Módulo 2: Desenvolvimento Front-End Básico

1. HTML5: Estrutura, Semântica e Boas Práticas

1.1 Introdução ao HTML5

HTML (HyperText Markup Language) é a linguagem de marcação padrão para criar páginas web. O HTML5 é a quinta e mais recente versão do HTML, trazendo novos elementos, atributos e comportamentos, além de um conjunto mais amplo de tecnologias que permite o desenvolvimento de websites e aplicações mais diversificados e poderosos.

Características do HTML5:

- Semântica aprimorada
- Suporte nativo para áudio e vídeo
- Canvas para gráficos e animações
- Armazenamento local
- Formulários aprimorados
- Geolocalização
- Web Workers para processamento em segundo plano
- WebSockets para comunicação bidirecional

1.2 Estrutura Básica de um Documento HTML5

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Título da Página</title>
  <link rel="stylesheet" href="estilos.css">
  <script src="script.js" defer></script>
</head>
<body>
  <header>
    <h1>Cabeçalho da Página</h1>
    <nav>
      <ul>
        <li><a href="#">Link 1</a></li>
        <li><a href="#">Link 2</a></li>
      </ul>
    </nav>
  </header>

  <main>
    <section>
      <h2>Seção Principal</h2>
      <p>Conteúdo da seção principal.</p>
    </section>

    <article>
      <h2>Artigo</h2>
      <p>Conteúdo do artigo.</p>
    </article>

    <aside>
      <h3>Barra Lateral</h3>
      <p>Conteúdo relacionado.</p>
    </aside>
  </main>

  <footer>
    <p>&copy; 2025 - Todos os direitos reservados</p>
  </footer>
</body>
</html>
```

1.3 Elementos Semânticos do HTML5

O HTML5 introduziu vários elementos semânticos que ajudam a descrever o significado do conteúdo, melhorando a acessibilidade, SEO e manutenção do código:

Elementos de Estrutura:

- <header>: Cabeçalho da página ou seção
- <nav>: Seção de navegação

- <main>: Conteúdo principal da página
- <section>: Seção genérica de conteúdo
- <article>: Conteúdo independente e autocontido
- <aside>: Conteúdo relacionado mas não essencial
- <footer>: Rodapé da página ou seção
- <figure> e <figcaption>: Conteúdo ilustrativo com legenda

Elementos de Texto:

- <h1> a <h6>: Cabeçalhos de diferentes níveis
- <p>: Parágrafo
- <blockquote>: Citação em bloco
- <cite>: Referência a uma obra
- <time>: Data/hora
- <mark>: Texto destacado
- : Ênfase (geralmente em itálico)
- : Forte ênfase (geralmente em negrito)

1.4 Metadados e SEO

Os metadados são informações sobre a página que não são exibidas diretamente, mas são importantes para navegadores, mecanismos de busca e outras aplicações web:

```
<head>
  <!-- Metadados básicos -->
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Título da Página - Palavra-chave Principal</title>

  <!-- Metadados para SEO -->
  <meta name="description" content="Descrição concisa da página em até 160 caracteres.">
  <meta name="keywords" content="palavra-chave1, palavra-chave2, palavra-chave3">
  <meta name="author" content="Nome do Autor">

  <!-- Open Graph para compartilhamento em redes sociais -->
  <meta property="og:title" content="Título para Compartilhamento">
  <meta property="og:description" content="Descrição para compartilhamento.">
  <meta property="og:image" content="https://exemplo.com/imagem.jpg">
  <meta property="og:url" content="https://exemplo.com/pagina">
  <meta property="og:type" content="website">

  <!-- Twitter Card -->
  <meta name="twitter:card" content="summary_large_image">
  <meta name="twitter:title" content="Título para Twitter">
  <meta name="twitter:description" content="Descrição para Twitter.">
  <meta name="twitter:image" content="https://exemplo.com/imagem.jpg">

  <!-- Canonical URL -->
  <link rel="canonical" href="https://exemplo.com/pagina-original">

  <!-- Favicon -->
  <link rel="icon" href="favicon.ico">
  <link rel="apple-touch-icon" href="apple-touch-icon.png">
</head>
```

1.5 Boas Práticas de HTML

Estrutura e Organização:

- Use a declaração DOCTYPE correta
- Especifique o idioma da página com o atributo lang
- Use indentação consistente para melhorar a legibilidade
- Organize o código em blocos lógicos
- Mantenha a hierarquia de cabeçalhos (h1 a h6) correta

Semântica:

- Use elementos semânticos em vez de genéricos quando possível
- Escolha elementos baseados no significado, não na aparência
- Use <button> para ações e <a> para navegação
- Evite usar tabelas para layout (use apenas para dados tabulares)

Acessibilidade:

- Adicione texto alternativo para imagens com alt
- Use atributos ARIA quando necessário

- Garanta que formulários sejam acessíveis (labels, fieldsets)
- Mantenha uma ordem de tabulação lógica
- Teste com leitores de tela

Performance:

- Carregue JavaScript com o atributo defer ou async quando apropriado
- Minimize o número de arquivos externos
- Otimize imagens e outros recursos
- Use lazy loading para imagens e vídeos

Compatibilidade:

- Teste em múltiplos navegadores
- Use polyfills para recursos não suportados em navegadores mais antigos
- Verifique a compatibilidade de recursos em caniuse.com (https://caniuse.com/)

2. CSS3: Estilização, Seletores e Layout

2.1 Introdução ao CSS3

CSS (Cascading Style Sheets) é a linguagem usada para estilizar e formatar documentos HTML. O CSS3 é a versão mais recente, trazendo recursos avançados como animações, transições, gradientes, sombras e layouts flexíveis.

Formas de Incluir CSS:

1. **CSS Inline:**

```
<p style="color: blue; font-size: 16px;">Texto com estilo inline</p>
```

2. **CSS Interno (na seção <head>):**

