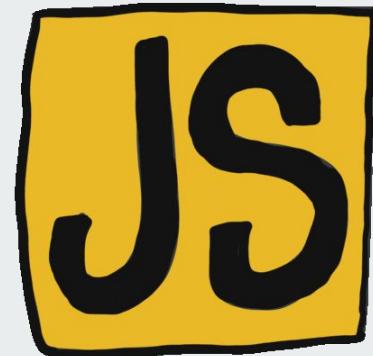
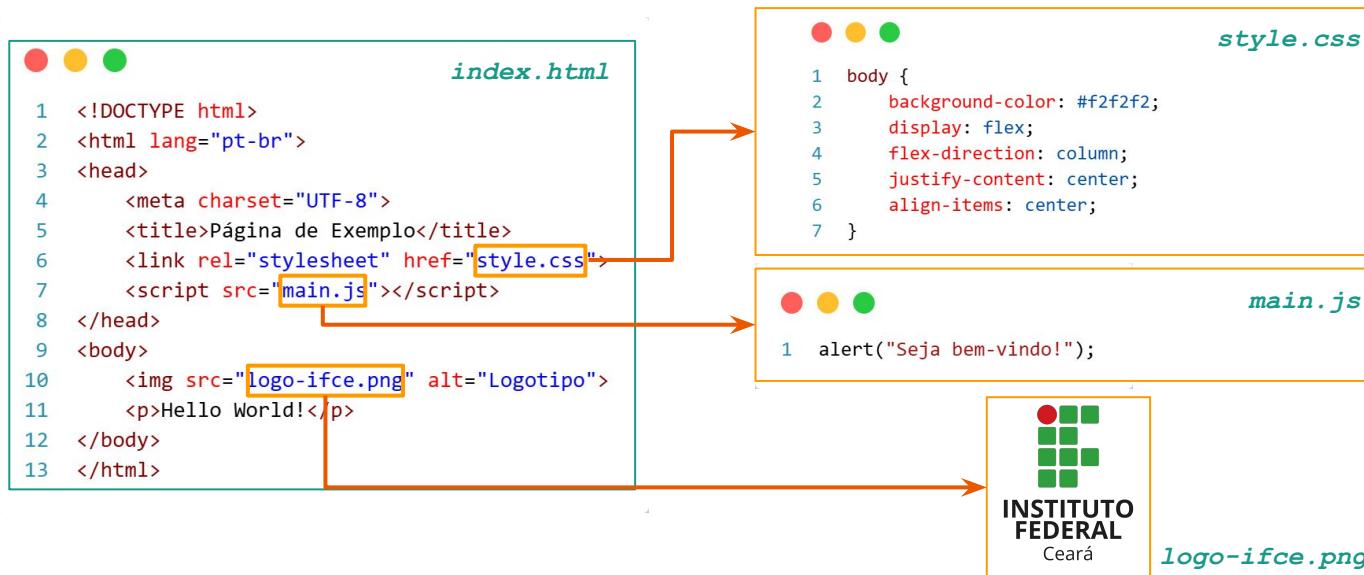

MANIPULAÇÃO DE DOM E EVENTOS





INTRODUÇÃO

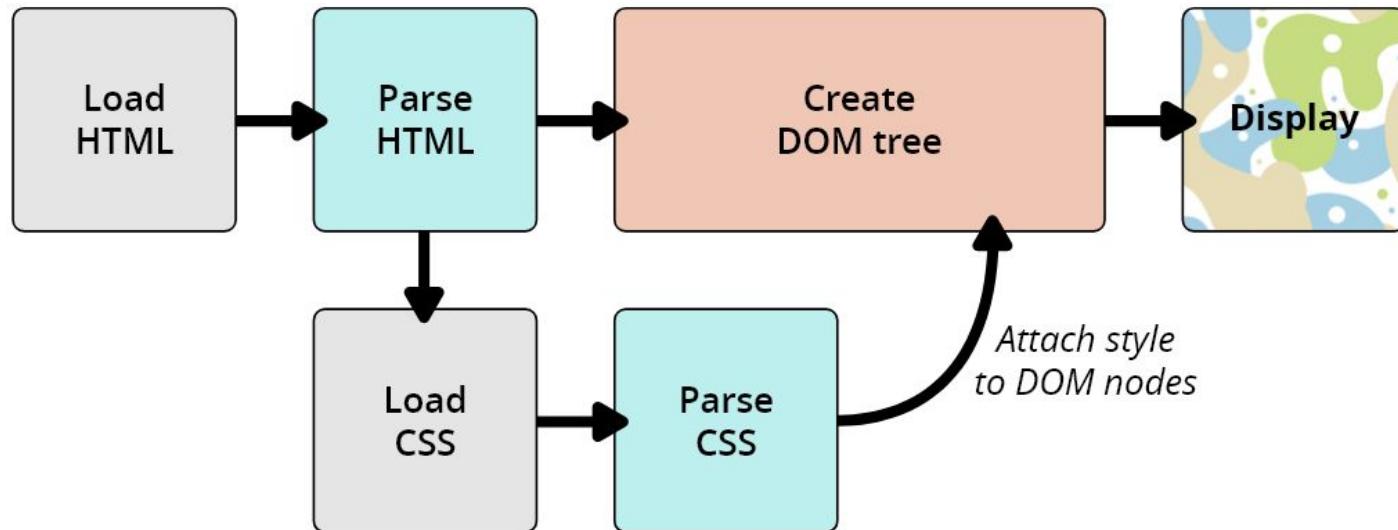
Relembrando a dinâmica de carregamento de um site web



