



CLASE 9

SERVER SIDE - NODE Y EXPRESS

Node JS

AGENDA

1. Node Js
2. Modularización1
3. Módulo Http1
4. Npm
5. Eventos1
6. Eventos2
7. Módulo Formidable1
8. Utilizando Módulos Propios
9. Módulo De Base De Datos
10. Express Js 3
11. Referencias

NODE JS



NODE JS

NODE JS²

- Utiliza programación asincrónica, de un solo thread, eliminando el waiting. Es no bloqueante entonces es más eficiente.
- Basado en el motor V8 de Google, entorno de ejecución que compila el código fuente JavaScript en máquina en lugar de interpretarlo.
- Comunidad de desarrolladores muy activa.

UN POCO DE HISTORIA

- 2010: gestor de paquetes NPM
- 2012: soporte nativo para Windows

LTS 22.13.1

NODE JS. CARACTERÍSTICAS¹

Utiliza programación asincrónica
Envía la tarea al file system, atiende el próximo pedido. Cuando el file server abre el archivo, el servidor retorna el contenido al cliente.



NODEJS. CARACTERÍSTICAS ²

Los archivos contienen tareas que pueden ser ejecutadas ante ciertos eventos, por ejemplo el acceso a un puerto del servidor.

Tienen extensión .js y deben ser iniciados en el servidor antes de tener algún efecto.

MODULARIZACIÓN¹

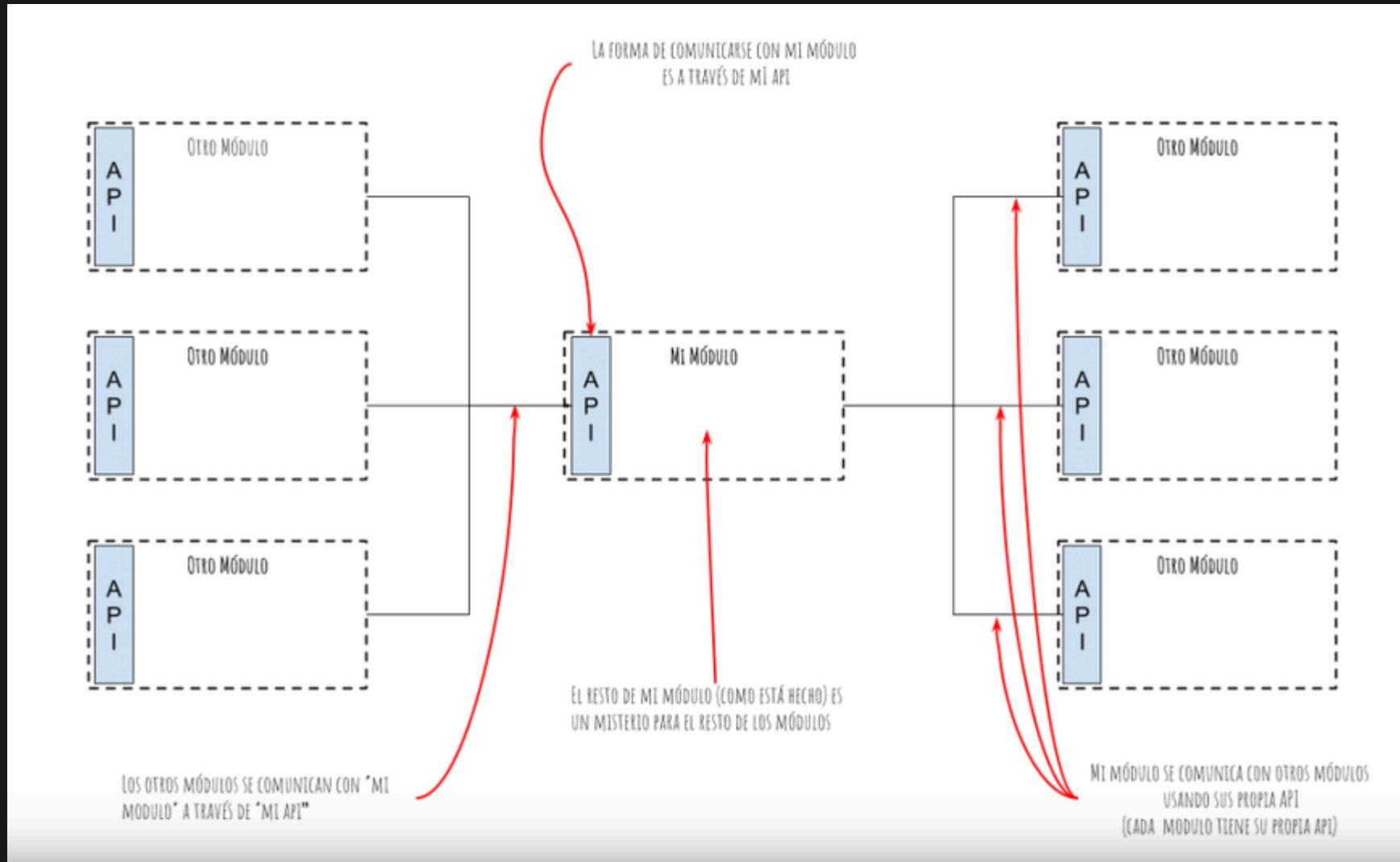
Dividir un problema en partes funcionales independientes es relevante para encapsular datos y operaciones. Es más sencillo mantener la calidad del código. Utilizar un único archivo con todo el código complica el mantenimiento del producto, detectar errores y solucionarlos manteniendo la calidad.

