

Introduction to Node.js

Node.js is an open-source and cross-platform JavaScript runtime environment. It is a popular tool for almost any kind of project!

Node.js runs the V8 JavaScript engine, the core of Google Chrome, outside of the browser. This allows Node.js to be very performant.

A Node.js app runs in a single process, without creating a new thread for every request. Node.js provides a set of asynchronous I/O primitives in its standard library that prevent JavaScript code from blocking and generally, libraries in Node.js are written using non-blocking paradigms, making blocking behavior the exception rather than the norm.

When Node.js performs an I/O operation, like reading from the network, accessing a database or the filesystem, instead of blocking the thread and wasting CPU cycles waiting, Node.js will resume the operations 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.

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

Node.js Concepts

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

However, it is not advisable to use Node.js for CPU intensive applications.

Node.js is primarily used to build network programs such as Web servers. However, you can build different types of applications such as command line applications, web applications, real-time chat applications, REST APIs etc.

Thousands of open-source libraries for Node.js are available, most of them hosted on the npm website, npm is a package manager for the JavaScript programming language. A number web frameworks can be used to accelerate the development of applications. Some of the popular frameworks are Express.js, Feathers.js, Koa.js, Sails.js, Meteor, and many others.

Number of IDEs such as Atom, JetBrains WebStorm, NetBeans, and Visual Studio Code support development of Node.js applications. Cloud-hosting platforms like Google Cloud Platform and AWS Elastic Beanstalk can be used to host Node.js applications when the response comes back.

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:

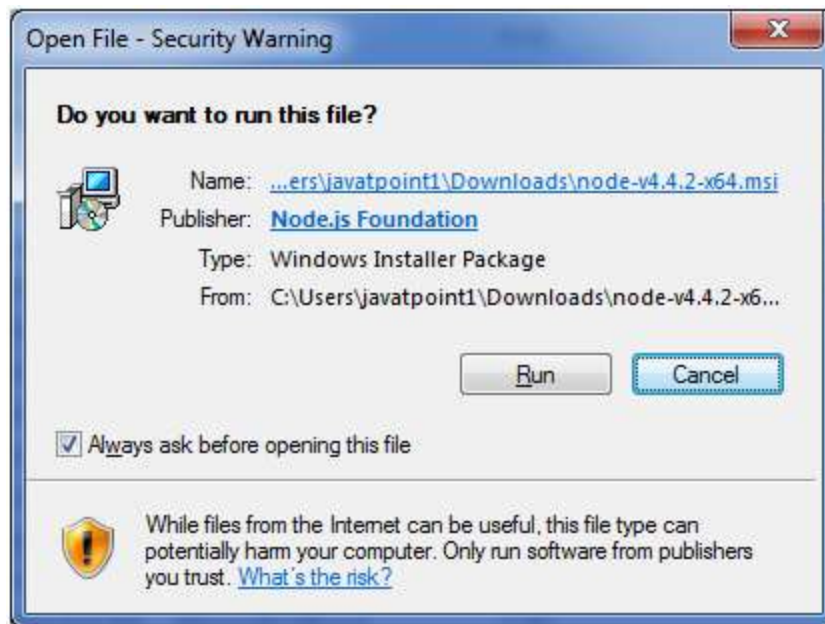
The source code written in source file is simply JavaScript. It is interpreted and executed by the Node.js interpreter.

How to download Node.js:

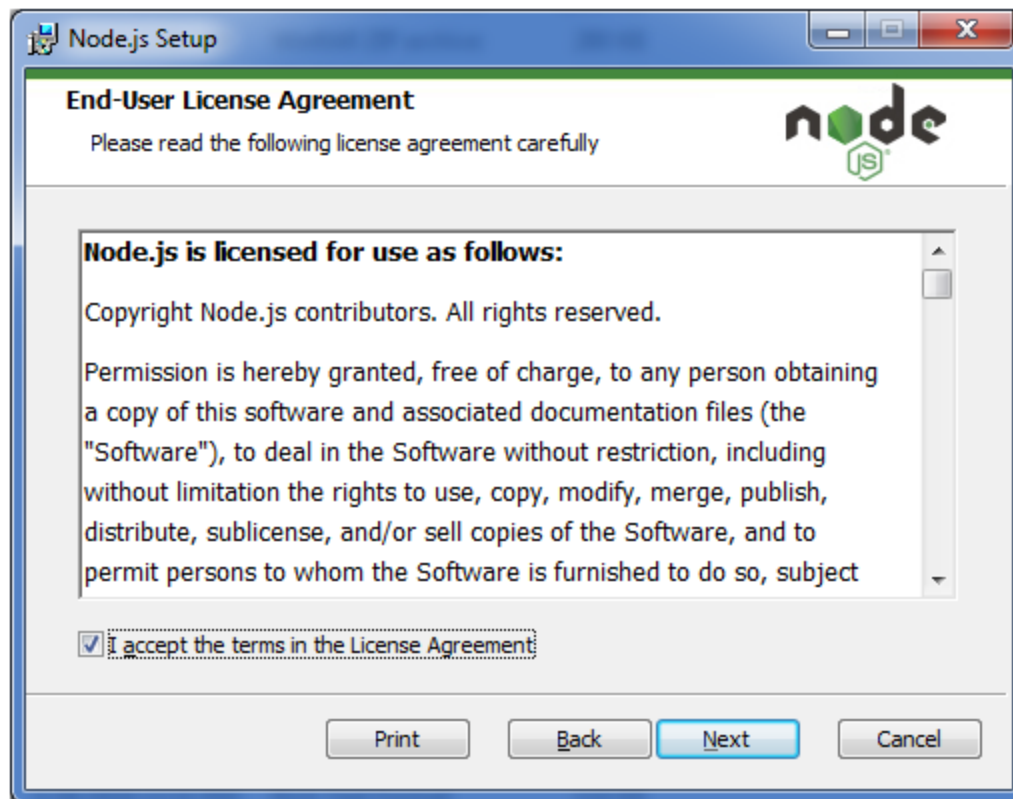
You can download the latest version of Node.js installable archive file from <https://nodejs.org/en/>



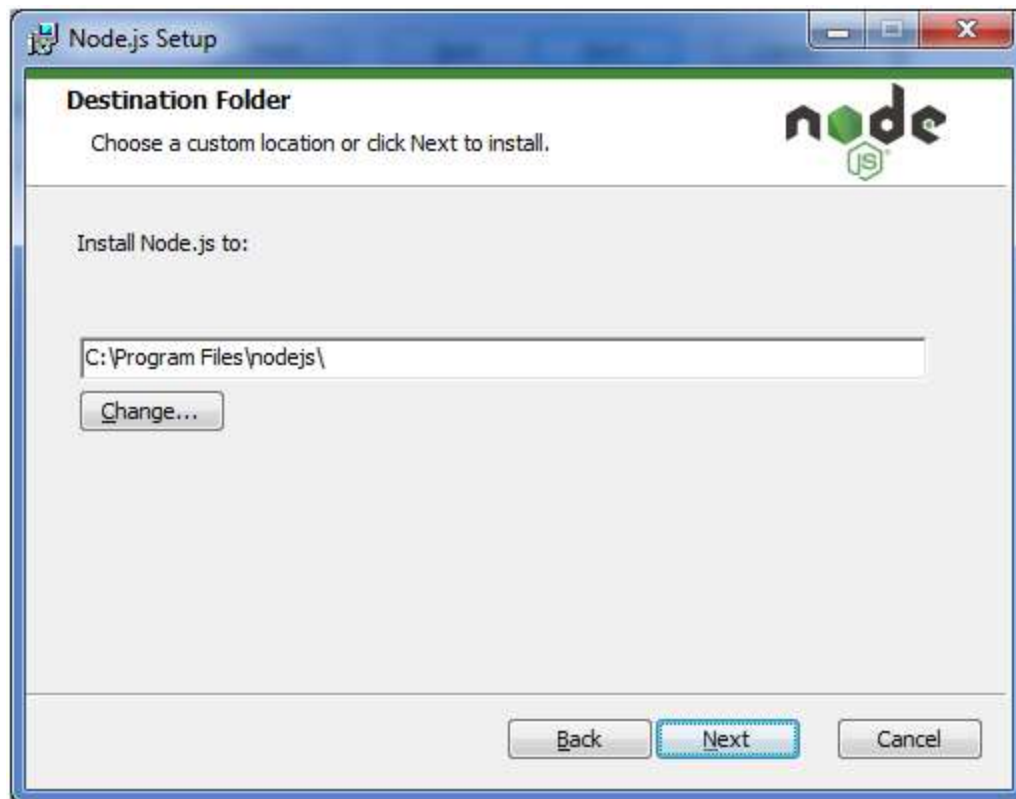
Here, you deploy the installation of node-v4.4.2 LTS recommended for most users.



