



6 DE NOVIEMBRE DE 2025

MANUAL DE USO DE NODE

HECHO POR DANIEL GARCÍA MÉNDEZ

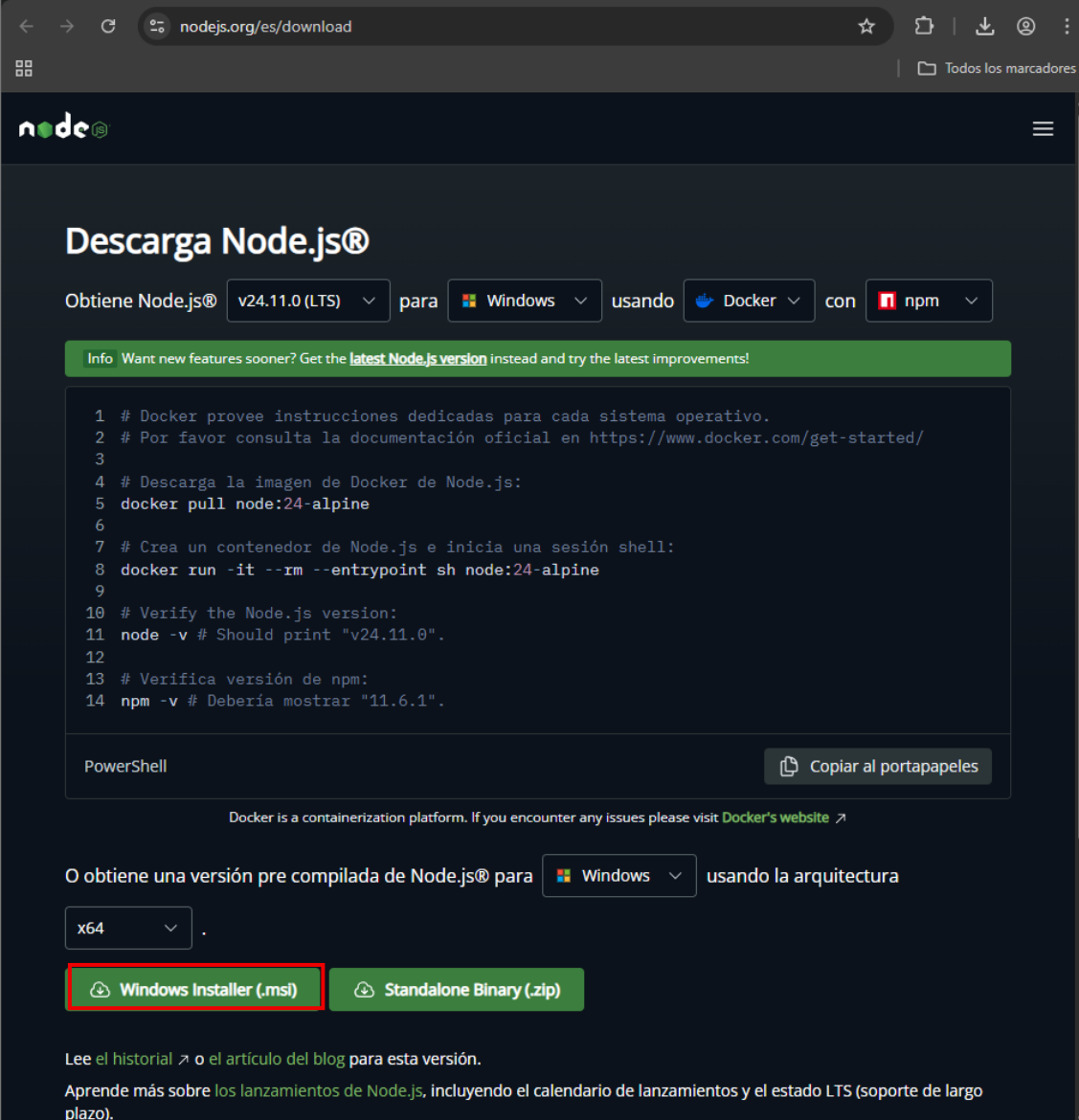


Contenido

Instalación de NODE y NPM	2
NPM paquetes	4
Cómo instalar un paquete con npm	6
Cómo desplegar un proyecto en node	7
Http	9
Cómo usar http	9
Creación de un servidor HTTP	10
Trabajar con encabezados HTTP	11
Configuración de encabezados de respuesta	12
Códigos de estado HTTP comunes	12
Ejemplo de métodos get y post en HTTP.....	13
Conclusiones	14

