Page No.
Date :

Name :- Patel Krima Vipulbhai

Roll No :- 48

Subject :- Full stack development

Semester :- M.Sc(IT) 1st-Sem

Date :- 24/7/2023.

Assignment :- 1

Page No.
Date :

Q.1. Node JS : Introduction, features, execution architecture.

→ Introduction :-
- It is an open source server environment
- It is free and run on various platforms.
- It uses javascript on the server.
- It uses asynchronous programming.
- It runs single threaded, non-blocking, asynchronous programming, which is very memory efficient.

→ Features :-
1. Single threaded
2. Asynchronous
3. Event Driven
4. Open Source
5. Performance
6. Highly scalable
7. NPM
8. No buffering.

→ Server achitecture :-
To manage several concurrent clients, Node.js employs a "Single Threaded Event Loop" design. The javascript event-based model and the javascript call back mechanism are employed in the Node.js Processing Model. It employs two Fundamental concepts :
1. Asynchronous Model
2. Non-blocking of I/O operations.

→ These feature enhance the scalability, performance, and throughput of node.js web applications.

✱ Components of the Node.js Architecture

1. Requests :-
Depending on the actions that a user needs to perform, the requests to the server can be either blocking or non-blocking.

2. Node.js server :-
The Node.js server accepts user requests, processes them and returns results to the users.

3. Event Queue :-
The main use of Event Queue is to store the incoming client requests and pass them sequentially to the Event loop.

4. Thread Pool :-
The thread pool in node.js server contains the threads that are available for performing operations required to process requests.

5. Event Loop :
It receives requests from the Event queue and sends out the responses to the clients.

6. External Resources :-
In order to handle blocking client requests, external resources are used. They can be of any type (computation, storage, etc.)

Q.2. modules with example in node.js
→ A set of functions you want to include in your application.

1. Built in modules :-
- Node.js has a set of built in modules which you can use without any further installation.
- Look at an example :-
To include a module, use the require() function with the name of module :

ex.
```
var http = require('http');
http.createServer(function, (req, res) {
    res.writeHead(200, {'Content-Type':
    'text/html'});
    res.end('Hello world');
}).listen(8000);
```

2. Create our own module :-
ex.
```
exports.myDateTime = function () {
    return Date();
};
```
- Use the exports keyword to make properties and methods available outside the module file.

ex. var dt = require ('./myDateTime');

res.write ("The date and time are currently : " + dt.myDateTime());

Q.3. Note on package with example.

→ A package in Node.js contains all the files you need for a module.

- Modules are javascript libraries you can include in your project.

→ 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".

Syntax :- 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.

★ Using a Package :-

- Once the package is installed, it is ready to use.

var uc = require ("upper-case");

ex.
```
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(8000);
```

Q.4. Use of package.json and package-lock.json

→ The package.json file is the heart of Node.js system. It is the manifest file of any Node.js project and contains the metadata the project. The package.json file is the essential part to understand, learn and works with the Node.js. It is the first step to learn about development in Node.js.

package.json file can be categorized into two categories:

1. Identifying metadata properties:
It basically consist of the properties to indentify the module/project such as the name of project, version of module, license, author of the project, description about the project etc.

2. Functional metadata properties:-
it consists of the functional values/properties of the project/module such as the entry/starting point of the module, dependencies in project, scripts being used, repository links of Node.

A package.json file can be created in two ways.

① Using npm init :
npm init
- It provides users with default values which are editable by the user.

② Writing directly to file :-
We can directly write into file with all the required information and can include it in the Node project.

Package-lock.json. :-
- It is automatically generated for any operations where npm modifies either the node-modules tree, or package.json.
- It describes the exact tree that was generated, such that subsequent installs are able to generate identical trees, regardless of intermediate dependency updates.
- It contains the name, dependencies and locked version of the project.

