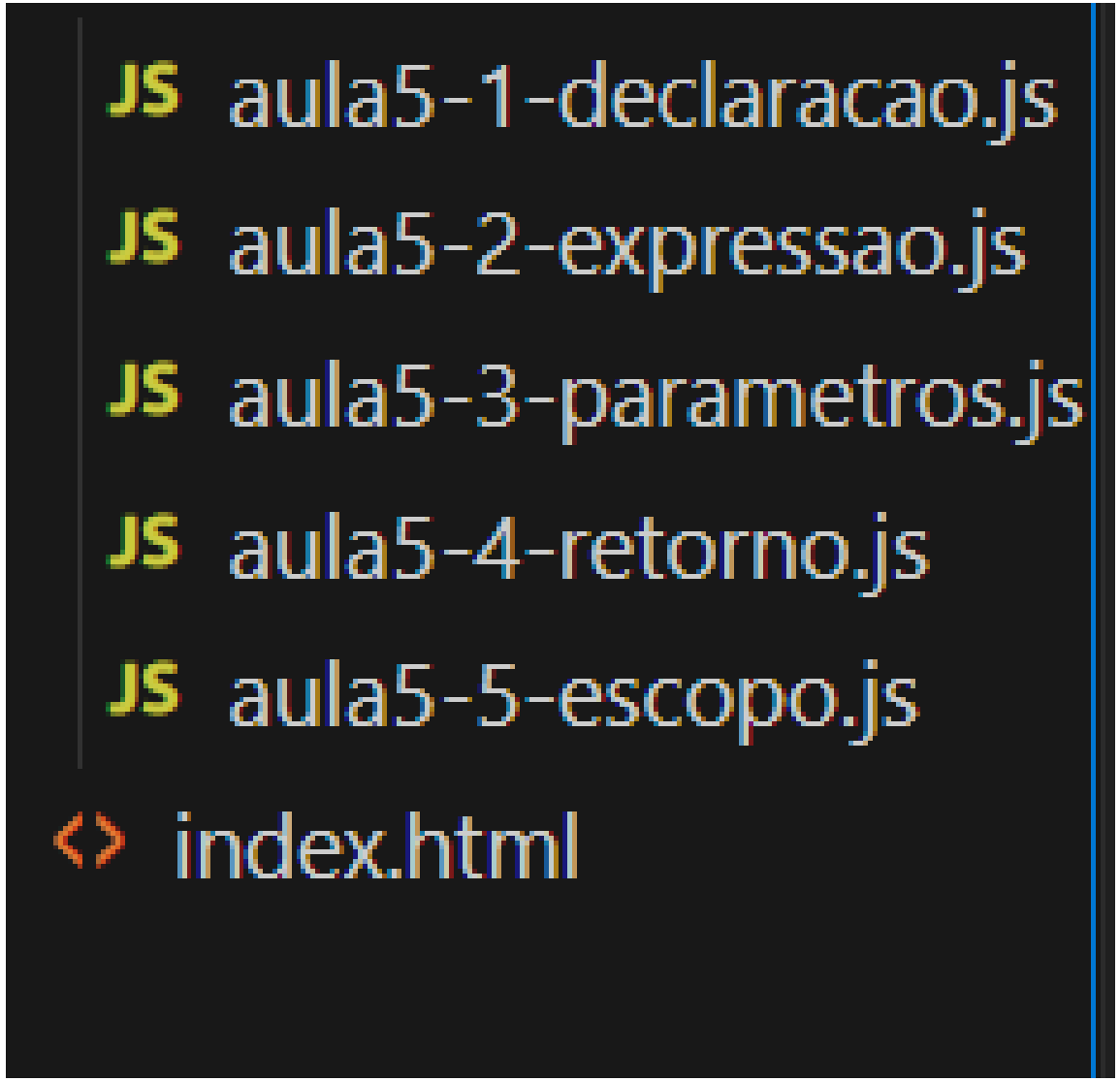
Aula 11

Funções


aula5-1-declaracao.js
aula5-2-expressao.js
aula5-3-parametros.js
aula5-4-retorno.js
aula5-5-escopo.js
index.html

index.html | aula5-5-escopo.js | aula5-4-retorno.js | aula5-3-parametros.js | aula5-2-expressao.js

index.html > html > body

```
1   <!DOCTYPE html>
2   <html lang="pt-BR">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Aula 5</title>
7       </head>
8   <body>
9       <h1>Aula 5: Funções em JavaScript</h1>
10      <h2>Uma função em JavaScript é um bloco de código reutilizável projetado para realizar uma
        tarefa específica.</h2>
11
12      <!-- <script src="js/aula5-1-declaracao.js"></script> -->
13      <script src="js/aula5-2-expressao.js"></script>
14      <!-- <script src="js/aula5-3-parametros.js"></script>
15      <script src="js/aula5-4-retorno.js"></script>
16      <script src="js/aula5-5-escopo.js"></script> -->
17  </body>
18  </html>
```

JS aula5-1-declaracao.js > ...

```
1   //DECLARAÇÃO DE FUNÇÕES (Function Declaration)
2   // - Forma padrão. Sofre hoisting (pode ser chamada antes da declaração).
3
4   minhaDeclaracao(); // Chamada ANTES da declaração (Hoisting)
5
6   //criando uma função
7   function minhaDeclaracao() {
8       console.log("1. Função declarada executada!");
9   }
10
11  minhaDeclaracao(); // Chamada APÓS a declaração
```

