Conteúdo do dia!

Site: Geração Tech
Curso: Formação em Desenvolvedor Web - Online
Livro: Conteúdo do dia!

Impresso por: JOÃO VITOR DE MELO FREITAS
Data: quinta-feira, 18 jul. 2024, 22:34

Índice

1. JavaScript

As páginas web são desenvolvidas usando três tecnologias padrão: HTML, CSS e JavaScript. O JavaScript é uma linguagem de programação que permite ao navegador atualizar dinamicamente o conteúdo do site.

Ele normalmente é executado pelo mesmo navegador usado para visualizar uma página web. Dessa forma, como ocorre com o CSS e o HTML, o comportamento exato de um código pode ser diferente de acordo com o navegador. Mas os navegadores mais comuns aderem à especificação ECMAScript.

Este é um padrão que unifica o uso do JavaScript na web e será a base desta lição, junto com a especificação HTML5, que especifica como o JavaScript precisa ser posto em uma página web para que um navegador possa executá-lo.

1.1. Guia Prático: Seu Primeiro "Hello World" com HTML e JavaScript no VSCode

Introdução

Neste guia, você aprenderá a criar seu primeiro "Hello World" usando HTML e JavaScript. Utilizaremos o Visual Studio Code (VSCode), um editor de código poderoso e amplamente utilizado por desenvolvedores.

Pré-requisitos

- **Visual Studio Code (VSCode)** instalado no seu computador.
- Noções básicas de HTML e JavaScript (mas não se preocupe, vamos cobrir o essencial).

Passo 1: Configurando o Ambiente

1. **Instale o Visual Studio Code**: Se ainda não tem o VSCode instalado, baixe e instale a partir do site oficial: Visual Studio Code.

2. **Crie uma Pasta de Projeto**: Crie uma nova pasta em seu computador onde você guardará seus arquivos de projeto. Nomeie a pasta como "HelloWorld".

Passo 2: Criando o Arquivo HTML

1. **Abra o VSCode** e a pasta "HelloWorld" que você criou:
   - Vá para **File > Open Folder...**
   - Selecione a pasta "HelloWorld" e clique em **Open**.
2. **Crie um Arquivo HTML**:
   - No VSCode, clique no ícone de **Novo Arquivo** (ou use o atalho `Ctrl + N`).
   - Salve o arquivo como `index.html` dentro da pasta "HelloWorld".
3. **Adicione o Código HTML Básico** ao arquivo `index.html`:

```
<!DOCTYPE html> <html lang="pt-br"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Meu Primeiro Hello World</title> </head> <body> <h1 id="mensagem">Hello, World!</h1> <script src="script.js"></script> </body> </html>
```

Passo 3: Criando o Arquivo JavaScript

1. **Crie um Arquivo JavaScript**:
   - No VSCode, clique no ícone de **Novo Arquivo**.
   - Salve o arquivo como `script.js` dentro da pasta "HelloWorld".
2. **Adicione o Código JavaScript** ao arquivo `script.js`:

```
// script.js // Seleciona o elemento HTML com o id 'mensagem' const mensagemElemento = document.getElementById('mensagem'); // Modifica o texto do elemento mensagemElemento.textContent = 'Olá, Mundo! Este é meu primeiro Hello World com JavaScript!';
```

Passo 4: Executando o Projeto

1. **Abra o Arquivo HTML no Navegador**:
   - No VSCode, clique com o botão direito no arquivo `index.html` e selecione **Open with Live Server**. Se você não tiver a extensão Live Server instalada, você pode instalá-la via Marketplace de Extensões do VSCode.
   - Alternativamente, você pode abrir o arquivo `index.html` diretamente no seu navegador (clicando duas vezes no arquivo ou arrastando-o para uma nova aba do navegador).
2. **Veja o Resultado**:
   - Você deve ver a mensagem "Olá, Mundo! Este é meu primeiro Hello World com JavaScript!" na página web aberta no seu navegador.