```
<head>
  <style>
    p {
      color: blue;
      font-size: 16px;
    }
  </style>
</head>
```

3. **CSS Externo (recomendado):**

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

2.2 Seletores CSS

Os seletores determinam quais elementos HTML serão estilizados por um conjunto de regras CSS:

Seletores Básicos:

- **Seletor de Elemento:** p { } (seleciona todos os elementos <p>)

- **Seletor de ID:** #meuId { } (seleciona o elemento com id="meuId")
- **Seletor de Classe:** .minhaClasse { } (seleciona todos os elementos com class="minhaClasse")
- **Seletor Universal:** * { } (seleciona todos os elementos)

Seletores de Combinação:

- **Descendente:** div p { } (seleciona todos os <p> dentro de)
- **Filho Direto:** div > p { } (seleciona apenas os <p> que são filhos diretos de)
- **Irmão Adjacente:** h1 + p { } (seleciona o <p> que vem imediatamente após um <h1>)
- **Irmãos Gerais:** h1 ~ p { } (seleciona todos os <p> que são irmãos de um <h1>)

Seletores de Atributo:

- [attr] - elementos com o atributo attr
- [attr="valor"] - elementos com attr="valor"
- [attr^="valor"] - elementos com attr começando com "valor"
- [attr$="valor"] - elementos com attr terminando com "valor"
- [attr*="valor"] - elementos com attr contendo "valor"

Pseudo-classes:

- :hover - quando o mouse está sobre o elemento
- :active - quando o elemento está sendo ativado
- :focus - quando o elemento tem foco
- :first-child - primeiro filho de seu pai
- :last-child - último filho de seu pai
- :nth-child(n) - enésimo filho de seu pai
- :not(seletor) - elementos que não correspondem ao seletor

Pseudo-elementos:

- ::before - cria um pseudo-elemento antes do conteúdo do elemento
- ::after - cria um pseudo-elemento depois do conteúdo do elemento
- ::first-letter - seleciona a primeira letra do texto
- ::first-line - seleciona a primeira linha do texto
- ::selection - seleciona a parte do texto que foi selecionada pelo usuário

2.3 Propriedades CSS Essenciais

Texto e Tipografia:

```
p {
    color: #333;
    font-family: 'Arial', sans-serif;
    font-size: 16px;
    font-weight: bold;
    line-height: 1.5;
    text-align: center;
    text-decoration: underline;
    text-transform: uppercase;
    letter-spacing: 1px;
    word-spacing: 2px;
}
```

Box Model:

```
div {
    width: 300px;
    height: 200px;
    padding: 20px;
    border: 1px solid #000;
    margin: 10px;
    box-sizing: border-box; /* Inclui padding e border na largura/altura */
}
```

Cores e Fundos:

```
div {
    color: #333;
    background-color: #f5f5f5;
    background-image: url('imagem.jpg');
    background-repeat: no-repeat;
    background-position: center;
    background-size: cover;
    opacity: 0.9;
}
```

Posicionamento:

```
div {
    position: relative; /* static, relative, absolute, fixed, sticky */
    top: 10px;
    right: 20px;
    bottom: 30px;
    left: 40px;
    z-index: 10;
}
```

Display e Visibilidade:

```
div {
    display: block; /* inline, inline-block, flex, grid, none */
    visibility: visible; /* hidden */
    overflow: auto; /* visible, hidden, scroll */
}
```

2.4 Layouts CSS Modernos

Flexbox:
O Flexbox é um modelo de layout unidimensional, ideal para organizar itens em uma linha ou coluna:

```css
.container {
    display: flex;
    flex-direction: row; /* row, column, row-reverse, column-reverse */
    flex-wrap: wrap; /* nowrap, wrap, wrap-reverse */
    justify-content: space-between; /* flex-start, flex-end, center, space-around, space-evenly */
    align-items: center; /* flex-start, flex-end, stretch, baseline */
    gap: 10px;
}

.item {
    flex: 1; /* flex-grow, flex-shrink, flex-basis combinados */
    align-self: flex-start; /* sobrescreve align-items para um item específico */
    order: 2; /* altera a ordem de exibição */
}
```

Grid:
O Grid é um modelo de layout bidimensional, perfeito para layouts complexos com linhas e colunas:

```css
.container {
    display: grid;
    grid-template-columns: repeat(3, 1fr); /* 3 colunas de tamanho igual */
    grid-template-rows: 100px auto 100px; /* altura específica para linhas */
    grid-gap: 10px; /* espaçamento entre células */
    grid-template-areas:
        "header header header"
        "sidebar content content"
        "footer footer footer";
}

.header { grid-area: header; }
.sidebar { grid-area: sidebar; }
.content { grid-area: content; }
.footer { grid-area: footer; }

.item {
    grid-column: 1 / 3; /* do início da coluna 1 até o início da coluna 3 */
    grid-row: 2 / 4; /* do início da linha 2 até o início da linha 4 */
}
```

2.5 Recursos Avançados do CSS3

Transformações:

```css
div {
    transform: rotate(45deg) scale(1.5) translateX(20px) skewY(10deg);
    transform-origin: center center;
}
```

Transições:

```
button {
    background-color: blue;
    transition: background-color 0.3s ease, transform 0.5s ease-in-out;
}

button:hover {
    background-color: red;
    transform: scale(1.1);
}
```

Animações:

```
@keyframes pulsar {
    0% { transform: scale(1); opacity: 1; }
    50% { transform: scale(1.2); opacity: 0.7; }
    100% { transform: scale(1); opacity: 1; }
}

.elemento {
    animation: pulsar 2s infinite ease-in-out;
}
```

Gradientes:

```
div {
    /* Gradiente linear */
    background: linear-gradient(to right, #ff0000, #0000ff);

    /* Gradiente radial */
    background: radial-gradient(circle, #ff0000, #0000ff);

    /* Gradiente cônico */
    background: conic-gradient(from 45deg, #ff0000, #0000ff, #ff0000);
}
```

Sombras:

```
div {
    /* Sombra de caixa */
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

    /* Sombra de texto */
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
}
```

Variáveis CSS (Custom Properties):

