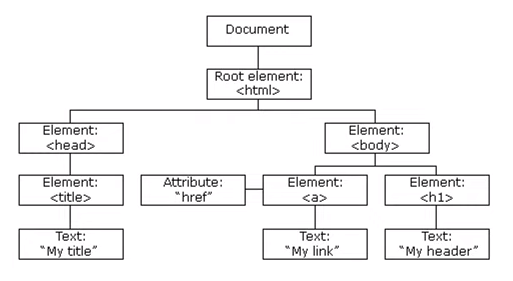
# O que é D.O.M

**DOM** significa: **Document Object Model**

O **DOM HTML** é um padrão de como **acessar e modificar os elementos HTML de uma página.**

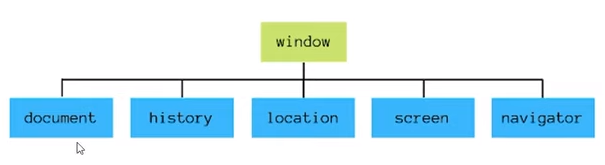
****

Cada elemento do DOM é chamado de **NÓ**.

# O que é B.O.M

**BOM** significa: **Browser Object Model**

Ele é o padrão de como **os browsers funcionam**, tudo que está dentro do **BOM** faz parte do Window, mas nem todo item que está dentro do **BOM** faz parte do **DOM.**



## Diferença entre D.O.M e B.O.M

O **DOM** faz parte do **BOM**, ele se encaixa no primeiro item do diagrama do **BOM**, pois o **DOM** nada mais é do que um documento

# Manipulando e Selecionando Elementos

A estrutura básica de um HTML é composta por **tags**, **identificadores**, **classes**

**Quando temos identificadores únicos, definimos o id, porém, quando temos mais de um identificador e precisamos de fazer a mesma ação com todos eles, usamos classes**

## Métodos

Para selecionar elementos de uma página, podemos fazê-lo de algumas formas:

**getElementById(‘titulo’);**

Retorna o elemento existente na tag

**getElementsByTagName(‘li’);**

Retorna um array com todos os elementos existentes na tag

**getElementsByClassName(‘textos’);**

Retorna um array com todos os elementos existentes na classe



**querySelectorAll(‘.primeira-classe .segunda-classe’);**

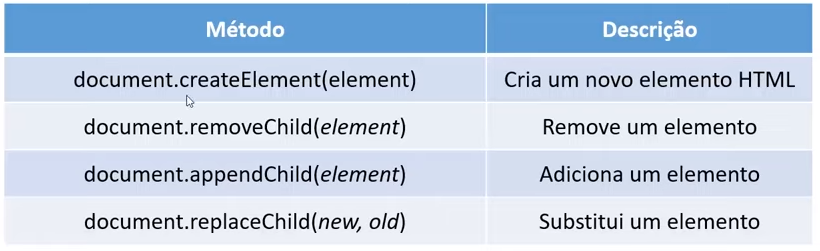
Podemos selecionar várias classes que o comando pegará todos os resultados que atenda ao critério

**querySelectorAll(‘li .opção’);**

Podemos selecionar o objeto e classe juntas também.

# 

# Métodos para Adicionar e Deletar um elemento HTML



# Trabalhando com Estilos

O **Element.classList** é uma propriedade somente leitura que retorna uma coleção [DOMTokenList (en-US)](https://developer.mozilla.org/en-US/docs/Web/API/DOMTokenList" \o "Currently only available in English (US)) ativa dos atributos de classe do elemento.

Usar **classList** é uma alternativa conveniente para acessar a lista de classes de um elemento como uma seqüência delimitada por espaço através de [element.className](https://developer.mozilla.org/pt-BR/docs/Web/API/Element/className).

## [**Sintaxe**](https://developer.mozilla.org/pt-BR/docs/Web/API/Element/classList#sintaxe)

const elementClasses = elementNodeReference.classList;

**elementClasses** é um DOMTokenList que representa o atributo de classe de **elementNodeReference**. Se o atributo de classe não foi definido ou está vazio **elementClasses.length** retorna 0. **element.classList**propriamente dito é somente leitura, embora você possa modificá-lo usando os métodos **add ()**e **remove ()**.

## [**Métodos**](https://developer.mozilla.org/pt-BR/docs/Web/API/Element/classList#m%C3%A9todos)

**add( String [, String] ):** Adicione valores de classe especificados. Se essas classes já existem no atributo do elemento, elas são ignoradas.

**remove( String [,String] ):** Remover valores de classe específicos.

**item ( Number ):** Retorna o valor da classe por índice na coleção.

**toggle ( String [, force] ):** Quando apenas um argumento está presente: Toggle class value; Ou seja, se a classe existir, em seguida, removê-lo e retornar false, se não, então adicioná-lo e retornar true. Quando um segundo argumento está presente: Se o segundo argumento é avaliado como true, adicione o valor especificado da classe e, se ele for avaliado como false, remova-o.