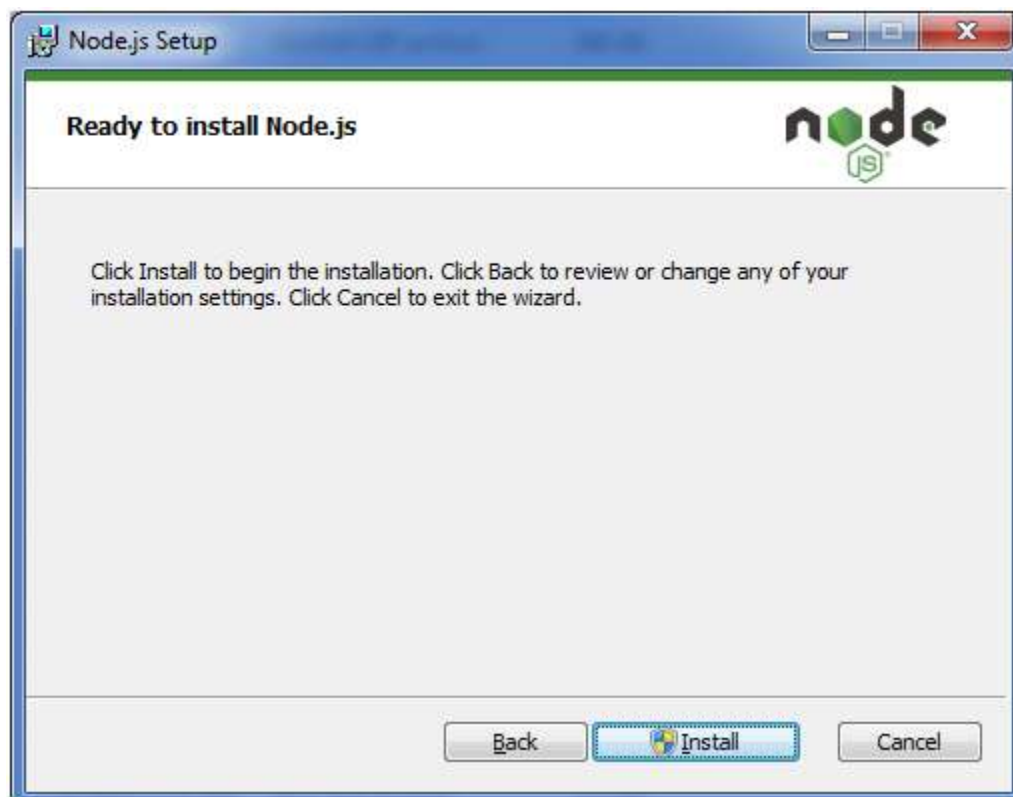
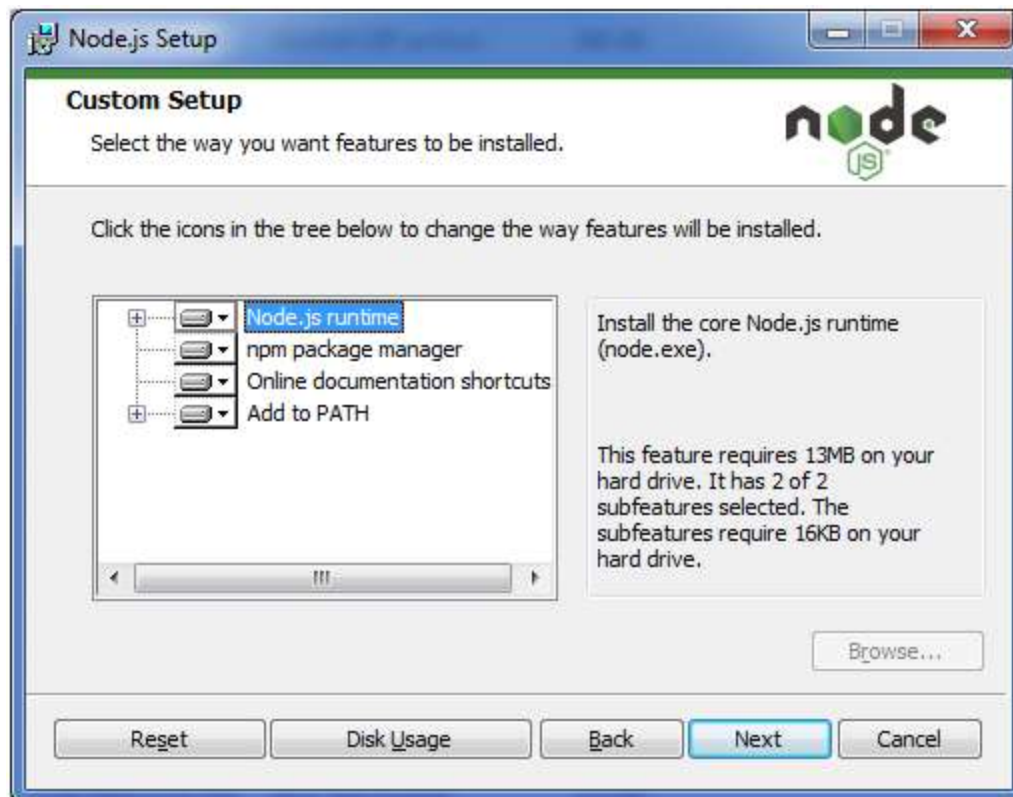
Accept the terms of license agreement.

