A22. JavaScript: Fundamentos, Manipulação do DOM

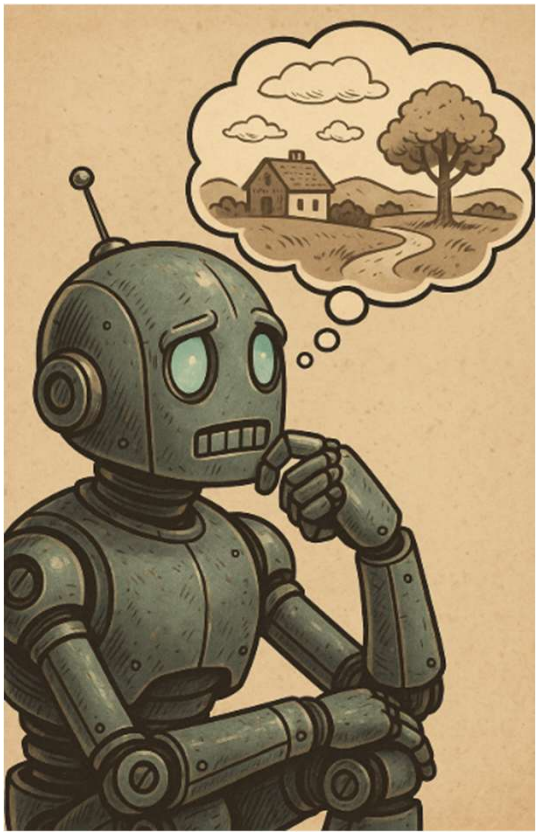
M6. JavaScript

Objetivo da aula

- Introduzir os fundamentos de JavaScript no navegador
- Trabalhar com variáveis, tipos de dados e operadores
- Aplicar estruturas condicionais e de repetição
- Compreender o DOM e manipulá-lo com JS
- Implementar eventos e listeners

Relembrando a última aula...

- ✓ Introdução a Layout Responsivo
- ✓ Flexbox
- ✓ Grid Layout
- ✓ Flexbox x Grid



Nossa aula...

- ✓ Introdução ao JavaScript
- ✓ Fundamentos de JavaScript
- ✓ Estruturas de Controle
- ✓ Funções
- ✓ DOM (Document Object Model)
- ✓ Eventos
- ✓ Debugging
- ✓ Boas Práticas

Introdução ao JavaScript

O que é JavaScript?

- Linguagem de programação interpretada, criada em 1995
- Executa diretamente nos navegadores
- Um dos **3 pilares da Web: HTML, CSS, JavaScript**

- ***Por que é importante?***
  - Dá vida às páginas web
  - Permite criar interações dinâmicas com o usuário
  - É a base para frameworks modernos (React, Angular, Vue.js)

Onde é usado?

- Navegadores (Front-End)
- Servidores (Node.js)
- Aplicativos móveis e desktop (React Native, Electron)
- IoT, jogos e inteligência artificial

Exemplos de uso

- Validar formulários antes de enviar
- Criar animações e efeitos visuais
- Alterar elementos sem recarregar a página
- Comunicar-se com servidores (APIs)

"JavaScript começa simples, mas é poderoso: tudo parte das variáveis, tipos e operadores."

Fundamentos de JavaScript

- **Declaração de variáveis**
  - var – antigo, não recomendado
  - let – variável com escopo de bloco
  - const – valor fixo, não pode ser reatribuído
- **Tipos de dados**
  - Primitivos: *string, number, boolean, null, undefined*
  - Estruturados: *array, object*
- **Operadores**
  - Aritméticos: + - * / %
  - Relacionais: ==, ===, !=, <, >, <=, =>
  - Lógicos: &&, ||, !

== (igualdade solta)
- Compara apenas os valores
- Faz conversão implícita de tipos
- Pode dar resultados inesperados

=== (igualdade estrita)
- Compara valores e tipos
- Não faz conversão automática de tipos
- É a forma mais recomendada

D1.
Vamos demonstrar...

Estruturas de Controle

- **Condicionais (tomada de decisão)**
  - if – Executa um bloco se a condição for verdadeira
  - else – Executa caso contrário
  - else if – Testa múltiplas condições
  - switch – Útil para várias comparações em sequência

D2.
Vamos demonstrar...

