Understanding Server-Side JavaScript

Objectives

To SME: Looks like th
title is not detailed in
given input deck. Kin
confirm

01

Fundamentals of Node.js

Understand and analyse the procedure of making basic application using Node.js

02

Introduction to NPM

Analyse NPM and its importance in Node.js

03

Modules

Analyse modules and how to use them in Node.js

04

File Operations

Understand and analyse different File Operations with examples

05

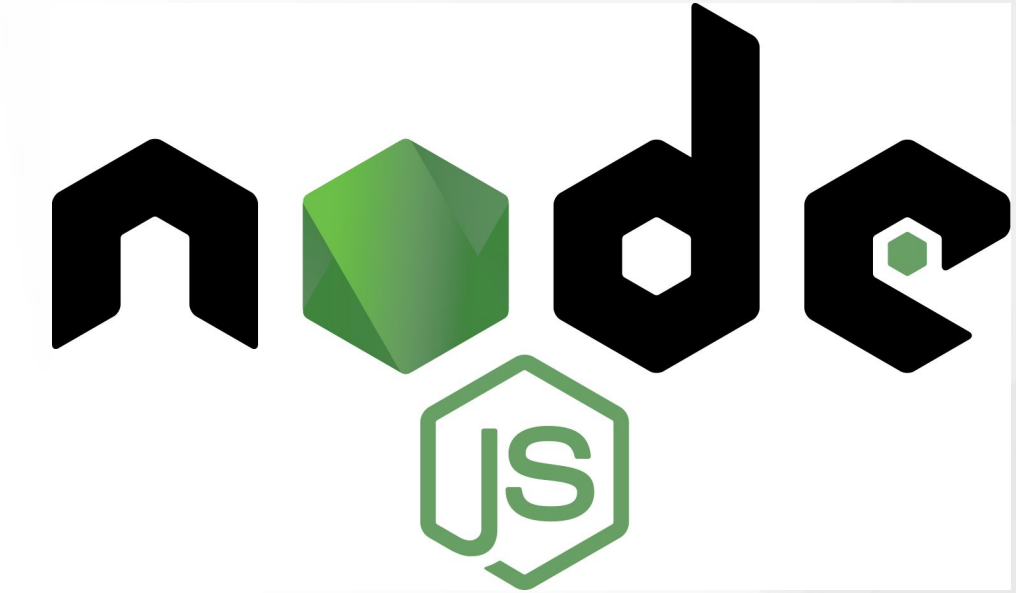
HTTP Methods

Analyse the concept of HTTP methods and their types

Fundamentals of Node.js

Node.js: An Introduction

- Node.js is an open source and cross-platform **JavaScript** runtime environment.
- It is used for server and networking applications.
- It runs JavaScript code outside a browser.
- Node.js uses **Google Java V8 engine** for execution.



Hero

Node.js: An Introduction

How browsers executes javascript code?

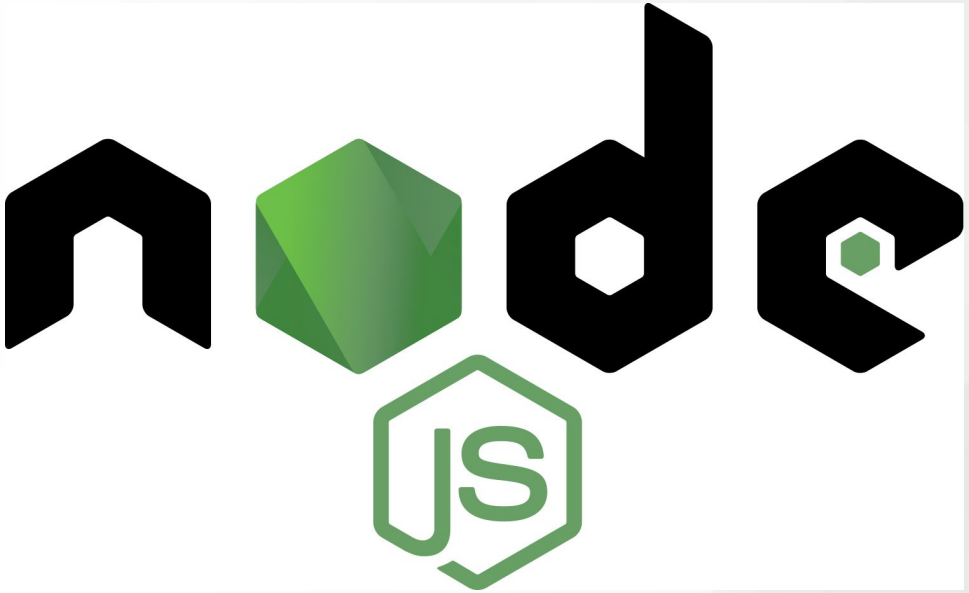
Ever wondered how browsers can execute js code? A browser is a collection of many different programs that performs specific tasks

One of these many programs is “**JS ENGINE**”. It is responsible to execute js code in the browser. In earlier days, only browsers were able to execute js code, that’s why js was primarily used in front end development.

Node.js: What about Server-side?

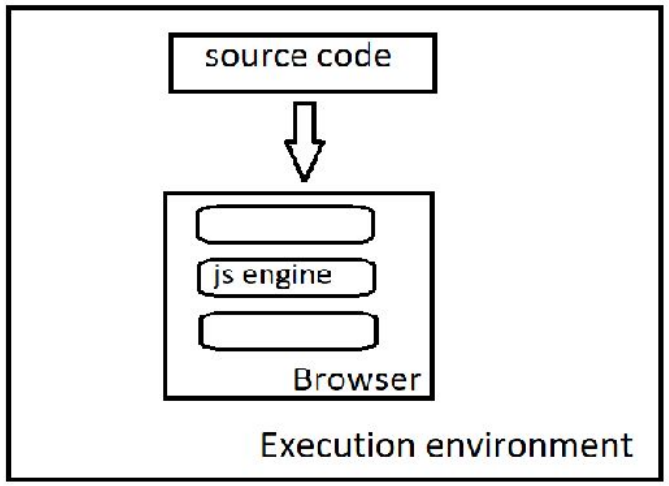
Then node js came into picture. Node js is a JavaScript runtime environment that is built on V8 js engine of chrome browser.

