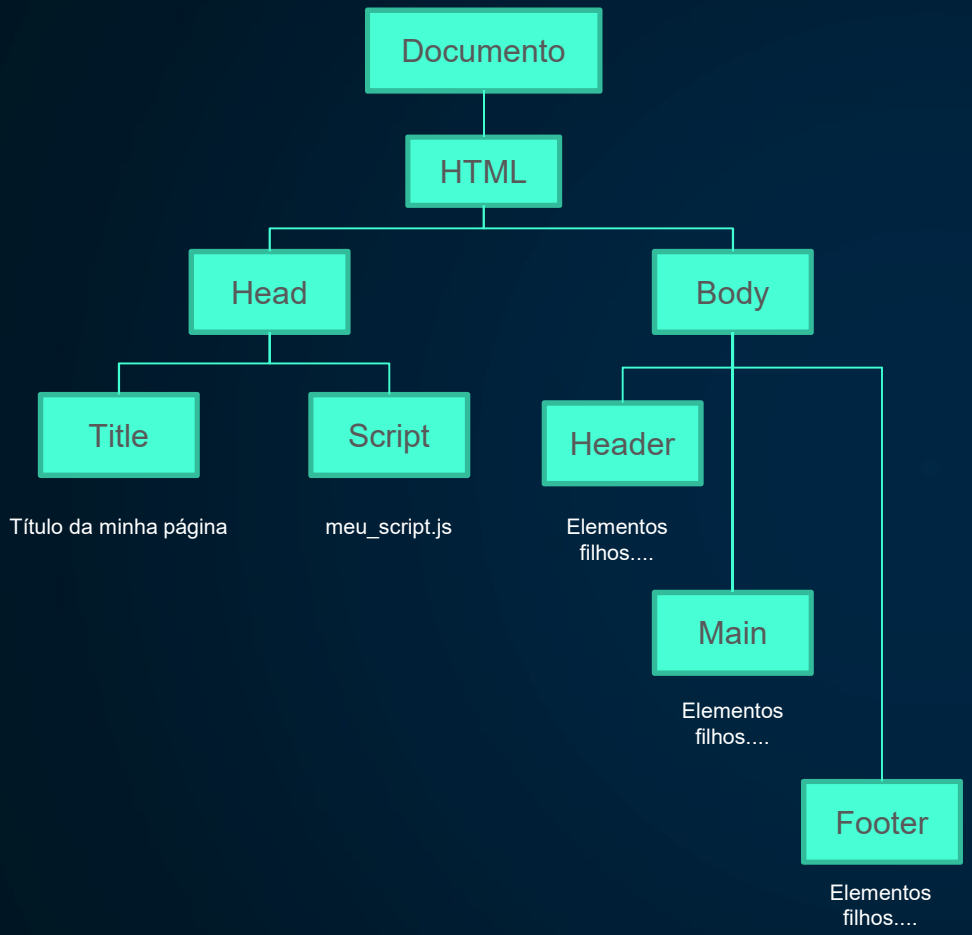
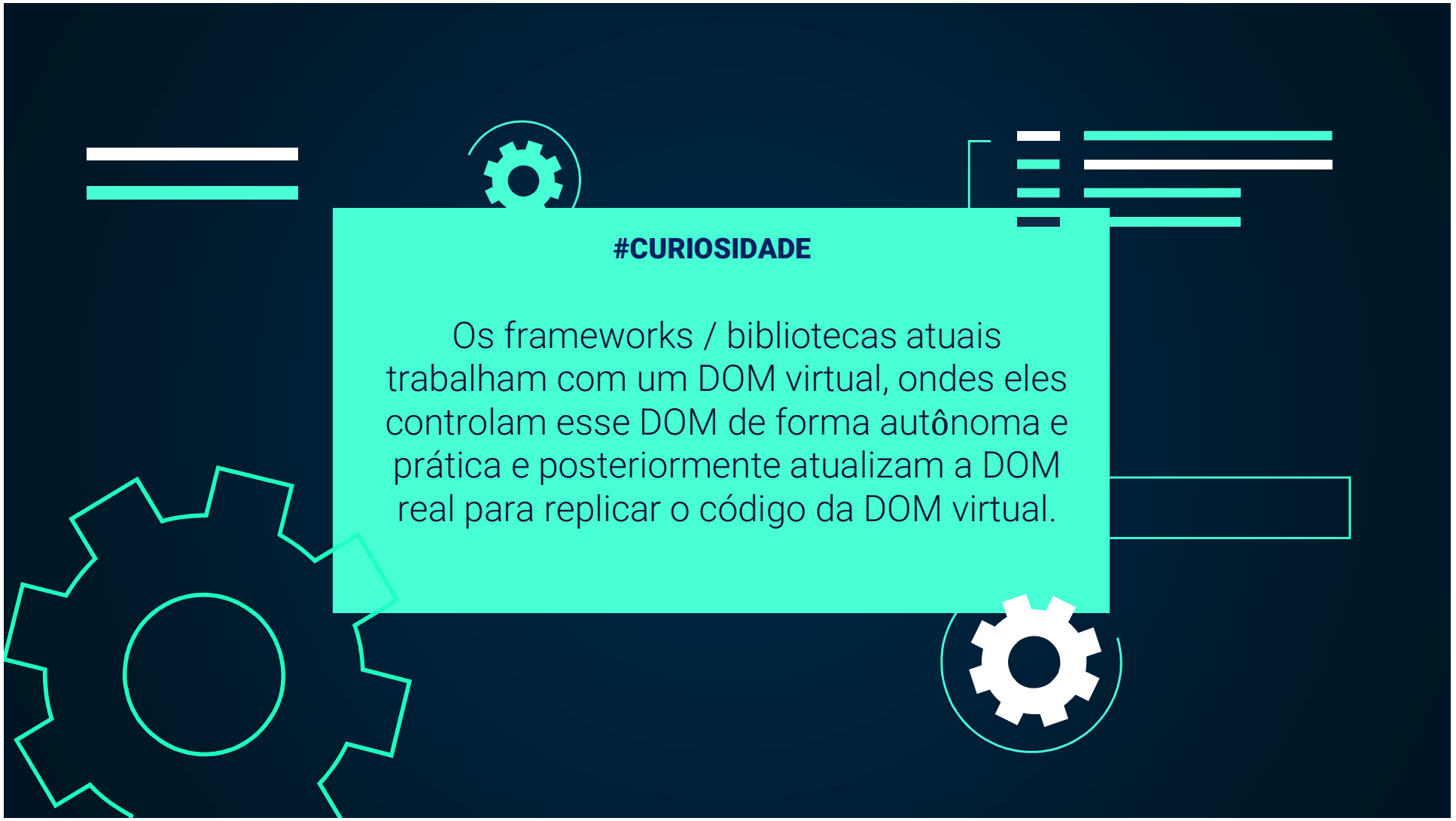

