Node.JS

Francisco Costa

Francisco Costa

- *Freelancer Software Engineer & External Consultant*
- Formador de linguagens de programação
- Adepto de boas práticas e metodologias de desenvolvimento de software

Francisco Costa

- jfcpcosta@gmail.com

- http://franciscocosta.net
- http://twitter.com/jfcpcosta
- http://linkedin.com/in/jfcpcosta
- https://github.com/jfcpcosta


Y
O
U

Programa

- Introdução ao **Node.JS**
- ES6
- Modulos
- Web e **HTTP**
- Bases de Dados: **NoSQL** e **Relacionais**
- WebSockets

Introdução ao Node.JS

Node.JS

É um **interpretador** de Javascript **assíncrono**

Open Source

Com foco em migrar o desenvolvimento em **Javascript** para o **servidor**

Node.JS

Criado por **Ryan Dahl** em 2009

Um ambiente de execução baseado no interpretador **V8 Javascript Engine**

Escrito em C++, Open Source e mantido pela **Google** e utilizado no **Google Chrome**

Node.JS

Orientado a **eventos**

A sua execução é **single-thread**

A única thread dísponivel é chamada de **Event Loop** e é responsável por executar o código javascript

Node.JS

O Node.JS tem um **gestor de dependências** integrado

NPM (Node Package Manager)

Podemos instalar bibliotecas muito facilmente

Node.JS

express

socket.io

mongoose

...

Node.JS

https://www.npmjs.com/

Node.JS

Extremamente **leve**

Podemos usar juntamente com o **Docker**

Bastante utilizado para criação de **micro-serviços** e **serverless**

Node.JS

Google Cloud

Node.JS

Adicionar funcionalidades facilmente através do **NPM**

Mesma linguagem no **backend** e no **frontend**

Rápido **deploy** da aplicação

Interpretador muito **eficiente** e com **pouco consumo** de recursos

Ambiente de Desenvolvimento

Requisitos

- Confortáveis com **JavaScript**
  - Funções **ES5** (*map*, *sort*, *filter*…)
  - Alguns conhecimentos **ES6**

Preparação do Ambiente de Desenvolvimento

- Instalar *NodeJS*
- Instalar o **Visual Studio Code** (o outro editor à escolha)

Instalação do Node.JS

Disponível em **duas** versões

LTS - Long Term Support

Versão mais atual com as últimas funcionalidades disponíveis

Instalação do Node.JS

LTS – 16.14.0

Current – 17.7.1

em 14 de Março de 2022

Instalação do Node.JS

Podemos instalar descarregando o **binário específico** para cada sistema operativo.

Podemos instalar usando um **gestor de pacotes** para o sistema operativo que estamos a usar (*homebrew*, *aptitude*, *chocolatey*…)

Instalação do Node.JS

https://nodejs.org/

Instalação do Node.JS

https://nodejs.org/en/download/package-manager/

Instalação do Node.JS

Podemos utilizar o **NVM** – Node Version Manager caso necessitemos de utilizar várias versões do Node.JS na mesma máquina

Instalação do Node.JS

Linux / MacOS:

https://github.com/nvm-sh/nvm

Windows:

https://github.com/coreybutler/nvm-windows

Hello, world

Hello, world

Tendo o Node.JS instalado na máquina e disponível no **PATH** passamos a ter dois **executáveis disponíveis**:

node e **npm**

node

Permite-nos executar um ficheiro Javascript no interpretador.

node

```
$ node src/app.js
```

node

```
console.log('Hello, world!');
```

npm

Permite-nos instalar dependências e executar tarefas no nosso projeto

npm

```
$ npm install moment
$ npm start
...
```

ES6

ECMAScript

- Também designado por **ES**
- Um standard para linguagens de programação de scripting *client-side*
- Primeira edição foi lançada em 1997
- **ES5** (*standard*) lançado em 2009 depois de um abandono do *standard*
- Sexta edição finalizada em 2015
  - **ES6** ou **ES2015**
- ES7 em 2016, ES8 em 2017…

ES6

- Scope
- `let`
- `const`
- Template literal
- Multiline string
- Optional parameters
- Arrow functions
- Rest operator
- Spread
- Generator functions
- Destructuring
- Classes

DEMO

ES6

Modúlos

Módulos

Um **módulo** em Node.JS é uma **funcionalidade** (simples ou complexa) organizada em **um** ou **múltiplos** ficheiros javascript, que podem ser **reutilizados** ao longo da aplicação.

Módulos

Cada módulo tem o seu **próprio contexto** e não interfere com outros módulos na mesma aplicação.

Módulos diferentes podem ser colocados em **ficheiros** javascript diferentes.

Módulos

Por defeito o Node.JS implementa **CommonJS standard modules**.

CommonJS define um **standard de modularização** para aplicações **CLI** ou **serverside**.

Módulos

Temos três tipos de módulos:

Core Modules,

Local Modules,

Third Party Modules

Core Modules

Os Core Modules, são as bibliotecas base que são incluídas com o Node.JS.

São carregados automaticamente com o a instalação do Node.JS.

No entanto para a sua utilização temos de os importar na mesma na nossa aplicação.

Core Modules

http

path

fs

...

Core Modules

```
const fs = require('fs');

fs.readFile('data.txt', (err, data) => console.log(data));
```

Local Modules

Os Local Modules, são os módulos criados por nós que representam a nossa aplicação.

Para os usarmos também necessitamos de os importar no contexto em que necessitamos deles.

Local Modules

```
const log = {
    info: (info) => {
      console.log('Info: ' + info);
    },
    warning: (warning) => {
      console.log('Warning: ' + warning);
    },
    error: (error) => {
      console.log('Error: ' + error);
    }
};

module.exports = log
```

Local Modules

```
const logger = require('./log');

logger.info('Hello, world');
```

Third Party Modules

Os Third Party Modules, são os módulos que instalamos através do NPM.

Mais uma vez, para os usarmos, precisamos de os importar através do require.

Third Party Modules

Para utilizarmos estes módulos precisamos de **inicializar** o NPM para o nosso projeto.

Isso irá criar um ficheiro **package.json**, onde será parametrizado todo o nosso projeto.

Third Party Modules

```
$ npm init
```

Third Party Modules

```
$ npm install moment
```

Third Party Modules

As dependências por defeito ficam numa pasta **node_modules**, que não deve ser adicionada ao controlo de versões.

Para voltarmos a criar esta pasta basta executar o comando:

```
$ npm install
```

Third Party Modules

```
const moment = require('moment');

const now = new Date();

const dateString = moment(now).format('Y-M-D H:mm:ss');

console.log(dateString);
```

ES6 Import, export

A partir da versão 12 do node passamos a poder utilizar os imports e exports que foram introduzidos no ES6 (ESM).

Para isso podemos passar a usar a extensão .mjs nos nossos ficheiros ou adicionar ao ficheiro package.json:

```
“type”: “module”
```

ES6 Import, export

Precisamos também de executar a nossa aplicação com a flag:

`--experimental-modules`

```
$ node --experimental-modules src/app.js
```

ES6 Import, export

```
export default function sum(a, b) {
  return a + b;
}
```

ES6 Import, export

```
import sum from './sum';


const res = sum(4, 6);

console.log(res);
```

ES6 Import, export

Este sistema passa a ser válido para Core Modules, Local Modules e Third Party Modules.

DEMO

Módulos: Core, Local e Third Party