Learn Docs Download

Menu

Introduction to Node.js

▶ TABLE OF CONTENTS

Node.js is an open-source and cross-platform JavaScript runtime environment. It is a popular tool for almost any kind of project!

Node.js runs the V8 JavaScript engine, the core of Google Chrome, outside of the browser. This allows Node.js to be very performant.

A Node.js app is run in a single process, without creating a new thread for every request. Node.js provides a set of asynchronous I/O primitives in its standard library that prevent JavaScript code from blocking and generally, libraries in Node.js are written using non-blocking paradigms, making blocking behavior the exception rather than the norm.

When Node.js performs an I/O operation, like reading from the network, accessing a database or the filesystem, instead of blocking the thread and wasting CPU cycles waiting, Node.js will resume the operations when the response comes back.

This allows Node.js to handle thousands of concurrent connections with a single server without introducing the burden of managing thread concurrency, which could be a significant source of bugs.

Node.js has a unique advantage because millions of frontend developers that write JavaScript for the browser are now able to write the server-side code in addition to the client-side code without the need to learn a completely different language.

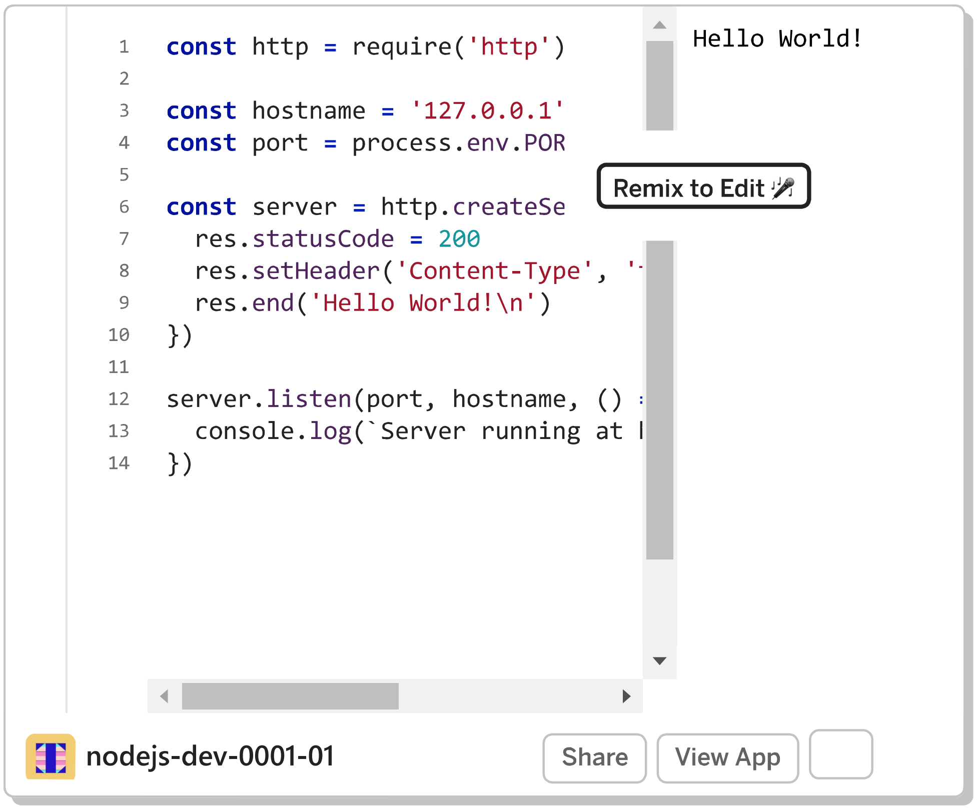
In Node.js the new ECMAScript standards can be used without problems, as you don't have to wait for all your users to update their browsers - you are in charge of deciding which ECMAScript version to use by changing the Node.js version, and you can also enable specific experimental features by running Node.js with flags.

A Vast Number of Libraries

npm with its simple structure helped the ecosystem of Node.js proliferate, and now the npm registry hosts over 1,000,000 open source packages you can freely use.

