

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

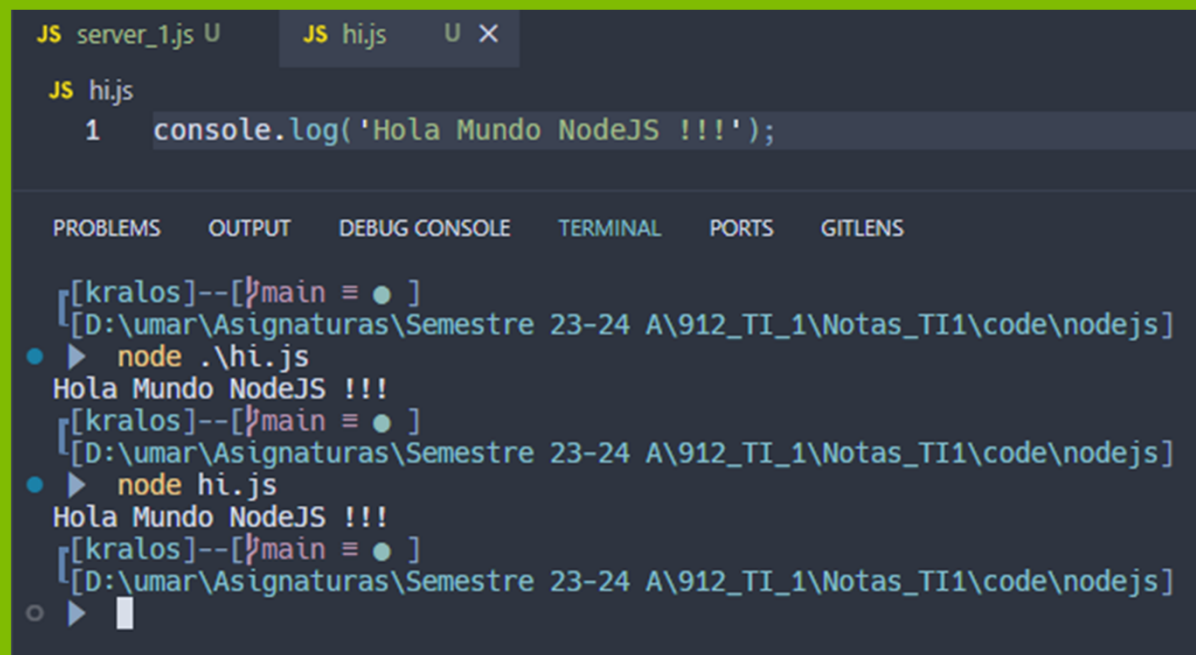
```
console.log('Hola Mundo NodeJS !!!');
```

nodejs

- Nota importante ojo con:



nodejs



The image shows a Visual Studio Code editor window with a dark theme. At the top, there are two tabs: 'server_1.js' and 'hi.js'. The 'hi.js' tab is active, showing a single line of JavaScript code: `console.log('Hola Mundo NodeJS !!!');`. Below the editor, there is a panel with tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', 'PORTS', and 'GITLENS'. The 'TERMINAL' tab is selected, displaying the output of running the script. The output shows three separate executions of the command `node hi.js`, each resulting in the message 'Hola Mundo NodeJS !!!'. The terminal also shows the current directory path: `D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs`.

```
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

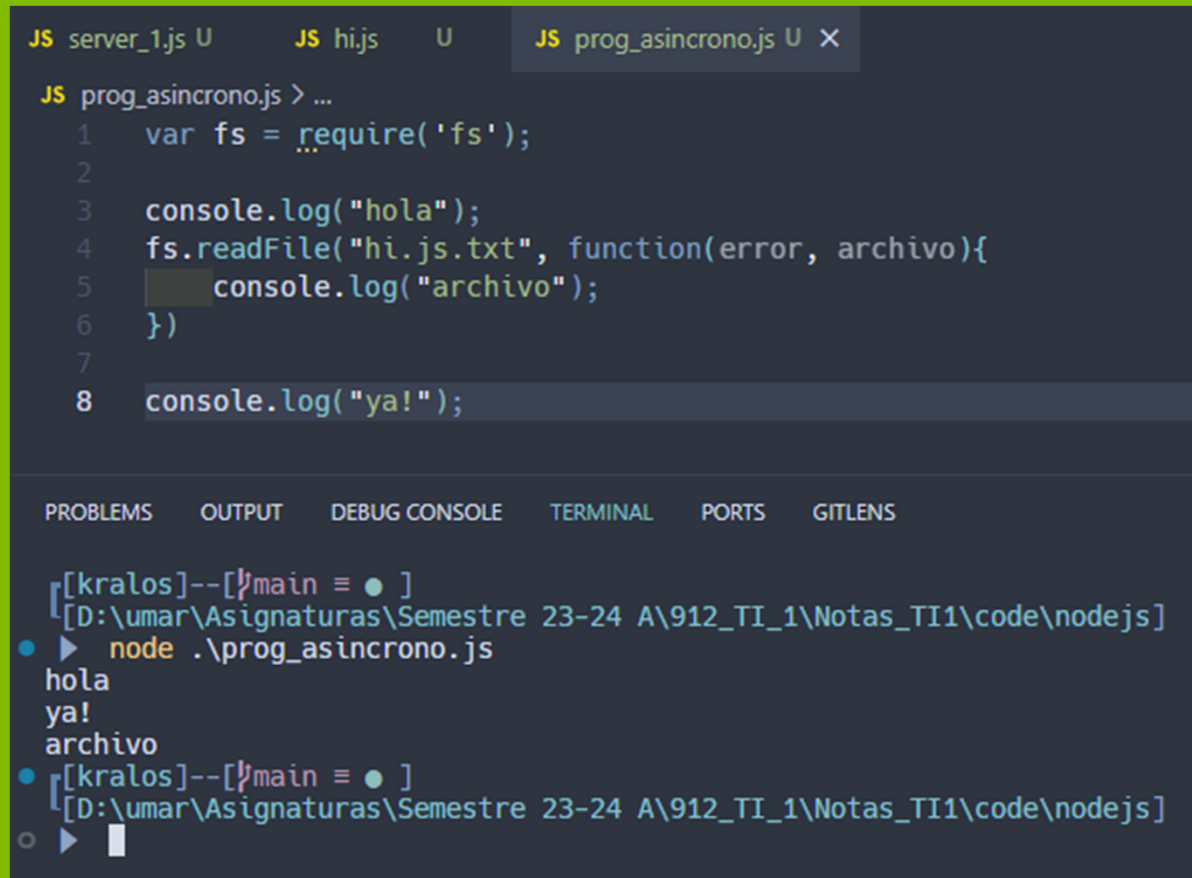
```
[kralos]--[!main ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs]
● ▶ node .\hi.js
Hola Mundo NodeJS !!!
[kralos]--[!main ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\code\nodejs]
● ▶ node hi.js
Hola Mundo NodeJS !!!
[kralos]--[!main ≡ ● ]
[D:\umar\Asignaturas\Semestre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

