Plataforma Node.js

Gestão de pacotes e módulos

Prof. Rommel Vieira Carneiro

Pacotes e Módulos

Pacote

Pode ser ...

a) ... uma pasta contendo um programa descrito por um arquivo package.json
b) ... um arquivo compactado (tar.gz) contendo (a)
c) ... uma URL que direciona para (b)
d) ... uma string <name>@<version> publicada no registro com (c)
e) ... uma string <name>@<tag> que aponta para (d)
f) ... uma string <name> que tenha uma tag latest que satisfaça (e)
g) ... uma URL git que, quando clonada, resulta em (a)

Fonte: About packages and modules (npm Docs)
Fonte da imagem: Package clipart

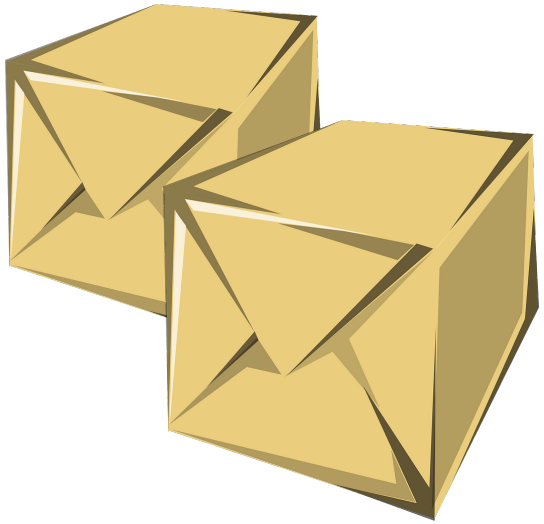
Módulo

Pode ser ...

- ... um arquivo Javascript
- ... uma pasta com um arquivo package.js que tenha um atributo main

Três tipos de módulos

- Módulos internos (core)
  ex: process, os, dns
- Módulos locais
  ex: lib_a, lib_b
- Módulos de terceiros
  ex: express, mongoose



Gerenciadores de Pacotes e Dependências

Os **gerenciadores de pacotes** auxiliam o desenvolvedor nas tarefas de instalação, atualização, configuração e remoção de programas e componentes (**Pacotes**).

Os **gerenciadores de dependências** atuam de forma similar, automatizando tarefas de download e configuração de pacotes, porém, com foco em um projeto específico.

♦ ♦ ♦

Enquanto um **gerenciador de pacotes** permite configurar todo o ambiente de desenvolvimento, um **gerenciador de dependências** configura um projeto específico.

NPM

- O NPM é, ao mesmo tempo, Gerenciador de Pacotes e um Gerenciador de dependências para o ambiente JavaScript e Node.Js
- O NPM é instalado juntamente com o Node.Js
- Possui um repositório de mais de 600.000 pacotes
- Possui versão pública e versão corporativa

Web Site: https://www.npmjs.com/

NPM – Contexto Global x Contexto Local

Contexto Global

- Refere-se ao computador como um todo
- Pasta no Linux e MacOs: **`/usr/local/lib/node_modules`**
  Pasta no Windows: **`%app_data%\Roaming\npm\node_modules`**

Contexto Local

- Refere-se ao projeto específico
- Configurações mantidas no arquivo **`package.json`**
- Módulos mantidos no diretório **`node_modules`**
- Utilize o npx para executar partes dos módulos localmente

```
$ npx nodemon        // Executa o nodemon instalado localmente no projeto
```

NPM – Primeiros passos

Nesta parte vamos verificar a versão do NPM e, caso necessário, atualizá-lo.

Em seguida, vamos inicializar um projeto, criando o arquivo de configurações com nome **`package.json`** na pasta raiz. Serão solicitadas as informações referentes ao módulo que compõem o arquivo de configurações.

```
$ npm –version                    // Verifica a versão do NPM
5.6.0

$ npm install npm@latest –g       // Atualiza a versão do NPM

$ npm init                        // Inicializa o arquivo package.json para o projeto
```

NPM – Instalação de Módulos

Os módulos podem ser instalados localmente ou globalmente.

Os módulos locais são colocados na subpasta **`node_modules`**, dentro do projeto e podem compor uma lista de dependências do projeto no arquivo **`package.json`**.

```
$ npm install express                    // Instalação do pacote express (EXEMPLO)
                                         // localmente no projeto atual

$ npm install express@3.0.0              // Instalação de versão específica

$ npm install express --save             // Opção --save informa para incluir pacote
                                         // nas dependências do projeto

$ npm install nodemon -g                 // Instalação do pacote globalmente

$ npm install nodemon --save-dev      // Opção –save-dev informa para incluir pacote
                                         // nas dependências de desenvolvimento do projeto
```

