**Phase 1**

**Day 1**

**24-05-2021**

**Web Developer**

**MERN Stack :**

**Mongo DB/MySQL Express Module React JS Node JS**

**Phase 1 :**

**Agile, Git, HTML,CSS, JavaScript**

**Project**

**Simply blogging**

**Phase 2**

**ES6 Features and React JS with Redux.**

**Project, Node JS Module**

**Building a To-Do App**

**Phase 3**

**Node JS Modules file handling, http module,**

**Express module (REST API),**

**Mongo DB,**

**Connecting Mongo DB data through Node JS**

**MongoDB and Mongoose**

**Socket programming**

**Project**

**Chatbox**

**Phase 4**

**Testing modules**

**Docker**

**Jenkin**

**AWS Overview ( ES3, RDS, S3 etc).**

**SVN : Java**

**Client1 or dev1 or Local Repository**

**Client2 or dev2 or Local RepositoryServer**

**Repository**

**Client3 or dev3 or Local Repository**

**GIT : Git is known as Sub Version Control System.**

**It is use to records the changes done in file or folder or application or projects.**

**Online shopping application**

**Git provide distributed sub version control system.**

**Login Module**

**Customer Module**

**Order Module**

**Manager module**

**Git commands**

**To make the folder as local repository as**

**git init**

**if you want to check the last command status we have run the command as**

**git status**

**If we want to move file from untrack phase to staging area. We have to run the command as**

**git add filename**

**if we want to move this file from staging area to local repository (folder).**

**git commit –m “Commit Message”;**

**git config --global user.email "you@example.com"git config --global user.name "Your Name"**

**git config –-global user.email “**[**abc@gmail.com**](mailto:abc@gmail.com)**”**

**git config -–global user.name “userName”**

**github : github is a part of Microsoft which provide remote repository for the git.**

**AWS**

**Azure**

**Google cloud**

**Oracle cloud**

**Etc**

**git add . ( all files and folder) adding the staging area.**

**Command to connect local repository to remote repository**

**git remote add origin URL**

**now to push the data to remote repository we have to use the command as**

**git push –u origin HEAD (u means upstream and HEAD last commit in that branch).**

**If we do any changes in local repository means created new file, updated existing file or deleted files**

**git add .**

**git commit –m “Message for that task”**

**git push –u origin HEAD**

**Steps :**

**git init**

**git status**

**git add .**

**git status**

**git commit –m “message for task”**

**git status**

**data store in local repository**

**git remote add origin URL**

**git push –u origin HEAD**

**next time or again and again whenever you do any changes in project or folder.**

**git add .**

**git commit –m “commit for task”**

**git push –u origin HEAD**

**Another way to make local folder as a local repository**

**git clone URL**

**git branch : A branch is simply light weighted movable pointer which hold more than one commit details.**

**When we create local repository we can see default branch ie master or main.**

**Syntax to create the branch**

**git branch branchname**

**To view branch names**

**git branch**

**To move use-defined branch**

**git checkout branchName**

**To delete the branch**

**git branch –D branchName**

**Command to create the branch and switch the branch**

**git branch branchname**

**git checkout branchName**

**Or**

**git checkout –b branchName**

**Manager**

**Manager create sample code and push in remove repository.**

**Ali Ajay**

**Git clone done by both developer**

**Login Application**

**BranchName BranchName**

**Ali\_Login Ajay\_Application**

**Git clone URL**

**: First time to download or clone remote repository in local machine.**

**And**

**git pull**

**: This command is use to updated new changes from remote repository to local existing repository.**

**git pull : we have run this command in default branch ie main or master.**

**git push : we have to push use-defined branch to remote repository**

**If we want to check all commit details**

**git log**

**Day 2**

**25-05-2021**

**HTML,CSS,JS ES5 and ES6. Bootstrap.**

**Web Application**

[**https://www.google.com**](https://www.google.com) **: URL**

**http: hypertext transfer protocol : set of rules**

**s : secure**

**www : world wide web**

**google : domain**

**com : commercial**

**Uniform Resource locator**

**Req(https/http)---🡪**

**Client Server**

**🡨-----Res (http/https)**

**HTML/HTML5**

**CSS/CSS3**

**JS (JavaScript)**

**HTML/HTML5--🡪 Display the content on web page**

**Display simple message**

**Add the image**

**Hyper link**

**List**

**Table**

**EmployeeDetails**

**Id Name Salary**

**100 Raju 12000**

**101 Ramesh 14000**

**Attribute : Attribute is known as properties of tags. Attribute we have to write in opening in the form of key-value pairs. Value may be single quote or double quote or without quote.**