Además de identificar y estar preparado para la detección de errores colaterales. Simplifica el trabajo en equipo. Facilita el reusode código. ¿Para qué hacer la rueda cada vez?

MODULARIZACIÓN²

Lo importante de un módulo es “cómo se usa” más que “como está hecho”. Un módulo puede ser visto como un conjunto de código y datos relacionados lógicamente. Como está implementado es un misterio para el resto del mundo.

MODULARIZACIÓN³



<https://gustavodohara.com/blogangular/los-modulos-javascript-agregarle-plugins-al-viejo-conocido-javascript/>

MODULARIZACIÓN⁴

- El encapsulamiento permite que, mientras el encabezado de la función no cambie, es posible cambiar el código sin que los otros módulos dejen de funcionar.
- Utilizar el código en todas las soluciones que se necesitan facilitando el reuso.
- Si bien es una ventaja para el mantenimiento el uso de los módulos, la administración de los mismos es un punto a trabajar.

MODULARIZACIÓN⁵

La complejidad de las soluciones en JS hizo necesario incluir módulos, que requieren de estrategias de empaquetado y distribución.

MODULARIZACIÓN⁵

Un módulo es como una librería de JS. Posee un conjunto de funciones para ser utilizadas en la aplicación.

Podemos crear nuestros propios módulos. Node posee un conjunto de librerías por defecto, como:

- fs: para manejar el file system
- http o https: para hacer que Node.js actue como un servidor http
- querystring: para manejar querystrings
- url: para parsear las urls
- zlib: para comprimir o descomprimir archivos
- events: para gestionar eventos

MÓDULO HTTP¹

Permite transferir datos sobre el protocolo http.
createServer() es un método para crear un servidor
http La función http.createServer() se ejecuta cada vez
que se intenta acceder al puerto 8080

Es posible:

Agregar un header

Manipular el queryString;

```
myfirst.js
1 var http = require('http');
2
3 http.createServer(function (req, res) {
4   res.writeHead(200, {'Content-Type': 'text/html'});
5   res.end('Seminario JS!');
6 }).listen(8080);
```

https://www.w3schools.com/nodejs/nodejs_http.asp

MÓDULO HTTP²

Permite crear un servidor Web básico.

