

## CLASE 9

# SERVER SIDE - NODE Y EXPRESS

Node JS

# AGENDA

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

# NODE JS



# NODE JS

- Servidor open source. El código está escrito en JS con soporte de algunos módulos en C++.
- El gestor de paquetes NPM flexibiliza y agiliza la generación de código.
- Portable para distintos sistemas operativos.
- Puede crear, eliminar, leer y escribir archivos en el servidor.
- Recolecta datos de un formulario.
- Agrega, elimina y modifica datos de la base de datos.

## NODE JS <sup>2</sup>

- 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 20.13.1

# NODE JS.CARACTERÍSTICAS<sup>1</sup>

Utiliza programación asíncrona  
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 <sup>2</sup>

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.



# MODULARIZACIÓN<sup>1</sup>

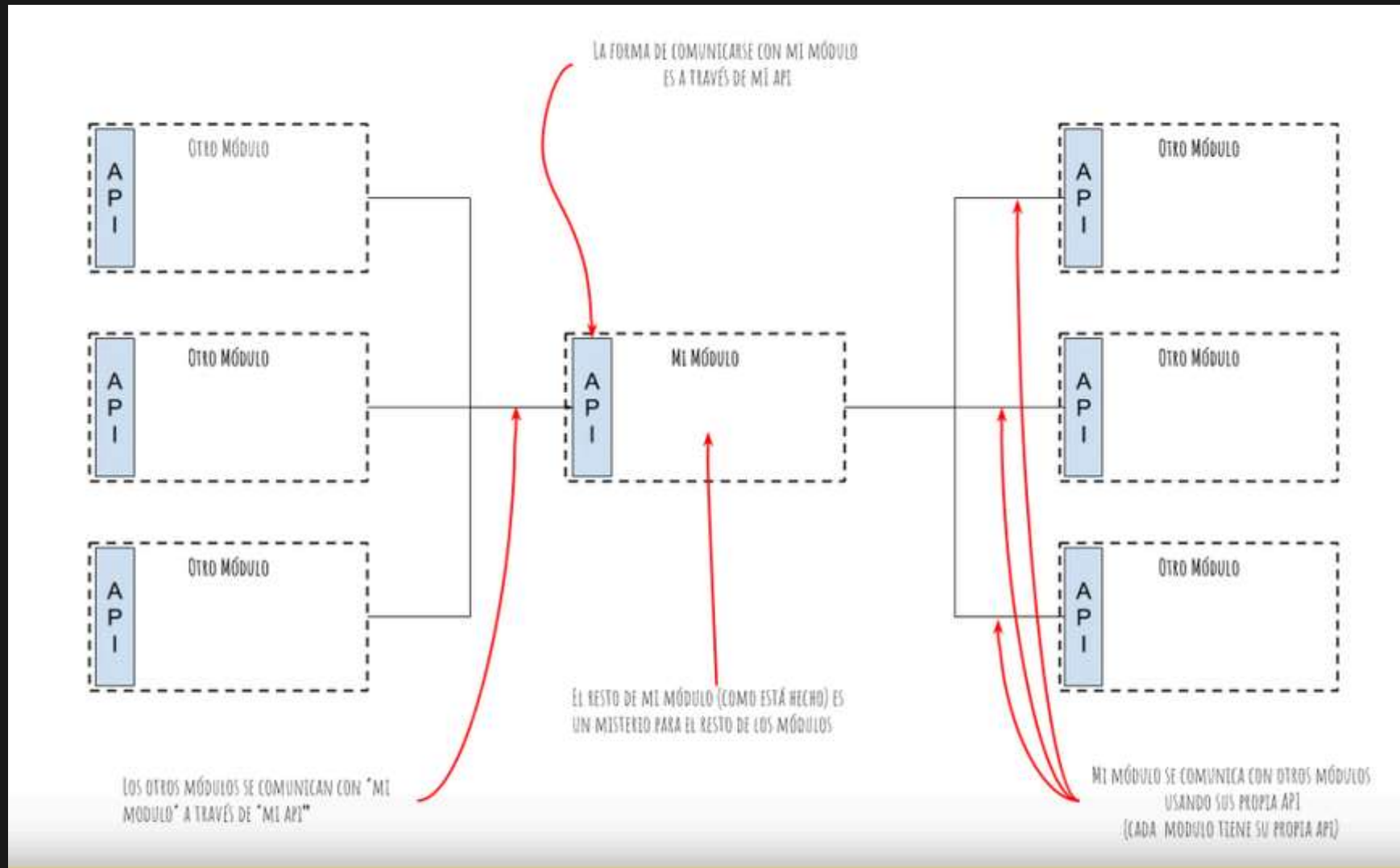
Dividir un problema en partes funcionales independientes es relevante para encapsular datos y operaciones. Es más sencillo mantener la calidad del código. Utilizar un único archivo con todo el código complica el mantenimiento del producto, detectar errores y solucionarlos manteniendo la calidad.