**<tagName name=”value”> </tagName>**

**Forms**

**HTML/HTML5 Form tag by default method consider as get.**

**If method is get the information send through URL in the form of query params. Like**

**URL?key=value&key=value&key=value**

**In Get method data send through url ie through head part. And body is empty.**

**Post method**

**The data send through body part of request.**

**CSS/CSS3---🡪 Apply presentation logic or look and feel for that content.**

**CSS : Cascading Style Sheet :**

**CSS provide lot of pre-defined properties which help to apply good look and feel application.**

**Using CSS we can achieve separation concern. Means actual contents and formatting style of separate.**

**Types of CSS files**

1. **Inline CSS**
2. **Internal CSS or embedded CSS**
3. **External CSS**

**Inline CSS**

**<tagName style=”property:value;property:value”>**

**</tagName>**

**Internal or Embedded CSS**

**Syntax**

**In between head tag we have to style stag**

**<style type=”text/css”>**

**selector {property : value;property:value}**

**</style>**

**Type of selector**

1. **Universal selector : \***
2. **Specific selector : tagname p, h1, to h6, div, span etc**
3. **Multi specific selector : tagname,tagName,tagName{property : value}**
4. **Class selector (local class selector ) :**

**tagName.className {property:value;property:value;}**

1. **Class selector ( global class selector )**

**.className {property:value;property:value;}**

1. **Id selector**

**#idName {property:value}**

1. **Child selector**

**parentNametag childTagName {property:value;}**

**Class Vs id**

**class : group of tags. So we can write more than one tag same class.**

**id : if we want to make unique ness between two tag when we have to use id.**

**<div>**

**<p class=”abc” id=”p1”>First para</p>**

**<p class=”xyz” id=”p2”>Second para</p>**

**<p class=”abc” id=”p3”>Third para</p>**

**<p class=”xyz” id=”p4”>Fourth para</p>**

**</div>**

**Using the id in JS we can read, write or update html contents(DOM).**

**External CSS file**

**JavaScript --🡪 Action(Events) on content or programming on web page or contents.**

**UI Developer : IDE**

**Micro soft : VSCode**

**MEAN Stack and MERN Stack**

**JavaScript : ES5**

**JavaScript was object based interpreter scripting language.**

**Object based or prototype based style Vs object oriented**

**OOPs : Object Oriented Programming system**

**Object, class, Inheritance, Polymorphism, Encapsulation, Abstraction etc.**

**Like C++, Java, Python, C# etc**

**Interpreter Vs Compiler**

**Both are translator : converting one format to another format.**

**Interpreter : it check the code line by line and convert : JavaScript**

**Compiler : Convert whole code at time. C, Java**

**Scripting Vs Programming**

**In programming language it generate another file which help to run the program like byte code, exe file or object file.**

**But in scripting directly we get the output.**

**JavaScript tags syntax**

**<script type=”text/JavaScript”> opening tag**

**</script> closing tag**

**This tag we have to write in between head tag or body tag of html we page.**

**We can write more than one tag in one html page.**

**If we want to display message through JavaScript we have to use pre-defined object ie document and write if function part of document object.**

**In JavaScript it is not mandatory every statement end with semicolon.**

**Variable and data types**

**To declare the variable in JavaScript we are using var keyword**

**var variableName**

**var a;**

**var n=10; // number type consider**

**var m=10.10; // number type consider**

**var fname=”Ajay Kumar”; //string type consider**

**var result = true; //Boolean type consider**

**var obj = new Date(); //object reference consider**

**document.write(a);// undefined (data type consider)**

**Operator :**

**Arithmetic Operator : +, -, \*, /, %**

**Conditional operator : >, >=, <, <=, ==, ===, !=**

**Logical operator : &&, ||, !**

**Assignment operator : =**

**Increment and decrement : ++, --**

**Ternary operator : condition ? true : false**

**typeof function or operator :**

**Day 3**

**26-05-2021**

**If statements**

**Simple if**

**if(condition){**

**true block**

**}**

**if else**

**if(condition) {**

**}else {**

**}**

**Nested if : if within another if.**

**if(condition) {**

**if(condition) {**

**}else {**

**}**

**}else {**

**if(condition) {**

**}**

**}**

**If else if or if ladder**

**if(condition) {**

**}else if(condition) {**

**}else if(condition) {**

**}else {**

**}**

**switch :**

**syntax**

**switch(variableName) {**

**case label1: block1;**

**break;**

**case label2: block2;**

**break;**

**case label3: block3;**

**break;**

**default default bock**

**break;**

**}**

**switch, case, break and default are keywords.**

**Looping : it is use to execute the set of statement again and again till the condition becomes false.**

**While loop**

**Do while loop**

**For loop**

**Initialization start and end**

**Condition true**

**Coding**

**Increment / decrement**

**Normal Function**

**Function is use to write set of instruction to perform a specific task.**

**Syntax to write normal function**

**User-defined function**

**function functionName(parameterList) {**

**}**

**JavaScript function can return any type of values without return keywords or return type.**

1. **Function no passing parameter and no return type.**
2. **Function passing parameter but no return type.**
3. **Function passing parameter and return type.**
4. **Function no passing parameter but return the value.**

