NodeJs

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial

Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro

SISTEMA NACIONAL DE GARANTÍA JUVENIL

01 Introducción

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial

Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro

SISTEMA NACIONAL DE GARANTÍA JUVENIL

¿Para qué?

¿Qué es Node.js?

Lo que todos sabemos

- Hay Javascript por alguna parte
- Backend
- ¿Algo que ver con NoSQL?
- Sirve para hacer chats

GOBIERNO DE ESPAÑA · MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO · EOI Escuela de organización industrial
Unión Europea · Fondo Social Europeo · Iniciativa de Empleo Juvenil · El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

¿Para qué?

¿Qué es Node.js?

Lo que no está tan claro

- ¿Es un framework para Javascript?
- ¿Es una librería?
- ¿Qué tiene que ver v8 con Node.js?
- **¿Para qué sirve (además de los chats)?**

GOBIERNO DE ESPAÑA · MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO · EOI Escuela de organización industrial
Unión Europea · Fondo Social Europeo · Iniciativa de Empleo Juvenil · El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

¿Para qué?

¿Qué es Node.js?

Node.js es:

"Una plataforma de software usada para construir aplicaciones de red escalables (especialmente servidores). Node.js utiliza JavaScript como lenguaje, y alcanza alto rendimiento utilizando E/S no bloqueante y un bucle de eventos de una sola hebra".

GOBIERNO DE ESPAÑA · MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO · EOI Escuela de organización industrial
Unión Europea · Fondo Social Europeo · Iniciativa de Empleo Juvenil · El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

¿Para qué?

¿Qué es Node.js?

Por ejemplo, si...

- ...para **ruby** tenemos **Rails**...
- ...para **python** tenemos **Django**...
- ...para **php** tenemos **Symphony**...
- ¿Podríamos decir que **Node.js** es el equivalente para **JavaScript**?...

GOBIERNO DE ESPAÑA · MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO · EOI Escuela de organización industrial
Unión Europea · Fondo Social Europeo · Iniciativa de Empleo Juvenil · El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

¿Para qué?

¿Qué es un *"lenguaje de programación"*?

- Una gramática que define la sintaxis del lenguaje
- Un intérprete/compilador que lo sabe interpretar y ejecutar
- **Mecanismos para interactuar con el mundo exterior (llamadas al sistema)**
- **Librería estándar (consola, ficheros, red, etc,...)**
- **Utilidades (intérprete interactivo, depurador, paquetes)**

GOBIERNO DE ESPAÑA · MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO · EOI Escuela de organización industrial
Unión Europea · Fondo Social Europeo · Iniciativa de Empleo Juvenil · El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

V8

Motor V8

v8 (JavaScript)

- Desarrollado en C++
- Una gramática que define la sintaxis del lenguaje
- Un intérprete/compilador que lo sabe interpretar y Ejecutar
- Mecanismos para interactuar con el mundo exterior (llamadas al sistema)
- Librería estándar (consola, ficheros, red, etc,...)
- Utilidades (intérprete interactivo, depurador, paquetes)

GOBIERNO DE ESPAÑA · MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO · EOI Escuela de organización industrial
Unión Europea · Fondo Social Europeo · Iniciativa de Empleo Juvenil · El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

V8

Motor V8

Node.js es algo más:

- Una filosofía sobre cómo hacer las cosas
- Un modelo de ejecución singular
- Muy enfocado hacia aplicaciones de red

GOBIERNO DE ESPAÑA · MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO · EOI Escuela de organización industrial
Unión Europea · Fondo Social Europeo · Iniciativa de Empleo Juvenil · El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

V8

Motor V8

Node.js se crea con un objetivo en mente:

• Escribir aplicaciones muy eficientes (E/S) con el lenguaje dinámico más rápido (v8) para soportar miles de conexiones simultáneas

• Sin complicarse la vida innecesariamente

- Sin paralelismo

- Lenguaje sencillo y muy extendido

- API muy pequeña y muy consistente

- Apoyándose en Eventos y Callbacks

V8

Motor V8

- No es la mejor opción para todos los casos: Si puedes hacerlo con Spring hazlo
- Evita las “soluciones totales” : Una necesidad, una herramienta

- Combina diferentes herramientas simples : Entiende lo que estás haciendo

- Flexibilidad > magia

- Tu código es tu responsabilidad

- Cada aplicación es un mundo

V8

Motor V8

Para entender **Node.js**, tenemos que entender dos ideas fundamentales:

- Concurrencia vs. paralelismo (asincronía)
  <> Paralelismo: varios actores realizando una acción cada uno **simultáneamente**
  <> Concurrencia: un solo actor, con varias tareas “activas” **entre las que va alternando**

- Eventos

V8

Motor V8

La potencia de **Node.js** es (curiosamente):

• Un modelo de ejecución **concurrente :** Muchos clientes o tareas activas

• Pero **NO paralelo**

- Una única hebra

Patrón Reactor

Un patrón de diseño para manejar eventos donde peticiones de servicio se transladan concurrentemente a un manejador central que se encarga de desmultiplexar las peticiones y despacharlas síncronamente mediante sus manejadores particulares asociados

