Olá! Seu tiverem dúvidas, me procurem no Teams: **Eduardo Coutinho <e.coutinho@avanade.com>**.

O objetivo desse texto é dar um “ponta pé inicial” na inclusão de vocês nos projetos de Node, vou repassar alguns conceitos básicos, alguns vocês já devem dominar, mas são conceitos bem importantes para ter uma fundação sólida de aprendizado.

Contents

1 Habilidades

Habilidades importantes para exercitar e evoluir constantemente:

- inglês
- curiosidade
- persistência
- aprendizado contínuo

Importante:

- foco em estudar uma coisa por vez

Existem mil assuntos diferentes borbulhando, sempre vai ter uma live, um vídeo, um curso, um livro ou imersão de uma coisa diferente, mas é importante não ir trocando de foco, escolher um só objetivo de estudo e se aprofundar nele.

Para o nosso projeto aqui, acredito que o mais importante é estudar Javascript principalmente. Depois o básico de Node, como utilizar o Docker e como escrever APIs Node com MonboDb e EventBus.

2 Caminho aprendizado programação

Conceitos importante para sempre estar de olho e ir evoluindo:

- lógica de programação
- primeira linguagem (boas opções são: javascript ou python)
- estrutura de dados
    - Vídeo Fabio Akita: https://www.youtube.com/watch?v=9GdesxWtOgs
- algoritmos

Esses são tópicos, para sempre ter em mente que precisam ser aprofundados. Existem sites de desafios de programação como HackerHank (https://www.hackerrank.com/) que trazem desafios interessantes para praticar questões de estrutura de dados e algoritmos. Apenas isso, tenha em mente que depois é importante sempre repassar esses conceitos e aperfeiçoar.

Lembrete: Para **todos** sem exceção, inclusive os que trabalham anos com uma tecnologia, sempre existe muito a aprender, então é importante aprendizado contínuo, interesse e humildade para aprender e ensinar.

3 Instalação

Abaixo uma lista das ferramentas importantes para instalar no seu computador.

Dica: Uma sugestão que faço sempre é utilizar o Windows em inglês (é possível mudar a linguagem no painel de controle), assim já vai praticando inglês e é muito mais fácil encontrar referências de um problema na internet quando a mensagem de erro está em inglês.

3.1 vscode

https://code.visualstudio.com/

Segundo pesquisa anual do StackOverflow é a IDE mais utilizada, instalem e procurem vídeos de como utilizá-la, como instalar plugins, teclas de atalho, etc.

3.2 git e github

Criem um repositório pessoal de vocês no github.com e adicionem os projetos mesmo que de estudo lá, para ir praticando o uso.

Importante instalar o git bash localmente: https://git-scm.com/downloads e praticar a execução de ações via linha de comando: git commit, git push, git pull, git branch, git merge. Esses sãos os comandos essenciais, deem uma pesquisada em um tutorial básico de uso deles.

3.3 WSL2 e Windows Terminal

Uma forma de rodar o Linux, como prompt, diretamente no Windows é utilizando o WSL2 (Windows Subsystem for Linux)

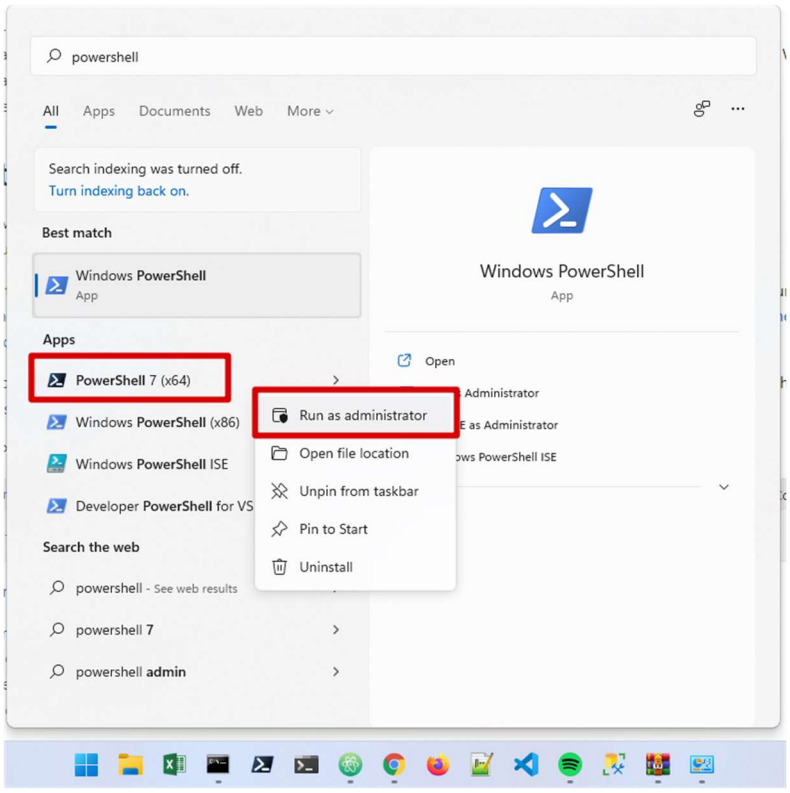
Documentação oficial: https://docs.microsoft.com/en-us/windows/wsl/setup/environment#set-up-your-linux-username-and-password

