LÓGICA DE PROGRAMAÇÃO EM JAVASCRIPT – DEV APRENDER

O que é uma variável?

Receber um valor e ser usado em qualquer lugar do programa

Let idade = 5;

* Let – tipo da variável
* Idade – nome da variável
* 5 = o valor da variável

Como devo nomear uma variável?

* Não devo usar número no começo
* Não devo usar espaço
* Utilizar a forma camelCase
* As letras maiúsculas e minúsculas fazem diferença no javaScript

Const - Devemos utilizar o const quando não vamos alterar os valores constantemente

Tipos primitivos

* String – let nome = “Brenner”;
* Number – let idade = 28;
* Boolean – Uma condição se é true ou false
* Undefined – Quando uma variável não está definida.
* Null – Casos em que quero resetar um valor

Tipos de referência

* Objetos – Junta as informações em lugar só e eu posso acessar pelas propriedades daquele objeto.

Ex:

let pessoa = {

nome: ‘Brenner’,

idade: 28,

sobrenome: ‘Alexandria’

};

* Arrays – É um conjunto de dados que pode ser acessado por um índice (uma posição exata nesse conjunto de dados).

Ex:

Let família = [26,45,50,17];

* Function – Ela faz parte da base de todo software, vai controlar todo o fluxo de execução, toda entrada de dados e saída, por isso devemos criar uma função. Para criar uma função pode usar um verbo + substantivo = resetaCor, transformarObjeto, alterarTamanho

Ex:

Function resetaCor() {

};

Tipos de funções

1 - Função que realiza uma tarefa e não devolve nada.

Ex:

Function dizerNome() {

Console.log(“Brennin”);

}

dizerNome();

2 – Faz um cálculo ou operação e retorna algo.

Ex:

Function multiplicaPorDois(valor) {

Return = valor \* 2;

}

Let resultado = multiplicaPorDois(5);

Console.log(resultado);

Operadores

* Operadores Aritméticos - + | - | \* | / | \*\*
* Operadores de implemento e decremento - ++ | --
* Operadores de atribuição -
* Operadores de comparação
* Operadores lógicos:

OU - && - retorna true se os dois operandos forem true.

OR - || - retorna true se pelo menos um dos operandos for true.

NOT - ! – operador de negação.

* Operadores bitwise
* Falsy – Undefined, null, 0, false, ‘ ’, NaN – not a number,
* Truthy – todos os outros valores que não entram em falsy.

IF .. ELSE

Sintaxe – if (condição) {

O código que vai ser executado

}

Caso eu queira outra condição -> Else if (condição) {

O código que vai ser executado

}

Caso nenhuma das condições acima não foram contempladas -> Else {}

Switch.. Case

É uma sequência de comparação.

Switch (variável) {

Case ‘colocar a condição que eu quero’ :

Código ou ação que eu queira.

Break;

Caso a condição não seja contemplada usamos o:

Default:

Console.log(“Essa opção não contempla a operação”)

}

Loop.. For

Declaramos tudo dentro do loop for.

For(variável, condição, implemento) {

}

Loop.. While

A diferença do laço while para laço for é que devemos declarar a variável fora do loop.

Let i = 0;

While (condição, implemento) {

}

Loop.. Do.. While

O Do while vai executar pelo menos uma vez primeiro para depois analisar a condição.

Let i = 0;

Do {

Console.log(“Digitando...”, i);

I++;

} while (i < 10);

Loop.. For.. in

Serve melhor para interação sobre propriedades de objetos ou elementos de um Array.

Const pessoa = {

Nome = ‘Brenner’,

Idade = 28

};

For(let chave in pessoa) {

Console.log(chave);

}

Se eu quiser olhar o valor dentro de chave devo acessar o que foi declarado, no caso, pessoa. Ex: console.log(chave, pessoa.nome); ou console.log(chave, pessoa[‘nome’]);

loop.. For.. Of

For(let cor of cores) {

Console.log(cor);

}

Factory Functions

Vai encapsular uma informação dentro do método que cria um objeto.

Constructor Functions

A mesma coisa do Factory Functions só que de forma diferente.

String

Tipo primitivo:

Const mensagem = ‘Minha primeira mensagem’;

Tipo objeto:

const outraMensagem = new String(‘Bom dia’);

Length – o tamanho da mensagem

Template Literal

* Object {}
* Boolean – true ou false
* Template ``

Date

Para trabalhar com datas

* Const date1 = new Data();
* Const date2 = new Data(‘March 06 2019 09:30’);
* Const date3 = new Data(2019, 02, 06, 9, 30);

Introdução a Arrays

As funções que mais utilizamos

Como adicionar novos elementos:

* No início – numeros.unshift(0);
* No meio – numeros.splice(1- qual posição quero inserir,0 – caso eu quisesse excluir, ‘a’ – o que eu adicionei);
* No fim – numeros.push(5);

Como encontrar elementos (PRIMITIVOS):

* Numeros.indexOf(2);
* Numeros.lastIndexOf(1); - Mostra onde o número aparece por último.

Como encontrar elementos (REFERÊNCIA)

* Usar o .find caso for um objeto. Ex:
* Primeiro criar o objeto – marcas –
* E dps criar a função:
* Const marca = marcas.find(Function(marca) {

Return marca.none === “a”;

});

Como remover elementos:

* Para remover no final – numeros.pop();
* Para remover o primeiro elemento – numeros.shift();
* Para remover o elemento do meio = numeros.splice();

Como esvaziar um Array:

* 1° - numeros = []; - a declaração precisa ser let.
* 2° - numeros.length = 0;
* 3° - numeros.splice(0, numeros.length);

Como combinar arrays:

* Para combinar os arrays usamos o .concat
* Const combinado = primeiro.concat(segundo);

Para dividir o Array:

* Usar o método slice

Operador spread (concatenar e combinar arrays)

Const primeiro [1, 2, 3, 4];

Const segundo [5, 6, 7, 8];

Const combinado = [...primeiro,...segundo];

// caso eu queira fazer alguma mudança, só adicionar

// Const combinado = [...primeiro, ‘a’, ...segundo];

Console.log(combinado);

For each

Const números = [1, 2, 3, 4, 5];

Números.forEach ((número) => Console.log(número))

Combinando Array

// Para adicionar algum elemento entre os números usamos o .join

Const número = [1, 2, 3, 4, 5];

Const combinado = numeros.join(‘.’)

Console.log(combinado);

// Para separar uma frase por espaço e ficar somente letras usamos .split

Const frase = “Olá bem vindo ao curso”

Const resultado = frase.split(‘ ’);

Como receber dados de usuário (INPUT)

// Utilizar o prompt

Introdução a manipulação do DOM

É basicamente todo o HTML que está em uma página.