Patrones de diseño

Patrón Reactor

Patrón Reactor

- Programación contra eventos
- Una vez puestos los manejadores, se pueden ejecutar en cualquier orden. El orden lo determina el orden en que aparezcan sucesos
- La ejecución de los manejadores **bloquea la hebra**
- Nunca hay dos manejadores ejecutándose al mismo tiempo
- Muy eficiente... cuando E/S >> ejecuión del manejador

GOBIERNO DE ESPAÑA · MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO · EOI Escuela de organización industrial
Unión Europea · Fondo Social Europeo · Iniciativa de Empleo Juvenil · El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

Patrones de diseño

Patrón Reactor

Muy enfocado hacia aplicaciones de red

• Mucha E/S : Por tanto, mucho tiempo con la CPU inactiva

• Para aprovechar ese tiempo, necesitas otros clientes
que lo puedan aprovechar

• Similar a un camarero en un bar

- Es un “Patron Reactor” del mundo real

- Para aprovecharlo, tiene que haber varios clientes!

- Un cliente no termina más deprisa por dedicarle un camarero sólo a él

GOBIERNO DE ESPAÑA · MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO · EOI Escuela de organización industrial
Unión Europea · Fondo Social Europeo · Iniciativa de Empleo Juvenil · El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

02 Instalación

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial

Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro

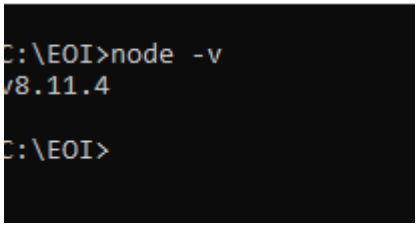
SISTEMA NACIONAL DE GARANTÍA JUVENIL

NodeJs

Descargar el instalador desde https://nodejs.org/es/

NodeJs

Una vez instalado ya tendremos disponible desde línea comando el comando ***node***



Recordad que ***Node.js*** ejecuta código Javascript y lo ejecuta todo de tirón. No stendremos que encargar de poner en nuestro código los manejadores necesarios para capturar eventos o cualquier otra cosa que ocurra en nuestro entorno.

03

Configuración Proyecto

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial

Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro

SISTEMA NACIONAL DE GARANTÍA JUVENIL

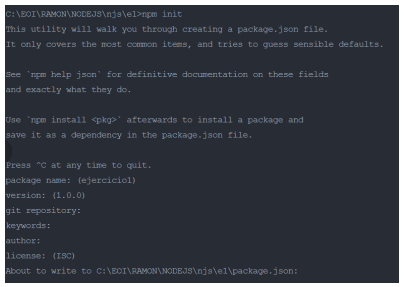
NodeJs

package.json

- Cualquier aplicación ***Node.js*** utiliza un fichero de configuración donde se guardan los paquetes necesarios del repositorio ***npm (Node Packet Manager)*** llamado ***package.json***

- Los proyectos se guardan en la carpeta que queramos

- Crear un fichero de configuración ***package.json ,*** ejecutar ***npm init*** dentro del directorio del proyecto, nos pedirá rellenar un poco de información. Por convenido siempre vamos a llamar a nuestro fichero principal de la aplicación ***app.js***



NodeJs

package.json

Una vez tenemos el fichero de configuración lo abrimos y vemos lo que ha creado

```
{
  "name": "ejercicio1",
  "version": "1.0.0",
  "description": "Ejercicios NodeJS",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "nodemon app"
  },
  "author": "",
  "license": "ISC"
}
```

Nombre del proyecto

Scripts para ejecutar con ***npm***
Por ejemplo ***npm start***

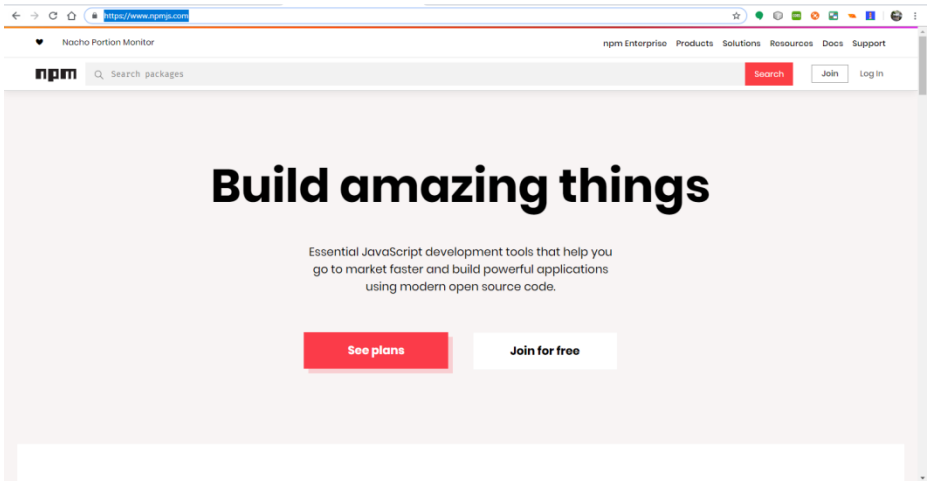
- Recordemos que una aplicación ***Node.js*** es una aplicación ***JavaScript*** que usa paquetes / librerías de un repositorio ***npm***