Q.5. Nodejs packages

→ 1. Express

- Expaess.js is the fastest, unopinionated, and simplest web framework for Node.js.
- Features of Expaess:-
  - Robust routing
  - Focus on high-quality performance.
  - Super high test coverage.
  - HTTP helpers.
  - content negation
  - Executable for developing apps & Apis faster.

2. Async JS

- It is heavily used in Node.js to ensure a non-blocking operations flow.
- AsyncJS provides several functions that include usual functions such as 'map', 'filter', 'reduce', 'each' as well as some common patterns for asynchronous flow control functions.

Advantage:

- A collection of Async functions helps to control the flow through the script.
- Help you avoid memory leaks.

3. Moment.js

- It is lightweight Javascript development tool for date and time manipulation.

- It makes dates and times easy to format, parse, validate and internationalize using a clean and concise API.

4. Axios:-
- Axios is an HTTP client API framework that supports promises to perform a request.
- Requests are used to make a communication with the server, then a framework like Axios will return a response with a promise to whether your request was fulfilled or rejected. It performs request such as GET, POST, DELETE, PUT.

5. Nodemon:-
- It is a monitoring tool and it helps Node.js developers by automatically restarting an application when file changes in the app directory are detected.
- With nodemon, you do not need any additional code or development method changes. It is a replacement wrapper for Node.js

6. Nodemailer:-
- It is Node.js application module that allows easy as pie email sending.

- 7. Express-Mailer is similar to Node-mailer used to send emails from your application and response object.

Q.6. npm introduction and commands with its use.

→ NPM stands for Node Package Manager and it is the package for the Node Java Script platform. It put modules in place so that node can find them, and manages dependency conflicts intelligently.

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

→ Commands :-

1. NPM Install Command :
   Syntax : npm install
   - install a package in the package.json file in the local node_modules folder.

2. NPM uninstall Command :
   Remove a package from the package.json file and removes the module from the local node_modules folder.
   npm uninstall

3. NPM update command :-
   - It use updates the specified package.
   npm update

4. NPM Global update command :-
   - It use the update action to each globally installed package.
   npm update -g

5. NPM deprecate command:
- It command will deprecate the npm registry for a package, providing a deprecation warning to all who attempt to install it.

npm deprecate

6. NPM Outdated Command:
- Checks the registry if any package is outdated. It prints a list of all packages which are outdated.

npm outdated.

7. NPM Doctor command:
Checks our environment so that our npm installation has what it needs to manage our Javascript packages.

npm doctor

8. NPM initialize Command:
Creates package.json file in our directory. It basically asks some questions and finally creates a package.json file in the current project directory.

npm init

9. NPM Start :-
Runs a command that is defined in the $cripts. If not defined it will run the node server.js command.

npm start.

10. NPM build Command
- used to build a package.
npm build.

11. NPM list command:
list all the packages as well as their dependencies installed.

12. NPM version :-
- used for package version.

13. NPM search:
for searches the npm registry for packages matching the search terms.

14. NPM Help:
documentation for a specified topic. It is used whenever the user needs help to get some reference.
npm help

15. NPM owner :-
Manages ownership of published packages.
It is used to manage package owners.

npm owner.

Q.7. Describe the use and working of following Nodejs packages. Important properties and method and relevant programs.

1. url :-

→ It is provides utilities for URL resolution and parsing. The getters and setters implement the properties of URL objects on the class prototype, and the URL class is available on the global object.

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

eg. Parse an address with the url.parse() method, and it will return a URL object with each part of the address as properties.

```
var url =require ("url");
var adr = 'http://localhost:8000/default.htm?
year =2017 & month =february';
var q = url.parse (adr, true);

Console.log (q.host);
Console.log (q.pathname);
console.log (q.search);
Console.log (q.query.month);
```

2. process, prm2.