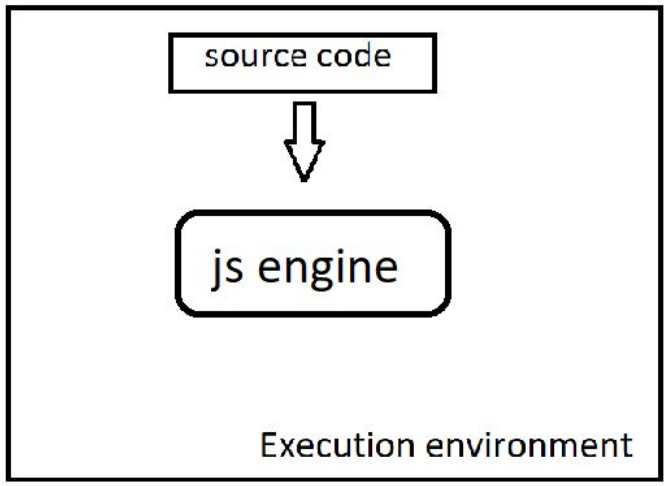
You can run js code directly by using node js. Because of this, it is now possible to build **server-side** applications using node.



Node.js: How Node is Different from a Browser?

- Browser is going to execute js on client side whereas Node is going to execute js on server side.
- Node js does not have **DOM** and other web APIs that browsers have.
- Document and window objects are not present in Node js
- Browsers don’t have many APIs that Node js have, example:- file system





Node.js: History

- The initial development of node js was led by Ryan Dahl in 2009. Only Linux and Mac os were supported in the initial release.
- In 2010, npm(node package manager) was introduced.
- The first version of node that supports windows was released in July 2011

Release	Status	Code name	Release date	Maintenance end
0.10.x	End-of-Life		2013-03-11	2016-10-31
0.12.x	End-of-Life		2015-02-06	2016-12-31
4.x	End-of-Life	Argon[68]	2015-09-08	2018-04-30
5.x	End-of-Life		2015-10-29	2016-06-30
6.x	End-of-Life	Boron[68]	2016-04-26	2019-04-30
7.x	End-of-Life		2016-10-25	2017-06-30
8.x	End-of-Life	Carbon[68]	2017-05-30	2019-12-31
9.x	End-of-Life		2017-10-01	2018-06-30
10.x	End-of-Life	Dubnium[68]	2018-04-24	2021-04-30
11.x	End-of-Life		2018-10-23	2019-06-01
12.x	Maintenance LTS	Erbium[68]	2019-04-23	2022-04-30
13.x	End-of-Life		2019-10-22	2020-06-01
14.x	Maintenance LTS	Fermium[68]	2020-04-21	2023-04-30
15.x	End of Life		2020-10-20	2021-06-01
16.x	Active LTS	Gallium[68]	2021-04-20	2024-04-30
17.x	**Current**		2021-10-19	2022-06-01
18.x	Planned		2022-04-19	2025-04-30

Legend: Old version | Older version, still maintained | **Latest version** | Future release

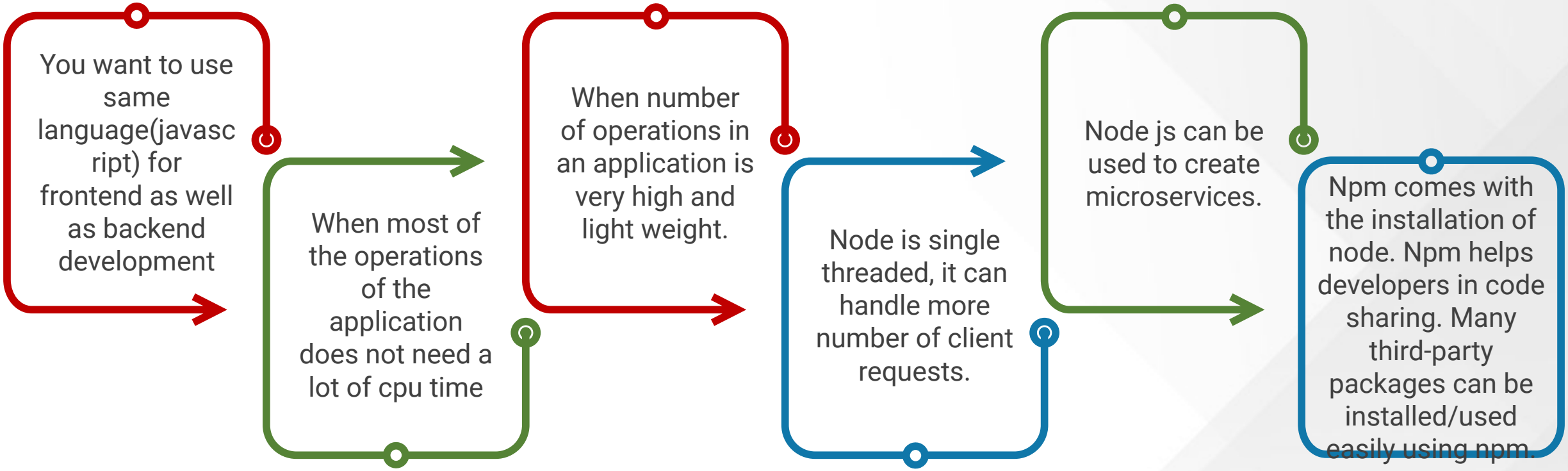
Node.js: Why Would I Use Node js?

Because of node, it is possible to use js for client side as well as server side

Applications can be built in short amount of time.

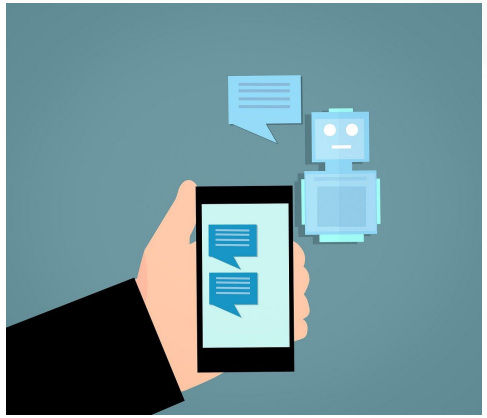
Scalable and flexible applications can be built.

Node.js: Why Would I Use Node js?



Node.js: Where Node js can be Used?

- Node can be used in building real-time applications such as messaging app or chat bot
- Node js can be used in IOT projects
- Node js can also be used for building audio or video streaming applications.



