Início do curso JavaScript do iniciante ao mestre

Daniel Tapias Morales

Vanilla JS é conhecido como JS Puro!

O que é JS?

* JS é uma linguagem Interpretada e não cupilada, o Browser interpreta o código!
* Tipagem dinâmica
* Multiparadigma => Pode usar o paradigma que quiser, orientado objeto ou não
* Linguagem principal client-side( mais usadas em sites)
* ECMASCRIPT é o nome oficial de JS => São a mesma coisa.

Babel é um conversor que converte o código JS para a versa ES5 que os Browser conseguem interpretar.

Browser Evergreen => São browser que se atualiza sozinho.

Introdução a logica (revisão)

\*Consultar a compatibilidade dos browsers

Kangax.github.io/compat-table/es6

Caniuse.com

Como o JS é tipagem fraca, a variável é dinâmica, podendo receber valor de outros tipos, exemplo:

Variavel recebeu uma string, ela pode receber um valor numérico depois.

UTILIZAR ESTE MEIO DE CONCATENAÇÃO DE VARIÁVEIS:



Verificando o tipo da variável:

Typeof

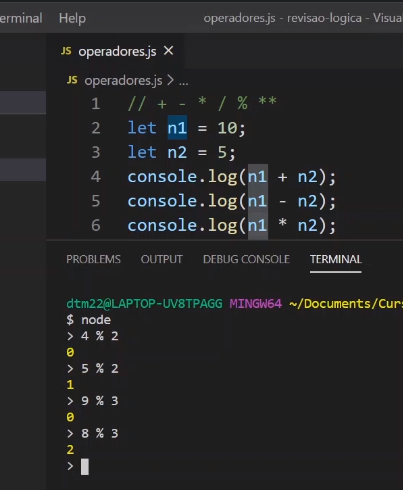
Conversão de variáveis (Numeros)

parseInt, parseFloat, Number()

Converter para string

toString

operadores aritméticos



OEPRADORES DE ATRIBUIÇÃO

+=

-=

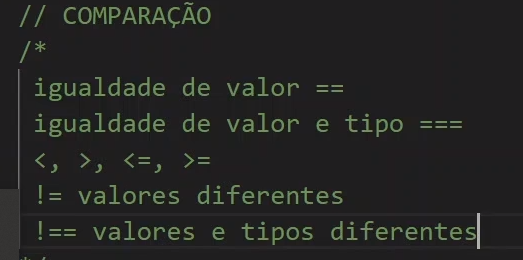
\*=

/=

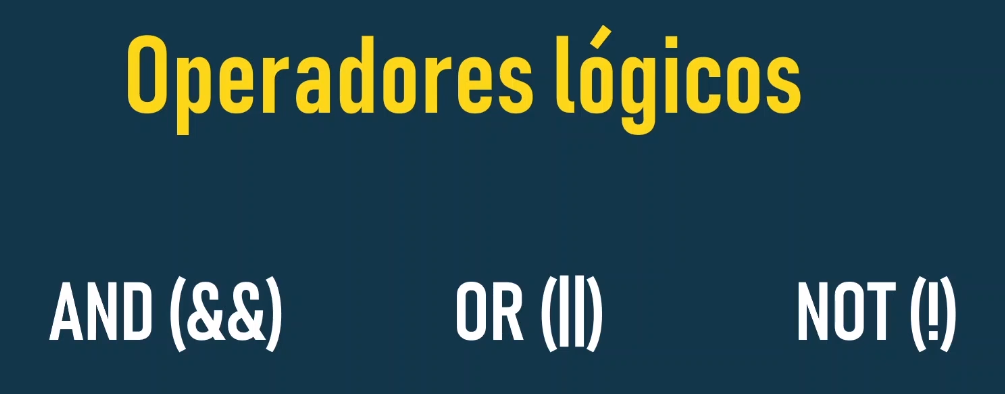
%=

\*\*= (FUNCIONA MAS NÃO UTILIZADO)

Operadores de comparação:



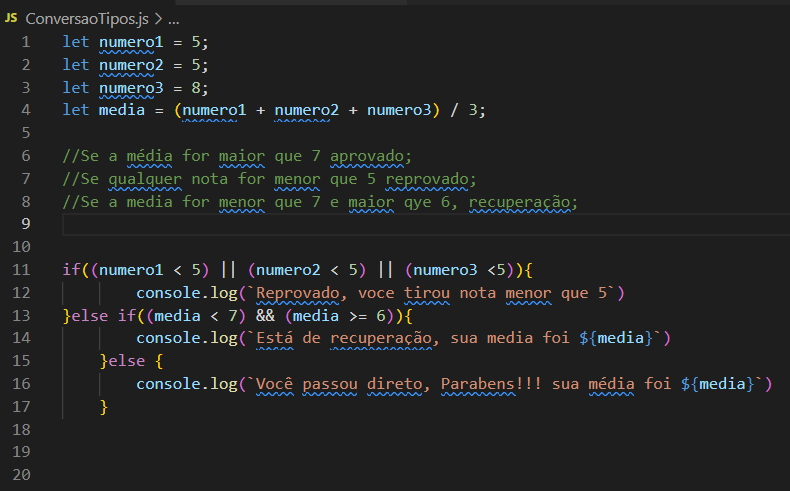
Operadores lógicos:



O OPERADOR && Tem preferencia de execução sobre os OR

Se na mesma empressão tiver comparando && e OR o && será executado primeiro.

