Apostila Java Script

1. Adicionando Script em HTML

Para adicionar código JavaScript diretamente em um arquivo HTML, usamos a tag `<script>`:

```
<!DOCTYPE html>
<html>
<head>
    <title>Exemplo de JavaScript Embutido</title>
</head>
<body>
    <h1>Olá, Mundo!</h1>

    <!-- Código JavaScript embutido -->
    <script>
        alert("Este é um alerta usando JavaScript embutid
o!");
    </script>
</body>
</html>
```

No exemplo acima, um alerta é exibido assim que a página é carregada.

2. Referenciando um Arquivo JavaScript Externo em HTML

Para organizar melhor o código, podemos manter o JavaScript em um arquivo separado e referenciá-lo com a tag `<script>`.

```
<!DOCTYPE html>
<html>
<head>
    <title>Exemplo de JavaScript Externo</title>
    <!-- Referência para o arquivo JavaScript externo -->
    <script src="script.js"></script>
```

```
</head>
<body>
    <h1>Olá, Mundo!</h1>
</body>
</html>
```

O conteúdo do arquivo `script.js` poderia ser, por exemplo:

```
alert("Este é um alerta a partir de um arquivo JavaScript e
xterno!");
```

3. Sintaxe de JavaScript

Alguns pontos fundamentais da sintaxe do JavaScript:

1. **Declaração de Variáveis**: Podemos usar `var`, `let` ou `const` para declarar variáveis.

```
var nome = "João"; // Variável global ou local
let idade = 25; // Escopo de bloco
const pais = "Brasil"; // Constante
```

1. **Estruturas de Controle**: Condicionais e loops.

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

// Loop
for (let i = 0; i < 5; i++) {
    console.log("Contagem: " + i);
}
```

1. **Funções**: Formas de definir e chamar funções.

```
// Função clássica
function saudacao(nome) {
    return "Olá, " + nome + "!";
}

console.log(saudacao("Ana"));

// Função de seta (arrow function)
const saudacaoArrow = (nome) => "Olá, " + nome + "!";
console.log(saudacaoArrow("Carlos"));
```

4. Comandos e Operadores

JavaScript possui vários comandos e operadores para operações aritméticas, lógicas, de comparação, entre outros.

1. **Operadores Aritméticos**: Para cálculos básicos.

```
let a = 10;
let b = 5;
console.log(a + b); // Soma: 15
console.log(a - b); // Subtração: 5
console.log(a * b); // Multiplicação: 50
console.log(a / b); // Divisão: 2
console.log(a % b); // Resto da divisão: 0
```

1. **Operadores Lógicos**: Para combinações e condições lógicas.

```
let x = true;
let y = false;
console.log(x && y); // E lógico: false
console.log(x || y); // OU lógico: true
console.log(!x); // NÃO lógico: false
```

1. **Operadores de Comparação**: Para comparar valores.

```
let p = 10;
let q = "10";
console.log(p == q); // Igual: true
console.log(p === q); // Estritamente igual: false
console.log(p != q); // Diferente: false
console.log(p !== q); // Estritamente diferente: true
console.log(p > 5); // Maior que: true
console.log(p < 5); // Menor que: false
```

5 - Funções em JavaScript

Funções são blocos de código reutilizáveis que podem ser executados quando chamados. Elas nos ajudam a estruturar o código de forma modular e a evitar a repetição.

1. Funções Sem Parâmetros

Funções sem parâmetros são úteis quando o comportamento desejado não requer entrada do usuário.

```
// Definindo uma função sem parâmetros
function cumprimentar() {
    console.log("Olá, Mundo!");
}

// Chamando a função
cumprimentar(); // Saída: Olá, Mundo!
```

No exemplo acima, a função `cumprimentar` não requer nenhum dado externo para ser executada.

2. Funções com Parâmetros

Funções podem aceitar entradas, chamadas de parâmetros. Isso permite que seu comportamento varie conforme as informações fornecidas.

```
// Definindo uma função com parâmetros
function somar(a, b) {
    console.log("A soma é: " + (a + b));
}

// Chamando a função com diferentes argumentos
somar(3, 5); // Saída: A soma é: 8
somar(10, 20); // Saída: A soma é: 30
```

Aqui, a função `somar` aceita dois parâmetros `a` e `b`, que são usados para calcular a soma.

3. Funções Sem Retorno

Funções podem realizar uma tarefa sem necessariamente retornar um valor. Geralmente, apenas executam instruções como imprimir mensagens.

