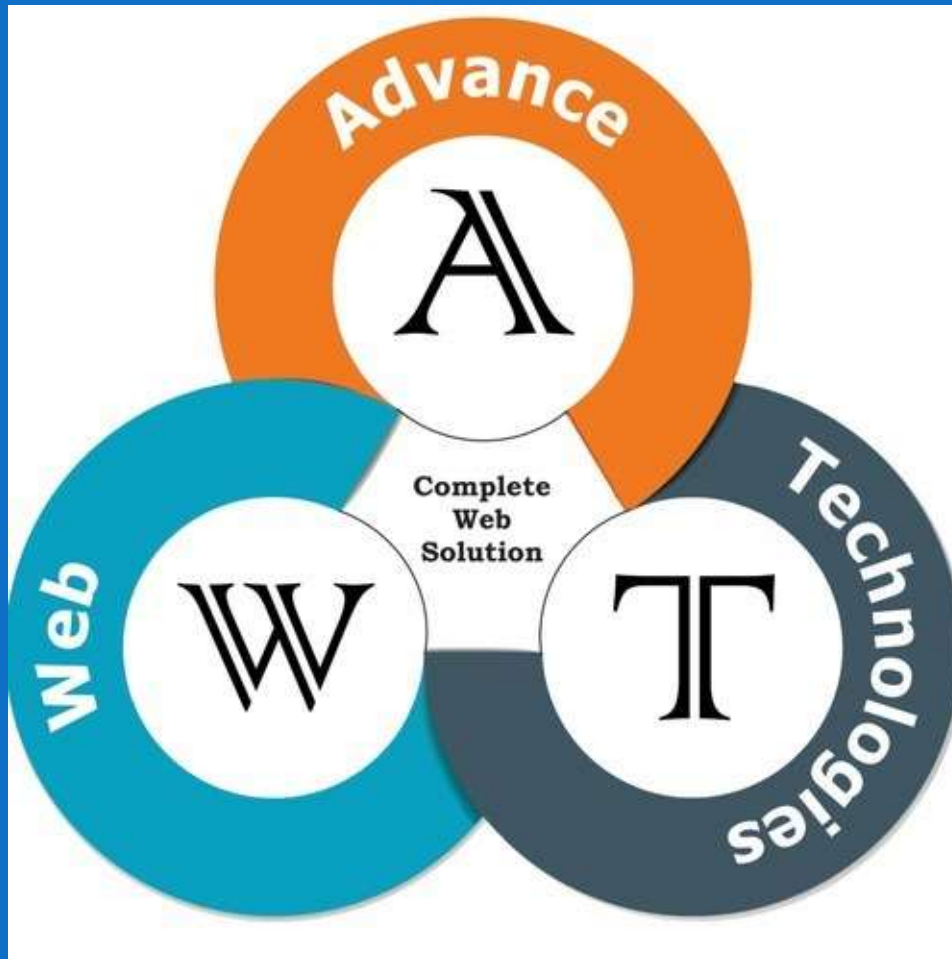


Object
Oriented
PHP

Advance
PHP

Laravel



Node.js



Marwadi
University

Department of
Computer Engineering

Unit 4
Node.js

Subject : 01IT0701 -
**Advance Web
Technology**

Prof. Jaydeep Ratanpara



- Introduction to Node.js
- Node Package Manager
- REPL Terminal
- Node.js Webserver – Server and Clients
- Creating a simple server
- Rendering HTML
- Rendering JSON Data
- Routing

Introduction to Node.js

What is Node.js ?

- Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications.
- It uses Google Chrome's JavaScript V8 Engine to execute code.
- Node.js allows you to run JavaScript on the server.

Introduction to Node.js

- ***What is Node.js ?***
- 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**
- A platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications.
- Lightweight and efficient
 - Perfect for data-intensive real-time applications running on distributed devices.
- Provides a rich library of various JavaScript modules
 - To simplify the web application development

Introduction to Node.js

Why Node.js ? : Node.js uses asynchronous programming!

A common task for a web server can be to open a file on the server and return the content to the client.

Here is how PHP or ASP handles a file request:

- Sends the task to the computer's file system.
- Waits while the file system opens and reads the file.
- Returns the content to the client.
- Ready to handle the next request.

Here is how Node.js handles a file request:

- Sends the task to the computer's file system.
- Ready to handle the next request.
- When the file system has opened and read the file, the server returns the content to the client.

Introduction to Node.js

What Can Node.js Do?

- Node.js can generate dynamic page content
- Node.js can create, open, read, write, delete, and close files on the server
- Node.js can collect form data
- Node.js can add, delete, modify data in your database

Features of Node.js

- **Extremely fast:**
 - Built on Google Chrome's V8 JavaScript Engine so its library is very fast in code execution.
- **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.

Features of Node.js

Single threaded:

- It follows Single threaded model with event looping. (which is responsible for executing the code, collecting and processing events, and executing queued sub-tasks).

No buffering:

- It 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 (fragment of data that is sent to all the servers by the client).

Features of Node.js

Highly Scalable:

- It is highly scalable because event mechanism helps the server to respond in a non-blocking way.

Open source:

- Node.js has an open source community which has produced many excellent modules to add additional capabilities to Node.js applications.

License:

- It is released under the MIT license.

Node Js Basic Usage

Node.js is widely used in the following applications:

- Real-time chats
- Internet of Things
- Complex SPAs (Single-Page Applications)
- Real-time collaboration tools
- Streaming applications
- Microservices architecture

