

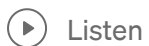
Integrating Pug with Express.js



Sung-Jie Hung | 洪崧傑 · [Follow](#)

Published in JavaScript in Plain English

4 min read · Jul 30



Objectives

Learn how to integrate Pug with Express.js by creating a simple multi-page app.

Tech Stack

Pug, express.js, node.js, nodemon

What is Pug?

Pug is a high-performance **template engine** heavily influenced by [Haml](#) and implemented with JavaScript for Node.js and browsers. Check out the official Pug doc [here](#).

Integrating with express.js

Pug fully integrates with Express, a popular Node.js **web framework**, as a supported view engine. Check out [Express's guide](#) for more detail on how to integrate Pug with Express.

Sample Usage

Let's dive right in! This blog post will illustrate how to create a straightforward **multi-page app** using Express and Pug. I assume that you have a basic understanding of using Node.js and JavaScript, so I won't go through the environment setup process.

Let's take a look at the project structure first. We have a folder called **views**, which contains all the Pug files. We also have a folder called **public**, which contains all the static files such as images, and CSS files. Last but not least, the app.js file is the entry point of this application.

```
1  .
2  └─ pug-expressjs-project/
3     └─ public/
4        └─ images/
5           └─ pug.png
6     └─ views/
7        └─ layout.pug
8        └─ about.pug
9        └─ home.pug
10    └─ package-lock.json
11    └─ package.json
12    └─ app.js
```

file_structure.txt hosted with ❤ by GitHub

[view raw](#)

App.js

First, we import the `express` module and create an instance of the `express` application. Then, we set the view engine to `Pug`. This **tells Express to use Pug as the template engine**. Next, we set the views directory to the `views` folder. This tells Express where to find the `Pug` files. Finally, we set the static directory to the `public` folder. This tells Express where to find the static files.

```
1  const express = require("express");
2  const app = express();
3
4  app.set("view engine", "pug");
5  app.use(express.static("public"));
```

app.js hosted with ❤️ by GitHub

[view raw](#)

Layout.pug

In the `views` folder, we create a `layout.pug` file. This file will be the **base template for all the pages**. It contains the basic HTML structure with the `head` and `body` tags. The `head` tag contains the title or link to CSS files. The `body` tag contains the header and the content block. The content block will be replaced by the content of each page in next step.

```
1  doctype html
2  html
3    head
4      title My Website
5    body
6      header
7        h1 My Website Header
8      block content
9      footer
10       p Sung-Jie Hung © 2023
```

layout.pug hosted with ❤️ by GitHub

[view raw](#)

Next, we create a `home.pug` file. This file will be **the home page of the application**. It extends the `layout.pug` file and replaces the content block with the content of the home page.

```
1  extends layout
2
3  block content
4    h2 Welcome to My Website
5    p This is the home page content.
```

home.pug hosted with ❤ by GitHub

[view raw](#)

We create a about.pug file. This file will be **the about page of the application**. It extends the layout.pug file and replaces the content block with the content of the about page too.

```
1  extends layout
2
3  block content
4    h2 About me
5    p This is Willie!
6    p I am a software engineer and I love pug!
7    img(he src='/images/pug.png', height='150px', width='150px')
```

about.pug hosted with ❤ by GitHub

[view raw](#)

After creating all the Pug files, we can add the **route handlers** to the app.js file. We create a route handler for the home page and the about page. The route handler for the home page renders the home.pug file. The route handler for the about page renders the about.pug file:)

```
1  const express = require("express");
2  const app = express();
3
4  app.set("view engine", "pug");
5  app.use(express.static("public"));
6
7  // Home page route
8  app.get("/", (req, res) => {
9    res.render("home");
10 });
11
12 // About page route
13 app.get("/about", (req, res) => {
14   res.render("about");
15 });
16
17 app.listen(3000);
```

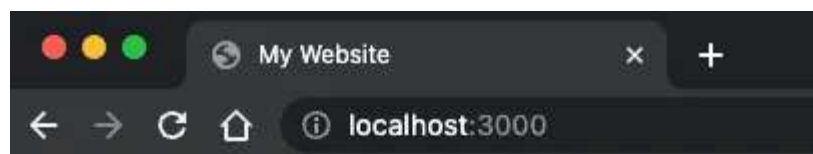
app.js hosted with ❤ by GitHub

[view raw](#)

Let's take a look at the browser

Now, we can start the server. We can see that the home page and the about page are rendered correctly. Isn't it cool?

