What is Node.js

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:

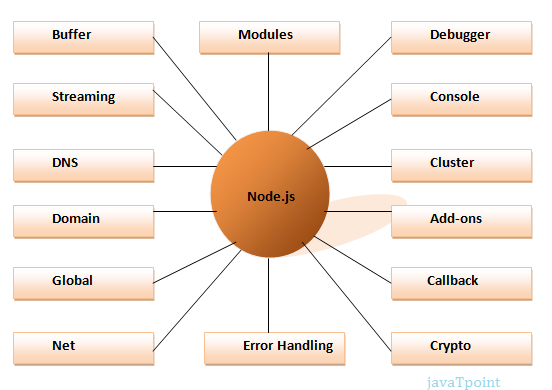
Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

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.

# **Node.js First Example**

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

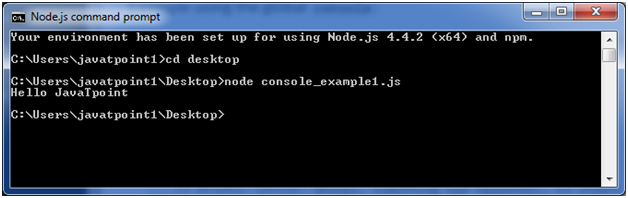
## Node.js console-based Example

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:

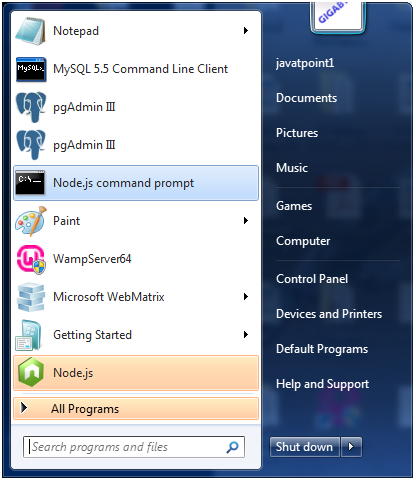
1. **Import required module:**The first step is to use ?require? directive to load http module and store returned HTTP instance into http variable. For example:
   1. var http = require("http");
2. **Create server:**In the second step, you have to use created http instance and call http.createServer() 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

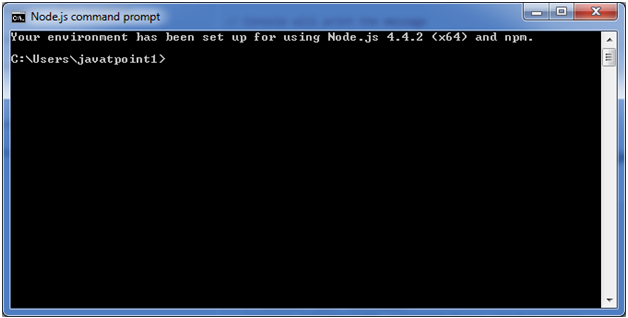
* 1. var http = require("http");
  2. http.createServer(function (request, response) {
  3. // Send the HTTP header
  4. // HTTP Status: 200 : OK
  5. // Content Type: text/plain
  6. response.writeHead(200, {'Content-Type': 'text/plain'});
  7. // Send the response body as "Hello World"
  8. response.end('Hello 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.



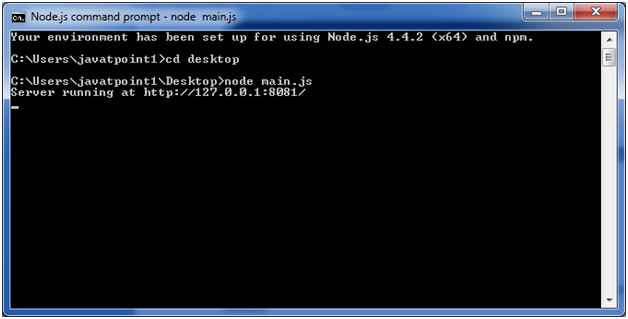
Now command prompt is open:



**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:

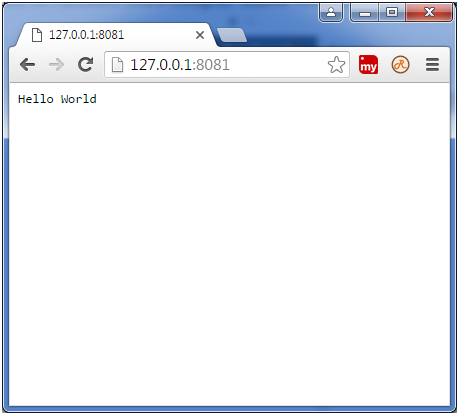
1. node main.js



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.



