Avaiya Jinal A.

Page No. 1
Date

Roll.No - 2 , Sub:- Node.js

Assignment - 1

Que-1 Node.js : Introduction , Features , execution architecture.

- Node.js is an open-source , cross platform javascript run-time environment that executes javascript code outside of a browser.
- Node.js was written and introduced by Ryan Dahl in 2009.

Node.js = Runtime Environment + Javascript library.

Features :-

1. Extremly Fast :-
   - Node.js is built on google chrome's v8 javascript engine , so it's library is very fast in code execution

2. I/o is Asynchronus and Event Driven :-
   - All API's of node.js library are asynchronous i.e non-blocking . So a NodeJS based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of node.js helps the server to get a response from the previous API call . It is also a reason that it is very fast.

3. Single threaded :-
- Node.js follows a single threaded model with event looping.

4. Highly Scalable
- Node.js is highly scalable because event mechanism helps the server to respond in a non-blocking way

5. No buffering :-
- Node.js cuts down the overall processing time while uploading audio and video files. Node.js application never buffer any data. These applications simply output the data in chunks.

6. Open source :-
- Node.js has an open source community which has produced many excellent modules to add additional capabilities to Node.js applications.
License : Node.js is released under the MIT license

- execution architecture
- Node.js is operates on an event-driven, non-blocking I/O architecture. The core components of its execution architecture are :

1. Event Loop :-
- The event loop is the heart of Node.js. It continuously listens for events and executes associated callback functions. It allows Node.js to handle

asynchronous operations without blocking the execution of the entire application.

2. Event Emitters :-
- The event loop is the heart of Node.js. It continuously listen for events and executes associated callback functions. It allows Node.js to handle asynchronous operations without blocking the execution of the entire application.

3. Callback Queue :-
- When asynchronous operations are completed, their corresponding callbacks are placed in the callback queue. The event loop continuously checks this queue for pending callbacks and executes them one by one.

4. Thread Pool :-
- Although Node.js runs on a single thread, certain tasks like file system operations and cryptographic computations are CPU-intensive and can block the event loop. To handle such tasks efficiently, Node.js uses a thread pool, which consists of worker threads that can execute these blocking operations in parallel.

5. Libuv :-
- Libuv is a cross-platform library that provides the event loop implementation for Node.js. It abstracts the underlying system calls and handles I/O operations asynchronously on behalf of Node.js applications.

Ic-2 Note on modules with example.

- modules are Javascript libraries.
- A set of Functions you want to include in your applications. There are two type of modules :-
  i) Built-in
  ii) External.

i) Built in modules:-

- Node.js has a set of built-in modules which you can use without any further installation.
- To include a module, use the require() function with the name of the module:

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

Now, your application has access to the HTTP module and is able to create a server.

* Built-in http Module.

- The HTTP module can Create an HTTP Server that listens to server ports and gives a response back to the client.

- Use the CreateServer() method to create an HTTP Server.
  The Function passed into the http.createServer() method, will be executed when Someone tries to access the computer on port 8080.
- If the response from the HTTP Server is supposed to be displayed as HTML, you should include an HTTP header with the correct content type.

res. writeHead (200, {'Content-Type': 'text/html'});

- The function passed into http.createServer() has a req argument that represents the request from the client, as an object (http.IncomingMessage object)
- This object has a property called "url" which holds the part of the url that comes after the domain name.

http://localhost:8080/temp.

2) Utility modules:-

var http = require('http');

//create a server object
http.createServer(function(req, res){
  res.write('Hello world!');
  res.end();
}).listen(8080);

1e-3. Note on package with example.

- Npm is the default package manager for nodejs and is written entirely in javascript. Developed by Isaac Z. Schlueter, it was initially released in January January 12, 2010.
- npm manages all the packages and modules for Nodejs and consists of Command line client npm. It gets installed into the system with installation of Nodejs the required packages and modules in Node project are installed using Npm.
- A package contains all the files needed for a module and modules are the javascript libraries that can be included in Node project accord to the requirement of the project.
- NPM can install all the dependencies of a project through the package.json file. It can also update and uninstall packages. In the package.json file, each dependency can specify a range of valid versions using the semantic versioning scheme, allowing developers to auto update their packages while at the same time avoiding unwanted breaking changes.

Some Facts about NPM:-

- At the time of writing this article, Npm has 580096 registered packages. The average rate of growth of this number is 291/day which outraces every other package registry.
- Npm is open Source.
- The top npm packages in the descreasing order are: lodash, async, react, request, express.

Installing NPM:-

- To install NPM, it is required to install Node.js as NPM get installed with Node.js automatically.

Checking and updating npm version.

- version of npm installed on system can be checked using following syntax:-

```
npm -V
```

Installing package.

```
npm install package_name.
```

le-4 use of package.json and package-lock.json.

package.json.

- It is a metadata file that describes the project's dependencies, scripts, configuration, and other details
- It is typically created and modified manually by the developer to manage the project's dependencies and configuration.
- It lists the required dependencies and their version ranges, but not the exact version to b installed.
- It can be easily shared and commited to version control systems.

{

xample:- "name" : " your project name",
"version" : "1.0.0";
"description" : "your project description",
"main" : "your app.js",
"Script" : {
"test": "echo \"Error: no test specified \" && exit
},
"author" : "Author name",
"license" : "ISC",
"dependencies" : {
"dependency1" : "^1.4.0",
"dependency2" : "^1.5.2"
}

}

- package-lock.json

- It is a lockfile that provides an exact, deterministic list of all the installed package and their dependencies, including their exact version numbers.
- It is automatically generated by npm and updated whenever you install or update packages.
- It is used to ensure that the same dependencies are installed consistently across different environments and prevent conflicts due to different versions being installed.
- It is not meant to be manually modified and should be committed to the version control system to ensure consistency across all team members.

