Theory Assignment - 1

Name - Vinita . K . Patel

Roll . No - 61

Div - A

Semester - 7th

Subject - 701 / Application development using full stack.

Q-1 Nodejs : Introduction , features , execution architecture

i Introduction :-

- Node.js is an open source server environment
- Free
- Runs on various platforms (Windows , Linux , Unix , Mac , OS etc)
- Uses Javascript on the server.

- Node.js uses asynchronous programming

- A common task for a web server can be to open a file on the server and return the content to the client.

ii Features :-

- single thread
- Asynchronous
- Open source
- Event driven
  - similar to callback function
- Highly scalable
- No Buffering
- NPM (Node Package Manager)
- Performance

Execution Architecture :-

- Node.js is an event-driven, non-blocking I/O architecture

- It uses an event loop to handle asynchronous I/O operation and can handle a large number of concurrent connection with high throughput and low latency

- Node.js simply enters the event enters the event loop after executing the input script

- Node.js exits the event loop when there are no more callback to perform.

- This behaviour is like browser javascript the event loop is hidden from the user

- HTTP is a first class citizen in Node.js.

Q-2 Node.js Modules with Example.

→ Module - A set of function you want to include in your application

- 3 Types of Modules :-

i Core module
ii Local module
iii Third-party module

i Core module :-

- Node.js has many built-in modules that part of the platform.

- These modules can be loaded into the program by using the required function.

Syntax

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

The require() function will return a Javascript type depending on what the particular module return.

Example :-

```
const http = require ('http');
http.createServer (function (req, res) {
    res.writeHead (200, {'content-Type': 'text/html'});
    res.write ('welcome');
```

```
res.end();
}).listen(3000);
```

Core Modules	Description
- http	creates an HTTP server in Node.js.
- fs	used to handle file system
- path	includes methods to deal with file paths.
- os	provides information about the operating system

ii) Local Modules :-

- Unlike built-in and external modules, local modules are created locally in your Node.js application.

Filename : calc.js

```
exports.add = function (x,y) {
    return x+y;
};
exports.sub = function (x,y) {
    return x-y;
};
exports.mult = function (x,y) {
    return x*y;
};
```

```
exports.div = function (x,y) {
    return x/y;
};
```

Filename index.js

```
const calculator = require ('./calc');
let x = 50, y = 10;
console.log (" Addition of 50 and 10 is "
    + calculator.add (x,y));
console.log (" Subtraction of 50 and 10 is "
    + calculator.sub (x,y));
Console.log (" Multiplication of 50 and 10 is "
    + calculator.mult (x,y));
console.log (" Division of 50 and 10 is "
    + calculator.div (x,y));
```

iii) Third-party Module :-

- are module that are available online using the Node Package Manager (NPM)
- These modules can be installed in the project folder or globally
- Some of the popular third-party module are Mongoose, express, angular and React.

Example :-

```
npm install express
npm install mongoose
npm install -g @angular/cli
```

Note on package with example.

The package.json file is the heart of Node.js system. It is the manifest file of any Node.js project and contain the metadata of project.

→ Identifying metadata properties :-
- It basically consist of the properties to identify the module project such at at same time.

- Functional metadata properties :-
As the name suggests it consist of functional value of a project such as the starting point of module.

-5 Node.Js package :-

→ also known as npm packages are module or library written in Javascript that extended the functionality of Node.js application.

- Node.js package are essential for building scalable, efficient & feature rich application since they allow developers to resise code & avoid error.

Some popular node js package :-

1) Express
- A fast & minimalist web application

framework that simplifies building we
server & APIs.

2) Lodash :-
- a wide range of helper function fo
working with array, object, string
& more.

3) Mongoose :-
An object data modeling library that
provide an easy way to interact wit
Mongo DB database.

4) Moment.js :-
A library for parsing validating
manipulating & formatting dates and
time.

5) Multer :-
A middleware between / for handling
file upload in node.js application.

6) Socket.io
A lib library for real-time bidirectio
communication between the server & cli
using web socket.

Example :-

```
Const express = require ('express');
Const app = express()

app.listen (3000,() => {
  console.log ('server Started on port 300
});
```

Q4 Use of package.json and package-lock.json

→ i Every npm package and Node.js project has a package.json file with meta-data for a project.

The file resides in the root directory of every node.js package and appears after running the npm init command.

1. → The package.json file contain descriptive and functional meta-data about a project such as a name, version and dependencies

2 → The file provides the npm package manager with various information to help identify the project and handle dependencies.

3 → The package.json file is fully customizable and looks different for every project.

```
{
    "name": "example-name",
    "version": "1.0.0"
}
```

ii package-lock.json

is automatically generated for any oper where npm modifies either the node_modules tree, or package.json

It describes the exact tree that was generated, such that subsequent install are able to generate identical trees, regardless of intermediate dependency updates.

6 npm introduction and command with its use

→ NPM - Node package Manager

It allows for seamless node.js package management

npm consist of 3 component

1. Website
2. Registry
3. CLI

- NPM is already ready to run in computer
- NPM is known as the world's largest software Registry.

