**DOM (Document Object Model)**

O Document Object Model ( DOM ) é uma interface de programação para os documentos HTML e XML. Representa uma página de forma que os programas podem alterar a estrutura do documento, alterar o estilo e o conteúdo. O DOM representa o documento para nós e objetos, dessa forma, como linguagens de programação podem se conectar à página.

**Interfaces do DOM:**

**Document:**

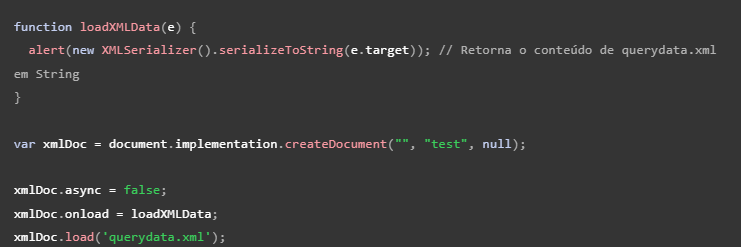
A interface Document serve como um ponto de entrada para o conteúdo da página (a árvore DOM, incluindo elementos como <body>e <table>) e fornece funcionalidades globais ao documento (como obter a URL da página e criar novos elementos no documento).

Propriedades:

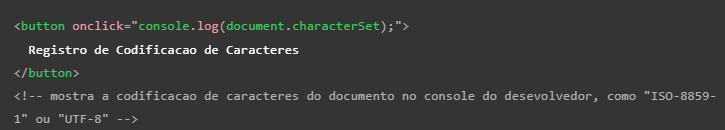
Document.all: Para acessar todos os elementos do documento.



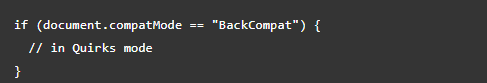
Document.async: Usado como document.load que indica uma requisição assíncrona.

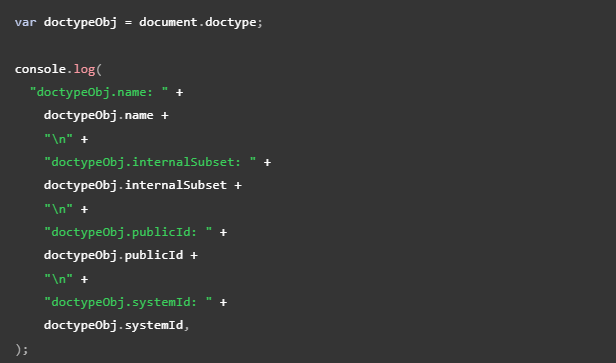


Document.characterSet: Retorna a codificação dos caracteres que estão sendo usados ​​pelo documento.

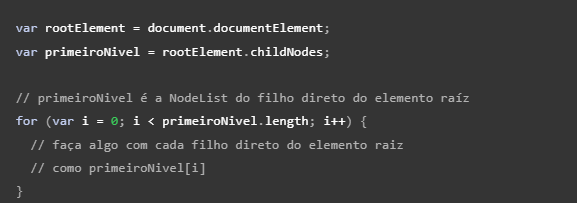


Document.compatMode: indica se o documento foi renderizado no modo Quirks ou Strict.

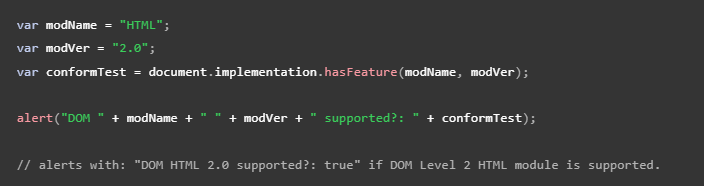


Document.doctype: Retorna o nome do tipo de conteúdo do documento HTML.

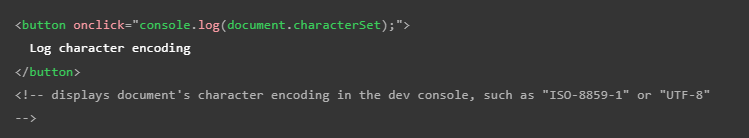
Document.documentElement: Retorna o Element que é um filho direto do documento. Para documentos HTML, este é normalmente o elemento HTML.

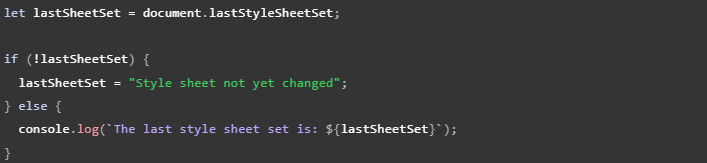


Document.implementation: Retorna a codificação usada quando o documento foi analisado.



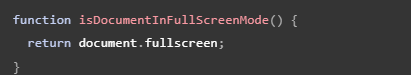
Document.inputEncoding: Retorna a codificação usada quando o documento foi analisado.



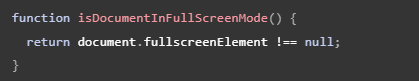
Document.lastStyleSheetSet: Retorna o nome do conjunto de folhas de estilo que foi habilitado pela última vez. Tem o valor null até que a folha de estilo seja alterada definindo o valor de [selectedStyleSheetSet](https://developer.mozilla.org/en-US/docs/Web/API/Document/selectedStyleSheetSet). 

