Node.js Tutorial

Node.js tutorial provides basic and advanced concepts of Node.js. Our Node.js tutorial is designed for beginners and professionals both.

Node.js is a cross-platform environment and library for running JavaScript applications which is used to create networking and server-side applications.

Our Node.js tutorial includes all topics of Node.js such as Node.js installation on windows and linux, REPL, package manager, callbacks, event loop, os, path, query string, cryptography, debugger, URL, DNS, Net, UDP, process, child processes, buffers, streams, file systems, global objects, web modules etc. There are also given Node.js interview questions to help you better understand the Node.js technology.

Node.js is a cross-platform runtime environment and library for running JavaScript applications outside the browser. It is used for creating server-side and networking web applications. It is open source and free to use. It can be downloaded from this link <https://nodejs.org/en/>

Many of the basic modules of Node.js are written in JavaScript. Node.js is mostly used to run real-time server applications.

The definition given by its official documentation is as follows:

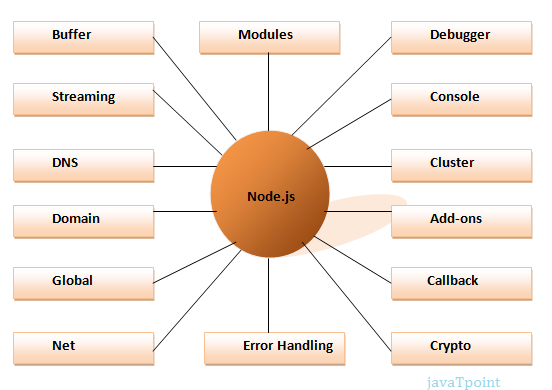
?Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.?

Node.js also provides a rich library of various JavaScript modules to simplify the development of web applications.

1. Node.js = Runtime Environment + JavaScript Library

**Different parts of Node.js**

The following diagram specifies some important parts of Node.js:



Features of Node.js

Following is a list of some important features of Node.js that makes it the first choice of software architects.

1. **Extremely fast:**Node.js is built on Google Chrome's V8 JavaScript Engine, so its library is very fast in code execution.
2. **I/O is Asynchronous and Event Driven:**All APIs of Node.js library are asynchronous i.e. non-blocking. So 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. It is also a reason that it is very fast.
3. **Single threaded:**Node.js follows a single threaded model with event looping.
4. **Highly Scalable:**Node.js is highly scalable because event mechanism helps the server to respond in a non-blocking way.
5. **No buffering:**Node.js cuts down the overall processing time while uploading audio and video files. Node.js applications never buffer any data. These applications simply output the data in chunks.
6. **Open source:**Node.js has an open source community which has produced many excellent modules to add additional capabilities to Node.js applications.
7. **License:**Node.js is released under the MIT license.

Install Node.js on Windows

To install and setup an environment for Node.js, you need the following two softwares available on your computer:

1. Text Editor.
2. Node.js Binary installable

**Text Editor:**

The text editor is used to type your program. For example: Notepad is used in Windows, vim or vi can be used on Windows as well as Linux or UNIX. The name and version of the text editor can be different from operating system to operating system.

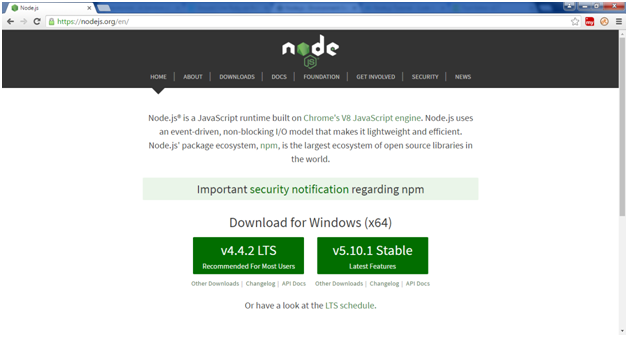
The files created with text editor are called source files and contain program source code. The source files for Node.js programs are typically named with the extension ".js".