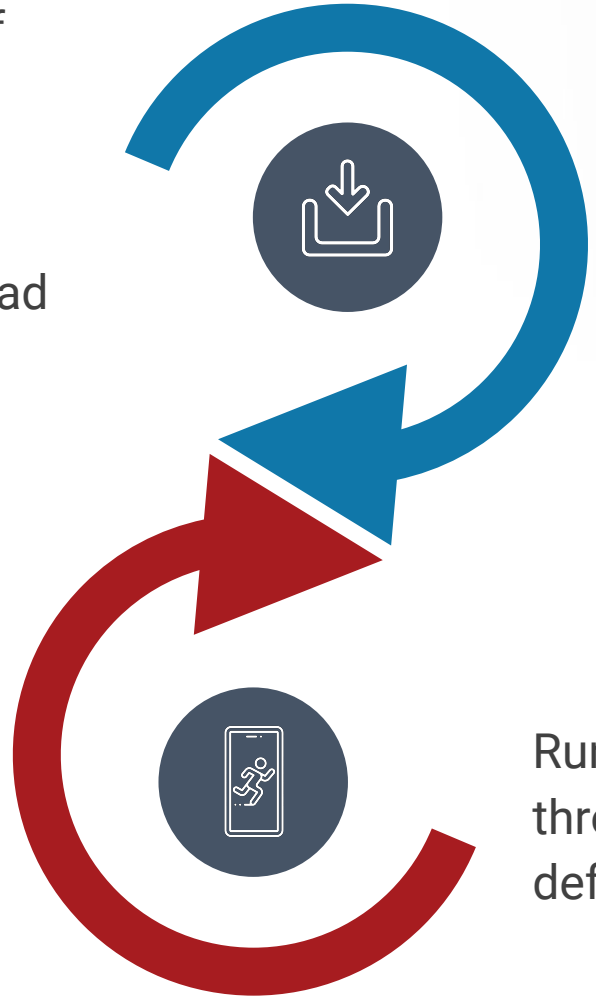
Node.js: Where Node js can be Used?

Some of the well-known applications that uses node js are:

- Netflix
- Paypal
- LinkedIn
- Uber

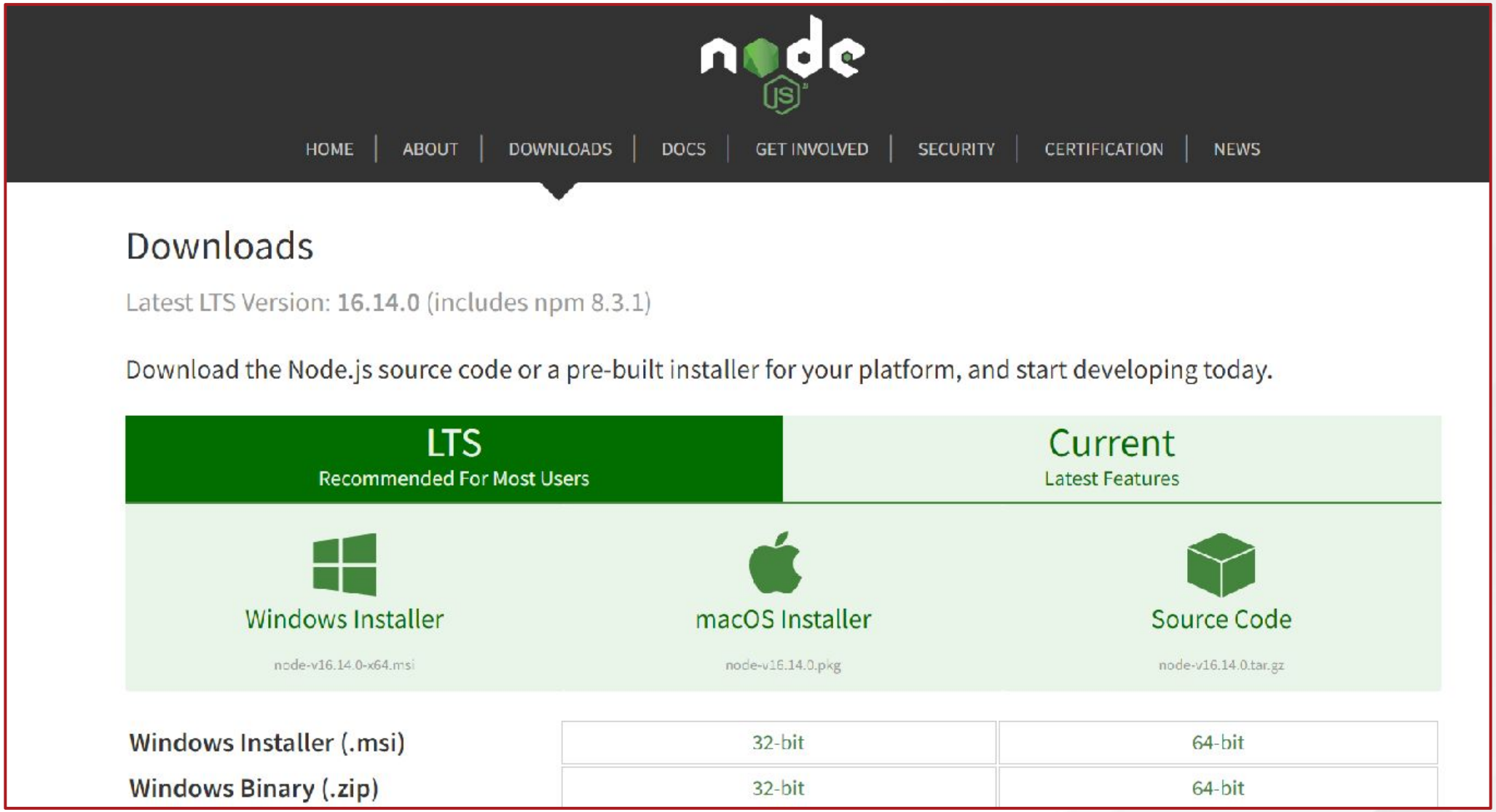
Node.js: Installation

Download the installer for “LTS” version of node from the link given below. It is the recommended version for most users. Download the installer according to the operating system of your system. Download link:- https://nodejs.org/en/download/



Run the installer and go through the steps. Keep the default values.

