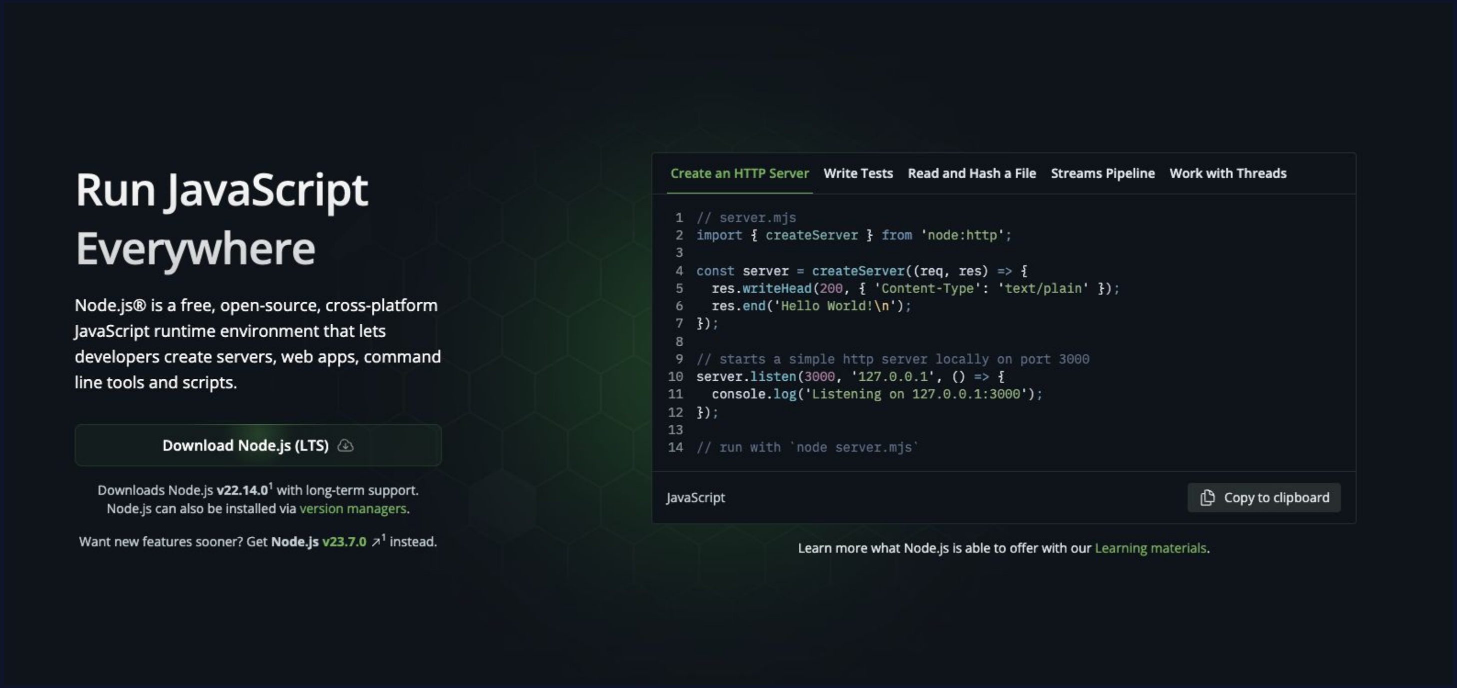
Node JS

The Nitty Gritty



Run JavaScript Everywhere

Node.js® is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command line tools and scripts.

Downloads Node.js **v22.14.0**[1] with long-term support.
Node.js can also be installed via version managers.

Want new features sooner? Get **Node.js** v23.7.0 ↗[1] instead.

```javascript
// server.mjs
import { createServer } from 'node:http';

const server = createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/plain' });
  res.end('Hello World!\n');
});

// starts a simple http server locally on port 3000
server.listen(3000, '127.0.0.1', () => {
  console.log('Listening on 127.0.0.1:3000');
});

// run with `node server.mjs`
```

Learn more what Node.js is able to offer with our Learning materials.