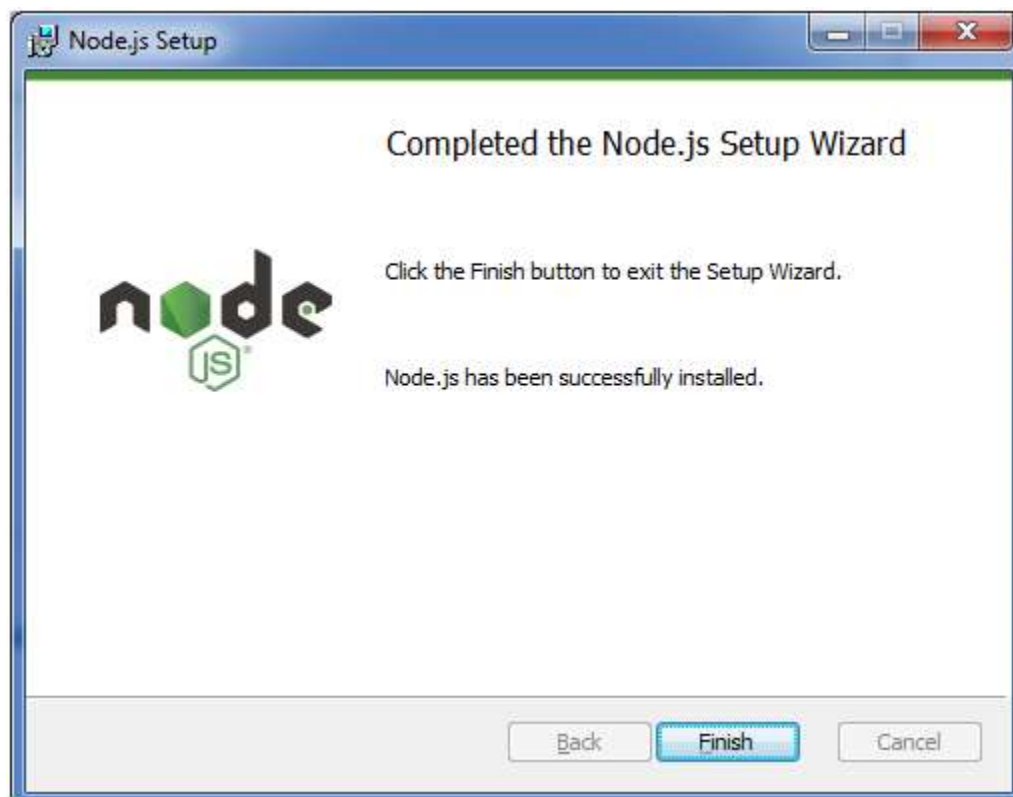
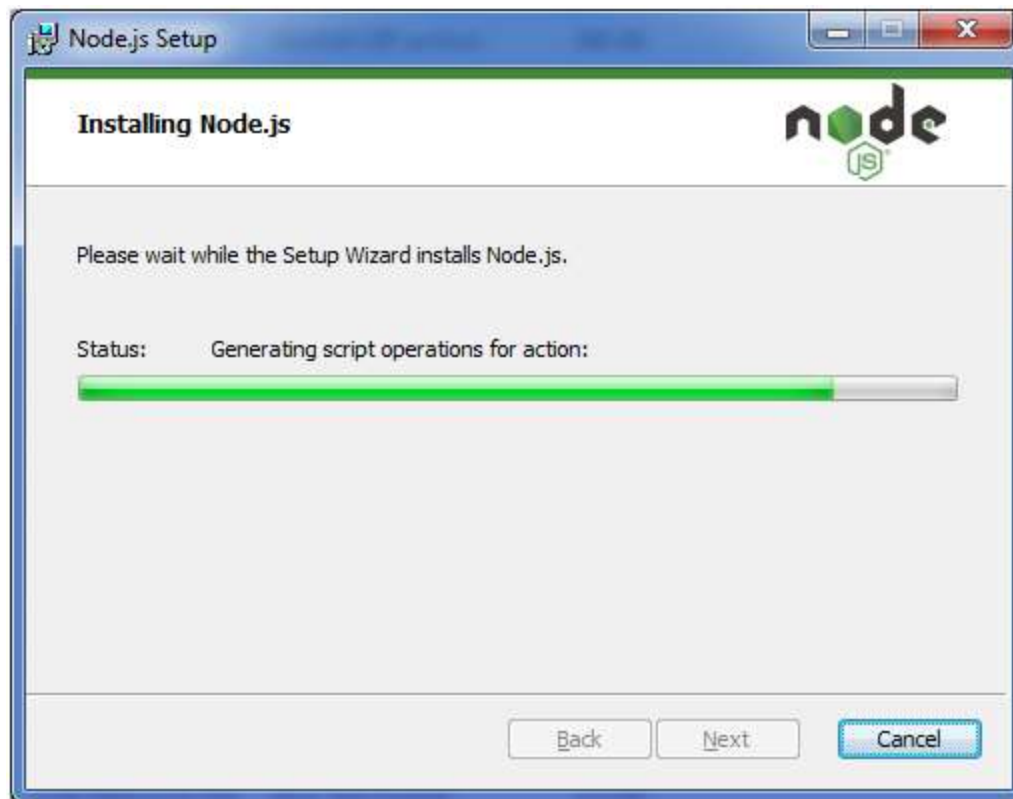