Node.js: Installation



Node.js: How to Run a js File Using Node

- Create a js file
- Write the code given in the image in that file and save it by the name “index.js”.
- Open cmd and make sure the current working directory is the folder that contains index.js file
- Type the command “node index.js”
- “Hello Node js” string will be printed on the screen.

index.js

```
1  function hello(){
2      console.log('Hello Node js');
3  }
4
5  hello();
```

C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.19044.1466]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Aishwarya Holkar\Desktop\code>node index.js
Hello Node js

C:\Users\Aishwarya Holkar\Desktop\code>
```

MCQ

Q1. State true or false, Is Node a programming language?

A. True
B. False

Q2. On which js enigne, node js is based on?

A. Chakra
B. V8
C. SpiderMonkey
D. React

Q3. State true or false, Is Node single-threaded?

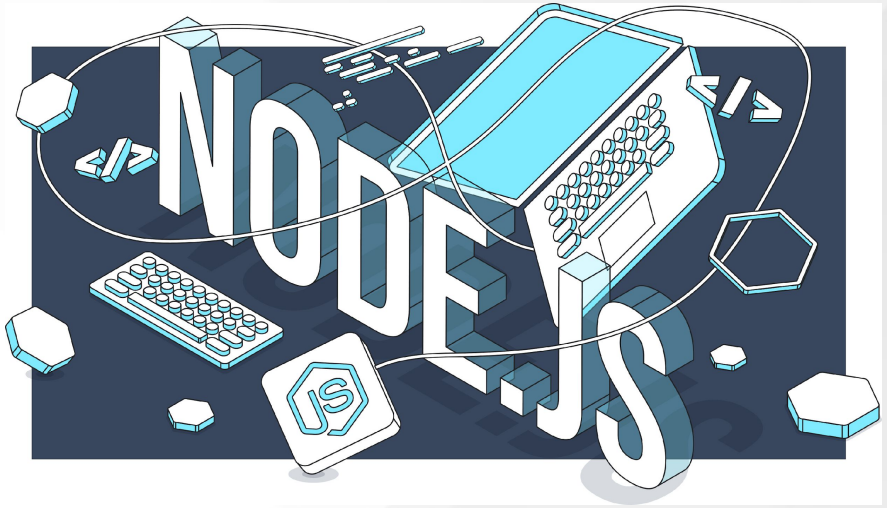
A. True
B. False

Q4. State true or false, node js can be used to build microservices?

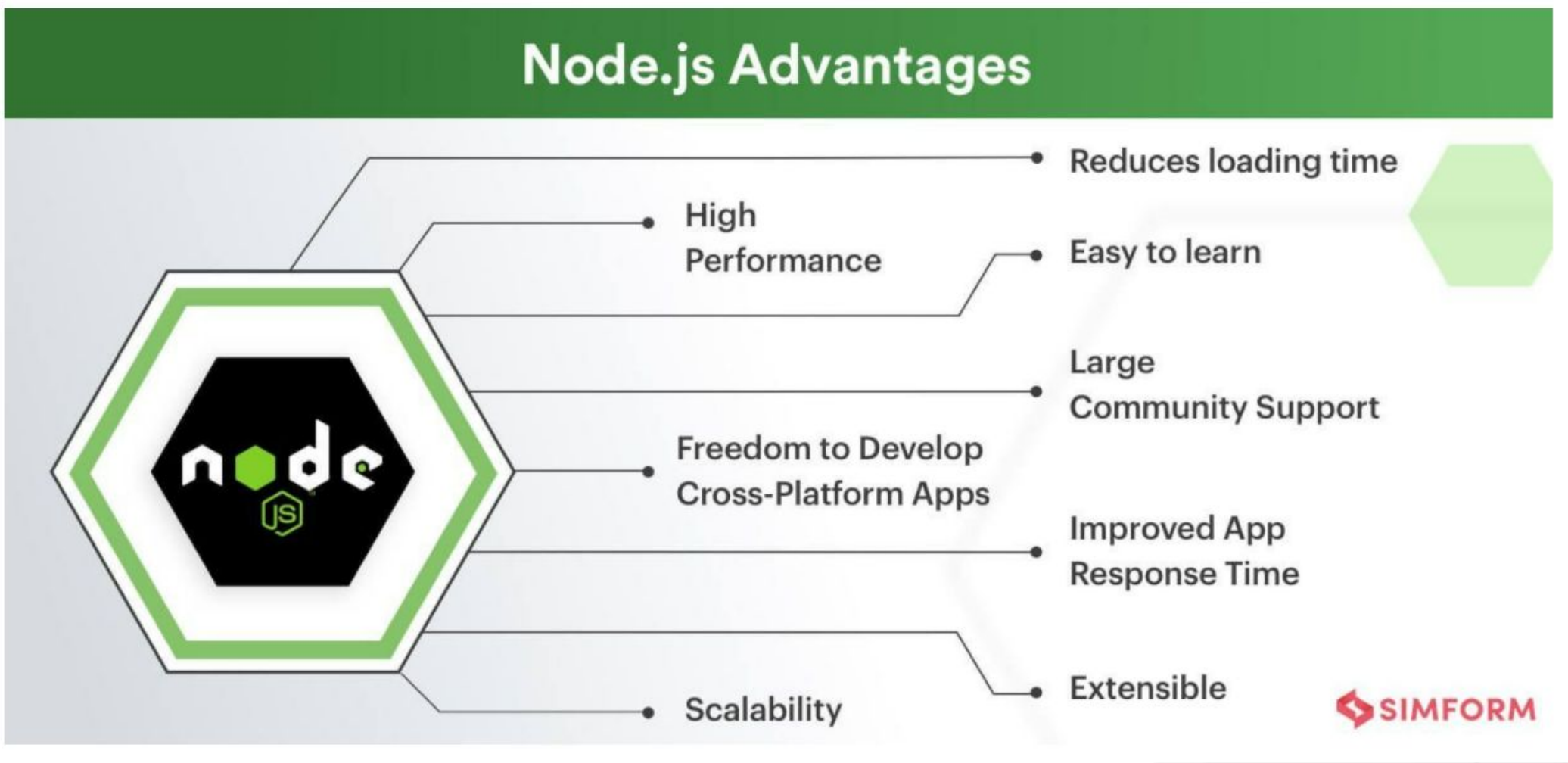
A. True
B. False

Applications of Node.js

- **Creating REST APIs** - A REST API in the context of Web Development is a web service that accepts requests and returns structured data (JSON in many cases).
- **Creating real-time services**: As Node.js supports asynchronous event-driven programming, it is well-suited to reactive real-time services.
- **Building microservices:** Since Node.js has a very lean core, it is best suited to building microservices. This is because you will only add dependencies that you need for the microservices, as opposed to the glut that comes with other frameworks.
- **Tooling:** For example, DevOps automations. These tools are used to reduce human assistance to processes that facilitate feedback loops between operations and development teams, so that iterative updates can be deployed faster to applications in production.



Summary


Node.js Advantages
Reduces loading time
High Performance
Easy to learn
Large Community Support
Freedom to Develop Cross-Platform Apps
Improved App Response Time
Extensible
Scalability
SIMFORM

Introduction to NPM

NPM - Node.js Package Manager

What is NPM?

- npm is the package manager for the Node.js platform.
- Also recognised as General Package Manager for JavaScript.
- The world's largest software library.
- When you have a Node.js project with a package.json file, you can run npm install from the project root and npm will install all the dependencies listed in the package.json. This makes installing a Node.js project from a git repo much easier!
- To install a public package, on the command line, run: npm install <packagename>
- To uninstall a package, use: npm uninstall <package-name>

Hero

NPM Global and Local Packages

What is NPM?

- Local packages are installed in the directory where you run npm install <package-name>, and they are put in the node_modules folder under this directory.
- Global packages are all put in a single place in your system (exactly where depends on your setup), regardless of where you run npm install -g <package-name>
- To view globally installed packages, run: npm list -g --depth 0
- Some examples of global packages - npm, nodemon, create-react-app, etc.

NPM - Node.js Package Manager

Need for NPM

- Installing and managing third party modules becomes very easy with NPM.
- With the help of NPX,node package execute(it comes with npm), you can directly execute any package without even installing that package.
- NPM can be used by developers to share their code with other developers

NPM - Packages

Commonly used libraries:-

- react
- express
- nodemon
- validator
- bcrypt
- passport

NPM – Package.json

package.json file contains the metadata about a project. It also contains the details about the dependencies of the project. It contains name, version, author and other fields.

package.json

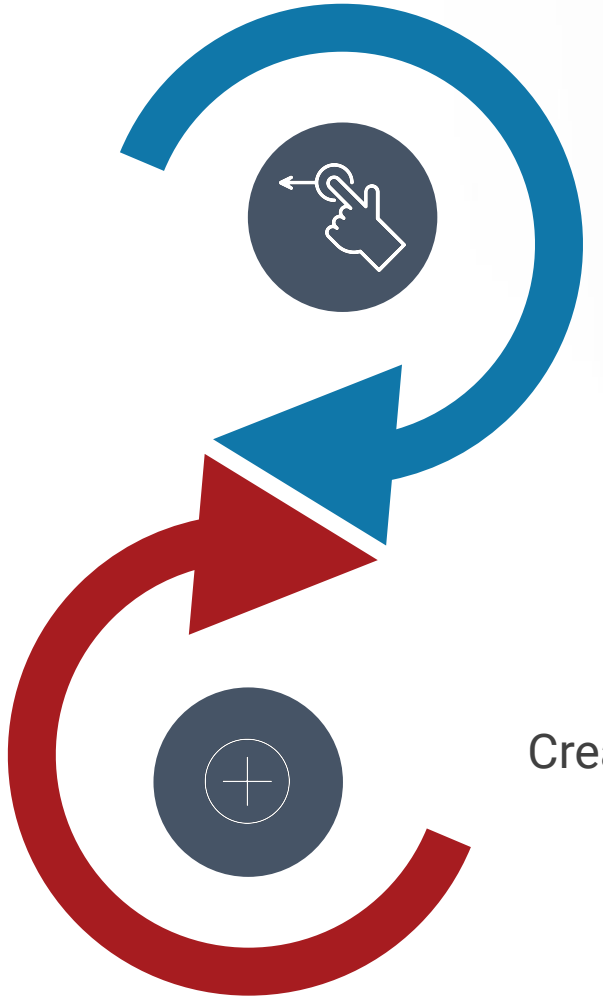
```
{
  "name": "my-app",
  "version": "1.0.0",
  "description": "a demo application",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

