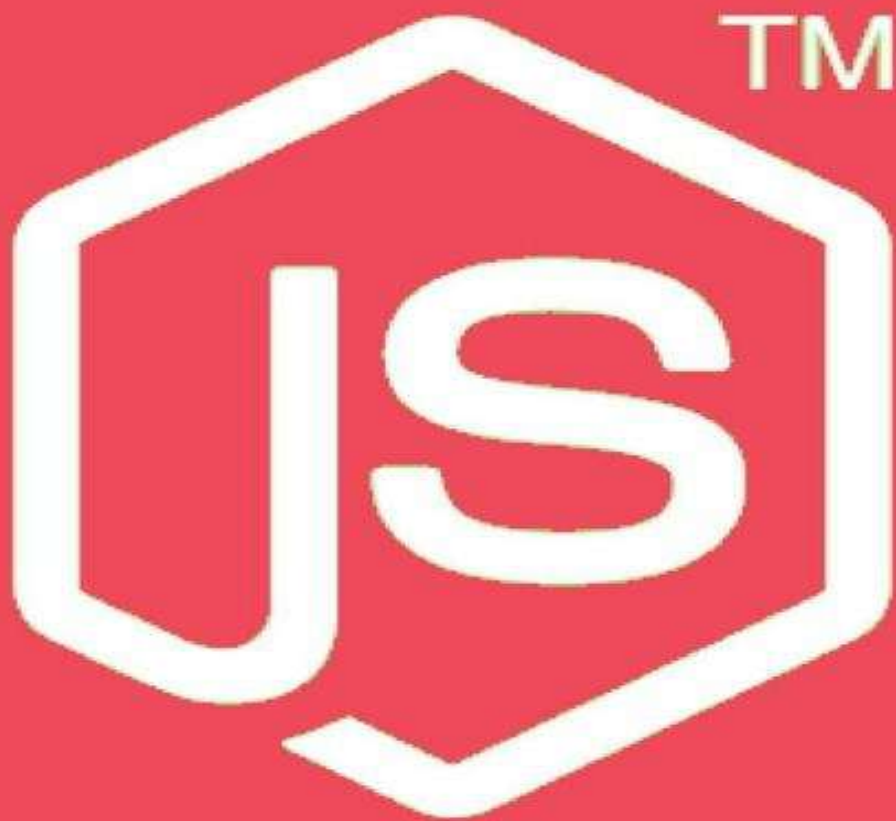


---



# Node Js

---



LEARN IN 1 DAY

KRISHNA RUNGTA

# Learn NodeJS in 1 Day

By Krishna Rungta

Copyright 2016 - All Rights Reserved – Krishna Rungta

**ALL RIGHTS RESERVED.** No part of this publication may be reproduced or transmitted in any form whatsoever, electronic, or mechanical, including photocopying, recording, or by any informational storage or retrieval system without express written, dated and signed permission from the author.

# Table Of Content

## **Chapter 1: Introduction**

1. [What is node.js](#)
2. [Why use Node.js](#)
3. [Features of Node.js](#)
4. [When to use and not use Node.js](#)

## **Chapter 2: Download & Install Node.js**

1. [How to install node.js](#)
2. [Installing node through a package manager](#)
3. [Running your first Hello world application in Node.js](#)

## **Chapter 3: Modules**

1. [What are modules in Node.js](#)
2. [Using modules in Node.js](#)
3. [Creating NPM modules](#)
4. [Extending modules](#)
5. [Publishing NPM Modules](#)
6. [Managing third party packages with npm](#)
7. [What is the package.json file](#)

## **Chapter 4: Create Server and Get Data**

## **Chapter 5: Node.js with Express**

1. [What is Express.js](#)
2. [Installing and using Express](#)
3. [What are Routes](#)
4. [Sample Web server using express.js](#)

## **Chapter 6: Node.js with MongoDB**

1. [Node.js and NoSQL Databases](#)
2. [Using MongoDB and Node.js](#)
3. [How to build a node express app with MongoDB to store and serve content](#)

## **Chapter 7: Promise, Generator, Event and Filestream**

1. [What are promises](#)

2. [Callbacks to promises](#)
3. [Generating promises with the BlueBird library](#)
4. [Creating a custom promise](#)
5. [Callbacks vs generators](#)
6. [Filestream in Node.js](#)
7. [Emitting Events](#)

## **Chapter 8: Testing with Jasmine**

1. [Overview of Jasmine for testing Node.js applications](#)
2. [How to use Jasmine to test Node.js applications](#)

# Chapter 1: Introduction

The modern web application has really come a long way over the years with the introduction of many popular frameworks such as bootstrap, Angular JS, etc. All of these frameworks are based on the popular JavaScript framework.

But when it came to developing server based applications there was just kind of a void, and this is where Node.js came into the picture.

Node.js is also based on the JavaScript framework, but it is used for developing server-based applications. While going through the entire tutorial, we will look into Node.js in detail and how we can use it to develop server based applications.

# What is node.js

Node.js is an open-source, cross-platform runtime environment used for development of server-side web applications. Node.js applications are written in JavaScript and can be run on a wide variety of operating systems.

