

Node.js and Express.js Simplified

A Practical Guide to Building, Testing, and Deploying Web Applications

OLUWATOBI SOFELA

Node.js and Express.js Simplified

A Practical Guide to Building, Testing, and Deploying Web Applications

This book is available at https://leanpub.com/node-and-express-simplified



© 2026 Oluwatobi Sofela

All rights reserved.

Except as permitted by applicable copyright law, no part of this publication may be reproduced, stored in a retrieval system, distributed, or transmitted in any form or by any means—including electronic, mechanical, photocopying, recording, or otherwise—without prior written permission from the publisher. Brief quotations may be used for reviews, commentary, and other purposes permitted by law.

Code examples in this book may be used, modified, and incorporated into your own personal or commercial software projects unless otherwise stated. This permission does not grant the right to reproduce, republish, distribute, or sell substantial portions of the book or its accompanying materials.

This book is provided for educational and informational purposes. Although reasonable care has been taken to ensure the accuracy of the information presented, software, libraries, APIs, platforms, services, and best practices may change over time. Readers should consult the relevant official documentation when appropriate.

This publication is provided “as is” without warranties of any kind, either express or implied. To the fullest extent permitted by applicable law, the author and publisher shall not be liable for any loss, damage, or other consequence arising directly or indirectly from the use of, or reliance on, the information or code contained in this book.

Tweet This Book!

Please help CodeSweetly by spreading the word about this book on Twitter!

The suggested hashtag for this book is #NodeJS.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#NodeJS

Also By CodeSweetly

Learn Coding Visually
Code React Sweetly
Creating NPM Package with TypeScript
Creating NPM Package with Vanilla JavaScript
Creating NPM Package with React TypeScript
The CSS Grid Guidebook
Visual CSS Grid
Visual CSS Flexbox
Creating NPM Package with ReactJS
CSS Flexbox

Contents

Introduction **1**
Welcome to Node.js and Express.js! 1
What You'll Learn 1
How You'll Learn 2
What You Should Already Know 2
The Tools You'll Need 2
A Personal Note 3
Let's Begin! 3

Getting Started with Node.js **5**
Install Node.js on Your System 6
Create a New Directory for Your Project 7
Create a package.json File 7
Why Node.js 8
Node's Module Support 12
Create HTTP Servers with Node.js 13
The `http.ServerResponse` APIs 20
How to Run JavaScript Code on the Command Line 31
What Can You Build with Node.js? 39

File Management in Node.js **41**
Types of Interfaces in the File System Module 41
Add the Callback-Based File System Module to Your Node.js App . . 41
Add the Promise-Based File System Module to Your Node.js App . . 41
Important File System Module Methods for Managing Files in Node.js 41
The node:fs Module Gives You APIs to Work with a Computer's File System 42

File Uploads in Node.js **43**
Create a New Project Directory 43
Create a `package.json` File 43

Create a Form for Uploading Files 43
Install the Formidable Package 43
Use the Formidable Module in Your Node.js Application 43
Run Your Application 43
What Is the Formidable `parse()` Method? 44
Specify Where Formidable Should Save the Uploaded Files 44

URL Management in Node.js 45
How to Use the Node.js `url` Module 45
URL Constructor – Parse URLs 45
fileURLToPath Function – Resolve File URLs to Paths 48
pathToFileURL Function – Resolve Paths to URLs 48
The `node:url` Module Provides APIs for Working with URLs 49

Serve Webpages in Node.js 50
Create a New Project Directory 50
Create the HTML Files for the Webpages You Wish to Serve 50
Configure a Web Server to Respond to Browser Requests by Serving Webpages 51
Run Your Application 51

Event Management in Node.js 52
Add the Events Module to Your Node.js Application 52
Create and Trigger Events with the `EventEmitter` API 52
Important Emitter Methods for Managing Events in Node.js 52
The `node:events` Module Provides APIs for Event Management . . 55

Use Express.js to Simplify HTTP Request Handling in Node 56
Create a Project Directory 56
Create a package.json File 56
Configure the Project as an ES Module Application 56
Why Express.js? 56
Raw Node.js Code vs. Express.js Code 57
Important Stuff to Know About Creating Express.js Web Servers in Node.js 57

Routes, Middleware, and Controllers in Express.js 59
Syntax of a Route in Express 59
Middleware vs. Route Handler 59
Examples 59

Important Things to Know about the `next` Parameter 60
Categories of Middleware in Express.js 60

Route vs. Query Parameters in Express.js 62
Create a New Project Directory 62
Create a package.json File 62
Configure the Project as an ES Module Application 62
Install Express 62
Configure a Web Server to Respond to Browser Requests 62
What Is a Query Parameter in Express? 62
What Is a Route Parameter in Express? 63

Router in Express.js 65
Create a New Project Directory 65
Create a `package.json` File 65
Configure the Project as an ES Module Application 65
Install Express 65
Group Related Routes and Middleware 65
Configure the Main Express Application 66
Run Your Express.js Application 66
Important Things to Know about Express Routers 66

Views in Express.js 68
Create a New Project Directory 68
Create a package.json File 68
Configure the Project as an ES Module Application 68
Install Express 68
What Are Static HTML Files in Express.js? 68
What Are Dynamic Views in Express.js? 70

Handling Form Data in Express.js 74
Validation vs Sanitization vs Output Escaping: What's the Difference? 74
Types of Form Validation 74
What Is Client-Side Form Validation? 74
What Is Server-Side Form Validation? 74
How to Use Express-Validator to Validate a Form's Data 74
Important Things to Know About Validation, Sanitization, and Output Escaping in Express 76
POST vs GET Method Attributes in HTML Forms 77

Connect a Frontend Application to a Node.js Backend **79**
Create a New Project Directory 79
Create the Frontend and Backend Project Directories 79
Configure the Project's Backend 79
Configure the Project's Frontend 80
Configure CORS in the Backend 80
Run Your Full-Stack Application 81
Send Data from the Frontend to the Express Backend 81
Use `express.json()` to Parse a JSON Request Body 82

Deploy a Full-Stack Application **83**
Deploy Your Backend Application to Render 83
Deploy Your Frontend Application to Vercel 84

Manage User Sessions in Express with Express-Session **87**
Create a Basic Express Application 87
Inspect Your App's Cookies in the Browser 88
Add Session Support with express-session 88
Understand the Three Session-Related Properties 88
Explore the Session Properties in Action 89

Test Express HTTP Endpoints with Supertest and node:test **90**
What's the Difference Between a Test Runner and an HTTP Testing Tool? 90
Create a Project Directory 90
Create a `package.json` File 90
Configure the Project as an ES Module Application 90
Install Express 90
Install the HTTP Testing Library 90
Test HTTP Requests 91
Run the App Script 91
Use the `node:test` Module to Run Tests 91
Configure the App's Web Server 92
Run the Express.js Web Server 92
Important Things to Know about Testing HTTP Endpoints 92

Use TypeScript with Express.js **94**
Create a Project Directory 94
Create a package.json File 94
Configure the Project as an ES Module Application 94

Install Express 94
Install TypeScript and Type Definitions 94
Configure a Web Server to Respond to Browser Requests 94
Run the Express Web Server 95

Add ESLint to an Express Application 97
Initialize ESLint's Configuration 97
Lint Your Code 97
Customize ESLint Rules 97
Specify the Files ESLint Should Ignore 97

Epilogue 98
What Next? 98
Keep Growing 98
One Last Favor 98
Final Words 98

Introduction

Welcome to Node.js and Express.js!

You know how to write JavaScript. Now you want to use it to build the part of a web application that receives requests, processes data, and sends responses back to the browser.

