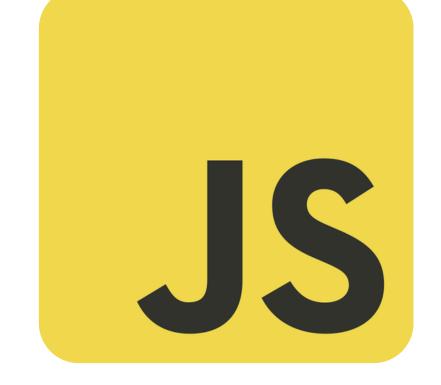
# JS para Desenvolvimento WEB





## Aula 2

Manipulação de DOM

## DOM

(Document Object Model)

Estrutura em forma de árvore criada pelo navegador para acessar e manipular o HTML da página.

Permite a manipulação do HTML carregado pelo próprio navegador e pelo Javascript.

```
<!doctype html>
<html lang="en-US">
  <head>
    <meta charset="utf-8" />
    <title>Exemplo de DOM</title>
  </head>
  <body>
    <section>
      <img
        src="imagem.png"
        alt="texto alt da imagem" />
      >
        Texto do parágrafo
        <a href="https://">Link para uma página da web</a>
      </section>
  </body>
  html>
```

```
DOCTYPE: html
└HTML lang="en-US"
 _HEAD
   -#text:
   -META charset="utf-8"
   -#text:
   -TITLE
    #text: Exemplo de DOM
   #text:
  -#text:
 -BODY
   -#text:
   -SECTION
    -#text:
     - IMG src="imagem.png" alt="texto alt da imagem"
     #text:
      -#text: Texto do parágrafo
      -A href="https://"
        ∟#text: Link para uma página da web
      #text:
     #text:
    #text:
```

https://software.hixie.ch/utilities/js/live-dom-viewer/

## DOM

Cada entrada na árvore se chama Node: HTML, HEAD, META, BODY, A, P, ...

- Root Node: HTML
- Child Node: Node 1 nível abaixo de outro. Por exemplo, IMG e P são "filhos" de SECTION.
- Parent Node: Node 1 nível **acima** de outro. Por exemplo, P é "pai" de A

#### Acessar Elementos

\_\_\_\_\_

O Javascript pode acessar o DOM através do objeto Document.

document.querySelector(<tag de busca>)

 Recebe como argumento uma string (tag) e encontra o primeiro elemento com aquela tag. Por exemplo:

document.querySelector("a");

Retorna um elemento.

#### Acessar Elementos

document.querySelectorAll(<tag de busca>)

• Recebe como argumento uma string (tag) e encontra **todos** os elemento com aquela tag. Por exemplo:

document.querySelectorAll("a");

Retorna uma lista de elementos.

#### Acessar Elementos

document.getElementById(<Id de busca>)

Recebe como argumento uma string (Id) e encontra o elemento com aquele
 Id. Por exemplo:

document.getElement ById("myId1");

Retorna um elemento.

### Modificar Elementos

É possível **modificar** atributos dos elementos do DOM:

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

link.textContent = "Novo texto visível do link";

link.href = "https://nova.url.do.link";

#### Criar Elementos

\_\_\_\_\_

É possível criar novos elementos para o DOM:

```
const para = document.createElement("p");
para.textContent = "Esse  é novo!";
```

Em seguida precisamos **anexar** o elemento criado a um elemento que já está no DOM:

```
const sect = document.querySelector("section");
    sect.appendChild(para);
```

#### Remover Elementos

\_\_\_\_\_

Também é possível **remover** elementos do DOM:

Se sabemos o "parent" do elemento:

```
const sect = document.querySelector("section");
const para = document.querySelector("p");
```

sect.removeChild(para);

#### Remover Elementos

• Se **não** sabemos o "parent" do elemento:

```
const para = document.querySelector("p");
    para.remove();
```

ou

para.parentNode.removeChild(para);

## Alterar CSS

É possível editar atributos CSS de elementos:

```
const para = document.querySelector("p");
```

```
para.style.color = "white";
para.style.backgroundColor = "black";
para.style.textAlign = "center";
```

Ou adicionar/remover classes a eles:

```
para.classList.add("class1");
para.classList.remove("class2");
```

## Event Listeners

É possível adicionar Event Listeners a elementos do DOM:

Um Event Listener **chama** uma **função** quando um certo **evento** o dispara (normalmente interações de usuário com algum elemento).

```
const input = document.querySelector('input');
const button = document.querySelector('button');
```

```
button.addEventListener('click', () => { /* código */ });
input.addEventListener('keydown', () => { /* código */ });
```

## Exemplo

Lista de compras

https://github.com/DeivisFelipe/Minicurso-JS

## Exercício

\_\_\_\_\_

Visualizador de cor em hexadecimal

https://github.com/DeivisFelipe/Minicurso-JS