

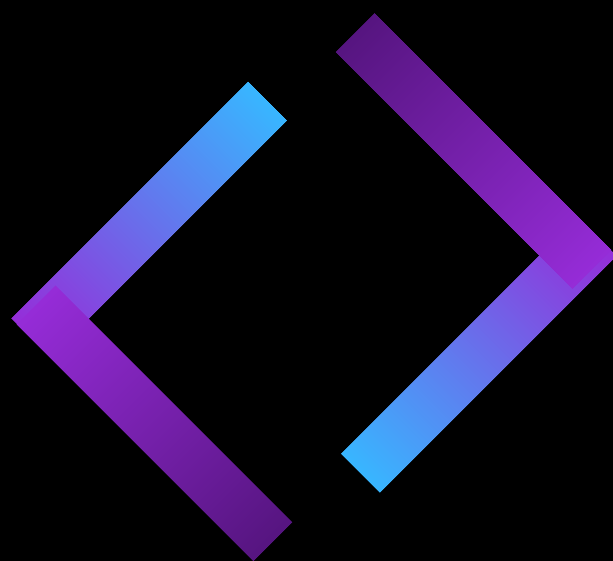
APOSTILA PET C3

HTML e CSS

Conhecimento retido
é conhecimento
perdido.



Ciências Computacionais



PETCode

HTML e CSS

PETCode

Índice

1. Introdução.....	3
1.1 O que é desenvolvimento web?.....	3
1.2 A importância do HTML e CSS.....	3
1.3 Ferramentas necessárias para começar.....	4
2. HTML - Estrutura.....	5
2.1 O que é HTML?.....	5
2.2 Estrutura básica de um documento HTML.....	5
2.3 Elementos e tags mais comuns.....	6
▪ <code><html></code> , <code><head></code> , <code><body></code>	
▪ <code><p></code> , <code><h1></code> a <code><h6></code> , <code><a></code> , <code></code> , <code></code> , <code></code> , <code></code> , <code><div></code> , <code></code>	
2.4 Boas práticas para escrever HTML.....	7
3. HTML - Conteúdo Avançado.....	9
3.1 Atributos e suas funções.....	9
3.2 Links internos e externos.....	9
3.3 Trabalhando com imagens (responsividade e SEO).....	10
3.4 Formulários: <code><form></code> , <code><input></code> , <code><textarea></code> , <code><button></code> , <code><select></code>	10
3.5 Elementos semânticos: <code><header></code> , <code><footer></code> , <code><section></code> , <code><article></code> , <code><aside></code> , <code><nav></code>	11
4. CSS - Estilo.....	13
4.1 O que é CSS?.....	13
4.2 Estrutura básica de um arquivo CSS.....	13
4.3 Formas de aplicar CSS: inline, interno e externo.....	14
4.4 Seletores básicos: por tag, id e classe.....	15
4.5 Propriedades mais usadas: <code>color</code> , <code>font-family</code> , <code>background</code> , <code>border</code> , <code>margin</code> , <code>padding</code> , <code>display</code>	16
5. CSS - Layout e Design.....	18
5.1 Modelo de caixa (box model).....	18

PETCode

Índice

5.2	Trabalhando com layouts responsivos.....	18
5.3	Grid layout.....	19
5.4	Flexbox.....	20
5.5	Media Queries e design responsivo.....	21
5.6	Tipografia: fontes, tamanhos e espaçamento.....	21
5.7	Estilização de links, botões e formulários.....	22
6.	HTML e CSS - Interação.....	24
6.1	Conectando HTML e CSS.....	24
6.2	Trabalhando com classes e IDs.....	25
6.3	Animações básicas com CSS.....	26
6.4	Pseudo-classes e pseudo-elementos (:hover, :before, :after).....	27
7.	Publicação.....	30
7.1	Testando localmente seu site.....	30
7.2	Introdução ao uso do Git e GitHub para publicar.....	31
7.3	Hospedagem gratuita: GitHub Pages e Netlify.....	32
7.4	Dicas.....	33
8.	Exercícios Práticos.....	34
8.1	Criando uma página simples com HTML.....	34
8.2	Aplicando estilo a uma página com CSS.....	35
8.3	Construindo um portfólio básico.....	36
8.4	Exercício final: página responsiva com menu e formulários.....	37
9.	Referências e Recursos.....	40
9.1	Documentação oficial (MDN, W3Schools).....	40
9.2	Ferramentas úteis: Visual Studio Code, extensões úteis, sites de paletas de cores.....	40
9.3	Sites para praticar.....	41
9.4	Recursos para design.....	41
9.5	Comunidades e fóruns para suporte.....	42

Introdução

O que é desenvolvimento web?

Desenvolvimento web é o processo de criar e manter websites. Ele envolve diversas tecnologias e linguagens que, juntas, permitem construir páginas que você acessa diariamente, como sites de notícias, blogs, redes sociais, e-commerce, entre outros.