- Home page



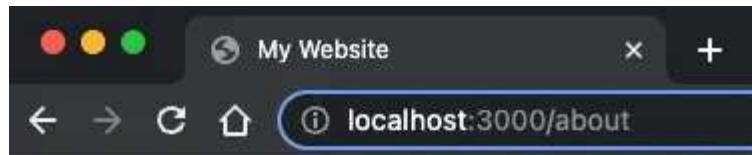
My Website Header

Welcome to My Website

This is the home page content.

Sung-Jie Hung © 2023

- About page



My Website Header

About me

This is Willie!

I am a software engineer and I love pug!



Sung-Jie Hung © 2023

Passing data from your Express.js routes

To make the Pug files more dynamic, we can **pass data from the route handlers** to the Pug files. Let's take a look at an example. We create a product route handler. The route handler passes an object to the product.pug file. The object contains the name, the price and the description of the product.

```
1 app.get("/product/:id", (req, res) => {
2   // Since we don't have a database, we'll hardcode the product information here
3   const product = {
4     id: req.params.id,
5     name: "Macbook Pro",
6     price: 1000,
7     description: "A laptop for code monkey like me 🐵",
8   };
9
10  res.render("product", { product: product });
11  });
```

app.js hosted with ❤️ by GitHub

[view raw](#)

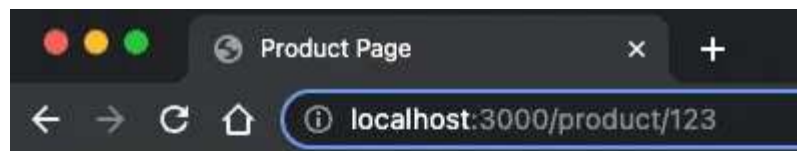
The `product.pug` file renders the page using the object passed from the route handler. It uses the Pug syntax to create a dynamic view.

```
1  doctype html
2  html
3    head
4      title Product Page
5    body
6      h1= product.name
7      p Price: ${product.price}
8      p Description: ${product.description}
```

product.pug hosted with ❤ by GitHub

[view raw](#)

Let's take a look at the browser. We can see that the product page is all set up!



Macbook Pro

Price: \$1000

Description: A laptop for code monkey like me 🐵

Implement Logic

We can also use the **Pug syntax to implement logic**. Let's take a look at another example. We create a profile route handler. The route handler passes an object to the `profile.pug` file. The object contains the name, the age and the `isAdmin` property of the user.

```
1 // User page route
2 app.get("/profile", (req, res) => {
3   // Since we don't have a database, we'll hardcode the user information here
4   const user = {
5     name: "Sung-Jie",
6     age: 24,
7     isAdmin: true,
8   };
9
10  res.render("profile", { user: user });
11 });
```

app.js hosted with ❤️ by GitHub

[view raw](#)