```css
:root {
    --cor-primaria: #3498db;
    --cor-secundaria: #2ecc71;
    --espacamento-padrao: 20px;
    --fonte-principal: 'Roboto', sans-serif;
}

.elemento {
    color: var(--cor-primaria);
    margin: var(--espacamento-padrao);
    font-family: var(--fonte-principal);
}
```

3. Git e Github

3.1 Introdução ao Controle de Versão

O controle de versão é um sistema que registra alterações em arquivos ao longo do tempo, permitindo recuperar versões específicas posteriormente. É essencial para o desenvolvimento de software, especialmente em equipes.

Benefícios do Controle de Versão:

- Histórico completo de alterações
- Trabalho simultâneo em equipe
- Ramificação para desenvolvimento paralelo
- Reversão para versões anteriores
- Rastreamento de problemas e bugs
- Documentação natural do projeto

Tipos de Sistemas de Controle de Versão:

- **Centralizados (CVCS)**: Um servidor central contém todos os arquivos versionados (ex: SVN)
- **Distribuídos (DVCS)**: Cada usuário tem uma cópia completa do repositório (ex: Git, Mercurial)

3.2 Conceitos Fundamentais do Git

Git é um sistema de controle de versão distribuído, criado por Linus Torvalds em 2005, inicialmente para o desenvolvimento do kernel Linux.

Áreas do Git:

- **Working Directory**: Diretório de trabalho onde você edita os arquivos
- **Staging Area (Index)**: Área intermediária onde as alterações são preparadas para commit
- **Repository (Local)**: Banco de dados local que armazena o histórico de commits
- **Repository (Remote)**: Repositório remoto, geralmente hospedado em serviços como GitHub

Estados dos Arquivos:

- **Untracked**: Arquivos que o Git não está acompanhando
- **Tracked**: Arquivos que o Git está acompanhando
    - **Unmodified**: Arquivos que não foram modificados desde o último commit
    - **Modified**: Arquivos que foram modificados desde o último commit

- **Staged**: Arquivos modificados que foram adicionados à área de staging

Objetos do Git:

- **Blob**: Conteúdo de um arquivo
- **Tree**: Diretório que pode conter blobs e outras trees
- **Commit**: Snapshot do repositório em um determinado momento
- **Tag**: Referência a um commit específico, geralmente usado para marcar versões

3.3 Comandos Básicos do Git

Configuração Inicial:

```
# Configurar nome e email
git config --global user.name "Seu Nome"
git config --global user.email "seu.email@exemplo.com"

# Configurar editor padrão
git config --global core.editor "code --wait"

# Verificar configurações
git config --list
```

Iniciar um Repositório:

```
# Criar um novo repositório
git init

# Clonar um repositório existente
git clone https://github.com/usuario/repositorio.git
```

Operações Básicas:

```
# Verificar status do repositório
git status

# Adicionar arquivos à área de staging
git add arquivo.txt      # Adicionar um arquivo específico
git add .                # Adicionar todos os arquivos modificados

# Criar um commit
git commit -m "Mensagem do commit"

# Ver histórico de commits
git log
git log --oneline        # Formato resumido
git log --graph          # Visualização gráfica

# Desfazer alterações
git checkout -- arquivo.txt  # Descartar alterações não staged
git reset HEAD arquivo.txt   # Remover arquivo da área de staging
git revert commit_hash       # Criar novo commit que desfaz alterações
git reset --hard commit_hash # Voltar para um commit específico (cuidado!)
```

Branches (Ramificações):

```
# Listar branches
git branch

# Criar um novo branch
git branch nome-do-branch

# Mudar para um branch
git checkout nome-do-branch

# Criar e mudar para um novo branch (atalho)
git checkout -b nome-do-branch

# Mesclar branches
git merge nome-do-branch

# Excluir um branch
git branch -d nome-do-branch  # Excluir branch já mesclado
git branch -D nome-do-branch  # Forçar exclusão
```

Repositórios Remotos:

```
# Listar repositórios remotos
git remote -v

# Adicionar um repositório remoto
git remote add origin https://github.com/usuario/repositorio.git

# Enviar alterações para o repositório remoto
git push origin branch-name

# Obter alterações do repositório remoto
git fetch origin
git pull origin branch-name  # fetch + merge
```

3.4 Fluxos de Trabalho com Git

Git Flow:
Um modelo de ramificação que define papéis específicos para diferentes branches:

- **master**: Código em produção
- **develop**: Código em desenvolvimento
- **feature/**: Branches para novas funcionalidades
- **release/**: Branches para preparação de releases
- **hotfix/**: Branches para correções urgentes em produção

GitHub Flow:
Um fluxo de trabalho mais simples, baseado em pull requests:

1. Criar um branch a partir do main
2. Fazer alterações e commits
3. Abrir um pull request

4. Discutir e revisar o código
5. Mesclar com o main quando aprovado
6. Implantar

Trunk-Based Development:
Um fluxo onde a maioria do desenvolvimento acontece diretamente no branch principal:

- Commits frequentes no branch principal
- Branches de curta duração para funcionalidades
- Uso de feature flags para funcionalidades incompletas
- Integração contínua rigorosa

3.5 GitHub e Colaboração

GitHub é uma plataforma de hospedagem de código-fonte que utiliza o Git para controle de versão, adicionando recursos de colaboração e gerenciamento de projetos.

Recursos do GitHub:

Repositórios:

- Públicos ou privados
- README.md para documentação
- Licenças de código aberto
- Arquivos .gitignore

Pull Requests (PRs):

- Propor alterações ao código
- Revisão de código
- Discussões e comentários
- Integração com CI/CD

Issues:

- Rastreamento de bugs
- Solicitações de recursos
- Planejamento de tarefas
- Referência a commits e PRs

Actions:

- Automação de fluxos de trabalho
- Integração contínua
- Implantação contínua
- Testes automatizados

Pages:

- Hospedagem de sites estáticos
- Documentação de projetos
- Portfólios

Projetos e Quadros Kanban:

- Gerenciamento visual de tarefas

- Organização de issues
- Acompanhamento de progresso

Boas Práticas para Colaboração:

- Escrever mensagens de commit claras e descritivas
- Manter PRs pequenos e focados
- Revisar código cuidadosamente
- Documentar o projeto adequadamente
- Usar issues para rastrear tarefas
- Seguir as diretrizes de contribuição do projeto

4. Tabelas e Formulários HTML

4.1 Tabelas HTML

As tabelas HTML são usadas para apresentar dados em linhas e colunas. Embora não devam ser usadas para layout, são essenciais para exibir dados tabulares.

Estrutura Básica de uma Tabela:

```
<table>
  <caption>Título da Tabela</caption>

  <thead>
    <tr>
      <th>Cabeçalho 1</th>
      <th>Cabeçalho 2</th>
      <th>Cabeçalho 3</th>
    </tr>
  </thead>

  <tbody>
    <tr>
      <td>Dado 1,1</td>
      <td>Dado 1,2</td>
      <td>Dado 1,3</td>
    </tr>
    <tr>
      <td>Dado 2,1</td>
      <td>Dado 2,2</td>
      <td>Dado 2,3</td>
    </tr>
  </tbody>

  <tfoot>
    <tr>
      <td colspan="2">Total:</td>
      <td>Valor</td>
    </tr>
  </tfoot>
</table>
```

Elementos de Tabela:

- <table>: Define uma tabela
- <caption>: Título da tabela
- <thead>: Grupo de linhas de cabeçalho
- <tbody>: Grupo de linhas de conteúdo
- <tfoot>: Grupo de linhas de rodapé
- <tr>: Linha da tabela
- <th>: Célula de cabeçalho
- <td>: Célula de dados

Atributos de Tabela:

- colspan: Número de colunas que uma célula deve ocupar
- rowspan: Número de linhas que uma célula deve ocupar
- headers: Associa células de dados com células de cabeçalho (para acessibilidade)
- scope: Define o escopo de uma célula de cabeçalho (row, col, rowgroup, colgroup)

Estilização de Tabelas com CSS:

```
table {
    width: 100%;
    border-collapse: collapse;
    margin-bottom: 20px;
}

caption {
    font-weight: bold;
    margin-bottom: 10px;
}

th, td {
    padding: 12px;
    text-align: left;
    border-bottom: 1px solid #ddd;
}

th {
    background-color: #f2f2f2;
    font-weight: bold;
}

tr:hover {
    background-color: #f5f5f5;
}

tr:nth-child(even) {
    background-color: #f9f9f9;
}
```

4.2 Formulários HTML

Os formulários HTML permitem coletar dados dos usuários para processamento. São essenciais para interação em aplicações web.

Estrutura Básica de um Formulário:

```
<form action="/processar" method="post">
  <fieldset>
    <legend>Informações Pessoais</legend>

    <div class="form-group">
      <label for="nome">Nome:</label>
      <input type="text" id="nome" name="nome" required>
    </div>

    <div class="form-group">
      <label for="email">Email:</label>
      <input type="email" id="email" name="email" required>
    </div>
  </fieldset>

  <fieldset>
    <legend>Preferências</legend>

    <div class="form-group">
      <label for="cor">Cor Favorita:</label>
      <select id="cor" name="cor">
        <option value="vermelho">Vermelho</option>
        <option value="azul">Azul</option>
        <option value="verde">Verde</option>
      </select>
    </div>

    <div class="form-group">
      <p>Interesses:</p>
      <label>
        <input type="checkbox" name="interesses" value="tecnologia"> Tecnologia
      </label>
      <label>
        <input type="checkbox" name="interesses" value="esportes"> Esportes
      </label>
      <label>
        <input type="checkbox" name="interesses" value="arte"> Arte
      </label>
    </div>
  </fieldset>

  <div class="form-group">
    <label for="mensagem">Mensagem:</label>
    <textarea id="mensagem" name="mensagem" rows="4"></textarea>
  </div>

  <div class="form-actions">
    <button type="reset">Limpar</button>
    <button type="submit">Enviar</button>
  </div>
</form>
```

Elementos de Formulário:

- <form>: Container para elementos de formulário
- <fieldset>: Agrupa elementos relacionados
- <legend>: Título para um fieldset
- <label>: Rótulo para um elemento de formulário
- <input>: Campo de entrada (vários tipos)
- <select> e <option>: Lista suspensa
- <textarea>: Campo de texto multilinha
- <button>: Botão

Tipos de Input:

- text: Texto de linha única
- password: Campo de senha
- email: Endereço de email
- number: Valor numérico
- tel: Número de telefone
- url: URL
- date, time, datetime-local: Valores de data e hora
- checkbox: Caixa de seleção
- radio: Botão de opção
- file: Upload de arquivo
- color: Seletor de cor
- range: Controle deslizante
- search: Campo de pesquisa
- hidden: Campo oculto

Atributos de Formulário:

- required: Campo obrigatório
- placeholder: Texto de exemplo
- pattern: Expressão regular para validação
- min, max, step: Limites para campos numéricos
- maxlength, minlength: Limites para campos de texto
- autocomplete: Controle de preenchimento automático
- disabled: Desabilita o campo
- readonly: Campo somente leitura
- autofocus: Foco automático ao carregar a página

Validação de Formulários:

Validação HTML5:

```
<input type="email" required pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}{{content}}quot;>
```

Validação com JavaScript:

```
const form = document.querySelector('form');

form.addEventListener('submit', function(event) {
    const email = document.getElementById('email');
    const emailRegex = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;

    if (!emailRegex.test(email.value)) {
        event.preventDefault();
        alert('Por favor, insira um email válido.');
        email.focus();
    }
});
```

Estilização de Formulários com CSS:

```
form {
    max-width: 600px;
    margin: 0 auto;
}

fieldset {
    margin-bottom: 20px;
    padding: 15px;
    border: 1px solid #ddd;
    border-radius: 4px;
}

legend {
    padding: 0 10px;
    font-weight: bold;
}

.form-group {
    margin-bottom: 15px;
}

label {
    display: block;
    margin-bottom: 5px;
    font-weight: bold;
}

input[type="text"],
input[type="email"],
input[type="password"],
select,
textarea {
    width: 100%;
    padding: 10px;
    border: 1px solid #ddd;
    border-radius: 4px;
    font-size: 16px;
```

```
}

input:focus,
select:focus,
textarea:focus {
    outline: none;
    border-color: #4CAF50;
    box-shadow: 0 0 5px rgba(76, 175, 80, 0.5);
}

input:invalid {
    border-color: #f44336;
}

button {
    padding: 10px 20px;
    background-color: #4CAF50;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 16px;
}

button[type="reset"] {
    background-color: #f44336;
}

button:hover {
    opacity: 0.9;
}

.form-actions {
    display: flex;
    justify-content: space-between;
}
```

5. Responsividade

5.1 Princípios de Design Responsivo

O design responsivo é uma abordagem que faz com que as páginas web se adaptem a diferentes tamanhos de tela e dispositivos, proporcionando uma experiência de usuário otimizada.

Princípios Fundamentais:

1. **Layout Fluido**: Usar unidades relativas (%, em, rem) em vez de fixas (px)
2. **Media Queries**: Aplicar estilos diferentes com base nas características do dispositivo
3. **Imagens Flexíveis**: Garantir que as imagens se adaptem ao tamanho do container
4. **Mobile First**: Começar o design para dispositivos móveis e depois expandir para telas maiores
5. **Breakpoints Estratégicos**: Definir pontos de quebra baseados no conteúdo, não em dispositivos específicos

5.2 Viewport e Unidades Relativas

Meta Tag Viewport:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Esta meta tag é essencial para o design responsivo, pois instrui o navegador a definir a largura da viewport igual à largura do dispositivo e a escala inicial como 1.0 (sem zoom).

Unidades Relativas:

- **Percentagem (%)**: Relativa ao elemento pai
- **em**: Relativa ao tamanho da fonte do elemento pai
- **rem**: Relativa ao tamanho da fonte do elemento raiz (html)
- **vw**: 1% da largura da viewport
- **vh**: 1% da altura da viewport
- **vmin**: 1% da menor dimensão da viewport
- **vmax**: 1% da maior dimensão da viewport

Exemplo de Uso:

```
html {
    font-size: 16px; /* Base para cálculos rem */
}

body {
    font-size: 1rem; /* 16px */
    line-height: 1.5;
}

h1 {
    font-size: 2.5rem; /* 40px */
}

.container {
    width: 90%; /* 90% da largura do pai */
    max-width: 1200px;
    margin: 0 auto;
}

.hero {
    height: 50vh; /* 50% da altura da viewport */
}
```

5.3 Media Queries

Media queries permitem aplicar estilos CSS com base em características do dispositivo, como largura, altura, orientação e tipo de mídia.

Sintaxe Básica:

```
@media media-type and (media-feature) {
    /* Regras CSS */
}
```

Tipos de Mídia:

- all: Todos os dispositivos
- screen: Telas de computador, tablet, smartphone
- print: Impressoras
- speech: Leitores de tela

Características de Mídia Comuns:

- width, min-width, max-width: Largura da viewport
- height, min-height, max-height: Altura da viewport
- orientation: Orientação do dispositivo (portrait ou landscape)
- aspect-ratio: Proporção da viewport
- resolution: Resolução do dispositivo
- hover: Capacidade de hover do dispositivo

Exemplos de Media Queries:

```
/* Estilos base (mobile first) */
.container {
    padding: 15px;
}

.nav {
    display: none;
}

.menu-toggle {
    display: block;
}

/* Tablets */
@media screen and (min-width: 768px) {
    .container {
        padding: 20px;
    }

    .column {
        width: 50%;
        float: left;
    }
}

/* Desktops */
@media screen and (min-width: 1024px) {
    .container {
        padding: 30px;
    }

    .nav {
        display: block;
    }
```

```
    .menu-toggle {
        display: none;
    }

    .column {
        width: 33.33%;
    }
}

/* Impressão */
@media print {
    .nav, .footer, .ads {
        display: none;
    }

    body {
        font-size: 12pt;
        line-height: 1.5;
        color: #000;
    }

    a::after {
        content: " (" attr(href) ")";
    }
}
```

5.4 Imagens Responsivas

CSS para Imagens Responsivas:

```
img {
    max-width: 100%;
    height: auto;
}
```

Elemento <picture> para Art Direction:

```
<picture>
    <source media="(min-width: 1200px)" srcset="imagem-grande.jpg">
    <source media="(min-width: 768px)" srcset="imagem-media.jpg">
    <img src="imagem-pequena.jpg" alt="Descrição da imagem">
</picture>
```

Atributo srcset para Densidade de Pixels:

```
<img
    src="imagem-1x.jpg"
    srcset="imagem-1x.jpg 1x, imagem-2x.jpg 2x, imagem-3x.jpg 3x"
    alt="Descrição da imagem">
```

Atributo srcset com sizes para Diferentes Tamanhos:

```
<img
    src="imagem-fallback.jpg"
    srcset="imagem-320w.jpg 320w, imagem-480w.jpg 480w, imagem-800w.jpg 800w"
    sizes="(max-width: 320px) 280px, (max-width: 480px) 440px, 800px"
    alt="Descrição da imagem">
```

5.5 Layouts Responsivos

Layout Fluido com Percentagens:

```
.container {
    width: 90%;
    max-width: 1200px;
    margin: 0 auto;
}

.row::after {
    content: "";
    display: table;
    clear: both;
}

.col {
    float: left;
    padding: 0 15px;
    box-sizing: border-box;
}

.col-1 { width: 8.33%; }
.col-2 { width: 16.66%; }
.col-3 { width: 25%; }
.col-4 { width: 33.33%; }
.col-5 { width: 41.66%; }
.col-6 { width: 50%; }
.col-7 { width: 58.33%; }
.col-8 { width: 66.66%; }
.col-9 { width: 75%; }
.col-10 { width: 83.33%; }
.col-11 { width: 91.66%; }
.col-12 { width: 100%; }