Node.js is based on an event-driven architecture and a non-blocking Input/Output API that is designed to optimize an application's throughput and scalability for real-time web applications.

Over a long period of time, the framework available for web development were all based on a stateless model. A stateless model is where the data generated in one session (such as information about user settings and events that occurred) is not maintained for usage in the next session with that user.

A lot of work had to be done to maintain the session information between requests for a user. But with Node.js there is finally a way for web applications to have a real-time, two-way connections, where both the client and server can initiate communication, allowing them to exchange data freely.

# Why use Node.js

We will have a look into the real worth of Node.js in the coming chapters, but what is it that makes this framework so famous. Over the years, most of the applications were based on a stateless request-response framework. In these sort of applications, it is up to the developer to ensure the right code was put in place to ensure the state of web session was maintained while the user was working with the system.

But with Node.js web applications, you can now work in real-time and have a 2-way communication. The state is maintained, and the either the client or server can start the communication.

# Features of Node.js

Let's look at some of the key features of Node.js

1. Asynchronous event driven IO helps concurrent request handling – This is probably the biggest selling points of Node.js. This feature basically means that if a request is received by Node for some Input/Output operation, it will execute the operation in the background and continue with processing other requests.

This is quite different from other programming languages. A simple example of this is given in the code below

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

    fs.readFile("Sample.txt", function(error, data)

    {

                console.log("Reading Data completed");