Brincando com operadores e IF e ELSE



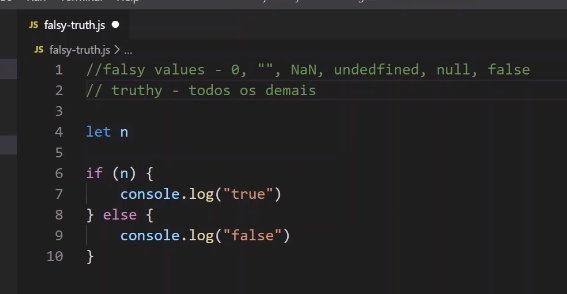
Utilizando IF na declaração de uma variável:



Declarando a variável msgMaiorIdade, ela vai receber true se idade for maior ou igual a 18 e false se for menor que 18.

Dentro da condições IF , alguns valores são considerado falsos, chamados de falsy truth

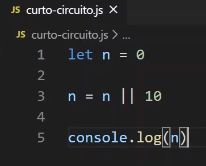
Exemplo:



CURTO CIRCUITO

Se o valor da variável n for zero a variável n recebe o valor de 10.

Como 0 é considerado FALSO e comparando com || o 10 é true, a variável recebe o valor True, retornando no console o valor 10.

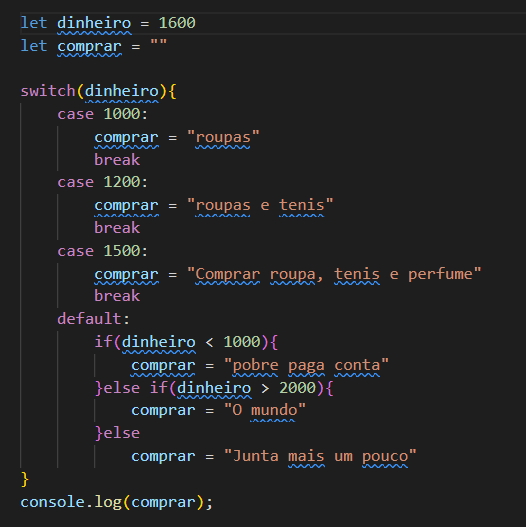


Muito cuidado com a utilização de curto circuito

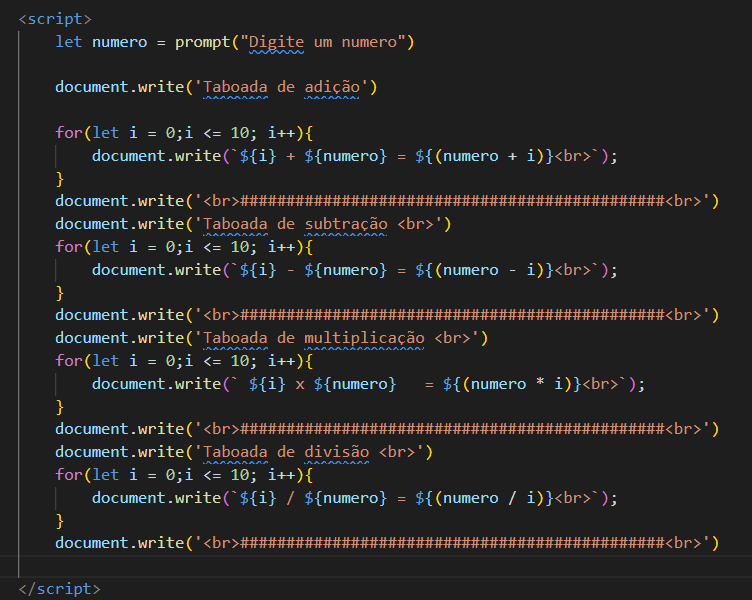
SWITCH COM IF

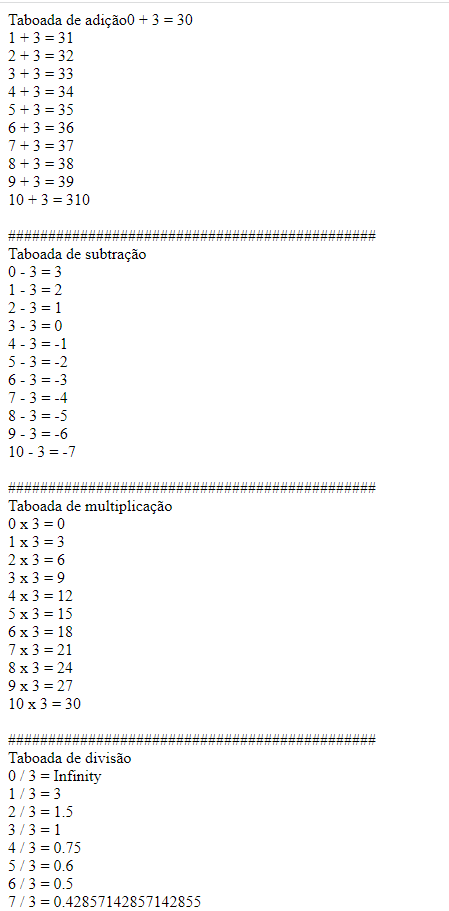
importante

* Switch compara valor da variável e também o tipo (===)



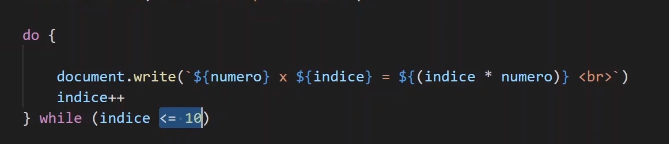
Utilizando loop para fazer uma taboada:



