Descomplicando a Programação - Iniciante

Este material foi desenvolvido para auxiliar iniciantes na jornada de aprendizado de programação. Aqui serão abordados desde os conceitos básicos até a criação de páginas web interativas, utilizando HTML, CSS e JavaScript. Independentemente de onde você esteja começando, este guia foi criado para tornar a programação acessível e descomplicada, ajudando você a construir uma base sólida para a sua carreira em tecnologia. Prepare-se para mergulhar neste mundo e dar os primeiros passos como desenvolvedor web!

por Júlia Bacchi

Dicas Iniciais

1 Resolução de Problemas

Uma habilidade crucial é saber resolver problemas de forma independente. Antes de pedir ajuda, tente entender o erro ou desafio por conta própria. Revise o código, experimente soluções diferentes, e use a documentação para buscar respostas. Isso ajuda a fortalecer sua capacidade de resolver problemas e te torna um programador mais confiante.

2 Recursos de Apoio

Aproveite os recursos valiosos de sites como o MDN (Mozilla Developer Network), W3Schools para encontrar conteúdo e informações sobre programação. Explore seus recursos para solucionar dúvidas e aprofundar seus conhecimentos. Mesmo que o conteúdo seja desenvolvido em inglês, todos possuem opção para visualizar em português, então não tenha medo!

3 Prática Constante

Não basta apenas assistir a conteúdos ou ler sobre programação, coloque em prática o que você aprende! Quando conhecer um novo conceito ou técnica, replique em um projeto próprio, mesmo que simples. A prática constante ajuda a fixar o conhecimento e a desenvolver suas habilidades.

4 Não Desanime

É normal sentir-se frustrado ao aprender a programar. Lembre-se de que todos passam por isso. Quando bater a frustração, faça uma pausa, respire fundo e busque tutoriais em vídeo ou fóruns online. Ver alguém explicar ou resolver um problema pode fazer toda a diferença.

Introdução às linguagens de Programação

Quando você começa a aprender programação, o primeiro passo geralmente é se familiarizar com as linguagens de marcação e estilo, HTML e CSS. Mas o que exatamente são essas linguagens?

HTML (HyperText Markup Language)

HTML é a linguagem de marcação usada para criar a estrutura de uma página web. Pense no HTML como o esqueleto de uma página: ele define o layout básico e os elementos que você vê em uma tela, como títulos, parágrafos, imagens, e links. Cada um desses elementos é representado por *tags* no código HTML, que organizam e estruturam o conteúdo.

CSS (Cascading Style Sheets)

CSS é a linguagem usada para definir a aparência visual das páginas web. Se o HTML é o esqueleto, o CSS é a pele e as roupas: ele permite estilizar os elementos criados com HTML, controlando cores, fontes, espaçamento, e o layout geral. Com CSS, você pode transformar uma página simples em algo visualmente atraente e responsivo, ajustando o design para diferentes dispositivos e tamanhos de tela.

Por onde Começar

Começar por HTML e CSS é uma ótima maneira de entender os fundamentos da web. Essas linguagens são relativamente fáceis de aprender, e seus efeitos são imediatamente visíveis no navegador, o que torna o aprendizado motivador. Com HTML, você cria a base; com CSS, você dá vida à página. Juntos, eles formam o alicerce para qualquer desenvolvedor web.

VS Code: Seu Ambiente de Desenvolvimento

O Visual Studio Code (VS Code) é um editor de código poderoso e amplamente utilizado por desenvolvedores ao redor do mundo. Ele é conhecido por sua versatilidade e facilidade de uso, tornando-o uma excelente escolha para iniciantes e profissionais. Nesta seção, você aprenderá a configurar e usar o VS Code para otimizar seu fluxo de trabalho e programar de forma eficaz.

Primeiros Passos

1

Download

Faça o download do VS Code em code.visualstudio.com da versão compatível com o seu sistema operacional e siga as instruções iniciais.

2

Criando o Projeto

No seu computador, crie uma nova pasta onde você armazenará seus arquivos de seu novo projeto.

No VS Code, em Arquivos, localize a pasta criada.

3

Como iniciar

O projeto deve ser aberto na íntegra no VS Code, pois arquivos individuais não fornecem a visão completa. Abrindo a pasta do projeto todos os arquivos e recursos serão acessíveis.

Entendendo o Ambiente

- Explorer: Na barra lateral esquerda, o Explorer exibe todos os arquivos e pastas do seu projeto. Isso permite que você navegue facilmente entre os arquivos.
- Editor: É a área principal de trabalho. Cada arquivo aberto aparecerá em uma aba no topo.
- Terminal integrado: O VS Code possui um terminal para executar comandos diretamente na interface do editor. Acesse-o em **Ver** > **Terminal**.