Example:-

```
{
  "name" : "your project name",
  "version" : "1.0.0",
  "lockfileversion" : 1,
  "requires" : true,
  "dependencies" : {
      "dependency1" : {
          "version : "1.4.0",
          "resolved" :
"http://registry.npmjs.org/dependency1/-/dependency1-1.4.0.tg
              "integrity" :
"sha512
e+Wh4ky2g/S1GvAbzDHpyRu7AnOomm4RHJmxhRz1CkfUT41brVhNDASE
     omWD9psBbsl3PDcoUHbVvu DiH2m+A=="
  },
```

```
"dependency2" : {
    "version" : "1.5.2",
    "resolved" :
"https://registry.npmjs.org/dependency2/-/dependency2-1.5.
    tgz", "integrity" :
"sha512-
 Won2IV5AhyE2Oqvf PIvxe3tupJuc9Ixcok BT Buiag TGO+P2CAyEc
 IxmftnL+2CTI6d5SIij4jcIDFnomsP2 4O=="
    }
  }
}
```

Que-5. Node.js package.

1) Async.js
- 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.
- It uses queues to monitor your workflow, allowing you to append additional tasks or attach extra callbacks, and handles errors with callbacks. This makes it a more versatile and robust solution for complex dependency management.

Example:

```
async function myFunction(inputValue)
{
    try {
        const a = await asyncFunc1('value');
        const b = await asyncFunc2(a);
        const c = syncFunc3(b);
        return await asyncFunc4(c);
    } catch (ex) {
        // handle exception
    }
}
```

2) mysql :-
- mysql is a Node.js client for the mysql protocol. Before using mysql to connect to your database ensure you

have mysql installed and configured in your machine. Then create a database and database tables that you can work with. Check instructions on how to create mysql database and tables.

```
Const {create connection} = require('mysql');
Const connection = create connection({
    host: "localhost",
    user: "root",
    Password: "",
    database: "my_database",
});
Connection.query('select * from name', function (err, result, field:
{
    if (err) {
        return console.log(err);
    }
    return console.log(result);
}).
connection.end();
```

3) 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 function that make Node.js coding easier and cleaner. Instead of writing a common function repeatedly you can use just a single line of code with the help of Lodash.

Unit - 6 npm introduction and command with its use

- Npm stand for Node package manager.
- Npm is a package manager for the Node javascript platform.
- Npm is known as the world's largest software registry. Open-source developers all over the world use npm to publish and share their source code.

Command :-

1) npm install :-
- This command is used to install package. You can either install packages globally or locally. when a package is installed globally, we can make use of the package's Functionality from any directory in our computer.
- On the other hand, if we install a package locally, we can only make use of it in the directory where it was installed. So any other folder or file in our computer can use the package.

2) npm uninstall :-
- This command is used to uninstall package.

3) npm init :-
- The init command is used to initialized a project. when you run this command, it creates a package.json file.

4) npm update:-
- use this command to update an npm package to its latest version.

5) npm restart :-
- used to restart a package. you can use this command when you'd want to stop and restart a particular project.

6) npm start
- used to start a package when required.

7) npm stop
- used to stop a package from running

8) npm version
- shows you the current npm version installed on your computer.

9) npm publish
- used to publish an npm package to the npm registry. This is mostly used when you have created your own package.

Q4c- Important properties and methods and relevant programs
Describe use and working of following node.js packages.

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

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

process :-
- works in node.js and browsers via the browser.js shim provided with the module.

```
require('process');
```

pm2 :-
- pm2 is a production process manager for Node.js applications with a built-in load balancer. It allows you to keep applications alive forever, to reload them without downtime and to facilitate common system admin tasks.
- Starting an application in production mode is as easy as :-

```
$ pm2 start app.js
```

Readline :-
- simple streaming readline module for node.js Reads a file and buffers new lines emitting a line event for each line.

npm install linebyline.

FS :- The fs module allows you to work with the file system performing various operations like reading writing or manupulating files.

```
const fs = require('fs');
fs.readFile('example-txt', 'utfs', (err, data
=> {
  if(err) throw err;
  console.log(data);
});
```

Event

Node's event emitter for all engines.

- This implements the Node.js events modules for environments that do have it > like browsers.

Events currently matches the Node.js 11.13.0 API

Console :-

- Drop in replacement for console across environment fix for missing methods.

npm install console --save

buffer

- The buffer module from node.js, with browserify, simple require('buffer') or use the Buffer global and you will get this module.
- The goal is to provide an API that is 100% identical to node's Buffer API. Read the official docs for the full list of properties instance methods, and class methods that are supported.

Query String.
- parse and stringify URL query string
- For browser usage, this package targets the latest version of Chrome, Firefox and Safari.

http :-
- Start using http in your project by running npm http. There are 1458 other projects in the npm registry using http.

V8
- V8 provide a direct interface to google's high performance javascript engine. The V8 engine is also used in chrome, nodejs, mongoDB, and many other Software. However each of

OS
- This is a Node.js core module.
- The os module provides information about the operating system on which Nodejs is running

```
Const os = require('os');
Console.log('platform:', os.platform());
Console.log('Architecture:', os.arch());
Console.log('Total memory(bytes):', os.totalmem());
```