Node Package Manager. NPM.

¿Para qué sirve NPM?

Como hemos visto NPM es un gestor de paquetes, eso quiere decir que puede:

- Descargar librerías js
- Actualizar en caso de nueva versión las librerías instaladas
- Descargar una versión en específico de la librería
- Gestionar las dependencias entre paquetes

Una de las ventajas de npm es que todos los paquetes los descarga de un repositorio de paquetes llamado npmjs.

NPM usa un fichero especial llamado **package.json** en el que se declaran las librerías y sus versiones. Esto es muy útil ya que puedes tener este archivo con todas las librerías que necesites para que con un simple comando se descarguen todas y no tengas que estar buscándolas en sus respectivos repositorios.

Lo bueno es que al repositorio de librerías puedes subir las tuyas propias de tal forma que si actualizas la librería y la subes al repositorio, con un comando dentro del proyecto en el que se usa se actualiza a la nueva versión.

Cómo instalar NPM

NPM viene incorporado en NodeJS, es decir, al descargar y instalar nodejs se instalará también npm automáticamente. NodeJS se utiliza para ejecutar código javascript fuera del navegador y npm es se gestor de paquetes y librerías.

Para instalar NPM, tanto en Windows, Linux y MAC, simplemente dirígete a su página oficial: https://nodejs.org/es/download/

Una vez descargado y instalado nodejs, comprueba si se ha instalado correctamente ejecutando en una terminal (en Windows pulsa Control + R y escribe **cmd**) el comando:

```
node -v
```

Si la salida del comando es la versión de nodejs es que se ha instalado correctamente, comprueba también que tienes npm ejecutando:

```
npm -v
```

Cómo actualizar NPM

Si quieres actualizar NPM a la última versión simplemente tienes que lanzar este comando:

```
npm install -g npm
```

O si quieres actualizar NPM en MAC:

```
sudo npm install -g npm
```

También puedes actualizar NPM usando tu gestor de paquetes de la distribución de Linux si lo has instalado así.

Tutorial NPM, comandos básicos

Antes de empezar con los comandos que tiene npm, es necesario saber como funcionan el número de versión que tienen los paquetes. Los paquetes de NPM suelen seguir un versionado semántico, es decir:

MAJOR.MINOR.PATCH

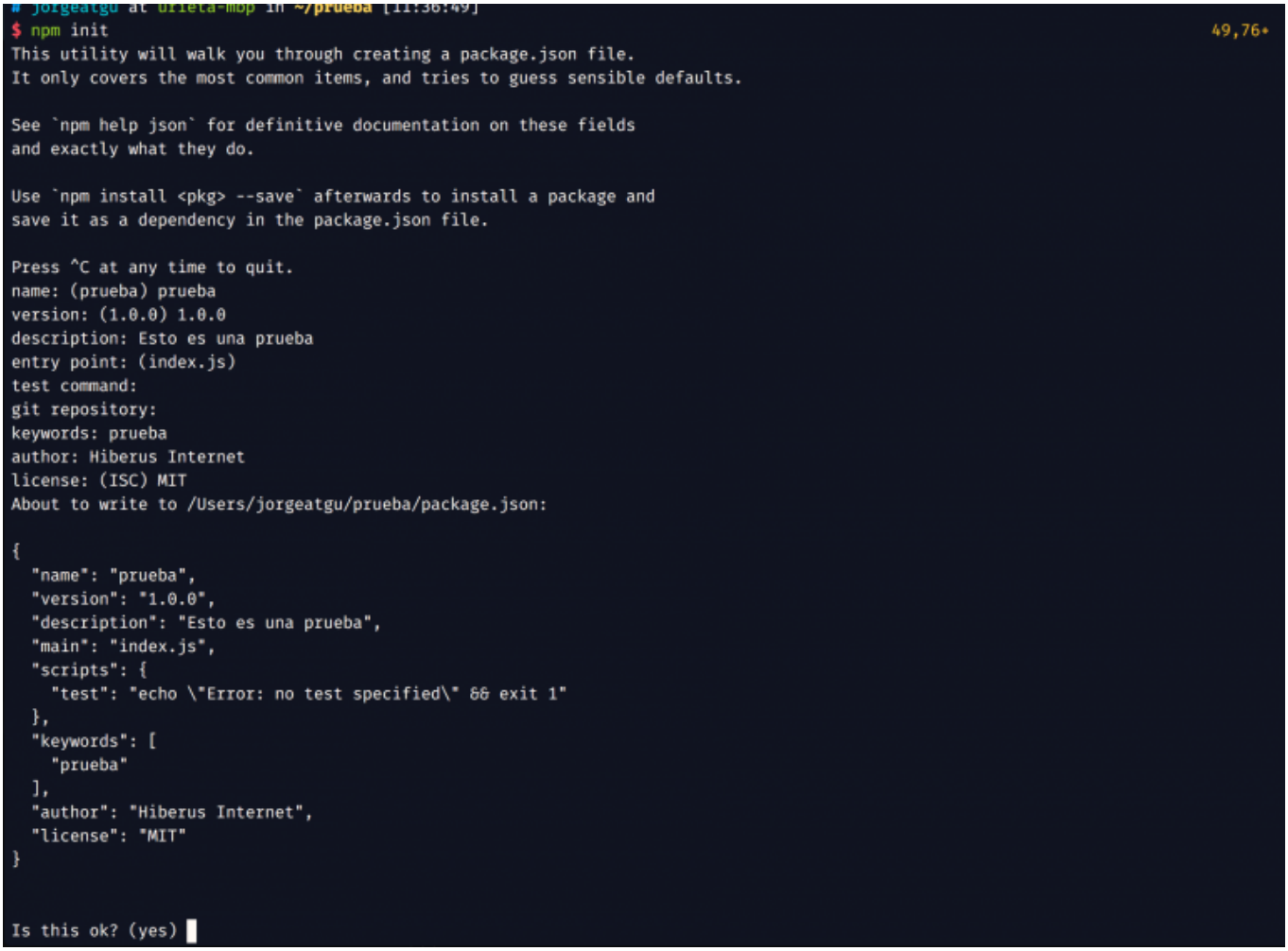
Por ejemplo dado un paquete con versión **5.4.2:**