O desenvolvimento web é dividido em duas principais áreas:

- **Front-end:** O lado visível de um site, que inclui o design, layout, cores e interatividade.
- **Back-end:** A parte que roda "nos bastidores", incluindo servidores, bancos de dados e lógica de negócios.

Nesta apostila, vamos focar no front-end, especificamente nos dois pilares principais:

- **HTML (HyperText Markup Language):** Define a estrutura e o conteúdo das páginas.
- **CSS (Cascading Style Sheets):** Define o estilo e a aparência das páginas.

A importância do HTML e CSS

HTML e CSS são a base do desenvolvimento web. Qualquer site ou aplicativo web que você utiliza diariamente depende dessas tecnologias. Entender como elas funcionam é essencial para quem deseja criar ou personalizar sites.

- **HTML** permite organizar o conteúdo, como textos, imagens e links.
- **CSS** possibilita personalizar a aparência, como cores, fontes e espaçamento.

Por que aprender HTML e CSS?

- **Simples de começar:** Você pode criar suas primeiras páginas com um editor de texto e um navegador.

- Amplamente utilizado: Qualquer plataforma ou framework moderno (React, Angular, etc.) começa com HTML e CSS.
- Recompensador: É empolgante ver suas ideias ganharem vida em uma página visual.

Ferramentas necessárias para começar

Para desenvolver páginas com HTML e CSS, você precisará de:

1. Um Editor de Código

Um editor de texto ou código para escrever HTML e CSS. Recomendamos:

- **Visual Studio Code:** Leve, poderoso e gratuito.
 - Instale extensões como "**Live Server**" para visualizar suas páginas em tempo real.

2. Um Navegador Web

É onde suas páginas serão exibidas. Utilize navegadores modernos como:

- **Google Chrome**
- **Mozilla Firefox**
- **Microsoft Edge**

3. (Opcional) Um Gerenciador de Arquivos

Se você deseja organizar seu projeto, é importante saber como lidar com pastas e arquivos no computador.

HTML – Estrutura

O que é HTML?

HTML (HyperText Markup Language) é a linguagem padrão usada para criar a estrutura básica de uma página web. Ele é responsável por organizar e descrever o conteúdo que será exibido no navegador, como textos, imagens, links e tabelas.

- HyperText: Refere-se à capacidade de criar links que conectam diferentes páginas ou partes de uma página.
- Markup Language: Uma linguagem de marcação que utiliza tags para definir o propósito de cada elemento no documento.

Exemplo de uma página HTML simples:

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Minha Primeira Página</title>
5    </head>
6    <body>
7      <h1>Bem-vindo ao HTML!</h1>
8      <p>Este é um exemplo de página básica.</p>
9    </body>
10 </html>
```

Estrutura básica de um documento HTML

Todo documento HTML segue uma estrutura padrão, composta por:

- **Declaração do tipo de documento**
 - A primeira linha `<!DOCTYPE html>` informa ao navegador que o documento usa a versão atual do HTML.
- **Elemento `<html>`**
 - Contém todo o conteúdo da página.

- **Elemento <head>**
 - Inclui informações sobre a página, como o título, links para estilos (CSS), scripts e metadados.
- **Elemento <body>**
 - Contém o conteúdo visível da página, como textos, imagens e links.

Exemplo detalhado:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <meta charset="UTF-8"> <!-- Define a codificação de caracteres -->
5    <title>Minha Página</title> <!-- Título da página exibido no navegador -->
6  </head>
7  <body>
8    <h1>Olá, mundo!</h1> <!-- Título principal -->
9    <p>Este é um parágrafo de exemplo.</p> <!-- Parágrafo de texto -->
10 </body>
11 </html>
```

Elementos mais comuns

No HTML, usamos **tags** para criar elementos. Cada elemento é formado por uma **tag de abertura**, um **conteúdo** e, geralmente, uma **tag de fechamento**

Exemplos de elementos básicos:

- **Título: <h1> a <h6>** (níveis de importância)
<h1>Título principal</h1>
<h2>Subtítulo</h2>
- **Parágrafos: <p>**
<p>Este é um parágrafo de texto.</p>
- **Links: <a>**
Visite o Google
- **Imagens: **

- Listas:

Ordenada (numerada): ``

```
<ol>
<li>Primeiro item</li>
<li>Segundo item</li>
</ol>
```

Não ordenada (com marcadores): ``

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
</ul>
```

- Divisões e spans:

Divisão de bloco: `<div>`

```
<div>
  <h2>Seção</h2>
  <p>Conteúdo dentro de uma divisão.</p>
</div>
```

Texto em linha: ``

```
<p>Este texto tem um <span style="color: red;">destaque</span>.
</p>
```

Boas praticas para escrever HTML

- Organização e legibilidade:
Use indentação para hierarquizar elementos.

```
1  <body>
2  <div>
3    <h1>Exemplo</h1>
4    <p>Parágrafo.</p>
5  </div>
6 </body>
```

- **Feche as tags corretamente:**

Sempre use tags de fechamento para evitar erros.

- **Comente o código quando necessário:**

Use comentários para explicar partes do código.