Document.mozFullScreen: “true” quando o documento está em full-screen mode.

Esta função simples informa se o modo de tela cheia está ativo ou não, usando a fullscreenpropriedade obsoleta:



Por outro lado, usa a fullscreenElementpropriedade atual para determinar a mesma coisa:

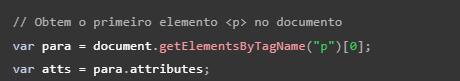


**Element:**

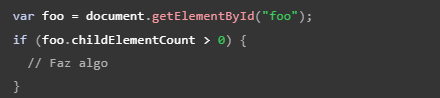
Uma interface Element é uma classe base mais geral de todos os objetos em uma [Document](https://developer.mozilla.org/pt-BR/docs/Web/API/Document) propriedade. Ela somente tem métodos e propriedades comuns para todos os tipos de elementos.

Propriedades:

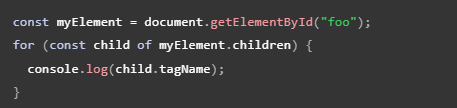
Element.attributes: Retorna uma [NamedNodeMap](https://developer.mozilla.org/en-US/docs/Web/API/NamedNodeMap" \o "Esta página está disponível apenas em inglês no momento) lista de todos os atributos associados ao elemento.



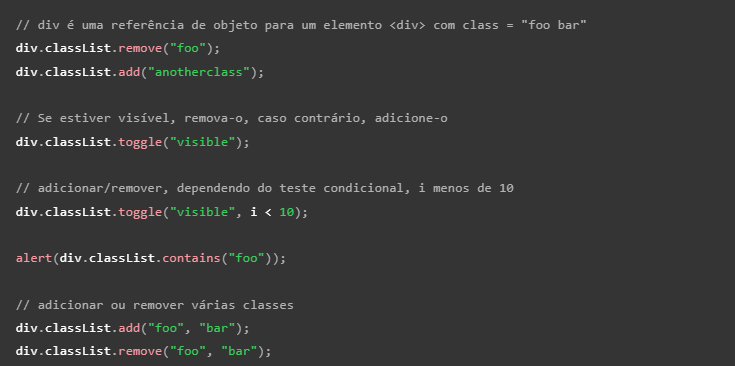
ParentNode.childElementCount: É um [Number](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Number)representando o número de nós filhos que são elementos.



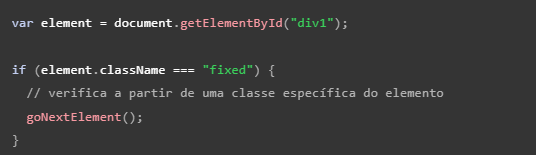
ParentNode.children: É um [HTMLCollection](https://developer.mozilla.org/pt-BR/docs/Web/API/HTMLCollection)ao vivo contendo todos os elementos filhos do elemento, como uma coleção.



Element.classList: Retorna um [DOMTokenList](https://developer.mozilla.org/en-US/docs/Web/API/DOMTokenList" \o "Esta página está disponível apenas em inglês no momento)contendo uma lista de atributos de classe.



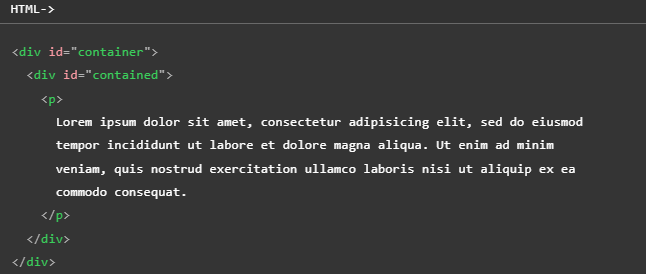
Element.className: É um [DOMString](https://developer.mozilla.org/pt-BR/docs/conflicting/Web/JavaScript/Reference/Global_Objects/String)representando a classe do elemento.

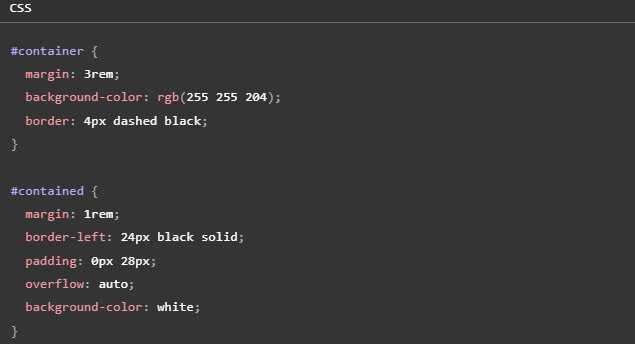


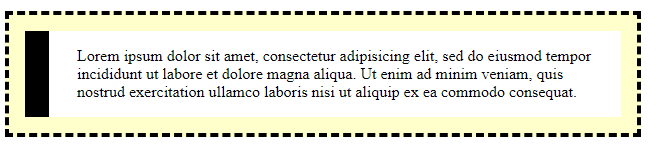
Element.clientHeight: Retorna um [Number](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Number)representando a altura interna do elemento.