# **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.

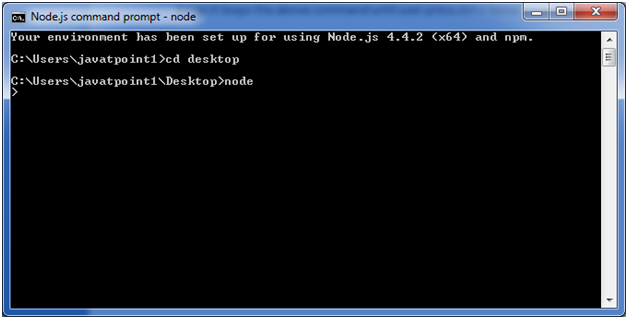
**Eval:**It takes and evaluates the data structure.

**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:

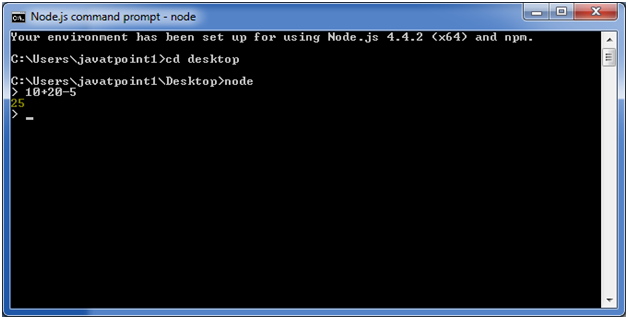


You can execute various mathematical operations on REPL Node.js command prompt:

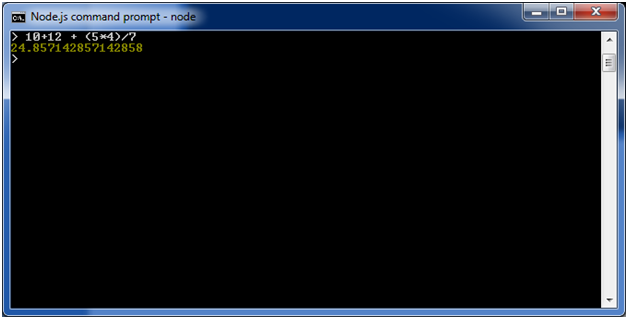
## Node.js Simple expressions

After starting REPL node command prompt put any mathematical expression:

1. Example: **>**10+20-5
2. 25



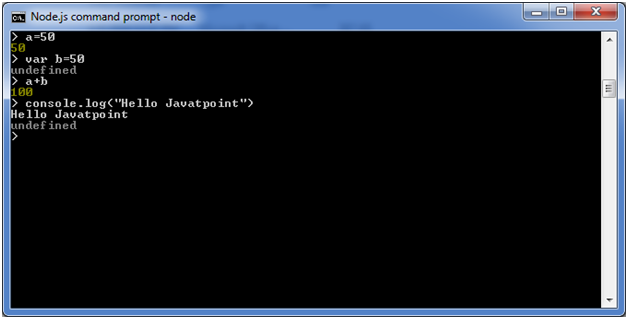
1. Example2: **>**10+12 + (5\*4)/7



## 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().

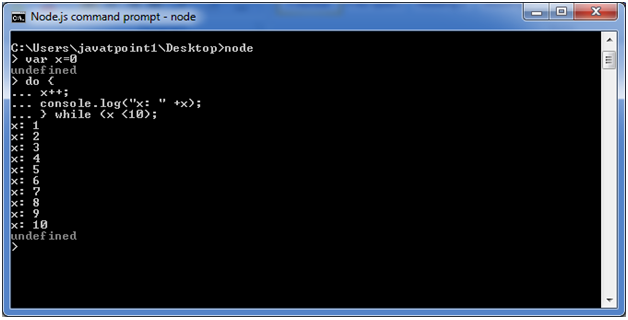
**Example:**



## 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

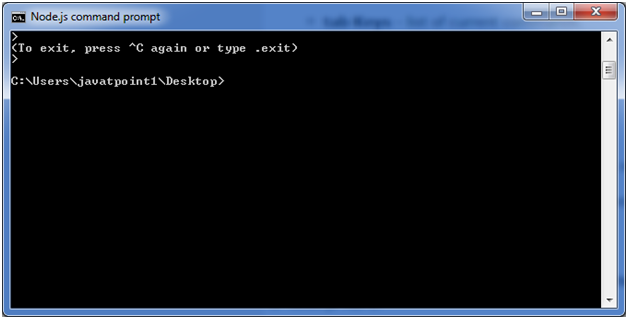
You can also use underscore \_ to get the last result.

**Example:**

## Node.js REPL Commands

## Node.js Exit REPL

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



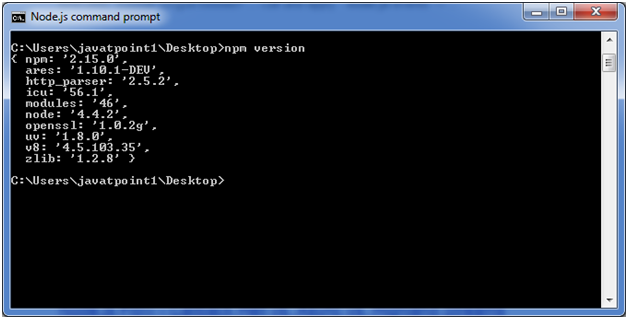
# **Node.js Package Manager**

Node Package Manager provides two main functionalities:

* It provides online repositories for node.js packages/modules which are searchable on search.nodejs.org
* It also provides command line utility to install Node.js packages, do version management and dependency management of Node.js packages.

The npm comes bundled with Node.js installables in versions after that v0.6.3. You can check the version by opening Node.js command prompt and typing the following command:

1. npm  version



## Installing Modules using npm

Following is the syntax to install any Node.js module:

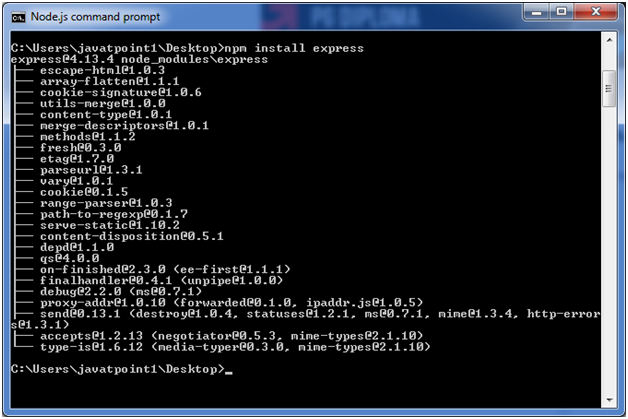
1. npm install **<**Module Name**>**

Let's install a famous Node.js web framework called express:

Open the Node.js command prompt and execute the following command:

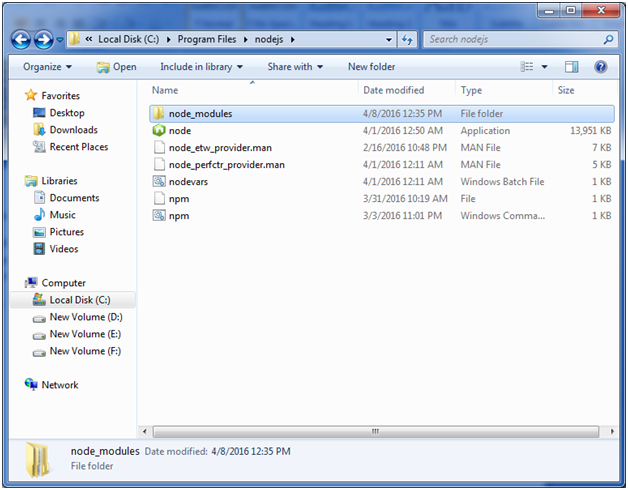
1. npm install express

You can see the result after installing the "express" framework.



