

# **Aplicações para a Internet II**

## **Node.JS**

2021/2022

# node.js

---

**Objetivo do Node:**  
**disponibilizar uma forma**  
**fácil de criar programas de**  
**rede escaláveis**

Criado por Ryan Dahl em 2009

# Desenvolvimento para a Web ...

---

Na Web, as interfaces de utilizador são standard

- HTML para markup
- CSS para style
- JavaScript para conteúdo dinâmico

Tradicionalmente

- Os dados são gerados no servidor, transferidos para o cliente e mostrados pelo browser
- As tecnologias server Side incluem linguagens como PHP, ASP.net, Rails, entre outras ... mas também node.js

# O que é node.js?

---

- Node.js é um **runtime** para construir aplicações web escaláveis de alta performance usando JavaScript ou um interpretador de JavaScript assíncrono **orientado a eventos**, desenvolvido para lidar com aplicações de rede escaláveis
- Arquitetura **single thread** com um sistema **não bloqueante**. Permite lidar com milhares de ligações concorrentes ou um número infinito de requisições por segundo com pouco recurso computacional
- Não é uma plataforma Web ... e não é uma linguagem

“Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine” - <https://nodejs.org>

# Porquê node.js?

---

- Non blocking I/O
- Assíncrono e orientado a eventos
- Google V8 javascript engine
- Single Thread with Event Loop
- Milhares de módulos
- Multiplataforma: Windows, Linux, Mac
- Uma única linguagem para Frontend e Backend
- Comunidade ativa
- Altamente escalável
- Executa na linha de comandos

# Porquê javascript?

---

- Callback's amigáveis
- Ubíquo
- Sem primitivas IO
- JavaScript é usado no lado do cliente
- node.js é JavaScript no lado do servidor
  - Usado no lado do cliente e do servidor

# Javascript engines ...

---

Todos os browsers têm a sua VM

- Firefox? *Spidermonkey*
- Internet Explorer? *Chakra*
- Chrome? *V8*
- Safari? *JavascriptCore*
- Opera? *Carakan*
- ...

# V8 Javascript engine ...

---

- O Chrome's V8 JavaScript engine é um sistema que interpreta, mas também compila e executa o código JavaScript.
- Faz a gestão da alocação de memória de forma muito eficiente
- Aumenta significativamente a velocidade com que programas desenvolvidos em JavaScript executam
- Por ser um sistema independente do browser, pode ser usado como ferramenta por várias plataformas, como o Google Chrome e o Node.js. O Node, já vem com uma versão do V8.



# node.js ...

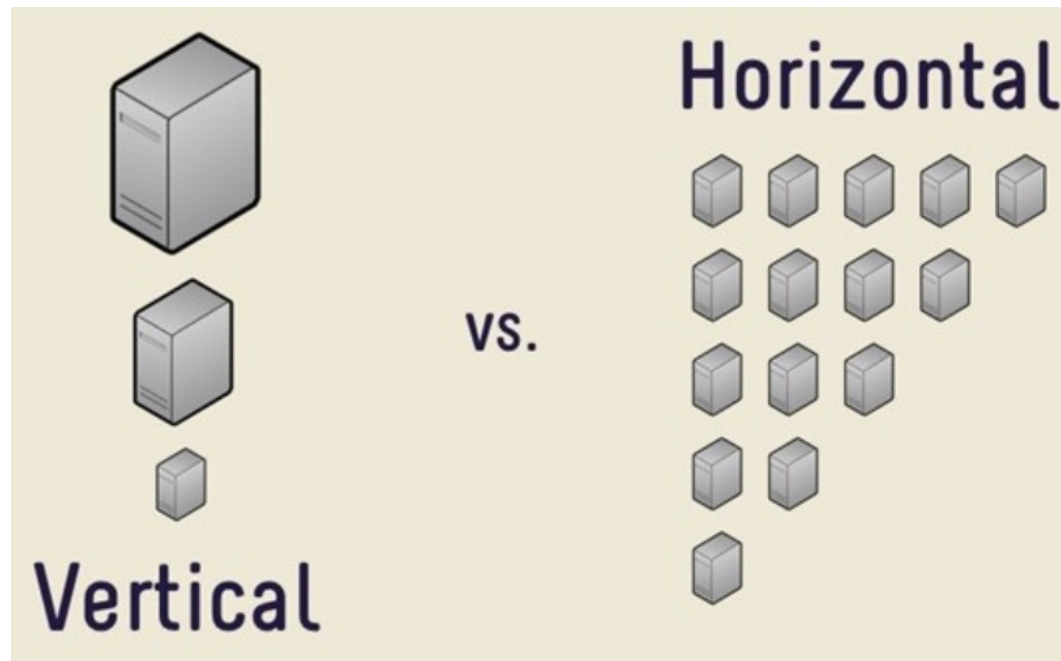
---

- Processamento assíncrono numa **única thread** (ao contrário de multithread clássico), o que minimiza a sobrecarga e a latência e maximiza a escalabilidade
- Ideal para aplicações que atendem muitos pedidos, mas não precisam de muita capacidade computacional por pedido
- Não é ideal para aplicações que usam a CPU intensivamente (CPU bound)
- Menos problemas com concorrência
- Servidores são normalmente baseados em threads. Node.js é baseado em eventos

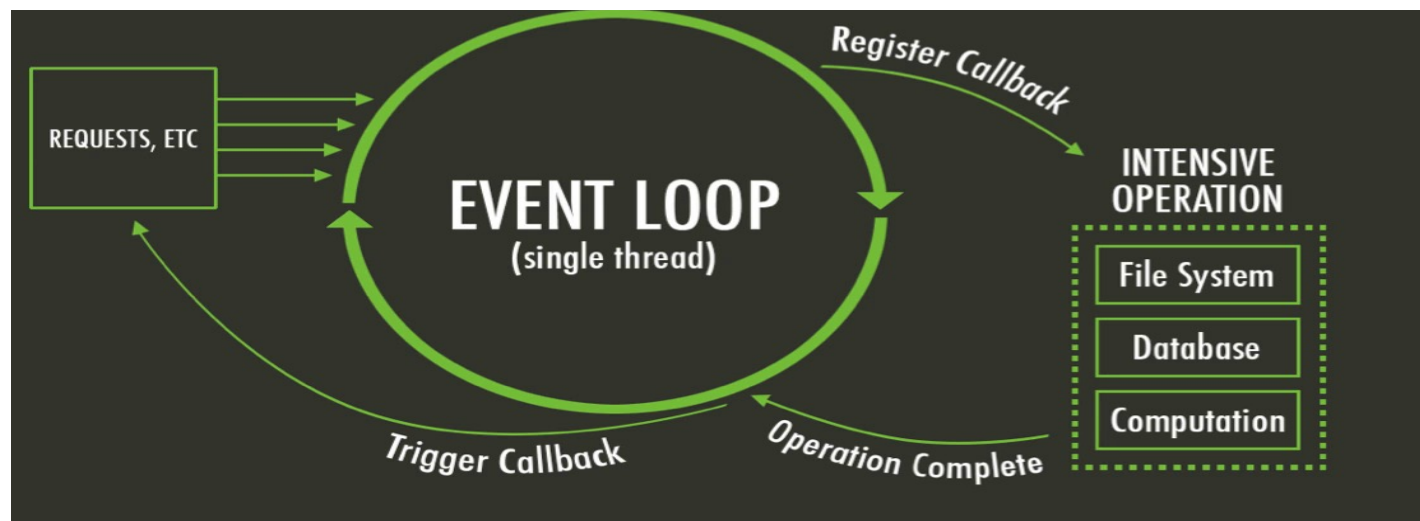
# node.js ... escalabilidade

---

- Node.js foi pensado para ser altamente escalável horizontalmente. Ao invés de investir em discos SSD, memória, pode-se virtualizar a máquina e/ou adicionar mais máquinas no mesmo ponto físico ou não, tornando assim sua escalabilidade muito mais simples, barata e eficaz.
- Na escala horizontal multiplicam-se os recursos