# Manipulação do DOM com JS

# SUMÁRIO




# 1. DOM

O Document Object Model ( DOM ) conecta páginas da Web a scripts ou linguagens de programação representando a estrutura de um documento.

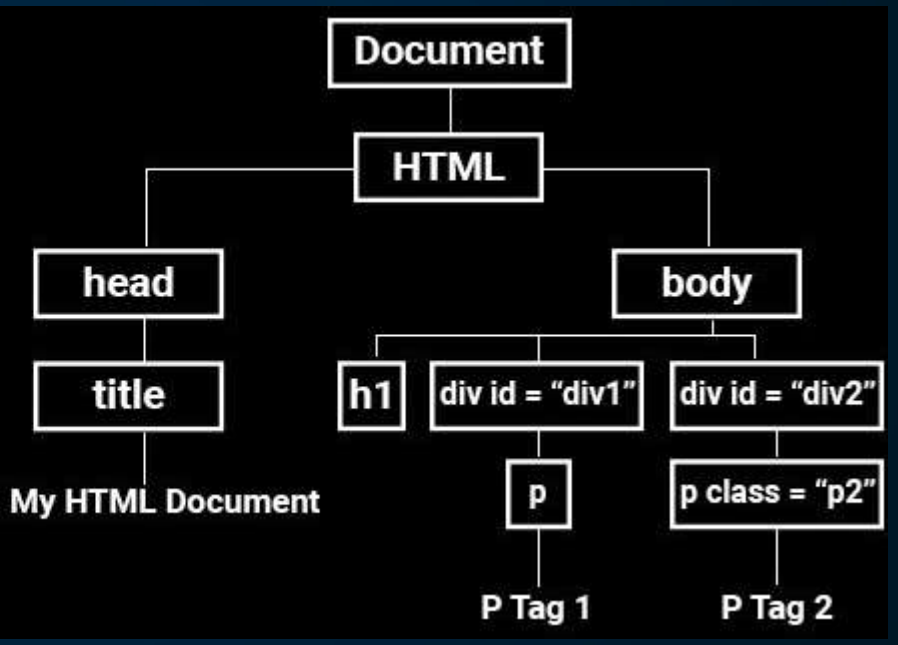

#CURIOSIDADE
Os frameworks / bibliotecas atuais trabalham com um DOM virtual, ondes eles controlam esse DOM de forma autônoma e prática e posteriormente atualizam a DOM real para replicar o código da DOM virtual.

