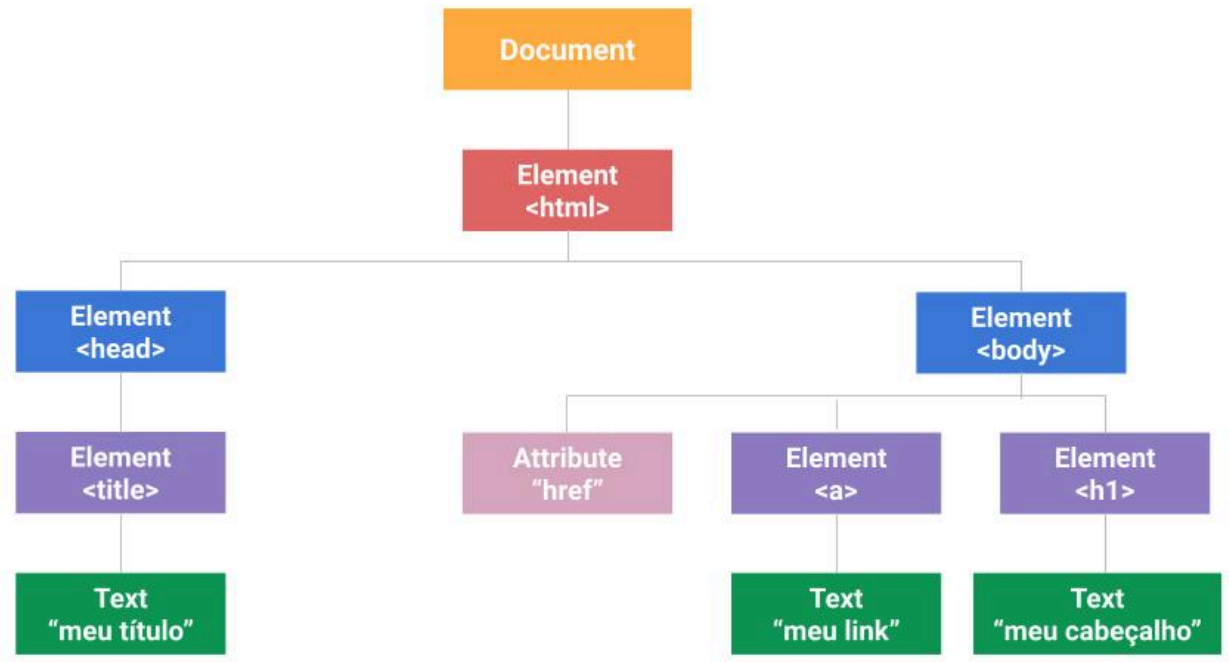
Manipulação de DOM

Link Notion: https://cherry-client-b8f.notion.site/Manipula-o-de-DOM-11b911d84e0d8074b3c9f88be820405b?pvs=74

O que é o DOM?

O DOM (Document Object Model) é a representação de dados dos objetos que compõem a estrutura e o conteúdo de um documento na Web.



Utilizando o node Window

O window pode ser muito útil para manipularmos ações baseadas nos recursos e informações que o navegador nos oferece.

Executar funções ao carregar a página

Podemos chamar funções no momento que a página é carregada com **onload** ou **DOMContentLoaded**.

```
//Exemplo com DOMContentLoaded
document.addEventListener('DOMContentLoaded', function() {
    console.log("O DOM foi completamente carregado e analisado.");
});

//Exemplo com onload
window.onload = function() {
    console.log("A página inteira, incluindo imagens e outros recursos, foi carregada.");
};
```

DOMContentLoaded

- Disparado quando o HTML foi completamente carregado e analisado, antes do carregamento de imagens, folhas de estilo, iframes, etc.
- Ideal quando você quer manipular ou acessar elementos do DOM assim que eles estão prontos.

onload

- Disparado somente quando toda a página foi carregada, incluindo imagens, estilos, scripts externos etc.
- Útil se você precisa garantir que todos os recursos da página estão disponíveis.

Popups

Podemos utilizar alert, confirm ou prompt por exemplo para mostrar janelas para o usuário.

Ex:

```
alert('Olá mundo') //exibe uma janela com uma mensagem na tela
prompt('Digite seu nome') //exibe uma janela com um campo de digitação
confirm('Você quer mesmo sair?') //exibe uma janela de confirmação
```

Redirecionamento

Podemos utilizar para redirecionar o usuário para alguma página específica.

Ex:

```
window.location.href = 'https://www.example.com'; // Redireciona para um novo site
```

Navegar pelo histórico

Também pode ser utilizado para fazer um botão de voltar ou avançar.

Ex:

```
window.history.back();  // Volta para a página anterior
window.history.forward(); // Avança para a próxima página
```

Recarregar a página

Pode ser utilizado para recarregar a página.

Ex:

```
window.location.reload(); // Recarrega a página atual
```

Fazer ações baseadas em tempo

Podemos utilizar *setTimeOut* ou *setInterval* para executar ações depois de um determinado tempo ou repetir ações com uma frequência de tempo.