# node.js event loop ...

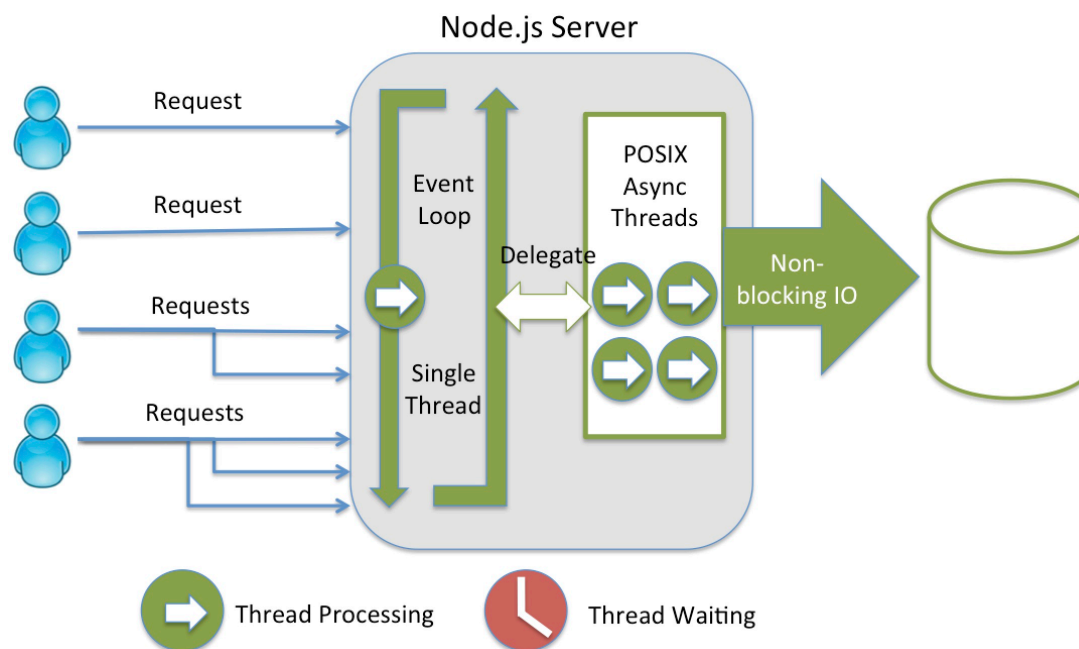


Existem algumas implicações neste modelo aparentemente simples e básico

- Evitar código síncrono, pois bloqueia o loop de eventos
- O que significa: callbacks, callbacks, ...

# node.js event loop ...

- O *Event Loop* fica à escuta de eventos que terminam o processamento. Quando um evento é concluído, ele dispara uma função de retorno (callback), que permite a execução de outras tarefas associadas ou dependentes dele



Fonte: <https://gist.github.com/ilyaigpetrov/f6df3e6f825ae1b5c7e2>




# Onde usar ...

---

- Chat/Messaging
- Real-time Applications
- I/O bound Applications
- Data Streaming Applications
- JSON APIs based Applications
- Single Page Applications
- Intelligent Proxies
- High Concurrency Applications
- ...

# Vamos começar ...

- No website <https://nodejs.org/en/> fazer o download para o sistema onde se pretende instalar
- Funciona de forma idêntica em Windows ou Linux
- Windows: <https://nodejs.org/dist/v12.14.1/node-v12.14.1-x64.msi>
- Linux: <https://nodejs.org/dist/v12.14.1/node-v12.14.1-linux-x64.tar.xz>

LTS Recommended For Most Users	Current Latest Features	
 Windows Installer <small>node-v12.14.1-x64.msi</small>	 macOS Installer <small>node-v12.14.1.pkg</small>	 Source Code <small>node-v12.14.1.tar.gz</small>

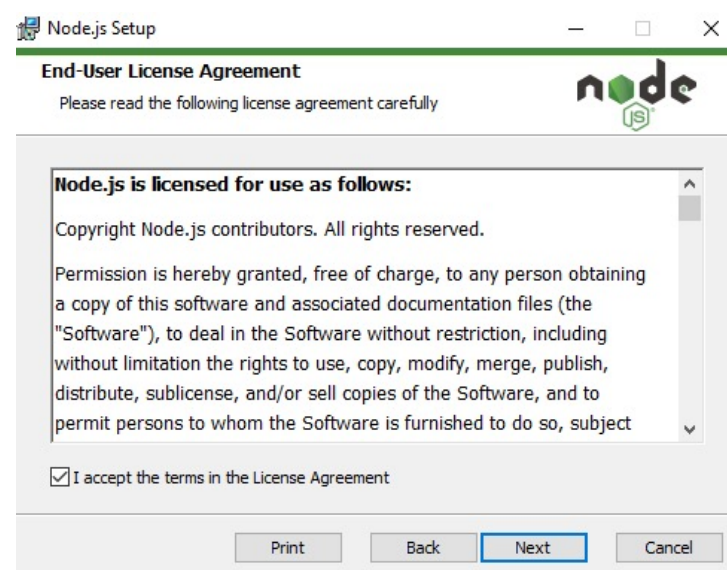
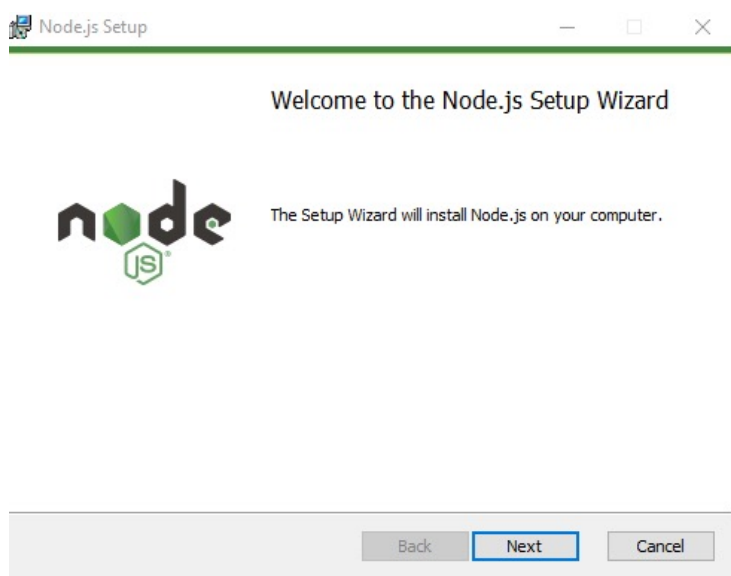
  

