FunctionUP Trainee

Meenakshi Lodhi Rajput

Node JS

Express JS &

Mongo DB Notes

NODE - JS

* What is node js :→

→ Node is not a language, It is a Server environment. Node js can connect with database.
→ Code and syntax is very similar to Javascript But not exactly the same.
→ Node js is free, open - source.
→ Node js use Crome's V8 engine to execute code
→ Single threaded with event loop.

* Why do we use Node js :→

→ Node js is mostly used for "API".
→ We can connect the same database with Web App, Mobile App.
→ Node js is easy to understand who know Js.
→ Node is super - fast for APIs.
→ With Node and javascript, we can become full stack Developer.

Note :→ → Node js runs on the server side.
→ Javascript is run on the browser.

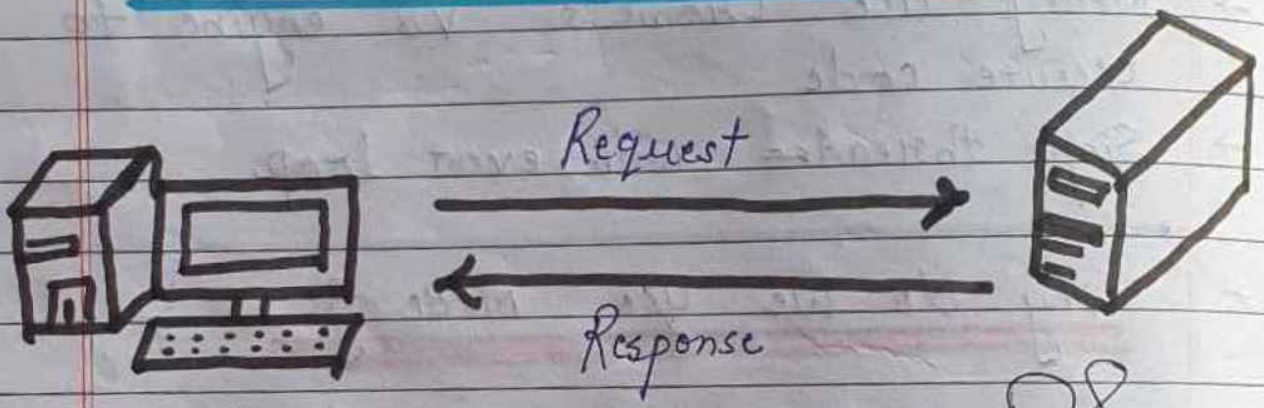
* History :→ Created by Ryan Dahl

First Release :→ May 27, 2009

→ Based on Google V8 Engine

→ written in C, C++, Javascript.

* Client and Server Sides :→



Client

↓

Client Side Programming languages

ex→ [JavaScript, HTML, CSS]

Server

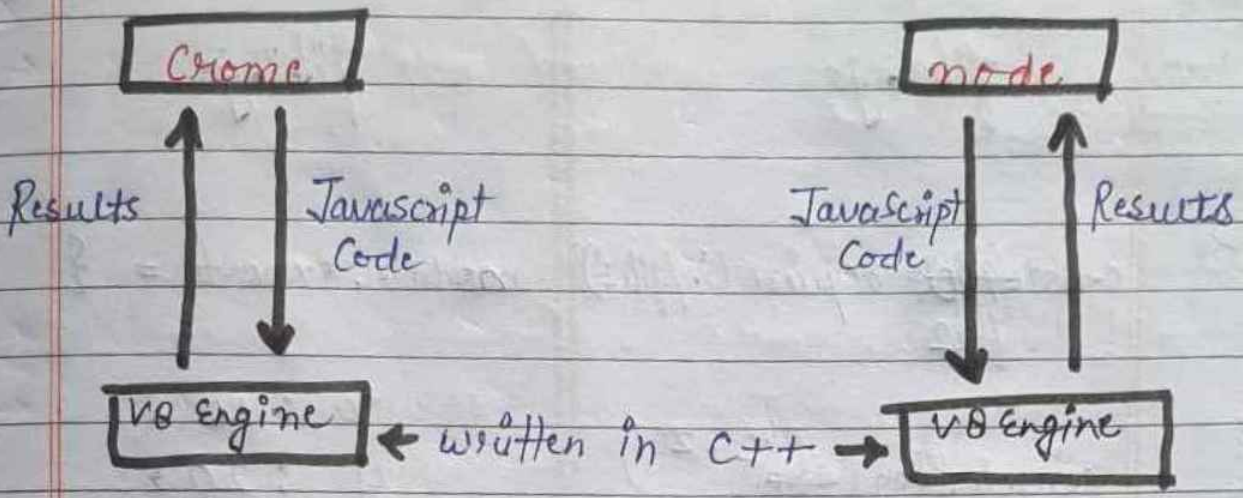
↓

Server Side Programming language

(Node.js)

Meenakshi Kapur

* How Node Js use JavaScript :→



Note→ commands to check versions of node js and npm :→

⇒ node -v
⇒ npm -v
⇒ code . (to open code editor)

* node package manager (npm) :→

→ NPM provides a public package repository, a specification for building packages, and a command line tool for working with packages.