Node.js and Express.js help you take that next step. Node.js lets you run JavaScript outside the browser and provides tools for tasks such as working with files and creating web servers. Express.js is a framework built on Node.js that simplifies handling HTTP requests and organizing your application's routes.

But learning how these pieces fit together can feel overwhelming. With so many APIs, packages, tools, and concepts to understand, it's easy to feel lost.

Node.js and Express.js Simplified gives you a practical path through the essentials. You'll learn how the code works, why you need it, and how to apply it as you build, connect, test, and deploy web applications.

What You'll Learn

The book begins with Node.js fundamentals, then introduces Express.js and shows how it simplifies common web development tasks. Along the way, you'll learn to:

- Work with files, handle file uploads, manage URLs, serve webpages, and respond to events.
- Handle HTTP requests with Express.js and use routes, route parameters, and query parameters.
- Organize application code with routers, controllers, and views.
- Receive and process data submitted through forms.
- Connect a frontend to a Node.js and Express.js backend.

- Test Express.js routes and deploy your applications so others can access them.

Together, these skills will help you understand how a browser and server work together, while also giving you a foundation for building your own applications.

How You'll Learn

This book pairs explanations with practical examples so you can see each concept in action. You'll work through the code step by step, observe the results, and learn how individual pieces contribute to a working application.

As you read, run the examples yourself. Change a value, try a different request, or adjust a route and see what happens. These small experiments help you understand the code well enough to adapt it to your own needs.

If you're new to Node.js and Express.js, follow the chapters in order. Take your time with unfamiliar ideas, and revisit earlier examples whenever you need a refresher. The goal is to become comfortable making decisions, investigating problems, and building on what you've learned.

What You Should Already Know

This book is for readers who know basic JavaScript and want to start building web applications with Node.js and Express.js. You'll get the most value from it if you're familiar with:

- JavaScript fundamentals, including variables, functions, arrays, and objects.
- Basic asynchronous JavaScript, including callbacks, promises, and `async/await`.
- Basic HTML, including links, forms, and input elements.
- Using a terminal to navigate folders and run commands.

You don't need previous experience with Node.js or Express.js. Familiarity with installing npm packages is helpful, but the examples will guide you through the commands you need.

The Tools You'll Need

Before you begin, have a code editor, a web browser, and the following installed:

- Node.js 24.15.0 or later.
- npm 11.14.0 or later.

Check your installed versions by running:

Figure 1. :

```
node --version && npm --version
```

Node.js includes npm, but the bundled version may be older than the one listed here. Check both versions and update npm separately if needed. The CodeSweetly package manager guide explains how to install, update, and check these tools.

When a chapter specifies a dependency version, use that version as you follow along. This keeps your setup aligned with the example and reduces differences caused by package updates. Newer major releases of Node.js, npm, or other dependencies may introduce changes, so a higher version number does not guarantee that every example will behave identically.

The file-creation commands assume an environment with `mkdir` and `touch` available, such as Bash or zsh on macOS or Linux, Git Bash on Windows, or a Linux shell through Windows Subsystem for Linux (WSL). You can also create the files and folders directly in your editor.

A Personal Note

At CodeSweetly, I write to make software development concepts easier to understand and apply. That same goal guides this book: clear explanations, useful examples, and room to learn at your own pace.

If a concept doesn't click the first time, return to the example and work through it in smaller steps. Understanding often grows through a few careful experiments. You can also find more tutorials and explanations at CodeSweetly.

Let's Begin!

By the end of this book, you'll have practiced building, connecting, testing, and deploying Node.js and Express.js applications. You'll also have a stronger understanding of how the pieces fit together and ready to apply that knowledge to your own projects.

Open your code editor, and let's get started with Node.js.

Getting Started with Node.js

Node.js is a runtime environment for executing JavaScript outside a web browser. It makes JavaScript usable in contexts beyond just web pages.

Node.js is neither a framework nor a programming language. Instead, it is an asynchronous, event-driven environment that provides components and APIs such as:

- JavaScript engine (Google's V8)
- Module systems (CommonJS and ES modules)
- Operating system APIs (`os`)
- File system APIs (`fs`)
- Networking APIs (`http`, `https`, and `net`)
- Memory and diagnostic APIs
- Event loop

These components make up Node.js's core infrastructure, enabling JavaScript to run outside of browsers.

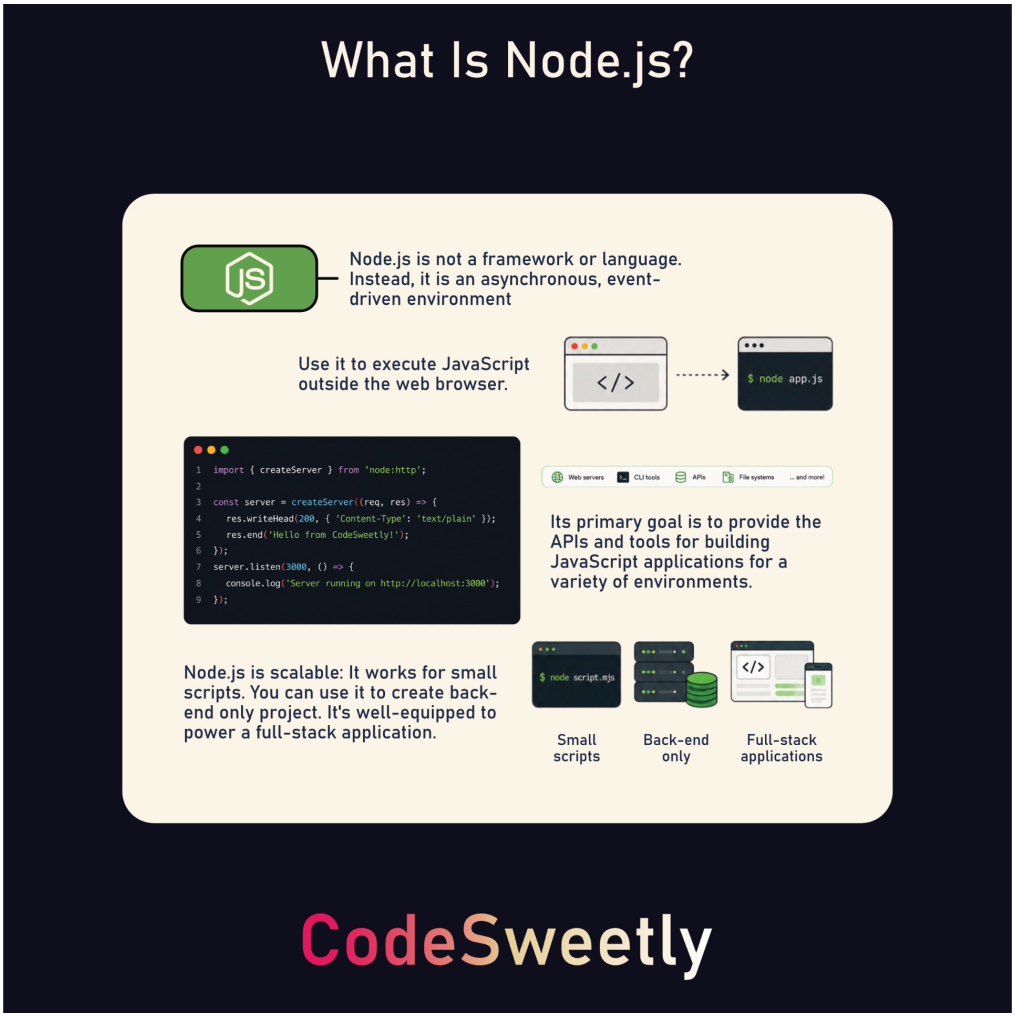

Figure 2. Node.js is a runtime environment providing the infrastructure to build and run any scale of JavaScript applications outside the browser, including servers, CLIs, scripts, and full-stack applications.

