Name : Patel Rutvi M BN : ICI-3
Roll : 30 Sub : OSWD Submition date : 24/7/23

Assignment :- 1

1] Nodejs. introduction, features, execution architecture

→

→ Nodejs is an open-source cross-Platform Javascript run-time environment that executes Java-script code outside of a browser.

→ Nodejs was written and introduced by Ryan Dahl in 2009.

* Features :-

1] Extremely Fast :-

Node.js is built on Google Chrom's V8 Javascript Engine. So its library is very fast in code execution.

2] I/o is Astinchronous and Event Driven :-

All API's of Node.js library are astinchronous Ex:- 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.

3] Single threaded :-

Node.js follows a single threaded model with event looping.

(2)

4] Highly Scalable :-

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

5] No buffering :-

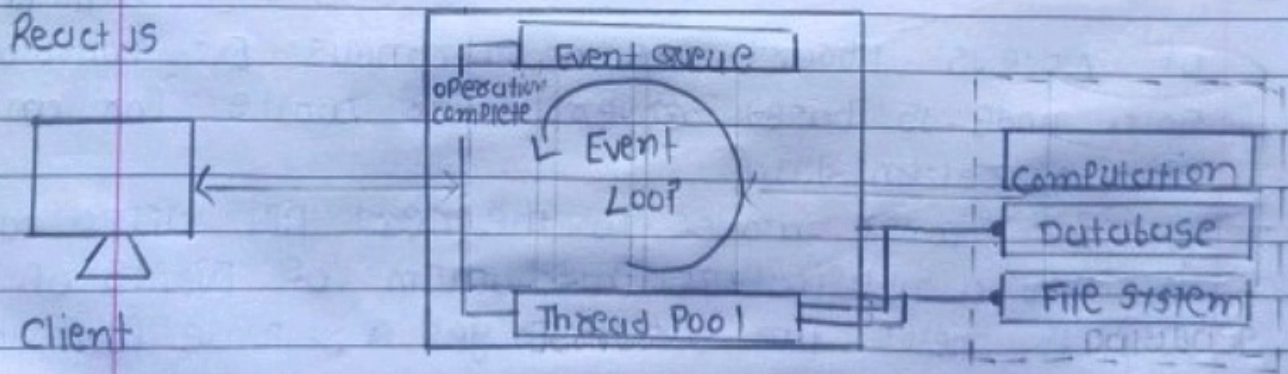
Node.JS cuts is down the overall processing time while uploading audio and video files.

→ Node.JS Applications never buffer any data.
→ These Applications simply output 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 Application. License released under the MIT license.

* Execution Architecture :-



(3)

1] Requests :-

incoming requests can be blocking (complex) non-blocking (simple) depending upon the tasks that a user wants to perform in a web application

2] Node.JS Server :-

Node.JS server is a server side platform that takes requests from users, process those requests and returns responses to the corresponding users

3] Event Queue :-

Event queue in a Node.JS server stores incoming requests and passes those requests one-by-one into the event loop

4] Thread Pool :-

Thread pool consists of all the threads available for carring out some tasks that might be required to fulfill client requests

5] Event Loop :-

Event Loop indefinitely receives requests and processes them, and then returns the responses to corresponding clients

6] External Resources :-

External resource are required to deal with blocking client requests. These resources can be for computation, data storage etc.

(4)

2] Note on node.js modules with example.

→

modules :-

A set of functions you want to include in your Application same as JavaScript libraries.

2 types of modules :- 1] Built-in-modules
2] External

1] Built-in-modules :-

NodeJS 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

Built-in http module :-

The HTTP module can create an HTTP server that listens to server 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)

* Creating Node.js Application

1] import Required module.

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

2] create server

```
http.createServer()
```

3] Testing Request & Response :

Now execute the main.js to start the server

```
$ node main.js
```

verify :- Server running at http://127.0.0.1:8081/

Utility modules :-

→ os Provides basic operating-system related utility functions

→ Path Provides utilities for handling and transforming file paths

→ net Provides both servers and clients as streams. Acts as a network wrapper

→ dns Provides functions to do Actual DNS lookup as well as to use underlying operating system name resolutionalities.

