Programação Front End – AULA 15

Matheus Moresco
Engenharia de Software – 3 Período
2025/01

Introdução

- Compreender o que é o DOM e como acessá-lo com JavaScript.
- Aprender a manipular elementos HTML dinamicamente.
- Utilizar eventos para tornar páginas interativas.

O que é o DOM?

- O **DOM (Document Object Model)** é uma **representação em forma de árvore** de todos os elementos HTML e CSS de uma página web. Ele permite que linguagens como **JavaScript** acessem, leiam e modifiquem o conteúdo, a estrutura e o estilo de uma página enquanto ela está sendo exibida no navegador.

- O **DOM** é como uma tradução da página HTML para uma estrutura que o JavaScript consegue entender e manipular.

O que é o DOM?

- Cada tag vira um **nó (node)** na árvore. Com JavaScript, você pode navegar por essa estrutura e **mudar coisas em tempo real**, como texto, cores, tamanhos, adicionar ou remover elementos.

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Minha Página</title>
  </head>
  <body>
    <h1>Olá, mundo!</h1>
    <p>Este é um parágrafo.</p>
  </body>
</html>
```

```
Document
└── html
    ├── head
    │   └── title
    └── body
        ├── h1
        └── p
```

Seleção de Elementos no DOM

- Métodos para seleção de elementos:
  - getElementById
  - getElementsByClassName
  - querySelector
  - querySelectorAll

Seleção de Elementos - getElementById

- Seleciona um único elemento com o atributo id informado.
- Usado quando você sabe o id único do elemento que quer manipular.
- Retorna: um único elemento (ou null se não encontrar).

```
<p id="mensagem">Olá!</p>
<script>
  const elemento = document.getElementById("mensagem");
  elemento.textContent = "Olá, DOM!";
</script>
```

Seleção de Elementos - getElementsByClassName

- Seleciona **vários elementos** que têm a mesma **classe CSS**.
- Usado quando você quer alterar vários elementos com a mesma classe.
- Retorna**:** uma coleção (HTMLCollection) de elementos.

```
<p class="item">Item 1</p>
<p class="item">Item 2</p>
<script>
  const itens = document.getElementsByClassName("item");
  for (let item of itens) {
    item.style.color = "blue";
  }
</script>
```

Seleção de Elementos - getElementsByTagName

- Seleciona todos os elementos com uma determinada tag HTML, como **div, p**, **h1**, etc.
- Usado quando você quer alterar vários elementos com a mesma Tag.
- Retorna**:** uma HTMLCollection (parecida com array).

```
<p>Parágrafo 1</p>
<p>Parágrafo 2</p>
<script>
  const paragrafos = document.getElementsByTagName("p");
  paragrafos[0].textContent = "Alterado!";
</script>
```

Seleção de Elementos - querySelector

- Seleciona o primeiro elemento que bate com um seletor CSS (como .classe, #id, tag, ou combinados).
- Retorna: o primeiro elemento que encontrar (ou null se não houver nenhum).

```
<div class="caixa">Texto</div>
<script>
  const div = document.querySelector(".caixa");
  div.style.backgroundColor = "yellow";
</script>
```

Seleção de Elementos - querySelectorAll

- Seleciona todos os elementos que batem com um seletor CSS. Semelhante ao querySelector, mas retorna todos.
- Retorna: uma NodeList.

```
<li class="item">Item 1</li>
<li class="item">Item 2</li>
<script>
  const itens = document.querySelectorAll(".item");
  itens.forEach(item => {
    item.style.fontWeight = "bold";
  });
</script>
```

Manipulação de Elementos

Manipular elementos no DOM com JavaScript é uma das partes mais poderosas e divertidas do desenvolvimento frontend. Com isso, você pode **mudar o conteúdo, estilo, atributos, classes e até criar ou remover elementos da página dinamicamente**.

Exemplos:

- Mudar o conteúdo de um texto (**textContent**, **innerHTML**)
- Alterar estilos (**style**)
- Adicionar/remover classes (**classList**)
- Criar novos elementos (**createElement**)
- Remover elementos (**remove**, **removeChild**)
- Mudar atributos (**setAttribute**, **getAttribute**)

Manipulação de Elementos

- Alterar o conteúdo do elemento:

```
const titulo = document.getElementById("titulo");
titulo.textContent = "Novo título!"; // texto puro
// ou
titulo.innerHTML = "<em>Texto com ênfase</em>"; // permite HTML
```

- Alterar estilos diretamente:

```
const caixa = document.querySelector(".caixa");
caixa.style.backgroundColor = "lightblue";
caixa.style.fontSize = "20px";
```

- Adicionar e remover classes:

```
const botao = document.querySelector("button");

botao.classList.add("ativo");
botao.classList.remove("inativo");
botao.classList.toggle("destaque"); // adiciona se não tiver, remove se já tiver
```

Manipulação de Elementos

- Criar e inserir novos elementos:

```
const lista = document.getElementById("minhaLista");

const novoItem = document.createElement("li");
novoItem.textContent = "Item novo!";
lista.appendChild(novoItem); // adiciona no final da lista
```

- Remover elementos:

```
const pai = document.getElementById("minhaLista");
const filho = pai.querySelector("li");
pai.removeChild(filho); // remove o filho do pai
```

- Trabalhar com atributos:

```
const link = document.querySelector("a");

link.setAttribute("href", "https://google.com"); // muda o destino do link
const destino = link.getAttribute("href"); // pega o valor atual do atributo
```

Manipulação de Eventos

A manipulação de eventos no DOM é o que permite que páginas web sejam interativas, é como dizer ao JavaScript: "Quando o usuário clicar aqui, faça isso".

Eventos são **ações do usuário** ou do navegador, como:

- **click** – quando o usuário clica em algo
- **input** – quando digita em um campo
- **submit** – quando envia um formulário
- **keydown** – quando pressiona uma tecla
- **mouseover** – quando passa o mouse por cima
- **load** – quando a página ou imagem termina de carregar

Como ouvir eventos?

- Usamos o método addEventListener() para dizer:

  "Quando esse evento acontecer nesse elemento, execute essa função“

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

<script>
  const botao = document.getElementById("meuBotao");
  botao.addEventListener("click", () => {
    alert("Você clicou no botão!");
  });
</script>
```

Função de callback

- A função que é executada quando o evento acontece é chamada de callback.
- Você pode passar uma função anônima ou uma função nomeada:

```
function minhaFuncao() {
  console.log("Evento disparado");
}

element.addEventListener("click", minhaFuncao);
```

Objeto do evento

- Dentro da função de callback, você pode usar o objeto event para acessar informações do evento:

```
element.addEventListener("click", (event) => {
  console.log(event.target); // qual elemento foi clicado
});
```

Remover eventos

- Você também pode remover um evento com **removeEventListener**, mas só funciona com funções nomeadas:

```
function minhaFuncao() {
  console.log("Evento!");
}

botao.addEventListener("click", minhaFuncao);
botao.removeEventListener("click", minhaFuncao);
```

Exemplo prático

- Adicionar um listner no botão de submit do formulário de cadastro do de curso.
- Depois de clicado, mostrar alert de sucesso.