This guide will help you understand Node.js's purpose by guiding you through a basic application setup. Let's begin with installing Node.js.

Install Node.js on Your System

The first step to creating a Node.js application is to have Node on your system. So, go to the Node.js website and install the latest LTS (long-term support) version.

After installation, run the following command in your terminal to display the Node.js version installed on your system.

Figure 3. :

```
node -v
```



The `-v` flag in the snippet above is shorthand for `--version`.

Once you've successfully installed Node.js on your system, we can proceed to create the project directory.

Create a New Directory for Your Project

Use the `mkdir` CLI command to create a new project directory as follows:

Figure 4. :

```
mkdir codesweetly-nodejs-app-001
```



You can use any name you prefer. In this guide, we'll use `codesweetly-nodejs-app-001` for demonstration.

Afterward, navigate to your project directory using the command line.

Figure 5. :

```
cd codesweetly-nodejs-app-001
```

Create a package.json File

Use npm to initialize a `package.json` file after navigating into the project directory.

Figure 6. :

```
npm init -y
```

Next, use the following command to delete the optional `main` field from `package.json`:

Figure 7. :

```
npm pkg delete main
```



The `main` field in `package.json` identifies a package's entry point when another program loads the package. We do not need it for this project because we will run our scripts directly. Leaving it in place is also harmless.

Why Node.js

JavaScript was originally created for web browsers, but other environments also provide the infrastructure to execute it.

Node's 2009 release provided an environment for running JavaScript outside the web browser. For example, let's create a script to run from our system's terminal.

1. Create a JavaScript file

Create the JavaScript file you want Node to run.

In a shell that supports `touch`, such as Bash or zsh, use the following command. Otherwise, create the file in your code editor:

Figure 8. :

```
touch console.js
```

2. Write your JavaScript program

Open the newly created JavaScript file and write your program:

Figure 9. console.js

```
console.log("=== Hello from the CodeSweetly Team! ===");
console.log("We hope you have fun Coding Sweetly with Node.js.");
console.log("Thank you for being part of the CodeSweetly community.");
console.log("=== Keep coding. Keep creating. Keep shipping. ===");
```

3. Run your JavaScript program

Installing Node.js makes the `node` command available for running a Node application from your command line.

Figure 10. :

```
node console.js
```

- `node`: The command for running Node.js scripts or the REPL.
- `console.js`: The JavaScript file you want Node to run.

You can also add the command to the `"scripts"` field of your project's `package.json` file:

Figure 11. package.json (line 3)

```
{
  "scripts": {
    "start": "node console.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  }
}
```

With this script in place, you can run your JavaScript program from your terminal like this:

Figure 12. :

```
npm run start
```

Once you execute your script, Node will print the file's output to your terminal. It will look like this:

Figure 13. :

```
$ npm run start

> codesweetly-nodejs-app-001@1.0.0 start
> node console.js

=== Hello from the CodeSweetly Team! ===
We hope you have fun Coding Sweetly with Node.js.
Thank you for being part of the CodeSweetly community.
=== Keep coding. Keep creating. Keep shipping. ===
```

As you can see, we've successfully executed JavaScript outside a web browser. That's precisely what Node.js helps us with. It is an asynchronous, event-driven runtime environment for running JavaScript code outside the web browser. You can also automate rerunning your program. Let's discuss how.

4. Automatically rerun your JavaScript program

By default, Node.js requires you to manually rerun your JavaScript file each time you make changes.

Repeating the manual process of executing your application as you make changes can be burdensome. Luckily, Node provides the `--watch` flag for automating the process:

Figure 14. :

```
node --watch filename.extension
```

The snippet above uses the `--watch` flag to start Node.js in watch mode. This causes Node to re-execute the specified script whenever you update any of the files in its dependency graph.

- `node`: The command for running Node.js scripts or the REPL.
- `--watch`: The flag for activating Node's watch mode.
- `filename.extension`: The script you want Node to execute.



Stop the running process using `Ctrl + C` on Windows, macOS, or Linux.

The `--watch` flag, by default, watches the entry point and all the modules it depends on. In other words, the watch command causes Node to monitor:

- The entry file (`filename.extension`)
- All files imported by the entry file
- All files imported in those imports
- And so on through the whole dependency graph

Node's default watch mode does not watch unrelated files. If you want Node to watch other files, use the `--watch-path` flag:

Figure 15. :

```
1  node --watch-path=./src --watch-path=./tests filename.extension
```

The `--watch-path` flag, in the snippet above, tells Node to watch all files in the `src` and `tests` directories, even if they are not in the entry point's dependency graph.

- `node`: The command for running Node.js scripts or the REPL.
- `--watch-path`: The flag for specifying the path you want Node to watch for changes.
- `filename.extension`: The script you want Node to execute.

Tip

- The `--watch-path` flag causes Node to ignore the dependency graph's modules unless they are part of `--watch-path`. Instead, Node.js will watch only the exact paths you specified, and nothing else.
- `--watch-path` enables watch mode itself. If you also pass `--watch`, Node still watches only the specified paths.
- `--watch-path` is supported on macOS and Windows, but not Linux. On Linux, use `--watch` to monitor the entry file and its dependencies.
- Watch commands do not reload the browser automatically. They simply detect file changes, so Node reruns the program you specified. As

> such, if your program sends data to browsers, you will need to refresh the browser yourself or use a third-party tool for auto-reloading.

Node.js supports two module systems. Let's learn about them.

Node's Module Support

Node.js supports both CommonJS (`.cjs`) and ECMAScript (`.mjs`) modules. This allows you to use your preferred module type to create Node.js applications.

For example, below is a `script.mjs` JavaScript file. Node will treat it as an ECMAScript module because it has a `.mjs` file extension.

Figure 16. script.mjs

```
1 import http from "node:http";
```

On the other hand, Node will regard the `script.cjs` JavaScript file below as a CommonJS module because it has a `.cjs` file extension.

Figure 17. script.cjs

```
1 const http = require("node:http");
```

Suppose you want to specify the module type for the `.js` files in your project. In that case, specify a `type` field in your `package.json` file like so:

Figure 18. package.json (line 6)

```
{
  "scripts": {
    "start": "node console.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "type": "module",
  "license": "ISC"
}
```

The `"type": "module"` field in the snippet above makes Node treat `.js` files governed by this `package.json` as ES modules. A nested `package.json` establishes its own scope. `.mjs` and `.cjs` files retain their respective module types.

Set the `"type"` field to `"commonjs"` to make Node treat `.js` files in that scope as CommonJS.

Some notes:

- The ES module system is the official standard for JavaScript.
- Kevin Dangoor started the project that became CommonJS in January 2009, before JavaScript had an official standard module system.
- ES modules became part of the ECMAScript standard in 2015. CommonJS remains supported in Node.js.

In this guide, we'll mainly use Node.js with ES modules, since ES modules are JavaScript's standard module system. To follow the examples, make sure the `"type"` field in your `package.json` file is set to `"module"`. You can do this by running the following command from your project directory:

Figure 19. :

```
npm pkg set type=module
```

When relevant, we'll compare CommonJS syntax.

Let's now discuss using Node.js to create web servers that receive and respond to requests from browsers.

Create HTTP Servers with Node.js

An HTTP server lets your JavaScript application handle requests from clients, such as browsers, and send responses.

There are three basic steps to setting up an HTTP server:

1. Create a new HTTP server instance.
2. Specify the port where you want the server to listen for connections.
3. Define how the server should respond to client requests.

Let's discuss the three steps in detail. To start, create an ES module for your project.

Figure 20. :

```
touch server.js
```

Afterward, open the newly created module and initialize a new instance of the HTTP `Server` object.