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

Incluye el módulo http

```
http.createServer(function (request, response) {  
    response.writeHead(200, {'Content-Type': 'text/html'});  
    response.end('Hello World!');  
}).listen(8080);
```

Función del módulo http

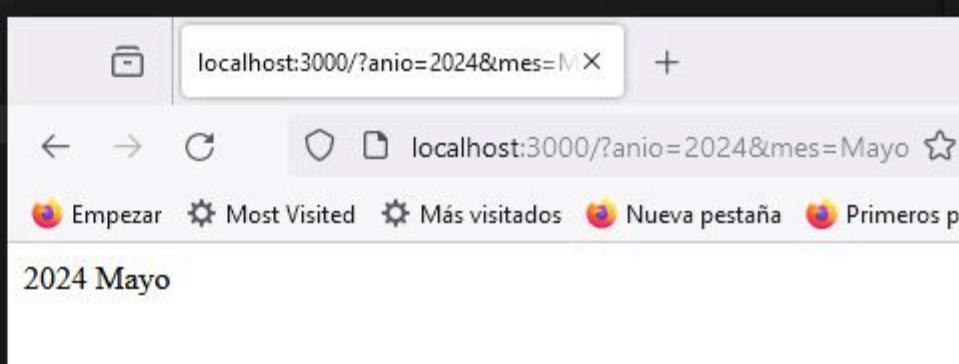
Requerimiento de un
cliente. Objeto
http.IncomingMessage

MÓDULO URL QUERYSTRING¹

url: propiedad del objeto http. IncomingMessage identifica la parte siguiente a la barra en la URL, seguida del nombre de dominio.

Es posible parsearla para analizar los resultados

```
examples > JS server-query-string.js > [o] url
5 Ejecutar server.js y client.js en terminales diferentes con los comandos node servidor.js y node cliente.js respectivamente
6
7 */
8 // Servidor en Node.js
9 const http = require('http');
10 const url = require('url');
11
12 const server = http.createServer((req, res) => {
13   res.writeHead(200, {'Content-Type': 'text/html'});
14   var q = url.parse(req.url,true).query;
15   var txt = q.anio + " " + q_mes;
16   res.end(txt);
17 }).listen(3000);
18
```



The screenshot shows a browser window with the URL `localhost:3000/?anio=2024&mes=Mayo`. The page content displays the string `2024 Mayo`, which is the result of parsing the query parameters from the URL.

MÓDULO URL - MÉTODOS Y PROPIEDADES

	Tipo	Recibe	Retorna
.parse(req, true)	Método	La url del cliente (req)	Un objeto con cada parte de la URL como propiedades
.host	Propiedad		El dominio de la URL
.pathname	Propiedad		La página de la URL
.search	Propiedad		Los parámetros
.query	Propiedad		Retorna un objeto JS con los parámetros

myfirst-url-querystring-parse.js

```
1 var url = require('url');
2 var adr = 'http://localhost:8080/index.htm?anio=2021&mes=mayo';
3 var q = url.parse(adr, true);
4
5 console.log(q.host);
6 console.log(q.pathname);
7 console.log(q.search);
8 var qdata = q.query;
9 console.log(qdata.mes);
```

```
pamadeo@LAPTOP-4E49AE20:~/Desktop$ node myfirst-url-quer
localhost:8080
/index.htm
?anio=2021&mes=mayo
mayo
```

MÓDULO FILESYSTEM - FS

Facilita la gestión de archivos del servidor.

- `require('fs');`

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

Que otros
puede tomar?