NodeJs

Dependencias

- Podemos consultar todos los paquetes / librerías que hay publicados en el repositorio de ***npm***

https://www.npmjs.com/



NodeJs

Dependencias

- Si necesitamos un paquete / librería para poder desarrollar nuestra App ***Node.js*** lo buscamos en el repositorio y una vez hayamos decidido cuál instalar /descargar, lo realizamos con el siguiente comando.
  - ***npm install <paquete>***
  - Si queremos guardar la referencia del paquete / librería en el fichero de configuración como una dependencia, además tenemos que poner ***npm install –save <paquete>***, esto nos guardará la referencia a al paquete en el elemento ***dependencies*** de nuestro fichero de configuración ***package.json***

```
"dependencies": {
  "nodemon": "^1.19.1"
}
```

NodeJs

Dependencias

- Vamos a instalar en nuestro proyecto base (***ejercicio1)*** un paquete llamado ***nodemon***, el cual reinicia nuestra aplicación cuando detecta que alguno de los ficheros de la misma ha cambiado.
  - ***npm install –save nodemon***

```
{
  "name": "ejercicio1",
  "version": "1.0.0",
  "description": "Ejercicios NodeJS",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "nodemon app"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "nodemon": "^1.19.1"
  }
}
```

NodeJs

Dependencias

- Los paquetes / librerías se instalan en la carpeta **node_modules** . Esta carpeta la podemos borrar para copiar la carpeta del proyecto a otro servidor ya que esta carpeta es bastante grande. Para instalar los paquetes otra vez (Dependencias guardadas en el ***package.json***) solo tenemos que ejecutar ***npm install*** en el raíz del proyecto, al mismo nivel que el fichero ***package.json***

- Para cargar los paquetes / librerías / ficheros , lo tenemos que hacer usando la palabra clave ***require('<paquete> o <ruta>');*** *o* ***import*** *,* según usemos **Commonjs Modules** o **ES Modules**

- Importar módulos (paquetes, otros ficheros)

- Garantía: una única vez

- Devuelve el módulo!

NodeJs

Dependencias (Ejemplo Commonjs Modules)

app.js

Pruebas-Common-Js-Modules > app.js > ...

```
1
2  let funciones=require("./funciones");
3
4
5  console.log(funciones.funciones.f2("HOLAsssss"));
```