Instalando o WSL2
Como requisito é necessário estar rodando Windows 10 versão 2004 ou superior (Build 20262 ou superior) ou Windows 11

Para verificar a versão do seu Windows, faça o seguinte: Iniciar / Executar / e digite “winver”

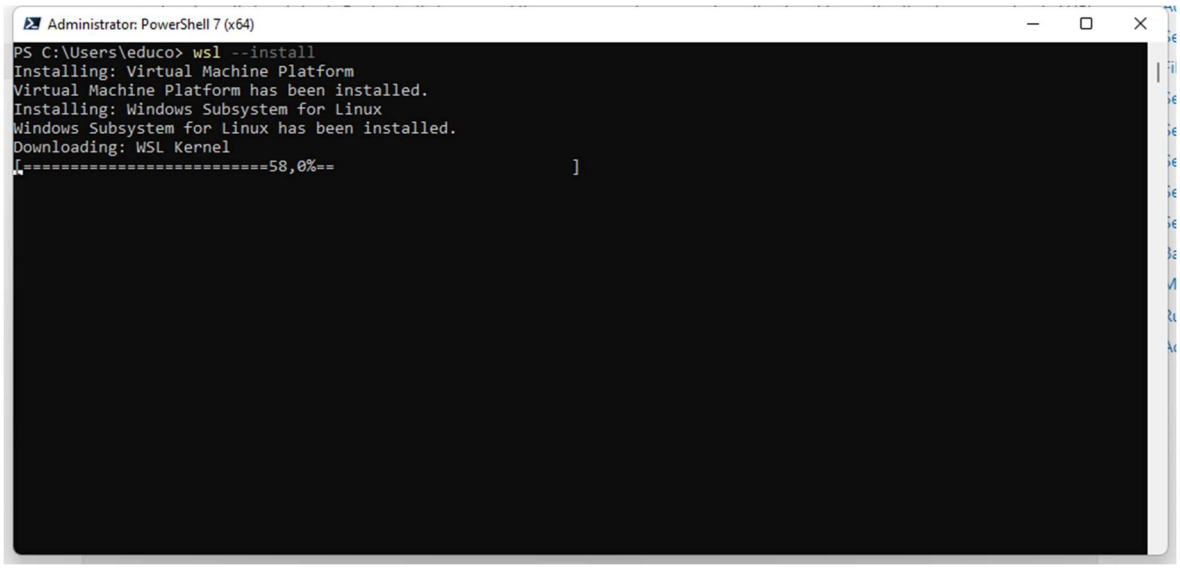
Se a sua versão não for compatível, execute o Windows Update. Se for compatível, para executar a instalação faça o seguinte:

Abra um prompt do PowerShell (como administrador):