Relembrando a dinâmica de carregamento de um site web



Relembrando a dinâmica de carregamento de um site web



Fonte: Mozilla Developer Network - https://developer.mozilla.org/en-US/docs/Learn_web_development/Getting_started/Web_standards/How_browsers_load_websites



DOCUMENT OBJECT MODEL

O que é DOM?

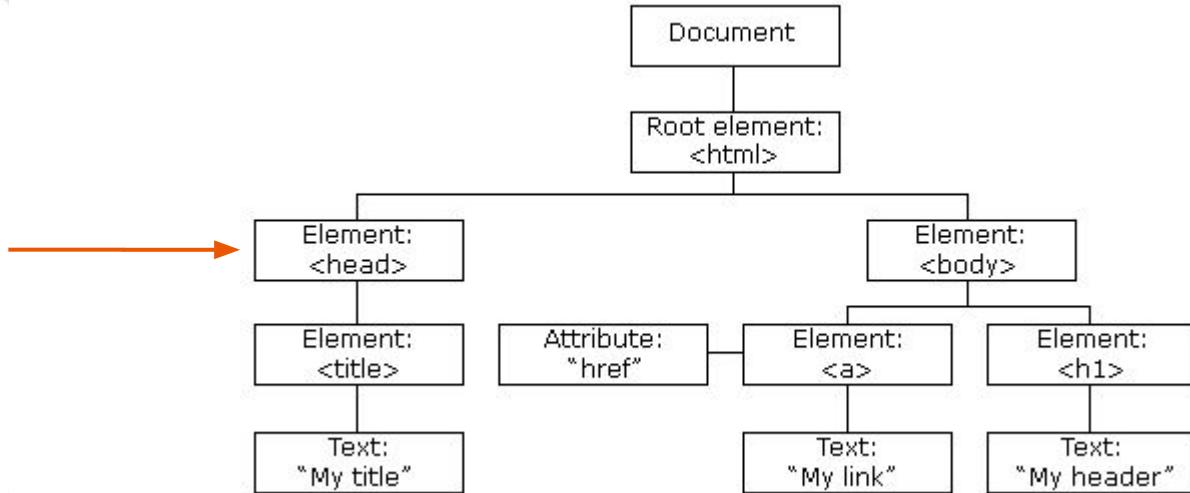
- *Document Object Model (DOM)* é a **representação de dados** dos **objetos** que compõem a **estrutura** e o **conteúdo** de um **documento** na Web
- DOM define ainda uma **API** para os documentos HTML que permite que os programas possam alterar a estrutura do documento (**Manipulação do DOM**)