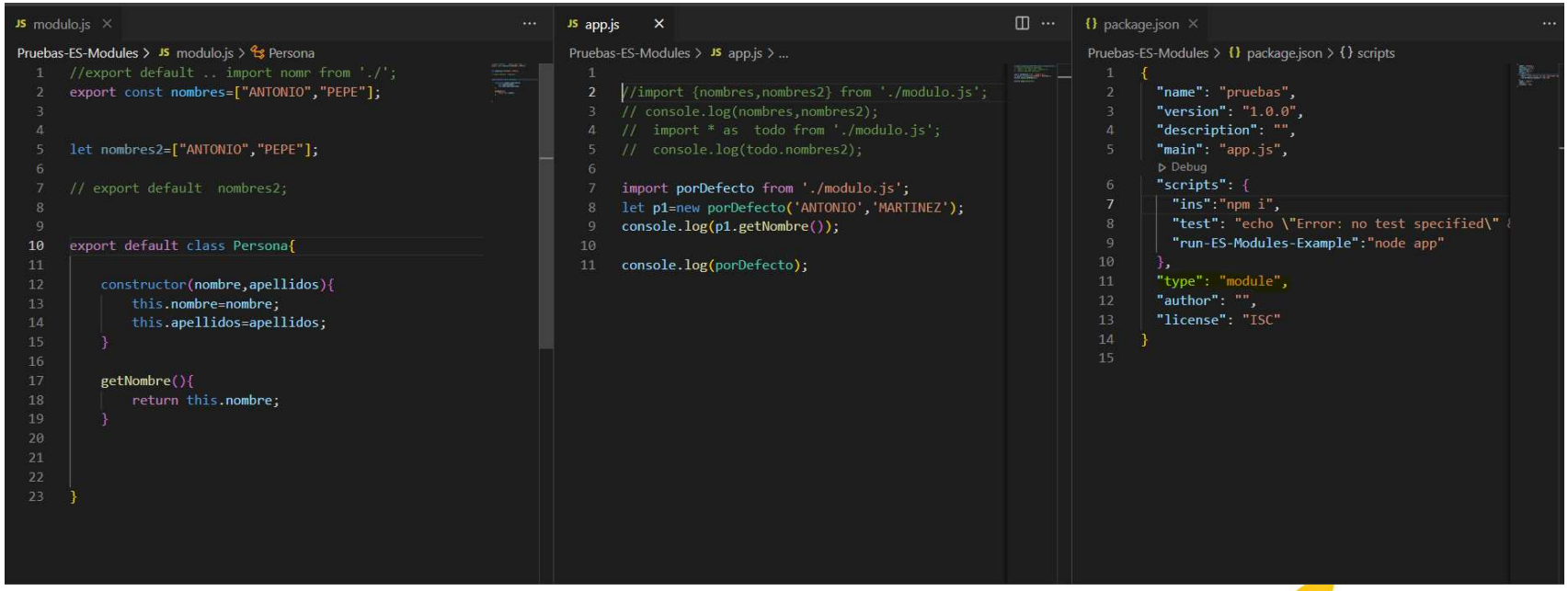
funciones.js

Pruebas-Common-Js-Modules > funciones.js > ...

```
1
2
3  exports.funciones={
4      nombre:'asassa',
5      f1:()=>{
6          return "f1";
7      },
8      f2:(p1)=>{
9          return `f2 ${p1}`;
10     }
11
12
13 };
```

NodeJs

Dependencias (Ejemplo ES Module)



modulo.js

```
//export default .. import nomr from './';
export const nombres=["ANTONIO","PEPE"];


let nombres2=["ANTONIO","PEPE"];

// export default  nombres2;


export default class Persona{

    constructor(nombre,apellidos){
        this.nombre=nombre;
        this.apellidos=apellidos;
    }

    getNombre(){
        return this.nombre;
    }


}
```

app.js

```
//import {nombres,nombres2} from './modulo.js';
// console.log(nombres,nombres2);
//  import * as  todo from './modulo.js';
//  console.log(todo.nombres2);

import porDefecto from './modulo.js';
let p1=new porDefecto('ANTONIO','MARTINEZ');
console.log(p1.getNombre());

console.log(porDefecto);
```

package.json

```
{
  "name": "pruebas",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  Debug
  "scripts": {
    "ins":"npm i",
    "test": "echo \"Error: no test specified\" 
    "run-ES-Modules-Example":"node app"
  },
  "type": "module",
  "author": "",
  "license": "ISC"
}
```

NodeJs

Paquetes / librerías

exports.<propiedad_a_exportar> = <valor>

- El otro lado del mecanismo. Podemos hacer ***require*** porque en algún otro fichero hay un ***exports.<propiedad_a_exportar>***
- Se puede exportar cualquier valor

Funciona como los módulos que vimos en TypeScript

NodeJs

Paquetes / Librerías

Ejemplo de importación de fichero como libreria

codigo.js

```
var lib = require("./libreria");
console.log(lib.propiedad);
```

libreria.js

```
console.log("una vez");
exports.propiedad = "Pública";
```

NodeJs

Paquetes / Librerías

entidades.js

```
Pruebas-Express > entidades.js > tablas
1  exports.tablas={
2
3      usuarios:[],
4      articulos:[]
5
6  }
```

app-all.js

```
Pruebas-Express > app-all.js > ...
1  const express = require('express');
2
3  const bodyParser= require("body-parser");
4
5  const entidades=require("./entidades");
6
7  const app = express();
8
```

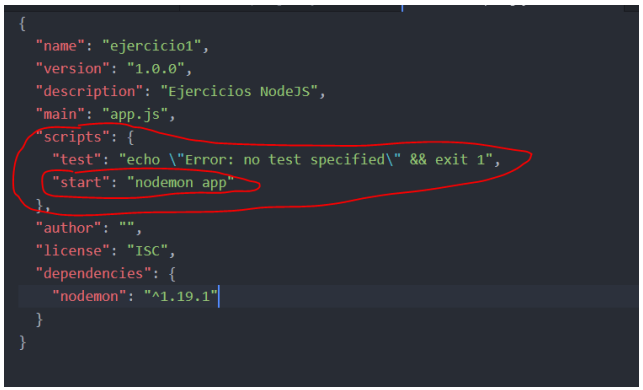
```
app.post('/entidades/articulo/',  (req, res,next) => {
    let params=req.body;

    try{
        entidades.tablas.articulos.push(req.body);
        res.sendStatus(200);
    }catch(err){
        res.sendStatus(500);
    }

});
```

NodeJs

Script inicio

- Ahora que ya tenemos configurada nuestra App con nuestro primer paquete (***nodemon***) , vamos a crear un Script de inicio de nuestra aplicación, para poder invocarlo con el comando ***npm***

- Cualquier script que creemos dentro del elemento ***scripts*** del fichero de configuración ***package.json*** se podrá invocar con el comando ***npm <nombre_script>***



NodeJs

Script inicio

- En este caso hemos configurado el Script ***start*** para que ejecute ***nodemon app (nodemon app.js),*** si no queremos usar ***nodemon*** tenemos que poner ***node app*** para poder ejecutar nuestra aplicación ***Node.js***

