**NPM**

**Índice**

1. Introdução
2. O que é?
3. Instalação 3.1. Windows 3.2. Linux 3.3. OSX
4. Inicialização
5. Gerenciamento de pacotes 5.1. Buscando 5.2. Se informando 5.3. Instalando 5.4. Atualizando 5.5. Removendo 5.6. Pacotes instalados 5.7. Cache 5.8. Reconstruindo 5.9. Sufixos de versões
6. Dicas
7. Conclusão

**1. Introdução**

Olá a todos meus amigos!

Depois de um ano sem por nada no ar, resolvi voltar a ativa com uma série de artigos visando ajudar alguns desenvolvedores que podem ter ficado perdidos no tempo no mercado de front-end.

Digo isso porque nos últimos anos esta área teve um verdadeiro BUM de novas tecnologias, ferramentas e abordagens e nós programadores frequentemente nos encontramos sem tempo para estudar coisas novas devido ao trabalho que sempre nos exige mais em menos tempo.

Quando finalmente um desenvolvedor resolve estudar, não é incomum se deparar com tanta coisa nova que não sabe por onde começar, e quando escolhe algo para estudar, descobre que este algo é baseado em outro algo que é baseado em outros e por aí a fora... oque pode ser profundamente desanimador.

Por isso, meu objetivo com estes artigos é de ajudar estes desenvolvedores a conseguirem se encontrar, a ter um chão onde pisar e serem capazes de olhar em volta e serem capazes de se orientar, podendo assim passar a estudar por conta própria com um novo animo e fogo no olhar.

Irei abordar neste primeiro artigo uma introdução ao NPM ao qual acredito ser uma tecnologia base utilizada por várias ferramentas.

**2. O que é?**

O NPM é um gerenciador de pacotes do NodeJS, Oque significa que possui online um repositório de projetos de código aberto os quais podem ser baixados com um único comando da linha de comando para dentro do nosso projeto.

O mais importante é que possuímos um controle de versões dos pacotes de tal maneira que ele alerta quando um pacote não é compatível com outro, ou seja se estamos tentando utilizar um plugin do jQuqey que não seja compatível com a versão do jQuery instalada, ele irá nos alertar e perguntar oque desejamos fazer (Atualizar jQuery?, Manter versão menos recente do plugin?, Não instalar o plugin?). Isso vale tanto para instalação como atualização.

**3. Instalação**

**3.1. Windows**

A instalação no Windows é bastante simples, basta baixar o instalador da versão instável do NodeJs no [site oficial](https://nodejs.org/) e executá-lo. O famoso Next, next, next do Windows.

Lembrando que o NPM faz parte do NodeJs, sendo por isso necessário instalar o NodeJs.

**3.2. Linux**

Existem várias maneiras de instalar o NodeJs/NPM no Linux, dependendo da sua distribuição.

Listarei o comando de instalação para algumas destas distribuições, para ver os comandos para outras distribuições [clique aqui para acessar a devida página na documentação oficial](https://nodejs.org/en/download/package-manager/).

**Arch Linux**

pacman -S nodejs npm

**Debian, Ubuntu, Mint e derivados**

sudo apt-get install -y nodejs

**Fedora**

yum -y install nodejs

**openSUSE**

zypper install nodejs4

**3.3. OSX**

curl "https://nodejs.org/dist/latest/node-${VERSION:-$(wget -qO- https://nodejs.org/dist/latest/ | sed -nE 's|.\*>node-(.\*)\.pkg</a>.\*|\1|p')}.pkg" > "$HOME/Downloads/node-latest.pkg" && sudo installer -store -pkg "$HOME/Downloads/node-latest.pkg" -target "/"

Para alternativas [clique aqui para acessar a devida página na documentação oficial](https://nodejs.org/en/download/package-manager/#osx).

**4. Inicialização**

<https://docs.npmjs.com/cli/init>

Uma vez que temos o NPM instalado na nossa máquina, vamos precisar inicializar ele no diretório do projeto onde desejamos utilizá-lo.

Para isso vamos acessar o diretório pela linha de comando e dentro dele executar o seguinte comando:

npm init

Será feita uma série de perguntas, não precisa se assustar com elas uma vez que não irão afetar o funcionamento do NPM no seu projeto.

Estas perguntas dizem respeito tão somente aos detalhes do seu projeto caso você venha a disponibilizá-lo no NPM, tais como nome, descrição, versão, repositório GIT e etc.

**5. Gerenciamento de pacotes**

**5.1. Busca**

Ocasionalmente acontecerá de não sabermos o nome do pacote que desejamos instalar, por exemplo o AngularJS, qual o nome? Será angular, angularjs, angularjs1?

Para resolver esta questão nós podemos efetuar uma busca, e existem duas maneiras para isso. Uma através da linha de comando e outra através de um navegador web.

**5.1.1 Linha de comando**

<https://docs.npmjs.com/cli/search>

Acesse o diretório do projeto pela linha de comando e dentro dele execute o seguinte comando:

npm search angular

O NPM irá buscar e exibir os resultados que baterem com "angular".

Uma informação importante é que na primeira vez que esse comando for executado na máquina, o NPM irá baixar uma lista completa de pacotes online e tentar montá-la na máquina. Em algumas máquinas esta operação poderá estourar a memória causando erro no comando.

**5.1.2. Navegador WEB**

Para buscar um pacote do NPM pelo navegador, basta acessar o [site do NPM](http://npmjs.com/) e entrar com o termo de pesquisa desejado na caixa de busca. Uma lista de pacotes associados será exibida.

**5.2. Se informando**

<https://docs.npmjs.com/cli/view>

Uma vez que temos um nome de pacote, pode ocorrer de querermos ter maiores informações a respeito dele, para isso também tempos um comando.

npm view angular

Serão exibidas todas as informações a respeito do pacote do angular.

**5.3. Instalando**

<https://docs.npmjs.com/cli/install>

Existem várias maneiras - e suas combinações - de instalar um pacote, veremos cada uma:

**5.3.1. Instalação simples**

Desejamos instalar o angular, simples assim.

Para isso acesse o diretório do projeto pela linha de comando e dentro dele execute o seguinte comando:

npm install angular

Desta forma o angular estará adicionado dentro do diretório "nodeModules" na raiz do projeto.

**5.3.2. Instalação de uma dada versão**

Desejamos instalar uma versão específica do angular (1.6.0 por exemplo).

Para isso acesse o diretório do projeto pela linha de comando e dentro dele execute o seguinte comando:

npm install angular@1.6.0

Desta forma o angular v1.6.0 estará adicionado dentro do diretório "nodeModules" na raiz do projeto.

**5.3.3. Instalação global**

Também é é possível instalar um pacote globalmente, de forma a estar disponível para o computador inteiro, independentemente do projeto.

Para isso acesse o diretório do projeto pela linha de comando e dentro dele execute o seguinte comando:

npm install -g gulp

Desta forma o gulp estará instalado globalmente em sua máquina e disponível para qualquer projeto.

**5.3.4. Salvando a dependência**

Se estamos adicionando um pacote ao projeto, possivelmente significa que ele é uma dependência para que o mesmo funcione corretamente.

Para que seja possível futuramente instalar todas as dependências com um único comando - sem ter que instalar uma a uma -, é necessário salvar as dependências.

Este processo também é necessário para que o NPM faça atualização dos pacotes e mantenha um controle sobre a compatibilidade das versões dos pacotes.

Para salvar uma dependência (durante ou depois da instalação) acesse o diretório do projeto pela linha de comando e dentro dele execute o seguinte comando:

npm install --save angular

Pronto, agora o arquivo packages.json do NPM no seu projeto possuirá uma posição para dependências com o nome do pacote (angular) e a versão do mesmo.

**5.3.5. Salvando a dependência de desenvolvimento**

Um pacote também pode ser uma dependência de desenvolvimento, ou seja, ele somente é necessário para o ambiente de desenvolvimento, nunca no ambiente de produção.

Para salvar uma dependência de desenvolvimento (durante ou depois da instalação) acesse o diretório do projeto pela linha de comando e dentro dele execute o seguinte comando:

npm install --save-dev angular

Pronto, agora o arquivo packages.json do NPM no seu projeto possuirá uma posição para dependências de desenvolvimento com o nome do pacote (angular) e a versão do mesmo.

**5.3.6. Instalando todas as dependências**

Digamos que acabamos de clonar um projeto e ele esta sem as suas dependências, porém com todas elas informadas no packages.json. Como podemos fazer para instalar todas elas de uma vez só?

Acesse o diretório do projeto pela linha de comando e dentro dele execute o seguinte comando:

npm install

Tão simples que chega a doer né?

**5.3.7. Instalando todas as dependências de produção**

Ao invés de instalar todas as dependências, queremos instalar somente aquelas que não sejam de dependência.

Para isso acesse o diretório do projeto pela linha de comando e dentro dele execute o seguinte comando:

npm install --production

**5.4. Atualizando**

<https://docs.npmjs.com/cli/update>

Também podemos atualizar nossos pacotes que estejam desatualizados, veja:

**Atualizar todos os pacotes do projeto**

npm update --save

**Atualizar todos os pacotes globais**

npm update -g

**Atualizar determinado pacote do projeto**

npm update --save angular

**Atualizar determinado pacote global**

npm update -g gulp

O --save (ou --save-dev) é necessário para que a mudança de versão também seja salva no package.json.

Lembrando que a atualização só irá rodar com pacotes que estiverem presentes na lista de dependências do package.json.