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit	
macOS Binary (.tar.gz)	64-bit	
Linux Binaries (x64)	64-bit	
Linux Binaries (ARM)	ARMv7	ARMv8
Source Code	node-v12.14.1.tar.gz	

# Instalar em Windows ...

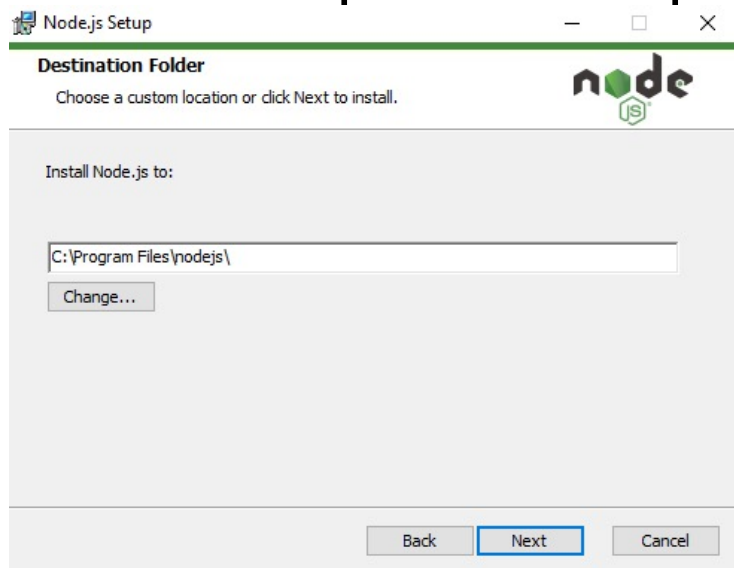
---

- Executar o ficheiro node-v12.14.1-x64.msi

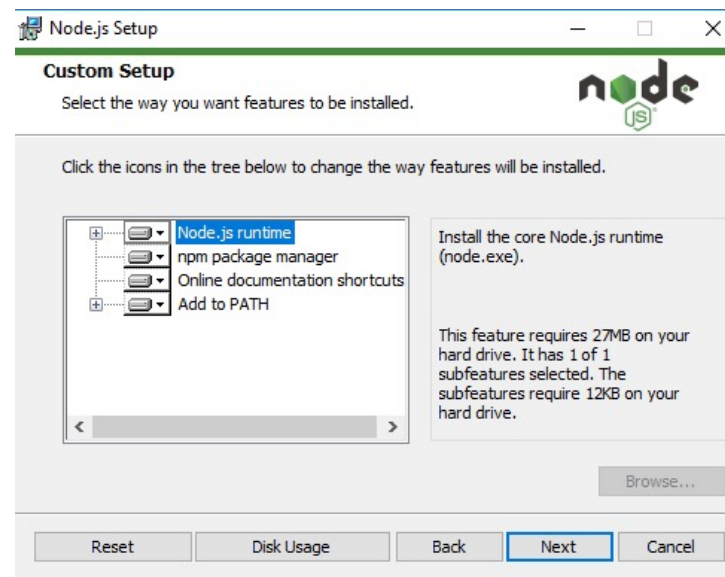


# Instalar em Windows ...

- Escolher a pasta onde pretende instalar



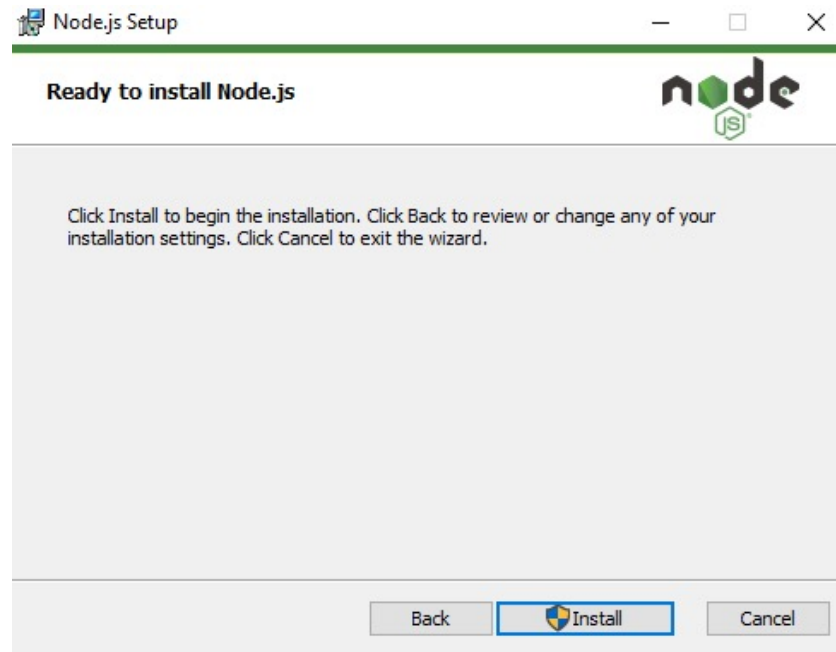
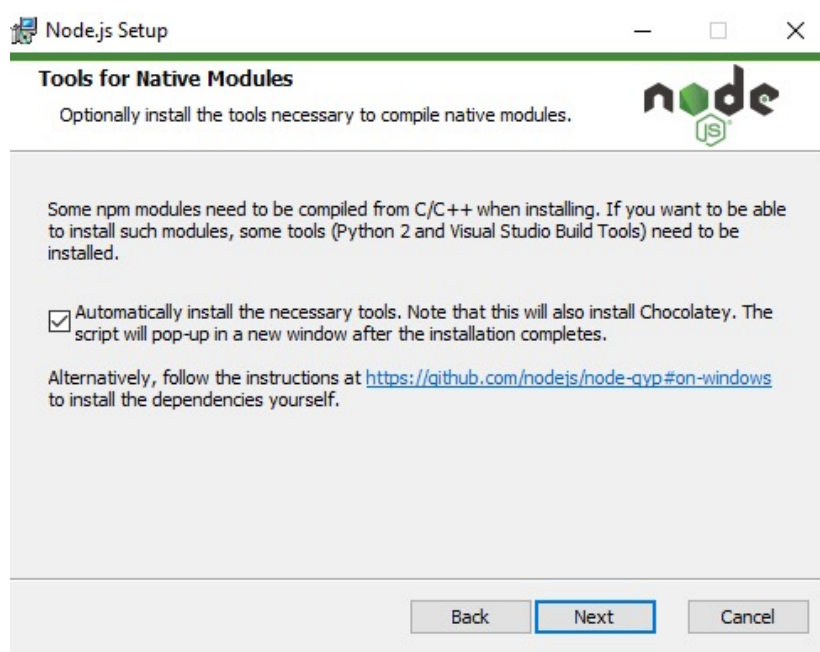
- Deixar todas as opções ativadas





# Instalar em Windows ...

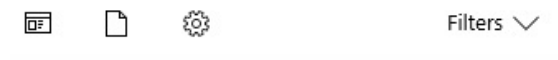
- Escolher se quer instalar outras ferramentas e clicar no botão install



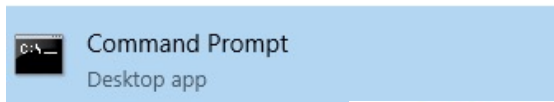
# Instalar em Windows ...

