Page No.:
Date: / /

Name : Mansi C. Patel.

Semester : 7th

Rollno. : 50

Subject : Full stack Development.

Code : 701

Date : 22-07-2023

Page No.:
Date: / /

Q-1. Node js : Introduction, features, execution architechure.

→ Node js is an open-source and cross-platform javascript runtime environment.

It is a popular tool for almost any kind of project.

Node js run The V8 js engine, the core of google chrome, outside of browser This allows Node js to be very performant.

A nodejs app runs in a single process without creating a thread for every request.

Node js provides a set of asynchronous. I/O primitives in its standard library. that prevent javascript code from blocking and generally. Libraries in node js are written using non-blocking paradigms, making blocking behavior exception rather than norm.

When node js performs an I/O operation, Like reading from network, accessing a DB or file system instead of blocking threat and wasting CPU cycle waiting.

Features.

- Asychronous and non-blocking
Node js is a javascript environment which uses a single thread to handle all request. All request are handled asychronously

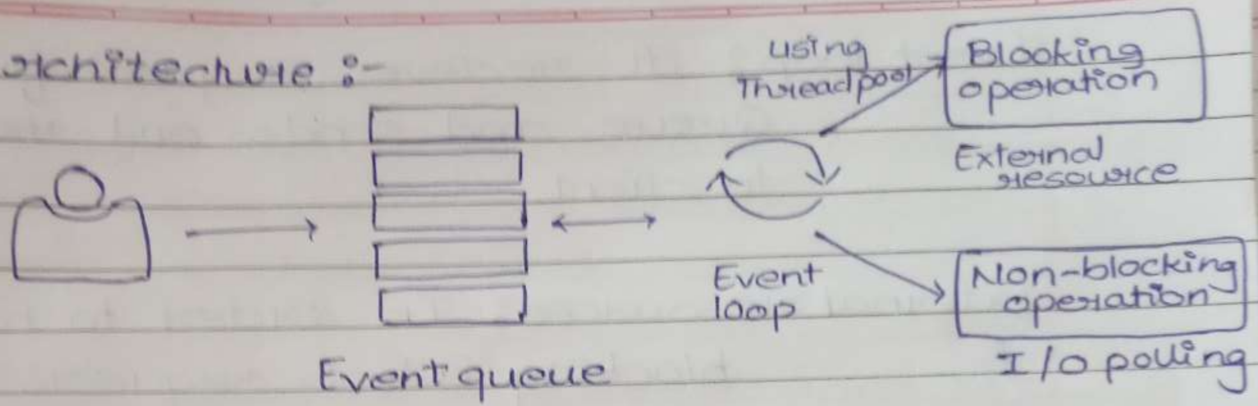
- Written in Javascript
Javascript is one of most used programming languages currently in industry. Almost all web application have javascript running.

- Scalable :
Another of node js features is scalability. This means that node js can handle a large number of request when needed.

- Cross-platform compatibility.
Node js features include cross-platform compatifity. This means it can run on many operating system like mac, Linux. and windows to name a few.

Page No.:
Date: / /

- Architechure :-



- Component of Node js architechure.

Requests : Depending on action that a user needs to perfoom request to servers can be either blocking or non-blocking.

Node js server : It accepts users requests, process them, and return result to users.

Even Queue :- The main use of event Queue is to store the incoming client request and pass them sequentatility.

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

Event loop: It recieves request from event Queue and sends out responses to client.

External resources: In order to handle blocking client requests external resources are used they can be of any type.

Q-2 Note on modules with examples.

→ In nodejs modules are blocks of encapsulated code that communicate with an external application on 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 js because of their reusability as well as ability to break down a complex piece of code into manageable chunks

Modules are of three types :-

-Core -Local - Third-Party.

- Core modules :
- node js has many builtin modules that are part of , platform and come with node js installation.

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

Syntax :

```
const module = require ('Module-name');
```

- The require() function return a javascript type depending on what particular module returns.

- The following example demonstrates how to use node.js http module to create a web server.

Example :-

```
const http = require ('http');
http. createServer (function (req, res) {
      res. writeHead (200, {"content-type':
        'text /html ' } );
      res. write (' Welcome to this page! ');
      res. end();
 }). listen (3000);
```

- Local Modules :

- Unlike built-in and external modules, local modules are created locally in your nodejs applications.

Let's create a simple calculating module that calculate various operations.