**Pre-defined global function**

1. **alert(“Message”): pop up message.**
2. **prompt(): This function help to take the value through keyboards.**
3. **parseInt() : it is use to convert string to integer.**
4. **parseFloat() : it is use to convert string to float.**
5. **eval() : it is use to convert string to number (with or without decimal).**
6. **confirm() : This function display pop message with 2 button ok and cancel. If you click on ok button it return true if click on cancel it return the false.**

**do {**

**alert 1: Add 2 Sub**

**prompt(): to receive choice**

**switch(){**

**case 1 take x and y value and display sum**

**case 2 take x and y value and display sub**

**}**

**Do you want to continue : confirm**

**}while()**

**alert(“thank you”)**

**events : event is a interaction between user and html tags or components or form tags.**

**Event provide bridge between html and JavaScript code.**

**Type of events**

**In JavaScript all events are start with on prefix followed by event name.**

**Type of events**

**onClick : button or p or div tags**

**onDblClick : button or p or div tag**

**onMouseOver : image tag**

**onMouseOut : image tag**

**onKeyUp : AJAX Google search engine**

**onKeyDown : chatting application**

**onFocus : enter in text field**

**onBlur : exit from text field**

**onChange : dropdown**

**onSubmit : Validation Form validation**

**onLoad : body tag**

**onUnload : body tag**

**Dropdown**

**<select name=”city” onChange=””>**

**<option value=”Bangalore”>Bangalore</option>**

**<option value=”Delhi”>Delhi</option>**

**<option value=”Mumbai”>Mumbai</option>**

**</select>**

**etc**

**Listener : Listeners are function which help to listen the generated events.**

**To generate the events we have to register the event on dom(Document Object Model) or HTML tags.**

**DOM : Document Object Model**

**index.html**

**In browser memory dom hierarchy will created.**

**<html>**

**<head>**

**<title>Message</title>**

**</head>**

**<body>**

**<p>Welcome to Web Page</p>**

**<script type=”text/JavaScript”>**

**for(var i=0;i<10;i++){**

**document.write(“Welcome”)**

**}**

**</script>**

**</body>**

**</html>**

**DOM means HTML tags**

**HTML**

**HEAD BODY**

**TITLE P**

**TextNode –Message TextNode: Welcome to**

**If we want to write (add), remove, update dom elements or tags dynamically.**

**Read, Write and Update : dynamically**

**DOM API : Document Object Model (Application Programming interface).**

**Java, Asp.net, Python, C/C++, JavaScript they provide DOM API to read, write and update data in html dynamically.**

**DOM API using JavaScript**

**If we want to read text field value using JavaScript**

**1st way**

**document.formname.textfieldname.value**

**2nd way**

document.getElementsByName("user")[0].value;

**3rd**

**document.getElementById(“user”).value**

**Form Validation**

**Using JavaScript**

**Using HTML5 Features**

**OOPs using ES5**

**objects :**

**object : any real world entity**

**properties or state – have –fields/variables**

**Person**

**behaviour –do/does -- functions / methods**

**Bank**

**Animal**

**Car**

**Employee**

**Etc**

**In JavaScript object are divided into two types.**

**Pre-defined objects**

**JavaScript follow object hierarchy**

**Object -🡪 properties or state**

**behaviour**

**object 🡪 property or state**

**behaviour**

**object -🡪property**

**behaviour**

**object**

**BOM : Browser object Model**

**window is top most object in BOM Hierarchy**

**DOM : Document object Model**

**document is top most object in DOM Hierarchy**

**User-defined objects**

**Day 4**

**25-05-2021**

**BOM : Browser Object Model :**

**In BOM hierarchy window is a top most object.**

**window.alert(“Welcome to JavaScript”)**

**or**

**alert(“Welcome to JavaScript”);**

**window.prompt(“Enter the name”)**

**or**

**alert(“Enter the name”)**

**window.document.write(“Welcome to JavaScript”)**

**or**

**document.write(“Welcome to JavaScript”)**

**synchronous and asynchronous communication**

**Statement level**

**L1**

**L2**

**L3**

**L4**

**Function call**

**dis1();**

**dis2();**

**dis3();**

**dis4();**

**Promise()**

**1st Req**

**2nd Req**

**3rd Req**

**4th Req**

**Client Server**

**In JavaScript window object contains setTimeout(), setInterval() and clearInterval() function. Which help to achieve asynchronous operations.**

