

Versión de nodejs v20.8.0

- node --version

```
[kralos]--[!main ≡ ● ]  
[D:\umar\Asignaturas\Sem  
▶ node --version  
v20.8.0
```

JS server_1.js > ...

```
1  const http = require('http');  
2  
3  const hostname = '127.0.0.1';  
4  const port = 3000;  
5  
6  const server = http.createServer((req, res) => {  
7    res.statusCode = 200;  
8    res.setHeader('Content-Type', 'text/plain');  
9    res.end('Hola Mundo');  
10 }); <- #6-10 const server = http.createServer  
11  
12 server.listen(port, hostname, () => {  
13   console.log(`El servidor se está ejecutando en http://${hostname}:${port}/`);  
14 });
```

127.0.0.1:3000

127.0.0.1:3000

Hola Mundo

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
[kralos]--[!main ≡ ● ]  
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs]  
▶ node .\server_1.js  
El servidor se está ejecutando en http://127.0.0.1:3000/  
□
```

nodejs

- Es un framework para implementar operaciones de entrada y salida. Está basado en eventos, streams y construido encima del motor de Javascript V8, que es con el que funciona el Javascript de Google Chrome.
- El hola mundo de nodejs, hi.js

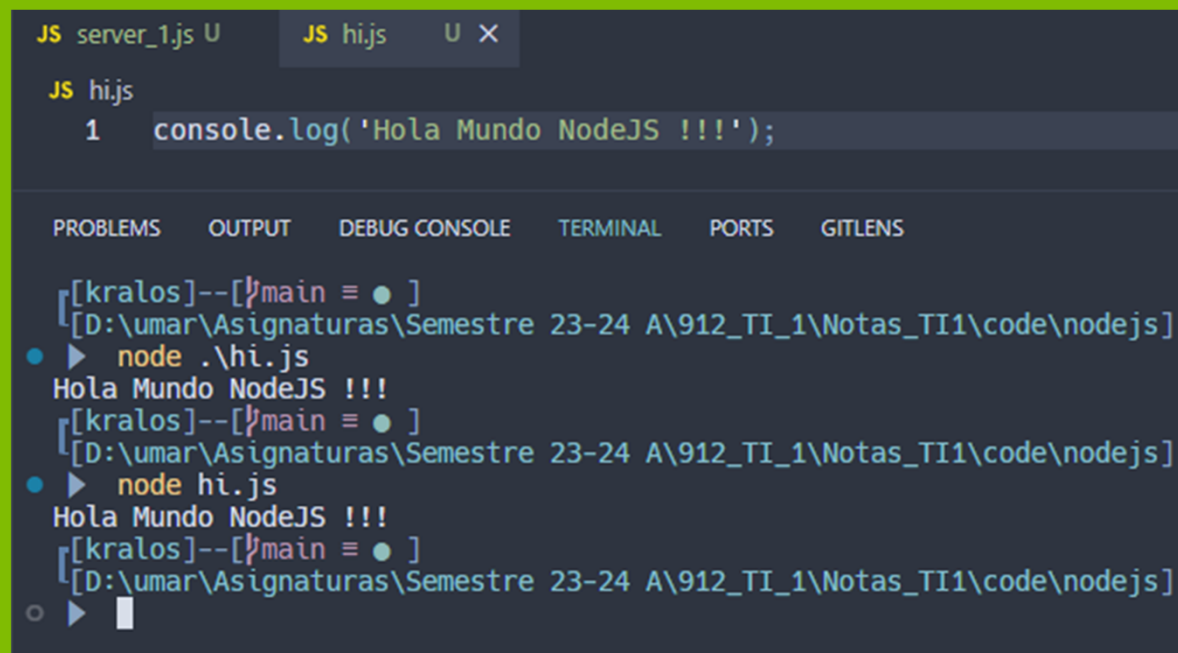
```
console.log('Hola Mundo NodeJS !!!');
```

nodejs

- Nota importante ojo con:

' ' " !

nodejs



The image shows a Visual Studio Code editor window with a file named `hi.js` open. The code in the file is:

```
1 console.log('Hola Mundo NodeJS !!!');
```

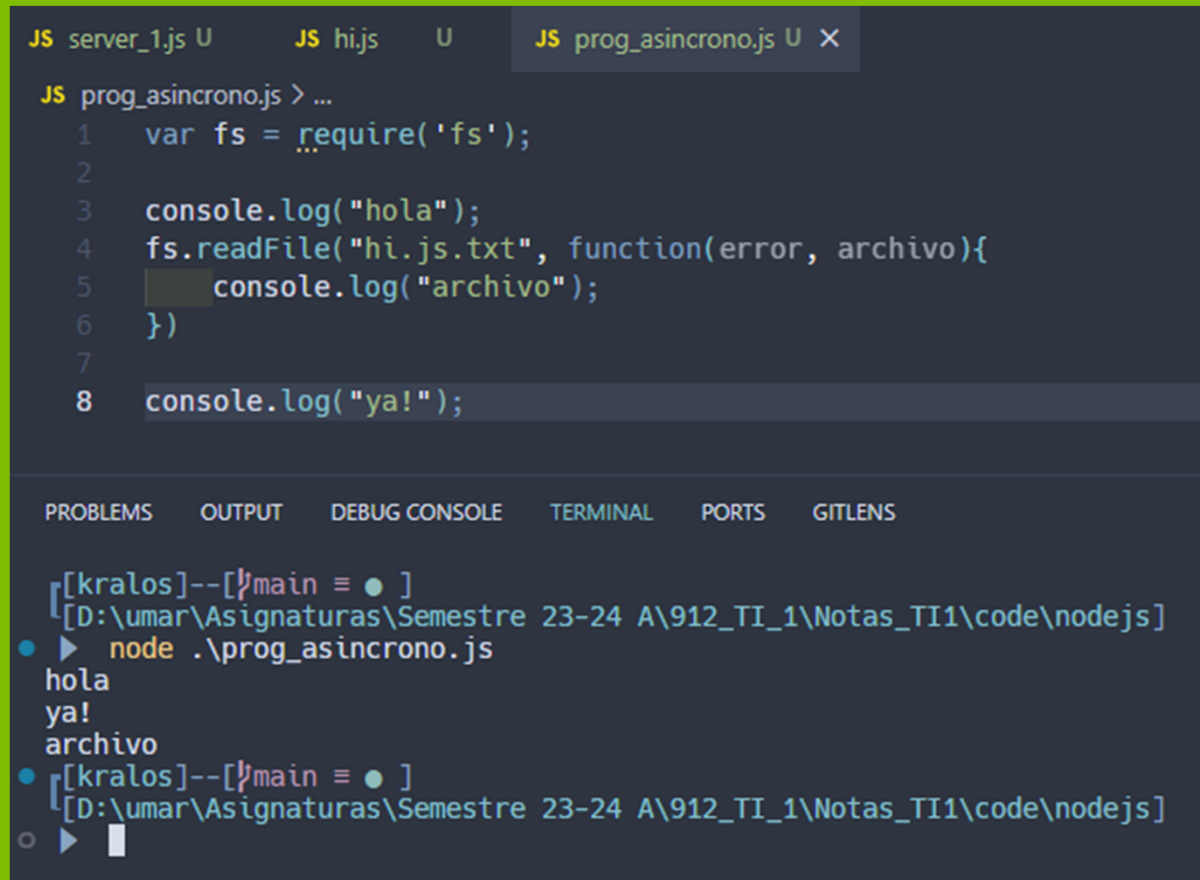
Below the editor, the integrated terminal is open, showing the output of running the script. The terminal has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and GITLENS. The output shows three successful runs of the command `node hi.js`, each resulting in the message `Hola Mundo NodeJS !!!`. The terminal prompt is `[kralos]--[!main ≡ ●]` and the current directory is `[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs]`.

Programación Asíncrona

- La filosofía detrás de Node.JS es hacer programas que no bloqueen la línea de ejecución de código con respecto a entradas y salidas, de modo que los ciclos de procesamiento se queden disponibles durante la espera. Por eso todas las APIs de NodeJS usan callbacks de manera intensiva para definir las acciones a ejecutar después de cada operación I/O, que se procesan cuando las entradas o salidas se han completado.

Programación Asíncrona

- Ejemplo, ver prog_asincrono.js



