**Java Script**

**Primeiros passos**

O Java Script é uma linguagem de alto nível utilizada principalmente em desenvolvimento Web, trazendo mais recursos funcionais e gráficos para aplicações.

Sobre a esturutra, existe algumas boas práticas a serem mencionadas antes do desenvolvimento na linguagem.

A primeira é a **externalidade** do arquivo js. Utilizando sintaxes de Javascript em uma estrutura HTML, por exemplo, é interessante que o código esteja externo a essa estrutura, permitindo maior fluidez do código.

Para isso usa-se <script src =“nome\_do\_arquivo.js ou Caminho\_do\_arquivo/ nome\_do\_arquivo.js”> </script>

Além disso, é interessante que o código de carregamento do arquivo, ou os trechos de sintaxe estejam ao final do código, antes do fechamento da *tag* ***body****.*

*<html>*

*<title></title>*

*<head></head>*

*<body>*

*<script src = “arquivo.js”></script>*

*</body>*

*</html>*

**Entendendo variáveis e seus valores**

**Tipagem**

O JS possui tipagem fraca, ou seja, a declaração acontece dinamicamente, com a linguagem entendendo que 1 é um número e que “1”(entre aspas) é *string*.

Existem, basicamente, **oito** tipos de valores, ou **tipos primitivos**. São eles: *Boolean, null, undefined, number, String, Array, Object* e *Function*.

A declaração de uma variável ocorre da seguinte forma:

*var Nome\_da\_var = valor;*

*console.log(Nome\_da\_var); ##* Uma forma de exibir o valor é com o comando:

**Exemplo:**

*var funcao = function() {}*

*console.log(função)*

Existem **três** formas de declarar variáveis:

**var –** Que apresenta escopo global e local, seu valor pode ser alterado.

**let –** Que apresenta escopo local de bloco e pode ter seu valor alterado.

**const –** Que apresenta escopo local de bloco, não pode ter seu valor alterado. Seu tipo é somente leitura e tem valor o valor inicial obrigatório, diferente da *var* e *let*.

Em Java Script, o escopo define uma limitação de visibilidade de um bloco de código.

No Escopo global, a variável é declarada fora do código e pode ser visto na execução por todo código.

No Escopo local, a variável é declarada dentro do código (uma função, por exemplo) e só pode ser visto dentro daquele bloco específico.

*var numero = 123;*

*console.log(numero);*

Ou

*function NOME(){*

*let numero = 123;*

*Console.log(numero);*

*}*

Para Criação das variáveis, é interessante seguir algumas regras:

* Nunca iniciar com números;
* Não usar espaço nos nomes das variáveis;
* Não usar palavras reservadas
* Declarar variáveis no topo do bloco do código;
* Não usar acento ou caracteres especiais.

**Operadores lógicos, de atribuição e aritméticos.**

No JS existem alguns operadores que podem se parecer entre si, mas apresentam significados diferentes.

**(=)** Significa atribuição de valor (var nome = “Eliabe”)

**(==)** Significa igualdade, usado para comparações. (“0” == 0 -> True)

**(===)** significa comparação idêntica e considera tipo e valor. (“0” === 0 -> False [0 string === 0 numeric])

Há também os operadores lógicos e relacionais:

**(>)** (Maior que) = 7 > 3 True

**(<)** (Menor que) = 7 < 3 False

**(>=)** (Maior ou igual a) = 7 <= 8 True

**(<=)** (Menor ou igual a) = 7 >= 8 False

**(&&)** - Considera todos valores True (7 >= 8 || 8 >= 8 False)

**(||)** - Considera ao menos um valor como True (7 >= 8 || 8 >= 8 True)

**(!-)** - Nega o valor

Ainda há os operadores Artiméticos:

**(+)** Adição

**(-)** Subtração

**(/)** Divisão real

**(\*)** Multiplicação

**(\*\*)** Potenciação

**(%)** Divisão inteira

Ex.:

*console.log(a && b);*

*console.log(a || b);*

*console.log(!b && a);*

*console.log(x == c);*

*console.log(x === z);*

**Vetores e Objetos**

Vetores(arrays) são um tipo de lista, ou lista de variáveis, onde cada variável possui um índice. Diferentes tipos de valores podem ser declarados.

Ex.: *let array = [“string”, 1, True, 2.5, [‘array1’]]*

O acesso ao item do *array* ocorre por meio de índice, que começa a partir de zero.

*Console.log(array[0]) #*gual a “string”

Para manipular um arrray, algumas funções facilitam esse processo no JS. São elas:

forEach() – itera um array item a item (parecido com a função for);

push() – Adciona um item ao final do array;

unshift() – adiciona item no início do array;

pop() – Remove o item no final do array;

shift() – remove o item no início do array;

indexOf() – Retorna o índice de um valor;

splice() – remove ou substitui um item pelo índice;

slice() -retorna uma parte de um array existente.

A estrutura básica de uso é simples, basta declarar o nome da array + função separado por ponto.

