Name : Rathod Mansi A.

Semester : 7th

Roll no : 69

Subject : Full stack development.

Question 1 : Nodejs : introduction, features, execution architecture.

* Introduction :

→ Node js is an open source server environment
→ Node. js runs the V8 JavaScript engine, the core of google chrome, outside of the browser. This allows node.js to be very performat.
→ A node.js app runs in a single process without creating a new thread for every request. Node.js provides a set of asynchronous I/O primitives in its standard library that prevent Js code from blocking
→ when Node.js Performs an I/O operation, like reading from the network, accessing a database or the file system. Instead of blocking the thread and wasting CPU cycle waiting, Node.js will resume the operations when the response comes back

* Features :

it's easy to get started and can be used f

Prototyping and agile development.

→ It provides fast and highly scalable services.

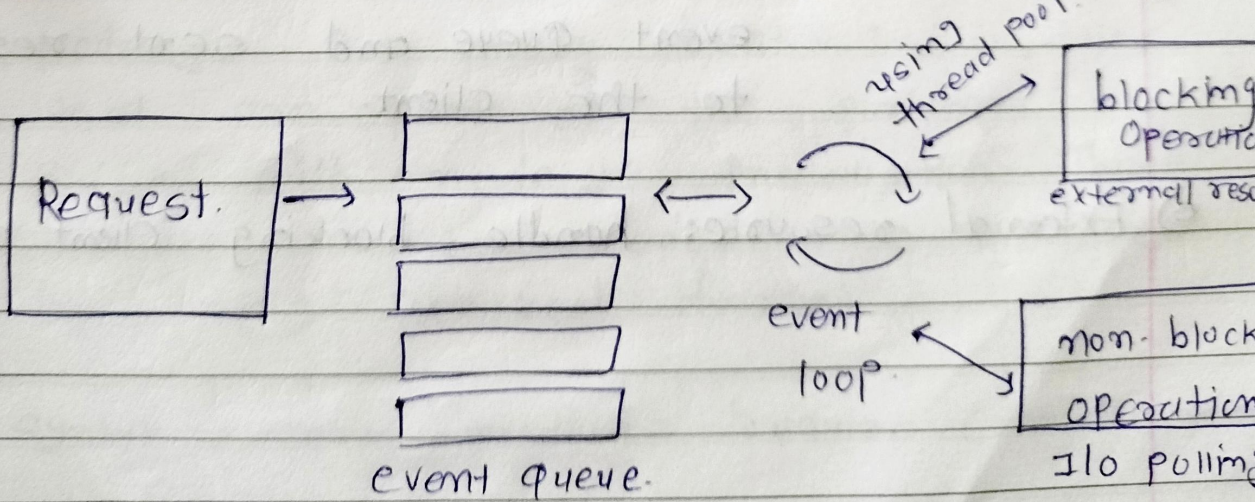
→ It uses JS everywhere, So it's easy for a JavaScript programmer to build back-end services using Node.js

→ source code cleaner and consistent.

→ Large ecosystem for open source library

→ It has Asynchronous or Non-blocking nature.

* execution Architecture.



* components of Node js architecture

1) Request : request can be either blocking or non-blocking

2) nodejs server : accept user request, process them and return result.

3) event queue: Store the incoming client request and pass them sequentially to the event loop.

4) thread Pool : contains the threads that are available for performing op. required to process request

5) Event loop: receive request from the event queue and sent response to the client.

6) External resource: handle blocking client requests

Question e: Note on module 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 single file or a collection of multiple files / foldeos.

Modules are of three types.

1) core module
2) local module
3) third party module.

1) core module :

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

Syntax : const module = require ('module_name

2) local module.

→ local modules are created locally in

node.js application.

→ This module also hides functionality that is not needed outside of the module.

3) Third party modules :

→ These modules are available online using the node package manager. These modules can be installed in the project folder or globally.

→ ex: Mongoose, express, angular and react.

Example :

```
Const http = require ('http');
http.createServer ( function (req, res)
{
    res.writehead (200, {'content-type' : 'text-
                                    html'});
    res.write ("welcome");
    res.end ();
}).listen (3000);
```

Question 3: Note on package with example.

→ A package is a folder tree described by a package.json file.
→ The package consists of the folder containing the package.json file and all subfolders until the next folder containing another package.json file, or a folder named node_modules.
→ A package in node.js contains all the files you need for a module.

Example :

```
var http = require('http');
var uc = require('upper-case');
http.createServer(function(req, res)
{
    res.writeHead(200, {'content-type':
                        'text/html'});
    res.write(uc.upperCase("Hello World"));
    res.end();
}).listen(8080);
```

→ above example, we can install upper case Package by npm i upper-case syntax.

Question 4 : use of package.json and package-lock.json

* package.json

→ Package.json is a versioning file used to install multiple packages in project.
→ As you initialize your node application, you will see three files installed in app that is node_module, Packag.json, and package.lock.json.
→ It records important metadata about a project which is required before publishing to NPM.
→ It also defines functional attributes of a project that npm uses to install dependencies, run scripts, and identify the entry point to our package.

* 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 the exact version of every installed dependency, including its sub dependencies and their versions.

→ The purpose of package-lock.json is to ensure that the same dependencies are installed consistently across different environments, such as development and production environments. It also helps to prevent issues with installing different package versions, which can lead to conflicts and errors.

→ Package-lock.json has just transformed the way packages are installed in a program.

Question 5: Nodejs packages.