- Podemos ver los nombres de scripts permitidos con el comando ***npm help --config***

```
Usage: npm <command>

where <command> is one of:
    access, adduser, bin, bugs, c, cache, completion, config,
    ddp, dedupe, deprecate, dist-tag, docs, doctor, edit,
    explore, get, help, help-search, i, init, install,
    install-test, it, link, list, ln, login, logout, ls,
    outdated, owner, pack, ping, prefix, profile, prune,
    publish, rb, rebuild, repo, restart, root, run, run-script,
    s, se, search, set, shrinkwrap, star, stars, start, stop, t,
    team, test, token, tst, un, uninstall, unpublish, unstar,
    up, update, v, version, view, whoami
```

Ejercicio1

Crear una aplicación **js** que realice lo siguiente

a) Cada 5 segundos muestre por consola el mensaje ***Hola, esta es mi primera aplicación con Node.js***

b) Crear un script en el fichero de configuración que se llame ***start*** y ejecute la aplicación con ***node*** en lugar de con ***nodemon***

04 EventEmitter

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial

Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro

SISTEMA NACIONAL DE GARANTÍA JUVENIL

Eventos

Nuestro código va a estar dirigido por eventos

- Node.js tiene su propio “estándar”
- Trae una implementación del **patrón Observador** (o Pub/Sub): **EventEmitter**
- Todas sus librerías (y casi todos los paquetes) siguen este modelo

GOBIERNO DE ESPAÑA · MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO · EOI Escuela de organización industrial
Unión Europea · Fondo Social Europeo · Iniciativa de Empleo Juvenil · El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

Eventos

Ejemplo de uso de EventEmitter :

Nos fijamos en la primera línea, estamos usando un paquete ***events*** que previamente hemos instalado con ***npm install events -g***

```
var EventEmitter = require(“events”).EventEmitter;
var pub = new EventEmitter();

pub.on("ev", (m) =>{
                console.log("[ev]", m);
});
pub.once("ev", (m) =>{
                console.log("(ha sido la primera vez)");
});
pub.emit("ev", "Soy un Emisor de Eventos!");
pub.emit("ev", "Me vas a ver muy a menudo...");
```

Ejercicio2

crear una aplicación **js** que realice lo siguiente

a) Implementar el ejemplo anterior ¿Qué es lo que ocurre?

05

Http

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial

Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro

SISTEMA NACIONAL DE GARANTÍA JUVENIL

Llamadas Http

Node.js tiene un servidor Web bastante interesante

- Asíncrono
  - No bloquea la hebra
  - Cada cliente conectado consume muy poquitos recursos
  - Genial para miles de conexiones simultáneas
- Relativamente rápido
- Interfaz sencilla
- HTTP puro y duro, sin adornos
- Basado en streams y eventos

Llamadas Http

```
var http = require("http");

var server = http.createServer();

server.on("request", function(req, res) {
                res.end(“<b>Hola, Mundo!</b>");
});

server.listen(3000);
```

req - > http.IncomingMessage (Parámetros que recibimos)
res -> http.ServerResponse (Respuesta que enviamos)

Llamadas Http

El servidor HTTP

• Eventos:

- ***connection***
- ***request***

• Operaciones:

- ***createServer***([requestCallback])
- **listen**(puerto, [hostname], [backlog], [callback])
- ***close***([callback])

Llamadas Http

http.IncomingMessage (*parametro "req"*)

- Representa la petición HTTP del cliente

- Propiedades:
  - ***req.headers***: cabeceras de la petición
  - ***req.method***: verbo HTTP
  - ***req.url***: url de la petición
  - ***req.conneciton.remoteAddress***: ip del cliente

Llamadas Http

http.ServerResponse (*parametro "res"*)

- Representa la respuesta del servidor
- Stream de escritura
- Operaciones adicionales:

  - ***res.writeHead(statusCode, [headers])***: código HTTP y cabeceras de la respuesta

  - ***res.statusCode***: [propiedad] Otra forma de establecer el código HTTP de la respuesta

  - ***res.setHeader(name, value)***: Otra forma de establecer las cabeceras, de una en una

Ejercicio3

Crear una aplicación **js** que realice lo siguiente

a) Implementar el ejemplo anterior del servidor Http

GOBIERNO DE ESPAÑA MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO EOI Escuela de organización industrial
Unión Europea Fondo Social Europeo Iniciativa de Empleo Juvenil El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

06 Promesas

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial

Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro

SISTEMA NACIONAL DE GARANTÍA JUVENIL

Llamdas Asíncronas

Node.js maneja la asincronía utilizando callbacks

- Podemos crear promesas usando el paquete ***promise***

```
var Promise = require('promise');

new Promise( (resolve, reject) => {
                ....
                resolve(<Devolvemos el resultado correcto>);
                ....
                reject(<Devolvemos el resultado erróneo>);

});
```

- Recordar que estamos trabajando con ***JavaScript***, podemos devolver lo que queramos (Objecto JSON, Texto ,...), mejor devolvemos Objetos JSON, mucho más fácil de trabajar y estructurar.