**The Node.js Runtime:**

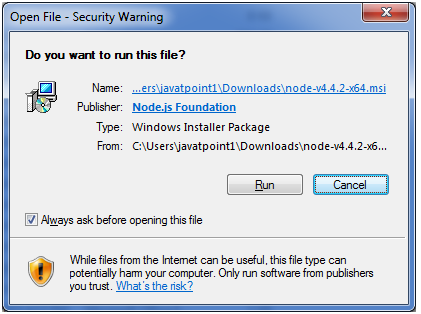
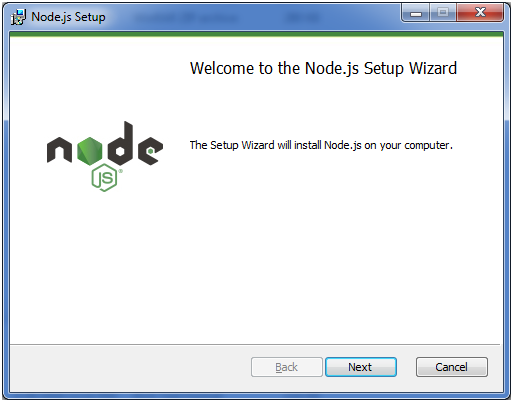
The source code written in source file is simply JavaScript. It is interpreted and executed by the Node.js interpreter.

**How to download Node.js:**

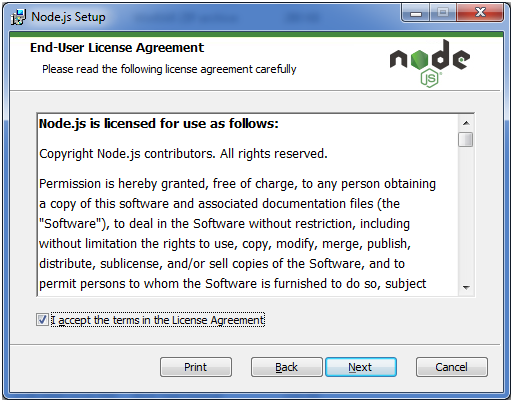
You can download the latest version of Node.js installable archive file from <https://nodejs.org/en/>



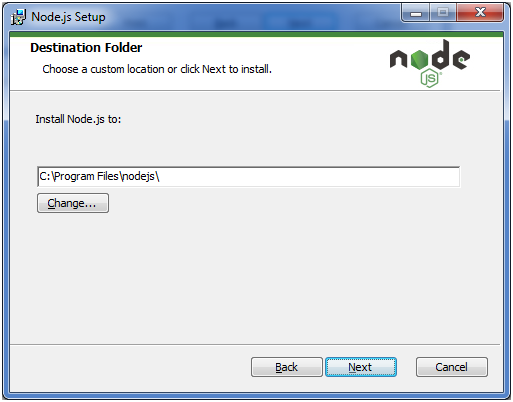
Here, you deploy the installation of node-v4.4.2 LTS recommended for most users.

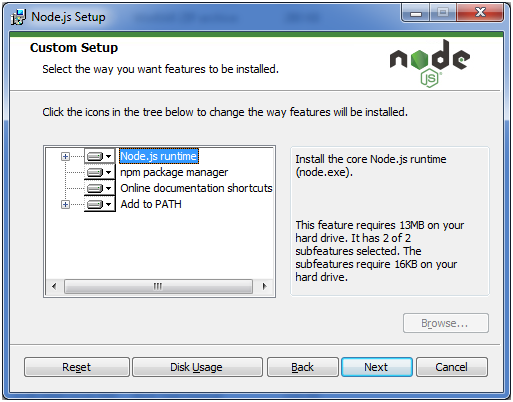
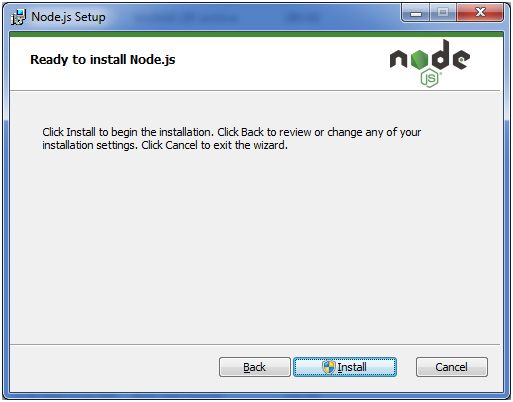
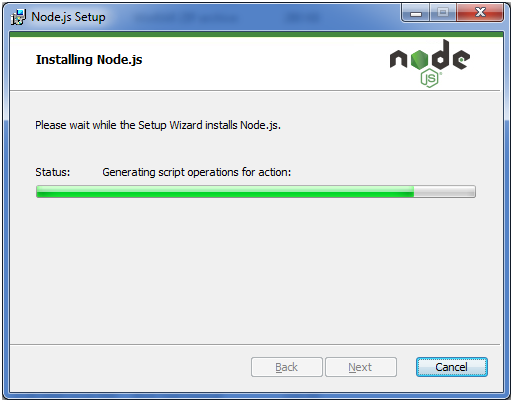
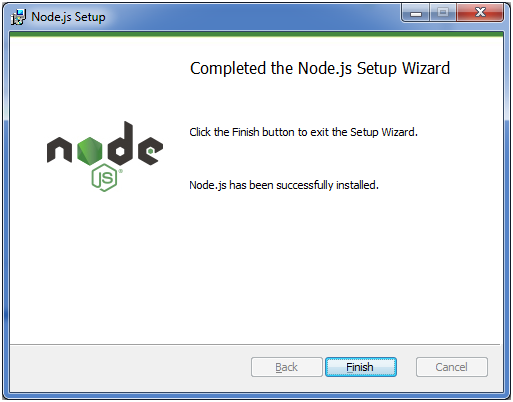
Accept the terms of license agreement.