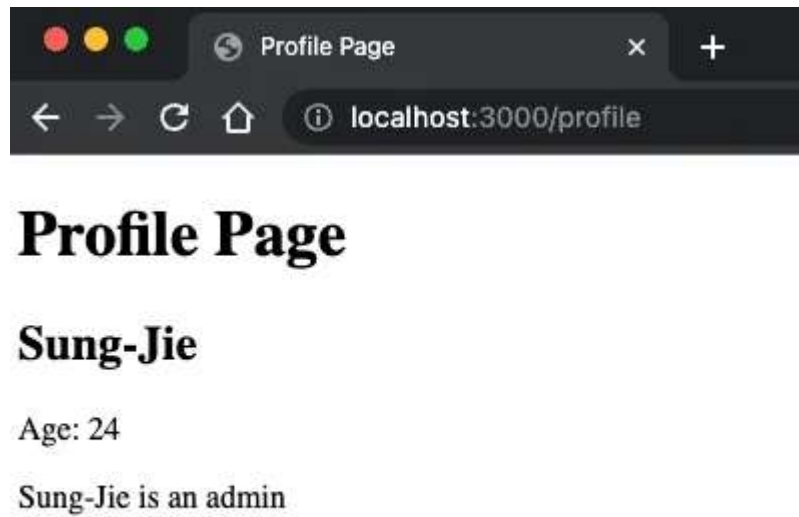
The profile.pug file renders the page using the object passed from the route handler. It uses the Pug syntax to implement logic. If the user is an admin, it will display the admin message. Otherwise, it will display the normal message.

```
1 doctype html
2 html
3   head
4     title Profile Page
5   body
6     h1 Profile Page
7     h2= user.name
8     p Age: #{user.age}
9
10    if user.isAdmin
11      p #{user.name} is an admin
12    else
13      p #{user.name} is not an admin
```

profile.pug hosted with ❤️ by GitHub

[view raw](#)

Let's take a look at the browser. We can see that the profile page renders the correct message. I am the admin of the website 🐒.

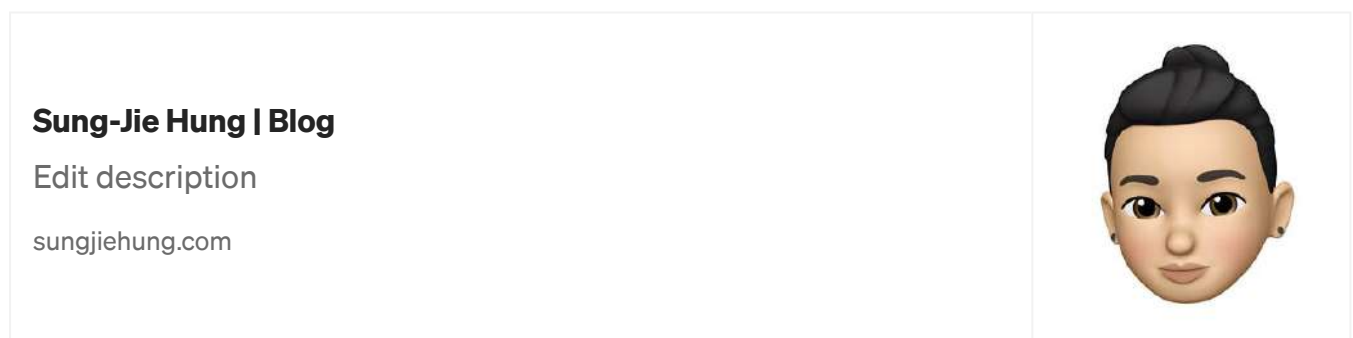


That's it! Pug helps us to implement logic in a really elegant way!

Conclusion

In this blog post, I illustrated how to create a simple multi-page app using Express and Pug. I also show how to pass data from the Express.js routes to the Pug files and how to use Pug syntax to implement logic. I hope you find this blog post useful. If you want to learn more about Pug, check out the official Pug doc [here](#). See you next time!

Check out ALL my blog posts



Resources

- [Pug Documentation](#)
- [Using template engines with Express](#)

More content at [PlainEnglish.io](https://plainenglish.io).

Sign up for our [free weekly newsletter](#). Follow us on [Twitter](#), [LinkedIn](#), [YouTube](#), and [Discord](#).