Estruturas de Controle

- **Laços de Repetição (loops)**
  - for – repete um bloco um número conhecido de vezes
  - while – repete enquanto a condição for verdadeira
  - do..while – executa pelo menos uma vez o bloco de repetição antes de testar a condição

D3.
Vamos demonstrar...

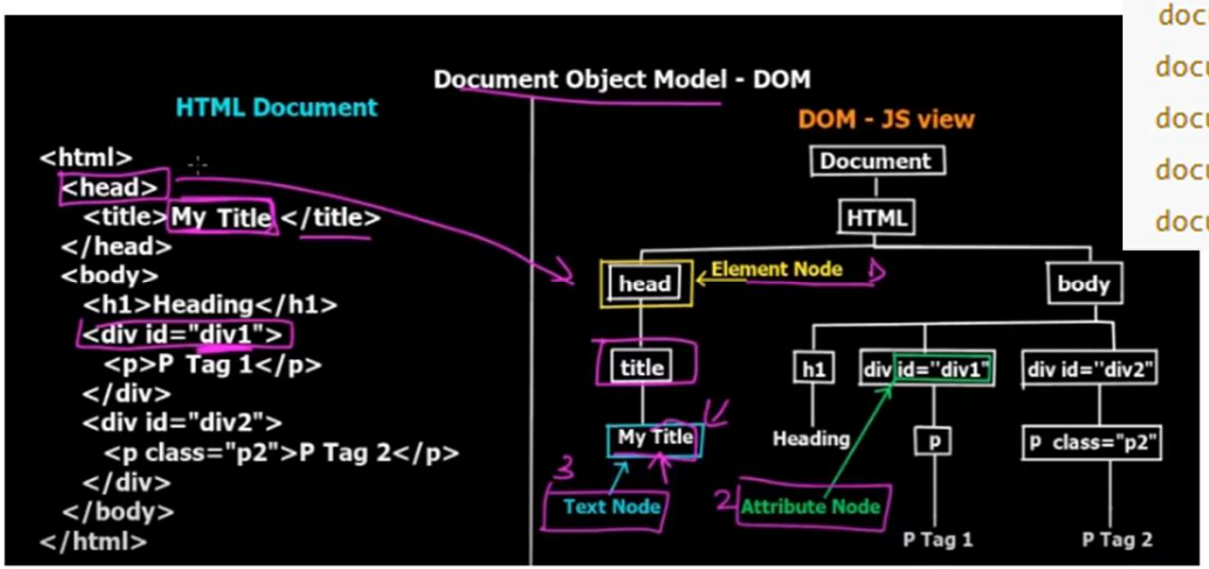
Funções

- ***O que são funções?***
  - Conjunto de instruções que realizam uma tarefa especifica
  - Permitem **reutilização de código** e organização
  - Podem receber **parâmetros** e retornar valores

D4.
Vamos demonstrar...

DOM (Document Object Model)

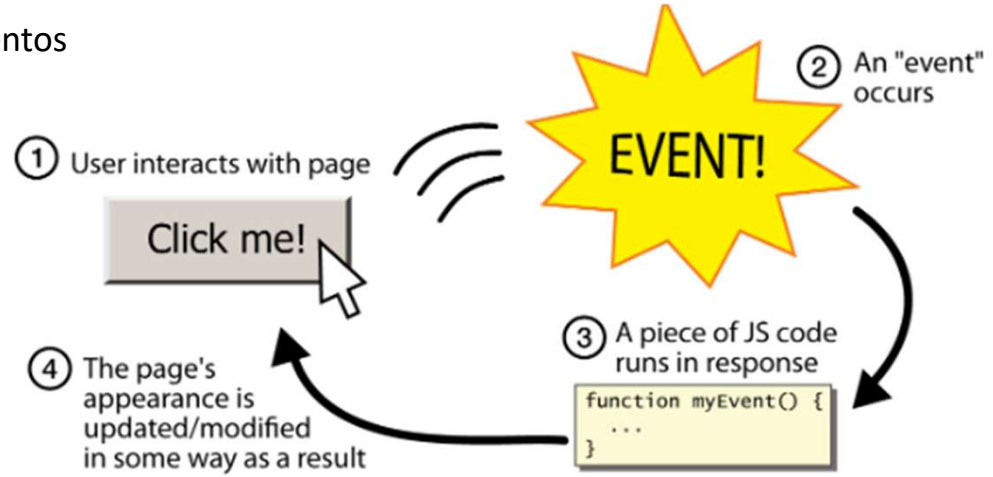
- ***O que é DOM?***
  - **Document Object Model**: representação em árvore da página HTML
  - Cada **tag** vira um **nó/objeto** que pode ser manipulado pelo JavaScript
  - Permite **acessar, modificar, adicionar ou remover** elementos dinamicamente




```
document.getElementById("titulo");       // por id
document.getElementsByClassName("item"); // por classe
document.getElementsByTagName("p");      // por tag
document.querySelector("#titulo");       // seletor CSS
document.querySelectorAll(".item");      // vários elementos
```

