Node.pdf

Pops_B

Aplicaciones Web

4º Grado en Ingeniería del Software

Facultad de Informática
Universidad Complutense de Madrid

INTRODUCCIÓN

MÓDULOS

Fichero:

- Se crea un fichero JS con la función a definir y ésta se asigna a la variable `module.exports`

`module.exports = fib;` (`modulofib.js`) Se pueden asignar varias funciones `module.exports = { ... }`

- Ahora podemos hacer uso de la función `fib` desde otro fichero

```
const fib = require("./modulofib.js");
```

`require(modulo)` → Ejecuta el fichero `módulo` pasado como parámetro, devolviendo el objeto asignado a `module.exports`
→ Siempre hay que indicar el path

Core

- No hay que indicar un path
- Tipos: os, `fs`, `path`, http, util ...

Paquete

- Se guardan en la carpeta `node_modules`
- Se importan con `require` sin indicar el path relativo

Para instalar las dependencias usar npm

`npm init` → nuevo proyecto ≈ `package.json`
`npm install xxxx --save` → dependencias, `package-lock.json`

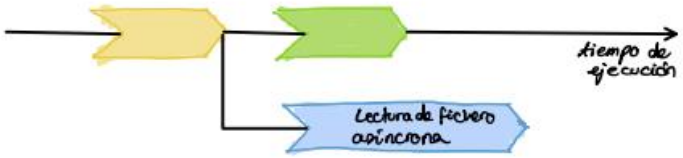
Instalar un paquete de forma global
`npm install -g nodemon`

MODELO ASÍNCRONO

Síncrono vs asíncrono

- Operación síncrona → La ejecución del programa **se detiene** hasta que dicha operación haya terminado
  Devuelven el resultado de la operación
  En caso de error, lanzan **excepciones**

- Operación asíncrona → La ejecución del programa **continua**, mientras la operación se realiza de manera **concurrente**
  No devuelve ningún valor, el resultado de la lectura se pasa como parámetro a la función callback
  No lanza excepciones, en caso de error es la función callback quien recibe la excepción como primer parámetro



