6 DE NOVIEMBRE DE 2025

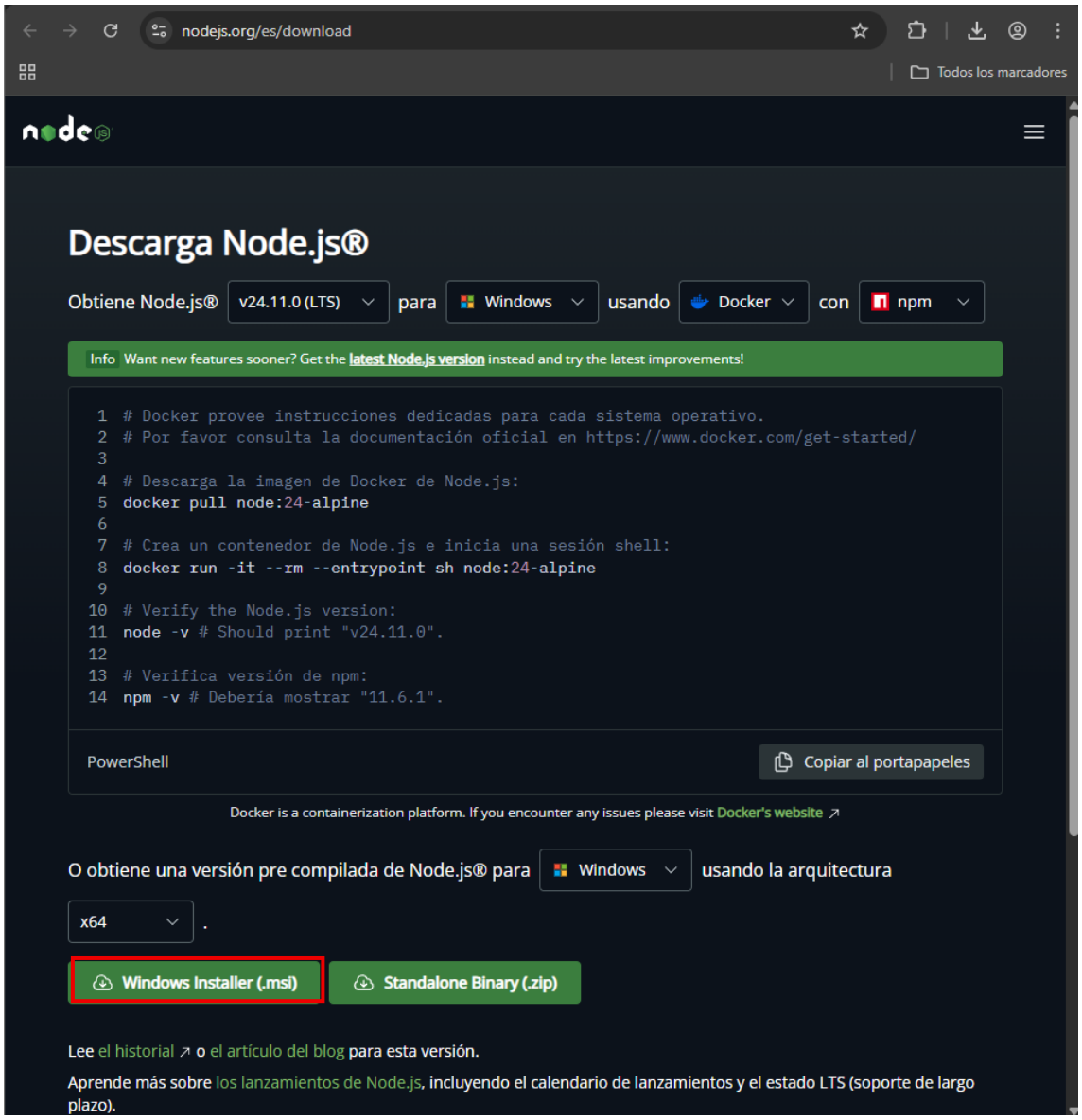
MANUAL DE USO DE NODE

HECHO POR DANIEL GARCÍA MÉNDEZ

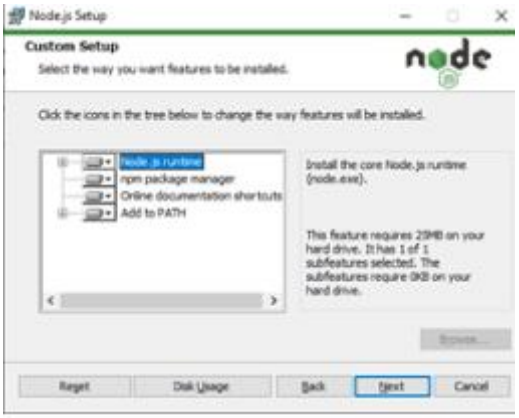
Contenido

Instalación de NODE y NPM

Para instalar node, nos iremos a su página oficial y darle a instalar como se indica en la imagen



