no HTML

Tabelas são usadas para organizar dados em linhas e colunas. No HTML, elas são criadas com a tag **<table>** e possuem uma estrutura básica composta por:

- ✅ **<table>** — Define a tabela;
- ✅ **<tr>** — Cria uma linha (table row);
- ✅ **<th>** — Cabeçalho da tabela (table header);
- ✅ **<td>** — Célula de dados (table data).

📌 Exemplo básico de tabela:

```
<table border="1">
  <tr>
    <th>Nome</th>
```

```
<th>Idade</th>
<th>Profissão</th>
</tr>
<tr>
<td>Ana</td>
<td>25</td>
<td>Desenvolvedora</td>
</tr>
<tr>
<td>João</td>
<td>30</td>
<td>Designer</td>
</tr>
</table>
```

📌 Tabela com <thead>, <tbody> e <tfoot>:

Essas tags ajudam a organizar melhor a tabela e tornam o código mais semântico:

```
<table border="1">
  <thead>
    <tr>
      <th>Produto</th>
      <th>Quantidade</th>
      <th>Preço</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Microscópio</td>
      <td>3</td>
      <td>R$ 5.000</td>
    </tr>
    <tr>
      <td>Béquer</td>
      <td>10</td>
      <td>R$ 15</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td colspan="3">Dados atualizados em 2025</td>
    </tr>
  </tfoot>
</table>
```

💡 **Dica:** O atributo `colspan` permite mesclar colunas, e o `rowspan` mescla linhas. A estilização é mais recomendada via CSS.

HTML Formatting Elements

**** – Negrito

```
<b>texto em negrito</b>
```

**** – Importante

```
<strong>texto importante</strong>
```

<i> – Itálico

```
<i>texto em itálico</i>
```

**** – Ênfase

```
<em>texto enfatizado</em>
```

<mark> – Texto destacado

```
<mark>texto destacado</mark>
```

Comentários em HTML

Comentários em HTML são usados para adicionar anotações no código que não aparecem no navegador. Eles são úteis para explicar partes do código ou desativar trechos temporariamente.

✓ **Sintaxe:** `<!-- comentário aqui -->`

✓ **Não são exibidos ao usuário,** apenas no código.

✓ **Podem ser usados para explicar ou organizar o HTML.**

Exemplo:

```
<!-- Este é um comentário em HTML -->
```










```
<h1>Bem-vindo!</h1>
```

```
<!-- Comentário para explicar a próxima seção -->
```

```
<p>Este é um parágrafo.</p>
```

HTML Forms (Formulários)

Formulários HTML são usados para coletar dados do usuário. Eles são compostos por campos de entrada como **<input>**, **<textarea>**, **<select>** e um botão de envio **<button>**. Todos esses elementos ficam dentro da tag **<form>**.

-  **<form>** — Contêiner onde todos os campos ficam.
-  **<input>** — Campo para texto, e-mail, senha, número etc.
-  **<label>** — Texto que nomeia cada campo.
-  **<textarea>** — Campo de texto com várias linhas.
-  **<button>** ou **<input type="submit">** — Botão de envio.
-  **<select>** — Menu suspenso (combobox)
-  **<option>** — Opções dentro de um select
-  **<fieldset>** — Agrupa elementos de formulário
-  **<legend>** — Título de um fieldset

Exemplo básico:

```
<form action="/enviar" method="post">
  <label for="nome">Nome:</label>
  <input type="text" id="nome" name="nome" required />

  <label for="email">Email:</label>
  <input type="email" id="email" name="email" required />

  <label for="mensagem">Mensagem:</label>
  <textarea id="mensagem" name="mensagem"></textarea>

  <button type="submit">Enviar</button>
</form>
```

✦ Você pode enviar esse formulário para um servidor usando o atributo **action=""** e definir o método com **method= "GET" ou "POST"** .

Métodos de Envio: GET e POST

Quando enviamos um formulário com **<form>**, usamos o atributo **method=""** para definir como os dados serão enviados ao servidor.



GET — Envia os dados pela URL.

Ideal para pesquisas, filtros ou formulários simples.

Exemplo de URL: `https://site.com/?nome=Joao&email=teste@mail.com`



POST — Envia os dados de forma “oculta” no corpo da requisição.
Ideal para dados sensíveis, como login, cadastro ou envio de mensagens.

Exemplo com GET:

```
<form action="/buscar" method="get">
  <label for="nome">Nome:</label>
  <input type="text" id="nome" name="nome" />
  <button type="submit">Buscar</button>
</form>
```

Exemplo com POST:

```
<form action="/enviar" method="post">
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" required />
  <label for="mensagem">Mensagem:</label>
  <textarea id="mensagem" name="mensagem"></textarea>
  <button type="submit">Enviar</button>
</form>
```

📌 **Dica:** use GET quando quiser apenas recuperar informações, e POST quando quiser enviar ou salvar dados.