NPM – Estrutura do arquivo package.json

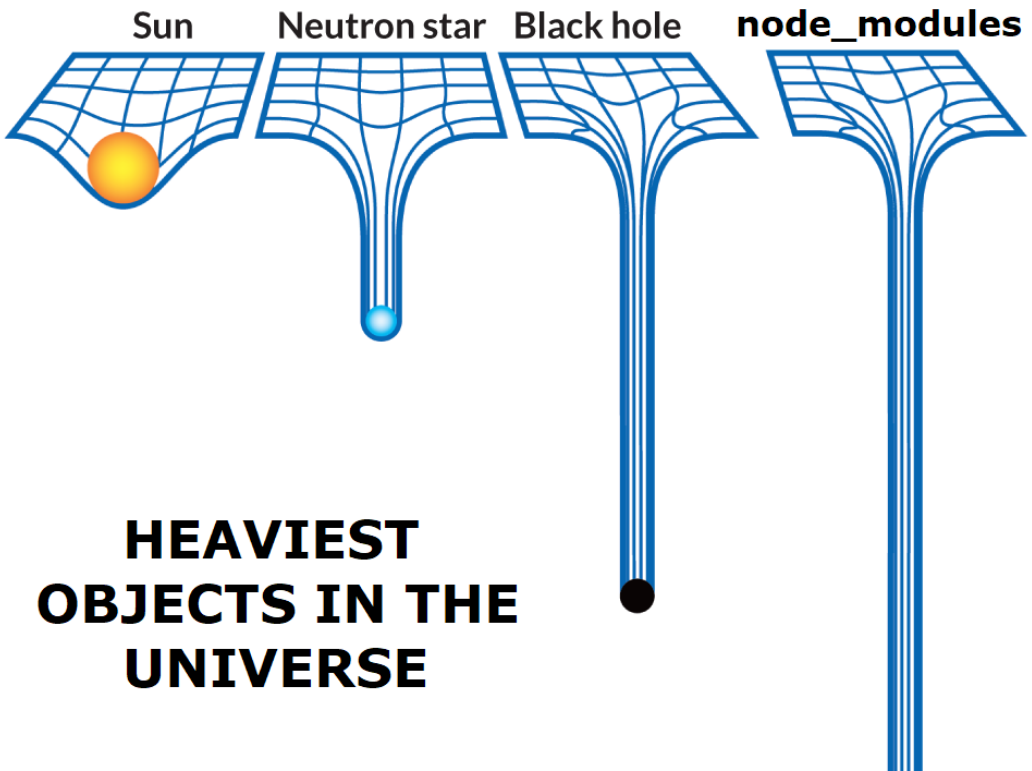
```
{
  "name": "projeto_teste",
  "version": "1.0.0",
  "description": "Projeto de Teste",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Rommel Carneiro",
  "license": "ISC",
  "dependencies": {
    "express": "^4.16.3"
  }
}
```

NPM – Estrutura do arquivo `package.json`

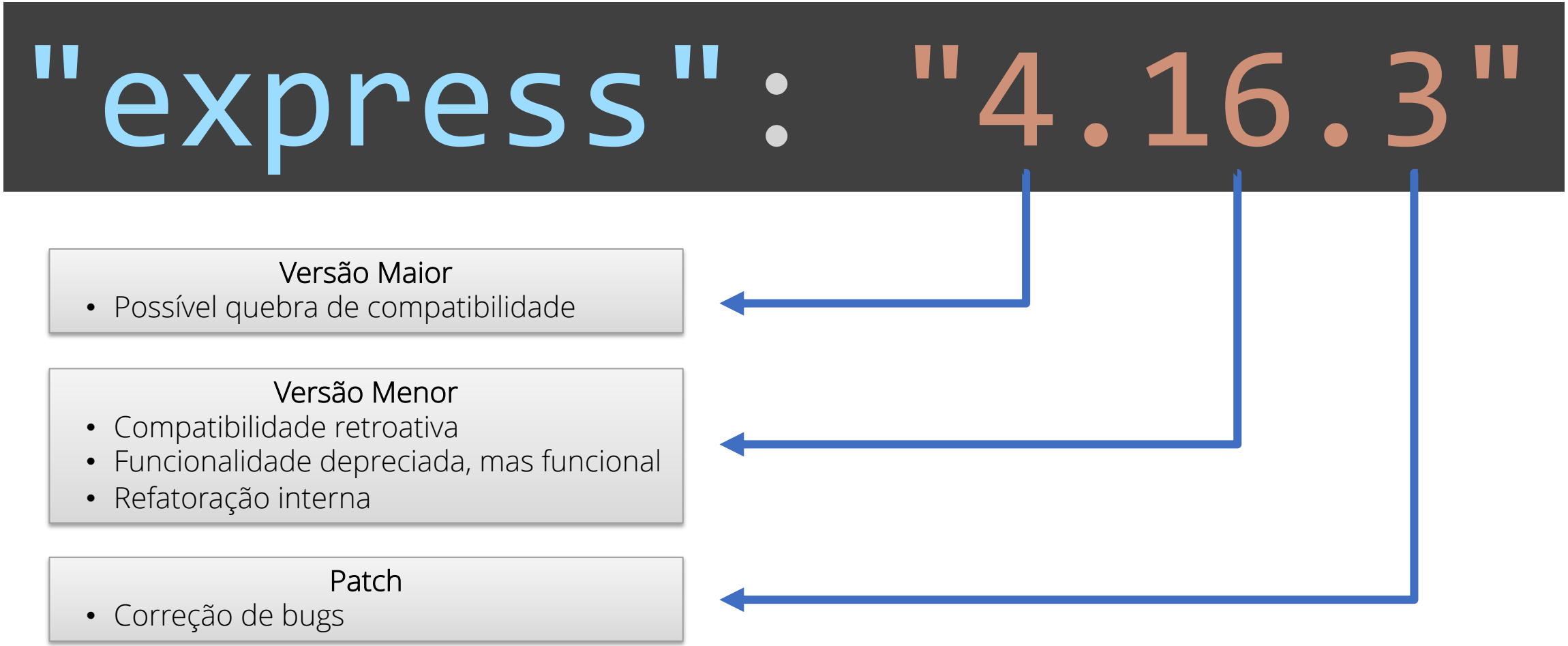
Atributo	Descrição
name	Nome do pacote
version	Versão do pacote
description	Descrição do pacote
homepage	Site Web do pacote
author	Autor do pacote
contributors	Nome dos colaboradores do pacote
dependencies	Lista de dependências associadas ao pacote
devDependencies	Lista de dependências do ambiente de desenvolvimento
main	Ponto de entrada do pacote
scripts	Comandos de script para o ambiente de desenv.
keywords	Palavras-chave associadas ao pacote