```
// Definindo uma função sem retorno
function mostrarMensagem(mensagem) {
    console.log("Mensagem: " + mensagem);
}

// Chamando a função
mostrarMensagem("Bem-vindo à aula de JavaScript!"); // Saíd
a: Mensagem: Bem-vindo à aula de JavaScript!
```

A função `mostrarMensagem` apenas imprime a mensagem fornecida, mas não retorna nada ao chamador.

4. Funções com Retorno

Funções também podem devolver um valor ao chamador com a instrução `return`.

```
// Definindo uma função com retorno
function calcularDobro(numero) {
    return numero * 2;
}
```

```
// Chamando a função e armazenando o resultado
let resultado = calcularDobro(7);
console.log("O dobro é: " + resultado); // Saída: O dobro
é: 14
```

6. Acesso a Elementos do HTML

Podemos acessar elementos HTML usando métodos como `getElementById` ou `querySelector`.

```
<!DOCTYPE html>
<html>
<head>
    <title>Acessando Elementos</title>
    <script>
        function alterarTexto() {
            let elemento = document.getElementById("paragra
fo");
            elemento.textContent = "Texto alterado via Java
Script!";
        }
    </script>
</head>
<body>
    <p id="paragrafo">Texto original</p>
    <button onclick="alterarTexto()">Alterar Texto</button>
</body>
</html>
```

No exemplo acima, o texto do parágrafo é alterado ao clicar no botão.

7. Alteração de Propriedades de Elementos HTML

Além de alterar texto, podemos mudar estilos e outras propriedades de elementos.

```
<!DOCTYPE html>
<html>
<head>
    <title>Alterando Estilos</title>
    <script>
        function mudarCor() {
            let elemento = document.getElementById("paragra
fo");
            elemento.style.color = "blue";
            elemento.style.fontSize = "20px";
        }
    </script>
</head>
<body>
    <p id="paragrafo">Texto com cor alterada</p>
    <button onclick="mudarCor()">Mudar Cor</button>
</body>
</html>
```

Neste exemplo, a cor e o tamanho da fonte do parágrafo são alterados quando o botão é clicado.

8. Eventos em JavaScript

Os eventos permitem que o JavaScript responda a ações dos usuários, como cliques ou digitação.

1. **Clique de Botão**: Usando `onclick`.

```
<!DOCTYPE html>
<html>
<head>
    <title>Evento de Clique</title>
    <script>
        function mostrarMensagem() {
            alert("Botão clicado!");
        }
    </script>
```

```
</head>
<body>
    <button onclick="mostrarMensagem()">Clique Aqui</button
>
</body>
</html>
```

1. **Mudança de Campo de Texto**: Usando `onchange`.

```
<!DOCTYPE html>
<html>
<head>
    <title>Evento de Mudança</title>
    <script>
        function exibirMensagem() {
            let valor = document.getElementById("campoText
o").value;
            alert("Texto digitado: " + valor);
        }
    </script>
</head>
<body>
    <input type="text" id="campoTexto" onchange="exibirMens
agem()">
</body>
</html>
```

1. **Evento de Teclado**: Capturando teclas pressionadas com `onkeydown`.

```
<!DOCTYPE html>
<html>
<head>
    <title>Evento de Teclado</title>
    <script>
        function capturarTecla(event) {
            let tecla = event.key;
            document.getElementById("saida").textContent =
"Tecla pressionada: " + tecla;
```

```
        }
    </script>
</head>
<body>
    <input type="text" onkeydown="capturarTecla(event)">
    <p id="saida">Pressione uma tecla</p>
</body>
</html>
```

1. **Carregamento de Página**: Usando `onload`.

```
<!DOCTYPE html>
<html>
<head>
    <title>Evento de Carregamento</title>
    <script>
        function aoCarregar() {
            alert("Página carregada com sucesso!");
        }
    </script>
</head>
<body onload="aoCarregar()">
    <h1>Bem-vindo!</h1>
</body>
</html>
```