**document.write(“1st Statement”) syn**

**document.write(“2nd Statement”): asyn**

**document.write(“3rd Statement”) asyn**

**document.write(“4th Statement “) syn**

**CSS style property in DOM CSS property**

**color:red color=red**

**font-size:24px; fontColor=24px;**

**background-color:yellow backgroundColor=yellow**

**Creating User-defined object using ES5 style (function style)**

**Up to ES5 no class keyword to create the objects. To create the user defined object we are using function itself.**

**Object : any real world entity**

**properties**

**behaviour**

**object is a concept.**

**If we want to describe the object we have to take the help of function in ES5 JavaScript.**

**In function style object creation we can add dynamic property.**

**ES6 Features**

**ES : ECMA :European Computer Manufacture Association**

**ECMA Script : it is a concept.**

**ES5**

**ES6 and ES7 new**

**JavaScript is a one of the implementation of ES5 as well as ES6.**

**TypeScript is a super set of JavaScript which support all features of ES6. Where Javascript support few features of ES6 or partial support of ES6.**

**TypeScript is also one of the implementation of ES6 Features.**

**Adding external JavaScript file**

**From ES6 onwards we can declare variable using var, let and const keyword.**

**Using var we can re-declare same variable once again. But using let we can’t re-declare same variable once again.**

**Using var we can declare global variable but using let we can declare local or block scope.**

**const : if we declare variable using const keyword we can’t change the value of that variable.**

**If we want to declare constant value using const keyword.**

**Normal function call may before or after function declaration.**

**But expression style calling must be after declaration.**

**IIFE : immediately invoked function expression**

**Syntax**

**1st 2nd**

**(functionBody)(functionCall)**

**This type of function we can call immediately and only once.**

**Callback function : passing the function it self or function name or function body to another function as a parameter is known as callback function.**

**Array with retrieve value using different ways**

**array : array is use to store more than one value of same or different types.**

**syntax to create the array**

**Literal style**

**let num1 =[100,200,300,400,500];**

**Object style**

**let num2 = new Array(100,200,300,400,500);**

**Day 5**

**30-05-2021**

**arrow function :**

**arrow function short cut syntax for the anonymous function with express style.**

**Syntax**

**Let/var functionName = ()=>document.write(“Arrow function”)**

**By default arrow function return result without return keyword.**

**If we want to write more than one line code in arrow function we have to use curly braces with or without return keyword.**

**Array methods**

**Push() : add the element at last**

**Unshift() add the element at beginning**

**Pop() : remove element from last**

**Shift() : remove element from beginning**

**Splice(): This method is use to add, remove and update the elements.**

**Array de structure concept.**

**Rest Operator and Spread Operator**

**Syntax Rest and Spread operator**

**…variableName**

**We can use with array**

**For Rest Operator …variablename on the left side of the de structure syntax.**

**For Spread operator …variable name on the right side of the array variable.**

**Spread operator**

1. **spread operator is use to create the clone of the array or duplicate array.**
2. **Spread operator is use to merge more than one array object.**
3. **Using spread operator we can merge two use-defined object property into one objects.**

**We can use with function**

**Creating user-defined object using ES6 style**

**ES6 style OOPs concept.**

**From ES6 onward we create to describe object we are using class keywords.**

**ES6 style**

**class className {**

**variableName = value;**

**variableName= value;**

**functionName() {**

**}**

**}**

**constructor**

**constructor is a type of special function which help to create the object.**

**Up ES5 function itself is behave like a constructor.**

**But from ES6 they provide constructor. If we want to write constructor from ES6 we have to create a function with name as constructor in lower case.**

