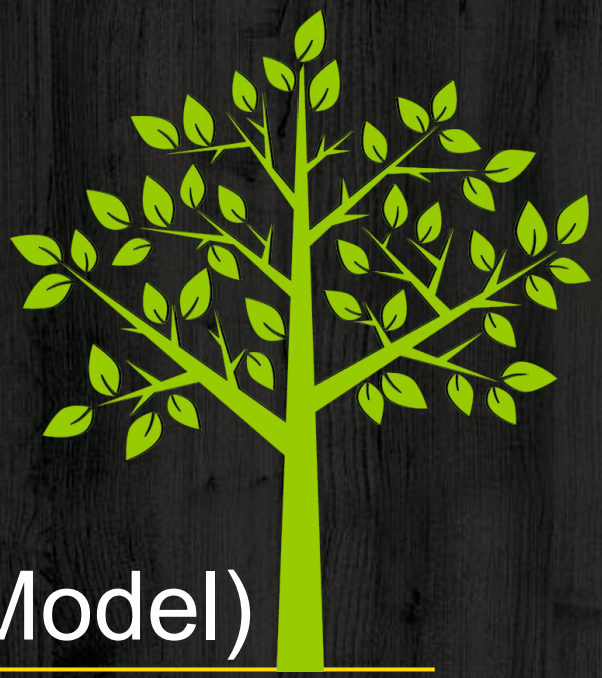
JavaScript

Programa do curso

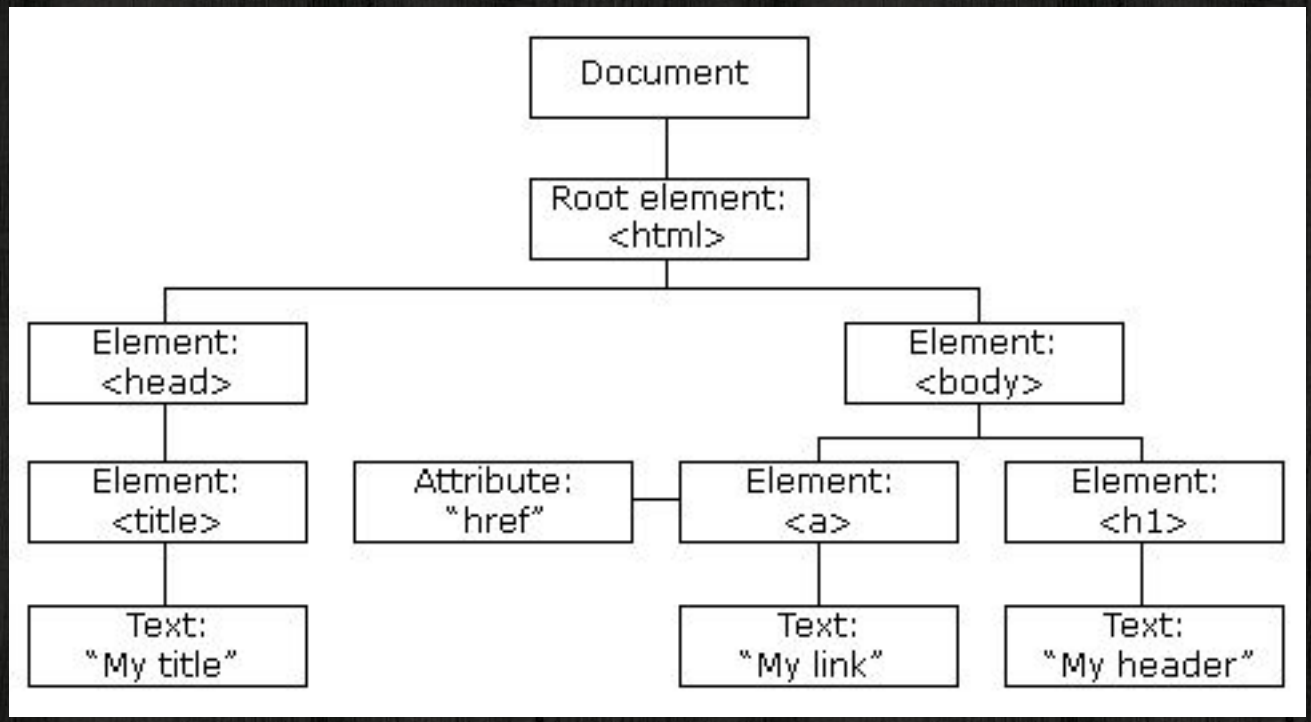
- **Aula 1: Introdução e objetos**
- Aula 2: JavaScript integrado a HTML
- Aula 3: DOM, seletores e elementos
- **Aula 4: DOM, seletores e elementos**
- Aula 5: Formulários
- Aula 6: Ajax
- Aula 7: Exercício Integrador



1. DOM (Document Object Model)

DOM - Definição

O **document object model (DOM)** oferece uma representação estrutural de um documento html.