Ex:

```
//Executa uma ação depois de determinado tempo
setTimeout(function() {
    console.log('Executado após 3 segundos');
}, 3000);

//Repete uma ação a cada x segundos
setInterval(function() {
    console.log('Executado a cada 2 segundos');
}, 2000);
```

Como eu acesso esses elementos via JS?

Nós temos 3 métodos principais:

document.getElementById(*id*) - Encontra um elemento no HTML pelo atributo id.

Ex:

```
const element = document.getElementById('meuElemento'); // Pega o elemento com ID "meuElemento"
```

document.getElementsByTagName(tag) - Encontra um elemento no HTML pela tag.

Ex:

```
const paragraphs = document.getElementsByTagName('p');
```

document.getElementsByClassName(classe) - Encontra um elemento no HTML pelo atributo class.

Ex:

```
const elements = document.getElementsByClassName('myClass');
```

Porém existem duas formas que são mais modernas e permitem utilizar seletores CSS.

document.querySelector(seletor CSS) - Encontra o primeiro elemento baseado no seletor CSS passado entre o parênteses.

Ex:

```
const element = document.querySelector('seletor-css');
```

document.querySelectorAll(seletor CSS) - Encontra todos os elementos baseados no seletor CSS passado entre os parênteses.

Ex:

```
const destaques = document.querySelectorAll('.destaque');

destaques.forEach(elemento => {
  elemento.classList.add("negrito");
});
```

Como eu capturo o valor de elementos de formulário?

Para capturar valores digitados pelo usuários, temos que acessar o atributo **value** em boa parte dos casos.

```
//HTML
//<input type="text" id="nome" />

const nome = document.querySelector("#nome").value;
console.log(nome);
```

Em alguns casos como Radio Buttons e Checkbox, temos que acessar de maneiras diferentes. O radio button, acessamos através do atributo **name, verificando a propriedade checked**.

Radio Button

```
//HTML
//<input type="radio" name="genero" value="masculino" /> Masculino
//<input type="radio" name="genero" value="feminino" /> Feminino


const genero = document.querySelector('input[name="genero"]:checked').value;
console.log(genero);
```

Já o checkbox, capturamos através da propriedade **checked, que retorna true ou false**.

Checkbox

```
//HTML
//<input type="checkbox" id="aceito" />

const aceito = document.querySelector("#aceito").checked;
console.log(aceito); // true ou false
```

Como alterar os elementos?

Com a manipulação de DOM nós conseguimos alterar todos os elementos do site de forma dinâmica via JS.

Alterando os textos

Nós temos 3 formas de inserir ou alterar um texto:

elemento.**textContent** - Retorna ou altera o conteúdo em um elemento HTML, afetando todos os elementos filho.

Ex:

```
const element = document.querySelector('#example');
console.log(element.textContent);
element.textContent = "Novo conteúdo de texto!";
```

elemento.**innerText** - Retorna ou altera o conteúdo em um elemento HTML, afetando todos os elementos filho, **porém respeitando a formatação do CSS**.

Ex:

```
const element = document.querySelector('#example');
console.log(element.innerText);
element.innerText = "Texto visível atualizado!";
```

elemento.**innerHTML** - Retorna ou altera o conteúdo em um elemento HTML, afetando todos os elementos filho, **porém respeitando a formatação do CSS e as tags HTML**.

Ex:

```
const element = document.querySelector('#example');
console.log(element.innerHTML);
element.innerHTML = "<p>Este é um <strong>novo</strong> parágrafo!</p>";
```

Alterando os atributos

Nós também podemos alterar os atributos de uma tag HTML via manipulação de DOM, como por exemplo o href de um link ou o src de uma imagem.

elemento.**atributo** - Altera o valor de um atributo existente.

Ex:

```
// Seleciona um botão pelo ID
const button = document.getElementById('meuBotao');

// Modifica diretamente o atributo 'disabled' (desativa o botão)
```

```
button.disabled = true;

// Altera diretamente o valor de 'id'
button.id = 'novoID';
```

elemento.**setAttribute(atributo, valor)** - Altera ou insere um novo atributo no elemento selecionado.

Ex:

```
// Seleciona uma imagem pelo ID
const img = document.getElementById('minhaImagem');

// Altera o atributo 'alt' da imagem
img.setAttribute('alt', 'Descrição da nova imagem');
```

Alterando o CSS

Nós também pode alterar o CSS de forma dinâmica, de acordo com as interações do usuário.

elemento.style.propriedade - altera a propriedade CSS do elemento selecionado.

Ex:

```
// Estrutura do Comando
element.style.propriedadeCSS = 'valor';

// Seleciona um parágrafo pelo ID
const paragraph = document.getElementById('meuParagrafo');

// Altera o estilo diretamente
paragraph.style.color = 'blue'; // Altera a cor do texto para azul
paragraph.style.fontSize = '20px'; // Altera o tamanho da fonte
paragraph.style.backgroundColor = 'lightyellow'; // Altera a cor de fundo
```

Também é possível, adicionar, remover ou alterar entre classes usando o elemento classList.

elemento.classList.add - Adiciona uma classe ao elemento selecionado.

elemento.classList.remove - Remove uma classe do elemento selecionado.

elemento.classList.toggle - Alterna entre adicionar e remover uma classe no elemento.