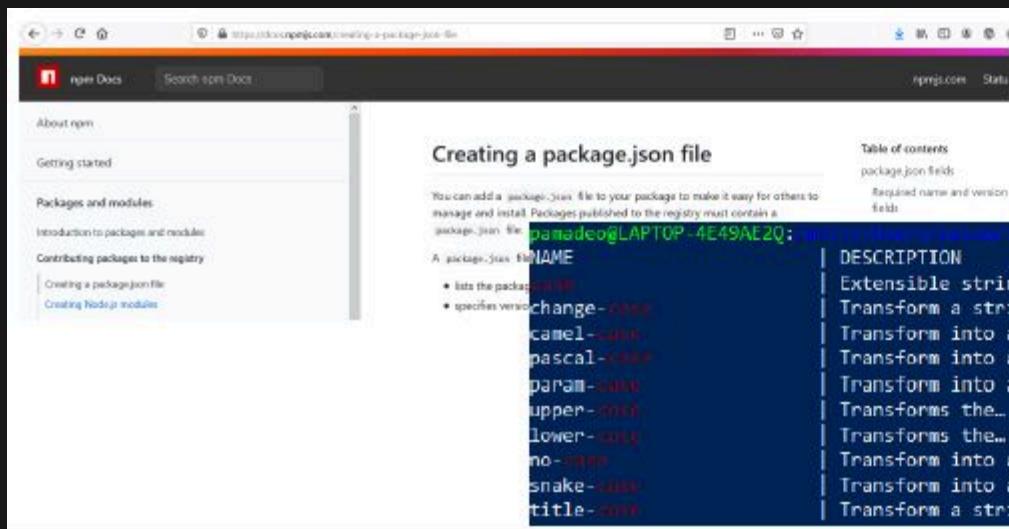
Métodos		
<code>.readFile()</code>	<code>fs.readFile('myfile1.html', function(err, data) { .. })</code>	Leer los archivos del servidor
<code>.appendFile()</code>	<code>fs.appendFile('myfile1.txt', 'Seminario!', function (err) { .. })</code>	Agrega contenido a un archivo y si no existe lo crea un archivo
<code>.open()</code>	<code>fs.open('myfile2.txt', 'w', function (err, file) { .. })</code>	Recibe como argumento el nombre del archivo y un flag. Si es w entonces retorna el archivo abierto para escritura. Si no existe lo crea.
<code>.writeFile()</code>	<code>fs.writeFile('myfile3.txt', 'Seminario JS!', function (err) { .. })</code>	Crea un nuevo archivo
<code>.unlink()</code>	<code>fs.unlink('myfile2.txt', function (err) { .. })</code>	Elimina un archivo
<code>.rename()</code>	<code>fs.rename('mynewfile1.txt', 'myrenamedfile.txt', function (err) { .. })</code>	Renombra un archivo

NPM

www.npmjs.com

npm se instala al instalar node

Un paquete es el conjunto de archivos que se necesitan para ejecutar un módulo o librería.



The screenshot shows a browser window displaying the npm documentation at <https://docs.npmjs.com/creating-a-package-json-file>. The page title is "Creating a package.json file". It includes a sidebar with links like "About npm", "Getting started", "Packages and modules", and "Contributing packages to the registry". The main content area has sections for "A package.json file", "Table of contents", and "Required name and version fields". Below this, there's a table of contents for "package.json fields". On the right side of the page, there's a section titled "Por línea de comandos" which shows a terminal window with the command \$ npm search case and its output.

DESCRIPTION	AUTHOR	DATE	VERSION	KEYWORDS
Extensible string...	=nbubna	2020-03-24	1.6.3	string case camel title upp...
Transform a string...	-blakeembrey	2020-12-02	4.1.2	change case convert transfo...
Transform into a...	-blakeembrey	2020-12-02	4.1.2	camel case camelcase camel...
Transform into a...	-blakeembrey	2020-12-02	3.1.2	pascal case camel capital c...
Transform into a...	-blakeembrey	2020-12-02	3.0.4	param case kebab hyphen das...
Transforms the...	-blakeembrey	2020-12-01	2.0.2	upper case uppercase locale...
Transforms the...	-blakeembrey	2020-12-02	2.0.2	lower case downcase locale...
Transform into a...	-blakeembrey	2020-12-02	3.0.4	no case space lower convert...
Transform into a...	-blakeembrey	2020-12-02	3.0.4	snake case underscore lower...
Transform a string...	-blakeembrey	2020-12-02	3.0.3	title case english capital...