Dicas e Conclusão

- **Personalização**: Tente modificar o conteúdo do arquivo `script.js` para alterar a mensagem ou adicionar novos elementos HTML e manipulá-los com JavaScript.
- **Exploração**: Explore mais recursos do VSCode, como o terminal integrado, extensões para facilitar o desenvolvimento e ferramentas de depuração.

Parabéns! Você criou e executou seu primeiro "Hello World" usando HTML e JavaScript no VSCode. Continue explorando e aprendendo mais sobre desenvolvimento web para criar projetos ainda mais incríveis.

1.2. Aula 01

dia 10 javascript vd 1

1.3. Lógica de Programação

1. Lógica de Programação

Lógica de programação é a base para o desenvolvimento de qualquer sistema computacional. Ela envolve o uso de um conjunto de regras e técnicas para resolver problemas através de algoritmos.

2. Algoritmos

Algoritmos são sequências finitas de instruções bem definidas e ordenadas que visam resolver um problema específico ou executar uma tarefa.

2.1 Raciocínio Lógico

O raciocínio lógico é essencial para a criação de algoritmos eficazes. Ele envolve a capacidade de analisar problemas, identificar soluções possíveis e implementá-las de forma sistemática.

3. Variáveis

Variáveis são utilizadas para armazenar dados temporariamente na memória do computador. Em JavaScript, uma variável pode ser declarada usando `var`, `let` ou `const`.

```
let nome = "João"; // Variável que pode mudar const idade = 25; // Constante, valor imutável
```

4. Constantes

Constantes são similares às variáveis, mas seus valores não podem ser alterados após a atribuição inicial. Em JavaScript, usamos `const` para declarar constantes.

```
const pi = 3.14159;
```

5. Funções

Funções são blocos de código reutilizáveis que executam uma tarefa específica. Elas podem receber parâmetros e retornar valores.

```
function saudacao(nome) { return `Olá, ${nome}!`; } console.log(saudacao("Maria")); // Saída: Olá, Maria!
```

6. Estruturas Condicionais

Estruturas condicionais permitem que você execute diferentes blocos de código com base em determinadas condições.

6.1 if/else

A estrutura `if/else` é a mais comum para realizar verificações condicionais.

```
let idade = 18; if (idade >= 18) { console.log("Você é maior de idade."); } else { console.log("Você é menor de idade."); }
```

6.2 switch

A estrutura `switch` é utilizada para realizar múltiplas verificações de igualdade de maneira mais organizada.

```
let dia = 3; switch (dia) { case 1: console.log("Domingo"); break; case 2: console.log("Segunda-feira"); break; case 3:
console.log("Terça-feira"); break; default: console.log("Dia inválido"); }
```

7. Estruturas de Repetição

Estruturas de repetição permitem executar um bloco de código múltiplas vezes.

7.1 for

A estrutura `for` é utilizada para repetir um bloco de código um número específico de vezes.

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

7.2 while

A estrutura `while` repete um bloco de código enquanto uma condição for verdadeira.

```
let contador = 0; while (contador < 5) { console.log(contador); contador++; }
```

8. Requisições

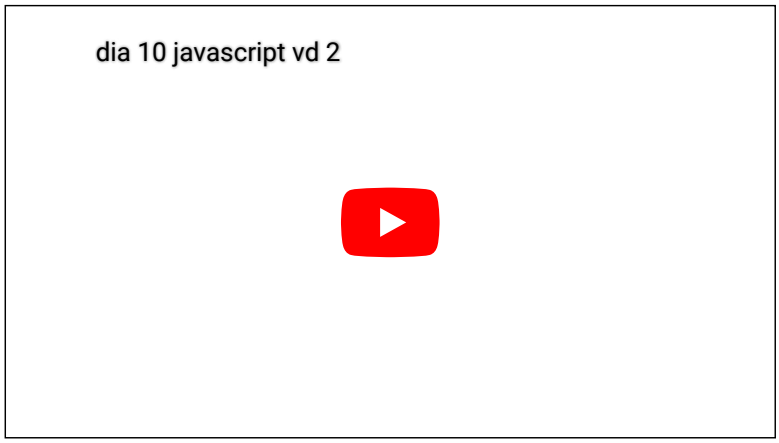
Requisições são utilizadas para comunicar-se com servidores web e obter ou enviar dados. Em JavaScript, podemos fazer requisições usando `fetch`.

```
fetch('https://api.exemplo.com/dados') .then(response => response.json()) .then(data => console.log(data)) .catch(error =>
console.error('Erro:', error));
```

