1... Introduction to Node JS

Objectives...

- To learn about Node JS.
- To study Different advantages of Node JS.
- To study about Traditional Web Server Model.
- To learn Node JS Process Model.
- To understand how to install Node JS on Windows.
- To learn Working in REPL.

1.1 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.
- Earlier the internet programming was divided in two phases:
  1) **Client Side Scripting:** Client side scripting is performed to generate a code that can run on the client end (browser) without needing the server side processing. Basically, these types of scripts are placed inside an HTML document.
  2) **Server Side Scripting:** Server side scripting is a technique of programming for producing the code which can run software on the server side. In simple words, any scripting or programming that can run on the web server is known as Server Side Scripting.

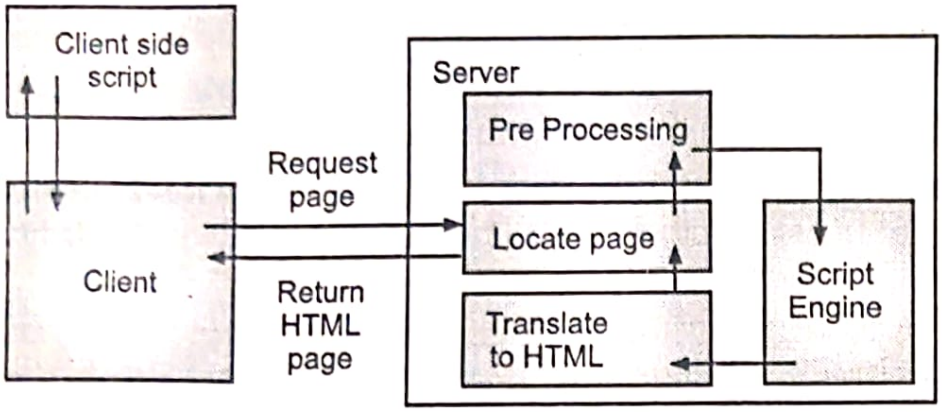


Fig. 1.1: Server Side Scripting

- Difference Between Client Side Scripting and Server Side Scripting:

Table 1.1: Difference between Client Side Scripting and Server Side Scripting

Sr. No.	Client Side Scripting	Server Side Scripting
1.	Web browsers execute client side scripting.	Web servers are used to execute server side scripting.
2.	It is also used for validations and functionality for user events.	They are basically used to create dynamic pages.
3	It cannot be basically used to connect to databases on web server.	It cannot be basically used to connect to databases on web server.
4.	These scripts cannot access file system that resides at web server.	It can also access the file system residing at web server.
5.	It can also used to create "cookies" that store data on user's computer.	Server side environment that runs on a scripting language is a web server.

Web Based Scripting Language:

- Since inception of web development, the client side scripting languages like VB Script, Java Script etc. and server side scripting languages like ASP, JSP, PHP etc. was different. At that time, developer was learning both for client end and server end. Till few years back Java Script was limited to client side scripting only.
- But in year 2009, an idea came in the mind of Ryan Dahl, a Google Engineer that, why not one language should work at both end client end as well as at server end. So he worked to run JavaScript outside client browser also. Means same JavaScript could be used at client end as well as at server end. So he used tool Chrome V8 engine and embedded in a C++ program and called it Node.exe which later on became Node JS.

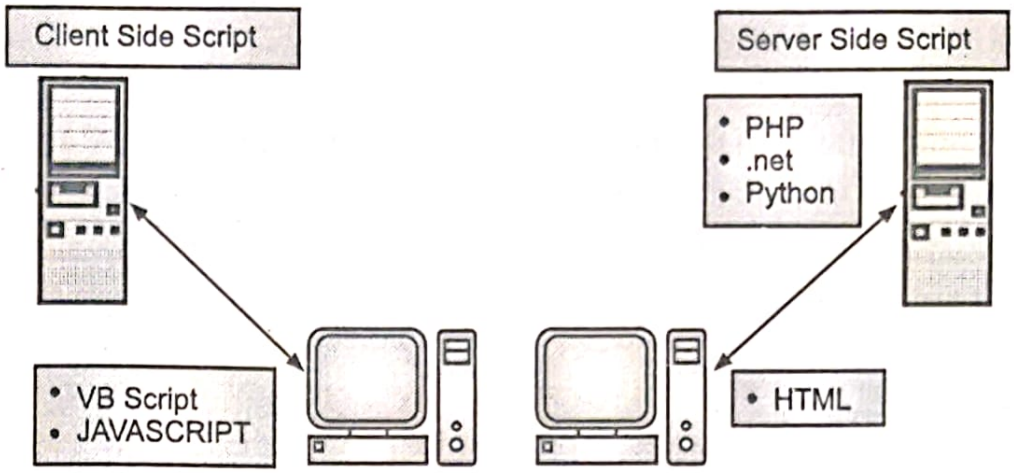


Fig. 1.2: Web Based Scripting Language

- Node JS is also based on the JavaScript framework, but it is used for developing server based applications. Node JS is not a framework and it's not a programming language also.
- We often use Node JS for building back end services like APIs, Web App or Mobile App. It is used in production by large companies such as PayPal, Uber, Netflix, Walmart and so on.

1.2 WHAT IS NODE JS? [S-22]

- A Node JS is a single process application; it does not create a new thread for every request.
- Node JS runs the V8 JavaScript engine, which is the core of Google Chrome browser.
- Node JS is an open source and cross platform JavaScript runtime environment. It is very easy and popular tool for developing any kind of websites.
- Node JS provides a collection 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.
- When Node JS performs an I/O operation, like reading from the network, accessing a database or the file system, instead of blocking the thread and wasting CPU cycles waiting, Node JS will resume the operations when the response comes back.
- This allows Node JS to handle thousands of concurrent connections with a single server without introducing the burden of managing thread concurrency.
- Node JS has a unique advantage because millions of frontend developers that write JavaScript for the browser are now able to write the server side code in addition to the client side code without the need to learn a completely different language.

1.2.1 Difference between Angular and Node JS

Table 1.2: Difference between Angular and Node JS

Sr. No.	Angular	Node JS
1.	It is an open source web application development framework.	It is a cross platform runtime environment for applications.
2.	It is written in TypeScript.	It is written in **C, C++** and JavaScript languages.
3.	Used for building single page client side web applications.	Used for building fast and scalable server side networking applications.
4.	Angular itself is a web application framework.	Node JS has many different frameworks like Sails.js, Partial.js, and Express.js, etc.
5.	Ideal for creating highly active and interactive web apps.	Ideal for developing small size projects.
6.	Helpful in splitting an app into MVC components.	Helpful in generating database queries.
7.	Suitable for developing real-time applications.	Suitable in situations where something faster and more scalable is required.

1.2.2 Difference between JavaScript and Node JS

Table 1.3: Difference between JavaScript and Node JS

Features	JavaScript	Node JS
1. Type	Programming Language.	Interpreter and environment for JavaScript.
2. Utility	Used for any client-side activity for a web application.	Used for accessing or performing any non-blocking operation of any operating system.
3. Running Engine	Spider monkey (Firefox), JavaScript Core (Safari), V8 (Google Chrome), etc.	V8 (Google Chrome)

1.2.3 Difference Between 'Front End' and 'Back End' Development

Table 1.4: Difference between Front End Development and Back End Development

Sr. No.	Front End Development	Back End Development
1.	Uses mark up and web languages like HTML, CSS, and JavaScript.	Uses programming and scripting languages like Python, Ruby, Perl, etc.
2.	Based on asynchronous requests and AJAX.	Based on Server Architecture.
3.	Better Accessibility.	Enhanced Security.
4.	Used for SEO.	Used for Backup.

