Node.JS Tutorial for beginners

Published On: October 1, 2024

Node.JS Tutorial for beginners

With Node.js, developers can create frontend and server-side JavaScript programs with ease. Learn how to accomplish this through this Node.JS tutorial developed for beginners.

Downl9oad Node.JS Tutorial PDF

Introduction to Node.JS

Since Node.js employs an asynchronous, event-driven approach, it is ideal for data-intensive applications. Developers utilize it to construct server-side web apps. In this Node.JS tutorial, we cover the following:

- Overview of Node.JS
- Node.JS Environment Setup
- Node.JS Modules
- Node.JS HTTP Module
- REPL Terminal
- Node.JS Package Manager (NPM)
- Advantages of Node.JS

Overview of Node.JS

Networking and server-side applications can be created with Node.js, a cross-platform environment,

and a library for JavaScript applications.

To make developing web applications easier, Node.js also offers a comprehensive library of different JavaScript modules. JavaScript Library + Runtime Environment is Node.js.

Node.js is lightweight and efficient, ideal for data-intensive real-time applications that span distributed devices, due to its event-driven, non-blocking I/O approach.

Node.JS Interview Questions and Answers

Features of Node.JS

The key features of Node.js that make it the top option for software architects are enumerated below:

- **Exceptionally quick:** Node.js's library executes code incredibly quickly because it is based on the V8 JavaScript Engine found in Google Chrome.
- **I/O is Event-Driven and Asynchronous:** The Node.js library's APIs are all non-blocking and asynchronous. Thus, a server running on Node.js never has to wait for an API to return data.
- **Single-threaded:** Node.js uses event looping to implement a single-threaded architecture.
- **Extremely Scalable:** Node.js is so scalable because its event structure makes it possible for the server to react without stalling.
- **No buffering:** When uploading audio and video files, Node.js reduces processing time overall. Applications built with Node.js never buffer data. These programs only provide fragmented data output.
- **Open source:** The Node.js community is open

MERN Stack Tutorial for Web Development Aspirants
Published On: October 14, 2024
MERN Stack Tutorial for Web Development Aspirants There is a growing need for competent MERN...

Tableau Developer Salary in Chennai
Published On: October 12, 2024
Introduction A Tableau Developer designs, develops, and maintains dashboards and visualizations using Tableau software. Key...

VMware Tutorial for Cloud Computing Aspirants
Published On: October 12, 2024
VMware Tutorial for Cloud Computing Aspirants VMware software allows you to run a virtual machine...

VBA Macros Tutorial

source and has created many fantastic modules to enhance the functionality of Node.js apps.
- **License:** The MIT license governs the release of Node.js.

If you are new to the web development field, our **HTML course in Chennai** would be a good start.

Node.JS Course Syllabus PDF

Node.JS Environment Setup

You can use our online environment to run the sample code, or you can install the Node.js environment locally on your computer. Using the Live Demo button found in the upper right corner of the code box below, try the following example:

console.log("Hello World!");

The Live Demo button opens a new browser tab with the online Node.js environment.

Local Environment Setup

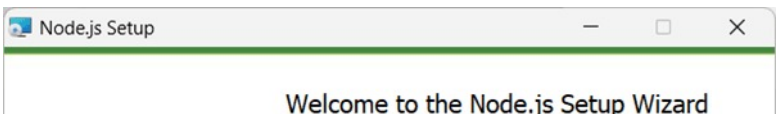
You may install Node.js on various operating systems, including Windows, Linux, Mac OS X, etc. The following computer tools are required:

- The Node.js binary installer
- Node Package Manager (NPM)
- IDE or Text Editor

Installation on Windows

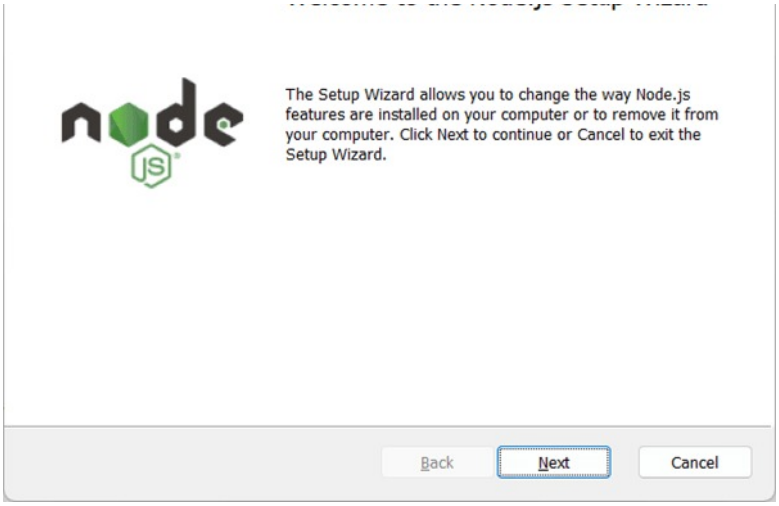
If your machine is running Windows 10 or Windows 11, download the 64-bit installer here: https://nodejs.org/dist/v20.9.0/node-v20.9.0-x64.msi.

- Double-click the downloaded file to begin the installation process.



for Beginners
Published On: October 10, 2024

VBA Macros Tutorial for Beginners VBA macros are programs that automate repetitive operations in Microsoft...

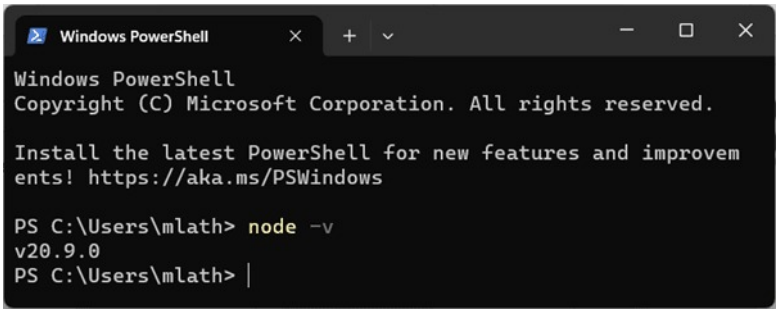


Node.JS Tutorial

You go through a few installation wizard steps during the installation process.

Additionally, it updates the system path with the Node.js executable's installation directory.

To confirm that Node.js has been installed successfully, execute node -v at the command prompt. The version of Node.js installed on your computer will be displayed, as seen below if the Node.js installation is successful.



Node.JS Tutorial 1

Installation on Ubuntu Linux

Download the tar file matching to extract the Linux binary from this link: *https://nodejs.org/dist/v20.9.0/node-v20.9.0-linux-x64.tar.xz.* Next, use the tar command to extract the binary:

tar -xf node-v20.9.0-linux-x64.tar.gz

Move the extracted files to the installation directory

/usr/local/node-v20.9.0.

sudo mv node-<version> /usr/local/node-v20.9.0

Make a symlink to access the executable located in the /usr/bin directory.

sudo ln -s /usr/local/node-v20.9.0/bin/node /usr/bin/node

With the following command, you can now confirm that the installation was done correctly:

node -v

Using the Ubuntu package manager

Try refreshing your local package index first with the following command:

sudo apt update

Now, install Node.js:

sudo apt install nodejs

As in the instance above, confirm the installation

node -v

Since Node version 0.6.0, Node Package Manager (NPM) has been part of the Node.js binaries from its official website; hence, you don't need to install it individually. Train your brain to become the best web developer with our **MEAN stack course in Chennai**.

Node.JS Modules

You can utilize the built-in modules in Node.js without having to install any other software.

Including the Existing Modules

Use the require() method and the module name to include a module:

var http = require('http');

Now that the HTTP module is available to your program, it can start a server:

```
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.end('Hello World!');
}).listen(8080);
```

Creating Your Own Modules

Your apps can incorporate your own modules with ease if you build them. The module that generates a date and time object is created in the example that follows:

Make a module that provides the time and date as of right now:

```
exports.myDateTime = function () {
  return Date();
};
```

To make properties and methods accessible from outside the module file, use the exports keyword. Create a file named "myfirstmodule.js" and save the above code there.

Building an application in Node.js

To create a "Hello, World!" web application with Node.js, you'll need the following three essential components:

- **Import necessary modules:** Node.js modules are loaded using the need directive.
- **Build a server:** A server that can accept requests from clients and operates similarly to the Apache HTTP Server.
- **Read request and return response:** The server generated in the previous phase will read the client's HTTP request, which may be from a console or a browser, and return the response.