@media screen and (max-width: 768px) {
    .col {
        width: 100%;
    }
}
```

Layout com Flexbox:

```
.container {
    max-width: 1200px;
    margin: 0 auto;
    padding: 0 15px;
}

.row {
    display: flex;
    flex-wrap: wrap;
    margin: 0 -15px;
}

.col {
    flex: 1 0 0%;
    padding: 0 15px;
    box-sizing: border-box;
}

@media screen and (max-width: 768px) {
    .row {
        flex-direction: column;
    }
}
```

Layout com Grid:

```css
.container {
    max-width: 1200px;
    margin: 0 auto;
    padding: 0 15px;
}

.grid {
    display: grid;
    grid-template-columns: repeat(12, 1fr);
    gap: 30px;
}

.span-1 { grid-column: span 1; }
.span-2 { grid-column: span 2; }
.span-3 { grid-column: span 3; }
.span-4 { grid-column: span 4; }
.span-5 { grid-column: span 5; }
.span-6 { grid-column: span 6; }
.span-7 { grid-column: span 7; }
.span-8 { grid-column: span 8; }
.span-9 { grid-column: span 9; }
.span-10 { grid-column: span 10; }
.span-11 { grid-column: span 11; }
.span-12 { grid-column: span 12; }

@media screen and (max-width: 768px) {
    .grid {
        grid-template-columns: 1fr;
    }

    [class*="span-"] {
        grid-column: 1;
    }
}
```

5.6 Padrões de Design Responsivo

Padrão Mostly Fluid:

- Layout fluido que se ajusta a diferentes tamanhos de tela
- Em telas menores, as colunas empilham verticalmente
- Mudanças mínimas entre breakpoints

Padrão Column Drop:

- Colunas são reorganizadas verticalmente em telas menores
- Cada coluna "cai" abaixo da anterior em um breakpoint específico

Padrão Layout Shifter:

- Mudanças significativas no layout entre breakpoints
- Não apenas empilhamento, mas reorganização completa

Padrão Off Canvas:

- Conteúdo menos importante (como navegação) fica fora da tela em dispositivos menores
- Pode ser acessado através de um botão (menu hambúrguer)

6. Pré-processadores CSS

6.1 Introdução aos Pré-processadores

Pré-processadores CSS são ferramentas que estendem as funcionalidades do CSS, adicionando recursos como variáveis, aninhamento, mixins e funções. O código escrito no pré-processador é compilado para CSS padrão que os navegadores podem entender.

Benefícios dos Pré-processadores:

- Código mais organizado e modular
- Reutilização através de variáveis e mixins
- Redução de repetição (DRY - Don't Repeat Yourself)
- Manutenção simplificada
- Recursos avançados não disponíveis no CSS puro

Pré-processadores Populares:

- **Sass/SCSS**: O mais popular e maduro
- **Less**: Mais próximo da sintaxe CSS
- **Stylus**: Sintaxe mais flexível e minimalista

6.2 SASS/SCSS

SASS (Syntactically Awesome Style Sheets) é um pré-processador CSS poderoso. Tem duas sintaxes: SASS (indentação) e SCSS (chaves e ponto-e-vírgula, mais próxima do CSS).

Instalação e Compilação:

```
# Instalação via npm
npm install -g sass

# Compilação básica
sass input.scss output.css

# Compilação com watch
sass --watch input.scss:output.css

# Compilação de diretório
sass --watch scss/:css/
```

Variáveis:

```
// Definindo variáveis
$cor-primaria: #3498db;
$cor-secundaria: #2ecc71;
$fonte-principal: 'Roboto', sans-serif;
$espacamento-padrao: 20px;

// Usando variáveis
body {
    font-family: $fonte-principal;
    color: $cor-primaria;
    margin: $espacamento-padrao;
}

a {
    color: $cor-secundaria;
}
```

Aninhamento:

```
nav {
    background-color: #333;

    ul {
        list-style: none;
        padding: 0;
        margin: 0;

        li {
            display: inline-block;

            a {
                display: block;
                padding: 10px 15px;
                color: white;
                text-decoration: none;

                &:hover {
                    background-color: #555;
                }
            }
        }
    }
}
```

Partials e Importação:

```scss
// _reset.scss
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

// _variables.scss
$cor-primaria: #3498db;
$cor-secundaria: #2ecc71;

// main.scss
@import 'reset';
@import 'variables';

body {
    font-family: sans-serif;
    color: $cor-primaria;
}
```

Mixins:

```scss
// Definindo um mixin
@mixin flexbox($direction: row, $justify: center, $align: center) {
    display: flex;
    flex-direction: $direction;
    justify-content: $justify;
    align-items: $align;
}

@mixin border-radius($radius) {
    -webkit-border-radius: $radius;
    -moz-border-radius: $radius;
    border-radius: $radius;
}

// Usando mixins
.container {
    @include flexbox(column, flex-start, stretch);
    padding: 20px;
}

.button {
    @include border-radius(5px);
    padding: 10px 15px;
}
```

Extensão/Herança:

```
%button-base {
    padding: 10px 15px;
    border: none;
    border-radius: 4px;
    font-size: 16px;
    cursor: pointer;
}

.button-primary {
    @extend %button-base;
    background-color: $cor-primaria;
    color: white;
}

.button-secondary {
    @extend %button-base;
    background-color: $cor-secundaria;
    color: white;
}

.button-outline {
    @extend %button-base;
    background-color: transparent;
    border: 1px solid $cor-primaria;
    color: $cor-primaria;
}
```

Funções:

```scss
// Funções integradas
$base-font-size: 16px;

h1 {
  font-size: $base-font-size * 2.5;
  color: darken($cor-primaria, 10%);
}

h2 {
  font-size: $base-font-size * 2;
  color: lighten($cor-primaria, 10%);
}

.overlay {
  background-color: rgba(0, 0, 0, 0.7);
}

// Funções personalizadas
@function calculate-width($col, $total: 12) {
  @return percentage($col / $total);
}