Después de ejecutar el msi nos saldrá el asistente de instalación de node, solo es hacer clic sobre next y el asistente nos hará un tour.



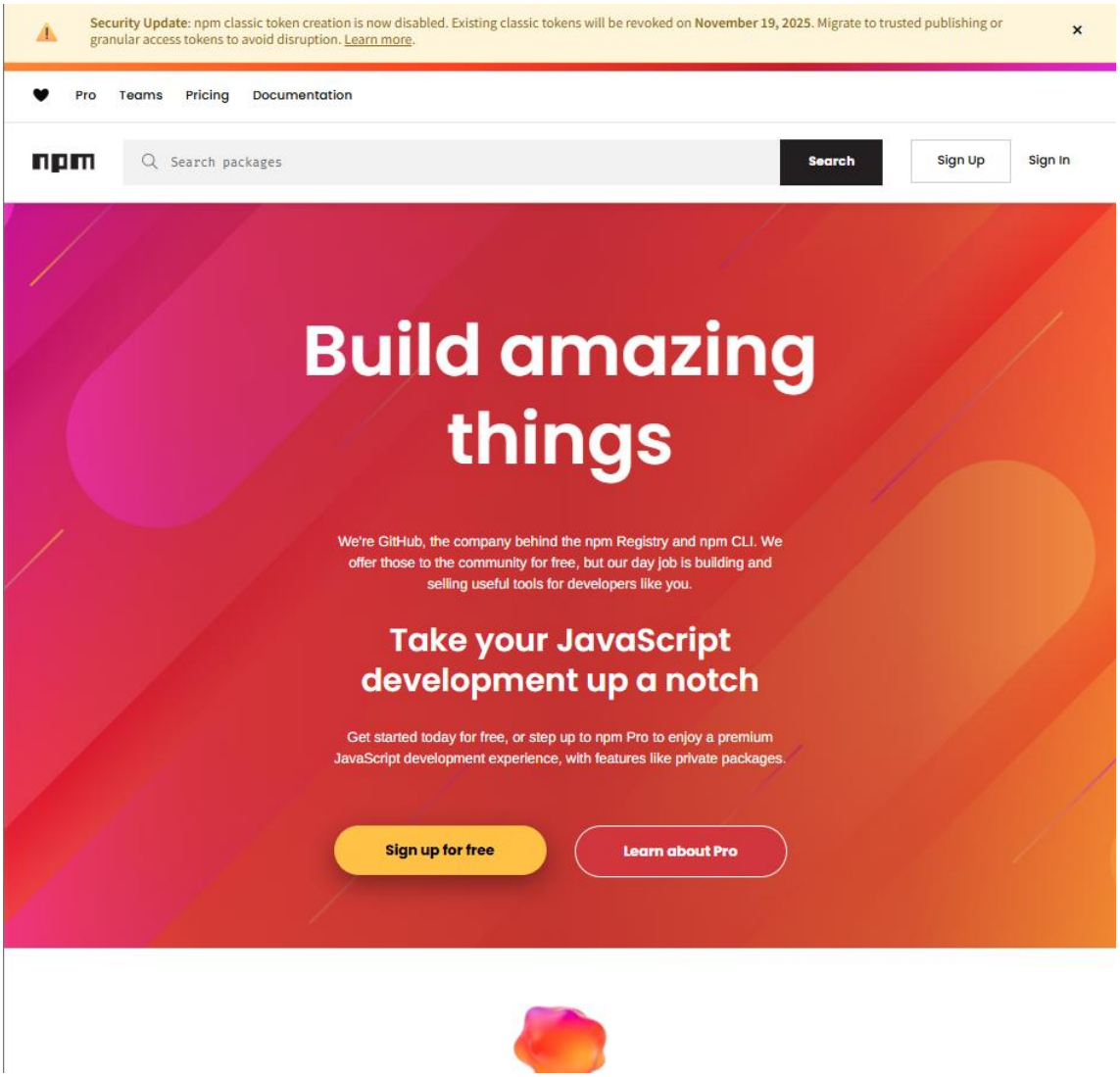
IMPORTANTE: Al instalar node, automáticamente se instala npm.