1.2.4 Features of Node JS [S-23]

1. Node JS is Asynchronous or Non-blocking nature.
2. Node JS is easy to launch and can be used for prototyping and agile development.
3. Node JS provides fast and highly scalable services.
4. It uses JavaScript everywhere so it's easy for a JavaScript programmer to build back end services using Node.js.
5. Source code cleaner, consistent and steady.
6. Large environment for open source library.

1.2.5 Application of Node JS [S-22]

- Node JS is best for usage in streaming or event based real-time applications like:

1. **Online vehicle booking and tracking system:** UBER was one of the first 3 companies that put Node JS into full production. They required an extremely fast and scalable cross platform technological solution that could handle an enormous amount of notifications and requests.
2. **Chat applications.**
3. **Game servers:** Fast and high performance servers that need to processes thousands of requests at a time, then this is an ideal framework.
4. **Social media:** It is virtually impossible without mobile use. According to Statista portal, LinkedIn had an average of 63M unique members using their mobile applications, which accounted for 59% of all unique members during the same 2016.
5. **Good for Collaborative Environment:** This is good for environments which manage document. In document management environment you will have

multiple people who post their documents and do constant changes by checking out and checking in documents.

6. **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.
7. **Streaming Servers:** Another ideal scenario to use Node is for multimedia streaming servers wherein clients have request's 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.

1.2.6 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.
  - **Yahoo, 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.

1.3 ADVANTAGES OF NODE JS [W-22, S-23]

1. **Fast processing and event based model:** V8 engine used in Node JS implementation was originally developed for the Chrome browser which is written in C++. Chrome's V8 is used to compile functions written in JavaScript into machine code, and it does the job at an impressive speed. Non-blocking input output and asynchronous request handling made Node JS capable of processing request without any delay. The implementation of event based model is easy common language for both server side as well as client side. Using common language synchronization happens fast which helps time applications and event based applications.
2. **Scalability:** Developers prefer to use Node JS because it is easily scales the application in both horizontal and vertical direction.
3. **Real time web apps:** Node JS is much more convenient for chat apps or gaming apps because of faster synchronization. Also, event loop avoids HTTP overload for Node JS development.

4. **Advantage of Caching:** It provides the caching of single module. Whenever there is any request for the first module, it gets cached in the application memory so you don't need to re-execute the code.
5. **Easy to learn and code:** Node JS is easy to learn and code because it uses JavaScript. If you are a front end developer and have a good grasp on JavaScript you can easily learn and build the application on Node JS.
6. **Hosting:** PaaS (Platform as a Service) and Heroku are the hosting platform for Node JS application deployment which is easy to use without facing any issue.
7. **Data Stream:** In Node JS HTTP request and response are considered as two separate events. They are data stream so when you process a file at the time of loading it will reduce the overall time and will make it faster when the data is presented in the form of transmissions. It also allows you to stream audio and video files at lightning speed.
8. **Corporate Support:** Most of the well known companies like Walmart, PayPal, Microsoft, Yahoo are using Node JS for building the applications. Node JS uses JavaScript so most of the companies are combining front end and back end teams together into a single unit.

1.4 TRADITIONAL WEB SERVER MODEL [W-22, S-23]

- In the traditional web server model, each request is handled by a dedicated thread from the thread pool.
- If no thread is available in the thread pool at any point of time then the request waits till the next available thread.
- Dedicated thread executes a particular request and does not return to thread pool until it completes the execution and returns a response.

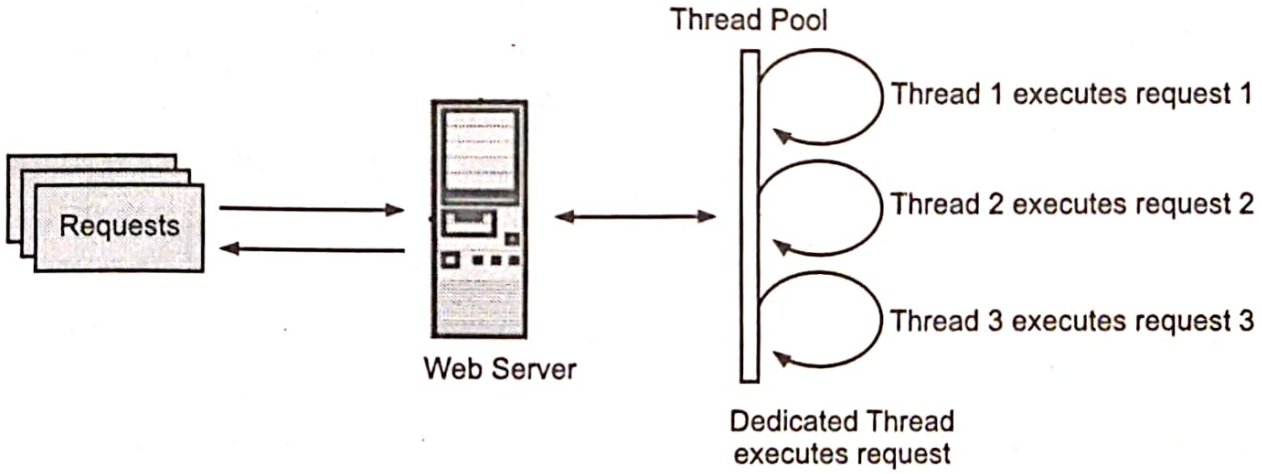


Fig. 1.3: Traditional Web Server Model

1.5 NODE JS PROCESS MODEL [S-22, W-22, S-23]

- Node JS processes user requests differently as compared to a traditional web server model.
- Node JS runs in a single process and the application code runs in a single thread and thereby needs less resource than other platforms.
- All the user requests to your web application will be handled by a single thread and all the I/O work or long running job is performed asynchronously for a particular request. So, this single thread doesn't need to wait for the request to complete, it is free to handle the next request.
- When asynchronous I/O work completes then it processes the request further and sends the response.
- An event loop is constantly watching for the events to be raised for an asynchronous job and executing callback function when the job completes. Internally, Node JS uses **libev** for the event loop which in turn uses internal C++ thread pool to provide asynchronous I/O.
- The following figure illustrates asynchronous web server model using Node.js.