```
<!-- Este é um comentário no HTML -->
```

- **Use atributos corretamente:**

Adicione descrições no atributo **alt** em imagens para acessibilidade.

Inclua links absolutos (URLs completas) ou relativos (dependendo da estrutura de pastas).

HTML – Estrutura

Atributos e suas funções

Os atributos são usados em tags HTML para adicionar informações extras ou personalizar seu comportamento. Eles são escritos na tag de abertura e possuem um formato de nome="valor".

Exemplos de atributos comuns:

- **id**: Identifica unicamente um elemento.
`<h1 id="titulo-principal">Título da Página</h1>`
- **class**: Agrupa elementos com características semelhantes.
`<p class="descricao">Este parágrafo pertence à classe "descricao".</p>`
- **src**: Especifica a fonte de imagens ou scripts.
``
- **alt**: Texto alternativo para imagens, importante para acessibilidade.
``
- **href**: Define o destino de links.
`Google`

Links internos e externos

Links internos:

Apontam para outras partes do mesmo site.

```
<a href="#sobre">Saiba mais sobre nós</a>
<!-- Link para um elemento com o id "sobre" -->
```

Links externos:

Apontam para outros sites.

```
<a href="https://www.github.com" target="_blank">Visite o GitHub</a>  
<!-- O atributo target="_blank" abre o link em uma nova aba -->
```

TRABALHO COM IMAGENS

Adicionando uma imagem:

```

```

Tornando imagens responsivas:

Adicione a propriedade CSS **width: 100%**; ou use o atributo **height** e **width** para manter as proporções.

```

```

Melhorando SEO:

1. Use nomes de arquivo descritivos, como **pessoas-feliz.jpg**.
2. Preencha o atributo **alt** com descrições precisas.

Formulários

Os formulários são usados para coletar informações dos usuários.

Elementos básicos de formulários:

```
<form action="/enviar-dados" method="POST">  
  <label for="nome">Nome:</label>  
  <input type="text" id="nome" name="nome" placeholder="Digite seu  
nome">
```

```
  <label for="email">Email:</label>  
  <input type="email" id="email" name="email" placeholder="Digite  
seu email">
```

```
  <label for="mensagem">Mensagem:</label>  
  <textarea id="mensagem" name="mensagem" rows="4"></textarea>
```

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

Atributos importantes:

- **action**: Define o destino dos dados do formulário.
- **method**: Especifica o método de envio (geralmente GET ou POST).
- **placeholder**: Exibe um texto informativo dentro de campos.

Elementos semânticos

Os elementos semânticos fornecem um significado claro ao conteúdo, ajudando na acessibilidade e na otimização para motores de busca (SEO).

Exemplos de elementos semânticos:

1. **<header>**: Cabeçalho da página ou de uma seção.

```
<header>
  <h1>Meu Site</h1>
  <nav>
    <a href="#sobre">Sobre</a>
    <a href="#contato">Contato</a>
  </nav>
</header>
```

2. **<footer>**: Rodapé da página.

```
<footer>
  <p>© 2025 Meu Site. Todos os direitos reservados.</p>
</footer>
```

3. **<section>**: Agrupa conteúdo relacionado.

```
<section id="sobre">
  <h2>Sobre Nós</h2>
  <p>Informações sobre a empresa.</p>
</section>
```

4. **<article>**: Representa um conteúdo independente (artigos, posts, etc.).

```
<article>
  <h3>Post no Blog</h3>
  <p>Conteúdo do artigo.</p>
</article>
```

5. **<aside>**: Informação complementar ou secundária.

```
<aside>
  <h4>Dicas</h4>
  <p>Veja também...</p>
</aside>
```

6. **<nav>**: Links de navegação.

```
<nav>
  <a href="#home">Home</a>
  <a href="#servicos">Serviços</a>
</nav>
```

Por que usar elementos semânticos?

1. Melhor organização e clareza do código.
2. Maior acessibilidade para leitores de tela.
3. SEO otimizado, ajudando mecanismos de busca a entenderem o conteúdo.

CSS - Estilo

O que é CSS?

CSS (Cascading Style Sheets) é uma linguagem usada para estilizar páginas HTML, controlando cores, fontes, espaçamento, layout e muito mais. Com o CSS, é possível transformar uma página simples em uma interface atraente e funcional.

Por que usar CSS?

1. **Separação de conteúdo e estilo:** O HTML organiza o conteúdo, enquanto o CSS cuida da aparência.
2. **Reutilização de estilos:** Um arquivo CSS pode ser aplicado a várias páginas.
3. **Customização avançada:** Permite criar designs responsivos e modernos.

Estrutura básica de um arquivo CSS

principais:

- **Seletor:** Define quais elementos serão estilizados.
- **Declaração:** Define o estilo aplicado, formado por uma propriedade e um valor.

Exemplo de uma regra CSS:

```
1  h1 {  
2    color: blue; /* Propriedade: cor do texto */  
3    font-size: 24px; /* Propriedade: tamanho da fonte */  
4  }
```

Formas de aplicar CSS

1.CSS inline (dentro do elemento HTML):

- Aplicado diretamente na tag usando o atributo `style`.