Luego para ver la versión de node y npm buscaremos el cmd y ejecutaremos **node –version** y **npm –versión**

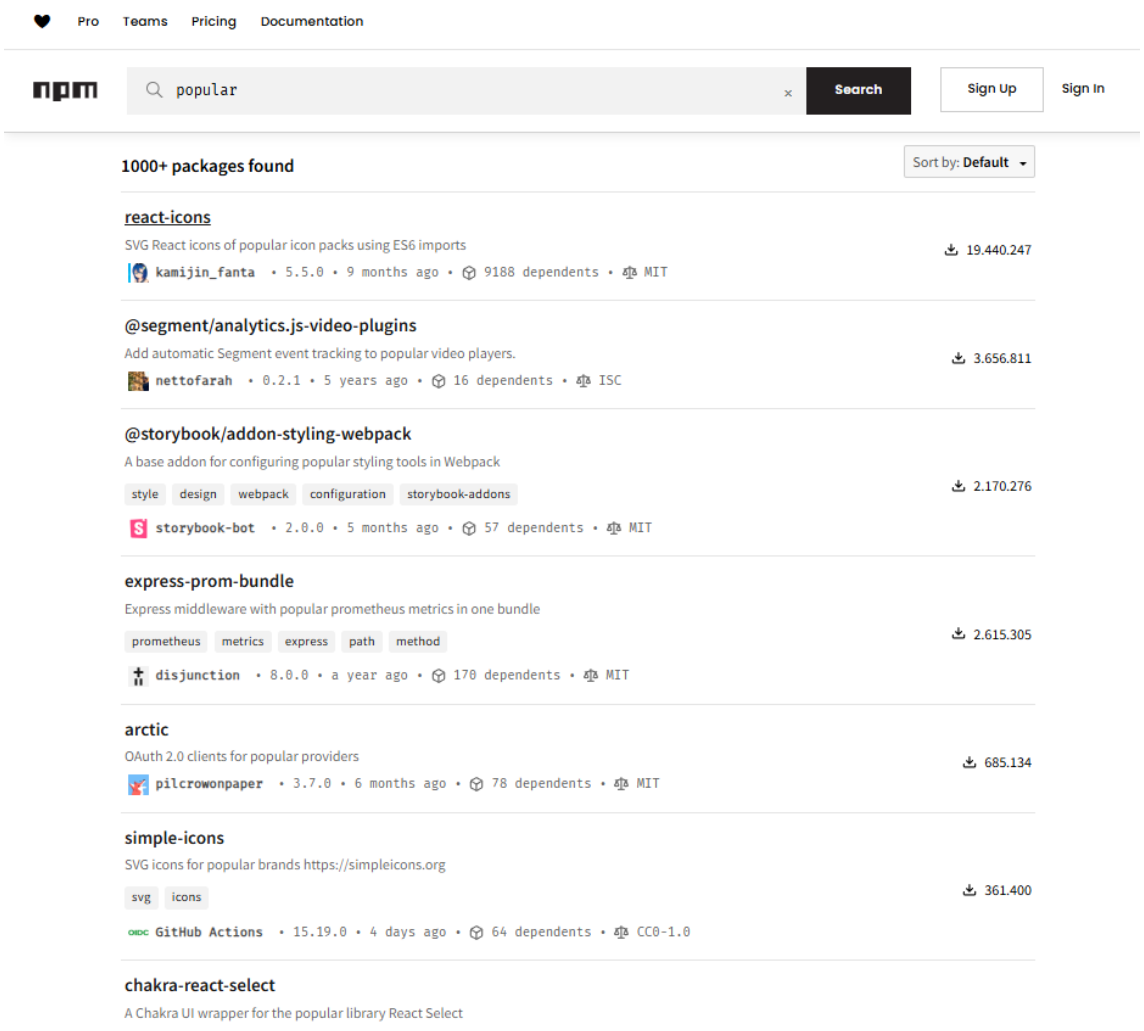
```
C:\Users\alumno>node --version
v22.19.0

C:\Users\alumno>npm --version
10.9.3
```