**Constructor get called automatically when we create the objects.**

**But in ES6 we can write only one constructor means we can’t write more than one constructor.**

**Day 6**

**31-05-2021**

**DOM Operation Insert and Delete DOM Element dynamically.**

**Project Details**

**Name\_Organization\_MERN\_Stack\_Repotitory :Repository**

**Local Machine**

**Create one folder**

**Phase 1**

**Create a Phase1 Project**

**Phase 2**

**Phase 3**

**You have to push this code in git repository.**

**Web Service :**

**Req java(req)**

**Client SBI XML/JSON HSBC**

**Spring boot asp.net**

**Res python**

**Node js Non java res**

**Web Service : Giving the Service for web application when two application running using different technologies.**

**SOAP Base only XML format**

**REST API Web Service xml and non xml ie json**

**Any other format.**

**JSON : JavaScript Object Notation**

**JSON is use to store the data in the form of key value pairs. Where key must in string format and value may be number, Boolean, string, array, complex object.**

**Syntax**

**{“key”:value,”key”:value,”key”:value}**

**In View side we have understand how to convert string to json and json to string.**

**Promise : Promise is a pre-defined object provided by JavaScript which help to handle asynchronous action’s eventually may be success values or failure values.**

**User-defined promise**

**Let pr = new Promise((res,rej)=> {**

**res(“successfully done”)**

**})**

**//promise handle asynchronous event data if promise resolved then callback function get called if promise rejected catch callback function call.**

**pr.then().catch()**

**JavaScript provide pre-defined method/function fetch() which help to consume/produce REST API from backend technologies develop in any language Java(Spring boot), Asp.net, Python, or Node JS.**

**Fetch() function return type of promise objects.**

**Rest API**

**Representational State Transfer Application Programming interface.**

**Backend technologies like Java, .net, python or Node JS expose their data (services) in the form of JSON using Rest API.**

**URL if they given data in the form of JSON (XML) They are REST api develop in any technologies.**

**HTML5 : Storage API**

**sessionStorage and localStorage**

**If we want to share the data between more than one JavaScript file may be ES5 or ES6 we can take the help of Storage API.**

**store.js**

**var a=10;**

**retrieve.js**

**document.write(a)**

**If we want to share the data between two js file then we can take help of sessionStorage or localStorage**

**sessionStorage.setItem(“key”,value);**

**localStorage.setItem(“key’,value);**

**key must be string**

**sessionStorage.getItem(“key”);**

**localStorage.getItem(“key”);**

**using key we can get the value from session scope.**

**This value is available till application close. Once application close the value form session store destroy.**

**Some time if you want to remove the value from session storage we can call**

**sessionStorage.removeItem(“key”);**

**If we store the value in localStorage it store the secondary memory. After close the after still we can get the value we open the application once again.**

**localStorage.removeItem(“key”);**

**Day 7**

**01-06-2021**

**Bootstrap**

**Bootstrap is a free open source front end CSS framework for HTML and CSS for develop responsive web application.**

**Meta view port tag ensure proper rendering or view and touch zooming.**

<meta name="viewport" content="width=device-width, initial-scale=1.0"/>

Using this tag we can make our web page as responsive web application.

Bootstrap provided set of pre-defined CSS classes which store in external CSS file.

Bootstrap provided CSS classes for all DOM elements or tags

Like div, p, button, form, table, etc.

If we want to add the bootstrap features to our web page

1st way

Refer the bootstrap URL using CDN (Content Delivery Network).

2nd way

Download the bootstrap file on our machine and link to that file.

3rd download the bootstrap using node js.

**Div tags classes**

**container : container is a fixed width size.**

**container-fluid : This type of classes take the full width of the viewport.**

**Button classes**

**Grid Layout**

**Bootstrap grid system allow up to 12 column across page.**

**You can divided the container or container-fluid in row and each row in column with space multiply by 12.**

**Container**

**row**

**col**

**jQuery : It is a external library function which contains lot of pre-defined function which internally connected to each other to read, write and update DOM element very easily.**

**$(“select tag using selector”).doActionOnThatTag()**

**Phase 2**

**Day 1**

**06-06-2021**

**Node JS Overview**

**jQuery is a type of external JavaScript library**

**Angular Framework is a type of JavaScript open source**

**framework.**

**Node JS : Node JS is not a library like jQuery nor framework like Angular Framework. It is a run time environment for JavaScript program it may be library or framework.**

**Like a JRE in Java.**

**Before Node JS JavaScript is known as Client Side scripting language running on browser.**

**After node JS script is known as client side as well as server side scripting language.**

**Node JS provide lot of pre-defined module (local module or external Module) which help to do file handling programing, creating web application, creating web service, connecting database(Using JavaScript).**