```
JS server_1.js U    JS hi.js    U    JS prog_asincrono.js U X

JS prog_asincrono.js > ...
1  var fs = require('fs');
2
3  console.log("hola");
4  fs.readFile("hi.js.txt", function(error, archivo){
5      console.log("archivo");
6  })
7
8  console.log("ya!");
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
[kralos]--[?main ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs]
• ▶ node .\prog_asincrono.js
hola
ya!
archivo
• [kralos]--[?main ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs]
○ ▶
```

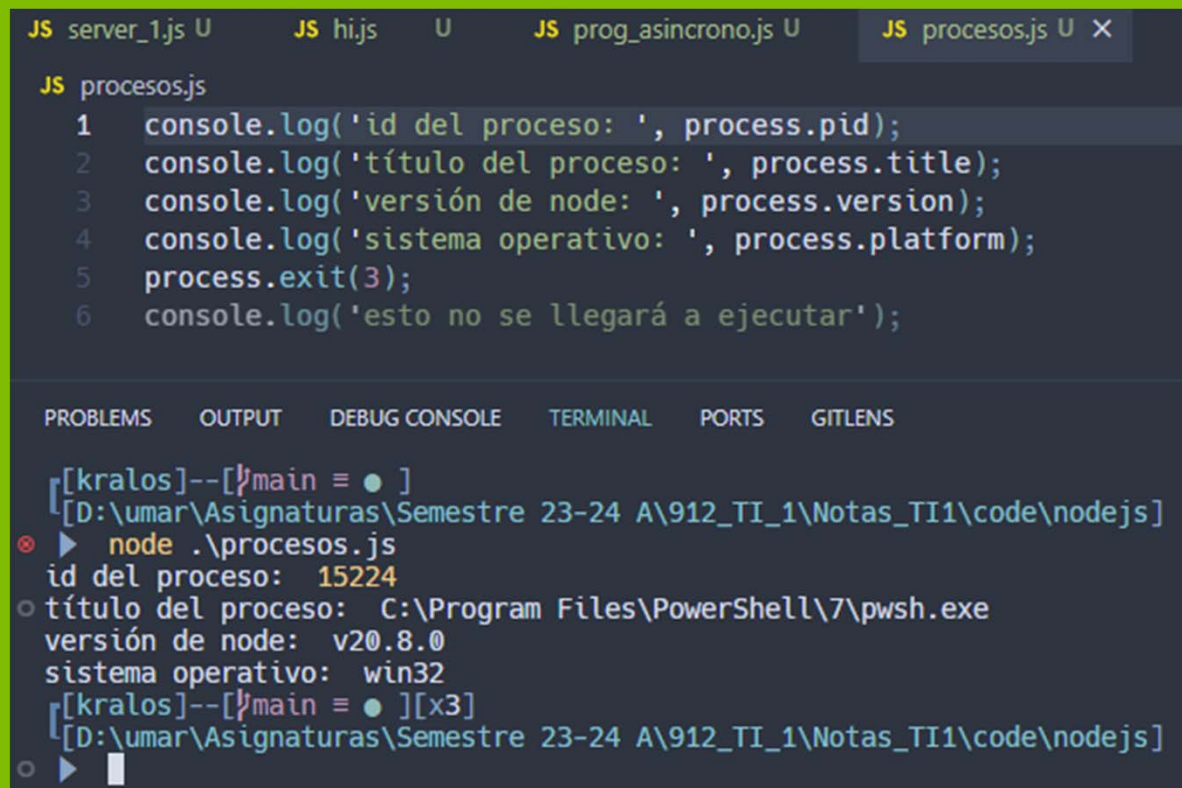
Programación orientada a eventos (POE)

- En Javascript del lado del cliente tenemos objetos como "window" o "document" pero en Node.JS no existen, pues estamos en el lado del servidor.
- Eventos que podremos captar en el servidor serán diferentes, como "uncaughtError", que se produce cuando se encuentra un error por el cual un proceso ya no pueda continuar. El evento "data" es cuando vienen datos por un stream. El evento "request" sobre un servidor también se puede detectar y ejecutar cosas cuando se produzca ese evento.

Single Thread (único hilo)

- Una de las características de NodeJS es su naturaleza "Single Thread". Cuando pones en marcha un programa escrito en NodeJS se dispone de un único hilo de ejecución.
- Esto, en contraposición con otros lenguajes de programación como Java, PHP o Ruby, donde cada solicitud se atiende en un nuevo proceso, tiene sus ventajas. Una de ellas es que permite atender mayor demanda con menos recursos, uno de los puntos a favor de NodeJS.

Single Thread (único hilo)

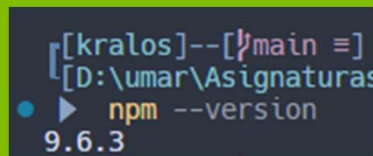


```
JS server_1.js U JS hi.js U JS prog_asincrono.js U JS procesos.js U X
JS procesos.js
1 console.log('id del proceso: ', process.pid);
2 console.log('título del proceso: ', process.title);
3 console.log('versión de node: ', process.version);
4 console.log('sistema operativo: ', process.platform);
5 process.exit(3);
6 console.log('esto no se llegará a ejecutar');