e então execute:

```
wsl –install
```

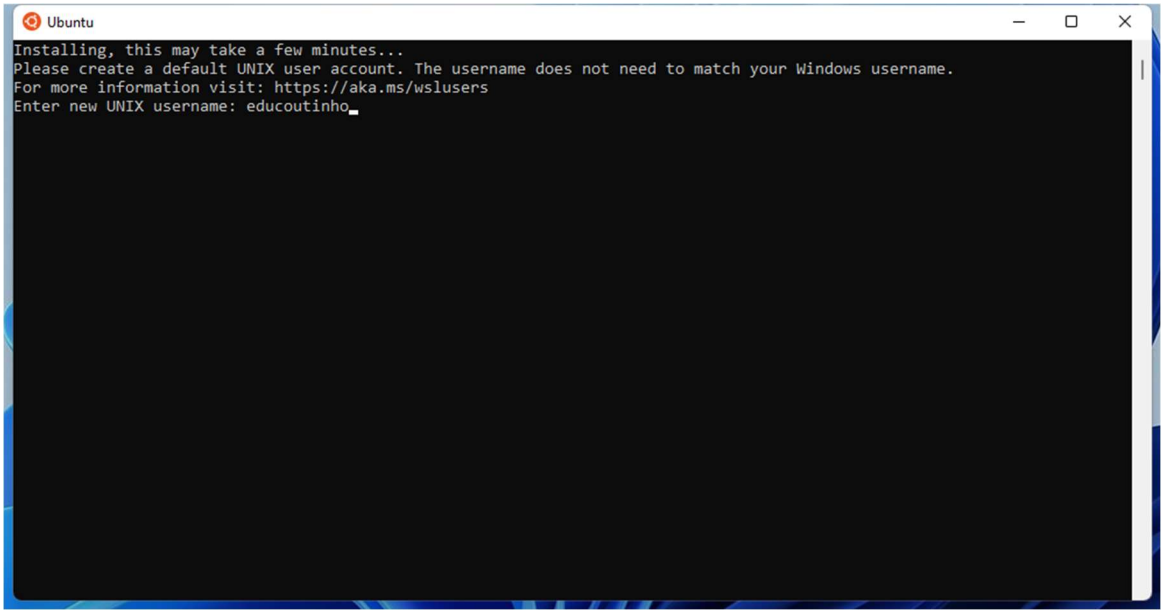


Por padrão, será instalado a distribuição Ubuntu do Linux

Quando o processo for finalizado, reinicie o computador.

Configurando a senha do usuário

Depois de reiniciar o computador será solicitado informar usuário e senha para a instalação linux:



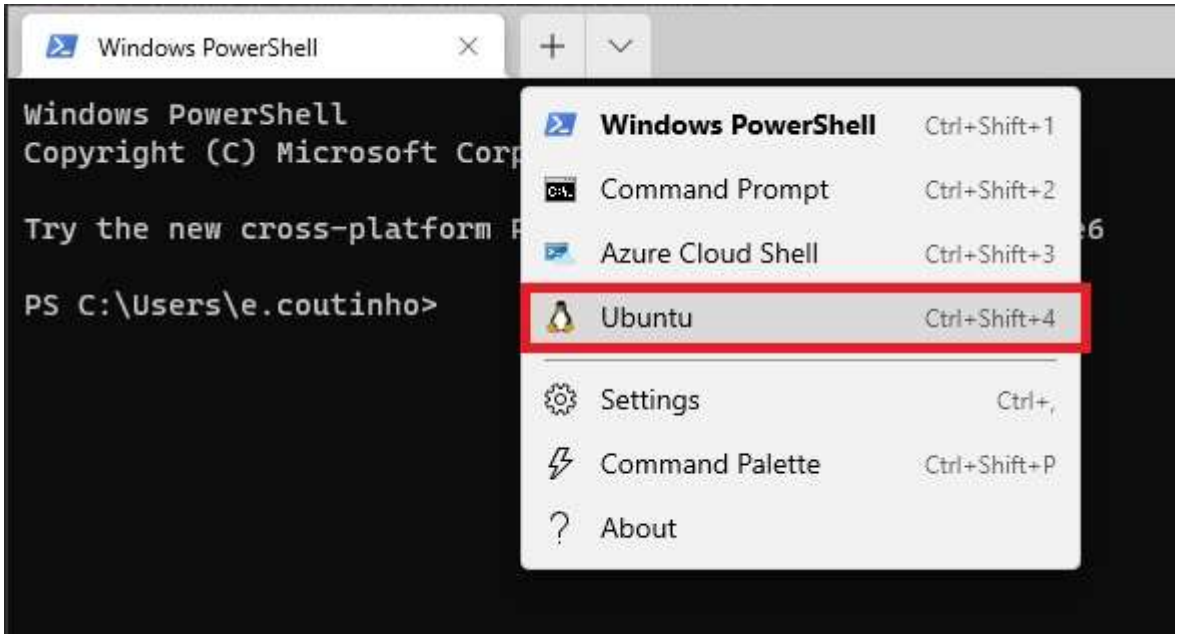
NOTA: Se essa janela não aparecer automaticamente após reiniciar, acesse: Iniciar e pesquisa por “Ubuntu”

Informe o usuário e senha que deseja utilizar.

Instalando o Windows Terminal

Baixe e instale o Windows Terminal: https://docs.microsoft.com/en-us/windows/terminal/get-started

Então, no Windows Terminal abra um prompt do Linux:



Atualizando os pacotes

No terminal, execute:

$ sudo apt update && sudo apt upgrade

Deem uma pesquisada sobre como utilizar o WSL, ele é muito útil. Que tal tentar criar um arquivo no Linux utilizando o editor **nano**?

3.4 Node com NVM

É possível baixar o Node e instalar ele diretamente no Windows, no entanto pode ser que seja necessário trabalhar com diferentes projetos, com diferentes versões do Node, no mesmo computador. Por isso, uma sugestão é utilizar o NVM, que é um gerenciador de versões do Node instaladas.

Instale o NVM
Se já tiver o Node instalado diretamente no Windows, desinstale o Node primeiro.

São duas etapas: 1- Instalar o NVM 2- Instalar o Node utilizar o NVM

1- Instalação do NVM
Onde você vai instalar? 1.1 - No Windows 1.2 - No WSL rodando no Windows 1.3 - No Linux

Se você está rodando no Windows uma sugestão é instalar tanto no Windows (1.1) quanto no WSL (1.2), cada ambiente é separado.