NPM paquetes

Si vamos al [sitio oficial de npm](), podremos encontrar gran variedad de paquetes.



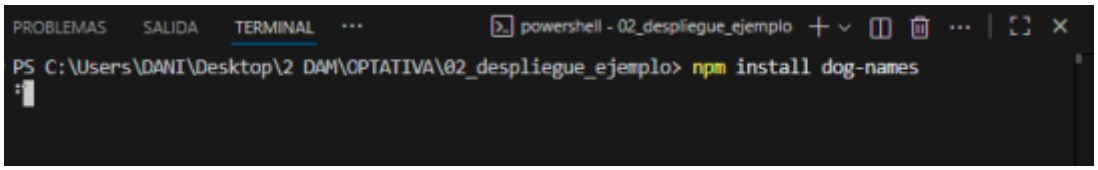
Si ponemos en el buscador “popular”, nos saldrán los paquetes más populares



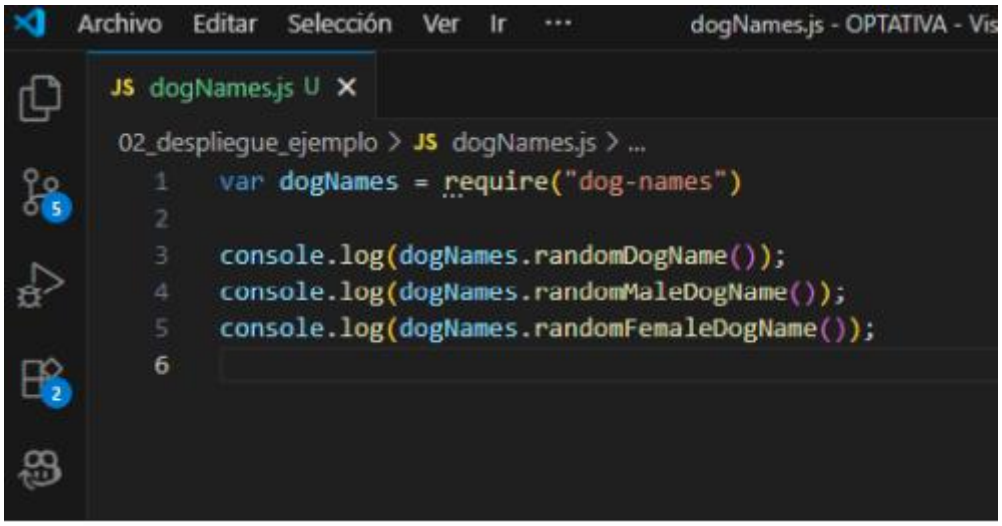
Cómo instalar un paquete con npm

Para instalar un paquete en npm, basta con agregar el comando **npm install nombre_del_paquete**

