Que es npm?

- npm es el manejador de paquetes por defecto para Node.js.
- Es el registro de software más grande del mundo, con aproximadamente 4 mil millones de descargas por semana.
- El registro contiene más de 600.000 paquetes.

npm. Tres componentes distintos.

1. **El sitio web:** Use el sitio web para descubrir paquetes, configurar perfiles y administrar otros aspectos de su experiencia npm. Por ejemplo, para gestionar el acceso a paquetes públicos o privados.
2. **La interfaz de línea de comando (CLI):** El CLI se ejecuta desde una terminal. Así es como la mayoría de los desarrolladores interactúan con npm.
3. **El registro:** El registro es una gran base de datos pública de software JavaScript y la metainformación que lo rodea.

npm. Distintos usos

- Adapte paquetes de código a sus aplicaciones o incorpore paquetes tal como están.
- Descargue herramientas independientes que puede usar de inmediato.
- Ejecute paquetes sin descargar usando npx.
- Compartir código con cualquier usuario de npm, en cualquier lugar.
- Restringir código a desarrolladores específicos.
- Form Orgs (organizaciones) para coordinar el mantenimiento, la codificación y los desarrolladores de paquetes.
- Forme equipos virtuales usando Orgs.
- Administre múltiples versiones de código y dependencias de código.
- Actualice las aplicaciones fácilmente cuando se actualice el código subyacente.
- Descubre múltiples formas de resolver el mismo rompecabezas.
- Encuentre otros desarrolladores que estén trabajando en problemas y proyectos similares.

npm-cli. npm init

- Crea un archivo package.json, si no existe, y escribe el objeto de configuración con las opciones que ingrese el usuario:

  npm init
- Si se lo invoca con -f, --force, -y o --yes, usará sólo los valores predeterminados y no le pedirá ninguna opción.

  npm init -y

npm-cli. package.json I

name: Las cosas más importantes en su package.json son los campos de nombre y versión. Esos son realmente necesarios, y su paquete no se instalará sin ellos. El nombre y la versión juntos forman un identificador que se supone que es completamente único.

version: Para el numero de version se utiliza el versionado semántico semver.

description: Es una cadena. Esto ayuda a las personas a descubrir su paquete, tal como aparece en la búsqueda npm.

keywords: Pon palabras clave en él. Esto ayuda a las personas a descubrir su paquete tal como figura en la búsqueda npm.

homepage: La página web del proyecto.

npm-cli. package.json II

bugs: La URL del rastreador de problemas de su proyecto y / o la dirección de correo electrónico a la que se deben informar los problemas. Estos son útiles para las personas que tienen problemas con su paquete.

license: Debe especificar una licencia para su paquete para que las personas sepan cómo se les permite usarlo.

people fields: author, contributors: El "autor" es una persona. "contribuidores" es una variedad de personas. Una "persona" es un objeto con un campo "nombre" y opcionalmente "url" y "correo electrónico".

main: El campo principal es un ID de módulo que es el punto de entrada principal para su programa. Es decir, si su paquete se llama foo, y un usuario lo instala, y luego requiere ("foo"), se devolverá el objeto de exportación de su módulo principal.

npm-cli. package.json III

bin: Muchos paquetes tienen uno o más archivos ejecutables que les gustaría instalar en el PATH. npm lo hace bastante fácil (de hecho, usa esta característica para instalar el ejecutable "npm").

man: Especifique un solo archivo o un array de nombres de archivos para que el programa man pueda encontrarlos.

directories: Los detalles de la especificación CommonJS Packages de algunas maneras que puede indicar la estructura de su paquete utilizando un objeto de directorios.

repository: Especifica el sitio del repositorio.

scripts: La propiedad "scripts" es un diccionario que contiene comandos de script que se ejecutan en varias ocasiones en el ciclo de vida de su paquete.

npm-cli. package.json IV

config: El objeto config se puede usar para configurar los parámetros de configuración utilizados en scripts.

