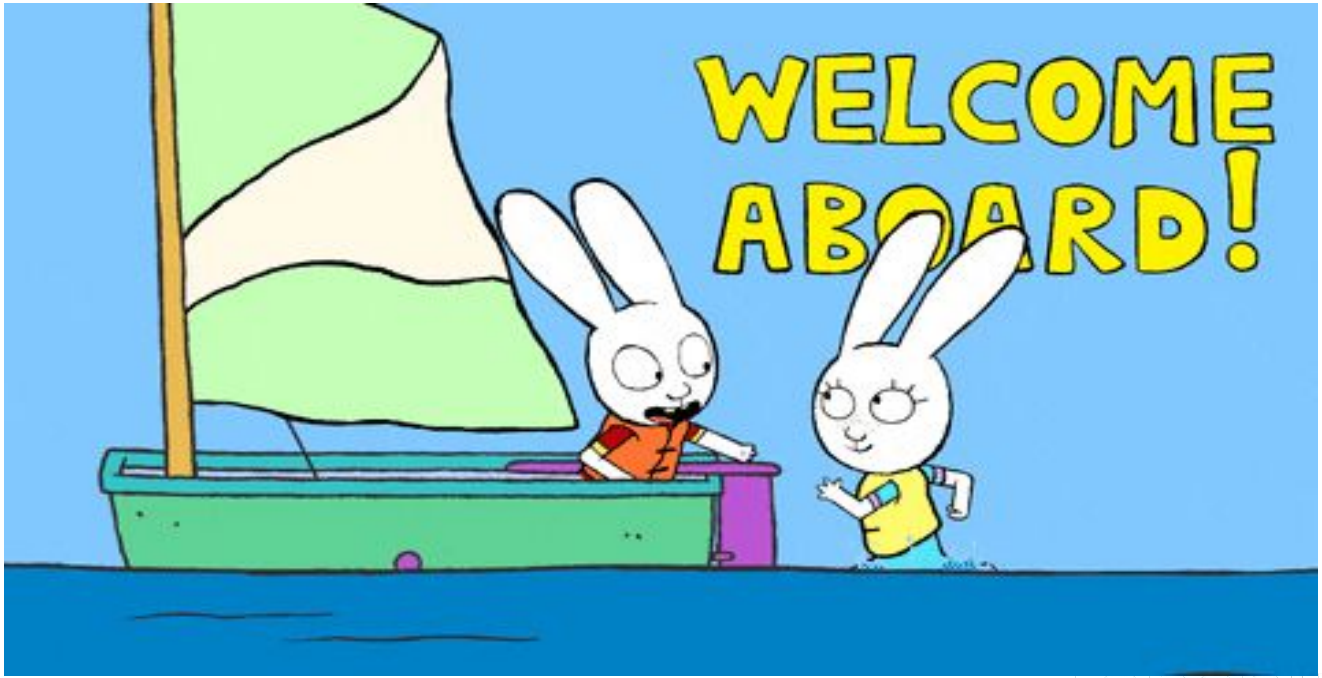
Bem vindes ao Backend!



Node e Package.json

Labenu_

Relembrando...

Labenu_

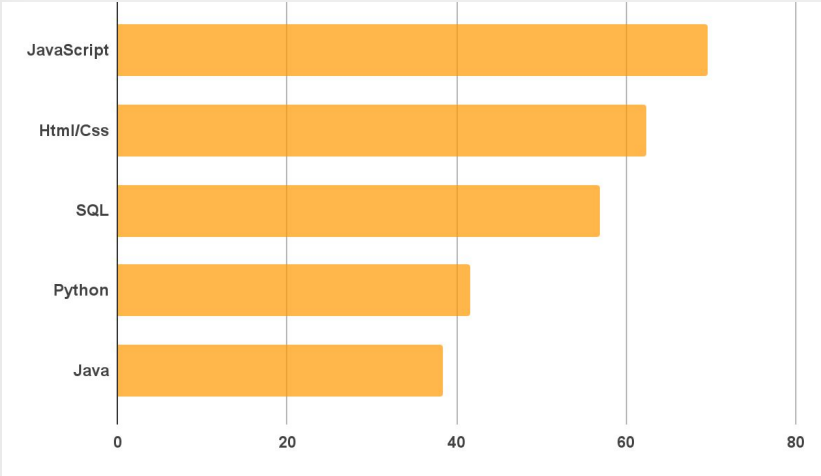
JSON e JS

JSON

```json
{
  "nome": "Mafalda",
  "idade": 20,
  "localizacao": {
    "pais": "Argentina",
    "cidade": null
  },
  "emprego": undefined,
  "amigos": ["Bob", "Carol"],
}
```

`JSON.parse(meuJson)`

Google LinkedIn



O que vamos ver hoje?

- NodeJs (NPM / Package.json)
- Backend com Node.js
- Criando Script Personalizados

Introdução 🧐

- No front-end o JS é executado pelo navegador
- Para instalar o React, utilizamos o NPM
- O que realmente acontece quando executamos o comando npm run start?