**Frontend backend**

**HTML/CSS/JavaScript JEE (Java)**

**Asp.net**

**Python**

**Php**

**Node JS**

**In Node JS (JavaScript Program (Server side scripting )) we can’t use window and document objects.**

**In Node JS we can’t use BOM and DOM hierarchy.**

**Why node js program we run through command prompt.**

**TypeScript**

**TypeScript is a super set of JavaScript which support all features of ES6, ES7.**

**ES6 using JavaScript doesn’t support data types.**

**But TypeScript support data types concept.**

**Like number, string, Boolean etc.**

**TypeScript file is .ts (TypeScript). We can’t include ts file in html page.**

**Transpiler : It is also type of compiler.**

**Transpiler is a tool or command that translate between source code at the same level of abstraction.**

1. **Typescript : This transpiler help to convert ts file to js file. Which we can include in html web page.**
2. **Babel**
3. **Traceur**

**With node we get by default another command ie npm**

**npm (node package manager).Like MVN (in Maven (Java))**

**This command is use to install external module or dependencies of JavaScript or Node JS.**

**Syntax**

**npm install –g moduleName (globally)**

**or**

**npm install moduleName (locally)**

**To convert ts to js it require command as tsc**

**To enable tsc we have to install typescript module**

**npm install –g typescript**

**Angular Framework : 2 to 12**

**Typescript is required mandatory.**

**React JS :**

**We can use ES5 JavaScript**

**We can use ES6 JavaScript**

**We can use typescript**

**Babel**

**Babel is a type of transpiler which help to convert ES6 to ES5 JavaScript(browser compatibility) with Pure JavaScript program.**

**Babel with React JS :**

**JSX (JavaScript and XML) : This JSX code can’t understand by browser so we have to convert JSX code to ES5 or ES6 JavaScript code which can be understand by browser.**

**Babel configuration to convert ES6 to ES5 code.**

**In Node Js application it require package.json file.**

**This file hold the all application configuration details, build details, module version details (dependencies details) etc.**

**npm init (this command is use to create package.json file)**

**Then install two external babel module**

**Installing locally.**

**npm install babel-cli**

**npm install babel-preset-env**

**then to convert demo.js (ES5 javascript code)**

**babel demo.js –d output**

**output is folder where file will generate.**

**Using babel demo.js –d output we have to convert each time to es5 file.**

**Create src folder and keep all JavaScript(Es6) files.**

**webpack:**

**webpack is a static module bundle.**

**Webpack treat all files and assets as module.**

**Module is a collection of variable, function, classes etc.**

**Using Webpack we can create dependency graph which describe how all modules are related to each other using require (import) and ­ statement between more than one files.**

**Using web pack we will create static bundle file which traverses all modules(files) to build the graph.**

**This static bundle file we can include in view page ie html page.**

**To connect to two file we have to take the help of import / require and export concept.**

**If you want to use the webpack features we have to install module**

**npm install –g webpack**

**npm install –g webpack-cli**

**please create one file with name**

**webpack.config.js file**

**React JS**

**React JS is a JavaScript library for building User Interface or UI Components.**

**React JS is front-end JavaScript library developed by Facebook. It is use to handle the view layer or presentation logic for web as well as mobile application (React native).**

**It is one of the most popular JavaScript library and has strong foundation community behind it.**

**React JS a open source.**

**React JS is a not a framework. It is library.**

**jQuery Vs React JS**

**Angular Vs React Vs Vue**

**jQuery do all operation on actual DOM But react JS provide Virtual dom if we any change it apply on virtual dom and then apply on actual dom.**

**jQuery is light weighted library where React is heavy compare to jquery.**

**Angular is a framework But react JS is a library.**

**Library only helps you in one aspect but where as framework helps us in many aspect.**

**Library Vs Framework**

**Angular is a framework React JS is a library.**

**Library is not standard. They focus on only one area depending upon type of library. Where framework is standard. The implementation of all design pattern (best practise) is taken care by framework. Framework is very big they do lot of task.**

**If we are planning to develop the application using framework 70 to 80% task is taken care by framework. Hardly we have to write 20 to 30% code to make final product.**

**Framework is like a template or protocol but not a final product.**

**Framework are heavy but library are light weighted.**

**Angular Vs React JS**

**React JS is library and Angular is a framework.**

**React JS is a library it only focus only UI not on look and feel.**

**MVC : Model View Controller : Angular framework base upon MVC Design pattern.**

**But React JS only focus on View in MVC.**

**React JS a open source library which help to improve the UI.**

**React JS provide virtual dom features.**

**React API : Application Programming interface.**

**Two modules**

**react : It is a open source js library or module which help to develop UI. If is fully component base UI.**

**react-dom : react-dom is a open source library or module which provide glue between react and actual DOM. When we want to show react component on DOM we need to use ReactDOM.render().**