---

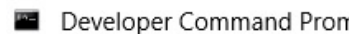
- Execute o *Command prompt* e escreva o comando `node -v`
- A mensagem indica o correto funcionamento do node.js com a respetiva versão



Best match



Apps



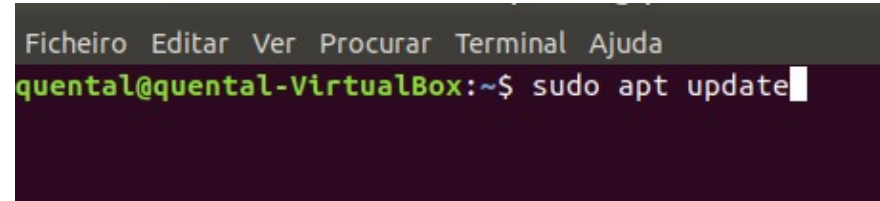
```
Your environment has been set up for using Node.js 12.14.1 (x64) and npm.  
C:\Users\quental>node -v  
v12.14.1
```

# Instalar em Linux ...

---

- Atualizar a cache APT

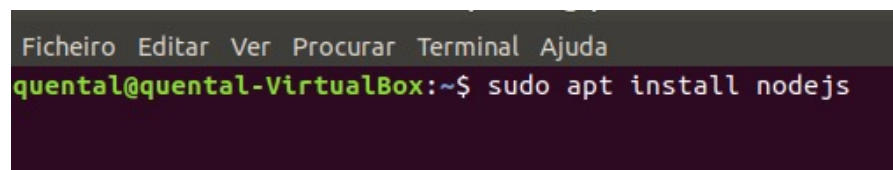
- `sudo apt update`



```
Ficheiro Editar Ver Procurar Terminal Ajuda
quental@quental-VirtualBox:~$ sudo apt update
```

- Instalar o Node.js

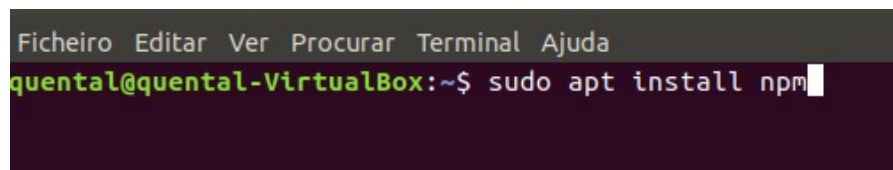
- `sudo apt install nodejs`



```
Ficheiro Editar Ver Procurar Terminal Ajuda
quental@quental-VirtualBox:~$ sudo apt install nodejs
```

- Instalar o pacote NPM para gerir o Node.js.

- `sudo apt install npm`

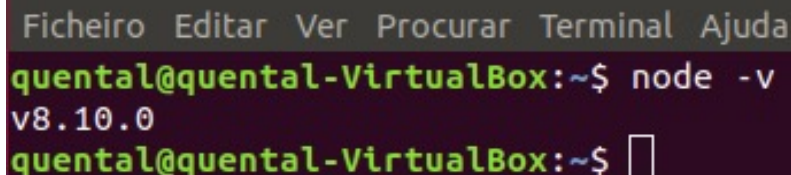


```
Ficheiro Editar Ver Procurar Terminal Ajuda
quental@quental-VirtualBox:~$ sudo apt install npm
```

# Instalar em Linux ...

- Verificar ...

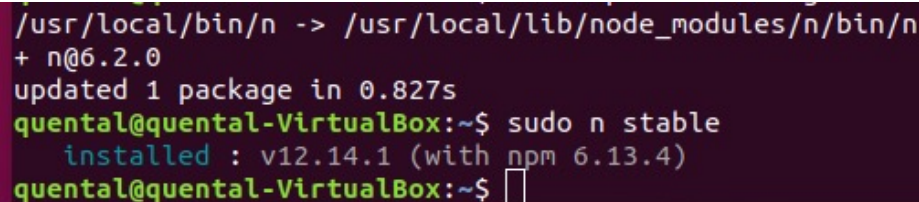
- node -v



```
Ficheiro Editar Ver Procurar Terminal Ajuda
quental@quental-VirtualBox:~$ node -v
v8.10.0
quental@quental-VirtualBox:~$
```

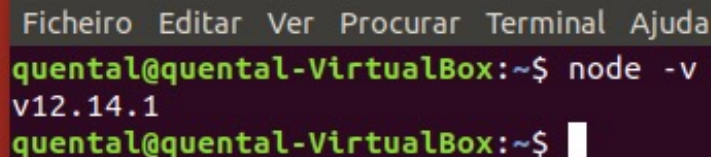
- Como a versão apresentada é a 8, vamos fazer o upgrade

- sudo npm cache clean -f
- sudo npm install -g n
- sudo n stable



```
/usr/local/bin/n -> /usr/local/lib/node_modules/n/bin/n
+ n@6.2.0
updated 1 package in 0.827s
quental@quental-VirtualBox:~$ sudo n stable
  installed : v12.14.1 (with npm 6.13.4)
quental@quental-VirtualBox:~$
```

- ... e verificamos novamente a versão (passou para 12.14.1)

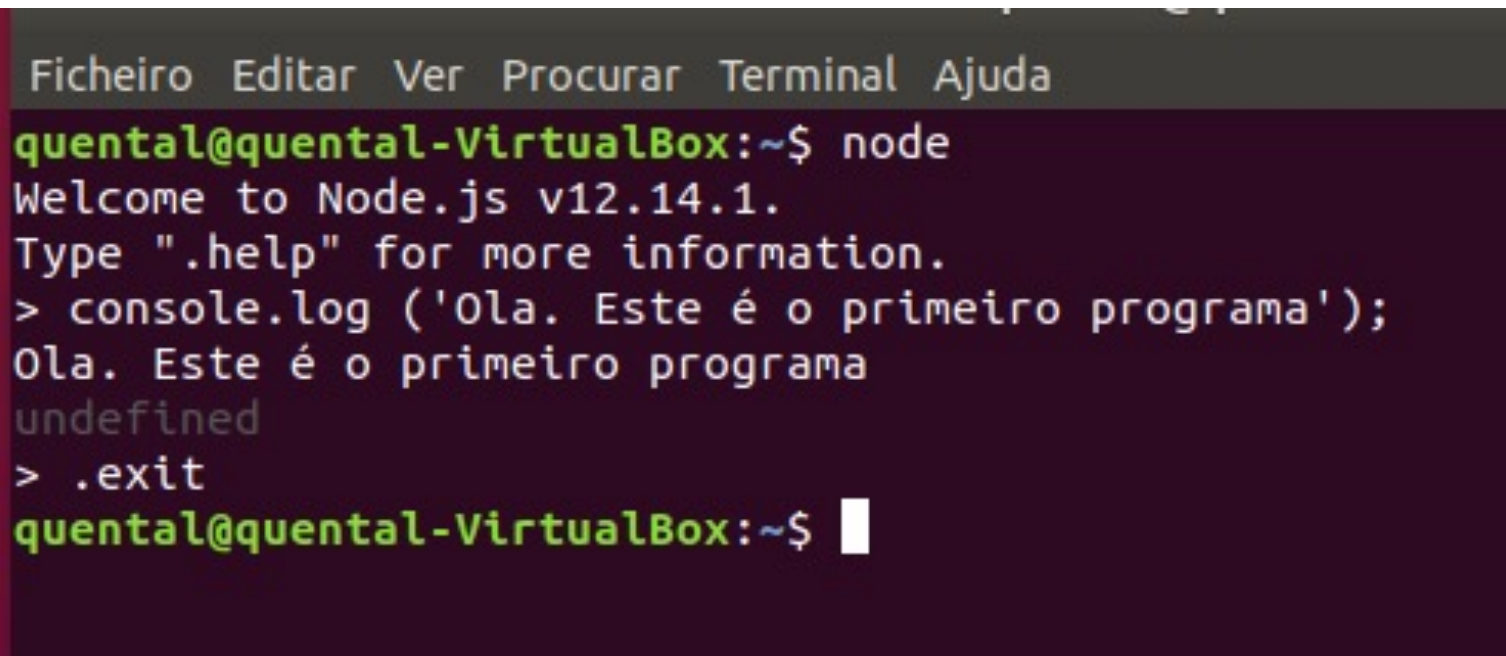


```
Ficheiro Editar Ver Procurar Terminal Ajuda
quental@quental-VirtualBox:~$ node -v
v12.14.1
quental@quental-VirtualBox:~$
```