Criando os Arquivos

1

Primeiro arquivo

Para iniciar um documento HTML, crie um arquivo no Explorer e nomeie-o como **index.html**. Para a estrutura básica do HTML, digite **!** (ou **html:5**) e pressione **Enter**. As tags essenciais serão preenchidas.

2

Subpastas

Você pode incluir subpastas como **styles**, para incluir posteriormente arquivos CSS e JavaScript, e **images**, para organizar e incluir todas as imagens do seu projeto.

3

Visualização

Para abrir o arquivo HTML no Navegador, localize **index.html** na pasta do projeto, ou dê duplo clique sobre ele no VS Code. Agora está tudo pronto para iniciar a codificar e acompanhar o andamento!

Importante!

> A cada modificação realizada, salve! Esta prática regular evita a perda de progresso e é essencial para manter o seu trabalho organizado e eficiente. Após salvar, volte ao navegador e atualize a página: Isso garantirá que as mudanças feitas no código apareçam na web.

Editores de código Online

Não quer instalar um editor no seu computador?

Experimente ferramentas online como o **CodeSandbox**! Você poderá escrever, testar e compartilhar código diretamente no seu navegador. Em todas as etapas de seu caminho na programação, ferramentas como essa oferecem uma maneira prática e acessível de experimentar com projetos sem precisar configurar um ambiente local. É especialmente útil para protótipos rápidos e colaboração em tempo real com outros desenvolvedores.

Comandos e Atalhos

Confira algumas das funcionalidades incríveis no mundo dos códigos!

Salvar e Visualizar

- **Salvar Arquivo: Ctrl + S** (Windows/Linux) ou **Command + S** (macOS) para salvar o arquivo que você está editando. Isso garante que todas as alterações sejam gravadas no disco.
- **Atualizar Página:** Após salvar as mudanças no código, pressione **Ctrl + R** (Windows/Linux) ou **Command + R** (macOS) para atualizar a página no navegador e visualizar as atualizações.

Estrutura HTML

- **Estrutura Básica:** Digite **!** e pressione **Enter**, ou **html:5** e pressione **Enter**.

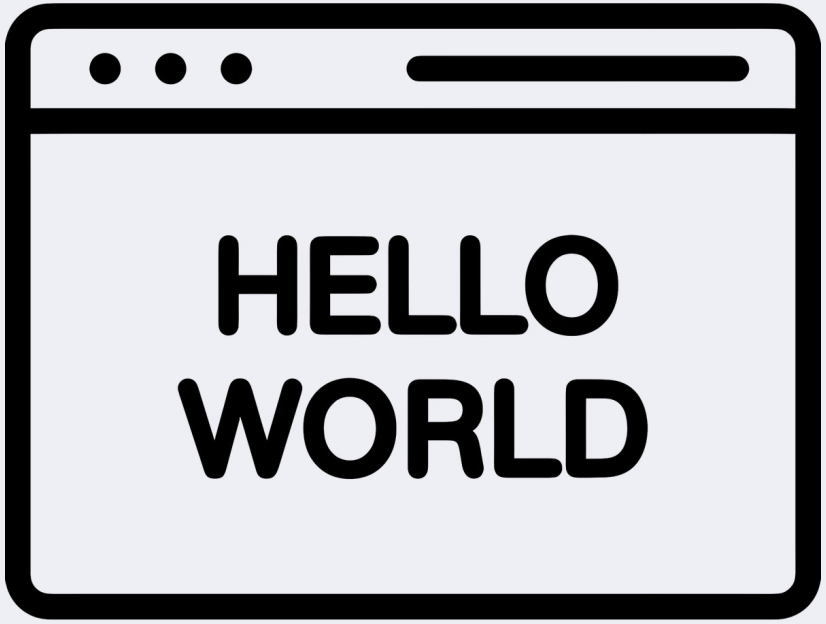
Navegação e Edição no VS Code

- **Pesquisa de Arquivos e Comandos:** Use Ctrl + P (Windows/Linux) ou **Command + P** (macOS) para abrir a barra de pesquisa rápida e encontrar arquivos ou executar comandos.
- **Pesquisa Global:** Para buscar uma palavra ou expressão em todo o projeto, use **Ctrl + Shift + F** (Windows/Linux) ou **Command + Shift + F** (macOS).
- **Trocar Entre Arquivos Abertos:** Use **Ctrl + Tab** (Windows/Linux) ou **Command + Tab** (macOS) para alternar entre os arquivos abertos no VS Code.

Ferramentas de Desenvolvimento

- **Inspecionar Elementos:** Use **Ctrl + Shift + I** (Windows/Linux) ou **Command + Option + I** (macOS) para abrir as ferramentas de desenvolvedor no navegador e inspecionar a estrutura HTML e CSS da página.

