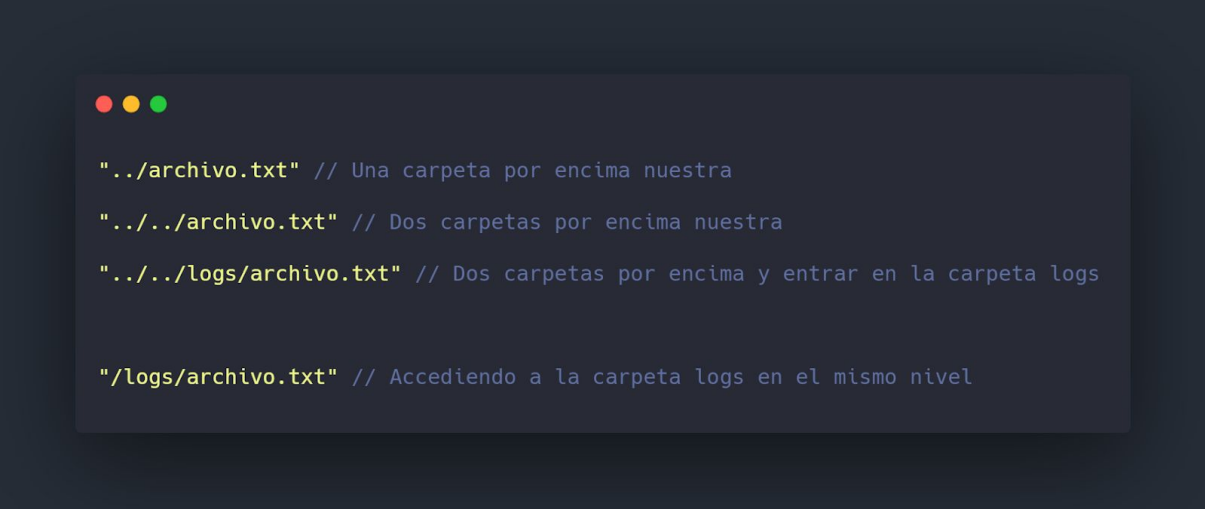


## Utilidades 3

### Enrutamiento

Cómo funciona el enrutamiento en JavaScript... En ocasiones queremos leer un archivo que no está estrictamente en nuestra ubicación, ya puede estar por encima en el sistema de archivos o por debajo

si queremos acceder a un archivo que está por encima de nuestra carpeta debemos usar los puntos suspensivos y si esta por debajo solo tenemos que

A dark-themed terminal window with three colored window control buttons (red, yellow, green) in the top-left corner. It contains four lines of text, each representing a file path with a comment in Spanish. The text is as follows:

```
../archivo.txt" // Una carpeta por encima nuestra
../../archivo.txt" // Dos carpetas por encima nuestra
../../logs/archivo.txt" // Dos carpetas por encima y entrar en la carpeta logs

/logs/archivo.txt" // Accediendo a la carpeta logs en el mismo nivel
```

### Module exports

Según programamos nuestro archivo crece y algo realmente útil sería poder separar nuestras funciones o clases en múltiples archivos, para mantener el orden.. Aquí es donde entra `module.exports`. Creemos unas funciones en un archivo `utils.js` y requiramos este archivo desde otros sitios



```
//utils.js
```

```
//Chec if number is prime
```

```
function isPrime(num) {  
    for (var i = 2; i < num; i++) {  
        if (num % i === 0) {  
            return false;  
        }  
    }  
    return num > 1;  
}
```


```
//Check if number is odd
```

```
function isOdd(num) {  
    return num % 2;  
}
```

```
module.exports = {  
    isPrime,  
    isOdd
```

```
}
```

Para usarlo desde otro archivo...tenemos que requerir al archivo desde una variable y usarlo desde la siguiente forma.



```
//index.js

const utils = require('./utils.js');

let elevenIsPrime = utils.isPrimer(11)

console.log(elevenIsPrime)
```

## Concatenación

Hay distintas formas de hacer concatenaciones.... veamos las dos principales formas.



```
const edad = 28;

const nickName = "GiR";

console.log('La edad de ' + nickName + ' es de ' + edad);

console.log(`La edad de ${nickName} es de ${edad}`);
```


## Promesas

Las promesas tienen dos formas de poder ser utilizadas... tenemos la forma "then"



```
axios.get('/user', {  
  params: {  
    ID: 12345  
  }  
})  
  
.then(function (response) {  
  // Se ejecuta cuando todo fue bien  
  console.log(response);  
})  
  
.catch(function (error) {  
  // Se ejecuta cuando hay un error  
  console.log(error);  
})  
  
.then(function () {  
  // Siempre se ejecuta  
});
```

y tenemos la forma de acortarlo... que es con un `await`. Y además almacenamos su valor en un variable.



```
const response = await axios.get('/user?ID=12345');
```

## **Foreach**

Un `foreach` es una forma rápida y sencilla de recorrer un array, no tenemos que iniciar un índice ... ni nada de lo que hacíamos anteriormente, veamos un ejemplo.

Mismo resultado, distinta sintaxis.



```
const numeros = [1,2,3,4,5,6,7,8,9,0]
```

```
for(let i = 0 ; i < numeros.length;i++){  
    console.log(numeros[i]);  
}
```

```
numeros.forEach(function(numero){  
    console.log(numero)  
})
```