.col-4 {
  width: calculate-width(4);  // 33.33333%
}
```

Diretivas de Controle:

```
// Condicionais
@mixin text-color($bg-color) {
    background-color: $bg-color;

    @if lightness($bg-color) > 50% {
        color: #000;
    } @else {
        color: #fff;
    }
}

.light-box {
    @include text-color(#f8f8f8);
}

.dark-box {
    @include text-color(#333);
}

// Loops
$grid-columns: 12;

@for $i from 1 through $grid-columns {
    .col-#{$i} {
        width: calculate-width($i, $grid-columns);
    }
}

$sizes: (small: 12px, medium: 16px, large: 24px);

@each $name, $size in $sizes {
    .text-#{$name} {
        font-size: $size;
    }
}
```

6.3 Less

Less é um pré-processador CSS que estende o CSS com recursos dinâmicos. Sua sintaxe é muito próxima do CSS, tornando-o fácil de aprender.

Instalação e Compilação:

```
# Instalação via npm
npm install -g less

# Compilação básica
lessc input.less output.css

# Compilação com watch (requer lessc-watch)
lessc --watch input.less output.css
```

Variáveis:

```
// Definindo variáveis
@cor-primaria: #3498db;
@cor-secundaria: #2ecc71;
@fonte-principal: 'Roboto', sans-serif;
@espacamento-padrao: 20px;

// Usando variáveis
body {
    font-family: @fonte-principal;
    color: @cor-primaria;
    margin: @espacamento-padrao;
}

a {
    color: @cor-secundaria;
}
```

Aninhamento:

```
nav {
    background-color: #333;

    ul {
        list-style: none;
        padding: 0;
        margin: 0;

        li {
            display: inline-block;

            a {
                display: block;
                padding: 10px 15px;
                color: white;
                text-decoration: none;

                &:hover {
                    background-color: #555;
                }
            }
        }
    }
}
```

Importação:

```
// reset.less
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

// variables.less
@cor-primaria: #3498db;
@cor-secundaria: #2ecc71;

// main.less
@import 'reset.less';
@import 'variables.less';

body {
    font-family: sans-serif;
    color: @cor-primaria;
}
```

Mixins:

```
// Definindo um mixin
.flexbox(@direction: row, @justify: center, @align: center) {
    display: flex;
    flex-direction: @direction;
    justify-content: @justify;
    align-items: @align;
}

.border-radius(@radius) {
    -webkit-border-radius: @radius;
    -moz-border-radius: @radius;
    border-radius: @radius;
}

// Usando mixins
.container {
    .flexbox(column, flex-start, stretch);
    padding: 20px;
}

.button {
    .border-radius(5px);
    padding: 10px 15px;
}
```

Operações:

```
@base-font-size: 16px;

h1 {
    font-size: @base-font-size * 2.5;
    margin-bottom: @base-font-size + 5px;
}

.container {
    width: 100% / 3;  // 33.33333%
}
```

Funções:

```
@cor-primaria: #3498db;

h1 {
    color: darken(@cor-primaria, 10%);
}

h2 {
    color: lighten(@cor-primaria, 10%);
}

.overlay {
    background-color: fade(#000, 70%);
}
```

Condicionais e Loops:

```
// Condicionais com mixins guardados
.text-color(@bg-color) when (lightness(@bg-color) > 50%) {
    color: #000;
}

.text-color(@bg-color) when (lightness(@bg-color) <= 50%) {
    color: #fff;
}

.light-box {
    background-color: #f8f8f8;
    .text-color(#f8f8f8);
}

.dark-box {
    background-color: #333;
    .text-color(#333);
}

// Loops com mixins recursivos
.generate-columns(@i: 1) when (@i <= 12) {
    .col-@{i} {
        width: (@i * 100% / 12);
    }
    .generate-columns(@i + 1);
}

.generate-columns();
```

6.4 Stylus

Stylus é um pré-processador CSS expressivo que oferece grande flexibilidade na sintaxe, permitindo omitir chaves, ponto-e-vírgula e até mesmo dois-pontos.

Instalação e Compilação:

```
# Instalação via npm
npm install -g stylus

# Compilação básica
stylus input.styl -o output.css

# Compilação com watch
stylus -w input.styl -o output.css
```

Variáveis:

```
// Definindo variáveis
cor-primaria = #3498db
cor-secundaria = #2ecc71
fonte-principal = 'Roboto', sans-serif
espacamento-padrao = 20px

// Usando variáveis
body
  font-family fonte-principal
  color cor-primaria
  margin espacamento-padrao

a
  color cor-secundaria
```

Aninhamento:

```
nav
  background-color #333

  ul
    list-style none
    padding 0
    margin 0

    li
      display inline-block

      a
        display block
        padding 10px 15px
        color white
        text-decoration none

        &:hover
          background-color #555
```

Importação:

```
// reset.styl
*
  margin 0
  padding 0
  box-sizing border-box

// variables.styl
cor-primaria = #3498db
cor-secundaria = #2ecc71

// main.styl
@import 'reset'
@import 'variables'

body
  font-family sans-serif
  color cor-primaria
```

Mixins:

```
// Definindo um mixin
flexbox(direction = row, justify = center, align = center)
  display flex
  flex-direction direction
  justify-content justify
  align-items align

border-radius(radius)
  -webkit-border-radius radius
  -moz-border-radius radius
  border-radius radius

// Usando mixins
.container
  flexbox(column, flex-start, stretch)
  padding 20px

.button
  border-radius(5px)
  padding 10px 15px
```

Funções:

```
base-font-size = 16px

h1
  font-size base-font-size * 2.5
  color darken(cor-primaria, 10%)

h2
  font-size base-font-size * 2
  color lighten(cor-primaria, 10%)

.overlay
  background-color rgba(0, 0, 0, 0.7)