An Example Node.js Application

The most common example Hello World of Node.js is a web server:



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

const hostname = '127.0.0.1'
const port = process.env.POR

const server = http.createSe
  res.statusCode = 200
  res.setHeader('Content-Type', '
  res.end('Hello World!\n')
})

server.listen(port, hostname, () =
  console.log(`Server running at 
})
```

```
Hello World!
```

This code first includes the Node.js `http` module.

Node.js has a fantastic standard library, including first-class support for networking.

The `createServer()` method of `http` creates a new HTTP server and returns it.

The server is set to listen on the specified port and host name. When the server is ready, the callback function is called, in this case informing us that the server is running.

Whenever a new request is received, the `request` event is called, providing two objects: a request (an `http.IncomingMessage` object) and a response (an `http.ServerResponse` object).

Those 2 objects are essential to handle the HTTP call.

The first provides the request details. In this simple example, this is not used, but you could access the request headers and request data.

The second is used to return data to the caller.

In this case with:

```
res.statusCode = 200
```

we set the statusCode property to 200, to indicate a successful response.

We set the Content-Type header:

```
res.setHeader('Content-Type', 'text/plain')
```

and we close the response, adding the content as an argument to `end()` :

```
res.end('Hello World\n')
```

Node.js Frameworks and Tools

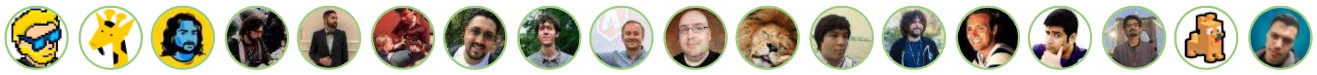
Node.js is a low-level platform. In order to make things easy and exciting for developers, thousands of libraries were built upon Node.js by the community.

Many of those established over time as popular options. Here is a non-comprehensive list of the ones worth learning:

- **AdonisJs**: A full-stack framework highly focused on developer ergonomics, stability, and confidence. Adonis is one of the fastest Node.js web frameworks.
- **Express**: It provides one of the most simple yet powerful ways to create a web server. Its minimalist approach, unopinionated, focused on the core features of a server, is key to its success.
- **Fastify**: A web framework highly focused on providing the best developer experience with the least overhead and a powerful plugin architecture. Fastify is one of the fastest Node.js web frameworks.
- **Gatsby**: A React-based, GraphQL powered, static site generator with a very rich ecosystem of plugins and starters.
- **hapi**: A rich framework for building applications and services that enables developers to focus on writing reusable application logic instead of spending time building infrastructure.
- **koa**: It is built by the same team behind Express, aims to be even simpler and smaller, building on top of years of knowledge. The new project born out of the need to create incompatible changes without disrupting the existing community.
- **Loopback.io**: Makes it easy to build modern applications that require complex integrations.
- **Meteor**: An incredibly powerful full-stack framework, powering you with an isomorphic approach to building apps with JavaScript, sharing code on the client and the server. Once an off-the-shelf tool that provided everything, now integrates with frontend libs React, Vue, and Angular. Can be used to create mobile apps as well.
- **Micro**: It provides a very lightweight server to create asynchronous HTTP microservices.
- **NestJS**: A TypeScript based progressive Node.js framework for building enterprise-grade efficient, reliable and scalable server-side applications.

- **Next.js**: A framework to render server-side rendered React applications.
- **Nx**: A toolkit for full-stack monorepo development using NestJS, Express, React, Angular, and more! Nx helps scale your development from one team building one application to many teams collaborating on multiple applications!
- **Sapper**: Sapper is a framework for building web applications of all sizes, with a beautiful development experience and flexible filesystem-based routing. Offers SSR and more!
- **Socket.io**: A real-time communication engine to build network applications.
- **Strapi**: Strapi is a flexible, open-source Headless CMS that gives developers the freedom to choose their favorite tools and frameworks while also allowing editors to easily manage and distribute their content. By making the admin panel and API extensible through a plugin system, Strapi enables the world's largest companies to accelerate content delivery while building beautiful digital experiences.

CONTRIBUTORS



EDIT THIS PAGE ON GITHUB

NEXT →

Trademark Policy
Privacy Policy
Code of Conduct
Security Reporting
About
Blog