→ TLS stand for Transport Layer security. it is the successor to secure sockets Layer. TLS along with SSL for cryptographic Protocols to secure communication over the web.

→ 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
→ Url to Parse URL strings
→ Util To Access utility functions
→ Vs To Access information about Vs (the Javascript engine)
→ zlib to compress or decompress files
→ buffer to handle binary data.

3] Note of node.js Pakage with example.

→

npm provide two main functionalities :

1] online repositories for node.j Packages / modules

Command line utility to install node.js Pakages do version management and dependency management of node.js Packages

(7)

→ Modules are Javascript libraries you can include in your Project.
→ NPM command bundled with Node.js installation

1) Know version :- $npm --version
2) Syntax to install nodeJS modules :-
$ npm install <Module name>
3) Node.js web framework module express :-
$ npm install express

4) use this module in JS file :-
var express = require('express');

Global vs Local installation :-
By default installs
→ any dependency in the local mode.
Package installation in node_modules
→ dependency module Access using require() method.
→ it created node_modules directory installed the express module.

$ ls -l → list down locally installed modules.
total 0
drwxr-xr-x 3 root root 20 Mar 17 02:23 node_modules

→ Global Package Stored in system directory.
→ Package use in CLI Function of any node.js
$ npm install express -g
→ List :- $ npm ls -g
→ Futures :- $ npm i express --save
→ dev dev :- $ npm i express -D / npm i express --save-dev

(8)

→ Uninstalling : $npm uninstall express.
→ update : $ npm update express
→ Search : $ npm search express

4] use of Package.json and Package-lock.json :
→

1] Package.json :

→ It contains basic information About the Project
→ It is versioning file used to install multiple Pakages in Project.
→ it mandator for every Project
→ it records important metadata About Project
→ it contain name, description, author, script, dependencies
→ contains metadata About Project

Ex :

```
{
"name" : " Project name",
"version" : " 1.0.0 ,
"description" : "description",
"main " : " APP.JS",
"scripts" : {
        "test": "echo \" Error: no test specified \" && exit 1",
},
"Author" : "Author name",
"license": "Isc",
"dependencies" : {
            " dependency1 ": "^1.4.0",
            " dependency2 : "^1.5.2"
          }
```

3

(9)

2] Pakage-lock.json :

→ it created for locking the dependency with the installed version.
→ it describes the exact tree that was generated to allow subsequent installs to have the identical tree
→ it Automatically generate for those operation where npm modifies either node_modules tree or Package.js
→ it Allow devs future
→ it contain name, dependencies, locked version.

Ex:

```
{
  "name" : " Project name",
  "version" : "1.0.0",
  "lockFileversion" : 1,
  "requires" : true,
  "dependency" : {
                  "dependency" : {
                                  "version" : "1.4.0",
                                  "resolved" :
"https://registry.npmjs.org/dependency1/-dependency1-
1.4.0.tgz", "integrity" : "sha512-URL },
                  "dependency" : {
                                  "version" : "1.5.2",
                                  "resolved" :
"https://registry.npmjs.org/dependency2/-/dependency2-
1.5.2.tgz"
                                  "sha512-
                                  "integrity" : URL
                  }
  }
}
```

(10)

5] NPM commands introduction and commands with its use.

→
- it stand for Node Pakage Manager
- it default Pakage manager for node.js and it writen in JavaScript.
- Develop by Issac z Schlueter. Released in January 12, 2010.
- NPM mangge all the Pakages and modules for node.JS and consists of command line client npm.
- it required Pakages and modules in node Project are installed using NPM.
- npm install all the dependency of project through the Package.json.
- Pakage.json can specify a range of valid versions using the semantic versioning scheme.
- Allow developer to Autho update their Pakages while at the same time avoiding unwanted brecking changes.
- npm is open source
- Top npm Pakages in decreasing order are :
  lodash, async, react, request, express

* Command :

1] NPM : Java Script Pakage manager
2] npm Access : set access level on Published Pakages
3] npm adduser : Add register user Account
4] npm Audit : Run a security Audit
5] npm bugs : Bugs for a Pakage in web browser
6] npm install : install Pakage
7] npm install-ci-test : install Project with a clean slate and tests.