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?

# MODULARIZACIÓN<sup>2</sup>

Lo importante de un módulo es “cómo se usa” más que “cómo 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.

# MODULARIZACIÓN<sup>3</sup>



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

# MODULARIZACIÓN<sup>4</sup>

- 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.
- Reusabilidad: utilizar el código en todas las soluciones que se necesitan.
- Mantenimiento: si bien es una ventaja el uso de los módulos, la administración de los mismos es un punto a trabajar.
- La complejidad de las soluciones en JS hizo necesario incluir módulos, que requieren de estrategias de empaquetado y distribución

# MODULARIZACIÓN<sup>5</sup>

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 actúe como un servidor http
- querystring: para manejar querystrings
- url: para parsear las urls
- zlib: para comprimir o descomprimir archivos
- events: para gestionar eventos



# MÓDULO HTTP<sup>1</sup>

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;

[https://www.w3schools.com/nodejs/nodejs\\_http.asp](https://www.w3schools.com/nodejs/nodejs_http.asp)

```
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);
```

# MÓDULO HTTP<sup>2</sup>

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

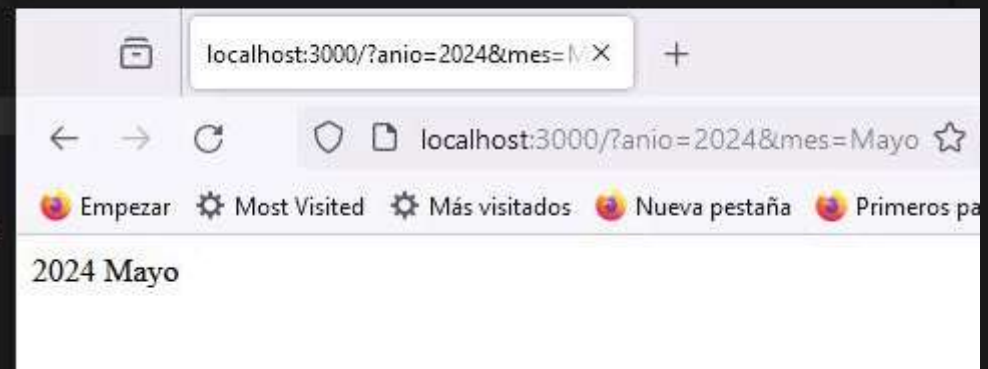
# MÓDULO URL QUERYSTRING<sup>1</sup>

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

Es posible paresearla para analizar los resultados

examples > JS server-query-string.js > 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
```



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

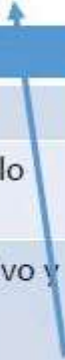
```
pamadeo@LAPTOP-4E49AE2Q: ~$ node myfirst-url-querystring-parse.js  
localhost:8080  
/index.htm  
?anio=2021&mes=mayo  
mayo
```

# 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

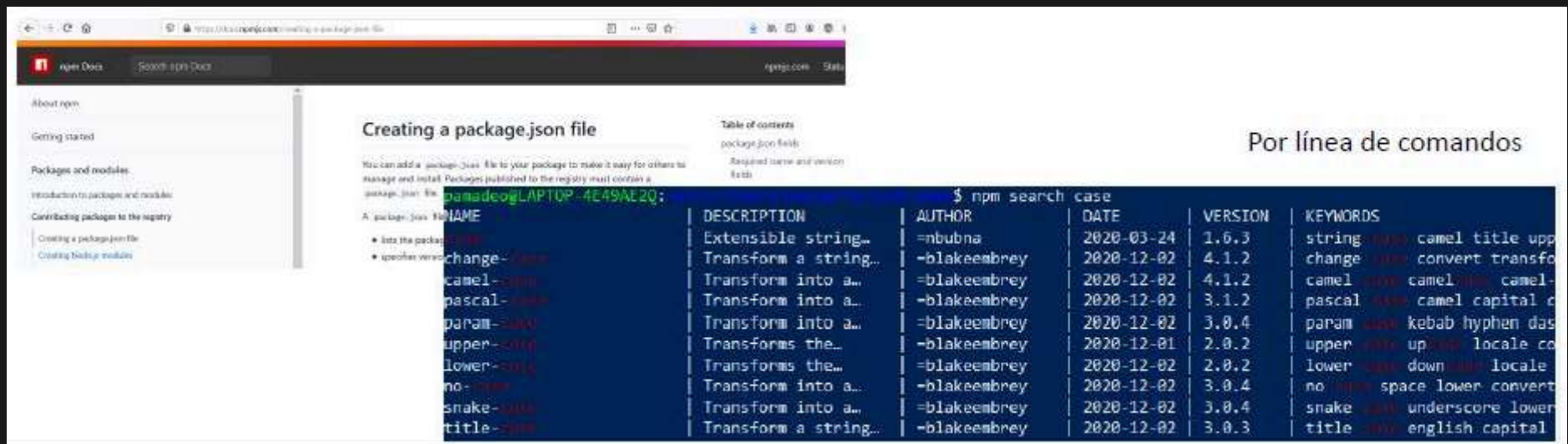


# NPM

[www.npmjs.com](http://www.npmjs.com)

npmse instala al instalar node