Initialize a new instance of the HTTP `Server` object

Node provides the `createServer` method for creating an instance of the HTTP `Server` object (`http.Server`).

You can use it in your project by importing it from Node's `http` module as follows:

Figure 21. server.js

```
import { createServer } from "node:http";

const server = createServer();
```

Here's what's going on:

- `import` statement: Imports the `createServer` API from Node's `http` library to the `server.js` ES module.
- `server`: A variable for storing the HTTP `Server` object that the `createServer()` function outputs.

Here's the CommonJS alternative:

Figure 22. server.cjs

```
const http = require("node:http");

const server = http.createServer();
```

Once you have created the local HTTP `Server` object, specify the server's port.

Configure the system's port where the server should run

The HTTP `Server` object provides a `listen()` method to configure the port on which the server should run and accept client requests.

Syntax

Figure 23. (line 5)

```
import { createServer } from "node:http";

const server = createServer();

server.listen(port, hostname, backlog, callback);
```

This form of the `listen()` method accepts the following arguments:

- `port`: (number) The port number where the server should run and listen for client requests. If omitted, the operating system will assign any unused port. You can use `server.address().port` to retrieve the port the server is listening on after it starts listening for client connections.
- `hostname`: (string) The hostname or IP address on which the server should accept client connections. If omitted, the server will default to either an unspecified IPv6 (`::`) address or an IPv4 (`0.0.0.0`) address. These unspecified addresses bind to all network interfaces for the applicable address family. You can use `server.address().address` to retrieve the bound IP address once it starts listening for client connections.
- `backlog`: (number) Maximum length of the pending connections' queue. 511 is the default value.
- `callback`: (function) The function to execute once the server starts listening for client requests.

Example

Figure 24. server.js

```
import { createServer } from "node:http";

const server = createServer();

server.listen(3000, "127.0.0.1", 511, () => {
  const info = server.address();
  console.log(`Server running at http://${info.address}:${info.port}/`);
});
```

Here's what's going on:

- `import` statement: Imports the `createServer` API from Node's `http` library to the `server.js` ES module.
- `server`: A variable for storing the HTTP `Server` object that the `createServer()` function outputs.
- `server.listen()`: Starts the server to listen for client connections. (Tip: The method emits a `listening` event once the server starts successfully.)

Here's the CommonJS alternative:

Figure 25. server.cjs

```
const http = require("node:http");

const server = http.createServer();

server.listen(3000, "127.0.0.1", 511, () => {
  const info = server.address();
  console.log(`Server running at http://${info.address}:${info.port}/`);
});
```

What is the `127.0.0.1` address?

The `127.0.0.1` address is an IPv4 loopback address that refers to your local computer. The hostname `localhost` also refers to your local computer, but it may resolve to `127.0.0.1` or the IPv6 loopback address, `::1`. You can use it as follows:

Figure 26. server.js

```
import { createServer } from "node:http";

const server = createServer();

server.listen(3000, "localhost", 511, () => {
  console.log(`Server running at http://localhost:3000/`);
});
```

If you run this server and visit its web address, the browser will wait for a response because we have not yet set up a request handler. Let's configure that now.

Configure the server to respond to client requests

The `createServer()` method accepts a `requestListener` callback that is invoked automatically whenever the server receives an HTTP request. Node allows you to use this callback to respond to requests.

Syntax

The `createServer()` method accepts two optional arguments. Here's the syntax:

Figure 27. (line 3)

```
import { createServer } from "node:http";

const server = createServer(options, callback);
```

- `options`: An object for customizing the server's behavior.
- `callback`: The `requestListener` function for handling and responding to client requests. It accepts two parameters:

Figure 28. :

```
import { createServer } from "node:http";

const server = createServer(options, function (request, response) {
  // the requestListener function's body
});
```

- `request` parameter: An `http.IncomingMessage` object that provides details about the client request.
- `response` parameter: An `http.ServerResponse` object for responding to the client requests.

Providing the `requestListener` callback function as `createServer`'s second argument causes Node.js to automatically register it as a listener for the `"request"` event. So, the syntax above is equivalent to:

Figure 29. :

```
import { createServer } from "node:http";

const server = createServer(options);

server.on("request", function (request, response) {
  // the requestListener function's body
});
```

`server.on("request", callback)` tells the server to listen for a request event and execute the callback on such an event.

Example

Figure 30. server.js (lines 8–13)

```
// Add the Node.js HTTP module
import { createServer } from "node:http";

// Specify the hostname and port to run the server
const hostname = "localhost";
const port = 3000;

// Create a new Server instance with a requestListener callback
const server = createServer((req, res) => {
  res.statusCode = 200; // Set an OK success (200) response status code
  res.setHeader("Content-Type", "text/plain"); // Define the media type of the
  ↪ response data
  res.end("Hello World!"); // Specify the response data and close the response
  ↪ stream
});

// Run the server on the specified port and hostname
server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

The snippet above used the `requestListener` callback function's response parameter to configure the server to respond to client requests. Here's the `server.on("request", callback)` alternative:

Figure 31. server.js (lines 8–17)

```
// Add the Node.js HTTP module
import { createServer } from "node:http";

// Specify the hostname and port to run the server
const hostname = "localhost";
const port = 3000;

// Create a new Server instance
const server = createServer();

// Listen for a request event and execute the requestListener callback
server.on("request", (req, res) => {
  res.statusCode = 200; // Set an OK success (200) response status code
  res.setHeader("Content-Type", "text/plain"); // Define the media type of the
  ↪ response data
  res.end("Hello World!"); // Specify the response data and close the response
  ↪ stream
});

```

```
// Run the server on the specified port and hostname
server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Now that your server is set up and the port configured, you can run the application.

Figure 32. :

```
node server.js
```

- `node`: The command for running Node.js scripts or the REPL.
- `server.js`: The JavaScript file you want Node to run.

While the server is running, if users request the app's resource at the server's web address, they will see a `Hello World!` response.



Figure 33. A Node.js server uses the `requestListener` callback to respond to client requests.



Stop the running process using `Ctrl + C` on Windows, macOS, or Linux.

Let's now discuss some of the `http.ServerResponse` object's APIs you can use to respond to HTTP requests.

The `http.ServerResponse` APIs

Below are some of the `ServerResponse` APIs for responding to client requests.

response.statusCode

The `response.statusCode` property sets the status code used when Node sends implicit response headers, such as on the first `write()` or `end()` call.

Syntax

Figure 34. (line 4)

```
import { createServer } from "node:http";

createServer((request, response) => {
  response.statusCode = <number>;
});
```

`<number>` is the status code value you want to send to the client. It typically falls within one of the following categories:

- **100 – 199:** Informational responses
- **200 – 299:** Successful responses
- **300 – 399:** Redirection messages
- **400 – 499:** Client error responses
- **500 – 599:** Server error responses

`200` is the default response status code.

Example

Figure 35. server.js (line 4)

```
import { createServer } from "node:http";

const server = createServer((req, res) => {
  res.statusCode = 200; // Set 200 response status code
  res.end(`${res.statusCode}`); // Use the status code as the response data
});

server.listen(8000, () => {
  console.log(`Server running at http://localhost:8000/`);
});
```

The snippet above uses `res.statusCode` to set the response's status code to `200` (success).

response.statusMessage

The `response.statusMessage` property sets the human-readable status message used when Node sends implicit response headers.

Syntax

Figure 36. (line 4)

```
import { createServer } from "node:http";

createServer((request, response) => {
  response.statusMessage = <string>;
});
```

`<string>` is the message you want to send to the client. `undefined` is the default, which makes Node default to the status code's standard message if you do not define a `statusMessage`.

