

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

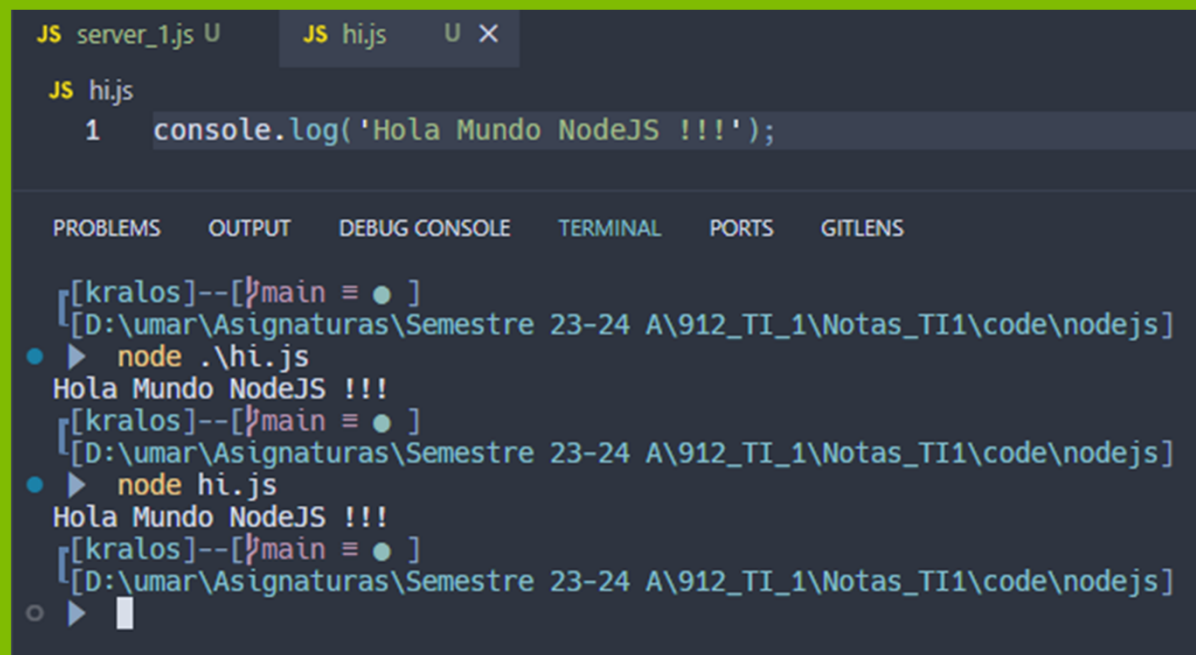
```
console.log('Hola Mundo NodeJS !!!');
```

nodejs

- Nota importante ojo con:



nodejs



The image shows a screenshot of the Visual Studio Code (VS Code) editor interface. At the top, there are two tabs: 'server_1.js U' and 'hi.js U X'. The 'hi.js' tab is active, displaying a single line of JavaScript code: `1 console.log('Hola Mundo NodeJS !!!');`. Below the editor, the 'TERMINAL' pane is open, showing the output of running the script. The terminal output consists of three identical blocks, each starting with a prompt `[kralos]--[!main ≡ ●]` followed by the directory path `[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs]`. The first block shows the command `node .\hi.js` being executed, resulting in the output `Hola Mundo NodeJS !!!`. The second block shows the command `node hi.js` being executed, also resulting in `Hola Mundo NodeJS !!!`. The third block shows the command `node` being executed, which appears to be incomplete or a placeholder. The terminal also shows a cursor at the end of the third command line.

```
JS server_1.js U JS hi.js U X
JS hi.js
1 console.log('Hola Mundo NodeJS !!!');

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

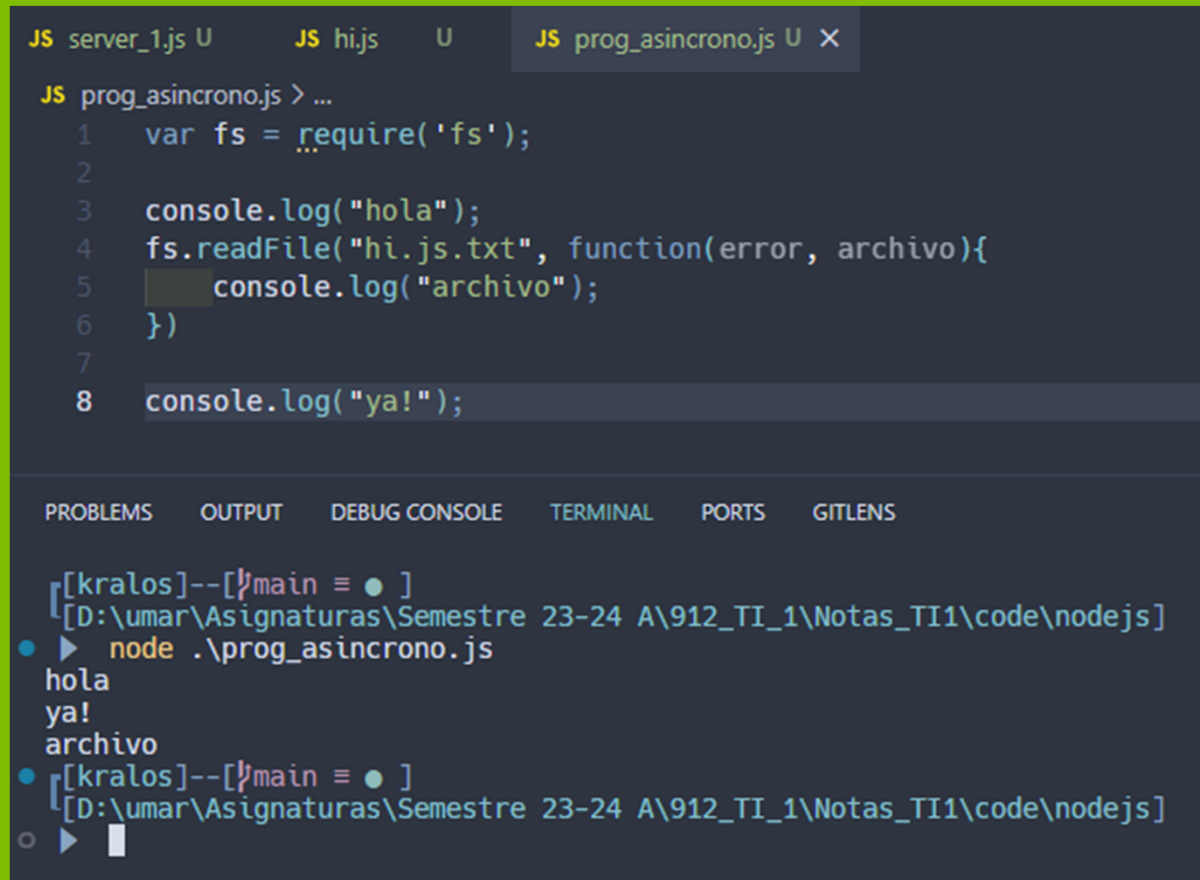
[kralos]--[!main ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs]
● ▶ node .\hi.js
Hola Mundo NodeJS !!!
[kralos]--[!main ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\code\nodejs]
● ▶ node hi.js
Hola Mundo NodeJS !!!
[kralos]--[!main ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\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]
○ ▶
```

