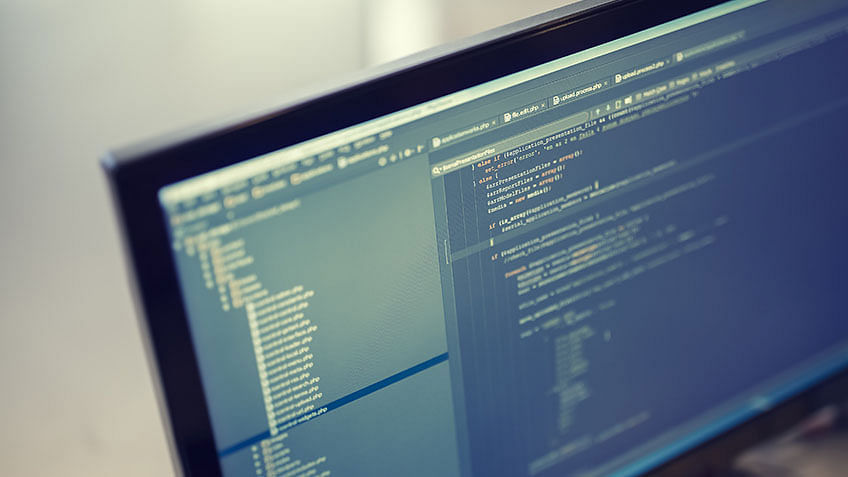
Node.js Express Tutorial: Create a User Management System

Lesson 5 of 8[By Simplilearn](https://www.simplilearn.com/authors/simplilearn)

Last updated on Feb 4, 202244071



[Previous](https://www.simplilearn.com/tutorials/nodejs-tutorial/nodejs-mongodb)[Next](https://www.simplilearn.com/tutorials/nodejs-tutorial/jwt-authentication)

Table of Contents

[What is Express.js?](https://www.simplilearn.com/tutorials/nodejs-tutorial/nodejs-express#what_is_expressjs)

[Features of Express.js](https://www.simplilearn.com/tutorials/nodejs-tutorial/nodejs-express#features_of_expressjs)

[A Basic “Hello World” Example in Node.js Express](https://www.simplilearn.com/tutorials/nodejs-tutorial/nodejs-express#a_basic_hello_world_example_in_nodejs_express)

[Node.js Express Application](https://www.simplilearn.com/tutorials/nodejs-tutorial/nodejs-express#nodejs_express_application)

[Let’s Code Now!](https://www.simplilearn.com/tutorials/nodejs-tutorial/nodejs-express#lets_code_now)

View More

[Node.js](https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs) is open-source and completely free, with countless developers worldwide using it to develop I/O intensive web applications, such as video streaming sites, single-page applications, online chat applications, and other web applications. The platform brings plenty of advantages to the table, making it a better choice than other server-side platforms, such as [Java](https://www.simplilearn.com/tutorials/java-tutorial) or PHP.

This Node.js Express tutorial is meant to help users with creating servers on Node.js Express, as well as defining API calls that can be connected to the frontend framework, such as React or Angular.  First, let’s learn more about what Express.js is and then implement it in a Node.js application to create a server.

What is Express.js?

|  |  |
| --- | --- |
| express | * Express is a flexible Node.js web application framework that provides a wide set of features to develop both web and mobile applications |
| expressjs | * Express.js makes it much easier and simpler to build a web server with the use of middleware, which can handle requests and responses |

Features of Express.js

The main features of Express.js include:

|  |  |
| --- | --- |
| /capability. | * The capability to design single-page, multi-page, and hybrid web applications |
|  | * Options for setting up middleware to respond to HTTP requests |
| define. | * It defines a routing table that is used to perform different actions based on the HTTP method and URL |
| html | * Enables users to dynamically render HTML pages based on passing arguments to templates |

A Basic “Hello World” Example in Node.js Express

Let’s now look at a basic web server that prints “Hello World” on the web browser.

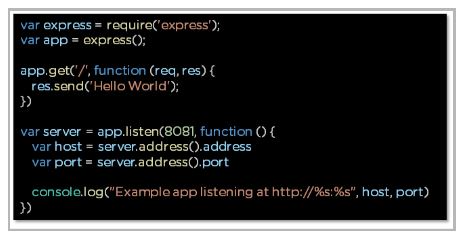


Fig: Express.js Hello World example

* The request object represents the HTTP request and contains properties for the request query string, parameters, body, HTTP headers, and so on
* The response object represents the HTTP response that an Express app sends when it receives an HTTP request



Fig: Express.js Hello World output

The following are the steps that we followed in this code to create a basic Express web server:

* Importing Express framework into our Node.js application
* Implementing a callback function with parameters ‘request’ and ‘response’
* The application will listen on the defined port, which in this case is “8081,” and variables ‘host’ and ‘port’ will contain the address and the port respectively
* Lastly, the console.log statement shows the address and port in the command prompt or terminal

Node.js Express Application

In this application, we will use Express.js to create a web server and then create API calls. These API calls will handle CRUD operations. CRUD stands for create, read, update, and Delete.

Prerequisites

Node.js installation

1. Download Node.js from <https://nodejs.org/en/download/>. Select the appropriate installer for your operation system requirements.

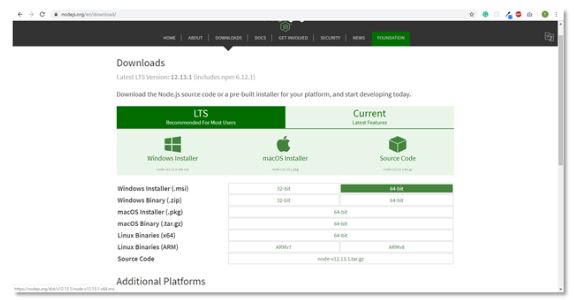


Fig: Node.js Official download page

2. Run the Node.js installer and accept the license agreement. You can leave the remaining settings as default. The installer will install Node.js and prompt you to click on the finish button.



Fig: Node.js Setup dialog box

3. To verify that Node.js was properly installed, open the command prompt and type this command: node --version

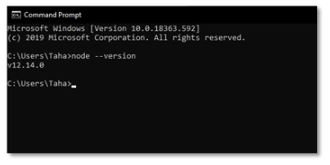


Fig: Command Prompt for verification of Node.js

4. When we install Node.js, NPM (Node Package Manager) is also installed. NPM includes several libraries that are used in web applications, such as React. Verify whether it is installed or not with the following command in CMD: npm --version

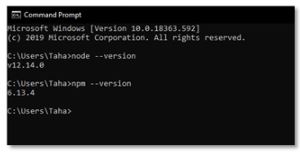


Fig: Command prompt for NPM verification

Text Editor

Install a text editor of your choice. We are using [Visual Studio Code](https://code.visualstudio.com/) in this tutorial, but you can also use other editors, such as Atom and Sublime Text, if you prefer those.

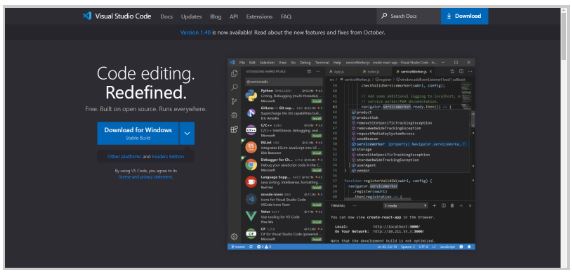


Fig: Visual Studio Code Official download page

Postman

We are using the Postman application to verify the output of the application. We will send and receive API calls and check if the web server is working properly.

To download the application, go to its [official website](https://www.postman.com/downloads/).

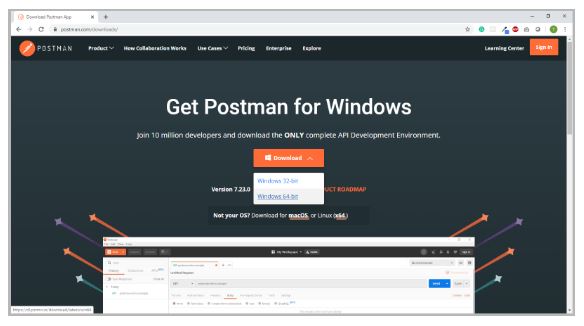


Fig: Postman Official download page

Project Setup

1. Create an empty folder and name it node express.
2. Open the newly created directory in VS Code, and inside the terminal, type npm init to initialize the project. Press the “Enter” key to leave the default settings as they are.

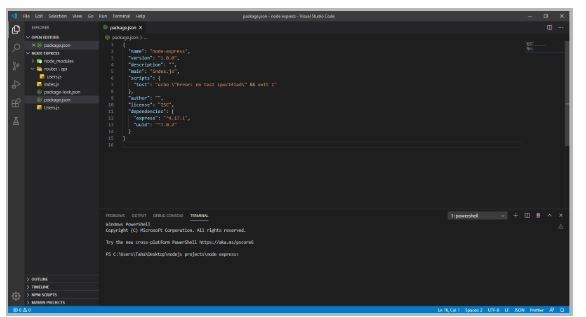


Fig: Node.js Express initial setup

Let’s Code Now!

In this Node.js Express tutorial, we are going to create our own Node.js application with Express.js, which makes creating a web server much easier than vanilla Node.js.

Let’s first take a look at what the project directory should look like by the end of this tutorial.

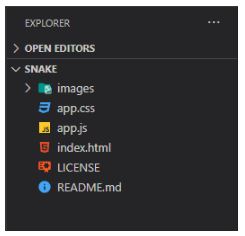


Fig: Project directory

* The file index.js acts as the main server for the Node.js application
* Users.js is a model that contains data on all users

ps

* Inside the folder routes/api, we define all the API routes that we need for our Node.js Express application

index.js

This is the file that acts as the main server and binds every API route together. We create this file in the root directory of the project and then add the following code:

const express = require("express");

const app = express();

app.use(express.json());

app.use(express.urlencoded({ extended: false }));

app.use("/api/users", require("./routes/api/users"));

app.listen(3000, () => console.log('Server started'));

* We first install the express module to use in our Node.js application
* Next, import the express module into the index.js file
* express.json() is a middleware function that parses JSON payload, if any, in the incoming API requests
* We eventually import all the routes that we will create in another directory
* Finally, we set the application to listen at port 3000, and return a message on the console stating that the server has started

Users.js

This file stores all user data in the user management system. Since this is a small project, we are not using a database here, and only using a list to store the users’ information.

const users = [

  {

    id: 1,

    name: "John",

    email: "john@gmail.com"

  },

  {

    id: 2,

    name: "Smith",

    email: "smith@gmail.com"

  },

  {

    id: 3,

    name: "Chris",

    email: "chris@gmail.com"

  },

  {

    id: 4,

    name: "Jack",

    email: "jack@gmail.com"

  }

];

module.exports = users;

* We create a list named users and then enter the data for four users inside that list—in the form of id, name, and email address
* Then, we export the module so that other files in the project can use it

FREE Java Certification Training

Learn A-Z of Java like never before[ENROLL NOW](https://www.simplilearn.com/learn-java-basics-skillup?utm_source=frs&utm_medium=skillup-course-banner&utm_campaign=frs-tutorial-skillup-course-promotion)



users.js

In this file, we add the API routes that we want to include in the application. First, we’ll create a folder named routes, and inside that, create another folder named api. After creating the folders, add a new file and name it users.js

const express = require("express");

const router = express.Router();

const uuid = require("uuid");

let users = require("../../Users");

router.get("/", (req, res) => {

  res.json(users);

});

router.get("/:id", (req, res) => {

  const found = users.some(user => user.id === parseInt(req.params.id));

  if (found) {

    res.json(users.filter(user => user.id === parseInt(req.params.id)));

  } else {

    res.sendStatus(400);

  }

});

router.post("/", (req, res) => {

  const newUser = {

    id: uuid.v4(),

    name: req.body.name,

    email: req.body.email

  };

  if (!newUser.name || !newUser.email) {

    return res.sendStatus(400);

