Guia Completo de Desenvolvimento Web

Sumário

---

1. Introdução ao Curso e Carreira de Desenvolvedor Web {#introdução}

O que é Desenvolvimento Web?

Desenvolvimento web é a prática de criar aplicações e sites que funcionam na internet. Um desenvolvedor web constrói desde simples páginas estáticas até aplicações complexas que processam dados em tempo real.

Oportunidades de Carreira

A carreira de desenvolvedor web oferece múltiplas especializações:

Frontend Developer: Trabalha com a interface visual que o usuário vê. Utiliza HTML, CSS e JavaScript para criar experiências interativas.

Backend Developer: Trabalha com servidores, bancos de dados e lógica de negócio. Gerencia o processamento de dados e autenticação.

Full-Stack Developer: Domina tanto frontend quanto backend, capaz de construir aplicações completas do zero até a publicação.

Especialistas: Web Designer (foco em design), DevOps (infraestrutura), Arquiteto de Software, etc.

Mercado Atual

De acordo com pesquisas recentes do Stack Overflow Developer Survey (2024-2025), as tecnologias mais procuradas são:

- **Linguagens**: JavaScript, HTML/CSS, Python, SQL, TypeScript
- **Frameworks**: Node.js, React.js, Express.js, Next.js
- **Ferramentas**: Docker, npm, Git
- **Plataformas em Nuvem**: AWS, Microsoft Azure, Google Cloud

Habilidades Essenciais para Iniciantes

1. **Pensamento lógico**: Capacidade de decompor problemas
2. **Conhecimento de algoritmos**: Básicos para resolver problemas
3. **Comunicação**: Trabalho em equipe é fundamental
4. **Aprendizado contínuo**: A tecnologia evolui constantemente
5. **Paciência**: Debugging e resolução de problemas requer persistência

---

2. Desenvolvimento de Software e Fundamentos {#desenvolvimento-de-software}

Ciclo de Vida do Desenvolvimento (SDLC)

O processo de desenvolvimento segue etapas bem definidas:

Planejamento: Define objetivos, escopo e recursos necessários.

Análise: Coleta requisitos e especifica o que o software deve fazer.

Design: Arquiteta a solução, define estrutura e componentes.

Implementação: Escreve o código seguindo o design.

Testes: Valida se o software funciona conforme especificado.

Deployment: Coloca a aplicação em produção.

Manutenção: Corrige bugs e adiciona melhorias.

Paradigmas de Programação

Programação Procedural: Código baseado em funções e procedimentos sequenciais. Exemplo: C, Pascal.

Programação Orientada a Objetos (OOP): Organiza código em objetos que possuem dados e comportamentos. Exemplo: Java, C++, Python.

Programação Funcional: Trata a computação como uma sequência de funções matemáticas. Exemplo: Lisp, Haskell.

Programação Declarativa: Descreve o resultado desejado sem focar em como conseguir. Exemplo: HTML, SQL.

Boas Práticas de Desenvolvimento

DRY (Don't Repeat Yourself): Evite duplicação de código. Reutilize funções e componentes.

KISS (Keep It Simple, Stupid): Mantenha soluções simples e diretas.

SOLID: Princípios para design orientado a objetos:

- S: Single Responsibility - uma classe com uma responsabilidade
- O: Open/Closed - aberto para extensão, fechado para modificação
- L: Liskov Substitution - subclasses devem substituir classes base
- I: Interface Segregation - interfaces específicas é melhor que gerais
- D: Dependency Inversion - dependa de abstrações, não implementações

Controle de Versão: Use Git para rastrear mudanças no código.

Documentação: Mantenha documentação clara e atualizada.

---

3. Redes de Computadores {#redes}

Conceitos Fundamentais

Uma rede de computadores é um conjunto de computadores conectados que compartilham dados e recursos.

Modelo OSI (Open Systems Interconnection)

O modelo OSI descreve 7 camadas de funcionamento das redes:

1. **Física**: Cabos, sinais, conectores

2. **Enlace**: Endereços MAC, switches
3. **Rede**: Roteamento, IP, protocolos
4. **Transporte**: TCP, UDP, garantia de entrega
5. **Sessão**: Estabelecimento de conexão
6. **Apresentação**: Criptografia, compressão
7. **Aplicação**: HTTP, HTTPS, FTP, SMTP

TCP/IP - Protocolo Principal da Internet

TCP (Transmission Control Protocol): Protocolo confiável que garante entrega de dados em ordem.

IP (Internet Protocol): Protocolo que roteia dados através de redes usando endereços IP.

Endereço IP: Identificador único de um computador na rede (exemplo: 192.168.1.1).

Porta: Identificador que especifica qual serviço usar (HTTP porta 80, HTTPS porta 443, SSH porta 22).

