**Guia de apoio sobre Funções e Manipulação do DOM em JavaScript**

**1. Funções em JavaScript**

**1.1 O que é uma Função?**

Uma **função** é um bloco de código que executa uma tarefa específica e pode retornar um valor. As funções ajudam a modularizar o código, tornando-o mais organizado, legível e reutilizável.

**Sintaxe de uma Função**

javascript

function nomeDaFuncao(parametro1, parametro2) {

// Bloco de comandos

return resultado;

}

**Exemplo Prático**

javascript

function dobro(num) {

return num \* 2;

}

let resultado = dobro(5); // Chamada da função

console.log(resultado); // Saída: 10

**1.2 Expressão de Função (Função Anônima)**

Uma **expressão de função** é uma função sem nome (função anônima) que pode ser atribuída a uma variável.

**Exemplo Prático**

javascript

let dobro = function(num) {

return num \* 2;

};

console.log(dobro(5)); // Saída: 10

**1.3 Função de Flecha (Arrow Function)**

As **arrow functions** são uma forma simplificada de escrever expressões de função. Elas foram introduzidas no ECMAScript 6 (ES6) e tornam o código mais conciso.

**Sintaxe de uma Arrow Function**

javascript

const nomeDaFuncao = (parametro1, parametro2) => {

// Bloco de comandos

return resultado;

};

**Exemplo Prático**

javascript

const gerarNumeroAleatorio = () => Math.floor(Math.random() \* 100);

console.log(gerarNumeroAleatorio());

**Observação**: Quando a arrow function possui apenas uma linha, o uso de chaves {} e da palavra return pode ser omitido.

**1.4 Função Construtora**

Uma **função construtora** é usada para criar objetos. É comum iniciar o nome da função construtora com letra maiúscula para diferenciá-la das funções normais.

**Exemplo Prático**

javascript

function Carro(marca, modelo, ano) {

this.marca = marca;

this.modelo = modelo;

this.ano = ano;

}

let meuCarro = new Carro("Volkswagen", "Fusca", 1980);

console.log(meuCarro.marca); // Saída: Volkswagen

Aqui, meuCarro é um objeto criado a partir da função construtora Carro.

**2. Manipulação do DOM (Document Object Model)**

**2.1 O que é o DOM?**

O **DOM (Document Object Model)** é uma representação em árvore de todos os elementos de um documento HTML ou XML. Ele permite que você interaja e manipule esses elementos com JavaScript, tornando as páginas web dinâmicas e interativas.

**2.2 Selecionando Elementos do DOM**

Para manipular o DOM, primeiro você precisa acessar os elementos desejados. Existem vários métodos para isso:

* getElementById: Seleciona um elemento pelo seu id.
* getElementsByClassName: Seleciona elementos por sua classe.
* querySelector: Seleciona o primeiro elemento que corresponde ao seletor CSS fornecido.
* querySelectorAll: Seleciona todos os elementos que correspondem ao seletor CSS fornecido.

**Exemplo Prático**

<!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 Manipulação DOM</title>

</head>

<body>

<div id="meuId">Texto original</div>

<button onclick="mudarTexto()">Clique para mudar o texto</button>

<script>

function mudarTexto() {

let elemento = document.getElementById("meuId");

elemento.innerText = "Novo texto"; // Modifica o conteúdo do elemento

}

</script>

</body>

</html>

**2.3 Manipulação de Estilos e Conteúdo**

Com JavaScript, você pode alterar o conteúdo e o estilo dos elementos selecionados.

**Exemplo Prático**

<!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 Manipulação DOM</title>

</head>

<body>

<div id="meuId">Texto original</div>

<button onclick="mudarEstilo()">Clique para mudar o estilo e o texto</button>

<script>

function mudarEstilo() {

let elemento = document.getElementById("meuId");

elemento.style.color = "red"; // Modifica a cor do texto

elemento.innerHTML = "<b>Texto em negrito</b>"; // Modifica o conteúdo HTML

}

</script>

</body>

</html>

**2.4 Eventos e Closures**

**2.4.1 Eventos**

Um **evento** é uma ação que ocorre em um elemento da página, como um clique do mouse ou uma tecla pressionada. Em JavaScript, você pode usar manipuladores de eventos para responder a essas ações.

**Exemplo de Evento de Clique**

<!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 Manipulação DOM</title>

</head>

<body>

<button id="meuBotao">Clique aqui</button>

<script>

document.getElementById("meuBotao").addEventListener("click", function() {

alert("Botão clicado!"); // Exibe um alerta quando o botão é clicado

});

</script>

</body>

</html>

**2.4.2 Closures**

Um **closure** ocorre quando uma função é definida dentro de outra função e mantém acesso às variáveis da função externa, mesmo após a execução da função externa.

**Exemplo Prático de Closure**

function saudacao(nome) {

let mensagem = "Olá, " + nome;

function mostrarMensagem() {

console.log(mensagem);

}

return mostrarMensagem;

}

let saudacaoParaJoao = saudacao("João");

saudacaoParaJoao(); // Saída: Olá, João

Neste exemplo, a função mostrarMensagem tem acesso à variável mensagem da função saudacao, mesmo após a função saudacao ter sido executada.

**2.5 Delegação de Eventos**

A **delegação de eventos** é uma técnica onde você usa um único manipulador de eventos em um elemento pai para controlar eventos em seus elementos filhos. Isso é eficiente e útil quando você tem muitos elementos filhos.

**Exemplo Prático**

<!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 Manipulação DOM</title>

</head>

<body>

<ul id="lista">

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

</ul>

<script>

document.getElementById("lista").addEventListener("click", function(event) {

if (event.target.tagName === "LI") {

event.target.style.color = "blue"; // Altera a cor do item clicado para azul

}

});

</script>

</body>

</html>

Neste exemplo, ao clicar em qualquer item <li> dentro da <ul id="lista">, a cor do texto será alterada para azul.

