Node.js is a very powerful JavaScript-based framework/platform built on Google Chrome's JavaScript V8 Engine. It is used to develop I/O intensive web applications like video streaming sites, single-page applications, and other web applications. Node.js is open source, completely free, and used by thousands of developers around the world.

Audience

This tutorial is designed for software programmers who want to learn the basics of Node.js and its architectural concepts. This tutorial will give you enough understanding on all the necessary components of Node.js with suitable examples.

Prerequisites

Before proceeding with this tutorial, you should have a basic understanding of JavaScript. As we are going to develop web-based applications using Node.js, it will be good if you have some understanding of other web technologies such as HTML, CSS, AJAX, etc.

Execute Node.js Online

For most of the examples given in this tutorial, you will find a **Try it** option, so just make use of this option to execute your Node.js programs on the spot and enjoy your learning.

Try the following example using the **Try it** option available at the top right corner of the below sample code box (on our website):

```
/* Hello World! program in Node.js */

console.log("Hello World!");
```

What is Node.js?

Node.js is a server side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009 and its latest version is v0.10.36. The definition of Node.js as supplied by its official documentation is as follows –

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

```
Node.js = Runtime Environment + JavaScript Library
```

Features of Node.js

Following are some of the important features that make Node.js the first choice of software architects.

- **Asynchronous and Event Driven** All APIs of Node.js library are asynchronous that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.

- **Very Fast** Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.

- **Single Threaded but Highly Scalable** - Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.

- **No Buffering** - Node.js applications never buffer any data. These applications simply output the data in chunks.

- **License** - Node.js is released under the MIT license.

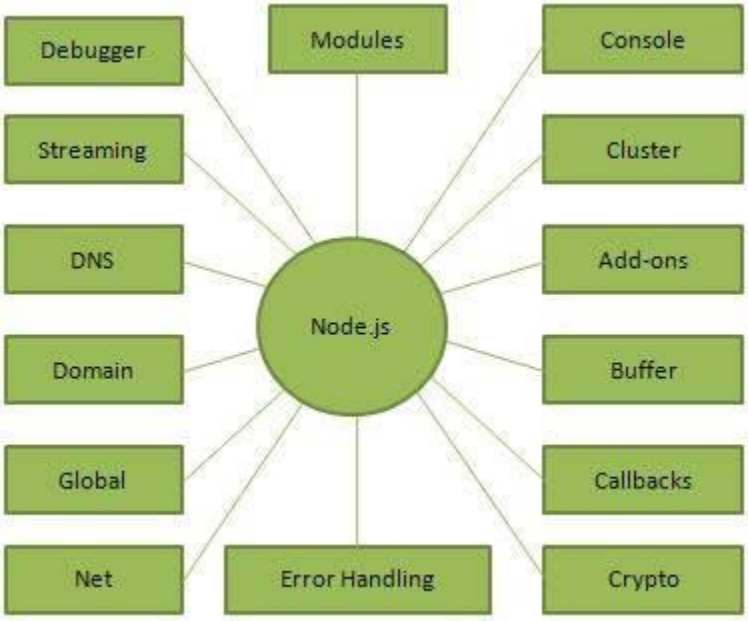
Who Uses Node.js?

Following is the link on github wiki containing an exhaustive list of projects, application and companies which are using Node.js. This list includes eBay, General Electric, GoDaddy, Microsoft, PayPal, Uber, Wikipins, Yahoo!, and Yammer to name a few.

- Projects, Applications, and Companies Using Node

Concepts

The following diagram depicts some important parts of Node.js which we will discuss in detail in the subsequent chapters.



Where to Use Node.js?

Following are the areas where Node.js is proving itself as a perfect technology partner.

- I/O bound Applications
- Data Streaming Applications

- Data Intensive Real time Applications (DIRT)
- JSON APIs based Applications
- Single Page Applications

Where Not to Use Node.js?

It is not advisable to use Node.js for CPU intensive applications.

Try it Option Online

You really do not need to set up your own environment to start learning Node.js. Reason is very simple, we already have set up Node.js environment online, so that you can execute all the available examples online at the same time when you are doing your theory work. This gives you confidence in what you are reading and to check the result with different options. Feel free to modify any example and execute it online.

Try following example using **Try it** option available at the top right corner of the below sample code box:

```
/* Hello World! program in Node.js */
console.log("Hello World!");
```

For most of the examples given in this tutorial, you will find **Try it**option, so just make use of it and enjoy your learning.

Local Environment Setup

If you are still willing to set up your environment for Node.js, you need the following two softwares available on your computer, (a) Text Editor and (b) The Node.js binary installables.

Text Editor

This will be used to type your program. Examples of few editors include Windows Notepad, OS Edit command, Brief, Epsilon, EMACS, and vim or vi.

Name and version of text editor can vary on different operating systems. For example, Notepad will be used on Windows, and vim or vi can be used on windows as well as Linux or UNIX.

The files you create with your editor are called source files and contain program source code. The source files for Node.js programs are typically named with the extension "**.js**".

Before starting your programming, make sure you have one text editor in place and you have enough experience to write a computer program, save it in a file, and finally execute it.

The Node.js Runtime

The source code written in source file is simply javascript. The Node.js interpreter will be used to interpret and execute your javascript code.

Node.js distribution comes as a binary installable for SunOS , Linux, Mac OS X, and Windows operating systems with the 32-bit (386) and 64-bit (amd64) x86 processor architectures.

Following section guides you on how to install Node.js binary distribution on various OS.

Download Node.js archive

Download latest version of Node.js installable archive file from Node.js Downloads. At the time of writing this tutorial, following are the versions available on different OS.

OS	Archive name
Windows	node-v0.12.0-x64.msi
Linux	node-v0.12.0-linux-x86.tar.gz
Mac	node-v0.12.0-darwin-x86.tar.gz
SunOS	node-v0.12.0-sunos-x86.tar.gz

Installation on UNIX/Linux/Mac OS X, and SunOS

Based on your OS architecture, download and extract the archive node-v0.12.0-**osname**.tar.gz into /tmp, and then finally move extracted files into /usr/local/nodejs directory. For example:

```
$ sudo apt-get install nodejs
```

Add /usr/local/nodejs/bin to the PATH environment variable.

OS	Output
Linux	export PATH=/usr/local/nodejs/bin
Mac	export PATH=$PATH:/usr/local/nodejs/bin
FreeBSD	export PATH=$PATH:/usr/local/nodejs/bin

Installation on Windows

Use the MSI file and follow the prompts to install the Node.js. By default, the installer uses the Node.js distribution in C:\Program Files\nodejs. The installer should set the C:\Program Files\nodejs\bin directory in window's PATH environment variable. Restart any open command prompts for the change to take effect.

Verify installation: Executing a File

Create a js file named **main.js** on your machine (Windows or Linux) having the following code.

```
/* Hello, World! program in node.js */
console.log("Hello, World!")
```

Now execute main.js file using Node.js interpreter to see the result:

```
$ node main.js
```

If everything is fine with your installation, this should produce the following result:

```
Hello, World!
```