Create a calc.js that has following code :

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

exports.sub = Function (x,y) {
            return x-y;
y;

exports.mul = Function (x,y) {
            return = x * y;
y;
```

index.js

```
const calculator = require (" ./calc ");
let x = 50 , y = 10;
console.log (" Addition = " + calculator.add
                                      (x,y));
```

Page No.:
Date: / /

- Third - party modules.

- They are modules that are available online using node package manager

- These modules can be installed in project folder or globally.

- Some of popular third party are mongoose, express, angular and react

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

Q-3 Note on package with example.

- Npm is a package manager for nodejs packages or modules if you like.

A package in node.js contains all files you need for a module.

Modules are javascript libraries you can include in your ~~[illegible]~~ projects.

→ Download a package

- Downloading a package is very easy.

- open command line interface and tell npm to download package you wants.

- I want to download a package called "upper-case".

```
c:\users\your name >npm install upper-case.
```

npm creates a folder named "nodemodules" where to package will be placed.

All packages you install in future will be placed in this folder.

Page No.:
Date: / /

→ using a package.

- Once package in installed it is ready to use.

- include upper-case package same way you include any other module.

```
var uc = require ('upper-case');
var http = require ('http');
http. createServer (Function (req, res)
{
   res.writeHead (200, {'content-Type':'text/html'});
   res.write (uc. uppercase ("Hello world");
   res. end();
}). Listen (8000);
```

Output:

```
          HELLO WORLD:
```

→ It will convert 'Hello world' lower case to 'HELLO WORLD' upper case

Q-4. Use of package.json and package-lock.json

→ Package.json

- The package.json file is heart of node.js system.

It is manifest file of any node.js project and contains metadata of project.

The package.json file is essential part to understand, learn and work with node js.

It is first step to learn about developement. node.js

Package.json file compraisies of
(i) Identifying metadata properties
(ii) Functional metadata properties

Creating package.json file in following ways

(i) Using npm init

(ii) writing directly to file

→ Package - lock.json.

- package-lock.json file is like a one-stop solution of your entire problem.

- package-lock.json is a file that is automatically generated by npm when a package is installed

- It records exact version of every installed dependency, including its sub-dependencies and that versions.

- The purpose of package-lock.json is to ensure that same dependencies are installed consistently accross different environments, such as development and production environment.

- It also helps to prevent issue with installing different packager versions, which can lead to conflicts and errors.

- package-lock.json is created by npm when you run npm install.

- it contains a detailed list of all packages, their dependencies their specific version numbers and location [usually mentioned in package.json file)

Purpose of package.json.

- Dependency version tracking.
- Consistent builds
- Faster and more reliable installs
- Security
- Reproducible builds.

Q-5. npm introduction and commands with its use.

→ npm stands for node package manager. and its .package manager for node js platform.

It put modules in place to that node can find them, and manages dependency conflicts intelligently.

npm is free to use.

most commonly it is used to publish discover, install as develop node program.

Commands :-

npm install
- it install a package in package.json file

npm update
- it update specified package.

npm init.
- create a package.json file, asks some questions and create package.json file.

npm start.
- runs a command that is defined in start property in scripts.

npm ls
- list all packages as well as their dependencies

Q-6 Describe use and working of following node js packages - url, process, pm2, readline, fs, events, console, buffer, querystring, http, V8, OS, zlib.

⟶ url

The 'url' module provides utilities of URL resolution and parsing. The getters and setters implement properties of URL objects on class prototype.

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

⟶ process

This module provides interaction with current . node.js process

```
const process = require ('process')
```

Page No.:
Date: / /

process object has a property 'args' which is an array containing the properties passed to node process.

→ pm2

It enables you to keep applications alive forever.

It also helps reload application without downtime, monitoring and clustering. It allows you to configure several different strategies for how your nodejs application should restart.

```
pm2 start app.js
```

→ readline.

allows. reading of input stream line by line. It wraps up process standard .input and output object.

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

Date: / /

→ fs.

helps us store, access and manages data on our operating system. commonly used features of fs module include.
fs. readfile to read data from a file.
fs. writefile to write data to a file and so on.

→ events

This module emits named events that can cause corresponding functions or callbacks to be called. It provides you with event emitter class that allows you to manage events in node application use on() method of eventemitter object to registor an eventhandler for an event.

→ Console.

This module is a global object that provides a simple .debugging console similar to js to display different levels of message.

→ Buffer

It is used to perform operation on raw, binary data. It refers to particular.

→ Querystring.

It is used to provide utilities for parsing and formatting URL querystring
It can be used to convert query string into json object and vice-versa

→ http

To use http server in node, we need to require http module. The http module creates an http server that listens to server ports and gives a response back to client.

→ zlib

It is used to provide compression and decompression functionalities