Name:- Chabhadiya Nayan P. Roll No : 07
Sub : Open Source Web Development
Submition Date: 24-07-2023
PAGE NO. 01
SEM :- MSC(ICT) SEM : 3
Assignment : 01

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 :-
NodeJS is built on google chrom's V8 Javascript engine so its library is very fast in code execution

2) I/O is Asynchronous and Event Driven :
All Api's of nodeJS library are asynchronous ex: non-blocking so a nodeJS 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 nodeJS helps the server to get a response from the previous Api call.

Scanned with CamSc

3) single thread.:
Nodejs follows a single thread model with event looping.

4) Highly Scalable:
Node.js highly scalable because in a non-blocking helps the server to respons in a non blocking way.

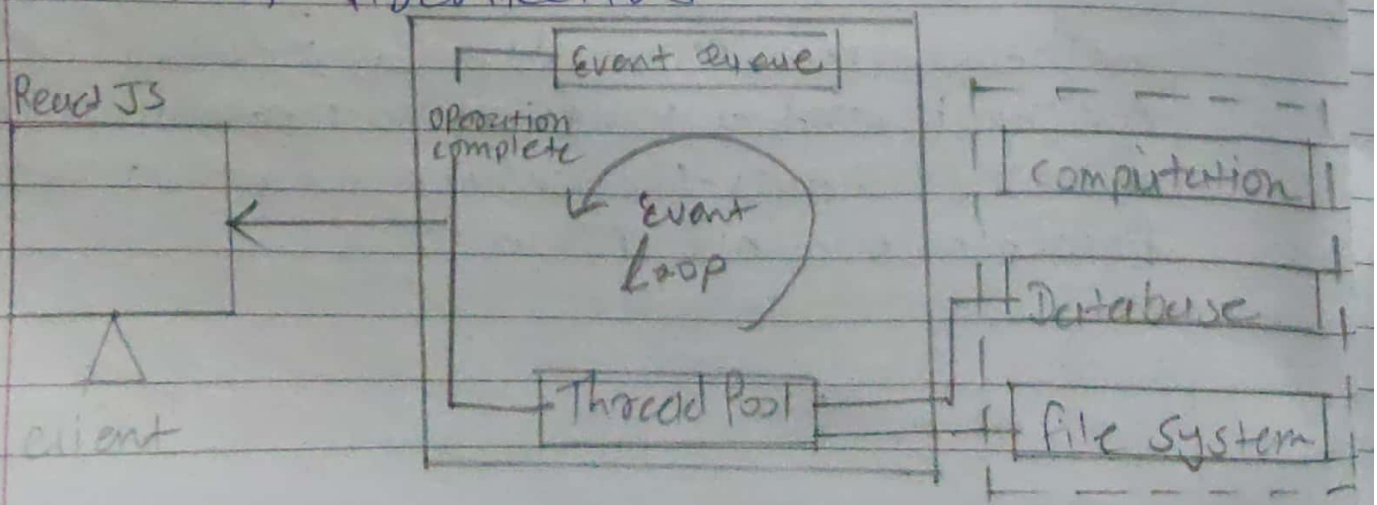
5) No buffering:
Nodejs cuts is is down the overall processing time while uploading audio and video files

-> Nodejs Applications never buffer any data

-> These applications simply output data in chunks

6) Open source:
Nodejs has an open source community which has produced many exellent modules to add additional capabilities to nodejs application License released under the MIT license.

* execution Architecture



Scanned with CamSc

1) Requests:

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

2) Nodejs server:

Nodejs server is a serverside platform that takes requests from users process those requests and returns response to the corresponding

3) Event queue:

Event queue in a nodejs 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 carrying out some tasks that might be required to full fill client requests.

5) Event Loop:

Event loop identifies receives requests and processes them, and then returns the response to corresponding clients

6) External Resource:

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

Scanned with CamSc

2) Note on nodejs modules with example.

=> modules:
A set of functions you want to include in your app same as JS libraries.

Types of Modules: 1) Built-in modules
2) External.