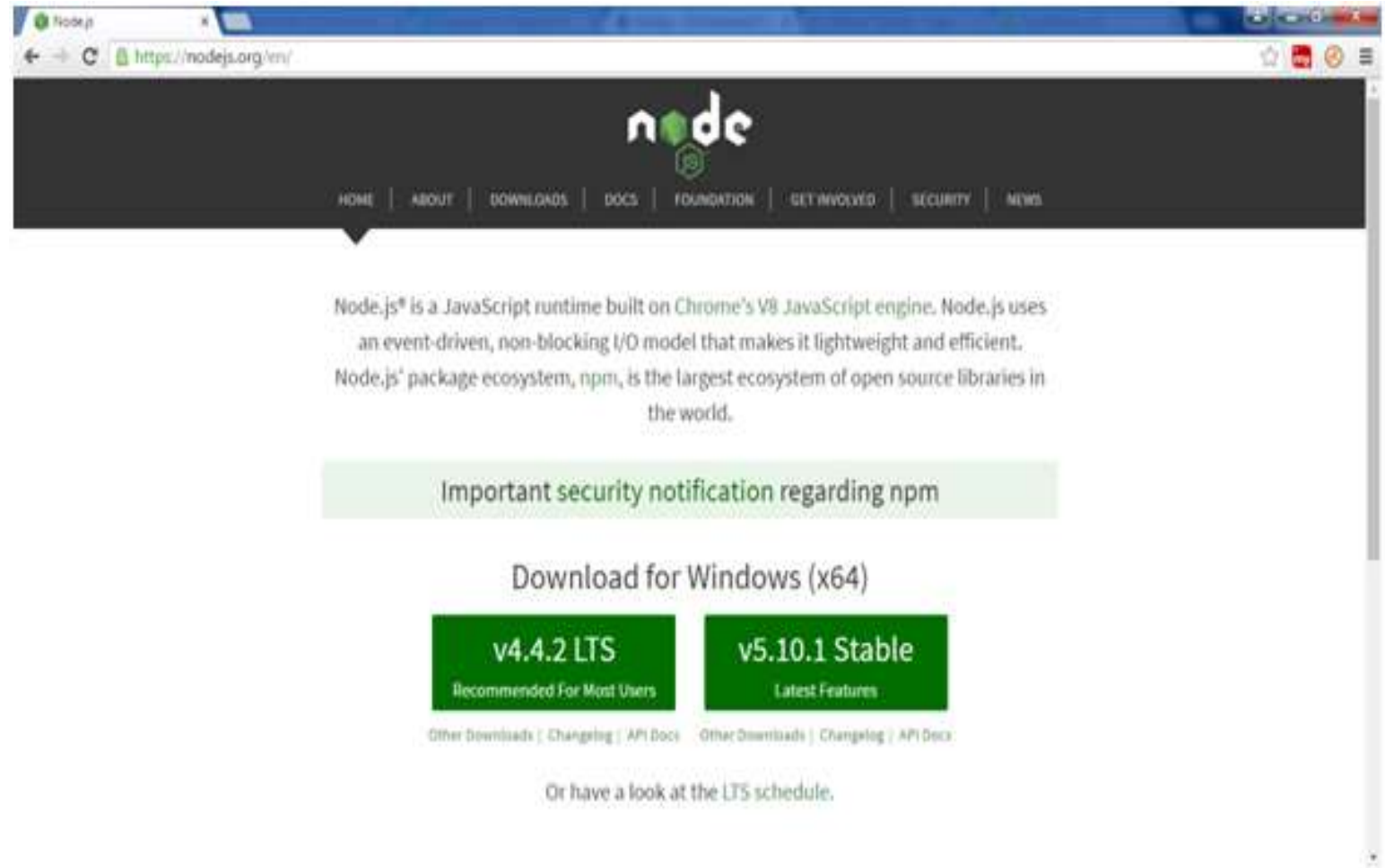
There are lot more applications of Node as it is very powerful backend component with lots of packages.

Node.Js Installation

How to download and install Node.js:

- To install and setup an environment for Node.js, you need the following two softwares available on your computer:
 - Text Editor.
 - Node.js Binary installable
- You can download the latest version of Node.js installable archive file from <https://nodejs.org/en/>

