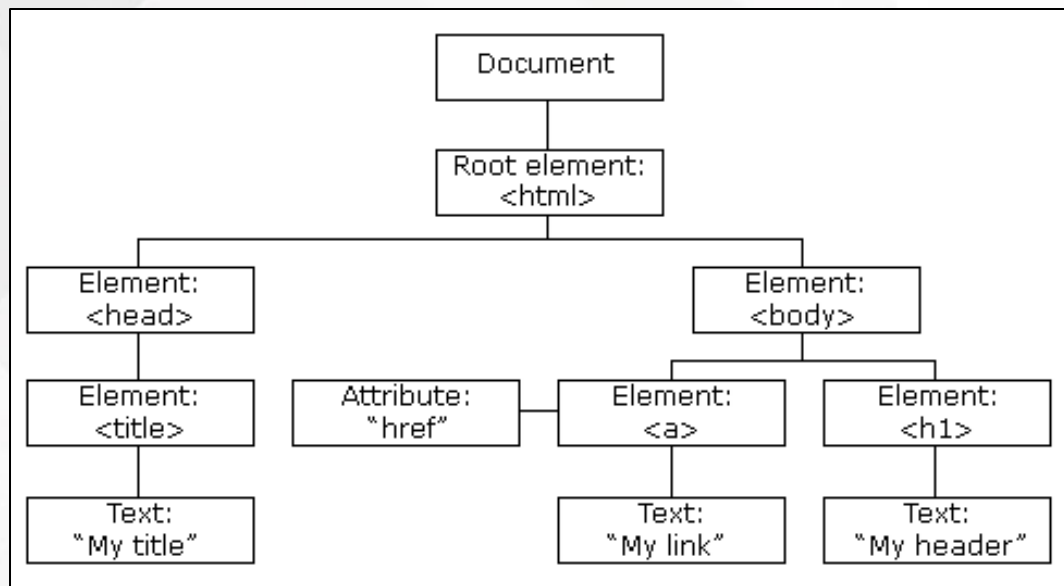


DOM - Document Object Model



DOM - Definição

O **document object model (DOM)** disponibiliza uma representação estrutural de um documento HTML

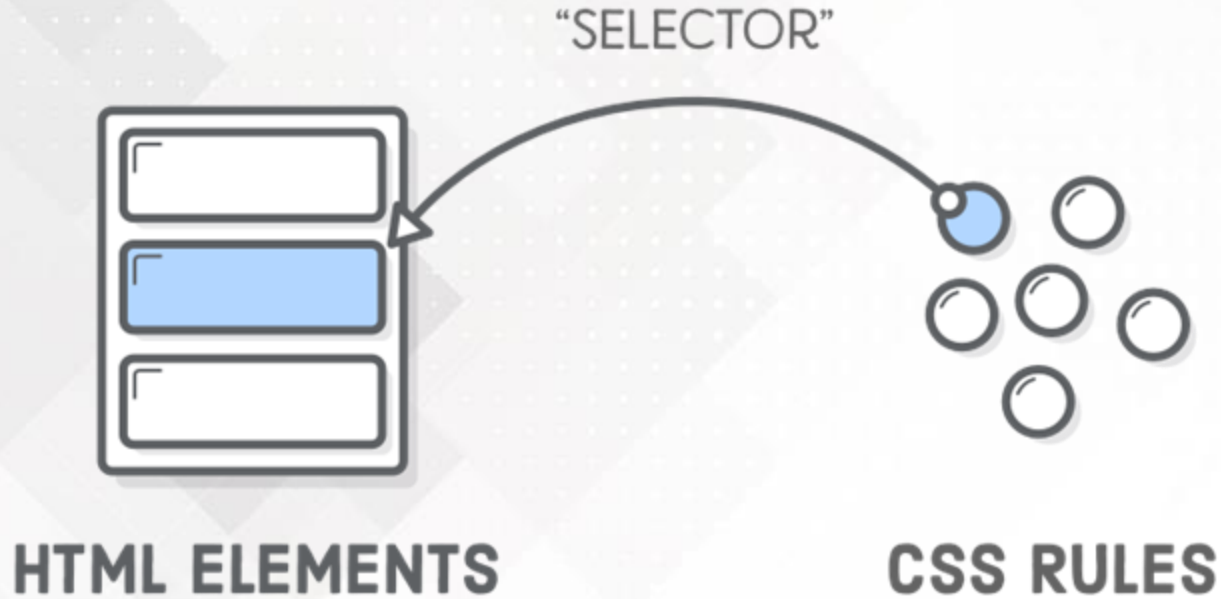


DOM - Definição

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 na página.
- Criar novos eventos na página.