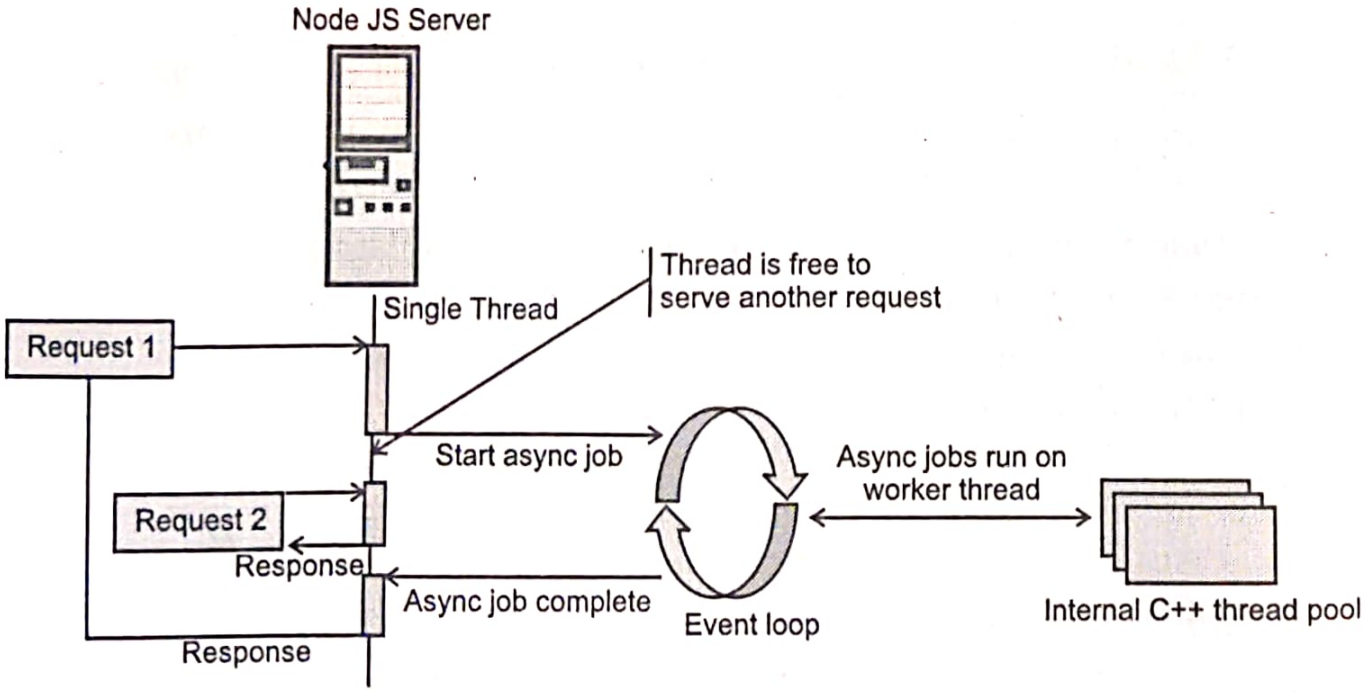


Fig. 1.4: Process Model using Node.js

- Node JS process model increases the performance and scalability with a few caveats.
- Node JS is not fit for an application which performs CPU intensive operations like image processing or other heavy computation work because it takes time to process a request and thereby blocks the single thread.

- Node JS comes with a built-in library that allows applications to act as a web server. Node JS event driven architecture and a non-blocking I/O API optimizes an application's throughput, Node JS excels when it comes to real-time communication.

Working of Node JS:

- Node JS is a virtual machine that uses JavaScript as its scripting language and runs on a v8 environment. It works on a single threaded event loop and a non-blocking I/O which provides high rate as it can handle a higher number of concurrent requests. Also, by making use of the 'HTTP' module, Node JS can run on any stand-alone web server.

Node JS is Single Threaded:

- Node JS uses a single threaded model in order to support asynchronous processing. With asynchronous processing, an application can perform better and is more scalable under web traffic. Thus, Node JS makes use of a single threaded model approach rather than typical thread based implementation.

1.5.1 Comparison between Request Handling in Node JS and a Traditional Server

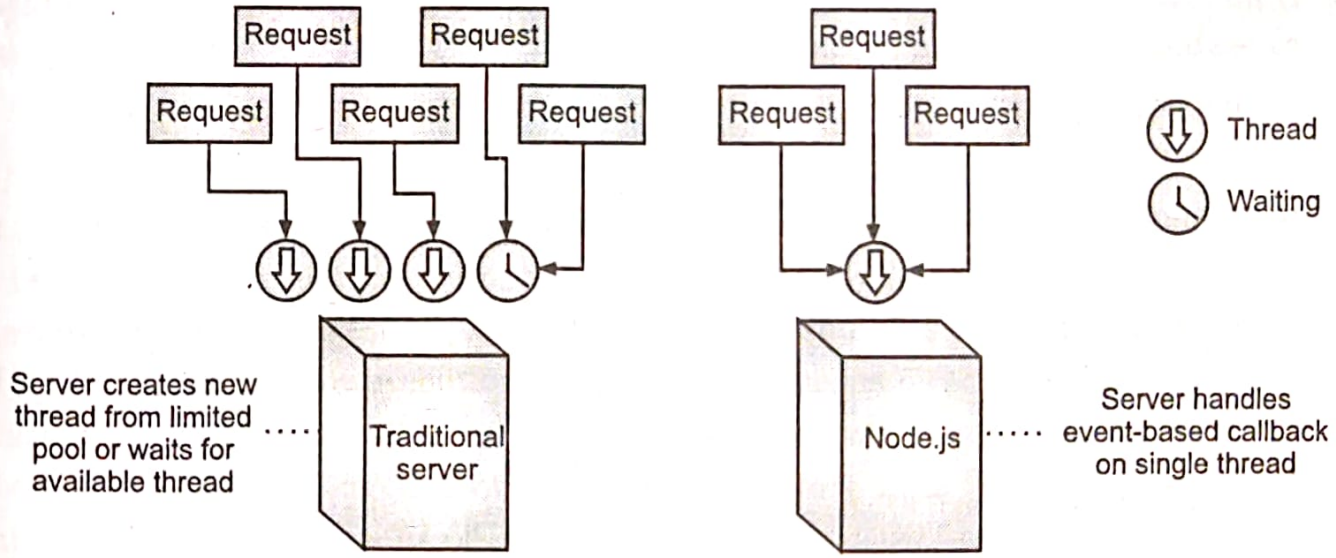


Fig. 1.5: Comparison between request handling in Node JS and a Traditional server

- Compared to traditional web-serving techniques each connection (request) spawns a new thread, taking up system RAM and eventually maxing-out at the amount of RAM available. Node.js operates on a single-thread, using non-blocking I/O calls, allowing it to support tens of thousands of concurrent connections held in the event loop.
- A quick calculation: assuming that each thread potentially has an accompanying 2 MB of memory with it, running on a system with 8 GB of RAM puts us at a theoretical

maximum of 4000 concurrent connections, plus the cost of context switching between threads. That is the scenario you typically deal with in traditional web server techniques. By avoiding all that, Node JS achieves scalability levels of over 1M concurrent connections.

1.6 BLOCKING AND NON-BLOCKING APPROCH

Blocking:

- It is also called as Synchronous Approach.
- An example of blocking is how some web servers like ones in Java or PHP handle requests. If your code does something blocking, like reading something from the database, your code "stalls" at that line and waits for the operation to finish. In that period, your machine is holding onto memory and processing time for a thread that is not doing anything. In order to cater other requests while that thread has stalled depends on your setup.
- Your server can spawn more threads to cater the request or, if you have a load balancing setup, forwards requests to the next available instance. This instills more setup, more memory consumed, more processing.
- Another Example, Imagine the situation of a restaurant. The waiter will take order from table1 give it to the kitchen and will wait until the chef prepares the food. This is called **blocking or synchronous nature.**

Non-blocking:

- It is also called as Asynchronous Approach.
- In contrast, non-blocking servers like ones made in Node JS, only use one thread to service all requests. This might sound counter-intuitive, but the creators designed it with the idea that the I/O is the bottleneck i.e. not computations. When requests arrive at the server, they are serviced one at a time. When the code serviced needs to query the DB for example, it sends off a request to the DB. However, instead of waiting for the response and stall, it sends the callback to a second queue and the code continues running. Now when the DB returns data, the callback gets queued in a third queue where they are pending execution. When the engine is doing nothing (stack empty), it picks up a callback from the third queue and executes it.
- Another Example, in contrast the waiter takes order from table1 and gives the order in the kitchen, now the waiter can serve another table let's say table2 and get their order. This is **asynchronous system**, as the chef is preparing the food, meanwhile a single waiter is serving different tables (taking orders and serving food). The waiter doesn't have to wait for the chef to cook the meal before he has to go to another table. This is what we call a **non-blocking** or **asynchronous** architecture.