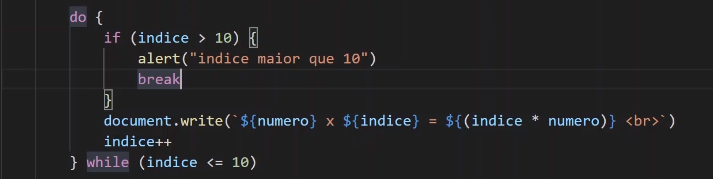


Do while

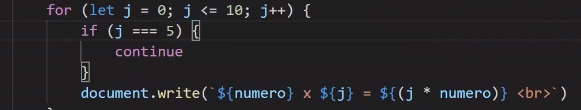
Utilizado quando você quer que rode pelo menos uma vez o código antes de faze a verificação while



Utilizando a palavra BREAK para sair do loop

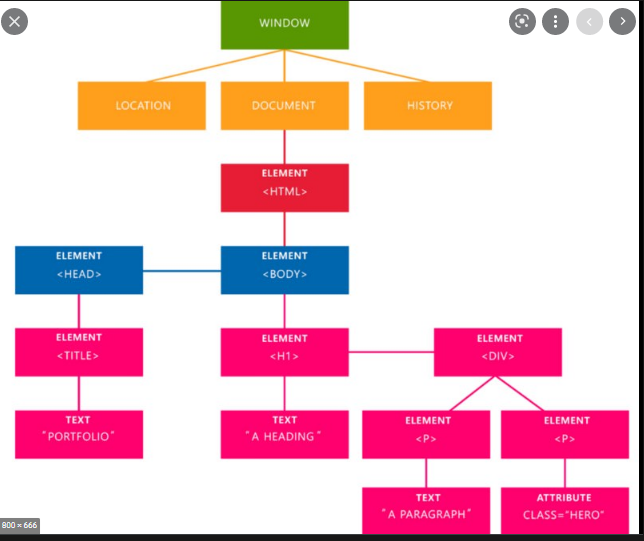


O COMANDO CONTINUE É utilizado para não executar uma parte do loop, por exemplo, ao validar que o J é 5 ele não executa e pula para o 6:



**DOM**

**DOCUMENT OBJECT MODEL**



A propriedade querySelector utiliza as propriedades do CSS ou seja para selecionar um elemento com id tem que por ( # )na frente do nome.

Ao selecionar elementos tipo array precisa selecionar o elemento dentro dele exemplo:

document.getElementsByClassName(“p”)[0].textContent

ARRAY

.every (Retorna true ou false)

Se dentro do array tiver um tipo de variável diferente do verificador o retorno será falso.

.soma (Retorna true ou false)

Se dentro do array tiver um tipo de variável igual do verificador o retorno será true.

.filter (Filtra os dados de um array)

Retorna apenas o valor especificado no filtro

.forEach (Loop mais elegante)

Retorna o que tem em cada posição do array, utilizado quando não quer usar um loop para buscar cada posição dentro do array

.map (Transforma os elementos do array de acordo com o parâmetro informado)

Retorna valores do array fazendo alguma transformação de dados de acordo com o informado para fazer.



Esta passando nesse comando para pegar todos os elementos numéricos e multiplicar cada um por ele mesmo.

Obs: arr1 já é uma função que busca no array apenas os números.

.indexof

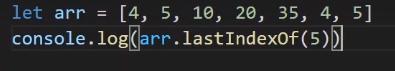
Retorna o primeiro valor encontrado dentro do array



Obs: Se não encontrar o elemento procurado o retorno vai ser -1

.lastindexof

Retorna o ultimo valor encontrado dentro do array



Obs: Se não encontrar o elemento procurado o retorno vai ser -1

.includes

Retorna falso ou true se encontrado o valor procurado na array

.find

Retorna o primeiro valor encontrado no array que satisfaz uma condição.

.findIndex

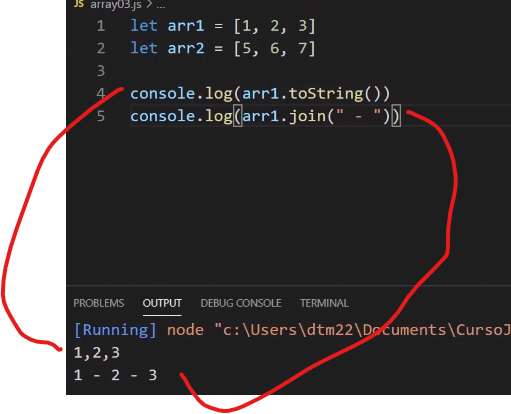
Retorna a posição do primeiro valor encontrado no array que satisfaz uma condição.

.toString

Transforma os elementos de um array em uma grande String

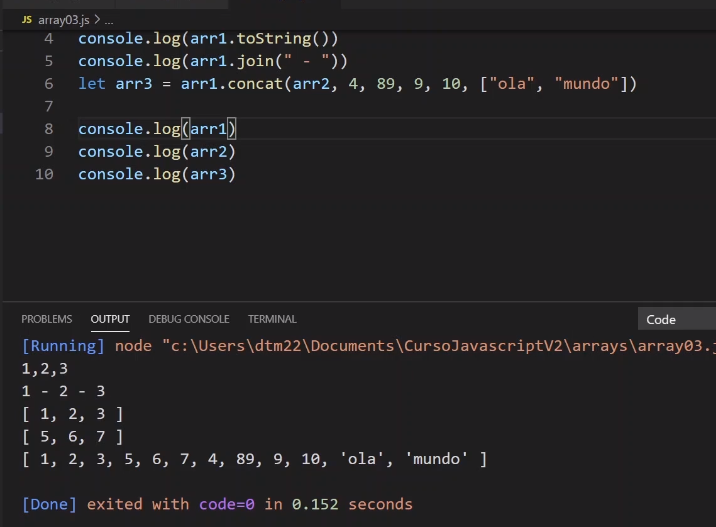
.join

Transforma os elementos de um array em uma grande String mas também coloca alguma informação dentro da string por exemplo:

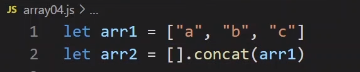


.concat (Concatenar)

Serve para juntar dois arrays.