# 1. DOM

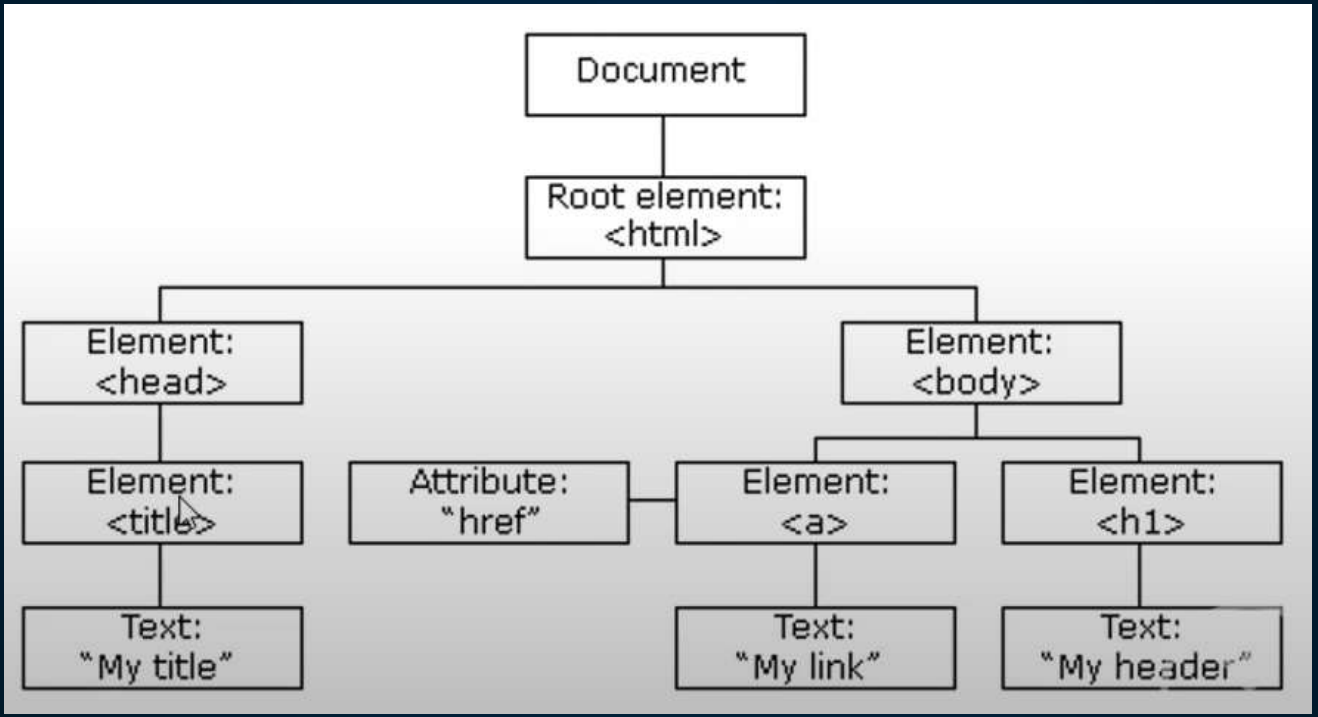
index.html

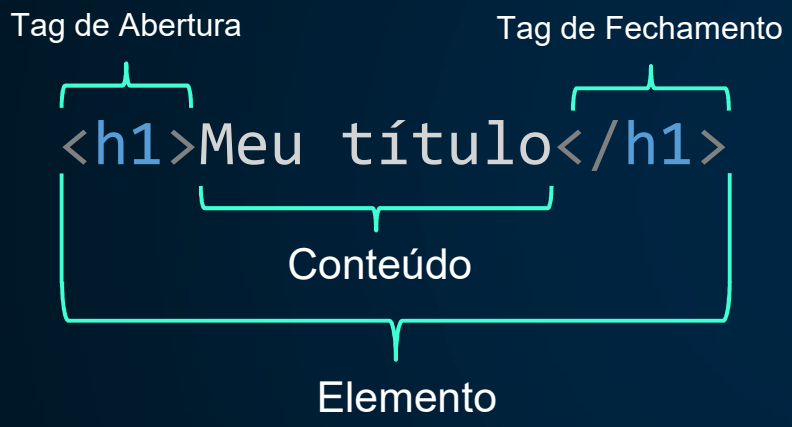
```html
<html>
    <head>
        <title>My HTML Document</title>
    </head>

    <body>
        <h1>Heading</h1>
        <div id="div1">
            <p>P Tag 1</p>
        </div>
        <div id="div2">
            <p class="p2">P Tag 2</p>
        </div>
    </body>
</html>
```