DOM - Definição

Com o DOM, o JavaScript consegue:

- Modificar elementos, atributos e estilos de uma página.
- Excluir qualquer elemento e atributo.
- Adicionar novos elementos ou atributos.
- Responder a todos os eventos HTML da página.
- Criar novos eventos HTML na página.

2. Seletores

Seletores

Para acessar os **elementos** de uma página, usamos seletores.

É possível acessar usando os nós, mas isso pode ser um pouco confuso.

Cada seletor pode retornar apenas **um elemento** ou um **array**.

O objeto document tem os seguintes seletores como método:

- **document.getElementById(ID)**
- **document.getElementsByClassName(classe)**
- **document.querySelector(cssQuery)**
- **document.querySelectorAll(cssQuery)**

document.getElementById()

```
<html>
<head>
</head>
<body>
    <h1 id="cabecalho">Bem-vindo!</h1>
    <script>
        document.getElementById("cabecalho").style.display = 'none';
    </script>
</body>
</html>
```

document.querySelector()

```
<html>
<head>
</head>
<body>
    <h1 class="cabecalho">Bem-vindo!</h1>
    <script>
        document.querySelector(".cabecalho").style.color = ‘blue’;
    </script>
</body>
</html>
```

querySelector() seleciona o **primeiro** elemento que cumpra a condição.

document.querySelectorAll()

```
<html>
<head>
</head>
<body>
    <h1 class="vermelho">Bem-vindo!</h1>
    <p class="vermelho">José</p>
    <script>
        document.querySelectorAll(".vermelho").forEach(
        function(element){
            element.style.color = 'red';
        });
    </script>
</body>
</html>
```

2. Modificar elementos

Atributos

Assim que um elemento é selecionado, é possível acessar os atributos dele usando a propriedade **attributes**.

Retorna um mapa (é como um array) que tem valores key/value com os nomes e valores dos atributos desse elemento.

```
elemento.attributes;
```

getAttribute() / setAttribute()

O método **getAttribute** aceita uma string como parâmetro com o nome do atributo que queremos obter. Retorna o valor do atributo. Caso ele não seja encontrado, retorna null.

```
elemento.getAttribute("href");
```

O método **setAttribute** permite adicionar um novo atributo ou modificar um existente.

```
elemento.setAttribute("class","vermelho"):
```

hasAttribute() / removeAttribute()

O método **hasAttribute** aceita uma string como parâmetro com o nome do atributo que queremos saber se existe no elemento.