NPMJS

```
pamadeo@LAPTOP-4E49AE2Q:/mnt/c/Users/pulim/JS/jc21-demo$ npm install upper-case
added 2 packages, and audited 3 packages in 11s
found 0 vulnerabilities
pamadeo@LAPTOP-4E49AE2Q:/mnt/c/Users/pulim/JS/jc21-demo$ node demo-files-upper-case.js
```

```
demo-files-upper-case.js
```

```
1 var http = require('http');
2 var uc = require('upper-case');
3
4 http.createServer(function (req, res) {
5   res.writeHead(200, {'Content-Type': 'text/html'});
6   res.write(uc.upperCase("Seminario JS!"));
7   res.end();
8 }).listen(8080); |
```



SEMINARIO JS!

EVENTOS¹

Pilar para la comunicación asincrónica. Forma parte del core de Node.

Permite crear objetos que nombre que reaccionen a los eventos emitidos.

¿Cómo funciona?

1. Creas una instancia EventEmitter.
2. Registras oyentes o listeners para ciertos eventos como on() o addListener()
3. Cuando algo ocurre, emites este evento con emit()
4. Todos los oyentes registrados a ese evento se ejecutarán en forma asincrónica.

EVENTOS²

```
var events = require('events');
var eventEmitter = new events.EventEmitter();
//Create an event handler:
var myEventHandler = function () {
    console.log('Escuche carga de archivo completa!');
}
//Asignar un manejador al evento
eventEmitter.on('carga-completa', myEventHandler);

//Disparo el evento
eventEmitter.emit('carga-completa');
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Cespi\JS\clases\ejemplos-clases-server-side\js21-demo> node myfirst-events.js
Escuche Carga completa!
PS C:\Users\Cespi\JS\clases\ejemplos-clases-server-side\js21-demo> █
```

MÓDULO FORMIDABLE¹

Parsea datos del formulario HTTP.
Facilita la gestión de archivos subidos en la aplicación,
por ejemplo a través de un formulario. Se crea un
objeto files que contiene métodos y propiedades para
la gestión de la subida de archivos.

MÓDULO FORMIDABLE²

```
var http = require('http');
var formidable = require('formidable');
var fs = require('fs');

http.createServer(function (req, res) {
  if (req.url == '/fileupload') {
    var form = new formidable.IncomingForm();
    form.parse(req, function (err, fields, files) {
      var oldpath = files.fileupload.path;
      var newpath = '/files_uploads/' + files.fileupload.r
      fs.rename(oldpath, newpath, function (err) {
        if (err) throw err;
        res.write('Archivo cargado y movido!');
        res.end();
      });
    });
  }
}).listen(3000);
```

UTILIZANDO MÓDULOS PROPIOS

```
<!--myfirstmodule.js-->
exports.myDateTime = function () {
    return Date();
};
```

Y luego

```
<!--using-myfirstmodule.js-->
var http = require('http');
var dt = require('./myfirstmodule');

http.createServer(function (req, res) {
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.write("La fecha actual es: " + dt.myDateTime());
    res.write("<br>");
    res.write(req.url);
    res.end();
}).listen(8080);
```

MÓDULO DE BASE DE DATOS

- Node brinda funciones para trabajar con bases de datos a través de módulos para manipular los datos.
- Es necesario instalar los módulos con npm
- Crear las conexión y realizar consultas.

- MongoDB
- MySQL
- Raspberry Pi

demo_db_connection.js

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

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword"
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
});
```

```
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var query = { address: "Park Lane 38" };
  dbo.collection("customers").find(query).toArray(function(err, result) {
    if (err) throw err;
    console.log(result);
    db.close();
  });
});
```

Create a database called "mydb":

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/mydb";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  console.log("Database created!");
  db.close();
});
```

EXPRESS JS³

Express es el framework más popular de Node para implementar servicios Web.

Agiliza todo el ciclo de desarrollo de software.

Integra motores de renderización de vistas para usar plantillas.

Procesa las peticiones middleware necesarias en el desarrollo Web como cookies, sesiones, parámetros URL, datos post, etc.

REFERENCIAS

Node JS. Entorno de ejecución JavaScript. Wikipedia
Express Framework Los 10 mejores frameworks de desarrollo Agregarle módulos al viejo y conocido JS