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
[kralos]--[P/main ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs]
node .\procesos.js
id del proceso: 15224
título del proceso: C:\Program Files\PowerShell\7\pwsh.exe
versión de node: v20.8.0
sistema operativo: win32
[kralos]--[P/main ≡ ● ][x3]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs]
```

Módulos en NodeJS

- En NodeJS el código se organiza por medio de módulos. Son como los paquetes o librerías de otros lenguajes como Java. Por su parte, NPM es el nombre del gestor de paquetes (package manager) que usamos en Node JS.
- El gestor de paquetes npm al descargar un módulo lo agrega a un proyecto local. Aunque cabe decir que también existe la posibilidad de instalar los paquetes de manera global en nuestro sistema.

A terminal window with a dark background. The prompt is [kralos]--[main ≡]. The current directory is [D:\umar\Asignaturas]. The command npm --version is entered, and the output is 9.6.3.

```
[kralos]--[main ≡]  
[D:\umar\Asignaturas]  
● ▶ npm --version  
9.6.3
```

Incluir módulos con "require"

- Javascript nativo no da soporte a los módulos. Esto es algo que se ha agregado en NodeJS y se realiza con la sentencia `require()`, que está inspirada en la variante propuesta por CommonJS.
- La instrucción `require()` recibe como parámetro el nombre del paquete que queremos incluir e inicia una búsqueda en el sistema de archivos, en la carpeta "`node_modules`" y sus hijos, que contienen todos los módulos que podrían ser requeridos.

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

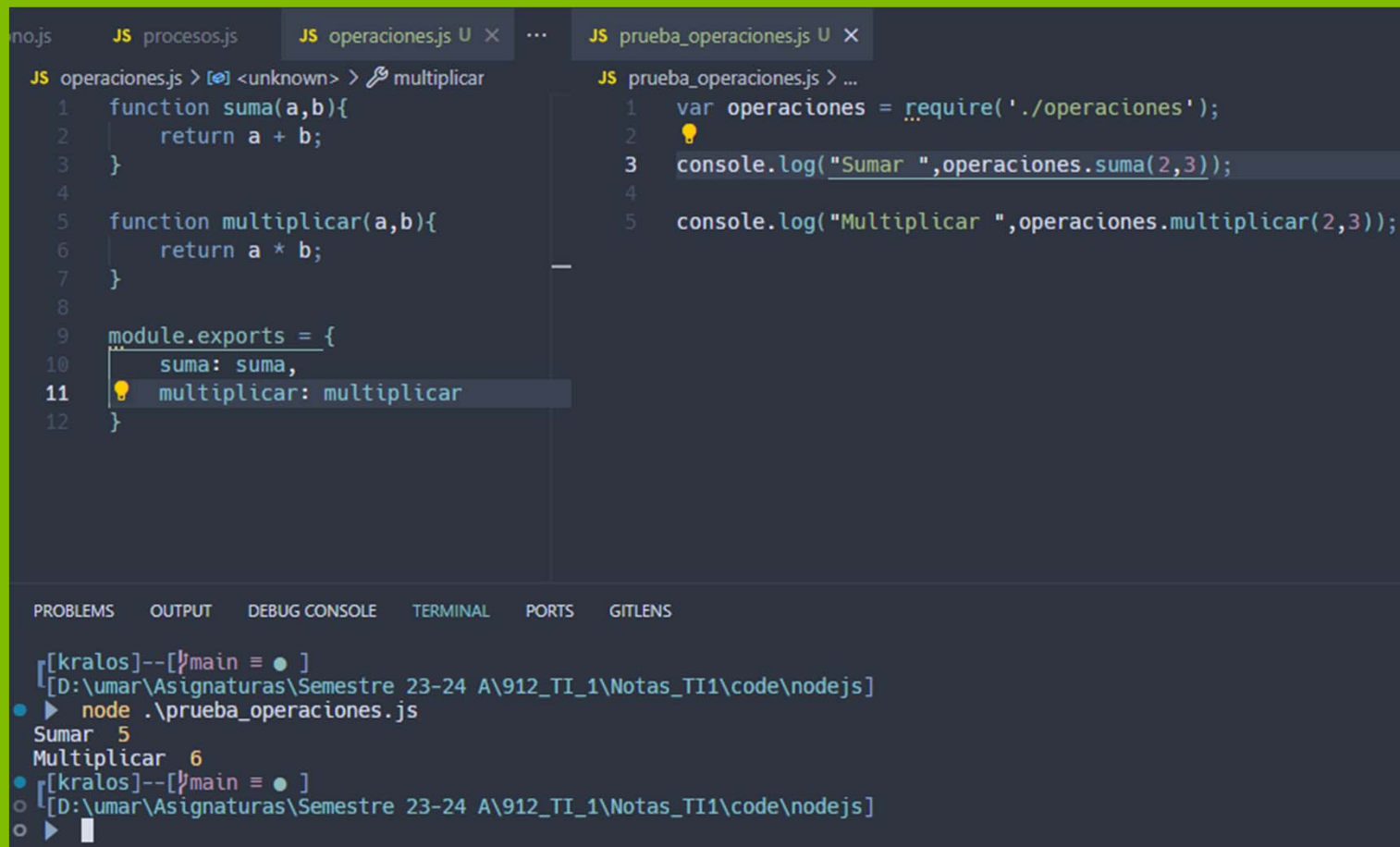
Módulos en NodeJS

- Existen distintos módulos que están disponibles de manera predeterminada en cualquier proyecto NodeJS y que por tanto no necesitamos traernos previamente a local mediante el gestor de paquetes npm.
- Esos toman el nombre de "Módulos nativos" y ejemplos de ellos son "http" para hacer un servidor web y "fs" para el acceso al sistema de archivos.

Exportando módulos en NodeJS

- Para exportar nuestros propios módulos usamos `module.exports`. Escribimos el código de nuestro módulo, con todas las funciones locales que queramos, luego hacemos un `module.exports = {}` y entre las llaves colocamos todo aquello que queramos exportar.

Módulos en NodeJS