Criando uma copia de um array utilizando concat



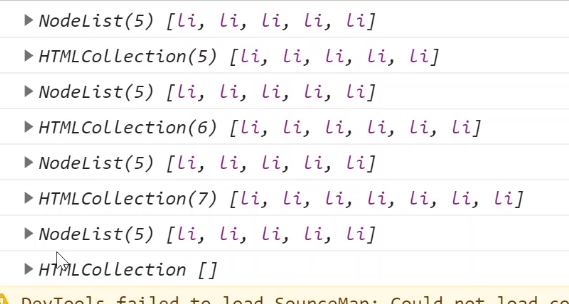
-----------------------------------------------------------------------------------------------------------------------------------------------------------

DOM Parte 2

Diferença entre nodelist e HTMLCollection?

Nodelist não atualiza com atualizações dos codigos, mas o HTMLCollection atualiza.





.ParentElement → Navega pelo DOM utilizando tags entre pai e filho.

.nextSibling e nextElementSibling → Navega entre irmãos.

.childNodes -> Vai mostrar os elementos e os textos. Mostrando todos os nós independente se for elemento ou não

.children -> Vai mostrar apenas os elementos, retornando um HTMLCollection

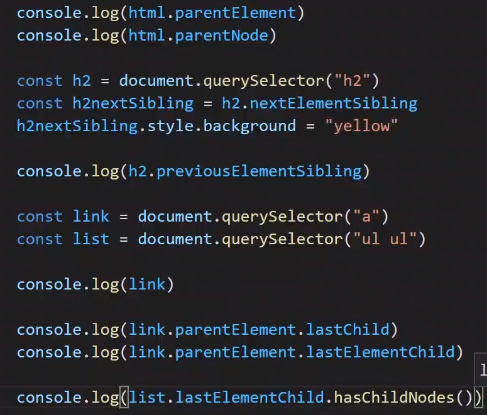
.firstChild -> Seleciona o primeiro nó.

.firtElementChild -> Seleciona o primeiro elemento.

.lastChild - > Seleciona o ultimo nó.

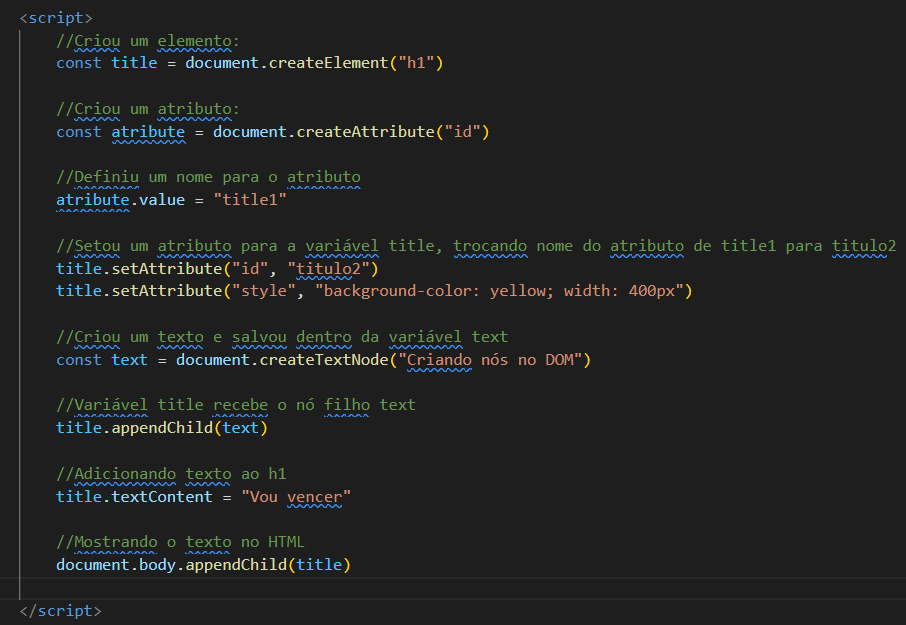
.lastElementChild -> Seleciona o ultimo elemento.

.hasChildNodes() -> Verifica se o elemento que estamos buscando tem algum nó ou não.(True or false)



CREATE DOM

Importante!

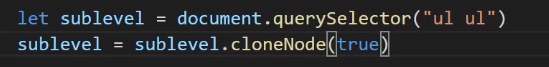


O método appendChild vai sempre inserir o nó texto depois do ultimo método do contêiner, nesse caso o conter utilizado foi o body, então antes de fechar o body ele vai inserir um h1 com texto “Vou vencer “

O método prepend é o mesmo que o appendChild mas no caso dele ele insere antes do primeiro filho do contêiner, ou seja após a abertura do body.



No exemplo acima é criado uma nova variável e adicionado com insertBefore, primero parâmetro é o próprio text e o segundo paramentro, você escolhe onde quer colocar, escolhendo o nó que quer colocar antes, nesse caso selecionou “ul”, então antes do primeiro filho ul do body a variável title2 vai entrar.