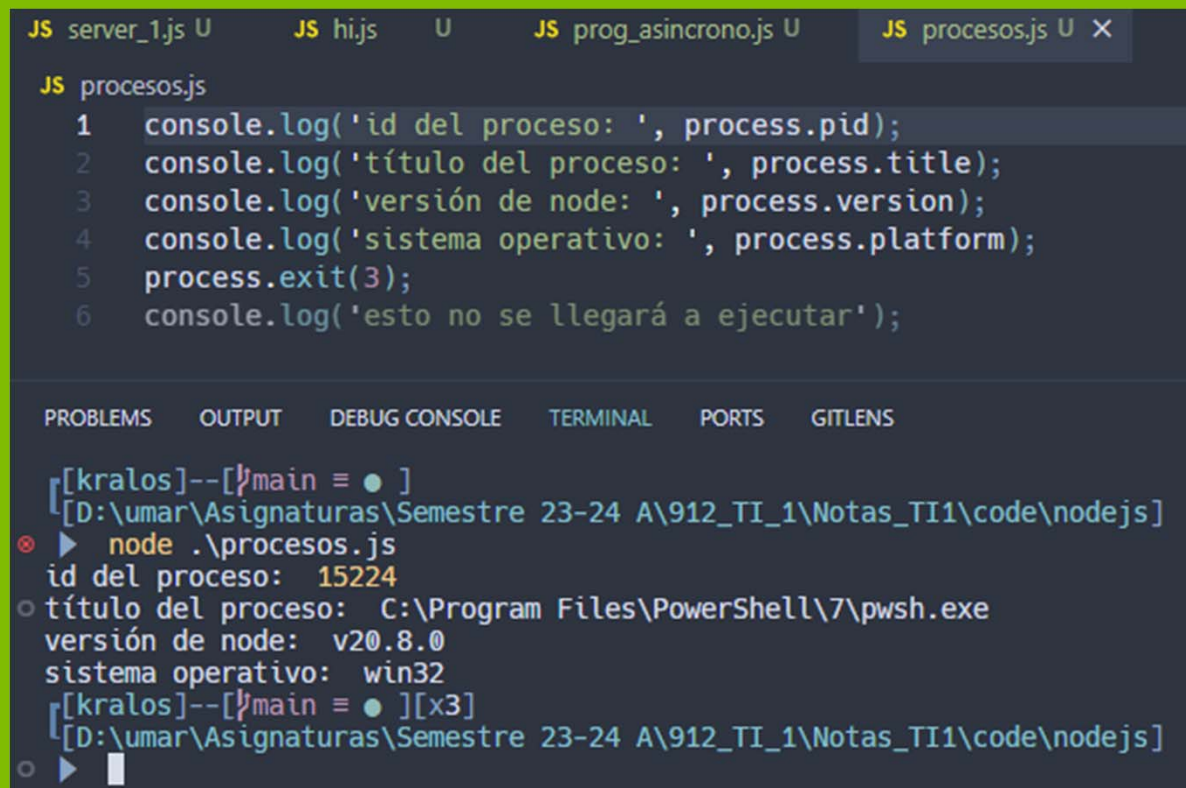
Programación orientada a eventos (POE)

- 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]
⊗ ▶ 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

Referencias

- *Manual de NodeJS.* (n.d.). Desarrolloweb.com. Retrieved October 9, 2023, from <https://desarrolloweb.com/manuales/manual-nodejs.html>