Pug

Expressjs

Nodejs

Nodemon

Template Engine



Follow



Written by Sung-Jie Hung | 洪崧傑

56 Followers · Writer for JavaScript in Plain English

CS @ UChicago | iOS & Android | <https://sungjiehung.com>

More from Sung-Jie Hung | 洪崧傑 and JavaScript in Plain English

Open in app ↗

Sign up

Sign in

●● Medium

🔍 Search



Sung-Jie Hung | 洪崧傑

Fetching data with TanStack Query (React Query)

Objectives

4 min read · Jul 19



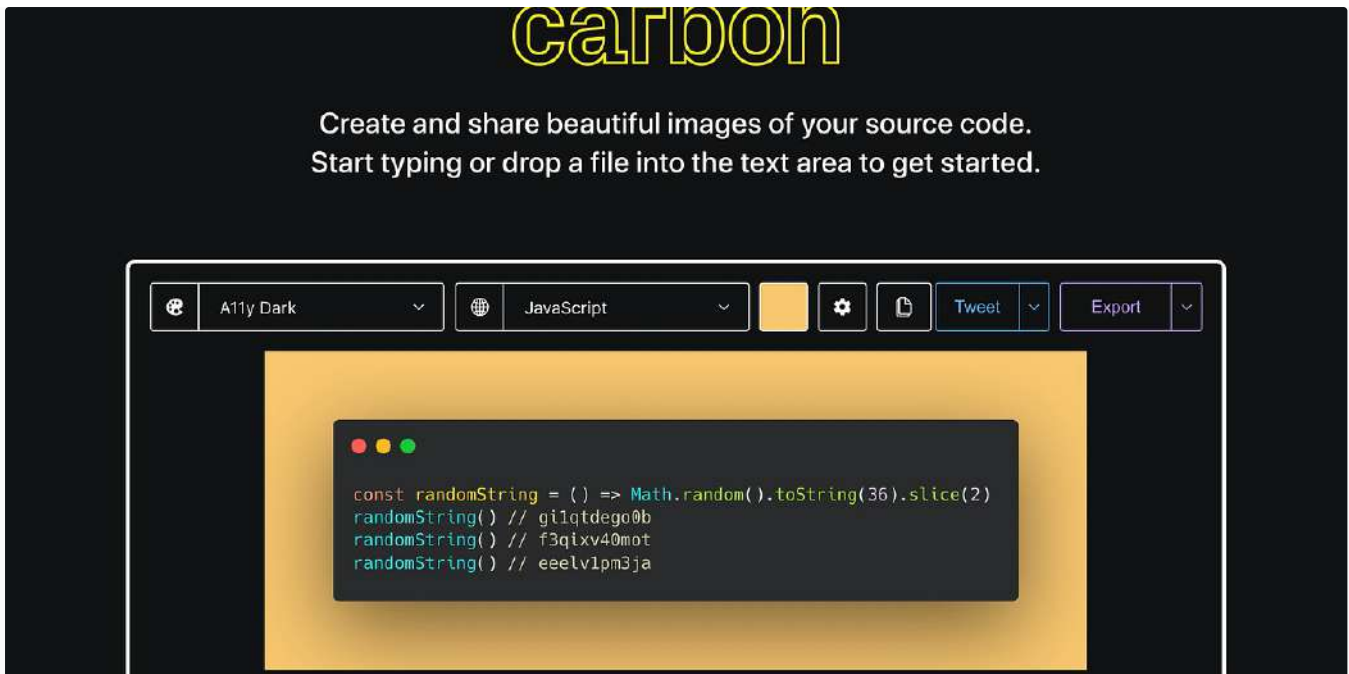
Hero in JavaScript in Plain English

Best VSCode Extensions That Every Developer Should Have (10x Productivity)

Unique extensions that will boost your productivity by 10x

7 min read · Jun 6





fatfish in JavaScript in Plain English

15 Killer Websites for Web Developers

99.9% of developers don't know all of them.