No exemplo acima ele Clonou a ul sublevel e todos seus filhos.



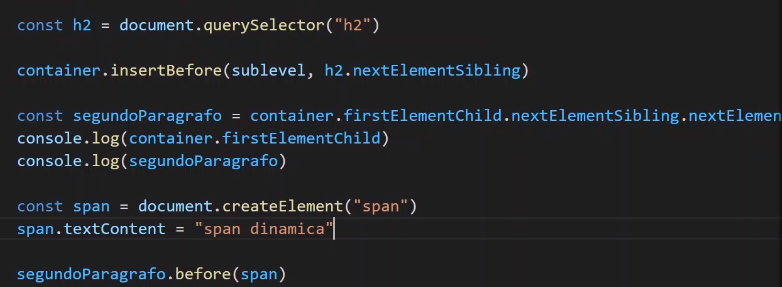
A constante segundo plano vai receber o valor do que esta dentro de container>primeiroelementofilho>dentro do próximo elemento filho > e novamente dentro do próximo elemento filho do próximo elemento filho. Resumindo:

Container\1 elemento filho\nextElemento\nextElemento

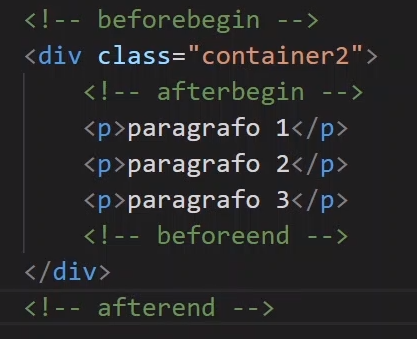


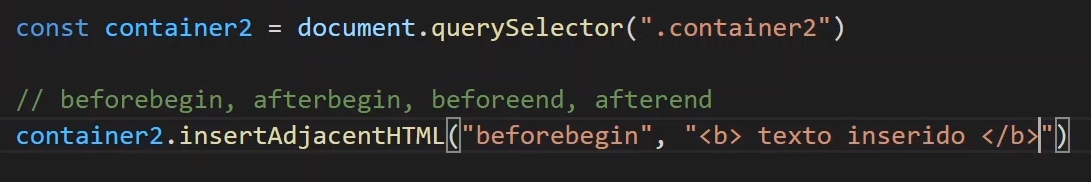
Então, inserindo o texto acima antes do nó segundoParagrafo..

Se usar .after no lugar do before, vai ser após o nó



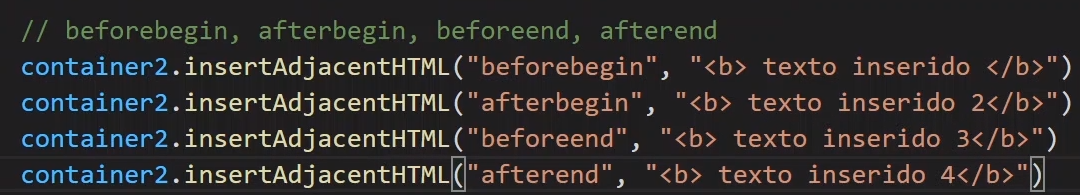
INSERT DOM



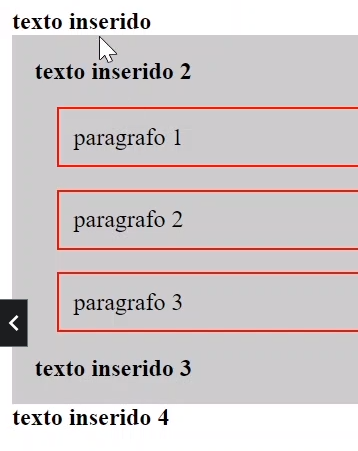


Nesses exemplos, você seleciona onde quer adicionar o nó e depois o html a ser inserido

Exemplo pratico:



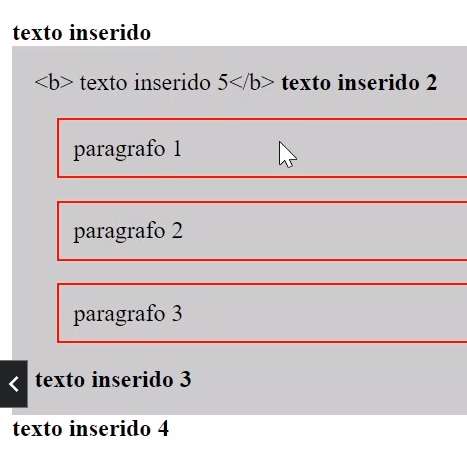
Resultado:



Existe também o modelo insertAdjacentText, ele vai inserir uma string e não uma HTML

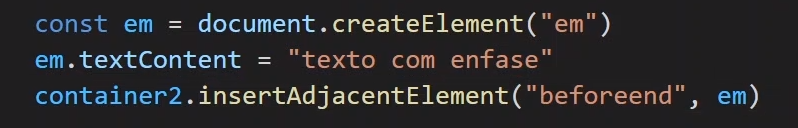


Resultado do exemplo:



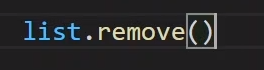
As tags html são interpretadas como texto

Para adicionar um elemento sem utilizar adjacentText, precisa criar o elemento antes e inserir com insertAdjacentElement.