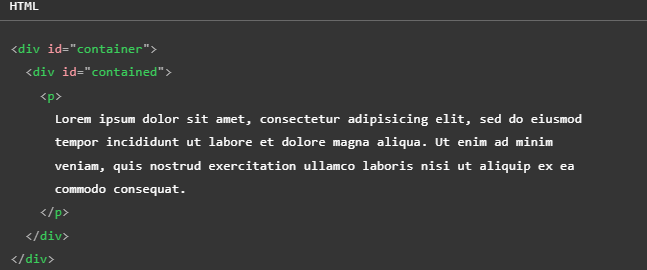
Element.clientLeft: Retorna um [Number](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Number) representando a largura da borda do elemento.

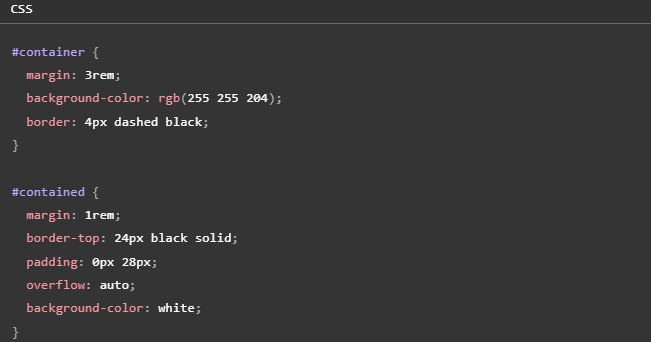


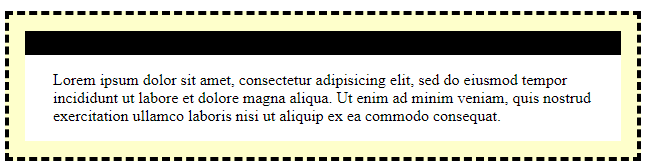




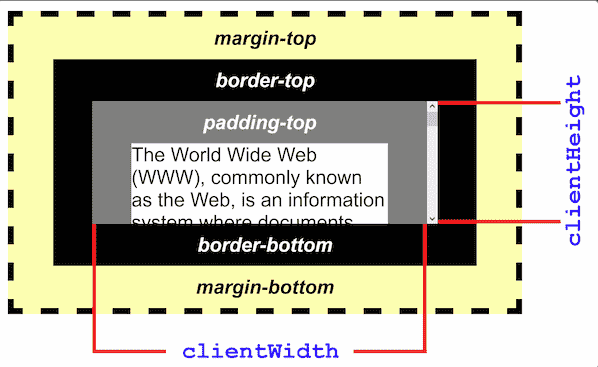
Element.clientTop: Retorna um [Number](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Number) representando a largura da borda do topo do elemento.





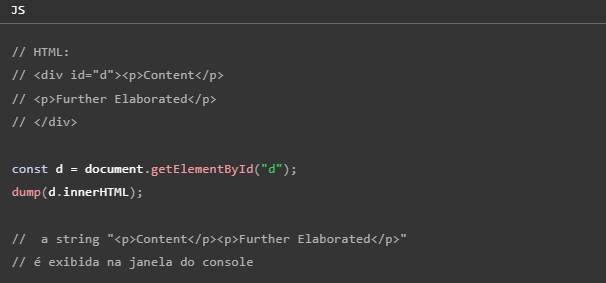


Element.clientWidth: Retorna um [Number](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Number) representando a largura interna do elemento.



Element.innerHTML: É uma [DOMString](https://developer.mozilla.org/pt-BR/docs/conflicting/Web/JavaScript/Reference/Global_Objects/String) representação da marcação do conteúdo do elemento.



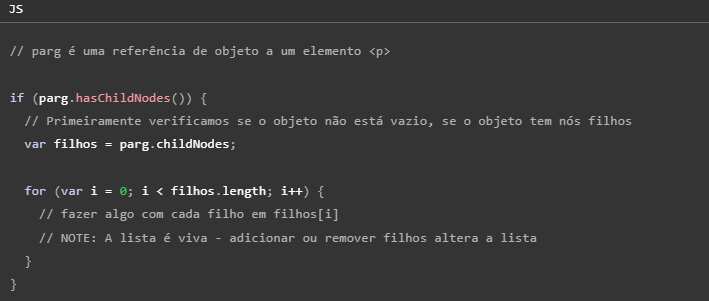


**Node:**

É uma interface de vários tipos do DOM herdado, e que permite que esses tipos sejam tratados de forma semelhante, por exemplo, herdando os mesmos métodos ou sendo testados da mesma forma.

Propriedades:

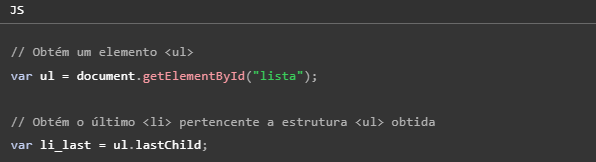
Node.chidNodes: Retorna um objeto [NodeList](https://developer.mozilla.org/pt-BR/docs/Web/API/NodeList) "vivo" contendo todos os filhos deste nó. Dizer que um objeto [NodeList](https://developer.mozilla.org/pt-BR/docs/Web/API/NodeList) é vivo significa que há alguma mudança em um dos filhos deste nó, o objeto [NodeList](https://developer.mozilla.org/pt-BR/docs/Web/API/NodeList) é atualizado automaticamente com tais mudanças.