Esses atalhos são práticos e ajudam a melhorar a eficiência ao programar, especialmente para quem está começando. Integrá-los ao seu fluxo de trabalho pode fazer uma grande diferença na produtividade.



<h1>Hello, World!</h1>

Esta mensagem é tradicionalmente o primeiro passo em qualquer jornada de codificação. Ela representa o primeiro código funcional e marca o início de sua aventura na tecnologia.

Este é um exercício básico e importante, pois é uma maneira de testar seu ambiente de desenvolvimento, garantindo que tudo esteja configurado corretamente.

Além disso, é um simples lembrete de que grandes jornadas começam com pequenos passos.

HTML: Estruturando sua página

Compreender a estrutura do HTML é fundamental para qualquer desenvolvedor web. O HTML é a base de todas as páginas da web, definindo como o conteúdo é organizado e apresentado. Quando você entende a estrutura, consegue criar páginas mais organizadas, acessíveis e otimizadas para os motores de busca. Além disso, uma boa estrutura facilita a manutenção do código e a colaboração em projetos, garantindo que outros desenvolvedores possam entender e trabalhar no seu código com facilidade. Em resumo, dominar o HTML é o primeiro passo para criar websites funcionais e eficientes.

Estrutura Básica de um HTML

Código	Descrição
<!DOCTYPE html>	Declaração do tipo de documento. Informa ao navegador que o documento é HTML5.
<html>	Elemento raiz. Contém todo o conteúdo da página.
<head>	Cabeça do documento. Inclui informações sobre o documento, como o título e links para arquivos CSS.
<title>	Dentro da <head>. Define o título da página (aparece na aba do navegador).
<meta charset="UTF-8">	Dentro da <head>. Define a codificação de caracteres para o documento.
<body>	Corpo do documento. Contém o conteúdo a ser criado visível da página, como textos, imagens e links.
<h1>, <h2>, <h3>, etc.	Cabeçalhos de diferentes níveis.
<p>	Define um parágrafo de texto.
<a href="URL" target="_blank">	Cria um link externo para páginas externas. O target faz com que abra em outra aba.
Ir para a seção	Para navegar dentro da mesma página.
<img src="URL" alt="Descrição">	Dentro do <body>. Adiciona uma imagem.

Exemplo de Código

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Minha Página</title>
</head>
<body>
  <h1>Bem-vindo ao HTML!</h1>
  <p>Este é um parágrafo de texto.</p>
  <a href="https://www.exemplo.com" target="_blank">Visite nosso site</a>
  <img src="imagem.jpg" alt="Descrição da Imagem">
</body>
</html>
```

Dicas de formatação de texto

Itálico: *SheCodes Workshops*

Negrito: **SheCodes Workshop**s

Quebra de linha: Welcome to my website
 We hope you enjoy your stay

Separação entre seções de conteúdo: Welcome to my website<hr /> Discover latest updates below

Semântica de HTML

Significa usar elementos que descrevem claramente o propósito do conteúdo que eles envolvem. Empregados corretamente, os elementos indicam claramente a função das seções dentro de uma página. Usar tags semânticas melhora a acessibilidade, facilita o entendimento do código por outros desenvolvedores, e ajuda os motores de busca a indexarem melhor o conteúdo.

Elementos Semânticos

Quais são os básicos?

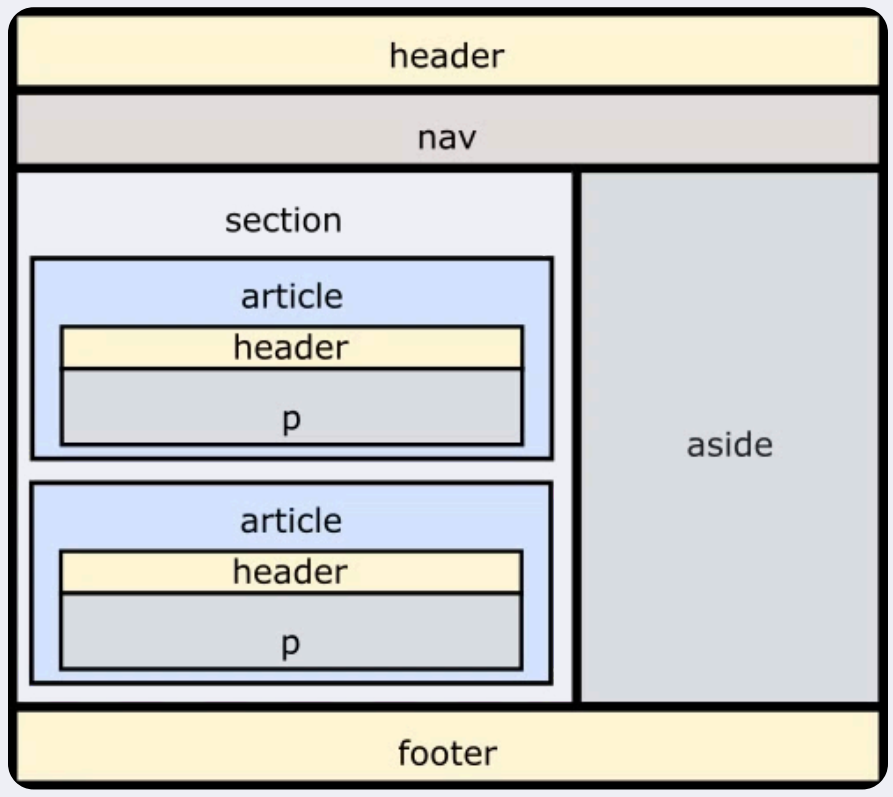
<header> para o cabeçalho de uma página.

<nav> para uma seção de links de navegação.

<section> para agrupar conteúdos relacionados.

<article> para um conteúdo independente, como uma postagem de blog.

<footer> para o rodapé da página.



▼ **<aside>**

> Este elemento pode não ser utilizado com tanta frequência, porém é parte da semântica e pode ser útil em contextos específicos, como destacar informações relacionadas ao conteúdo principal, barra lateral com links ou citações.

▼ **Veja aqui como distribuir esta estrutura em código**