Choose the location where you want to install.



Ready to install:

   s

# Node.js First Example

There can be console-based and web-based node.js applications.

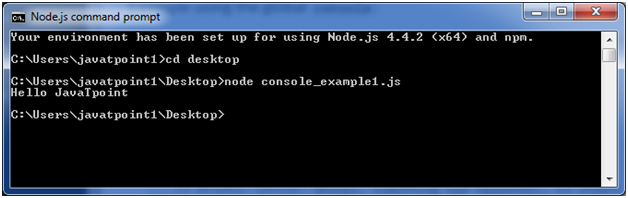
## **Node.js console-based Example**

*File: console\_example1.js*

1. console.log('Hello JavaTpoint');

Open Node.js command prompt and run the following code:

1. node console\_example1.js



Here, console.log() function displays message on console.

## **Node.js web-based Example**

A node.js web application contains the following three parts:

1. **Import required modules:** The "require" directive is used to load a Node.js module.
2. **Create server:**You have to establish a server which will listen to client's request similar to Apache HTTP Server.
3. **Read request and return response:** Server created in the second step will read HTTP request made by client which can be a browser or console and return the response.

**How to create node.js web applications**

Follow these steps:

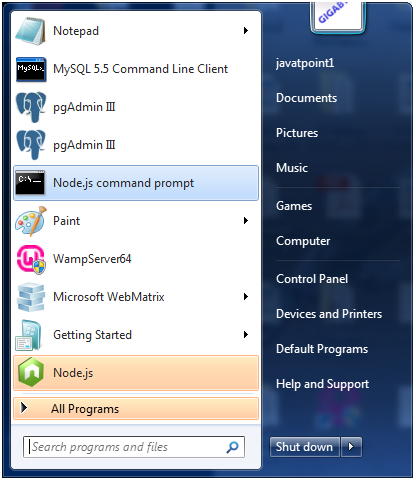
1. **Import required module:**The first step is to use ?require? directive to load http module and store returned HTTP instance into http variable. For example:
   1. var http = require("http");
2. **Create server:**In the second step, you have to use created http instance and call http.createServer() method to create server instance and then bind it at port 8081 using listen method associated with server instance. Pass it a function with request and response parameters and write the sample implementation to return "Hello World". For example:
   1. http.createServer(function (request, response) {
   2. // Send the HTTP header
   3. // HTTP Status: 200 : OK
   4. // Content Type: text/plain
   5. response.writeHead(200, {'Content-Type': 'text/plain'});
   6. // Send the response body as "Hello World"
   7. response.end('Hello World\n');
   8. }).listen(8081);
   9. // Console will print the message
   10. console.log('Server running at http://127.0.0.1:8081/');
3. **Combine step1 and step2 together** in a file named "main.js".

*File: main.js*

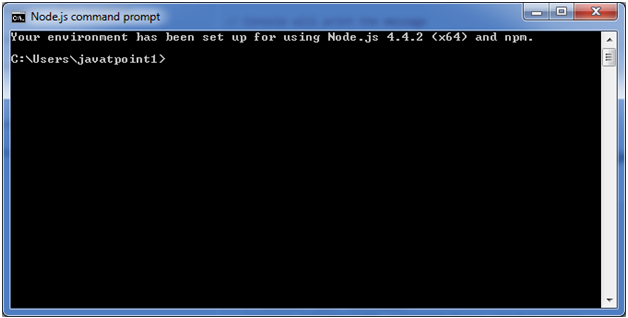
* 1. var http = require("http");
  2. http.createServer(function (request, response) {
  3. // Send the HTTP header
  4. // HTTP Status: 200 : OK
  5. // Content Type: text/plain
  6. response.writeHead(200, {'Content-Type': 'text/plain'});
  7. // Send the response body as "Hello World"
  8. response.end('Hello World\n');
  9. }).listen(8081);
  10. // Console will print the message
  11. console.log('Server running at http://127.0.0.1:8081/');

**How to start your server:**

Go to start menu and click on the Node.js command prompt.