Choose the location where you want to install.



Ready to install:





Node.js First Example

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

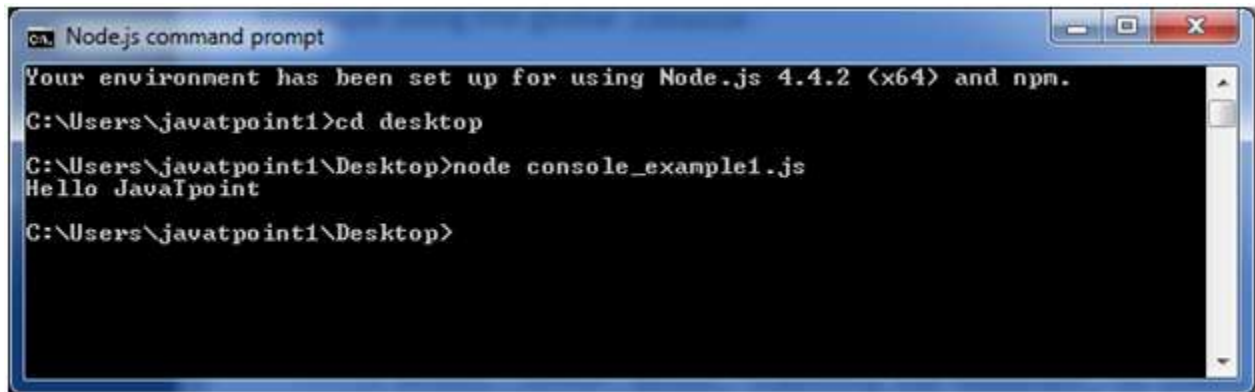
Node.js console-based Example

File: console_example1.js

1. `console.log('Hello JavaTpoint');`

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

1. `node console_example1.js`



```
Node.js command prompt
Your environment has been set up for using Node.js 4.4.2 (x64) and npm.
C:\Users\javatpoint1>cd desktop
C:\Users\javatpoint1\Desktop>node console_example1.js
Hello JavaTpoint
C:\Users\javatpoint1\Desktop>
```

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".