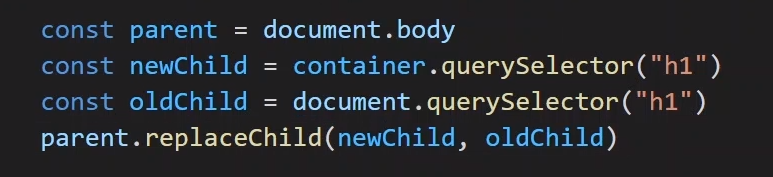


Remover elementos do DOM

Seleciona o nó .remove

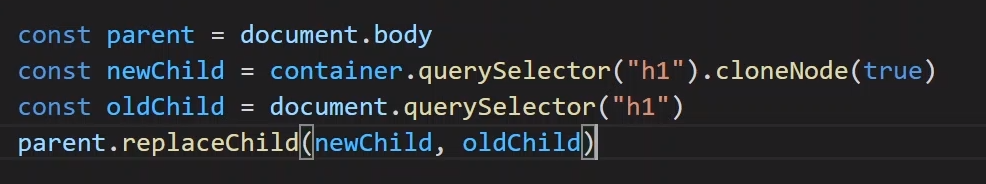


Movendo um no para o lugar do outro

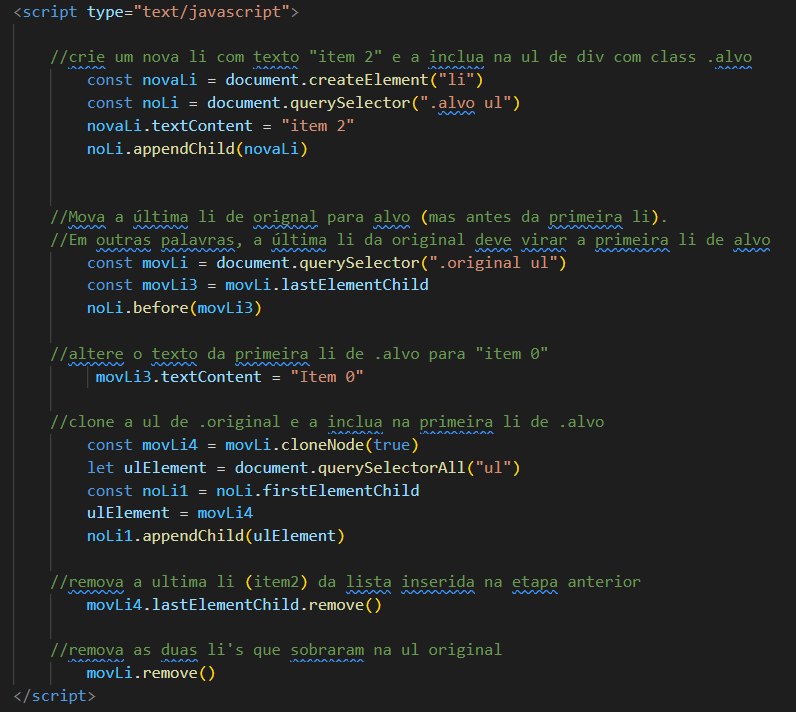


No exemplo acima o nó que estava dentro do .conteiner vai substituir o no que estava dentro de body

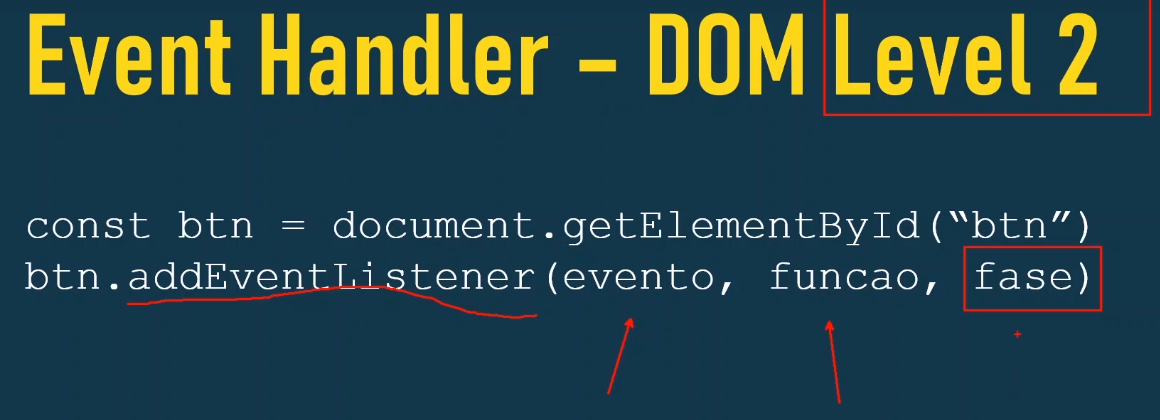
Caso queira manter o antigo novo nos dois locais, precisa clonar, ficando dessa forma



Solucionei todo o desafio:

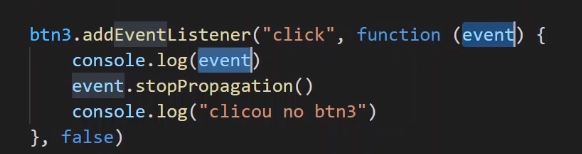


EVENTOS



Eventos add  
addEventListener

Stop propagação de evento burble



----------------------------------------------------------------------------------------------------------------

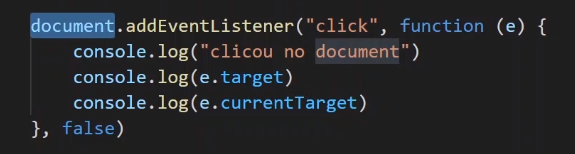
Delegando evento

No evento abaixo vemos um evento inserido no container e delegados a todos seus botoes.