HTTP e HTTPS

HTTP (HyperText Transfer Protocol): Protocolo sem estado para transferência de documentos web.

HTTPS: Versão segura do HTTP com criptografia SSL/TLS.

Métodos HTTP principais:

- GET: Solicita um recurso
- POST: Envia dados para processamento
- PUT: Atualiza um recurso completo
- DELETE: Remove um recurso
- PATCH: Atualiza parte de um recurso

Códigos de Status:

- 200-299: Sucesso
- 300-399: Redirecionamento
- 400-499: Erro do cliente

- 500-599: Erro do servidor

DNS (Domain Name System)

Traduz nomes de domínio (exemplo.com) para endereços IP, permitindo acesso fácil aos sites.

---

4. Principais Linguagens Web {#linguagens-web}

HTML (HyperText Markup Language)

Propósito: Estruturar conteúdo de página web.

Características:

- Linguagem de marcação, não de programação
- Baseada em tags/elementos
- Define semântica do conteúdo
- Caso-insensível mas melhor praticar lowercase

Elementos Principais:

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Título da Página</title>
  </head>
  <body>
    <h1>Cabeçalho Principal</h1>
    <p>Parágrafo de texto</p>
    <a href="https://exemplo.com">Link</a>
    <img src="imagem.jpg" alt="Descrição">
  </body>
</html>
```

CSS (Cascading Style Sheets)

Propósito: Estilizar e posicionar elementos HTML.

Características:

- Responsável pela aparência visual

- Utiliza seletores, propriedades e valores
- Suporta responsividade (mobile-first)

Exemplo:

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

h1 {
  color: #333;
  font-size: 2em;
}

@media (max-width: 600px) {
  h1 {
    font-size: 1.5em;
  }
}
```

JavaScript

Propósito: Adicionar interatividade e lógica à página.

Características:

- Linguagem de programação completa
- Executa no navegador (client-side)
- Manipula o DOM dinamicamente
- Suporta programação orientada a objetos e funcional

Exemplo:

```javascript
```

```
// Função simples
function saudacao(nome) {
  return `Olá, ${nome}!`;
}

// Manipular DOM
document.getElementById('botao').addEventListener('click', () => {
  console.log('Botão clicado');
});

// Requisição AJAX
fetch('https://api.exemplo.com/dados')
  .then(resposta => resposta.json())
  .then(dados => console.log(dados));
```

Python

Propósito: Backend, ciência de dados, automação.

Características:

- Sintaxe limpa e legível
- Dinâmico e interpretado
- Grande comunidade e bibliotecas

SQL

Propósito: Consultar e manipular dados em bancos de dados.

Características:

- Linguagem declarativa
- Estruturada em SELECT, INSERT, UPDATE, DELETE
- Essencial para qualquer desenvolvedor

Exemplo:

```sql
SELECT * FROM usuarios WHERE idade > 18;
INSERT INTO usuarios (nome, email) VALUES ('João', 'joao@email.com');
UPDATE usuarios SET nome = 'Silva' WHERE id = 1;
DELETE FROM usuarios WHERE id = 5;
```

---

5. DOM - Document Object Model {#dom}

O que é DOM?

O DOM é uma representação em árvore de um documento HTML. Cada elemento HTML torna-se um nó que pode ser acessado e modificado via JavaScript.

Estrutura do DOM

```
Document
├── html
│   ├── head
│   │   ├── meta
│   │   └── title
│   └── body
│       ├── h1
│       ├── p
│       └── div
│           └── span
```

Acessando Elementos

```javascript
// Por ID
const elemento = document.getElementById('meu-id');

// Por classe
const elementos = document.getElementsByClassName('minha-classe');

// Por seletor CSS
const elemento = document.querySelector('.classe');
const elementos = document.querySelectorAll('.classe');

// Por tag
const paragrafos = document.getElementsByTagName('p');
```

Manipulando o DOM

```javascript
```

```javascript
// Alterando conteúdo
elemento.textContent = 'Novo texto';
elemento.innerHTML = '<strong>Texto em negrito</strong>';

// Alterando atributos
elemento.setAttribute('href', 'https://novo-link.com');
elemento.id = 'novo-id';

// Alterando estilos
elemento.style.color = 'red';
elemento.style.fontSize = '20px';

// Adicionando/removendo classes
elemento.classList.add('ativa');
elemento.classList.remove('inativa');
elemento.classList.toggle('destaque');

// Criando elementos
const novo = document.createElement('div');
novo.textContent = 'Conteúdo novo';
document.body.appendChild(novo);

// Removendo elementos
elemento.remove();
elementoPai.removeChild(elemento);
```