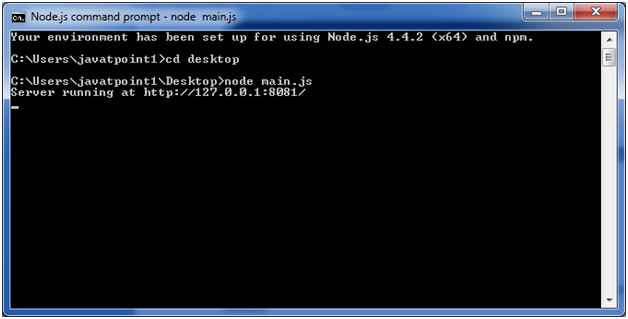
Now command prompt is open:



**Set path:**Here we have save "main.js" file on the desktop.

So type **cd desktop** on the command prompt. After that execute the main.js to start the server as follows:

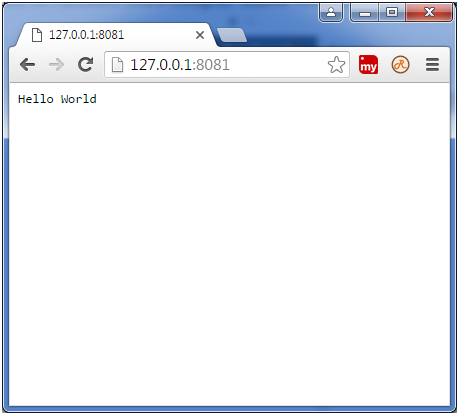
1. node main.js



Now server is started.

**Make a request to Node.js server:**

Open http://127.0.0.1:8081/ in any browser. You will see the following result.



Now, if you make any changes in the "main.js" file, you need to again run the "node main.js" command.

# Node.js REPL

The term REPL stands for **Read Eval Print**and**Loop**. It specifies a computer environment like a window console or a Unix/Linux shell where you can enter the commands and the system responds with an output in an interactive mode.

## **REPL Environment**

The Node.js or node come bundled with REPL environment. Each part of the REPL environment has a specific work.

**Read:** It reads user's input; parse the input into JavaScript data-structure and stores in memory.

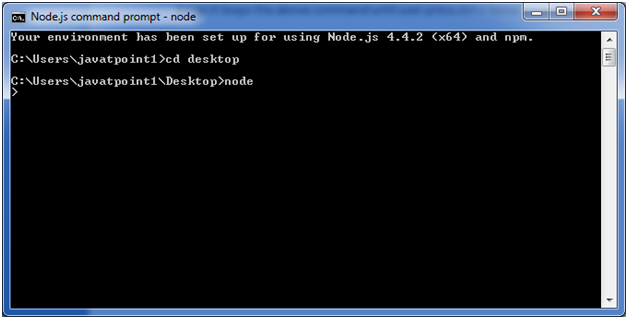
**Eval:**It takes and evaluates the data structure.

**Print:**It prints the result.

**Loop:** It loops the above command until user press ctrl-c twice.

## **How to start REPL**

You can start REPL by simply running "node" on the command prompt. See this:

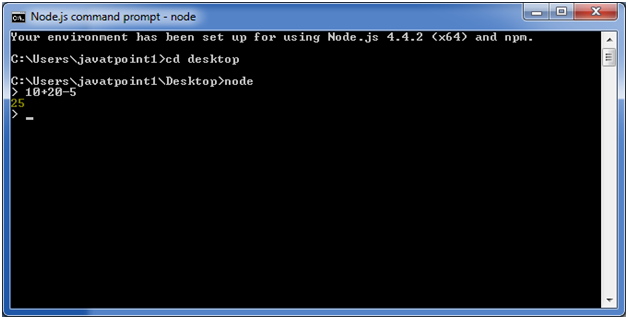


You can execute various mathematical operations on REPL Node.js command prompt:

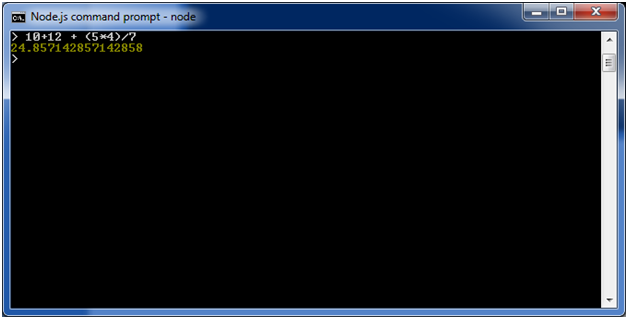
## **Node.js Simple expressions**

After starting REPL node command prompt put any mathematical expression:

1. Example: **>**10+20-5
2. 25



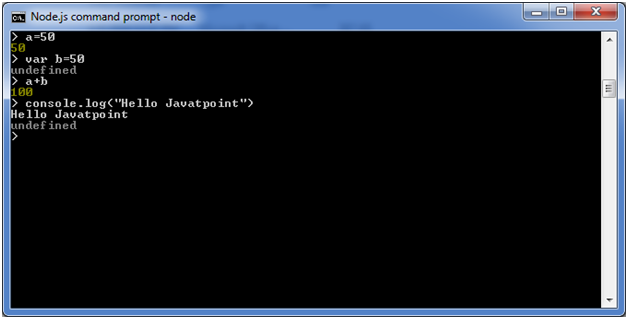
1. Example2: **>**10+12 + (5\*4)/7



## **Using variable**

Variables are used to store values and print later. If you don't use **var**keyword then value is stored in the variable and printed whereas if **var** keyword is used then value is stored but not printed. You can print variables using console.log().

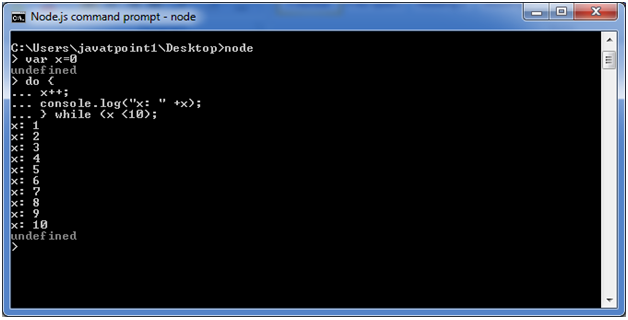
**Example:**



## **Node.js Multiline expressions**

Node REPL supports multiline expressions like JavaScript. See the following do-while loop example:

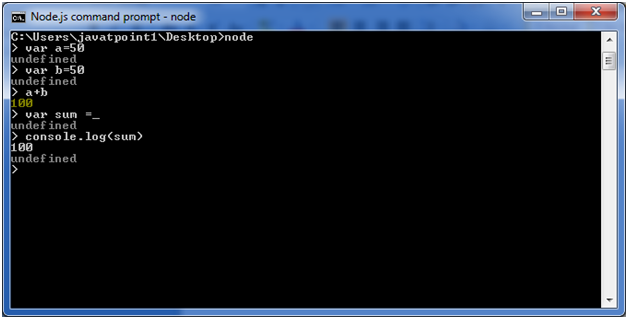
1. var x = 0
2. undefined
3. **>** do {
4. ... x++;
5. ... console.log("x: " + x);
6. ... } while ( x **<** **10** );



## **Node.js Underscore Variable**

You can also use underscore \_ to get the last result.

**Example:**

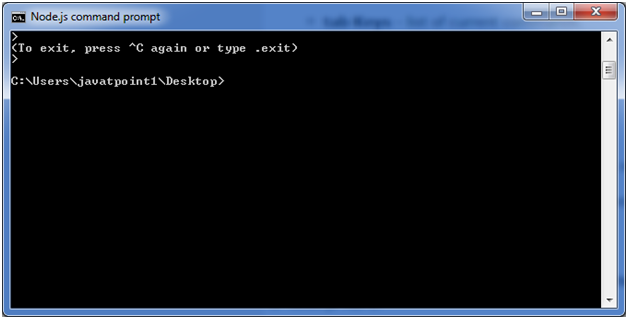


## **Node.js REPL Commands**

|  |  |
| --- | --- |
| Commands | Description |
| ctrl + c | It is used to terminate the current command. |
| ctrl + c twice | It terminates the node repl. |
| ctrl + d | It terminates the node repl. |
| up/down keys | It is used to see command history and modify previous commands. |
| tab keys | It specifies the list of current command. |
| .help | It specifies the list of all commands. |
| .break | It is used to exit from multi-line expressions. |
| .clear | It is used to exit from multi-line expressions. |
| .save filename | It saves current node repl session to a file. |
| .load filename | It is used to load file content in current node repl session. |

## **Node.js Exit REPL**

Use ctrl + c command twice to come out of Node.js REPL.

ss