1.1- Instalação do NVM no Windows

Faça o download do arquivo nvm-setup.zip no site do NVM e execute o instalador:
https://github.com/coreybutler/nvm-windows/releases

Para testar a instalação, abra um prompt e execute:

```
nvm --version
```

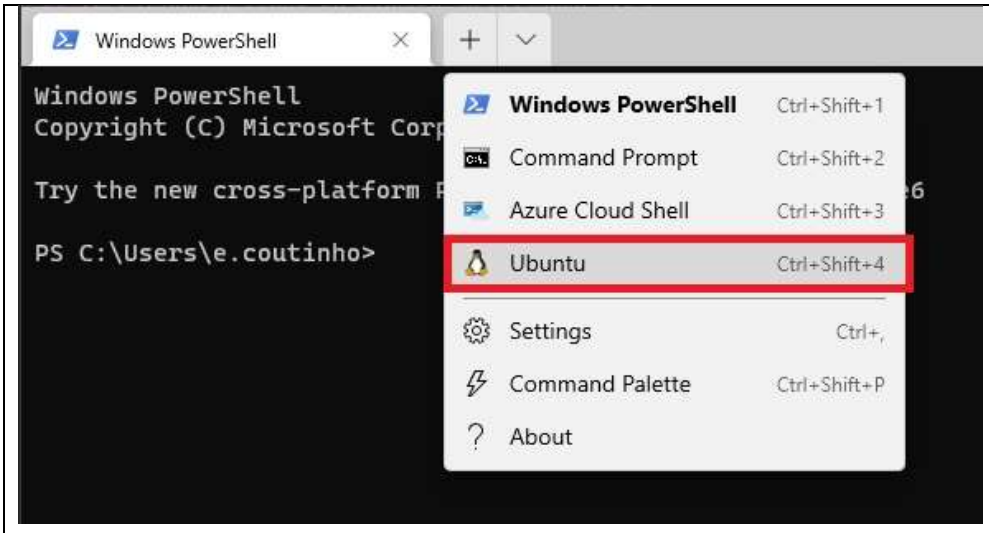
Agora, siga os passos do tópico abaixo: Instale o Node usando o NVM

NOTA: Você pode ter o node instalado tanto no Windows, quanto no Linux rodando dentro do WSL, basta repetir o processo nos dois ambientes.

1-2- Instalação do NVM no WSL rodando no Windows

Veja aqui instruções de como instalar o WSL: Rodando o Linux junto com o Windows

Então, abra um prompt do WSL:



Para instalar o NVM, execute:

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash
```

Nota: Por favor pegue o link da versão mais recente no site: https://github.com/nvm-sh/nvm#installing-and-updating

Depois execute:

```
export NVM_DIR="$([ -z "${XDG_CONFIG_HOME-}" ] && printf %s "${HOME}/.nvm" || printf %s
"${XDG_CONFIG_HOME}/nvm")"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
```

Para verificar a instalação execute:

```
nvm --version
```

Agora, siga os passos do tópico abaixo: Instale o Node usando o NVM

1.3- Instalação do NVM no Linux

Para instalar o NVM abra um prompt de comando e execute:

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash
```

Nota: Por favor pegue o link da versão mais recente no site: https://github.com/nvm-sh/nvm#installing-and-updating

Depois execute:

```
export NVM_DIR="$([ -z "${XDG_CONFIG_HOME-}" ] && printf %s "${HOME}/.nvm" || printf %s
"${XDG_CONFIG_HOME}/nvm")"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
```

Para verificar a instalação execute:

```
nvm --version
```

Agora, siga os passos do tópico abaixo: Instale o Node usando o NVM

2- Instalação do Node utilizando o NVM

Agora, utilizando o NVM vamos instalar o Node:

Instale o Node no Windows utilizando o NVM

Acesse o site do Node: https://nodejs.org/en/ e verifique qual a última versão LTS disponível.

No meu caso a última versão LTS disponível é a 16.14.0, então vou instalar ela.

Abra um prompt de comando e digite:

```
nvm install 16.14.0
```

O NVM irá baixar a versão do Node e instalar. Depois da instalação execute:

```
nvm use 16.14.0
```

Então, a versão estará pronta para uso, você pode testar:

```
node --version
```

Repetindo o mesmo processo “nvm install [version]” / “nvm use [version]” você pode instalar outras versões.

Mais informações sobre a instalação do Node usando NVL no Windows, aqui: https://docs.microsoft.com/en-us/windows/dev-environment/javascript/nodejs-on-windows

Mais informações sobre como usar o NVM, aqui: https://github.com/coreybutler/nvm-windows

3.5 Instalar o podman

O Docker é uma ferramenta essencial para trabalhar nos projetos, no entanto, em janeiro de 2022 o Docker Desktop alterou os termos de licença e agora ele não é mais free para uso corporativo, por isso fomos orientados a remover o Docker Desktop de nossos computadores. Uma alternativa é instalar o podman no WSL.