NPM – Package.json

How to create package.json?

There are two ways of creating this file.

Using “npm init” command

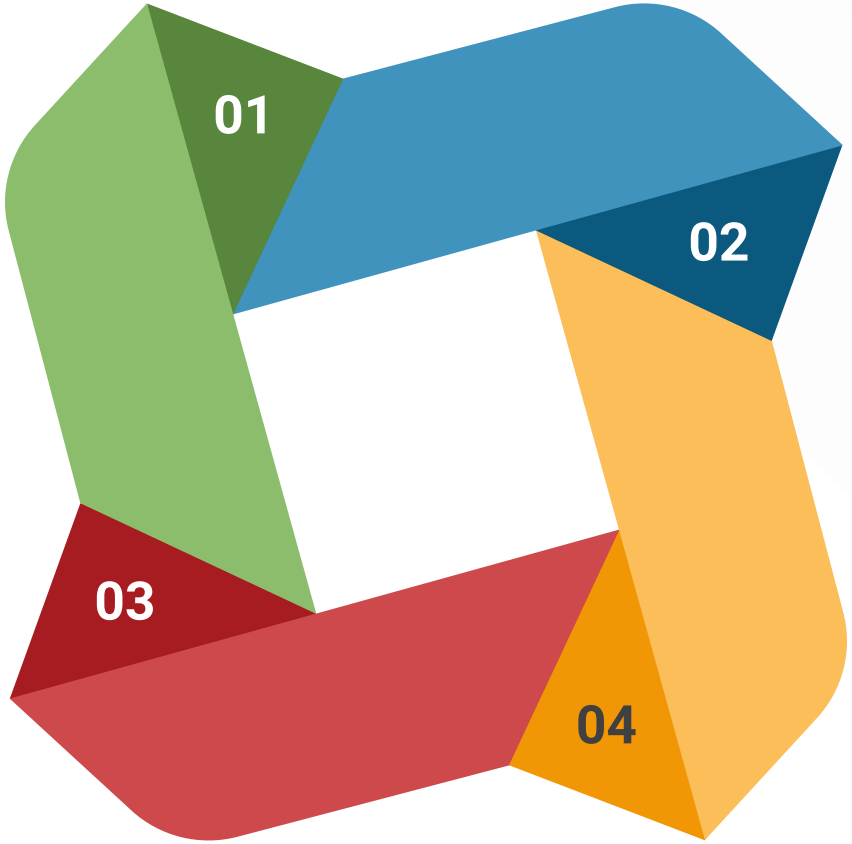


Creating file directly

Summary

npm is the package manager for the Node.js platform.

Also recognised as General Package Manager for JavaScript.



