Guia Completo de Programação Web

MÓDULO 1: INTRODUÇÃO AO CURSO E CARREIRA DE DESENVOLVEDOR WEB

1.1 O que é Desenvolvimento Web?

Desenvolvimento web é o processo de criar aplicações e sites que funcionam na internet. É uma área em constante evolução que combina criatividade, lógica e tecnologia para resolver problemas reais.

1.2 Por que Estudar Desenvolvimento Web?

A demanda por desenvolvedores web está em crescimento exponencial. Empresas de todos os tamanhos precisam de presença na internet, criando oportunidades abundantes. Os salários são competitivos, o trabalho remoto é comum, e você pode construir seus próprios projetos.

1.3 Características de um Desenvolvedor Web

Um bom desenvolvedor web possui: pensamento lógico, capacidade de resolver problemas, paciência para debugar código, disposição para aprender constantemente, atenção aos detalhes, compreensão de design básico, e boas práticas de organização de código.

1.4 Áreas de Especialização

Front-End: Trabalha com a interface visual (HTML, CSS, JavaScript). Foca no que o usuário vê e interage.

Back-End: Trabalha com a lógica de servidor (Python, PHP, Node.js). Foca no processamento de dados e regras de negócio.

Full-Stack: Domina tanto front-end quanto back-end, podendo trabalhar em todas as camadas de uma aplicação.

1.5 Ferramentas Essenciais

- **Editor de Código**: Visual Studio Code, Sublime Text
- **Navegadores**: Chrome, Firefox, Edge
- **Controle de Versão**: Git e GitHub
- **Terminal/Console**: Essencial para trabalhar com ferramentas
- **Navegador DevTools**: Para debug e inspeção de código

MÓDULO 2: INTRODUÇÃO A DESENVOLVIMENTO DE SOFTWARE, REDES E LINGUAGENS WEB

2.1 Ciclo de Desenvolvimento de Software

O processo segue estas etapas:

Planejamento: Define objetivos, escopo e requisitos do projeto.

Análise: Estuda as necessidades detalhadamente e especifica soluções.

Design: Cria a estrutura e arquitetura da aplicação.

Desenvolvimento: Escreve o código seguindo o design planejado.

Testes: Verifica se tudo funciona corretamente e identifica bugs.

Deployment: Publica a aplicação em um servidor acessível ao público.

Manutenção: Corrige bugs encontrados e implementa melhorias.

2.2 Conceitos Fundamentais de Redes de Computadores

Modelo Cliente-Servidor: O cliente (navegador) faz requisições, o servidor responde com dados. Você digita um URL no navegador (cliente), o servidor recebe a requisição e envia a página web de volta.

HTTP/HTTPS: Protocolos que definem como dados são transmitidos na web. HTTP é inseguro, HTTPS criptografa os dados (note o "S" de Seguro).

DNS: Sistema de Nomes de Domínio. Traduz endereços como "google.com" em números IP (192.168.0.1).

IP: Endereço único de um computador na rede. IPv4 usa formato como 192.168.0.1, IPv6 é mais moderno.

Porta: Número que especifica qual serviço usar no servidor. Porta 80 é HTTP, 443 é HTTPS, 3000 é comum para desenvolvimento.

URL: Endereço web completo. Exemplo: "https://www.exemplo.com:443/pagina?id=1#secao"

2.3 Principais Linguagens Web

HTML (HyperText Markup Language): Define a estrutura e conteúdo das páginas web. Ela cria elementos como títulos, parágrafos, links e imagens.

```
Exemplo básico:
<h1>Meu Título</h1>
<p>Meu conteúdo</p>
```

CSS (Cascading Style Sheets): Controla a aparência visual. Define cores, fontes, tamanhos, posicionamento.

```
Exemplo básico:
h1 {
  color: blue;
  font-size: 24px;
}
```

JavaScript: Adiciona interatividade e comportamento dinâmico. Permite animações, validações, requisições sem recarregar a página.

```
Exemplo básico:
function saudacao() {
  console.log("Olá!");
}
```

PHP: Linguagem de servidor que processa requisições e interage com banco de dados.

Python: Linguagem versátil usada no back-end (Django, Flask).

Node.js: Permite usar JavaScript no servidor.

2.4 Fluxo de uma Requisição Web

1. Usuário digita URL no navegador
2. Navegador envia requisição HTTP para o servidor
3. Servidor processa a requisição (acessa banco de dados se necessário)
4. Servidor envia resposta HTML de volta
5. Navegador recebe e renderiza o HTML
6. Clonar repositório, fazer alterações e fazer push no Git
7. Criar página com animações CSS
8. Estudar código de sites que você gosta

ESTRUTURA COMPARATIVA DAS LINGUAGENS

HTML

```html
<!-- DOCUMENTO -->
<!DOCTYPE html>
<html lang="pt-BR">

<!-- CABEÇALHO: Metadados -->
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Meu Site</title>
</head>

<!-- CORPO: Conteúdo visível -->
<body>
  <!-- SEÇÃO 1: Cabeçalho do site -->
  <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>

  <!-- SEÇÃO 2: Conteúdo principal -->
  <main>
    <article>
      <h1>Título Principal</h1>
      <p>Texto do artigo...</p>
    </article>
  </main>

  <!-- SEÇÃO 3: Rodapé -->
  <footer>
    <p>&copy; 2024 Meu Site</p>
  </footer>
</body>
</html>
```

CSS

```
CSS
```

```css
/* VARIÁVEIS - Reutilização de valores */
:root {
  --cor-primaria: #007bff;
  --cor-secundaria: #6c757d;
  --espacamento: 20px;
  --fonte-principal: 'Arial', sans-serif;
}

/* RESET - Normalizar estilos padrão */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

/* ESTILOS GERAIS */
body {
  font-family: var(--fonte-principal);
  font-size: 16px;
  line-height: 1.6;
  color: #333;
  background-color: #fff;
}

/* COMPONENTES - Reutilizáveis */
.botao {
  display: inline-block;
  padding: var(--espacamento);
  background-color: var(--cor-primaria);
  color: white;
  text-decoration: none;
  border-radius: 5px;
  transition: background-color 0.3s ease;
}

.botao:hover {
  background-color: var(--cor-secundaria);
}

/* LAYOUT */
.container {
  max-width: 1200px;
  margin: 0 auto;
  padding: var(--espacamento);
}
```

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

  .container {
    padding: 10px;
  }
}
```

JavaScript

```
javascript
```

```javascript
// 1. SELEÇÃO DE ELEMENTOS
const titulo = document.getElementById('titulo');
const botoes = document.querySelectorAll('.botao');
const formulario = document.querySelector('form');

// 2. MANIPULAÇÃO DOM
titulo.textContent = 'Novo Título';
titulo.classList.add('destaque');

// 3. EVENTOS
botoes.forEach(botao => {
  botao.addEventListener('click', function(evento) {
    console.log('Botão clicado!');
  });
});

formulario.addEventListener('submit', function(evento) {
  evento.preventDefault();
  console.log('Formulário enviado');
});

// 4. FUNÇÕES REUTILIZÁVEIS
function validarEmail(email) {
  const regex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
  return regex.test(email);
}

// 5. LÓGICA DE APLICAÇÃO
if (validarEmail('usuario@email.com')) {
  console.log('Email válido');
} else {
  console.log('Email inválido');
}

// 6. ARMAZENAMENTO TEMPORÁRIO
let usuario = {
  nome: 'João',
  email: 'joao@email.com',
  ativo: true
};

console.log(usuario.nome);

// 7. REQUISIÇÕES (será aprendido depois)
fetch('/api/dados')
  .then(resposta => resposta.json())
```