```
<header>
  <h1>Meu Site</h1>
</header>
<nav>
  <ul>
   <li><a href="#home">Home</a></li>
   <li><a href="#about">Sobre</a></li>
  </ul>
</nav>
<main>
  <p>Conteúdo principal da página.</p>
</main>
<footer>
  <p>&copy; 2024 Meu Site</p> </footer>
```

Agrupamento com

A tag ** ** é uma das mais versáteis no HTML, usada principalmente para agrupar elementos e criar seções em uma página web. Embora por si só não adicione estilo ou comportamento, ela serve como um contêiner que pode ser estilizado com CSS e manipulado posteriormente.

Como utilizar

Agrupamento de Elementos: Agrupe blocos de conteúdo, como textos, imagens, e outros elementos.

Criação de Layouts: Crie layouts estruturados usando técnicas como flexbox e grid.

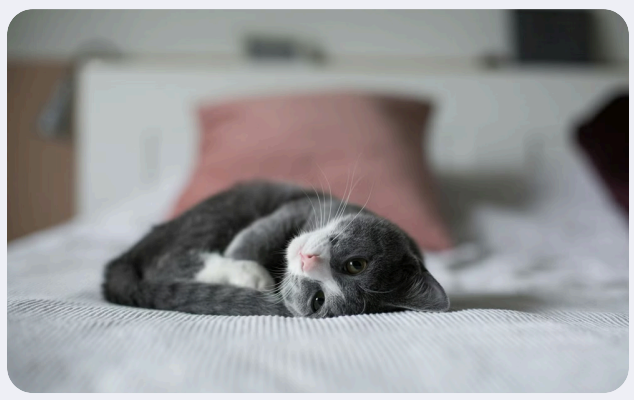
Aplicação de Estilos: Adicione classes ou IDs ao ** ** para aplicar estilos específicos.

> ### Dicas!
>
> **Organização**: Use ** ** para organizar seu código em seções lógicas, facilitando a manutenção e o entendimento.
>
> **Semântica**: Quando possível, use tags HTML5 mais semânticas, como **<header>**, **<section>** e **<footer>** para tornar o conteúdo mais acessível e melhorar o SEO (Search Engine Optimization/ Otimização pra motores de busca).

Incorporando Imagens

Imagens enriquecem o conteúdo web, tornando-o mais atraente e informativo. Utilize a tag <img> para inserir imagens em suas páginas.

Imagens não só enriquecem o conteúdo web, tornando-o mais atraente e informativo, mas também podem ser ajustadas e otimizadas para melhorar a experiência do usuário.



Entendendo os atributos

Para inserir uma imagem, utilize o atributo **src** *(source/fonte)* para especificar o caminho da imagem. O atributo **alt** *(alternative text/texto alternativo)* fornece uma descrição alternativa para a imagem, importante para acessibilidade.

▼ **Exemplo:**

```
<img
src="caminho/para/imagem.jpg
" alt="Descrição da Imagem">
```

Ajustando medidas

O tamanho da imagem pode ser controlado usando os atributos **width** e **height**, ajustando os valores conforme necessário. É considerada uma melhor prática separar o estilo (CSS) da estrutura (HTML), torna o código mais organizado, facilita a manutenção e permite ajustes globais com mais facilidade.

▼ **Exemplo:**

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

Elementos Visuais

Para imagens vetoriais, considere usar o formato SVG. É escalável e mantém a qualidade em qualquer tamanho. Ícones também são muito utilizados, eles ajudam a guiar o usuário de maneira intuitiva. Um exemplo comum é a biblioteca Font Awesome, com ícones de redes sociais, ferramentas, ações, e outros.

▼ **Exemplo:**