Instalación de NODE y NPM

Para instalar node, nos iremos a su [página oficial](https://nodejs.org/es/download) y darle a instalar como se indica en la imagen



The screenshot shows the Node.js download page for Windows. The page title is "Descarga Node.js®". Below the title, there are dropdown menus to select the version (v24.11.0 LTS), platform (Windows), and package manager (npm). A green info bar suggests getting the latest version. A code block contains instructions for installing Node.js using Docker. Below the code block, there is a "Copiar al portapapeles" button. A note mentions Docker is a containerization platform. Below that, there are options to get a pre-compiled version for Windows using the x64 architecture. Two buttons are visible: "Windows Installer (.msi)" and "Standalone Binary (.zip)". At the bottom, there are links to the history and blog for this version, and a link to learn more about Node.js releases.

nodejs.org/es/download

Descarga Node.js®

Obtiene Node.js® **v24.11.0 (LTS)** para **Windows** usando **Docker** con **npm**

Info Want new features sooner? Get the [latest Node.js version](#) instead and try the latest improvements!

```
1 # Docker provee instrucciones dedicadas para cada sistema operativo.
2 # Por favor consulta la documentación oficial en https://www.docker.com/get-started/
3
4 # Descarga la imagen de Docker de Node.js:
5 docker pull node:24-alpine
6
7 # Crea un contenedor de Node.js e inicia una sesión shell:
8 docker run -it --rm --entrypoint sh node:24-alpine
9
10 # Verify the Node.js version:
11 node -v # Should print "v24.11.0".
12
13 # Verifica versión de npm:
14 npm -v # Debería mostrar "11.6.1".
```

PowerShell [Copiar al portapapeles](#)

Docker is a containerization platform. If you encounter any issues please visit [Docker's website](#)

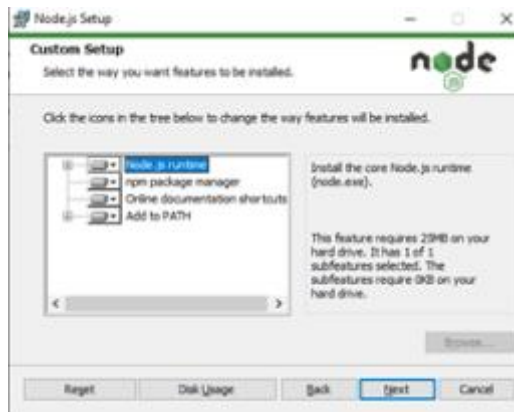
O obtiene una versión pre compilada de Node.js® para **Windows** usando la arquitectura **x64**.

[Windows Installer \(.msi\)](#) [Standalone Binary \(.zip\)](#)

Lee el [historial](#) o el [artículo del blog](#) para esta versión.

Aprende más sobre [los lanzamientos de Node.js](#), incluyendo el calendario de lanzamientos y el estado LTS (soporte de largo plazo).

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.