Website
CLI (Command line Interface)
Registory ↓ Library (packages).

* Import - Export in node - js :→

file1.js

```
const file2 = require('./file2')

console.log(file2.z())
```

file2.js

```
module.exports = {
    x : 10,
    y : 20,
    z : function () {
        return 10;
    }
}
```

Note — filter function in arrays

e.g.

```
const arr = [2,4,7,6];

let result = arr.filter
        ((item) => {
    return item >= 4
})
console.warn(result)
```

* Core Module → e.g. - console, fs, Buffer, http etc.

global module
(We can use global modules without import.)
e.g. console.log(--dirname);
(--filename);

Non-global Module.
(we import non-global modules before using it)
e.g. fs

e.g. const fs = require('fs');
fs.writeFileSync("code.txt", "This is my code");
filename
content

Note→ we can declare fs variable with another name and place it below the fs.writeFileSync().

* To make basic Server :→

module.

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

function

parameters

```
http.createServer((req, resp) => {
    resp.write("<h1> Hello, I am Meenakshi");
    resp.end();                    //to end Server)
}).listen(4500);
```

port no

Note→ http module takes (req, resp) and a function.

Note:→ HTTP Method with CRUD →

- POST → Create
- GET → Read
- PUT → Update
- DELETE → Delete

⇒ The package.json file contains metadata about the project, as well as, a list of application and development dependencies.

* Package.json :→

⇒ to create package.json ⇒

command ⇒ `npm init` is used to configure a project.

→ It consists projed informations →
- Name
- Version
- Dependencies
- Licence
- Main file etc.

Packages or ~~modules~~ methods :→

→ Packages can be installed locally or globally.
→ Local packages are stored locally in a project, in the node module folder.
→ Global package are stored globally on system.

→ Typically, local packages are code libraries used by project.

→ And global packages are executables used to perform some operations on a project such as running tasks

→ Local packages are available only within their specific project, and global packages are available System wide.

---

* To install & unstall Packages :-

→ The npm ~~install~~ program is used to manage packages.

→ Packages can be installed with the install command, and unstalled with the uninstall command.

e.g. npm install nodemon -g
npm ~~in~~ i nodemon

npm uninstall nodemon -g
npm unstall nodemon

Note → -g, installs & unstalls the package globally.

e.g. const colors = require('colors');
console.log("package", bgBlue);
(/red, /yellow etc.)
↑ we can use

* To recover package.json = npm install.

Note → we should not push node_modules folder.
⇒ we can ignore this folder by .gitignore file
/node_modules

* Nodemon (time saving module) :-

→ To install

commands npm i nodemon -g

→ npm i nodemon sudo -g // for mac or ubuntu.

* nodemon is a tool that helps develop Node.js based application by automatically restarting the node application when file changes in the directory are detected.

* Node js is async

e.g. 1. run first script
2. run second script (complex data)
→ --- it will not wait to finish 2nd script
3. run third script
4. continue ---

* To Make a simple API →

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

http.createServer ((req, resp) => {
resp.writeHead (200, { 'Content-Type':
                       'application/json' });
resp.write (JSON.stringify ());
resp.end ();

}).listen (5000);
```

(200 ↓ status code; 'application/json' ↓ Data in JSON format)

* To set input from command line :→

Note→ `console.log (process);`

↓ object

(or) `console.log (process.argv);`

↳ argument vector

(or) `console.log (process.argv[2]);`

↑ to get 2nd value of process object.

* To create file :→

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

const input = process.argv;
fs.writeFileSync (input[3], input[4])
```

input[3] ↑ filename

input[4] ↑ file data

Note→ To delete file :→ We will use if-else condition

⇒ fs.unlinkSync (input[3]).

* To remove ⇒ remove <~~filenam~~ <text>
(or)
<filename>

* To show file list :→

→ to create multiple files using loop :→

```
Const fs = require ('fs');
const path = require ('path');
```

```
const dirPath = path.join(__dirname, 'files');

for (i=0; i<5; i++)
  {
    fs.writeFileSync(dirPath + "/hello" + i + ".txt",
                     "a different test files");
  }
```

⇒ to show files :→

```
fs.readdir(dirPath, (err, files) => {
  files.forEach((item) => {
    console.log("file name is", item)
  })
})
```

---

* CRUD operations with file System :→

→ Create file
→ Read file
→ Update file
→ Rename file
→ Delete file.