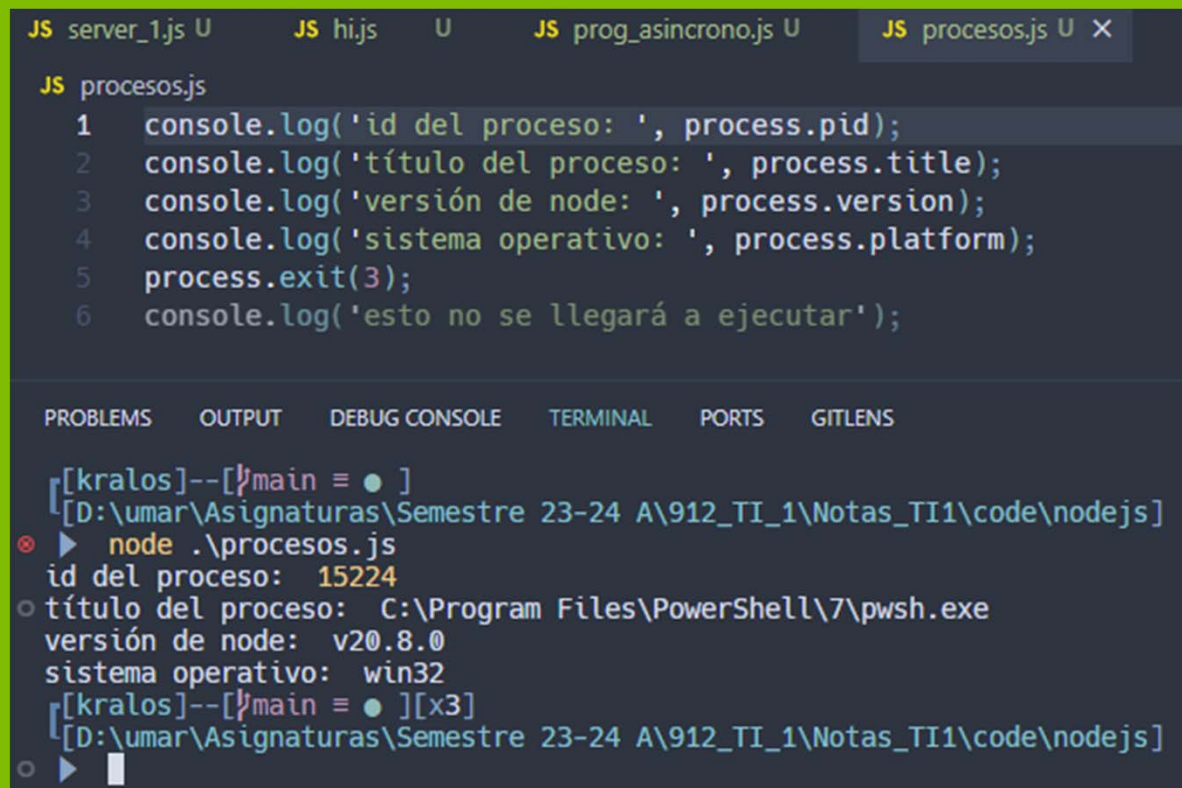
Programación orientada a eventos (POE)

- 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)

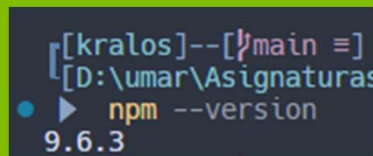


```
JS server_1.js U    JS hi.js U    JS prog_asincrono.js U    JS procesos.js U X
JS procesos.js
1 console.log('id del proceso: ', process.pid);
2 console.log('título del proceso: ', process.title);
3 console.log('versión de node: ', process.version);
4 console.log('sistema operativo: ', process.platform);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

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS
[kralos]--[Pmain ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs]
❶ ▶ 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]--[Pmain ≡ ● ][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 screenshot of a terminal window with a dark background. The prompt is [kralos]--[main ≡]. The current directory is [D:\umar\Asignaturas]. The command npm --version has been executed, 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");
```

