Name : Navadiya Nensi B. Submission Date : 24/7/23

Roll No : 22

Subject : OSWD (304) MSCIT - 3

PAGE NO. :
DATE : / /

Assignment : 1

Que : 1 Nodejs : Introduction , features , execution architecture

⟹ Introduction :

- Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a brocoser.
- Node.js was written and introduced by Ryan Dahl in 2009.
- Node.js

⟹ Features :

1. Extremely fast :
- Node.js is built on Google Chrome's V8 JavaScript Engine, so its library is very fast in code execution.

2. 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 serves to get a response from the previous API call. It is also a reason that it is very fast.

PAGE NO.:
DATE: / /

3. Single threaded :
- Node.js follows a single threaded model with event looping.

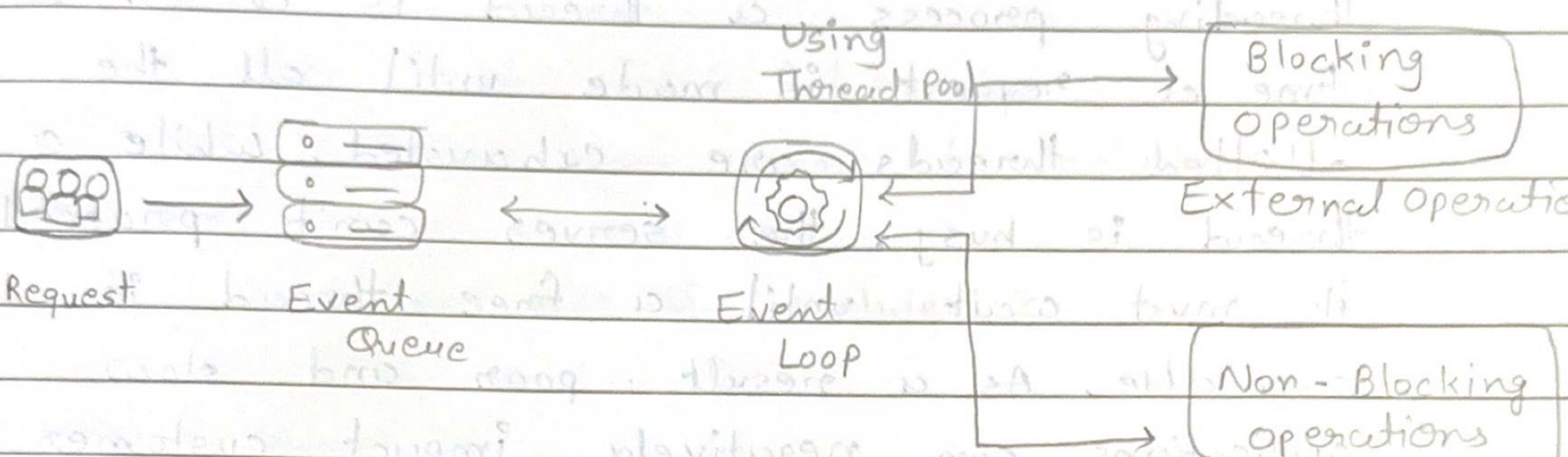
4. Highly Scalable :
- Node.js is highly scalable because event mechanism helps the server to respond in a non-blocking way.

5. No buffering :
- Node.js 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.

6. Open Source :
- Node.js has an open source community which has produced many excellent modules to add additional capabilities to Node.js applications. License : Node.js is released under the MIT license.

PAGE NO. :
DATE : / /

⟹ Execution Architecture :



- Node.js offers a "single - Threaded Event Loop" architecture to manage concurrent requests without creating multiple threads and using fewer threads to utilize fewer resources. That's why developers prefer Node.js architecture to take the advantages Node.js has.
- Another reason for the popularity of Node.js is its callback mechanism and Javascript - event - based Model. Node.js event loop architecture enables Node.js to execute blocking I/O operations in a non - blocking way. Also, you can easily scale your Node.js application with its single thread than multi - thread per request under ordinary web loads.
- Actually, Node.js has two types of threading :
  - Multi - Threading
  - Single Threading

- Multi - threading is a program execution model that creates multiple threads within a process. These threads execute independently but concurrently

share process resources. During this multiple-threading process, a thread is chosen each time a request is made until all the allotted threads are exhausted. While a thread is busy, the server can't proceed; it must wait until a free thread is available. As a result, poor and slow applications can negatively impact customer experience.