Step 1: Import the necessary module

The need directive is used to load the http module, as seen below, and the delivered HTTP instance is saved in an http variable.

var http = require("http");

Step 2: Build a server

Create Server Using the previously generated http instance, we execute the http.createServer() function to create a server instance. Next, we bind the server instance to port 3000 by using the listen method that is connected to it. Assign the request and response arguments to a function.

Syntax for createServer()

http.createServer(requestListener);

Every time the server receives a request from a client, a function called requestlistener is called. This function processes the incoming request and then generates a server response.

A ServerResponse object is returned by the requestlistener function, which accepts request HTTP requests and response objects from the Node.js runtime.

```
listener = function (request, response) {

  // Send the HTTP header

  // HTTP Status: 200 : OK

  // Content Type: text/plain

  response.writeHead(200, {'Content-Type':
'text/html'});
```

The above function includes the content-type headers and status code to the ServerResponse along with the Hello World message.

This function feeds into the createserver() function as an argument. The server is configured to listen on a specific port; let's use 3000 as the port for incoming requests.

Step 3: Read the request and return the response

Create an example implementation that will consistently return "Hello World". Save the script that follows as hello.js.

```
http = require('node:http');

listener = function (request, response) {

  // Send the HTTP header

  // HTTP Status: 200 : OK

  // Content Type: text/html

  response.writeHead(200, {'Content-Type':
'text/html'});

  // Send the response body as "Hello World"

  response.end('<h2 style="text-align: center;">Hello
World</h2>');

};

server = http.createServer(listener);

server.listen(3000);

// The message will print on the console.

console.log('Server running at
http://127.0.0.1:3000/');
```

Type the following command into the PowerShell terminal.

```
PS D:\nodejs> node hello.js

Operating server: http://127.0.0.1:3000/
```

The application launches the Node.js server in listen

mode on port 3000 on localhost. Now launch a browser and type the URL into it: http://127.0.0.1:3000/. The browser displays the Hello World message exactly as it should.

Node.JS HTTP Module

The HTTP module with Node.js enables data communication over the Hypertext Transfer Protocol (HTTP). Use the require() method to incorporate the HTTP module.

var http = require('http');

Node.js as a Web Server

An HTTP server that responds to client requests and listens on server ports can be created by the HTTP module. To create an HTTP server, use the createServer() method:

Example

```
var http = require('http');

http.createServer(function (req, res) {

  res.write('Hello World!'); //write a response to the client

  res.end(); //end the response

}).listen(8080); //the server object listens on port 8080
```

Add an HTTP Header

If HTML is the intended format for the HTTP server response, you must add an HTTP header containing the appropriate content type:

Example

```
var http = require('http');

http.createServer(function (req, res) {

  res.writeHead(200, {'Content-Type': 'text/html'});

  res.write('Hello World!');

  res.end();

}).listen(8080);
```

Get a fundamental understanding with our **Web Designing Course in Chennai**.

REPL Terminal

One interactive shell is integrated into the Node.js runtime, allowing you to run commands one at a time. The Node.js interactive shell runs on the acronym REPL, which stands for READ, EVALUATE, PRINT, and LOOP.

The Node.js interactive REPL terminal is comparable to a Powershell, Command Prompt, or Linux bash terminal. It carries out the subsequent duties:

- **Read:** This function takes in user input, parses it into a JavaScript data structure, and then stores it in memory.
- **Eval:** Takes the data structure and assesses it.
- **Print:** To print the result.
- **Loop:** The terminal is prepared to take in the user's next input.

Simply type node (without the javascript file name as done previously) in the command terminal to launch the Node.js REPL on your PC. You'll see the Node.js prompt >.

```
D:\nodejs>node

Welcome to Node.js v20.9.0.

Type ".help" for more information.
```

>

The REPL feature of Node is quite useful when playing with Node.js scripts and troubleshooting JavaScript code. You can test any JavaScript or Node.js expression by placing it in front of the > prompt.

Example

> 10+20

30

> "Hello"+"World"

'HelloWorld'

> a=10

10

> b=20

20

> a+b

30

> Math.random()

0.5423940959293392

>

As you notice, the terminal is ready to receive the next command after reading, evaluating, and displaying the command.

To start the REPL, press ctrl+c twice, or ctrl+D, or enter .exit in front of > symbol.