Event Listeners

```javascript
// Evento de clique
elemento.addEventListener('click', (evento) => {
  console.log('Clicado');
});

// Evento de submit
formulario.addEventListener('submit', (evento) => {
  evento.preventDefault(); // Impede recarregar página
  console.log('Formulário enviado');
});

// Eventos comuns: change, input, mouseover, mouseout, keydown, load
```

Properties Úteis do DOM

```javascript
```

```
// Navegação entre elementos
elemento.parentElement;
elemento.firstChild;
elemento.lastChild;
elemento.nextSibling;
elemento.previousSibling;

// Informações
elemento.innerText;
elemento.innerHTML;
elemento.outerHTML;
elemento.nodeType;
elemento.nodeName;

// Dimensões
elemento.offsetWidth;
elemento.offsetHeight;
elemento.offsetTop;
elemento.offsetLeft;
```

---

6. Hospedagem e Publicação de Sites {#hospedagem}

Tipos de Hospedagem

Hospedagem Compartilhada: Múltiplos sites em um mesmo servidor. Econômico mas com limitações.

VPS (Virtual Private Server): Servidor virtual dedicado. Mais controle e melhor performance que compartilhado.

Servidor Dedicado: Servidor físico exclusivo. Máxima performance e controle.

Cloud Hosting: Hospedagem em nuvem (AWS, Azure, Google Cloud). Escalável e flexível.

Static Hosting: Para sites estáticos (GitHub Pages, Netlify). Gratuito ou muito barato.

Servidores Web

Apache: Servidor web tradicional, multiplataforma, amplamente usado.

Nginx: Servidor web moderno, leve, excelente performance.

IIS: Servidor web da Microsoft para Windows.

Publicando sua Aplicação

Passo 1: Escolha um Host

- Iniciantes: GitHub Pages, Netlify, Vercel (gratuitos)
- Pequenos projetos: Hostinger, Bluehost, SiteGround
- Empresas: AWS, Azure, Google Cloud

Passo 2: Configure Domínio

- Registre seu domínio (GoDaddy, Namecheap, etc.)
- Aponte DNS para seu host

Passo 3: Implante Aplicação

- Upload via FTP/SFTP (Filezilla)
- Git push para GitHub Pages
- Deploy automático via CI/CD

Passo 4: Configure SSL/TLS

- Protocolo HTTPS para segurança
- Muitos hosts oferecem certificados gratuitos (Let's Encrypt)

Deploy com Git

```bash
# Inicializar repositório
git init
git remote add origin https://github.com/usuario/repo.git

# Fazer commit
git add .
git commit -m "Initial commit"

# Enviar para repositório
git push -u origin main
```

---

7. Desenvolvimento Full-Stack {#full-stack}

O que é Full-Stack?

Um desenvolvedor Full-Stack trabalha com frontend, backend e infraestrutura, podendo construir aplicações completas do zero até a publicação.

Stack Typical

Frontend:

- HTML, CSS, JavaScript
- Frameworks: React, Vue, Angular
- TypeScript para tipagem estática

Backend:

- Node.js/Express (JavaScript)
- Python/Django ou Flask
- PHP/Laravel
- Banco de dados: PostgreSQL, MySQL, MongoDB

Ferramentas:

- Git para versionamento
- Docker para containerização
- APIs RESTful ou GraphQL

Fluxo de Desenvolvimento Full-Stack

```
1. Design da aplicação (arquitetura)
2. Criar banco de dados
3. Implementar API backend
4. Testar API (Postman)
5. Construir interface frontend
6. Integrar frontend com backend
7. Testes e-2-e
8. Deploy em produção
9. Monitoramento e manutenção
```

Exemplo Simples: MERN Stack (MongoDB, Express, React, Node)

```javascript
// Backend (Node.js + Express)
const express = require('express');
const app = express();

app.get('/api/usuarios', (req, res) => {
  res.json({ usuarios: ['João', 'Maria'] });
});

app.listen(3001, () => console.log('Servidor rodando'));

// Frontend (React)
import React, { useState, useEffect } from 'react';

function App() {
  const [usuarios, setUsuarios] = useState([]);

  useEffect(() => {
    fetch('/api/usuarios')
      .then(res => res.json())
      .then(dados => setUsuarios(dados.usuarios));
  }, []);

  return (
    <div>
      {usuarios.map(u => <p key={u}>{u}</p>)}
    </div>
  );
}
```

---

8. WordPress e GitHub Pages {#wordpress-github}

WordPress

O que é: Sistema de gerenciamento de conteúdo (CMS) baseado em PHP que permite criar sites sem necessidade de escrever muito código.

Quando usar: Blogs, sites corporativos, lojas online (com WooCommerce), portais.