- However, Node.js uses a single-threaded processing model. Single-threaded architecture handles Input/output operations by using event loops to handle blocking operations in a non-blocking way, whereas multi-thread architecture processes every request as a single thread.

PAGE NO. :
DATE : / /

Que : 2 Note on modules with example.

- Modules are JavaScript Libraries :
- A set of functions you want to include in your application.
- There are two type of modules :
  (1) Built - in
  (2) External

Built - in Modules

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

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

Now your application has access to the HTTP module, and is able to create a server.

(1) Built - in http Module :

- The HTTP module can create an HTTP server that listens to servers ports and gives a response back to the client.
- Use the createServer() method to create an HTTP server
- The function passed into the http.createServer() method, will be executed when someone tries to access the computer on port 8080.
- If the response from the HTTP server is supposed to be displayed as HTML, you should include an HTTP header with the correct content type :

```
res.writeHead(200, {'Content-Type': 'text/html'});
```

- The function passed into the http.createServer() has a req argument that represents the request from the client, as an object (http.IncomingMessage object).
- This object has a properly called "url" which holds the part of the url that comes after the domain name : http://localhost:8080/emp

(2) Utility Modules :

- os Provides basic operating-system related utility func.
- path Provides utilities for handling and transforming file paths.
- net Provides both servers and clients as streams. Acts as a networks wrapper.
- dns Provides functions to do actual DNS lookup as well as to use underlying operating system name resolution functionalities.
- domain Provides ways to handle multiple different I/O operations as a single group.
- crypto Provides cryptographic functionality that includes a set of wrappers for open SSL's hash HMAC, cipher, decipher, sign and verify functions.
- tls TLS stands for Transport Layer Security. It is the successor to secure Sockets Layer (SSL). TLS along with SSL is used for cryptographic protocols to secure communication over the web. TLS uses public-key cryptography to encrypt messages. It encrypts communication generally on the TCP Layer.

PAGE NO. :
DATE : / /

- querystring To handle URL query strings
- readline To handle readable streams one line at the time
- stream To handle streaming data
- string_decoder To decode buffer objects into strings
- timers To execute a function after a given number of milliseconds.
- tty Provides classes used by a text terminal
- url To parse URL strings
- util To access utility functions
- v8 To access information about V8 (the JavaScript engine)
- zlib To compress or decompress files
- buffer To handle binary data

PAGE NO.:
DATE: / /

Que : 3 Note on Package with example.

- NPM (Node Package Manager) is the default package manager for Node.js and is written entirely in Javascript. Developed by Isaac Z. Schlueter, it was initially released in January 12, 2010.
- NPM manages all the packages and modules for Node.js and consists of command line client npm. It gets installed into the system with installation of Node.js. The required packages and modules in Node project are installed using NPM.
- 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 the project.
- NPM can install all the dependencies of a project through the package.json file. It can also update and uninstall packages. In the package.json file, each dependency can specify a range of valid versions using the semantic versioning scheme, allowing developers to auto-update their packages while at the same time avoiding unwanted breaking changes.
- npm is open source

Example :
Installing the express package into the project. Express is the web development framework used by the Node.

Syntax :

```
npm install express
```

To use express in the Node, follow the below syntax:

Syntax:

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

To uninstall the express package

Syntax:

```
npm uninstall package_name -g
```

Que: 4 Use of package.json and package-lock.json

package.json:

- The 'package.json' file is a critical component of Node.js projects and is used to manage various aspects of the project, including its metadata, dependencies, scripts, and more. It is a JSON (Javascript Object Notation) file that provides essential information and configurations for the project, making it easy to manage, share, and distribute Node.js applications.

Here are some common uses of 'package.json'.

1. Project Metadata
2. Dependencies
3. Dev Dependencies
4. Scripts
5. version Management
6. Dependency locking

- ex:

```
{
  "name" : "my-node-app",
  "version" : "1.0.0",
  "description" : "A sample Node.js application",
  "author" : "John",
  "license" : "MIT",
  "dependencies" : {
    "express" : "^4.17.1",
    "lodash" : "^4.17.21".
  },
  "devDependencies" : {
    "mocha" : "^9.0.0",
    "eslint" : "^7.32.0"
  },
  "scripts" : {
    "start" : "node index.js",
    "test" : "mocha",
    "lint" : "eslint ."
  }
}
```

package-lock.json :

- The 'package-lock.json' file is a crucial component in Node.js projects that use npm (Node package Manager) for managing dependencies. It was introduced in npm version 5.0.0 to ensure deterministic and reproducible builds. Its main purpose is to lock the version of every installed package, along with its dependencies, to prevent potential conflicts on

unexpected version upgrades when the project is deployed or shared with other developers.