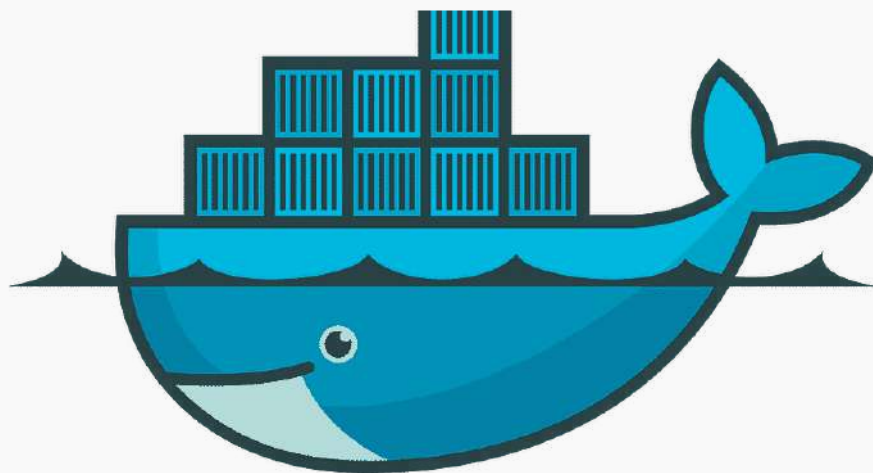
★ · 4 min read · Aug 9



2.5K



25



docker



Sung-Jie Hung | 洪崧傑

Persisting data using Docker Volume and Bind Mount

Objectives

4 min read · Jul 24



See all from Sung-Jie Hung | 洪崧傑

See all from JavaScript in Plain English

Recommended from Medium



Ankit Detroja

Unveiling JavaScript: The Double-Bang (!!)

What is the Double-Bang Operator?

1 min read · 5 days ago



133



Badih Barakat in Stackademic

Next.js API: Connect to MySQL Database

Next.js is a React framework that can be used to build static and dynamic websites and web applications. It is known for its performance...

★ · 10 min read · Oct 23

👏 48 💬



Lists



Stories to Help You Grow as a Software Developer

19 stories · 608 saves



Karthik Reddy Singireddy

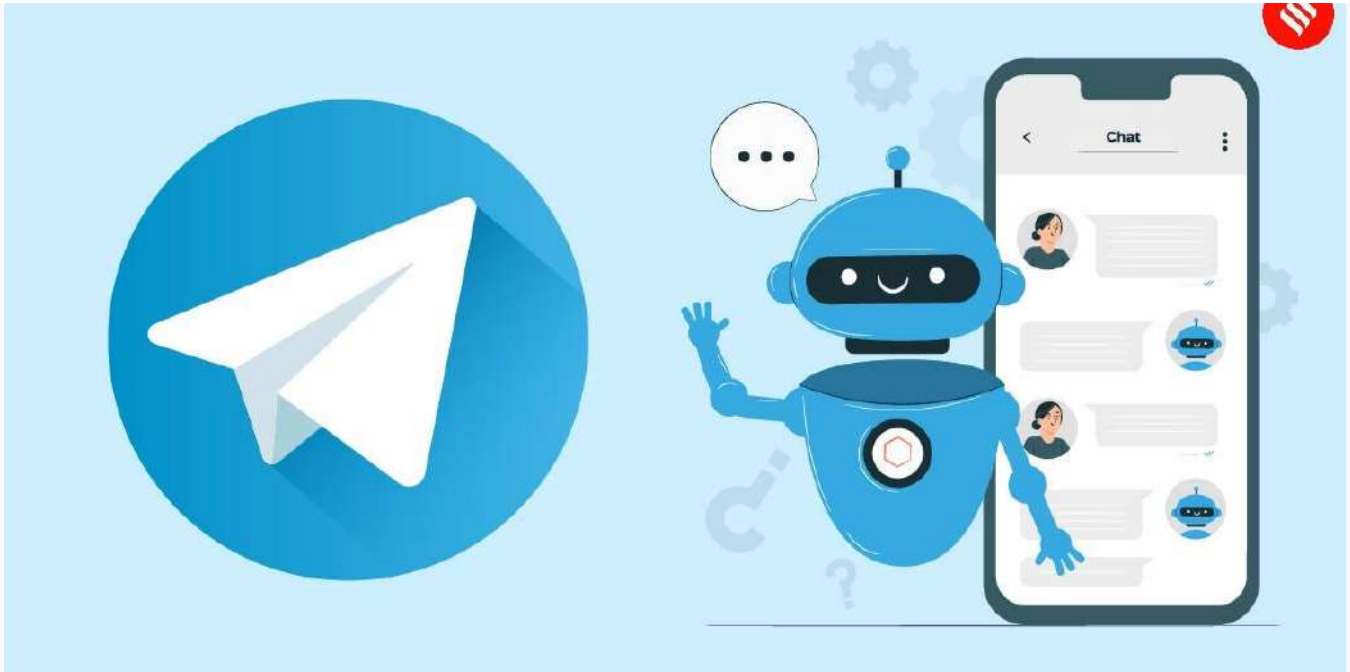
Building a RESTful API with Node.js, Express.js, Sequelize(ORM), and Swagger for CRUD Operations