Instalando WordPress

Pré-requisitos:

- Servidor com PHP e MySQL (MAMP, XAMPP, WAMP)
- Acesso ao servidor via FTP ou SSH

Processo:

1. Download do WordPress.org
2. Criar banco de dados MySQL
3. Upload para servidor
4. Executar instalação (wp-admin/install.php)
5. Criar usuário admin

Estrutura WordPress

- **Posts**: Conteúdo que aparece em feed chronológico
- **Pages**: Conteúdo estático (Sobre, Contato)
- **Temas**: Templates que definem design
- **Plugins**: Extensões de funcionalidade
- **Widgets**: Componentes reutilizáveis (barra lateral)
- **Taxonomias**: Categorias e tags para organizar conteúdo

Painel de Administração

O dashboard oferece:

- **Posts/Pages**: Criar e gerenciar conteúdo
- **Mídia**: Upload e organização de imagens
- **Appearence**: Temas e customização
- **Plugins**: Instalar e ativar plugins
- **Users**: Gerenciar usuários
- **Settings**: Configurações gerais

Plugin WooCommerce

Para criar loja virtual:

1. Instale WooCommerce do diretório de plugins
2. Configure configurações (moeda, país, imposto)
3. Configure gateway de pagamento

4. Crie produtos

5. Customize aparência da loja

GitHub Pages

O que é: Serviço de hospedagem gratuito do GitHub para sites estáticos.

Características:

- Gratuito
- Integrado com Git
- Suporta Jekyll (gerador de sites estáticos)
- Domínio gratuito ou use seu domínio

Publicando no GitHub Pages

Método 1: Repositório User/Organization

```bash
# Crie repositório chamado username.github.io
git clone https://github.com/username/username.github.io
cd username.github.io

# Adicione arquivos
echo "Hello World" > index.html

git add .
git commit -m "Initial commit"
git push -u origin main

# Acesse em https://username.github.io
```

Método 2: Repositório Project

```bash
# Repositório pode ter qualquer nome
git clone https://github.com/username/meu-projeto
cd meu-projeto

# Configure branch gh-pages nas settings
# Arquivos em gh-pages ficarão disponíveis em:
# https://username.github.io/meu-projeto/
```

---

9. HTML Essencial {#html}

Estrutura Base

```html
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Descrição da página">
  <title>Título do Site</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <header>Cabeçalho</header>
  <nav>Menu de navegação</nav>
  <main>Conteúdo principal</main>
  <footer>Rodapé</footer>

  <script src="script.js"></script>
</body>
</html>
```

Elementos Semânticos

```html
<header>      <!-- Cabeçalho da página/seção -->
<nav>         <!-- Navegação -->
<main>        <!-- Conteúdo principal -->
<article>     <!-- Conteúdo independente -->
<section>     <!-- Seção temática -->
<aside>       <!-- Conteúdo lateral -->
<footer>      <!-- Rodapé -->
<figure>      <!-- Figura com legenda -->
<figcaption>  <!-- Legenda de figura -->
<time>        <!-- Data/hora -->
```

Elementos de Texto

```html
```

```
<h1> até <h6>     <!-- Cabeçalhos de diferentes níveis -->
<p>               <!-- Parágrafo -->
<strong>          <!-- Texto importante (negrito semântico) -->
<em>              <!-- Ênfase (itálico semântico) -->
<b>               <!-- Negrito visual -->
<i>               <!-- Itálico visual -->
<mark>            <!-- Texto marcado/destacado -->
<code>            <!-- Código inline -->
<pre>             <!-- Código pré-formatado -->
<blockquote>      <!-- Citação em bloco -->
<ul>              <!-- Lista não-ordenada -->
<ol>              <!-- Lista ordenada -->
<li>              <!-- Item de lista -->
<dl>              <!-- Lista de definições -->
<dt>              <!-- Termo -->
<dd>              <!-- Definição -->
```

Elementos de Mídia

```html
<img src="imagem.jpg" alt="Descrição" />
<picture>
  <source media="(max-width: 600px)" srcset="pequena.jpg">
  <img src="grande.jpg" alt="Imagem responsiva">
</picture>

<video width="320" height="240" controls>
  <source src="video.mp4" type="video/mp4">
  Seu navegador não suporta vídeo
</video>

<audio controls>
  <source src="audio.mp3" type="audio/mpeg">
  Seu navegador não suporta áudio
</audio>

<iframe src="https://example.com"></iframe>
```

Atributos Globais Importantes

```html
```

```
id="identificador"          <!-- Identificador único -->
class="classe1 classe2"      <!-- Classes para CSS/JS -->
style="color: red;"          <!-- Estilos inline -->
data-custom="valor"          <!-- Dados customizados -->
title="Dica"                <!-- Texto de hover -->
lang="pt-BR"                <!-- Idioma -->
dir="ltr"                   <!-- Direção (ltr/rtl) -->
role="button"               <!-- Papel ARIA para acessibilidade -->
aria-label="Descrição"      <!-- Descrição para leitores de tela -->
```

