

NODEJS

TEORÍA

¿Qué es NodeJs y qué características tiene?

- Es un entorno de ejecución de JS
- Es de código abierto
- Es multiplataforma (Incorporado hasta en Nintendo switch)
- Es asíncrono aunque monohilo
- Orientado a eventos
- Usa el motor de JS de Chrome

1. Instala NODEJS
2. Crea una nueva carpeta. Ejemplo Práctica Node
3. Ve a la carpeta desde la terminal
4. Escribe **code** . para abrir VS CODE dentro de la carpeta
5. Crea un programa JS que saque en consola Hola mundo
6. Abre la terminal del VS CODE y ejecuta el programa con **node nombre_programa.js**

```
7. const http = require('node:http')
8. const server = http.createServer((req, res) => {
9.   res.writeHead(200, { 'Content-Type': 'text/html' })
10.  res.end('<h1>Hola mundo</h1>')
11.})
12.
13. server.listen(0, () => {
14.  console.log(
15.    `Servidor escuchando en el puerto
16.    http://localhost:${server.address().port}`,
17.  )
18.})
```

Ejemplo de programa que monta un servicio web. El 0 en el listen significa que busque un puerto libre.

INSTALAR ESLINT Y PRETTIER

Uno de los principales problemas es que suelen dar conflictos en el editor el LINTER con el FORMATEADOR de código, el primero ESLINT lo que hace es detectar errores y el segundo PRETTIER se encarga del formato (tipos de comillas, si usamos ; o no, si tabulamos o usamos espacios...). Es importante usar estos programas cuando se programa dentro de un equipo y se usan programas de control de versiones como GIT porque si no se pueden introducir cambios FANTASMAS es decir que abres un fichero y solo cambias una línea pero el formateador te

NODEJS

cambia varias líneas y GIT detecta los cambios y cree que has cambiado más cosas de las que has hecho, por eso todo el equipo debería usar la misma configuración.

Sigue los pasos al pie de la letra.

1. Abre una terminal desde VS CODE, asegúrate que estas en la carpeta de tu proyecto e instala las **extensiones** oficiales de ESLINT y PRETTIER

A mí también me gusta la extensión **Error Lens** para que los errores te salgan desplegados en vez de un pequeño subrayado rojo.

2. Inicializa el proyecto con **npm init** para que se cree el fichero package.json
3. Instala ESLINT (versión STANDARD que ya trae reglas preconfiguradas, -D para solo desarrollo)

npm i -D standard

4. Instala Prettier (solo para desarrollo)

npm i -D prettier

5. Configura prettier. Crea el fichero **.prettierrc**

```
6. {  
7.   "useTabs": true,  
8.   "singleQuote": true,  
9.   "jsxSingleQuote": true,  
10.  "arrowParens": "avoid",  
11.  "semi": false  
12.}  
13.
```

Ver web de prettier -> DOC -> Configurar Prettier -> Opciones

- 6.- Instala el paquete para eliminar conflictos entre prettier y eslint

npm i -D eslint-config-prettier (es la versión oficial)

- 7.- Modifica **package.json** añadiendo

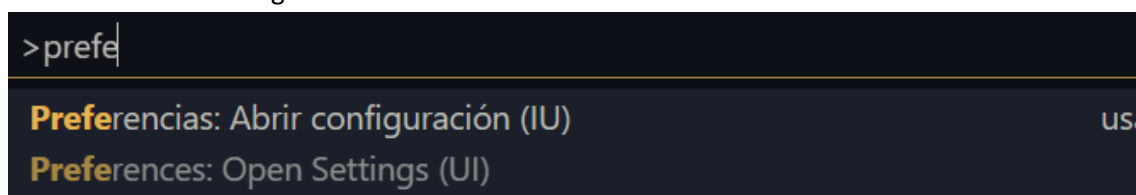
```
,  
  "eslintConfig": {  
    "extends": [  
      "standard",  
      "eslint-config-prettier"  
    ]  
  }  
}
```