Node.js

Labenu_

Node.js 🎀

- Tornou possível o uso do JS do lado do servidor (backend)
- Utiliza a engine V8 criada pelo Google para interpretar o código JS
- Rápido crescimento e adoção pela comunidade dada sua performance e facilidade de uso
- Utiliza o NPM para gerenciar os pacotes (dependências)

NPM 📦

- Node Package Manager
- Já vem junto com o Node.js
- Gerencia as dependências do projeto
- Permite a criação e execução de instruções ou conjuntos de instruções

package.json 📦

- Fica **na pasta raiz** do projeto e utiliza o NPM como administrador de dependências

- Guarda informações e configurações, tais como: o nome do projeto, quem criou, as dependências e os scripts

- Os scripts podem ser executados com o comando: **npm run nomeDoScript**

criando o package.json 🎞️

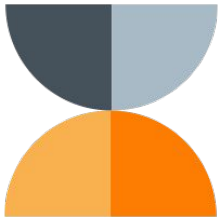
- É a referência para os comandos do NPM:
    - `npm init`: cria um package.json (aparecerá perguntas que deverão ser preenchidas)
    - `npm init -y`: cria um package.json (com valores padrão)

Vamos ver na prática! 🔬

dependências - package.json 🎞️

- **Dependencies**
  - Dependências para que o projeto **seja executado**

- **devDependencies**
  - Dependências para que o projeto **seja desenvolvido**



Pausa para relaxar 😴

- **NPM** é uma ferramenta que facilita o acesso a bibliotecas de código JS
- Criamos o package.json com o comando ***npm init -y***
- O comando ***npm install*** instala dependências, que podem ser *dependencies* ou *devDependencies*

Backend com Node.js

Labenu_

Backend com Node.js 🎀

- Aqui no backend nós **não estamos mais no navegador**
- **Não temos acesso** aos métodos nativos do navegador, tais como: *alert*, *confirm*, *prompt* e *localStorage*
- É tudo dentro do **terminal**

Backend com Node.js 🎀

- Por enquanto, os dados serão salvos na memória dos processos enquanto a aplicação estiver rodando

- Bem parecido com o estado no React

- Podemos enviar dados para o programa utilizando o **process.argv**

O process.argv

- O **process.argv** é nativo do Node e consiste em um array de strings recebidas nos argumentos do comando. Os dois primeiros argumentos são fixos:

```
node nome_do_arquivo.extensão
```

  - **process.argv[0]**: o primeiro argumento é o próprio node
  - **process.argv[1]**: o segundo argumento é o arquivo que vamos executar

O process.argv

- A partir do **process.argv[2]**, nós podemos atribuir valores

```
node nome_do_arquivo.extensão valor
```

- Podemos passar informações, ao rodar os comandos utilizando o argv

Vamos ver na prática! 🔬

Criando scripts personalizados

Labenu_

Scripts personalizados

- Durante o front-end, nós utilizamos o comando **npm run start** para executar os nossos projetos;

- Isso acontece pois o react cria **scripts** no package.json para que certas sequências de comandos sejam executadas;

- Podemos fazer o mesmo com nossos projetos, alterando a área de **scripts** do arquivo package.json

Scripts personalizados

- Por padrão, esta área vem preenchida desta forma:

```
{
 "name": "exemplo",
 "version": "1.0.0",
 "description": "",
 "main": "index.js",
 "scripts": {
   "test": "echo \"Error: no test specified\" && exit 1"
 },
 "keywords": [],
 "author": "",
 "license": "ISC"
}
```

Scripts personalizados

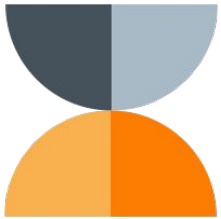
- E podemos adicionar múltiplos scripts, com diferentes comandos em cada um

```
{
 "name": "exemplo",
 "version": "1.0.0",
 "description": "",
 "main": "index.js",
 "scripts": {
   "start": "clear && node ./index.js",
   "serve": "clear && echo “Hello” && node ./index.js"
 },
 "keywords": [],
 "author": "",
 "license": "ISC"
}
```

Vamos ver na prática! 

Exercício 1

Crie um arquivo **somaNumeros.js** e um **package.json**. Crie um script personalizado, chamado **soma** . Passe os números 5 e 10 como argumentos e imprima a somatória desses dois números no terminal.



Pausa para relaxar 😴

Para os nossos testes de hoje, usaremos:

- **Entradas:** process.argv[2] (3, 4, 5...)
- **Saídas:** console.log / console.table
- **Rodar o projeto:** node nomeDoArq.js

Praticando

Labenu_

Coding Together 👫

Crie um pacote do Node contendo um script chamado ***pesquisar-usuarios***. Ele deve receber uma string pelo terminal e imprimir uma tabela com as pessoas cujo nome contém a *string* informada.

Resumo

Labenu_

Resumo 📓

- Node.js é um **interpretador** de Javascript
- Seu foco foi simplificar a manipulação de arquivos
- O Node.js não possui tudo que o navegador injeta para gente quando usamos JS em sites, porém possui recursos a mais

Resumo 📓

- **npm** nos ajuda muito a cuidar do gerenciamento das **bibliotecas externas**

- Para iniciar novos projetos de Node.js, usamos o comando **npm init -y**

- Acessamos argumentos passados aos nossos comandos e scripts através de **process.argv**

Dúvidas? 🧐

Labenu_

Obrigado(a)!