Example

Figure 37. server.js (line 5)

```
import { createServer } from "node:http";

const server = createServer((req, res) => {
  res.statusCode = 404;
  res.statusMessage = "Sorry, page not found"; // Set a response message
  res.end(`${res.statusMessage}`); // Use the status message as the response
  ↪ data
});

server.listen(8000, () => {
  console.log(`Server running at http://localhost:8000/`);
});
```

The snippet above uses `res.statusMessage` to set a human-readable status message for the server response's status code.

response.setHeader

The `response.setHeader()` method sets an HTTP response header that Node.js will send to the client when it sends the response headers.

Calling `setHeader()` does not send the header immediately. It stores the header so that Node can send it later–for example, when you call `response.write()` or `response.end()`.

Syntax

Figure 38. (line 4)

```
import { createServer } from "node:http";

const server = createServer((request, response) => {
  response.setHeader("Header-Name", "header-value");
});
```

You can also use an array to define multiple values for a single header:

Figure 39. (line 4)

```
import { createServer } from "node:http";

const server = createServer((request, response) => {
  response.setHeader("Header-Name", ["value1", "value2", "value3"]);
});
```



Suppose a header with the same name already exists among the headers waiting to be sent. In that case, Node will replace the existing value with the latest one.

Example

Figure 40. server.js (line 4)

```
import { createServer } from "node:http";

const server = createServer((req, res) => {
  res.setHeader("Content-Type", "text/html"); // Set the response header's
  ↪  content type
  res.end(`${res.getHeader("Content-Type")}`); // Use the content type's value
  ↪  as the response data
});

server.listen(8000, () => {
  console.log(`Server running at http://localhost:8000/`);
});
```

The snippet above uses `res.setHeader` to set the header that the server sends to the client in response to the client's request. The value can also be an array:

Figure 41. server.js (line 4)

```
import { createServer } from "node:http";

const server = createServer((req, res) => {
  res.setHeader("Set-Cookie", ["name=codesweetly", "age=5"]); // Set two
  ↪ cookies in the response
  res.end(`${res.getHeader("Set-Cookie")}`); // Use the cookie header values as
  ↪ the response data
});

server.listen(8000, () => {
  console.log(`Server running at http://localhost:8000/`);
});
```



`setHeader()` caches non-string values without modifying them. But Node converts non-string data to strings when sending them to the client. So, while `getHeader()` may retrieve non-string values, Node sends header values as strings.

response.writeHead

The `response.writeHead()` method sets the HTTP response's status code, status message (optional), and headers that the Node.js server will send to the client.

Syntax

Figure 42. (line 4)

```
import { createServer } from "node:http";

createServer((request, response) => {
  response.writeHead(statusCode, statusMessage, headers);
});
```

- `statusCode`: A 3-digit number representing the HTTP response's status code that the Node.js server will send to the client.

- `statusMessage`: An optional string representing the human-readable status message that the Node.js server will send to the client. If omitted, Node uses an existing `response.statusMessage` value or the status code's standard message.
- `headers`: An optional object or array representing the HTTP response headers that the server will send to the client. If you use an array, Node will interpret the even-numbered indices as header names and the odd-numbered indices as header values.



- Call `writeHead()` only once per response. After it commits the headers, they can no longer be changed.
- Call `writeHead()` before writing any response body (`write()` or `end()`) since headers precede the body in HTTP responses.
- `writeHead()` returns the same `http.ServerResponse` object as the `response` parameter of `createServer()`'s callback. So, you can chain it with other response methods like `end()`.

Example

Figure 43. server.js (lines 4–10)

```
1  import { createServer } from "node:http";
2
3  const server = createServer((req, res) => {
4    res
5      .writeHead(200, "OK", {
6        "Content-Type": "text/plain",
7        "CodeSweetly-Header": "simplified/tutorials",
8        "Your-Custom-Header": "your-custom-value",
9      })
10     .end("Hello from CodeSweetly's Server!");
11 });
12
13 server.listen(8000, () => {
14   console.log(`Server running at http://localhost:8000/`);
15 });
```

The snippet above uses `writeHead` to send three headers to the client. We also chained an `end()` method to the `ServerResponse` object returned by `writeHead()`.

Here's the array equivalent of the `headers` argument:

Figure 44. server.js (lines 4–13)

```
import { createServer } from "node:http";

const server = createServer((req, res) => {
  res
    .writeHead(200, "OK", [
      "Content-Type",
      "text/plain",
      "CodeSweetly-Header",
      "simplified/tutorials",
      "Your-Custom-Header",
      "your-custom-value",
    ])
    .end("Hello from CodeSweetly's Server!");
});

server.listen(8000, () => {
  console.log(`Server running at http://localhost:8000/`);
});
```

Let's now discuss the difference between `writeHead()`, `statusCode`, `setHeader`, and `statusMessage`.

writeHead() vs. statusCode vs. statusMessage vs. setHeader()

The `writeHead()` method sets the status code and, optionally, the status message and headers for one HTTP response. Here's how it works differently from the implicit alternatives.

Data type

- `writeHead()`: A *method* of the `http.ServerResponse` object
- `statusCode`: A *property* of the `http.ServerResponse` object
- `statusMessage`: A *property* of the `http.ServerResponse` object
- `setHeader()`: A *method* of the `http.ServerResponse` object

Purpose

- `writeHead()`: Sets the response's status code, status message (optional), and headers.
- `statusCode`: Sets (or gets) the response's status code.

- `statusMessage`: Sets (or gets) the response's status message.
- `setHeader()`: Sets the response's header. (You can use `getHeader()` to retrieve the header.)

Timing

- `writeHead()`: Commits the status and headers. It does not guarantee immediate network transmission.
- `statusCode`: Sets the status code before implicit headers are sent.
- `statusMessage`: Sets the status message before implicit headers are sent.
- `setHeader()`: Stores a header for later transmission, for example with the first `write()` or `end()` call.

Repeated use

Before the headers are committed, you can assign `statusCode` and `statusMessage` or call `setHeader()` repeatedly. The latest value replaces the previous one.

Call `writeHead()` only once per response, before `write()` or `end()`.

Precedence

The status code and any status message supplied to `writeHead()` override the corresponding properties. Its headers merge with those set by `setHeader()`, overriding values with matching names.

Caching

Headers supplied directly to `writeHead()` without prior use of `setHeader()` are not cached for retrieval with `getHeader()`. The `statusCode` and `statusMessage` properties remain readable.



- Read `response.statusCode` and `response.statusMessage` directly. They are properties, not methods.
- Use `getHeader()` to retrieve the cached headers set by `setHeader()`.

response.write

The `response.write()` method sends a chunk (portion) of the HTTP response body to the client. In other words, rather than sending all the data at once, you can use `write()` to send it in chunks. It is an efficient way to stream or send large amounts of data.

Syntax

Figure 45. (line 4)

```
1  import { createServer } from "node:http";
2
3  createServer((request, response) => {
4    response.write(chunk, encoding, callback);
5  });
```

- `chunk`: A string, buffer, or Uint8Array representing the portion of the HTTP body you want to send to the client.
- `encoding`: An optional string specifying how Node should encode a string chunk into a byte stream. `"utf8"` is the default value.
- `callback`: An optional function, which Node will call when it sends the chunk of data to the client.



- You can call `write()` multiple times to send (flush) successive chunks of data to the client.
- If you have not called `writeHead()`, the first `write()` sends the implicit headers with the first body chunk. Later calls send additional body data separately. (Implicit headers are generated automatically from the response properties and stored headers.)
- The `write()` method does not work with request methods or response status codes that do not support content writing.