# 1. DOM

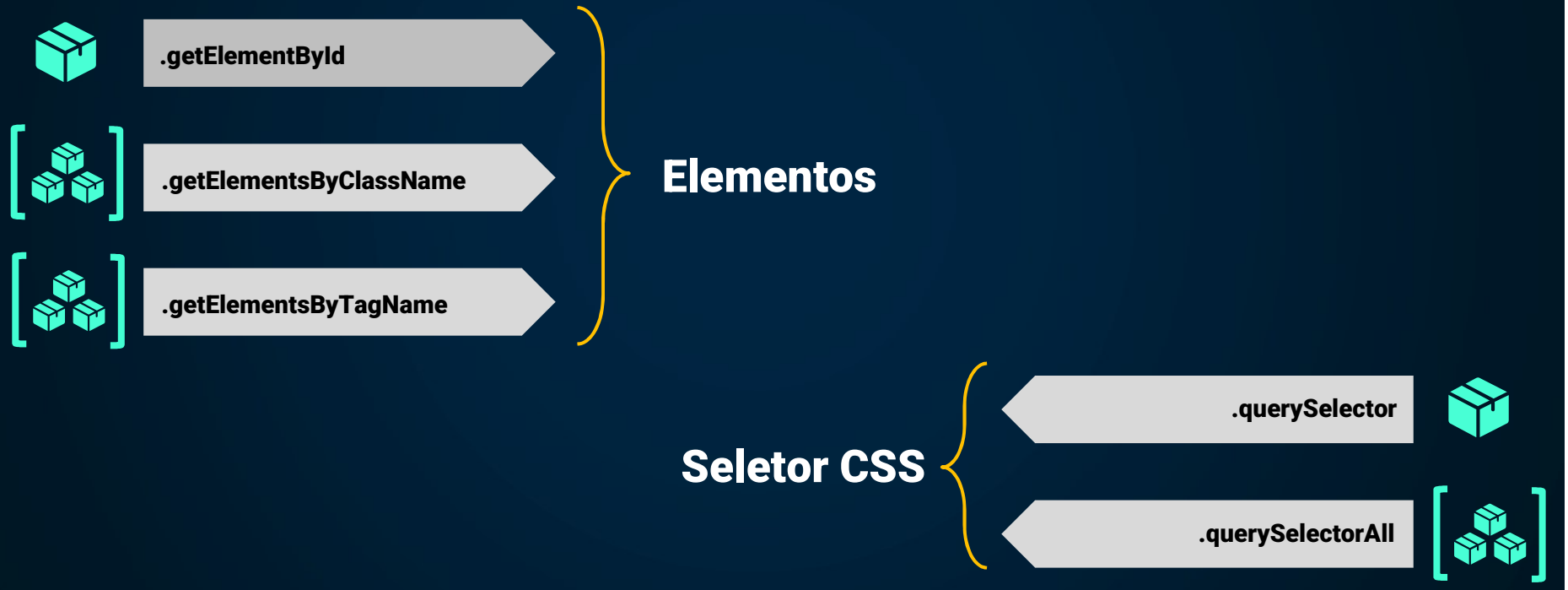

Document
Root element:
<html>
Element:
<head>
Element:
<body>
Element:
<title>
Attribute:
"href"
Element:
<a>
Element:
<h1>
Text:
"My title"
Text:
"My link"
Text:
"My header"



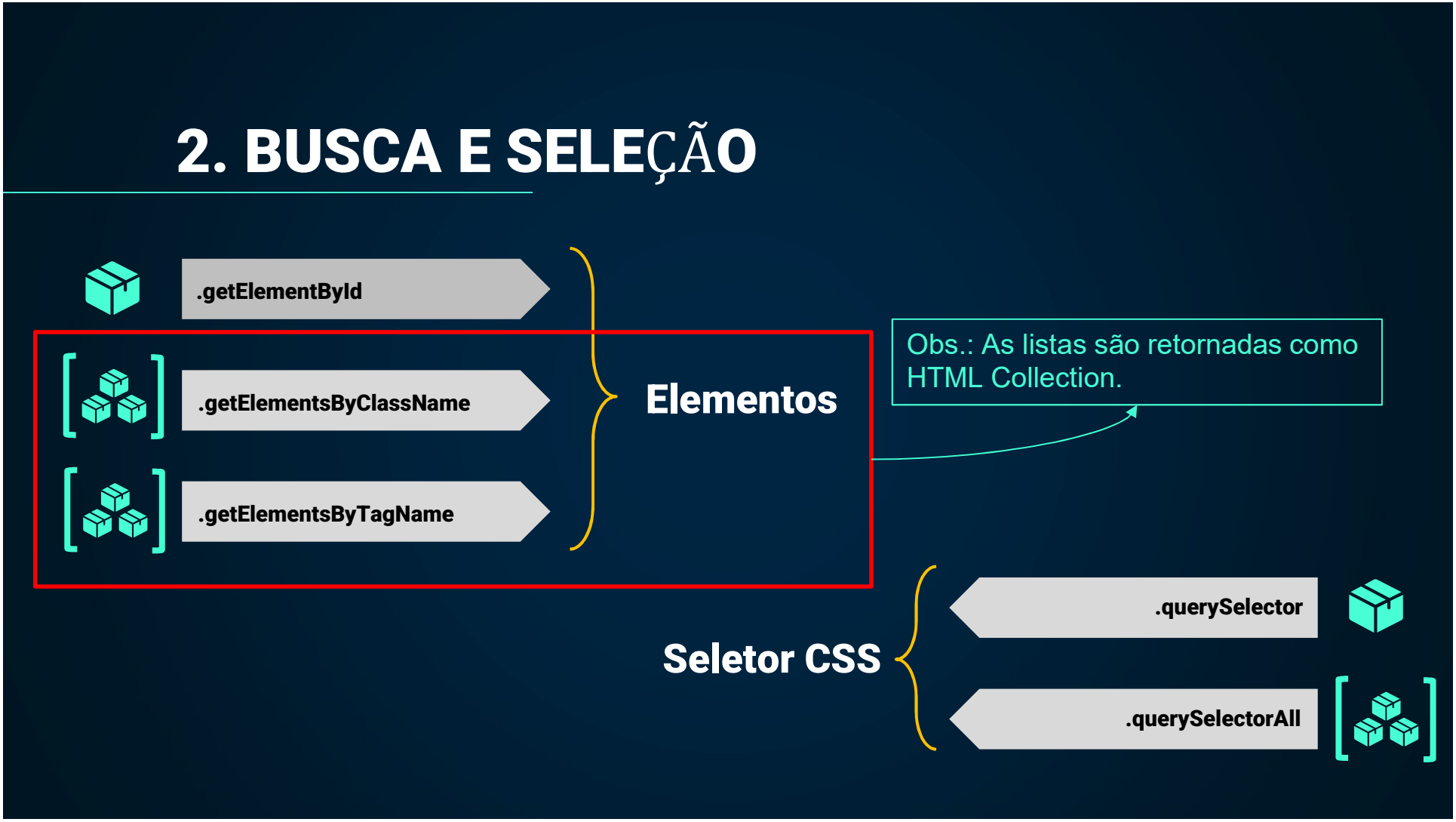

# 2. BUSCA E SELEÇÃO

Para manipularmos um elemento, primeiro é necessáro localizar esse elemento no DOM e esse processo é feito através de métodos que buscam e selecionam um ou mais elementos / nós.

# 2. BUSCA E SELEÇÃO



# 2. BUSCA E SELEÇÃO



# 2. BUSCA E SELEÇÃO

**.getElementById** Retorna a referência do elemento através do seu ID

## Sintaxe

```
var elemento = document.getElementById(id);
```

## Exemplo

```
<!DOCTYPE html>
<html>
<head>
  <title>Exemplo getElementById</title>
  <script>
  function mudarCor(novaCor) {
    var elemento = document.getElementById("para1");
    elemento.style.color = novaCor;
  }
  </script>
</head>
<body>
  <p id="para1">Algum texto de exemplo</p>
  <button onclick="mudarCor('blue');">Azul</button>
  <button onclick="mudarCor('red');">Vermelho</button>
</body>
</html>
```

https://developer.mozilla.org/pt-BR/docs/Web/API/Document/getElementById

# 2. BUSCA E SELEÇÃO

**.getElementsByClassName**

Retorna um vetor de objetos com todos os elementos filhos que possuem o nome da classe dada.

## Sintaxe

```
var elementos = document.getElementsByClassName(nomes); // ou:
var elementos = rootElement.getElementsByClassName(nomes);
```

## Exemplo

```
1  //Retorna todos os elementos que possuem a classe 'teste'
2  document.getElementsByClassName('teste');
3
4  //Retorna todos os elementos que possuem as classes 'vermelho' e 'teste'
5  document.getElementsByClassName('vermelho teste');
6
7  //Retorna todos os elementos que possuem a classe 'teste' dentro do elemento que possui o ID 'principal'
8  document.getElementById('principal').getElementsByClassName('teste');
```

https://developer.mozilla.org/pt-BR/docs/Web/API/Document/getElementsByClassName

# 2. BUSCA E SELEÇÃO

**.getElementsByTagName**

Retorna um vetor de objetos com todos os elementos filhos que possuem o nome de tag fornecido .

## Sintaxe

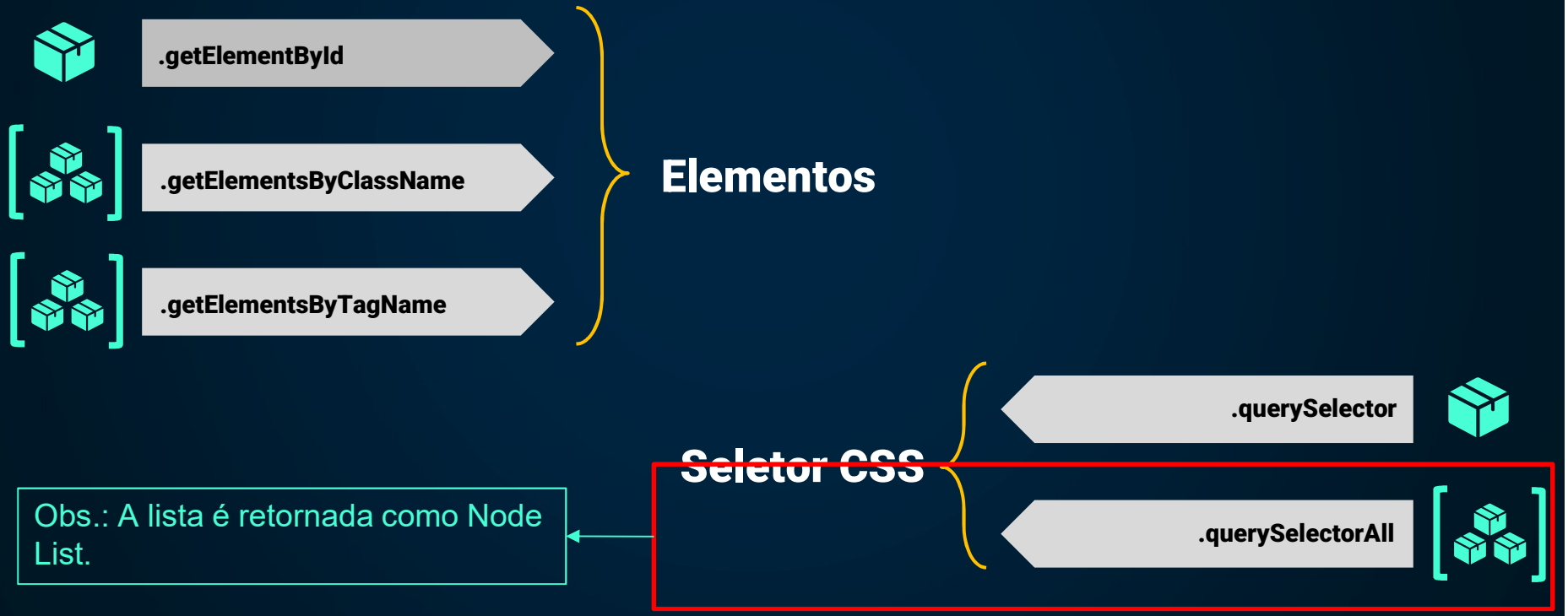
```
var elemento = document.getElementsByTagName(tag);
```

## Exemplo

```
1  // Retorna todos os elementos que possuem a tag "h1"
2  document.getElementsByTagName("h1")
3
4  // Retorna todos os elementos que possuem a tag "div"
5  document.getElementsByTagName("div")
```

https://developer.mozilla.org/en-US/docs/Web/API/Element/getElementsByTagName

# 2. BUSCA E SELEÇÃO



# 2. BUSCA E SELEÇÃO

.querySelector

Retorna o primeiro elemento dentro do documento (usando ordenação em profundidade, pré-ordenada e transversal dos nós do documento) que corresponde ao grupo especificado de seletores.

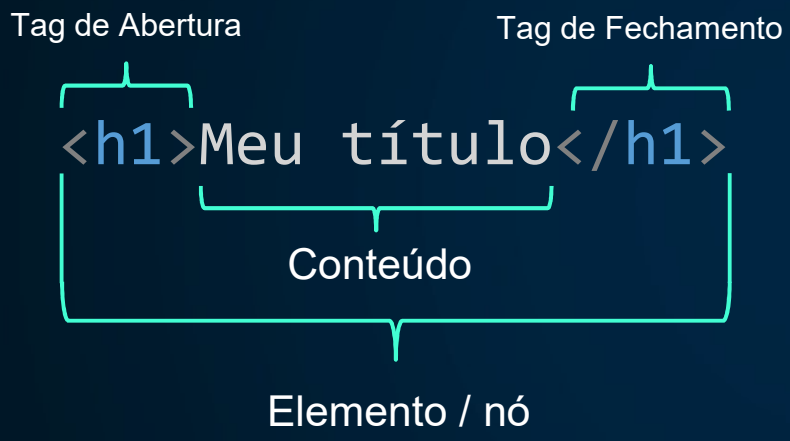
## Sintaxe

```
var elemento = document.queryselector(css_selector);
```