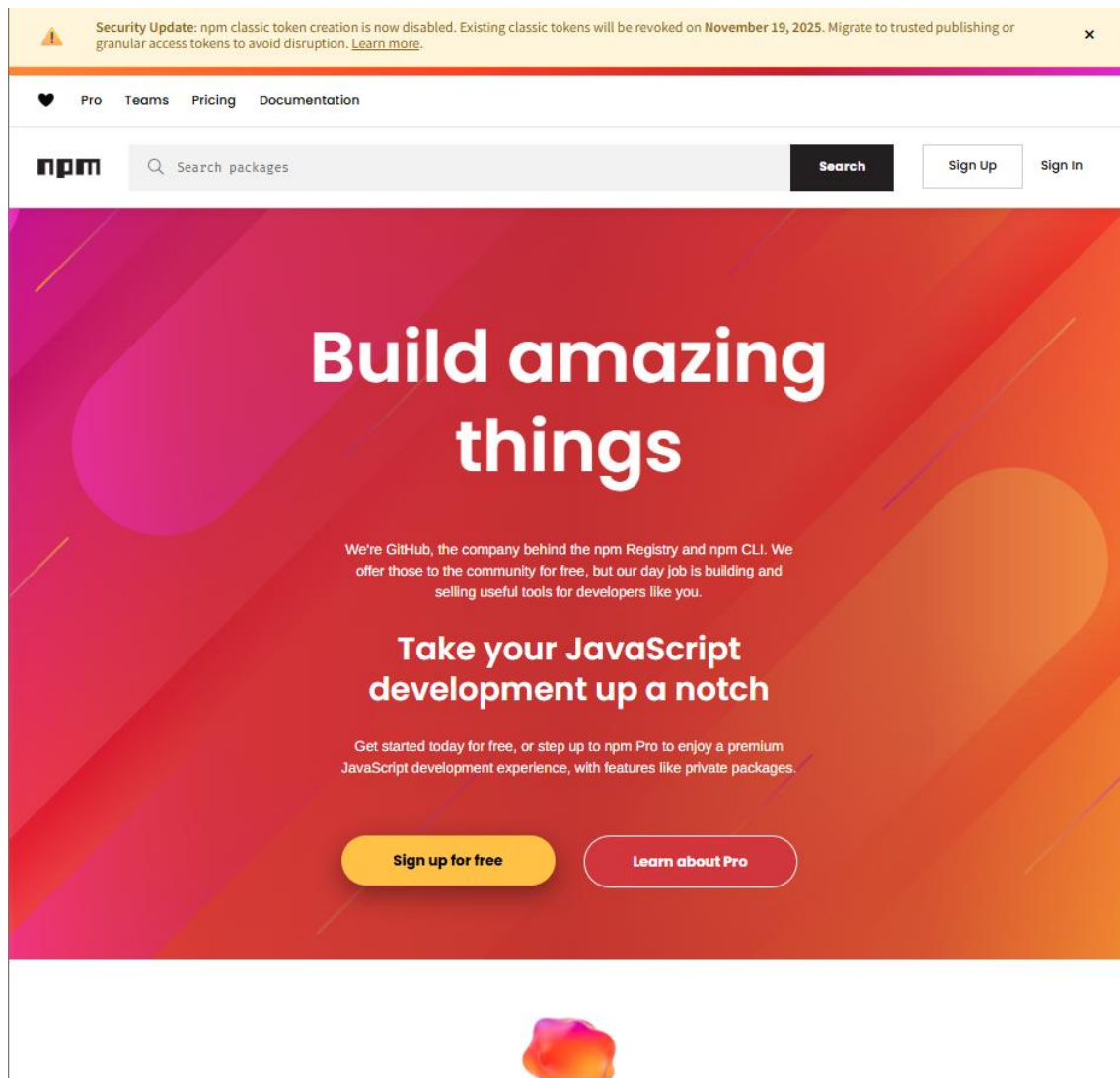
Luego para ver la versión de node y npm buscaremos el cmd y ejecutaremos **node –version** y **npm –versión**

```
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](https://www.npmjs.com), podremos encontrar gran variedad de paquetes.



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


 × Search Sign Up Sign In

1000+ packages found

Sort by: Default

react-icons


SVG React icons of popular icon packs using ES6 imports

 kamijin_fanta • 5.5.0 • 9 months ago • 9188 dependents • MIT

19,440,247

@segment/analytics.js-video-plugins

Add automatic Segment event tracking to popular video players.


 nettofarah • 0.2.1 • 5 years ago • 16 dependents • ISC

3,656,811

@storybook/addon-styling-webpack

A base addon for configuring popular styling tools in Webpack

style design webpack configuration storybook-addons


 storybook-bot • 2.0.0 • 5 months ago • 57 dependents • MIT

2,170,276

express-prom-bundle

Express middleware with popular prometheus metrics in one bundle


prometheus metrics express path method

 disjunction • 8.0.0 • a year ago • 170 dependents • MIT

2,615,305

arctic

OAuth 2.0 clients for popular providers


 pilcrowonpaper • 3.7.0 • 6 months ago • 78 dependents • MIT

685,134

simple-icons

SVG icons for popular brands <https://simpleicons.org>

svg icons

 GitHub Actions • 15.19.0 • 4 days ago • 64 dependents • CC0-1.0

361,400

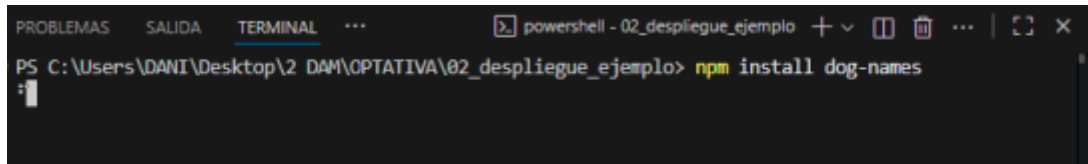
chakra-react-select

A Chakra UI wrapper for the popular library React Select

Cómo instalar un paquete con npm

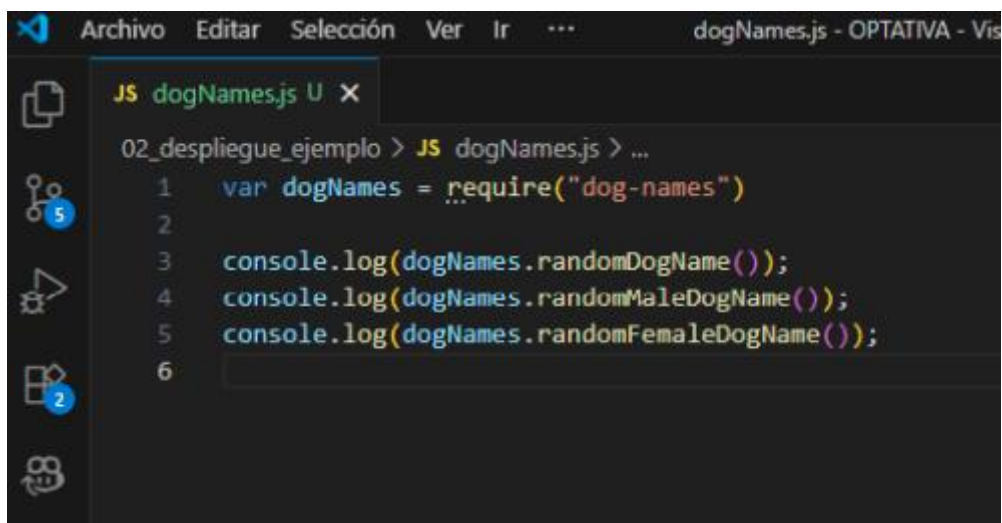
Para instalar un paquete en npm, basta con agregar el comando **npm install nombre_del_paquete**

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