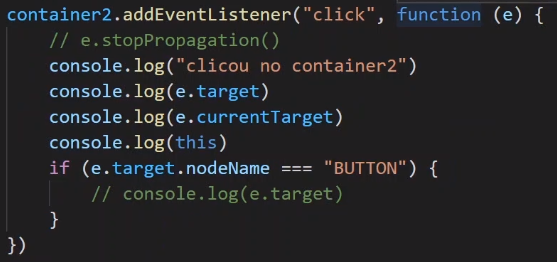
O evento ocorre apenas quando é clicado no elemento dentro do conteiner.



----------------------------------------------------------------------------------------------------------------------

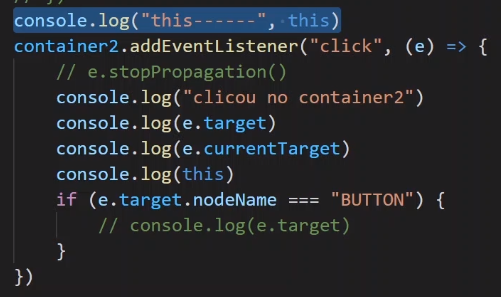
currentTarget é o próprio document no exemplo abaixo, e o target é o botão.

No exemplo abaixo o this aponta para o próprio container



this utilizando arrow function vira statico e aponta para o elemento global window

Veja abaixo



Exemplo de criação de formulário



No exemplo acima vemos a captura de dados com as constances txtTitulo e btn.

Adicionamos um evento ao btn, evento de click e adicionamos uma function com parametro (e).

Se o valor de txtTitulo for vazio, entra no if e executa o alerta. O envio de dados e prevenido com a função e.preventDefault(). E é dado focus no campo titulo para preenchimento.

---------------------------------------------------------------------------------------------------



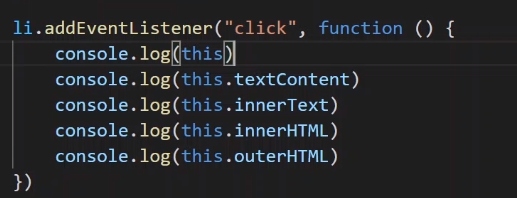
Capturando valores com as const e adicionando eventos a txtDescricao.

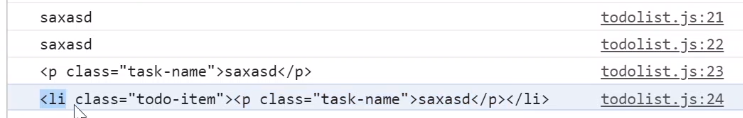
O Evento keyup e keydown é executado junto com o click de entrada e saida de dados

clicou em um tecla já executa os dois eventos.

O Indicado é utilizar o evento input, pois mesmo inserindo com os formas de copiar e colar ou mouse, o evento input ocorre certo e normalmente.

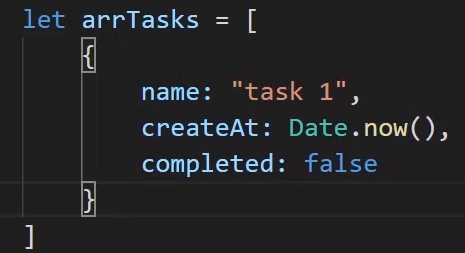
------------------------------------------------------------------------------------------------------------------------



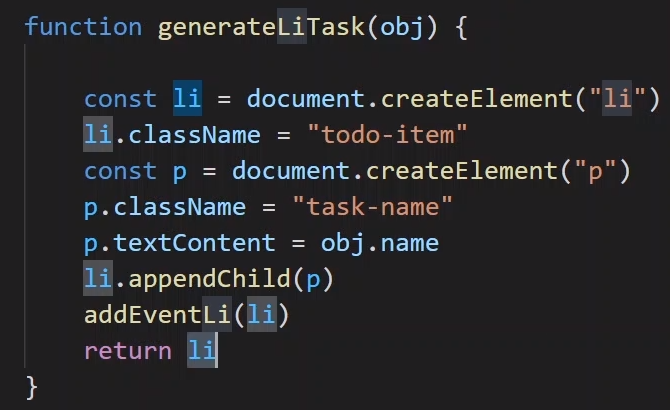


No exemplo acima é mostrado a diferença de retornos.

Aula 139. Consumir uma estrutura de dados

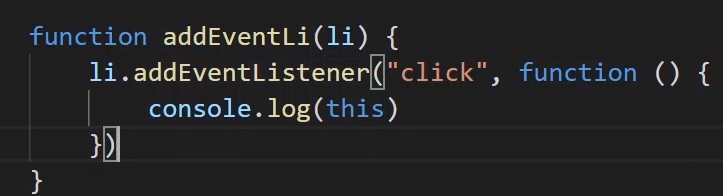


Uma estrutura construtora, composta por um array de objetos, cada posição do array tem um objeto que tem 3 propriedades, nome, data e boleano.



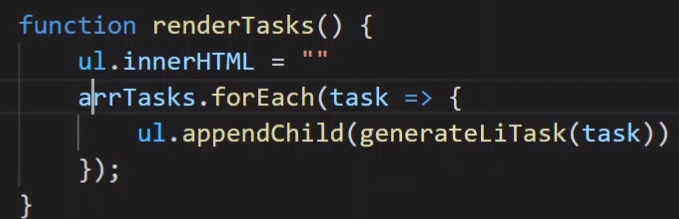
Esta função recebe o objeto do construtor acima e prepara o li, deixando pronto para inserirmos dentro da ul. Retornando assim uma li do objeto.

