3. Desarrollo de servicios con Node.js

Anexo I. Más ejercicios de introducción con Node.js

Programación de Servicios y Procesos

Arturo Bernal
Nacho Iborra
Javier Carrasco

Tabla de contenido

1. Más sobre *lodash*

Como has visto, *"lodash"* es una biblioteca de Node muy popular, y millones de personas la utilizan a diario en sus desarrollos. Lo acabas de probar en un ejercicio realmente simple antes, durante esta sesión. Se explorarán algunas otras características de esta biblioteca en este nuevo desafío.

Ejercicio 1

Cree un proyecto llamado ***"Exercise_Lodash"*** en tu espacio de trabajo. Se probarán algunos métodos útiles de la API para ver cómo funcionan.

Comienza instalando *"lodash"* en este proyecto (recuerda usar *npm init* para crear el archivo *"package.json"* y después usar *npm install*). Después, añade y prueba estos fragmentos de código:

- En primer lugar, define una variable para almacenar una lista de personas con sus atributos, como por ejemplo:

```
let people = [
    {firstName: "Nacho", lastName: "Iborra", age: 39},
    {firstName: "Juan", lastName:"Perez", age: 55},
    {firstName: "Mario", lastName:"Calle", age: 4},
    {firstName: "Ana", lastName:"Jiménez", age: 33},
    {firstName: "Yeray", lastName:"Perez", age: 2},
    {firstName: "Javier", lastName:"Carrasco", age: 44}
];
```

- Después, utiliza la función *map* de *Lodash* para obtener un vector con solo los nombres de cada persona. Imprime este vector en la consola (*console.log*).
- A continuación, intenta combinar las funciones *map*, *orderBy* y *takeWhile* para mostrar solo los nombres de las personas mayores de 18 años, en orden de edad descendente.
- Finalmente, combina las funciones *map* y *sum* para calcular el promedio de las edades en la lista anterior.

También puede consultar algunas otras funciones útiles de esta biblioteca, como generadores de números aleatorios, utilidades matemáticas, etc.

2. Manejo de fechas: *moment*

"moment" es otra biblioteca popular en el repositorio de NPM. Se utiliza para tratar con fechas: crearlas con un formato de entrada determinado, comparar fechas, etc. Puedes consultar la API de esta biblioteca.

Ejercicio 2

Cree un ejercicio llamado ***"Exercise_ Moment"*** en tu espacio de trabajo. En este ejercicio se crearán algunas fechas, se comprobarán si son válidas según un patrón dado y se harán algunas comparaciones entre ellas. Sigue estos pasos:

- En primer lugar, como de costumbre, instala la biblioteca *"moment"* escribiendo los comandos correspondientes (*npm init* y *npm install*).
- Después, crea la fecha actual con esta instrucción:

```
let now = new moment();
```

- Ahora, define dos fechas en dos variables separadas, con el formato español típico (DD/MM/AAAA). Para hacer esto, utiliza el siguiente constructor:

```
let date1String = "07/03/2013";
let date1 = new moment(date1String, "dd/MM/YYYY");
```

- Utiliza el método *isValid()* para comprobar si son válidos o no. Muestra el resultado en la consola. Después, intenta con una fecha no válida (es decir, una fecha que no siga el patrón anterior, como "07-marzo-2011").
- A continuación, compara ambas fechas válidas entre ellas. Echa un vistazo al método *diff* para ver cómo funciona. Luego, imprime la diferencia en días y en años entre ambas fechas.
- Finalmente, inténtalo con una fecha y hora completas. Defina una variable con una fecha y hora algunos minutos después de ahora, y utiliza el método *fromNow()* para determinar cuántos minutos existen entre ahora y su fecha y hora.

Siéntate libre de explorar otras características de esta biblioteca, como permitir múltiples patrones de entrada o sumar/restar tiempo a una fecha determinada para crear otra.

3. Un módulo global útil: *nodemon*

Para terminar con este anexo, se instalará un módulo global (revisa el contenido de la Parte I del tema para ver cómo instalar módulos globales con el comando *npm*). Puedes encontrar una amplia variedad de módulos globales para probar, pero este te resultará especialmente útil. Es el módulo *"nodemon"*.

Nodemon es un monitor que re-ejecuta automáticamente aplicaciones Node cuando se realiza (y guardes) cualquier cambio en el código. A veces es un dolor de cabeza volver a ejecutar manualmente las aplicaciones, especialmente cuando estás probando muchos cambios pequeños. Con *nodemon* no tienes que preocuparse por eso, simplemente escribe los cambios y el programa se reiniciará automáticamente.

Una vez instalado, puedes utilizarlo en cualquier aplicación con solo escribir *Nodemon* en lugar del comando *node* para ejecutarlo.

Ejercicio 3

Una vez que hayas instalado *"nodemon"* globalmente en tu sistema, intenta ejecutar con él el primer ejercicio de este anexo (en lugar de escribir *node file.js*, utiliza *nodemon file.js*).

Después, intenta cambiar algunos valores en la colección de personas y guarda los cambios. Verás cómo todas las pruebas que realizas sobre la colección se vuelven a ejecutar automáticamente.