

# INTRODUCCIÓN A NODE.JS

**DEV.FX**  
DESARROLLAMOS(PERSONAS);

dev



# OBJETIVOS

1. Entender que es Node.js, qué problemas resuelve y donde podemos utilizarlo.
2. Aprender a instalar Node.js en nuestro equipo y comprobar que funcione correctamente.
3. Comprender el uso básico de la línea de comandos con el fin de poder movernos entre carpetas y ver su contenido.
4. Saber usar la línea de comandos para poder ejecutar nuestros archivos JavaScript con Node.js
5. Aprender el flujo de trabajo de ejecución de archivos JavaScript con Node.js desde Visual Studio Code.

# NODE.JS

JavaScript es uno de los lenguajes de programación más populares del mundo.

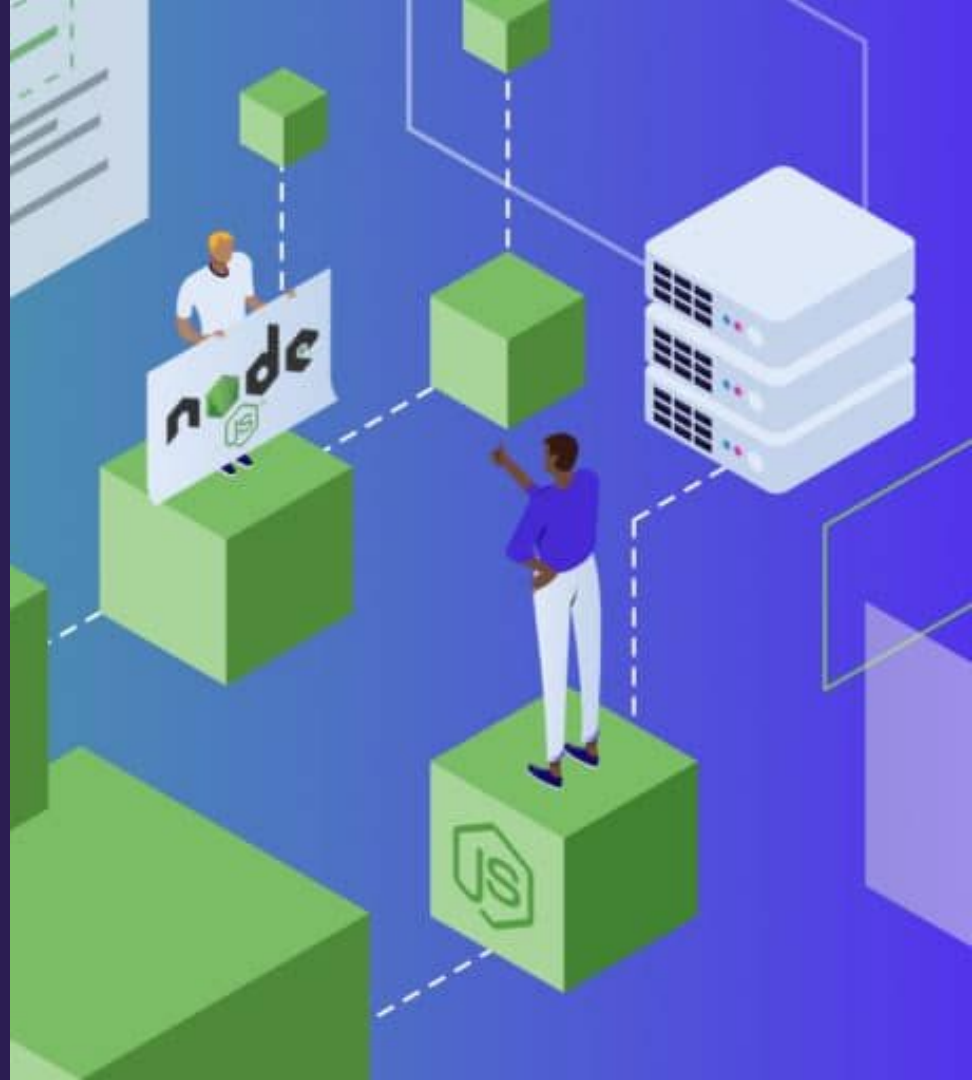
Durante sus primeros 20 años, JavaScript se utilizó principalmente para la creación de scripts del lado del cliente.

Más tarde llegó Node.js, que es un **entorno de ejecución** que incluye todo lo necesario para ejecutar un programa escrito en **JavaScript del lado del servidor**.