```
  .then(dados => console.log(dados))
  .catch(erro => console.error(erro));
```

GUIA DE ESTUDO POR TÓPICO

TÓPICO: Introdução ao Curso

O que estudar:

- História da web
- Mercado de trabalho
- Ferramentas essenciais
- Como começar a aprender

Para lembrar na prova:

- Diferencie front-end, back-end e full-stack
- Cite ferramentas principais (VSCode, Git, DevTools)
- Explique por que você quer ser dev web

TÓPICO: Redes e Software

O que estudar:

- Modelo cliente-servidor
- HTTP/HTTPS
- DNS e IP
- Ciclo de vida do software

Para lembrar na prova:

- Cliente faz requisição, servidor responde
- HTTPS é HTTP seguro (S = Segurança)
- DNS traduz domínio em IP
- Fases: Planejamento → Análise → Design → Dev → Testes → Deploy → Manutenção

Exemplo prático: Quando você acessa google.com:

1. Navegador envia requisição HTTP
2. DNS converte "google.com" em IP
3. Servidor Google recebe requisição
4. Servidor envia HTML, CSS, JavaScript
5. Navegador renderiza a página

TÓPICO: DOM

O que estudar:

- Estrutura hierárquica do DOM
- Como navegador constrói DOM
- Manipulação com JavaScript

Para lembrar na prova:

- DOM é árvore de nós
- Cada elemento HTML é um nó
- JavaScript pode selecionar e modificar nós

Exemplo de árvore DOM:

```
document
└── html
    ├── head
    │   └── title: "Meu Site"
    └── body
        ├── header
        ├── main
        │   └── h1: "Bem-vindo"
        └── footer
```

TÓPICO: Hospedagem e Publicação

O que estudar:

- Tipos de hospedagem
- FTP/SFTP
- Domínios

- DNS
- GitHub Pages

Para lembrar na prova:

- Compartilhada é mais barata, VPS e dedicada mais caros
- GitHub Pages é gratuito para sites estáticos
- Você precisa registrar domínio e apontar DNS

TÓPICO: Full-Stack, WordPress, Git

O que estudar:

- Stacks populares (MEAN, LAMP, MERN)
- WordPress basics
- Git e GitHub

Para lembrar na prova:

- Full-Stack domina front e back
- WordPress é CMS que não precisa código
- Git controla versões do código

Comandos Git essenciais:

```
git init           # Inicializar repo
git add .          # Adicionar arquivos
git commit -m ""   # Criar versão
git push           # Enviar para servidor
git pull           # Baixar atualizações
```

TÓPICO: HTML

O que estudar:

- Estrutura básica
- Tags semânticas
- Formulários
- Atributos

Para lembrar na prova:

- `<!DOCTYPE html>` é obrigatório
- Use tags semânticas: `<header>`, `<nav>`, `<main>`, `<footer>`
- `<head>` = metadados, `<body>` = conteúdo
- Formários com `<form>`, `<input>`, `<label>`

Checklist HTML:

- [ ] Tem doctype?
- [ ] Tem html, head, body?
- [ ] Meta charset UTF-8?
- [ ] Meta viewport (responsividade)?
- [ ] Title descritivo?
- [ ] Tags semânticas?
- [ ] Alt em imagens?
- [ ] Labels com inputs?

TÓPICO: CSS

O que estudar:

- Seletores
- Box model
- Layouts (Flexbox/Grid)
- Responsividade
- Animações

Para lembrar na prova:

- Cascata: especificidade determina qual regra ganha
- Box Model: margin → border → padding → content
- Flexbox para layouts 1D, Grid para 2D
- Media queries para responsividade

Ordem de CSS:

```
css
```

```css
/* 1. Reset */
* { margin: 0; padding: 0; }

/* 2. Estilos gerais */
body { font-family: Arial; }

/* 3. Componentes */
.botao { padding: 10px; }

/* 4. Layout */
.container { display: flex; }

/* 5. Responsividade */
@media (max-width: 768px) { }
```

TÓPICO: Formulários

O que estudar:

- Tipos de input
- Validação HTML5
- Validação JavaScript
- Envio de dados

Para lembrar na prova:

- `<form action="" method="POST">`
- `<input type="email">` com atributo `required`
- `<label for="">` associado a input
- Validação com `pattern`, `minlength`, `required`

Formulário completo:

```html
```

```html
<form action="enviar.php" method="POST">
  <label for="email">Email:</label>
  <input
    type="email"
    id="email"
    name="email"
    required
  >
  <button type="submit">Enviar</button>
</form>
```

TÓPICO: WordPress

O que estudar:

- Instalação
- Painel administrativo
- Criação de conteúdo
- Temas e plugins
- Customização básica

Para lembrar na prova:

- Artigos vs Páginas (posts vs páginas estáticas)
- Temas controlam aparência
- Plugins adicionam funcionalidades
- Pode adicionar CSS customizado

PERGUNTAS PARA REVISAR

Nível Iniciante

1. O que significa "Full-Stack"?
2. Qual a diferença entre HTTP e HTTPS?
3. O que é uma tag HTML?
4. Para que serve CSS?
5. O que é responsividade?

Nível Intermediário

6. Explique o Box Model do CSS
7. Qual a diferença entre `display: flex` e `display: grid`?
8. Como funciona uma requisição cliente-servidor?
9. O que é o DOM?
10. Como validar um formulário com HTML5?

Nível Avançado

11. Explique a cascata do CSS e especificidade
12. Qual a diferença entre `position: absolute` e `position: relative`?
13. Como você publicaria um site no GitHub Pages?
14. Qual a estrutura de pastas típica de um projeto?
15. Como melhorar performance de um site?

RESPOSTAS DAS PERGUNTAS

Respostas Iniciante

1. O que significa "Full-Stack"? Um desenvolvedor full-stack trabalha tanto no front-end (interface do usuário) quanto no back-end (lógica do servidor). Domina todas as camadas.

2. Qual a diferença entre HTTP e HTTPS? HTTP é inseguro, dados viajam em texto aberto. HTTPS criptografa os dados, é seguro. O "S" significa Seguro.

3. O que é uma tag HTML? Tag é uma instrução que marca elementos. Exemplo: `<p>texto</p>` marca um parágrafo.

4. Para que serve CSS? CSS estiliza e posiciona elementos HTML. Define cores, fontes, tamanhos, layouts.

5. O que é responsividade? Site adapta-se a diferentes tamanhos de tela (celular, tablet, desktop) usando media queries.

Respostas Intermediário

6. Explique o Box Model do CSS Cada elemento tem: content (conteúdo) →

padding (espaço interno) → border (borda) → margin (espaço externo).

7. Qual a diferença entre flex e grid? Flexbox é 1D (linha ou coluna). Grid é 2D (linhas e colunas simultaneamente).

8. Como funciona requisição cliente-servidor? Cliente (navegador) envia HTTP request para servidor. Servidor processa e envia HTTP response com dados.

9. O que é o DOM? DOM é representação em árvore dos elementos HTML. Permite JavaScript acessar e modificar a página.

10. Como validar formulário com HTML5? Use atributos: `required`, `type="email"`, `minlength`, `pattern`, `max`, `min`.

Respostas Avançado

11. Explique cascata do CSS e especificidade Regras CSS em cascata: inline > ID > classe > elemento. Mais específica vence menos específica.

12. Diferença entre position absolute e relative? Relative posiciona relativo à sua posição normal. Absolute posiciona relativo ao primeiro ancestor posicionado.

13. Como publicar no GitHub Pages? Crie repo "usuario.github.io", faça push de arquivos, acesse em usuario.github.io.

14. Estrutura de pastas típica?