Retorna um valor booleano.

```
elemento.hasAttribute(nomeAtributoEmString);
```

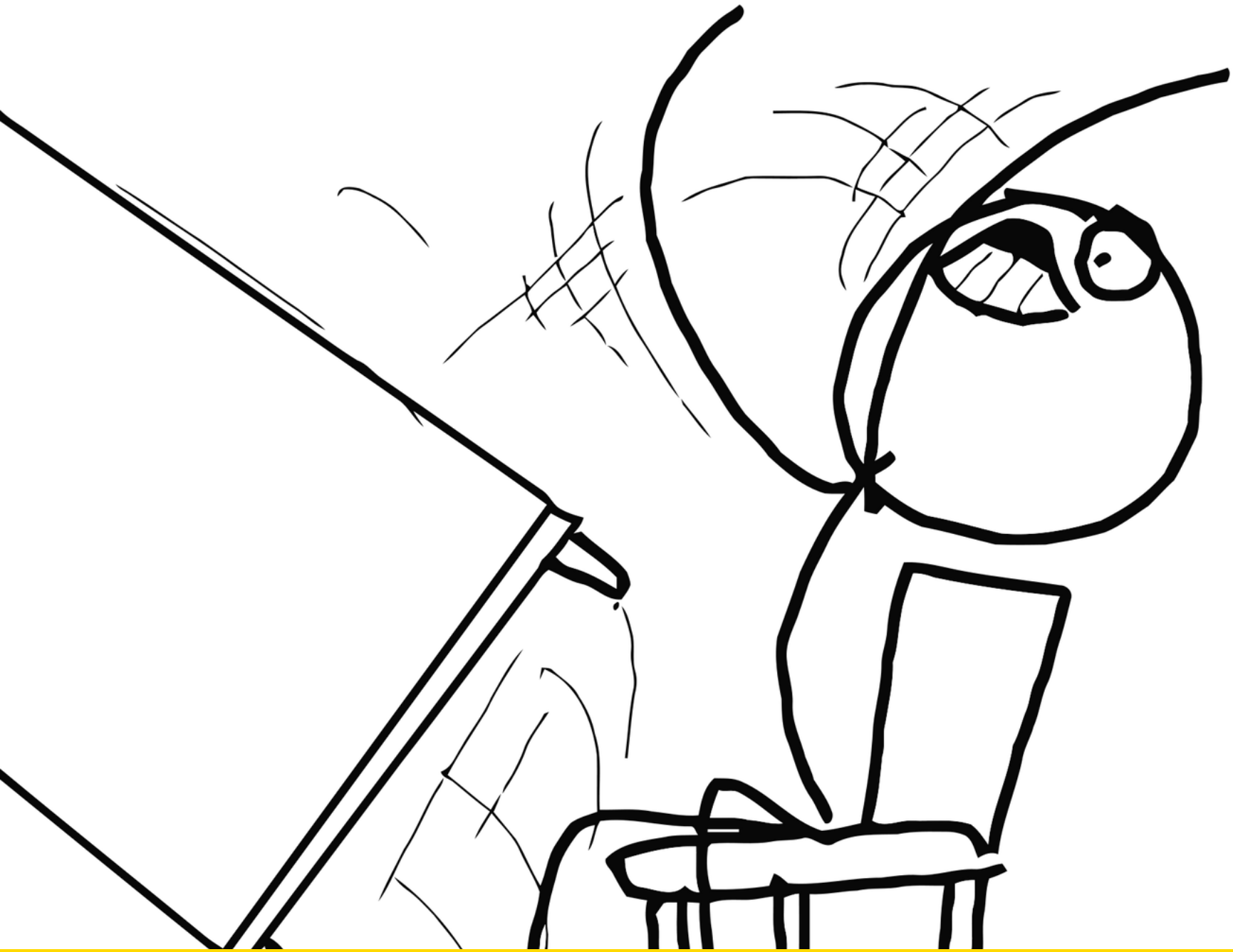
Com o método **removeAttribute,** podemos remover um atributo existente.

```
elemento.removeAttribute(nomeAtributo):
```

Estilos

- Os elementos HTML têm uma propriedade chamada *style*, que retorna um objeto literal que representa os estilos desse objeto.
- É possível adicionar ou modificar seus atributos.
- Os nomes das propriedades CSS em JavaScript são escritos no seguinte formato: **nomeDePropiedadeDeCss**.

```
elemento.style.color = "red"; // configuramos a cor vermelha
elemento.style.fontWeight = "bold"; // configuramos o weight como negrito
```



Vamos praticar!

Prática 1 - **DOM - Seletores e Atributos**

3. Elemento

Criando elementos - createElement

O método **createElement** permite criar novos elementos HTML.

createElement aceita como parâmetro strings com nomes de tags HTML (a, div, span, li, ul, etc).

```
var btn = document.createElement("BUTTON");
```

O método **createTextNode** permite criar novos textos HTML.

```
var texto = document.createTextNode("Olá, sou um texto");
```

Inserindo elementos - appendChild

appendChild permite inserir um nó dentro de outro.

```
<script>
	let li = document.createElement("LI");
	let liText = document.createTextNode("Water");
	li.appendChild(liText);
	document.getElementById("myList").appendChild(li);
</script>
<body>
	<ul id="myList"></ul>
</body>
```

textContext / innerHTML

textContent permite ler ou escrever conteúdo como **texto**.

```
elemento.textContent = "texto";

elemento.textContent; // texto
```

innerHTML permite ler ou escrever conteúdo em **HTML** de um nó.

```
<div id="cabecalho"></div>

let elemento = document.getElementById("cabecalho");

elemento.innerHTML = "<h1>Meu elemento HTML</h1>";
```

Resultado:

```
<div id="cabecalho"><h1>Meu elemento HTML</h1></div>
```

Removendo elementos - removeChild

O objeto nó tem um método chamado **removeChild** que permite remover nós filhos.

Para poder remover um nó, primeiro precisamos selecioná-lo.

```
<div id=’pai’>
      <div id=’filho’></div>
</div>
```

```
let pai = document.getElementById(‘pai’);

let filho = document.getElementById(‘filho’);

pai.removeChild(filho);
```

Vamos praticar!

Prática 2 - **Elementos**

Obrigado!

Perguntas?