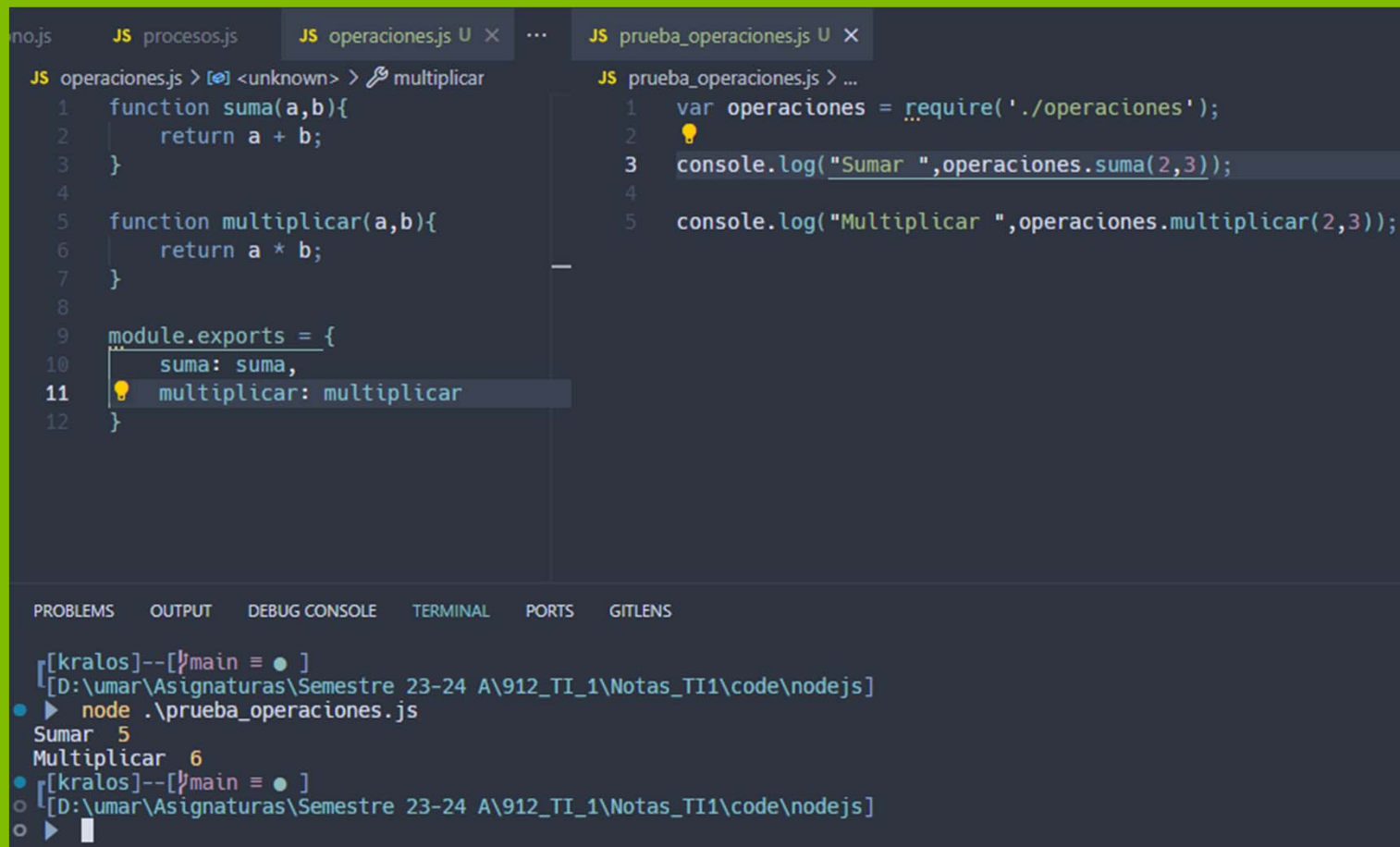
Módulos en NodeJS

- 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));

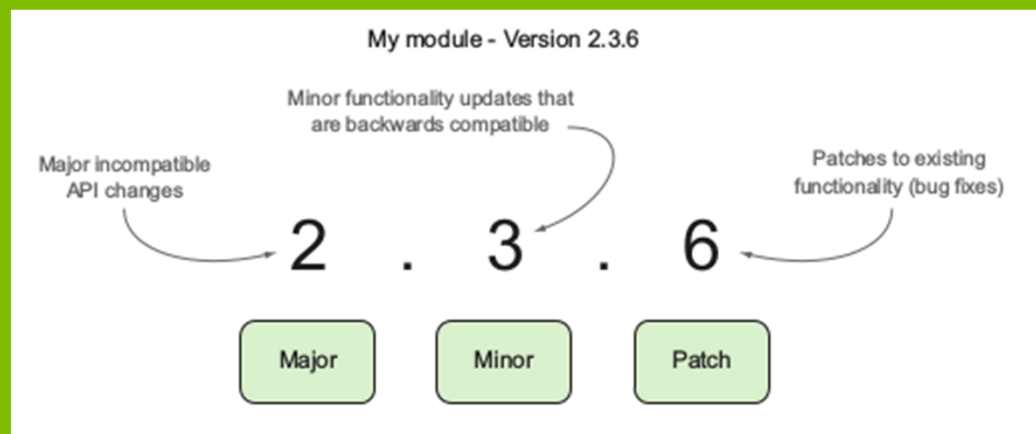
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