# Primeiro programa ...

---

- Em Ubuntu
  - Executar node
  - Console.log ('Meu primeiro programa');


A screenshot of a terminal window with a dark background. At the top, there is a menu bar with the options 'Ficheiro', 'Editar', 'Ver', 'Procurar', 'Terminal', and 'Ajuda'. Below the menu bar, the terminal shows the command 'quental@quental-VirtualBox:~\$ node' being entered. The output of the command is 'Welcome to Node.js v12.14.1. Type ".help" for more information.' followed by a prompt '>'. The user enters 'console.log ('Ola. Este é o primeiro programa');' and the terminal outputs 'Ola. Este é o primeiro programa' followed by 'undefined' and another prompt '>'. The user enters '.exit' and the terminal returns to the shell prompt 'quental@quental-VirtualBox:~\$' with a cursor.

```
Ficheiro  Editar  Ver  Procurar  Terminal  Ajuda
quental@quental-VirtualBox:~$ node
Welcome to Node.js v12.14.1.
Type ".help" for more information.
> console.log ('Ola. Este é o primeiro programa');
Ola. Este é o primeiro programa
undefined
> .exit
quental@quental-VirtualBox:~$
```

# Primeiro programa ...

---

- Em Windows
  - Executar node
  - Console.log ('Meu primeiro programa');

 Node.js command prompt

```
C:\Users\quental>node
Welcome to Node.js v12.14.1.
Type ".help" for more information.
> console.log ('Ola. Este é o primeiro programa');
Ola. Este é o primeiro programa
undefined
> .exit

C:\Users\quental>
```

# Página em servidor HTTP ...

---

- O Node.js permite que se crie facilmente um servidor HTTP.
- Vamos criar uma página que devolva “Olá – Esta é a primeira página Web com node”
- Criar um ficheiro com, por exemplo, o nome server.js e inserir o seguinte código:

```
var http = require('http');  
  var server = http.createServer (function(req, res) {  
    res.writeHead(200);  
    res.end('Olá Aqui está a nossa primeira página Web');  
  });  
  server.listen(80);
```

# Página em servidor HTTP ...


---

- Executar o comando *node server.js*

 server.js - Bloco de notas

Ficheiro Editar Formatar Ver Ajuda

```
var http = require('http');
    var server = http.createServer(function(req, res) {
        res.writeHead(200);
        res.end('Ola <br>Aqui esta a nossa primeira pagina Web');
    });
server.listen(80);
```

 Node.js command prompt - node server.js

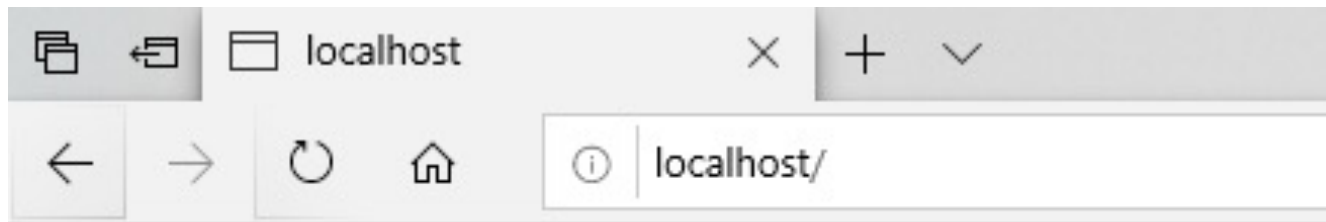
```
C:\Users\quental>node server.js
```



# Página em servidor HTTP ...

---

- Abrir um browser, e inserir o endereço `http://localhost`



Ola  
Aqui esta a nossa primeira pagina Web

# <NPM/>



- 
- NPM é o nome reduzido de Node Package Manager (Gestor de Pacotes do Node).
  - O NPM consiste nas duas principais funções:
    - É um repositório online para publicação de projetos de código aberto para o Node.js;
    - É um utilitário que corre na linha de comando que interage com o repositório online, que ajuda na instalação de pacotes, gestão de versão e gestão de dependências.
  - Existem muitas aplicações no repositório e são adicionadas novas todos os dias, como refere a própria documentação do NPM (<https://www.npmjs.com/>);
  - Os pacotes são instalados via linha de comandos.

# <NPM/>



- Como instalar o NPM:

O NPM facilita a partilha e a reutilização de códigos para *developers* de JavaScript.

- O NPM é distribuído com o NodeJS, significa que ao instalar o NodeJS, automaticamente é instalado o NPM.
- Como verificar se o NPM está instalado e que versão está instalada:
  - Correr o comando “npm -v”

```
C:\>npm -v  
6.13.4
```

# <NPM versions/>



- O NPM é um projeto separado do NodeJS, o que significa que tem um ciclo de vida independente e como consequência deve ser atualizado de forma independente.
  - Correr o comando “npm install npm@latest -g” para atualizar o npm para a versão mais atual.

```
C:\>npm install npm@latest -g
C:\Users\0101448\AppData\Roaming\npm\npm -> C:\Users\0101448\AppData\Roaming\npm\node_modules\npm\bin\npm-cli.js
C:\Users\0101448\AppData\Roaming\npm\npx -> C:\Users\0101448\AppData\Roaming\npm\node_modules\npm\bin\npx-cli.js
+ npm@6.13.7
added 434 packages from 860 contributors in 11.189s
```

- Correr o comando “npm install npm@next -g” para atualizar o npm para a versão mais atual, ainda não lançada. Poderá fazer sentido para poder testar os pacotes antes da nova versão ser lançada

```
C:\>npm install npm@next -g
C:\Users\0101448\AppData\Roaming\npm\npx -> C:\Users\0101448\AppData\Roaming\npm\node_modules\npm\bin\npx-cli.js
C:\Users\0101448\AppData\Roaming\npm\npm -> C:\Users\0101448\AppData\Roaming\npm\node_modules\npm\bin\npm-cli.js
+ npm@6.13.7
updated 1 package in 11.482s
```

# <package.json/>



- 
- Como regra geral, qualquer projeto que onde seja usado o Node.js é necessário ter um ficheiro package.json.
  - O que é um ficheiro package.json?
    - Um ficheiro package.json pode ser descrito como um manifesto do projeto, que inclui os pacotes e aplicações dos quais depende e metadados específicos, como o nome, a descrição e o autor do projeto.