Simplesmente recebe a li da função de cima e adiciona um evento a ele.

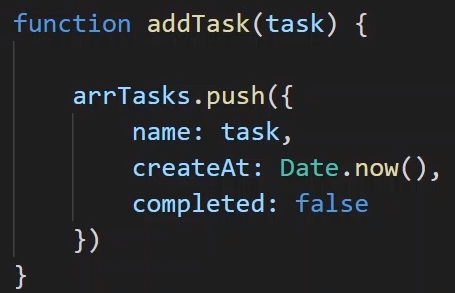


A função abaixo é responsável por adicionar os li dentro da ul.

Primeiro limpa a ul, depois chama a função generateLiTask, gerando a li e depois coloca ela dentro da ul com o appendChild.

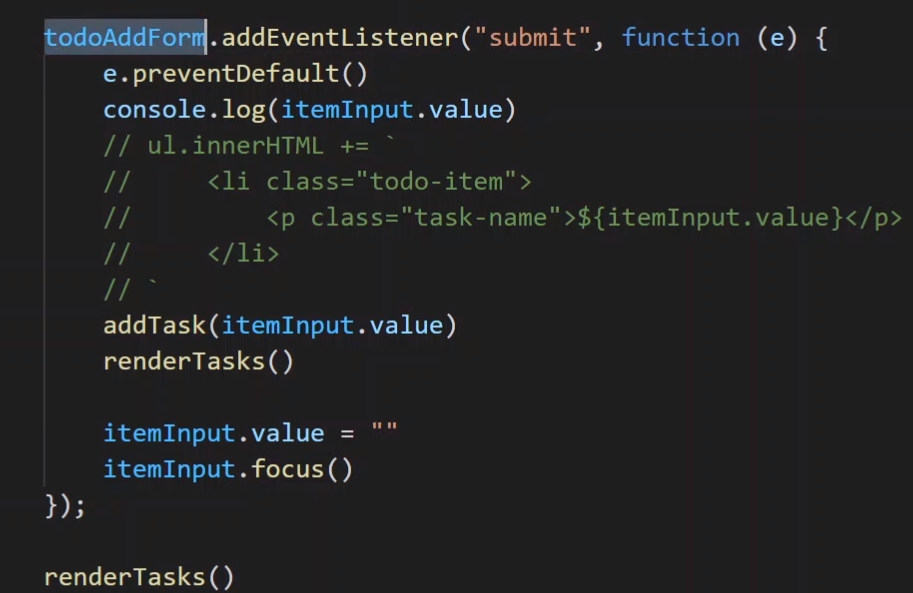


A função addTask simplesmente adiciona uma tarefa, ou seja, cria um objeto dentro do array toda vez que clica em add.



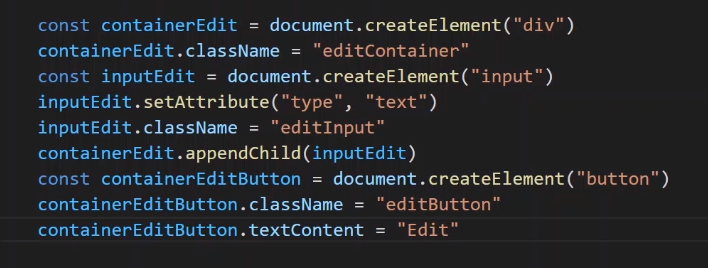
A função abaixo é o botão de adicionar tarefa, ele chama a função addTask e o renderTask() para mostrar. Depois ele limpa o campo de entrada e dá focus nele.

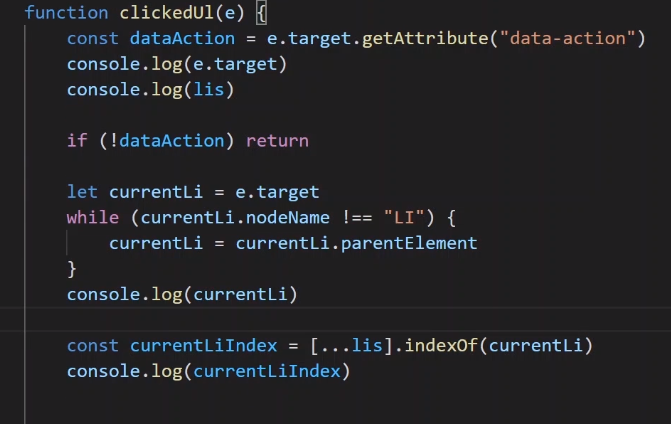
O comando e.preventDefaut evita que seja enviado informação sem ter clicado em submit.



renderTasks() é quem verifica a array de objetos e mostra todos objetos na tela.

Criando uma div e botoes para as tarefas criadas:





dataAction recebe o valor pego do atributo data-action

Se o retorno for diferente de dataAction (por exemplo se for vazio, null) já vai retornar.

Caso contrario. Variável currentiLi recebe o valor de e.target

Se o nome do nó currentLi for diferente de li, currentiLi recebe o valor de currentLi.parentElement. E o loop retorna mais uma vez.

Ou seja, o loop quer pegar o li do dataAction, então enquanto não achar o li o loop vai retornando.

A constante currentLiIndex vai receber o indice do primeiro elemento que encontra de currentLi. Exemplo.

Se dataAction for o clique no botao editar, vai retornar indice 0, porque esta na primeira posição do array, se clicar em excluir, vai aparecer o indice 1 porque esta no segundo lugar do array.

AULA 151