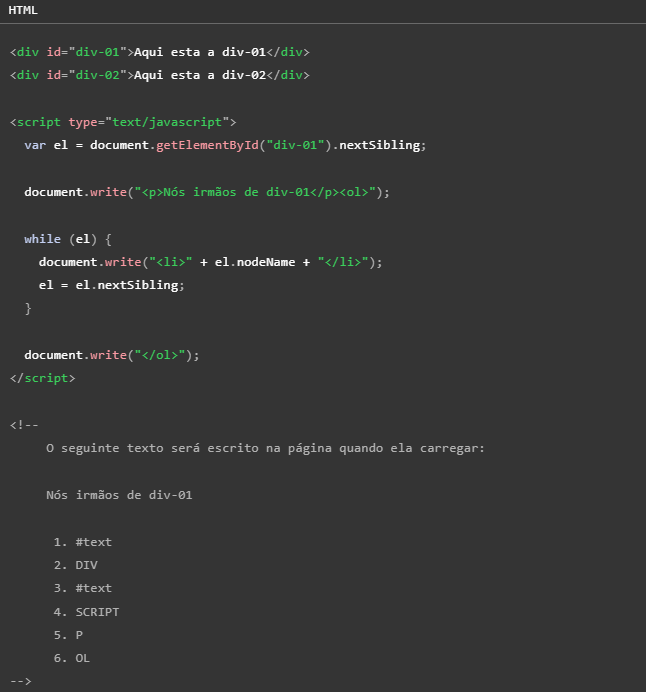
Node.firstChild: Retorna um [Node](https://developer.mozilla.org/pt-BR/docs/Web/API/Node) representando o primeiro filho direto do nó ou null, caso o nó não tenha nenhum filho.



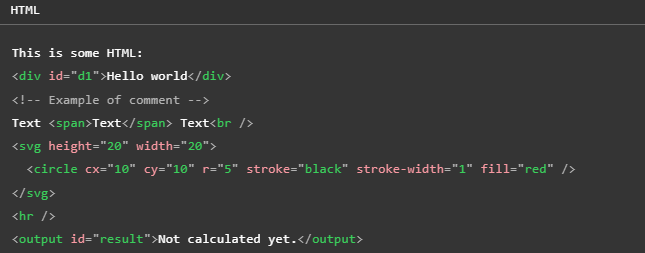
Node.lastChild: Retorna um [Node](https://developer.mozilla.org/pt-BR/docs/Web/API/Node) representando o último filho direto do elemento ou null, caso o elemento não tenha nenhum filho.

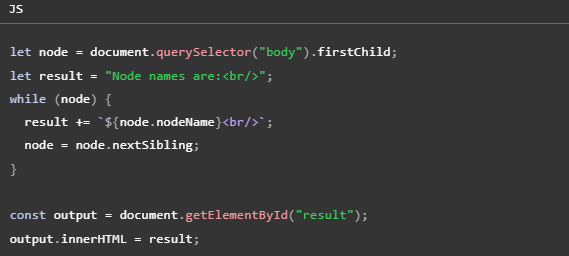


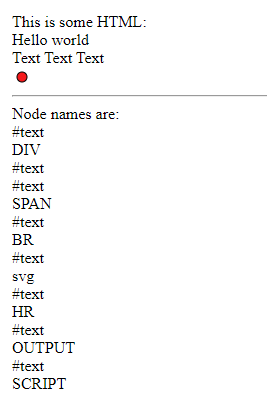
Node.nextSibling: Retorna um [Node](https://developer.mozilla.org/pt-BR/docs/Web/API/Node)representando o próximo elemento na árvore ou null, caso tal nó não exista.



Node.nodeName: Retorna uma [DOMString](https://developer.mozilla.org/pt-BR/docs/conflicting/Web/JavaScript/Reference/Global_Objects/String) contendo o nome do elemento, do Node. A estrutura do nome irá mudar conforme o tipo do elemento. Veja as diferenças na documentação do método [Node.nodeName](https://developer.mozilla.org/en-US/docs/Web/API/Node/nodeName" \o "Esta página está disponível apenas em inglês no momento).

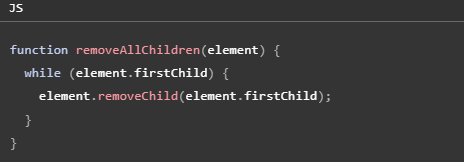




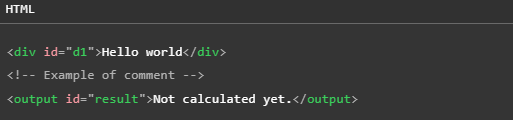


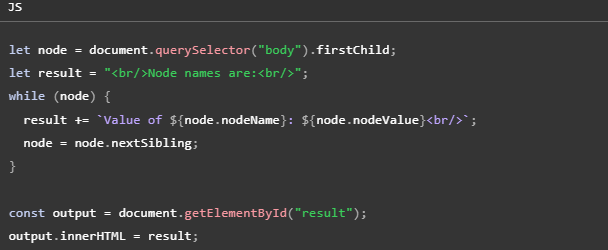
Node.nodePrincipal: Uma interface nsIPrincipal representando o nó principal.

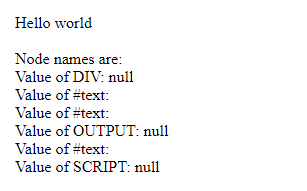
Esta função remove cada primeiro filho de um elemento, até que não reste nenhum:



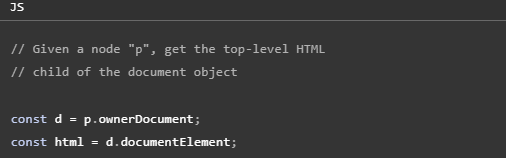
Node.nodeValue: Retorna / Definir o valor do nó atual.







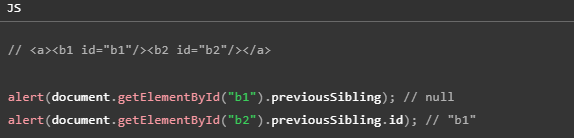
Node.ownerDocument: Retorna o [Document](https://developer.mozilla.org/pt-BR/docs/Web/API/Document) que esse nó pertence. Se o nó em si é um documento, retorna null.



Node.parentNode: Retorna um [Node](https://developer.mozilla.org/pt-BR/docs/Web/API/Node) que é pai desse nó. Se não existe tal nó, como, por exemplo, se esse nó é o topo da árvore ou se ele não participa de uma árvore, essa propriedade retorna null.



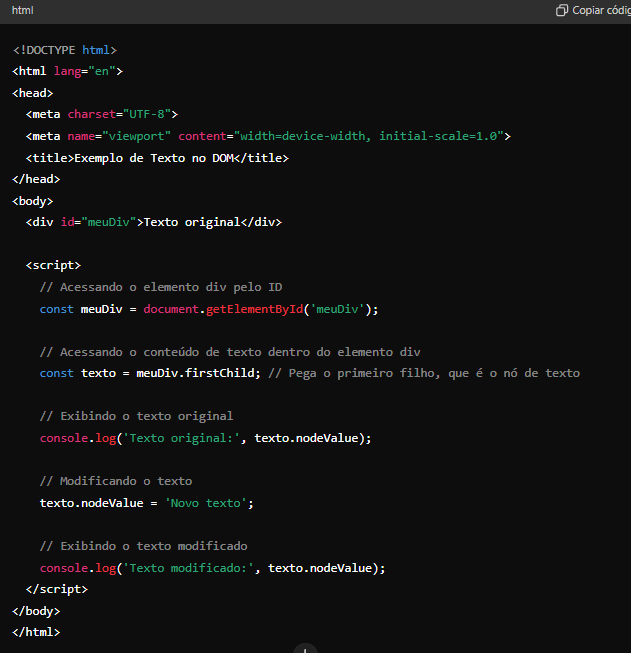
Node.previousSiblings: Retorna um [Node](https://developer.mozilla.org/pt-BR/docs/Web/API/Node) representando o último nó em uma árvore ou nullse não existe tal nodo.



**Text:**

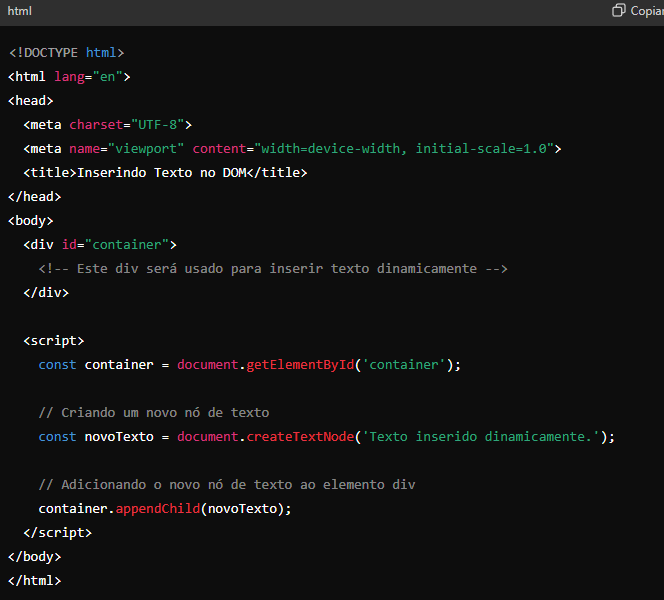
A interface Text no DOM (Document Object Model) representa o conteúdo de texto dentro de um elemento HTML. Ela é utilizada para manipular e acessar partes de texto dentro de nós (nodes) de elementos HTML.

Propriedades:



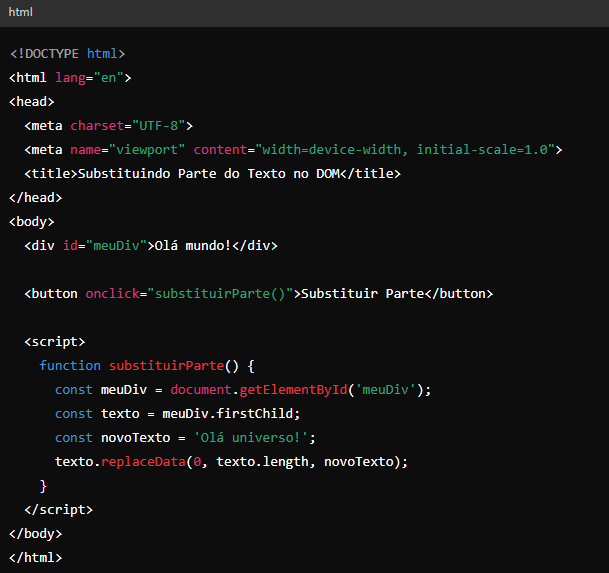
meuDiv.firstChild: acessa o nó de texto dentro do elemento <div>.

texto.nodeValue: permite acessar e modificar o valor do texto.