1.7 FRAMEWORKS AND TOOLS OF NODE JS

- Node JS is a low level platform. To make things easy, exciting and simple for developers, thousands of libraries were built upon Node JS by the community.
- A Node JS framework is just some abstract design, built out of Node JS that represents the control flow of the given framework's design. So it is almost like the skeleton of a program.
- Many of those established over time as popular options. Here is a non comprehensive list of the ones worth learning:
  - **Meteor:** An incredibly powerful full stack framework, powering you with an isomorphic approach to building apps with JavaScript, sharing code on the client and the server.
  - **AdonisJs:** A full stack framework highly focused on developer ergonomics, stability, and confidence. Adonis is one of the fastest Node JS web frameworks.
  - **hapi:** A rich framework for building applications and services that enables developers to focus on writing reusable application logic instead of spending time building infrastructure.
  - **Fastify:** A web framework highly focused on providing the best developer experience with the least overhead and powerful plugin architecture. Fastify is one of the fastest Node JS web frameworks.
  - **Gatsby:** A React based, GraphQL powered, static site generator with a very rich ecosystem of plugins and starters.
  - **Micro:** It provides a very lightweight server to create asynchronous HTTP micro services.
  - **Loopback.io:** Makes it easy to build modern applications that require complex integrations.
  - **Nest.JS:** A Type Script based progressive Node JS framework for building enterprise grade efficient, reliable and scalable server side applications.
  - **Nx:** A toolkit for full stack monorepo development using NestJS, Express, React, Angular, and more. Nx helps scale your development from one team building one application to many teams collaborating on multiple applications.
  - **Sapper:** Sapper is a framework for building web applications of all sizes, with a beautiful development experience and flexible file system based routing. Offers SSR and more.
  - **Socket.io:** A real-time communication engine to build network applications.

- **Express:** It provides one of the simplest yet powerful ways to create a web server. Its minimalist approach, unopinionated, focused on the core features of a server, is the key to its success.
- **Next.js:** A framework to render server side rendered React applications.
- **Strapi:** Strapi is a flexible, open source Headless CMS that gives developers the freedom to choose their favorite tools and frameworks while also allowing editors to easily manage and distribute their content. By making the admin panel and API extensible through a plugin system, Strapi enables the world's largest companies to accelerate content delivery while building beautiful digital experiences.

1.8 INSTALL NODE JS ON WINDOWS [W-22, S-23]

- 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?

- 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.

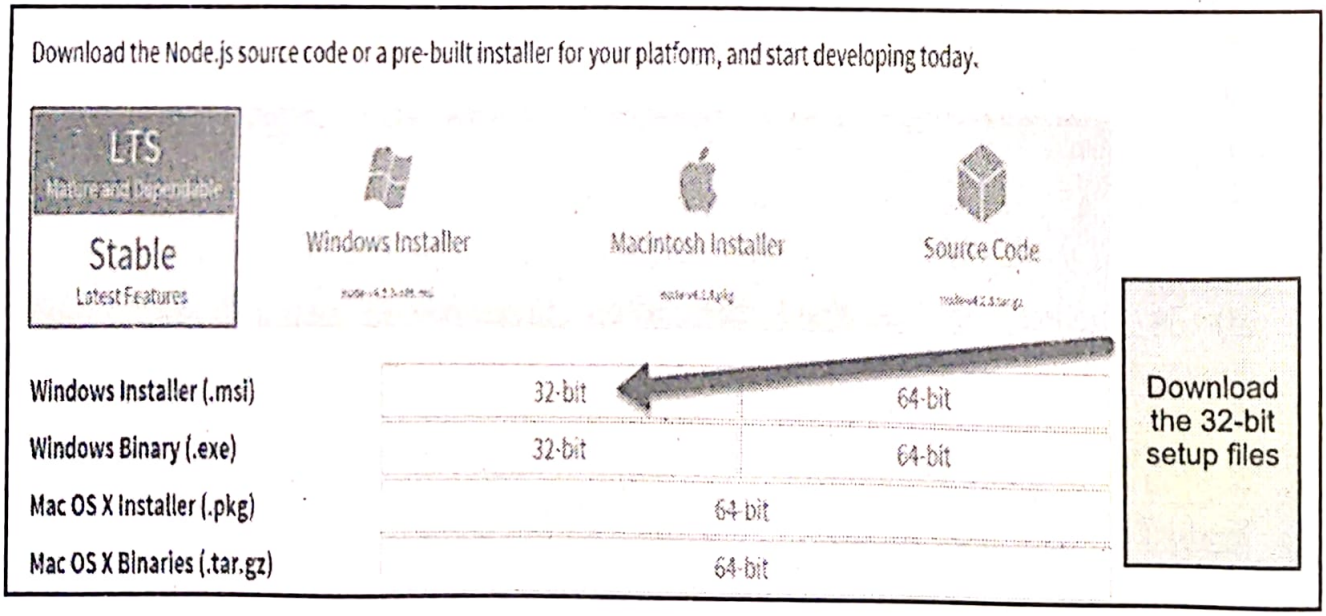


Fig. 1.6

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.