```
no.js  JS procesos.js  JS operaciones.js U X  ...  JS prueba_operaciones.js U X

JS operaciones.js > [?] <unknown> > multiplicar
1  function suma(a,b){
2      return a + b;
3  }
4
5  function multiplicar(a,b){
6      return a * b;
7  }
8
9  module.exports = {
10     suma: suma,
11     multiplicar: multiplicar
12 }

JS prueba_operaciones.js > ...
1  var operaciones = require('./operaciones');
2
3  console.log("Sumar ",operaciones.suma(2,3));
4
5  console.log("Multiplicar ",operaciones.multiplicar(2,3));

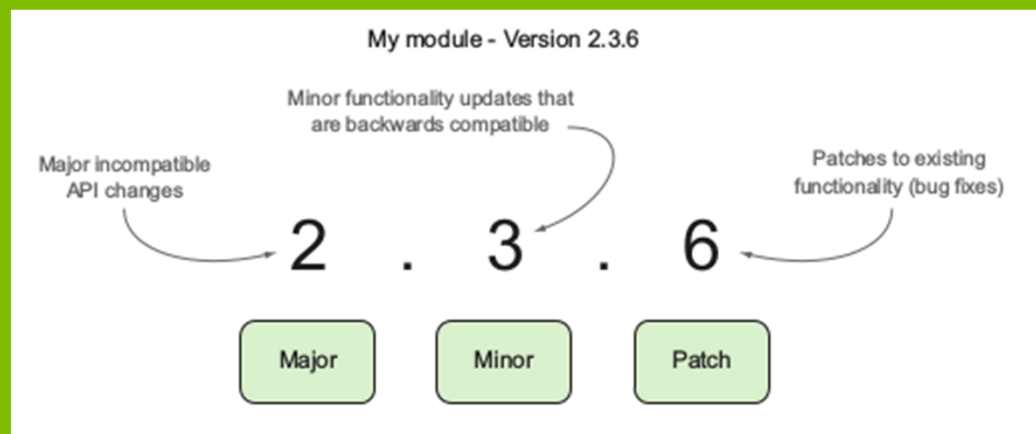
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

