NPM

Comandos basicos

$ npm init
$ npm init -y

Sirve para crear el archivo package.json el cual contiene toda la información acercada nuestro proyecto

$ npm set init.author.email <email>
$ npm set init.author.name <name>

Asignar nuestros datos como en 'git', pero no es tan obligatorio hacer esto.

Instalación de dependencias

Las dependencias se deben instalar en nuestra carpeta raíz de nuestro proyecto.

$ npm i <pkg>

$ npm install <pkg> | npm install <pkg> --save (por defecto) | npm install <pkg> -S

Por defecto se instala como una independencia **requerida** para el proyecto, es decir, que paquete que instalas es necesario para vivir en producción

$ npm install <pkg> -D | npm install <pkg> --save-dev

Este flag nos va a permitir establecer que paquete que vamos a instalar solo es necesario en nuestro entorno local o el entorno de desarrollo.

$ npm install <pkg> -g

Instalar un paquete de forma global. Esto permite que podamos utilizar este paquete en diferentes proyectos. Por lo general, se deben instalar estos paquetes con permisos de administrador.

Si no queremos colocar permisos de administrador a cada rato, podemos seguir esta guía: **https://docs.npmjs.com/resolving-eacces-permissions-errors-when-installing-packages-globally**

$ npm list -g --depth 0 (profundidad)

Ver los paquetes que están instalados de forma global.

$ npm list

Para listar los paquetes que tiene un proyecto en especifico

$ npm install <pgk> -O

Podemos instalar de forma opcional un paquete con este comando.

$ npm install <pgk> --dry-run (simula la instalación)

Este flag indica que el paquete no va a ser instalado dentro del proyecto, simplemente es una simulación, nada mas nos muestra el output como si se fuese instalado

$ npm install <pgk> -f | npm install <pkg> --force

Instalar algún paquete de forma forzada. Nos va a permitir instalar este paquete forzando esa instalación a que sea desde el ultimo recursos desde el servidor de NPM

$ npm install <pgk>@<versión>

Para instalar algún paquete con una versión especifica

Actualizar y eliminar

$ npm outdate

Para **ver** si se encuentran desactualizados los paquetes y, además nos indica la ultimas versiones de esos paquetes

$ npm update

Podemos actualizar los paquetes que se encuentran desactualizados.

$ npm install <pkg>@latest

Podemos actualizar un paquete en especifico a su ultima versión con este comando.