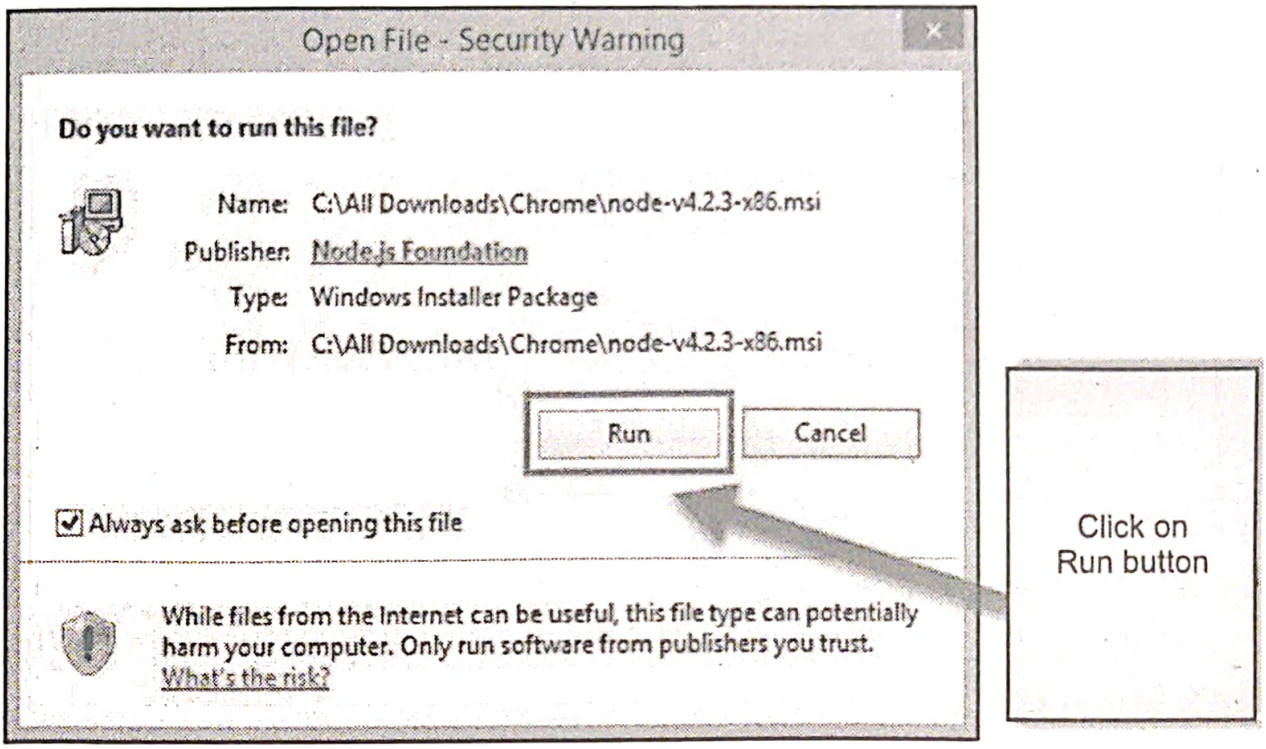


Fig. 1.7

Step 3: In the next screen, click the "Next" button to continue with the installation.

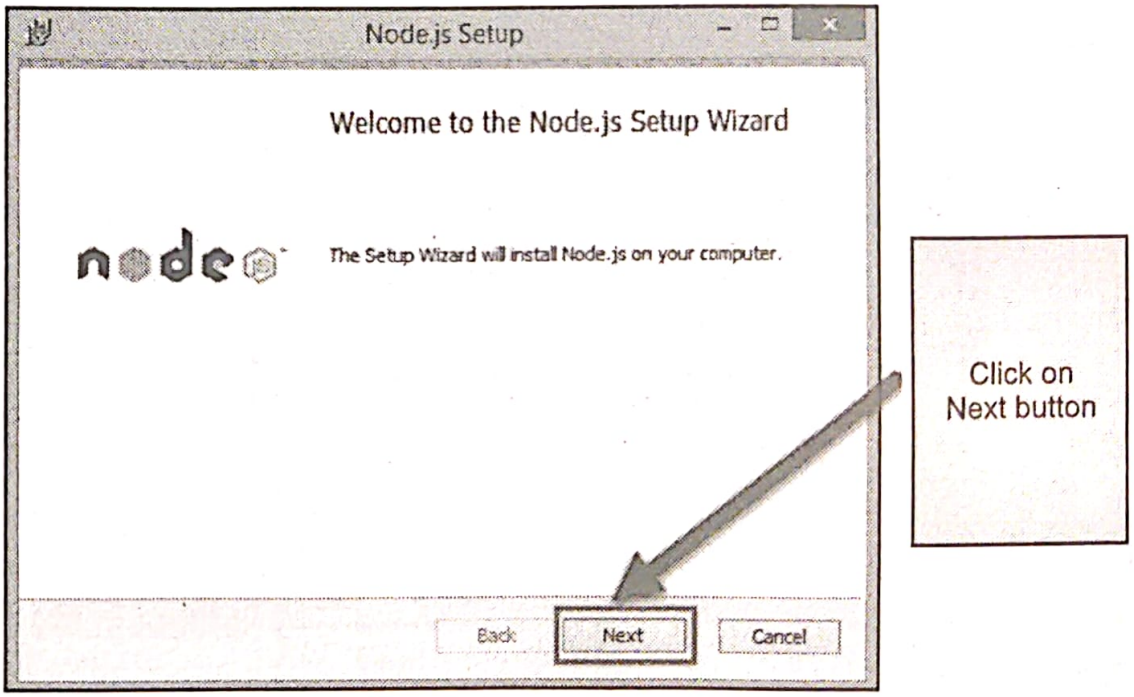


Fig. 1.8

Step 4: In the next screen, accept the license agreement and click on the Next button.

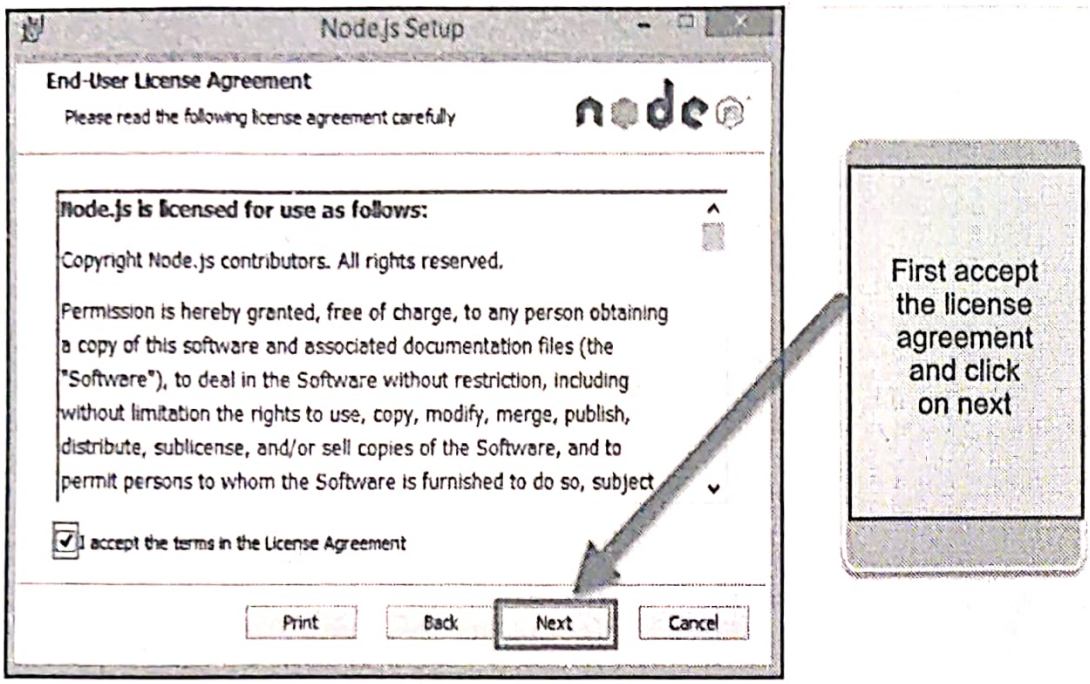


Fig. 1.9

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.