```
PROBLEMAS  SALIDA  TERMINAL  ...  powershell - 02_despliegue_ejemplo + v [] [] ... | {} x
PS C:\Users\DANI\Desktop\2 DAM\OPTATIVA\02_despliegue_ejemplo> npm install dog-names
```

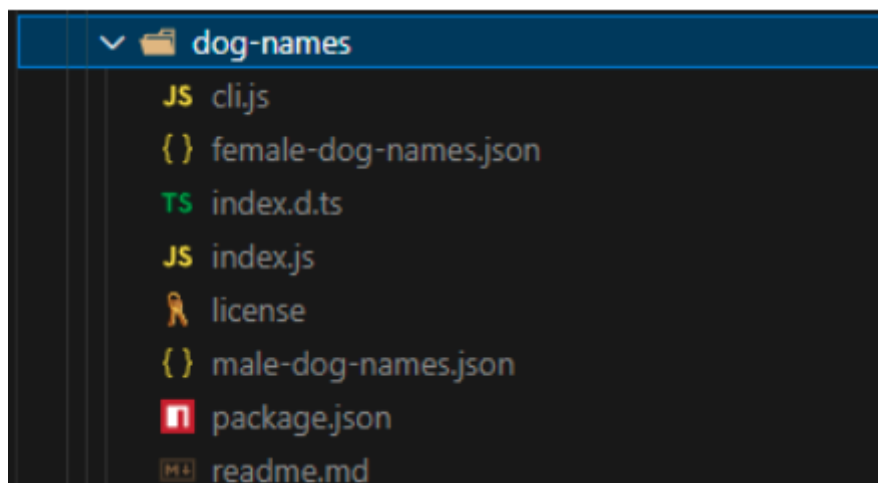
Nos crearemos un archivo js en este caso para probarlo, este paquete genera nombres aleatorios de perros.



```
Archivo  Editar  Selección  Ver  Ir  ...  dogNames.js - OPTATIVA - Vis
JS dogNames.js U X
02_despliegue_ejemplo > JS dogNames.js > ...
1  var dogNames = require("dog-names")
2
3  console.log(dogNames.randomDogName());
4  console.log(dogNames.randomMaleDogName());
5  console.log(dogNames.randomFemaleDogName());
6
```

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.