Node js

Open-source

Cross-platform

JavaScript runtime environment

Node js

Open-Source: Node.js is open-source, meaning its source code is freely available to the public. Anyone can view, modify, and contribute to its development. This fosters collaboration and transparency.

Cross-Platform: Node.js is designed to work on multiple operating systems, including Windows, macOS, and various Linux distributions. This cross-platform compatibility allows developers to write code that runs consistently across different environments.

JavaScript Runtime Environment: Node.js provides a runtime environment for executing JavaScript code outside of a web browser. It includes features like the V8 JavaScript engine (used by Google Chrome), which interprets and executes JavaScript code efficiently.

Key points need to know

Browser javascript and node js

Basic node command

Node js module

Browser javascript

Runs in web browsers.

Allows manipulation of the DOM (Document Object Model).

Global objects like window, document, and navigator are available.

Node js

Runs on servers and local machines.

Provides file system access for reading, writing, and more.

Uses the global object and has a different set of globals.

Basic node command

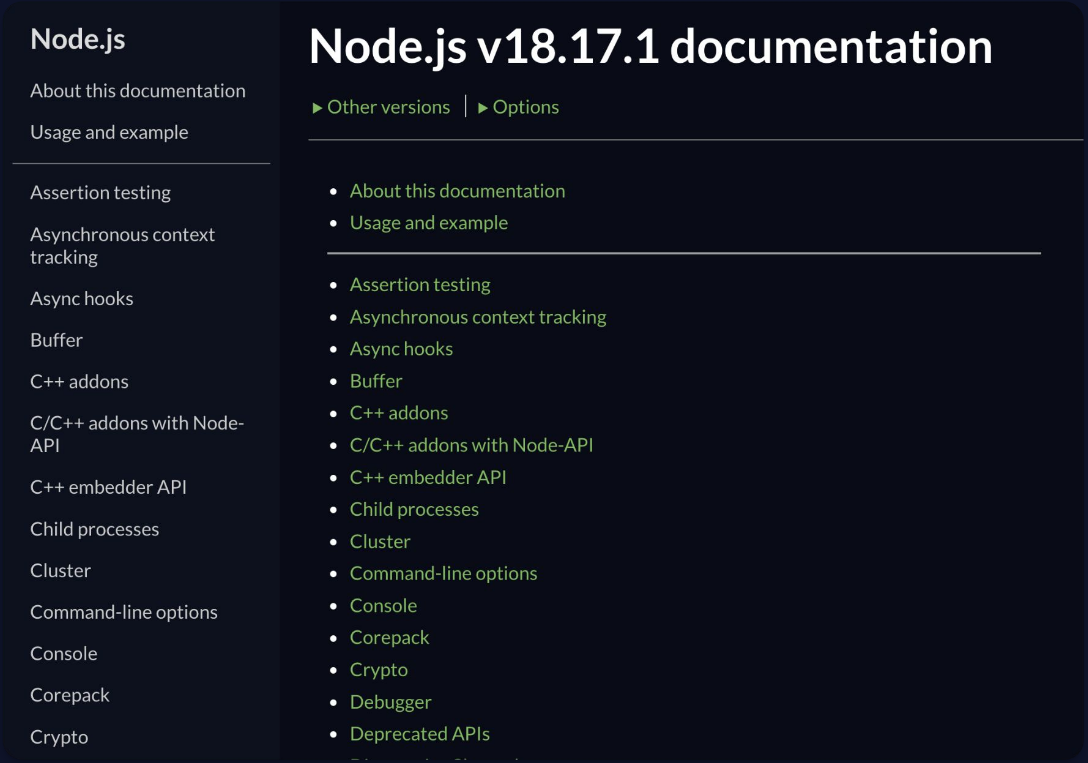
- node -v
- node
- node index.js
- npm init
- npm install ‘package’
- npm run ‘script’
- npx ‘package’