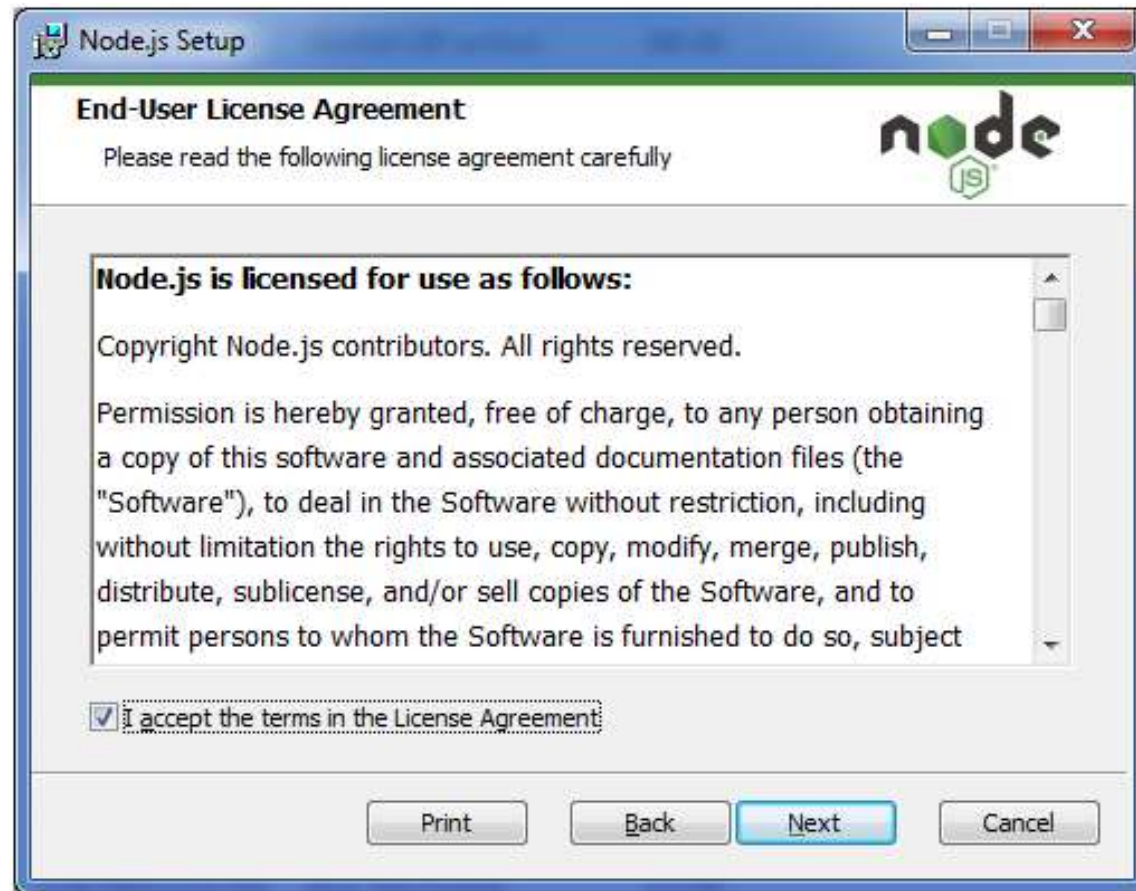
Node.Js Installation



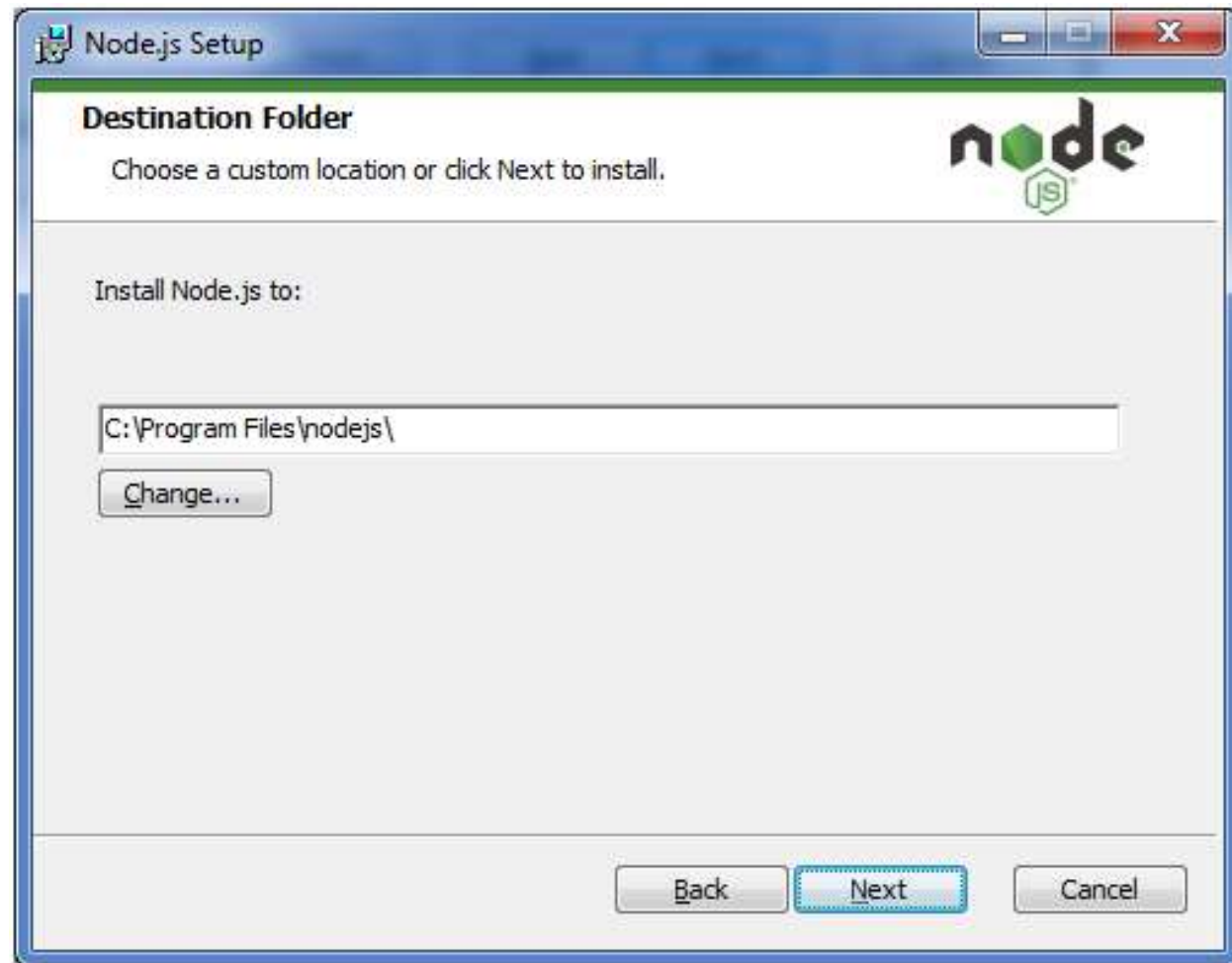
Node.Js Installation



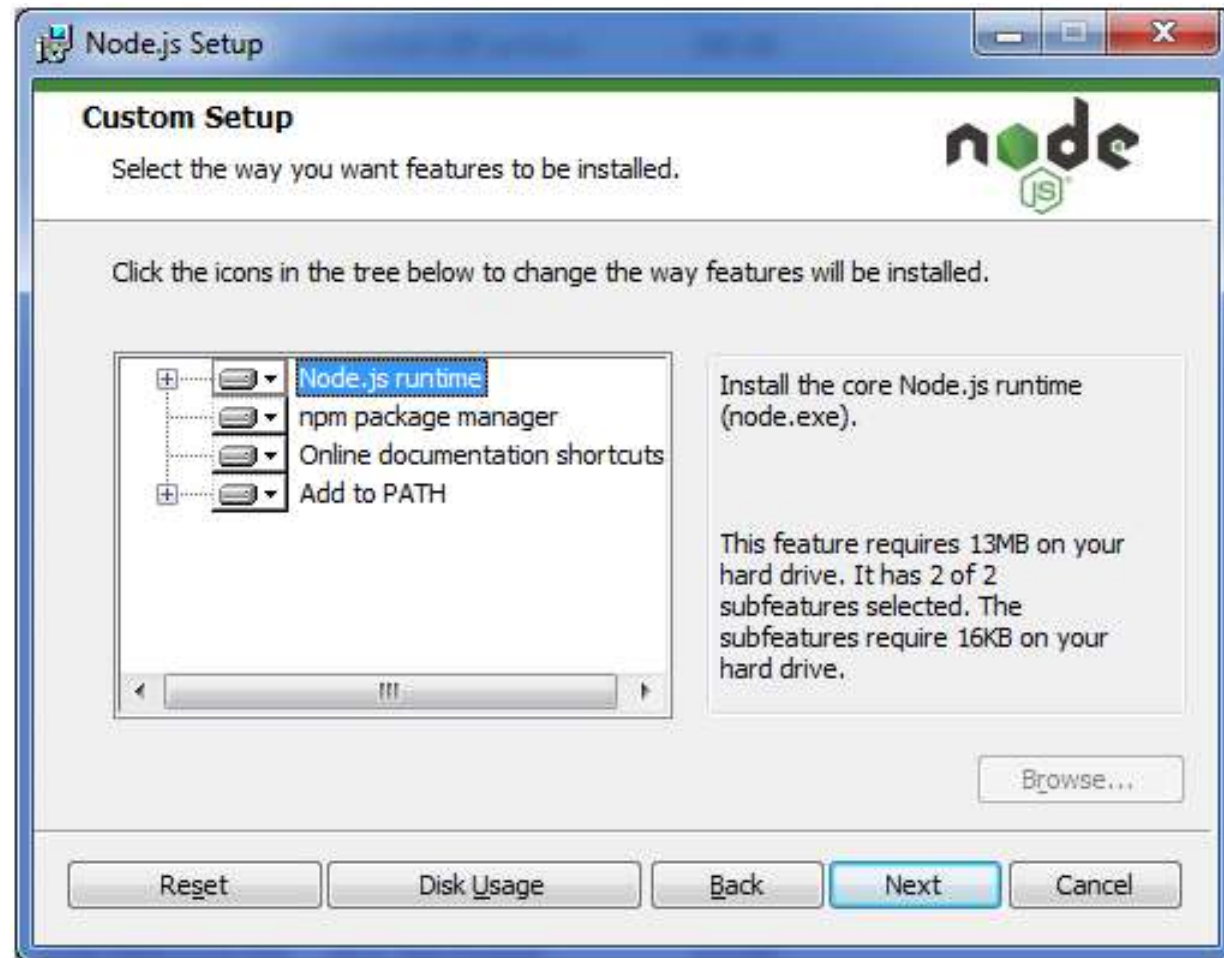
Node.Js Installation



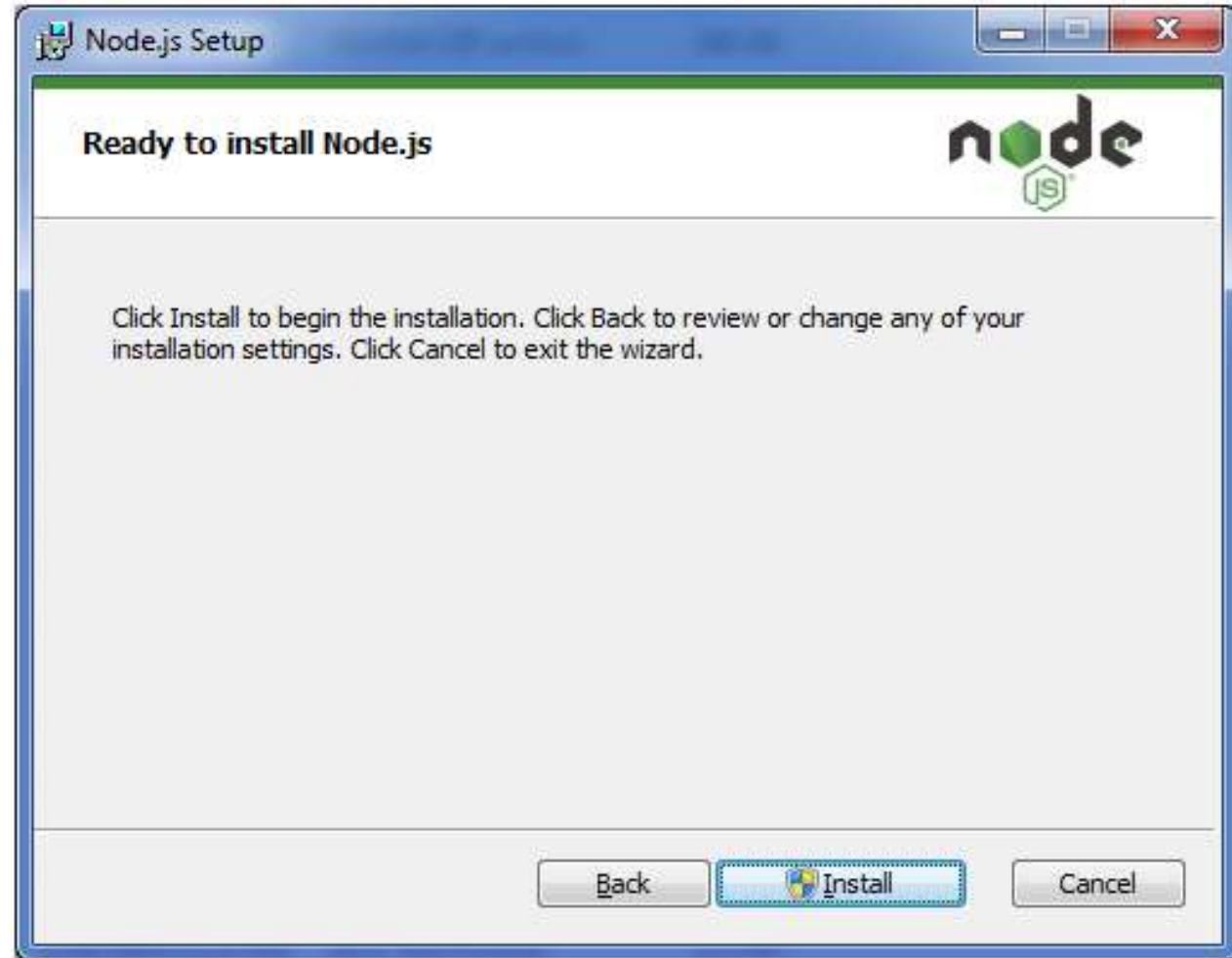
Node.Js Installation



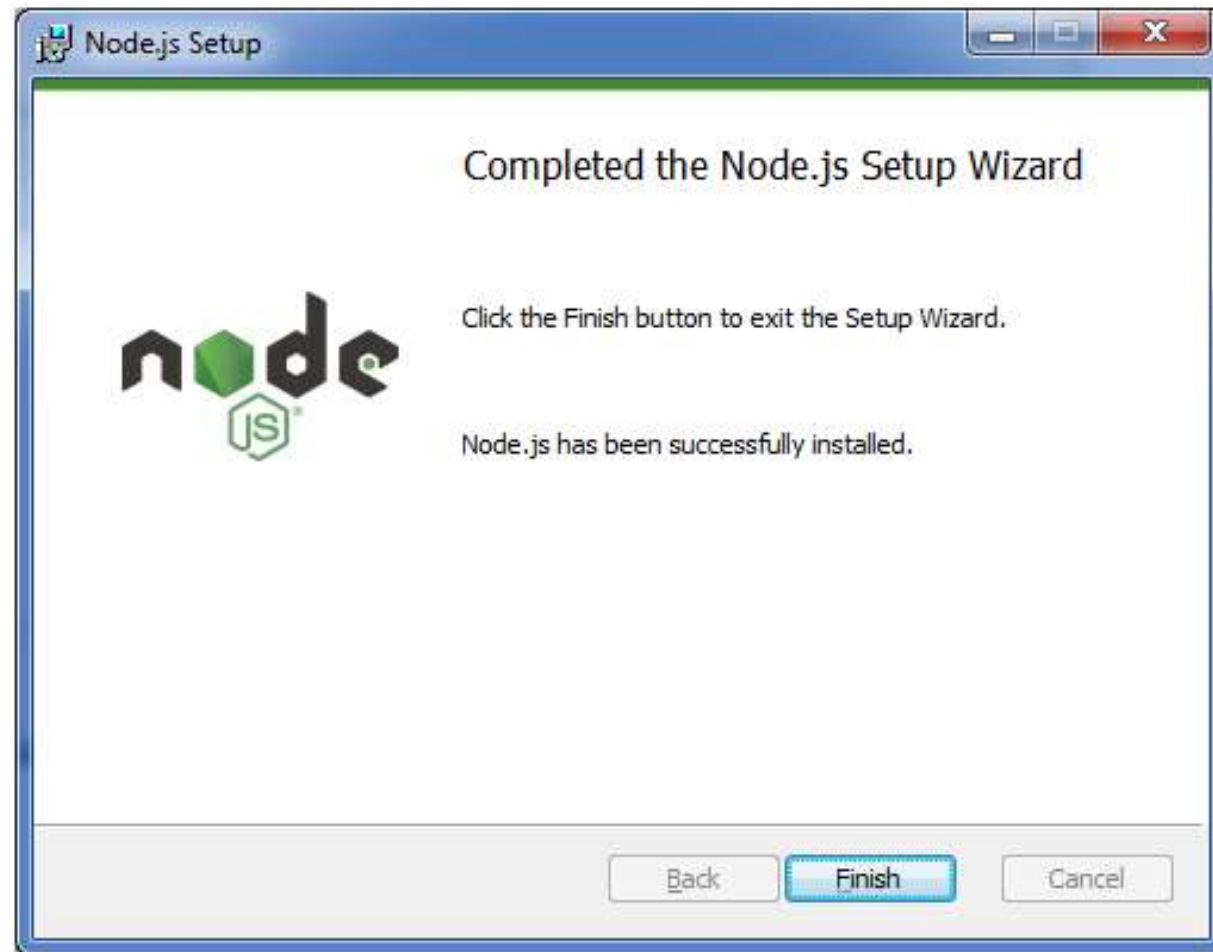
Node.Js Installation



Node.Js Installation



Node.Js Installation



Node.Js Exmample

Node.js Console-based Example

Create .js file and add the following code in it.

```
console.log('Hello World');
```

Save file with the name **Example1.js**

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

```
node Example1.js
```