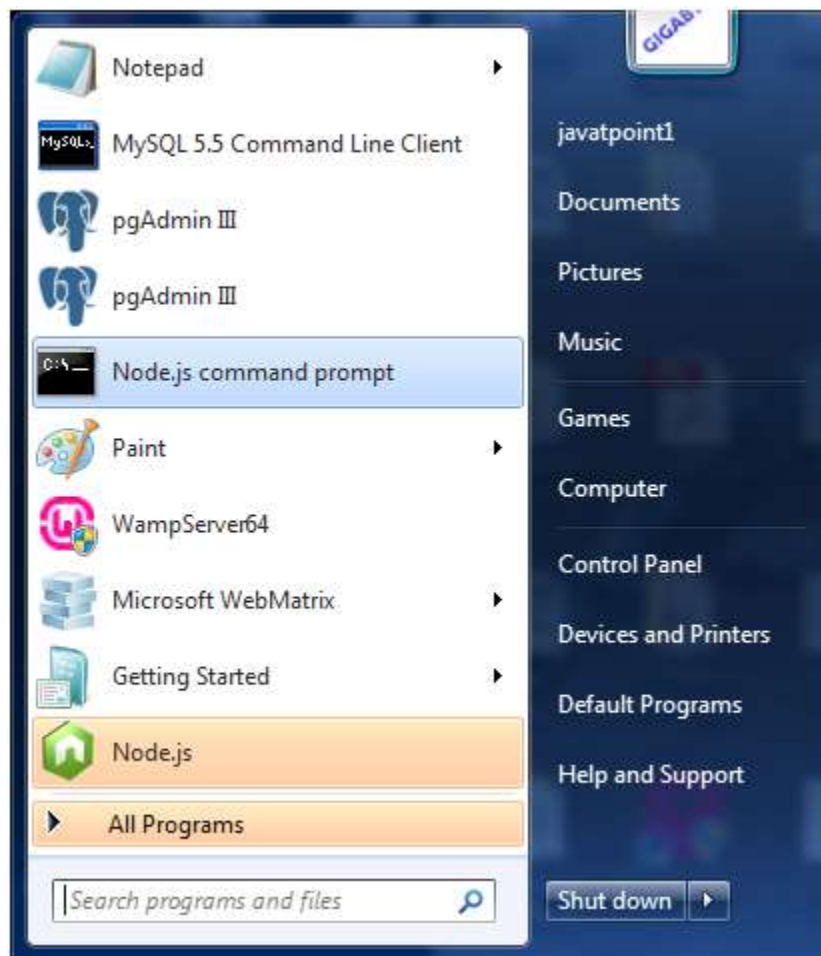
File: main.js

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:

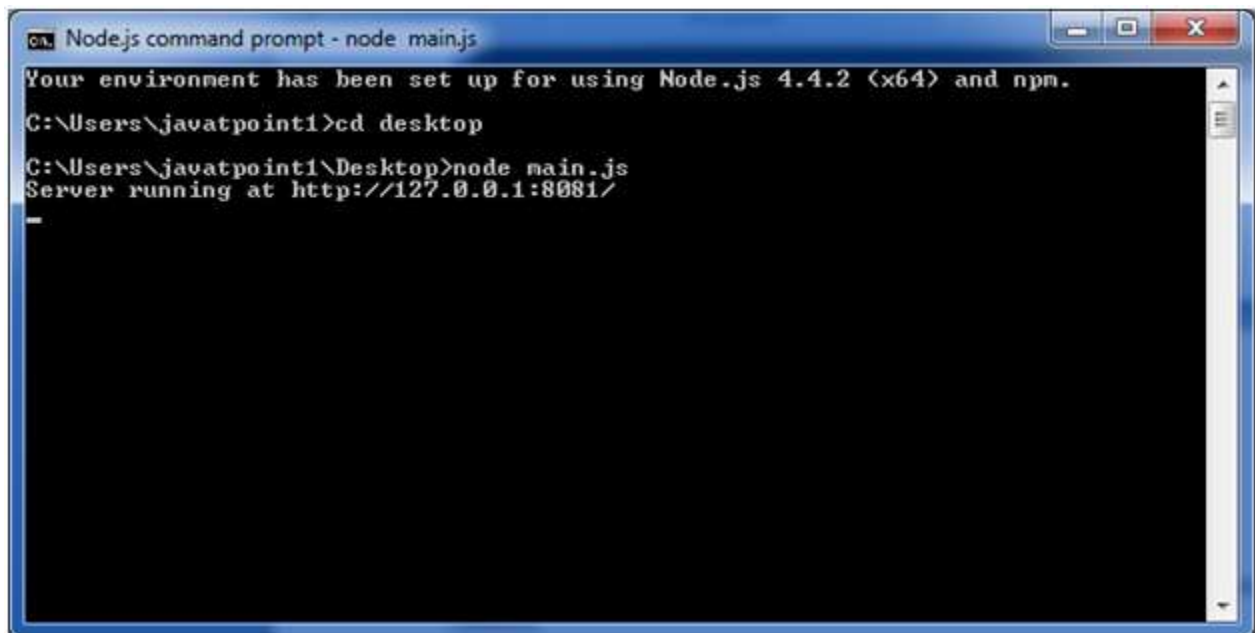
A screenshot of a Windows command prompt window titled "Node.js command prompt". The window has a blue title bar with standard minimize, maximize, and close buttons. The text inside the window reads: "Your environment has been set up for using Node.js 4.4.2 (x64) and npm." followed by the current directory path "C:\Users\javatpoint1>".

```
Node.js command prompt
Your environment has been set up for using Node.js 4.4.2 (x64) and npm.
C:\Users\javatpoint1>
```