↳ Uso de funciones **callback**

1) Callback
2) Promise } Alternativas
3) Async

readFile (fichero, opcs, **callback**)
↓
FIN → invoca

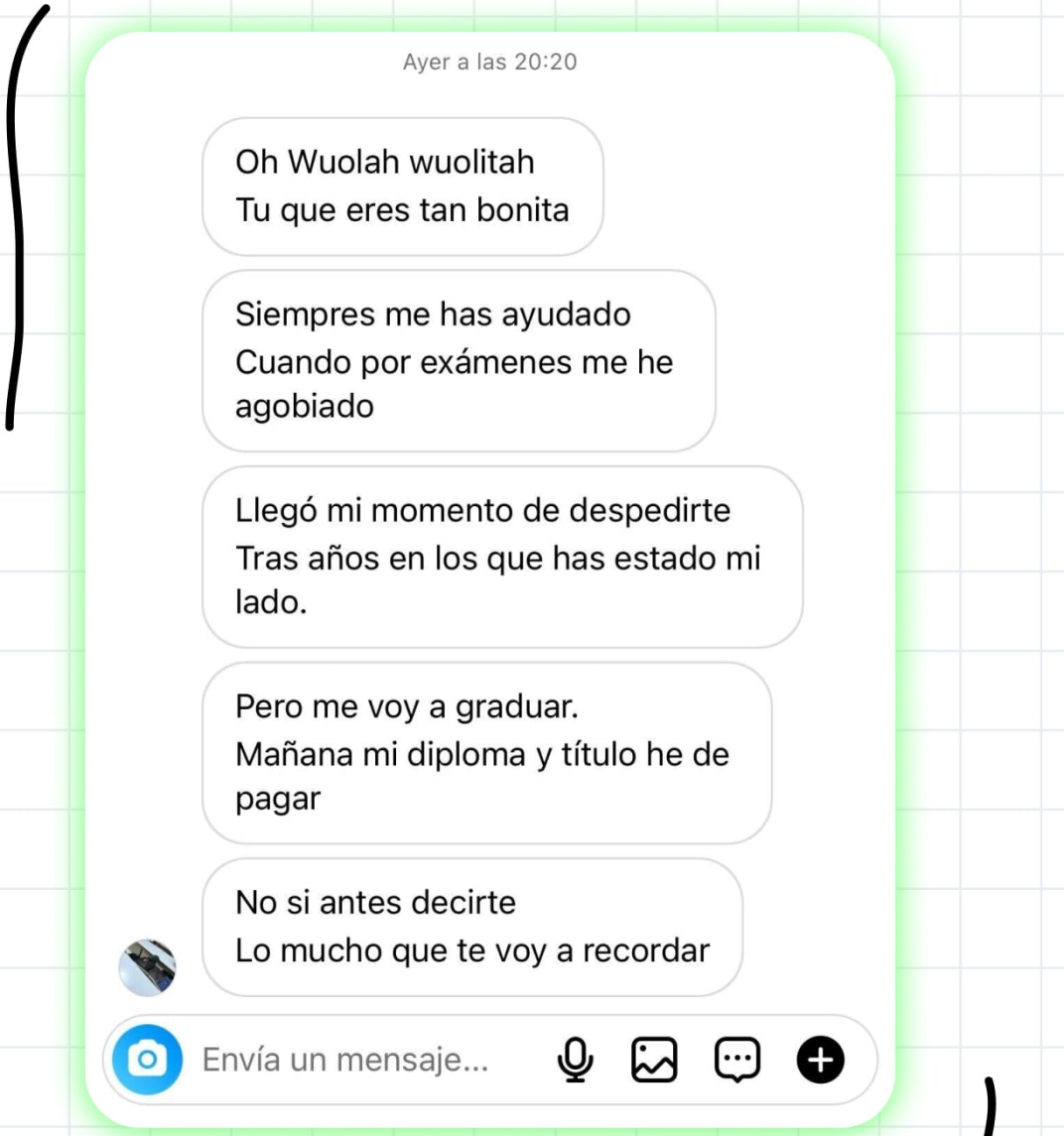
Lectura → OK (error, resultado)
(null, contenido)
↳ Falla (error, undifined)

```
"use strict";

const fs = require("fs");

fs.readFile("FichTexto.txt",
            { encoding: "utf-8" },
            function(err, contenido) {
                if (err) {
                    console.log("Se ha producido un error:");
                    console.log(err.message);
                } else {
                    console.log("Fichero leído correctamente:");
                    console.log(contenido);
                }
            }
);
```

→ Notación flecha (solo en funciones anónimas)
→ Más común

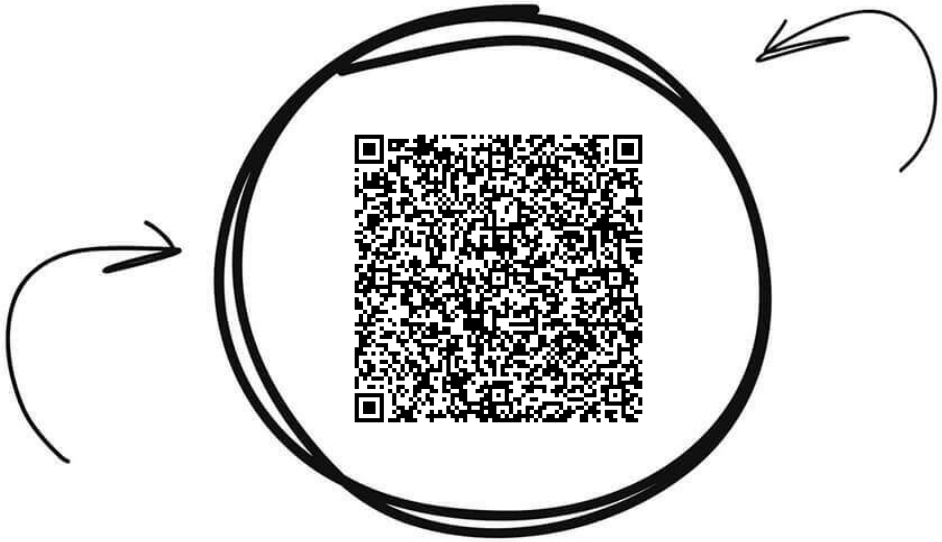
Que no te escriban poemas de amor cuando terminen la carrera 🚩🚩🚩🚩🚩🚩