Eventos

- ***O que são eventos?***
  - Ações do usuário ou do navegador que podem ser detectadas pelo JavaScript
  - **Exemplos**: clique de mouse, pressionar tecla, carregar página, enviar formulário
  - O JS pode **ouvir** (listen) e **responder** (handler) a esses eventos
- **Eventos mais comuns**
  - onclick – clique em elemento
  - onmouseover – mouse sobre o elemento
  - onchange – mudança de input/select
  - onkeyup / onkeydown – teclado
  - onsubmit – envio de formulário
  - onload – quando a página termina de carregar



Maneiras de Usar Eventos

1. Atributo direto no HTML (menos recomendado)

```html
<button onclick="alert('Você clicou!')">Clique aqui</button>
```

2. Função no JS

```html
<button onclick="saudacao()">Clique aqui</button>
<script>
  function saudacao() {
    alert("Olá, seja bem-vindo!");
  }
</script>
```

3. addEventListener (**recomendado**)

```html
<script>
  let botao = document.querySelector("button");
  botao.addEventListener("click", () => {
    alert("Evento via addEventListener!");
  });
</script>
```

D5.
Vamos demonstrar...

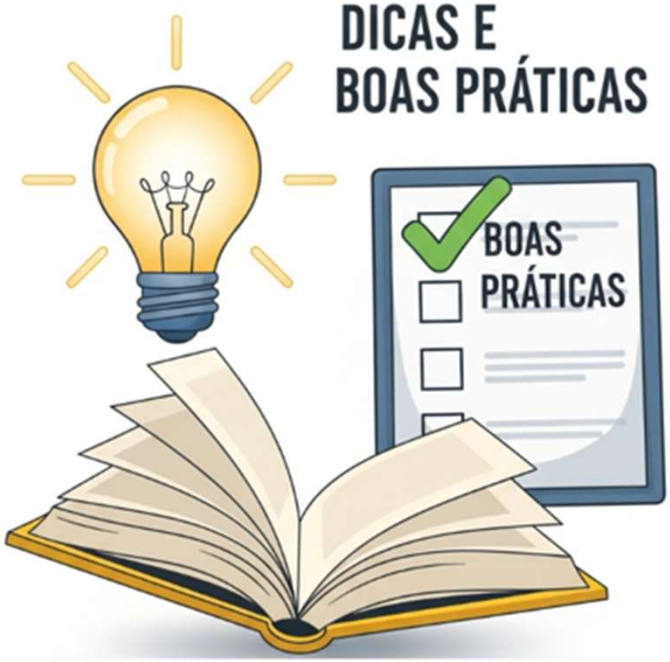
Debbugging

- ***O que é Debugging?***
  - Processo de **identificar e corrigir erros** no código
  - No navegador, usamos o **DevTools** (Console e Sources)
- **Principais ferramentas**
  - console.log() – exibir mensagens
  - console.error() – destacar erros
  - console.table() – exibir arrays e objetos em tabela
  - breakpoints – para execução em um ponto para inspecionar variáveis

D6.
Vamos demonstrar...

Boas Práticas

1. **Clareza e legibilidade**
   - Use nomes descritivos: let idadeAluno e não let x
   - Indente o código corretamente
2. **Constância e Escopo**
   - Use const sempre que o valor não mudar
   - Use let para variáveis mutáveis
   - Evite var
3. **Separação de responsabilidades**
   - Deixe o HTML limpo – evite onclick no HTML
   - Use JS separado em arquivo externo
4. **Evite repetição de código**
   - Crie funções reutilizáveis
5. **Comentários moderados**
   - Comente o **porquê** do código, não o óbvio
6. **Testar sempre**
   - Teste incrementando pequenas partes antes do projeto todo



Perguntas?! Dúvidas?!?