Árvore de objetos

- O modelo DOM HTML de uma página é construído em formato de árvore de objetos



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <title>My Title</title>
5 </head>
6 <body>
7   <a href="#">My link</a>
8   <h1>My header</h1>
9 </body>
10 </html>
```



Por que manipular o DOM?

- **Atualizações dinâmicas**
 - Podemos alterar toda a estrutura de elementos do documento
- **Interatividade em páginas web**
 - Podemos definir ações que serão executadas quando algum evento ocorrer

Métodos e Propriedades do DOM

- **Seleção de Elementos**

- `document.getElementById(id)`
- `document.getElementsByTagName(nomeDaTag)`
- `document.querySelector(seletor)`
- `document.querySelectorAll(seletor)`

→ O método `querySelector()` retorna o primeiro elemento encontrado, enquanto o método `querySelectorAll()` retorna uma lista de todos os elementos que correspondem ao seletor especificado.

Métodos e Propriedades do DOM

- **Alteração de conteúdo (propriedades)**
 - `element.innerHTML` (retorna ou modifica o HTML interno do elemento)
 - `element.innerText` (retorna ou modifica o texto interno do elemento)
- **Alteração de estilos**
 - Adição/remoção de classes com `classList`
 - Adicionar uma classe: `element.classList.add("minhaClasse")`
 - Remover uma classe: `element.classList.remove("minhaClasse")`
 - Alternar a aplicação de uma classe: `element.classList.toggle("minhaClasse")`

Métodos e Propriedades do DOM

- **Alteração/definição de atributos**

- `element.attribute` (altera o valor de um atributo de um elemento)
 - Também pode ser utilizado o método `setAttribute(atributo, valor)`

- **Alteração de estilos com o atributo `style`**

- **Sintaxe:** `element.style.property = new style`
- **Exemplo:**



```
1 let btnAdicionar = document.getElementById("btn-add");
2
3 btnAdicionar.style.backgroundColor = "#363636";
4 btnAdicionar.style.color = "#FFFFFF";
```

Métodos e Propriedades do DOM

- Alteração/definição de atributos
 - `element.attribute` (altera o valor de um atributo de um elemento)
 - Também pode ser utilizado o método `setAttribute(atributo, valor)`
- Alteração de estilos com o atributo
 - **Sintaxe:** `element.style.property`
 - **Exemplo:**

→ Perceba que o nome da propriedade no DOM é o mesmo nome da propriedade CSS correspondente, porém ao invés de utilizar “-” para separar nomes compostos, aqui utilizamos o padrão camelCase.



```
1 let btnAdicionar = document.getElementById("btn-add");
2
3 btnAdicionar.style.backgroundColor = "#363636";
4 btnAdicionar.style.color = "#FFFFFF";
```

Métodos e Propriedades do DOM

- **Criando, adicionando e removendo elementos**
 - `document.createElement(element)` – criar um elemento HTML
 - `document.appendChild(element)` – adiciona um elemento HTML ao documento
 - `element.appendChild(element)` – adiciona um elemento HTML dentro do elemento selecionado
 - `document.removeChild(element)` – remove um elemento HTML do documento
 - `element.appendChild(element)` – adiciona um elemento HTML dentro do elemento selecionado



TRATAMENTO DE EVENTOS



Eventos em JavaScript

- **O que são eventos?**
 - Eventos são **ações ou ocorrências detectadas pelo navegador** enquanto interage com a página web.
 - Ações como cliques, teclas pressionadas, movimento do mouse, carregamento da página, finalização de uma requisição HTTP, etc
 - Eles permitem que os desenvolvedores capturem essas ações e executem funções específicas em resposta.



Eventos em JavaScript

- **Manipulando eventos:**
 - A manipulação de eventos em JavaScript envolve três etapas:
 1. Seleção do elemento DOM.
 2. Definição de um “ouvinte de evento”: utiliza-se o método `addEventListener()` para associar uma função ao evento de um determinado tipo.
 - Pode-se ainda, definir um manipulador de evento usando manipuladores inline (atributos especiais no próprio elemento HTML, como: `onclick`, `onkeydown`, `onsubmit`...)
 3. Execução de uma **função callback**: a função associada é executada quando o evento ocorre.

Eventos em JavaScript

- Adicionando manipulador de evento com **addEventListener()**
 - O método `addEventListener()` recebe o tipo de evento a ser monitorado e uma função que será executada quando o evento ocorrer:

- **Sintaxe:** `element.addEventListener(event, function)`

- **Exemplo:**



```
1 let botao = document.getElementById("btn");
2
3 botao.addEventListener("click", clickBotao);
4
5 function clickBotao() {
6     console.log("O botão foi clicado.");
7 }
```



PRÁTICAS

Aprendendo na prática

- **Projeto 1: To-Do List**

- Vamos implementar um projeto simples de uma aplicação **To-Do List** (Lista de Tarefas)
- A figura ao lado mostra como deve ficar nossa aplicação
- Roteiro para a prática:

<https://github.com/lucas-lfm/web1-2025.2-base/tree/main/atividades/pratica-dom-01>



Desafio

- **Projeto 2: Galeria de Imagens**
 - Vamos implementar um projeto simples de uma galeria de imagens dinâmica.
 - A figura ao lado mostra como deve ficar o resultado final.
 - Ao clicar em uma miniatura de uma imagem, ela deve ser apresentada na área de destaque.
 - Perceba que as demais miniaturas (que não estão selecionadas), devem ser apresentadas em preto e branco.