Caracteres Especiais

```
html

&lt;                <!-- < -->
&gt;                <!-- > -->
&amp;               <!-- & -->
&quot;              <!-- " -->
&apos;              <!-- ' -->
               <!-- Espaço não-quebrável -->
&copy;              <!-- © -->
&reg;               <!-- ® -->
&deg;               <!-- ° -->
&euro;              <!-- € -->
```

---

10. CSS e Styling {#css}

Formas de Incluir CSS

```
html

<!-- CSS Inline -->
<p style="color: red;">Texto vermelho</p>

<!-- CSS Interno -->
<style>
  p { color: blue; }
</style>

<!-- CSS Externo (recomendado) -->
<link rel="stylesheet" href="styles.css">
```

Seletores

```css
/* Elemento */
p { color: blue; }

/* Classe */
.destaque { color: red; }

/* ID */
#principal { font-size: 2em; }

/* Atributo */
input[type="text"] { border: 1px solid gray; }

/* Combinadores */
div p { }              /* Descendentes */
div > p { }           /* Filhos diretos */
p + span { }          /* Próximo irmão */
p ~ span { }          /* Qualquer irmão seguinte */

/* Pseudo-classes */
a:hover { color: red; }
input:focus { outline: 2px solid blue; }
li:first-child { font-weight: bold; }
li:nth-child(2n) { background: gray; }
input:checked { }
p:not(.destaque) { }

/* Pseudo-elementos */
p::first-line { color: red; }
p::first-letter { font-size: 2em; }
div::before { content: ">>"; }
div::after { content: "<<"; }
```

Box Model

```css
```

```css
/* Margin: espaço fora da borda */
/* Border: borda do elemento */
/* Padding: espaço dentro da borda */
/* Content: conteúdo do elemento */

div {
  margin: 10px;              /* Margem em todos os lados */
  margin: 10px 20px;         /* Superior/inferior, esquerda/direita */
  margin: 10px 20px 30px;    /* Superior, esquerda/direita, inferior */
  margin: 10px 20px 30px 40px; /* Superior, direita, inferior, esquerda */

  padding: 10px;
  border: 1px solid black;
  box-sizing: border-box;    /* Largura inclui padding e border */
}
```

Display

```css
display: block;        /* Ocupa toda a largura -->
display: inline;       /* Ocupa apenas espaço necessário -->
display: inline-block; /* Híbrido: inline mas aceita width/height -->
display: flex;         /* Layout flexível -->
display: grid;         /* Layout em grid -->
display: none;         /* Não aparece na página -->
display: hidden;       /* Aparece mas não ocupa espaço -->
```

Flexbox

```css
```

```css
.container {
  display: flex;
  flex-direction: row;           /* row, column, row-reverse, column-reverse -->
  justify-content: center;       /* Alinhamento horizontal -->
  align-items: center;           /* Alinhamento vertical -->
  gap: 10px;                     /* Espaço entre items -->
  flex-wrap: wrap;               /* Quebra em múltiplas linhas -->
}

.item {
  flex: 1;                       /* Cresce para preencher espaço -->
  flex-basis: 200px;            /* Tamanho base -->
  flex-grow: 1;                 /* Crescimento -->
  flex-shrink: 1;               /* Encolhimento -->
}
```

Grid

```css
.container {
  display: grid;
  grid-template-columns: 1fr 2fr 1fr;  /* 3 colunas -->
  grid-template-rows: 100px auto 50px; /* 3 linhas -->
  gap: 10px;                           /* Espaço entre células -->
  grid-auto-flow: row;                 /* Dense, row, column -->
}

.item {
  grid-column: 1 / 3;                  /* Ocupa coluna 1 e 2 -->
  grid-row: 1 / 3;                     /* Ocupa linha 1 e 2 -->
}
```

Cores

```css
```

```css
/* Nomes */
color: red;

/* Hexadecimal -->
color: #ff0000;
color: #f00; /* Forma curta -->

/* RGB -->
color: rgb(255, 0, 0);
color: rgba(255, 0, 0, 0.5); /* Com transparência -->

/* HSL -->
color: hsl(0, 100%, 50%);
color: hsla(0, 100%, 50%, 0.5);
```

Tipografia

```css
font-family: Arial, sans-serif;           /* Fonte -->
font-size: 16px;                          /* Tamanho -->
font-weight: bold; /* 400, 700, 900 */   /* Espessura -->
font-style: italic;                       /* Estilo -->
line-height: 1.5;                         /* Altura de linha -->
text-align: center; /* left, right, justify */ /* Alinhamento -->
text-decoration: underline;               /* Decoração -->
letter-spacing: 2px;                      /* Espaço entre letras -->
text-transform: uppercase;                /* Transformação de caso -->
```

