**JAVASCRIPT**

**Comentários 🡪** usamos // para fazer comentários de uma linha em Javascript e /\* \*/ para fazer comentários de duas ou mais linhas. Atalhos para comentar no vscode 🡪 control + /

**Imprimir 🡪** Para imprimir resultados no console use 🡪 console.log()

**Variáveis 🡪** São atribuições que podem mudar de valor ao longo do código. Para atribuir uma variável use 🡪 var x = 1.

**Constantes 🡪** São atribuições imutáveis, ou seja, os valores não podem ser alterados ao longo do código. Para atribuir uma constante use (obs: uma boa prática é declarar o nome de uma constante em letras maiúsculas) 🡪 const PRECO = 2.

**Funções 🡪** São blocos de código que podem ser chamados para serem executados a qualquer momento. Para criar uma função faça 🡪 function nome(x,y) {

soma = x + y

return soma

}

Nesse caso, x e y são os parâmetros que serão atribuídos para a soma. Por fim, o return vai armazenar a soma de x + y como resultado da função.

**Javascript em uma página da web**

Para importar o arquivo Javascript para o arvquivo HTML, use <script src=“ ”> </script>

- **DOM 🡪** Significa Document Object Model 🡪 É a estrutura dos elementos de uma janela.

Diagrama

Descrição gerada automaticamente

**Frameworks e Bibliotecas**

São ferramentas desenvolvidas para facilitar e agilizar a programação em Javascript. Essas ferramentas disponibilizam diferentes funções que você pode usar e que você não conseguiria apenas através do Javascript. Mais populares: VueJS, Angular, React e jQuery (caindo na popularidade).

**Sintaxe e Operadores**

Teclado de computador

Descrição gerada automaticamente

Tabela

Descrição gerada automaticamente

Tabela

Descrição gerada automaticamente

Tabela

Descrição gerada automaticamente

Tabela

Descrição gerada automaticamente

Diagrama

Descrição gerada automaticamente

**Variáveis e Tipos**

1. Atribuições:

var 🡪 declara um valor dinâmico com escopo global.

let 🡪 declara um valor dinâmico com escopo de bloco. É o mais recomendado!

const 🡪 declara um valor fixo com escopo de bloco. É recomendado usar Upper case para nomear uma const.

Tabela

Descrição gerada automaticamente

1. Estruturas de Dados:

Existem dois tipos:

- Tipos primitivos: numbers / strings / boolean / null / undefined.

- Tipos compostos/não primitivos: objects, arrays.

Basicamente, tipos primitivos não tem métodos dentro deles.

Agora vamos entender sobre cada um deles:

* Strings: Comumente utilizada para textos. Os valores são declarados entre aspas ou crases.
* Numbers: Podem ser números inteiros ou decimais.
* Booleans: Sempre retornam True ou False. São chamados quando atribuímos algum tipo de comparação.
* Arrays (vetores): São listas iteráveis de elementos.

Interface gráfica do usuário, Aplicativo, Tabela, Excel

Descrição gerada automaticamente

Para adicionar um novo elemento na última posição do array use array.push() e use array.unshift() para adicionar no primeiro elemento. Use array.pop() para excluir o último elemento e array.shift() para exluir o primeiro elemento.

* Objects (dicionários): Um objeto é composto de chaves e valores.

Uma imagem contendo Texto

Descrição gerada automaticamente

Para adicionar um novo elemento dentro de um objeto faça: object.nome = “ ” ou object[nome] = “ ”. Para ver somente os valores use object.values() e para ver as chaves use object.keys().

* Empty / Undefined / Null: Todos são valores que te retornam false.

Quadro de comunicações

Descrição gerada automaticamente

**Funções**

Estrutura comum de funções:

function nome(parâmetros) {

código;

return;

}

O return para a função, ou seja, a função deixa de ser executada ao chegar no return e ele retorna e armazena um valor para essa função.

Tipos de Funções:

1. Anônima 🡪 São atribuídas a uma variável e não possuem um nome.

Texto

Descrição gerada automaticamente

1. Auto invocável 🡪 IIFE (Imediately Invoked Function Expression) 🡪 São executadas logo após a declaração da função. Possuem parênteses no final.

Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

1. Callbacks 🡪 É uma função passada como argumento para outra função.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

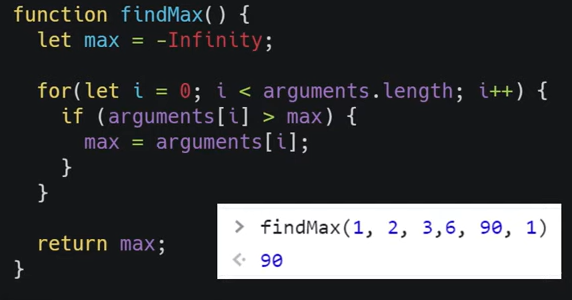
Parâmetros:

* Valores padrão 🡪 Se você quiser atribuir um valor padrão para algum parâmetro, faça:

Texto

Descrição gerada automaticamente

* Objeto “arguments” 🡪 É como um objeto array correspondendo a todos os argumentos passados para a função.