```
projeto/
├── index.html
├── css/
│   └── style.css
├── js/
│   └── script.js
├── img/
│   └── logo.png
└── README.md
```

15. Como melhorar performance? Comprimir imagens, minificar CSS/JS, lazy loading, usar CDN, cache, reduzir requisições.

GLOSSÁRIO TÉCNICO

API: Conjunto de regras que permite comunicação entre programas

Backend: Parte lógica do servidor (dados, regras de negócio)

Cache: Armazenamento temporário para acesso rápido

CDN: Rede de servidores distribuídos globalmente

CLI: Interface de linha de comando (Terminal)

CMS: Sistema de gerenciamento de conteúdo (WordPress)

CSS: Linguagem de estilos para HTML

DOM: Representação em árvore de um documento HTML

Framework: Estrutura reutilizável para desenvolvimento

Frontend: Interface com usuário (HTML, CSS, JavaScript)

Full-Stack: Desenvolvedor que domina front e back

Git: Sistema de controle de versão

GitHub: Plataforma para repositórios Git

Hospedagem: Serviço que armazena seu site em servidor

HTML: Linguagem de marcação para estrutura web

HTTP: Protocolo de transferência de hipertexto

HTTPS: HTTP seguro com criptografia

IDE: Ambiente de desenvolvimento integrado

JavaScript: Linguagem de programação para web

Localhost: Seu computador como servidor local

Plugin: Extensão que adiciona funcionalidade

Repositório: Pasta que armazena projeto com versões

Responsivo: Site que adapta a diferentes telas

Servidor: Computador que fornece serviços/dados

Stack: Conjunto de tecnologias usadas

Tag: Elemento HTML entre `< >`

Terminal: Interface de texto para comandos

Theme: Tema que controla aparência (WordPress)

URL: Endereço web completo

Validação: Verificar se dados estão corretos

Widget: Pequeno componente funcional

DICAS FINAIS PARA PROVA

Antes da prova:

- Durma bem
- Revise resumos
- Faça exercícios práticos
- Organize materiais

Durante a prova:

- Leia perguntas com cuidado
- Responda questões fáceis primeiro
- Mostre seu raciocínio
- Revise antes de entregar

Se for prova prática:

- Comece estruturando HTML
- Depois estilize com CSS
- Por fim, adicione JavaScript
- Teste em navegador
- Use DevTools para debug

Se for prova teórica:

- Defina conceitos claramente
- Dê exemplos práticos
- Explique "por quê" não apenas "como"

- Cite fontes/referências

MATERIAIS DE REFERÊNCIA PARA ESTUDAR

Documentações Oficiais:

- MDN Web Docs: mdn.org
- HTML Spec: html.spec.whatwg.org
- W3C: w3.org

Plataformas de Aprendizado:

- Codecademy: codecademy.com
- freeCodeCamp: freecodecamp.org
- Udemy: udemy.com
- Coursera: coursera.org

Sites Práticos:

- CodePen: codepen.io
- JSFiddle: jsfiddle.net
- Replit: replit.com

Comunidades:

- Stack Overflow
- GitHub
- Reddit r/webdev
- Dev.to

CRONOGRAMA DE ESTUDO RECOMENDADO

Semana 1: Leia módulos 1-2, faça anotações

Semana 2: Estude módulo 3-4, pratique DOM

Semana 3: Aprenda HTML (módulo 5), crie página simples

Semana 4: Domina CSS (módulo 6), estilize sua página

Semana 5: Estude Formulários (módulo 7), valide dados

Semana 6: Aprenda WordPress (módulo 8), crie site

Semana 7: Revise tudo, faça exercícios práticos

Semana 8: Simule prova, revise dúvidas

BOA SORTE NA PROVA! 🎓

Você tem todo o conhecimento necessário. Estude com foco, pratique bastante e não tenha medo de errar durante o aprendizado. Erros são oportunidades de aprender!

Lembre-se: programação é aprendizado contínuo. Mesmo após a prova, continue estudando e construindo projetos! CSS é aplicado para estilização 7. JavaScript é executado para interatividade

---

MÓDULO 3: DOM, HOSPEDAGEM E PUBLICAÇÃO DE SITES WEB

3.1 O Que é DOM (Document Object Model)?

DOM é a representação em árvore de uma página HTML. Cada elemento HTML se torna um "nó" no DOM, permitindo que JavaScript acesse e modifique elementos.

Estrutura hierárquica:

```
Document
  ├── html
      ├── head
      │   ├── title
      │   └── meta
      └── body
          ├── header
          ├── section
          └── footer
```

3.2 Como o Navegador Constrói o DOM

1. **Parsing de HTML:** O navegador lê o HTML linha por linha
2. **Tokenização:** Quebra o HTML em tokens (tags, atributos, conteúdo)
3. **Construção do DOM:** Cria a árvore de nós
4. **Parsing de CSS:** Processa estilos e cria a CSSOM (CSS Object Model)

5. **Render Tree:** Combina DOM e CSSOM

6. **Layout:** Calcula posição e tamanho de cada elemento

7. **Paint:** Desenha os elementos na tela

3.3 Manipulação do DOM com JavaScript

```javascript
// Selecionar elementos
const titulo = document.getElementById('meu-titulo');
const paragrafos = document.querySelectorAll('p');

// Modificar conteúdo
titulo.textContent = 'Novo Título';
titulo.innerHTML = '<strong>Novo Título</strong>';

// Modificar atributos
titulo.setAttribute('class', 'destaque');
titulo.removeAttribute('disabled');

// Modificar estilos
titulo.style.color = 'red';
titulo.style.fontSize = '20px';

// Adicionar/remover classes
titulo.classList.add('ativo');
titulo.classList.remove('inativo');
titulo.classList.toggle('visivel');

// Adicionar elementos
const novo = document.createElement('p');
novo.textContent = 'Novo parágrafo';
document.body.appendChild(novo);

// Remover elementos
titulo.remove();
```

3.4 Tipos de Hospedagem Web

Hospedagem Compartilhada: Seu site compartilha servidor com outros. Opção econômica para iniciantes. Exemplo: Hostinger, GoDaddy.

VPS (Virtual Private Server): Você tem uma máquina virtual exclusiva com mais controle. Intermediário em preço e controle.

Servidor Dedicado: Servidor físico inteiro é seu. Caro mas oferece máximo controle e performance.

Cloud Hosting: Infraestrutura flexível que cresce com sua demanda. AWS, Google Cloud, Microsoft Azure.

GitHub Pages: Hospedagem gratuita para sites estáticos. Integrado com Git.

3.5 Processo de Publicação de Sites Web

Passo 1 - Preparação Local: Desenvolva e teste seu site no seu computador.

Passo 2 - Registrar Domínio: Escolha um nome e registre em um serviço como Registro.br.

Passo 3 - Contratar Hospedagem: Escolha um plano conforme suas necessidades.

Passo 4 - Configurar DNS: Aponte seu domínio para o servidor da hospedagem.

Passo 5 - Fazer Upload: Envie seus arquivos via FTP ou painel de controle.

Passo 6 - Testar Online: Acesse seu site e verifique se tudo funciona.

Passo 7 - Monitoramento: Mantenha seu site atualizado e seguro.

3.6 FTP (File Transfer Protocol)

Protocolo para transferir arquivos do seu computador para o servidor. Você precisa de:

- Endereço do servidor FTP
- Usuário e senha
- Software FTP (FileZilla, WinSCP)

MÓDULO 4: DESENVOLVIMENTO FULL-STACK, WORDPRESS E GITHUB PAGES

4.1 Desenvolvimento Full-Stack

Full-Stack significa dominar tanto front-end quanto back-end. Um desenvolvedor full-stack pode trabalhar em todas as partes de uma aplicação web.

Stack Comum - MEAN:

- **M**ongoDB: Banco de dados
- **E**xpress: Framework back-end
- **A**ngular: Framework front-end
- **N**ode.js: Ambiente JavaScript no servidor

Stack Comum - LAMP:

- **L**inux: Sistema operacional
- **A**pache: Servidor web
- **M**ySQL: Banco de dados
- **P**HP: Linguagem back-end

Stack Comum - MERN:

- **M**ongoDB: Banco de dados
- **E**xpress: Framework back-end
- **R**eact: Biblioteca front-end
- **N**ode.js: Ambiente JavaScript

4.2 WordPress - O Que É?

WordPress é um sistema de gerenciamento de conteúdo (CMS) que permite criar sites sem programar muito. É usado por mais de 40% dos sites da web.

Vantagens: Fácil de usar, tema e plugins disponíveis, SEO amigável, comunidade grande.

Desvantagens: Menos controle que código custom, pode ser lento com muitos plugins, segurança depende de manutenção.

4.3 Instalando WordPress Localmente

1. Baixe WordPress de wordpress.org
2. Instale um servidor local (XAMPP, Laragon, Local by Flywheel)
3. Crie um banco de dados MySQL
4. Coloque WordPress na pasta do servidor
5. Acesse em localhost/wordpress
6. Siga o assistente de instalação

4.4 Estrutura de Arquivos WordPress