[kralos]--[?main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs]
● ▶ node .\prueba_operaciones.js
Sumar 5
Multiplicar 6
● [kralos]--[?main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs]
○ ▶
```

Comando npm

- Es un comando que funciona desde la línea de comandos de NodeJS. Por tanto lo tenemos que invocar con npm seguido de la operación que queramos realizar.
 - npm install paquete -g - Instala paquete globalmente.
 - npm install paquete - Instala el paquete de forma local.
 - npm uninstall paquete -g - Desinstala paquete global
 - npm uninstall paquete - Desinstala paquete local

Módulos en NodeJS



Comando npm

- `npm install package@0.0.0` - Instala una versión específica
- `npm list -g --depth=0` - Lista los paquetes globales instalados
- `npm list --depth=0` - Lista los paquetes locales instalados
- `npm view paquete versión` - Muestra la versión del paquete
- `npm search package` - Busca un paquete
- `npm update -g` - Actualiza los paquetes globales
- `npm update` - Actualiza los paquetes locales
- `npm outdated -g` - Lista los paquetes globales anticuados
- `npm outdated` - Lista los paquetes locales anticuados
- `npm -v` - Muestra la versión de npm que está instalada
- `npm install -g npm-check-updates` - instala paquete para chequear actualizaciones

Otro uso importante de NPM

- Te ayuda a acelerar la fase de desarrollo mediante el uso de dependencias preconstruidas (Crear un archivo package.json).
- npm init
 - El comando init se utiliza para inicializar un proyecto. Crea un archivo package.json.
 - Al ejecutar npm init, se te pedirá que proporciones cierta información sobre el proyecto que estás inicializando.
 - Para saltarte el proceso de proporcionar la información puedes simplemente ejecutar el comando npm init -y.

Otro uso importante de NPM

The screenshot shows a Visual Studio Code editor with a project named 'proyecto_0'. The file explorer on the left lists several JavaScript files: 'package.json', 'hijs', 'operaciones.js', 'procesos.js', 'prog_asincrono.js', 'prueba_operaciones.js', and 'server_1.js'. The 'package.json' file is selected and its content is displayed in the editor. The content is a JSON object with the following properties: 'name' (proyecto_0), 'version' (1.0.0), 'description' (empty string), 'main' (index.js), 'scripts' (a test script that echoes an error message and exits with code 1), 'keywords' (empty array), 'author' (empty string), and 'license' (ISC). Below the editor, the 'TERMINAL' tab is active, showing the command 'npm init -y' and its output, which is the same JSON object shown in the editor.

```
proyecto_0 > package.json > ...
1 {
2   "name": "proyecto_0",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC"
12 } <- #1-12 { "name": "proyecto_0", "version": "1.0.0", "description": ""...
13
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
[kralos]--[main ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_0]
● ▶ npm init -y
Wrote to D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_0\package.json:

{
  "name": "proyecto_0",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Otro uso importante de NPM

- Si existe el archivo package.json
- **npm install** Instalar paquetes, descritos en el archivo package.json , de forma global o local.
- **npm uninstall** Desinstalar paquetes, descritos en el archivo package.json , de forma global o local.

Otro uso importante de NPM

- **npm update**

- Utiliza este comando para actualizar paquetes a su última versión.

- **npm restart**

- Se utiliza para reiniciar un paquete o proyecto.

- **npm start**

- Se utiliza para iniciar un paquete o proyecto.

- **npm stop**

- Se utiliza para detener la ejecución de un paquete o proyecto.

Express

- Es el framework más usado en NodeJS, que nos facilitará la creación de servidores web en muy pocos minutos, personalizados según nuestras necesidades.
- Permitirá no solo servir archivos estáticos, sino atender todo tipo de solicitudes complejas que impliquen realizar cualquier tipo de acción, configurar rutas de manera potente, trabajar de detalladamente con las cabeceras del HTTP y las respuestas, etc. Incluso desarrollar tu propio API REST de una manera más o menos sencilla.