A Step-by-Step Guide to Creating a Powerful RESTful API with Modern Technologies

6 min read · Jul 21

👏 17 💬





 Gabriel Rossetto

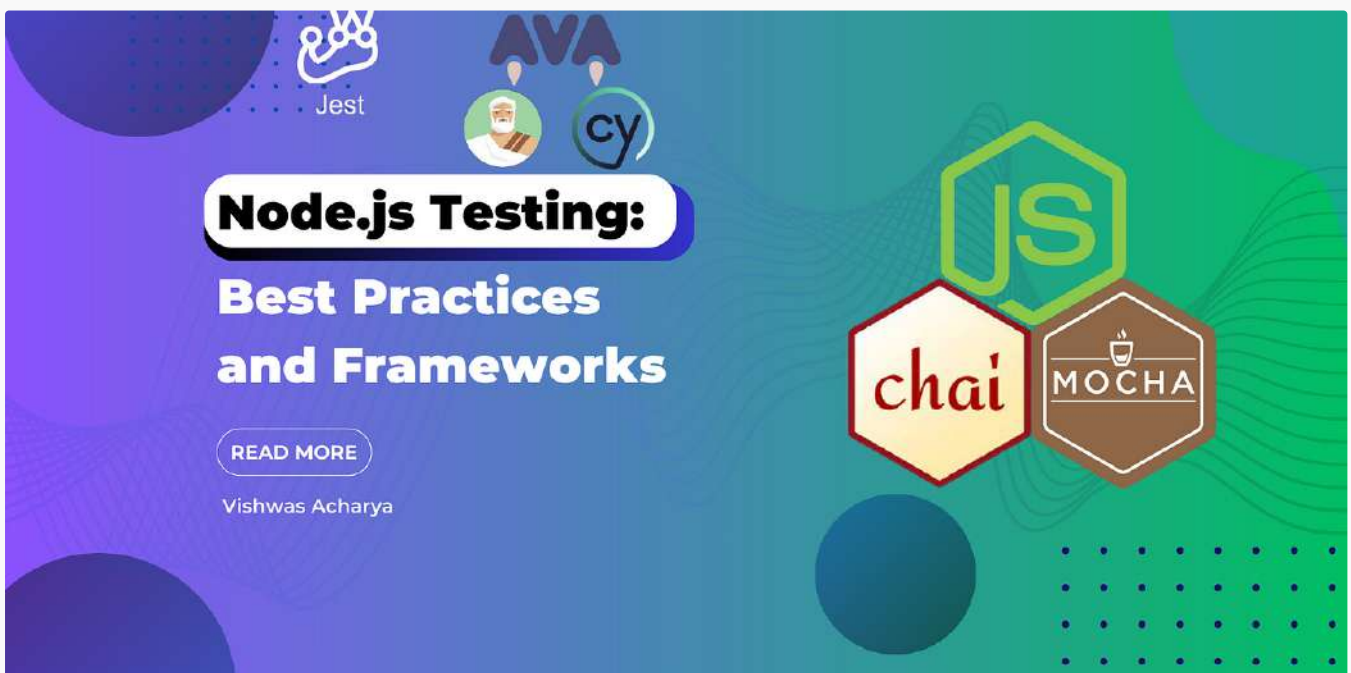
Creating a Simple Telegram Bot using Node.js: A Step-by-Step Guide


In this step-by-step guide, we'll walk through the process of creating a simple Telegram bot using Node.js. By the end of this tutorial...