```
wordpress/
├── wp-admin/           (Painel administrativo)
├── wp-content/         (Temas e plugins)
│   ├── themes/        (Seus temas)
│   ├── plugins/       (Seus plugins)
│   └── uploads/       (Imagens e arquivos)
├── wp-includes/        (Arquivos núcleo)
├── wp-config.php       (Configurações do banco)
└── index.php           (Ponto de entrada)
```

4.5 GitHub Pages - Hospedagem Gratuita

GitHub Pages permite hospedar sites estáticos gratuitamente diretamente de um repositório GitHub.

Tipos:

- **Repositório de usuário:** seu-usuario.github.io (site pessoal)
- **Repositório de projeto:** seu-usuario.github.io/nome-projeto

4.6 Publicando no GitHub Pages

Passo 1: Crie um repositório chamado "seu-usuario.github.io"

Passo 2: Clone o repositório:

```
bash

git clone https://github.com/seu-usuario/seu-usuario.github.io
cd seu-usuario.github.io
```

Passo 3: Adicione seus arquivos HTML, CSS, JavaScript:

```
index.html
style.css
script.js
```

Passo 4: Faça commit e push:

```
bash
```

```
git add .
git commit -m "Primeiro site"
git push origin main
```

Passo 5: Acesse seu-usuario.github.io em alguns minutos!

4.7 O que é Git?

Git é um sistema de controle de versão que permite rastrear mudanças no código, colaborar com outros e voltar a versões anteriores.

Conceitos-chave:

Repositório: Pasta que armazena seu projeto com histórico de versões.

Commit: "Fotografia" do seu código em um momento específico. Sempre inclua mensagem descritiva.

Branch: Cópia do projeto onde você trabalha sem afetar o principal.

Merge: Combina mudanças de um branch para outro.

4.8 Comandos Git Essenciais

bash

```
# Inicializar repositório
git init

# Clonar repositório remoto
git clone https://github.com/usuario/repo.git

# Ver status de mudanças
git status

# Adicionar arquivos para commit
git add arquivo.txt
git add .               (todos os arquivos)

# Criar um commit
git commit -m "Descrição das mudanças"

# Enviar para repositório remoto
git push origin main

# Baixar atualizações do remoto
git pull origin main

# Ver histórico de commits
git log

# Criar novo branch
git branch novo-branch

# Trocar de branch
git checkout novo-branch

# Combinar branches
git merge novo-branch
```

MÓDULO 5: HTML - LINGUAGEM DE MARCAÇÃO

5.1 O que é HTML?

HTML significa "HyperText Markup Language" (Linguagem de Marcação de Hipertexto). Usa tags/elementos para estruturar conteúdo. Tags são instruções que dizem ao navegador como exibir o conteúdo.

5.2 Estrutura Básica de um Documento HTML

```html
<!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="style.css">
  </head>
  <body>
    <h1>Olá Mundo!</h1>
    <p>Este é meu primeiro site.</p>
    <script src="script.js"></script>
  </body>
</html>
```

Explicação:

`<!DOCTYPE html>`: Declara que é HTML5 (versão atual).

`<html lang="pt-BR">`: Tag raiz, idioma português Brasil.

`<head>`: Contém metadados, título, links de CSS.

`<meta charset="UTF-8">`: Define codificação de caracteres.

`<meta name="viewport">`: Torna o site responsivo em celulares.

`<title>`: Título que aparece na aba do navegador.

`<body>`: Contém todo o conteúdo visível.

5.3 Tags HTML Essenciais

Títulos e Textos:

```html
```

```
<h1>Título Principal</h1>
<h2>Subtítulo 2</h2>
<h3>Subtítulo 3</h3>
<h4>Subtítulo 4</h4>
<h5>Subtítulo 5</h5>
<h6>Subtítulo 6</h6>

<p>Parágrafo de texto normal.</p>

<strong>Texto em negrito importante</strong>
<b>Texto em negrito sem semântica</b>

<em>Texto em itálico com ênfase</em>
<i>Texto em itálico sem semântica</i>

<small>Texto pequeno</small>
<mark>Texto marcado/destacado</mark>
<del>Texto deletado</del>
<ins>Texto inserido</ins>
<sub>Texto subscrito</sub>
<sup>Texto sobrescrito</sup>
```

Listas:

```
html
```

```html
<!-- Lista não ordenada (com pontos) -->
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>

<!-- Lista ordenada (com números) -->
<ol>
  <li>Primeiro</li>
  <li>Segundo</li>
  <li>Terceiro</li>
</ol>

<!-- Lista de definições -->
<dl>
  <dt>HTML</dt>
  <dd>Linguagem de marcação</dd>
  <dt>CSS</dt>
  <dd>Linguagem de estilos</dd>
</dl>
```

Links e Imagens:

```
html
```

```html
<!-- Link simples -->
<a href="https://www.google.com">Clique aqui</a>

<!-- Link para outra página local -->
<a href="sobre.html">Sobre</a>

<!-- Link com atributo target -->
<a href="https://www.google.com" target="_blank">Abre em nova aba</a>

<!-- Imagem -->
<img src="imagem.jpg" alt="Descrição da imagem" width="300" height="200">

<!-- Imagem responsiva -->
<img src="imagem.jpg" alt="Descrição" style="max-width: 100%; height: auto;">

<!-- Figura com legenda -->
<figure>
  <img src="imagem.jpg" alt="Descrição">
  <figcaption>Legenda da imagem</figcaption>
</figure>
```

5.4 Elementos Semânticos

Elementos semânticos descrevem o significado do conteúdo, melhorando SEO e acessibilidade.

```
html
```

```html
<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>

<main>
  <article>
    <h1>Título do Artigo</h1>
    <p>Conteúdo do artigo aqui.</p>
  </article>

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

<footer>
  <p>&copy; 2024 Meu Site. Todos os direitos reservados.</p>
</footer>
```

Significados:

`<header>`: Cabeçalho do site ou seção.

`<nav>`: Área de navegação com links.

`<main>`: Conteúdo principal da página.

`<article>`: Conteúdo independente como artigo ou post.

`<section>`: Seção temática de conteúdo.

`<aside>`: Conteúdo complementar/barra lateral.

`<footer>`: Rodapé do site ou seção.

5.5 Tabelas em HTML

```html
```

```html
<table border="1">
  <thead>
    <tr>
      <th>Nome</th>
      <th>Idade</th>
      <th>Cidade</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>João</td>
      <td>25</td>
      <td>São Paulo</td>
    </tr>
    <tr>
      <td>Maria</td>
      <td>30</td>
      <td>Rio de Janeiro</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td colspan="3">Total: 2 pessoas</td>
    </tr>
  </tfoot>
</table>
```

Elementos:

`<table>`: Define a tabela.

`<thead>`: Cabeçalho da tabela.

`<tbody>`: Corpo da tabela.

`<tfoot>`: Rodapé da tabela.

`<tr>`: Linha (table row).

`<th>`: Célula de cabeçalho (table header).

`<td>`: Célula de dados (table data).

5.6 Atributos Globais Importantes