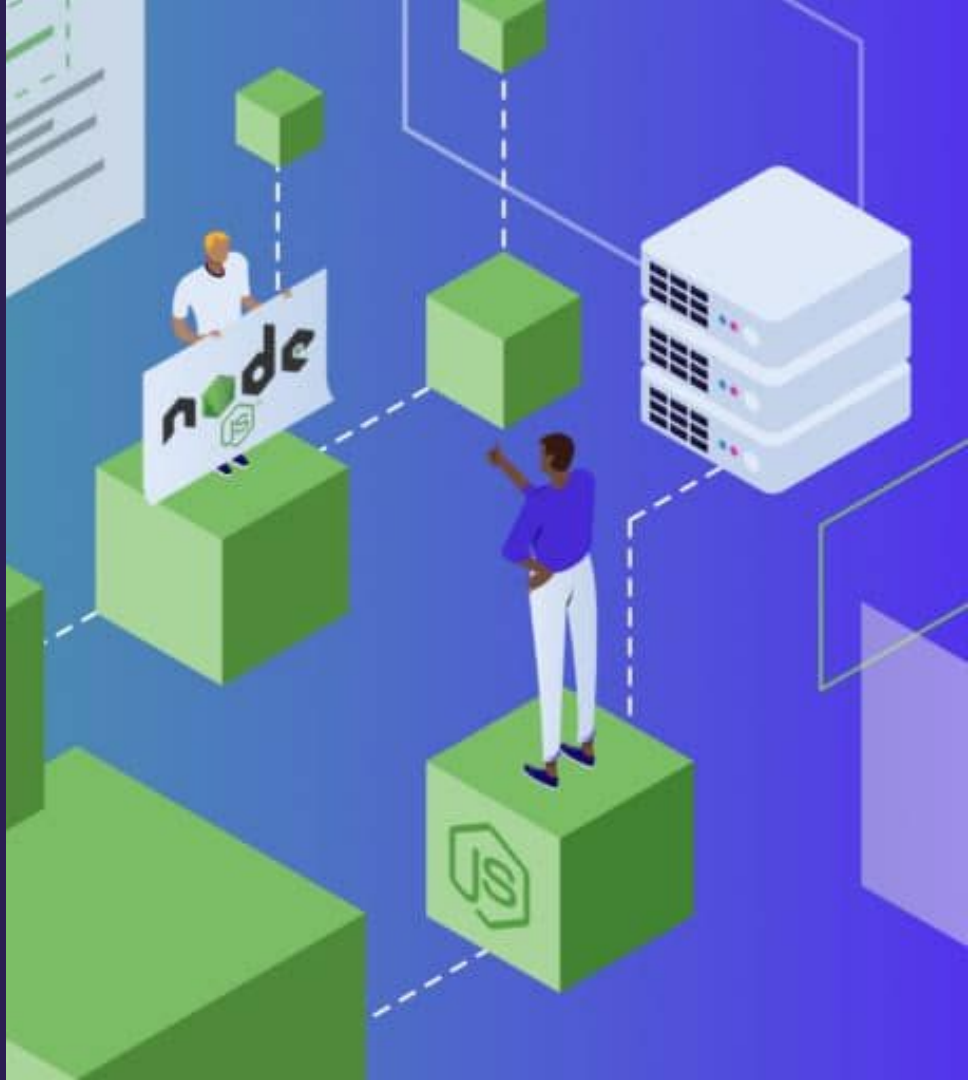
# Qué no es `node.js`?

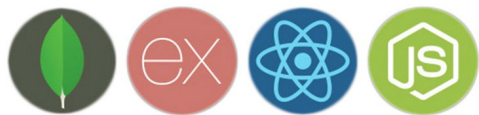
- NO es un lenguaje de programación
- NO es una página web
- No es una framework
- No es un servidor



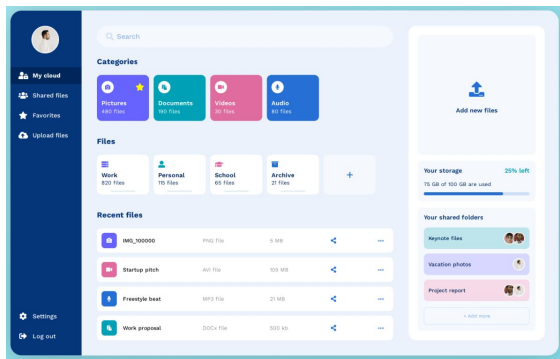
# NODE.JS ES:

- Es un entorno de ejecución de JavaScript
- Es de código abierto y multiplataforma.
- Es de un solo hilo (asíncrono).
- Arquitectura orientada a eventos
- Se ejecuta en el motor de ejecución de JavaScript V8.
- Es compatible con un amplio repositorio de paquetes de código abierto que facilitan el desarrollo de aplicaciones (npm).