Seletores



Seletores



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

Cada seletor pode retornar apenas **um elemento** ou uma **lista de elementos**.

O objeto document tem os seguintes seletores como método:

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



document.getElementById()



```
<script>
```

```
  let h1 = document.getElementById("titulo")
```

```
  h1.style.display = 'none';
```

```
</script>
```

```
<h1 id="titulo">Bem-vindo!</h1>
```



document.querySelector(CSSQuery)



```
<script>
```

```
  let vermelho = document.querySelector(".vermelho");
```

```
  vermelho.style.color = 'red';
```

```
</script>
```

```
<h1 class="vermelho">Bem-vindo!</h1>
```

```
<p class="vermelho">Esse é meu site :)</p>
```



document.querySelectorAll(CSSQuery)



```
<script>
```

```
  let vermelhos = document.querySelectorAll(".vermelho");
```

```
  for (let i = 0; i < vermelhos.length; i++) {
```

```
    vermelhos[i].style.color = 'red'
```

```
  }
```

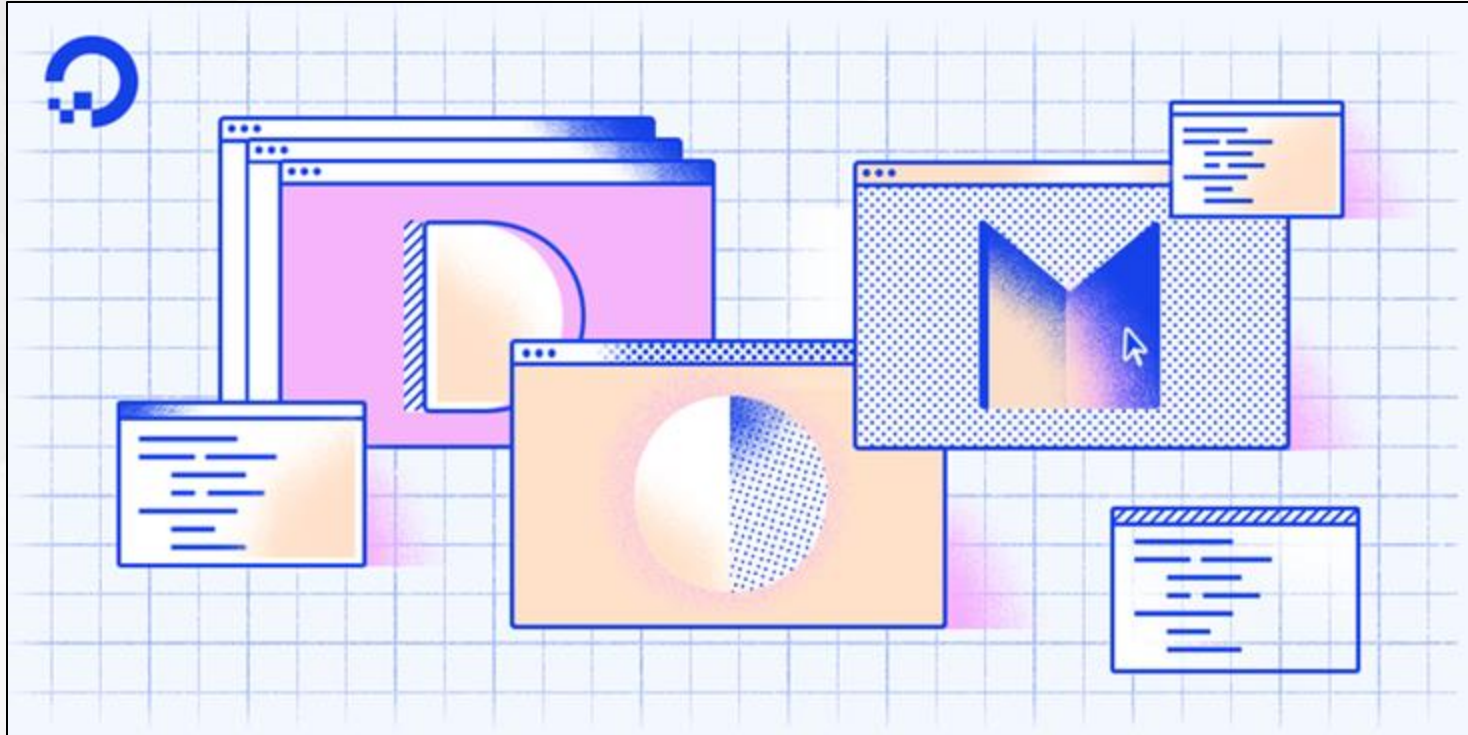
```
</script>
```

```
<h1 class="vermelho">Bem-vindo!</h1>
```

```
<p class="vermelho">Esse é meu site :)</p>
```



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 os nomes e valores dos atributos deste 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 fonte para negrito`

Elemento

<header>

<nav>

<section>

<article>

<aside>

<footer>

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).