1) Built-in modules.
Nodejs has a set of build 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 app has access to the HTTP module and is able to create servers

Built-in http module:
The http module can create an http server that listen to ports and give O/P

-> Use the create server() method to create an http server,

-> The function passed into the http server() method, will be able to access to computer on port 8080 or any you want

Scanned with CamSc

→ If the response from the http server is suppose to be display as html we should include an http header with the correct content the

→ 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.)

creating app.

1) import Required module
var http = require("http");

2) Create server
http.createServer();

3) Test req & res
now execute the main.js to start server
$ node main.js
verify: server at http://localhost:8000/

Utility modules:

→ os provides basic operating-system related utility functions.

→

→ path provides utilities for handling and transforming file paths

Scanned with CamSc

→ TLS stand for Transport layer security it secure socket layer TLS along with SSL

=> querystring to handle URL query strings

=> stream to handle streaming data

→ string decoder To decode buffer objects into strings

=> Url to parse url strings

→ Util To Access utility functions.

→ zlib to compress or decompress files.

→ buffer to handle binary data

3) Note of Nodejs Package with example

=> NPM provide two main functionality

1) online repositories for nodejs packages

→ Command line utility to install nodejs packages do version management and dependency management of nodejs packages.

Scanned with CamSc

→ module are Javascript libraries you can include in your project.

→ 1) know version: npm --version

2) Syntax to install: npm i <package>

3) nodejs framework: npm i express

4) nodejs web: var express = require('express');

Global vs Local install

By default installs any dependency in the local mode.

→ package installation in node_modules

→ dependency module access using require()

$ ls -l - list down modules
total 0
drwxr-xr-x 3 root root 20 Mar 20 07 2023

→ Global Package stored in system directory

→ package we in cd function
$ npm i. express -g

List :- $ npm ls -g
features : npm i express --save
dev dev : npm i express -D
npm i express --save-dev

Scanned with CamSc

-> 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 package

=> it mandatory for every project

-> it records important data
-> it containing name description etc
-> contains metadata about project

Ex :

```
{
"name": "Project1",
"version": 1.0.0,
"description": "D",
"main": "APP.JS",
"script": {
        "test": "echo 1 'error: ' && exit 1"
},
"Author": Author name",
"licence": "ISC"
"dependency": {"dependency1": "^1.4.0",
}
```

Scanned with CamSc

2) Package-lock.json

-> it created for locking the dependency with the installed version

-> it describe the exact tree that generated

-> It allow devs future

-> It contain name dependencies, locked version

Ex:-

```
{
  "name": "Proj-1",
  "version": 1.0.0
  "lockfileversion": 1,
  "requires": true,
  "dependency": {
        "dependency": {
                version: "14.0"
"resolved" "http://dependency url.:"sha512"
  "dependency": {
        "version": "1.3.2",
  "resolved": http://url",
  }
}
}
```

Scanned with CamSc

Q) NPM commands introduction and commands with its use

-> it stand for node Package Manager

-> it default package manager for node and it written in JS

=> Develop by Isaac Schlueter Reload at 12-01-14.

-> npm manage all the package and modules for nodeJS and consists of command line client npm.

=> npm install all the dependency of project

-> Package.json can specify a range of valid version using the Semantic versioning

=> Allow developer to Auto update their package while at the same time avoiding unwanted breaking change.

-> npm is open source

-> Top npm package in design
lodash, async, react, request, express

command
1) npm = Package manager
2) npm Access = set access level
3) npm adduser = Add user account

Scanned with CamSc

4) npm audit : Run a security
5) npm bugs : Bug for package.
6) npm i : install package
7) npm i --ci --test: install project as tests.
8) npm i --test = run and install tests
9) npm link : link package
10) npm login : register and Register
11) npm ls : list packages.
12) npm logout:

6) Describe use and working of following nodejs packages.

1) URL:
The URL module of a web address into readable parts.
=> To include the URL module use require() method.

