Anotações JS

Em JavaScript, podemos colocar ou não o ponto e vírgula no fim das linhas de código;

Podemos apenas separar blocos de código com chaves, sem necessariamente usar alguma palavra reservada da linguagem;

Sites para executar o JS sem precisar de um editor de texto:

- repl.it

- jsfiddle.net

OBS: Pode usar o console do browser também.

Tipos numéricos são do tipo number e não int, float, double e tals;

Observações (navegador):

console.log('Olá mundo !')

alert('Olá mundo !') // para a execução do código no browser até clicar no 'OK', e no node aparecerá um erro "undefined"

const confirma = confirm('Confirma ?') // retorna um valor boolean

const nome = prompt('Digite o seu nome') // como se fosse um scanf de C, porém retorna uma string

// Para inverter um array: [varA, varB, varC] = [varB, varC, varA]

OBS2: Além do document.write, também podemos usar o document.body.innerHTML(‘Adicionar um texto no HTML’), e assim adicionar um texto no html da página (no body)

LEMBRAR DO CONCEITO DE ATRIBUIÇÃO POR REFERÊNCIA OU POR VALOR:

REFERÊNCIA: ponteiro, ou seja, caso instanciarmos um objeto ou função e depois criarmos um outro objeto recebendo esse anterior, o que será recebido (atribuído) nesse novo objeto, será o endereço de memória do objeto anterior, ou seja, caso mude algo em qualquer um dos dois objetos, e chamar ele depois, esse valor será alterado.

VALOR: Caso não estejamos atribuindo a uma constante ou variável, um objeto ou retorno de uma função, o(a) mesmo(a) não receberá o endereço, mas sim o seu valor, onde caso chamemos um dos dois, o valor de cada um será diferente.

NULL OU UNDEFINED: no código

DIFERENÇAS DE VAR E LET:

VAR:

Está presente em todo código, e isso é ruim quando usamos ela em web por exemplo (maioria dos casos), pois no browser sempre que alguém utilizar essa variável declarada com var, será modificado no site inteiro (tudo no browser é window, já no node é por módulos, ou seja, cada “código fonte” terá uma variável declarada como var de um jeito).

OBS: Em funções esse conceito de variável global não se adequa, pois caso seja declarada dentro dela, fora da mesma ela não terá sido declarada.

LET:

Além da diferença dita no VAR, em LET podemos redeclarar a mesma variável mas em escopos diferentes; já no VAR, também podemos e pode ser no mesmo escopo. Exemplo:

let numero

{

let numero

}

HOISTING

Içamento de variáveis declaradas com a palavra reservada “var”, caso haja um uso dessa variável antes da sua declaração. Exemplo:

console.log(a);

var a = 2;

console.log(a);

Resposta: primeiro undefined pois haverá **apenas** esse “içamento” da declaração da variável, e não a sua atribuição, onde após a linha que foi feita essa atribuição haverá a execução do comando console.log(a) mostrando no console o número 2, já que o mesmo foi atribuído na linha anterior.

OBS: com função declarada assim: function nomeFuncao() { }; o nome da função também ocorre o hoisting.

FUNÇÃO vs OBJETO => no código

OBS SOBRE FUNÇÃO:

Como as funções podem ser declaradas no decorrer do código, sem estar em um lugar específico como em C ou em Java, caso uma variável por exemplo não tenha sido declarada dentro dela, ela será executada pois foi declarada anteriormente (e fora da função)

OBS:

const d1 = new Date(0)

const d2 = new Date(0)

No caso acima, o operador de igualdade e identidade retornarão falsos, pois está sendo comparado o endereço de memória (objeto / em java a variável no Stack aponta para a referência na memória dinâmica) // para array funciona da mesma forma \*

Já no caso: console.log(d1.getTime() === d2.getTime()) retornará verdadeiro, assim como caso seja por igualdade, pois o valor passado foi o mesmo (0), e o retorno da função é a mesma pois foi usado a mesma função

console.log(undefined == null) //retorna verdadeiro

console.log(undefined === null) // retorna falso

**DIFERENÇAS DE BROWSER PARA NODE:**

Escopo global do browser: window

Outra forma de acessar (num contexto global): this. Obs: (this === window)

var declara uma variável global que é diferente de let (já explicado anteriormente)

- Caso use var, podemos mostrar pelo window ou this assim:

var a = 3;

window.a //a resposta será 3

- Já no let e const não acontece isso acima.

- Uma função também é colocada dentro do window, ao menos que não seja atribuída a um let ou const

OBS: Uma forma de fugir do escopo global é usando objeto, onde apenas o nome do objeto estará, e assim o que há dentro dele não poderá ser chamado pelo this ou window, apenas pelo nome do objeto criado através da palavra this. Exemplo: this.nomeObjeto (onde o this não vai ser igual ao window dessa forma, mas sim ao nomeObjeto\*)

OBSERVAÇÕES

* Utilizando um number ao invés de ter uma variável fracamente tipada

Number.prototype.entre = function (inicio, fim) {

//por ser um number, será acessado o valor atual pela variável this:

return this >= inicio && this <= fim

}

//utilizando:

const imprimirResultado = function (nota) {

if(nota.entre(9, 10)){

console.log(‘Quadro de Honra’)

} else if (nota.entre(7, 8.99)) {

console.log(‘Aprovado’)

}

}

OBS2: o Math.floor(), Math.ceil() e o Math.random() podem ser bem utilizados, principalmente os dois primeiros caso precise diferenciar um float ou double de um int, pois em JS o tipo dos dois juntos é: number.

OBS3: O Math.random() retorna um valor entre 0 e 1.

OBS4: Para converter um number para string, podemos usar o .toString()

OBS5: podemos usar o document.querySelector(‘.form’); por exemplo para pegar o formulário dentro do documento como se fosse uma classe, id, nome e outros

OBS6: Podemos usar o form.onsubmit = function (evento) { //para capturar o evento submit

evento.preventDefault(); // para prevenir os comportamentos padrões

console.log(‘Foi enviado’)

}; OU usando o .addEventListener que é um recurso mais moderno do JS

function recebeEventoForm (evento) {

evento.preventDefault(); // para prevenir os comportamentos padrões

}

form.addEventListener(‘submit’, recebeEventoForm); // para escutar o evento de submit e executar o evento recebeEventoForm

Observações sobre funções:

- podemos colocar argumentos nas funções sem ter passado parâmetros na declaração dela, onde eles serão salvos na variável “arguments” como um array; apenas se usarmos na declaração, a palavra “function”.

-caso passassemos menos argumentos que parâmetros e mandarmos imprimi-los, será mostrado o valor undefined.