```
html
```

```html
<!-- ID - identificador único -->
<p id="paragrafo-1">Este parágrafo tem um ID</p>

<!-- Class - para aplicar estilos CSS -->
<p class="destaque">Este parágrafo está destacado</p>

<!-- Style - estilo inline -->
<p style="color: red; font-size: 18px;">Parágrafo vermelho</p>

<!-- Title - tooltip ao passar mouse -->
<p title="Descrição ao passar mouse">Passe o mouse aqui</p>

<!-- Data attributes - armazenar dados customizados -->
<p data-usuario-id="123" data-tipo="premium">Dado customizado</p>

<!-- Disabled - desabilita elemento -->
<button disabled>Botão desabilitado</button>

<!-- Hidden - esconde elemento -->
<p hidden>Este parágrafo está escondido</p>
```

5.7 Boas Práticas em HTML

1. **Use tags semânticas** em vez de divs genéricos
2. **Sempre feche tags** properly
3. **Use atributo alt em imagens** para acessibilidade
4. **Mantenha indentação consistente** para legibilidade
5. **Use nomes descritivos** para IDs e classes
6. **Minimize divs aninhadas** - mantenha estrutura simples
7. **Separe conteúdo (HTML), estilo (CSS) e comportamento (JavaScript)**

MÓDULO 6: CSS - ESTILIZAÇÃO E LAYOUT

6.1 O que é CSS?

CSS significa "Cascading Style Sheets" (Folhas de Estilo em Cascata). Controla a aparência visual dos elementos HTML, definindo cores, fontes, espaçamento, posicionamento.

6.2 Formas de Inserir CSS

Estilo Inline (Menos recomendado):

```html
<p style="color: blue; font-size: 18px;">Texto azul</p>
```

Tag Style Internal (Para páginas simples):

```html
<!DOCTYPE html>
<html>
<head>
  <style>
    p {
      color: blue;
      font-size: 18px;
    }
  </style>
</head>
<body>
  <p>Texto azul</p>
</body>
</html>
```

Arquivo CSS Externo (Melhor prática):

```html
<!-- No arquivo HTML -->
<head>
  <link rel="stylesheet" href="style.css">
</head>
```

```css
/* No arquivo style.css */
p {
  color: blue;
  font-size: 18px;
}
```

6.3 Seletores CSS

Seletor de Elemento:

```css
```

```
p {
  color: red;
}
```

Seletor de ID (único):

```
css

#header {
  background-color: navy;
}
```

HTML correspondente: `<div id="header"></div>`

Seletor de Class (múltiplos elementos):

```
css

.destaque {
  background-color: yellow;
  font-weight: bold;
}
```

HTML correspondente: `<p class="destaque">Destaque</p>`

Seletor de Atributo:

```
css

input[type="email"] {
  border: 1px solid blue;
}

a[target="_blank"] {
  color: purple;
}
```

Seletores Combinadores:

```
css
```

```css
/* Descendente - qualquer p dentro de div */
div p {
  color: red;
}

/* Filho - apenas p filhos diretos de div */
div > p {
  color: blue;
}

/* Irmão adjacente - p imediatamente após h1 */
h1 + p {
  margin-top: 0;
}

/* Irmão geral - qualquer p que vem após h1 */
h1 ~ p {
  color: gray;
}
```

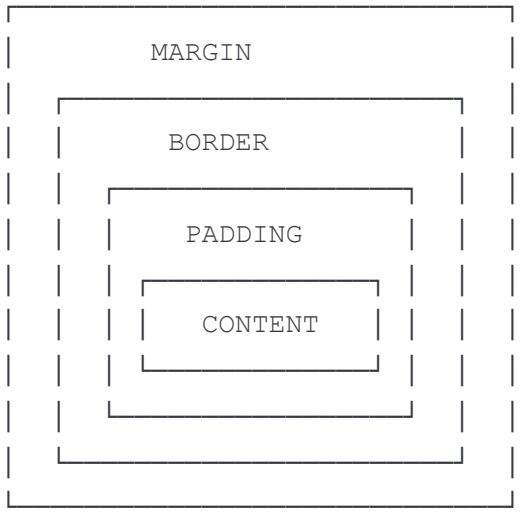
Pseudo-classes:

```
css
```

```css
/* Quando mouse está sobre elemento */
a:hover {
  color: red;
  text-decoration: underline;
}

/* Primeiro filho */
li:first-child {
  font-weight: bold;
}

/* Último filho */
li:last-child {
  border-bottom: none;
}

/* Enésimo filho */
tr:nth-child(2n) {
  background-color: #f0f0f0;
}

/* Link já visitado */
a:visited {
  color: purple;
}

/* Quando elemento recebe foco */
input:focus {
  border-color: blue;
  outline: none;
}

/* Quando elemento está desabilitado */
button:disabled {
  opacity: 0.5;
  cursor: not-allowed;
}
```

6.4 Box Model

Todo elemento em CSS é um box com:



Exemplo prático:

```css
div {
  width: 200px;
  height: 100px;

  /* Espaço externo (fora da borda) */
  margin: 20px;

  /* Espaço interno (dentro da borda) */
  padding: 15px;

  /* Borda */
  border: 2px solid black;

  /* Conteúdo */
  background-color: lightblue;
}
```

Valores de Margin e Padding:

```css
```

```css
/* Todos os lados */
margin: 20px;

/* Vertical (topo/baixo) e horizontal (esquerda/direita) */
margin: 10px 20px;

/* Topo, horizontal, e fundo */
margin: 10px 20px 15px;

/* Topo, direita, fundo, esquerda (no sentido do relógio) */
margin: 10px 20px 15px 5px;

/* Lados individuais */
margin-top: 10px;
margin-right: 20px;
margin-bottom: 15px;
margin-left: 5px;
```

6.5 Propriedades de Texto

```
css
```

```css
p {
  /* Cor do texto */
  color: #333333;

  /* Fonte */
  font-family: 'Arial', 'Helvetica', sans-serif;

  /* Tamanho da fonte */
  font-size: 16px;

  /* Peso/Espessura */
  font-weight: bold;        /* bold, normal, 100-900 */

  /* Estilo */
  font-style: italic;       /* italic, normal, oblique */

  /* Altura da linha */
  line-height: 1.6;

  /* Espaçamento entre letras */
  letter-spacing: 0.05em;

  /* Alinhamento */
  text-align: center;       /* left, right, center, justify */

  /* Decoração */
  text-decoration: underline;  /* underline, overline, line-through, none */

  /* Transformação */
  text-transform: uppercase;  /* uppercase, lowercase, capitalize */

  /* Sombra do texto */
  text-shadow: 2px 2px 4px rgba(0,0,0,0.3);
}
```

6.6 Cores em CSS

Nomes de cores:

```css
color: red;
color: blue;
color: lightblue;
```

Hexadecimal (mais comum):

```css
color: #FF0000;     /* Vermelho */
color: #00FF00;     /* Verde */
color: #0000FF;     /* Azul */
color: #FFFFFF;     /* Branco */
color: #000000;     /* Preto */
```

RGB:

```css
color: rgb(255, 0, 0);        /* Vermelho */
color: rgba(255, 0, 0, 0.5); /* Vermelho semi-transparente */
```

HSL:

```css
color: hsl(0, 100%, 50%);     /* Vermelho */
color: hsla(0, 100%, 50%, 0.5); /* Vermelho semi-transparente */
```

6.7 Background

```css
```

```css
div {
  /* Cor de fundo */
  background-color: lightblue;

  /* Imagem de fundo */
  background-image: url('fundo.jpg');

  /* Repetição */
  background-repeat: no-repeat;  /* repeat, repeat-x, repeat-y, no-repeat */

  /* Posição */
  background-position: center;   /* left, right, center, top, bottom */
  background-position: 50% 50%;

  /* Tamanho */
  background-size: cover;        /* contain, cover, 100% 100%, 500px 300px */