Instalação do Podman (Windows)

O site oficial do Podman é: https://podman.io/

As instruções de instalação no Windows estão em: https://www.redhat.com/sysadmin/podman-windows-wsl2

As instruções do Podman estão em: https://podman.io/getting-started/installation

Para realizar a instalação siga os passos descritos abaixo:

Precisa ter o WSL2 instalado
Se não tiver o WSL instalado, veja o processo descrito acima de como fazer a instalação.

Rode os comandos abaixo no Terminal do WSL:

Instale o podman

Para executar a instalação, execute os comandos abaixo dentro de um terminal do WSL do Linux:

```
. /etc/os-release
```

```
sudo sh -c "echo 'deb
http://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/x${NAME}_${VERSION
_ID}/ /' > /etc/apt/sources.list.d/devel:kubic:libcontainers:stable.list"

wget -nv
https://download.opensuse.org/repositories/devel:kubic:libcontainers:stable/x${NAME}_${VERSION_I
D}/Release.key -O Release.key

sudo apt-key add - < Release.key

sudo apt-get update -qq

sudo apt-get -qq -y install podman

sudo mkdir -p /etc/containers

echo -e "[registries.search]\nregistries = ['docker.io', 'quay.io']" | sudo tee
/etc/containers/registries.conf
```

Testar

Para testar a instalação, execute:

```
podman --version
```

Alias Docker para Podman

Todos os comandos do Docker ou do Podman são iguais, no entanto ao invés de rodar o comando assim:

```
docker ps -a
```

Você precisará utilizar o podman, assim:

```
podman ps -a
```

Para fazer com que o comando “docker” funcione, você pode incluir um alias para o comando “docker” executar o comando “podman”.

Edite o arquivo .bash_aliases:

```
nano ~/.bash_aliases
```

Adicione a linha:

```
alias docker=podman
```

Pressione CTRL+X para salvar, Y para confirmar e então ENTER.

Depois, execute:

```
source ~/.bash_aliases
```

Então, você poderá executar:

```
docker ps -a
```

Instalação do Podman Compose

Para utilizar o equivalente ao “docker compose” com o podman instale o “podman compose”.

O github do projeto é: https://github.com/containers/podman-compose

Para realizar a instalação siga os passos descritos abaixo.

Rode os comandos abaixo no Terminal do WSL:

Verifique se o Python está instalado:
Execute:

```
python3 --version
```

Se não tiver o PIP3 instalado, instale:
Execute:

```
pip3 --version

sudo apt update
sudo apt get pip3
```

Instale as seguintes dependências:
Execute:

```
sudo pip3 install python-dotenv
sudo pip3 install pyyaml
```

Instale o podman-compose:

Execute:

```
sudo pip3 install podman-compose
```

Vamos fazer um teste:
Execute:

```
podman-compose --help
```

Alias docker-compose para podman-compose
Para fazer com que o comando “docker-compose” funcione, você pode incluir um alias para o comando “docker-compose” executar o comando “podman-compose”.

Edite o arquivo .bash_aliases:

```
nano ~/.bash_aliases
```

Adicione a linha:

```
alias docker-compose=podman-compose
```

Pressione CTRL+X para salvar, Y para confirmar e então ENTER.

Depois, execute:

```
source ~/.bash_aliases
```

Então, você poderá executar:

```
docker compose –help
```

Não é muito importante agora se aprofundar no kubernetes, que é uma automação de containers, mas é importante depois dar uma olhada em como subir um Docker.

3.6 Postman

Instale o Postman: https://www.postman.com/downloads/

Pesquisem como utilizar ele, por exemplo para consultar uma API pública, por exemplo:

- free apis: https://github.com/public-apis/public-apis
- https://dukengn.github.io/Dog-facts-API/

3.7 Chrome debug tools

Importante saber como usar o Debug tools do Chrome, para por exemplo olhar o HTML ou o Javascript de uma página.

4 Conceitos web

Depois quero falar com vocês sobre conceitos genéricos como:

- como que um site é carregado no seu browser quando você digita a url no browser?
- conceitos (servidores, rede, TCP, DNS, requisição, códigos HTTP, etc.)
- API
- Json

5 Javascript

A linguagem que será utilizado no Node é o Javascript, então é importante ter um conhecimento bom de Javascript, abaixo alguns pontos importantes:

5.1 Entendendo Javascript

https://developer.mozilla.org/en-US/docs/Web/JavaScript

O Javascript é uma linguagem de programção com as seguintes características:

- Interpretada
- Tipagem dinâmica
- Baseada em objetos
- Funções de primeira classe

Pesquisar o que esses termos significam.