## Exemplo

```
1  // Retorna todos os elementos que possuem a tag "h1"
2  document.getElementsByTagName("h1")
3
4  // Retorna todos os elementos que possuem a tag "div"
5  document.getElementsByTagName("div")
```

https://developer.mozilla.org/en-US/docs/Web/API/Element/getElementsByTagName



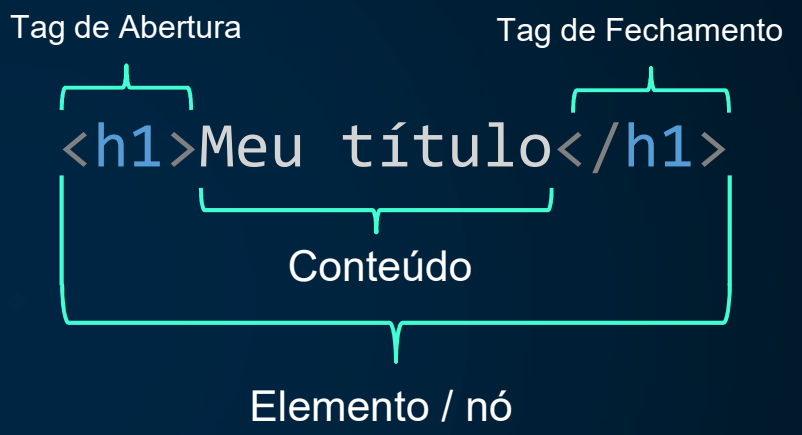

# 3. CONTEÚDO

Existem várias formas de manipularmos o conteúdo de um elemento / nó.

# 3. CONTEÚDO

- .textContent
- .innerText
- .innerHTML
- .value (inputs)

## Tag




## Input de formulário

conteúdo

# 3. CONTEÚDO

**.textContent**

Retorna o texto do elemento selecionado e seus descendentes (filhos).

## Sintaxe

```
elemento.textContent
```

## Exemplo

Dado o código HTML e atribuindo ele à um elemento com um método de busca e seleção.

```
<h1>Título <span style="display: none">principal </span>do meu site</h1>
```
→ elemento

```
elemento.textContent
```
→ Título principal do meu site

# 3. CONTEÚDO

**.innerText**

Retorna o texto do elemento selecionado e seus descendentes (filhos).

## Sintaxe

```
elemento.innerText
```

## Exemplo

Dado o código HTML e atribuindo ele à um elemento com um método de busca e seleção.

```
<h1>Título <span style="display: none">principal </span>do meu site</h1>
```
→ elemento

`elemento.innerText` → Título do meu site → Perceba que o texto “principal” foi ignorado.

# 3. CONTEÚDO

**.innerHTML – ATENÇÃO!**

Retorna o código HTML do elemento selecionado e seus descendentes (filhos).

## Sintaxe

```
elemento.innerHTML
```

## Exemplo

Dado o código HTML e atribuindo ele à um elemento com um método de busca e seleção.

```
<h1>Título <span style="display: none">principal </span>do meu site</h1>
```

```
elemento.innerHTML ⟶ "<h1>Título <span style="display: none">principal
</span>do meu site</h1>"
```

# 3. CONTEÚDO

**.value**

Manipula o valor do elemento especificado.

## Sintaxe

```
elemento.value
```

## Exemplo

Dado o código HTML e atribuindo ele à um elemento com um método de busca e seleção.

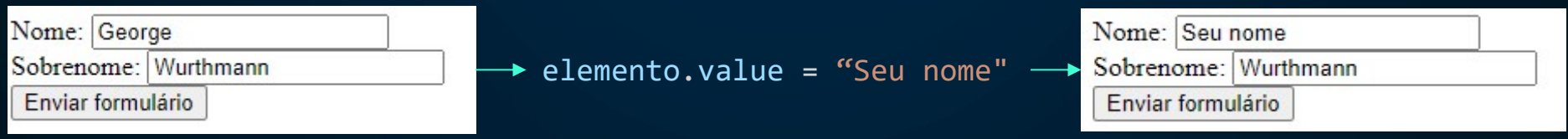
# 3. CONTEÚDO

**.value** — Manipula o valor do elemento especificado.

**Exemplo** Dado o código HTML e atribuindo ele à um elemento com um método de busca e seleção.

```
<form>
    Nome: <input type="text" name="nome" value="George"><br>
    Sobrenome: <input type="text" name="sobrenome" value="Wurthmann"><br>
    <input type="submit" value="Enviar formulário">
</form>
```

elemento



# 4. ESTILOS

.classList

.add()

.remove()

.toggle()

**Elemento com classe**

```
<h1 class="fundo-escuro texto-vermelho fonte-ampliada">Meu título</h1>
```

# 4. ESTILOS

**.classList** — Retorna uma coleção ativa dos atributos de classe do elemento.

## Sintaxe

```
elemento.classList
```

## Exemplo

Dado o código HTML e atribuindo ele à um elemento com um método de busca e seleção.

```
<h1 class="fundo-escuro texto-vermelho fonte-ampliada">Meu título</h1> ⟶ elemento

elemento.classList ⟶ [“fundo-escuro”, “texto-vermelho”, “fonte-ampliada”]
```

# 4. ESTILOS

**.classList**

O .classList possui 3 métodos para a manipulação das classes:

**.add()**

Adiciona uma classe ao elemento.

