Node Js

A Server side platform which is built on the top of google chrome's V8 java script engine which was initially built and developed by Mr. Rayn Dal in the year 2009.

* using nodejs we could able to build faster and scalable n/w applications.

* Nodejs internally uses event driven and non-blocking I/O mechanism (or) model which makes it light weight and efficient perfect for data intensive real time applications which run on distributive system / devices.

* Nodejs is open source and cross platform run-time environment using which we could develop server side & networking based applications

* java script is the programming language through which we could able to add instructions within node js

* Node js comes with rich library of various java script modules which simplyfies development of web application using Node js.

* Node js is a combination of runtime environment and java script library.

Following are set of features makes Node js the popular server used for developing web application

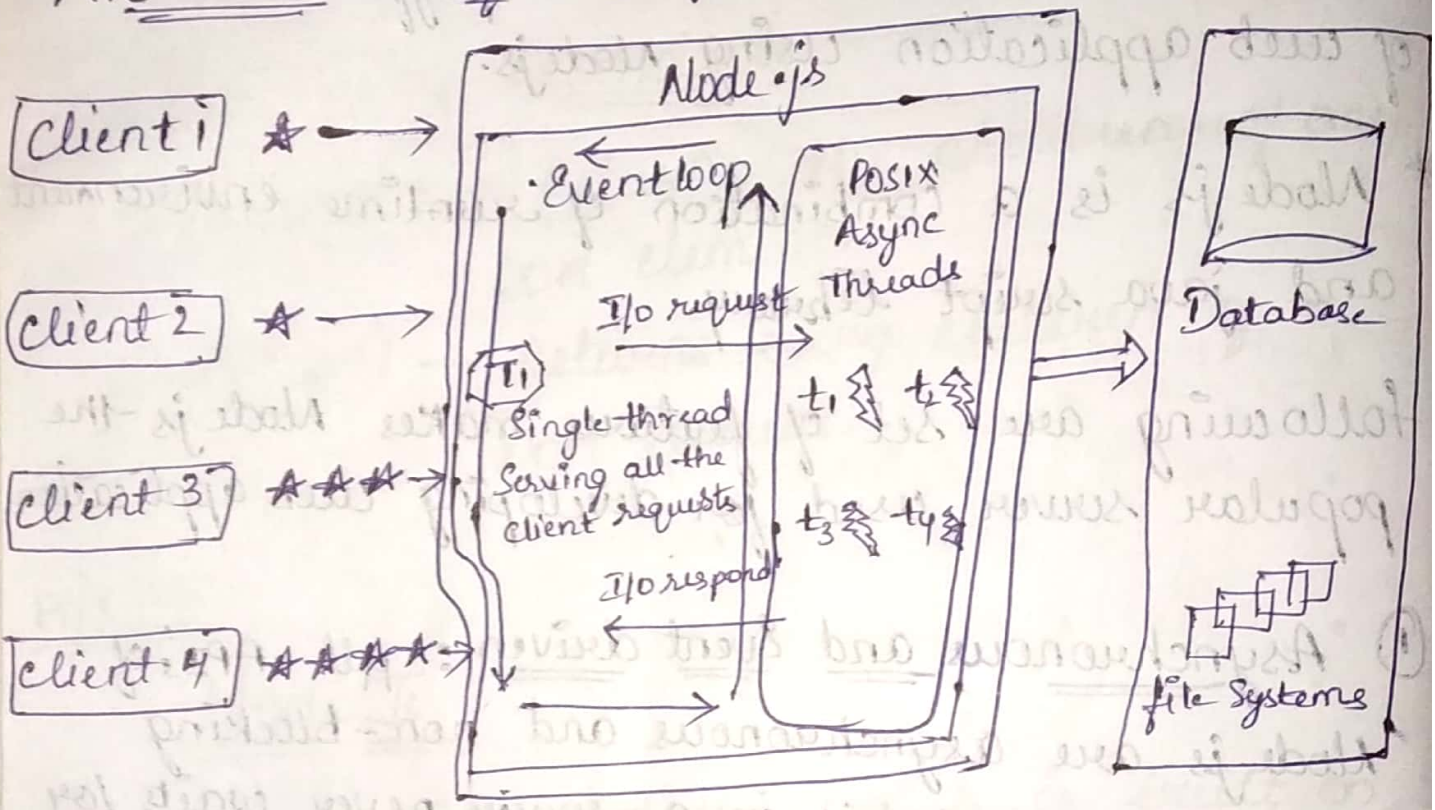
① Asynchronous and Event driven : All APIs of Node js are asynchronous and non-blocking which means Nodejs is a server never waits for any API to return data, the server moves to the next API after calling it, a notification mechanism of events of Nodejs help server to get response from previous API calls.