Example

Figure 46. server.js (lines 5–7)

```
import { createServer } from "node:http";

const server = createServer((req, res) => {
  res.writeHead(200, "OK", { "Content-Type": "text/plain" });
  res.write("First chunk");
  res.write(" | Second chunk");
  res.write(" | Third chunk");
  res.end();
});

server.listen(8000, () => {
  console.log(`Server running at http://localhost:8000/`);
});
```

The snippet above uses `write()` to send three chunks of body data to the client.

response.end

The `response.end()` method completes the response to an HTTP request. It signals to Node that you've finished sending all response headers and the body to the client. In these examples, call this method to finish the response. It does not necessarily close the underlying connection, which may be reused.

Syntax

Figure 47. (line 4)

```
import { createServer } from "node:http";

createServer((request, response) => {
  response.end(chunk, encoding, callback);
});
```

- `chunk`: An optional string, buffer, or Uint8Array representing the portion of the HTTP body you want to send to the client before ending the stream. If specified, it is equivalent to calling `response.write(chunk)` followed by `response.end()`.

- `encoding`: An optional string specifying how Node should encode a string chunk into a byte stream. `"utf8"` is the default value.
- `callback`: An optional function, which Node will call when the data streaming ends.

Example: Use the end() method to finish streaming data to an HTTP client

Figure 48. server.js (lines 8–10)

```
1  import { createServer } from "node:http";
2
3  const server = createServer((request, response) => {
4    response.writeHead(200, "OK", { "Content-Type": "text/plain" });
5    response.write("First chunk\n");
6    response.write(" | Second chunk\n");
7    response.write(" | Third chunk\n");
8    response.end(" | Last data chunk.", "utf8", () => {
9      console.log("Data streaming completed! All data has been sent.");
10   });
11 });
12
13 server.listen(8000, () => {
14   console.log(`Server running at http://localhost:8000/`);
15 });
```

The snippet above uses the `end()` method to properly end data streaming to the client.



- The `\n` escape sequence in the string above represents a newline character.
- The newline character (`\n`) helps create line breaks in strings.
- Set the response status and headers using `writeHead()`, `statusCode`, `statusMessage`, or `setHeader()` before writing the body with `write()` or `end()`.

How to Run JavaScript Code on the Command Line

Node.js comes with a built-in REPL, which stands for read-eval-print loop. You can use it to test small pieces of code.

Start a REPL session in Node.js

To begin a REPL session, just type `node` in your terminal (without any arguments) and press the Enter key.

Figure 49. :

```
node
```

The `node` command starts Node.js in REPL mode, and your terminal will display messages similar to the following:

Figure 50. :

```
$ node
Welcome to Node.js v24.15.0.
Type ".help" for more information.
>
```

The message above means the command line is ready and waiting for you to enter JavaScript or REPL code. Here are some examples.

Example 1: Print a JavaScript value

Figure 51. (lines 4–5)

```
$ node
Welcome to Node.js v24.15.0.
Type ".help" for more information.
> console.log("=== Hello from the CodeSweetly Team! ===");
=== Hello from the CodeSweetly Team! ===
undefined
>
```



To exit the REPL, press `Ctrl + D` on Windows, macOS, or Linux, or type `.exit` in the terminal.

The snippet above told Node.js to print the `=== Hello from the CodeSweetly Team! ===` string. The REPL is smart enough to know when you want to display values, so you usually do not need to use `console.log()`.

Here's an example:

Figure 52. (lines 4–5)

```
$ node
Welcome to Node.js v24.15.0.
Type ".help" for more information.
> "=== Hello from the CodeSweetly Team! ==="
'=== Hello from the CodeSweetly Team! ==='
>
```

Each time you run some code, the REPL does the following:

1. **R**eads the code.
2. **E**valuates it.
3. **P**rints the result and waits for your next input.
4. **L**oops (repeats) these steps after each input until you exit the REPL session.

Example 2: Execute code continuously

Figure 53. (lines 4, 6, 8, 10)

```
$ node
Welcome to Node.js v24.15.0.
Type ".help" for more information.
> 3 + 7
10
> "Code" + "Sweetly"
'CodeSweetly'
> 100 > 7
true
> .exit
```

As you can see, REPL lets you keep running code until you exit the session.

Example 3: Write multiline code

To write multiline code, such as a function, press `Enter` after each line. The REPL will recognize that you wish to continue on a new line.

Figure 54. (lines 4–9)

```
$ node
Welcome to Node.js v24.15.0.
Type ".help" for more information.
> if (new Date().getHours() < 21) {
... "Today is special!"
... } else {
... "Last minutes embody great opportunities!"
... }
'Today is special!'
> .exit
```

The example above demonstrates running a multiline `if` statement in a REPL session. The displayed result depends on the local time when you run it.

In addition to `.exit`, REPL provides several other useful commands. Below are some of the most common.

REPL commands

- `.help`: Print the REPL's dot commands.
- `.break`: Abort the multiline expression currently being entered. Unlike `.break`, `.clear` can also reset the context of a programmatically created REPL.
- `.editor`: Allows you to write multiple expressions in editor mode and run them all at once. Press `Ctrl + D` on Windows, macOS, or Linux to exit editor mode and execute your code.
- `.load`: Load code from a JavaScript file into the REPL session.
- `.save`: Save all the JavaScript code you entered in this REPL session to a file.

REPL also has built-in variables for storing the previous result and error. Let's look at how they work.

REPL variables

REPL has the following variables predefined:

- **Underscore (_):** Stores the previous operation's result.
- **Error (`_error`):** Stores the last error thrown.

Example 1: Access the last operation's result

The REPL assigns the result of your last operation to an internal underscore variable (_). You can use this variable in your next command.

Figure 55. (line 6)

```
$ node
Welcome to Node.js v24.15.0.
Type ".help" for more information.
> 3 + 7
10
> _ + " CodeSweetly " + "books"
'10 CodeSweetly books'
> .exit
```

The example above uses the underscore variable (_) to retrieve the result of the last operation.

Tip

To disable the automatic assignment to the underscore variable, set the variable to a custom value:

Figure 56. (line 8)

```
$ node
Welcome to Node.js v24.15.0.
Type ".help" for more information.
> 3 + 7
10
> _ + " CodeSweetly " + "books"
'10 CodeSweetly books'
> _ = 70
Expression assignment to _ now disabled.
70
> _ + " CodeSweetly " + "books"
'70 CodeSweetly books'
> .exit
```

Example 2: Access the last error thrown

The REPL assigns the last error to an internal error variable (`_error`), or to `undefined` if no error has occurred.

Figure 57. (line 6)

```
$ node
Welcome to Node.js v24.15.0.
Type ".help" for more information.
> "My String".push("Text")
Uncaught TypeError: "My String".push is not a function
> _error
[TypeError: "My String".push is not a function]
> .exit
```

The example above uses the error variable (`_error`) to retrieve the most recent error.

Tip

Assigning a custom value to `_error` disables its automatic updates. The example below leaves automatic updates enabled and shows how a newly thrown error replaces the previous value:

Figure 58. (line 8)

```
$ node
Welcome to Node.js v24.15.0.
Type ".help" for more information.
> "My String".push("Text")
Uncaught TypeError: "My String".push is not a function
> _error
[TypeError: "My String".push is not a function]
> throw new Error("=== Default Error ===")
Uncaught Error: === Default Error ===
> _error
[Error: === Default Error ===]
> .exit
```

Node.js also lets you run the REPL from a JavaScript file for advanced customization. The following section explains how to do this.

Run the REPL from a JavaScript file

To run the REPL from a JavaScript file, import it from the `node:repl` module like this:

Figure 59. :

```
import repl from "node:repl";
```

After importing the `repl` module, you can use its APIs to create, launch, and manage a REPL instance. Below are a few examples.

