**Fundamentos do JavaScript: tipos, variáveis e funções**

Fundamentos – Porque é importante vermos? Para resolvermos bugs o quanto antes

JavaScript é conhecida como a “linguagem da web”, embora tenha se desenvolvido no front-end, atualmente é mais utilizado no back-end, ou server side (lado do servidor) das aplicações web.

Usaremos durante o curso o NodeJS

**Tipos de dados**

Tem três tipos principais que são os primitivos:

* Números (number) – salva os números
* Strings (string)– salva os caracteres
* Booleanos (boolean)– para armazenar verdadeiros ou falsos
* Null – é um tipo especial, pois pode ser traduzido como “ausência de valor” e pode ser atribuído como valor de uma variável (por exemplo, se queremos verificar se uma entrada já possui um valor, avaliamos com o null)
* Undefined – também representa “ausência de valor”, porém de uma outra forma, já que o null se refere a um valor atribuído a uma variável que existe e foi iniciada, e undefined se refere ao valor de uma variável que não foi inicializada.

Criando uma constante: const nomedaconstante = valor;

Podemos usar os operadores + - \* / para as operações básicas da matemática

Math.PI = retorna o valor de pi

Math.round(valorparaarredondar) – arredonda números decimais

.toFixed(númerodecasas) – consegue controlar o número de casas decimais após a vírgula

toLocaleString() – converte um número para uma string

toLocaleString(‘pt-BR’, {style: ‘currency’, currency: ‘BRL’} – serve para transformar um número na moeda real brasileira, que tem esses argumentos:

* Style: o estilo do formato a ser utilizado, que pode usar:
  + Decimal – para representar números simples
  + Currency – colocar valores para moeda
  + Percent – para representar percentuais

Math.ceil() – retorna o maior número inteiro que é maior que o número passado – exemplo: Math.ceil(11.123)=12

Math.floor() – retorna o menor número inteiro que é menor que o número passado, exemplo: Math.floor(11.789)=11

**Ordenar números no JavaScript**

Let numeros = [0,1,2]

numeros.sort(); - essa função serve para ordernar números em uma lista array

Porém, pode ser que não ocorra da ordem esperada, e deve ser utilizada a seguinte solução

O sort recebe opcionalmente uma função de comparação que, dados dois valores, deve devolver um número inteiro:

* Se for 0 indica que são iguais
* Se for -1 indica que o primeiro valor é menor
* Se for 1 indica que o segundo valor é menor

function comparaNumeros(a,b) { if (a == b) return 0; if (a < b) return -1; if (a > b) return 1; }

lista.sort(comparaNumeros); // [1, 5, 8, 9, 10, 12, 15]

e podemos enxugar da seguinte forma:

lista.sort((a, b) => a - b); // [1, 5, 8, 9, 10, 12, 15]

**Como executar no terminal do VSCode um arquivo JS usando node**

node arquivo.js

**Ponto flutuante (números decimais)**

Podemos colocar só usando o ponto mesmo para representar essa casa decimal, de forma direta: 3.3, e se usarmos o 0,5 por exemplo, usamos apenas: .5

**Not a number (não é um número)**

Isso acontece caso uma operação matemática não tenha um número envolvido, por exemplo se usarmos uma string e multiplicamos, irá retornar esse erro

**Tipo string (tipos textos)**

Serve para guardar tipos textos, tudo o que não for número ou booleano, podemos guardar caracteres, pontuação, tudo.

Const texto1 = “Mensagem aqui”; - para criar uma string, abrimos as aspas duplas, e podemos fazer em aspas simples também, logo, para string não há diferença para o uso de aspas simples ou duplas na hora de criar uma string. Mas para fazermos uma citação, devemos abrir as aspas simples e fazer a citação em aspas duplas, e o mesmo serve da forma contrária, da seguinte forma

const citação = ‘Reirysson disse: “oi”’;

**Template string ou template literal**

Template strings são as que permitem expressões embutidas, podemos usar string multi-linhas e interpolação de string com elas, as quais são envolvidas pelos acentos gravar (` `), e podem possuir placeholders, que são indicados por um cifrão seguido de chaves (${expression}), essas, bem como o texto em volta delas são passados a uma função, uma função padrão apenas concatena as partes em uma string única, se existir uma epressão precedendo a template string (função tag exemplo) a template string é definida como “tagged template string”, no caso, a expressão tag é chamada pela template string processada, que você pode manipular antes de produzir o resultado. Para isso, temos o seguinte exemplo:

Const nome = “Reirysson”;

Const profissão = “Programador”;

Console.log(`Meu nome é ${nome} e eu sou um ${profissão}`);

**Concatenação – juntar duas strings (+)**

Podemos usar o sinal de adição para juntar duas ou mais strings, por exemplo:

Const citação = “Meu nome é: “;

Const nome = “Reirysson”;

Console.log(citação+nome); - “Meu nome é: Reiryson”;

**toLowerCase() – converter todos os caracteres da string para letra minúscula**

exemplo:

const cidade = “belo horizonte”;

const input = “Belo Horizonte”;

const inputMinusculo – input.toLowerCase();

console.log(cidade === inputMinusculo); // true

**lenght – para saber quantos caracteres uma string possui**

const senha = “minhasenha123”;

console.log(senha.length) // 13 caracteres

**const – para variáveis que não se alteram**

**Tipo booleano**

Não é um número ou texto, e ele só tem dois valores, que é true ou false, é um dos mais usados para questão de comparação, em, lações if, for, while.

= - atribui um valor a uma variável

Podemos fazer de forma direta usando no console.log por exemplo e comparar valores, strings.

**Case-sensitive – diferença de letras maiúsculas e minúsculas**

Temos algumas convenções (boas práticas e padronização) para se usar:

* camelCase – para palavras que inicia com letra minúscula e a primeira letra de cada palavra seguida é com letra maiúscula
* snake\_case – para onde todas as palavras são minúsculas
* kebab-case – para vai ter variáveis e funções
* PascalCase – para casos em que todas as palavras começam com letra maiúscula
* **Nunca colocar espaço e iniciar uma declaração com número**

**Variáveis**

Temos três tipos, que são os seguintes:

* Var – o fato de a var poder ser declarada e depois ser substituída, acaba deixando a mesma solta, não tem uma regra especifica e acaba resultando em certos eventos que não sejam previstos. É uma variável global, logo funciona em qualquer lugar que declare a variável – **é muita propensa a bugs**
* Let – É parecido com o var, pois consegue armazenar o valor, manipular, só que tem uma diferença de escopo, e o mesmo só funciona dentro do local onde foi declarado. Exemplo:
  + If(true) {

Let a = 10;

}

Console.log(a);

Dessa forma, irá aparecer um erro, pois o let a só funciona dentro do if, já dessa forma:

If(true) {

Let a = 10;

Console.log(a);

}

Irá funcionar, pois o let a está declarado dentro do mesmo laço de repetição

* Const - tem o mesmo escopo do let, pois funciona só dentro do local onde foi declarado, só que tem uma particularidade diferente do let, logo a constante não pode ser alterada, uma vez definida, será o valor que foi definido como constante, isso para números inteiros simples, mas já para objetos conseguimos acessar a propriedade e conseguir fazer a alteração, logo uma constante não pode ser reatribuída mas não quer dizer que ela não pode ser modificada se ela for do tipo que tenha propriedades, em que essas podem ser modificada. Sempre que a variável constante, temos que atribuir um valor para iniciar