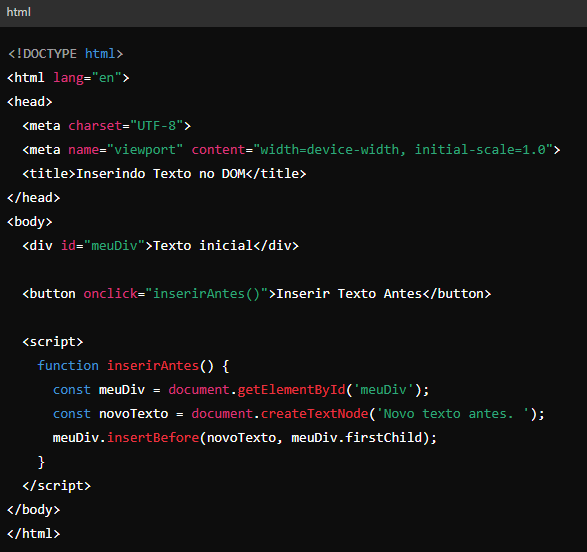


document.createTextNode('Texto inserido dinamicamente.') cria um novo nó de texto com o conteúdo especificado.

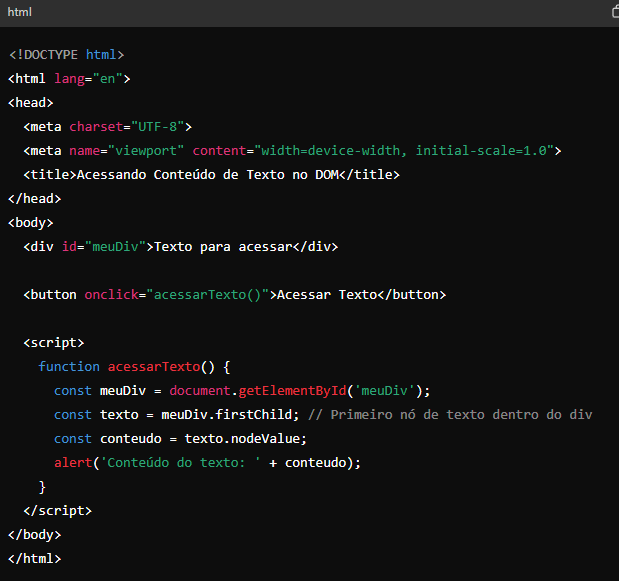
container.appendChild(novoTexto) insere o nó de texto criado como um filho do elemento <div> com id container.



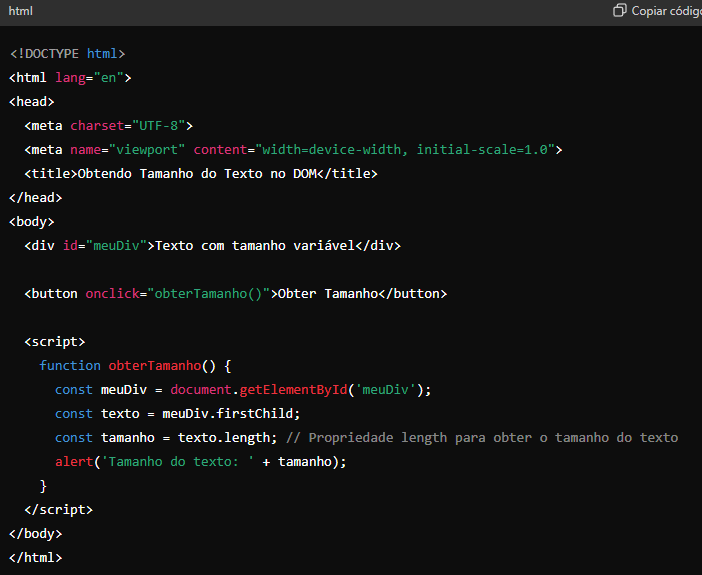
Este exemplo substitui parte do texto dentro do elemento <div> usando o método replaceData() da interface Text.



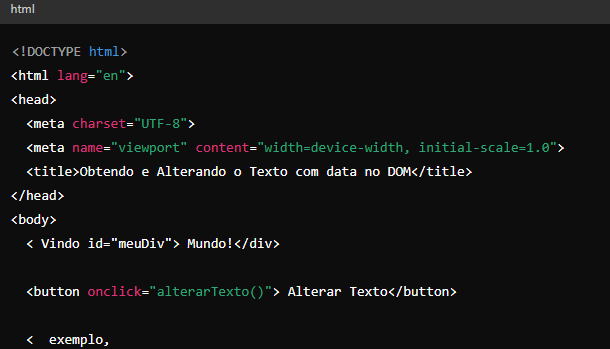
Este exemplo insere texto antes do texto existente dentro do elemento <div> quando o botão é clicado, usando o método insertBefore().