**.remove()**

Remove uma classe do elemento.

**.toggle()**

Se o elemento possuir classe informa ele remove, caso contrário adiciona.

# 4. ESTILOS

**.add()** Adiciona a classe informada ao elemento selecionado

## Sintaxe

```
elemento.classList.add()
```

## Exemplo

Dado o código HTML e atribuindo ele à um elemento com um método de busca e seleção.

```
<h1 class="fundo-escuro texto-vermelho">Meu título</h1>
```
→ elemento

```
elemento.classList.add(“fonte-ampliada”)
```
→ Comando executado.

```
<h1 class="fundo-escuro texto-vermelho fonte-ampliada">Meu título</h1>
```

# 4. ESTILOS

**.remove()** — Remove a classe informada ao elemento selecionado

## Sintaxe

```
elemento.classList.remove()
```

## Exemplo

Dado o código HTML e atribuindo ele à um elemento com um método de busca e seleção.

```
<h1 class="fundo-escuro texto-vermelho fonte-ampliada">Meu título</h1>
```
→ elemento

```
elemento.classList.remove("fonte-ampliada")
```
→ Comando executado.

```
<h1 class="fundo-escuro texto-vermelho">Meu título</h1>
```

# 4. ESTILOS

**.toggle()**

Alterna a classe informada ao elemento selecionado

## Sintaxe

```
elemento.classList.toggle()
```

## Exemplo

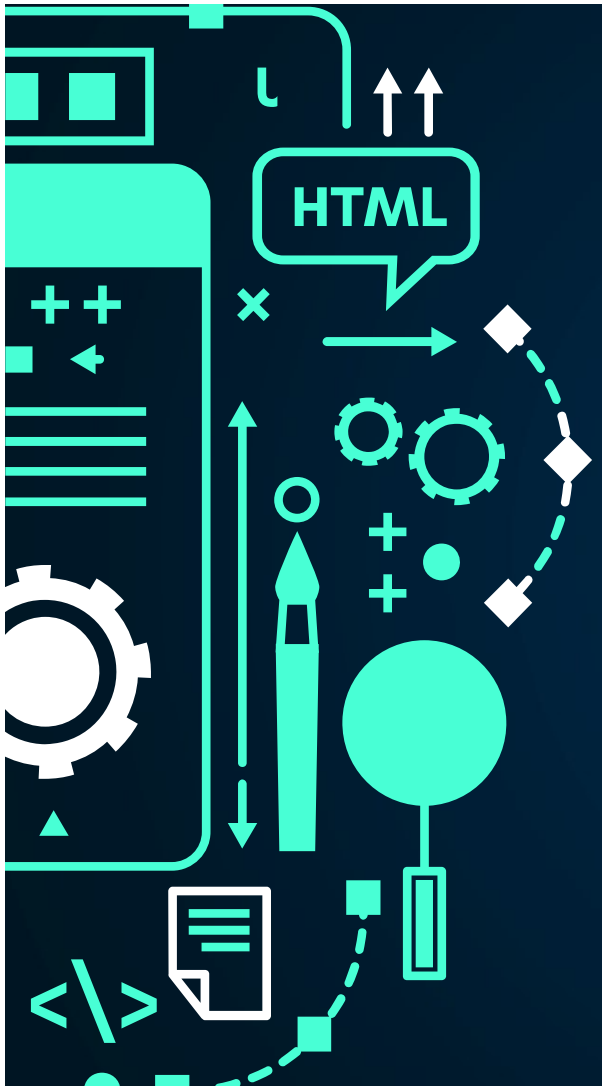
Dado o código HTML e atribuindo ele à um elemento com um método de busca e seleção.

```
<h1 class="fundo-escuro texto-vermelho">Meu título</h1>
```

→ elemento

```
elemento.classList.toggle(“fonte-ampliada”) e elemento.classList.toggle(“texto-vermelho”)
```

```
<h1 class="fundo-escuro fonte-ampliada">Meu título</h1>
```




# Vamos praticar?

1. Desenvolver uma página com um botão que alterna o tema entre claro e escuro.
2. Dado um formulário de cadastro, validar se os campos estão preenchidos.

# Obrigado

Professor George Wurthmann
@geo-wurth
georgewurthmann