Introducción a node y typescript

Alejandro Mármol Muñoz

Node.js es un entorno de tiempo de ejecución de JavaScript (de ahí su terminación en .js haciendo alusión al lenguaje JavaScript). Este entorno incluye todo lo que se necesita para ejecutar un programa escrito en JavaScript. JavaScript solo podía ejecutarse en el navegador, y como respuesta a ese problema se creó node, que se podría ejecutar en los ordenadores como si de aplicaciones independientes se tratara.

En Node.JS el código se encuentra estructurado por módulos, que a medida que vamos trabajando iremos necesitando agregar más, por lo que además también usaremos el NPM (Node Package Manager). **NPM** Es un gestor de paquetes desarrollado en su totalidad bajo el lenguaje JavaScript, a través del cual podemos obtener cualquier librería con tan solo una línea de código

Programaremos node en replit, dada sus múltiples ventajas. **Replit** es un entorno de desarrollo online para programar desde el navegador. No requiere instalar ningún programa y puede acceder a la nube, por lo que es muy cómodo y versatil.

En vez de programar con JavaScript, usaremos TypeScript. **TypeScript** es un lenguaje de programación evolucionado de JavaScript, que soluciona numerosos problemas de su predecesor. Por último, usaremos JSON para guardar la información. **Json** es un formato que almacena información estructurada y se utiliza principalmente para transferir datos entre un servidor y un cliente. El archivo es básicamente una alternativa más simple y liviana al XML.

Procedemos a crear nuestro programa en typescript compilado a javascript, mediante el entorno de desarrollo replit que nos permite acceder a la nube y al gestor de paquetes npm, y guardando la información en formato json:

1º- Accedemos a Replit ➡ https://replit.com/

Una vez registrados con nuestro correo, podremos proceder a crear el proyecto.

Cliqueamos en New repl y seleccionamos el lenguaje Node e introducimos el nombre del repositorio, en este caso Ej1-Introducción.

2º- Creamos la estructura del proyecto.

Se creará un index.js, el archivo inicial ejecutable. Crearemos una estructura de archivos óptima para el desarrollo:

- Se crea el directorio src, y dentro un archivo typescript (<nombre>.tp) que contendrá el código del programa.
- Un archivo llamado .replit que contendrá “run = "npm start"”, que nos permitirá utilizar al gestor de paquetes antes mencionado.

3º- Paquete json

Procedemos a obtener el paquete json. Cliqueamos a la sección “packages” y buscamos “json”. Se acoplará automáticamente dos archivos : package.json y package-lock.json. El primer es es un fichero que contiene información acerca de tu proyecto (nombre, versión, etc) además de listar los paquetes de los que depende. El segundo es un corrector de incompatibiliad de versiones(nombre que mismo me le he apodado) : si alguien clonase un repositorio y este tiene una versión distinta a la que se usó al programarlo, este paquete instala la versión correspondiente, evitando así numerosos errores.

Luego, creamos un archivo “tsconfig.json”. Este indicará al proyecto que se está trabajando con TypeScript. Lo colocas en la raíz de carpetas del proyecto y en él situamos un JSON con todas las configuraciones de trabajo para el transpilador de TypeScript:

"target": "es5"

“outDir” : “`./dist`”

“rootDir” : “src”

target: indicamos que queremos que compile a código Javascript escrito con el estándar ES5.

outDir: indicamos el directorio donde se van a colocar los archivos Javascript, ".js", una vez transpilados.

rootDir: indica dónde están los archivos fuente, con el código TypeScript, ".ts", que debe ser traducido.

Cuando se haya compilado el archivo ts en la ruta rootDir, se guardará el nuevo js en una carpeta llamada Dist, como hemos escrito en el outDir.

Por último, en los scripts del paquete.json, introducimos:

"ts": "tsc -w",

“dev": "nodemon ./dist/index.js",

"start": "node ./dist/index.js"

En estos script declarámos qué archivo ejecutará, sustituyendo el main predefinido por el que se encuentra en la carpeta dist, que contiene el codigo compilado.

4º- Programar

Ya lo tenemos todo listo para programar. En este caso, como ejemplo, crearemos un bucle sencillo que dice los números del 1 al 99 que sean primos:

```
for (var x = 1; x < 100; x++)
{numeroPrimo(x);}

function numeroPrimo(num){
let confirmacion = false;

for (var x = 2; x < num; x++) {
if (num % x == 0 && confirmacion == false) {
console.log("El numero " + num + " no es primo");
confirmacion = true;
}
}
if (confirmacion==false){console.log("El numero "+num+" es primo")}
}
```

Una vez creado el programa, escribimos el comando tsc -w para que se compile el archivo y le damos a run para ejecutar el programa.