**React JS as well as Angular framework is use to create SPA means Single page application.**

**Multi page application**

**One.html welcome.html**

**Hyperlink**

**Using button**

**Submit button**

**Using JavaScript function**

**In multipage application whole page loaded once again in browser memory.**

**Using SPA we can load the only part of the page rather than whole page.**

**Component is use to control the view or part of view on web page.**

**Using Component we can create user-defined tags whenever we use that tag the code written in component get display on view page.**

****

**Day 2**

**07-06-2021**

**Online editor to do React JS program**

**Codepen**

**React provide two library react and react-dom**

**react library help us to create react component.**

**read-dom provide the bridge between react and actual dom.**

**Creating react component**

**1st way : React.createElement()**

**const element = React.createElement("p",{},"Welcome to React JS");**

**const element = React.createElement("p",{"class":"myClass"},"Welcome to React JS");**

**const root = document.querySelector("#root");**

**ReactDOM.render(element,root);**

**2nd way : using normal function or arrow function style**

**function functionName() {**

**}**

**function MyComponent() {**

**const element = React.createElement("p",{"class":"myClass"},"Welcome to React JS");**

**return element;**

**}**

**const root = document.querySelector("#root");**

**ReactDOM.render(MyComponent(),root);**

**JSX : JavaScript and XML**

**While writing function name we have to follow pascal naming rules. Means first letter of function name must be upper case.**

**3rd way : using class style ES6**

**React library provide features to create user-defined tags using function or class style. Those function or class return JSX code.**

**<b>Welcome </b>**

**<abc color=”red”>hello</abc>**

**Facebook provide create-react-app external node module which help to create the project.**

**If we create project using create-react-app it add all external dependencies which require to run the react js program like installing react, react-dom, babel, webpack still more.**

**npm install –g create-react-app**

**After installation successfully create create-react-app we have to create a project**

**create-react-app project-name**

**Ex :**

**create-react-app welcome-app**

**Here**

**create-react-app is command or module or library**

**welcome-app is a project name**

**create-react-app is a external module provided by facebook which help to create sample react js projects.**

**After created project successfully move inside a project using a command**

**cd project-name**

**cd welcome-app**

**open this project in Vs code**

**1st option code .**

**Or open any other way**

**Then run project we have to run the commands**

**(terminal or command prompt must be open inside a project folder ie where package.json)**

**npm start**

**After executed project 100% it automatically open the default browser using URL as**

[**http://localhost:3000**](http://localhost:3000)

**Project structure**

**node\_module : This folder contains all required node js external module or dependencies which help to run the project.**

**public : This folder contains first page of the react js application.**

**index.html**

**if you want any static content depending upon project you can write but not advisable.**

**In this page you can find div tag with id attribute with value as root.**

<div id="root"></div>

**In div tag component get render (send or display) using ReactDOM.render() function.**

**In public folder you can keep all static resource resources for the project like images.**

**package.json : This file contains all react js configuration details.**

**Next expand the src folder.**

**Then open the App.js file**

**If we export using default then in another page we can import without curly braces that component or variable or class.**

**In one file we can export default only one component or variable or functions.**

**In one file we can export default only one and more than normal export.**

**React Component**

**A component is a small, reusable chunk of code that is responsible for one job. That job is often to render some HTML code.**

**In React JS we can create the component using Normal function style (ES5 style), using arrow function style (ES6 style) as well as class style (ES6 style).**

**Every component return React.createElement may one tag or more than one tag with help of JSX.**

**The component allow you to split your UI or application into independent, reusable piece of code.**

**Header component**

**Footer component**

**JSX must be return only one tag.**

**If we are planning to return more than one tag it must be wrap in another like div tag.**

**<div>**

**Set of tags**

**</div>**

**<>**

**Set of tags**

**</>**

**JSX : JavaScript and XML : Using JSX features we can write html code inside a JavaScript function.**

**That html code can contains static data as well as dynamic data.**

**JSX expression**

**{5+2}**

**{10/2}**

**{variableName}**

**{objectValue}**

**In React JS we will use more ternary operator to do some condition.**

**Ternary operator**

**condition ? true : false**

**According to JSX every dom element when retrieve using map we have to provide unique id ie known as key.**

**Using key JSX maintain the unique ness between two tags if two tags contains same information or different information.**

**Creating component using class style**

**When creating class component first we have to import it**

import React from "react"