NPM – Pasta node_modules

- Repositório dos módulos do projeto
- Pasta **`node_modules/.bin`** → Command Line Interfaces (CLI)
  Execução via **`npx`**
- Inclusão no **`.gitignore`**



NPM – Versionamento Semântico



Fonte: Semantic Versioning - https://semver.org

NPM – Versionamento Semântico

Utilize o versionamento semântico para estabelecer a dinâmica de atualização dos pacotes do projeto.

```
{
  . . .
  "dependencies": {
     "express": "4.16.3",      // Versão exata 4.16.3
     "express": "^4.16.3",     // Aceita versões 4.*.*
     "express": "~4.16.3",     // Aceita versões 4.16.*
     "express": ">4.16.3",     // Aceita versões superiores a 4.16.3
     "express": ">=4.16.3",    //  ||  superiores ou iguais a 4.16.3
     "express": "<4.16.3",     //  ||  inferiores a 4.16.3
     "express": "<=4.16.3",    //  ||  inferiores ou iguais a 4.16.3
  }
}
```

NPM – Listagem de Módulos Instalados

Para verificar os módulos instalados localmente, use o comando **`npm list`**.
É possível verificar ainda os módulos globais com o comando **`npm list -g`**.

OBS: A lista traz todas as dependências em forma de árvore. Use o parâmetro **`--depth=0`** para informar que você deseja ver apenas o primeiro nível da árvore.

```
$ npm list                  // Lista módulos instalados no projeto

$ npm list -g --depth=0     // Lista módulos instalados globalmente
```

NPM – Atualização de Módulos

Para verificar os módulos desatualizados, use o comando **`npm outdated`**.
Para atualizar um módulo, utilize o comando **`npm update`**.

```
$ npm outdated          // Verifica os pacotes desatualizados
Package         Current  Wanted  Latest  Location
@angular/cli      1.1.3   1.7.4   6.0.3
grunt             1.0.1   1.0.2   1.0.2
http-server      0.10.0  0.10.0  0.11.1

$ npm update grunt      // Atualiza o modulo grunt
```

NPM – Desinstalação de Módulos

Para desinstalar módulos locais do projeto ou globais, utilize o comando **`npm uninstall`**.

```
$ npm uninstall express      --> Desinstalação do modulo express
                                 da pasta do projeto projeto atual

$ npm uninstall express -g   --> Desinstalação do módulo globalmente
```

NPM – Configurações do ambiente

Para verificar as configurações do ambiente do NPM tais como: caminho do Node.JS e pasta dos módulos globais utilize o comando **`npm config list`**

```
$ npm config list
; cli configs
metrics-registry = "https://registry.npmjs.org/"
scope = ""
user-agent = "npm/6.0.1 node/v10.0.0 win32 x64"

; globalconfig C:\Users\UserX\AppData\Roaming\npm\etc\npmrc
msvs_version = "2015"
python = "C:\\Users\\UserX\\.windows-build-tools\\python27\\python.exe"

; builtin config undefined
prefix = "C:\\Users\\UserX\\AppData\\Roaming\\npm"

; node bin location = C:\Desenvolvimento\nodejs\node.exe
; cwd = C:\Desenvolvimento\xampp\htdocs\AulasWeb\Lab-CSS-Sprites
; HOME = C:\Users\UserX
; "npm config ls -l" to show all defaults.
```