Eventos en NodeJS

TSR 2021 Juansa Sendra, Grupo B

Eventos

- La práctica totalidad del API de nodeJS utiliza una arquitectura asincrónica dirigida por eventos
  - Un emisor puede generar distintos tipos de eventos, identificados mediante un nombre (string)
    - Ej: un stream emite 'close', 'drain', 'error', 'finish', 'pipe', 'unpipe'
    - Ej: un socket emite 'close', 'connection', 'error', 'listening'
  - Podemos asociar una o más acciones a cada evento
    - cada acción es un callback y representa actividades pendientes
    - establece un orden: mantiene un vector de acciones por evento, y se activan por orden
- En nuestros programas podemos tratar esos eventos, pero también declarar emisores de nuevos eventos
  - Para poder definir emisores de eventos necesitamos el módulo `events`

Eventos.- Estructura aplicación

- Si todos los eventos son externos (API)
  - Ej.- eventos de teclado, a través de sockets, etc.
  - asociar accions a los eventos
- Si creamos eventos internos (desde el propio programa)
  - importar la biblioteca `events`
  - crear un emisor (o varios)
  - asociar acciones a los eventos
  - emitir eventos
    - Es habitual utilizar `setTimeout` o `setInterval`
    - `setTimeout(f,ms) // programa f para ejecutarse tras ms milisegundos`
    - `setInterval(f,ms) // repite la ejecución de f cada ms milisegundos`

Eventos.- Emisor

- Todos los emisores son instancias de la clase events.EventEmitter
- Para emitir un evento utiliza `emisor.emit('nombreEvento', arg, arg,..)`
    - tantos argumentos como necesitemos (posiblemente ninguno)
- En caso de error interno el emisor genera el evento 'error'

Eventos.- Acciones

- `emisor.on('nombreEvento', callback)` asocia el callback al evento
  - Los argumentos de `emisor.emit('..', args) son los parámetros del callback
  - Cada vez que llega el evento activa la acción
- `emisor.once('nombreEvento', callback)` asocia el callback al evento
  - activa sólo una vez, y dá de baja la acción
- podemos asociar varios callbacks al mismo evento, y ante la llegada del evento se activan en el orden en que se han registrado
- Si no tenemos ninguna acción asociada al evento `emisor.on('error',..)` y llega ese evento, el programa aborta
- Una acción activada se ejecuta inmediatamente (si no estemos ejecutando otra cosa), o se anota en la lista de eventos

Eventos.- Example

```
const ev = require('events')
const emisor = new ev.EventEmitter
emisor.on('event', (i)=>{
        console.log('Llega el evento '+i)
})
emisor.emit('event',1)
emisor.emit('event',2)
```