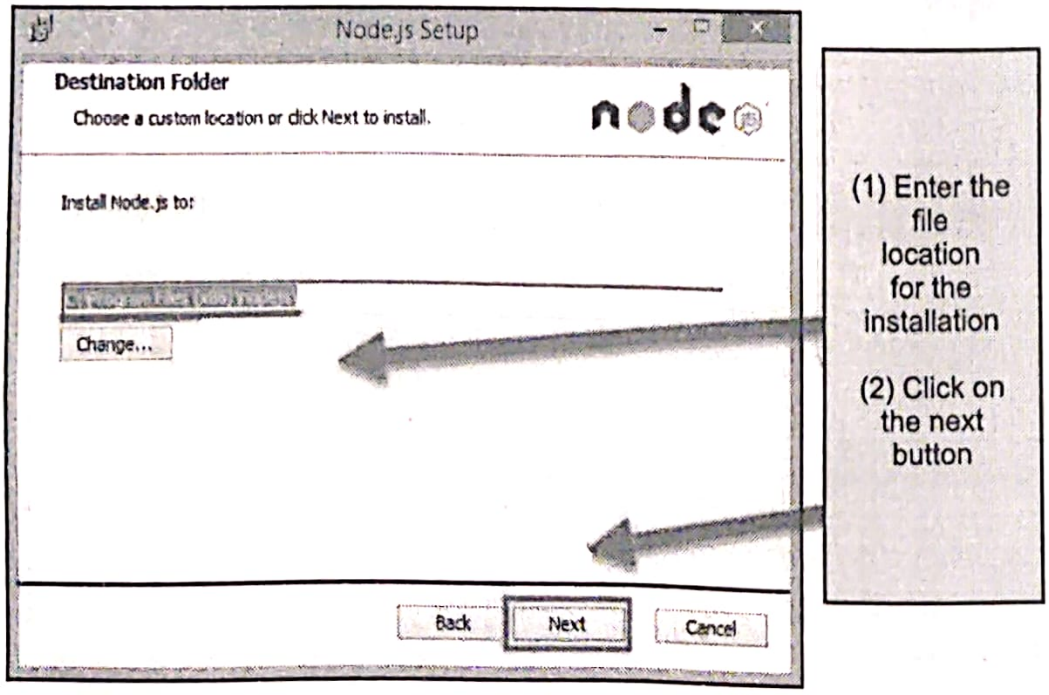


Fig. 1.10

Step 6: Accept the default components and click on the next button.

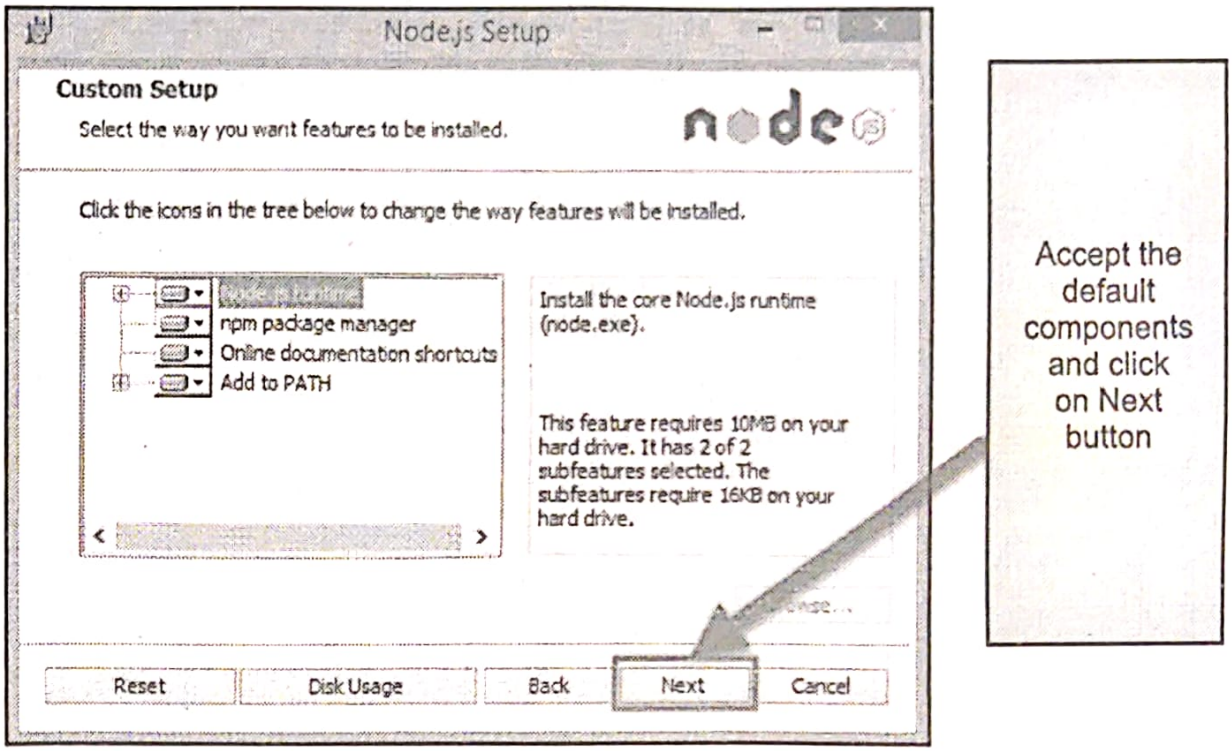


Fig. 1.11

Step 7: In the next screen, click the Install button to start the installation.

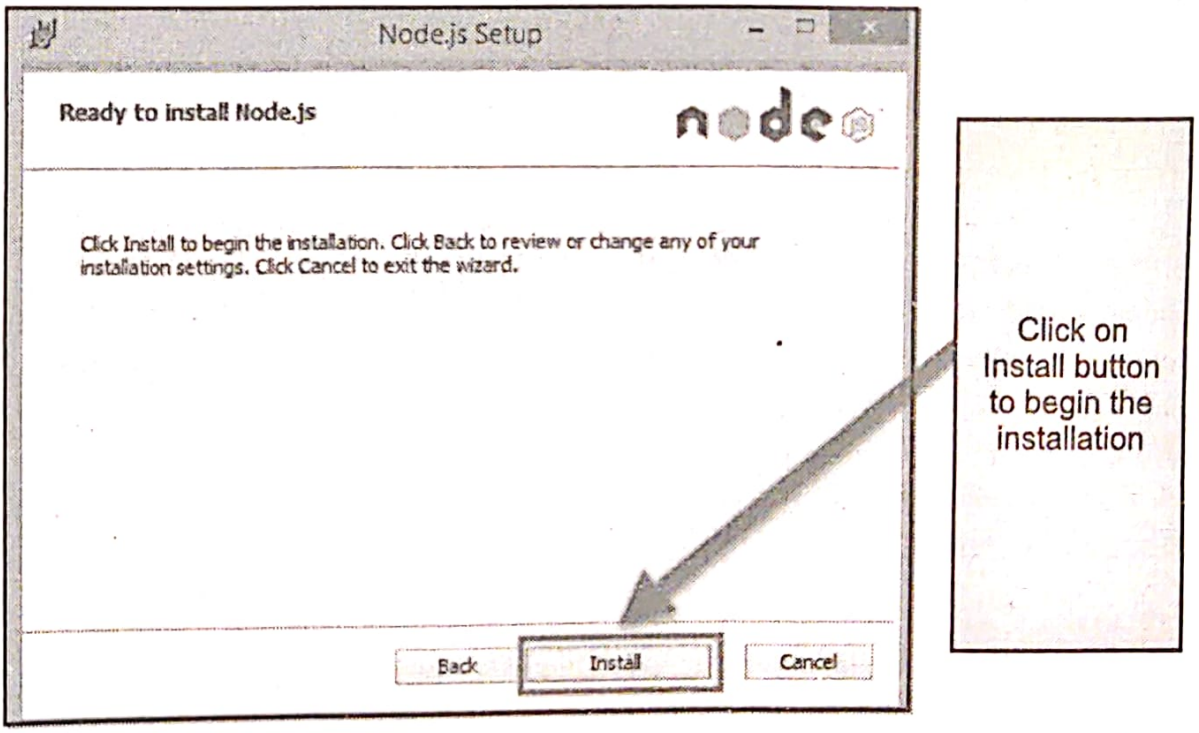


Fig. 1.12

Step 8: Click the Finish button to complete the installation.