// Funções personalizadas
calculate-width(col, total = 12)
  return (col / total) * 100%

.col-4
  width calculate-width(4)  // 33.33333%
```

Condicionais e Loops:

```
// Condicionais
text-color(bg-color)
  background-color bg-color

  if lightness(bg-color) > 50%
    color #000
  else
    color #fff

.light-box
  text-color(#f8f8f8)

.dark-box
  text-color(#333)

// Loops
grid-columns = 12

for i in 1..grid-columns
  .col-{i}
    width (i / grid-columns) * 100%

sizes = {
  small: 12px,
  medium: 16px,
  large: 24px
}

for name, size in sizes
  .text-{name}
    font-size size
```

6.5 Organização de Projetos com Pré-processadores

Estrutura de Arquivos (Exemplo com SCSS):

```
scss/
|-- base/
|   |-- _reset.scss
|   |-- _typography.scss
|   |-- _utilities.scss
|
|-- components/
|   |-- _buttons.scss
|   |-- _forms.scss
|   |-- _navigation.scss
|   |-- _cards.scss
|
|-- layout/
|   |-- _header.scss
|   |-- _footer.scss
|   |-- _grid.scss
|   |-- _sidebar.scss
|
|-- pages/
|   |-- _home.scss
|   |-- _about.scss
|   |-- _contact.scss
|
|-- themes/
|   |-- _default.scss
|   |-- _dark.scss
|
|-- abstracts/
|   |-- _variables.scss
|   |-- _mixins.scss
|   |-- _functions.scss
|
|-- vendors/
|   |-- _bootstrap.scss
|   |-- _jquery-ui.scss
|
|-- main.scss
```

Arquivo Principal (main.scss):

```
// Abstracts
@import 'abstracts/variables';
@import 'abstracts/mixins';
@import 'abstracts/functions';

// Vendors
@import 'vendors/bootstrap';
@import 'vendors/jquery-ui';

// Base
@import 'base/reset';
@import 'base/typography';
@import 'base/utilities';

// Layout
@import 'layout/grid';
@import 'layout/header';
@import 'layout/footer';
@import 'layout/sidebar';

// Components
@import 'components/buttons';
@import 'components/forms';
@import 'components/navigation';
@import 'components/cards';

// Pages
@import 'pages/home';
@import 'pages/about';
@import 'pages/contact';

// Themes
@import 'themes/default';
@import 'themes/dark';
```

Boas Práticas:

1. **Modularização**: Divida o código em arquivos pequenos e focados
2. **Nomenclatura Consistente**: Use convenções como BEM (Block Element Modifier)
3. **Comentários**: Documente seções importantes e decisões de design
4. **Variáveis Globais**: Centralize variáveis em um único arquivo
5. **Reutilização**: Crie mixins e placeholders para código repetitivo
6. **Organização**: Siga uma estrutura de arquivos lógica
7. **Minificação**: Configure a compilação para produzir CSS minificado
8. **Sourcemaps**: Habilite para facilitar a depuração

Recursos Adicionais

Documentação Oficial

- HTML5 - MDN Web Docs (https://developer.mozilla.org/pt-BR/docs/Web/HTML)
- CSS3 - MDN Web Docs (https://developer.mozilla.org/pt-BR/docs/Web/CSS)

- Git Documentation (https://git-scm.com/doc)
- GitHub Docs (https://docs.github.com/)
- Sass Documentation (https://sass-lang.com/documentation)
- Less Documentation (https://lesscss.org/usage/)
- Stylus Documentation (https://stylus-lang.com/)

Tutoriais e Cursos

- HTML & CSS - W3Schools (https://www.w3schools.com/)
- Git & GitHub Crash Course - Traversy Media (https://www.youtube.com/watch?v=SWYqp7iY_Tc)
- Responsive Web Design Fundamentals - Google Developers (https://developers.google.com/web/fundamentals/design-and-ux/responsive)
- Sass Crash Course - Traversy Media (https://www.youtube.com/watch?v=nu5mdN2JIwM)

Ferramentas

- Can I Use (https://caniuse.com/) - Compatibilidade de recursos web
- HTML Validator (https://validator.w3.org/)
- CSS Validator (https://jigsaw.w3.org/css-validator/)
- Autoprefixer (https://autoprefixer.github.io/) - Adiciona prefixos de fornecedor automaticamente
- Sassmeister (https://www.sassmeister.com/) - Playground online para Sass
- Less2CSS (http://lesscss.org/less-preview/) - Conversor online de Less para CSS

Exercícios Práticos

1. **Estrutura HTML Semântica**:
    - Crie a estrutura HTML de um blog com cabeçalho, navegação, artigos, barra lateral e rodapé.
    - Use elementos semânticos apropriados.
    - Adicione metadados para SEO.
2. **Estilização com CSS**:
    - Estilize a estrutura do blog criada no exercício anterior.
    - Implemente um esquema de cores consistente.
    - Crie um layout com Flexbox ou Grid.
    - Adicione transições e animações para interações.
3. **Controle de Versão**:
    - Inicialize um repositório Git para o projeto do blog.
    - Faça commits para cada etapa significativa.
    - Crie um branch para uma nova funcionalidade.
    - Mescle o branch com o branch principal.
    - Publique o repositório no GitHub.
4. **Formulário de Contato**:
    - Adicione um formulário de contato ao blog.
    - Inclua campos para nome, email, assunto e mensagem.
    - Implemente validação HTML5.
    - Estilize o formulário para ser atraente e usável.

5. **Design Responsivo**:

    - Torne o blog responsivo para dispositivos móveis, tablets e desktops.
    - Implemente um menu hambúrguer para dispositivos móveis.
    - Teste em diferentes tamanhos de tela.
    - Use media queries para ajustar o layout.

6. **Pré-processador CSS**:

    - Converta os estilos CSS do blog para um pré-processador (Sass, Less ou Stylus).
    - Organize o código em arquivos parciais.
    - Use variáveis para cores e tamanhos.
    - Implemente mixins para código reutilizável.
    - Compile o código para CSS.