Node.Js Exmample

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.

Node.Js Exmample

1.Import required modules:

The first step is to use `require` directive to load http module and store returned HTTP instance into http variable. For example:

```
var http = require("http");
```

Node.Js Exmample

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:

```
http.createServer(function (request, response) {  
  // Send the HTTP header  
  // HTTP Status: 200 : OK  
  // Content Type: text/plain  
  response.writeHead(200, {'Content-Type': 'text/plain'});  
  // Send the response body as "Hello World"  
  response.end('Hello World\n');  
}).listen(8081);  
  
// Console will print the message  
console.log('Server running at http://127.0.0.1:8081/');
```

Node.Js Exmample

3. Combine **step1** and **step2 together** in a file named "main.js".

```
var http = require("http");

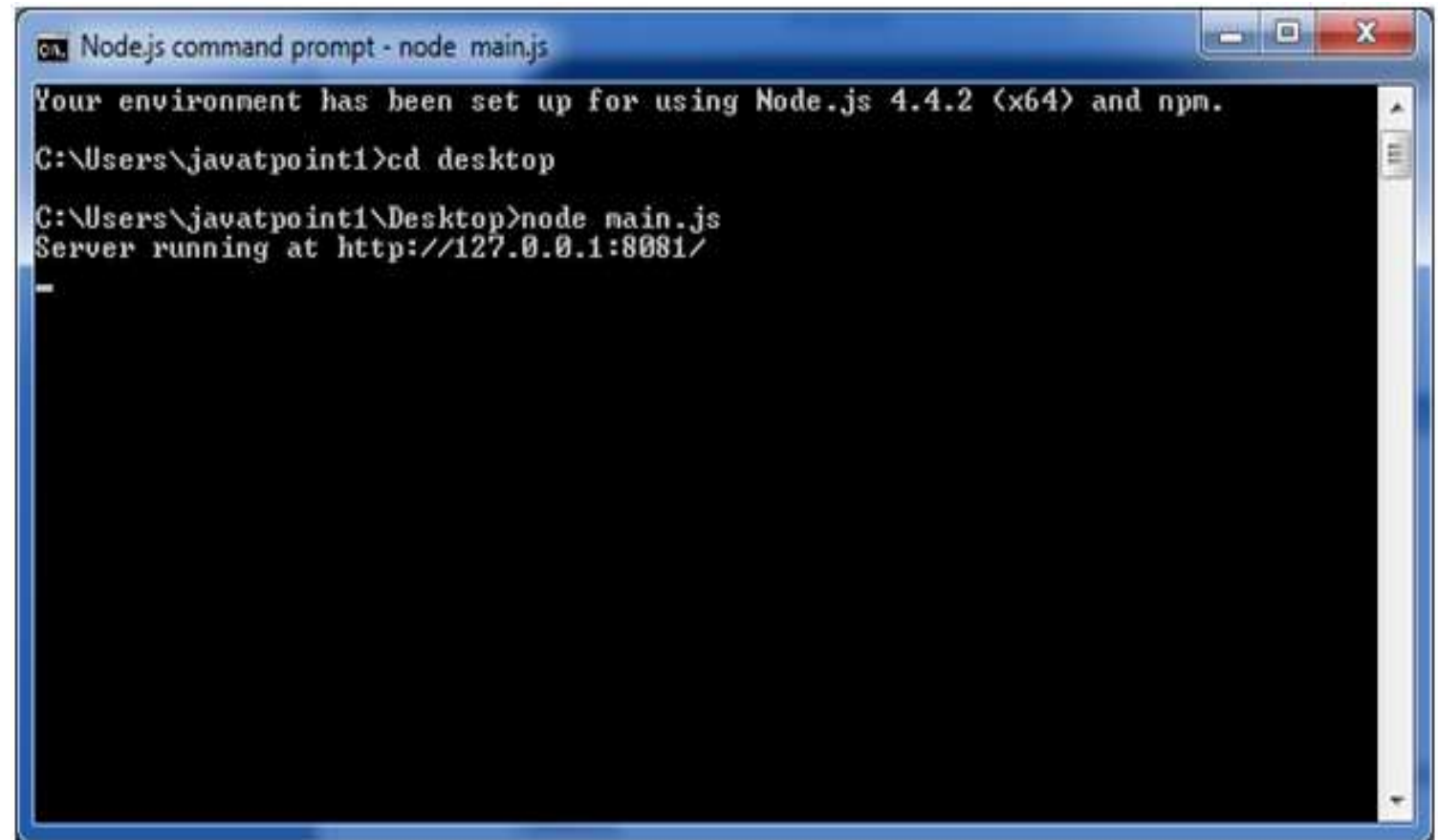
http.createServer(function (request, response) {
  response.writeHead(200, {'Content-Type': 'text/plain'});
  // Send the response body as "Hello World"
  response.end('Hello World\n');
}).listen(8081);

// Console will print the message

console.log('Server running at http://127.0.0.1:8081/');
```

Node.Js Exmample

How to start your server?

