Name :- Om Pandey

Roll :- 35

Sem :- 7th

Subject :- Full Stack Development

Theory Assignment - I

Q1 Node js - Introduction, features, execution architecture.

ns Node js is an open saurce, cross platform JS runtime environment that allows developers to run JavaScript code on the server side. It was built on chrome's v8 JS engine which provides fast & efficient ~~exection~~ execution of the code

Features

- Asynchronous & Non-Blocking

Node js uses an event-driven, non-blocking I/O model which allows it to handle multiple requests concurrently without getting blocked by long running operations.

- Single-Threaded Event loop

Node js operates on a single threaded event-loop, which means it uses a single process to handle multiple requests

- Cross Platform

Node js runs on multiple operating system including win, macOS and various Linux distributions.

Date
Page

Execution Architecture

- The architecture of Node.js revolves around its event-driven, non-blocking I/O model

a) Event Loop

Event loop is mainly responsible for handelling events and callbacks in a non-blocking manner. The Event loop continuously checks for pending events, executes their associated callbacks and then moves to the next event.

b) Event-driven & Non-blocking I/O

This means it can initiates I/O tasks and then move on to other tasks without waiting for the I/O to complete. When the I/O operation is finished a callback is triggered to handle the result.

c) Callbacks

Asynchronous functions in node.js uses callback to return results once the operation is completed. These callbacks are key of node.js.

Q2 Note on module with example

Ans - In node.js the modules are used to encapsulate the code and provide a way to organize, share and reuse functionality across different parts of an application. Each file in Node.js is treated as a module and we can export specific parts of module to make them accessible to other parts of the application.

We can create modules as follows:-

- Make a file name abc.js and write the following code:-

```
// Export a function
module.exports.addition = function (num) {
    return num + num
};
```

- Now we can use this file in another file named app.js :-

```
const add = require('./abc.js');
const num = 8;
const result = add.addition(num)
console.log(`${result}`);
```

Q3 Note on package with example.

→ In node js packages refers to the ~~collection~~ collection of modules, code and configurations files organized together to serve a specific purpose or functionality.

Packages are managed using Node Package manager and can be easily installed into node js apps.

NPM is default package manager for nodejs and provides aces to a vast ecosystem of open-source packages that can extend the capabilities of your application.

To create a node package you have to follow the following steps:-

- Initialize the package using npm-init command under terminal. Then fill the necessary fields and then press enter then it will form a file named as package.json.

- Create a file named greeting.js

// greetings.js

```
exports.greeting = function(name) {
    return 'Hello ${name}!
};
```

- Update the package to specify the entry point for the package. Open the file and add.

```
"main": "greetings.js"
```

Now this package is ready to get installed and used by different files.

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

Ans a) The package.json file is a manifest for your node.js project. It contains meta data about the project including version, description, author info, entry points and most importantly the list of dependencies required to run the project. This file acts as the central configuration for your project. They are used as follows:-

1) Dependency Management

→ The 'dependencies' section lists the external packages and their respective versions required for your project to function correctly. When someone else wants to run your project then they can use package.json file to install all the required dependencies with a simple command.

2) Scripts

This section in file allows you to define custom scripts under the 'script' section. These can be used to automate the tasks like running tests, starting the application etc.

3) Metadata

This file includes 'metadata' about the project like the project's name, version, description, author, license, repository URL and other details.

b) Package-lock.json

This was introduced in ~~node~~ npm v5 to provide consistent dependency resolution. It is automatically generated when we use 'npm install' command. This file is used for locking down the exact version of dependencies you ~~are~~ installed in your project.

The key purpose:-

a) Dependency version locking

The file contains an exact, resolved version of each installed dependency and its sub-dependency. This ensures that all developers and servers working on the project use the same version, minimizing potential issues caused by version mismatches.

2) Faster and reproducible install

By using the information in 'package-lock.json', npm can ~~skip~~ skip any unnecessary dependencies resolution during installation, making the process faster. It also guarantees that anyone else who installs the dependencies gets the exact same version you have, thus ensuring reproductibility.

Both the file that is package.json and package-lock.json should be commited to version controll to ensure consistent and predictable project setups for everyone collaborating on the project

Q5 Node js packages

Ans In the content to node js, packages refer to external modules or library that can be installed and used in your Node js application to extend their functionality. Node js packages are managed ~~to extend their~~ using the npm or yarn which provide access to a vast ecosystem of open-source packages contributed by developers worldwide. These packages can be easily installed, updated or removed to enhance the capabilities of Node js applications.

There are various types of packages but the most common in use are as follows :-

1) Utility Package :- provides general utility functions that can be used across different projects. like 'lodash'.

