Page No.
Date :

Name : Prajapati Ayushi. H

Sem : 7th

Roll No : 63

Subject : Full Stack Developme

Subject code : 701

(1) Nodejs : Introduction, features, execution architecture.

→ Nodejs is an open-source and cross-platform runtime env. for executing javascript code outside a browsers.

→ It is not a framework neither a programming language.

→ It is often used for building back-end services like APIs

→ It is Runtime env. + javascript lib.

• Features of nodejs:

→ It is easy to get started and can be used for prototyping and agile development

→ It provides fast and highly scalable services.

→ It uses javascript everywhere, so it's easy for a javascript programmer to build back end services using node.js.

→ Source code cleaner and consistent.

Page No.
Date :

→ large ecosystem for open source library.

→ It has asynchronous or non-blocking nature.

• Execution Architecture :

→ Nodejs runs on chrome v8 engine which converts javascript code into machine code. It is highly scalable, lightweight, fast and data intensive.

→ Nodejs accept the request from the clients and sends the response, while working with nodejs handles them with a single thread.

→ TO operate I/O operation or requests nodejs use the concepts of threads

→ Nodejs basically works on two-concept :

- Asynchronous
- Non-blocking I/O.

→ Non-blocking I/O -

- Non-blocking i/o means working with multiple requests without blocking the thread for a single request.

→ I/O basically interacts with external systems such as files, databases.

→ Asynchronous -

→ Asynchronous is executing a callback function.

→ The moment we get the response from the other server or database it will execute a callback function.

→ Callback functions are called as soon as some work is finished and this is because the node.js uses an event-driven architecture.

→ Event loop -

→ The event loop contains a single thread and is responsible for handling easy tasks like executing callbacks and network I/O.

→ eventloop is the heart of node.js.

→ When we start our node application the event loop starts running right away.

Page No.
Date :

(2) Note on modules with example.

→ In Node.js modules are the blocks of encapsulated code that communicate with an external applica" on the basis of their related functionality.

→ Modules can be single file or a collection of multiple file / folder.

→ Modules are of three types :

- Core modules
- local modules
- Third-party modules.

→ Core modules :

- Node.js has many built-in modules that are part of platform and come with node.js installation.
- These modules can be loaded into the program by using the required function.

Syntax :-

```
const module = require ('module_name');
```

Eg :

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

→ Package.json :-

- core modules are http, assert, fs, path, process, os, url, querystring.

- Local modules :-

- Unlike built-in and external modules, local modules are created locally in your node.js application.
- Let's create

Filename : calc.js

```
exports.add = function (x, y)
    { return x + y; y;
```

Filename : index.js

```
const addition = require ('./calc');
let x = 5, y = 10;

console.log (" addition of 5 + 10 is " +
            addition.add (x, y));
```

Run the file by command,

node index.js

→ Third-party module :

- Third-party modules are module that are available online using node package manager (npm).
- These modules can be installed in the project folder or globally.
- Eg :- npm i express.

(3) Note on package with example.

→ A package contains all the files needed for a module and modules are the javascript libraries that can be included in node project according to the requirement of project.

→ Downloading a package is very easy.

→ Eg : npm i package name.

→ NPM creates a folder named "node_module" where the package will be placed.

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

Page No.
Date :

→ Now once you install the package you can now include the package.

var package = require('package_name');

Eg:

→ npm i http
→ npm i upper-case
→ 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);

(4) Use of package.json and package-lock.json

→ Package.json & package-lock.json files are integral part of nodejs project and are commonly used when working with npm.

→ Package.json can is a meta data that contain which packages will be used in the project, the name of the project, whose the author.

→ It also contain the ~~comm~~ script for the command ~~which we~~ so that won't have to write it again & again.

→ The ds name of dev dependency's are also present in package.json

→ Package-lock.json keeps a record of the exact versions of all dependencies and their transitive dependencies installed in the node_modules directory.

→ This ensures that all developers working on the project as well as the produc^n environment, get precisely the same versions of the dependencies.

→ Package-lock.json file is automatically

generated and should not be manually modified.

(5) npm introduction and commands with its use.

→ npm stands for Node Package Manager and it is the package manager for node javascript platform.

→ It put modules in place so that node can find them, and manages dependency conflicts intelligently.

→ npm is free to use.

→ Most commonly, it is used to publish, discover, install & develop node programs.

→ Commands :-

- npm install
  - It install a package in the package.json file.

- npm update
  - it update the specified package.

- npm init
  - Creates a package.json file, asks some questions & create package.json file.

- npm start
  - Runs a command that is defined in start property in the scripts.

- npm ls
  - List all the packages as well as their dependencies.

(6) Describe use and working of following node's packages. - url, process, pm2, readline, fs, events, console, buffer, querystring, http, v8, os, zlib.

→ **url :**

The 'url' module provides utilities of URL resolution & parsing. The getters & setters implement the properties of URL objects on the class prototype.

```
const url = require('url');
```

→ process

This module provides interaction with current node.js process.

```
const process = require('process')
```

process object has a property 'argv' which is an array containing the properties passed to node process.

→ pm2

It enables you to keep applications alive forever. It also helps reload applica" w/o downtime, monitoring & clustering. It allows you to configure several diff" strategies for how your node.js applica" should restart

```
pm2 start app.js
```

→ readline

allows the reading of input stream line by line. It wraps up the process standard input & ~~auto~~ output objects.

```
const readline = require('readline');
```

→ fs

Helps us store, acces and manage data on our operating system. commonly used features of fs module include fs.readFile to read data from a file. fs.writeFile to write data to a file. and so on.

→ events:

This module emits named events that can cause corresponding functions or callbacks to be called. It provides you with EventEmitter class that allows you to manage events in the node application. use on() method of EventEmitter object to register an event handler for an event.

→ console

This module is a global object that provides a simple debugging console similar to javascript to display different levels of message.

→ Buffer

It is used to perform opera" on raw binary data. It refers to the particular

→ querystring

It is used to provide utilities for parsing and formatting URL querystring. It can be used to convert query string into JSON object and vice-versa.

→ http

It is TO use HTTP server in node, we need to require the HTTP module. The HTTP module creates an HTTP server that listens to server ~~por~~ ports & gives a response back to client.

→ zlib

is used to provide compression & decompression functionalities.