[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



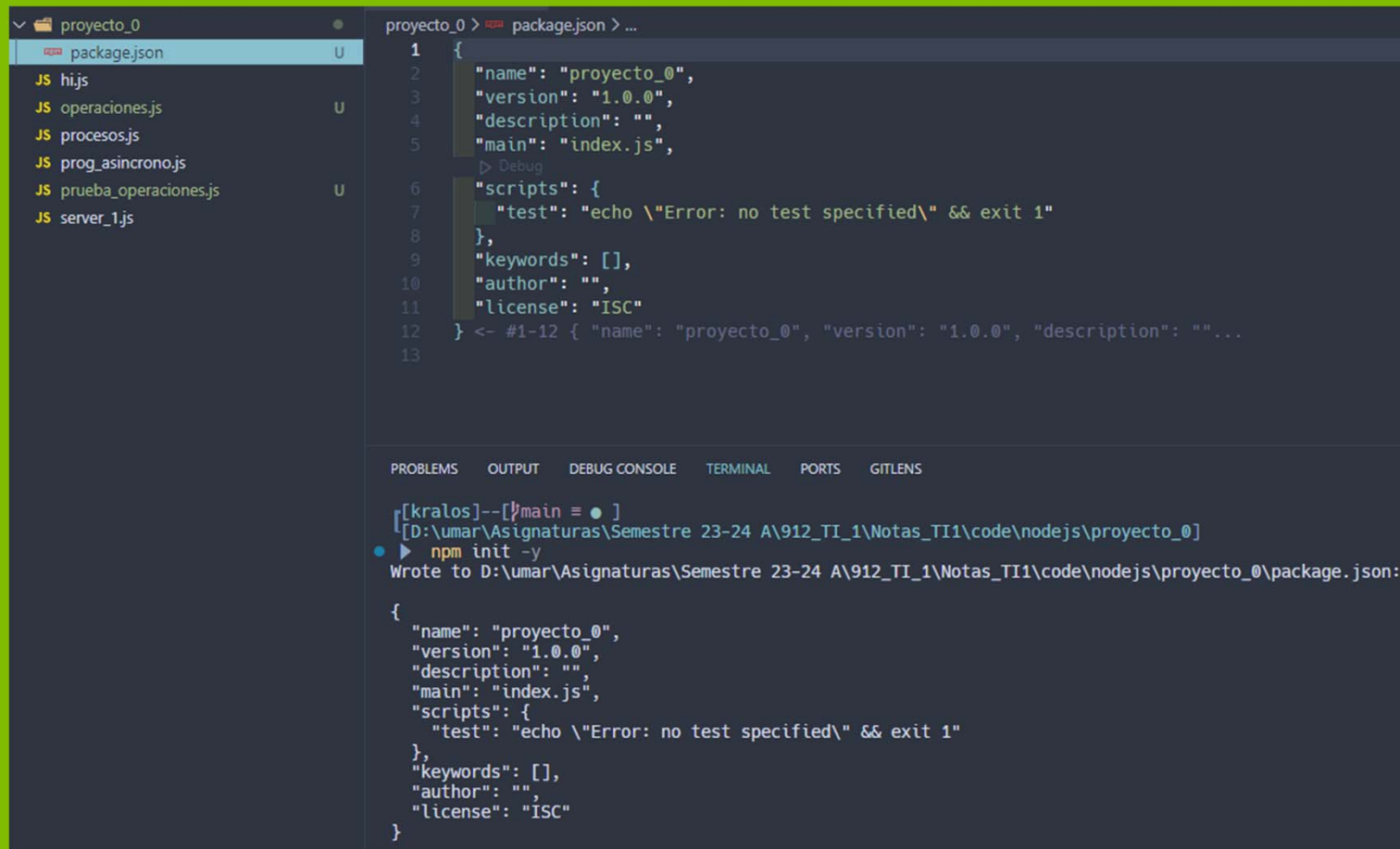
Comando npm

- `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



```
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.

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/>