```
<i class="fab fa-instagram"></i>
```

Listas, Tabelas e Formulários

Listas

Listas Ordenadas (<ol>): Criam uma lista numerada.

Listas Não Ordenadas (<ul>): Criam uma lista com marcadores.

Itens de Lista (<li>): Definem os itens dentro de listas.

▼ **Exemplo:**

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

Tabelas

Elementos de Tabela (<table>, <tr>, <td>, <th>): Criam tabelas com linhas e colunas.

▼ **Exemplo:**

```
<table>
 <tr>
  <th>Cabeçalho 1</th>
  <th>Cabeçalho 2</th>
 </tr>
 <tr>
  <td>Celula 1</td>
  <td>Celula 2</td>
  </tr>
</table>
```

Formulários

Formulário (<form>): Contém elementos para coleta de dados.

Campos de Entrada (<input>, <textarea>, <select>): Usados para receber informações.

Botões (<button>, <input type="submit">): Para enviar o formulário.

▼ **Exemplo:**

```
<form action="/submit">
<label for="nome">Nome:
</label>

<input type="text"
id="nome" name="nome">

<input type="submit"
value="Enviar"> </form>
```

CSS: Estilizando sua Página

Como vimos anteriormente, CSS é a linguagem utilizada para estilizar elementos HTML. Agora que já entendemos os conceitos básicos da estrutura de uma página, explore as diversas propriedades do CSS para criar páginas web visuais e atraentes.

Como aplicar CSS em um documento HTML

As duas principais de aplicar estilos ao seu projeto são: **CSS Externo** e **CSS Interno**.

1. **Adicionar CSS Interno ao HTML**: Diretamente no HTML, utilize a tag **<style>** na seção **<head>**. Isso é útil para estilos que são específicos para uma única página e não precisam ser reutilizados em outras páginas.
2. **Adicionar CSS Externo:** O arquivo CSS precisa ser vinculado ao HTML. Isso é feito utilizando a tag **<link rel="stylesheet" href="style.css" />** dentro da seção **<head>** do seu documento HTML. O atributo **href** Define o caminho para o arquivo, então deverá ter o mesmo nome do seu arquivo CSS.

▼ **Nota!**

> Ambas as abordagens têm seus usos e benefícios, mas em geral, o CSS externo é a melhor prática devido à sua capacidade de manter o código mais limpo e organizado, facilitar a manutenção e melhorar o desempenho do seu projeto.

Seletores: Classes e IDs

Para aplicar estilos de forma eficiente, utilizamos seletores que ajudam a identificar quais elementos HTML devem receber determinados estilos. Dois dos seletores mais comuns são as **classes** e os **IDs**. Embora ambos sejam utilizados para aplicar estilos, eles têm propósitos e regras de uso distintas:

Classes

Permitem aplicar estilos a vários elementos HTML ao mesmo tempo. Elas são muito úteis quando desejamos estilizar vários elementos de maneira semelhante ou para aplicar um conjunto específico de estilos em diferentes partes da página.

- **Definição**: As classes são definidas no CSS com um ponto (**.**) seguido pelo nome da classe.
- **Uso em HTML**: Para usar uma classe em um elemento HTML, você deve atribuí-la ao atributo **class** desse elemento.

▼ **Exemplo:**

> No HTML:
>
> ```
> Este é um exemplo de como usar classes no CSS para estilizar diferentes elementos.
>
> Classes permitem aplicar estilos comuns a vários elementos.
> ```
>
> No CSS:
>
> ```
> .description {
>  color: #666;
>  font-size: 1.2em;
>  line-height: 1.6;
>  margin-bottom: 15px;
> }
> ```

IDs

São usados para identificar um único elemento na página. Eles são ideais quando precisamos aplicar estilos a um elemento específico que não se repete em nenhum outro lugar da página.

- **Definição**: IDs são definidos no CSS com um cerquilha (**#**) seguido pelo nome do ID.
- **Uso em HTML**: Para usar um ID, você deve atribuí-lo ao atributo **id** do elemento HTML.

▼ **Exemplo:**

> No HTML:
>
> ```
> <header id="header">
>  Bem-vindo ao meu Website
> </header>
> ```
>
> No CSS:
>
> ```
> #header {
>  background-color: blue;
>  color: white;
>  padding: 20px;
>  text-align: center;
> }
> ```

Personalizando Textos e Cores

Opções de Texto

Personalize o texto com propriedades como alinhamento, tamanho, peso, família da fonte e decoração para melhorar a legibilidade e o estilo da sua página.

▼ **Exemplo:**

text-align: left | right | center | justify | initial | inherit;

Font-size: tamanho da fonte (px)

Font-weight: largura da fonte (números 100, 200 ou normal | bold | bolder | lighter)

Font-family: Georgia, serif;

text-decoration: underline;

Variedades de Cores

Explore diferentes maneiras de definir cores. Você pode aplicar cores a vários elementos, como o fundo, o texto, ícones e até imagens.

▼ **Exemplo:**

Cor de Fundo:

- Cor sólida: `background: green;`
- Gradiente: `background: linear-gradient(#e66465, #9198e5);`

Cor da Fonte:

- Cor nativa: `color: blueviolet;`
- Hexadecimal: `color: #00ff00;`

Cor RGB:

- Sem transparência: `color: rgb(135, 93, 241);`
- Com transparência (RGBA): `color: rgba(135, 93, 241, 0.5);`