(11)

8] npm install-test : install Packages(s) and run tests
9] npm link : Symlink a Package folder
10] npm login : register user Account
11] npm logout :
12] npm ls : list of install Packages.

6] Describe use and working of following nodeJS Packages.

→

1] URL :

The URL module splits up a web address into readable 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=2018&month=Jun';

var q = url.Parse(adr, true);
console.log(q.host);
console.log(q.Pathname);
console.log(q.search);
var q.d = q.query
console.log(q.data, method);
```

→ 2] process, pm2 (external Pakage) :-

Pm2 (Process manages 2) is a Popular Production - ready process manages for node.js application deploy your nodjs application ensuring high easy scaling

→ Pm2 Provides features like Process monitoring autometic restarts, load balancing and more; & managing node.js app in Production enviourment.

use :-

Pm2 can stop, start and restart process easily, ensuring that your node.js applications runing continuosly.

1) load balancing :-

Pm2 allows you to run multiple instances of your application to distribute incoming request

2] Autometic Restart :-

Pm2 can autometically restart your application if it crashes or encounters errors.

* Readline :-

Readline module in node.js allowing the reading of input Stream line by line. This module wraps up the process standard output and process standard input objects.

To module makes it eagier 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 statment to the console.

```
const readline = require ('readline');
const rl = readline. (create Interface
        (process stdin, process. stdout );
rl. set prompt ('what's your age');
rl. Prompt ();
rl. on ('line', (age) => {
    console. log ('Age received by the
        user : ${age}');
rl. close ();
});
```

3] fs :

node.js file system allows you to work with the file system on your computer

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

comman use of file system module :

- Read file
- Create file
- update file
- delete file
- Rename file

• 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. write (data);
      return res. end ;
  });
}) / listen (8080);
```

• Create File :

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

```
fs.appendFile ('File.txt', 'Hello content',
                function (err)
{
  if (err) throw err;
  console.log ('Solved');
});
```

3] event :

To include the built-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);
eventEmitter.emit ('screen');
```

4] Console :

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

Ex :

```
Const fs = require ('fs');
Const out = fs.createwrite stream ('./stdout.log');
Const err = fs.createwrite stream ('./stderr.log');
const myobject = new console.Console (out, err);
myobject.log ('This is the First example');
```

global console :- it is used without calling

require ('console');

5] Buffer :-

The buffer in node.JS is used to perform operations on raw binary data. generally, buffer refers to particular memory location in memory buffer and array have some similarities

Some methods of buffer :

Buffer.all (File
Buffer.from (initialization)
Buffer.write (data)
tostring ()
Buffer.isBuffer (object.)

Ex :-

```
const buffer1 = Buffer.allo (100);
const buffer2 = Buffer.from ([1, 2, 3]);
const buffer3 = new Buffer ('GFG');

buffer.write ('Happy learning');
const a= buffer1.tostring ('utf-8');
console.log (a);
console.log (buffer1.length);
console.log
const data = buffer.Dest.tostring ('utf-8');
console.data ();
console.log (data);
```

- Query String :-

Querystring module provides a

Way of Parsing the url query string

methods of querystring :

escape ()
Parse ()
StringHTF1 ()
unescape ()

```
var querystring = require ('query string');
var q = querystring.Parse ('year = 2018 & month =
                            March');
console.log (q.year);
```

6) http :-

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

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

→ use createserver () method to create an http server

Eg :-

```
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);
```

7] V8 :-

V8 is high-performence 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 interpreting it

- Some important components of the V8 JavaScript engine :

- garbage collection
- JS interpter
- web Assembly.

- garbage collector :

V8 JavaScript includes a garbage collector it free up the memory used by objects that are no longer needed.

- JS interpreter :

V8 ignition first interprets JavaScript code, which is bite interpreter, ignition reads 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

8] Zlib :

The zlib module Provide a way of zip and unzip files

Syntax :

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

- methods of zlib :

constants
create.deflate ()
createGunzip()
createUnzip()

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

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