Ex:

```
// Estrutura do Comando
element.style.propriedadeCSS = 'valor';
element.classList.add('classe1', 'classe2');
element.classList.remove('classe1');
element.classList.toggle('classe1');

//Exemplo

// Seleciona um botão pelo ID
const button = document.getElementById('meuBotao');

// Adiciona uma classe 'ativo'
button.classList.add('ativo');
```

```
// Remove uma classe 'desabilitado'
button.classList.remove('desabilitado');

// Alterna uma classe 'destaque' (adiciona se não estiver presente, remove se estiver)
button.classList.toggle('destaque');
```

Como criar novos elementos?

Para criar novos elementos, nós utilizaremos um método chamado createElement, que cria um novo elemento a ser adicionado ao HTML posteriormente.

document.createElement(tag) - Cria um novo elemento HTML que normalmente é acesso por uma variável.

Ex:

```
// Estrutura do Comando
const novoElemento = document.createElement('tag');

//Exemplo
const novoParagrafo = document.createElement('p'); // Cria um novo elemento <p>
```

Como adicioná-lo no HTML

Para adicioná-lo ao HTML, temos diferentes formas:

elementoPai.appendChild - Insere o novo elemento como o **último filho** do elemento pai.

Ex:

```
const parent = document.getElementById('container');
const newElement = document.createElement('div');
newElement.textContent = "Este é o novo filho";
parent.appendChild(newElement);
```

elementoPai.append - Insere o novo elemento como o **último filho** do elemento pai, mas aceitando texto e múltiplos elementos como argumentos separados por vírgula.

Ex:

```
const parent = document.getElementById('container');
const newElement = document.createElement('div');
newElement.textContent = "Este é o novo filho";
parent.append(newElement, " - texto extra");
```

elementoPai.prepend - Insere o novo elemento como o **primeiro filho** do elemento pai.

Ex:

```
const parent = document.getElementById('container');
const newElement = document.createElement('div');
newElement.textContent = "Este é o novo primeiro filho";
parent.prepend(newElement);
```

elementoPai.insertBefore - Insere o novo elemento **antes de um elemento filho específic**o.

Ex:

```
const parent = document.getElementById('container');
const referenceElement = document.getElementById('existingChild');
const newElement = document.createElement('div');
newElement.textContent = "Este é inserido antes do filho existente";
parent.insertBefore(newElement, referenceElement);
```

elementoPai.replaceChild - Substitui um elemento filho existente por um novo elemento.

Ex:

```
const parent = document.getElementById('container');
const oldChild = document.getElementById('oldChild');
const newElement = document.createElement('div');
newElement.textContent = "Este substitui o filho antigo";
parent.replaceChild(newElement, oldChild);
```

elementoSelecionado.insertAdjacentElement - Insere o novo elemento em uma posição específica relativa a outro elemento no DOM. Posições possíveis:

"beforebegin": Antes do elemento
"afterbegin": Como o primeiro filho
"beforeend": Como o último filho
"afterend": Após o elemento

Ex:

```
const referenceElement = document.getElementById('existingElement');
const newElement = document.createElement('div');
newElement.textContent = "Este é inserido após o elemento";
referenceElement.insertAdjacentElement('afterend', newElement);
```

Eventos

Para adicionar eventos como clique de mouse, acionamento de teclas, utilizaremos o addEventListener, que recebe como parâmetro o evento que deve acontecer e a função que será ativada quando esse evento acontecer.

Ex:

```
button.addEventListener('click', function () {
    // Exibe o tipo de evento (ex: 'click')
    console.log('Clicou!');
}
```

Também é possível adicionar múltiplos eventos à um mesmo elemento.

Ex:

```
        // Evento de clique
        button.addEventListener('click', function() {
            console.log('Botão clicado!');
        });

        // Evento de mouse sobre o botão
```

```
    button.addEventListener('mouseover', function() {
        console.log('Mouse sobre o botão!');
        // Muda a cor de fundo do botão
        button.style.backgroundColor = 'lightgreen';
    });

    // Evento de mouse sair do botão
    button.addEventListener('mouseout', function() {
        console.log('Mouse saiu do botão!');
        // Restaura a cor de fundo do botão
        button.style.backgroundColor = 'lightcoral';
        });
```

Acessando as informações de um evento

Todo evento recebe como parâmetro um objeto “event” que possui todas as informações sobre aquele evento e com isso conseguimos manipular as informações, como por exemplo saber qual tecla ou botão do mouse foi pressionado e assim disparar uma ação.

```
//Exemplo detectando a tecla Esc pressionada:

document.addEventListener('keydown', function(event) {
    if (event.key === 'Escape') {
        console.log('Você pressionou a tecla Esc!');
    }
});
```

Referências

https://www.w3schools.com/js/js_htmldom_navigation.asp

https://www.toptal.com/developers/keycode