Exemplo:

*let arrayteste = [“string”, 1, True, 2.5, [‘array1’]];*

*arrayteste.pop();*

*arrayteste.splice(0, 3);*

**Objetos**

Objetos são dados que possuem propriedades e valores que definem suas características e devem ser declaradas entre chaves. Parecido com o o dicionário de python.

Exemplo:

*var xicara = {*

*cor: ‘azul’,*

*tamanho: ‘pequeno’,*

*funcao: tomar\_café()*

*}*

Para manipular um objeto basta criar uma variável e atribuir a ela o nome do objeto seguido do item, separado por ponto.

Exemplo:

*let box = {numero: 10, testo: 'Teste', logico: true, matrix: ['matrix'], caixa: 'Objeto\_Interno'};*

*console.log(box);*

*var primeiro = box.numero;*

*console.log(primeiro);*

**Estruturas condicionais**

São estruturas para realizar tarefas sobre determinada condição.

Exemplo de condição com If e else e else if:

*Var jog1 = 0;*

*Var jog2 = 0;*

*Var placar;*

*if (jog1 > 0) {*

*console.log(“Jogador 1 fez gol”)*

*} else if (jog2 > 0) {*

*console.log(“Jogador 2 fez gol”)*

*} else {*

*console.log(“Ninguém fez gol”)*

*}*

Além disso, é possível aninhar condições, ou seja, um ***if*** dentro de outro ***if****,* por exemplo.

*If (jog1 < 0){*

*if (jog1 > 0) {*

*console.log(“Jogador 1 fez gol”)*

*} else if (jog2 > 0) {*

*console.log(“Jogador 2 fez gol”)*

*} else {*

*console.log(“Ninguém fez gol”)*

*}*

}

Também é possível simplificar um condicional ***if*** com o **IF Ternário**, ou seja, toda estrutura if em uma única linha.

Exemplo:

*Jog1 != -1 && jog2 != -1 ? console.log(“Os jogadores são válidos”) :*

Outra estrutura condicional é o “switch/case”.

Exemplo:

*switch (placar){*

*case placar = jog1 > jog2:*

*Console.log(“Jogador 1 ganhou”);*

*break;*

*case placar = jog2 > jog1:*

*Console.log(“Jogador 2 ganhou”);*

*break;*

*default:*

*Console.log(“Ninguém ganhou”);*

*}*

**Laços de repetição**

Estruturas condicionais que repetem uma instrução até atingir determinada condição.

Exemplos de **FOR, FOR IN e FOR OF**:

*let obj = [“val1”, [“val2”, [“val3”, [“val4”]*

*for (let índice = 0; índice < obj.length; índice++){*

*console.log(indice);*

*}*

*for (let i in obj) {*

*console.log(i)*

*}*

*For (i of obj){*

*Console.log(i)*

*}*

A diferença entre o **in** e **of** é que o primeiro, exibe o índice do item e o segundo o valor do item.

O **for of** não pode ser utilizado com objetos, caso tente, será necessário declarar junto a propriedade e o resultado será uma quebra do valor em várias linhas.

**While**

Executa uma instrução enquanto determinada condição for verdadeira, sendo a verificação feita antes da execução;

Exemplo:

*var a = 10;*

*while (a < 10) {*

*a++;*

*console.log(a);*

*}*

Há também o **Do/while** que executa uma instrução até que dada condição seja falsa e a verificação é realizada após a execução;

Exemplo:

*var a = 10;*

*do{*

*a++;*

*console.log(a);*

*} while (a < 10) {*

*a++;*

*}*

**Funções**

Funções são blocos de instrução para execução de uma determinada tarefa.

Modelo:

*Function nomeDaFunlçao(){*

*${Instrução};*

*}*

*nomeDaFunlçao();*

Exemplo de função com parâmetros:

*function teste(x, y){*

*soma = x + y;*

*divisao = x/y;*

*console.log(soma);*

*console.log(divisao);*

*}*

*teste(4, 6);*

Existem X tipos de funções:

Um tipo de funções são as **funções declarativas**, nelas é obrigatório que a função tenha um nome, caso contrário, no momento da execução será apresentado um erro.

Exemplo:

*function declarativa(x, y){*

*soma = x + y;*

*console.log(soma);*

*};*

*declarativa (4, 6);*

Outro tipo de função são as **expressões de funções** que são atribuídas à expressões, nesse caso a nomeação é opcional, pois o JS atribui à função o nome da variável.

Exemplo:

*var função1 = function (x, y){*

*soma = x + y;*

*console.log(soma);*

*};*

*declarativa (4, 6);*

O último tipo é a **Arrow Function** que são funções com sintaxe curta. Elas sempre serão anônimas e portanto não podem ser nomeadas. A estrutura é um pouco diferente também, sendo declarada com parênteses e arrow (=>)

Exemplo:

*var função2 = (x, y)=>{*

*soma = x + y;*

*console.log(soma);*

*};*

*declarativa (4, 6);*

: