**EBook Gratis**

# APRENDIZAJE

# npm

Free unaffiliated eBook created from
**Stack Overflow contributors.**

#npm

# Tabla de contenido

# Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: npm

It is an unofficial and free npm ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official npm.



Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

# Capítulo 1: Empezando con npm

## Observaciones

npm es el gestor de paquetes predeterminado para Node.js. Está escrito en su totalidad en JavaScript y permite instalar y distribuir principalmente módulos de JavaScript en el registro. Los paquetes se tratan en el formato CommonJS , y son reconocibles por la presencia de un archivo `package.json` .

## Versiones

| Versión | Fecha de lanzamiento |
|---|---|
| v4.0.0 | 2016-10-21 |
| v3.10.0 | 2016-06-17 |
| v3.9.0 | 2016-05-06 |
| v2.15.0 | 2016-03-11 |
| v3.8.0 | 2016-02-26 |
| v3.7.0 | 2016-01-29 |
| v3.6.0 | 2016-01-21 |
| v3.5.0 | 2015-11-20 |
| v3.4.0 | 2015-11-06 |
| v3.3.0 | 2015-08-14 |
| v2.14.0 | 2015-08-14 |
| v3.2.0 | 2015-07-25 |
| v3.1.0 | 2015-07-03 |
| v2.13.0 | 2015-07-03 |
| v3.0.0 | 2015-06-26 |
| v2.12.0 | 2015-06-19 |
| v2.11.0 | 2015-05-22 |
| v2.10.0 | 2015-05-08 |

| Versión | Fecha de lanzamiento |
|---|---|
| v2.9.0 | 2015-04-24 |
| v2.8.0 | 2015-04-10 |
| v2.7.0 | 2015-02-27 |
| v2.6.0 | 2015-02-13 |
| v2.5.0 | 2015-01-30 |
| v2.4.0 | 2015-01-23 |
| v2.3.0 | 2015-01-16 |
| v2.2.0 | 2015-01-09 |
| v2.1.0 | 2014-09-26 |
| v2.0.0 | 2014-09-13 |
| v1.4.0 | 2014-02-13 |

# Examples

### Instalación o configuración

# Instalar

`npm` se incluye con Node.js , por lo que si instala Node.js automáticamente también tendrá `npm` instalado. Puedes elegir entre una versión **actual** y una versión **LTS.**

### Windows

Para Microsoft Windows puede descargar un **instalador de MSI** desde https://nodejs.org/en/download/ .

### OS X

Para Apple OS X puede descargar un **instalador PKG** desde la misma ubicación https://nodejs.org/en/download/ .

### Linux

Para Linux puede usar su administrador de paquetes para instalar Node.js y npm.

También puede compilar Node.js desde la fuente y todavía obtendrá `npm` . También hay un script

que puede ejecutar que instalará `npm` :

```
curl -L https://www.npmjs.com/install.sh | sh
```

# Actualiza npm a la última versión

El método recomendado para actualizar su instalación `npm` es simplemente tener `npm` instalado:

```
npm install -g npm@latest
```

Alternativamente, puede actualizar a la versión actual de LTS en lugar de a la última versión:

```
npm install -g npm@lts
```

También puede instalar cualquier versión de Node (y npm) con `nvm` . Al instalar globalmente con `npm` con una instalación `nvm` , no necesita usar `sudo` (o Ejecutar como administrador en Windows).

## Instalar paquetes

Observe que se pueden instalar paquetes. Este comando instala la versión más reciente disponible de los paquetes nombrados:

Tanto a nivel local como mundial.

La instalación local significa que **npm** instala su paquete en el directorio de trabajo actual. Los módulos de nodo entran en `./node_modules` , los ejecutables entran en `./node_modules/.bin/` . Por lo general, querrá instalar módulos locales para usar dentro de su programa, como una dependencia, y funcionarán solo en el lugar donde están instalados.

```
npm install <package names>
```

Taquigrafía:

```
npm i <package names>
```

`npm` puede interactuar con un archivo `package.json` en el directorio actual de varias maneras útiles, a través de las `dependencies` objetos y `devDependencies` almacenadas en `package.json` (instalación de múltiples módulos):

El comando `npm install` sin parámetros.

```
npm install
```

instala todos los paquetes nombrados como claves de objeto en los objetos `dependencies` y `devDependencies` en `package.json` , utilizando restricciones de versiones semánticas como lo indican los valores del objeto.

Al desarrollar nuevo software:

Use la opción `-S` para agregar los `<package names>` y las versiones de los módulos npm que está instalando y que siempre deben incluirse con su módulo. Se anexa a la lista de `dependencies` rastreadas en el archivo `package.json` , después de la instalación.

```
npm i <package names> -S
```

Use la opción `-D` para agregar los `<package names>` y las versiones de los módulos npm que está instalando y que otros desarrolladores necesitan para desarrollar o probar su módulo. Se anexa a la lista de `devDependencies` rastreadas en el archivo `package.json` , después de la instalación.

```
npm i <package names> -D
```

Donde `lodash` y `mocha` son nombres de paquetes.

## Instalación de paquetes globales

Instalar un paquete global

Los paquetes instalados globalmente eliminan los módulos en `{prefix}/lib/node_modules` y ponen los archivos ejecutables en `{prefix}/bin` , donde `{prefix}` suele ser algo como `/usr/local` . Instalar un módulo global significa que sus binarios terminan en su `PATH` entorno `PATH` . Por lo general, querrá instalar un módulo global si se trata de una herramienta de línea de comandos o algo que desee usar en su shell.

```
npm install --global package-name
```

Eliminar un paquete global

```
npm uninstall --global package-name
```

Nota: el argumento `--global` se puede simplificar a `-g` . Entonces, por ejemplo, el primer comando podría haber sido `npm install -g package-name` , con el mismo resultado exacto.

Nota: en los sistemas * nix, la instalación de paquetes globales puede requerir permisos de superusuario. No hacerlo fallará con: `EACCES` . En ese caso, ejecute:

```
sudo npm install --global package-name
```

## Actualizando paquetes

En cada aplicación, el ciclo de vida llega un día en que sus componentes deben actualizarse. Todos conocen el dolor de actualizar cada dependencia individual una por una. Bueno, aquí solo necesitas emitir el comando:

```
npm update (-g)
```

Si la "-g" está ahí, npm actualizará los paquetes globales.

## Usando npm para manejar dependencias

¿Así que quieres implementar tu aplicación en múltiples sitios? ¿Y su proyecto tiene demasiadas dependencias para instalarlos uno por uno? Npm tiene una solución solo ejecute el siguiente comando:

```
npm init
```

En la carpeta raíz del proyecto, siga las instrucciones en pantalla (escriba el valor deseado y luego presione Intro) y luego, si desea guardar una dependencia, agregue:

```
--save
```

despues de ti

```
npm install
```

comandos por ejemplo:

```
npm install mypackagename --save
```

Y luego esa dependencia se guardará, entonces no tiene que mover la carpeta "node_modules". Para instalar todos los problemas de dependencia guardados:

```
npm install
```

y todas las dependencias guardadas serán instaladas.

Lea Empezando con npm en línea: https://riptutorial.com/es/npm/topic/2061/empezando-con-npm

# Capítulo 2: ¿Qué NPM hace?

## Introducción

Le permite administrar fácilmente diferentes paquetes (módulos) y hacer un seguimiento de qué versión ha instalado.

## Examples

### Inicializar un nuevo proyecto NPM

```
npm init
```

Esto iniciará un nuevo proyecto de NPM para usted, solo presione Entrar hasta que deje de hacerle preguntas.

Ahora notará que tiene un nuevo archivo llamado *package.json* . Este archivo, entre otras cosas, hará un seguimiento de los paquetes o módulos que ha instalado en su proyecto.

Lea ¿Qué NPM hace? en línea: https://riptutorial.com/es/npm/topic/10586/-que-npm-hace-

# Capítulo 3: npm scripts

## Sintaxis

- La propiedad "scripts" en `package.json` permite ejecutar paquetes npm localmente.
- El script `"karma": "karma"` referencia al script de shell `karma` el directorio `node_modules/.bin` . Esta referencia debe ser grabada y debe aplicarse un alias para que se use en otros scripts npm, como `"test": "karma start"` .

## Observaciones

# Scripts pre-reconocidos

- `prepublish` : ejecutar antes de que se publique el paquete
- `publish` , `postpublish` : Ejecutar después de que el paquete se publica
- `preinstall` : Ejecutar antes de instalar el paquete
- `install` , `postinstall` : Ejecutar después de instalar el paquete
- `version` : ejecute antes de `preversion version` del paquete
- `postversion` : Ejecutar después de golpear la versión del paquete
- `pretest` , `test` , `posttest` : `npm test` por el `npm test`
- `prestop` , `stop` , `poststop` : `poststop` por el comando `npm stop`
- `prestart` , `start` , `poststart` : `poststart` por el comando `npm start`
- `prerestart` , `restart` , `postrestart` : `postrestart` por el comando `npm restart` . Nota: `npm restart` ejecutará los scripts de detención e inicio si no se proporciona un script de `restart` .