📁 Tipos de `<input>` mais usados

O elemento `<input>` muda completamente sua função dependendo do valor do atributo `type=""`. Veja os mais comuns:

- ✓ **text** – Campo de texto comum
- ✓ **email** – Valida e-mails automaticamente
- ✓ **password** – Oculta os caracteres digitados
- ✓ **number** – Aceita apenas números
- ✓ **date** – Seleciona data no calendário
- ✓ **color** – Abre seletor de cores
- ✓ **file** – Permite enviar arquivos
- ✓ **radio** – Opções únicas (um grupo)
- ✓ **checkbox** – Marcar múltiplas opções

- ✓ **range** – Slider (barra ajustável)
- ✓ **submit** – Botão de enviar formulário

Exemplo prático:

```
<form>

  <label>Nome: <input type="text" name="nome" /></label>
  <label>Email: <input type="email" name="email" /></label>
  <label>Senha: <input type="password" name="senha" /></label>
  <label>Data de nascimento: <input type="date" name="data" /></label>
  <label>Foto de Perfil: <input type="file" name="foto" /></label>
  <label>Cor favorita: <input type="color" name="cor" /></label>

  <p>Sexo:</p>
  <label><input type="radio" name="sexo" value="m" /> Masculino</label>
  <label><input type="radio" name="sexo" value="f" /> Feminino</label>

  <p>Interesses:</p>
  <label><input type="checkbox" name="html" /> HTML</label>
  <label><input type="checkbox" name="css" /> CSS</label>
  <label><input type="checkbox" name="js" /> JavaScript</label>

  <label>Experiência (0 a 10): <input type="range" min="0" max="10" /></label>

  <button type="submit">Enviar</button>
</form>
```



Cores no CSS

No HTML, podemos aplicar cores utilizando CSS — seja diretamente na tag (com o atributo **style**) ou usando uma folha de estilo externa. Podemos alterar a cor do texto (**color**) ou do fundo (**background-color**).

✓ Exemplo prático:

```
<!DOCTYPE html>
<html>
<body>
  <div style="background-color:Tomato;">
    <h1 style="color:blue;">A Blue Heading</h1>
    <p style="color:red;">A red paragraph.</p>
  </div>
</body>
</html>
```

🎯 Como as cores podem ser escritas no CSS?

- ✓ **Nome da cor:** red, blue, tomato
- ✓ **Hexadecimal:** #FF5733 (mais usado no design)
- ✓ **RGB:** rgb(255, 99, 71)
- ✓ **RGBA (com transparência):** rgba(255, 99, 71, 0.5)
- ✓ **HSL:** hsl(9, 100%, 64%)

🌈 Exemplos de diferentes formatos:

```
<p style="color:tomato;">Cor por nome</p>
  <p style="color:#ff6347;">Cor Hexadecimal</p>
  <p style="color:rgb(255, 99, 71);">Cor RGB</p>
  <p style="color:rgba(255, 99, 71, 0.5);">RGB com transparência</p>
  <p style="color:hsl(9, 100%, 64);">Cor usando HSL</p>
```

💡 **Dica:** Evite usar o atributo `style=""` diretamente no HTML. O ideal é separar o CSS em um arquivo próprio para manter o código organizado.

🎨 Border e Background no CSS

Além de colorir textos, o CSS também permite modificar o **fundo** de elementos com `background-color` e criar bordas com `border`.

✅ Exemplo prático com background e border:

```
<!DOCTYPE html>
<html>
<body>

  <div style="background-color: lightblue; border: 2px solid navy; padding: 10px;">
    <h2 style="color: darkblue;">Título com borda</h2>
    <p style="color: darkred;">Parágrafo com fundo azul claro.</p>
  </div>

</body>
</html>
```

📌 Como funciona a propriedade border?

A sintaxe básica é:

```
border: [largura] [estilo] [cor];
```

- ✓ **Largura:** 1px, 2px, 5px...
- ✓ **Estilo:** solid, dashed, dotted, double
- ✓ **Cor:** red, #000, rgb(0,0,0)

```
div {  
  border: 3px dashed #ff4500; /* laranja tracejado */  
}
```

🎨 Como funciona o background-color?

Essa propriedade altera a cor do fundo de qualquer elemento HTML:

```
body {  
  background-color: #f0f0f0;  
}  
  
div {  
  background-color: rgba(255, 99, 71, 0.3); /* Tomato com transparência */  
}
```

💡 **Dica:** você pode combinar background e border juntos para criar cartões, caixas de destaque, botões e muito mais.

🎨 Ligando CSS ao HTML (Arquivo Externo)

Em vez de usar estilos diretamente nas tags com o atributo `style=""`, o ideal é criar um arquivo CSS separado e vinculá-lo (linkar) ao HTML. Isso deixa o código mais organizado e profissional.

✓ 1. Estrutura recomendada de arquivos:

```
projeto/  
├── index.html  
└── styles.css
```

✓ 2. Dentro do HTML, use a tag <link>:

```
<!DOCTYPE html>  
<html lang="pt-BR">  
  <head>  
    <meta charset="UTF-8">  
    <title>Meu Site</title>  
    <link rel="stylesheet" href="styles.css"> <!-- ✓ Link do CSS -->
```

```
</head>
<body>
<h1>Olá, mundo!</h1>
<p>Este é um exemplo com CSS externo.</p>
</body>
</html>
```

✓ 3. E no arquivo styles.css:

```
body {
  background-color: #f5f5f5;
  font-family: Arial, sans-serif;
}

h1 {
  color: blue;
}

p {
  color: red;
}
```

💡 **Dica:** a tag <link> sempre deve ser colocada dentro do <head> do HTML. Isso garante que o CSS seja carregado antes do conteúdo aparecer na tela.