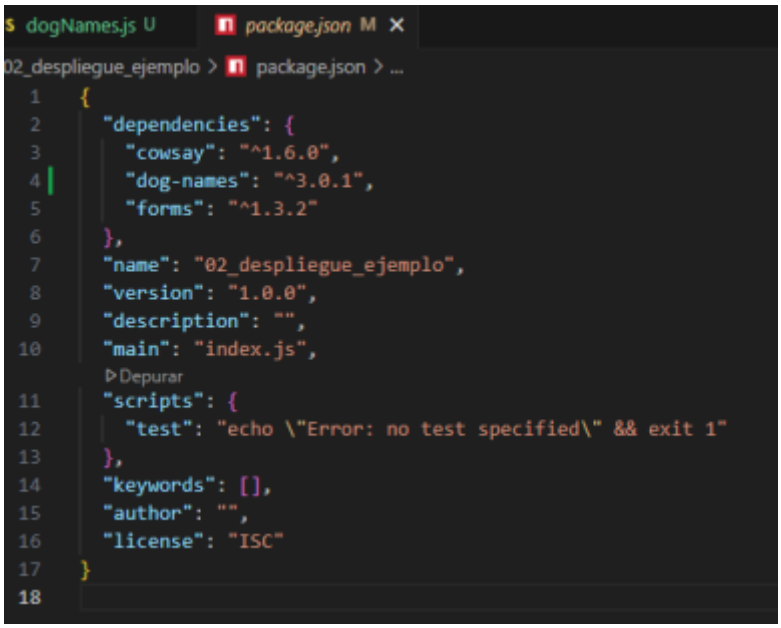
Ejemplo de uso del comando junto al paquete dog-names:

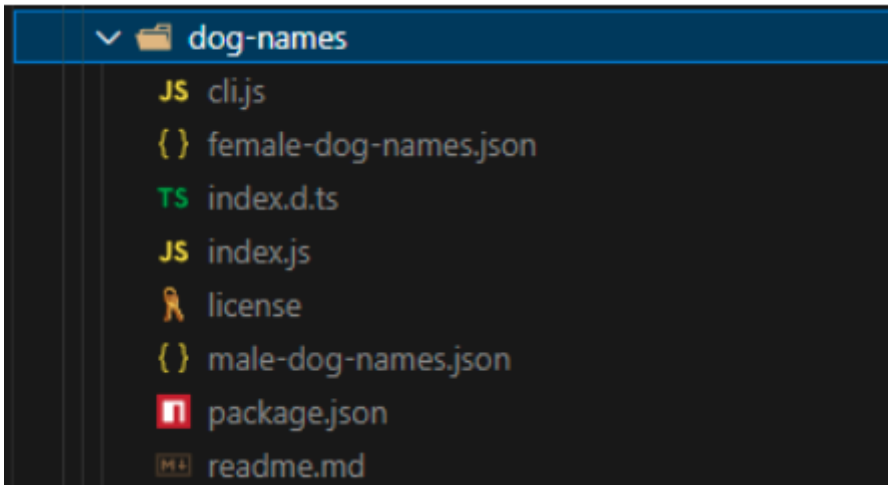


Nos crearemos un archivo js en este caso para probarlo, este paquete genera nombres aleatorios de perros.



Cada vez que instalemos un paquete, aparecerá en el package.json y se creará una carpeta en nodeModules del proyecto, cuando eliminemos un paquete, todo lo relacionado con este, será eliminado de las dependencias del proyecto.

