Funciones síncronas, asíncronas y callbacks

TSR 2018-19 Francisco Torres

Lo que ya sabemos

- Una función **síncrona** se ejecuta cuando se invoca
- Ej `readFileSync` lee un fichero de forma síncrona, `writeFileSync` escribe fichero de forma síncrona:

```
var fs = require('fs');
fs.writeFileSync('mydata.txt', 'Hello Node\nHello boys!')
var buffer = fs.readFileSync('mydata.txt')
console.log(buffer.toString())
console.log('ejecutando otras instrucciones')
console.log('raiz(2) =', Math.sqrt(2));
```

```
Hello Node
Hello boys!
ejecutando otras instrucciones
raiz(2) = 1.4142135623730951
```

Ejecución asíncrona

- La variante **asíncrona** ejecuta la operación en 2º plano (no bloquea al resto del programa)
- Muchas funciones predefinidas en Node tienen comportamiento asíncrono (ej `readFile`)

```
var fs = require('fs');
fs.writeFileSync('mydata.txt', 'Hello Node\nHello boys!')
fs.readFile('mydata.txt') // ???? como se invoca
console.log('ejecutando otras instrucciones')
console.log('raiz(2) =', Math.sqrt(2))
```

```
ejecutando otras instrucciones
raiz(2) = 1.4142135623730951
```

Invocación función asíncrona

- Debe proporcionar como último argumento una función **callback**
- Compara las funciones de lectura de ficheros:
  - Síncrona: `fs.readFileSync(path[, options])`
  - Asíncrona: `fs.readFile(path[, options], callback)`
- `path` = ruta del fichero a leer
- `options` = opciones sobre el modo de lectura
- `callback` = función a invocar cuando termine la ejecución de la función asíncrona

Invocación función asíncrona ...

- Añadimos una función callback al ejemplo:

```
var fs = require('fs');
fs.writeFileSync('mydata.txt', 'Hello Node\nHello boys!')
fs.readFile('mydata.txt', ()=>{console.log("leido")})
console.log('ejecutando otras instrucciones')
console.log('raiz(2) =', Math.sqrt(2))
```

```
ejecutando otras instrucciones
raiz(2) = 1.4142135623730951
leido
```

Invocación función asíncrona ...

- Pero deseamos mostrar el contenido del fichero, no un mensaje `leido`
- La función callback necesita 2 argumentos `(err,data)`:
  - Si la función asíncrona se ejecuta sin errores, `data` contiene su resultado
  - Si hay un error en la ejecución de la función síncrona, `err` contiene info. sobre el error

Invocación función asíncrona ...

- Ejecución sin errores

```
var fs = require('fs');
var myCB = function (err, data) {
   if (err) console.error(err.stack)
   else console.log(data.toString())
}
fs.writeFileSync('mydata.txt', 'Hello Node\nHello boys!')
fs.readFile('mydata.txt', myCB)
console.log('ejecutando otras instrucciones')
console.log('raiz(2) =', Math.sqrt(2))
```

```
ejecutando otras instrucciones
raiz(2) = 1.4142135623730951
Hello Node
Hello boys!
```

Invocación función asíncrona ...

- Cuando intenta leer fichero inexistente

```
var fs = require('fs');
var myCB = function (err, data) {
   if (err) console.error(err.stack)
   else console.log(data.toString())
}
fs.readFile('none', myCB)
console.log('ejecutando otras instrucciones')
console.log('raiz(2) =', Math.sqrt(2))
```

```
ejecutando otras instrucciones
raiz(2) = 1.4142135623730951
Error: ENOENT: no such file or directory, open '...\none'
    at Error (native)
```

Otro ejemplo

- Lee y procesa números desde fichero

```
var fs = require('fs')
fs.writeFileSync('mydata.txt', '1 2 3 4 5 6 7 8')
var doble = (str) => str.split(" ").map(n => 2*n)
var f = function (err, data) {
   if (err) console.error(err.stack)
   else console.log(doble(data.toString()))
}
fs.readFile('mydata.txt', f)
console.log('ejecutando otras instrucciones')
console.log('raiz(2) =', Math.sqrt(2))
```

```
ejecutando otras instrucciones
raiz(2) = 1.4142135623730951
[ 2, 4, 6, 8, 10, 12, 14, 16 ]
```