② Faster in processing : As node js is built on google chrome V8 java script engine, it is very faster in executing instructions

③ Single threaded but highly scalable : Internally Node js uses a single threaded model with event looping, the event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable.

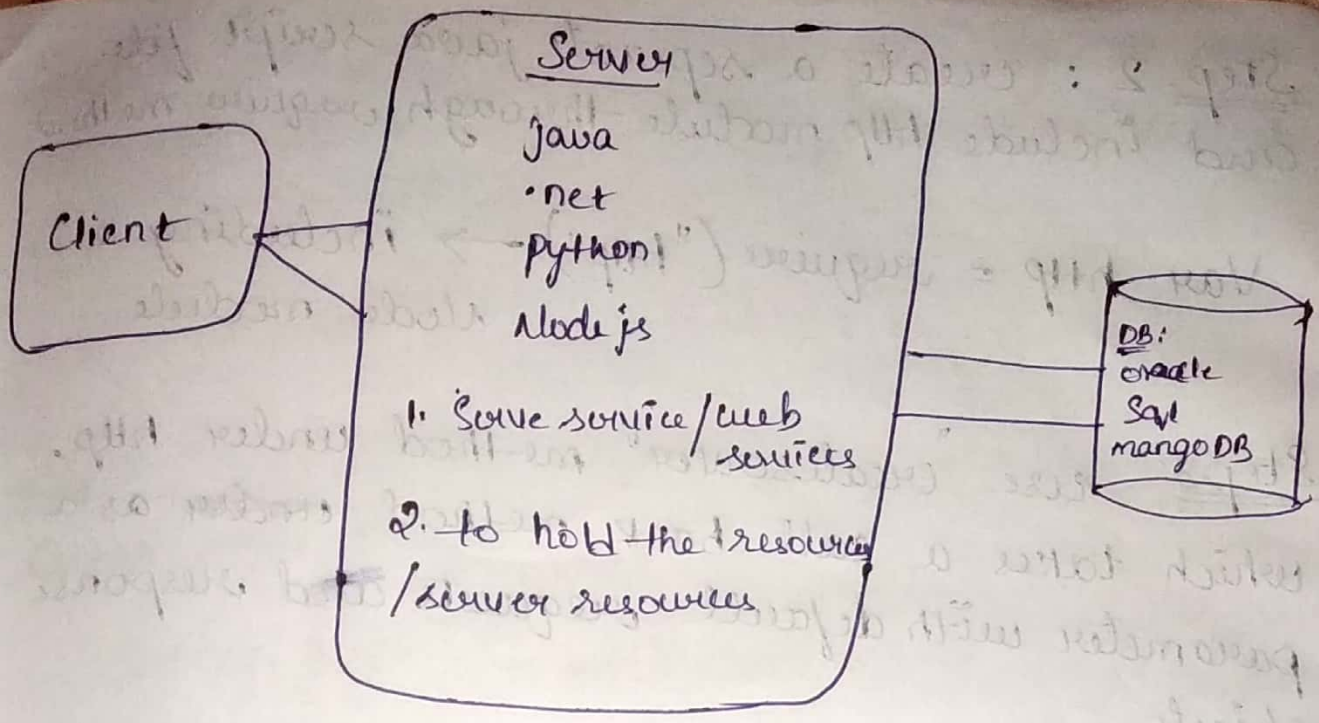
(4) No Buffering: Nodejs applications never buffer any data, these applications simply output data in chunks

Architecture of Nodejs:



NPM (Node package Manager)

* Npm is a online code repository publishing of open source Node.js code; it is also a command line utility for interacting with NPM repository for package installation and version management.
* It is through NPM we could work with installation (or) uninstallation (or) upgradation of node modules. Through NPM we could even start (or) stop Node server



19/5/20

Nodejs Module

A module in Nodejs is a simpler or complex functionality organized in single or multiple java script files which can be reused throughout the Nodejs application. Each module in Nodejs has its own context so that it doesn't interfere with other modules or doesn't interrupt the global scope

* It is through Npm we install (or) uninstall, version manager of any module can be done.