Llamdas Asíncronas

- Las promesas las utilizamos en el lado del servidor que hacen las llamadas críticas dentro de nuestra App : conexiones a base de datos, procesos de cálculo , ...

- ¿Cómo trabajo con las promesas?

- Tenemos que crear funciones que devuelvan promesas

```
function hazAlgoCritico(){
        return new Promise(....);
}
```

Llamadas Asíncronas

- Para trabajar con las promesas que devuelven las funciones tenemos que hacer lo siguiente :

```
hazAlgoCritico.then(

        ok =>{
                <Aquí proceso el resultado devuelto por
                la promesa cuando no ha sucedido un error>
        },
        error =>{
                <Aquí proceso el resultado devuelto por
                la promesa cuando ha sucedido un error>

        }
);
```

ok , error -> Objeto JSON devuelto por la promesa

Ejercicio4

Crear una aplicación **js** que realice lo siguiente

a) Crear una función llamada ***hazAlgoCritico*** que reciba un único parámetro llamado ***p1***. Si ***p1=='1'*** la promesa debe devolver ***{resultado:'EJECUTADO CON ÉXITO'}***, si ***p1=='0'*** la promesa debe devolver ***{resultado:'EJECUTADO CON ERROR'}***

b) Invocar a la función ***hazAlgoCritico*** con ***p1=1 ,***después con ***p1=0*** y por último con ***p1=2***

Llamadas Asíncronas

Los callbacks tienen muchas ventajas

• Muy fáciles de entender e implementar

• Familiares para el programador JavaScript

• Extremadamente flexibles (clausuras, funciones de primer orden, etc, ...)

• Un mecanismo universal de asincronía/continuaciones

Pero...

GOBIERNO DE ESPAÑA MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO EOI Escuela de organización industrial
Unión Europea Fondo Social Europeo Iniciativa de Empleo Juvenil El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

Llamadas Asíncronas

Una manera alternativa de modelar asincronía

- Construcción explícita del flujo de ejecución
- Separación en bloques consecutivos
- Manejo de errores más controlado
- Combinación de diferentes flujos asíncronos

GOBIERNO DE ESPAÑA · MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO · EOI Escuela de organización industrial
Unión Europea · Fondo Social Europeo · Iniciativa de Empleo Juvenil · El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

Llamadas Asíncronas

.then(success, [error])

- Concatena bloques
- El nuevo bloque (success)...
  - Sólo se ejecuta si el anterior se ha ejecutado sin errores
  - Recibe como parámetro el resultado del bloque anterior
  - Devuelve el valor que se le pasará el siguiente bloque

GOBIERNO DE ESPAÑA · MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO · EOI Escuela de organización industrial
Unión Europea · Fondo Social Europeo · Iniciativa de Empleo Juvenil · El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

Llamadas Asíncronas

➡ Si es un **dato inmediato**, se pasa tal cual

➡ Si es una **promesa**, se resuelve antes de llamar al siguiente bloque

• El segundo parámetro pone un manejador de error

- Equivalente a llamar a .fail(error)

• .then(...) **siempre devuelve una nueva promesa**

GOBIERNO DE ESPAÑA · MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO · EOI Escuela de organización industrial
Unión Europea · Fondo Social Europeo · Iniciativa de Empleo Juvenil · El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

Llamadas Asíncronas

Las promesas se **resuelven** o se **rechazan**

Si se resuelven:

- Se resuelven a un valor (si es un bloque, su valor de retorno)
- Representan el estado *"OK, puede seguir el siguiente"*

Si se rechazan:

- Representan un error
- La ejecución cae hasta el siguiente manejador de errores
- Se saltan todos los estados desde el error hasta el manejador

Llamadas Asíncronas

var Promise = require('promise');

```
return new Promise(function (resolve, reject) {
    sql.connect("mssql://usuario:password@servidor/bd").then(function (ok) {
        var sp = new sql.Request();
        for (var pos in parametros) {
            sp.input(pos, parametros[pos]);
        }
        sp.execute(procedimiento).then(function (resultado) {
            resolve(resultado);
        }).catch(function (err) {
            reject('<<ERROR>> ' + err);
        });
    });

});
```

07 Express

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial

Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro

SISTEMA NACIONAL DE GARANTÍA JUVENIL

Middleware

Un framework web para Node.js

- Estrictamente web (microframework)
- Podemos definir nuestro servicios REST
- Sencillo y flexible
- Muy popular
- Se adapta muy bien a la filosofía de Node
- Similar a Sinatra, Sylex, Flask, Spark, ...

GOBIERNO DE ESPAÑA · MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO · EOI Escuela de organización industrial
Unión Europea · Fondo Social Europeo · Iniciativa de Empleo Juvenil · El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

Middleware

Express nos va ayudar con...

- Rutas
- Parámetros
- Formularios y subida de ficheros
- Cookies
- Sesiones
- Templates

Middleware

Más concretamente, express...