M E R N



# Por qué aprender node.js

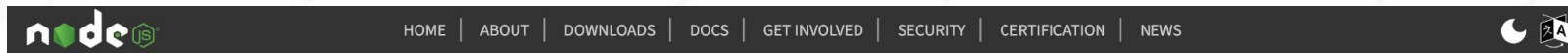
1. Demanda del mercado
  - a. Stack MERN  
(Mongo DB, Express, React, Node.js)
  - b. Stack MEAN  
(Mongo DB, Express, Angular, Node.js)
2. Es JavaScript, ese ya te lo sabes!
3. Podemos desarrollar Web Apps, APIs, Apps de Escritorio, etc...
4. Una comunidad gigante.
5. Rápido, escalable, facil de desplegar (gratis o barato)

# ¿CÓMO INSTALAMOS NODE.JS?

**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev

# Descarga de Node.js



Node.js® is an open-source, cross-platform JavaScript runtime environment.

Security releases now available

Download for macOS

18.18.2 LTS

Recommended For Most Users

20.8.1 Current

Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#)   [Other Downloads](#) | [Changelog](#) | [API Docs](#)

For information about supported releases, see the [release schedule](#).

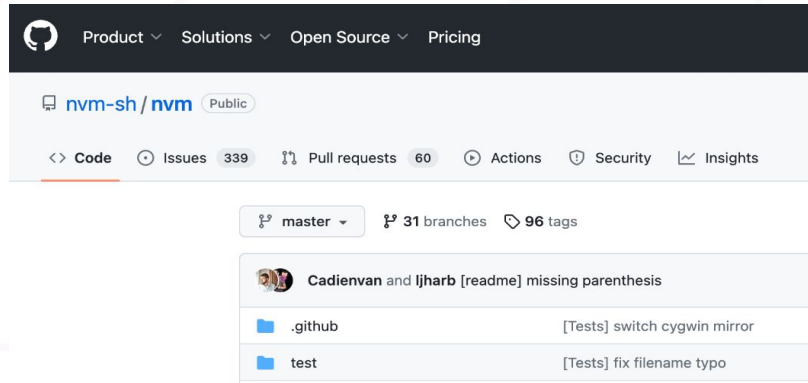
Descargar  
la última  
versión que  
tenga LTS