5.2 Uso básico

Pesquise como criar um arquivo Javascript e executar ele no browser, algo como:

Index.html:

```
<html>
<body>
    <script>
        (function(){
            "use strict";

            console.log('Hello world!');
        })();
    </script>
</body>
</html>
```

Como abrir o console do Google Chrome? Como visualizar essa mensagem?

Como separar isso em um arquivo test.js e chamar o JS no index.html ?

6 Básico da linhagem e estrutura de dados

Abaixo uma visão inicial, exercitem e explorem os conceitos abaixo. Tem exercícios no HackerRank também de aprendizado de Javascript.

Variáveis

Forma antiga de declaração:

```
var valor = 15;
var valor2 = 2.51;
var nome = 'Coutinho';
```

NOTA: Prefira utilizar as formas abaixo, “const” ou “let” para declaração. Todas variáveis declaradas utilizando “var” e que estejam fora de uma função ou bloco tem escopo global.

Constante:

```
const valor = 15;
const valor2 = 2.51;
const nome = 'Coutinho';
```

Variável escopo local:

```
let valor = 15;
let valor2 = 2.51;
let nome = 'Coutinho';
```

Tipos:

- Boolean
- Null
- Undefined
- Number
- BigInt
- String
- Symbol

Exemplos de uso:

```
const valor = 15; // Number
const valor2 = 2.51; // Number
const nome = "Coutinho"; // String
const usuario = {nome:"Eduardo", sobreNome:"Coutinho"}; // Object
const aceita = true; // Boolean
```

Verificar o tipo de um valor

```
typeof(15); //number
```

```
typeof(1.9); //number
typeof('abacaxi'); //string
typeof(true); //boolean
typeof([1, 2, 3]); //object
typeof({nome: 'Eduardo', sobreNome: 'Coutinho'}); //object
typeof(function add(a, b) { return a+b; }); //function
```

Conversão para número:

```
Number(9.2) = 9.2
Number(' 2 ') = 2
Number('2abc') = NaN
```

Tentar converter um valor não numérico para número, retorna NaN (Not a Number)
Exemplos:

```
Number('a') // NaN
```

NaN não é comparável

```
NaN === NaN //false
```

Então para verificar se algo é NaN utilize a função “isNaN”:

```
isNaN("a") //true
isNaN(18) //false
```

Funções

Para criar uma função utilize “function”:

Exemplos de funções:

1- Sem retorno e sem entrada de valor:

```
function hello() {
	console.log('Ola!');
}
hello();
```

2- Sem retorno e com entrada de valor:

```
function hello(name) {
	console.log('Ola ' + name);
}
hello('Coutinho');
```

3- Com retorno de valor:

```
function sum(x, y) {
	return x + y;
}
console.log(sum(2, 3));
```

Parâmetros Rest:

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Functions/rest_parameters

Permite receber um número indefinido de parâmetros em uma função.

```
function somar(...numeros) {
   let total = 0;
   for (const num of numeros)
      total += num;
   return total;
}
somar(2, 5, 10, 100);
```

É possível combinar com parâmetros padrões, por exemplo:

```
function atualizarTotal(texto, ...numeros) {
      let total = 0;
      for (const num of numeros)
      total += num;
   console.log(texto + total);
}
atualizarTotal('Total: ', 2, 5, 10, 100);
```

Arrow functions

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Functions/Arrow_functions

Como no tipo padrão de funções parâmetros e retorno são opcionais

1) Sintaxe padrão:

```
const add = (a, b) => {
      const result = a + b;
      return result;
};
console.log(add(2, 3));
```

isso é o equivalente a escrever:

```
const add2 = function(a, b) {
      return a + b;
}
console.log(add2(2, 3));
```

2) Sintaxe simplificada para 1 parâmetro: Se exatamente 1 parâmetro for recebido o parênteses ao redor do parâmetro pode ser omitido

```
const log = message => {
      console.log(message);
};
log();
```

3) Sintaxe para nenhum parâmetro: Se a função não receber nenhum parâmetro o parênteses não pode ser omitido

```
const greet = () => {
```

```
      console.log('Hi there!');
};
greet();
```

4) Se a função possuir somente uma expressão no body Se a função possuir somente uma expressão no body é posível omitir as chaves “{}” no corpo da função e o “return”

```
const add = (a, b) => a + b;
console.log(add(2, 3));
```

5) Retornar objeto Para retornar um objeto é necessário parênteses ao redor da declaração do objeto se não as chaves serão interpretadas como o corpo da função

```
const loadPerson = pName => ({name: pName });
```

Strings

Concatenação

```
console.log('Eduardo' + ' ' + 'Coutinho');
```

Template strings: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/template_strings

```
let valor = 2;
console.log(`O valor é ${valor}`);
```