3 min read · Jul 10



135



 Vishwas Acharya

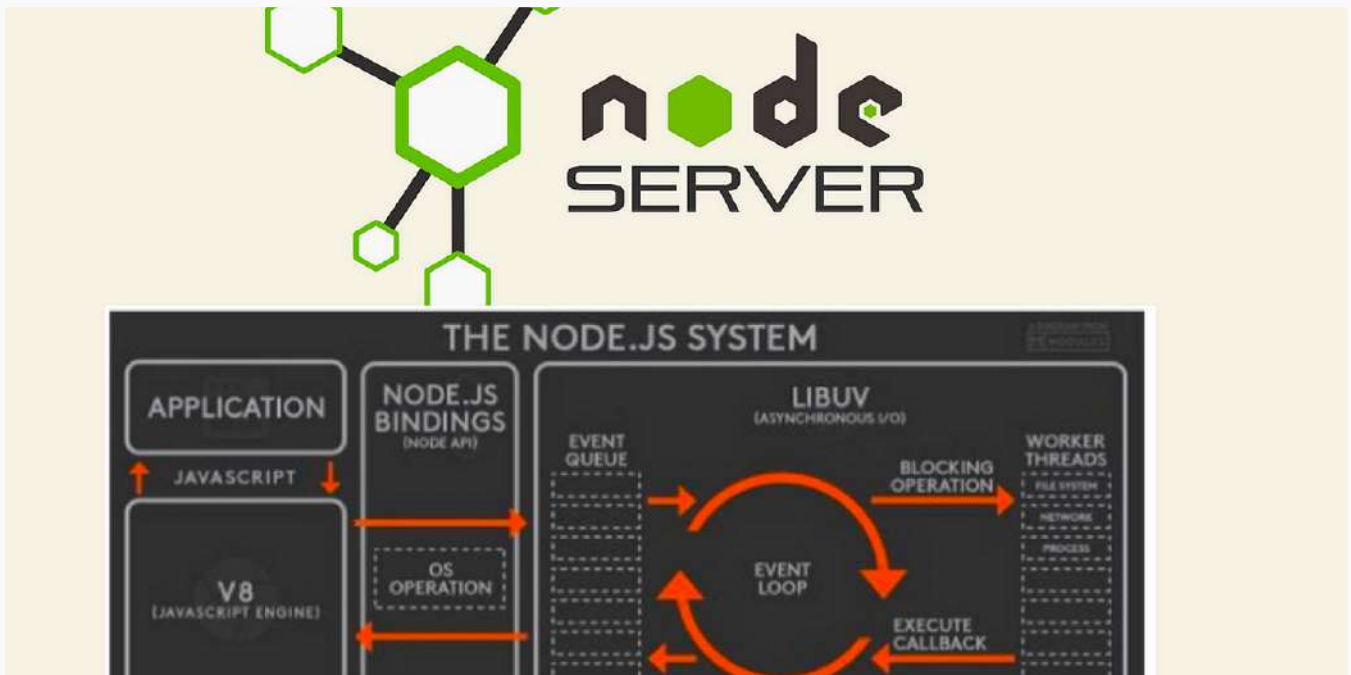
Node.js Testing: Best Practices and Frameworks for Reliable Apps


Discover the best practices and frameworks for effective Node.js testing. Ensure reliability and quality in your applications.

10 min read · Jul 10



30



 Vitaliy Korzhenko

Creating a Simple Web Server in Node.js

In this article, we will explore the process of building an HTTP server from scratch in Node.js. We will dive into the fundamentals of...

2 min read · Jul 3



20



See more recommendations