<https://nodejs.org>

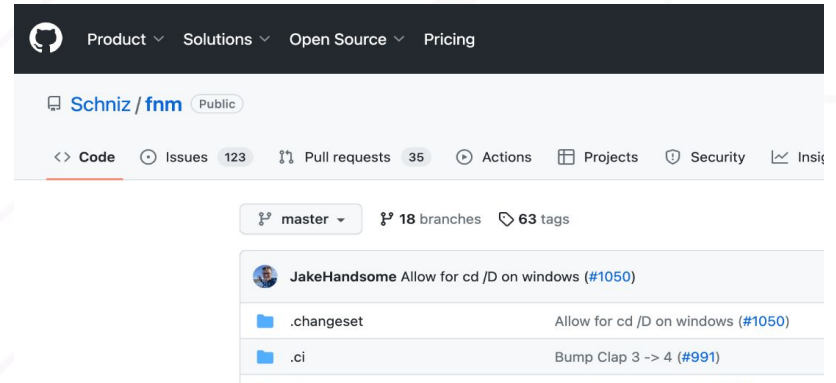
DEV.F



# Alternativas



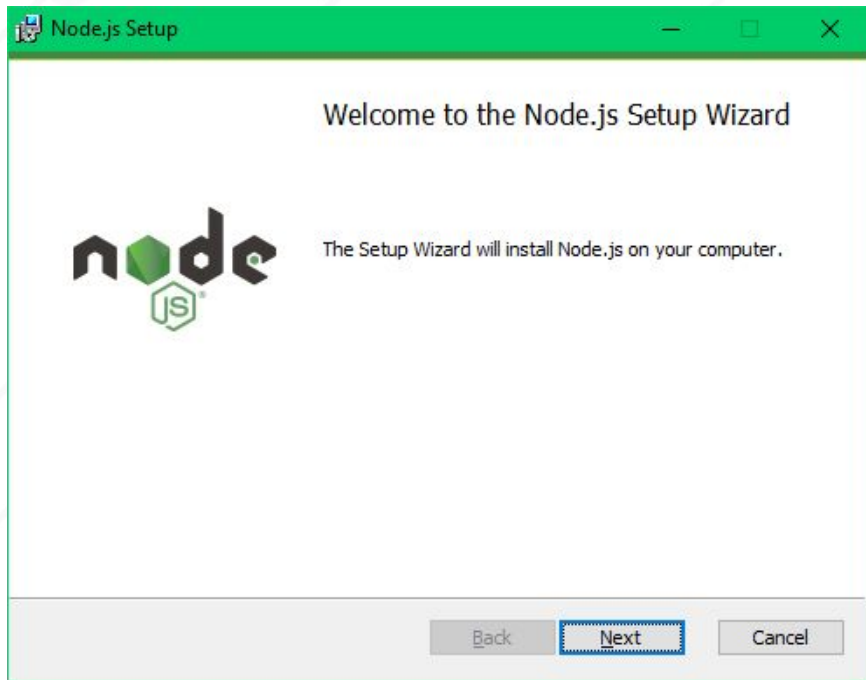
<https://github.com/nvm-sh/nvm>



<https://github.com/Schniz/fnm>

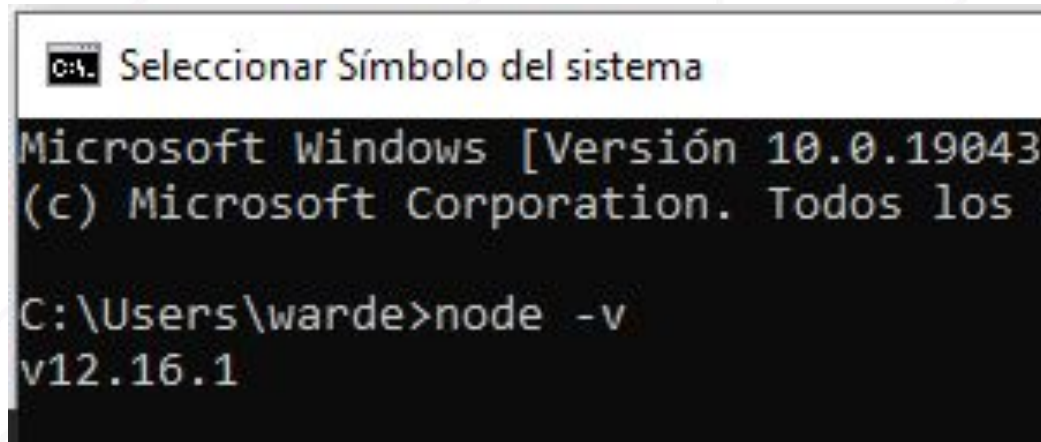
<https://www.rust-lang.org/>

# Proceso de Instalación



El Next... Next...  
Next... de toda la  
vida.

# Comprobar Instalación



```
C:\> Seleccionar Símbolo del sistema

Microsoft Windows [Versión 10.0.19043]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\warde>node -v
v12.16.1
```

Ir a una línea de comandos, y ejecutar el comando:

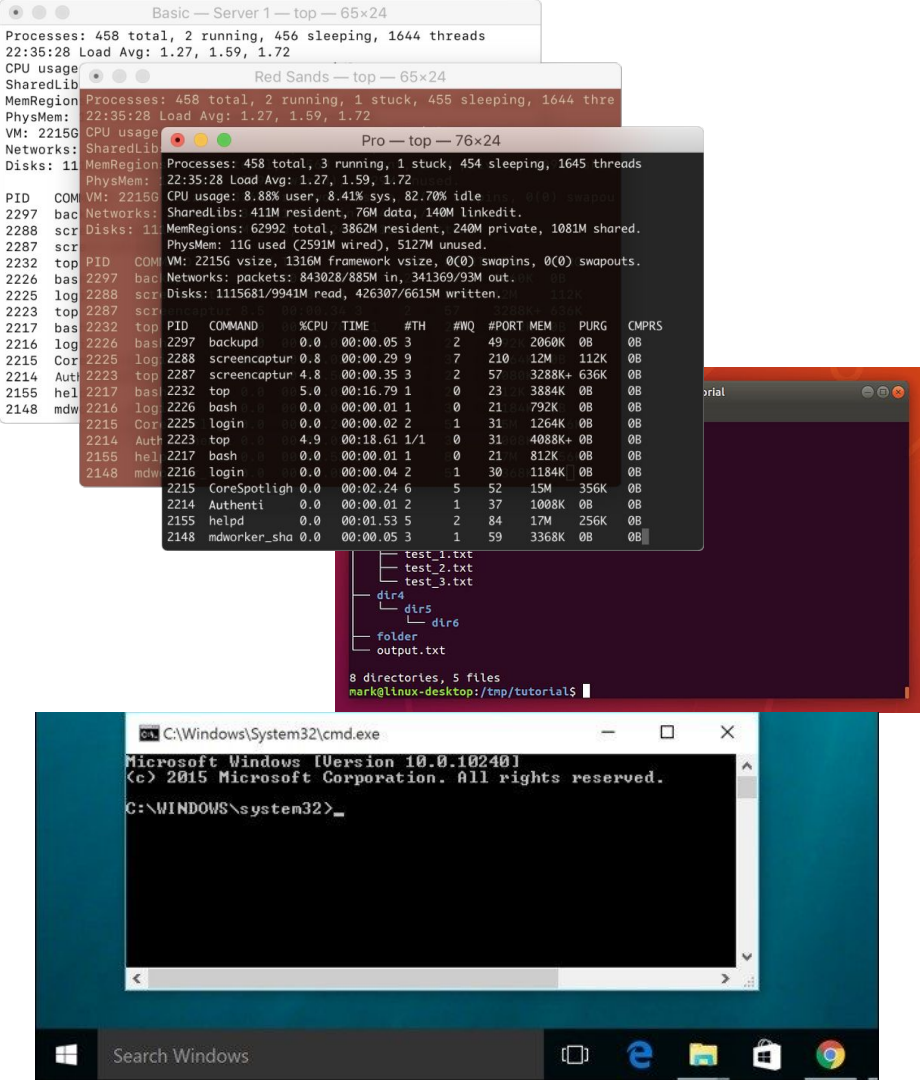
**node -v**

Deberemos poder ver la versión de node.js que tenemos instalada.

# ¿CÓMO USO NODE.JS?

**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev



# Breve Recordatorio: Terminal y Línea de Comandos

La interfaz de línea de comandos es un tipo de interfaz de usuario de computadora que permite a los usuarios dar instrucciones a algún programa informático o al sistema operativo por medio de una línea de texto simple.

# Resumen de Comandos Elementales

## Linux / Mac OS

### **pwd**

*(print working directory)*

Este comando imprime la ubicación de tu directorio de trabajo actual.

`$ pwd`

### **ls**

*(list)*

Imprime el contenido de un directorio.

`$ ls`

### **cd directorio**

*(change directory)*

Permite moverte a otro directorio

`$ cd Documents/DEVF`

## Windows

### **cd**

*(current directory)*

Sin pasarle ningún parámetro adicional, cd te muestra tu ubicación actual

`$ cd`

### **dir**

*(directory)*

Imprime el contenido de un directorio.

`$ dir`

### **cd directorio**

*(change directory)*

Permite moverte a otro directorio

`$ cd Documents/DEVF`

`$ cd ..` ← Nos permite regresar un nivel



# DEV.F

## **PRO TIP #1**

*Cuando escribimos nombres de archivos o carpetas en una terminal, podemos presionar la tecla **Tab** (tabulador) para **autocompletar**.*



**DEV.F**

## **PRO TIP #2**

*En una terminal, si presionamos la tecla de flecha hacia arriba o hacia abajo, podemos desplazarnos entre comandos que hemos usado anteriormente*





DEV.F

### **PRO TIP #3**

Cuando ejecutamos un comando que **“bloquea nuestra terminal”**, podemos forzar la salida (terminar el proceso) con la combinación de teclas: **CTRL + C**

# Usando node.js

Node.js command prompt

```
c:\Users\sdkca\Desktop\nodeScripts>node index.js
Let's talk about Carlos.
He's 172 centimeters tall.
He's 71 kilograms heavy.
Actually that's not too heavy.
He's got Brown eyes and Black hair.
His teeth are usually White depending on the coffee.
If I add 18, 172, and 71 I get 261. I don't know what
that means but, whatever.
finished

c:\Users\sdkca\Desktop\nodeScripts>_
```

En la línea de comandos,  
ejecutar:

**node archivo.js**

De esta forma  
ejecutaremos el archivo  
indicado.