```
<table>
 <tr>
  <th>Cabeçalho 1</th>
  <th>Cabeçalho 2</th>
 </tr>
 <tr>
  <td>Celula 1</td>
  <td>Celula 2</td>
  </tr>
</table>
```

Fundamentos de CSS

Dentre os fundamentos essenciais do CSS, estão o layout e posicionamento de elementos, técnicas de espaçamento, controle de dimensões e manipulação de imagens e backgrounds. Esses conceitos são importantes para criar designs responsivos e organizados, oferecendo uma base sólida para quem está começando no desenvolvimento web.

Layout e Posicionamento

Flexbox

Uma abordagem simplificada para criar layouts que se ajustam ao espaço disponível. É ótimo para criar layouts de uma única dimensão (linha ou coluna). Usando propriedades como **flex-direction, justify-content,** e **align-items**, você pode controlar a distribuição e o alinhamento dos elementos.

Grid

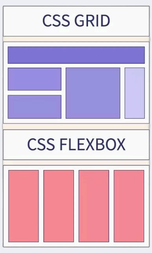
Um sistema bidimensional que permite criar layouts mais complexos. Ele usa linhas e colunas para organizar os elementos, permitindo controles precisos sobre onde os elementos são posicionados. As propriedades principais incluem **grid-template-columns, grid-template-rows,** e **grid-area**.

Position

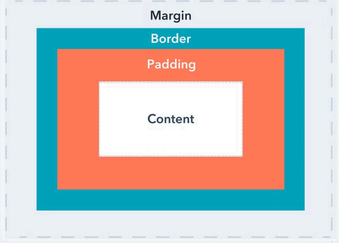
Posição dos elementos. **static, relative, absolute, fixed** e **sticky**.

Display

Exibição dos elementos. **block, inline, inline-block, none**, etc.



Espaçamento



Margin

Espaço externo ao redor de um elemento, cria distância entre elementos adjacentes.

Border

Adiciona bordas ao redor de um elemento. As propriedades incluem **border-width, border-style,** e **border-color**.

Border-Radius

Arredonda os cantos das bordas, permitindo criar elementos com cantos suavizados ou circulares.

Padding

Espaço interno ao redor do conteúdo de um elemento, separando-o das bordas do próprio elemento.

Dimensões

Width

Width: Define a largura de um elemento.

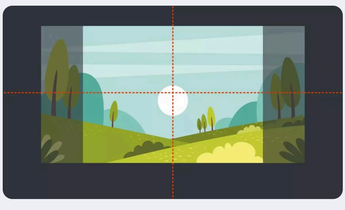
Max-width e Min-width (Largura Máxima e Mínima): Limitam o tamanho máximo ou mínimo que um elemento pode ter.

Height

Height: Define a altura de um elemento.

Max-height e Min-height (Altura Máxima e Mínima): Limitam o tamanho máximo ou mínimo em altura.

Backgrounds e Objetos



Background Image

Permite adicionar uma imagem de fundo a qualquer elemento. A propriedade **background-image** define o caminho da imagem a ser usada.

Background properties

Background-Repeat (Repetição do Fundo): Controla se e como a imagem de fundo é repetida (**repeat, no-repeat, repeat-x, repeat-y**).

Background-Position (Posição do Fundo): Define a posição inicial da imagem de fundo dentro do elemento.

Background-Size (Tamanho do Fundo): Controla o tamanho da imagem de fundo (**cover, contain**, valores específicos em unidades de medida).

Background-Attachment (Anexo do Fundo): Define se o fundo deve rolar com o conteúdo (**scroll**) ou permanecer fixo (**fixed**).

Object-fit e Object-Position

Object-Fit (Ajuste do Objeto): Ajusta como uma imagem ou vídeo deve ser encaixado dentro do contêiner (**fill, contain, cover, none, scale-down**).

Object-Position (Posição do Objeto): Define a posição da imagem ou vídeo dentro do contêiner, especialmente quando não está sendo totalmente preenchido.

JavaScript: Interatividade e Dinamismo

JavaScript é a linguagem de programação que adiciona interatividade e dinamismo às páginas web. Compreender os conceitos básicos permite criar experiências web mais envolventes e interativas.

Introdução ao JavaScript

Considerada uma das principais tecnologias da web, essa linguagem é utilizada em conjunto com HTML e CSS para criar páginas dinâmicas, como animações, respostas a eventos do usuário, validação de formulários e muito mais.

Como incluir o JavaScript em uma página HTML

Para adicionar JavaScript a uma página web, você precisa incluí-lo no seu arquivo HTML. Isso pode ser feito de duas maneiras principais:

1. **JavaScript Interno:** Diretamente dentro do arquivo HTML usando a tag **<script>**. Esse código é executado quando a página é carregada.
2. **JavaScript Externo:** Em um arquivo separado com a extensão **.js**, normalmente dentro na subpasta source. Inclua-o em sua página HTML usando a tag **<script src="script.js"></script>** dentro de **<body>**, após todo o conteúdo inserido.

Como testar?

No arquivo **script.js**, escreva console.log('Hello, World!');

▼ **Exemplo:**

No HTML

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Exemplo de JavaScript Externo</title>
</head>
<body>
    <h1>Olá, mundo!</h1>
    <script src="script.js"></script> <!-- Inclui o arquivo JavaScript externo -->
</body>
</html>
```