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

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

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



Inserindo elementos - appendChild

appendChild permite inserir um nó dentro de outro.

```
let li = document.createElement("LI");
```

```
let textoLi = document.createTextNode("Item lista");
```

```
li.appendChild(textoLi);
```

```
document.getElementById("minhaLista").appendChild(li);
```

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


textContent / innerHTML

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

```
elemento.textContent = "texto";
```

```
elemento.textContent;    // texto
```

innerHTML permite escrever conteúdo em um elemento.

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

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

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

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.

```
let elemento = document.getElementById(ID);  
let elementoFilho = elemento.children.item(nroItem);  
elemento.removeChild(elementoFilho);
```

Intervalo

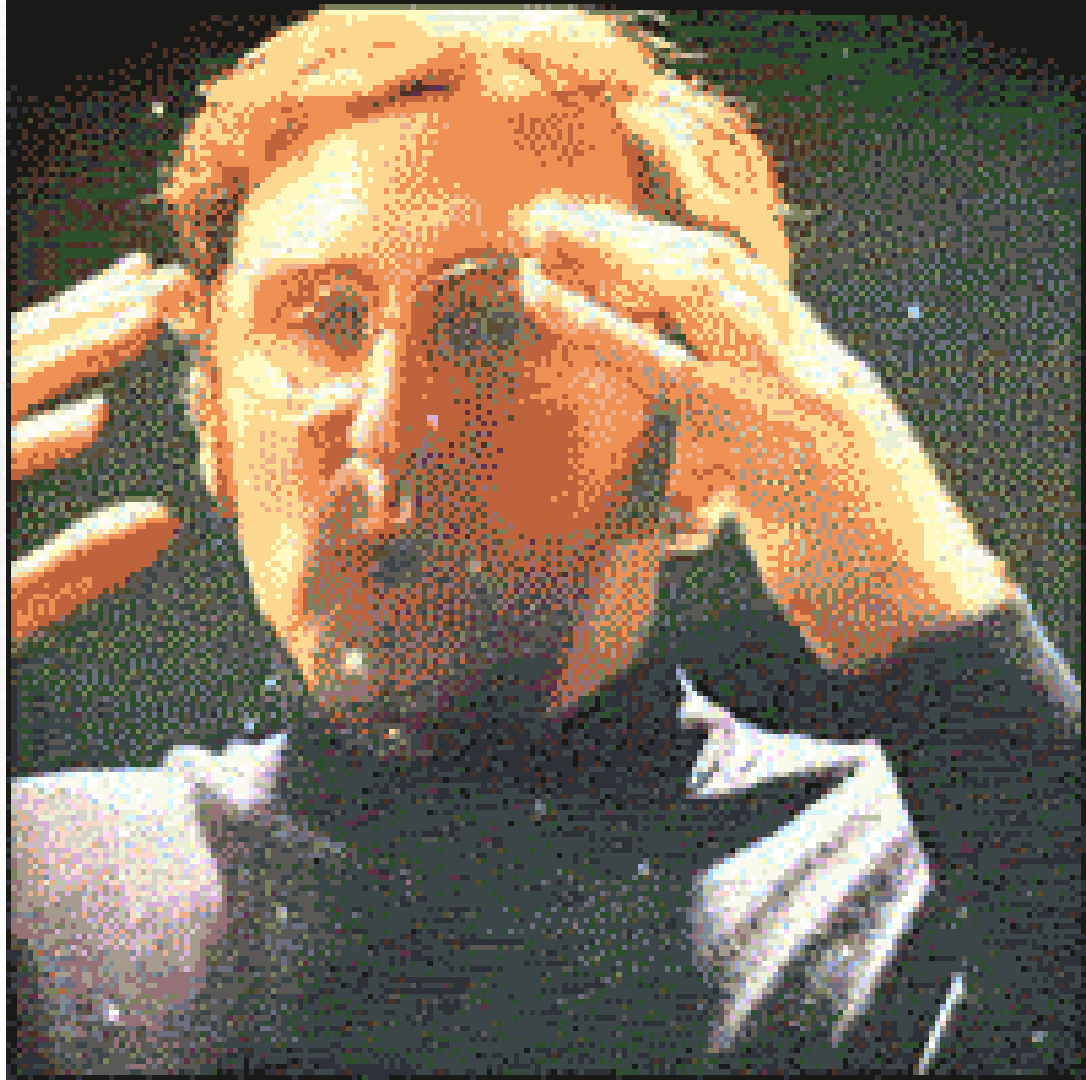


Eventos

Javascript - na última aula...

- Conceitos básicos de JS
- Estrutura DOM
- Manipulação de elemento e atributos
- Eventos

Interação do usuário através de eventos



DOM - Eventos

Um evento é uma coisa que acontece no navegador ou algo que o usuário faz.

Alguns exemplos:

- A página terminou de carregar
- A página foi rolada
- Um botão foi clicado

O JavaScript permite agir quando esses eventos acontecem.

DOM - Eventos

Existem duas formas de registrar um evento:

A primeira é estabelecendo uma propriedade diretamente no elemento

```
elemento.onNomeDoEvento = function () {}
```

As mais usadas são:

- ◆ onclick
- ◆ onchange
- ◆ onmouseover
- ◆ onmouseout
- ◆ onkeydown
- ◆ onload

DOM - Eventos

A segunda é utilizando **addEventListener**.

```
elemento.addEventListener(tipoDeEvento, funcaoGerenciadora);
```

tipoDeEvento: é uma string com o nome do tipo de evento

funcaoGerenciadora: é uma função invocada quando o evento acontece.

Por exemplo:

```
elemento.addEventListener("click", function(){  
    alert("Ai! Você clicou em mim!");  
});
```

Eventos - this

Podemos utilizar a palavra reservada **this**, que neste contexto faz referência ao objeto que executou o evento.

```