⇒

```
const fs = require('fs');
const path = require('path');
const dirPath = path.join(__dirname, 'crud')
const filePath = `${dirPath}/data.text`;
```

folder name

filename

```
fs.writeFileSync(filePath, 'This is a simple text file');
                                    // to create file

fs.readFile(filePath, 'utf8', (err, item) => {
  console.log(item)
})                                  // to read file

fs.appendFile(filePath, 'and file name is data.txt', (err)=>{
  if(!err) console.log("file is updated")
})
                                    // to update file

fs.rename(filePath, `${dirPath}/world.txt`, (err)
  if(!err) console.log("file name is updated")
})
                                    // to rename file
```

renamed

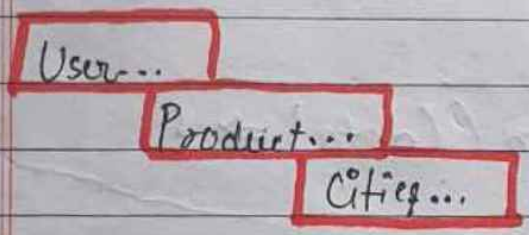
Note ⇒ buffer = temporary memory location

fs.unlinkSync (`${dirPath}/world.txt`)
// to delete file.

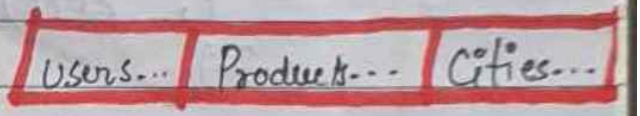
---

* Asynchronous and Synchronous :→

In asynchronous, Second task do not wait to finish first task.



In synchronous operations tasks are performed one at a time



Node ⇒ Javascript and Node are Asynchronous.

eg.

```
console.log("start exe...");
setTimeout(() => {
        console.log("logic exe...")
}, 2000)
    console.log("complete exe...")
```

* To handle asynchronous data in Node.js

→ we will use promises.

e.g.

```
let a = 20;
let b = 0;

let waitingData = new Promise ((resolve, reject) => {
    setTimeout (() => {
        resolve (30)
    }, 2000)
})

waitingData.then ((data) => {
    b = data;
    console.log (a+b)
})
```

API = Application Programming Interface

* **Express JS** :→ It's a framework built to create Node.js web based applications.

⇒ **Core features of Express framework :→**

→ Allows to set up middlewares to respond to HTTP requests.

→ Defines a routing table which is used to perform different action based on HTTP method and URL.

→ Allows to dynamically render HTML Pages based on passing arguments to templates.

⇒ `npm install express. --save`

→ To install express locally in the node.js- workspace directory, and save it in the dependencies list.

`npm install express -g --save.`

→ To install the express framework globally using npm if we want to create web application using node terminal.

e.g.

```
const express = require('express');
const app = express();

app.get("/home", (req, res) => {
  res.send("welcome, This is a Home Page");
});

app.get("/about", (req, res) => {
  res.send("welcome, this is about Page");
});

app.get("/help", (req, res) => {
  res.send("welcome, this is help page");
});
```

(display)

* Render HTML and JSON :→

```
app.get("/about", (req, res) => {
  res.send("
    <input type="text" placeholder="user name">
    <button>Click Me</button>
  ");
```

⇒ app.get("/help", (req, res) => {
res.send([
{
name: "Meenakshi",
Id: 101
},
{
name: "Rajput",
Id: 102
}
]);

To create links :->

⇒
app.get("/home", (req, res) => {
res.send(`
<h1> welcome to Home page </h1>
 go to about page
`);
});

→ app.get("/about", (req, res) => {
res.send(`
<input type = "text" placeholder = "user name"/>
<button> Click Me </button>
 go to Home Page `));

* To Create HTML Page in Node.js:

→ Create HTML pages (index.html, contact.html

then →

```
const express = require ('express');
const path = require ('path');

const app = express();
const publicPath = path.join (__dirname, 'pub
app.use (express.static (publicPath));

app.listen (5000);
```

* To remove extension from URL :→

```
app.get ('', (req, res) => {
    res.sendFile (`${publicPath}/index.html`)
})
```

→ to show by default page :→

```
app.get ('*', (_, res) => {
    res.sendFile (`${publicPath}/help.html`)
})
```

* Template Engine :→

→ It is used to create dynamic pages
eg. ejs template.

→ In index.js file setup →

app.set ('view engine', 'ejs');

⇒ Then create a page.
folder (view) → profile.ejs (file)

⇒ write html code in ejs file.

eg
```
<!DocType html>
<html lang = "en">
<head>
  <title> Profile Page </title>
</head>
<body>
 <h1> Welcome <%= user.name %></h1>
 <h3> Email : <%= user.email %></h3>
 <h3> City : <%= user.city %> </h3>
</body>
</html>
```

=> In index.js :-

```
app.get('/profile', (req, res) => {
  const user = {
    name: 'Meenakshi Lodhi Rajput',
    city: 'Meerut'
  }
  res.render('profile', {user});
});
```

Note:- we can create dynamic html pages.