JS aula5-2-expressao.js > ...

```
// TÓPICO 2: EXPRESSÃO DE FUNÇÃO (Function Expression)
// - Função atribuída a uma variável. NÃO sofre hoisting.
console.log("\n--- JS: aula5-2-expressao.js ---");

// Erro
//minhaExpressao();

const minhaExpressao = function() {
    console.log("2. Função por expressão executada!");
};

minhaExpressao(); // Chamada APÓS a atribuição

const expressaoNomeada = function nomeInternoDaFuncao() {
    console.log("2.1. Expressão de função nomeada (nome interno útil para debug).");
};
expressaoNomeada();
```

JS aula5-3-parametros.js > ...

```
// TÓPICO 3: PARÂMETROS E ARGUMENTOS
// - Parâmetros: "variáveis" na definição da função.
// - Argumentos: "valores reais" passados na chamada da função.
console.log("\n--- JS: aula5-3-parametros.js ---");

function saudar(nome, saudacao = "Olá") { // 'saudacao' tem valor padrão
    console.log(`3. ${saudacao}, ${nome}!`);
}

saudar("Aluno");                    // Usa o padrão para 'saudacao'
saudar("Professor", "Bom dia");     // Fornece ambos os argumentos
saudar("Maria", undefined);         // Força o uso do padrão para 'saudacao' mesmo com
argumento 'undefined'
```

s > JS aula5-4-retorno.js > ...

```js
// TÓPICO 4: RETORNO DE VALORES (return)
// - Funções podem "devolver" um resultado usando 'return'.
// - Sem 'return' explícito, a função retorna 'undefined'.
console.log("\n--- JS: aula5-4-retorno.js ---");

function calcularArea(largura, altura) {
    if (largura <= 0 || altura <= 0) {
        return "Dimensões inválidas"; // Retorno antecipado para validação
    }
    return largura * altura; // Retorna o resultado do cálculo
    // console.log("Isso não executa"); // Código após return não é alcançado
}

let area1 = calcularArea(5, 10);
console.log("4. Área (5x10):", area1); // 50

let areaInvalida = calcularArea(5, -2);
console.log("4.1. Área (5x-2):", areaInvalida); // Dimensões inválidas

function semRetorno() {
    // Nenhuma instrução 'return'
}
let resultadoSemRetorno = semRetorno();
console.log("4.2. Função sem retorno explícito:", resultadoSemRetorno); // undefined
```

```js
// TÓPICO 5: ESCOPO DE VARIÁVEIS - VAR vs LET (Exemplo Único e Direto)
console.log("--- JS: aula5-5-escopo.js (Ultra Curto) ---");
console.log("Um exemplo para diferenciar escopo de 'var' e 'let'.");

function demonstrarDiferencaEscopo() {
    if (true) {
        //ecopo de função
        var statusComVar = "VAR: Visível fora do bloco, dentro da função.";
        //escopo de bloco
        let statusComLet = "LET: Visível APENAS dentro deste bloco.";



        console.log("DENTRO DO BLOCO:");
        console.log(statusComVar);  // Acessível
        console.log(statusComLet);  // Acessível
    }

    console.log("\nFORA DO BLOCO (mas dentro da função):");
    console.log(statusComVar);  // Acessível! 'var' tem escopo de função.
    // console.log(statusComLet); // ERRO! statusComLet is not defined. 'let' tem escopo de bloc
                                  // Descomente a linha acima para ver o erro.
}

demonstrarDiferencaEscopo();
```

Atividade

Exercício 1: Saudação Personalizada 🗣️

- **Arquivo:** `exercicio1-saudacao.js`
- **Enunciado:** Crie uma função chamada `cumprimentar` que aceite um **nome** como argumento e exiba no console uma mensagem de saudação personalizada, como "Olá, [Nome]! Bem-vindo(a)!".

Exercício 2: Dobro do Número 🔢

- **Arquivo:** `exercicio2-dobro.js`
- **Enunciado:** Escreva uma função chamada `calcularDobro` que receba um **número** como parâmetro e **retorne** o dobro desse número. Teste a função com diferentes números e exiba o resultado retornado no console.

Exercício 3: Verificar Maioridade 🔞

- **Arquivo:** `exercicio3-maioridade.js`
- **Enunciado:** Crie uma função chamada `verificarMaioridade` que receba uma **idade** como argumento. A função deve **retornar** `true` se a idade for 18 ou maior, e `false` caso contrário. Imprima o resultado da verificação no console para algumas idades.

Exercício 4: Calcular Média de Dois Números 📊

- **Arquivo:** `exercicio4-media.js`
- **Enunciado:** Elabore uma função chamada `calcularMedia` que aceite **dois números** como parâmetros. A função deve calcular a média aritmética desses dois números e **retornar** o resultado. Demonstre seu uso.

Exercício 5: Apresentar Informações 📝

- **Arquivo:** `exercicio5-apresentacao.js`
- **Enunciado:** Crie uma função chamada `apresentarUsuario`. Esta função deve aceitar três argumentos: **nome**, **idade** e **cidade**. A função deve construir e exibir no console uma frase que apresente o usuário, por exemplo: "Conheçam [Nome], de [Idade] anos, que mora em [Cidade]."

Adapte o programa para receber nome, idade e cidade de um formulário HTML (DOM)