Se puede deducir que la propiedad `"scripts"` en `package.json` es una herramienta muy poderosa. Puede usarse como una herramienta de compilación, similar a la de Grunt y Gulp, pero con más de 250,000 paquetes disponibles. Los scripts NPM ejecutan paquetes npm instalados localmente en su proyecto desde el directorio `node_modules/.bin` .

## Examples

### Ejecutando karma localmente

fragmento de `package.json`

```
{
    "scripts":
        "test": "karma start",
        "karma": "karma"
    }
}
```

### Ejecutando npm scripts

Hay dos tipos de scripts npm, y el comando para ejecutar cada uno es ligeramente diferente. El primer tipo de scripts npm son scripts "pre-reconocidos". Estos scripts son reconocidos automáticamente por npm y no necesitan un prefijo especial (como verá para el otro tipo) para ejecutarlos. El otro tipo de scripts son scripts "personalizados". Estos scripts no son reconocidos previamente por npm y tienen que estar precedidos por un comando especial para ejecutarlos. Hay una lista de scripts pre-reconocidos en la sección de comentarios.

Para ejecutar scripts pre-reconocidos:

`npm start` o `npm test`

Para ejecutar scripts personalizados necesita usar el comando de `run` :

```
npm run karma
```

## ¿Qué son los scripts npm y cómo se activan?

Los scripts npm son comandos que `npm` ejecutará cuando se le llame con los argumentos adecuados. El poder y la sensación de esto es NO instalar los paquetes npm de manera global contaminando su entorno.

La diferencia entre las secuencias de comandos pre-reconocidas y personalizadas se basa en la palabra de `run` entre las etiquetas, las **secuencias de comandos `custom` deberán `run` entre npm y el nombre de la secuencia de comandos**

En función de esto, podemos diferenciar y crear diferentes tareas o scripts para ejecutar con npm.

Dado el siguiente ejemplo en el archivo `package.json` :

```
{
  "name": "MyApp",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "mocha --recursive ./tests/",
    "test:watch": "npm run test -- -w",
    "start": "nodemon --inspect ./app.js",
    "build": "rm -rf ./dist/ && gulp build"
  }
...
}
```

Podemos ver diferentes tareas a ejecutar:

- `npm test` bien ya que es un script pre-reconocido
- `npm run test` bien ya que es una forma válida de ejecutar un script npm
- `npm run test:watch` también funcionaría, y se llama prueba de ejecución de npm dentro de sí mismo
- `npm run build`

Antes de ejecutar `gulp build` elimine la carpeta `dist` que se encuentra en el directorio (suponiendo que esté en Linux o que se reconozca el comando `rm` )

Lea npm scripts en línea: https://riptutorial.com/es/npm/topic/4842/npm-scripts

# Capítulo 4: Publicación

## Examples

### Publicar repositorio público

Para publicar repositorios públicos con una cuenta npm gratuita, inicialmente debe publicar el módulo con acceso público. Una forma de hacer esto es configurar la configuración para que `npm` lea en `packages.json` siguiente manera:

```
"publishConfig": {
  "access": "public"
},
```

También puede usar la bandera `--access=public` con el comando `npm publish` . De lo contrario, aparecerá el mensaje de error ligeramente confuso, "necesita una cuenta de pago para realizar esta acción".

```
npm publish // if you modified packages.json
```

o

```
npm publish --access=public
```

### Firmar etiqueta Git

Para firmar para trabajar necesita una clave GPG predeterminada configurada. Puede activarlo o desactivarlo de la siguiente manera:

```
npm config set sign-git-tag <true or false>
```

Lea Publicación en línea: https://riptutorial.com/es/npm/topic/3025/publicacion

# Creditos

| S. No | Capítulos | Contributors |
|---|---|---|
| 1 | Empezando con npm | acdcjunior, Ana, Aurora0001, Cezar Augusto, Community, eppsilon, GilZ, Kenan, Konstantin A. Magg, KriszDev, Paul, Protectator, RationalDev, Tim Anstee |
| 2 | ¿Qué NPM hace? | Kuhan |
| 3 | npm scripts | Alejandro Vales, DankestMemes |
| 4 | Publicación | Pier Paolo Ramon, RationalDev |