Node.JS Developer Salary in Chennai

Multiline Expression

Node REPL supports multiline expressions, much like JavaScript does. Let's look at the following while loop in action:

```
> x=0

0

> do {

... x++;

... console.log("x: "+x);

... }

... while (x<5);

x: 1

x: 2

x: 3

x: 4

x: 5

undefined

>
```

After the opening bracket, you instantly see the three dots... when you press Enter. Node verifies expression continuity automatically.

Underscore Variable

To obtain the final result, use underscore (_):

```
> var x=10

undefined

> var y=20

undefined
```

> x+y

30

> var z= _

undefined

> z

30

>

Dot Commands

Certain special commands in the REPL begin with a dot. They're

- **.help:** It displays the help for dot commands
- **.editor:** This activates editor mode, making it simple to write multiline JavaScript code. To execute the code you wrote in this mode, press Ctrl-D.
- **.break:** This command will stop further input when you enter a multi-line expression. equivalent to hitting Ctrl-C.
- **.clear:** It removes any multi-line expression that is presently being input and restores the REPL context to an empty object.
- **.load:** A JavaScript file is loaded about the active working directory.
- **.save:** This saves everything you typed during the REPL session to a file (name the file).
- **.exit:** This closes the window (same as pressing C twice).
- **Up/Down Keys:** It helps to view command history and change earlier orders.
- **tab Keys:** A list of the available commands.

Accelerate your career with our **AngularJS training in Chennai**.

NPM Package Manager

The command-line tool called NPM, or Node Package Manager, is used to install Node.js

packages and control their versions and dependencies.

- Additionally, it offers a searchable online repository for node.js packages and modules at *https://www.npmjs.com/*.

You can click this page to view a detailed guide on NPM commands.

The NPM tool comes with the Node.js binaries for all OS platform bundles if you are using a later version of Node.js (0.6.0 or higher). Verify the NPM version using the command terminal:

PS C:\Users\mlath> npm -v

10.1.0

If you possess an earlier version of NPM, you must use the following command to update it to the most recent version.

npm install npm -g

Keep in mind that YARN and PNPM tools are available as substitutes for NPM.

- YARN facilitates faster package installations and uniform dependency management across computers or safe offline environments.
- For Node.js packages, another quick and disk-space-efficient package manager is called Performant NPM (PNPM).

Installing external packages from the NPM repository is necessary if your Node.js application depends on any of them. Local or global installation are the two ways that NPM packages are installed. Installing a package locally is the default. Gain expertise with web development through our **JavaScript training in Chennai**.

Installing Package Locally

Any Node.js module may be installed using a

straightforward syntax:

npm install <Module Name>

Example:

npm install express

You can now use this module as follows in your JS file:

var express = require('express');

Installing a package in local mode involves doing so in the node_modules directory, which is found in the folder that houses the Node application.

The require() method can be used to access packages that are installed locally.

- To add a dependent item to the package.json file for your application, use the option –save after the install command.
- A JSON file called package.json is used in Node.js projects to handle dependencies. It includes project details including name, version, and dependencies.
- The package.json file is used by the npm package manager to install and manage dependencies.

The package.json file for a Node.js project is typically located in the root directory. You can execute the npm init command to create it.

To start a new Node.js project, create a new folder and execute the pnm init command inside it:

PS D:\nodejs\newnodeapp> npm init

This tool will guide you through the package creation process.JSON data set. It attempts to predict reasonable defaults and only covers the most frequently used elements.

Use `npm install <pkg>` afterward to install a

package and save it as a dependency in the package.json file.

Press ^C at any time to quit.

package name: (newnodeapp) newnodeapp

version: (1.0.0)

description: Test Node.js App

entry point: (index.js)

test command:

git repository:

keywords: test, nodejs

> *author: mvl*
>
> *license: (ISC)*
>
> Writing to *D:\nodejs\NewNodeApp\package.json* is about to occur.
>
> *{*
>
> *“name”: “newnodeapp”,*
>
> *“version”: “1.0.0”,*
>
> *“description”: “Test Node.js App”,*
>
> *“main”: “index.js”,*
>
> *“scripts”: {*
>
> *“test”: “echo \”Error: no test specified\” && exit 1”*

```
  },
  "keywords": [
    "test",
    "nodejs"
  ],
  "author": "mvl",
  "license": "ISC"
}
```

In this "project.JSON," if we install the express package locally, the next step adds a dependent item to the package.

