**DOM**

O DOM (Document Object Model) é a modelagem de todo o HTML, proporcionando interatividade às páginas web com a introdução do JavaScript a partir da versão 5 do HTML. Pode ser comparado a uma árvore genealógica invertida, onde o elemento mais alto é a janela do navegador (window), seguido pelo documento (document) como objeto global, tendo a tag html como raiz e suas descendências representando os elementos HTML através de ramificações.

**DOM & HTML**

O HTML é uma linguagem de marcação que define a estrutura básica de uma página da web, enquanto o DOM (Document Object Model) é uma representação em memória dessa estrutura criada pelo navegador. O HTML cria a estrutura estática da página, enquanto o DOM permite interações dinâmicas com o conteúdo HTML, como adicionar, remover ou modificar elementos e atributos usando scripts como JavaScript. Em resumo, o HTML cria a estrutura inicial da página, enquanto o DOM representa a página atual em tempo de execução, permitindo interações dinâmicas.

**Acessar um elemento específico no DOM?**

Para acessar um elemento específico no DOM, você pode usar métodos fornecidos pelo JavaScript. Aqui estão algumas maneiras comuns de fazer isso:

* **getElementById**: Este método retorna uma referência ao elemento com o ID especificado.
* **getElementsByClassName**: Este método retorna uma coleção de todos os elementos com a classe especificada.
* **getElementsByTagName**: Este método retorna uma coleção de todos os elementos com a tag especificada.
* **querySelector**: Este método retorna o primeiro elemento que corresponde ao seletor CSS especificado.
* **querySelectorAll**: Este método retorna uma NodeList de todos os elementos que correspondem ao seletor CSS especificado.

**DOMContentLoaded**

O evento DOMContentLoaded é disparado quando o documento HTML é completamente carregado e analisado, antes mesmo dos recursos externos como CSS e imagens serem carregados. Isso indica que o DOM está pronto para ser manipulado por scripts. É útil para executar código JavaScript assim que o DOM estiver disponível, permitindo inicializar variáveis, configurar manipuladores de eventos ou realizar outras tarefas dependentes da estrutura do DOM, antes que todos os recursos da página tenham sido totalmente carregados.

**Adicionar um novo elemento ao DOM**

Há um conjunto de funções que você pode usar para criar elementos inteiramente novos e adicioná-los à página.

Para criar um novo elemento, use o comando chamado [createElement](https://developer.mozilla.org/en-US/docs/Web/API/document.createElement):

var imgEl = document.createElement("img");

Para anexá-lo à página, chame [appendChild](https://developer.mozilla.org/en-US/docs/Web/API/Node.appendChild) no elemento pai de destino:

document.body.appendChild(imgEl);

Da mesma forma, você também pode usar [insertBefore](https://developer.mozilla.org/en-US/docs/Web/API/Node.insertBefore), [replaceChild](https://developer.mozilla.org/en-US/docs/Web/API/Node/replaceChild), [removeChild](https://developer.mozilla.org/en-US/docs/Web/API/Node/removeChild), e [insertAdjacentHTML](https://developer.mozilla.org/en-US/docs/Web/API/Element/insertAdjacentHTML).

**innerHTML,Finalidade no DOM & textContent**

A propriedade innerHTML em JavaScript permite acessar e modificar o conteúdo HTML interno de um elemento DOM. Retorna uma string representando o HTML de todos os descendentes do elemento, incluindo as tags HTML. A diferença principal entre innerHTML e textContent é que innerHTML inclui tags HTML e formatação, enquanto textContent retorna apenas o texto sem formatação. innerHTML é útil para manipular o HTML, como adicionar novos elementos ou alterar a estrutura HTML, enquanto textContent é adequado para obter apenas o texto puro, sem formatação.

**Remover elemento no DOM**

var elementoParaRemover = document.getElementById("id-do-elemento");

elementoParaRemover.remove();

**NodeList e HTMLCollection no contexto do DOM?**

No contexto do DOM (Document Object Model), tanto NodeList quanto HTMLCollection são coleções de nós (ou elementos) retornados por métodos de seleção de elementos. Eles representam conjuntos de elementos do DOM, mas têm algumas diferenças:

NodeList:

* Retornado por métodos como querySelectorAll() e childNodes.
* Pode conter qualquer tipo de nó: elementos, texto e comentários.
* É uma coleção estática, refletindo os nós presentes no momento de sua criação.
* Semelhante a um array, mas não é um array real; pode ser iterado ou convertido em array.

HTMLCollection:

* Retornado por métodos como getElementsByTagName() e getElementsByClassName().
* Contém apenas elementos HTML, excluindo texto e comentários.
* É uma coleção ao vivo, atualizando-se automaticamente para refletir alterações no DOM.
* Comporta-se como uma lista de elementos HTML, com propriedades e métodos específicos de coleção.

Em resumo, a diferença principal está na composição dos elementos que contêm (todos os tipos de nós vs. apenas elementos HTML) e no comportamento em relação às mudanças no DOM (estático vs. ao vivo). Escolha entre eles dependendo das suas necessidades para selecionar e manipular elementos do DOM.

**Manipular classes de um elemento no DOM**

Para manipular as classes de um elemento no DOM usando JavaScript, você pode usar métodos e propriedades fornecidos pela API do DOM. Aqui estão algumas maneiras de fazer isso:

**classList**: A propriedade classList fornece métodos para adicionar, remover e alterar classes em um elemento. Ela é especialmente útil porque fornece métodos para manipular classes sem precisar manipular diretamente a string de classes.

* + **add(className)**: Adiciona uma classe ao elemento.
  + **remove(className)**: Remove uma classe do elemento.
  + **toggle(className)**: Adiciona a classe se ela não estiver presente, caso contrário, remove-a.
  + **contains(className)**: Verifica se o elemento possui a classe especificada.

**Procedimento & Função**

Procedimentos e funções são blocos de código reutilizáveis em programação, mas diferem em comportamento e retorno:

Procedimento:

* Executa uma tarefa ou conjunto de tarefas.
* Geralmente não retorna valores, usados para ações como imprimir, manipular dados ou alterar o estado de um sistema.
* Pode aceitar parâmetros para personalização.

Função:

* Realiza uma operação específica e retorna um valor.
* Pode aceitar parâmetros para processamento.
* Não altera o estado do sistema ou realiza ações externas, apenas cálculos ou processamentos.

Em resumo, procedimentos executam ações sem retorno de valor, enquanto funções realizam cálculos e retornam resultados. Ambos são úteis para organizar e reutilizar código, escolhidos conforme as necessidades específicas de uma tarefa.