  }

  users.push(newUser);

  res.json(users);

});

//Update User

router.put("/:id", (req, res) => {

  const found = users.some(user => user.id === parseInt(req.params.id));

  if (found) {

    const updateUser = req.body;

    users.forEach(user => {

      if (user.id === parseInt(req.params.id)) {

        user.name = updateUser.name ? updateUser.name : user.name;

        user.email = updateUser.email ? updateUser.email : user.email;

        res.json({ msg: "User updated", user });

      }

    });

  } else {

    res.sendStatus(400);

  }

});

//Delete User

router.delete("/:id", (req, res) => {

  const found = users.some(user => user.id === parseInt(req.params.id));

  if (found) {

    users = users.filter(user => user.id !== parseInt(req.params.id))

    res.json({

      msg: "User deleted",

      users

    });

  } else {

    res.sendStatus(400);

  }

});

module.exports = router;

* Import the necessary modules in this file:
  + Import express to create API routes
  + Import uuid to generate ids for new users
  + Import users to use user data
* router.get(‘/’): This route displays the users’ data in the API response
* router.get(‘/:id’): We define this route to search for a user’s data using their ID. It returns the data if found; otherwise, it sends an error message
* router.post(‘/’): This is a POST API request that enables us to add a user into the student management system
* router.put(‘/:id’): We define another API route that takes in the ID of a particular user, and then updates the data of that user if found; otherwise, it sends an error message
* router.delete(‘/:id’): Finally, create the last route for our Node.js application that takes in a user’s ID as input and deletes that user if found; otherwise, it sends an error message

That’s all the code we needed to create a web server using Express.js in a Node.js application.

Next, we’ll learn how to verify that the web server is working properly. We can start the server by using the following command in the terminal: node index.js

Postman

Postman is an intuitive and powerful application that web developers use to check if the API requests in their applications are working properly.

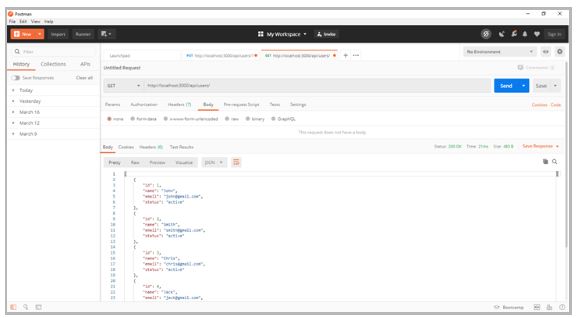


Fig: Postman Application

* Conclusively, we use Postman for API calls:
  + GET to retrieve data: http://localhost:3000/api/users/
  + POST with JSON body to add a user: http://localhost:3000/api/users/
  + PUT with JSON body to update data for user-id: http://localhost:3000/api/users/1
  + DELETE to delete a user: http://localhost:3000/api/users/1

All API routes should work as expected.

If you have any questions for us, please leave them in the comment section below and we'll have our experts get back to you promptly.

Learn top skills including Angular, Spring Boot, Hibernate, Servlets, and JSPs, as well as MVC, web services, and SOA to build highly web scalable apps with the [Full Stack Java Developer Masters Program](https://www.simplilearn.com/java-full-stack-developer-certification-training-course?source=GhPreviewCTAText).

Get Ahead of the Curve and Master Node.js Today

Now that you know how to create a simple web server in this Node.js Express tutorial, you may be wondering how you can obtain the skills necessary to take advantage of its rising popularity. Fortunately, there are some great options to learn this exciting and practical skill set at your own pace. Simplilearn’s [Node.js Certification training course](https://www.simplilearn.com/node-js-certification-training?referrer=search&tag=node.js) will give you a great foundation in this popular platform, combining live, instructor-led training, self-paced tutorials, and hands-on projects to help you become career-ready upon completion. You can also explore the [Node.js interview Question](https://www.simplilearn.com/tutorials/nodejs-tutorial/nodejs-interview-questions) to prepare yourself to crack your next interview. Get started today and seize your future!