Express

```
EXPLORER: PROYECTO_0
> node_modules
package-lock.json
package.json

package.json M X
package.json > ...
You, 1 second ago | 1 author (You)
You, 3 days ago • update notas ...
1 {
2   "name": "proyecto_0",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC",
12  "dependencies": {
13    "express": "^4.18.2"
14  }
15 } <- #1-12 { "name": "proyecto_0", "version": "1.0.0", "description": ""

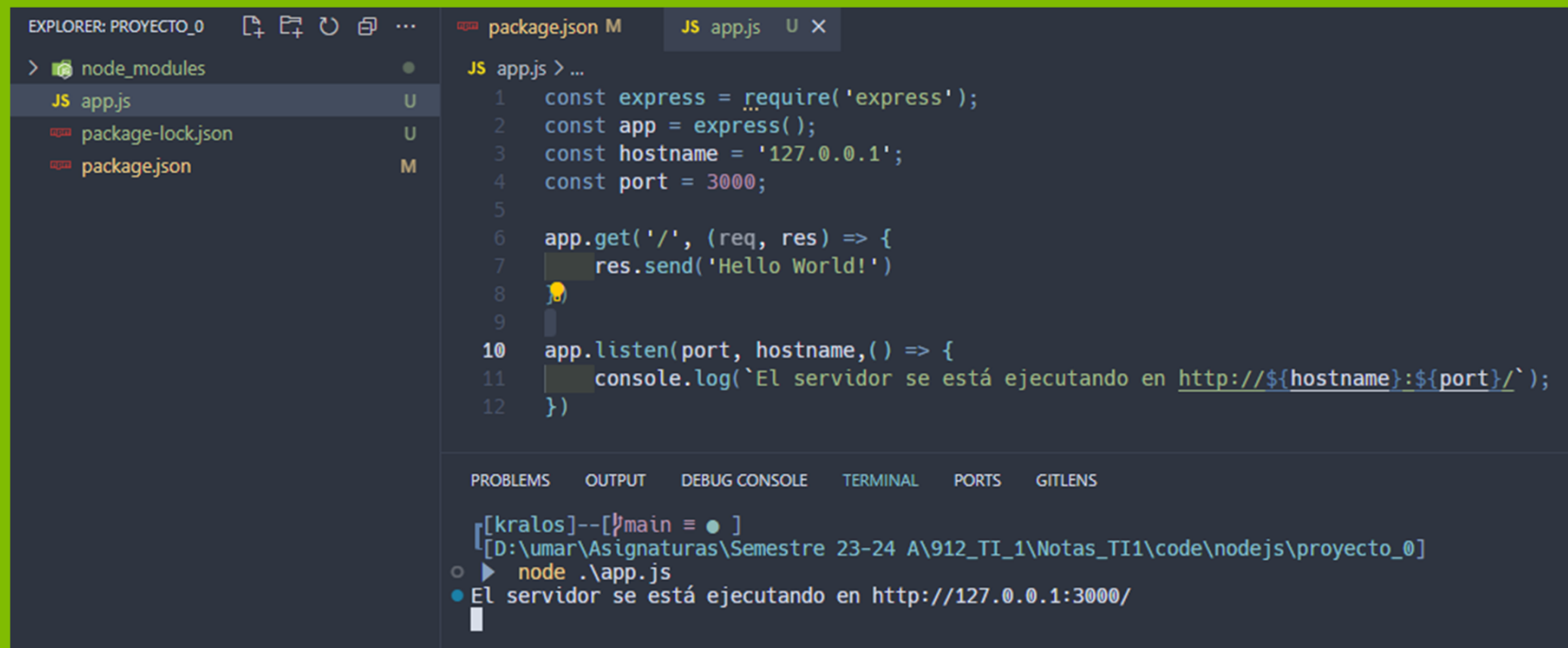
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
[kralos]--[!main =]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_0]
• ▶ npm install express

added 58 packages, and audited 59 packages in 9s

9 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
[kralos]--[!main = • ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_0]
○ ▶
```

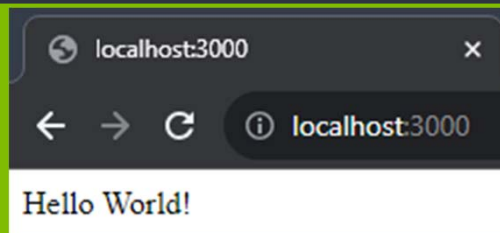

Express



The screenshot shows the Visual Studio Code editor interface. On the left, the Explorer sidebar displays the file structure of a project named 'PROYECTO_0', including 'node_modules', 'app.js', 'package-lock.json', and 'package.json'. The main editor area shows the 'app.js' file with the following code:

```
1 const express = require('express');
2 const app = express();
3 const hostname = '127.0.0.1';
4 const port = 3000;
5
6 app.get('/', (req, res) => {
7   res.send('Hello World!')
8 })
9
10 app.listen(port, hostname, () => {
11   console.log(`El servidor se está ejecutando en http://${hostname}:${port}/`);
12 })
```

Below the code editor, the terminal window is open, displaying the output of the command 'node .\app.js'. The output shows the server starting and listening on 'http://127.0.0.1:3000/'.



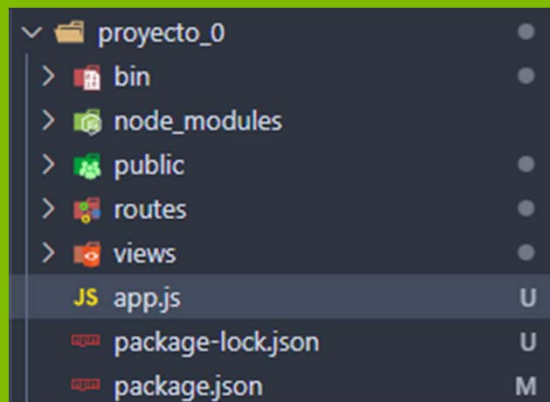
Modulo {express-generator}

- Para crear rápidamente un esqueleto de aplicación.

```
[kralos]--[!main ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_0]
○ ▶ npx express-generator

warning: the default view engine will not be jade in future releases
warning: use '--view=jade' or '--help' for additional options

destination is not empty, continue? [y/N] █
```



```
destination is not empty, continue? [y/N] y

create : .
create : ./package.json
create : ./app.js
create : ./public/javascripts
create : ./public/images
create : ./public/stylesheets
create : ./public/stylesheets/style.css
create : ./public
create : ./routes
create : ./routes/index.js
create : ./routes/users.js
create : ./views
create : ./views/index.jade
create : ./views/layout.jade
create : ./views/error.jade
create : ./bin
create : ./bin/www

install dependencies:
> cd . && npm install

run the app:
> SET DEBUG=proyecto-0:* & npm start
```