Funções de string

```
string.split()
```

Coleções

Array

```
const valores = [ 7, 2, 3 ];
const frutas = [ 'abacaxi', 'banana', 'uva' ];
```

Funções de Arrays

Referência: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

.push()

Adiciona um item no final do array

```
let valores = [ 10, 20, 30 ]
valores.push(40)
console.log(valores)      // [10, 20, 30, 40]
```

```
valores.push(50, 60, 70)
console.log(valores)      // [10, 20, 30, 40, 50, 60, 70]
```

.pop()

Retorna e remove o último item do array

```
let valores = [ 10, 20, 30 ]
let val = valores.pop()
console.log(valores) //10, 20

console.log(val) //30
```

.shift()

Remove o primeiro item do array

```
let valores = [ 10, 20, 30 ]
let val = valores.shift()

console.log(valores) //20, 30

console.log(val) //10
```

.unshift()

Adiciona um ou mais itens no início do array

```
let valores = [ 10, 20, 30 ]

valores.unshift(5)
console.log(valores)      // [5, 10, 20, 30]

valores.unshift(2, 3)
console.log(valores)      // [2, 3, 5, 10, 20, 30]
```

.slice()

Retorna uma cópia do array com os itens indicados, iniciando no primeiro parâmetro e indo até o índice do segundo, mas sem incluí-lo

Sintaxe:

slice(start, end)

- start (Optional) = Zero-based index at which to start extraction.
- end (Optional) = Zero-based index before which to end extraction. slice extracts up to but not including end. For example, slice(1,4) extracts the second element through the fourth element (elements indexed 1, 2, and 3).

Exemplo:

```
let frutas = ['banana', 'manga', 'abacaxi', 'uva', 'laranja']
```

```
let ret = frutas.slice(1, 4) //Seleciona os índices: 1, 2 e 3 (4 não incluído)
console.log(ret); //['manga', 'abacaxi', 'uva']
```

.splice()

Altera o conteúdo de um array removendo ou substituindo elementos existentes e/ou adicionando novos itens

Sintaxe:

splice(start, deleteCount, item1, item2, itemN)

- start = The index at which to start changing the array.
- deleteCount (Optional) = An integer indicating the number of elements in the array to remove from start.
- item1, item2, ... (Optional) = The elements to add to the array, beginning from start.

Exemplos:

```
let frutas = []
let ret = '';
frutas = ['banana', 'manga', 'abacaxi', 'uva', 'laranja']
ret = frutas.splice(1, 0, 'figo')     //adiciona um item no índice 1 e remove 0 itens
console.log('frutas=', frutas); //['banana', 'figo', 'manga', 'abacaxi', 'uva', 'laranja']
console.log('ret=', ret); //[]

frutas = ['banana', 'manga', 'abacaxi', 'uva', 'laranja']
ret = frutas.splice(1, 2) //iniciando na posição 1, remove 2 itens
console.log('frutas=', frutas); //['banana', 'uva', 'laranja']
console.log('ret=', ret); //['manga', 'abacaxi']

frutas = ['banana', 'manga', 'abacaxi', 'uva', 'laranja']
ret = frutas.splice(1)    //remove todos os itens a partir índice 1
console.log('frutas=', frutas); //['banana']
console.log('ret=', ret); //['manga', 'abacaxi', 'uva', 'laranja']

frutas = ['banana', 'manga', 'abacaxi', 'uva', 'laranja']
ret = frutas.splice(-1, 1)      //remove o último item do array
console.log('frutas=', frutas); //['banana', 'manga', 'abacaxi', 'uva']
console.log('ret=', ret); //['laranja']

frutas = ['banana', 'manga', 'abacaxi', 'uva', 'laranja']
ret = frutas.splice(0, 2)       //seleciona índices de 0 até 2, sem incluir o 2
console.log('frutas=', frutas); //['banana', 'manga', 'abacaxi', 'uva', 'laranja']
console.log('ret=', ret); //['banana', 'manga']

frutas = ['banana', 'manga', 'abacaxi', 'uva', 'laranja']
ret = frutas.splice(-3, -1) //['banana', 'manga', 'abacaxi', 'uva', 'laranja']
console.log('frutas=', frutas); //['banana', 'manga', 'abacaxi', 'uva', 'laranja']
console.log('ret=', ret); //['abacaxi', 'uva']
.indexOf()
```

Obtém o índice de um item no array, ou se não encontrar o item retorna -1

```
let valores = [ 10, 20, 30 ]
```

```
console.log(valores.indexOf(10)) //0
console.log(valores.indexOf(20)) //1
console.log(valores.indexOf(30)) //2
console.log(valores.indexOf(70)) //-1
```

.includes()

Verifica se um item está presente no array. É o mesmo que array.indexOf() !== -1

```
let valores = [ 5, 12, 8, 130, 44 ]
let ret = valores.includes(8)
console.log(ret) //true
```