**Explicação:**

* **HTML**: A estrutura contém uma lista (<ul>) com três itens (<li>), todos dentro de um elemento com o id "lista".
* **JavaScript**: O código adiciona um ouvinte de eventos (addEventListener) à lista. Quando qualquer item da lista (<li>) é clicado, o JavaScript verifica se o elemento clicado (event.target) é um LI. Se for, ele altera a cor do texto desse item para azul.

**Atividades 1:**

Aqui estão alguns exercícios para praticar a transformação de funções nomeadas em funções anônimas:

**Exercício 1**

**Função Nomeada:**

javascript

function quadrado(numero) {

return numero \* numero;

}

let resultado = quadrado(4);

console.log(resultado); // Saída: 16

**Transforme a função quadrado em uma função anônima atribuída a uma variável.**

**Exercício 2**

**Função Nomeada:**

javascript

function saudacao(nome) {

return `Olá, ${nome}!`;

}

let mensagem = saudacao('Maria');

console.log(mensagem); // Saída: Olá, Maria!

**Transforme a função saudacao em uma função anônima atribuída a uma variável.**

**Exercício 3**

**Função Nomeada:**

javascript

function soma(a, b) {

return a + b;

}

let resultado = soma(10, 5);

console.log(resultado); // Saída: 15

**Transforme a função soma em uma função anônima atribuída a uma variável.**

**Exercício 4**

**Função Nomeada:**

javascript

function verificaPar(num) {

return num % 2 === 0;

}

let resultado = verificaPar(7);

console.log(resultado); // Saída: false

**Transforme a função verificaPar em uma função anônima atribuída a uma variável.**

**Exercício 5**

**Função Nomeada:**

javascript

function calculaFatorial(n) {

if (n === 0) return 1;

return n \* calculaFatorial(n - 1);

}

let resultado = calculaFatorial(5);

console.log(resultado); // Saída: 120

**Transforme a função calculaFatorial em uma função anônima atribuída a uma variável.**

**Exercício 6**

**Função Nomeada:**

javascript

function dobroDeCadaElemento(array) {

return array.map(num => num \* 2);

}

let resultado = dobroDeCadaElemento([1, 2, 3]);

console.log(resultado); // Saída: [2, 4, 6]

**Transforme a função dobroDeCadaElemento em uma função anônima atribuída a uma variável.**

**Exercício 7**

**Função Nomeada:**

javascript

function juntarStrings(str1, str2) {

return str1 + str2;

}

let resultado = juntarStrings('Olá', ' Mundo!');

console.log(resultado); // Saída: Olá Mundo!

**Transforme a função juntarStrings em uma função anônima atribuída a uma variável.**

**Atividades 2:**

Aqui estão quatro exemplos que mostram como transformar funções normais em funções de flecha (arrow functions) em JavaScript. Vamos incluir tanto funções com parâmetros quanto funções sem parâmetros.

**Exemplo 1: Função Simples com um Parâmetro**

**Função Normal:**

javascript

function quadrado(num) {

return num \* num;

}

console.log(quadrado(4)); // Saída: 16

**Exemplo 2: Função Simples com Múltiplos Parâmetros**

**Função Normal:**

javascript

function soma(a, b) {

return a + b;

}

console.log(soma(5, 3)); // Saída: 8

**Exemplo 3: Função Sem Parâmetros**

**Função Normal:**

javascript

function saudacao() {

return "Olá, mundo!";

}

console.log(saudacao()); // Saída: Olá, mundo!

**Exemplo 4: Função com Bloco de Código (Múltiplas Linhas)**

**Função Normal:**

javascript

function multiplicarPorDois(num) {

let resultado = num \* 2;

return resultado;

}

console.log(multiplicarPorDois(7)); // Saída: 14

**Atividades 3:**

Aqui estão três atividades adicionais para praticar o uso de funções construtoras em JavaScript:

**1. Atividade: Criar uma Função Construtora para um Livro**

**Objetivo:** Criar uma função construtora chamada Livro que define as propriedades de um livro e exibe informações sobre o livro.

**Instruções:**

1. Crie a função construtora Livro com as propriedades titulo, autor, e anoPublicacao.
2. Adicione um método chamado exibirInfo que retorna uma string com as informações do livro. Ex.: let meuLivro = new Livro("1984", "George Orwell", 1949);
3. Crie um objeto meuLivro usando o construtor Livro e chame o método exibirInfo.

**2. Atividade: Criar uma Função Construtora para um Produto**

**Objetivo:** Criar uma função construtora chamada Produto que define as propriedades de um produto e calcula o preço com desconto.

**Instruções:**

1. Crie a função construtora Produto com as propriedades nome, preco, e desconto.
2. Adicione um método chamado precoFinal que calcula e retorna o preço após aplicar o desconto. Ex.: let meuProduto = new Produto("Smartphone", 1200, 15);
3. Crie um objeto meuProduto usando o construtor Produto e chame o método precoFinal.

**3. Atividade: Criar uma Função Construtora para um Estudante**

**Objetivo:** Criar uma função construtora chamada Estudante que define as propriedades de um estudante e calcula a média das notas.

**Instruções:**

1. Crie a função construtora Estudante com as propriedades nome e notas (um array de números).
2. Adicione um método chamado calcularMedia que calcula e retorna a média das notas. Ex: let meuEstudante = new Estudante("Ana", [7.5, 8.0, 9.0]);
3. Crie um objeto meuEstudante usando o construtor Estudante e chame o método calcularMedia.