# <package.json/>



- O ficheiro package.json, contém metadados específicos para o projeto, independentemente se é uma Web Application, um módulo Node.js. ou apenas uma biblioteca JavaScript.
- Estes metadados ajudam a identificar o projeto e atuam como uma linha de base para utilizadores e developers obterem informações sobre o projeto.

```
{
  "name": "metaverse", // The name of your project
  "version": "0.92.12", // The version of your project
  "description": "The Metaverse virtual reality. The final outcome of all virtual worlds, augmented reality, and the Internet.", // The description of your project
  "main": "index.js"
  "license": "MIT" // The license of your project
}
```

- É estruturado no formato Json que permite uma leitura fácil dos metadados.

# <package.json/>



- Criar um ficheiro package.json manualmente, pode ser algo difícil, contudo é possível utilizar a linha de comandos, e através do npm, fazer a criação automática do ficheiro.
- Para isso deverá ser usado o comando “npm init”:vc

```
C:\>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: _
```

# <package.json/>



- Com o comando iniciado é possível a criação do ficheiro passo a passo.
- O ficheiro package.json fica criado na diretoria do projeto.

```
package name: ai
version: (1.0.0) 1
Invalid version: "1"
version: (1.0.0) 1.0.0
description: Primeiro projeto
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\package.json:

{
  "name": "ai",
  "version": "1.0.0",
  "description": "Primeiro projeto",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes)
```



# <package.json/>



- 
- Outro aspecto importante do ficheiro package.json é que ele contém a coleção de dependências de qualquer projeto.
  - Essas dependências são os módulos dos quais o projeto depende para funcionar corretamente.
  - Ter essas dependências no package.json do seu projeto permite que o mesmo instale as versões dos módulos dos quais depende.
  - Ao executar o comando de instalação apropriado, dentro do projeto, é possível instalar todas as suas dependências descritas no ficheiro package.json.

# <package.json/>



- Permite a separação de dependências necessárias para a ambiente de produção e as dependências necessárias para o ambiente de desenvolvimento.
- No ambiente de produção, é provável que não seja necessária uma ferramenta para monitorização dos ficheiros CS, por exemplo, como será necessário no ambiente de desenvolvimento.

```
{
  "name": "metaverse",
  "version": "0.92.12",
  "description": "The Metaverse virtual reality. The final outcome of all virtual worlds, augmented reality, and the Internet.",
  "main": "index.js",
  "license": "MIT",
  "devDependencies": {
    "mocha": "~3.1",
    "native-hello-world": "^1.0.0",
    "should": "~3.3",
    "sinon": "~1.9"
  },
  "dependencies": {
    "fill-keys": "^1.0.2",
    "module-not-found-error": "^1.0.0",
    "resolve": "~1.1.7"
  }
}
```

# <Comandos/>



- Comandos npm essenciais:

## npm init

- O comando npm init é uma ferramenta que permite, passo a passo, definir a estrutura do projeto. Define:
  - Nome do projeto,
  - Versão inicial do projeto,
  - Descrição do projeto,
  - O entry point do projeto (o ficheiro main),
  - O comando de teste do projeto,
  - Repositorio GIT do projeto (caso exista),
  - Keywords do projeto (tags relacionadas com o projeto),
  - Licença do projeto.

# <Comandos/>



- Comandos npm essenciais:

**npm install <modulo>**

- Comando que permite a instalação de um modulo necessário para o projeto.

**npm install <modulo> --save**

- Comando que permite a instalação de um modulo necessário para o projeto e cria o registo no ficheiro package.json, criado assim a dependência do módulo no projeto.

**npm install <modulo> --save-dev**

- Comando que permite a instalação de um modulo necessário para o projeto e cria o registo no ficheiro package.json, criado assim a dependência do módulo no projeto para o ambiente de desenvolvimento.

# <Comandos/>



- Comandos npm essenciais:

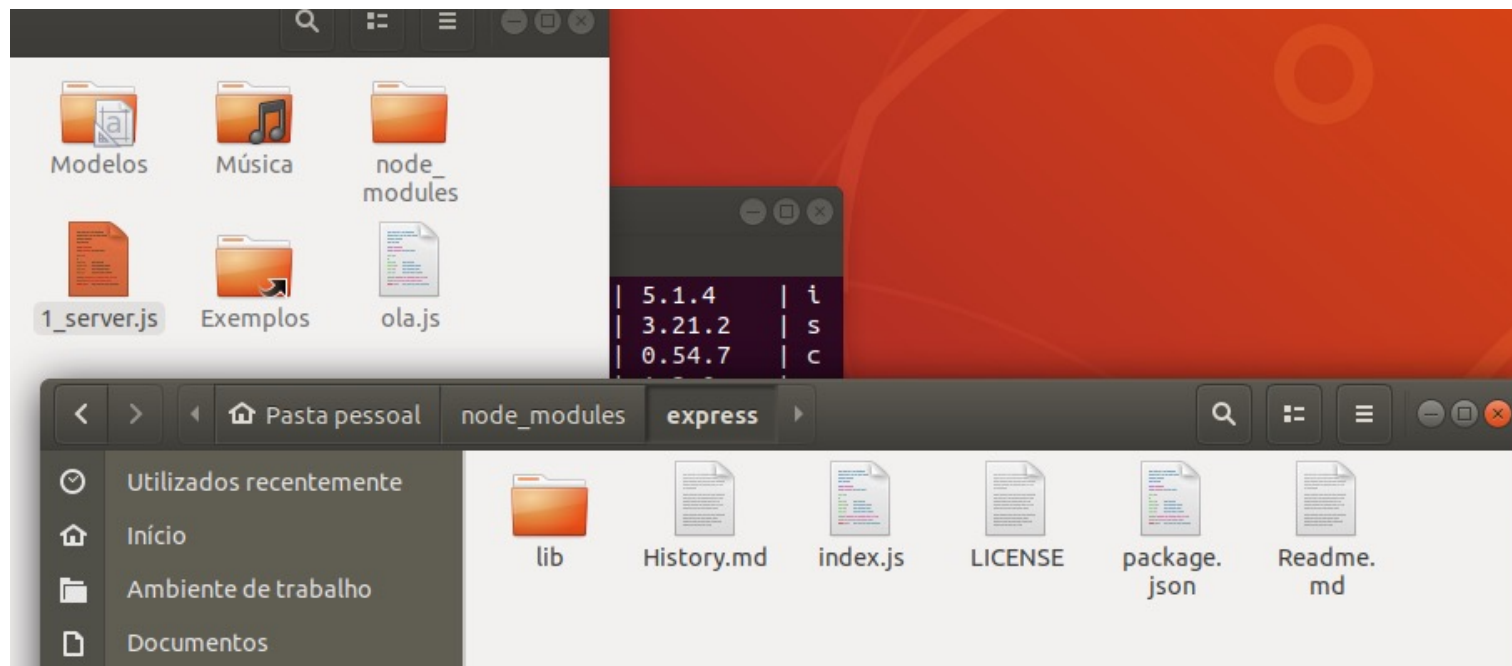
## `npm install <modulo> --global`

- Comando que permite a instalação de um modulo de forma global no sistema.
- Os módulos globais podem ser muito úteis - existem várias ferramentas, utilitários onde o seu uso geral pode ser uma vantagem.
- Para saber quais os módulos globais instalados, podemos recorrer ao comando: `npm --global list`
- É possível a instalação de vários módulos em simultâneo:

```
npm i express moment lodash mongoose body-parser webpack
```

# Instalação global vs local


- Por omissão, o NPM instala qualquer dependência no modo local: instalação do pacote no diretório *node\_modules*, localizado na pasta em que a aplicação Node está presente
- Pacotes locais são acessíveis pelo método `require()`. Por exemplo, quando instalamos o módulo *express*, é criado o diretório *node\_modules* no diretório atual.