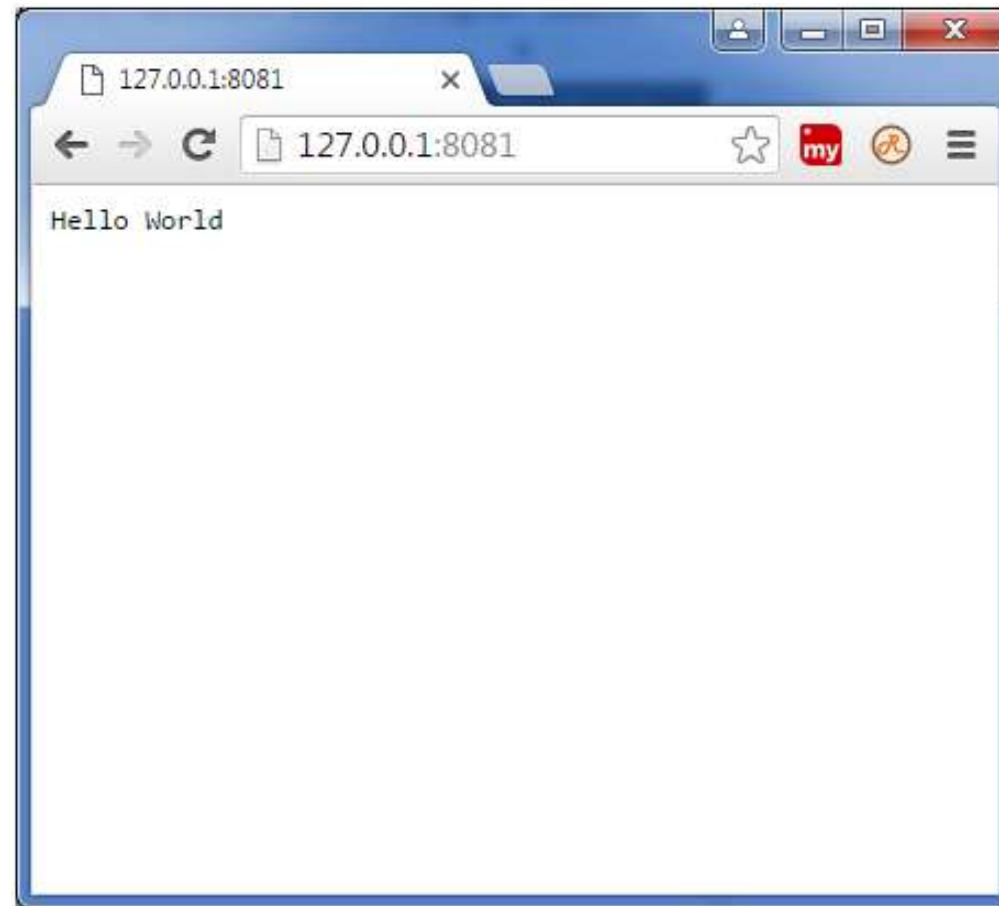


```
on. Node.js command prompt - node main.js
Your environment has been set up for using Node.js 4.4.2 (x64) and npm.
C:\Users\javatpoint1>cd desktop
C:\Users\javatpoint1\Desktop>node main.js
Server running at http://127.0.0.1:8081/
-
```


Node.Js Exmample

Now make a request

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



Node.js Package Manager (NPM)

NPM : Node Package Manager

NPM is a package manager for Node.js packages, or modules if you like.

www.npmjs.com hosts thousands of free packages to download and use.

The NPM program is installed on your computer when you install Node.js

What is a Package?

- A package in Node.js contains all the files you need for a module.
- Modules are JavaScript libraries you can include in your project.

Node.js Package Manager (NPM)

Download a Package

- Downloading a package is very easy.
- Open the command line interface and tell NPM to download the package you want.
- I want to download a package called "upper-case":

```
C:\Users\Your Name>npm install upper-case
```

NPM creates a folder named "node_modules", where the package will be placed.

All packages you install in the future will be placed in this folder.

Node.js Package Manager (NPM)

Using a Package

- Once the package is installed, it is ready to use.
- Include the "upper-case" package the same way you include any other module

```
var uc = require('upper-case');
```

Node.js Package Manager (NPM)

Create a Node.js file that will convert the output "Hello World!" into upper-case letters:

```
var http = require('http');
var uc = require('upper-case');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write(uc.upperCase("Hello World!"));
  res.end();
}).listen(8080);

console.log('Server running at http://127.0.0.1:8080/');
```

Save the code above in a file called "demo_uppercase.js", and initiate the file.
You will see the result at : **http://localhost:8080**

Node.js Package Manager (NPM)

NPM

NPM stands for Node Package Manager, responsible for managing all the packages and modules for Node.js.

Node Package Manager provides two main functionalities:

- Provides online repositories for node.js packages/modules.
- Provides command-line utility to install Node.js packages and also manages Node.js versions and dependencies

Node.js Package Manager (NPM)

Installing Modules using npm

Syntax:

```
npm install <Module Name>
```

Example to install famous Node.js web framework called express

```
npm install express
```

You can also check the version using :

```
npm version
```

Node.js Package Manager (NPM)

Create Your Own Modules

- You can create your own modules, and easily include them in your applications.

Example : Create a module that returns the current date and time:

```
exports.myDateTime = function () {  
  return Date();  
};
```

Use the **exports** keyword to make properties and methods available outside the module file.

Save the code above in a file called "myfirstmodule.js"

Node.js Package Manager (NPM)

Include Your Own Module

```
var http = require('http');  
var dt = require('./myfirstmodule');  
  
http.createServer(function (req, res) {  
  res.writeHead(200, {'Content-Type': 'text/html'});  
  res.write("The date and time are currently:  
" + dt.myDateTime());  
  res.end();  
}).listen(8080);
```

Notice that we use ./ to locate the module, that means that the module is located in the same folder as the Node.js file.

Save the code above in a file called "demo_module.js", and initiate the file:

Rendering HTML

- Create one HTML page to be used with Node.js
- Test.HTML File is as follows:

```
<html>  
<head>  
<title>Node Page</title>  
</head>  
<body>  
<h1>Node Js Learning</h1>  
</body>  
</html>
```

- To Access this page using Node.js we need to define a route in our server.js file.

Rendering HTML and Anchor Tag Routing

- Create one new HTML page Testnew.HTML File is as follows:

- **Testnew.html**

```
<html>
<head>
<title>Node Page</title>
</head>
<body>
<h1>Node Js Learning</h1>

<a href="index">Click</a>
</body>
</html>
```

test.html

```
<html>
<head>
<title>Node Page</title>
</head>
<body>
<h1>Node Js Learning</h1>

<a href="new">Click</a>
</body>
</html>
```

- To Access this page using Node.js we need to define a route in our server.js file.