  /* Fixação (para efeito parallax) */
  background-attachment: fixed;
}

/* Shorthand */
background: lightblue url('fundo.jpg') no-repeat center/cover fixed;
```

6.8 Borders e Shadows

```
css
```

```css
div {
  /* Border simples */
  border: 2px solid red;

  /* Borders individuais */
  border-top: 2px solid red;
  border-right: 2px solid green;
  border-bottom: 2px solid blue;
  border-left: 2px solid yellow;

  /* Arredondamento */
  border-radius: 10px;          /* Todos os cantos */
  border-radius: 10px 20px 30px 40px;  /* Topo-esq, topo-dir, bai-dir, bai-esq */

  /* Sombra */
  box-shadow: 5px 5px 10px rgba(0,0,0,0.3);

  /* Sombra múltipla */
  box-shadow: 2px 2px 5px red, -2px -2px 5px blue;

  /* Estilo da borda */
  border-style: solid;      /* solid, dashed, dotted, double */

  /* Borda individual com estilos diferentes */
  border-top: 2px solid red;
  border-right: 3px dashed green;
  border-bottom: 4px dotted blue;
  border-left: 1px double black;
}
```

6.9 Display e Posicionamento

Display - Como elemento é exibido:

```
css
```

```css
/* Bloco - ocupa linha inteira, respeita margin/width/height */
display: block;
/* Exemplos: div, p, h1, section */

/* Inline - fica na mesma linha, ignora width/height */
display: inline;
/* Exemplos: span, a, strong */

/* Inline-block - fica na linha mas respeita width/height */
display: inline-block;

/* Nenhum - elemento não é exibido */
display: none;

/* Flex - layout flexível (muito usado) */
display: flex;

/* Grid - layout em grid (muito usado) */
display: grid;
```

Position - Como posicionar elemento:

```css
/* Estático - posição normal do fluxo (padrão) */
position: static;

/* Relativo - relativo à sua posição normal */
position: relative;
top: 10px;      /* Move 10px para baixo */
left: 20px;     /* Move 20px para direita */

/* Absoluto - relativo ao primeiro ancestor posicionado */
position: absolute;
top: 50px;
left: 100px;

/* Fixo - relativo à viewport (fica fixo ao scroll) */
position: fixed;
top: 0;
right: 0;

/* Sticky - muda entre relativo e fixo ao scroll */
position: sticky;
top: 0;
```

6.10 Flexbox - Layout Flexível

Flexbox é um sistema de layout que facilita alinhamento e distribuição de elementos.

```
css
```

```css
/* Contenedor flex */
.container {
  display: flex;

  /* Direção dos itens (padrão: row) */
  flex-direction: row;        /* row, column, row-reverse, column-reverse */

  /* Quebra de linha */
  flex-wrap: wrap;            /* wrap, nowrap, wrap-reverse */

  /* Alinhamento horizontal (eixo principal) */
  justify-content: center;
  /* flex-start, flex-end, center, space-between, space-around, space-evenly */

  /* Alinhamento vertical (eixo transversal) */
  align-items: center;
  /* flex-start, flex-end, center, stretch, baseline */

  /* Alinhamento de múltiplas linhas */
  align-content: center;

  /* Espaço entre itens */
  gap: 20px;
}

/* Itens flex */
.item {
  /* Quantas partes o item ocupa */
  flex: 1;

  /* Crescimento do item */
  flex-grow: 1;

  /* Redução do item */
  flex-shrink: 1;

  /* Tamanho base */
  flex-basis: 200px;

  /* Alinhamento individual */
  align-self: center;
}
```

Exemplo prático - Menu horizontal com Flexbox:

```
html
```

```
<style>
  nav {
    display: flex;
    gap: 20px;
    background-color: navy;
    padding: 15px;
  }

  nav a {
    color: white;
    text-decoration: none;
    flex: 1;
    text-align: center;
  }

  nav a:hover {
    background-color: #333;
    border-radius: 5px;
  }
</style>

<nav>
  <a href="#home">Home</a>
  <a href="#sobre">Sobre</a>
  <a href="#contato">Contato</a>
</nav>
```

6.11 CSS Grid - Layout em Grade

Grid é um sistema de layout bidimensional muito poderoso.

```
css
```

```css
.container {
  display: grid;

  /* Define colunas */
  grid-template-columns: 1fr 1fr 1fr;  /* 3 colunas iguais */
  grid-template-columns: 200px 1fr;    /* 1ª coluna 200px, 2ª preenche resto */
  grid-template-columns: repeat(4, 1fr); /* 4 colunas iguais */

  /* Define linhas */
  grid-template-rows: 100px auto 50px;

  /* Espaçamento */
  gap: 20px;
  column-gap: 15px;
  row-gap: 10px;

  /* Alinhamento */
  justify-items: center;   /* Horizontal */
  align-items: center;     /* Vertical */
}

.item {
  /* Posição na grade */
  grid-column: 1 / 3;  /* Da coluna 1 até 3 */
  grid-row: 1 / 2;     /* Da linha 1 até 2 */
}
```

Exemplo prático - Layout com Grid:

```html
```

```
<style>
  .grid-layout {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 20px;
    padding: 20px;
  }

  .item {
    background-color: lightblue;
    padding: 20px;
    border-radius: 5px;
    text-align: center;
  }
</style>

<div class="grid-layout">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
  <div class="item">Item 4</div>
  <div class="item">Item 5</div>
  <div class="item">Item 6</div>
</div>
```

6.12 Responsividade - Media Queries

Media queries permitem aplicar CSS diferentes em diferentes tamanhos de tela.

```
css
```

```css
/* Estilos padrão (mobile first) */
body {
  font-size: 14px;
  width: 100%;
}

.container {
  display: grid;
  grid-template-columns: 1fr;  /* 1 coluna em celular */
}

/* Tablets e acima (768px) */
@media (min-width: 768px) {
  body {
    font-size: 16px;
  }

  .container {
    grid-template-columns: repeat(2, 1fr);  /* 2 colunas */
  }
}

/* Desktop e acima (1024px) */
@media (min-width: 1024px) {
  .container {
    grid-template-columns: repeat(3, 1fr);  /* 3 colunas */
  }
}

/* Breakpoints comuns */
@media (max-width: 480px) { }   /* Celular pequeno */
@media (min-width: 481px) { }   /* Celular grande */
@media (min-width: 768px) { }   /* Tablet */
@media (min-width: 1024px) { }  /* Desktop pequeno */
@media (min-width: 1440px) { }  /* Desktop grande */
```

6.13 Transições e Animações

Transições - Mudanças suaves entre estados:

```
css
```

```css
button {
  background-color: blue;
  color: white;
  transition: background-color 0.3s ease-in-out;
}

button:hover {
  background-color: darkblue;
}

/* Propriedades individuais */
transition-property: background-color;
transition-duration: 0.3s;
transition-timing-function: ease-in-out;
transition-delay: 0.1s;
```

Animações - Sequências mais complexas:

```
css

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

.caixa {
  animation: deslizar 2s ease-in-out infinite;
}

/* Propriedades */
animation-name: deslizar;
animation-duration: 2s;
animation-timing-function: ease-in-out;
animation-delay: 0.5s;
animation-iteration-count: infinite;  /* or number */
animation-direction: alternate;        /* normal, reverse, alternate */
```

6.14 Transforms - Transformações Visuais

```css
div {
  /* Mover */
  transform: translate(50px, 30px);
  transform: translateX(50px);
  transform: translateY(30px);

  /* Escalar */
  transform: scale(1.5);         /* 1.5x do tamanho */
  transform: scaleX(2);          /* 2x largura */
  transform: scaleY(0.5);        /* 0.5x altura */

  /* Rotacionar */
  transform: rotate(45deg);      /* 45 graus */
  transform: rotateX(45deg);     /* Rotação 3D em X */
  transform: rotateY(45deg);     /* Rotação 3D em Y */
  transform: rotateZ(45deg);     /* Rotação 3D em Z */

