Introducción a NodeJS REPL

(node shell)

TSR 2018-19 Juansa Sendra

Para más información sobre REPL

REPL *(Read Eval Print Loop)*

Entorno interactivo (node shell) para:

- Desarrollar y probar código de forma ágil
  - *Resolver dudas sobre sintaxis/funcionamiento de js*
  - *Ejecuta código escrito en cualquier editor*
- Depurar fragmentos de código
  - *Escribe código en editor, cárgalo en REPL, y trabaja de forma interactiva*

```
Para seguir este texto necesitas un editor y una terminal
```

REPL

Desde la consola escribe node

- el prompt cambia a `>`
- podemos evaluar cualquier expresión `> 23 + 32`
- el sistema lee la expresión (Read), la evalua (Eval), e imprime el resultado (Print), tras lo cual vuelve al principio (Loop)

```
55
>
```

- Para salir pulsa CTRL-C dos veces consecutivas

REPL

```
> 23 + 32
55
> "hola, " + "lector"
"hola, lector"
> let x = 10, y = 20
undefined
>x+y
30
> "otorrinolaringologo".length
19
> [4,2,8,12].includes(6)
false
>
```

REPL

Si la expresión a evaluar ocupa varias líneas:

- En cada línea nueva aparece `...` (indica que continuamos con la expresión)
- Vuelve al modo normal cuando finaliza la definición o expresión actual, se ejecuta `.break`

```
> function suma(a,b) {
... return a+b
... }
undefined
> suma(10,30)
40
>
```

Ejecucion fichero externo

- Edita el fichero `f.js` con cualquier editor
- Ejecuta en al consola `node f.js`

```
// fichero fact.js
function fact(n) {
   if (n==1) return 1
   else {
      return n*fact(n-1)
   }
}
console.log(fact(5))
```

```
node fact.js
120
```

Depuración con REPL

-Ejecuta `.load f.js` desde REPL

- carga y ejecuta el fichero, y se mantiene en el REPL
- puedes probar funciones, inspeccionar variables, etc.

Depuración con REPL

```
> .load fact.js
// fichero fact.js
function fact(n) {
   if (n==1) return 1
      else {
            return n*fact(n-1)
               }
               }
               console.log(fact(5))
120
undefined
> fact(0)
RangeError: Maximum call stack size exceeded
    at fact (repl:2:14)
    at fact (repl:5:22)
    ...
```

Dot commands

- Son órdenes especiales que empiezan por `.`
- Las más útiles son
  - `.help` explica las distintas órdenes dot
  - `.exit` sale de REPL
  - `.editor` para escribir expresiones multilínea
  - `.break` sale del modo editor
  - `.save fichero` guarda en ese fichero la sesión actual del trabajo (todo lo escrito en REPL)
  - `.load fichero` carga el fichero en la sesión actual

También te interesa saber ...

- Las teclas `arriba/abajo` permiten recorrer la historia de los comandos anteriores
- `tab` autocompleta expresiones u órdenes
  - Si no hay una finalización unica, otro `tab` muestra las alternativas
  - Si no escribimos nada, `tab+tab` muestra los módulos que node exporta por defecto
  - Si x es el nombre de un modulo/objeto, `x.` `tab+tab`, muestra sus funciones/propiedades
- `_` representa el resultado de la última expresión