- ~~How~~ Here's how 'package-lock.json' is used:
  1. Dependency Resolution
  2. Deterministic Installs
  3. Faster Installations
  4. Security and Integrity
  5. Continuous Integration (CI) and Deployment:

Que: 5 Node.js packages

- Node.js packages, also known as modules or libraries, are reusable blocks of code that extend the functionality of Node.js applications. These packages are typically distributed via the Node Package Manager (NPM) and can be easily installed and used in Node.js projects. They help developers save time by providing pre-built functionalities and solutions to common programming tasks.
- There are thousands of Node.js package available on the NPM registry, covering a wide range of functionalities, such as web frameworks, database connectors, utility functions, authentication modules, and more. Here are some popular Node.js packages:

PAGE NO.:
DATE: / /

1. Express.js :
- A minimalist web framework for building web applications and APIs. It simplifies the process of handling HTTP requests, routing, middleware, and more.

2. Lodash :
- A utility library that provides a large set of functions for working with arrays, objects, strings, and other data types. It helps with data manipulation and & handling.

3. Axios :
- A popular HTTP client that allows you to make HTTP requests from Node.js applications, enabling communication with external APIs.

4. MongoDB / Mongoose :
- MongoDB is a NoSQL database, and Mongoose is an Object Data Modeling (ODM) library that simplifies working with MongoDB databases in Node.js.

5. Socket.io :
- A real-time communication library that enables bidirectional event-based communication between the server and clients, commonly used for chat applications and real-time updates.

PAGE NO. :
DATE : / /

Que : 6 npm introduction and commands with its use.

- npm is a short form of Node Package Manager, which is the world's largest software registry. The open-source web project developers use it from the entire world to share and borrow package. The npm also acts as a command-line utility for the Node.js project for installing packages in the project, dependency management, and even version management.

Commands :-

1. npm init
   - This is used to create the package.json file.

2. npm ls : This is used to list the package or modules in your project together with their dependencies.

3. npm i <package> or npm install <package>
   - This is the command used to install a package.

4. npm un <package> or npm uninstall <package>
   - This is used to uninstall a package.

5. npm publish
   - This is used to publish a package.

6. npm run
   - This lists the available scripts that can be run on your project.

PAGE NO.:
DATE: / /

7. npm stop
- This is used to stop a script.

8. npm cache clean --force
- This force cleans the npm cache.

9. npm search <package>
- This is used to search for a package in your project folder.

10. npm view <package>
- This is used to show data about a package and print it to the screen.

Que : 7 Describe use and working of following Node.js packages also important properties and relevant programs.

(1) url :
- The url module slips up a web address into redable parts.
- To include the url module, use require() method,

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

- 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:8080/default.html?
          year=2017&month=february';

var q = url.parse(adr, true);

console.log(q.host);

console.log(q.pathname);

Console.log(q.search);

var qdata = q.query;

Console.log(qdata.month);
```

(2) process, pm2 (external package):

- pm2 (process manager 2) is a popular, production-ready process manager for Node.js application. It helps you manages and deploy your node.js applications, ensuring high availablity, reliablity and easy scaling.
- pm2 provides features like process monitoring, autometic restarts, load balancing and more, making it a great tool for manging node.js app in production environment.

→ use of pm2 :

→ Process management :
pm2 can stop, start and restart process easily, ensuring that your Node.js applications runing continuosly.

→ load balancing :
pm2 allows you to run multiple instances of your applications to distribute incoming request.

→ Autometic restart :
pm2 can autometically restart your application if it crashes or encounters errors.

(3) readline :
- Readline module in node.js allows the reading of input stream line by line. This module wraps up the process standards output and process standard input objects.
- To module makes it easier to input and read the output given by the user. To use this module, create a new javascript file and write the following code at the start of application :

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

- readline module comes with different methods to interact with user.

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

let rl = readline.creatInterface
        (process.stdin, process.stdout);
```

- Here createInterface() method takes two arguments. The first argument will be for standard input and 2nd for standard output.
- we can also use setprompt() method is used to set the particular statement to the console.