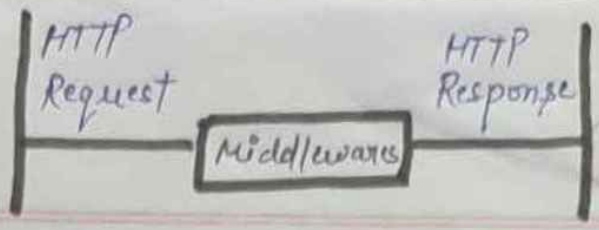
---

* Middleware :->

-> Middleware functions are functions that have access to the request object (req), the response object (res), and the next function in the application's request-response cycle.

The next function is a function in the Express router which, when invoked, executes the middleware succeeding the current middleware.

Middleware working →



e.g.

```
const express = require ('express');
const app = express();

const reqFilter = (req, res, next) => {
    next();
}
app.use(reqFilter).
```

⇒ Types of Middleware :→

- Application - level middleware
- Route - level middleware
- Error - handling middleware
- Built - in middleware
- Third - party middleware.

Meena Ke Code Kaafi

mongeDB

→ MongeDB is a NoSQL database.
→ The Data stored in a collection.
→ collection don't have row and columns.
→ Data is stored in the form of object.

e.g.

```
{ _id: "....";
  name: "Meenakshi Lodhi Rajput",
  address: {
      street : "123 street",
      city : "Meerut",
      state : "UP",
      zip : "12345"
      }
}
```

NOTE.

* CRUD operations in Mongo DB. :→
(CREATE, READ, UPDATE, DELETE)

* Database Commands :→

→ To view all database :→ show dbs

→ To Create a new or switch databases :→
→ use dbName.

→ To view current Database :→ db

→ Delete Database :— db.dropDatabase()

* Collection Commands :→

→ To show collections :— show collections
→ To create a collection :— db.createCollection('name')
→ To Drop a collection :— db.comments.drop()
collection name.

* Document Commands :→

→ To show :→ db.collectionName.find()

etc.....

* To connect database :→

→ install mongodb ⇒ npm i mongodb

index.js

```
const {MongoClient} = require ('mongodb');
const url = 'mongodb://localhost:27017';
const database = 'functionUp' ← database name
const client = new MongoClient (url);

async function getData ()
{
  let result = await client.connect ();
  let db = result.db (database);
  let collection = db.collection ('product');
  let response = await collection.find ({}).toArray ();
  console.log (response);
}
getData ();
```

collection name

Note

* Basic API with MongoDB :→

```
const dbConnect = require("./mongodb");
const express = require('express');
const app = express();
```

→ get method :→

```
app.get('/', async (res, req) => {
let data = await dbConnect();
data = await data.find().toArray();
res.send(data);
});
```

→ post method :→

```
app.post('/', (req, res) => {
  res.send({name : 'Meenakshi'})
})
```

→ put method :→

```
app.put("/", (req, res) => {
  res.send({lastName : 'Lodhi Rajput'})
})
```

→ delete method :→

```
app.delete ("/:id", (req, res) => {
  res.send ("done")
})

app.listen (5000)
```

* Mongoose :→ npm i mongoose

→ To define a model, derive a custom schema from Mongoose's Schema and compile the schema to a model.

e.g.
```
const mongoose = require ('mongoose');
Const main = async () => {
await mongoose.connect ("mongodb://localhost:27017/e-com");
const productSchema = new mongoose.Schema({
    name : string
    price : Number
});
```

Schema →

* model :→ It connects nodejs with mongoDB by using schemas.

e.g.
```
Const main = async () => {
Const product = mongoose.model('product', productSchema);
let data = new product ({ name: "max 3", price: 200 });

const result = await data.save();
console.log(result);
}
```

Note:→ then write schemas in mongoDB.
→ we can apply CRUD operations using mongoose.
→ And creati APIs in mongoose.

* OS Module in Node js :→

```
const os = require ('os');
```
↳ to show system configuration

```
// console.log (os.arch());
```

to show free memory ↓ ↑ to divide memory

```
// console.log (os.freemem()/(1024*1024*1024));
// console.log (os.totalmem()/(1024*1024*1024));

// console.log (os.hostname()); → to know username
// console.log (os.platform());
                    ↳ windows etc.

// console.log (os.userInfo()); → to show user information
```

Other Topics →

* Events and Event Emitter in Node js :-
* REPL (Read- Eval- Print - Loop) :→