**Node JS with Server Setup**

* Node is a JavaScript Runtime
* Node js is open source server environment
* It allows you to run JS on server
* Its totally free and its also platform independent
* It runs on windows, linux, Mac OS, Unix etc

**Why Node JS**

* Send the task to the computer file system
* Node js can create, open, read, write, delete and close files on server
* Generate dynamic web pages
* Connect database and process data cruds

**History**

Node. js was **written initially by Ryan Dahl in 2009**, about thirteen years after the introduction of the first server-side JavaScript environment, Netscape's LiveWire Pro Web. The initial release supported only Linux and Mac OS X. Its development and maintenance was led by Dahl and later sponsored by Joyent.



**Node JS Modules**

* Node module is JS library
* A set of functions, assets etc for future apps without installation
* There are three types of modules   
  -> custom   
  -> built In   
  -> developers package module

* **Create a node module**   
  In any js file just we need to export our modules and then require other file to access out modules   
  - Create a JS file for building modules   
  - write some modules elements   
  - now export your module element   
    
  module.exports = our modules elements   
    
  - now require your module to other file for access   
  const data = require(‘module path’);
* Modules scaffolding   
  To add more module elements in one export we can use scaffolding   
   const obj = {};
* Module with ES6+ syntax

**File System Module**

* Its a build in modules
* Read, write, create, update, delete, rename any file
* We can use also sync for file system process
* First require http module   
  const fs = require(fs);
* Now Create any file with data write   
  fs.writeFile(‘file\_name’ ,’set\_data’, (errors, data) => {  
    
  });
* Now Rename any file name   
  fs.rename(‘old\_file\_name’ ,‘new\_file\_name’, (errors) => {  
    
  });
* Now Delete any file   
  fs.unlink(‘file\_name’, (errors) => {  
     
  });
* Now Read any file data with data read   
  fs.readFile(‘file\_name’ , (errors, data) => {  
     
  });
* Now append some data to any file   
  fs.appendFile(‘file\_name’ , ‘new\_data’ (errors, data) => {  
     
  });
* Now open a file   
  fs.open(‘file\_name’ , ‘mode’, (errors, file) => {  
     
  });
* Create a folder   
  fs.mkdir(‘path’ , {} , (errors, file) => {  
     
  });

**Path Module**

* Any kind of path related work
* Require path first   
  const path = require(‘path’);
* Some path methods   
  - basename   
  - dirname   
  - extname   
  - parse   
  - join
* How to use   
  First get your app url  
   path.method(url);

**URL Module**

* Any kind of URL related work
* Require path first   
  const url = require(‘url’);   
    
  const req = url.parse(‘app\_url’, true);
* Get url request params   
  - host   
  - hostname   
  - pathname   
  - search   
  - searchParams

**HTTP Module for Create a server**

* It is a built in module
* It is used to create a server
* First require http module   
  const http = require(‘http’);
* Now create a server with listener   
  http.createServer().listener(post\_number);
* Now server callback with request & response   
  http.createServer( (req, res ) => {  
   res.write(‘ server starting ’);  
   res.write(‘ server starting ’);  
   return res.end():   
  }).listener(port\_number, ()=> {  
   // After server load   
  });
* Now create a basic router with http request   
  http.createServer( (req, res ) => {  
   req.url use it to make a route   
  }).listener(port\_number);

**Developer Package**

* Reusable module or library management system
* Npm or yarn is a package manager for node JS
* We can get any developer module from npm or yarn
* Install any package by **npm**   
  npm install package\_name   
  npm i package\_name   
  npm i package\_name --save   
  npm i package\_name --save-dev   
  npm i package\_name -D
* Install any package by **yarn**    
  yarn add package\_name
* npm init for package.json file   
  npm init
* npm package default setup   
  npm init -y

**Colors Package**

* Colors package is used for set terminal text color
* Now install nodemon   
  npm install colors
* Now setup modemon in package json   
  “Start” : “nodemon entry filename”

**Nodemon Package**

* Nodemon is used for server reload after save
* Now install nodemon   
  npm install -g nodemon
* Now setup modemon in package json   
  “Start” : “nodemon entry filename”

**Setup Environment variables**

* Our system configuration store in env
* First install dotenv package   
  npm install dotenv
* **Now add config**   
  Import dotenv from ‘dotenv’;  
  dotenv.config();
* **Now create a .env file in your root & set variables**   
  .env
* **Access env variables**   
  process.env.VAR\_NAME

**Essential Array Methods**

* Find
* Filter
* findIndex
* lastIndexOf
* Some
* Join
* Split
* Reduce

**Get Data with POST Request**

* We can receive data from req with event   
    
  let body = ‘’;  
  req.on(‘data’, (chunk) => {  
   body += chunk.toString();  
  });  
  req.on(‘end’ () => {  
   totalbodyData = body;   
  });

**ES5 & ES6 export import**

* There are two type of module in JS   
  - common JS   
  - module
* Common JS is default system for ES5
* Module system is modern JS Syntax
* Set module JS system   
  Go to package.json file and set type   
  “type” : “module”
* Now export module by using ES6+ system   
  export const ageCal = () => { }
* Import module from ES6 +  
  Import { ageCal } from ‘module.js’;
* Now export default   
  export default ageCal ;
* esm package for replace .js syntax   
  npm install esm

**Status Code**

* '100': 'Continue',
* '101': 'Switching Protocols',
* '102': 'Processing',
* '103': 'Early Hints',
* '200': 'OK',
* '201': 'Created',
* '202': 'Accepted',
* '203': 'Non-Authoritative Information',
* '204': 'No Content',
* '205': 'Reset Content',
* '206': 'Partial Content',
* '207': 'Multi-Status',
* '208': 'Already Reported',
* '226': 'IM Used',
* '300': 'Multiple Choices',
* '301': 'Moved Permanently',
* '302': 'Found',
* '303': 'See Other',
* '304': 'Not Modified',
* '305': 'Use Proxy',
* '307': 'Temporary Redirect',
* '308': 'Permanent Redirect',
* '400': 'Bad Request',
* '401': 'Unauthorized',
* '402': 'Payment Required',
* '403': 'Forbidden',
* '404': 'Not Found',
* '405': 'Method Not Allowed',
* '406': 'Not Acceptable',
* '407': 'Proxy Authentication Required',
* '408': 'Request Timeout',
* '409': 'Conflict',
* '410': 'Gone',
* '411': 'Length Required',
* '412': 'Precondition Failed',
* '413': 'Payload Too Large',
* '414': 'URI Too Long',
* '415': 'Unsupported Media Type',
* '416': 'Range Not Satisfiable',
* '417': 'Expectation Failed',
* '418': "I'm a Teapot",
* '421': 'Misdirected Request',
* '422': 'Unprocessable Entity',
* '423': 'Locked',
* '424': 'Failed Dependency',
* '425': 'Too Early',
* '426': 'Upgrade Required',
* '428': 'Precondition Required',
* '429': 'Too Many Requests',
* '431': 'Request Header Fields Too Large',
* '451': 'Unavailable For Legal Reasons',
* '500': 'Internal Server Error',
* '501': 'Not Implemented',
* '502': 'Bad Gateway',