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

Const rl = readline.CreateInterface (process.stdin,
                                     stdout);

rl.setprompt ('What's your age?');

rl.prompt();

rl.on ('line', (age) => {
    Console.log ('Age received by the user: ${age}');

rl.close();

});
```

PAGE NO.:
DATE: / /

(4) fs :

- Node.js file system allows you to coork with the file system on your computer.

- var fs = require ('fs');

- Commem use of file system module :
  - Readfile
  - createfile
  - updatefile
  - Deletefile
  - Renamefile

→ Read files :

The fs.readfile() method is used to read files on your computer.

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

var fs = require ('fs');

http.createServer (function (req, res) {
    fs.readfile ('demo.html', function (err, data) {
        res.corite (data);
        return req.end();
    });
}).listen (8000);
```

→ Create file :

```
var fs = require ('fs');
fs.appendfile ('file1.txt', 'Hello Content',
        function (err) {
    if (err) throw err;
    console.log ('solved');
});
```

(5) events :

- To include the buit-in Events module use the require() method, In addition, all event properties and methods are an instance of an eventemitter object.

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

var eventEmitter = new event.EventEmitter ();

var myeventHandler = function () {
    console.log ('I have a Screen');
}
eventEmitter.on ('Screen', myEventHandler),
event Emitter.emit ('Screen');
```

(6) console :

- Node.js console module object that provides a simple debugging console similar to JavaScript to display different levels of message.

→ example :

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

const out = fs.createWriteStream ('./stdout.log');

const err = fs.createwritestream ('./stderr.log');

const myobject = neco console.Console (out, err);

myobject.log ('This is the first example.');
```

- global console : it is used without calling.
require ('console');

(7) buffer :

- The buffer in node.js is used to perform operations on raco binary data. generally, buffer refers to perticular memory location in memory. buffer and array have some similarities.

→ Some methods of buffer :

- Buffer.alloc (slie)
- Buffer.from (initialization)
- Buffer.write (data)

- tostring()

- Buffer.isBuffer(object)

- Ex:

```
Const buffer1 = Buffer.alloc(100);
Const buffer2 = Buffer.from([1,2,5]);
Const buffer3 = new Buffer('GFG');

buffer1.write('Happy learning');

Const a = buffer1.tostring('utf-8');
console.log(a);

Console.log(buffer1.length);

Const data = buffersDest.tostring('utf-8');

Console.data();
Console.log(data);
```

(8) querystring :

- querystring module provides a way of parsing the url query string.

- methods of querystring :
  - escape()
  - parse()
  - stringlyfy()
  - unescape()

- var querystring = require ('query String');
var q = querystring.parse ('year = 2017 & month = february');

console.log (q.year);

(9) http :

- Node.js has a built-in module called HTTP, which allows Node.js to transfer data over Hyper Text Transfer Protocol.

var http = require ('http');

- Use creatServer() method to create an http server.

- Ex :

```
var http = require ('http');
http.createServer (function (req, res) {
    res.write ('Hello world');
    res.end();
}).listen (8080);
```

→ Read the query string :

- function passed into the http.createServer() has a req argument that represent the request from the client, as an object.

```
var http = require ('http');
http.createServer (function (req, res) {
```

```
res.writeHead(200, {'content-type':
                    'text/html'});
res.write(req.url);
res.end();
}).listen(8080);
```

(10) V8 :

- V8 is a high-performance JavaScript engine developed by google and used in google chrome, the open-source browser from google. It was designed to improve the performance of web applications by compiling Javascript into native machine code rather than interpereting it.

→ some important components of the V8 javascript engine :

- garbage collection
- JS interpreter
- web Assembly

- garbage collector :
V8 javascript includes a garbage collector. it s free up the memory used by objects that are no longer needed.

- JS interpreter :
V8 ignition first interprets java-script code, which is bytecode interpreter, ignition needs the code and evaluates it, performing is not

as efficient as machine code.

- Web Assembly :
  web assembly is binary instruction format for a stack - based virtual machine.

(11) zlib :
- The zlib module provide a way of zip and unzip files.
- The syntax :

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

→ methods of zlib :-

Constants
createDeflate()
create Unzip()
create Gzip()

- var zlib = require ('zlib');

```
var fs = require ('fs');
var gzip = zlib.createGzip();
var r = fs.createReadStream ('./demofile.txt');
var w = fs.createWriteStream ('./mygzipfile.txt.gz');
r.pipe (gzip).pipe (w);
```