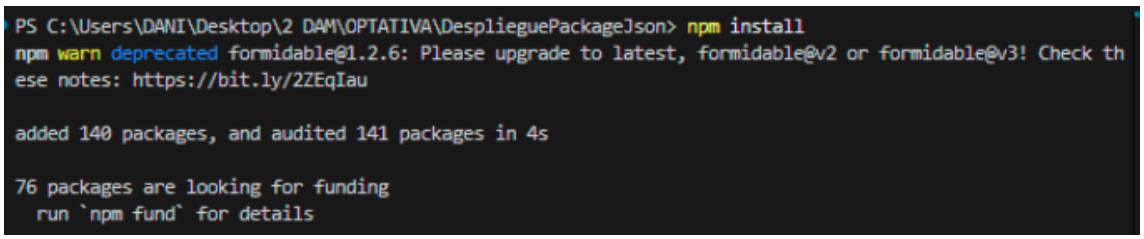


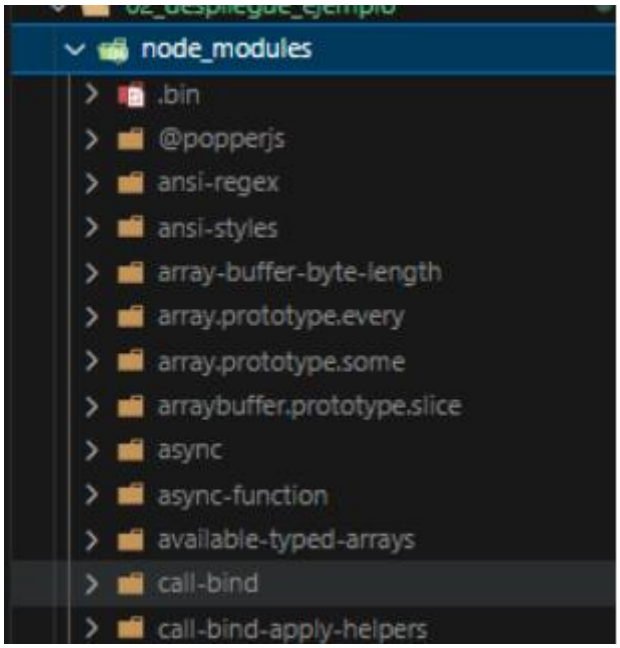


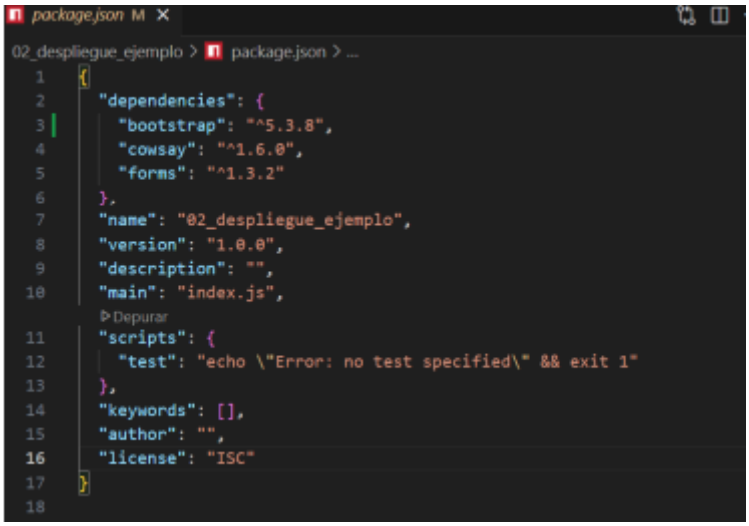
Cómo desplegar un proyecto en node

Para desplegar un proyecto, solo tendremos que poner el comando **npm -i.**

En este caso, voy a hacer un despliegue de un package.json propio, solo pegaremos el .json en la nueva carpeta y veremos que el nuevo proyecto importa todas las dependencias y paquetes del antiguo proyecto a importar.




node_modules
.bin
@popperjs
ansi-regex
ansi-styles
array-buffer-byte-length
array.prototype.every
array.prototype.some
arraybuffer.prototype.slice
async
async-function
available-typed-arrays
call-bind
call-bind-apply-helpers


package.json M
02_despliegue_ejemplo > package.json > ...
{
"dependencies": {
"bootstrap": "^5.3.8",
"cowsay": "^1.6.0",
"forms": "^1.3.2"
},
"name": "02_despliegue_ejemplo",
"version": "1.0.0",
"description": "",
"main": "index.js",
Depurar
"scripts": {
"test": "echo \"Error: no test specified\" && exit 1"
},
"keywords": [],
"author": "",
"license": "ISC"
}

Http

Node.js incluye un potente módulo HTTP integrado que le permite crear servidores HTTP y realizar solicitudes HTTP.