**5.5. Removendo**

<https://docs.npmjs.com/cli/uninstall>

Também podemos remover um determinado pacote que não desemos mais, veja:

**Remoção simples**

npm uninstall angular

**Remoção salvando nas dependências**

npm uninstall --save angular

**Remoção global**

npm uninstall -g gulp

**5.6. Pacotes instalados**

<https://docs.npmjs.com/cli/ls>

Para ver quais os pacotes estão instalados no projeto, acesse o diretório do projeto pela linha de comando e dentro dele execute o seguinte comando:

npm ls

O NPM

poderá informar de pacotes instalados que não se encontram no packages.json.

Para ver os pacotes globais instalados:

npm ls -g

**5.7. Cache**

<https://docs.npmjs.com/cli/cache>

O NPM possui um sistema de cache que armazena os pacotes baixados de forma que, ao tentarmos instalar um deles em um novo projeto, ele não realiza um novo download, somente copia do cache.

Existem alguns comandos do cache que podem vir a ser úteis:

**Listar pacotes cacheados**

npm cache ls

**Limpar cache**

npm cache clean

**5.8. Reconstruindo pacotes**

<https://docs.npmjs.com/cli/rebuild>

Eventualmente você irá atualizar o NPM, processo no qual ele é removido e uma nova versão instalada.

Quando isso ocorre, o NPM "perde a memória" dos pacotes instalados e os mesmos param de funcionar.

Para resolver este problema podemos executar um comando que irá reconstruir todos os pacotes para a nova versão do NPM, veja:

npm rebuild

npm rebuild -g

**5.9. Sufixos de versões**

A grande vantagem do NPM é o controle de compatibilidade dentre os pacotes, isso se deve ao fato de, através dos sufixos no package.json, podermos especificar quais as versões podemos utilizar (fato também utilizado pelos desenvolvedores dos pacotes).

Isso nos permite controlar melhor a estabilidade dos pacotes do projeto. Irei aqui mostrar como isso funciona.

Primeiro vamos entender como funciona uma versão de acordo com o [SEMVER](http://semver.org/). Suponhamos uma versão 1.2.3, eis oque significa cada posição:

* Primeira posição - É a posição que diz respeito a mudanças no pacote que alterar a compatibilidade do mesmo, alguma função removido ou algum parâmetro alterado... Se o seu projeto não for alterado para se adaptar a esta mudança, ele irá quebrar.
* Segunda posição - É a posição que diz respeito a novas funcionalidades ou recursos no pacote, porém que não tenham alterado a compatibilidade do mesmo. Nenhuma alteração em seu projeto será necessária para continuar funcionando.
* Terceira posição - É a posição que diz respeito a correções no pacote, porém que não tenham alterado a compatibilidade do mesmo. Nenhuma alteração em seu projeto será necessária para continuar funcionando.

Sabendo disso, vamos compreender os sufixos:

* \* - Se encontrar um asteristico no lugar versão do pacote, significa que o NPM poderá instalar/atualizar sempre para a versão mais recente do pacote.
* 1.2.3 - Se encontrar uma versão sem qualquer prefixo no pacote, sígnica que o NPM irá sempre instalar aquela versão, nunca podendo atualizar.
* ^1.2.3 - Se encontrar uma versão sufixada com um circunflexo, significa que o NPM poderá atualizar somente para versões que alteram a terceira posição da versão (ou seja, atualizações de correção que mantenham a compatibilidade)
* ~1.2.3 - Se encontrar uma versão sufixada com um til, significa que o NPM poderá atualizar somente para versões que alteram a segunda e a terceira posição da versão (ou seja, atualizações novos recursos e de correção, porém mantendo a compatibilidade)
* >1.2.3 - Se encontrar uma versão sufixada com um um simbolo de maior, significa que o NPM poderá instalar e atualizar somente versões maiores que a versão dada.
* >=1.2.3 - Se encontrar uma versão sufixada com um um simbolo de maior ou igual, significa que o NPM poderá instalar e atualizar somente versões maiores ou igual que a versão dada.
* <1.2.3 - Se encontrar uma versão sufixada com um um simbolo de menor, significa que o NPM poderá instalar e atualizar somente versões menores que a versão dada.
* <=1.2.3 - Se encontrar uma versão sufixada com um um simbolo de menor ou igual, significa que o NPM poderá instalar e atualizar somente versões menores ou igual que a versão dada.

Dependendo de como forem adicionados os sufixos, o NPM poderá informar alguma incompatibilidade. Por exemplo... Se utilizamos uma versão fixa e antiga do jQuery e um plugin do jQuery com asteristico (ou seja, o mais recente), esta versão não será compatível com o jQuery. Logo, o NPM irá informar e perguntar oque fazer.

**6. Dicas**

Existem algumas poucas dicas que gostaria de acrescentar.

Primeiro, a maior parte dos comandos (ver links da documentação) aceitam o parâmetro --json para exibir o resultado no formato JSON.

Segundo, a linha de comando permite enviar a saída do comando para um arquivo, combinando este fato com a dica acima nós podemos fazer por exemplo:

npm view --json angular > detalhes.json

E teremos todos os detalhes do angular disponíveis no arquivo detalhes.json.