SINTAXE BÁSICA JAVASCRIPT

1. ENTENDENDO VARIAVEIS E SEUS VALORES

* Como funciona a tipagem em js – funciona como uma regra de uso de dados quanto mais forte for a tipagem, mas obrigatório é a declaração do tipo de dado. A tipagem do javascript é fraca, a declaração dos dados acontece de forma dinâmica. EX: ao criar uma variável com valor numérico ele já entende que o numero 1 será number, se o valor da variável estiver dentro de uma aspas será entendido como string.
* Tipos primitivos – As variáveis podem guardar os tipos de dados, chamados primitivos, são eles: boolean (verdadeiro ou falso), null, undefinied, number, strings, array, object, function.
* O que são variáveis – São dados que variam, como se fosse uma caixa que você guarda vários valores.
* Como se declara variável – Existem 3 modos: var (variável genérica pode ter seu valor alterado, se não tiver um valor inicial será tratado como null; let (escopo local de bloco, pode ter seu valor alterado, se não for atribuído valor será null (undefinied), a let é utilizada em escopo local. Const – escopo local de bloco, é uma variável que não tem seu valor alterado é somente leitura, e obrigatoriamente deve ter um valor inicial.
* Escopo – Define a limitação e visibilidade de um bloco código.
* Escopo global – Quando a variável é declarada fora de qualquer bloco, sua visibilidade fica disponível em todo código
* Escopo local – quando a variável é declarada dentro de um bloco de código, e sua visibilidade pode ficar disponível ou não
* **Regras de uso de variáveis – Iniciar as variáveis sempre com letras, underline ou cifrão $, nunca inicie com números, e não utilize espaços. EX :** var 1nome (errado); var nome ou var \_nome (correto)
* Não usar espaços e sim use o camelCase ou \_. EX: var nome completo (errado); var nomeCompleto ou var nome\_completo
* Não usar palavras reservadas para criar variáveis EX: var function (errado), pois já é um método um elemento nativo do js, por isso não se pode utilizar.
* **Declare as variáveis no topo do bloco de código.**
* Diferença entre atribuição, comparação e comparação idêntica.
* Atribuição – o Sinal de igualdade (=) não significa igual mas sim atribuição.
* Comparação – Se utiliza dois sinais de igual (==), ex: “0” == 0, sempre retornará um valor boolean (verdadeiro ou falso ).
* Comparação idêntica – nesta comparação não vamos apenas comparar valor mas também o tipo, nesse caso utiliza-se 3 sinais de igual (===). Ex : “0” === 0; nesse caso será falso, pois embora o valor seja igual o tipo é diferente pois um é string e o outro number.
* Operadores aritméticos, relacionais e lógicos:
* Os operados aritméticos são operadores matemáticos com valor numérico : +, -, /(divisão real), \* ,%(divisão inteira), \*\*(potenciação).
* Operadores relacionais consultam a relação entre valores :
* >maior que ;
* < menor que;
* >= maior igual a;
* <=menor igual a;
* Operadores lógicos são tipos de operadores que consultam valores lógicos e são três tipos : && “e” – que considera que todos os valores sejam true; II “ou” – considera que qualquer valor seja true; ! “não” – inverte o valor de verdadeiro para falso.

1. VETORES E OBJETOS

* O que são vetores ou arrays – são um tipo de lista, ou matriz de variável, onde cada variável possui um índice, os valores podem ser de vários tipos EX: É uma caixa e dentro dela tem outras caixas menores onde você pode guardar qualquer coisa. EX: let array = [“string”, 1, true] usa-se colchetes e pode guardar qualquer valores dentro de seu indicie inclusive outros arrays . EX:

Let array [“string”, 1, true, false, [‘array1’], [‘array2’]]. O indicie só é acessado por numero inteiro onde o 0 é o primeiro indicie e eles são separados por virgula.

