1) Node JS : Introduction, features, execution architecture.

->

Introduction :-

node js is an open source and cross platform runtime environment for executing. it enables developers to execute javascript code outside of a web browser. allowing server side scripting and development of scalable applications.

Features :

1) Asynchronous and non-blocking I/O :

it is designed to handle asynchronous operations efficiently making it ideal for I/O intence applications. it uses non blocking event driven architecture

2) Single threaded and event driven :-

it operates on single-threaded event loop, which handles all asyne I/O operations it avoids overhead of thread creation and context switching.

3) NPM :-

it is a default package manager for node.js, providing access to thousands of open source packages and modules.

4) Fast execution :-

node leverages v8 engine which compiles javascript code into machine code.

5) cross platform :-

it is compatible with various operating systems including windows, macos and linux.

* Execution architecture :-

1) Event loop :- the heart of nodejs is an event loop. it is a continuous loop that constantly checks for pending I/O operations, timers and callbacks.

2) Non-blocking I/O :- when non blocking I/O operation is initiated, it doesn't wait for operation to complete. instead it continues executing next line of code

3) Callbacks: it uses call backs to handle result of async operations. when an IO operation is completed corresponding callback function is added

4) Event queue: it holds all the callbacks waiting to be executed. event loop continuously checks event queue for pending callbacks

5) Event emitters: nodejs uses event emitters to handle events and event listners. this mechanism facilities communication between parts of app.

6) libuv: nodejs relies on libuv library to handle low-level async IO operations.

2) Note on modules with example.

→ In nodejs modules are reusable blocks of code that encapsulate functionality and allow you to organize your code into separate files.

they help in maintaining code reusability, modularity and keep the codebase clean

* Creating a module:

1) Create a new file named 'circle.js'
2) Define functionality you want.

```
Const calculatedArea = (radius) => {
    return Math.PI * radius * radius
}

module.exports = calculatedArea.
```

* Using modules:

```
Const calculatedArea = require('./circle')
Const radius = 5
Const area = calculatedArea(radius)
```

This is how we can create different modules, export them and require them in another file to reusability.

3) Note on package with example.

→ In Nodejs a package refers to a collection of modules, often accompanied by a package.json file that contains metadata about package, its dependencies version and other information.

Packages are managed and distributed using node package manager (NPM).

NPM is a default package manager for node.js that provides vast repos

* Creating a package:

Step-1: Create new directory for your package

Step-2: initialize your project by running "npm init". this will create package.json.

Example:-

1) create a new directory for package e.g. 'random-generator'
2) navigate to directory by command prompt.
3) run npm init command.

4) create index.js

```
const getRandom = (min, max) => {
    return Math.floor(Math.random()
}
module.exports = getRandom
```

* using the package

```
const getRandom = require('../random
                          generator')

const randomNumber = getRandom(1,10)
console.log(randomNumber)
```

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

->

package.json file is the heart of any node.js project. it contains meta data about project including.

- project name, version and description
- Entry point
- list of dependencies.
- list of devdependencies.
- project repo, author, licen

This file is essential for sharing your project with others and ensuring that they have the necessary dependencies to run project correctly.

When we create new node.js project using npm init package.json is created

-> package-lock.json :-

package-lock.json is automatically generated by npm when you install packages. It serves two main purposes.

1) Dependency locking :-

it locks the specific version of packages that were installed in project. this ensures that whenever someone else installs the project dependencies using npm i they get exact same version of package

2) Faster and reliable installation :-

npm can avoid unnecessary dependency resolution and network requests during installation. it includes complete tree of package versions.

5) Node.js packages:-

In node.js packages refer to collections of code, modules and dependencies that extend functionality of application.

Packages are managed using NPM. NPM provides access to a vast ecosystem of open source packages that you can integrate into projects.

1) npm registry:-

it is a centralized repo where developers publish and share their packages.

2) package installation:-

we use 'npm install' command followed by package name. NPM will download and install packages along all dependencies.

3) Dependency management:-

node.js packages often have dependencies on other packages. when you install package NPM automatically fetch dependencies.

4) versioning :-

each package in npm registry has a version number. that follows semantic versioning scheme. this helps ensure that you can control the version.

5) scoped packages :-

some packages have a namespace prefix, known as a scope. scoped packages are typically used.

6) package and package-lock.json :-

As mentioned earlier, package.json file contains metadata about your project and its dependencies, while package lock.json file ensures the specific versions of dependencies are installed.

6) NPM introduction and command with its use

→ it is a package manager for node, allowing developers to easily discover, install, manage and share reusable code packages. it comes bundled with Node.js. So you can start using it immediately after installing node.js on system.

npm is one of the largest package registries globally, offering wide variety of open-source packages contributed by community.

* NPM Commands:

1) npm init:

initializes a new node.js project and creates a package.json file.

2) npm install:

installs project dependencies listed in package.json file. When you clone project with others you run this command.

3) npm uninstall :-

it uninstalls a package from project. it removes it from node_modules

4) npm update :

updates packages to their latest versions based on package.json file.

5) npm run :-

executes scripts defined in the 'scripts' section of package.json.

6) npm init -y :-

initializes a new node.js project with default settings without asking

?) important properties and methods and relevent programs.

->

URL:

it provides utilities for URL resolution and parsing.

-> url.parse (urlString [, parseQueryString, slashesDenoteHost]]).

ex:

```
Const url = require('url');
Const urlString = 'https://www.a.com:8080/
                   Path?Query=Param';
Const parsedUrl = url.parse (urlString,
                             true)
Console.log (parsedUrl);
```

process:-

it provides information and control over current processes.

- process.argv
- process.env
- process.cwd()

Pm 2 :-

its a popular process manager for node.js

- it needs to be installed first using npm.

- npm install -g Pm2

readline :-

it provides an interface for reading data from a readable stream.

- readline.createInterface
- rl.question (query, callback)

fs :-

- fs.readfile (Path [, options], callback)
- fs.writefile (file, data [, options], callback

Events :-

- eventEmitter.on (event, listner)

- eventEmitter.emit (event [, ...args])

Console:-

- Console.log ([data][, ...args])
- Console.error ([data][, ...args])

buffer:-

provides way to work with binary data directly.

- Buffer.alloc (size[, fill[, encoding]])
- Buffer.from (array)

Query String:-

provides utility for parsing and formatting URL query string.

- queryString.parse()
- query string.Stringify()

http:-

provides functionality for creating HTTP servers and clients.

- http.createServer([options][,requestListener]).

- http.get()

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

const server = http.createServer((req,res) => {
    res.end("hello")
});

v8 :-

const v8 = require('v8');
Const heapstats = v8.getheapstatistics();

os :-

const os = require('os');
Const cpuInfo = os.cpus()[2];

const totalmemory = os.totalmem();
```

zlib :-

zlib.gzip()
zlib.gunzip()

```
-> Const readline = require ('readline');
   Const rl = readline.createInterface({
       input: process.stdin,
       output: process.stdout
   });
   rl.question ('your name?', (name) => {
       log (name);
       rl.close();
   });
```

```
-> Const buffer1 = Buffer.alloc (8)
   Const buffer2 = Buffer.from ([1,2,3,4])
```

```
-> Const queryString = require ('query string')
   const querystring = 'param = value 1&
                        param 2 = value 2';

   Const parsequery = queryString.parse
                        (queryString);
```