- El primer número (5) indica una versión grande del paquete, es decir, cuando este número cambia. se supone que ha habido un cambio grande en el paquete que rompe lo que había con la versión anterior.
- El segundo número (4) indica una versión menor, es decir, en esta versión se resuelven errores anteriores y se añaden cosas pero sin romper la versión anterior, es decir, esta versión será compatible con una versión anterior y por lo tanto se puede actualizar con seguridad.
- El último número (2) indica arreglo de errores y pequeños fallos pero sin añadir funcionalidad, por lo tanto también es compatible con versiones anteriores.

NPM init. Inicialización

Este comando sirve para crear el archivo **package.json**. Te guiará y preguntará una serie de preguntas para rellenar el archivo y así no lo tengas que crear a mano

```
npm init
```



NPM install. Cómo descargar de paquetes

Este es el comando que más veces vas a usar. Como su nombre indica sirve para instalar paquetes. Los paquetes se descargarán y se meterán automáticamente en una carpeta llamada **node_modules** por lo que no te asustes si la ves en tu proyecto.

Lo más normal es que añadas esta carpeta al **.gitignore** para que no se suba al repositorio git. Este comando tiene muchas variantes:

Descargar todas las dependencias:

```
npm install
```

NPM install asecas lo que hace es leer el archivo del package.json para instalar todas las dependencias que encuentre. Si el paquete ya estaba instalado va a intentar actualizarlo.

Lo primero que se hace cuando te bajas un repositorio de git con paquetes npm es ejecutar este comando para que descargue todas las librerías, de lo contrario no funcionará.

Descargar paquete en particular:

```
npm install <nombre-paquete>
```

Por ejemplo **npm install jquery**. Descarga el paquete y lo mete en la carpeta node modules. Para buscar el nombre del paquete lo puedes hacer en el repositorio de npmjs: **https://www.npmjs.com/**

Descargar paquete y añadir versión exacta:

```
npm install <nombre-paquete> --save-exact
```

Por ejemplo `npm install dayjs –save-exact` . Este comando es igual que el anterior con la diferencia de que al guardarse en el **package.json** se guarda con la versión exacta en ese momento, es decir, nunca se va actualizar. Esto permite mantener siempre la misma versión para que todo funcione igual.

