**JAVASCRIPT**

**Objetivos**

1. **Entender a história da linguagem**
2. **Familiarizar-se com a sintaxe**
3. **Utilização de recursos básicos**
4. **Criar seu primeiro projeto**
5. **Conhecer possibilidades no mercado de trabalho**

**Aplicações**

* **Web**
* **Mobile**
* **Smartwatches**
* **Games**
* **Internet of Things**
* **APIs**

**Recursos básicos da Linguagem**

**Recursos básicos**

Para comentar o código usa o // para uma linha e /\* \*/ para varias linhas. Se apertar “**ctrl + /**” e se quiser comentar varias linhas é só selecionar as linhas e dar o comando.

Em JavaScript temos o termo variável representado por **“var”**, como o nome diz ele pode ser variado.  
e temos a constante representado por **“const”**, onde não é possível alterar o valor pois como o nome diz é constante. Como boa pratica para leitura de código ao usar uma constante coloca o nome dela em letras maiúsculas.

**Funções**

Para declarar uma função usa a expressão “function” e em seguida dá o nome dela ex: fuction soma (a + b) {   
aí você pode usar 2 opções para mostrar o valor o console.log () para mostrar o valor no console ou o return para que ele retorne o valor. Para chamar a função você digita o nome dela ex: soma (3, 5).

**Console**

**Executando um arquivo .js**

**JavaScript em uma página da web**

Para rodar o console é necessário que você instale o Node.js na sua máquina, no vscode para que tenha a saída no console é necessário que use o comando console.log e depois no terminal digite node e o nome do arquivo que quer rodar, ex:

Crie um arquivo com nome teste.js

No vscode digite:

console.log (“Olá Mundo!”)

e no terminal:

node teste.js

Assim irá aparecer a mensagem de Olá Mundo! No seu terminal.  
Lembrando que o seu terminal precisa estar na pasta que o arquivo está salvo.

**Estrutura de projeto** É comum em projetos separar os arquivos por pastas, css, js e no html criar um arquivo index e deixar ele separado das outras pastas.

**Inserindo JavaScript em uma página HTML** digite o código

script *src*="./assets/js/scripts.js"/script

**Interagindo com elementos DOM** Ele mostra tudo o que tem disponível naquele documento, ou seja, a estrutura de elementos dentro da janela.

Para trocarmos a cor de uma fonte.

Abra a página clique com o direito e vá em inspecionar, em seguida console e digite:

document .getElementsByTagName('h1')

var heading1 = document.getElementsByTagName('h1')[0]

heading1

<h1​Olá, Mundo!​/h1>​

heading1.style.color = 'red'

'red'

**Variáveis**

**Let** só funciona dentro de um bloco. Sempre que usar o let tem que declarar ele antes.

**Var** é uma variável que pode ser utilizada dentro e fora do bloco.

**Constantes**

São declaradas em SNAKE\_UPPER\_CASE, tem Escopo de bloco e não faz hosting. Não pode reatribuir uma constante e também não pode redeclarar.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Var | const | let |
| Escopo | Global ou local | BLOCO | BLOCO | |
| Redeclarar | SIM | NÃO | NÃO | |
| Reatribuir | SIM | NÃO | SIM | |
| Hoisting | SIM | NÃO | NÃO | |

**Estruturas de dados**

Strings Utilizado para textos

**Tipos de função**

Usa-se a function para declarar uma função, variáveis criadas dentro de uma função só podem ser utilizadas dentro dela.

Return serve para a função parar de ser executada.

**Função anônima** uma variável pode armazenar uma função. Ela não tem nome e usa ela na declaração da variável ou constante.

**Função auto invocável** uma função anônima entre parênteses, seguida por outro par de parênteses, que representa sua chamada.  
(  
 function() {

Let name = “DIO”  
 return name;  
}

)();

Também pode ser utilizada com parâmetros ou armazenar em uma variável.

**Call-backs** é uma função passada como argumento para outra função, utilizando call-backs, você tem maior controle da ordem de chamadas.

**Parâmetros**

**Valores padrão** para declarar o valor nulo e só atribuir o número 1 na hora de declarar a variável;

**Objeto “arguments” Um** array com todos os parâmetros passados quando a função foi invocada.

**Arrays**

**Spread:** uma forma de lidar separadamente com elementos, o que parte de um array se torna um elemento independente.

**Rest:** combina os argumentos em um array. O que era um elemento independente se torna parte de um array.

**Objetos  
Object Destruction:** entre chaves {} podemos filtrar apenas os dados que nos interessam em um objeto.

**Loops**

**If** / **Else** declaração 1 é feita no if e caso seja falso colocamos o else para mostrar um ou outro. No JS não existe elseif só as palavras separadas.

**Switch:** Equivale a uma comparação de valor. Sempre precisa de um valor default, ideal para quando precisa comparar muitos valores.

**For:** Loop dentro de elementos iteráveis (arrays, Strings).

**While:** Enquanto, ele executa instruções até que a condição se torne falsa.

**This**

**O que é this?** A palavra reservada this é uma referencia de contexto. Para retornar aquilo que você quer.

**Manipulando seu valor**

**Call** envia o valor que você chama. É possível passar parâmetros para essa função separando-os por virgulas.

**Apply** da mesma forma que o call ele chama o objeto, na hora de passar os argumentos ele passa apenas dentro de um array.

**Bind** ele clona a estrutura da função e aplica o valor do objeto como parâmetro.

**Arrow function**

Ela diminui uma linha de código usando um =>, caso exista apenas uma linha, você pode dispensar as chaves e o return, caso exista apenas um parâmetro pode dispensar os parênteses. A Arrow function não faz Hosting! Outras restrições são no this, não existe o objeto “arguments” e no construtor também não pode ser utilizado.

Sets são estruturas que armazenam apenas valores únicos.

Métodos adicionar add consultar has e deletar delete

Set possui valores únicos

Em vez da propriedade length consultasse o numero de registros pela propriedade size

Não possui os métodos map, filter, reduce etc.