dependencies: Las dependencias se especifican en un objeto simple que asigna un nombre de paquete a un rango de versión.

devDependencies: Dependencias que se usarán durante el desarrollo y no en producción.

engines: Se puede especificar la versión a utilizar de distintos motores (nodejs, npm).

npm-cli. npm install I

- Este comando instala un paquete y cualquier paquete del que este dependa en la carpeta node_modules.

a) una carpeta que contiene un programa descrito por un archivo package.json

b) un tarball comprimido que contiene (a)

c) una url que resuelve a (b)

d) un <nombre> @ <versión> que se publica en el registro (ver npm-registry) con (c)

e) un <nombre> @ <etiqueta> (ver npm-dist-tag) que apunta a (d)

f) un <nombre> que tiene una etiqueta "más reciente" que satisface (e)

g) una <url remota de git> que se resuelve en (a)

npm-cli. npm install II

- **npm install:** Instala todos los módulos listados como dependencias en el archivo package.json.
- **npm install <paquete>:** Instala el paquete indicado en la carpeta node_modules local.
- **npm install <paquete> -g o --global:** Instala el paquete indicado en la carpeta node_modules global.
- Además de -g y --global, existen diferentes flags que se pueden usar con este comando.

npm-cli. npm install III. Flags

- **-P, --save:** El paquete se guardará en el campo *dependencies* del package.json. Es el comportamiento por defecto.
- **-D, --save-dev:** El paquete se guardará en el campo *devDependencies* del package.json.
- **-O, --save-optional:** El paquete se guardará en el campo *optionalDependencies* del package.json.
- **--no-save:** Previene el guardado del paquete.

npm-cli. npm update

- Este comando actualiza todos los paquetes enumerados en la última versión (especificada por la configuración de la etiqueta).
- También instalará los paquetes faltantes. Al igual que con todos los comandos que instalan paquetes, el indicador --dev hará que devDependencias también se procesen.
- Si se especifica el distintivo -g, este comando actualizará paquetes instalados globalmente.
- Si no se especifica ningún nombre de paquete, se actualizarán todos los paquetes en la ubicación especificada (global o local).

npm update gulp

npm-cli. npm uninstall

- Esto desinstala un paquete, eliminando por completo todo lo que npm instaló en su nombre.
- En el modo global (es decir, con -g o -global agregado al comando), desinstala el contexto del paquete actual como un paquete global.
- La desinstalación npm toma 3 indicadores exclusivos y opcionales que guardan o actualizan la versión del paquete en su paquete principal. json:
  - **-S, --save:** El paquete se eliminará de tus dependencias.
  - **-D, --save-dev:** El paquete se eliminará de tus devDependencies.
  - **-O, --save-optional:** El paquete se eliminará de sus OptionalDependencies.
  - **--no-save:** el paquete no se eliminará de su archivo package.json.

npm-cli. npm docs

- Este comando intenta adivinar la ubicación probable de la URL de la documentación de un paquete y luego intenta abrirla usando el parámetro de configuración --browser.
- Puede pasar múltiples nombres de paquetes a la vez.
- Si no se proporciona un nombre de paquete, buscará un package.json en la carpeta actual y usará la propiedad del nombre.

npm docs gulp gulp-concat

npm-cli. npm ls

- Este comando se imprimirá para extender todas las versiones de los paquetes que están instalados, así como sus dependencias, es una estructura de árbol.
- Imprimirá paquetes extraños, faltantes e inválidos.
- El árbol que se muestra es el árbol de dependencia lógica, basado en las dependencias del paquete, no en el diseño físico de la carpeta node_modules.
- Existen diferentes flag para este comando.