(a nosotros por suerte nos pasa)



WUOLAH

Aplicaciones Web



Banco de apuntes de la UCM

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

MORALEJA

Si un fragmento del programa depende del resultado de una llamada asíncrona, no debe ir tras dicha llamada:

```
fs.readFile("fich.txt", function(err, contenido) {
    ...
});
// hacer algo con el contenido
```

Debe ir dentro de la función callback:

```
fs.readFile("fich.txt", function(err, contenido) {
    // hacer algo con el contenido
    ...
});
```

ACCESOS A BASES DE DATOS

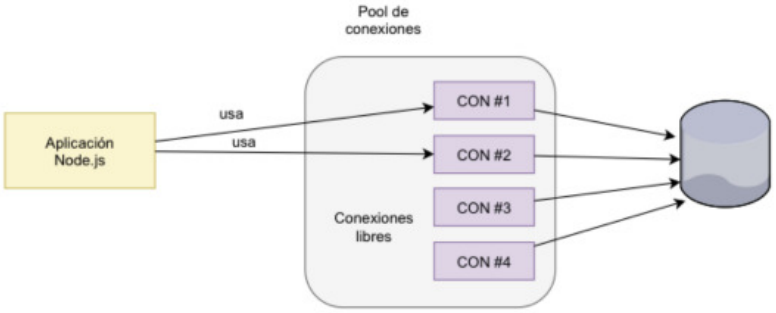
XAMPP { Apache [init] / MySQL [init] [admin]

visual
Para parar el pool
ctrl+C

```
npm install mysql --save
```

Pool de conexiones

Contenedor de conexiones permanentemente abiertas a una determinada BBDD



Código completo

```
const mysql = require("mysql");
const pool = mysql.createPool({ host:  "localhost",
                                 user:  "root",
                                 password: "",
                                 database: "miBD" });
pool.getConnection(function(err, connection) {
    if (err) {
        console.log(`Error al obtener la conexión: ${err.message}`);
    }
    else {
        connection.query(
            "SELECT Nombre, Apellidos FROM Contactos",
            function(err, filas) {
                connection.release();
                if (err) {
                    console.log('Error al realizar la consulta');
                }
                else {
                    filas.forEach(function(fila) {
                        console.log(`${fila.Nombre} ${fila.Apellidos}`);
                    });
                }
            });

    }
});
```

Creación de Pool
Obtiene conexión del Pool
Ejecuta (sin) consultas y modificaciones
Resultado de la consulta
Devuelve la conexión al pool

Operación completa

1. Configuración del pool
2. Solicitud de conexión
3. Ejecución de consulta (y liberación de la conexión)

si lees esto me debes un besito

Que no te escriban poemas de amor
cuando terminen la carrera ►►►►►►

(a nosotros por suerte nos pasa)

No si antes decirte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de pagar

Llegó mi momento de despedirte
Tras años en los que has estado mi lado.

Siempres me has ayudado
Cuando por exámenes me he agobiado

Oh Wuolah wuolitah
Tu que eres tan bonita

INSERT, UPDATE, DELETE en query()

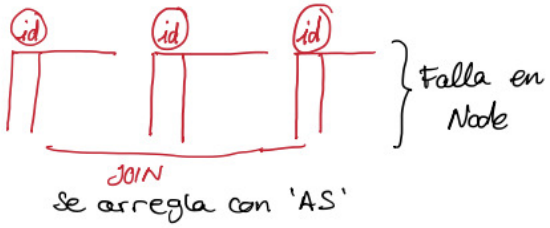
No reciben datos de consulta, la callback recibe un objeto con distintos atributos

- affectedRows
- insertId (Auto_increment solo)

Consultas

SELECT nombre AS nom, apellidos AS ap FROM contactos

[{ nombre: ___, apellido: ___ },
  { nombre: ___, apellido: ___ }
  :
]



↳ Where id = "1 OR TRUE"
↓
Muestra el contenido de toda la tabla } Se evita con una consulta paramétrica

Inyección SQL

let id = "1; DROP table X";

Se evitan con **consultas paramétricas**, que son consultas con **huecos** (donde se quieran insertar datos variables), representados con **?**