Este módulo es esencial para crear aplicaciones web y API en Node.js.

Características clave

- Crea servidores HTTP para gestionar las peticiones y enviar las respuestas.
- Realizar solicitudes HTTP a otros servidores
- Gestionar diferentes métodos HTTP (GET, POST, PUT, DELETE, etc.)
- Trabajar con encabezados de solicitud y respuesta
- Gestionar datos de transmisión para grandes cargas útiles

Cómo usar http

Para utilizar el módulo HTTP, inclúyalo en su aplicación utilizando el require()método:

```
// Using CommonJS require (Node.js default)
const http = require('http');

// Or using ES modules (Node.js 14+ with "type": "module" in package.json)
// import http from 'http';
```

Creación de un servidor HTTP

El createServer()método del módulo HTTP crea un servidor HTTP que escucha las solicitudes en un puerto específico y ejecuta una función de devolución de llamada para cada solicitud.

Ejemplo básico de servidor HTTP

```
// Import the HTTP module
const http = require('http');

// Create a server object
const server = http.createServer((req, res) => {
  // Set the response HTTP header with HTTP status and Content type
  res.writeHead(200, { 'Content-Type': 'text/plain' });

  // Send the response body as 'Hello, World!'
  res.end('Hello, World!\n');
});

// Define the port to listen on const PORT = 3000;

// Start the server and listen on the specified port
server.listen(PORT, 'localhost', () => {
  console.log(`Server running at http://localhost:${PORT}/`);
});
```

Explicación del código

1. http.createServer()- Crea una nueva instancia de servidor HTTP
2. La función de devolución de llamada se ejecuta para cada solicitud con dos parámetros:
    - req- El objeto de solicitud (http.IncomingMessage)
    - res- El objeto de respuesta (http.ServerResponse)
3. res.writeHead()- Establece el código de estado y los encabezados de la respuesta.
4. res.end()Envía la respuesta y finaliza la conexión.
5. server.listen()- Inicia el servidor en el puerto especificado

Ejecutando el servidor

1. Guarda el código en un archivo llamadoserver.js
2. Ejecutar el servidor usando Node.js:

node server.js

Visite http://localhost:3000 en su navegador para ver la respuesta.

Trabajar con encabezados HTTP

Las cabeceras HTTP te permiten enviar información adicional con tu respuesta.

Este res.writeHead()método se utiliza para establecer el código de estado y las cabeceras de respuesta.