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

parse the address the URL module use require() method and it will return a URL object with

```
var url = require('url');
var adr = 'http://localhost:8080/default.html?year=2018&month=Jan';
var q = url.parse(adr, true);
console.log(q.host);
var q.d, i.d = q.query;
console.log(data, method);
```

Scanned with CamSc

* 1) process. pm2 (external Package):

pm2 is a popular Production reality process manager for nodejs application deploy your node app.

-> pm2 provides feature like Process monitoring automatic restarts, load balancing and more managing nodejs ap in production environment.

Use:

pm2 can stop, start and restart process easily ensuring that your Node.js application running continously.

1) load balancing:

pm2 allow you to run multiple instance of your app to request

2) Automatic Restart:

pm2 can automatically restart your app if any changes done

* read line:

readline module in nodejs allowing the reading of input stream line by line this module wraps up the process standard o/p and process standard input objects.

=> To module makes it easier to input and read the output given by the user to use this

Scanned with CamSc

module create a new js file and write the following code.

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

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

-> const readline = require("readline");
let rl = readline.createInterface(process);
Here createInterface() method takes two arguments.

-> we can also use setPrompt() method it used to set the particular statement to the console

```
const readline = require('readline');
const rl = readline.createInterface(
    process.stdin, process.stdout);
rl.setPrompt('whats age of you?');
rl.prompt();
rl.on('line', (age) => {
    console.log('Age received by the
        user: ${age}');
rl.close();
});
```

Scanned with CamSc

3) fs :

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

var fs = require ('fs')
common use of file system module.

- Read file
- create file
- update file
- Delete file
- Rename file

Read file :

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

```
var Http = require ('HTTP');
var fs = require ('fs');
http create server (function (req, res) {
    fs. read file ('demo.html', function (err, data) {
        res.write(data)
        return res.end();
    });
}).listen(8080);
```

create file:

```
var fs = require ('fs');
fs.appendfile ('file.txt', "Hello") function
    (err).
```

Scanned with CamSc

```
});
    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 event emitter object

```
var events = require ('events');
var eem = new events.EventEmitter();
var mcH = function() {
    console.log("I am");
};
em.on("scream", mcH);
em.emit('scream');
```

4) console:

Nodejs console module object that provides a simple debugging console similar to JS.

Ex

```
const fs = require ('fs');
const out = fs.createWriteStream("std.log");
const err = fs.createWriteStream
const myObject = new console.Console(
.out, err);
myObject.log("first");
global.console: it calling without
require('console').
```

Scanned with CamSc

⇒ buffer

The buffer in nodejs is used to perform operations on raw binary data generally buffer refers to particular memory.

Some methods of buffer

Buffer.all (Alc)
Buffer.from (initialization)
Buffer.write (data)
toString ()
Buffer.isBuffer (object)

Ex

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

buffer.write("Hello");
const a = buffer1.toString('Utf-8');
console.log(a);
console.log(buffer1.length);
const data = bufferObj.toString('utf8');
console.log(data);
```

query string:

querystring module provides way of parsing the URL query string.

method of querystring:

escape()	stringify()
parse()	unescape()

Scanned with CamSc

```
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');
```

→ We createServer() method to create http server

Eg:

```
var http = require('http');
http.createServer(function (req, res)
{
    res.write("Hellow world");
    res.end();
}).listen(8000);
```

* 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)
{
```

Scanned with CamSc

```
res.writeHead(200, {"Content-Type"
                : "text/html"});
req.write(res, req);
res.end();
}).listen(8000);
```

7) V8:

V8 is high-performance JS 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

- Some important component of the V8 JS. engine:

- garbage collection
- JS interpreter
- web Assembly

• garbage collection

V8 JS includes a garbage collections it free up the memory used by objects that are no longer needed

• JS interpreter:

V8 ignition first interprets JS code which is bite interpreter ignition reads the code and evaluates it performing is not as efficient as machine code.

Scanned with CamSc

· web Assembly

web assembly it binary instruction format for a stack-based virtual machine

8) zlib:

The zlib module provide a way of zip and unzip file

Syntax:

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

· methods of zlib:

· constants
create Deflate()
create Gunzip()
create Gzip()

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

Scanned with CamSc