Posicionamento

```css
position: static;      /* Padrão, normal -->
position: relative;    /* Relativo ao seu posicionamento normal -->
position: absolute;    /* Relativo ao pai posicionado -->
position: fixed;       /* Relativo ao viewport (fica fixo ao scroll) -->
position: sticky;      /* Híbrido: normal até scroll, depois fixo -->

/* Propriedades de posicionamento -->
top: 10px;
right: 10px;
bottom: 10px;
left: 10px;
z-index: 10;           /* Ordem de sobreposição -->
```

Responsividade

```css
/* Mobile-first approach -->
.container {
  display: grid;
  grid-template-columns: 1fr;
}

/* Tablet -->
@media (min-width: 768px) {
  .container {
    grid-template-columns: 1fr 1fr;
  }
}

/* Desktop -->
@media (min-width: 1024px) {
  .container {
    grid-template-columns: 1fr 1fr 1fr;
  }
}
```

Efeitos e Animações

```css
```

```css
/* Transições -->
transition: all 0.3s ease;
transition-property: color, background;
transition-duration: 0.3s;
transition-timing-function: ease, linear, ease-in, ease-out;
transition-delay: 0.1s;

/* Transformações -->
transform: rotate(45deg);
transform: scale(1.5);
transform: translateX(10px);
transform: skew(10deg);

/* Animações -->
@keyframes deslizar {
  0% { left: 0; }
  50% { left: 50%; }
  100% { left: 100%; }
}

div {
  animation: deslizar 2s infinite;
}

/* Sombras -->
box-shadow: 2px 2px 5px rgba(0,0,0,0.3);
text-shadow: 2px 2px 4px gray;

/* Opacidade -->
opacity: 0.5;

/* Filtros -->
filter: blur(5px);
filter: brightness(1.2);
filter: contrast(1.5);
```

---

11. Formulários Web {#formularios}

Estrutura Base

```
html
```

```html
<form action="/enviar" method="POST" enctype="multipart/form-data">
  <!-- Campos do formulário -->
  <input type="text" name="nome" required>
  <button type="submit">Enviar</button>
</form>
```

Tipos de Input

```
html
```

```html
<!-- Texto -->
<input type="text" name="nome" placeholder="Digite seu nome">

<!-- Email -->
<input type="email" name="email" required>

<!-- Senha -->
<input type="password" name="senha">

<!-- Número -->
<input type="number" name="idade" min="0" max="120">

<!-- Telefone -->
<input type="tel" name="telefone" pattern="[0-9]{3}-[0-9]{4}-[0-9]{4}">

<!-- URL -->
<input type="url" name="website">

<!-- Data -->
<input type="date" name="data_nascimento">

<!-- Hora -->
<input type="time" name="horario">

<!-- Data e Hora -->
<input type="datetime-local" name="agendamento">

<!-- Mês -->
<input type="month" name="mes">

<!-- Semana -->
<input type="week" name="semana">

<!-- Cor -->
<input type="color" name="cor" value="#ff0000">

<!-- Range/Slider -->
<input type="range" name="volume" min="0" max="100" value="50">

<!-- Checkbox -->
<input type="checkbox" name="termos" id="termos">
<label for="termos">Concordo com os termos</label>

<!-- Radio Button -->
<input type="radio" name="genero" value="m" id="masculino">
<label for="masculino">Masculino</label>
```

```
<input type="radio" name="genero" value="f" id="feminino">
<label for="feminino">Feminino</label>

<!-- File Upload -->
<input type="file" name="arquivo" accept=".pdf,.doc,.docx">

<!-- Hidden -->
<input type="hidden" name="user_id" value="123">

<!-- Search -->
<input type="search" name="busca" placeholder="Buscar...">

<!-- Submit -->
<input type="submit" value="Enviar">

<!-- Reset -->
<input type="reset" value="Limpar">

<!-- Button -->
<input type="button" value="Clique-me" onclick="alerta()">
```