  /* Inclinar */
  transform: skew(10deg, 20deg);
  transform: skewX(10deg);
  transform: skewY(20deg);

  /* Combinações */
  transform: translate(50px, 30px) rotate(45deg) scale(1.2);
}
```

6.15 Opacity e Visibilidade

```css
div {
  /* Opacidade (0 a 1) */
  opacity: 0.5;          /* 50% transparente */

  /* Visibilidade */
  visibility: hidden;   /* Hidden mas ocupa espaço */
  visibility: visible;

  /* Display */
  display: none;         /* Não aparece e não ocupa espaço */
}
```

6.16 Boas Práticas em CSS

1. **Use classes em vez de IDs** para estilos reutilizáveis
2. **Mobile first** - desenvolva para celular, depois expanda
3. **Evite !important** - indica problema na cascata
4. **Use variáveis CSS:**

```css
:root {
  --cor-primaria: #007bff;
  --cor-secundaria: #6c757d;
  --espacamento: 20px;
}

body {
  color: var(--cor-primaria);
  padding: var(--espacamento);
}
```

5. **Organize o CSS** em seções lógicas
6. **Reutilize classes** em vez de duplicar estilos
7. **Prefira Flexbox/Grid** a positioning absoluto

MÓDULO 7: FORMULÁRIOS WEB

7.1 O que é um Formulário Web?

Um formulário web permite que usuários enviem dados para o servidor. É a principal forma de interação entre usuário e aplicação.

7.2 Estrutura Básica de Formulário

```html
```

```html
<form action="processar.php" method="POST">
  <!-- Campos do formulário -->
  <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">

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

Atributos:

`action`: URL para onde os dados serão enviados

`method`: POST (envia dados no corpo) ou GET (envia na URL)

7.3 Elementos de Input

Texto:

```html
<input type="text" name="nome" placeholder="Digite seu nome">
<input type="email" name="email" placeholder="seu@email.com">
<input type="password" name="senha" placeholder="Sua senha">
<input type="number" name="idade" min="0" max="120">
<input type="tel" name="telefone" placeholder="(XX) XXXXX-XXXX">
<input type="url" name="website" placeholder="https://exemplo.com">
<input type="search" name="busca" placeholder="Pesquisar...">
```

Seleção:

```html
<!-- Checkbox - múltiplas seleções -->
<input type="checkbox" name="interesses" value="esportes"> Esportes
<input type="checkbox" name="interesses" value="musica"> Música
<input type="checkbox" name="interesses" value="leitura"> Leitura

<!-- Radio button - apenas uma seleção -->
<input type="radio" name="genero" value="masculino"> Masculino
<input type="radio" name="genero" value="feminino"> Feminino
<input type="radio" name="genero" value="outro"> Outro
```

Data e Hora:

```html
<input type="date" name="data_nascimento">
<input type="time" name="hora">
<input type="datetime-local" name="evento">
<input type="month" name="mes">
<input type="week" name="semana">
<input type="color" name="cor_favorita">
```

Outros:

```html
<input type="range" name="volume" min="0" max="100">
<input type="file" name="documento" accept=".pdf,.doc">
<input type="hidden" name="usuario_id" value="123">
```

7.4 Select (Menu Suspenso)

```html
```

```html
<label for="pais">País:</label>
<select id="pais" name="pais">
  <option value="">Selecione um país</option>
  <option value="br">Brasil</option>
  <option value="pt">Portugal</option>
  <option value="us">Estados Unidos</option>
</select>

<!-- Select com grupos -->
<select name="estado">
  <optgroup label="Sudeste">
    <option value="sp">São Paulo</option>
    <option value="rj">Rio de Janeiro</option>
  </optgroup>
  <optgroup label="Nordeste">
    <option value="ba">Bahia</option>
    <option value="pe">Pernambuco</option>
  </optgroup>
</select>

<!-- Select múltiplo -->
<select name="interesses" multiple size="4">
  <option value="esportes">Esportes</option>
  <option value="musica">Música</option>
  <option value="leitura">Leitura</option>
  <option value="viagem">Viagem</option>
</select>
```

7.5 Textarea (Texto Multi-linha)

```html
<label for="mensagem">Mensagem:</label>
<textarea
  id="mensagem"
  name="mensagem"
  rows="5"
  cols="50"
  placeholder="Digite sua mensagem aqui..."
></textarea>
```

7.6 Labels e Associação

```html
```

```html
<!-- Label associado ao input -->
<label for="email">Email:</label>
<input type="email" id="email" name="email">

<!-- Label englobando o input -->
<label>
  <input type="checkbox" name="concordo">
  Concordo com os termos
</label>
```

7.7 Validação HTML5

```html
<!-- Campo obrigatório -->
<input type="text" name="nome" required>

<!-- Padrão específico -->
<input type="email" name="email" pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$">

<!-- Comprimento -->
<input type="password" name="senha" minlength="8" maxlength="16">

<!-- Valores numéricos -->
<input type="number" name="idade" min="18" max="100" step="1">

<!-- Campo desabilitado -->
<input type="text" name="teste" disabled>

<!-- Campo apenas leitura -->
<input type="text" name="id" value="123" readonly>
```

7.8 Exemplo Completo de Formulário

```html
```

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <title>Formulário de Cadastro</title>
  <style>
    form {
      max-width: 500px;
      margin: 50px auto;
      padding: 20px;
      border: 1px solid #ddd;
      border-radius: 5px;
      background-color: #f9f9f9;
    }

    .form-group {
      margin-bottom: 20px;
    }

    label {
      display: block;
      margin-bottom: 5px;
      font-weight: bold;
      color: #333;
    }

    input, select, textarea {
      width: 100%;
      padding: 10px;
      border: 1px solid #ddd;
      border-radius: 4px;
      box-sizing: border-box;
      font-size: 14px;
    }

    input:focus, select:focus, textarea:focus {
      outline: none;
      border-color: #007bff;
      box-shadow: 0 0 5px rgba(0, 123, 255, 0.5);
    }

    button {
      background-color: #007bff;
      color: white;
      padding: 10px 20px;
      border: none;
```

```
      border-radius: 4px;
      cursor: pointer;
      font-size: 16px;
      width: 100%;
    }

    button:hover {
      background-color: #0056b3;
    }

    .error {
      color: red;
      font-size: 12px;
      margin-top: 5px;
    }
  </style>
</head>
<body>
  <form action="cadastro.php" method="POST">
    <h2>Cadastro de Usuário</h2>

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

    <div class="form-group">
      <label for="email">Email:</label>
      <input
        type="email"
        id="email"
        name="email"
        placeholder="seu@email.com"
        required
      >
    </div>

    <div class="form-group">
      <label for="telefone">Telefone:</label>
      <input
        type="tel"
```

```
      id="telefone"
      name="telefone"
      placeholder="(XX) XXXXX-XXXX"
    >
  </div>

  <div class="form-group">
    <label for="data_nascimento">Data de Nascimento:</label>
    <input
      type="date"
      id="data_nascimento"
      name="data_nascimento"
      required
    >
  </div>

  <div class="form-group">
    <label for="genero">Gênero:</label>
    <select id="genero" name="genero">
      <option value="">Selecione</option>
      <option value="masculino">Masculino</option>
      <option value="feminino">Feminino</option>
      <option value="outro">Outro</option>
    </select>
  </div>

  <div class="form-group">
    <label>Interesses:</label>
    <label>
      <input type="checkbox" name="interesses" value="esportes">
      Esportes
    </label>
    <label>
      <input type="checkbox" name="interesses" value="musica">
      Música
    </label>
    <label>
      <input type="checkbox" name="interesses" value="leitura">
      Leitura
    </label>
  </div>