⚙️ Conectando JavaScript com HTML

O JavaScript é a linguagem que adiciona interatividade às páginas web. Assim como o CSS, ele pode ser inserido de três formas principais:

✓ 1. JavaScript Inline (dentro da tag)

Usado para ações simples, direto no HTML:

```
<button onclick="alert('Olá!')">Clique aqui</button>
```

Exemplo:

```
<h1>My First JavaScript</h1>

<button type="button"
onclick="document.getElementById('demo').innerHTML = Date()">
Click me to display Date and Time.</button>

<p id="demo"></p>
```

✓ 2. JavaScript Interno (dentro do HTML)

Inserido dentro da tag `<script>` no final do **body**:

```
<!DOCTYPE html>
<html>
<body>
<h1>Exemplo</h1>
<button id="btn">Clique</button>

<script>
  document.getElementById("btn").onclick = function() {
    alert("Você clicou no botão!");
  };
</script>
</body>
</html>
```

✓ 3. JavaScript Externo (em outro arquivo)

É a forma mais profissional — o JS fica em um arquivo separado:


```
projeto/
├── index.html
├── script.js  <-- arquivo JavaScript
```

index.html

```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Externo</h1>
<button id="btn">Mostrar Alerta</button>
<script src="script.js"></script> <!-- ✓ Conectando o JS -->
</body>
</html>
```

script.js

```
document.getElementById("btn").addEventListener("click", function() {
  alert("JavaScript carregado de um arquivo externo!");
});
```

 **Dica:** Sempre coloque o `<script src="script.js">` no final do `<body>` — assim o HTML é carregado antes do JavaScript.

HTML + CSS + JavaScript — Trabalhando Juntos

Para criar páginas web completas, utilizamos três tecnologias que funcionam em perfeita sintonia:

- ✓ **HTML** — Estrutura e conteúdo da página (texto, imagens, botões);
- ✓ **CSS** — Aparência visual (cores, fontes, layout, responsividade);
- ✓ **JavaScript** — Comportamento e interatividade (cliques, animações, validações, APIs);

📌 Exemplo de HTML, CSS e JS funcionando juntos:

```
<!DOCTYPE html>
  <html lang="pt-BR">
  <head>
    <meta charset="UTF-8">
    <title>Exemplo Completo</title>
    <link rel="stylesheet" href="styles.css"> <!-- CSS Externo -->
  </head>
  <body>
    <h1 id="titulo">Bem-vindo!</h1>
    <button id="botao">Clique em mim</button>

    <script src="script.js"></script> <!-- JavaScript Externo -->
  </body>
</html>
```

📁 styles.css:

```
body {
  text-align: center;
  background-color: #f0f0f0;
  font-family: Arial, sans-serif;
}

h1 {
  color: #007bff;
}

button {
  padding: 10px 20px;
  font-size: 16px;
  cursor: pointer;
  border: none;
  background-color: #007bff;
  color: white;
  border-radius: 5px;
}

button:hover {
```

```
background-color: #0056b3;  
}
```

script.js:

```
document.getElementById("botao").addEventListener("click", function() {  
    document.getElementById("titulo").textContent = "Você clicou no botão!";  
});
```

💡 **Resumo:** HTML cria a estrutura, CSS embeleza e JavaScript dá vida à página. Juntos, eles formam a base de qualquer site ou aplicativo web moderno.

✅ Conclusão: Como tudo se conecta

Ao longo deste módulo, aprendemos que uma página web é construída a partir da combinação de três tecnologias principais:

- ✓ **HTML** — É a base da página. Define *o que* aparece: textos, imagens, formulários, tabelas e estrutura semântica.
- ✓ **CSS** — Controla *como* tudo é visualizado: cores, fontes, espaçamento, bordas, responsividade e layout.
- ✓ **JavaScript** — Torna a página interativa: cliques, animações, validações de formulário, efeitos dinâmicos e comunicação com servidores.

💡 **A ideia principal:** HTML constrói o conteúdo, CSS deixa bonito e JavaScript dá movimento. Quando trabalhados em conjunto, eles criam sites modernos, responsivos e funcionais.

📖 *Este projeto foi pensado para revisar conceitos fundamentais e reforçar meus conhecimentos em desenvolvimento web com base na prática, visualidade e exemplos reais.*

Próximo Módulo: CSS

<> Teste você mesmo:

```
1 <h1>Olá Mundo!</h1>
```

Pré-visualização:

Olá Mundo!



Acesse nosso repositório no GitHub e deixe sua contribuição

© 2025 Adriano Souza Fosneca Desenvolvedor Full-Stack Jr. Todos os direitos reservados.