Elementos de Formulário Avançados

```
html
```

```html
<!-- Textarea -->
<textarea name="mensagem" rows="5" cols="40" placeholder="Digite sua mensagem"></tex

<!-- Select (Dropdown) -->
<select 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 Optgroup -->
<select name="continente">
  <optgroup label="América">
    <option value="br">Brasil</option>
    <option value="us">EUA</option>
  </optgroup>
  <optgroup label="Europa">
    <option value="pt">Portugal</option>
    <option value="es">Espanha</option>
  </optgroup>
</select>

<!-- Select Múltiplo -->
<select name="habilidades" multiple>
  <option value="html">HTML</option>
  <option value="css">CSS</option>
  <option value="js">JavaScript</option>
</select>

<!-- Datalist (Sugestões) -->
<input list="navegadores" name="browser">
<datalist id="navegadores">
  <option value="Chrome">
  <option value="Firefox">
  <option value="Safari">
</datalist>

<!-- Output (Resultado de cálculo) -->
<input type="number" id="n1" value="5"> +
<input type="number" id="n2" value="3"> =
<output id="resultado">8</output>

<!-- Progress -->
<progress value="70" max="100"></progress>
```

```html
<!-- Meter (Medição) -->
<meter value="6" min="0" max="10" low="3" high="7"></meter>

<!-- Details e Summary -->
<details>
  <summary>Mais informações</summary>
  <p>Conteúdo detalhado que fica oculto até clique</p>
</details>
```

Validação de Formulários

```html
<!-- Atributos de Validação -->
<form>
  <!-- Required -->
  <input type="text" required>

  <!-- Min/Max -->
  <input type="number" min="1" max="100">

  <!-- MinLength/MaxLength -->
  <input type="text" minlength="3" maxlength="50">

  <!-- Pattern (Expressão regular) -->
  <input type="text" pattern="[A-Za-z]{3}" title="Apenas 3 letras">

  <!-- Step (Incremento para números) -->
  <input type="number" step="5">

  <!-- Multiple (Múltiplos arquivos) -->
  <input type="file" multiple>

  <!-- Accept (Tipos de arquivo aceitos) -->
  <input type="file" accept="image/*">
</form>
```

Validação com JavaScript

```javascript
```

```javascript
// Validar formulário simples
const form = document.querySelector('form');

form.addEventListener('submit', (e) => {
  e.preventDefault();

  const email = form.email.value;
  const senha = form.senha.value;

  if (!email.includes('@')) {
    alert('Email inválido');
    return;
  }

  if (senha.length < 6) {
    alert('Senha deve ter pelo menos 6 caracteres');
    return;
  }

  // Enviar formulário
  form.submit();
});

// API Constraint Validation
const input = document.querySelector('input[type="email"]');

input.addEventListener('blur', () => {
  if (!input.validity.valid) {
    console.log(input.validationMessage);
    input.classList.add('erro');
  } else {
    input.classList.remove('erro');
  }
});
```

CSS para Formulários

```css
```

```css
/* Estilo base -->
input, textarea, select {
  font-family: Arial, sans-serif;
  font-size: 16px;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 4px;
  box-sizing: border-box;
}

/* Focus -->
input:focus, textarea:focus, select:focus {
  outline: none;
  border-color: #4CAF50;
  box-shadow: 0 0 5px rgba(76, 175, 80, 0.3);
}

/* Estados -->
input:disabled {
  background-color: #f5f5f5;
  cursor: not-allowed;
}

input:invalid {
  border-color: red;
}

input:valid {
  border-color: green;
}

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

/* Grupos de campo -->
fieldset {
  border: 1px solid #ddd;
  padding: 15px;
  border-radius: 4px;
  margin-bottom: 15px;
}
```

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

/* Botões -->
button, input[type="submit"] {
  background-color: #4CAF50;
  color: white;
  padding: 10px 20px;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  font-size: 16px;
}

button:hover {
  background-color: #45a049;
}

button:active {
  transform: scale(0.98);
}

/* Responsive -->
@media (max-width: 600px) {
  input, textarea, select {
    width: 100%;
  }
}
```

Exemplo Completo de Formulário

```
html
```

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Formulário de Cadastro</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f4;
      padding: 20px;
    }

    .container {
      max-width: 500px;
      margin: 0 auto;
      background-color: white;
      padding: 30px;
      border-radius: 8px;
      box-shadow: 0 2px 10px rgba(0,0,0,0.1);
    }

    h1 {
      margin-bottom: 20px;
      color: #333;
    }

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

    label {
      display: block;
      margin-bottom: 5px;
      color: #555;
      font-weight: bold;
    }

    input, textarea, select {
      width: 100%;
```

```
      padding: 10px;
      border: 1px solid #ddd;
      border-radius: 4px;
      font-size: 14px;
    }

    input:focus, textarea:focus, select:focus {
      outline: none;
      border-color: #4CAF50;
      box-shadow: 0 0 5px rgba(76, 175, 80, 0.3);
    }

    button {
      width: 100%;
      padding: 12px;
      background-color: #4CAF50;
      color: white;
      border: none;
      border-radius: 4px;
      cursor: pointer;
      font-size: 16px;
      font-weight: bold;
    }

    button:hover {
      background-color: #45a049;
    }

    .error {
      color: red;
      font-size: 12px;
      margin-top: 5px;
    }

    .success {
      background-color: #d4edda;
      color: #155724;
      padding: 15px;
      border-radius: 4px;
      margin-bottom: 20px;
      display: none;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="success" id="sucesso">Formulário enviado com sucesso!</div>
```

```
<h1>Cadastro de Usuário</h1>

<form id="meuFormulario">
  <div class="form-group">
    <label for="nome">Nome Completo:</label>
    <input type="text" id="nome" name="nome" required minlength="3">
    <div class="error" id="erro-nome"></div>
  </div>

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

  <div class="form-group">
    <label for="telefone">Telefone:</label>
    <input type="tel" id="telefone" name="telefone" placeholder="(11) 98765-4321
  </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" required>
      <option value="">Selecione...</option>
      <option value="m">Masculino</option>
      <option value="f">Feminino</option>
      <option value="o">Outro</option>
    </select>
  </div>

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

  <div class="form-group">
    <input type="checkbox" id="termos" name="termos" required>
    <label for="termos">Concordo com os termos de serviço</label>
  </div>

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

