Aula 13 – Arrays e Manipulação do DOM

PARTE 1 — ARRAYS

Conceito

Um **array** é uma estrutura que guarda vários valores em uma única variável.
Cada valor tem um **índice (posição)**, que começa em **0**.

```
let alunos = ["Ana", "Bruno", "Carlos"];
```

🔍 Acessar elementos

```
console.log(alunos[0]); // "Ana"
console.log(alunos[2]); // "Carlos"
```

🖍️ Modificar elementos

```
alunos[1] = "Beatriz";
console.log(alunos); // ["Ana", "Beatriz", "Carlos"]
```

⚙️ Operações básicas

Operação	Método	Exemplo	Resultado
Inserir no final	`.push()`	`alunos.push("Daniel")`	["Ana", "Beatriz", "Carlos", "Daniel"]
Remover o último	`.pop()`	`alunos.pop()`	remove "Daniel"
Inserir no início	`.unshift()`	`alunos.unshift("Laura")`	["Laura", "Ana", "Beatriz", "Carlos"]
Remover o primeiro	`.shift()`	`alunos.shift()`	remove "Laura"
Remover por índice	`.splice()`	`alunos.splice(1, 1)`	remove o item do índice 1 ("Beatriz")
Saber o tamanho	`.length`	`alunos.length`	3

🔁 Percorrer a lista (com `for`)

```
for (let i = 0; i < alunos.length; i++) {
    console.log(i + ": " + alunos[i]);
}
```

PARTE 2 — CONECTANDO JS E HTML (DOM)

O que é o DOM

O **DOM (Document Object Model)** é uma representação da página como uma **árvore de elementos**.
Cada tag HTML vira um **nó (node)** que pode ser acessado e modificado via JavaScript.

Exemplo visual:

```
document
 └── html
     └── body
         ├── h1
         └── ul
             ├── li
             └── li
```

🪄 O objeto `document`

É o ponto de entrada para manipular o DOM.

Selecionar elementos

`getElementById`

Seleciona um elemento com base no seu `id`.

```
<p id="mensagem">Olá!</p>
```

```
let p = document.getElementById("mensagem");
console.log(p.innerText); // "Olá!"
```

`getElementsByClassName`

Seleciona todos os elementos de uma classe (retorna uma **HTMLCollection**).

```
<p class="aluno">Ana</p>
<p class="aluno">Bruno</p>
```

```
let alunos = document.getElementsByClassName("aluno");
console.log(alunos[0].innerText); // "Ana"
```

Modificar conteúdo

Propriedade	Uso	Exemplo
`.innerText`	Altera o texto visível	`p.innerText = "Bem-vindo!"`
`.innerHTML`	Altera o HTML interno	`div.innerHTML = "<b>Olá</b>"`
`.value`	Altera/obtém o valor de inputs	`input.value = "Novo texto"`

Criar e remover elementos

Criar um elemento:

```
let novoItem = document.createElement("li");
novoItem.innerText = "Novo produto";
document.getElementById("lista").appendChild(novoItem);
```

Remover um elemento:

Agora vamos usar a forma mais direta, removendo o próprio elemento com `.remove()`.

```
<p id="mensagem">Texto temporário</p>

let mensagem = document.getElementById("mensagem");
mensagem.remove(); // remove o parágrafo do DOM
```

Definir atributos de um elemento (`setAttribute`)

```
let botao = document.getElementById("botao");
botao.innerText = "Clique aqui";
botao.setAttribute("id", "btnAdicionar");
botao.setAttribute("class", "botao-principal");
```

O método `setAttribute(atributo, valor)` é usado para **definir ou alterar atributos HTML** de um elemento via JavaScript.
Ele é especialmente útil para ajustar propriedades como `id`, `class`, `src`, `href`, `type`, `value` e outras.

💡 Exemplos comuns:

```
botao.setAttribute("disabled", true); // desativa um botão
input.setAttribute("placeholder", "Digite seu nome");
link.setAttribute("href", "https://www.exemplo.com");
```

Exemplo prático - Lista de produtos

Desenvolva uma página web que permita **gerenciar uma lista de produtos**.
A página deve conter:

- Um campo de entrada (`<input>`) para o usuário digitar o nome do produto;
- Uma lista (`<ul>`) para exibir os produtos cadastrados;
- Um botão para **adicionar** o produto à lista;

O comportamento da página deve ser o seguinte:

- Ao clicar no botão **Adicionar**, o produto digitado deve ser inserido **ao final da lista**;
- Caso o usuário clique em **Adicionar** sem informar um nome, uma **mensagem de aviso** (`alert`) deve ser exibida;