# Instalação global vs local

---

- Pacotes / dependências instalados globalmente são armazenados no diretório do sistema.
- Tais dependências podem ser usadas na função CLI (Command Line Interface) de qualquer node.js, mas não podem ser importadas usando o `require ()` no aplicativo Node diretamente
- Vamos instalar o módulo `express` usando a instalação global

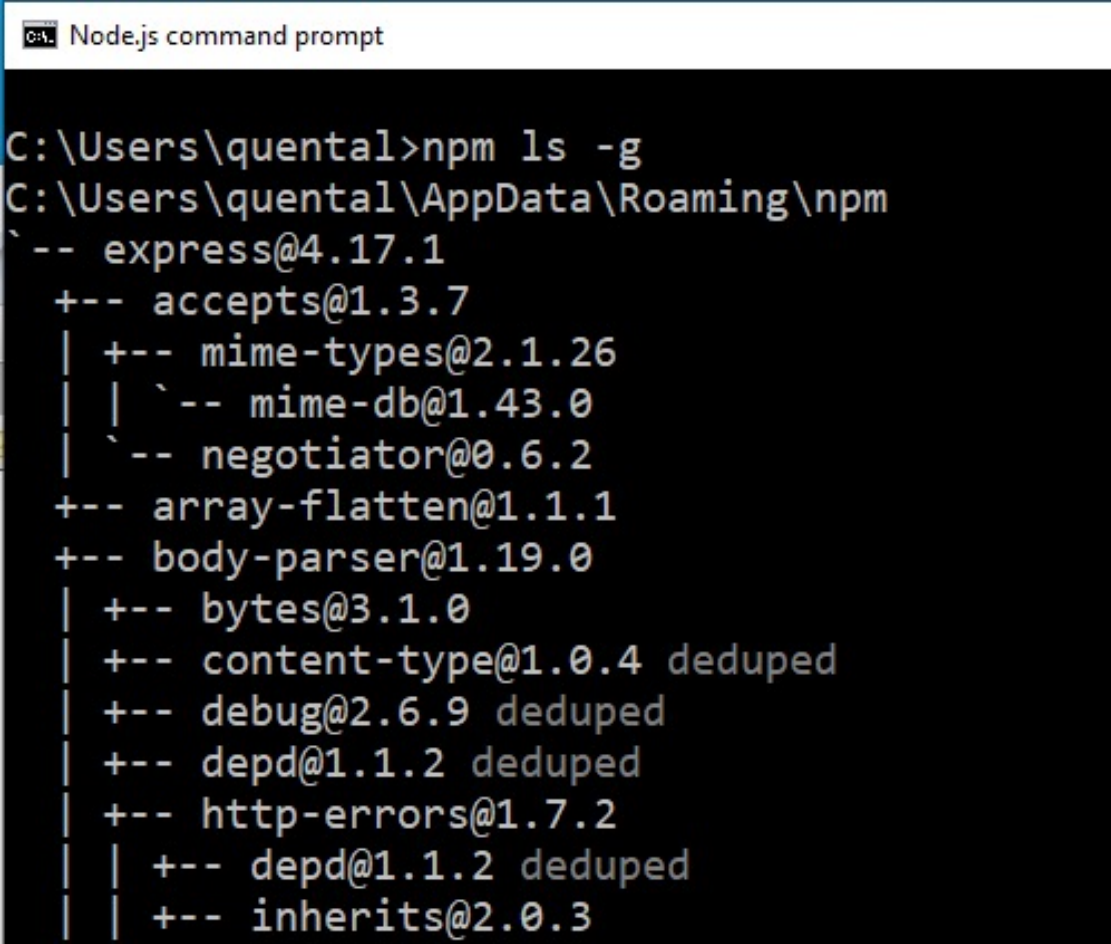
 Node.js command prompt

```
C:\Users\quental>npm install express -g
+ express@4.17.1
added 50 packages from 37 contributors in 8.67s
```

# Instalação global vs local

---

- Ver versão dos módulos e local de instalação: ***npm ls -g***



```
Node.js command prompt

C:\Users\quental>npm ls -g
C:\Users\quental\AppData\Roaming\npm
`-- express@4.17.1
   |-- accepts@1.3.7
   |   |-- mime-types@2.1.26
   |   |   `-- mime-db@1.43.0
   |   `-- negotiator@0.6.2
   |-- array-flatten@1.1.1
   |-- body-parser@1.19.0
   |   |-- bytes@3.1.0
   |   |-- content-type@1.0.4 deduped
   |   |-- debug@2.6.9 deduped
   |   |-- depd@1.1.2 deduped
   |   |-- http-errors@1.7.2
   |   |   |-- depd@1.1.2 deduped
   |   |   |-- inherits@2.0.3
```



# Criar um módulo

---

- É necessário gerar o package.json. Vamos gerar o esqueleto básico do package.json
- npm init

npm

```
C:\Users\quental>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (quental)
```

# Criar um módulo

---

- Conteúdo do package.json

```
license: (ISC)
About to write to C:\Users\quental\package.json:

{
  "name": "quental",
  "version": "1.0.0",
  "description": "Primeiro package.json",
  "main": "exemplo.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js"
  },
  "author": "",
  "license": "ISC"
}
```

- Registrar utilizador no repositório npm: npm adduser

```
C:\Users\quental>npm adduser
Username: cq
Password:
Email: (this IS public) quental@estgv.ipv.pt
```