2) Authentication packages :- Passport, bcrypt & JWT token.

and many more.

To use such packages you need to create a package.json file that lists the dependencies your project require you can also manually edit the file.

Example:- if you want to install express then:-

- using npm:- npm install express --save

And then you can use the same in your file like here in 'app.js'.

```
=> const express = require('express');
   const app = express()
   app.get('/', (req, res) => {
           res.send("Hello");
       });
   app.listen(3000, () => {
           console.log("started at 3000");
       });
```

Q. npm introduction and command with its use.

→ It is a command line tool that comes bundled with node js. It is the default package manager for Node js, allowing developers to easily install manage and share open-source node js packages and modules.

NPM commands

1) npm-init :- the use initializes a new node js project and creates a package.json.

2) npm install :- Use to install the dependencies listed in package.json file for current project.

3) npm install (package) :- To install a specific package and to save it as a project dependencies.

4) npm ~~inst~~ install (package) --save-dev.

Installs a package as a development dependencies, meaning it is required for development and testing purpose but not for production applications.

5) and others like npm update, npm uninstall <package-name> and npm update <package-name>, npm list are mainly used for the basic things in node js.

6) npm run <file name> Runs custom scripts defined in scripts section of package.json.

Q7 Describe use and working of following node.js packages. Important properties and methods and relevant programs.

Ans

1) Url

The url module provides utilities for URL resolution and parsing. It is used to work with URL's and extract information from them

Eg:- const url = require('url')
const urlString = "https://www.google.com";
const parsedUrl = new Url(urlString);
console.log(parsedUrl.host);
console.log(parsedUrl.pathname);
console.log(parsedUrl.search params);

2) process, pm2 (external package)

process object provides info and control over the node.js process. It allows interacting with the current process and accessing environment variable. getting command line Arguments

console.log(process.argv);

pm2 is an internet package used to manage node js prows. It provides tools for prows. It provides tools for prows monitoring, scaling and cluster management.

```
# install pm2 g/t belly
  npm install -g pm2
  pm2 start app.js
```

3) readline

The 'readline' module provides an interface for reading input streams line by using the package. Commonly used to interact with users in the command line.

```
→ const read = require('readline');
  const rl = read.createInterface({
        input: prows.stdin,
        output: prows.stdout
      });
  rl.question("Hello, who are you?
  (name) => {
        console.log(`Hello ${name}
    rl.close();
});
```

4) fs

This module provides file system related functionality, allowing reading writing and manipulating files.

Reading a file :-

```
const fs = require ('fs')
fs.readfile ('example.txt', 'utf8'
    (err, data) => {
        if (err) {
            console.error ("Error", err);
            return;
        }
        console.log ('File content', data);
});
```

5) Events

This module provides an event-driven architecture for building apps that can emit and listen to events

```
- const eventEmitter = require ('events');
  class myEmm extends EventEmitter {}
  const myEmm = new myEmm ();
  myEmm.on ('greet', (name) => {
        console.log ('Hello', ${name} );
  });
  myEmm.emit ('greet', 'Om');
```

6) Console

This module provide a simple debugging console that can be used to log messages during development

eg:- console.log('This is log message');

7) Buffer

This module provides a way to handle binary data. It is used to work raw binary data in Node.js.

```
var buffer = Buffer.from('abc');
console.log(buffer);
```

=> 61, 62, 63

8) QueryString

This provides utilities for working with query string in URL's

```
const query = require('querystring')
const param = query.parse("name=Om&age
console.log(param);
```

→ {name: 'Om', age: '20'}

http

This module provides a set of functions and classes to create HTTP server and make HTTP requests

```
const http = require('http');
const server = http.createServer((req, res) =>
    { res.writeHead(200, {'Content-Type':
            'Text/plain'});
      res.end("Hello");
    });
server.listen(3000, () => {
      console.log("at- 3000");
  });
```

V8

This exposes API's related to the V8 javascript engine, providing acess to performance and memory related data.

```
const V8 = require('v8');
console.log(V8.getHeapStatistics());
```

11) OS

This module provides operating system related functionality such as information about the host OS.

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

12) zlib

The zlib module provides compression and decompression functionality using gzip and deflate.

```
const zlib = require('zlib');
const data = "This is some data";
zlib.gzip(data, (err, dataCompressed) =>
    {
        if (err) { console.error("Error", err); }
        return;
        console.log('dataCompressed', dataCompressed);

//unzip
zlib.unzip(compressedData, (err, decompressed
    Data) => { if (err) console.log("Error", err)
        return;
    console.log('Decompressed', decompressed
        Data.toString());
    });
});
```