# Qué perdemos usando node.js?

fetch

DOM

window

append

createElement

querySelector

document

getElementById

DEV.FM

dev

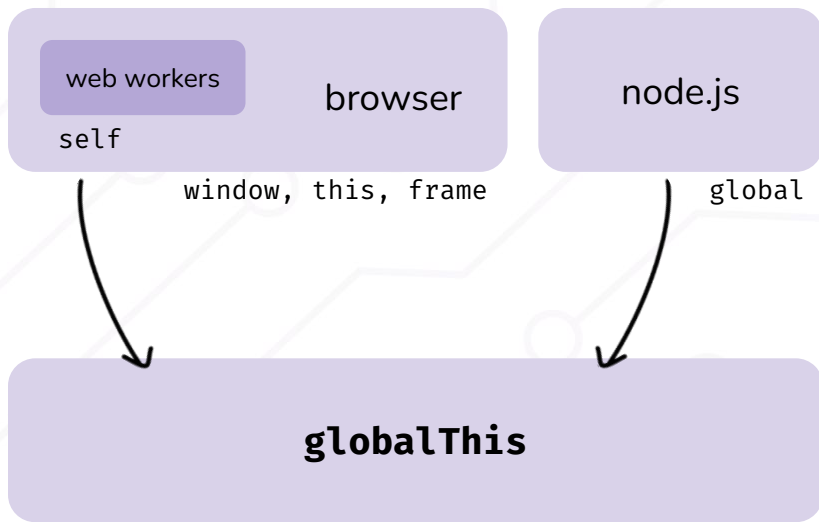
# globalThis

Cuando usábamos JavaScript en el navegador teníamos acceso a la variable global **window**.

Dado que node.js no se ejecuta necesariamente en el navegador no tenemos acceso a window, pero existe una variable la variable global **global**.

Para no tener nombres diferentes para la misma variable en diferentes entornos, usamos la variable **globalThis**, lo que hace referencia a **global** en node.js y a **window** en el navegador.

[globalThis - MDN](#)



Un Objeto para gobernarlos a todos

**DEV.F**

# Vamonos al código!



dev

# CommonJS

```
1 const Math = {};  
2  
3 function add (a, b) {  
4   return a + b;  
5 }  
6  
7 Math.suma = add;
```

```
1 function add (a, b) {  
2   return a + b;  
3 }  
4  
5 exports.suma = add;
```

```
1 const math = require('./math.js');  
2 console.log(math.suma(2,5));
```

```
1 function add (a, b) {  
2   return a + b;  
3 }  
4  
5 module.exports = {  
6   add  
7 };
```

```
1 const { add } = require('./math.js');  
2 console.log(add(2,5));
```

# ES Modules



```
1 import { add } from './math.mjs';  
2 console.log(add(3,4));
```



```
1 export function add (a, b) {  
2     return a + b;  
3 }
```

# Extensiones para uso de módulos

- .js** - usa CommonJS
- .cjs** - usa CommonJS
- .mjs** - usa ES Modules





# Módulos nativos

**DEV.F**  
DESARROLLAMOS(PERSONAS);

dev

# Node.js v18.18.2 documentation

► Other versions | ► Options

- About this documentation
- Usage and example

- Assertion testing
- Asynchronous context tracking
- Async hooks
- Buffer
- C++ addons
- C/C++ addons with Node-API
- C++ embedder API
- Child processes
- Cluster
- Command-line options
- Console
- Corepack
- Crypto
- Debugger

- Deprecated APIs
- Diagnostics Channel
- DNS
- Domain
- Errors
- Events
- File system
- Globals
- HTTP
- HTTP/2
- HTTPS
- Inspector
- Internationalization
- Modules: CommonJS modules
- Modules: ECMAScript modules
- Modules: node:module API
- Modules: Packages

- Net
- OS
- Path
- Performance hooks
- Permissions
- Process
- Punycode
- Query strings
- Readline
- REPL
- Report
- Single executable applications
- Stream
- String decoder
- Test runner
- Timers
- TLS/SSL
- Trace events
- TTY
- UDP/datagram
- URL
- Utilities

- V8
- VM
- WASI
- Web Crypto API
- Web Streams API
- Worker threads
- Zlib

[Documentación  
node.js LTS](#)