```
1 <p style="color: red;">Texto em vermelho</p>
```

2.CSS interno (no cabeçalho do HTML):

- Usado dentro de uma tag `<style>` no elemento `<head>`.

```
1 <head>
2   <style>
3     body {
4       background-color: lightgray;
5     }
6   </style>
7 </head>
```

3.CSS externo (arquivo separado):

- Arquivo com extensão `.css` vinculado ao HTML com a tag `<link>`.

```
1 <head>
2   <link rel="stylesheet" href="estilos.css">
3 </head>
```

Arquivo `estilos.css`:

```
1 body {
2   background-color: lightgray;
3 }
```

Seletores básicos

Seletores por tipo (tag):

Estiliza todos os elementos de um tipo específico.

```
p {  
  font-family: Arial, sans-serif;  
}
```

Seletores por classe:

Aplica estilos a elementos com a mesma classe.

HTML:

```
<p class="destaque">Texto destacado</p>
```

CSS:

```
.destaque {  
  color: orange;  
}
```

Seletores por ID:

Aplica estilos a um único elemento com o mesmo ID.

HTML:

```
<h1 id="titulo-principal">Título</h1>
```

CSS:

```
#titulo-principal {  
  font-size: 32px;  
}
```

Propriedades mais usadas

Cores e fundos:

- **color**: Cor do texto.

```
p {  
  color: blue;  
}
```

- **background-color**: Cor de fundo.

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

Fontes e textos:

- **font-family**: Define a fonte.

```
h1 {  
  font-family: "Arial", sans-serif;  
}
```

- **font-size**: Tamanho da fonte.

```
p {  
  font-size: 16px;  
}
```

- **text-align**: Alinhamento do texto.

```
h1 {  
  text-align: center;  
}
```

Bordas e espaçamento:

- **border**: Estiliza bordas.

```
div {  
  border: 2px solid black;  
}
```

- **margin**: Espaçamento externo.

```
p {  
  margin: 10px;  
}
```

- **padding**: Espaçamento interno.

```
div {  
  padding: 20px;  
}
```

Exibição e layout:

- **display**: Define como os elementos são exibidos.

```
div {  
  display: flex;  
}
```

CSS – Layout e Design

Modelo de Caixa (Box Model)

O **modelo de caixa** é a base de como os navegadores renderizam os elementos HTML. Cada elemento é representado como uma caixa retangular composta por:

1. **Conteúdo**: O espaço onde texto e imagens aparecem.
2. **Preenchimento (padding)**: Espaço entre o conteúdo e a borda.
3. **Borda (border)**: A borda ao redor do preenchimento.
4. **Margem (margin)**: Espaço externo entre a borda do elemento e outros elementos.

Exemplo visual:

```
div {  
  width: 200px; /* Largura do conteúdo */  
  padding: 10px; /* Espaço interno */  
  border: 2px solid black; /* Borda */  
  margin: 20px; /* Espaço externo */  
}
```

Trabalhando com layouts responsivos

Um layout responsivo adapta-se a diferentes tamanhos de tela, como desktops, tablets e celulares.

Usando unidades flexíveis:

- % (**percentual**): Proporcional ao elemento pai.
- **em** e **rem**: Baseadas no tamanho da fonte.
- **vh** e **vw**: Proporcional à altura e largura da janela.

Exemplo:

```
body {  
  font-size: 16px; /* Base para rem */  
}  
div {  
  width: 50%; /* Metade da largura do pai */  
  height: 50vh; /* Metade da altura da janela */  
}
```

Media Queries:

Permite alterar estilos com base no tamanho da tela.

```
@media (max-width: 768px) {  
  body {  
    font-size: 14px;  
  }  
}
```

Flexbox

O Flexbox é uma ferramenta poderosa para criar layouts flexíveis e alinhados.

Conceitos básicos:

1. **Container flexível:** O elemento pai que define o contexto flex.
2. **Itens flexíveis:** Elementos filhos no container.

Propriedades principais do container:

- **display: flex;** Ativa o Flexbox.
- **justify-content:** Alinha os itens no eixo principal.
 - Valores: **flex-start**, **center**, **space-between**, **space-around**.
- **align-items:** Alinha os itens no eixo transversal.
 - Valores: **stretch**, **center**, **flex-start**, **flex-end**.

Exemplo:

```
.container {  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  height: 100vh;  
}
```

Flexbox

O Grid Layout é ideal para criar layouts bidimensionais (linhas e colunas).

Conceitos básicos:

1. **Container de grid:** Define o contexto grid.
2. **Itens de grid:** Elementos filhos dentro do grid.

Propriedades principais:

- **display: grid;** Ativa o Grid Layout.
- **grid-template-columns** e **grid-template-rows:** Define colunas e linhas.