.find()

Retorna o valor do primeiro item que atende a condição definida

```
let valores = [ 5, 12, 8, 130, 44 ]
let ret = valores.find(element => element > 10)
console.log(ret) //12
.findIndex()
```

Retorna o índice do primeiro item que atende a condição definida

```
let valores = [ 5, 12, 8, 130, 44 ]
let ret = valores.findIndex(element => element > 10)
console.log(ret) //1
```

.filter()

Retorna uma cópia do array, com todos os itens que atendem a condição definida

```
let valores = [ 5, 12, 8, 130, 44 ]
let ret = valores.filter(element => element > 10)
console.log(ret) //12, 130, 44
```

.forEach()

Executa uma função para cada item de um array

```
const valores = [ 5, 12, 8, 130, 44 ]
valores.forEach((element) => { console.log(element); })
```

.sort()

Ordena os itens de um array. Para realizar a ordenação para elemento do array é convertido para String, isso gera alguns diferenças na ordenação de uma sequencia de números.

```
const frutas = ['banana', 'manga', 'abacaxi', 'uva', 'laranja']
frutas.sort()
console.log(frutas) //['abacaxi', 'banana', 'laranja', 'manga', 'uva']

const numeros = [1, 30, 4, 21, 100000];
```

```
numeros.sort()
console.log(numeros) //[1, 100000, 21, 30, 4]
```

.reverse()

Coloca os itens do array na ordem inversa, o primeiro item vira o último

```
const frutas = ['banana', 'manga', 'abacaxi', 'uva', 'laranja']
frutas.reverse()
console.log(frutas) //['laranja', 'uva', 'abacaxi', 'manga', 'banana']

const numeros = [1, 30, 4, 21, 100000];
numeros.reverse()
console.log(numeros) //[100000, 21, 4, 30, 1]
```

DICA: .reverse() pode ser utilizado para escrever uma palavra ao contrário, para isso basta:

1- Usar split() para transformar a palavra em array com cada letra em uma posição:

```
'calendario'.split('') //['c', 'a', 'l', 'e', 'n', 'd', 'a', 'r', 'i', 'o']
```

2- Usar o método reverse() para inverter o array

```
'calendario'.split('').reverse() //['o', 'i', 'r', 'a', 'd', 'n', 'e', 'l', 'a', 'c']
```

3- E então juntar as letras novamente em uma string:

```
'calendario'.split('').reverse().join('') //oiradnelac
```

.reduce()

array.join()

.map()
method creates a new array with the results of calling a provided function on every element in the calling array.

```
const array1 = [1, 4, 9, 16];
const map1 = array1.map(x => x * 2);
console.log(map1); // expected output: Array [2, 8, 18, 32]
```

Spread operator

Utilizado para:

1- criar um novo array, com os valores do array original, ao invés de copiar a referência do array

```
const array2 = [...array1];
```

2- para converter um array em valores isolados:

Math.min(2, -1, 3);
const prices = [2, -1, 3];

Math.min(...prices);

Array Destructuring

Referência: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment

```
const valores = [10, 20, 30, 40];
const [a, b] = valores;
console.log(a); //10
console.log(b); //20

const nameArray = [ 'Eduardo', 'Coutinho', 'mail@gmail.com', '11 96666-6666'];
const [ firstName, lastName, ...outrasInformacoes] = nameArray;
console.log(firstName) // Eduardo
console.log(outrasInformacoes); //["mail@gmail.com", "11 96666-6666"]
```

Set

Permite o armazenamento de valores únicos. A interação sobre os itens é realizado na ordem de inserção.

Referência: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Set

```
const valores = new Set();
valores.add(5); // valores {5}
valores.add(1); // valores {5, 1}
valores.add(1); // valores {5, 1} não é alterado porque o valor 1 já estava no conjunto
```

Funções de set

```
// declaração
const valores = new Set();

//add() -- adiciona um item
valores.add(5); // valores {1}
valores.add(1); // valores {5, 1}

//size -- retorna a quantidade de itens armazenados
valores.size;      //2

//has() -- verifica se um item estão no conjunto
valores.has(1); // true
valores.has(3); // false, 3 não está no conjunto

//delete() -- remove um item do conjunto
valores.delete(1) // valores {5}

//interar sobre os itens
const valores = new Set();
valores.add(5);
valores.add(1);
valores.add('laranja');
for (let item of valores) { console.log(item); }
// 5
// 1
```

```
// laranja

//converter array em Set
let frutas = ['abacaxi', 'banana', 'uva', 'uva'];
const valores = new Set(frutas);
console.log(valores); // Set(3) {"abacaxi", "banana", "uva"}
```

Map

Objeto simples de chave/valor

const personData = new Map();

WeakSet

WeakMap