- Construye sobre http
- Procesando la petición por un stack de middleware que se encarga de decorar las peticiones
  - Asocia rutas a manejadores
  - Decorar los objetos req y res (parseo de parámetros, multipart, etc,...)
  - Rendear templates
- Nosotros escogemos qué middlewares queremos usar, y en qué orden

Middleware

Servidor Web simple con Express

```
let express = require("express“);

app = express();


app.listen(3000);
```

Ejercicio5

Crear una aplicación **js** que realice lo siguiente

a) Crear un servidor Web simple con Express

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial
Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

Añadir funcionalidades

Podemos añadir una gran cantidad de módulos a Express que realicen alguna funcionalidad concreta, vamos a nombrar las más interesantes :

Partimos que nuestro servidor Express está guardado en la variable ***app***

body-parser : paquete que nos permite parsear el cuerpo de las peticiones Http

Ejemplo : establecemos un límite de 50 Mb para el cuerpo de la petición y podemos recibir texto, html, css y Json

bodyParser.json({limit: "50mb", type: "text/html/css/json"})

method-override : paquete que nos permite sobreescribir métodos Http

Cabeceras de respuesta

Configuración de la cabecera de la respuesta del servidor : podemos configurar cualquier parámetro de la cabecera de la Http, para controlar lo que nuestro acepta en cuanto se le envíe una petición REST

```
app.use(        function (req, res, next) {

                res.header("Access-Control-Allow-Origin", "*");

                res.header("Access-Control-Allow-Headers", "Origin, X-
                Requested-With, Content-Type, Accept,Authorization");

                res.setHeader('Access-Control-Allow-Methods', 'GET,                    POST,DELETE,PUT');

                next();
});
```

Añadir funcionalidades

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

const bodyParser= require("body-parser");


const app = express();

let router = express.Router();


app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());


app.use(router);
```

Definir rutas

Definición de las rutas: las rutas son las URL parciales , que vamos a exponer para que respondan como servicios REST, es decir, tenemos nuestro servidor ***http*** que arranca un ***nombre de máquina o dirección IP*** y añadiríamos las rutas al final de lo anterior para construir la URL de nuestro servicio REST.

Ejemplo :

http://localhost:3000 -> sería nuestro servidor Web arrancado con Express

Si le añadimos la ruta ***'/clientes'*** ahora tendríamos la ruta a nuestro servicio REST como ***http://localhost:3000/clientes*** y es dentro de esa ruta donde tendríamos que añadir la funcionalidad necesaria para obtener los clientes desde nuestra fuente de datos (Base de dato, Servicio Web , ...)

Definir rutas

Definición de las rutas: la definición es muy simple. Vamos a definir rutas que van a actuar como servicios REST por lo que tendrán que responder a operaciones tipo ***POST,PUT,DELETE,...***

En todas las rutas que vamos a ver esto es común :

req -> Objeto que maneja la petición realizada
res -> Objeto que maneja la respuesta a enviar

RUTA CON OPERACIÓN GET

```
app.get('/<nombre_ruta>', function (req, res){
        <lógica de la operación GET>

})
```

Definir rutas

req -> Objeto que maneja la petición realizada
res -> Objeto que maneja la respuesta a enviar

RUTA CON OPERACIÓN GET

```
app.get('/<nombre_ruta>', function (req, res){
        <lógica de la operación GET>

})
```

- Para ***recuperar los parámetros en este tipo de petición*** , usamos (***req.query***) , dentro de req.query están todos los parámetros envíados a la petición. Después accedemos a ellos como siempre , usando el **.**

```
let data = req.query;
data.usuario....
```

Definir rutas

req -> Objeto que representa la petición realizada
res -> Objeto que representa la respuesta a enviar

RUTA CON OPERACIÓN GET

```
app.get('/<nombre_ruta>', function (req, res){
        <lógica de la operación GET>

})
```

- Para enviar la respuesta de nuestro servicio REST usamos ***res.send(<respuesta>)***

```
res.send({status: 500, data: 'Error procesando petición'});
```

Definir rutas

RUTA CON OPERACIÓN POST / PUT

app.put(*'/<nombre_ruta>', function (req, res){....});*

app.post(*'/<nombre_ruta>', function (req, res){*
<lógica de la operación GET>

})

- Para ***recuperar los parámetros en este tipo de petición*** , usamos (***req.body***) , dentro de req.body están todos los parámetros envíados a la petición. Después accedemos a ellos como siempre , usando el **.**

  let data = ***req.body****;*
  data.usuario....

- Para enviar la respuesta de nuestro servicio REST usamos ***res.send(<respuesta>)***

  res.send({status: 500, data: 'Error procesando petición'});

Definir rutas

- Dependiendo de la versión de ***Express*** que se esté utilizando, ***bodyParser*** y otras funcionalidades se deberán usar según lo indique en la API, por ello hay que tener muy en cuenta en todo momento que versión de ***Express*** estamos usando y cuál es su API correcta.

Definir rutas