INSERT INTO contactos (Nombre, Apellidos) VALUES (?, ?);

query(consulta, valores_marcadores, callback)

```
// Suponemos que la variable `id` contiene el identificador
// introducido por el usuario
connection.query(
    `SELECT Nombre, Apellidos FROM contactos WHERE Id = ?`,
    [id],
    function(err, rows) {
    if(err){
        console.log('Error en la consulta');
    }
    else {
        console.log(rows);
    }
    });
```

SERVIDORES WEB

```
"use strict";
const http = require("http");

// Establecimiento de la función callback del servidor
const servidor = http.createServer(function(request, response) {
    // ...
});

// Inicio del servidor
servidor.listen(3000, function(err) {
    if (err) {
        console.log(`Error al abrir el puerto 3000: ${err}`);
    } else {
        console.log("Servidor escuchando en el puerto 3000.");
    }
});
```

Cómo es el Server

Incoming msg → method, URL, headers

Server Response → Construir la respuesta (métodos)

callback

Arranca el Server

Ejemplo servidor http://localhost:3000/index.html

Iniciar un servidor en el puerto 3000

```
const http = require('http');

// Establecimiento de la función callback del servidor
let servidor = http.createServer(function(request, response){
    // funcionalidad del servidor
    console.log(`Método ${request.method}`);
    console.log(`URL ${request.url}`);
    console.log(request.headers);
    response.statusCode = 200;
    response.setHeader("Content-Type", "text/html");
    response.write('<!DOCTYPE html>');
    response.write('<html>');
    response.write('<head>');
    response.write('<title>Página de prueba</title>');
    response.write('<meta charset="utf-8">');
    response.write('</head>');
    response.write('<body></ul>');
    for (let i = 0; i < 10; i++) {
        response.write(`<li>Item ${i}</li>`);
    }
    response.write('</ul></body></html>');
    response.end();
});

// Inicio del servidor
servidor.listen(3000, function(err) {
    if(err) {
        console.log("Error al iniciar el servidor");
    }
    else {
        console.log("Servidor escuchando en el puerto 3000")
    }
});
```

Consideramos que está todo bien y devolvemos una página

Tipo de contenido de la respuesta

(Obligatoria) Terminamos la respuesta

si lees esto me debes un besito

Combinar el servidor con una BBDD

```
const http = require("http");
const mysql = require("mysql");
const pool = mysql.createPool({
    host: "localhost",
    user: "root",
    password: "",
    database: "mibd"
});


function consultaBD(callback) {
    pool.getConnection(function (err, conexion) {
        if (err) {
            callback(err);
        }
        else {
            conexion.query("SELECT Nombre, Apellidos, COUNT(corr.Correo) as NumCorreos " +
                "FROM Contactos con LEFT JOIN Correos corr ON con.Id = corr.Id " +
                "GROUP BY con.id", function (err, rows) {
                    conexion.release();

                    if (err) {
                        callback(err);
                    }
                    else {
                        callback(null, rows);
                    }
                });
        }
    });
}
```

```
const servidor = http.createServer(function (request, response) {
    consultaBD(function (err, filasBD) {
        if (err) {
            response.statusCode = 500;  // Internal Server Error
            console.error(err);
        }
        else {
            response.statusCode = 200;
            devolverPagina(response, filasBD);
        }
    });
});
```

```
function devolverPagina(response, filasBD) {
    response.setHeader("Content-Type", "text/html");
    response.write('<html>');
    response.write('<head>');
    response.write('<title>Base de datos de teléfonos</title>');
    response.write('<meta charset="utf-8">');
    response.write('<style>th, td { border: 1px solid }</style>');
    response.write('</head>');
    response.write('<body>');
    response.write('<table>');
    response.write('<tr><th>Nombre</th><th>Apellidos</th>' + '<th>Número direcciones</th></tr>');

    filasBD.forEach(function (fila) {
        response.write('<tr>');
        response.write(`<td>${fila.Nombre}</td>`);
        response.write(`<td>${fila.Apellidos}</td>`);
        response.write(`<td>${fila.NumCorreos}</td>`);
        response.write('</tr>');
    });

    response.write('</table>');
    response.write('</body></html>');
    response.end();
}

servidor.listen(3000, function (err) {
    if (err) {
        console.log("Error al iniciar el servidor");
    }
    else {
        console.log("Servidor escuchando en el puerto 3000")
    }
});
```