→ A process object is a global object, so it can be accessed from anywhere. As it is a predefined library, so we don't have to download it to our system globally.

eg.

```
var process = require('process');
Console.log(process.cwd());
Console.log(process.version());
Console.log(process.platform);
Console.log(process.features);
```

3. readline :-

→ Readline Module in Node.js allows the reading of input stream line by line. This module wrapsup the process standard output and process standard input objects. The readline module makes it easier to input and read the output given by the user. To use this module, create a new Java script file and write the code.

ex.
```
const readline = require ('readline');
let r1 = readline.createInterface( process.stdin,
                                   process.stdout);

r1.question ('What is your age?', (age) => {
    Console.log ("Your age is : '+age)
});
r1.close();
```

Properties :-

① question () - used for asking questions from the user

② close () - It will close the interface.

③ setPrompt() - It is used to set the particular statement to the console.

④ prompt() - displaying the statement which is set in setPrompt() method.

⑤ On() - takes the first argument as a 'line' event. This event is invoked whenever the user presses Enter key.

4. Fs :-

- It allows you to work with the file system on your computer.

Properties :-

Fs.readFile() - is used to read files

Fs.appendFile() - appends specified content to a file. IF file not exist, it will be created.

Fs.open() - takes a 'flag' as the second parameters, if it is "w" for "writing", the specified file is opened for following. writing.

Fs.writeFile() - replace the specified file and content if it exists.

Fs.unlink() - deletes the specified file.

Fs.rename() - rename specific file.

ex.

```
var Fs. = require ('fs');
fs.appendFile ("my.txt", "Helloworld", function(e){
        if (e) throw e;
    Console.log ('saved');
});

fs.rename ("my.txt", "demo.txt", function (e){
        if (e) throw e;
    Console.log ('File renamed');
});
```

5. Event :-

- Nodejs has a built in module, called 'Events' where you can create, fire and listen for your own events.

→ EventEmitter class :
It is a core of a events module. it provides method to handle and emit events.

→ Event handling :
This method is used to register an event listen for a specific event.

eg.
```
const EE = require ('events');
class MyClass extends EventsEmitter {
    constructor () {
        super ();
    }
}
Const obj = new MyClass()
objEon ('event', () => {
    .log ('customer event');
});
obj. emit ('event');
```

6. Console :-

→ It is a built in Object in node.js and web browsers that provides a simple debugging and logging mechanism.

→ It allows you to interact with the standard output, error, debug info and errors during the execution of your code.

Page No.
Date :

Methods :

① Console.log() → Points the provided data and objects to the console spread by spaces.
→ It automatically converts the values to string.

② Console.error() → points to the standard error stream instead of standard output.

③ Console.warn() → Points to standard error stream but it is often used for less server warning or alerts.

④ Console.clear() → It is clear the console.

7. Buffer :
- The buffer class is a built in global object that provids a way to handle binary data.
- A buffer is like an array of, integers but each element is represented by single byte of data.

ex. Buffer alloc(5);
Buffer from([65, 66, 67]);

8. query string :
→ The query string is a part of the URL that comes after the question mark '?'. It is used to send data to the server as key value pairs.

→ Passing Query strings :-
queryString.parse(str[,sep[,eq[,option]]]);

8 http :-

- The http module in node.js is a core module that provids functionality to create HTTP servers and make HTTP request.

ex. const http = require ("http");
const server = http.createServer((req, res) => {
res.end("Hello world");
}).listen(8000);

9. V8 :-

It is an open-source js engine developed by google, It is written in C++ and is primarily used to execute Javascript code in web browser and server-side environments.

- V8 is at the core of many popular js environments including google chrome. Node.js and several other runtime environments.

10. OS:

The 'os' module in Node.js is a core module that provides operating system related utilities

os.platform();

const os = require ("os");
console.log (os.platform());

11. Zlib :-

- It module in Node.js is a core module that provides compression and decompression functionalities using the Zlib library.

ex. zlib.deflate(buf[, options], callbacks)

```
Const zlib = require('zlib');
Const input = Buffer.from('Hello zlib
                          compress');
zlib.deflate(input, core, compressed)=>{
    if (err) console.log(err);
});
```