NODEJS

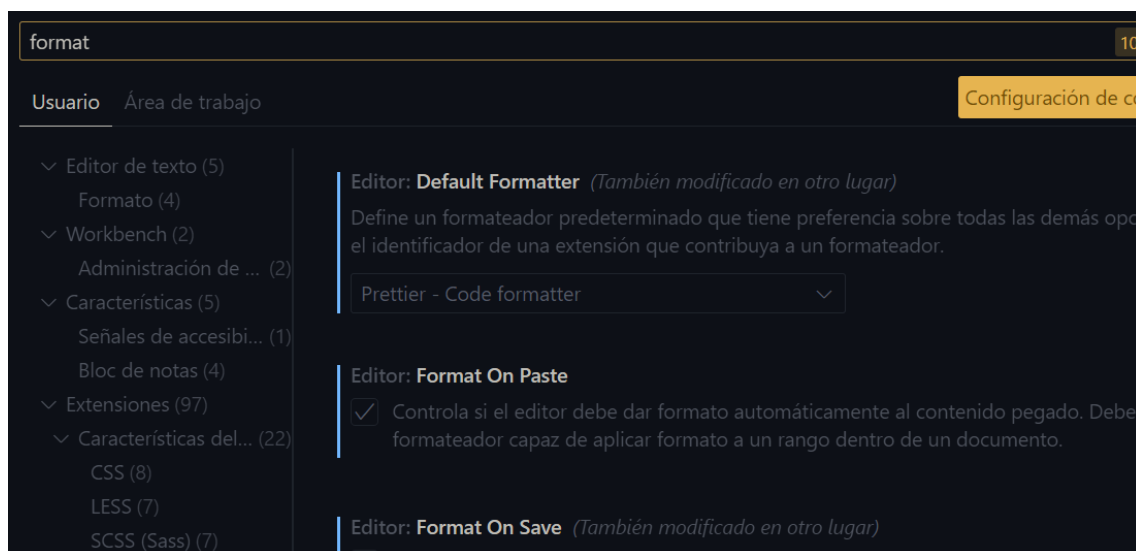
Esto permite configurar ESLINT sin tener que crear un fichero de configuración para ESLINT, aunque debes hacerlo si quieres personalizar reglas..

Le tenemos que decir que extienda STANDARD y que use el módulo para eliminar conflictos.

8.- Puedes ir a la configuración de VS CODE



Preferencias del editor.



Y puedes activar **Format On Paste** para que para cuando copies código te lo formatee y sobre todo activa **Format On Save** para que al grabar el fichero lo formatee.

9.- Si te fijas en el package.json cuando instalas módulos te añade las **dependencias** que necesita tu proyecto (que se encuentran en la carpeta modules)

```
"devDependencies": {
  "eslint-config-prettier": "^9.1.0",
  "prettier": "^3.3.2",
  "standard": "^17.1.0"
},
```

NODEJS

Revisa la documentación de node_semver <https://github.com/npm/node-semver> para saber lo que hace ^ ~ > ...

Módulos: CommonJS vs ES Modules

Leer <https://lenguajejs.com/automatizadores/introduccion/commonjs-vs-es-modules/>

CommonJS

- Carga de módulos SINCRONA
- **// module-name.js**
module.exports = {
 /* ... */
}

// index.js
const module = require("./module-name.js");
const package = require("package");

module.method();
- Node lo usa por defecto para los .js y también para cjs

ESModules

- Carga ASINCRONA
- **// module.js**
export const data = 42;
export const method = () => console.log("Hello");

// index.js
import { data, method } from "./module.js";
- Para usarlos en NODE debes usar extensión .mjs

NODEJS

CIS

CommonJS

```
const elem = require('module');

module.exports = {
};
```



ESM

ES Modules

```
import { elem } from './module.js';

export const elem = {
};
```