**contains( String ):** Verifica se o valor da classe especificado existe no atributo de classe do elemento.

## [**Exemplos**](https://developer.mozilla.org/pt-BR/docs/Web/API/Element/classList#exemplos)**:**

// div é uma referência de objeto para um elemento <div> com class = "foo bar"

div.classList.remove("foo");

div.classList.add("anotherclass");

// Se estiver visível, remova-o, caso contrário, adicione-o

div.classList.toggle("visible");

// adicionar/remover, dependendo do teste condicional, i menos de 10

div.classList.toggle("visible", i < 10 );

alert(div.classList.contains("foo"));

// adicionar ou remover várias classes

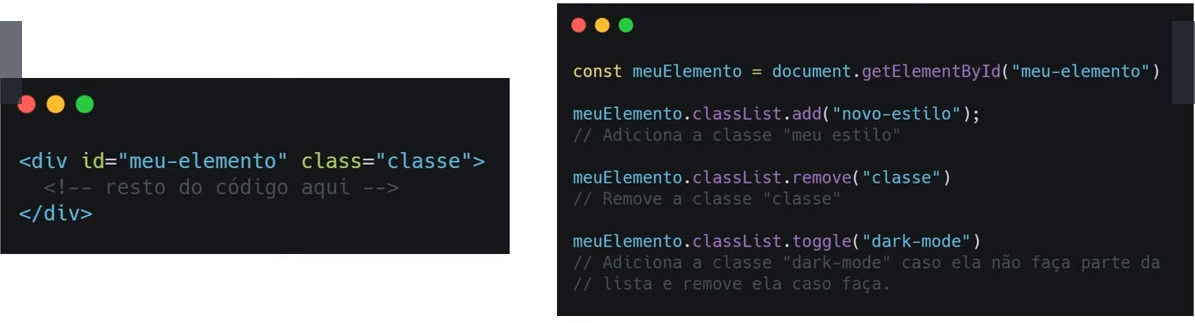
div.classList.add("foo","bar");

div.classList.remove("foo", "bar");

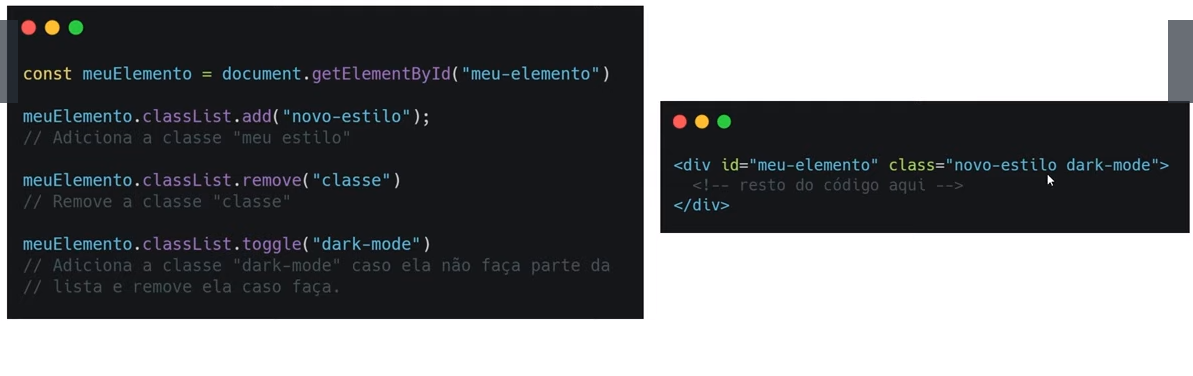
As versões do Firefox antes de 26 não implementam o uso de vários argumentos nos métodos add / remove / toggle. Veja <https://bugzilla.mozilla.org/show_bug.cgi?id=814014>

## [**Especificações**](https://developer.mozilla.org/pt-BR/docs/Web/API/Element/classList#especifica%C3%A7%C3%B5es)

| **Especificações** | **Status** | **Comentário** |
| --- | --- | --- |
| [HTML Living Standard The definition of 'Element.classList' in that specification.](https://html.spec.whatwg.org/multipage/dom.html#dom-classlist) | Padrão em tempo real | Observe dentro da especificação HTML relacionada ao [class](https://developer.mozilla.org/pt-BR/docs/Web/HTML/Global_attributes" \l "attr-class) attribute. |
| [DOM The definition of 'Element.classList' in that specification.](https://dom.spec.whatwg.org/#dom-element-classlist) | Padrão em tempo real | Definição inicial |



Código executado



Acessando diretamente um elemento



Desta forma estamos acessando o estilo direto do **P** e alterando o valor.

# Eventos

É qualquer ação que um usuário faça em uma página, quando ele interage com a página.

**Mouse**

**mouseover:** quando entramos em um elemento, acionamos o mouseover.

**mouseout:** quando saímos de um elemento, acionamos o mouseout.

**Click**

**click:** quando clicamos uma vez com o mouse, acionamos o evento click.

**dbclick**: quando clicamos duas vezes com o mouse, acionamos o evento dbclick.

**Atualização**

**change:** sempre queum campo mudar, alterar, acionamos o evento change.

**load:** sempre que uma página for carregada, acionamos o evento load.

**Event listener**

Diretamente no JavaScript, cria um evento que vai ser acionado no momento em que o usuário realizar determinada ação.

**Exemplo:**

const botao = document.getElementById(“meuBotao”);

botao.addEventListener(“click”, outraFuncao);

Outra forma de se fazer a chamada, é atribuindo diretamente no HTML