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



The screenshot shows the npm website's 'Creating a package.json file' guide. On the right, a terminal window displays the command `$ npm search case` and its output, which is a table of search results.

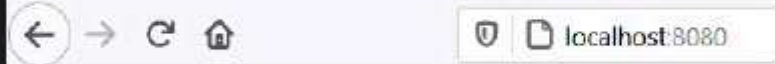
NAME	DESCRIPTION	AUTHOR	DATE	VERSION	KEYWORDS
case	Extensible string...	=nbubna	2020-03-24	1.6.3	string case camel title upp
change-case	Transform a string...	=blakeembrey	2020-12-02	4.1.2	change case convert transfo
camel-case	Transform into a...	=blakeembrey	2020-12-02	4.1.2	camel case camelcase camel-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

# NPMJS

```
pamadeo@LAPTOP-4E49AE2Q: ~/demo $ npm install upper-case
added 2 packages, and audited 3 packages in 11s
found 0 vulnerabilities
pamadeo@LAPTOP-4E49AE2Q: ~/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.toUpperCase("Seminario JS!"));
7   res.end();
8 }).listen(8080);
```



SEMINARIO JS!

# EVENTOS

- El módulo Event permite gestionar objetos que pueden disparar y escuchar eventos.
- Objeto EventEmitter
- Es posible asignar manejadores de eventos a eventos propios.
- emit() : método que dispara un evento.

myfirst-events.js

```
1 var events = require('events');
2 var eventEmitter = new events.EventEmitter();
3
4 //Create an event handler:
5 var myEventHandler = function () {
6   console.log('Escuche un Algo!');
7 }
8
9 //Assign the event handler to an event:
10 eventEmitter.on('algo', myEventHandler);
11
12 //Fire the 'algo' event:
13 eventEmitter.emit('algo');
```

```
pamadeo@LAPTOP-4E49AE2Q: ~$ node myfirst-events.js
Escuche un Algo!
pamadeo@LAPTOP-4E49AE2Q: ~$
```



# MÓDULO FORMIDABLE<sup>1</sup>

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 FORMIDABLE<sup>2</sup>

```
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();      });
    });
  }
}).
```

# 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 la 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 JS <sup>3</sup>

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