# Criar um módulo

---

- Registrar utilizador no repositório npm
  - `npm adduser`

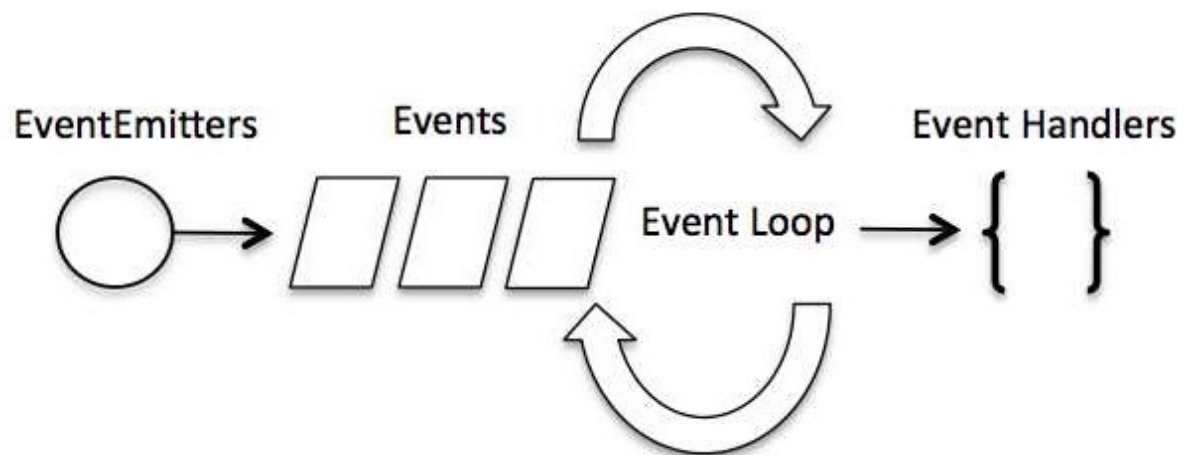
```
C:\Users\quental>npm adduser
Username: cq
Password:
Email: (this IS public) quental@estgv.ipv.pt
```

- Publicar módulo
  - `npm publish`
  - Se tudo estiver bem, o módulo fica disponível para instalação como qualquer outro módulo do node.js

# Programação orientada a eventos

---

- Numa aplicação orientada a eventos, geralmente há um loop principal que escuta eventos e, em seguida, dispara uma função de retorno de chamada quando um desses eventos é detetado.
- O node.js possui vários eventos incorporados, disponíveis através do módulo de eventos e da classe *EventEmitter*, que são usados para ligar eventos e escuta de eventos



# Programação orientada a eventos

---

```
// Import events module
```

```
var events = require('events');
```

```
// Create an EventEmitter object
```

```
var EventEmitter = new events.EventEmitter();
```

- Sintaxe para ligar um evento a um event handler

```
// Bind event and event handler as follows
```

```
eventEmitter.on('eventName', eventHandler);
```

```
// Fire an event
```

```
eventEmitter.emit('eventName');
```

# Programação orientada a eventos

---

- **Exemplo** (código guardado em main.js) e executar *node main.js*. Consultar métodos em [https://nodejs.org/api/events.html#events\\_class\\_eventemitter](https://nodejs.org/api/events.html#events_class_eventemitter)

```
// Import events module
var events = require('events');
// Create an EventEmitter object
var EventEmitter = new events.EventEmitter();
// Create an event handler as follows
var connectHandler = function connected() {
    console.log('connection succesful.');
```

    // Fire the data\_received event

```
    EventEmitter.emit('data_received');
} // Bind the connection event with the handler
EventEmitter.on('connection', connectHandler);
// Bind the data_received event with the anonymous function

EventEmitter.on('data_received', function() {
    console.log('data received succesfully.');
```

});

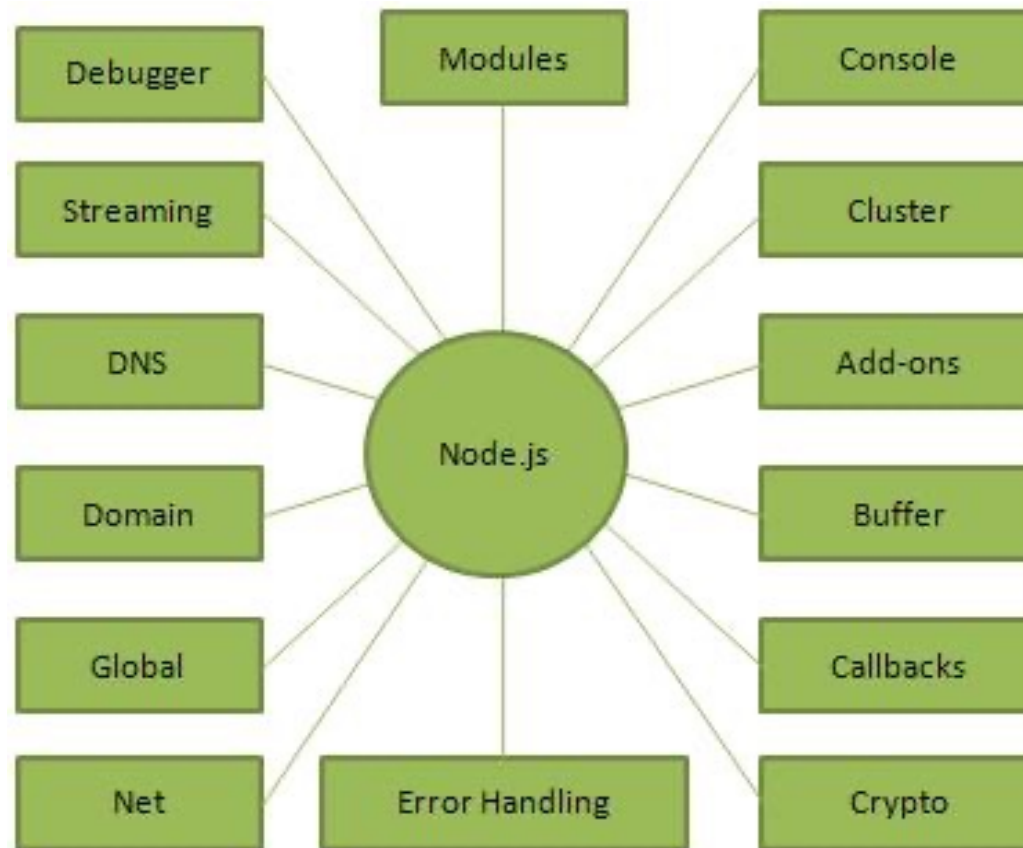
```
// Fire the connection event
EventEmitter.emit('connection');
console.log("Program Ended.");
```



The image part with relationship ID rld3 was not found in the file.

# Módulos do node.js ...

---



# Alguns módulos ...

---

- **Os Module:** Provides basic operating-system related utility functions.  
<https://nodejs.org/api/os.html>
- **PATH Module:** Provides utilities for handling and transforming file paths.  
<https://nodejs.org/api/path.html>
- **NET Module:** Provides both servers and clients as streams. Acts as a network wrapper.  
<https://nodejs.org/api/net.html>
- **DNS Module:** Provides functions to do actual DNS lookup as well as to use underlying operating system name resolution functionalities.  
<https://nodejs.org/api/dns.html>



# Bibliografia ...

---

- Node.js API: <https://nodejs.org/api>
- Learn node.js: <https://www.tutorialspoint.com/nodejs/>
- Whatis V8: <https://v8.dev/>
- GitHub: <https://github.com/goldbergonyi/nodebestpractices>
- Configurar npm: <https://docs.npmjs.com/files/package.json>
- Rubens, João; *Primeiros passos com node.js*, Casa do código, ISBN 978-85-5519-285-2
- Instalação local versus global: <https://nodejs.org/en/blog/npm/npm-1-0-global-vs-local-installation/>
- Atualizar packages: <https://docs.npmjs.com/updating-packages-downloaded-from-the-registry>

Endereços úteis:

- Módulos disponíveis em <https://www.npmjs.com/>
- Express: <http://expressjs.com/>
- Executar testes online:  
[https://www.tutorialspoint.com/execute\\_nodejs\\_online.php](https://www.tutorialspoint.com/execute_nodejs_online.php)