↳ Same as Answer 3

Question: 6 npm introduction and commands with its use.

→ npm stands for node package manager

→ It allows for seamless nodejs Pactages management.

→ you can install, share and manage it

→ npm consist of three components:

1) website
2) registry
3) AI

* Installing npm.

→ npm comes with the node js.
→ so, you don't have to install it seperately.
→ you can install node js from their official website.
→ After installing node, you can check version of it

* commands :

1) npm init.

→ It asks you for some data like author name, description etc.

→ you can just press enter for defaults.

→ To quickly create a package.json file you can use command
npm init -y

2) npm install
→ install a package in package.json file.

3) npm Unistall.
→ Remove package from package.json file.

4) npm update.
→ updates the specified package.

5) npm start.
→ Runs a command that is defined in the start property

6) npm build :
→ used to build a package.

Question 7: Describe and use following nodejs packages, important properties and methods and relevant programs.

i) url

→ This module has utilities for URL resolution and parsing meant to have feature parity with node.js core URL module.

* properties:

1) href
2) protocol
3) host
4) auth
5) hostname
6) port
7) Pathname

* method:

1) parse → return an object

2) format → return formatted URL string

3) resolve : take a base URL, href URL and resolve them as a browser for anchor tag.

ii) process

→ The goal of this module is not to be a full fledged alternative to the builtin process module.

→ This module mostly exists to provide the nextTick functionality and little more.

syntax: npm i process.

iii) readline.

→ allow the reading of input stream line by line. This module wraps up the Process standard output and process standard input objects.

```
var readline = require ('linebyline');
rl = readline ('./file-in-win1251.txt', {
    retainBuffer: true
});
rl.on("line", function (data, linecount)
{
    var line = iconv.decode (data, 'win1251');
});
```

iv) fs

→ This module allows you to work with the file system on computer.

Syntax: var fs = require ('fs');

* common use:

1) Read files
2) create files
3) update files
4) delete files
5) Rename files.

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

Example.

```
var http = require ("http");
var fs = require ('fs');
http.createServer (function (req, res) {
  fs.readFile ('xyz.html', function (err, data)
{ res.writeHead (200, {});
  res.write (data);
  return res.end ();
});  }).listen (8080);
```

v) events.

→ node's event emitter for all engines.
→ This implements the Node.js events module for environments that don't have it. like browsers.
→ events module uses ES5 features.

Example:

```
var EventEmitter = require('events')
var ee = new EventEmitter()
ee.on('message', function (text) {
  console.log(text)
})
ee.emit('message', 'hello world')
```

vi) console.

→ Dzop - in replacement for console - a cross environment fix for missing methods.

Installation

```
npm install console --save.
```

and import it:

```
import console from 'console'
```

→ is a global object that provides a simple debugging console similar to JavaScript to display different levels of message.

vii) buffer.

→ Provides a way of handling streams of binary data.

→ is a global object
→ not necessary to import it using the required keyword.

properties

1) alloc
2) allocUnsafe
3) allocUnsafSlow
4) byteLength
5) compare
6) concat
7) copy
8) equals
9) fill
10) form
11) includes.

viii) Query string

→ Provides utilities for parsing and formatting URL query strings. It can be accessed using :

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

* Method.

1) decode()

→ alias for querystring.parse().

2) encode()

→ alias for query string.stringify().

3) escape

→ Performs URL percent-encoding on the give str in a manner that is optimized for specific requirements of URL query strings.

ix) http

→ allows Node.js to transfer data over the Hyper Text Transfer protocol (HTTP)

⇒ HTTP module can create an HTTP server that listens to server ports and gives a response back to the client.

→ use createServer() method to create an HTTP server

Example:

```
var http = require('http');
http.createServer(function(req, res)
{
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write('Hello');
    res.end();
}).listen(8080);
```

X) V8

→ is an open source JS engine developed by the Chromium project for the Google Chrome.

→ represents interfaces and event specific to the version of V8. It provides methods

viii) Query string

→ provides utilities for parsing and formatting URL query strings. It can be accessed using :

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

* Method.

1) decode()

→ alias for queryString.parse().

2) encode()

→ alias for queryString.stringify().

3) escape

→ performs URL percent-encoding on the given str in a manner that is optimized for specific requirements of URL query strings.

ix) http

→ allows Node.js to transfer data over the Hyper Text Transfer protocol (HTTP)

→ HTTP module can create an HTTP server that listens to server ports and gives a response back to the client.

→ use createServer() method to create an HTTP server

Example:

```
var http = require('http');
http.createServer(function(req, res)
{
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write('Hello');
    res.end();
}).listen(8080);
```

X) V8

→ is an open source JS engine developed by the Chromium project for the Google Chrome.

→ represents interfaces and event specific to the version of V8. It provides methods.

to get information about heap memory through v8. getHeapStatistics.

use syntax:

```
const v8 = require ('v8').
```

xi) os

→ provide information about os system.

Syntax:

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

* methods and properties

1) arch()
2) constants
3) cpus()
4) endianness()
5) EOL
6) freemem()
7) hostname()
8) networkinterfaces()
9) release()
10) type()

xii) zlib

→ compress a file into a gzip file.

→ provides a way of zip and unzip files.

Syntax:

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

* properties and methods.

1) constants
2) createDeflate()
3) createDeflateRaw()
4) createGunzip()
5) createGzip()
6) createGzip
7) createInflate()
8) createInflateRaw()
9) gzip
10) gzipSync()
11) unzip
12) unzipSync().