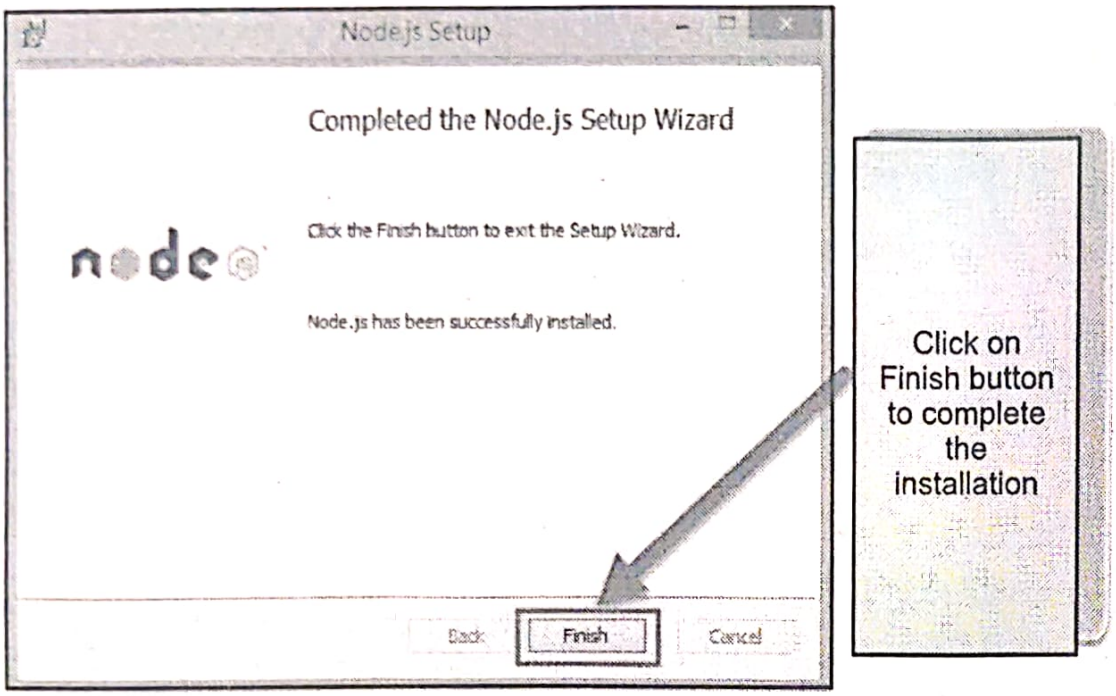


Fig. 1.13

1.8.1 Installing Node through a Package Manager

- The other way to install Node JS on any client machine is to use a "package manager".
- On windows, the node package manager is known as Chocolatey. It was designed to be a decentralized framework for quickly installing applications and tools that you need.
- To install Node JS via Chocolatey, the following steps need to be performed.

Step 1: Installing Chocolatey – The Chocolatey website (`https://chocolatey.org`)

- The first steps is to run the below command in the command prompt windows. This command is taken from the Chocolatey web site and is the standard command for installing Node JS via Chocolatey.

```
@powershell -NoProfile -ExecutionPolicy Bypass -Command "iex ((new-
objectwet.webclient).DownloadString('https://chocolatey.org/install.ps1'
))" && SET PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin
```

- This command is a PowerShell command which calls the remote PowerShell script on the Chocolatey website. This command needs to be run in a PowerShell command window.
- This PowerShell script does all the necessary work of downloading the required components and installing them accordingly.

Step 2: The next step is to install Node JS to your local machine using the Chocolatey, package manager.

- This can be done by running the below command in the command prompt.

```
cinst nodejs install
```

- If the installation is successful, you will get the message of the successful installation of Node.js.
- Note: If you get an error like "C:\ProgramData\chocolatey\lib\libreoffice\tools\chocolateyInstall.ps1" Then manually create the folder in the path.

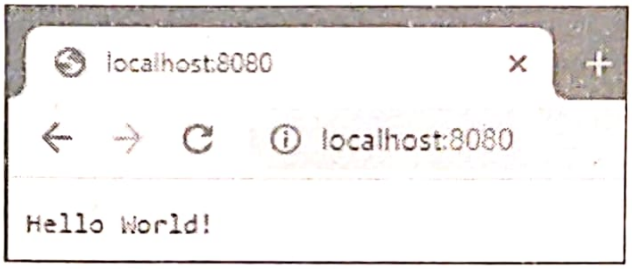
1.9 RUNNING FIRST "HELLO WORLD" APPLICATION IN NODE JS [S-23]

- Once you have downloaded and installed Node JS on your computer, let's try to display "Hello World" in a web browser.
- Create file Node JS with file name `app.js`
- The most common example "Hello World" of Node JS is a web server.

Program 1.1: Simple "Hello World" program.

```
const http = require('http')
const hostname = '127.0.0.1'
const port = process.env.PORT
const server = http.createServer((req, res) => {
 res.statusCode = 200
 res.setHeader('Content-Type', 'text/plain')
 res.end('Hello World!\n')
})
server.listen(port, hostname, () => {
 console.log(`Server running at http://${hostname}:${port}/`)
})
```

Output:



Explanation:

- The basic functionality of the "require" function is that it reads a JavaScript file, executes the file, and then proceeds to return an object. Using this object, one can then use the various functionalities available in the module called by the require function. So in our case, since we want to use the functionality of http and we are using the require(http) command.

- The createServer() method of http creates a new HTTP server and returns it. The server is set to listen on the specified port and host name. When the server is ready, the callback function is called, in this case informing us that the server is running.
- Whenever a new request is received, the request event is called, providing two objects: a request (an `http.IncomingMessage` object) and a response (an `http.ServerResponse` object).
- These 2 objects are essential to handle the HTTP call. The first provides the request details. In this simple example, this is not used, but you could access the request headers and request data. The second is used to return data to the caller.
- In this case with: `res.statusCode = 200`
- We set the `statusCode` property to 200, to indicate a successful response.
- We set the Content-Type header: `res.setHeader('Content-Type', 'text/plain')` and we close the response, adding the content as an argument to end( ): `res.end('Hello World\n')`

Executing the code:

Step 1: Save the file on your computer: `C:\Users\Your Name\ app.js`

Step 2: In the command prompt, navigate to the folder where the file is stored. Enter the command: `Node app.js`

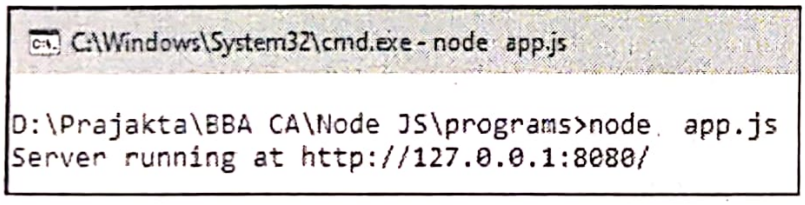


Fig. 1.14

Step 3: Now, your computer works as a server! If anyone tries to access your computer on port 8080, they will get a "Hello World!" message in return!