npm-cli. npm ls --flags

- **npm ls --json:** Muestra la información en formato JSON.
- **npm ls --long:** Muestra la información de manera extendida.
- **npm ls --global:** Muestra las dependencias globales.
- **npm ls --depth=0:** Muestra la profundidad máxima de visualización del árbol de dependencias.
- **npm ls --prod:** Mostrar sólo el árbol de dependencias para paquetes en dependencies
- **npm ls --dev:** Mostrar sólo el árbol de dependencias para paquetes en devDependencies

npm-cli. npm config

- Administra los archivos de configuración de npm.
- **npm config set key value:** Configura la clave con el valor proporcionado, si este es omitido se configura el booleano true.
- **npm config get key:** Obtiene el valor de la clave especificada.
- **npm config list:** Muestra todos los valores de configuración. Usar -l para mostrar los valores predeterminados o --json para mostrar en formato JSON.
- **npm config delete key:** Borra la clave especificada.

package.json. scripts I

npm admite la propiedad "scripts" del archivo package.json para las siguientes secuencias de comandos:

- prepublish
- prepare
- prepublishOnly
- prepack
- postpack
- publish
- preinstall
- install
- postinstall
- preuninstall
- postuninstall
- preversion
- version
- postversion
- test
- stop
- start
- restart
- preshrinkwrap
- shrinkwrap
- postshrinkwrap

package.json. scripts II

Además, se puede ejecutar cualquier comando que no esté en la lista anterior con la instrucción *run*.

npm run miComando

npm start

npm test

package.json. scripts III

Los scripts se ejecutan en un entorno en el que hay mucha información disponible sobre la configuración de npm y el estado actual del proceso.

Los parámetros de configuración se ponen en el entorno con el prefijo npm_config_. Por ejemplo, puede ver la configuración de root comprobando la variable de entorno npm_config_root.

process.env.npm_package_root

package.json. scripts IV

Los scripts también se pueden ejecutar pasandole argumentos.

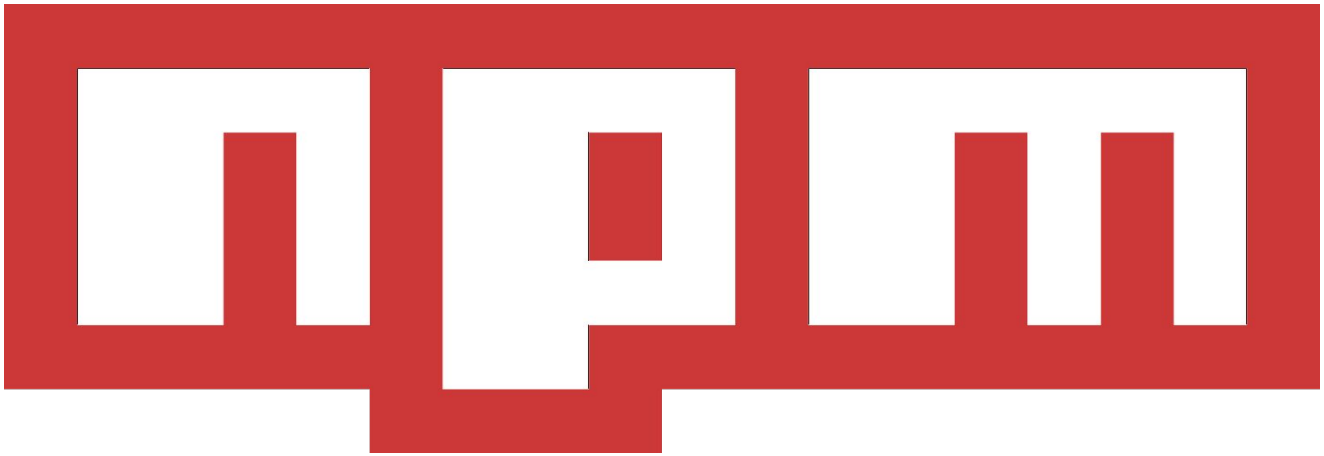
npm run miScript --miArg=arg1

Y lo tendre visible en el objeto **process.env.npm_config_miArg**

o

npm run miScript --miArg arg1

Y lo tendre disponible en el array **process.arg**



PREGUNTAS