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

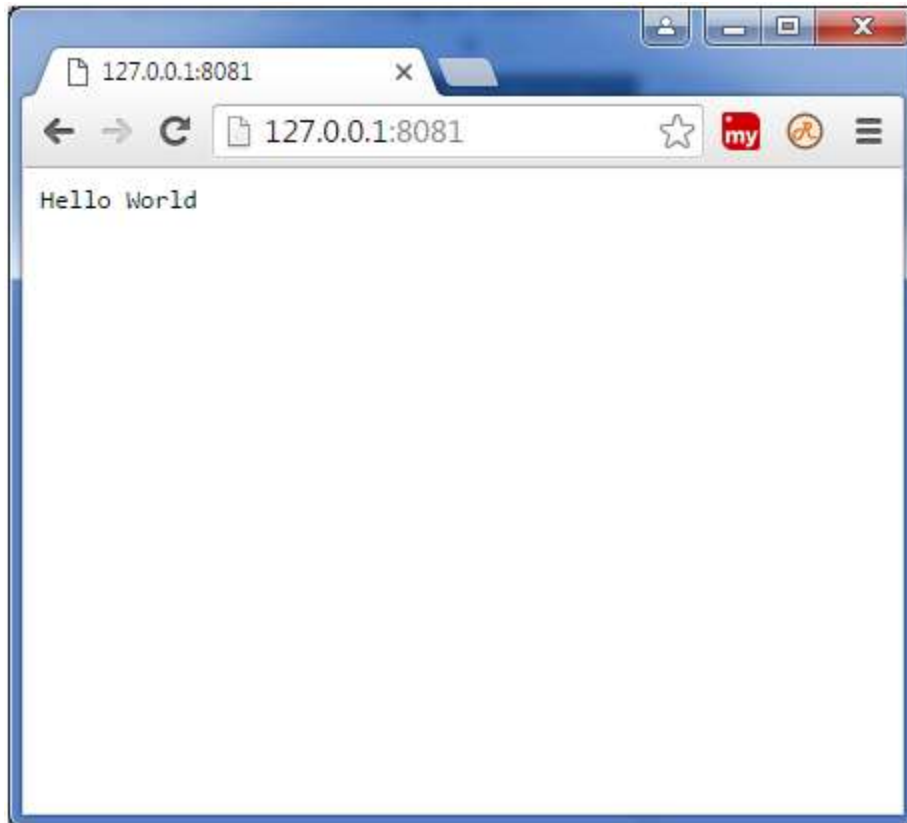
A screenshot of a Windows command prompt window titled "Node.js command prompt - node: main.js". The window has a blue title bar with standard minimize, maximize, and close buttons. The text inside the window shows the sequence of commands and output: "Your environment has been set up for using Node.js 4.4.2 (x64) and npm.", "C:\Users\javatpoint1>cd desktop", "C:\Users\javatpoint1\Desktop>node main.js", and "Server running at http://127.0.0.1:8081/" followed by a horizontal line.

```
Node.js command prompt - node: main.js
Your environment has been set up for using Node.js 4.4.2 (x64) and npm.
C:\Users\javatpoint1>cd desktop
C:\Users\javatpoint1\Desktop>node main.js
Server running at http://127.0.0.1:8081/
_
```

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