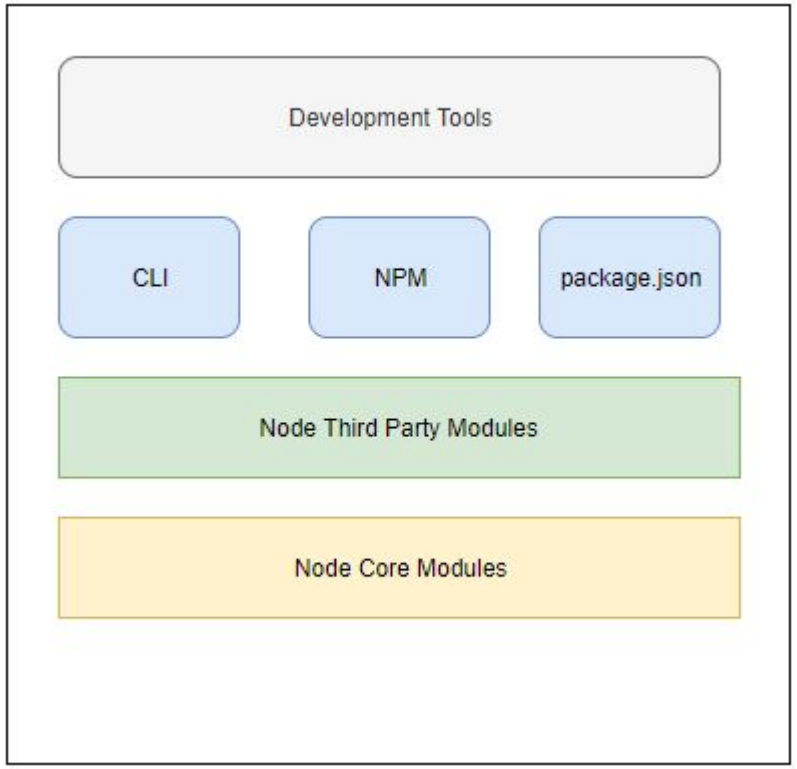
Installing and managing third party modules becomes very easy with NPM.

package.json file contains the metadata about a project. It also contains the details about the dependencies of the project.

Modules

Modules

- A module is a collection of code located in a file.
- There are two types of Modules:
  - Core and Local modules.
- Modules are exported using module.exports.
- Modules are imported using require() function

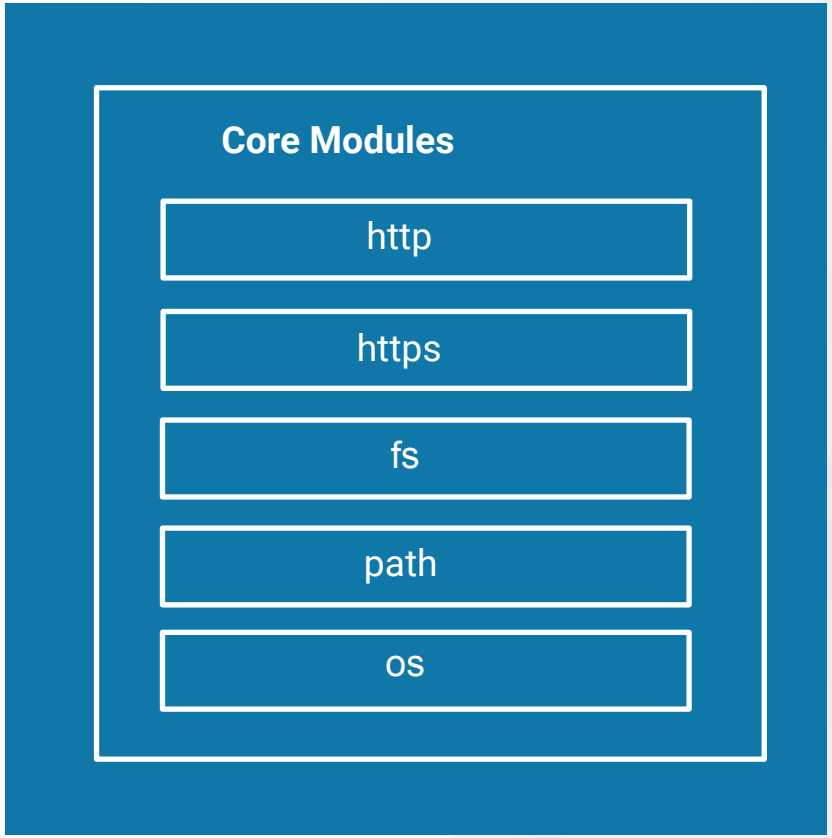


Module Categorisation

Core Modules

- Built-in (native) modules: These are modules that come with Node.js itself; you don't have to install them separately.
- Third-party modules: These are modules that are often installed from a package repository.

Custom Modules - These are modules that you have created within your application.



Module Categorisation

Module name	Description
http	This module allows node application to transfer data over http protocol. Also provides createServer() method which creates an http server.
https	Can be used to transfer data over http TLS/SSL protocol, a secure http protocol.
fs	It is used to perform file related operations such as create a file, read from a file etc.
path	It contains methods that provides a way for working with file and directory paths.
os	Provides information about the system's operating system.

Module: Import and Export(ES6)

We have to export an element(function or variable) so that it can used in another js file. There are two types of export:

- **Default**:- There can be only one default export in a js file.
- **Named**:- There can be any number of named exports.

```
1  import show from './index.mjs';
2  import {helloNode} from './index.mjs';
3
4  show();
5  helloNode();
```

app.js

```
1
2  function helloJs(){
3      console.log('hello javascript');
4  }
5
6  function helloNode(){
7      console.log('hello node');
8  }
9
10 var studentId = 101;
11
12 var studentName = "abc";
13
14 export default helloJs;
15
16 export {helloNode};
17
18 export {studentId,studentName};
```

index.js

Module: Import and Export(ES6)

To use an exported element in another file, we have to import that element.

- To import **default elements**, we don't have to use { }
- To import **named elements**, we must use { }

```
1  import show from './index.mjs';
2  import {helloNode} from './index.mjs';
3
4  show();
5  helloNode();
```

app.js

```
1
2  function helloJs(){
3      console.log('hello javascript');
4  }
5
6  function helloNode(){
7      console.log('hello node');
8  }
9
10 var studentId = 101;
11
12 var studentName = "abc";
13
14 export default helloJs;
15
16 export {helloNode};
17
18 export {studentId,studentName};
```

index.js

Using Modules

- We will see how to create a custom module and import into our app.
- In the example, the custom module (myfirstmodule) is used with the core ‘http’ module.

```
//Create file app.js
var dt = require('./myfirstmodule');    //This is
custom module

console.log(dt.myDateTime());
```