Configuración de encabezados de respuesta

Ejemplo: Configuración de varios encabezados

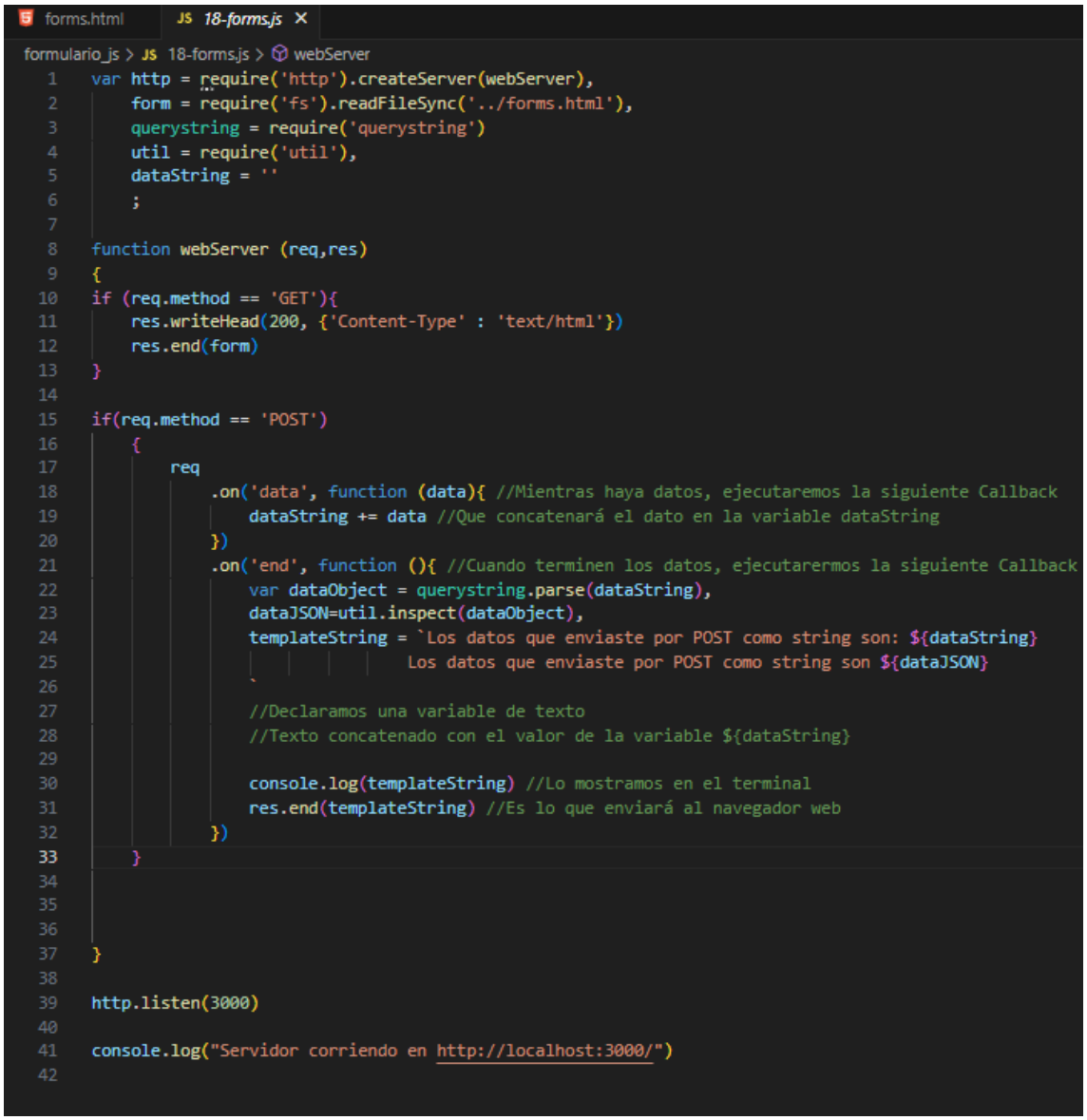
```
const http = require('http');

const server = http.createServer((req, res) => {
  // Set status code and multiple headers
  res.writeHead(200, {
    'Content-Type': 'text/html',
    'X-Powered-By': 'Node.js',
    'Cache-Control': 'no-cache, no-store, must-revalidate',
    'Set-Cookie': 'sessionid=abc123; HttpOnly'
  });

  res.end('<h1>Hello, World!</h1>');
});

server.listen(3000, () => {
  console.log('Server running at http://localhost:3000/');
});
```

Códigos de estado HTTP comunes

Código	Mensaje	Descripción
200	DE ACUERDO	Respuesta estándar para solicitudes HTTP exitosas
201	Creado	La solicitud se ha atendido y se ha creado un nuevo recurso.
301	Mudarse permanentemente	El recurso se ha trasladado a una nueva URL.
400	Solicitud incorrecta	El servidor no puede procesar la solicitud debido a un error del cliente.
401	No autorizado	Se requiere autenticación.

403	Prohibido	El servidor rechaza la autorización de la solicitud.
404	Extraviado	No se pudo encontrar el recurso solicitado.
500	Error Interno del Servidor	Se produjo una situación inesperada

Ejemplo de métodos get y post en HTTP



Conclusiones

No me gusta trabajar sobre presión, no trabajo de forma correcta y me salto algunos pasos importantes.