Modulo {express-generator}

- **npx** es también una herramienta CLI cuyo propósito es facilitar la instalación y la gestión de las dependencias alojadas en el registro npm.
- Por ejemplo, tenemos en nuestro proyecto “ESLint” antes teníamos que buscar el binario en node_modules.

```
./node_modules/.bin/eslint yourfile.js |
```

Modulo {express-generator}

- Ahora npx lo buscará por nosotros.

```
npx eslint yourfile.js
```

- En nuestro proyecto instala y ejecuta express-generator, y como podemos ver reemplaza y agrega código.

Modulo {express-generator}

```
JS app.js U package.json M X
code > nodejs > proyecto_0 > package.json > {} dependencies
You, 1 second ago | 1 author (You)
1 {
2   "name": "proyecto-0",
3   "version": "0.0.0",
4   "private": true,
5   "scripts": {
6     "start": "node ./bin/www"
7   },
8   "dependencies": {
9     "body-parser": "~1.16.0",
10    "cookie-parser": "~1.4.3",
11    "debug": "~2.6.0",
12    "express": "~4.14.1",
13    "jade": "~1.11.0",
14    "morgan": "~1.7.0",
15    "serve-favicon": "~2.3.2"
16  } <- #8-16 "dependencies": You, 1 se
```

Modulo {express-generator}

```
[kralos]--[?main ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_0]
• ▶ npm install
npm WARN deprecated transformers@2.1.0: Deprecated, use jstransformer
npm WARN deprecated constantinople@3.0.2: Please update to at least constantinople 3.1.1
npm WARN deprecated jade@1.11.0: Jade has been renamed to pug, please install the latest version of pug instead of jade
○
added 61 packages, removed 14 packages, changed 26 packages, and audited 112 packages in 9s

19 vulnerabilities (1 low, 1 moderate, 11 high, 6 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

Luego de muchos “npm audit fix –force”

```
• ▶ npm audit fix --force
npm WARN using --force Recommended protections disabled.

up to date, audited 86 packages in 1s

10 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
[kralos]--[?main ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_0]
○ ▶
```

Modulo {express-generator}

- Ejecutando la App

- CMD

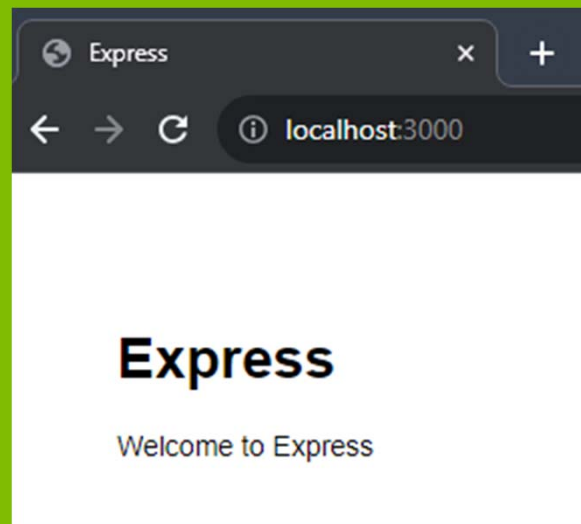
```
> set DEBUG=app:* & npm start
```

- PowerShell

```
PS> $env:DEBUG='myapp:*'; npm start
```

Modulo {express-generator}

```
[kralos]--[?main ≡ ●]  
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_0]  
▶ $env:DEBUG='app:*'; npm start  
  
> proyecto-0@0.0.0 start  
> node ./bin/www  
  
GET / 200 57.282 ms - 170  
GET /stylesheets/style.css 200 29.348 ms - 111  
□
```



Referencias

- *Manual de NodeJS.* (n.d.). Desarrolloweb.com. Retrieved October 9, 2023, from <https://desarrolloweb.com/manuales/manual-nodejs.html>
- *Nodejs — Chuletas.* (n.d.). Github.io. Retrieved October 10, 2023, from <https://jolav.github.io/chuletas/nodejs/>