1. Website :-

It allows you to find third-party package set up profile and manage your package

2. Registry :-

It is a large Public database of Javascript code.

3. CLI (command line Interface)

It is runs from a terminal to allow you to interact with npm

It helps for installing, updating and uninstalling packages and managing dependencies.

NPM command :-

i) npm init - Using to initialize a project

ii) npm init - using to instantly initialize a projec

iii) npm install - Using to install module

iv Package.json - Using to install module and save them to your package.json as a dependency.

→ Installing a Module using npm

- following is the syntax to install :-

-npm install <Module name>

Lets install a famous node.js we frame - work called express.

Open the Node.js command prompt & execute it.

→ Uninstalling a Module :-

* npm uninstall express

- To node.js module is uninstalled you can verify by using following command.

1) npm Is
- you can see that the module is empty now.

→ searching module :-

"npm search express" command is used to search express or module.

1) npm search express.

Q-7 Describe use & working of following

1) Url

- The Url module splits up a web address into readable parts.
- To include the URL module, use the require() method

Syntax var url = require('url');

Example :-

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

http.createServer(function(req,res){

var querystring = url.parse(req.url, true);
console.log(querystring);
}).listen(4200);
```

Properties

1. 'url.href' - A string representing the full URL
2. 'url.protocol - The protocal scheme of the URL
3. 'url.host' - The hostname of the URL.
4. 'url.port' - The port number of the URL

Process :-

Is a built in global object in Node.js that provides information and control over the current Node.js process.

It allow you to interact with current process, access command-line argument.

Properties :-

1. 'process.env' - An object containing the environment variable
2. 'process.pid' - The process id of current Node.js process.
3. 'process.argv' - An array containing the command line argument passed to process

Method :-

1. 'process.exit' - Use to terminate the code
2. 'process.on' - method allow to register event handler.
3. 'process.nextTick' - Method allow or add a callback function to next tick queue

```
process.on ('before exit'. (code => {
console.log ('Process before exit event
                with code:', code);
});

process.on ('exit', (code) => {
    console.log ('exit', code);
});

console.log ('this message is displayed first
```

iii) Readline

It provide an interface for reading data from a Readable stream one line at a time.

Use the 'require' keyword to import the file

```
var calculator = require ('./calc');
var x = 50, y = 20

console.log (" Addition " + calculator.add (x, y));
console.log (" Subraction " + calculator.sub (x, y
console.log (" Multiplication" + calculator.Mul (x,
Console.log (" Divison" + calculator.div (x,
```

iv FS (file system)

- The node fs module enables interacting with the file system in a way on standard POSIX function.

Syntax - import as fs from node:fs / promises;

const fs = require ('fs');

```
fs.readFile ('example.txt', 'utf8', (err, data) => {
  if (err){
     console.error ('Error reading the file:', err);
  }
   else {
     console.log ('File content :', data);
  }
});
```

v Events

The events module provide a way of working with event

Example :-

```
var event = require ('events');
var event emitter = new events.EventEmitter();

eventEmitter.on ('Scream', function() {
console.log ('A scream is deleted !');
});
Event Emitter.emit ('scream');
```

vi Console

The console class can be used to create a simple logger with configurable output stream & can be accessed using either require ('node: console'). console or console. console.

syntax

```
const { console } = require ('node:
                              console');
```

vii Buffer

- Buffer object are used to represent a fixed - length sequence of byte.
- Many Node.js API support buffer.

```
import {Buffer} from 'node:buffer';
const buf1 = Buffer.alloc (10);
const buf2 = Buffer.alloc (10, 1);
const buf3 = Buffer.allocUnsafe (10);
const buf4 = Buffer.allocFrom ([1,2,3]);
const buf6 = Buffer.From ('test');
const buf7 = Buffer.From ('test', latin');
```

viii) Query string

The node: query string module provides utilities for parsing formatting URL query string.

It can be accessed using

```
const querystring = require ('node:
                              querystring');
```

IX HTTP

- To use the HTTP server & client one must require ('node:http');
- HTTP message headers are represent by an object like this :

```
{
  'content-length' : '123';
  'content-type' : text/plain;
  'connection' : 'keep-alive';
  'host' : 'example.com';
  'accept' : '*', }
```

X V8

Node.js is reffered to as a runtime environment since it contains everything you need to run a Javascript program

This V8 engine is at the heart of node.js.

```
const v8 = require ('v8');
const sampleObject = {
  name: 'John Doe',
  age: 30,
```

```
email: 'john.doe@example.com',
};
const serializedData = v8.serialize(sampleObject);
console.log('serialized Data:', serializedData);
```

xi OS

The node os module provide operating system-related utility method & properties It can be accessed using

```
const os = require('node:os');

const homeDir = os.homedir();
Console.log('Home Directory:', homedir);
```

xii zlib

The node zlib module provide compression functionality implemented using zlib deflate / Inflate & Brotli

```
const zlib = require('zlib);
const inputData = 'welcome here';

zlib.deflate(inputData, (err, compressedData)=>{

if(err){
  console.error('compression Error:', err);
}
  else{
  console.log('compressed Data:', compressed
                                    Data);
```