Conclusão

Este guia apresenta os conceitos básicos de lógica de programação e sua aplicação em JavaScript, abordando algoritmos, variáveis, constantes, funções, estruturas condicionais, estruturas de repetição e requisições. Compreender e dominar esses conceitos é fundamental para o desenvolvimento de aplicações eficientes e funcionais.

1.4. Aula 02



1.5. Variáveis e Constantes em JavaScript

Introdução

Variáveis e constantes são fundamentais na programação. Elas permitem armazenar e manipular dados de maneira eficiente. Em JavaScript, as variáveis podem ser declaradas usando `var`, `let` ou `const`, cada uma com suas próprias características e usos específicos.

Variáveis

Declaração e Inicialização

Em JavaScript, uma variável pode ser declarada usando `var`, `let` ou `const`. A escolha da palavra-chave afeta o escopo e a mutabilidade da variável.

- **var**: Utilizada desde as primeiras versões de JavaScript. Tem escopo de função ou global, o que pode levar a problemas como hoisting.

```
var nome = "João";
```

- **let**: Introduzida no ECMAScript 6 (ES6). Tem escopo de bloco, o que torna seu comportamento mais previsível e seguro.

```
let idade = 25;
```

Reatribuição de Valores

As variáveis declaradas com `let` podem ser reatribuídas.

```
let cidade = "São Paulo"; cidade = "Rio de Janeiro"; // Valor atualizado
```

Constantes

Declaração e Inicialização

Constantes são declaradas usando a palavra-chave `const` e devem ser inicializadas no momento da declaração. Elas não podem ser reatribuídas depois de definidas.

```
const pi = 3.14159;
```

Imutabilidade

Embora o valor de uma constante não possa ser alterado, se a constante for um objeto ou array, suas propriedades ou elementos podem ser modificados.

```
const carro = { marca: "Toyota", modelo: "Corolla" }; carro.modelo = "Camry"; // Permitido
```

Sintaxe do JavaScript

A sintaxe do JavaScript define como os programas devem ser escritos e interpretados. Aqui estão alguns elementos essenciais da sintaxe do JavaScript:

Declaração de Variáveis

- `var` para declarações de variáveis de escopo global ou de função.
- `let` para declarações de variáveis de escopo de bloco.

- `const` para declarações de constantes.

Estruturas de Controle

- Condicionais: `if`, `else if`, `else`
- Laços de repetição: `for`, `while`, `do while`

```
if (condicao) { // bloco de código } else { // bloco de código } for (let i = 0; i < 5; i++) { // bloco de código }
```

Funções

Declaração e expressão de funções.

```
function saudacao(nome) { return `Olá, ${nome}!`; } const saudacao = function(nome) { return `Olá, ${nome}!`; };
```

Tipos de Dados do JavaScript

JavaScript possui vários tipos de dados primitivos e de objetos.