Descargar paquete global:

```
npm install -g <nombre-paquete>
```

Por ejemplo **npm install -g @angular/cli**. Esto sirve para instalar un paquete globalmente, es decir, se suele utilizar para instalar utilidades en línea de comandos para poder utilizarlas mediante línea de comandos. Por ejemplo instalando globalmente angular cli, consigues que a partir de ese momento, puedes utilizar el comando **ng** para crear

NPM uninstall. Eliminando paquetes

Lo contrario a instalar un paquete, lo elimina de la carpeta **node_modules** y de las dependencias del package.json. Por ejemplo:

```
npm uninstall jquery
```

NPM update. Cómo actualizar paquetes y dependencias

Al tener paquetes instalados, en algún momento vamos a querer actualizarlos, pues con este comando puedes hacerlo, podemos hacerlo así:

```
npm update
```

Actualiza todos los paquetes del package.json en caso de que hayan sido actualizados.

```
npm update jquery
```

Como ves arriba, también se puede actualizar paquetes en particular pasando su nombre. Como he explicado antes, npm al usar versionado semántico, puedes indicar que tipo de actualización quieres en cada paquete:

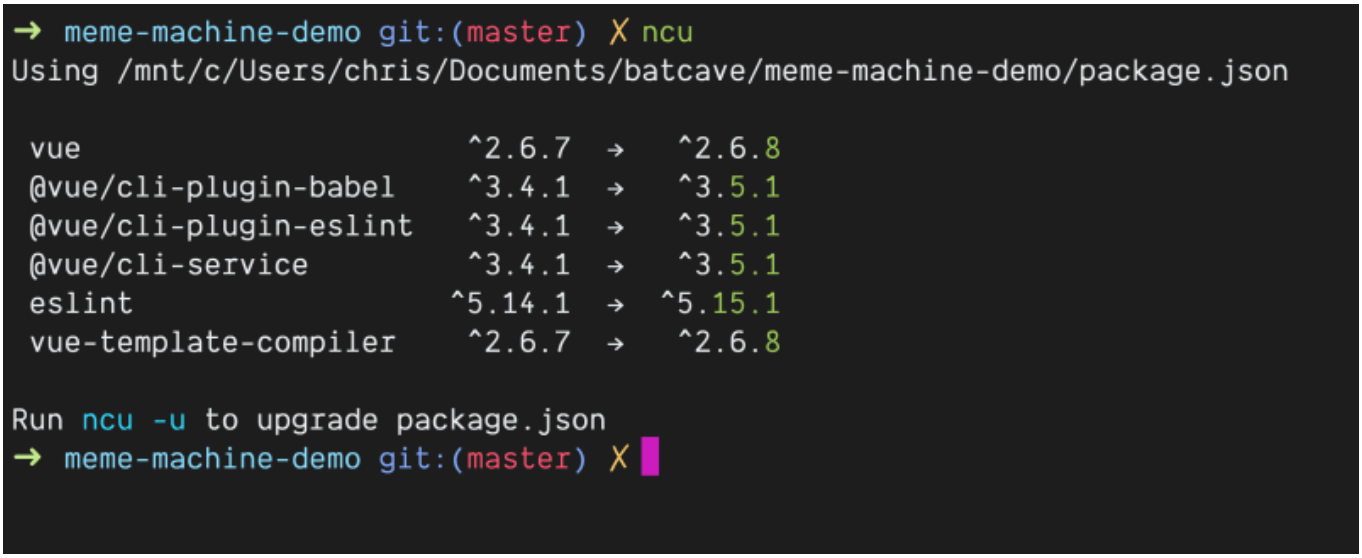
- Poniendo ~1.0.4, por ejemplo, solo va a actualizar el último número, es decir, va a subir solo de versión patch
- Poniendo ^1.0.4, por ejemplo, solo va a actualizar los dos últimos números, es decir, va a subir solo de versión minor
- Poniendo simplemente un asterisco en la versión *, va actualizar también de versión mayor.

NPM al instalar paquetes por defecto, te coloca el ^ para que actualice versiones patch y minor.

Como recomendación te aconsejo que instales este paquete para tener un listado de posibles actualizaciones de los paquetes:

```
npm i -g npm-check-updates
```

