Assignment - 1

Name : Shrushti C. Pastagiya

Roll No : 043

M.SC (IT) 7th Sem

Sub : Application Development using Full Stack

Code : 701

Date : 21st July, 2023

PAGE NO.
DATE :

Que : 1

(1) Nodejs : Introduction, features, execution architecture.

⟹

- Introduction :

Node.js is an open-source and cross-platform runtime environment for executing Javascript code outside a browser. You need to remember that NodeJS is not a framework and it's not a programming language. Most people are to use Node.Js for building back-end services like APIs like Web App or Mobile App. It's used in Production by large companies such as Paypal, Uber, Netflix, Walmart, and so on.

Node.js = RunTime Environment + Javascript Library

- Features :

There are other programming languages also which we can use to build back-end services so what makes Node.js different.

1. It's easy to get started and can be used for prototyping and agile development.
2. It provides fast and highly scalable services.
3. It uses Javascript everywhere, so it's easy for a Javascript programmer to build back-end services using Node.Js.

4. Source code cleaner and consistent.
5. Large ecosystem for open source library.
6. It has Asynchronous or Non-blocking nature.

- Architecture :

Node.js is a Javascript-based platform that is mainly used to create I/O-intensive web applications such as chat apps, multimedia streaming sites etc. It is built on google chrome's V8 javascript engine. A web Application is software that runs on a server and is rendered by a client browser that accesses all the application's resources through the internet.

To manage several concurrent clients, Node.js employ's a "single Threaded Event loop" design. The Javascript event-based model and the Javascript callback mechanism are employed in the node.js Processing Model. It employs two fundamental concepts :

1. Asychronous Model
2. Non-blocking of I/O operations

* Components of the Node.js Architecture :

- Requests :

① Depending on the actions that a user

needs to perform, the requests to the server can be either blocking or non-blocking.

- Node.js Server :

The node.js server accepts user requests, processes them, and returns results to the users.

- Event Queue :

The main use of Event queue is to store the incoming client requests and pass them sequentially to the Event loop.

- Thread pool :

The Thread Pool in a Node.js server contains the threads that are available for performing operations required to process requests.

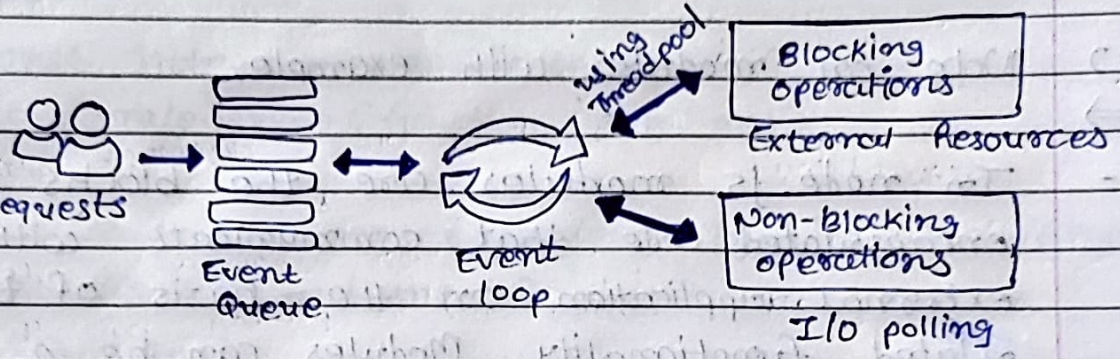
- Event loop :

Event loop receives requests from the Event Queue and sends out the responses to the clients.

- External Resources :

In order to handle blocking client requests, external resources are used. They can be of any type.

* Workflow of Node.js Server :



- Users send requests to the server for performing operations.
- The requests enter the Event Queue first at the server-side.
- The Event queue passes the requests sequentially to the event loop. The event loop checks the nature of the request.
- Event loop processes the non-blocking requests which do not require external resources and returns the responses to the corresponding clients.
- for blocking requests, a single thread is assigned to the process for completing the task by using external resources.
- After the completion of the operation, the request is redirected to the Event loop which delivers the response back to the Client.

PAGE NO.
DATE :

Que : 2

(2) Note on 'modules with example.

⟹

- In node.js modules are the blocks of encapsulated code that communicate with an external application on the basis of their related functionality. Modules can be a single file or a collection of multiple files/folders. The reason programmers are heavily reliant on modules is because of their reusability as well as the ability to break down a complex piece of code into manageable chunks.

- Modules are of three types:
  - Core Modules
  - local Modules
  - Third-Party Modules

- Core modules :-

  Node.js has many builtin modules that are part of the platform and come with node.js installation. These modules can be loaded into the program by using the required function.