Server.js -

```
const bodyParser = require('body-parser');
const express = require('express');
var app = express();
app.get('/', (req, res) => {
  console.log('Hello World');
});

app.listen(4500, () => console.log('Server Started at Port 4500'));

var path = require("path");

app.get('/index', function(req, res){
  res.sendFile(path.join(__dirname + '/test.html'));
});

app.get('/new', function(req, res){
  res.sendFile(path.join(__dirname + '/testnew.html'));
});
```

Node.js REPL

REPL stands for Read Eval Print Loop and it represents a computer environment like a Windows console or Unix/Linux shell where a command is entered and the system responds with an output in an interactive mode.

Node.js or Node comes bundled with a REPL environment.

It performs the following tasks –

REPL Environment → Read Eval Print and Loop

- **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.

Node.js REPL

REPL Environment ➔ Read Eval Print and Loop

- Computer environment / shell
 - You can enter the commands
 - System responds with an output in an interactive mode
- REPL starts by running "node" command on the command prompt.
- Can also execute various mathematical operations.
- Variables
 - Used to store values and print later.
 - Ex:
 - `a = 50`
 - `var b = 50`
 - `a+b`

Node.js REPL

REPL Environment → Read Eval Print and Loop

- Multiline expressions
- REPL supports multiline expressions like JavaScript.
- Ex:

```
var x = 0;  
do {  
  x++;  
  console.log("x: " + x);  
} while ( x < 10 );
```
- Underscore Variable
 - Use underscore `_` to get the last result
 - Ex: `a-b`
 `var diff = _`

Node.js REPL Commands

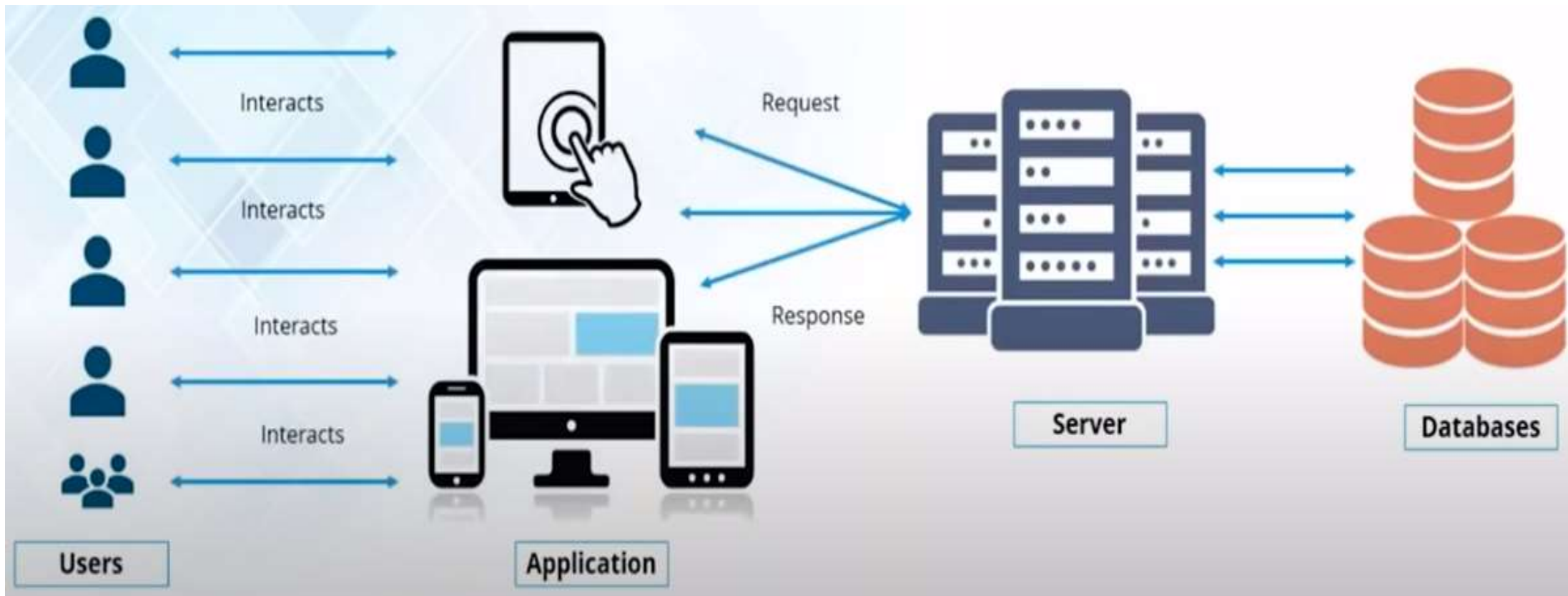
Commands	Used to
ctrl + c	Terminate the current command.
ctrl + c twice	Terminate the node repl.
ctrl + d	Terminate the node repl.
up/down keys	See command history and modify previous commands.
tab keys	Specify the list of current command.
.help	Specify the list of all commands.
.break	Exit from multi-line expressions.
.clear	Exit from multi-line expressions.
.save filename	Save current node repl session to a file.
.load filename	Load file content in current node repl session.