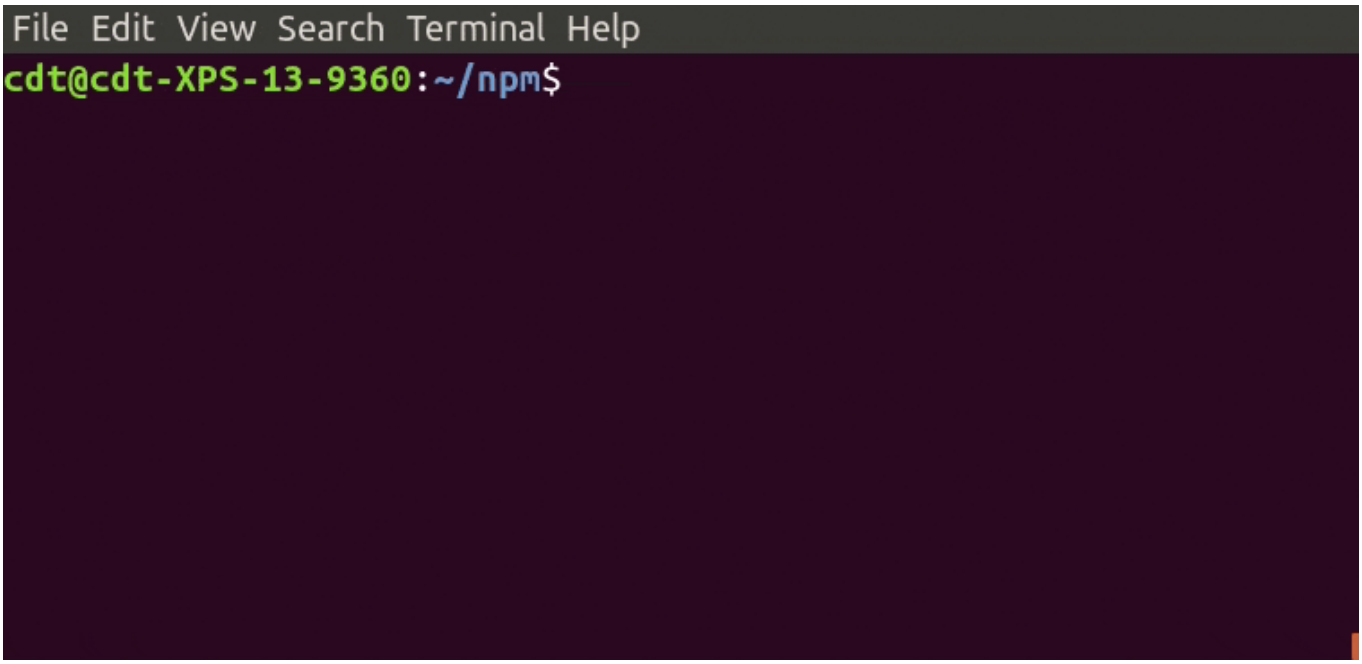
Una vez instalado, simplemente ejecuta **ncu**



NPM list. Listado de versiones

Muestra el nombre de todos los paquetes instalados en el proyecto en el que estemos situados, nos da el nombre del paquete, su versión y las dependencias que haya encontrado

```
npm list
```



Este comando tiene un parámetro especial para indicar la profundidad en la que va a buscar:

```
npm list depth <numero>
```

Es decir, si indicamos que queremos una profundidad de 0, solo va a mostrar los paquetes que hay en el package.json. Una profundidad de 1 mostrará esos paquetes más sus dependencias directas, y así sucesivamente.

NPM outdated. Paquetes desactualizados

Imprime una lista con los paquetes desactualizados. Si queremos actualizarlos podemos hacerlo como hemos visto antes con **npm update**:

```
npm outdated
```

Cómo instalar paquetes NPM desde repositorios Github o Gitlab

Si da la casualidad que el paquete que quieres instalar no se encuentra en el repositorio de paquetes de npmjs, no te preocupes porque también puedes instalar paquetes directamente desde el repositorio de github.

Esto también lo puedes usar para tener tus propias librerías localizadas en repositorios de github.

```
npm install git://github.com/usuario/repositorio.git#v0.1.0
```

Además al final puedes indicar la versión que quieres descargar (tirará de los tags del repositorio) y también puedes tirar de ramas para no tener problemas de versiones:

```
npm install git://github.com/usuario/repositorio.git#develop
```

Lo bueno es que puedes cambiar **gitlab.com** por tu servidor de git, por ejemplo gitlab o bitbucket. Además con el mismo comando puedes descargar paquetes npm de repositorios git privados.

Ejecutar tareas y comandos npm

Otro uso típico de es el de ejecutar pequeños comandos npm de una forma más cómoda. Si has usado cualquier framework para el frontend, o has trabajado con nodejs estos comandos ya los habrás usado.

Los comandos los puedes definir en el **package.json**, en la sección scripts:

```
{
    "scripts": {
        "dev": "node lib/server-development",
        "prod": "node lib/server-production"
    },
}
```

Así de sencillo, en este caso se ha creado un par de comandos para ejecutar en desarrollo o en producción. Para ejecutar los comandos simplemente se situa con la consola en el proyecto y se ejecuta el comando que quiera mediante:

```
npm run <nombre_comando>
```

Por ejemplo:

```
npm run dev
```

Puedes poner dentro de los comandos que defines cualquier código **bash** por lo que puede ser útil para automatizar ciertas tareas en el frontend sin tener que recurrir a **gulp** o a **webpack.**

Conclusiones