```
.container {  
  display: grid;  
  grid-template-columns: 1fr 1fr 1fr; /* 3 colunas iguais */  
  grid-template-rows: auto auto; /* 2 linhas ajustáveis */  
}
```

- **gap:** Define o espaçamento entre linhas e colunas.

```
.container {  
  gap: 20px;  
}
```

Exemplo:

```
.container {  
  display: grid;  
  grid-template-columns: 1fr 2fr; /* 2 colunas: 1 parte e 2 partes */  
  gap: 10px;  
}  
  
.item {  
  background-color: lightblue;  
}
```

Media Queries e design responsivo

As Media Queries são usadas para criar designs que se adaptam a diferentes dispositivos. Você pode ajustar o layout, tamanhos de fonte e outros estilos com base em larguras específicas.

Exemplo:

```
body {  
  font-size: 16px;  
}  
  
@media (max-width: 768px) {  
  body {  
    font-size: 14px;  
  }  
}  
  
@media (max-width: 480px) {  
  body {  
    font-size: 12px;  
  }  
}
```

Tipografia

A tipografia é essencial para a legibilidade e aparência do texto.

Propriedades principais:

- **font-family:** Define a fonte.

```
body {  
  font-family: "Arial", sans-serif;  
}
```

- **font-size**: Controla o tamanho da fonte.

```
h1 {  
  font-size: 2rem;  
}
```

- **line-height**: Ajusta o espaçamento entre linhas.

```
p {  
  line-height: 1.5;  
}
```

Estilização de links, botões e formulários

Links:

Use pseudo-classes para criar efeitos de interação:

```
a {  
  text-decoration: none;  
  color: blue;  
}  
a:hover {  
  color: red;  
}
```

Botões:

```
button {  
  background-color: #007bff;  
  color: white;  
  padding: 10px 20px;  
  border: none;  
  border-radius: 5px;  
  cursor: pointer;  
}  
button:hover {  
  background-color: #0056b3;  
}
```

Formulários:

```
input, textarea {  
  width: 100%;  
  padding: 10px;  
  border: 1px solid #ccc;  
  border-radius: 4px;  
}  
input:focus, textarea:focus {  
  border-color: #007bff;  
  outline: none;  
}
```

HTML e CSS - Interação

Neste capítulo, exploraremos como conectar HTML e CSS, aplicando estilos aos elementos e criando interações modernas com animações e pseudo-classes.

Conectando HTML e CSS

Existem três formas principais de aplicar CSS ao HTML:

1. CSS Inline

O estilo é aplicado diretamente no elemento HTML usando o atributo **style**.

```
<p style="color: red; font-size: 20px;">Texto em vermelho</p>
```

2. CSS Interno

Os estilos são adicionados dentro da tag **<style>** no cabeçalho do documento HTML.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    body {
      background-color: lightblue;
    }
  </style>
</head>
<body>
  <h1>Bem-vindo!</h1>
</body>
</html>
```

3. CSS Externo

O CSS é armazenado em um arquivo separado (estilos.css) e vinculado ao HTML usando a tag **<link>** no cabeçalho.

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="estilos.css">
</head>
<body>
  <h1>Bem-vindo!</h1>
</body>
</html>
```

Arquivo **estilos.css**:

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

Essa é a abordagem mais recomendada, pois separa o conteúdo (HTML) do estilo (CSS), facilitando a manutenção do código.

Trabalhando com classes e IDs

Classes

As classes permitem aplicar estilos a múltiplos elementos com características semelhantes. Use o atributo `class` no HTML e seletores com ponto (.) no CSS.

HTML:

```
<div class="box">Caixa 1</div>
<div class="box">Caixa 2</div>
```

CSS:

```
.box {
  width: 100px;
  height: 100px;
  background-color: blue;
  color: white;
  text-align: center;
  margin: 10px;
}
```

IDs

Os IDs identificam unicamente um elemento. Use o atributo `id` no HTML e seletores com `#` no CSS.

HTML:

```
<div id="box-principal">Caixa Principal</div>
```

CSS:

```
#box-principal {  
  width: 200px;  
  height: 200px;  
  background-color: green;  
  color: white;  
  text-align: center;  
}
```

Animações básicas com CSS

As animações no CSS trazem vida aos elementos e tornam a página mais interativa.

Transições

Permitem animações suaves entre dois estados.

Exemplo:

```
<button class="botao">Clique Aqui</button>  
  
.botao {  
  background-color: blue;  
  color: white;  
  padding: 10px 20px;  
  border: none;  
  border-radius: 5px;  
  transition: background-color 0.3s ease;  
}  
  
.botao:hover {  
  background-color: darkblue;  
}
```

Keyframes

Usados para criar animações mais complexas.

Exemplo:

```
<div class="animado">Animação</div>

@keyframes mover {
  0% {
    transform: translateX(0);
  }
  50% {
    transform: translateX(50px);
  }
  100% {
    transform: translateX(0);
  }
}

.animado {
  width: 100px;
  height: 100px;
  background-color: red;
  animation: mover 2s infinite;
}
```

Pseudo-classes e pseudo-elementos

Pseudo-classes

As pseudo-classes são usadas para estilizar elementos em estados específicos, como ao passar o mouse.

Exemplo:

```
a{
  color: blue;
}

a:hover {
  color: red;
  text-decoration: underline;
}
```

Pseudo-elementos

Os pseudo-elementos estilizam partes específicas de um elemento.

Exemplo:

```
p::first-line {  
  font-weight: bold;  
  color: green;  
}
```

Criando interações modernas

Efeito hover em botões

Crie animações simples ao passar o mouse sobre botões.

HTML:

```
<button class="hover-animado">Passe o Mouse</button>
```

CSS:

```
.hover-animado {  
  background-color: purple;  
  color: white;  
  padding: 10px 20px;  
  border: none;  
  border-radius: 5px;  
  transition: transform 0.3s;  
}
```

```
.hover-animado:hover {  
  transform: scale(1.1);  
}
```

Menus suspensos

Crie menus interativos com CSS.

HTML:

```
<nav class="menu">
  <ul>
    <li>Home</li>
    <li>Serviços
      <ul class="submenu">
        <li>Web Design</li>
        <li>SEO</li>
      </ul>
    </li>
    <li>Contato</li>
  </ul>
</nav>
```

CSS:

```
.menu ul {
  list-style: none;
  padding: 0;
}

.menu li {
  position: relative;
  display: inline-block;
  padding: 10px 20px;
}

.submenu {
  display: none;
  position: absolute;
  top: 100%;
  left: 0;
  background-color: #ddd;
  list-style: none;
}

.menu li:hover .submenu {
  display: block;
}
```

Publicação

Depois de criar e estilizar uma página com HTML e CSS, o próximo passo é torná-la acessível ao público. Este capítulo ensinará como testar localmente e publicar seus projetos na internet usando ferramentas gratuitas.

Testando localmente seu site

Antes de publicar, é importante verificar se o site funciona como esperado no ambiente local.

Passos:

1. Crie uma pasta para o projeto.

- Inclua os arquivos HTML, CSS e imagens.

2. Abra o arquivo HTML no navegador.

- Clique com o botão direito no arquivo `.html` e escolha "Abrir com" seguido do navegador de sua preferência.

3. Use um editor de código com servidor local.

- Recomendamos usar o Visual Studio Code com a extensão Live Server:
 - Instale o Live Server na aba de extensões.
 - Abra seu arquivo HTML no VS Code.
 - Clique em "Go Live" no canto inferior direito para visualizar o site no navegador.
 -

Benefícios do Live Server:

- Atualizações em tempo real ao salvar o código.
- Simula um servidor local, útil para testar recursos que dependem de caminhos relativos.

Introdução ao uso do Git e Github

O que é Git e GitHub?

- Git: Sistema de controle de versão que ajuda a rastrear mudanças no código.
- GitHub: Plataforma para armazenar e compartilhar projetos Git na nuvem.

Configurando Git no projeto:

1. Instale o Git: Baixe o instalador no [site oficial do Git](#).

2. Inicie um repositório Git no projeto:
`git init`

3. Adicione os arquivos ao repositório:
`git add.`
`git commit -m "Primeiro commit"`

Subindo o projeto para o GitHub:

1. Crie um repositório no GitHub.
2. Conecte o repositório local ao GitHub:
`git remote add origin https://github.com/seu-usuario/nome-do-repositorio.git`
3. Envie os arquivos para o GitHub:
`git push -u origin main`

Publicando com GitHub Pages

O GitHub Pages é uma solução gratuita para hospedar sites estáticos diretamente de um repositório.

Como usar:

1. **Acesse o repositório no GitHub.**
2. Vá para **Configurações > Pages**.
3. Em "Source", escolha a branch **main** e a pasta raiz.
4. Clique em "Save".

Seu site estará disponível em:

<https://seu-usuario.github.io/nome-do-repositorio/>

Publicando com Netlify

O **Netlify** é outra ferramenta gratuita para hospedar sites estáticos, com integração simplificada e suporte a domínios personalizados.

Passos para publicar:

1. **Crie uma conta no [Netlify](#).**
2. **Conecte seu repositório do GitHub:**
 - Clique em "Add New Site" e escolha "Import an existing project".
 - Conecte o repositório onde está seu projeto.
3. **Configure o build:**
 - Para sites estáticos com HTML e CSS, basta usar a pasta raiz (/).
4. **Implante o site.**
 - O Netlify fornecerá um link automático (ex.: <https://meu-site.netlify.app>).

Hospedagem em outras plataformas gratuitas

1. Vercel

- Semelhante ao Netlify, ideal para sites estáticos e frameworks como Next.js.
- Link: <https://vercel.com>

2. Google Firebase

- Serviço de hospedagem gratuito com ferramentas adicionais, como banco de dados e autenticação.

- Link: <https://firebase.google.com>
- 3. Cloudflare Pages**
- Hospedagem rápida e gratuita para projetos estáticos.
 - Link: <https://pages.cloudflare.com>

Dicas para publicação

- 1. Otimize suas imagens:**
 - Reduza o tamanho das imagens antes de enviar para o servidor.
 - Use ferramentas como [TinyPNG](#).
- 2. Use nomes descritivos para arquivos:**
 - Prefira nomes como [sobre-nos.html](#) em vez de [pagina1.html](#).
- 3. Teste em diferentes dispositivos e navegadores:**
 - Certifique-se de que o site funciona em celulares, tablets e desktops.
- 4. Certifique-se de que os links estão funcionando:**
 - Verifique todos os links internos e externos antes de publicar.

Exercícios Práticos

Este capítulo apresenta exercícios práticos para consolidar os conhecimentos de HTML e CSS adquiridos nos capítulos anteriores. As atividades começam com tarefas simples e gradualmente aumentam em complexidade, culminando na criação de um site completo e responsivo.

Criando uma página simples com HTML

Objetivo:

Criar uma página HTML que contenha:

- Um título.
- Um parágrafo.
- Uma lista ordenada e uma lista não ordenada.
- Um link para outro site.

Instruções:

1. Crie um arquivo chamado pagina.html.
2. Adicione a estrutura básica do HTML.
3. Insira os elementos solicitados.

Exemplo de código:

```
<!DOCTYPE html>
<html>
<head>
  <title>Minha Página Simples</title>
</head>
<body>
  <h1>Bem-vindo à Minha Página</h1>
  <p>Este é um exemplo de parágrafo.</p>

  <h2>Minha Lista Ordenada</h2>
  <ol>
    <li>Item 1</li>
    <li>Item 2</li>
  </ol>
```

```
<h2>Minha Lista Não Ordenada</h2>
<ul>
  <li>Item A</li>
  <li>Item B</li>
</ul>
<p>Visite<a href="https://www.google.com" target="_blank">Google<
/a>.</p>
</body>
</html>
```

Aplicando estilo a uma página com CSS

Objetivo:

Criar um arquivo CSS para estilizar a página HTML do exercício anterior.

Instruções:

1. Crie um arquivo chamado **estilos.css**.
2. Adicione regras de estilo para:
 - Alterar a cor do fundo.
 - Alterar a fonte do título.
 - Adicionar bordas às listas.

Exemplo de código CSS:

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

h1 {
  color: darkblue;
  text-align: center;}

ol, ul {
  border: 1px solid #ccc;
  padding: 10px;
  width: 300px;
  margin: auto;}
```

Atualize o HTML para incluir o CSS:

```
<link rel="stylesheet" href="estilos.css">
```

Construindo um portfólio básico

Objetivo:

Criar uma página de portfólio com informações pessoais e um resumo de projetos.

Estrutura sugerida:

- **Seção 1:** Título e descrição pessoal.
- **Seção 2:** Lista de habilidades (use listas não ordenadas).
- **Seção 3:** Projetos (use links para os projetos).

Exemplo de código:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="estilos.css">
  <title>Meu Portfólio</title>
</head>
<body>
  <header>
    <h1>João Silva</h1>
    <p>Desenvolvedor Front-End</p>
  </header>
  <section>
    <h2>Habilidades</h2>
    <ul>
      <li>HTML</li>
      <li>CSS</li>
      <li>JavaScript</li>
    </ul>
  </section>
  <section>
    <h2>Projetos</h2>
    <ul>
      <li><a href="https://github.com/seu-usuario/projeto1"
target="_blank">Projeto 1</a></li>
      <li><a href="https://github.com/seu-usuario/projeto2"
target="_blank">Projeto 2</a></li>
    </ul>
  </section>
</body>
</html>
```

Exercício final: Página responsiva

Objetivo:

Criar uma página responsiva que inclua:

- Um cabeçalho com navegação.
- Um banner.
- Uma seção de informações com layout de duas colunas.
- Um rodapé.

Instruções:

1. Crie um arquivo `index.html` e um `style.css`.
2. Use **Flexbox** ou **Grid** para criar o layout.
3. Adicione **Media Queries** para ajustar o layout em telas menores.

Exemplo de código HTML:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="style.css">
  <title>Página Responsiva</title>
</head>
<body>
  <header>
    <nav>
      <ul>
        <li><a href="#home">Home</a></li>
        <li><a href="#sobre">Sobre</a></li>
        <li><a href="#contato">Contato</a></li>
      </ul>
    </nav>
  </header>
  <section id="banner">
    <h1>Bem-vindo ao Meu Site</h1>
  </section>
  <section id="sobre">
    <div class="coluna">
      <h2>Sobre Mim</h2>
      <p>Texto sobre mim.</p>
    </div>
    <div class="coluna">
      
    </div>
  </section>
```

```
<footer>
  <p>© 2025 João Silva. Todos os direitos reservados.</p>
</footer>
</body>
</html>
```