Este exemplo utiliza a propriedade nodeValue para acessar e exibir o conteúdo de texto dentro do elemento <div> quando o botão é clicado.



Neste exemplo, a propriedade length é usada para obter o tamanho do texto dentro do elemento <div>, exibindo-o em um alerta quando o botão é clicado.



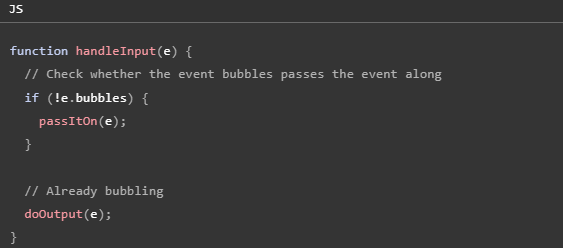
Obtendo e Alterando o Texto com a propriedade ‘data

**Event:**

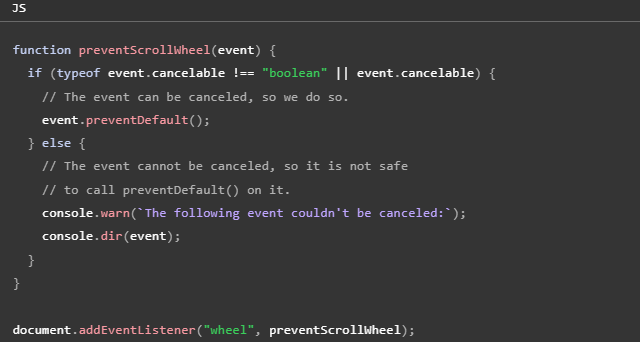
Uma interface de eventos representa qualquer evento do DOM. Ele contém propriedades comuns e métodos para qualquer evento.

Propriedades:

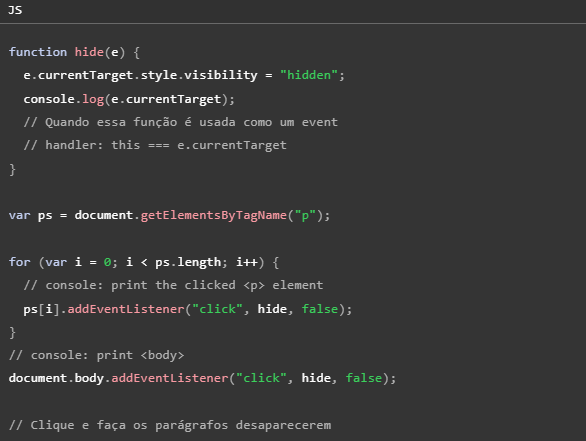
Event.bubbles: Um booleano que indica se o evento surge em bolha pelo DOM ou não.



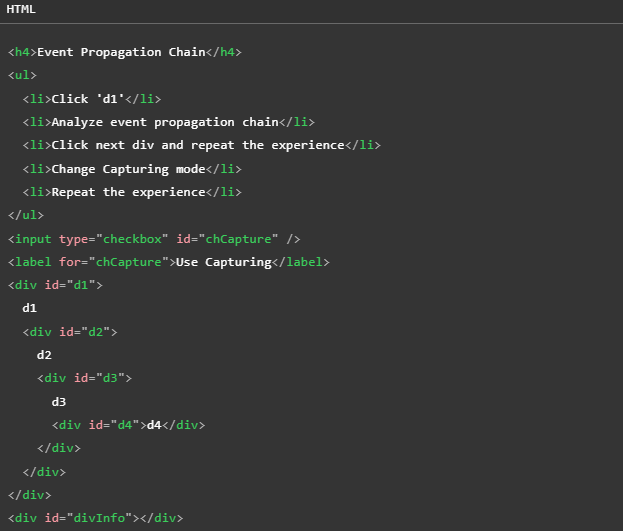
Event.cancelable: Um booleano que indica se o evento foi cancelado.

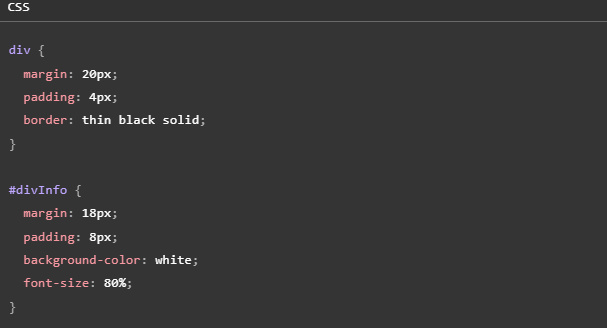


Event.currentTarget: Uma referência para o alvo registrado atualmente para o evento.