Example 1: Start a REPL command-line session from a JavaScript file

Figure 60. my-repl-file.js (line 2)

```
import repl from "node:repl";
repl.start();
console.log("REPL session started. Type your commands below:");
```

The example above uses the `repl.start()` method to create and launch a new REPL Server instance. Run the file from your command line as shown below:

Figure 61. :

```
node my-repl-file.js
```

Node.js will enter REPL mode, and your terminal will display messages similar to the following:

Figure 62. :

```
$ node ./my-repl-file.js
> REPL session started. Type your commands below:
```

Example 2: Define a custom input prompt

By default, REPL uses a greater-than sign and a space (`> `) as its prompt. You can change this by providing a prompt argument to the `repl.start()` method.

Syntax

Figure 63. :

```
import repl from "node:repl";

repl.start(promptValue);
```

- `import` statement: Imports the `repl` API from Node's `repl` library.
- `repl.start()`: The method for creating and launching an interactive REPL instance.
- `promptValue`: A string or object for specifying a custom REPL prompt.

Example

Figure 64. my-repl-file.js (line 2)

```
import repl from "node:repl";
repl.start("===+ ");
console.log("REPL session started. Type your commands below:");
```

In the example above, the custom prompt is set to `===+ `. You can also use an object as shown below:

Figure 65. my-repl-file.js (line 2)

```
import repl from "node:repl";
repl.start({ prompt: "===+ " });
console.log("REPL session started. Type your commands below:");
```

When you run the file from your command line, your terminal will display messages similar to the following:

Figure 66. :

```
$ node ./my-repl-file.js
===+ REPL session started. Type your commands below:
```

Example 3: Evaluate the user's command-line inputs

The options object passed to `repl.start({ ... })` has several properties for customizing the REPL environment. For example, you can use the `eval` property to handle user input.

Here's an example:

Figure 67. my-repl-file.js (line 12)

```
import repl from "node:repl";

function stringToUppercase(code, context, replResourceName, callback) {
  const userInput = code.trim();
  if (Number.isFinite(+userInput)) {
    callback(new Error(`(Invalid input) ${userInput} must not convert to a
    ↪  finite number`));
  } else {
    callback(null, `Formatted Input = ${userInput.toUpperCase()}`);
  }
}

repl.start({ prompt: "Enter a text > ", eval: stringToUppercase });
```

The example above uses the `eval` property to define a custom function for processing user input in the REPL session. It rejects input that converts to a finite number and converts other input to uppercase. The input already arrives as a string; this is a numeric-conversion check, not a string-type check.



- Number.isFinite(): Checks if its argument is a finite number.
- `+userInput`: Uses the unary plus operator (+) to attempt to convert `userInput` to a number. It works similarly to the Number() function.

Before we proceed to the next chapter, let's recap what Node.js is.

What Can You Build with Node.js?

Node.js is not just for creating web servers. You can use it for a wide variety of projects, such as:

- Command-line developer tools
- Web applications
- Automation scripts
- Deployment scripts
- Filesystem tools

For example, the snippet below uses Node's `createWriteStream` API to write data to a file.

Figure 68. write-stream.js

```
import { createWriteStream } from "node:fs";

const myWritableStream = createWriteStream("myOutputfile.txt");

myWritableStream.write("First data chunk.\n");
myWritableStream.write(" | Second data chunk.\n");
myWritableStream.write(" | Third data chunk.\n");
myWritableStream.end(" | Last data chunk.", "utf8", () => {
  console.log("Success! The data streaming has ended.");
});
```

Some notes:

- If the specified file does not exist, Node will create it automatically. With the default settings shown here, an existing file is overwritten.
- The `node filename.extension` command still works for running a Node.js script that streams data to a file. The `node` command is a universal CLI tool for running all Node.js scripts regardless of their purposes.

And that's it! This chapter used a basic application setup to explain what Node.js is, its purpose, and how you can use it to run JavaScript programs outside a web browser. In the following chapter, we will discuss how to use Node to manage files.

File Management in Node.js

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Types of Interfaces in the File System Module

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Add the Callback-Based File System Module to Your Node.js App

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Add the Promise-Based File System Module to Your Node.js App

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Important File System Module Methods for Managing Files in Node.js

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

fs.writeFile() – Write data to a file

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

fs.readFile() – Read a file's content

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

fs.appendFile() – Append data to a file

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

fs.rename() – Rename a file or folder

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

fs.unlink() – Delete a file or symbolic link

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

The node:fs Module Gives You APIs to Work with a Computer's File System

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

File Uploads in Node.js

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a New Project Directory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a `package.json` File

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a Form for Uploading Files

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Install the Formidable Package

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Use the Formidable Module in Your Node.js Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Run Your Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

What Is the Formidable `parse()` Method?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

The callback-based `parse()` method

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

The promise-based version of the `parse()` method

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Specify Where Formidable Should Save the Uploaded Files

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

URL Management in Node.js

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

How to Use the Node.js `url` Module

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

URL Constructor – Parse URLs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Syntax of the URL constructor

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Examples of the URL constructor

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

The `URLSearchParams` APIs for reading and writing to a URL's query string

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

searchParams.append() – Add a new query parameter to the URL

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Syntax

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

searchParams.get() – Retrieve the first value matching a specified name

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Syntax

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

searchParams.getAll() – Retrieve all values matching a specified name

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Syntax

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

searchParams.size – Find out the total number of search parameters in a URL

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Syntax

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

searchParams.set() – Set a query parameter's value

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Syntax

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

searchParams.delete() – Remove query parameters from the URL

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Syntax

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

fileURLToPath Function – Resolve File URLs to Paths

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Syntax of the `fileURLToPath()` function

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Examples of the fileURLToPath function

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

pathToFileURL Function – Resolve Paths to URLs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Syntax of the `pathToFileURL()` function

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Examples of the pathToFileURL function

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

The `node:url` Module Provides APIs for Working with URLs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Serve Webpages in Node.js

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a New Project Directory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create the HTML Files for the Webpages You Wish to Serve

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

`index.html` (homepage)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

`about.html` (about page)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

`contact.html` (contact page)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

`404.html` (error page)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure a Web Server to Respond to Browser Requests by Serving Webpages

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Run Your Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Event Management in Node.js

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Add the Events Module to Your Node.js Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create and Trigger Events with the `EventEmitter` API

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Syntax

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Example 1: How to register a listener and trigger an event

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Example 2: How to pass arguments to an event listener

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Example 3: How to add multiple listeners to an event

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Important Emitter Methods for Managing Events in Node.js

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

emitter.once() – running an event listener only once

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

emitter.listeners() – checking all the listeners attached to an event

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

emitter.listenerCount() – counting the listeners attached to an event

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

emitter.eventNames() – getting the event names registered on an emitter object

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

emitter.getMaxListeners() – checking the listener warning threshold

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

emitter.prependListener() – adding a listener to the beginning of an event's listeners array

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

emitter.prependOnceListener() – adding a one-time listener to the beginning of an event's listeners array

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

emitter.removeListener() – removing a listener from an event's listeners array

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Syntax

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Example: Remove a single listener

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Example: Remove multiple instances of the same listener

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Example: Removing a listener does not cancel its call in the current emission

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

emitter.removeAllListeners() – removing all listeners

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Syntax

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Example 1: Remove a specific event's listeners

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Example 2: Remove all listeners from a specific `EventEmitter` instance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

The `node:events` Module Provides APIs for Event Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Use Express.js to Simplify HTTP Request Handling in Node

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a Project Directory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a package.json File

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure the Project as an ES Module Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Why Express.js?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

1. Install Express

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

2. Create the entry script

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

3. Use Express.js to configure the project's Node.js web server

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

4. Run the Express.js web server

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Raw Node.js Code vs. Express.js Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Example 1: Use raw Node.js to handle web requests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Example 2: Use Express.js to handle web requests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Important Stuff to Know About Creating Express.js Web Servers in Node.js