Tipos Primitivos

1. **String**: Cadeia de caracteres.

```
let texto = "Hello, World!";
```

2. **Number**: Números inteiros e de ponto flutuante.

```
let numeroInteiro = 42; let numeroDecimal = 3.14;
```

3. **Boolean**: Valores lógicos (`true` ou `false`).

```
let verdadeiro = true; let falso = false;
```

4. **Undefined**: Uma variável declarada que ainda não foi inicializada.

```
let indefinido; console.log(indefinido); // undefined
```

5. **Null**: Representa a ausência intencional de um valor.

```
let vazio = null;
```

6. **Symbol**: Um valor único e imutável, usado como identificador.

```
let simbolo = Symbol("descricao");
```

Tipos de Objetos

1. **Object**: Coleção de propriedades.

```
let pessoa = { nome: "Ana", idade: 30 };
```

2. **Array**: Lista ordenada de valores.

```
let frutas = ["maçã", "banana", "laranja"];
```

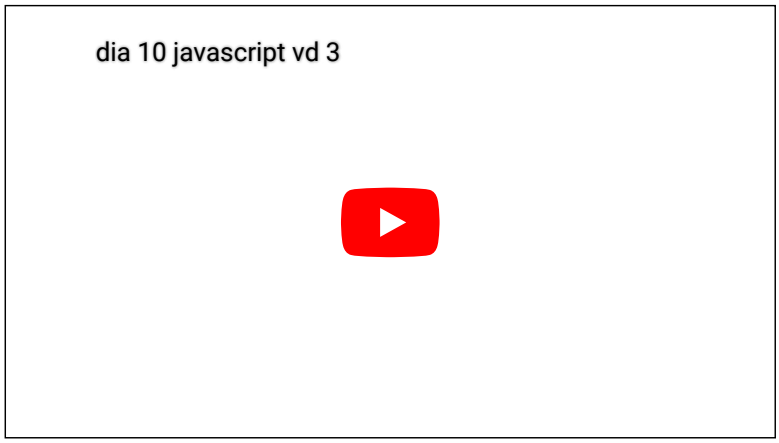
3. **Function**: Bloco de código reutilizável.

```
function soma(a, b) { return a + b; }
```

Conclusão

Entender variáveis, constantes e os tipos de dados em JavaScript é essencial para qualquer desenvolvedor. Variáveis permitem armazenar dados que podem mudar, enquanto constantes armazenam dados imutáveis. A sintaxe do JavaScript é flexível e poderosa, permitindo a criação de código eficiente e de fácil manutenção. Com o domínio desses conceitos, você estará bem equipado para enfrentar desafios de programação e desenvolver aplicações robustas.

1.6. Aula 03



1.7. Lógica de Programação com Operações Aritméticas e Concatenação em JavaScript

Introdução

Neste guia, vamos explorar a lógica de programação utilizando operações aritméticas entre números e operações entre números e strings em JavaScript. Vamos entender como a linguagem trata esses diferentes tipos de operações e veremos exemplos práticos de concatenação.

Operações Aritméticas

Operações aritméticas são operações matemáticas básicas que podemos realizar em números. Em JavaScript, as principais operações aritméticas são:

1. **Adição (`+`)**
2. **Subtração (`-`)**
3. **Multiplicação (`*`)**
4. **Divisão (`/`)**
5. **Módulo (`%`)**: Resto da divisão entre dois números
6. **Incremento (`++`)**: Aumenta o valor da variável em 1
7. **Decremento (`--`)**: Diminui o valor da variável em 1

Exemplos de Operações Aritméticas

javascript

Copiar código

```
let a = 10; let b = 5; let soma = a + b; // 15 let subtracao = a - b; // 5 let multiplicacao = a * b; // 50 let divisao = a / b; // 2 let modulo = a % b; // 0 console.log(`Soma: ${soma}`); console.log(`Subtração: ${subtracao}`); console.log(`Multiplicação: ${multiplicacao}`); console.log(`Divisão: ${divisao}`); console.log(`Módulo: ${modulo}`);
```

Operações entre Números e Strings

Quando realizamos operações entre números e strings em JavaScript, o comportamento pode variar dependendo da operação:

1. **Adição (`+`)**: Se um dos operandos for uma string, JavaScript realiza a concatenação.
2. **Outras operações (`-`, `*`, `/`, `%`)**: JavaScript tenta converter a string em um número e, em seguida, realiza a operação.