Syntax :

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

* Program :

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

- Local Modules :-

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

* Program : (Calc.js)

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

(index.js)

```
const calculator = require('./calc');
let x = 50, y = 10;
Console.log(" Addition of 50 and 10 is "
+ calculator.add(x, y));
```

PAGE NO.
DATE :

- Third-party modules :

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

example :
- npm install express
- npm install mongoose
- npm install -g @ angular/cli

Que : 3

(3) Note on package with example.

⟹

- In node.js, a package refers to a collection of code. Usually in the form of modules that is organized and distributed together. Packages can includes various functionalities, utilies and libraries and they are managed using the node Package manager (NPM).

- Npm is a pocoerful tool that allocos developers to install manage and share packages easily. It comes

bundled with Node.js. So, there's no need to install it seperately. npm maintains a vast repository of public packages and it also allows developers to publish their own packages for others to use.

- Installing Packages:

After creating the project, next step is to incorporate the packages and modules to be used in the Node project. To install packages and modules in the project use the following syntax:

```
npm install package_name
```

- 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:

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

* Program:

```
const express = require ('express');
const numbers = [1,2,3,4,5];
const sum = express.sum(numbers);
const person = {
    name : 'william',
    age : 20,
    city : 'New york'
};
const keys = express.keys(person);
```

Que : 4

(4) Use of package.json and package-lock.json.

⟹

- Package.json :

The 'package.json' file is a JSON file that contains metadata about your node.js. Project and its dependencies. It servers as a manifest for the project and includes information such as the projects name, version, author, license, scripts, And most importantly the list of dependencies required to run the project.

- The 'package.json' file is typically located at the root of your project directory. It can be created manually or generated using the npm init command.

- package-lock.json:

The package-lock.json file is automatically generated by NPM when you install or modify dependencies in your projects. It is designed to provide a detailed and deterministic record of the exact versions of packages that are installed in the 'node_modules' directory.

- This file ensures that all developers working on the project use the same version on dependencies, preventing potential conflicts or unexpected behavior due to version mismatches.

Que: 5

(5) Node.js Packages.

⟹

1. Express:

Express.js is the fastest, unopinionated and simplest web framework for Node.js. It was flexibly created to build simple pages, multi-pages and hybrid apps with robust features for web and mobile development. The framework is small with undisputed performances benefits. It is the leading Node.js framework designed to help you build web apps and APIs with powerful tooling for HTTP servers.

2. Lodash :

Lodash is a modern Javascript library that provides utility functions. Lodash is inspired by the famous underscore.js utility library. Lodash has built-in functions that make node.js coding easier and cleaner. Instead of writing a common function repeatedly, you can just a single line of code with the help of lodash.

3. Moment.js :

Moment.js is a 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. Axios performs requests such as GET, POST, DELETE and PUT.

5. Karma :

As a Node.js developer, you need to test your application to make sure it is stable, reliable

and showcasing good performance. Karma is the tool for this job. The main aim of karma is to provide developers a productive testing environment.

6. AsyncTS :

Asynchronous is heavily used in node.js to ensure a non-blocking operations flow. Asynchronous I/O permits other processing to continue even before the first transmission has finished.

7. Nodemon :

Nodemon 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.

Que: 6

(6) npm introduction and commands with its use.

→

- npm stands for node package manager. It allows for seamless node.js package management. You can install, share and manage node.js packages.
- npm consists of three components:
  1. Website
  2. Registry
  3. CLI

* Basic npm commands:

- Installing packages:-
  1. locally:
     A locally installed package can be accessed only on the folder you've downloaded it.
- npm install <package_name> --save-dev

2. Globally:

A globally installed packages works anywhere on the machine. To install global packages you've to use -g flag.

- npm install -g node-static

- Updating packages :

since we have installed packages sometimes we need to update our packages to get new features. To do that, you've to use

- npm update <package_name>

for specific package,

- npm update, to update all packages.
- for global packages, you've to use -g.

npm update <package_name> -g

- Uninstalling packages :

Sometimes you don't need a particular package and you want to remove it. It's not a good idea to manually remove the package from the node_modules folder as it can be a dependency for the other packages. So, to safely remove a package you've to use the command.

- npm uninstall <package_name>

for global package,

- npm uninstall <package_name> -g

- list of installed package :

To get the list of installed package, use the command

- npm list

- Installing from package.json :
  If you want to share your project then you may not want to share all your node modules. To do that, you have to use command,
  - npm install

- getting help :
  npm CLI has built-in help command. You can access it by
  - npm help

  for particular command,
  - npm <command> -h

  you can also search npm documentation for help,
  - npm help-search <command>

Que : 7

(7) Describe use, properties, working, method and relevant program of following. Nodejs.