function minhaFuncao() {  
    // this é o elemento que executou o evento  
    console.log(this)  
}
```

```
elemento.addEventListener('click', minhaFuncao);
```

Eventos - removeEventListener

Para remover um *addEventListener* inserido, utilizamos:

```
elemento.removeEventListener(tipoDeEvento, funcaoGerenciadora)
```

- **tipoDeEvento**: é necessário que seja o mesmo evento do `addEventListener`
- **funcaoGerenciadora**: é necessário que seja a mesma função do `addEventListener`

Eventos - preventDefault()

Para evitar a execução de um evento, por padrão, utilizamos:

```
document.querySelector("#link").addEventListener("click",  
function(event) {  
    event.preventDefault();  
    // o link não vai mais para o Google  
});
```

```
<a id="link" href="http://www.google.com.br/">Google</a>
```

Eventos - Mouse

O objeto **event** associado ao mouse tem atributos que permitem saber a posição dele com **clientX** e **clientY**.

```
elemento.addEventListener('click', function(event) {  
    event.clientX;  
    event.clientY;  
});
```

Eventos - Teclado

Também é possível controlar os eventos disparados quando as teclas são pressionadas, utilizando os eventos `keypress`, `keydown` e `keyup`.

```
elemento.addEventListener('keypress', function(event) {  
    let tecla = event.keyCode;  
    if (tecla == 27) {  
        alert("Você apertou escape!!");  
    }  
});
```

Timers



Timers - setTimeout

O JavaScript tem funções nativas que permitem atrasar a execução de qualquer código.

A função **setTimeout** é utilizada quando queremos que o código seja executado uma vez depois de um tempo estabelecido.

setTimeout(funcao, tempoDeEspera);

Exibe um alert depois de **3** segundos (3000 milissegundos):

setTimeout(function(){ alert("Hello"); }, 3000);

Timers - setInterval

Por meio dessa função, podemos executar o mesmo código várias vezes em um intervalo regular.

setInterval(função, tempoDeEspera);

A cada 3 segundos, aparece o alert:

setInterval(function(){ alert("Hello"); }, 3000);

Timers - clearTimeout / clearInterval

Para interromper um *timeout*, utilizamos:

```
let delay = setTimeout(function(){ alert("Hello"); }, 3000);  
clearTimeout(delay);
```

Para interromper um *interval*, utilizamos:

```
let intervalo = setInterval(function(){ alert("Hello"); }, 3000);  
clearInterval(intervalo);
```

Até a próxima aula!