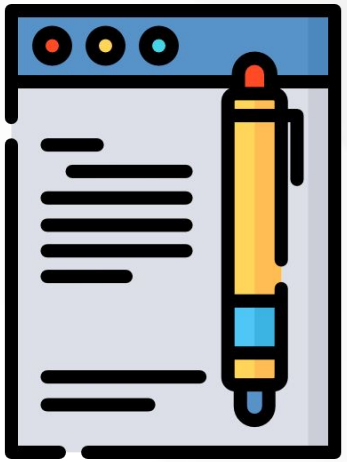
```
//Create file myfirstmodule.js



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

Summary

- A module is a collection of code located in a file.
- There are two types of Modules:
  - Core and Local modules.
- http module allows node application to transfer data over http protocol. Also provides createServer() method which creates an http server.

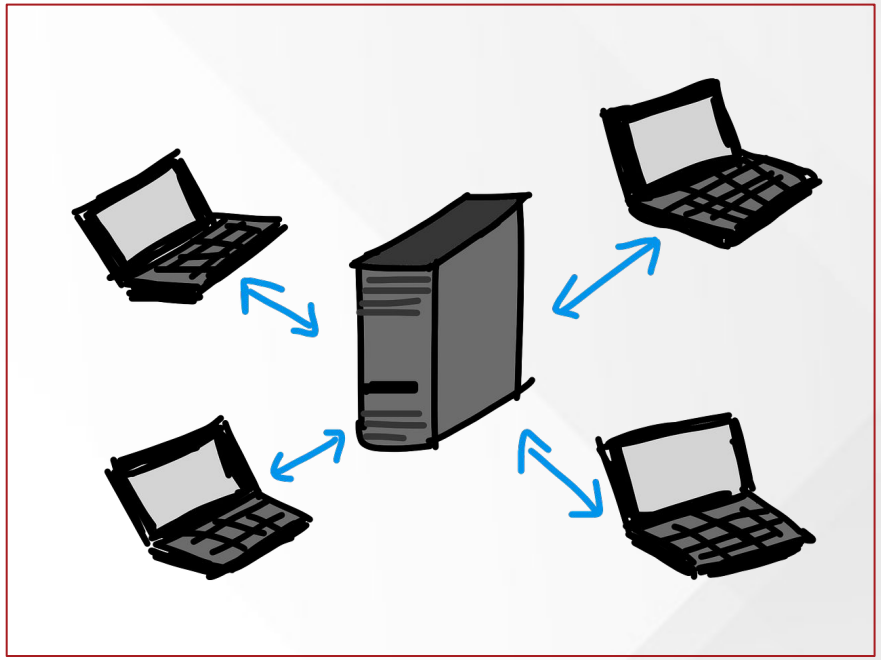


HTTP Model

HTTP: Hypertext Transfer Protocol

HTTP protocol is used for transferring data over the web. It is a stateless protocol and is based on request/response model.

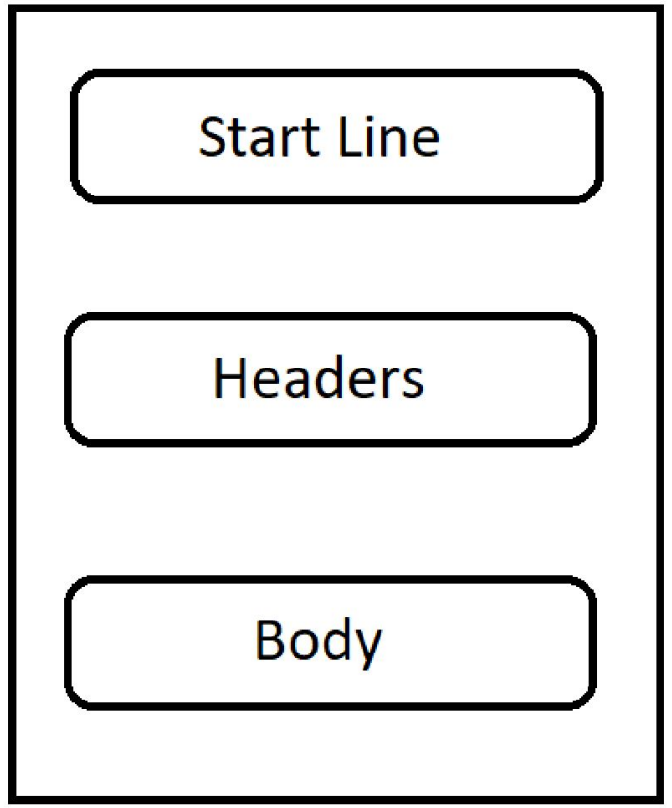
A client sends a “request” to the server. The server then processes the “request” and then sends appropriate “response” back to the client.



HTTP: Hypertext Transfer Protocol

The basic structure of request and response is same.

- **Start Line:-** contains http method(GET/POST etc) and url. In case of response, it contains the status code as well.
- **Headers:-** contains metadata about the message(request/response) such as content-type, content-length,date etc.
- **Body:-** Not all requests/reponses have this part. It contains the data.

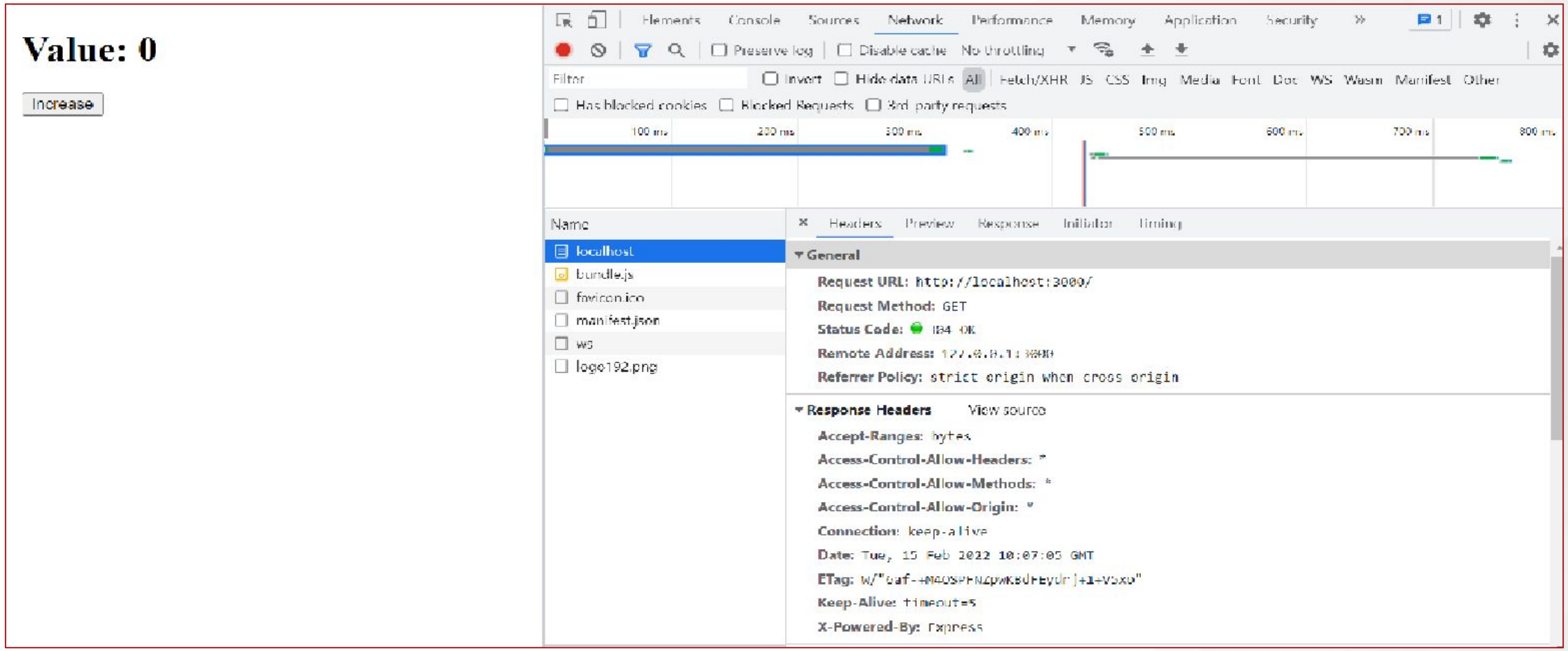


HTTP: Hypertext Transfer Protocol

How to view request and response headers(chrome)?

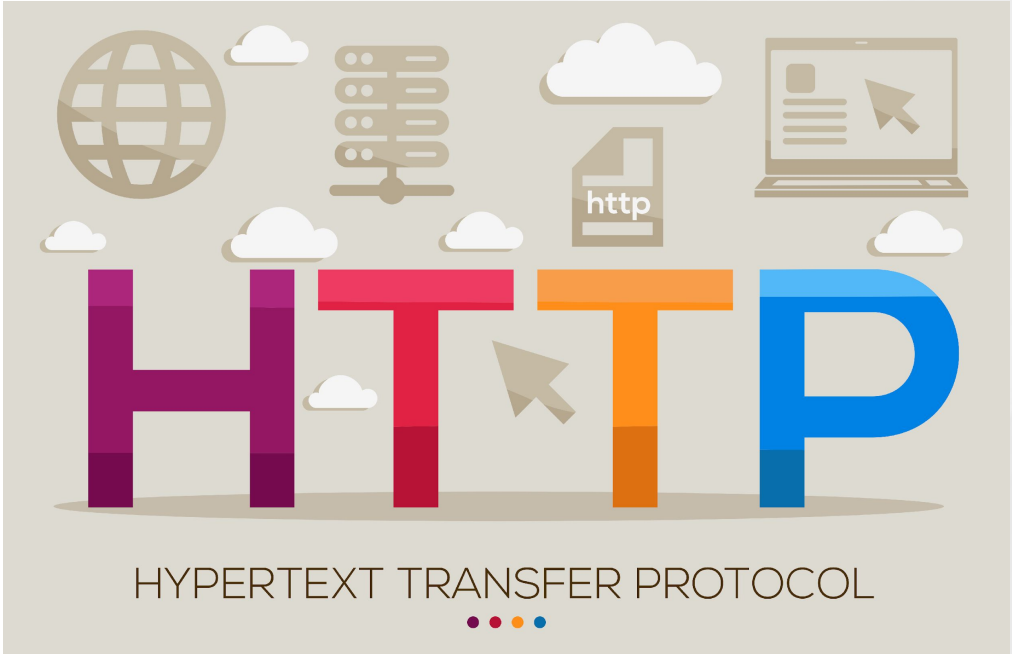
- Right click anywhere on the screen
- Select “Inspect” option
- Click on “Network” tab
- Refresh the page, select one from the “name” section.

HTTP: Hypertext Transfer Protocol



HTTP Methods

GET - Use GET requests to retrieve resource representation/information only – and not to modify it in any way.
POST - Is used to submit an entry to a specified resource, often causing a change of state.
PUT - Use PUT APIs primarily to update existing resource. If the resource does not exist, then API may decide to create a new resource or not.
DELETE - As the name applies, DELETE APIs are used to delete resources that are identified by the Request-URI.
PATCH - HTTP PATCH requests are to make partial updates on a resource.



HTTP Request and Response

- How to make an HTTP request?
- Customising an HTTP request.
- Request methods and samples using (GET and POST)

Creating HTTP Server using Node.js

- We have seen how to run a js file using node. Now, let's build a web server to view result by calling a URL on a browser.
- Create a file named app.js containing the code given here .
- Now from your terminal, run the web server using ‘node app.js’.

Visit http://localhost:3000 and you will see a message saying, "Hello World".