Following are the basic steps needed to create a http Node server

Step 1: Doconload ~~and include~~ Node http module through npm

Syntax: npm install http → downloading a module

Step 2 : create a seperate java script file and include http module through require method

```
Var http = require("http");
```
→ including a node module

Step3: use "createserver" method under http, which takes a call back method as a parameter with default requests and response objects

```
http."createserver"(function(req, res){
    ... // code to handle req & response.
});
```

Step4: using response object we can specify the type of content being returned from the server.

Syntax:
```
res.writeHead(200, {'content-type':
                    'text/html'});
```
using writeHead method we can specify the response status code along with the response text type

Step5: using response.write method we can write the content to be responded from the server

```
res.write("<content ....>");
```

Step 6: respond.end is an pre-defined method through which we can send request using user response.

```
res.end();
```

Step 7: Make the server to listen to a particular port no.

```
server.listen(<port number>);
```

21/5/20

Performing Node operations in Node js through fs Module:

"fs" is a pre-defined node module through which we could able to perform read, write or append operations on any file. Following are the steps to be followed while making use of fs module

Step 1: Download fs module through npm

```
npm i fs
```

Step 2: Include and create a reference for fs module

Syntax: `Var fs = require("fs");`

Step 3: use the following pre-defined methods under fs module through which we could able to do any file operations in files

1.

// Reading data from Existing file.

fs.readfile("<filename>", function(err, data){

// err object holds data if there is any error while reading file

// data holds the actual data of file

})

// creating new file using fs module

fs.appendFile("<filename>", "<content to be written>", function(err){

// err will be thrown if there is any error while creating/writing to file.

})

// Creating or writing through fs.open() method.

fs.open("filename", "<operation type>", function (err, file){

})

fs.writeFile("filename", "content", function (err) {

// write file method will replaces all file content with new one

});

Delete files:

To delete file using fs module

```
fs.unlink("filename", function(err){
  // err if there is any err while deleting files
})
```

Renaming files

```
fs.rename("filename", "newfilename", function
    ... err                          (err){
})
```

Node Express js framework

Express js is a web application framework for node js through which we could able to create flexible node js web applications. It is one of the most popular web framework provides following mechanisms.

1. It write handlers with requests with different http at different url paths
2. It integrates with view rendering engines in order to generate response by inserting data into templates.
3. It sets the common web application settings like the port to use for connecting, location of templates that are used etc...

④ Express itself is fairly minimalist developers have created compatible middleware packages to address atmost any web development problem

⑤ There are libraries to work with cookies, sessions, user logins, url params, post data, security headers and so on.

⑥ Express was released on 2010 and current latest version is 4.17.*

22/5/20

Folder structure got created through Express Generator

① node_modules: under which all the node downloaded node modules of the current app will be stored. This folder name cannot be renamed.

② package.json file: It holds a json object with all the configuration info of the current module.

(i) It holds list of dependencies node module

(ii) Name and Version number of current application.

(iii) Author name and Contact details

(iv) Starting point of current application (node start/server start) etc.

(3) Bin folder: a pre-defined folder holds a pre-defined file 'www' holds the set of instructions through which we could create http server, making the server to listen a default port number handling the errors if any.

* Bin folder and www file name can be renamed to a custom name, corresponding changes to be done at package.json

(4) Routes folder: Holds set of java script files, where each java script file represents a single webservice. Any no. of webservices can be added within node server, all the webservices js files has to be placed under routes folder.

* Index.js, users.js are the default services available under routes folder.

* Once the services are added under routes folder, corresponding route mapping or url mapping has to be done under app.js file

Steps to create a webservice under a node Express server

Step 1: create a external js file under routes folder.

Step 2: under the js file, create an instance of Express module.

Eg: Var express = require('express');

Step 3: create a router instance through a router class under express module.

Eg: Var router = express.Router();

Step 4: using the router reference, through GET/POST method add the business logic of the corresponding webservice under call back method.

Eg:
```
router.get("/", function(req, res) {
    ... // Business logic of this webservice
    res.send();
});

router.post("/", function(req, res) {
    ... // Business
    res.send();
});
Module.exports = router;
```

Step 5: under app.js file specify the router path through which web services can be accessible.