Step 4: Start your internet browser, and type in the address: `http://localhost:8080`

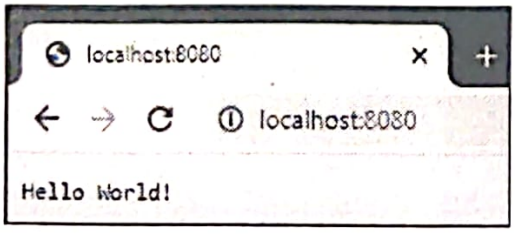


Fig. 1.16

1.10 WORKING IN REPL [S-22, W-22, S-23]

- The REPL feature of Node is very useful in experimenting with Node JS code and to debug JavaScript code. It is a quick and easy way to test simple Node.js/JavaScript code.

- REPL stands for Read Eval Print Loop and it represents a runtime computing environment.
- REPL is like a Windows console or Unix/Linux shell where a command is entered and the system responds with an output in an interactive mode.
- To launch the REPL (Node shell), open command prompt (in Windows) or terminal (in Mac or UNIX/Linux) and type node as shown below. It will change the prompt to > in Windows and MAC.

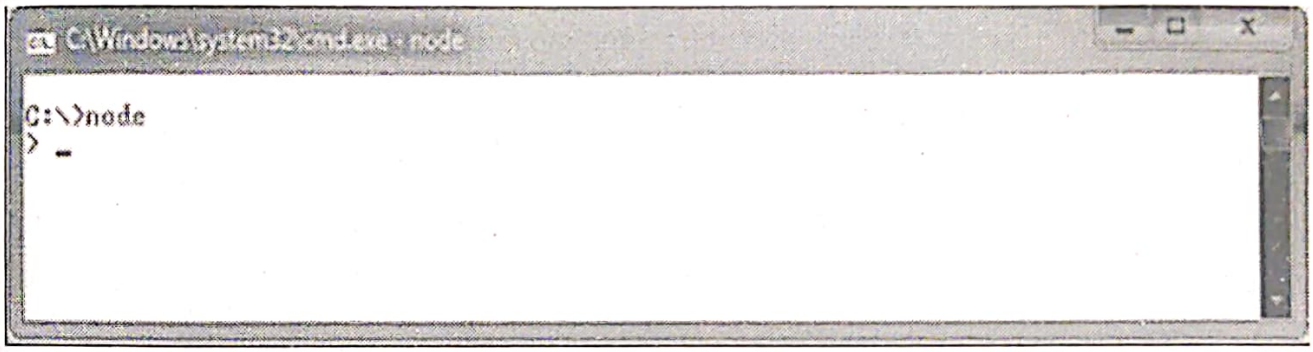


Fig. 1.17: Launch Node JS REPL

- Node JS or Node comes bundled with a REPL environment. It performs the following tasks:
  - **Read:** It is used to reads user's input, parses the input into JavaScript data structure, and stores in memory.
  - **Eval:** It is used to takes and evaluates the data structure.
  - **Print:** It is used to prints the result.
  - **Loop:** Loops the above command until the user presses `ctrl-c` twice.

Simple Expression:

- Let's try a simple mathematics at the Node JS REPL command prompt:

```
> 2 + 7
9
> 2 + (1 * 5) - 4
3
>
```

- Along with adding two integers the + operator is also used to concatenates two strings as in browser's JavaScript.

```
> "Corona" + "Virus"
Corona Virus
```

Variables:

- We can make use variables to store values and print later like any simple program.
- To store value we use **var** keyword.
- To simply display we don't use **var** keyword.

- If **var** keyword is not used, then the value is stored in the variable and printed. Whereas if **var** keyword is used, then the value is stored but not printed. You can print variables using **console.log()**.

```
> a = 100
100
> var b = 200
undefined
> a + b
300
> console.log("Hello World")
Hello World
```

Multiline Expression:

- Node REPL supports multiline expression similar to JavaScript. Following is the example of do-while loop .

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

- Comes automatically when you press Enter after the opening bracket. Node automatically checks the continuity of expressions.

Underscore Variable:

- You can use underscore (_) to get the last result:

```
node
> var p = 100
undefined
> var q = 200
undefined
> p + q
300
> var sum = _
undefined
> console.log(sum)
300
undefined
>
```

REPL Commands:

- **.help**: List of all commands.
- **.break**: Exit from multiline expression.
- **.clear**: Exit from multiline expression.
- **.save filename**: Save the current Node REPL session to a file.
- **.load filename**: Load file content in current Node REPL session.
- **ctrl + c:** Terminate the current command.
- **ctrl + c twice:** Terminate the Node REPL.
- **ctrl + d:** Terminate the Node REPL.
- **Up/Down Keys**: See command history and modify previous commands.
- **tab Keys:** List of current commands.

Quitting REPL:

- As mentioned above, you will need to use `ctrl-c` twice to come out of Node JS REPL.

```
node
>
(^C again to quit)
>
```

Summary

- Over the years, most of the applications were based on a stateless request-response framework. In these sorts 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.
- 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.

Check Your Understanding

1. How do you kill a process in Node.js?
   (a) CTRL+A (b) CTRL+K
   (c) CTRL+B (d) CTRL+C
2. Node uses _____ engine in core.
   (a) Chrome V8 (b) Microsoft Chakra
   (c) Node EN (d) Internet Explorer Engine
3. How many Node object methods are available?
   (a) 16 (b) 17
   (c) 18 (d) 19
4. What is the default scope in Node JS application?
   (a) Global to function (b) Local to object
   (c) Local (d) Global
5. Which extension is used to save Node JS files?
   (a) .java (b) .node.
   (c) .txt (d) .js

6. REPL stands for_____.
   (a) Read Eval Print Loop (b) Research Eval Program Learn
   (c) Read Earn Point Learn (d) Read Eval Point Loop
7. Node JS is _____ Language.
   (a) Server side (b) Client side
   (c) Both (d) None
8. Command to start Node REPL is____.
   (a) $ node repl (b) $ node start
   (c) $ node (d) $ node Console
9. Node JS is_____.
   (a) Synchronuos (b) Asynchronous

ANSWER KEY

1. (d)	2. (a)	3. (c)	4. (c)	5. (d)
6. (a)	7. (a)	8. (c)	9. (b)	

Practice Questions

Q.I: Answer the following questions in short.

1. What is Node JS?
2. What are the advantages of Node JS?
3. What is Client side scripting?
4. What is Server side scripting?
5. Explain Web server.
6. REPL stands for?
7. What are applications of Node JS?
8. NodeJS is a framework. True or False.

Q.II: Answer the following questions.

1. What is difference between Node JS and Agular JS?
2. What is difference between Node JS and Java Script?
3. What is difference between Client Side Scripting and Server Side Scripting?
4. Explain Node JS Process Model?
5. Write steps to install Node JS on Windows through Dashboard?
6. Write steps to install Node JS through Package Manager?

7. Node JS can be use to develop which Applications?
8. Explain REPL in the context of Node.js.
9. What is difference between front end and back end development?
10. Explain asynchronous and synchronous in Node.js.

Q.III: Define the terms.

1. Client
2. Server
3. Web Server
4. Middleware
5. Blocking
6. Non Blocking

❖❖❖