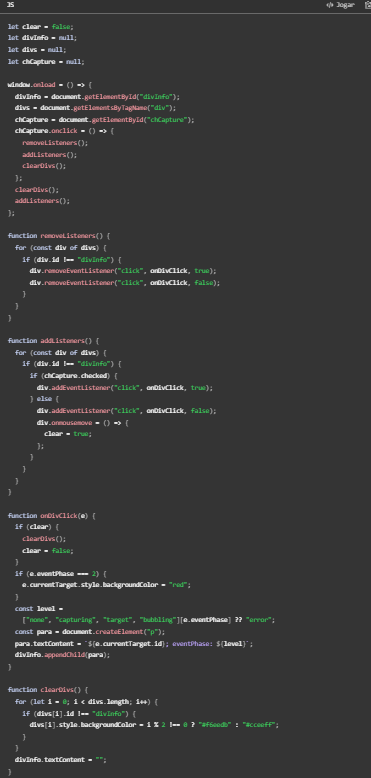


Event.eventPhase: Indica que a fase do fluxo de eventos está sendo processada.

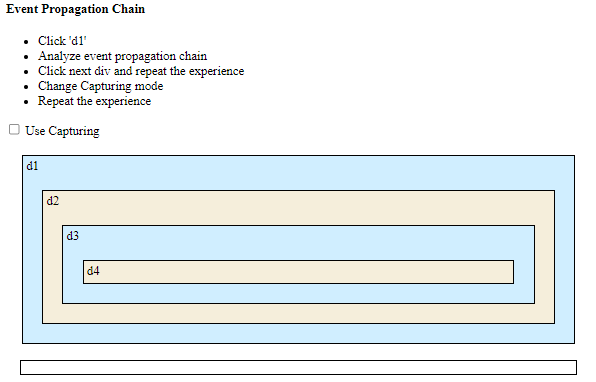




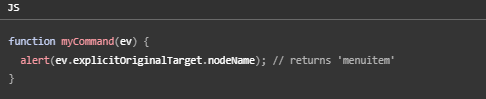
O javascript:

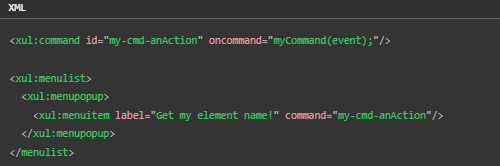


Resultado:



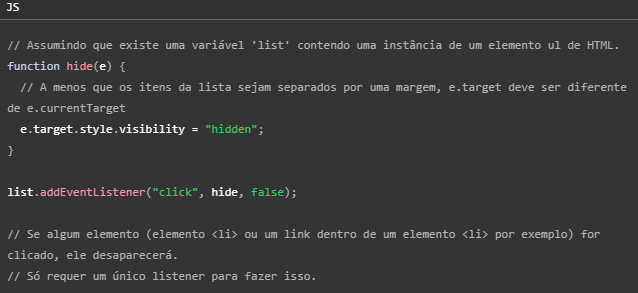
Event.explicitorOriginalTarget: O objetivo original explícito do evento (específico do Mozilla).



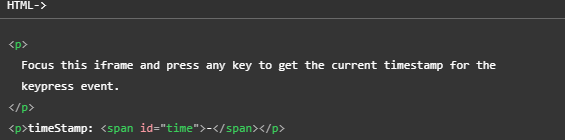


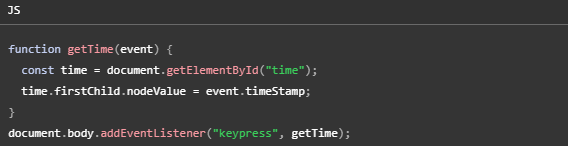
Event.target: Uma referência ao objectivo para o qual o evento foi originalmente despachado.

A propriedade **event.target** pode ser usada para implementar a delegação de eventos:

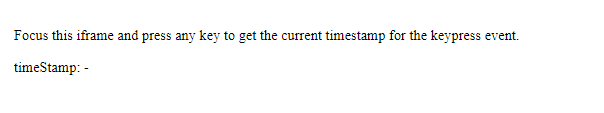


Event.timeStamp: O tempo em que o evento foi criado.

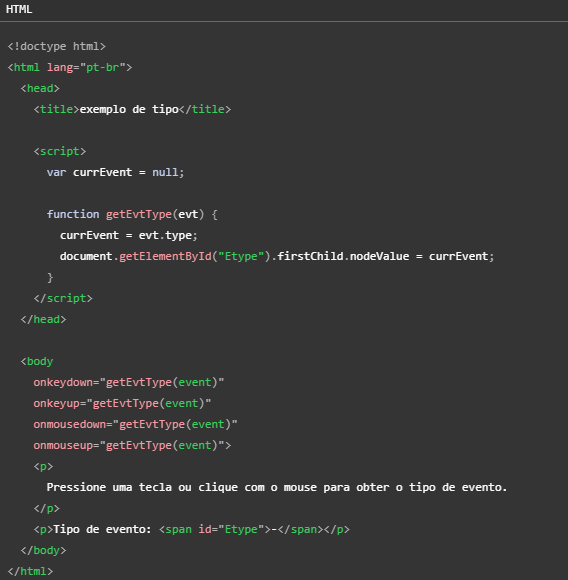




Resultado:



Event.type: O nome do evento (não diferencia maiúsculas de minúsculas).



Event.isTrusted: Indica se ou não o evento foi iniciado pelo navegador (depois de um clique do usuário, por exemplo) ou por um script (usando um método de criação de evento, como [event.initEvent](https://developer.mozilla.org/en-US/docs/Web/API/Event/initEvent) ).