Exemplos de Operações entre Números e Strings

javascript

Copiar código

```
let numero = 10; let texto = "5"; let resultado1 = numero + texto; // "105" (concatenação) let resultado2 = numero - texto; // 5 (subtração após conversão) let resultado3 = numero * texto; // 50 (multiplicação após conversão) let resultado4 = numero / texto; // 2 (divisão após conversão) console.log(`Adição (concatenação): ${resultado1}`); console.log(`Subtração: ${resultado2}`); console.log(`Multiplicação: ${resultado3}`); console.log(`Divisão: ${resultado4}`);
```

Concatenação

Concatenação é o processo de unir duas ou mais strings. Em JavaScript, usamos o operador `+` para concatenar strings.

Exemplo de Concatenação

javascript

Copiar código

```
let saudacao = "Olá, "; let nome = "Maria"; let mensagem = saudacao + nome + "! Seja bem-vinda."; console.log(mensagem); // "Olá, Maria! Seja bem-vinda."
```

Podemos também usar a template string (template literals) para uma concatenação mais prática:

javascript
Copiar código

```
let mensagem2 = `${saudacao}${nome}! Seja bem-vinda.`; console.log(mensagem2); // "Olá, Maria! Seja bem-vinda."
```

Exemplo Completo

Vamos criar um exemplo completo que envolve operações aritméticas, operações entre números e strings, e concatenação:

javascript
Copiar código

```
// Declaração de variáveis let numero1 = 8; let numero2 = 3; let texto1 = "4"; let saudacao = "Olá"; let nome = "João"; // Operações aritméticas let soma = numero1 + numero2; let diferenca = numero1 - numero2; let produto = numero1 * numero2; let quociente = numero1 / numero2; // Operações entre números e strings let somaMisturada = numero1 + texto1; // "84" (concatenação) let subtracaoMisturada = numero1 - texto1; // 4 (subtração após conversão) // Concatenação de strings let mensagem = saudacao + ", " + nome + "!"; let mensagem2 = `${saudacao}, ${nome}!`; // Saída dos resultados console.log(`Soma: ${soma}`); // 11 console.log(`Diferença: ${diferenca}`); // 5 console.log(`Produto: ${produto}`); // 24 console.log(`Quociente: ${quociente}`); // 2.6666... console.log(`Soma Misturada: ${somaMisturada}`); // "84" console.log(`Subtração Misturada: ${subtracaoMisturada}`); // 4 console.log(`Mensagem: ${mensagem}`); // "Olá, João!" console.log(`Mensagem2: ${mensagem2}`); // "Olá, João!"
```

Conclusão

Neste guia, exploramos a lógica de programação em JavaScript, focando em operações aritméticas, operações entre números e strings, e concatenação. Entender como JavaScript lida com esses tipos de operações é essencial para escrever código eficiente e evitar erros comuns. Com os exemplos fornecidos, você deve estar bem equipado para aplicar esses conceitos em seus próprios projetos.

1.8. Aula 04

dia 10 javascript vd 4

1.9. Estrutura Sequencial em JavaScript

Introdução

A estrutura sequencial é a mais básica das estruturas de controle de um programa. Em uma estrutura sequencial, as instruções são executadas uma após a outra, na ordem em que aparecem.