⟹

1. Url :
   The 'URL' module provides utilities for URL resolution and parsing.

Property:

- href - provides us the complete url string
- host - Gives us host name and port number
- hostname - hostname in the url
- path - gives us path name of the url
- pathname - provides host name, port and pathname
- port - gives us Port no. specified in url
- auth - authorization part of url
- protocol - Protocol used for the request
- Search - Returns query string attached with url

Program:

```
const url = require('url');
const urlobject = {
    protocol : 'https:',
    hostname : 'www.example.com',
    port : '8080',
    pathname : '/path/to/resource',
    query : { id : '123' },
    hash : '#section'
};
const formattedUrl = url.format(urlobject);
console.log(formattedUrl);
```

2. process, pm2 (external package):

The process module is used to interact with the node.js process, get info. about the runtime environment & handle events & signals.

property:

process.argv - An array that contains command-line arguments passed to the node.js process.

process.env - An object representing the environment variables of the current process.

process.cwd() - Returns the current working directory of the Node.js process.

Program:

```
console.log(process.env.NODE_ENV);
```

3. Use of pm2 (Process Manager 2):

pm2 is used to manage and monitor node.js application in production to ensure they stay online, stable & efficiently use system resources.

Property:

npm install pm2 -g : install pm2 globally
pm2 start app.js : start a node.js application with pm2
pm2 list : list running processes
pm2 logs - monitor logs
pm2 reload app.js - reload application.

4. readline:

The primary use of the readline module is to read input from users or other readable streams and process it line by line.

Properties:

readline.createInterface - This method creates a new 'readline.Interface' instance.
resume() - Resumes the input stream after it has been paused.
close() - close the readline.Interface, releasing any associated resources.

Program:

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

PAGE NO.
DATE :

```
const rl = readline.createInterface({
    input : process.stdin,
    output : process.stdout
});
rl.question('What is your name?', (name) =>
    { console.log(`Hello, ${name}!`);
      rl.close();
});
```

5. FS :

The primary use of the 'fs' module is to interact with the file system to read, write, manipulate files and directories.

Property :

fs.readline(path[, options], callback -
Asynchronously reads the contents of a file.

fs.writefilesync(path, data[, options]);
synchronously writes data to a file.

program :

```
const fs = require('fs');
fs.readline('example.txt', 'utf8', (err, data) => {
    if (err) {
```

```
    console.error('Error reading the file:', err);
  } else {
    console.log('File content:', data);
  }
});
```

6. Console :

The primary use of console is for printing messages and loging various types of data to the console.

Property :

```
console.log([data][, ...args]);
console.error([data][, ...args]);
```

Program :

```
console.log('Hello, world!');
Console.error('This is an error');
```

7. buffer :

The primary use of buffer class is to handle binary data, including encoding, decoding and manipulation of raw data in the form of bytes.

property :

buffer.length - Returns the length of the buffer in bytes.

program:

```
const buf1 = Buffer.alloc(10);
const buf2 = Buffer.from('Hello', 'utf8');
```

8. querystring:

The use of querystring is to handle URL query strings, parse them into javascript objects & Stringify Javascript Objects into query string.

Property:

```
querystring.escape(str)
querystring.unescape(str)
```

Program:

```
const querystring = require('querystring');
const querystring = 'name=abc&age=30';
const parsedobj = querystring.parse(querystring);
console.log(parsedobj);
```

9. V8 :

The use of V8 to execute javascript code. The V8 package is not part of the standard Node.js modules and is not exposed directly to the users.

methods:

Just-In-Time compilation
Garbage Collection
ECMAScript support
Asynchronous Execution

10. OS :

The primary of os is the retrieve information about the operating system such as platform, architecture, network interface and system memory.

property :

os.platform() - Returns the os platform
os.arch() - Returns the CPU architecture of the os.
os.cpus() - Returns an array of objects containing info about each CPU / core available on the system.

PAGE NO.
DATE:

Program:

```
const os = require('os');
Console.log('platform :', os.platform());
console.log('architecture :', os.arch());
```

12. Zlib:

The use of zlib is to compress and decompress data using various compression algorithm, such as a gzip, Deflate and Brotli.

Property:

```
Zlib.deflate(buffer, callback)
Zlib.deflateSync(buffer)
Zlib.deflateRaw(buffer, callback)
```

Program:

```
const zlib = require('zlib');
Const inputString = 'Hello!!';
Const inputBuffer = Buffer.from(inputString);
zlib.gzip(inputBuffer, (err, compressedData) => {
  if(err){
    console.error('compression error:', err);
  }else{
  console.log('compressed Data:',
  compressedData.toString('base64'));
}});
```