No arquivo script.js

```
console.log('JavaScript externo está funcionando!');
```

Testando diretamente no Navegador

Você pode abrir o console através de **F12** ou **Ctrl+Shift+I** (Windows/Linux) ou **Cmd+Option+I** (Mac), e depois selecionando a aba "Console". Aqui, você pode digitar código JavaScript e ver os resultados instantaneamente. É uma ótima forma de experimentar pequenos trechos de código.

Conceitos básicos de JavaScript

1

Variáveis

Variáveis armazenam dados em JavaScript e podem conter diferentes tipos de dados, como numbers, strings, booleans, objects e arrays.

2

Operadores

Operadores são símbolos que realizam operações em variáveis e valores, como aritméticos, de comparação e lógicos.

3

Funções

São blocos de código reutilizáveis que executam uma tarefa específica. Elas ajudam a organizar o código e a evitar repetições.

4

Estruturas Condicionais

As estruturas condicionais permitem executar código com base em condições. **If / Else:** Executa um bloco de código se a condição for verdadeira ou falsa.

5

Loops

Permitem repetir uma operação várias vezes. **For:** Repete um bloco de código um número específico de vezes. **While:** Repete um bloco de código enquanto uma condição for verdadeira.

6

Seletores

Seletores são usados para selecionar elementos HTML em JavaScript e manipulá-los. Por exemplo, para alterar o texto de um elemento.

7

Eventos

São ações que ocorrem em uma página web, como clicar em um botão ou mover o mouse. JavaScript permite responder a esses eventos e criar interatividade.

8

Console.log

Essa função exibe uma mensagem no console do navegador. É uma ferramenta útil para depuração, permitindo verificar valores de variáveis e o fluxo do código.

Aplicando os Conceitos

Nesta seção, vamos explorar os conceitos fundamentais que darão vida aos seus sites e aplicativos, entendendo como funciona essa linguagem e como utilizá-la no seu projeto.

Variáveis e tipos de Dados

Variáveis

Armazenam valores que podem ser usados e modificados ao longo do código. Há três palavras-chave para declarar:

var: Declara uma variável com escopo global ou local, mas pode causar confusão com o escopo.

let: Declara uma variável com escopo de bloco, mais seguro e recomendável.

const: Declara uma variável que não pode ser reatribuída, ou seja, seu valor não muda.

Tipos de Dados

Número: Valores numéricos, como **42** ou **3.14**.

String: Representa textos, como **"Hello, world!"**.

Boolean: Representa verdadeiro ou falso, como **true** ou **false**.

let age = 30; // Número
let name = "Ana"; // String
let isStudent = true; // Boolean

Operadores

Aritméticos

let a = 5 + 3; // 8 let b = 10 % 3; // 1

Atribuição

let x = 10; x += 5; // x é agora 15

Comparação

let isEqual = (5 === 5); // true

let isNotEqual = (5 !== 3); // true

Lógicos

let a = true && false; // false

let b = true || false; // true

Funções: Definição e como Aplicar

Funções são blocos de código projetados para executar tarefas específicas e podem ser reutilizadas em diferentes partes do seu programa. Elas ajudam a organizar o código e evitar repetição.

```
function greet(name) {
return Hello, ${name}!;
}
console.log(greet("Alice")); // Hello, Alice!
```

Importante saber

Funções podem aceitar parâmetros, que são valores fornecidos quando a função é chamada, e retornar resultados, que são valores que a função produz após a execução.