Exemplo de Prompt e `console.log`

Usando `prompt` para Entrada de Dados

O método `prompt` é utilizado para solicitar ao usuário que insira dados, exibindo uma caixa de diálogo no navegador. Esse método retorna uma string com o valor inserido pelo usuário.

javascript
Copiar código

```
// Solicita ao usuário que insira seu nome let nome = prompt("Digite seu nome:"); // Exibe o nome no console
console.log("Olá, " + nome + "!");
```

Usando `console.log` para Saída de Dados

O método `console.log` é utilizado para exibir mensagens no console do navegador, o que é útil para depuração e registro de informações.

javascript
Copiar código

```
// Declaração de variáveis let a = 10; let b = 20; // Soma de duas variáveis let soma = a + b; // Exibe o resultado da soma
no console console.log("A soma de " + a + " e " + b + " é: " + soma);
```

O Console do Navegador

O console do navegador é uma ferramenta de desenvolvimento que permite aos desenvolvedores depurar e registrar mensagens enquanto estão escrevendo o código. A maioria dos navegadores modernos, como Google Chrome, Firefox e Edge, possuem um console integrado.

Acessando o Console do Navegador

Para abrir o console do navegador:

- **Google Chrome**: Pressione `Ctrl + Shift + J` (Windows/Linux) ou `Cmd + Option + J` (Mac).
- **Firefox**: Pressione `Ctrl + Shift + K` (Windows/Linux) ou `Cmd + Option + K` (Mac).
- **Edge**: Pressione `Ctrl + Shift + I` e depois clique na aba "Console".

Depuração de Erros

Depurar erros é um processo essencial no desenvolvimento de software. Aqui estão algumas técnicas comuns para depuração em JavaScript:

Usando `console.log`

Adicionar mensagens de log no código pode ajudar a entender o fluxo do programa e identificar onde os erros estão ocorrendo.

javascript
Copiar código

```
let x = 5; let y = 0; console.log("Valor de x: " + x); console.log("Valor de y: " + y); let resultado = x / y;
console.log("Resultado: " + resultado);
```

Usando Breakpoints

Breakpoints permitem pausar a execução do código em pontos específicos para inspecionar o estado do programa.

1. Abra o console do navegador.
2. Vá até a aba "Sources" (Fontes).

3. Encontre o arquivo JavaScript que você deseja depurar.
4. Clique no número da linha onde deseja adicionar o breakpoint.
5. Atualize a página para pausar a execução do código no breakpoint adicionado.

Mensagens de Erro

Quando um erro ocorre, o console do navegador exibe uma mensagem de erro que pode ajudar a identificar e corrigir o problema.

javascript
Copiar código

```
// Código com erro let numero = "dez"; let resultado = numero / 2; // Gera um NaN (Not a Number) console.log("Resultado: " + resultado);
```

Convertendo Strings para Números

Em JavaScript, é comum precisar converter uma string que representa um número em um tipo de dado numérico. Existem várias maneiras de fazer isso:

Usando `parseInt` e `parseFloat`

`parseInt` converte uma string em um número inteiro, enquanto `parseFloat` converte uma string em um número de ponto flutuante.

javascript
Copiar código

```
let strNumero1 = "42"; let strNumero2 = "3.14"; let intNumero = parseInt(strNumero1); let floatNumero = parseFloat(strNumero2); console.log("Número inteiro: " + intNumero); // 42 console.log("Número de ponto flutuante: " + floatNumero); // 3.14
```

Usando o Operador Unário `+`

Uma maneira rápida de converter uma string em número é usar o operador unário `+`.

javascript
Copiar código

```
let strNumero = "50"; let numero = +strNumero; console.log("Número: " + numero); // 50
```

Usando `Number()`

A função `Number` converte uma string em um número. É uma maneira mais explícita de conversão.

javascript
Copiar código

```
let strNumero = "100"; let numero = Number(strNumero); console.log("Número: " + numero); // 100
```

Conclusão

A estrutura sequencial é fundamental para a programação. Utilizando `prompt` e `console.log`, podemos interagir com o usuário e depurar nosso código. Entender o console do navegador e como depurar erros é crucial para o desenvolvimento eficiente. Saber converter strings em números é uma habilidade útil que permite manipular dados de maneira mais flexível em JavaScript.

1.10. Aula 05

dia 10 javascript vd 5