* Arrays 🡪 Para lidar com arrays como parâmetros tem a técnica spread que é uma forma de lidar separadamente com os elementos do array. A sintaxe dessa técnica é apenas colocar três pontos (...) antes do argumento.

Texto

Descrição gerada automaticamente

Nesse caso, 1 é argumento para x, 2 para y e 3 para z. Outra técnica para lidar com arrays é o rest, que é o contrario do spread, ou seja, ele combina os argumentos do array e os utiliza conjuntamente. Para utiliza-lo use três pontos no parâmetro da função.

Texto

Descrição gerada automaticamente

* Objetos 🡪 Para lidar com objetos tem a técnica Object Destructuring que é usado para filtrar apenas os dados que interessam do objeto. Para usa-lo coloque chaves e coloque a chave do valor que você quer retornar.

Texto

Descrição gerada automaticamente

**Condicionais**

1. If/else 🡪 condicional básico
2. Switch/case

Texto

Descrição gerada automaticamente com confiança média

**Loops**

1. for 🡪 Existe o “for” básico. Mas além disso existe o “for in” e o “for of”.
   1. for in:

Texto

Descrição gerada automaticamente

* 1. for of:

Texto

Descrição gerada automaticamente

1. while 🡪 While básico.
   1. do while 🡪 A diferença do “do while” para o while é que o “do while” sempre vai executar pelo menos uma vez.

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

**This**

this é uma palavra reservada no Javascript e é uma referência para o contexto.

Texto

Descrição gerada automaticamente

Tabela

Descrição gerada automaticamente

Manipulando valor do this:

1. Call 🡪 Essa função indica qual objeto você está se referindo ao usar this.

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

1. Apply 🡪 Sua funcionalidade é quase igual à da call, mas tem uma pequena diferença que é quando você vai passar argumentos para uma função eles são passados em formas de array.

Tela preta com letras brancas

Descrição gerada automaticamente

1. Bind 🡪 Ele clona a estrutura da função que foi chamado e vai aplicar o valor do objeto passado como parâmetro.

Texto

Descrição gerada automaticamente

**Arrow Functions**

Sintaxe:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Note que ela é representada por essa seta =>. Em uma arrow function, caso exista apenas uma linha, você não precisa usar o return, porque ela já presume que é pra retornar o valor após a seta =>. E caso exista apenas um parâmetro você também pode dispensar os parênteses, como no exemplo a seguir:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Obs: Arrow function não faz hoisting!

Algumas restrições:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Coleções**

1. Map 🡪 Estrutura: Possui as entradas no formato chave e valor. Pode ser declarado por new Map() e pode ser iterado pelo loop “for of”.
   1. Métodos 🡪 Para adicionar elementos no Map use o set 🡪 myMap.set(). Para ler o valor dentro do map use o get 🡪 myMap.get(). Para deletar use delete 🡪 myMap.delete().
   2. Diferença de Map e Objetos 🡪 A principal diferença é que o Map aceita qualquer valor para chaves, já os objetos aceitam apenas strings. Maps possuem a propriedade length e é utilizado quando o valor das chaves forem desconhecidos.
2. Set 🡪 Estrutura: Armazenam valores únicos (sem serem repetidos).
   1. Métodos 🡪 Para adicionar um valor no set use o add 🡪 mySet.add(). Para ler use o has 🡪 mySet.has(). Para deletar use o delete 🡪 mySet.delete().
   2. Diferença de Set e Arrays 🡪 A principal diferença é que os Sets não aceitam valores repetidos. Além disso, ao invés do length o Set usa o size. Também não possui os métodos filter, map, reduce, etc.

**3 métodos importantes para Array**

1. Map 🡪 É um método de manipulação de arrays, criando um novo array a partir de um já existente.
   1. Sintaxe 🡪 array.map(call-back, thisArg) 🡪 callback é a função a ser executada em cada elemento e o thisArg é opcional e serve para se você quiser fazer uma operação dentro da função callback.
   2. forEach 🡪 A diferença entre map e forEach é o valor retornado, pois o forEach necessita ser armazenado em alguma variável.
2. Filter 🡪 É literalmente um filtro que vai criar um novo array apenas com os valores que passaram pelo filtro.
   1. Sintaxe 🡪 array.filter(callback, thisArg)
3. Reduce 🡪 Executa uma função em todos os elementos do array e retorna um valor único.
   1. Sintaxe 🡪 array.reduce(callback, initialValue) 🡪 o callback é a função a ser executada a partir do acumulador e o initialValue é opcional e é o valor sobre o qual o acumulador vai operar.