```
  </form>
</div>

<script>
  const form = document.getElementById('meuFormulario');
  const sucessoMsg = document.getElementById('sucesso');

  form.addEventListener('submit', (e) => {
    e.preventDefault();

    // Limpar mensagens de erro
    document.querySelectorAll('.error').forEach(el => el.textContent = '');

    const nome = document.getElementById('nome').value;
    const email = document.getElementById('email').value;
    const termos = document.getElementById('termos').checked;

    let valido = true;

    // Validar nome
    if (nome.trim().length < 3) {
      document.getElementById('erro-nome').textContent = 'Nome deve ter pelo menos
      valido = false;
    }

    // Validar email
    const regexEmail = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if (!regexEmail.test(email)) {
      document.getElementById('erro-email').textContent = 'Email inválido';
      valido = false;
    }

    // Validar termos
    if (!termos) {
      alert('Você deve concordar com os termos');
      valido = false;
    }

    if (valido) {
      sucessoMsg.style.display = 'block';
      form.reset();

      // Enviar dados para servidor (exemplo)
      const dados = new FormData(form);
      fetch('/enviar-formulario', {
        method: 'POST',
        body: dados
```

```
        })
        .then(resposta => resposta.json())
        .then(dados => console.log('Sucesso:', dados))
        .catch(erro => console.error('Erro:', erro));
    }
  });
  </script>
</body>
</html>
```

Resumo de Melhores Práticas

Para Iniciantes

1. **Comece com o básico:** HTML → CSS → JavaScript nessa ordem
2. **Pratique muito:** A programação se aprende fazendo
3. **Use ferramentas certas:** Editor de código (Visual Studio Code) e navegador com DevTools
4. **Leia documentação:** MDN Web Docs é excelente
5. **Estude projetos existentes:** Inspecione sites com DevTools
6. **Faça projetos pessoais:** Portfolio é importante para conseguir emprego
7. **Participe de comunidades:** Stack Overflow, GitHub, Discord de dev
8. **Mantenha-se atualizado:** Tecnologia evolui rápido

Roadmap Sugerido para Aprender

Mês 1-2: Fundamentos

- HTML semântico
- CSS responsivo
- JavaScript vanilla
- DOM e Event Listeners

Mês 3-4: Intermediário

- Git e GitHub
- Formulários web
- APIs REST

- Fetch API

Mês 5-6: Frontend Avançado

- Framework (React/Vue/Angular)
- Estado da aplicação
- Async/Await
- Testes

Mês 7-8: Backend Básico

- Node.js/Express ou Python/Django
- Banco de dados (PostgreSQL/MongoDB)
- Autenticação
- APIs próprias

Mês 9+: Full-Stack e Deployment

- Integração Frontend + Backend
- Docker
- CI/CD
- Servidores em nuvem

Recursos Essenciais

- **MDN Web Docs**: https://developer.mozilla.org (documentação oficial)
- **W3Schools**: Tutoriais interativos
- **FreeCodeCamp**: Cursos longos e estruturados
- **Codecademy**: Aprendizado interativo
- **GitHub**: Código de exemplo
- **Stack Overflow**: Respostas para problemas
- **CSS-Tricks**: Artigos sobre CSS e design
- **Dev.to**: Artigos da comunidade

Conclusão

Desenvolvimento web é uma carreira desafiadora e recompensadora. O mercado

está sempre em busca de profissionais qualificados, com salários competitivos e diversas oportunidades de trabalho (remoto, autônomo, agência, startup).

Os conceitos apresentados neste guia formam a base necessária para começar a jornada. Lembre-se que a prática é fundamental - quanto mais projetos você fizer, mais rápido evoluirá.

Dica final: Não tente aprender tudo de uma vez. Estabeleça um plano, seja consistente, e os resultados virão. Todo grande desenvolvedor começou exatamente onde você está agora.