Node.js Web Module

What is Web Server

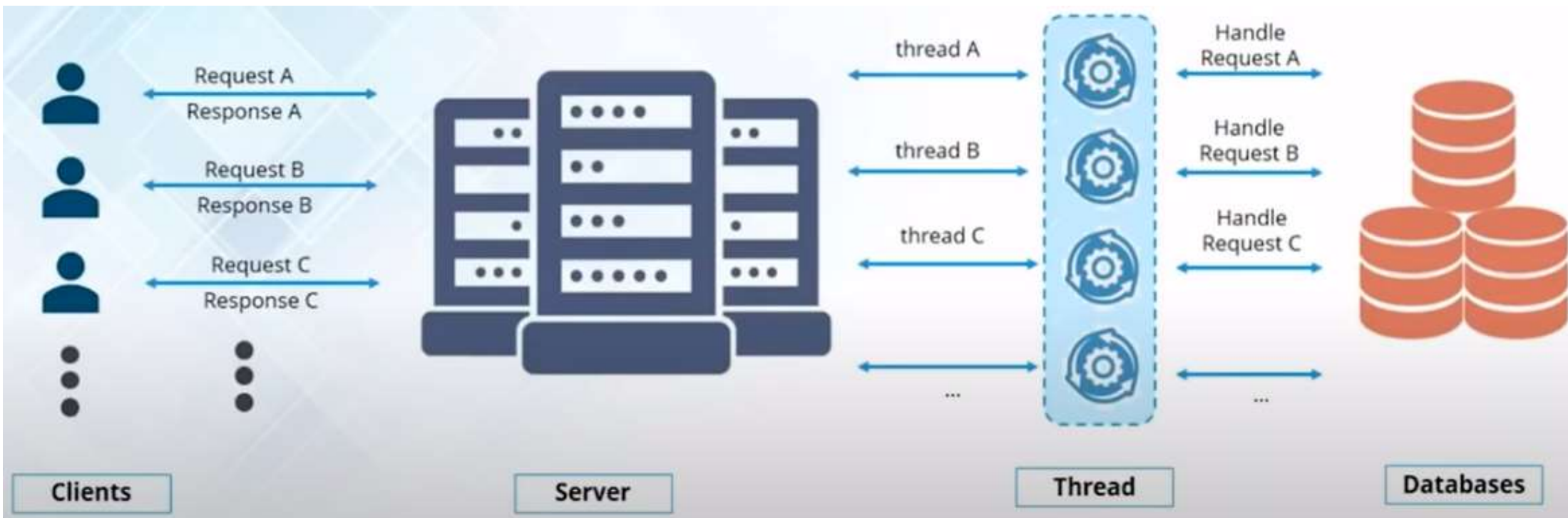
- A software program that handles HTTP requests sent by HTTP clients like web browsers, and returns web pages in response to the clients.
- Web servers usually respond with html documents along with images, style sheets and scripts.
- Support server side scripts using scripting language or redirect to application server which perform the specific task like
 - Getting data from database,
 - Perform complex logic etc.
- and then sends a result to the HTTP client through the Web server.

Client – Server Architecture



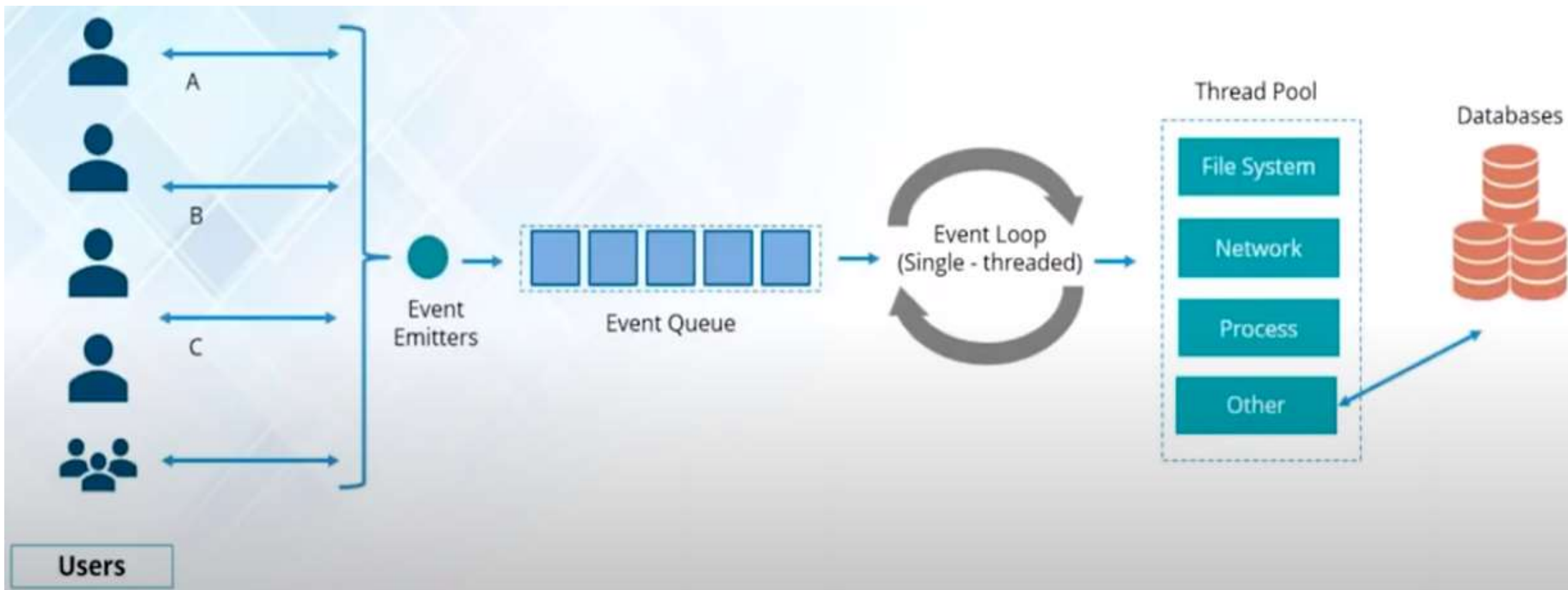
Multi Thread Model

- Each request is handled by a separate thread.
- **Limitations:**
 - If thread pool exhausted, then incoming request thread has to wait. (**Solution:** **scalability** but **costly**)
 - If a thread acquires a lock in the shared resources, it will block other thread and creates bottleneck situation; Hence it will decrease performance.



Single Thread Model

- Node.js is event-driven, handling all requests asynchronously from single thread
- Almost no function in Node directly performs I/O, so the process never blocks



Success Stories

Introduction to Node.js



Nexflix used JavaScript and NodeJS to transform their website into a single page application.

NETFLIX

Uber has built its massive driver / rider matching system on Node.js Distributed Web Architecture.



Linked in



Web Application Architecture

A web application can be divided in 4 layers

- **Client Layer:**
 - Contains web browsers, mobile browsers or applications which can make HTTP request to the web server.
- **Server Layer:**
 - Contains Web server which can intercepts the request made by clients and pass them the response.
- **Business Layer:**
 - Contains application server which is utilized by web server to do required processing. This layer interacts with data layer via data base or some external programs.
- **Data Layer:**
 - Contains databases or any source of data.

CRUD Operations

- Step:1- Download POSTMAN
- Step:2- Create a New Directory named nodetest
- Step:3- Go to terminal and reach to nodetest
- Step:4- npm init
- Step:5- npm install express mysql body-parser –save
- Step:6- Create index .js file and [GO TO CODE : CRUD OPERATION OF NODE](#)
- Step:7- NODE INDEX.JS
- Step:8- To perform Insert, Update and Delete Use POSTMAN

Node Js Routing

```
app.get('/', function (req, res) {  
  res.send('Hello World!') })
```

```
app.post('/', function (req, res) {  
  res.send('Hello World!') })
```

```
app.put('/', function (req, res) {  
  res.send('Hello World!') })
```

```
app.delete('/', function (req, res) {  
  res.send('Hello World!') })
```

Node Js routing is mainly used to execute CRUD Operations or REST APIs



THANK YOU