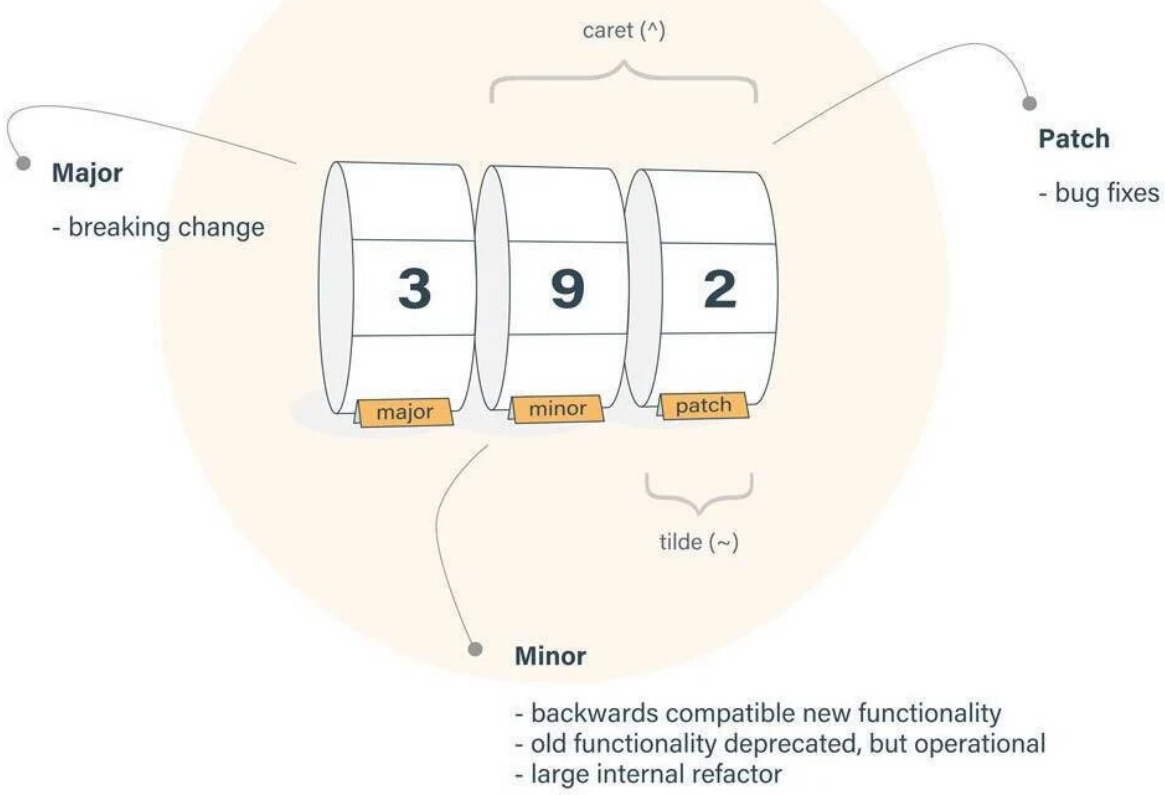
$ npm uninstall <pkg>

Para desinstalar o eliminar un paquete en especifico. Esto lo elimina del package.json y node_moduls

$ npm uninstall <pkg> --no-save

Nos permite desinstalar o eliminar un paquete, pero sin eliminarlo del package.json pero si del node_moduls

Símbolos ^ y ~



Si mantenemos el caret dentro de la configuración del package.json estamos garantizando que cuando nosotros hagamos una actualización o tengamos un cambio que podamos realizar, vamos a hacer actualización solo de los cambios menores y de los parches o bug fix de este paquete.

`$ npm install <pkg> --sabe-exact | npm install <pkg> -E`

Para instalar la dependencia en su versión exacta

Ejecutar tareas (script)

`$ npm run <script-name>`

Podemos ejecutar nuestros propios scripts con este comando.
Ver mas: https://platzi.com/login/?next=/comentario/885229/

Solución de problemas

Uno de los problemas que podemos toparnos en la construcción de nuestros proyectos, trabajando con un equipo, es que nuestros archivos de node_moduls no estén correctamente instalados o tengamos una versión anterior.

Una forma de solucionarlo es eliminar la carpeta de 'node_moduls' o ejecutar un comando que a nosotros nos va a dar seguridad de limpiar ese 'cache' que pueda llegar a existir.

`$ npm cache clean -f | npm cache clean --force`

Elimina la cache

`$ npm cache verify`

Con esto vamos a poder ver si ya la cache ha sido eliminada y que todas las

instalaciones de nuestros recursos van a ir hacia los servidores de NPM

$ npm audit
Para ver las vulnerabilidades que tenemos en nuestro proyecto.

$ npm audit --json
nos genera un json con información un poco mas detallada de lo que esta pasando con estos paquetes que instalamos.

$ npm audit fix
Para solucionar las vulnerabilidades que tengamos en nuestro proyecto. Básicamente, actualiza a la ultima versión nuestros paquetes con las dependencias que requerían

Probar nuestro paquete de npm localmente
Ya debemos tener todo el proyecto configurado, para luego ejecutar los siguientes comandos:

$ npm link
Esto nos creara un enlace simbólico en la carpeta global (.npm-global) que se vincula al paquete o proyecto donde se ejecuto el comando. El nombre de este enlace simbólico se toma el campo "name" del package.json. Lo recomendado es que el valor de "name" inicie con una @.
Eje: "name": "@boogst/<proyect-name>"

$ npm link "@boogst/<proyect-name>"
Si queremos usar el paquete anterior en algún proyecto debemos referenciarlo con este comando. Recuerda que debes estar ubicado en el proyecto "nuevo" que quieres implementar este paquete.

$ npm unlink "@boogst/<proyect-name>"
Este comando nos permite deslinkear el paquete.

publicar nuestro paquete
$ npm adduser
Para agregar un usuario de NPM

$ npm login
Para iniciar sección a un usuario

$ npm publish
Para publicar mi proyecto a NPM. Recuerda que debes estar ubicado en la raíz de tu proyecto.

$ npm unpublish -f
Para des publicar un paquete. Recuerda que debes estar ubicado en la raíz de tu proyecto.

$npm versión <major | minor | patch>

Nos permite actualizar la versión de nuestro proyecto o paquete.

Link de apoyo: https://medium.com/canariasjs/creando-componentes-en-react-y-publicando-en-npm-16eee85f9fba

NOTAS

Luis Andrés Villegas Sanchez