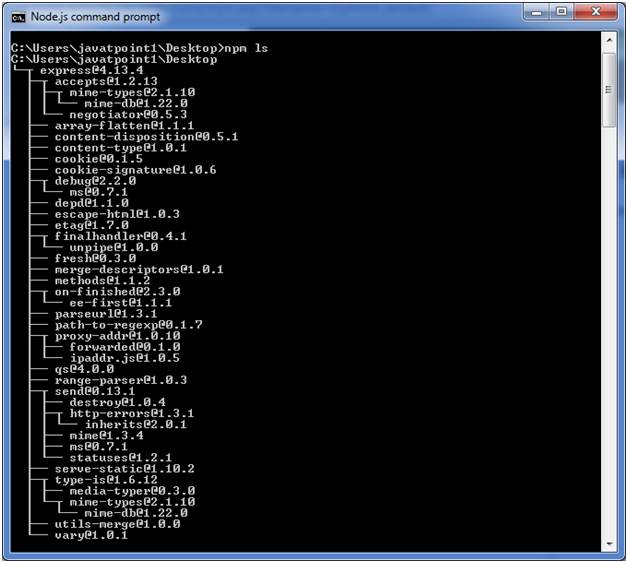
## Global vs Local Installation

By default, npm installs dependency in local mode. Here local mode specifies the folder where Node application is present. For example if you installed express module, it created node\_modules directory in the current directory where it installed express module.



You can use npm ls command to list down all the locally installed modules.

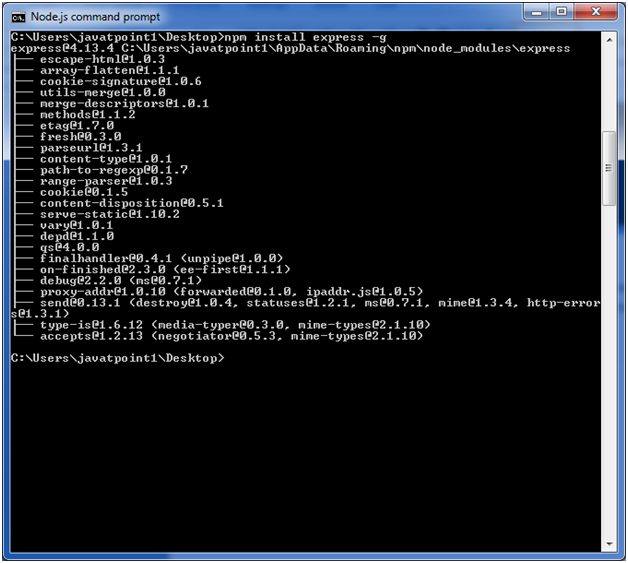
Open the Node.js command prompt and execute "npm ls":



Globally installed packages/dependencies are stored in system directory. Let's install express module using global installation. Although it will also produce the same result but modules will be installed globally.

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

1. npm install express -g

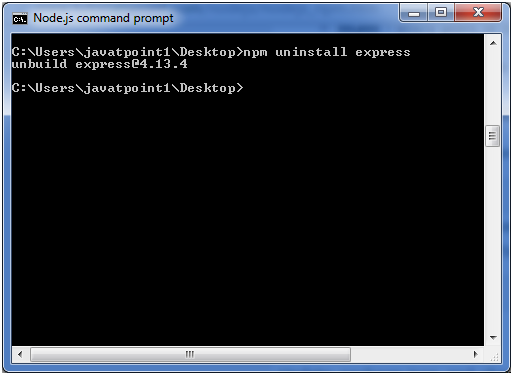


Here first line tells about the module version and its location where it is getting installed.

## Uninstalling a Module

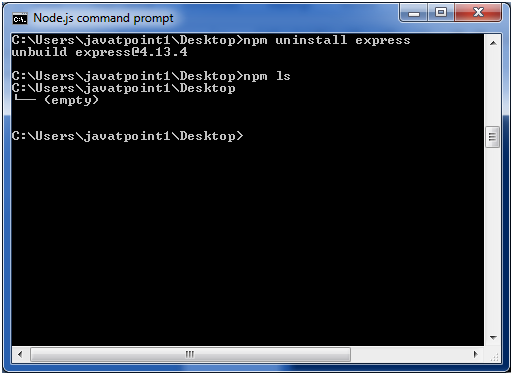
To uninstall a Node.js module, use the following command:

1. npm uninstall express



The Node.js module is uninstalled. You can verify by using the following command:

1. npm ls



You can see that the module is empty now.

## Searching a Module

"npm search express" command is used to search express or module.

1. npm search express