D:\nodejs\newnodeapp>npm install express –save

This folder's package.json file will be modified to:

```
  {
    "name": "newnodeapp",
    "version": "1.0.0",
    "description": "Test Node.js App",
    "main": "index.js",
    "scripts": {
```

> *"test": "echo \"Error: no test specified\" && exit 1"*
>
> *},*
>
> *"keywords": [*
>
> *"test",*
>
> *"nodejs"*
>
> *],*
>
> *"author": "mvl",*
>
> *"license": "ISC",*
>
> *"dependencies": {*
>
> *"express": "^4.18.2"*
>
> *}*
>
> *}*

Within the node_modules subfolder of the package folder, the express package code is located.

Installing all of the project dependencies at once can be achieved by running npm install if you have already added them to your "*package.json*" file.

The npm install command can be used with the -save-dev parameter to add the package as a DevDepndency.

- **–save-dev:** It installs and adds the entry in the *package.json* file *devDependencies*.
- **–no-save:** Installs the file but does not add the dependency to the package.json file.
- **–save-optional:** It installs and adds the item to the package.json file *optionalDependencies*

- **–no-optional:** Installing optional dependencies will not be possible with this.

Installing Package Globally

The system directory is where globally installed packages and dependencies are stored.

Any node can use these dependencies.js CLI (Command Line Interface) method, however, require() cannot be used to import them straight into a Node application. Let's try installing the express module globally now.

npm install express -g

Similar results will be obtained, but this time the module will be installed globally.

The global packages are located in the

- /usr/lib/node_modules folder on Linux
- C:\Users\your for Windows

username\AppData\Roaming\npm\node_modules

Update Package

Open the command prompt or terminal in your project folder, then type the following update command to update the package installed locally in your Node.js project.

npm update <package name>

To update the current ExpressJS module to the newest version, use the following command.

PS D:\nodejs\newnodeapp> npm update express

audited 63 packages in 2s as of right now.

11 packages need financing;

See `npm fund' for more details.

found 0 vulnerabilities

Uninstall Packages

Use this command to delete a local package from your project to uninstall it from its dependencies.

npm uninstall <package name>

The following command can be used to delete the ExpressJS for the application.

PS D:\nodejs\newnodeapp> npm uninstall express

eliminated 62 packages and examined one package, discovering no vulnerabilities.

The package entry will also be removed from the dependencies list in the package.json file.

Hone your skills with our **MERN full-stack job seeker course** to find your dream career faster.

Advantages of Node.JS

Numerous benefits of Node.js include:

- **Fast Performance:** Node.js's event-driven, single-threaded architecture allows it to handle several requests at once without stuttering.
- **Scalability:** Node.js is a great option for expanding enterprises because it's simple to scale both vertically and horizontally.
- **Seamless JSON support:** JavaScript is used by Node.js to interface with JSON, the primary data transmission format.
- **Access to a large library of tools:** A large library of tools is available through the Node Package Manager (NPM) to assist developers in creating applications more rapidly and effectively.
- **Based on JavaScript:** One of the most widely used programming languages among developers, JavaScript serves as the

foundation for Node.js.

- **Simple to understand:** It's simple to pick up and adjust to Node.js.

Conclusion

Almost any type of project can benefit from the use of Node.js. One can obtain careers on many job profiles upon acquiring knowledge of Node.js. We hope this Node.JS tutorial has provided a fundamental understanding of web development with JavaScript. Join SLA for the best **Node.JS training in Chennai**.

Share on your Social Media

Softlogic Academy

Softlogic Systems

KK Nagar [Corporate Office]

No.10, PT Rajan Salai, K.K. Nagar, Chennai – 600 078.
Landmark: Karnataka Bank Building
Phone: +91 86818 84318
Email: enquiry@softlogicsys.in
Map: Google Maps Link

OMR

No. E1-A10, RTS Food Street
92, Rajiv Gandhi Salai (OMR),
Navalur, Chennai - 600 130.
Landmark: Adj. to AGS Cinemas

Phone: +91 89256 88858
Email: info@softlogicsys.in
Map: Google Maps Link

Courses

- Python
- Software Testing
- Full Stack Developer
- Java
- Power BI
- Clinical SAS
- Data Science
- Embedded
- Cloud Computing
- Hardware and Networking
- VBA Macros
- Mobile App Development
- DevOps

Social Media Links

Review Sources

- Google
- Trustpilot
- Glassdoor
- Mouthshut
- Sulekha
- Justdial
- Ambitionbox
- Indeed
- Software Suggest
- Sitejabber