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

It's common to call `app.listen()` last

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

You can set the port number using environment variables

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Common Express.js response methods

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Express's response methods do not terminate the HTTP request handler's execution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Routes, Middleware, and Controllers in Express.js

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Syntax of a Route in Express

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Middleware vs. Route Handler

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Examples

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Express route with a single handler function

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Express route with multiple middleware functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Express route with an array of middleware functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Express route with a combination of an array of middleware functions and individual callbacks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Important Things to Know about the `next` Parameter

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Categories of Middleware in Express.js

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Application-level middleware

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Router-level middleware

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Error-handling middleware

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Built-in middleware

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Third-party middleware

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Note: Middleware categories in Express are not mutually exclusive

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Route vs. Query Parameters in Express.js

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a New Project Directory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a package.json File

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure the Project as an ES Module Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Install Express

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure a Web Server to Respond to Browser Requests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

What Is a Query Parameter in Express?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Syntax of the query parameter

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

How to access a query parameter

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

What Is a Route Parameter in Express?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Syntax of the route parameter

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

How to access a route parameter

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Important things to know about the route parameter

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Dots and dashes between route parameters are literal text

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Routers do not automatically inherit their parent's route parameters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Router in Express.js

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a New Project Directory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a `package.json` File

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure the Project as an ES Module Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Install Express

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Group Related Routes and Middleware

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a directory for the routers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create the route files

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a mini-app for the book-related routes and middleware

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Time to practice with routers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure the Main Express Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Run Your Express.js Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Important Things to Know about Express Routers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Routers do not inherit parent route parameters by default

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Routers can be mounted on other routers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Views in Express.js

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a New Project Directory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a package.json File

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure the Project as an ES Module Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Install Express

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

What Are Static HTML Files in Express.js?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create static HTML files

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

index.html (homepage)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

about.html (about page)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

contact.html (contact page)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

404.html (error page)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure a web server to serve static HTML files

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Run your Express.js application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a directory for static files

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Move all static files into the `public` folder

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Update the server to retrieve static files from the `public` folder

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

What Are Dynamic Views in Express.js?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

What is EJS?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

What are EJS template tags?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Types of EJS template tags

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Closing tag (%>)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Scriptlet tag (<%)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Escaped output tag (<%=)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Unescaped output tag (<%-)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Literal opening tag (<%%)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Newline-trimmed ending tag (-%>)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Whitespace slurp opening tag (<%_)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Whitespace slurp closing tag (_%>)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Comment tag (<%#)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

How to use EJS in an Express project

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Install EJS

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure a web server to serve dynamic views

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Run the Express app

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a directory for view templates

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create an index view template

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Update the server to retrieve view templates from the `views` folder

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

How to create reusable EJS templates (partials)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Syntax of the include() method

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

How to include a partial in a view template

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a partials directory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a partial view template

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Update the main view template

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Run your server from the root directory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Handling Form Data in Express.js

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Validation vs Sanitization vs Output Escaping: What's the Difference?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Types of Form Validation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

What Is Client-Side Form Validation?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

What Is Server-Side Form Validation?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

How to Use Express-Validator to Validate a Form's Data

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a new project directory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a package.json file

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure the project as an ES module application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Install the required packages

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure a web server to respond to browser requests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create the routes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create the controllers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

body()

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

validationResult()

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create the views

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

views/index.ejs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

views/infoForm.ejs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Run your Express.js application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Important Things to Know About Validation, Sanitization, and Output Escaping in Express

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Why use the escaped output tag (<%=)?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

EJS's HTML escaping is effective for normal HTML output

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Avoid escaping user-supplied data permanently

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

POST vs GET Method Attributes in HTML Forms

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Usage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Data visibility

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Data size

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Bookmarkable query

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Shareable query

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Repeated requests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Connect a Frontend Application to a Node.js Backend

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a New Project Directory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create the Frontend and Backend Project Directories

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure the Project's Backend

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a package.json file

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure the project as an ES module application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Install Express

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure a web server to respond to browser requests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure the Project's Frontend

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a package.json file

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a webpage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Handle unsuccessful HTTP responses and network failures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure CORS in the Backend

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Install the `cors` package in your backend

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Allow requests from the frontend

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Run Your Full-Stack Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Start the Node.js backend server

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Run the frontend with a local server

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Send Data from the Frontend to the Express Backend

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Add a form to the webpage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Add a `POST /quote` route to the server

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Run the full-stack application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Use `express.json()` to Parse a JSON Request Body

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Update the webpage's form

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Update the server to parse a JSON request body

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Run your application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Deploy a Full-Stack Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Deploy Your Backend Application to Render

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure the backend to use environment variables for environment-specific settings

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Initialize a Git repository to track the backend changes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Specify the backend files Git should ignore

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Stage and commit your backend changes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Set up a GitHub remote repository for the backend application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Push your backend commits to the remote repository

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Sign in or sign up on the Render website

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a Web Service on Render

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Deploy Your Frontend Application to Vercel

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure the frontend to use environment variables for environment-specific settings

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Update the hardcoded environment-specific settings

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create the frontend JavaScript module

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Specify Parcel as the project's build tool

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Initialize a Git repository to track the frontend changes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Specify the frontend files Git should ignore

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Stage and commit your frontend changes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Set up a GitHub remote repository for the frontend application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Push your frontend commits to the remote repository

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Sign in or sign up on the Vercel website

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Deploy your frontend application to Vercel

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Manage User Sessions in Express with Express-Session

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a Basic Express Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a project directory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a package.json file

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure the project as an ES Module application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Install Express

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure a web server to respond to browser requests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Run the Express.js web server

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Inspect Your App's Cookies in the Browser

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Add Session Support with express-session

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Understand the Three Session-Related Properties

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

sessionStore

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

sessionID

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

session

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Explore the Session Properties in Action

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

1. Install the express-session library

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

2. Configure your app to identify returning clients across multiple requests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

3. Run the Express App

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

4. Send Requests to the Server

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

5. Initialize the Session

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Test Express HTTP Endpoints with Supertest and node:test

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

What's the Difference Between a Test Runner and an HTTP Testing Tool?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a Project Directory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a `package.json` File

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure the Project as an ES Module Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Install Express

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Install the HTTP Testing Library

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Test HTTP Requests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Run the App Script

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Use the `node:test` Module to Run Tests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create the route file

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create the app's test script

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Write the test case

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Update the `app.js` script

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Run the test

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure the App's Web Server

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Run the Express.js Web Server

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Important Things to Know about Testing HTTP Endpoints

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Expectations run in the order of definition

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Supertest requests support promises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

expect() calls accept the done callback

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

expect()'s behavior depends on its arguments

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

The optionalEndFunction callback sends an HTTP request

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Use TypeScript with Express.js

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a Project Directory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Create a package.json File

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure the Project as an ES Module Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Install Express

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Install TypeScript and Type Definitions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Configure a Web Server to Respond to Browser Requests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Run the Express Web Server

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Precompile TypeScript into JavaScript before running it

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

1. Configure the TypeScript compiler

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

2. Compile the project's TypeScript code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

3. Execute the compiled web server

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Transpiling TypeScript at runtime using a TypeScript runner

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

1. Install the project's TypeScript runner

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

2. Execute the TypeScript web server

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Add ESLint to an Express Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Initialize ESLint's Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Lint Your Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Customize ESLint Rules

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Specify the Files ESLint Should Ignore

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Epilogue

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

What Next?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Keep Growing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

One Last Favor

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.

Final Words

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/node-and-express-simplified.