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

```
        rows="4"
        placeholder="Digite sua mensagem aqui..."
      ></textarea>
    </div>

    <div class="form-group">
      <label>
        <input type="checkbox" name="concordo" required>
        Concordo com os termos e condições
      </label>
    </div>

    <button type="submit">Cadastrar</button>
    <button type="reset">Limpar</button>
  </form>
</body>
</html>
```

7.9 Validação com JavaScript

```
html
```

```html
<!DOCTYPE html>
<html>
<head>
  <title>Validação de Formulário</title>
</head>
<body>
  <form id="meuForm">
    <label for="nome">Nome:</label>
    <input type="text" id="nome" name="nome">
    <span id="erroNome"></span>

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

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

  <script>
    const form = document.getElementById('meuForm');

    form.addEventListener('submit', function(e) {
      e.preventDefault();  // Impede envio padrão

      // Validar nome
      const nome = document.getElementById('nome').value.trim();
      const erroNome = document.getElementById('erroNome');

      if (nome === '') {
        erroNome.textContent = 'Nome é obrigatório';
        erroNome.style.color = 'red';
        return false;
      } else if (nome.length < 3) {
        erroNome.textContent = 'Nome deve ter pelo menos 3 caracteres';
        erroNome.style.color = 'red';
        return false;
      } else {
        erroNome.textContent = '';
      }

      // Validar email
      const email = document.getElementById('email').value.trim();
      const erroEmail = document.getElementById('erroEmail');
      const regexEmail = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

      if (email === '') {
```

```
        erroEmail.textContent = 'Email é obrigatório';
        erroEmail.style.color = 'red';
        return false;
      } else if (!regexEmail.test(email)) {
        erroEmail.textContent = 'Email inválido';
        erroEmail.style.color = 'red';
        return false;
      } else {
        erroEmail.textContent = '';
      }

      // Se passou em todas as validações
      alert('Formulário válido! Enviando...');
      // form.submit();  // Descomente para enviar de fato
    });
  </script>
</body>
</html>
```

MÓDULO 8: WORDPRESS - CMS PRÁTICO

8.1 Por que Usar WordPress?

WordPress é utilizado por mais de 40% de todos os sites na internet. É fácil de usar, seguro, e tem comunidade ativa. Você pode criar desde blogs até e-commerce.

8.2 Instalação do WordPress Localmente

Opção 1 - Com XAMPP:

1. Baixe XAMPP (Apache + MySQL)
2. Inicie Apache e MySQL
3. Acesse phpMyAdmin em localhost/phpmyadmin
4. Crie novo banco de dados: "wordpress_local"
5. Baixe WordPress em wordpress.org
6. Descompacte em C:\xampp\htdocs\wordpress
7. Acesse localhost/wordpress
8. Siga o assistente

Opção 2 - Com Laragon (mais fácil):

1. Baixe Laragon
2. Instale e inicie
3. Clique direito > Quick App > WordPress
4. Siga o assistente
5. Acesse em localhost/wordpress

Opção 3 - Local by Flywheel (melhor interface):

1. Baixe Local
2. Clique em "Create Site"
3. Escolha nome e preencha dados
4. Siga o assistente

8.3 Interface do WordPress

Painel Administrativo:

- **Dashboard:** Visão geral do site
- **Artigos:** Criar, editar posts
- **Páginas:** Páginas estáticas
- **Mídia:** Gerenciar imagens e arquivos
- **Comentários:** Moderar comentários
- **Aparência:** Temas e widgets
- **Plugins:** Estender funcionalidades
- **Usuários:** Gerenciar usuários
- **Configurações:** Ajustes gerais

8.4 Criando Conteúdo no WordPress

Criar um Post:

1. Vá em Artigos > Adicionar Novo
2. Preencha título
3. Digite conteúdo no editor
4. Defina categoria
5. Adicione tags

6. Configure imagem destaque

7. Clique "Publicar"

Criar uma Página:

1. Vá em Páginas > Adicionar Nova

2. Similar aos posts mas para conteúdo estático

3. Usado para Sobre, Contato, etc

8.5 Temas WordPress

Temas controlam a aparência do seu site. Você pode:

1. Ir em Aparência > Temas

2. Procurar temas gratuitos ou premium

3. Instalar e ativar

4. Personalizar com Personalizador

Temas populares:

- Astra
- OceanWP
- GeneratePress
- Neve
- Hello Elementor

8.6 Plugins WordPress

Plugins adicionam funcionalidades ao WordPress.

Plugins essenciais:

- **Yoast SEO:** Otimização para mecanismos de busca
- **Akismet:** Proteção contra spam
- **WooCommerce:** Transformar em loja online
- **Elementor:** Page builder visual
- **Contact Form 7:** Criar formulários
- **Jetpack:** Backup e segurança
- **Wordfence:** Segurança adicional

Instalando plugin:

1. Vá em Plugins > Adicionar Novo

2. Pesquise o plugin

3. Clique "Instalar Agora"

4. Clique "Ativar"

8.7 Personalizando WordPress com CSS

Você pode adicionar CSS customizado sem editar temas:

1. Vá em Aparência > Personalizador

2. Procure por "CSS Adicional" ou "Custom CSS"

3. Adicione seu CSS

Exemplo:

```css
/* Mudar cor do título */
h1 {
  color: #007bff;
  font-size: 36px;
}

/* Estilizar posts */
.post {
  background-color: #f9f9f9;
  padding: 20px;
  border-radius: 5px;
}
```

8.8 WordPress vs HTML/CSS Puro

Aspecto	WordPress	HTML/CSS
Facilidade	Muito fácil	Requer código
Customização	Limitada sem código	Completa
Segurança	Gerenciada	Sua responsabilidade
Plugins	Muitos disponíveis	Precisa programar
Performance	Boa com otimização	Potencialmente melhor
Aprendizado	Rápido	Mais tempo

RESUMO ESTRUTURAL DE TODAS AS LINGUAGENS

HTML - Estrutura

```
<!DOCTYPE html>
<html>
├── <head>
│   ├── <meta>
│   ├── <title>
│   ├── <link> (CSS)
│   └── <style>
└── <body>
    ├── <header>
    ├── <nav>
    ├── <main>
    │   ├── <article>
    │   └── <section>
    ├── <aside>
    └── <footer>
```

Função: Conteúdo e estrutura

Tags principais: div, p, a, img, form, table, ul, ol, h1-h6

CSS - Estilos

```
Seletores
  └── Propriedades CSS
      ├── Color
      ├── Font
      ├── Box Model
      ├── Layout (Flexbox/Grid)
      ├── Posicionamento
      ├── Transições
      └── Animações
```

Função: Aparência visual

Conceitos: Cascata, Especificidade, Box Model, Responsividade

JavaScript - Comportamento

```
Eventos
  └── DOM Manipulation
      ├── Seleção
      ├── Modificação
```

```
    ├── Adição/Remoção
    └── Validação
```

Função: Interatividade dinâmica

Conceitos: Variáveis, Funções, Eventos, API DOM

DICAS FINAIS PARA PROVA

1. **Estruture bem o HTML** - use tags semânticas
2. **Estilize com CSS** - mobile first, use Flexbox/Grid
3. **Adicione interatividade** - com JavaScript e eventos
4. **Valide formulários** - HTML5 + JavaScript
5. **Teste responsividade** - em diferentes telas
6. **Use ferramentas** - DevTools do navegador
7. **Organize código** - indentação e comentários
8. **Pratique projetos** - coloque em GitHub Pages
9. **Entenda conceitos** - não apenas memorize sintaxe
10. **Estude regularmente** - programação requer prática constante

---

EXERCÍCIOS PRÁTICOS SUGERIDOS

1. Criar página pessoal em HTML/CSS
2. Fazer portfolio responsivo com Flexbox
3. Criar formulário com validação JavaScript
4. Publicar página no GitHub Pages
5. Instalar WordPress e criar conteúdo
6.