```
$ dogNames.js U  package.json M X
02_despliegue_ejemplo > package.json > ...
1  {
2    "dependencies": {
3      "cowsay": "^1.6.0",
4      "dog-names": "^3.0.1",
5      "forms": "^1.3.2"
6    },
7    "name": "02_despliegue_ejemplo",
8    "version": "1.0.0",
9    "description": "",
10   "main": "index.js",
11   "scripts": {
12     "test": "echo \"Error: no test specified\" && exit 1"
13   },
14   "keywords": [],
15   "author": "",
16   "license": "ISC"
17 }
18
```



Cómo desplegar un proyecto en node

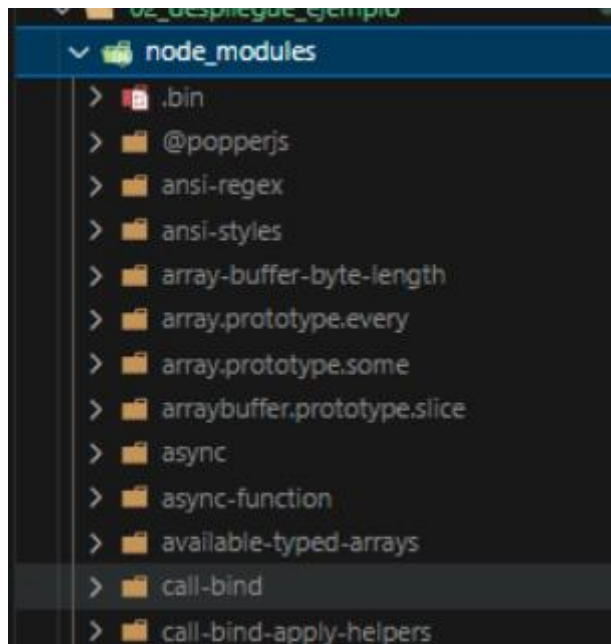
Para desplegar un proyecto, solo tendremos que poner el comando **npm -i**.

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.

```
PS C:\Users\DANI\Desktop\2 DAM\OPTATIVA\DesplieguePackageJson> npm install
npm warn deprecated formidable@1.2.6: Please upgrade to latest, formidable@v2 or formidable@v3! Check these notes: https://bit.ly/2ZEQiaU

added 140 packages, and audited 141 packages in 4s

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

```
package.json M X
02_despliegue_ejemplo > package.json > ...
1 {
2   "dependencies": {
3     "bootstrap": "^5.3.8",
4     "cowsay": "^1.6.0",
5     "forms": "^1.3.2"
6   },
7   "name": "02_despliegue_ejemplo",
8   "version": "1.0.0",
9   "description": "",
10  "main": "index.js",
11  "scripts": {
12    "test": "echo \"Error: no test specified\" && exit 1"
13  },
14  "keywords": [],
15  "author": "",
16  "license": "ISC"
17 }
```

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 llamado `server.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

```
forms.html x
forms.html > html
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <meta charset="UTF-8">
5    <title>Envío de Formularios con Node.js</title>
6  </head>
7  <body>
8    <form method="POST">
9      <div>
10       <label for="nombre">Nombre:</label>
11       <input type="text" name="nombre" required>
12     </div>
13     <div>
14       <label for="email">Email:</label>
15       <input type="text" name="email" required>
16     </div>
17     <div>
18       <input type="submit">
19     </div>
20   </form>
21 </body>
22
23 </html>
```

```
forms.html JS 18-forms.js X
formulario_js > JS 18-forms.js > webServer
1 var http = require('http').createServer(webServer),
2   form = require('fs').readFileSync('../forms.html'),
3   querystring = require('querystring')
4   util = require('util'),
5   dataString = ''
6   ;
7
8 function webServer (req,res)
9 {
10  if (req.method == 'GET'){
11    res.writeHead(200, {'Content-Type' : 'text/html'})
12    res.end(form)
13  }
14
15  if(req.method == 'POST')
16  {
17    req
18      .on('data', function (data){ //Mientras haya datos, ejecutaremos la siguiente Callback
19        dataString += data //Que concatenará el dato en la variable dataString
20      })
21      .on('end', function (){ //Cuando terminen los datos, ejecutaremos la siguiente Callback
22        var dataObject = querystring.parse(dataString),
23          dataJSON=util.inspect(dataObject),
24          templateString = `Los datos que enviaste por POST como string son: ${dataString}
25                          | Los datos que enviaste por POST como string son ${dataJSON}
26                          `
27        //Declaramos una variable de texto
28        //Texto concatenado con el valor de la variable ${dataString}
29
30        console.log(templateString) //Lo mostramos en el terminal
31        res.end(templateString) //Es lo que enviará al navegador web
32      })
33  }
34
35
36
37 }
38
39 http.listen(3000)
40
41 console.log("Servidor corriendo en http://localhost:3000/")
42
```

Conclusiones

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