* Como se manipula um array – Ao ser declarado ele traz consigo uma serie de métodos padrão do javascript para manipula-los. São eles :
  + forEach () – executa uma função para cada indicie de um array (faz uma repetição para cada item ou índice dentro de um array
  + push () – adiciona um item no final do array;
  + pop () – remove o item do final do array;
  + shift () – remove o item no inicio do array;
  + unshift () – adiciona um item no inicio do array;
  + indexOf () – retorna o índice de um valor;
  + splice () – remove ou substitui um item pelo índicie;
  + slice () – retorna parte de uma array já existente;
* O que são Objetos – Dados que possuem propriedades e valores que definem suas características, deve ser declarado entre chaves {}. EX: var xicara = { cor: ‘azul’, tamanho: ‘p’, função: tomarCafe();
* Manipular Objetos – As propriedades de objetos podem ser atribuídas a variáveis facilitando a manipulação do objeto. Isso é chamado de desestruturação. EX : var xicara = {cor: ‘azul’, tamanho: ‘p’, função: tomarCafe()}

Var cor = xicara.cor;

Var tamanho = xicara.tamanho;

Var função = tomarCafe();

1. ESTRUTURAS CONDICIONAIS

* Estrutura condicionais – são instruções para realizar determinada tarefa a partir de uma condição, seja decisão ou repetição. EX: Um jogo precisa mudar o placar toda vez que um jogador marca um ponto. Palavras reservadas:
  + IF – (SE) usada para estabelecer uma condição
  + ELSE – (SE NÃO) no caso se a condição não for atendida podemos utilizar o else.
  + ELSE IF – Casa haja mais de uma condição utilizamos else if para estabelecer outra condição.
  + NINHO DE IF – Também podemos utilizar um if dentro de outro, chamamos isso de aninhamento de if’s.
  + IF TERNÁRIO – Utilizado para fazer uma verificação em uma única linha. EX : [condição]? [instrução1]:[instrução2] //lembre-se de usar interrogação e dois pontos.
  + SWITCH/CASE – EX : switch (${Expressão}){

Case 1: ${instrucao};

Break ( é necessáro que se use o break para parar a instrução, se não ficará em loop)

* Estrutura de Repetição – São estruturas condicionais que repetem uma instrução ate que eles atinjam determinada condição.
  + FOR – Funciona como uma repetição de instrução até que a condição seja falsa. EX: for ([expressaoInicial];[condicao];[incremento]{

declaração}

* + FOR/IN – funciona como uma repetição a partir de uma propriedade. EX: for ([indice] in [objeto ou array]) { decçaracao }
  + FOR/OF – Funciona como uma repetição a partir de um valor. EX: for ([indice] of [array]) {declaração

} Obs: for/of não funciona om objects pois as propriedades variam diferente dos índice de um array que sempre serão números inteiros.

* + WHILE – Executa uma instrução enquanto determinada condição for verdadeira, a verificação é feita antes da execução. EX: var a = 0; while (a < 10) {

a++; console.log (a); }

* + DO/WHILE – Executa uma instrução ate que determinada condição seja falsa, e a verificação é feita depois da execução

1. FUNÇÕES E SUAS PARTICULARIDADES

* O que são funções – são blocos de comandos e instruções para execução de determinada tarefa: EX: function nomeDaFuncao () {

${instrução}

nomeDaFuncao ()

* Funcoes com parâmetros – As funções podem receber em sua declaração parâmetros, que servem como variáveis de estado, onde sua atribuição pode ser feita durante a chamada. EX : function nomeDaFuncao(parâmetro) {

${instrução};}

nomeDaFuncao(valorDoParametro)

1. APROFUNDANDO FUNÇÕES

* Tipos de função:
  + Função declarativas: Possuem uso mais comum, usa-se a palavra reservada function, nesse caso o nome da function é obrigatória;
  + Expressões de Funções: São funções atribuídas a funções, e a sua nomeação é opcional.
  + Arrow function: são funções de expressão de sintaxe curto, sempre serão anônimas, e não podem ser nomeadas. Ex: var função = () => {

${instrução} }

* Funções aritméticas – métodos nativos : (SERÁ FEITO UMA CALCULADORA EM JAVASCRIPT)
  + Number() – para converter valores em números
  + Prompt() – para registrar entradas de usuário
  + Alert() – para mostrar mensagem ao usuário
  + Template strings – para usar strings junto com expressões.