Exemplo de código CSS:

```
/* Estilos gerais */
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
}
```

```
header {
  background-color: #333;
  color: white;
  padding: 10px 0;
}
```

```
header ul {
  list-style: none;
  display: flex;
  justify-content: center;
}
```

```
header ul li {
  margin: 0 15px;
}
```

```
header ul li a {
  color: white;
  text-decoration: none;
}
```

```
#banner {
  background-color: lightblue;
  text-align: center;
  padding: 50px 0;
}
```

```
#sobre {
  display: flex;
  gap: 20px;
  padding: 20px;
}

.coluna {
  flex: 1;
}

footer {
  background-color: #333;
  color: white;
  text-align: center;
  padding: 10px 0;
}

/* Responsividade */
@media (max-width: 768px) {
  #sobre {
    flex-direction: column;
    text-align: center;
  }
}
```

Referencias e Recursos

Para se tornar um desenvolvedor web competente, é essencial consultar materiais de referência confiáveis e utilizar ferramentas que facilitem o aprendizado e o desenvolvimento. Este capítulo reúne os principais recursos, documentações e ferramentas úteis para quem está começando ou quer se aprofundar em HTML e CSS.

Documentações oficiais

As documentações são a fonte mais confiável para aprender e tirar dúvidas sobre HTML e CSS.

1. MDN Web Docs (Mozilla Developer Network)

- Documentação detalhada e exemplos práticos de HTML, CSS e JavaScript.
- Link: <https://developer.mozilla.org/>

2. W3Schools

- Tutoriais e exemplos interativos de HTML, CSS e outras tecnologias web.
- Link: <https://www.w3schools.com/>

3. HTML Living Standard

- Especificação oficial do HTML mantida pela WHATWG.
- Link: <https://html.spec.whatwg.org/>

4. CSS Specification

- Especificação oficial do CSS publicada pelo W3C.
- Link: <https://www.w3.org/Style/CSS/Overview.en.html>

Ferramentas úteis

1. Editores de código

- **Visual Studio Code (VS Code):** Editor de código leve, personalizável e com suporte a extensões.
 - Link: <https://code.visualstudio.com/>
- **Sublime Text:** Simples e rápido, ideal para iniciantes.
 - Link: <https://www.sublimetext.com/>

2. Extensões recomendadas para VS Code

- Live Server: Visualize suas alterações em tempo real no navegador.
- Prettier: Formata automaticamente seu código.
- HTML CSS Support: Melhora a autocompletação para HTML e CSS.

3. Ferramentas de depuração

- DevTools: Ferramentas de desenvolvimento integradas aos navegadores (Chrome, Firefox, Edge).
 - Acesse pressionando **F12** ou **Ctrl + Shift + I**.
- Responsively App: Teste a responsividade do site em vários dispositivos simultaneamente.
 - Link: <https://responsively.app/>

Sites para prática

1. CodePen

- Uma plataforma online para criar e compartilhar experimentos com HTML, CSS e JavaScript.
- Link: <https://codepen.io/>

2. JSFiddle

- Ambiente interativo para testar trechos de código HTML, CSS e JavaScript.
- Link: <https://jsfiddle.net/>

3. Frontend Mentor

- Oferece desafios práticos de desenvolvimento front-end com design pré-definido.
- Link: <https://www.frontendmentor.io/>

4. FreeCodeCamp

- Plataforma de aprendizado com exercícios interativos e projetos práticos.
- Link: <https://www.freecodecamp.org/>

Recursos para design e estilos

1. Sites de paletas de cores

- Colors: Ferramenta para criar e explorar paletas de cores.
 - Link: <https://colors.co/>

- **Adobe Color:** Criação de esquemas de cores baseados em teoria de cores.
 - Link: <https://color.adobe.com/>
- 2. **Bibliotecas de ícones**
 - **Font Awesome:** Ícones escaláveis para projetos web.
 - Link: <https://fontawesome.com/>
 - **Material Icons:** Ícones do Google Material Design.
 - Link: <https://fonts.google.com/icons>
- 3. **Geradores de gradientes**
 - **CSS Gradient:** Criação de gradientes para aplicar no CSS.
 - Link: <https://cssgradient.io/>

Comunidades e fóruns

1. **Stack Overflow**
 - Comunidade para tirar dúvidas e buscar soluções de problemas técnicos.
 - Link: <https://stackoverflow.com/>
2. **Reddit**
 - Subreddits úteis:
 - **r/web_design:** https://www.reddit.com/r/web_design/
 - **r/frontend:** <https://www.reddit.com/r/frontend/>
3. **Grupos no Discord e Slack**
 - **DevCommunity:** Comunidade com canais de HTML e CSS.
 - Link: <https://devcommunity.org/>

Livros recomendados

1. **HTML and CSS: Design and Build Websites** - Jon Duckett
 - Livro visualmente atraente, ideal para iniciantes.
2. **CSS Secrets** - Lea Verou
 - Dicas avançadas para melhorar suas habilidades em CSS.
3. **Don't Make Me Think** - Steve Krug
 - Abordagem prática sobre design e usabilidade.



Créditos:

Luiz Fernando Viana Ciriaco

Yan Karlo da Silva Veiga Vasconcellos Dutra