Modules

Local modules

Core modules

Third party modules


Node.js

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

Node.js v18.17.1 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

Screenshot by Xnapper.com

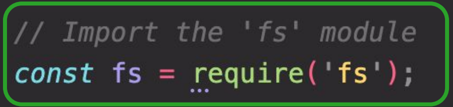
https://nodejs.org/docs/latest-v22.x/api/index.html

fs (File System):

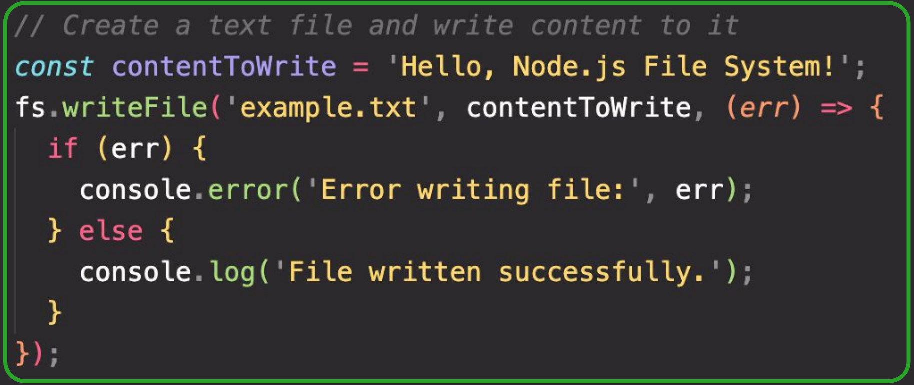
Provides methods for working with files and directories.

Examples: read, write, delete, and manipulate files.

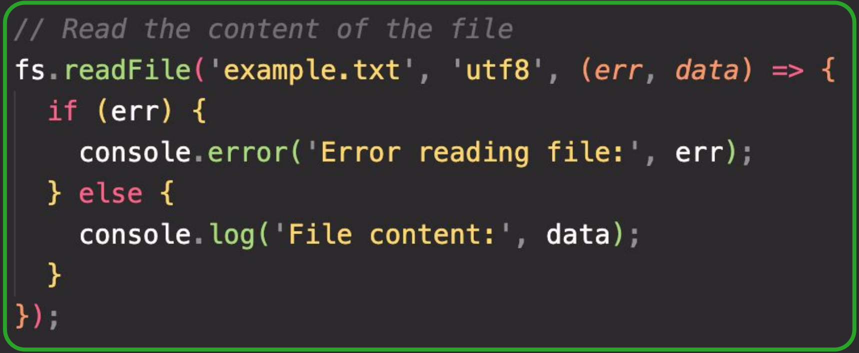
Create and read file



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



```
// Create a text file and write content to it
const contentToWrite = 'Hello, Node.js File System!';
fs.writeFile('example.txt', contentToWrite, (err) => {
  if (err) {
    console.error('Error writing file:', err);
  } else {
    console.log('File written successfully.');
  }
});
```



```
// Read the content of the file
fs.readFile('example.txt', 'utf8', (err, data) => {
  if (err) {
    console.error('Error reading file:', err);
  } else {
    console.log('File content:', data);
  }
});
```

path:

Helps with file and directory path manipulation.

Ensures cross-platform compatibility in path handling.

Handling file and path directory

```javascript
// Import the 'path' module
const path = require('path');
```

```javascript
// Create a file path
const filePath = path.join(__dirname, 'files', 'example.txt');
```

```javascript
// Normalize and display the file path
console.log('Normalized File Path:', path.normalize(filePath));

// Get the file extension
console.log('File Extension:', path.extname(filePath));

// Get the directory name
console.log('Directory Name:', path.dirname(filePath));

// Get the file name
console.log('File Name:', path.basename(filePath));
```