    });
```

- The above code snippet looks at reading a file called Sample.txt. In other programming languages, the next line of processing would only happen once the entire file is read.
  - But in the case of Node.js the important fraction of code to notice is the declaration of the function ('function(error,data)'). This is known as a callback function.
  - So what happens here is that the file reading operation will start in the background. And other processing can happen simultaneously while the file is being read. Once the file read operation is completed, this anonymous function will be called and the text "Reading Data completed" will be written to the console log.
2. Node uses the V8 JavaScript Runtime engine, the one which is used by Google Chrome. Node has a wrapper over the JavaScript engine which makes the runtime engine much faster and hence processing of requests within Node also become faster.
  3. Handling of concurrent requests – Another key functionality of Node is the ability to handle concurrent connections with a very minimal overhead on a single process.
  4. The Node.js library used JavaScript – This is another important aspect of development in Node.js. A major part of the development community are already well versed in javascript, and hence, development in Node.js becomes easier for a developer who knows javascript.



5. There are an Active and vibrant community for the Node.js framework. Because of the active community, there are always keys updates made available to the framework. This helps to keep the framework always up-to-date with the latest trends in web development.

## **Who uses Node.js**

Node.js is used by a variety of large companies. Below is a list of a few of them.

- Paypal – A lot of sites within Paypal have also started the transition onto Node.js.
- LinkedIn - LinkedIn is using Node.js to power their Mobile Servers, which powers the iPhone, Android, and Mobile Web products.
- Mozilla has implemented Node.js to support browser APIs which has half a billion installs.
- Ebay hosts their HTTP API service in Node.js

# When to use and not use Node.js

Node.js is best for usage in streaming or event-based real-time applications like

1. Chat applications
2. Game servers – Fast and high-performance servers that need to process thousands of requests at a time, then this is an ideal framework.
3. Good for collaborative environment – This is good for environments which manage documents. In document management environment you will have multiple people who post their documents and do constant changes by checking out and checking in documents. So Node.js is good for these environments because the event loop in Node.js can be triggered whenever documents are changed in a document managed environment.
4. Advertisement servers – Again here you could have thousands of requests to pull advertisements from the central server and Node.js can be an ideal framework to handle this.
5. Streaming servers – Another ideal scenario to use Node is for multimedia streaming servers wherein clients have requests to pull different multimedia contents from this server.

Node.js is good when you need high levels of concurrency but less amount of dedicated CPU time.

Best of all, since Node.js is built on javascript, it's best suited when you build client-side applications which are based on the same javascript framework.

## When to not use Node.js

Node.js can be used for a lot of applications with various purposes, the only scenario where it should not be used is if there are long processing times which is required by the application.

Node is structured to be single threaded. If any application is required to carry out some long running calculations in the background. So if the server is doing some calculation, it won't be able to process any other requests. As discussed above, Node.js is best when processing needs less dedicated CPU time.

# Chapter 2: Download & Install Node.js

To start building your Node.js applications, the first step is the installation of the node.js framework. The Node.js framework is available for a variety of operating systems right from Windows to Ubuntu and OS X. Once the Node.js framework is installed you can start building your first Node.js applications.

Node.js also has the ability to embedded external functionality or extended functionality by making use of custom modules. These modules have to be installed separately. An example of a module is the MongoDB module which allows you to work with MongoDB databases from your Node.js application.

# How to install node.js




The first steps in using Node.js is the installation of the Node.js libraries on the client system. To perform the installation of Node.js, perform the below steps;

**Step 1)** Go to the site <https://nodejs.org/en/download/> and download the necessary binary files. In our example, we are going to the download the 32-bit setup files for Node.js.

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

**LTS**  
Mature and Dependable

**Stable**  
Latest Features

 Windows Installer <small>node-v4.2.3-x86.msi</small>	 Macintosh Installer <small>node-v4.2.3.pkg</small>	 Source Code <small>node-v4.2.3.tar.gz</small>
--	--	---

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.exe)	32-bit	64-bit
Mac OS X Installer (.pkg)	64-bit	
Mac OS X Binaries (.tar.gz)	64-bit	

Download the 32-bit installer

**Step 2)** Double click on the downloaded .msi file to start the installation. Click the Run button in the first screen to begin the installation.

## Open File - Security Warning

Do you want to run this file?



Name: C:\All Downloads\Chrome\node-v4.2.3-x86.msi

Publisher: [Node.js Foundation](#)

Type: Windows Installer Package

From: C:\All Downloads\Chrome\node-v4.2.3-x86.msi

Run

Cancel

Always ask before opening this file



While files from the Internet can be useful, this file type can potentially harm your computer. Only run software from publishers you trust.  
[What's the risk?](#)

Click the  
Run button

**Step 3)** In the next screen, click the "Next" button to continue with the installation

## Node.js Setup

Welcome to the Node.js Setup Wizard



The Setup Wizard will install Node.js on your computer.

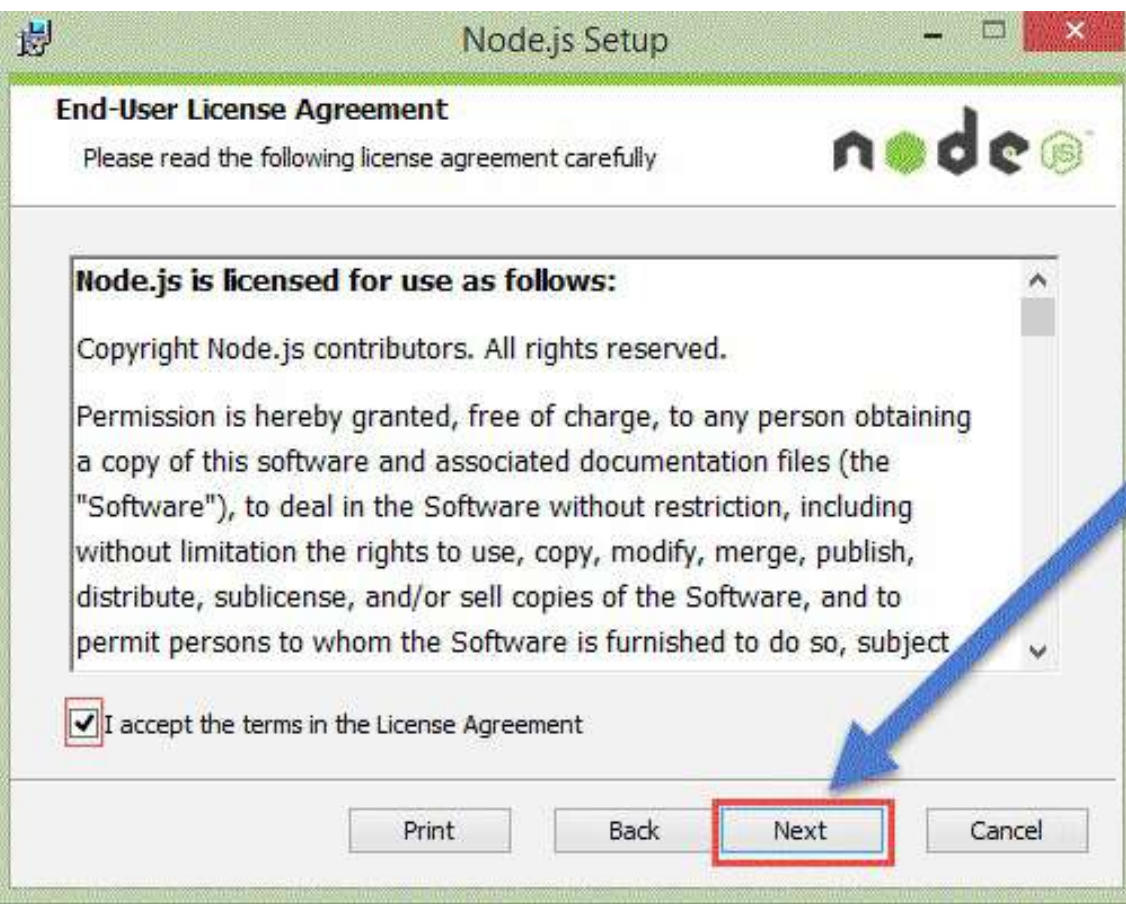
Back

Next

Cancel

Click the  
Next button

**Step 4)** In the next screen Accept the license agreement and click on the Next button.



Accept the license agreement and click the Next

**Step 5)** In the next screen, choose the location where Node.js needs to be installed and then click on the Next button.

1. First enter the file location for the installation of Node.js. This is where the files for Node.js will be stored after the installation.
2. Click on the Next button to proceed ahead with the installation.