```
app.get('/prueba/', (req, res,next) => {

    res.send("<b>ESTO ES UNA PRUEBA</b>");

});

/**** EJEMPLO QUERY PARAMS : http://localhost:8084/pruebaQueryParams?p1=v1&p2=v2 -> req.query={"p1":"v1","p2":"v2"} */
app.get('/pruebaQueryParams/', (req, res,next) => {

    res.send(req.query);

});

/***** EJEMPLO PATH PARAM : http://localhost:8084/pruebaRouteParams/MANOLO/MARTINEZ -> req.params={"nombre":"MANOLO","apellidos":"MARTINEZ"} */

app.get('/pruebaRouteParams/:nombre/:apellidos', (req, res,next) => {

    res.send(req.params);
});


/***** EJEMPLO POST */

app.post('/pruebaPost/',  (req, res,next) => {

    res.send(req.body);
});


app.listen(8084, function() {
    console.log("Node server running on http://localhost:8084/prueba");
});
```

Ejercicio6

Crear una aplicación **js** que realice lo siguiente

a) Crear un servidor Express con las siguientes rutas

GET /prueba1 : devuelve *{estado:200,resultado:'PRUEBA1'}*
POST /prueba1 : devuelve *el body que recibe*

08 Bases de Datos

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial

Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro

SISTEMA NACIONAL DE GARANTÍA JUVENIL

Acceso a MySql

```
var mysql      = require('mysql');

var connection = mysql.createConnection({
  host     : '<nombre maquina o dirección ip>',
  user     : '<usuario>',
  password : '<contraseña>',
  database : '<nombre_bd>'
});

connection.connect();

exports.funciones={
  consulta:(consulta)=>{
    connection.query(`${consulta}`, function (error, results, fields) {
      if (error) throw error;
      console.log('Consulta: ', results);    });
 }
}
```

09

Templates Pug

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial

Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro

SISTEMA NACIONAL DE GARANTÍA JUVENIL

Pug

- Paquete ***pug***

- Para usarlo con nuestra aplicación ***Node.js*** solo bastaría con poner lo siguiente en el código

```
app.set("view engine", "pug");

// Con esta línea establecemos la carpeta de las plantillas Pug
app.set("views", path.join(__dirname, "views"));

// Con esta línea establecemos como directorio raíz el actual del proyecto
app.use(express.static(__dirname + '/'));
```

Pug

- Renderizar una plantilla Pug en unservicio REST

```
app.get('/pruebas/', (req, res) => {

// Renderiza la plantilla Pug llamada home.pug que está en el directorio
// establecido anteriormente y le pasa el parámetro usuario
 res.render("home", {
   usuario: 'EL USUARIO'
  });

});
```

Pug

- Plantilla ***home.pug***

```
doctype html
html(lang='es')
    head
        link(rel='stylesheet', href='css/bootstrap.min.css')
        link(rel='stylesheet', href='css/estilo.css')
        title Servicio de cambio de contraseña
    body
    header
        div.container
            section
                div.card.text-center
                    div.card-header
                        b
                            font(color="blue") CAMBIO DE CONTRASEÑA
                    div.card-body
                        div.row
                            div.col-3.offset-3
                                h5.card-title
                                    b Usuario
                            div.col-4
                                p.card-text #{usuario}
                                input(type="hidden",id="usuario",value=usuario)
                        div.row
                            div.col-3.offset-3
                                span
                                    b Contraseña anterior
                            div.col-3
                                input(type="password", placeholder="Contraseña anterior",id="passAnt")
                        div.row
                            div.col-3.offset-3
                                span
                                    b Contraseña nueva(*)
                            div.col-3
                                input(type="password", placeholder="Contraseña nueva",id="passNuevo1")
                        div.row
                            div.col-3.offset-3
                                span
                                    b Repetir contraseña(*)
                            div.col-3
                                input(type="password", placeholder="Repetir contraseña",id="passNuevo2")
                        div.card-footer.text-muted
                            button(type="button", id="pass2",onclick="cambiar()",class="btn btn-primary btn-lg") Cambiar

    footer
        div.center (*) Longitud mínima de 8 carácteres, al menos una letra mayúscula y una minúscula, al menos un número y al menos un símbolo (.,#)
        div.jumbotron.oculto(id="mensaje")
    script(src='js/bootstrap.min.js')
```

Ejercicio7

crear una aplicación **js** que realice lo siguiente

a) Crear un servidor Express que implemente las APIs realizadas con *Java* ***(Este ejercicio no hay que entregarlo)***

GOBIERNO DE ESPAÑA · MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO · EOI Escuela de organización industrial
Unión Europea · Fondo Social Europeo · Iniciativa de Empleo Juvenil · El FSE invierte en tu futuro
SISTEMA NACIONAL DE GARANTÍA JUVENIL

https://nodejs.org/es/

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, COMERCIO Y TURISMO
EOI Escuela de organización industrial

Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro

SISTEMA NACIONAL DE GARANTÍA JUVENIL