▼ **Veja aqui!**

> ```
> function add(a, b) {
>  return a + b;
> }
> console.log(add(5, 3)); // 8
> ```

Mais funcionalidades do JavaScript

Você aprenderá a usar estruturas de controle para criar lógicas dinâmicas, a manipular o DOM (Document Object Model) para transformar e interagir com o conteúdo da sua página, e a trabalhar com eventos para tornar suas aplicações interativas e responsivas. Vamos começar a transformar suas ideias em realidade com o poder do JavaScript!

Estruturas de Controle

Condicionais

if: Executa um bloco de código se a condição for verdadeira.

else: Executa um bloco de código se a condição do **if** for falsa.

else if: Adiciona condições adicionais.

switch: Substitui múltiplos **if** e **else** para comparação de valores.

▼ **Exemplo:**

```
let score = 85;

if (score > 90) {
  console.log("Excelente");
} else if (score > 75) {
  console.log("Bom");
} else {
  console.log("Precisa melhorar");
}
```

Loops

for: Itera sobre um bloco de código um número fixo de vezes.

while: Itera enquanto a condição for verdadeira.

do...while: Itera pelo menos uma vez, depois continua enquanto a condição for verdadeira.

▼ **Exemplo:**

```
for (let i = 0; i < 5; i++) {
  console.log(i); // 0, 1, 2, 3, 4
}
let count = 0;
while (count < 5) {
  console.log(count); // 0, 1, 2, 3, 4
  count++;
}
```

Manipulação do DOM

O que é o DOM?

O Document Object Model (DOM) é uma interface que representa a estrutura do documento HTML ou XML como uma árvore de objetos. Cada elemento HTML (como ** **, **<h1>**, **<p>**, etc.) é representado como um nó na árvore do DOM. Isso permite que você, através do JavaScript, acesse, manipule e modifique o conteúdo e a estrutura da página web dinamicamente.

Selecionar e Modificar Elementos

Exemplo:

```
let header = document.querySelector("h1");
header.textContent = "Novo Título";
```

Explicação:

1. **document.querySelector("h1")**: Esse método é utilizado para selecionar o primeiro elemento **<h1>** que aparece no documento. O **document** representa toda a página, e o **querySelector** permite selecionar elementos usando seletores CSS.
2. **header.textContent = "Novo Título";**: Aqui estamos alterando o conteúdo de texto do elemento **<h1>** que foi selecionado. O texto dentro desse elemento é substituído por "Novo Título". Ou seja, se antes o título fosse "Bem-vindo", ele agora será "Novo Título".

Eventos

Escutar e responder a Eventos

Em JavaScript, você pode configurar o código para "escutar" eventos, como cliques, teclas pressionadas, movimentos do mouse, entre outros. Quando o evento ocorre, uma função é executada automaticamente em resposta.

Exemplo:

```
document.querySelector("#button").addEventListener("click", function() { alert("Botão clicado!");
});
```

Explicação:

1. **document.querySelector("#button")**: Aqui, estamos selecionando um elemento com o ID **"button"**. O símbolo **#** no seletor indica que estamos buscando um ID específico.
2. **.addEventListener("click", function() {...})**: Esse método **addEventListener** é usado para "escutar" um tipo específico de evento em um elemento. No caso, estamos escutando o evento **"click"** (quando o botão é clicado).
3. **function() { alert("Botão clicado!"); }**: Essa é a função que será executada quando o evento de clique ocorrer. No exemplo, mostramos um alerta com a mensagem "Botão clicado!". Você pode substituir essa função por qualquer ação que deseja que ocorra ao clicar no botão.

"Esses são exemplos básicos, mas muito poderosos, que abrirão as portas para você criar e controlar o comportamento das suas páginas web!"

link desconhecido

Parabéns por chegar até aqui!

Agora que você dominou os fundamentos de HTML, CSS e JavaScript, está pronto para dar os próximos passos e se aventurar em conceitos mais avançados que vão elevar suas habilidades de desenvolvimento.

Download

O que vem pela frente...

O próximo módulo vai te ajudar a criar projetos ainda mais complexos e interativos, que podem abrir novas oportunidades na sua carreira.

Descomplicando a Programação - Intermediário

Um pouco sobre o que você vai Aprender!

- Aprofundamento em HTML, CSS e JavaScript
- Conheça novos elementos e técnicas para melhorar a acessibilidade do seu site
- Explore as funcionalidades de Flexbox, Grid, animações e como criar layouts responsivos que funcionam perfeitamente em qualquer dispositivo
- Descubra como manipular APIs, usar frameworks, e criar experiências de usuário mais interativas e envolventes
- Crie do zero um projeto real para desenvolver sua prática!

Continue praticando!

Quero continuar!

Se você gostou do conteúdo e te ajudou a descomplicar a programação, acompanhe os próximos módulos e faça parte desse projeto!