Un primer proyecto en NodeJS

Teoría

Ya sabemos cómo ejecutar un archivo con código JavaScript dentro, utilizando el comando "*node [nombreDel archivo.js]*". Veamos ahora cómo hacer cuando nuestro programa ya no entra en un único archivo suelto.

El archivo *"package.json"*

Los proyectos de NodeJS cuentan con un **archivo de configuración** en formato JSON que, entre otras cosas, permite especificar el nombre del proyecto, versión, nombre del autor, etc.

Al instalar NodeJS, éste trae incluido un utilitario llamado NPM (Node Package Manager, el cual veremos en mayor profundidad más adelante). NPM cuenta con un comando que, a través de un cuestionario con una serie de preguntas, nos ayuda a generar la inicialización de nuestro programa o sistema. Al ejecutarlo desde una consola en la carpeta del proyecto generará allí automáticamente el archivo con los datos requeridos. Si aún no hemos creado una carpeta para nuestro proyecto, comencemos por ahí. Podemos hacerlo desde los menúes contextuales del sistema operativo, o mediante el comando por consola 'mkdir' seguido del nombre de la carpeta por crear. Una vez creada, debemos posicionarnos en esa ruta y ejecutar el comando desde la terminal.

Ejemplo desde la terminal:

```
$ mkdir miProyecto
$ cd miProyecto
$ npm init
```

Se nos pedirá la siguiente información:

- package name (nombre del paquete / proyecto) *// no puede contener espacios!*
- version (versión)
- description (descripción)
- entry point (archivo que será usado como punto de entrada / main)
- test command (en caso de haber alguno, comando con el cual se ejecutarán los tests)
- git repository (link al repositorio de git donde se almacenará el versionado)
- keywords (palabras clave con que se podrá encontrar tu proyecto una vez publicado)
- author (autor)
- license (tipo de licencia, en caso de tenerla)

Finalmente, generará un archivo con el siguiente formato JSON:

```
{
  "name": "mi-proyecto",
  "version": "1.0.0",
  "description": "este es mi proyecto",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "maquino",
  "license": "ISC",
  "keywords": []
}
```

Si se lo desea, se puede ejecutar el comando con la opción “`--yes`” (o su versión corta, "-y") para evitar tener que llenar cada campo, uno por uno, y generar un archivo con todos valores por defecto:

```
$ npm init -y
```

El nombre por defecto para el proyecto se copia del nombre del directorio donde se encuentra el mismo. Si este nombre contiene espacios, la generación automática no podrá realizarse y lanzará un error, al no poder inferir un nombre válido.

Configuración adicional

A partir de la versión 13 de NodeJS, se ha agregado una nueva actualización que permite utilizar una nueva sintaxis para importar y exportar módulos, que se asemeja a la utilizada en otros lenguajes de programación. Para poder habilitar esta nueva característica (de momento, experimental), deberemos agregar al documento un nuevo par *"clave": "valor"* al comienzo de la lista de pares incluidos por defecto, en el que definamos que nuestro proyecto será de tipo "módulo":

```
{
  "type": "module",
  "name": "mi-proyecto",
  "version": "1.0.0",
  ...
}
```

Otras observaciones

Con el objetivo de mantener los archivos de nuestro proyecto ordenado, existen algunas convenciones. Entre ellas, cuidaremos de tener nuestros archivos de configuración en el directorio raíz del proyecto, y de crear una carpeta aparte para los archivos con el código fuente de nuestro programa, de nombre "src".

Puntos de inicio del proyecto

Dentro de las numerosas opciones de configuración que podemos detallar en el package.json, encontraremos un objeto ‘scripts’ que nos permitirá definir comandos personalizados para ejecutar el proyecto desde distintos puntos de entrada. Veamos el siguiente ejemplo:

```
{
  ...
  "scripts": {
    "start": "node src/main.js",
    "test": "node test/testAll.js"
  }
}
```

En este caso, encontramos definidos dos puntos de inicio para nuestro programa: uno que (probablemente) iniciará el servidor, y otro que ejecutará los tests del sistema.
Para ejecutar el sistema utilizando alguno de esos puntos de entrada definidos, se usará el comando ***npm xxxx***, en donde ***xxxx*** es el nombre que se le dió al script (en este caso, ‘start’ o ‘test’).

```
$ npm start
$ npm test
```

Npm cuenta con un set de comandos predefinidos que pueden ser usados como nombres de scripts (entre ellos, start y test). Si se desea crear otros nombres de scripts más personalizados, también es posible.

```
{
  ...
  "scripts": {
    "comenzar": "node src/main.js",
    "pruebas": "node test/testAll.js"
  }
```

```
}
```

La única diferencia será que para poder ejecutarlos desde la consola, se deberá anteponer la palabra “run” al nombre del script. Ejemplo:

```
$ npm run comenzar
$ npm run pruebas
```