```
var http = require('http’);


http.createServer(function (req, res) {
  res.writeHead(200,
{'Content-Type': ‘text/html'});  // Not necessary. But better follow this professional way.
  res.end('Hello World!');
}).listen(3000);
```

Understanding the Code

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



http.createServer(function (req, res) {

  res.writeHead(200,
{'Content-Type': 'text/html'});    res.end('
Hello World!');

}).listen(3000);
```

HTTP Module is imported.

createServer function sets up the HTTP server

createServer function holds an anonymous function as a parameter.

The anonymous function handles requests and responses.

The anonymous function is called whenever a request is made to the server.

writeHead function to set the response status code as first parameter and content type as second.

write “Hello World” by passing as a parameter. Since it’s the end of response, we call it with res.end().

Listen to port mentioned here.

Understanding the Code

```
1  <html>
2      <head>
3          <title>GET request form</title>
4      </head>
5
6      <body>
7
8          <form action="http://localhost:3000" method="GET">
9              Name: <input type='text' name='nm'/><br/><br/>
10             Id: <input type='text' name='id'/><br/><br/>
11             <input type='submit' value='submit'>
12         </form>
13     </body>
14 </html>
```

index.html

Get Request

```
var http = require('http');
var url = require('url');

http.createServer(function (req, res) {

    var qs=url.parse(req.url, true);
    var path = url.parse(req.url, true).pathname;
    console.log(path);
    console.log(req.method);

    if(path === '/' && req.method === 'GET')
    {
        console.log(qs.query);

        res.writeHead(200, {'Content-Type': 'text/plain'});
        res.write(qs.query.nm);
        res.end();
    }else{
        res.writeHead(200, {'Content-Type': 'text/plain'});
        res.write("invalid method/url");
        res.end();

    }
}).listen(3000,() => {console.log('server started at 3000')});
```

app.js

Hands-on

Thank You!