**Then we have write simple code**

class MyClass extends React.Component{

    render(){

      return <div>This is Class component Example</div>

    }

}

**Phase 2 : Day 3**

**08-09-2021**

**create-react-app create-state-and-props**

**React Component contains two type of variable**

**props (properties)**

**state**

**If component created using ES6 class style we can use props and state variables.**

**If component created using ES5 function style or ES6 arrow style we can use only props no state. From new version of React JS using react hook methods we can use state in function or arrow function components.**

**props :**

**Component’s props (obviously short for properties) in React.**

**Props are similar to argument for function component(normal function or arrow function) or constructor in ES6 class style.**

**Props are ready-only to component (immutable) can’t change.**

**A component’s props is an object. It hold the information about that components.**

**Props can hold number, string, Boolean, object, json or function etc.**

**If you want to access props in class component we have to use this.props but in function component we can use directly props no this.**

**<SayHello user=”Ravi”></SayHello>**

**<font color=”red”> Hi </font>**

**<font color=”green”> Hello </font>**

**<font color=”blue”> How r you </font>**

**Using props we are creating user-defined attribute for components.**

**Props is use to receive the value for attribute of that component.**

**Using props we can pass the value from parent component to child component.**

**It advisable inside one file we have to create only one components.**

**State : Unlike props, state is a private variable and strictly belong to single components. Component state we can modified or change over time in response to user action.**

**State is a heart of react component which help describe the component behaviour**

**These states are use to store the information or data about that component.**

**If we created component using class syntax we can create or initialize the state variable inside a constructor using syntax as**

**After super keyword.**

**this.state = {key:value}**

**value may be number, string, Boolean, array type or object type.**

**We can change the state using**

**setState({key:value}) using some events.**

**React form and events with setState**

**create-react-app react-form-and-events**

**React Events :**

**Handling event with react js is very similar to handling events on DOM element using JS.**

**React event type are named using camelCase like**

**In JavaScript**

**onclick**

**onClick**

**But in React JS**

**onClick**

**onDblClick**

**onMouseOver**

**onChange**

**onSubmit**

**etc**

**HTML with DOM**

**<input type=”button” value=”click” onclick=”fun()”/>**

**in React JS we have to use syntax as**

**<input type=”button” value=”click” onClick={this.fun}/>**

**In normal function we can’t access this ie current object or component instance.**

**But in arrow ES6 this ie current object or component instance bound automatically to arrow functions.**

/without bind we can't access component object.

**So if you want to access this ie current objet or component object in normal function we have to bind that function in constructor.**

**In App component create id, name and salary, gender, hobbies, as state variable then assign the value through forms and display those details in Child component DisplayEmployee. Hint pass the value using props to DisplayEmployee components.**

**Display in property format.**

**App must be class component and DisplayEmployee must be functional components.**

**Functional component takes props as parameter and return JSX.**

**Component communication : sharing the data.**

**Life cycle of component**

**How to call REST API using fetch and axios**

**Get, post, put and delete.**

**Phase 2 : Day 4**

**09-06-2021**

**Calling Rest API using React JS**

**In Plain JavaScript using fetch() function we retrieve the REST API develop any in language.**

**Syntax**

**fetch(“url”).then(res=>res.json()).then(data=>console.log(data)).catch(e=>console.log(e))**

**fetch is a core API so if we want to retrieve rest api using fetch we have deal with two promise(then).**

**But React JS use third party module or plugin ie axios. Using axios we can retrieve data from REST API. Axios internally fetch, promise and ajax.**

**axios provide some extra features.**

**axios we get(), post(), put() and delete() resources from backend technologies.**

**According to REST API**

**get() : get resources employee, task, customer, order etc.**

**post() : create resources insert records in employee, task, customer, order etc.**

**put(): update the resources, in employee resource update salary using id, in customer resource update address, phone number, age using cid.**

**delete() : delete the resource, delete employee using id, delete customer using cid etc.**

**We will learn using Resource using REST API in phase 3**

**product.json file**

**store 1 or 2 product details**

**using node js with help of json-server module we can make product.json file behave like a server (REST API).**

**First we have to product.json file with few data.**

**Then install external module as**

**npm install –g json-server**

**open the terminal in location where product.json file is present**

**Then run the command as**

**json-server product.json**

[**http://localhost:3000/product**](http://localhost:3000/product)

**using this URL we can call get(), post(), put() and delete() the data from product.json file.**

**We have to install third party module ie axios in current project using command as**

**npm install axios**

**axios all method return type is promise.**

**axios.post(“url”,dataInJsonFormat).then(result=>console.log(result)).catch(e=>console.log(e));**

**Component communication**

**Sharing the data between two components (state variables).**

**Parent -🡪 Child -🡪 Using props**

**Child -🡪 Parent -🡪 using props with callback function**

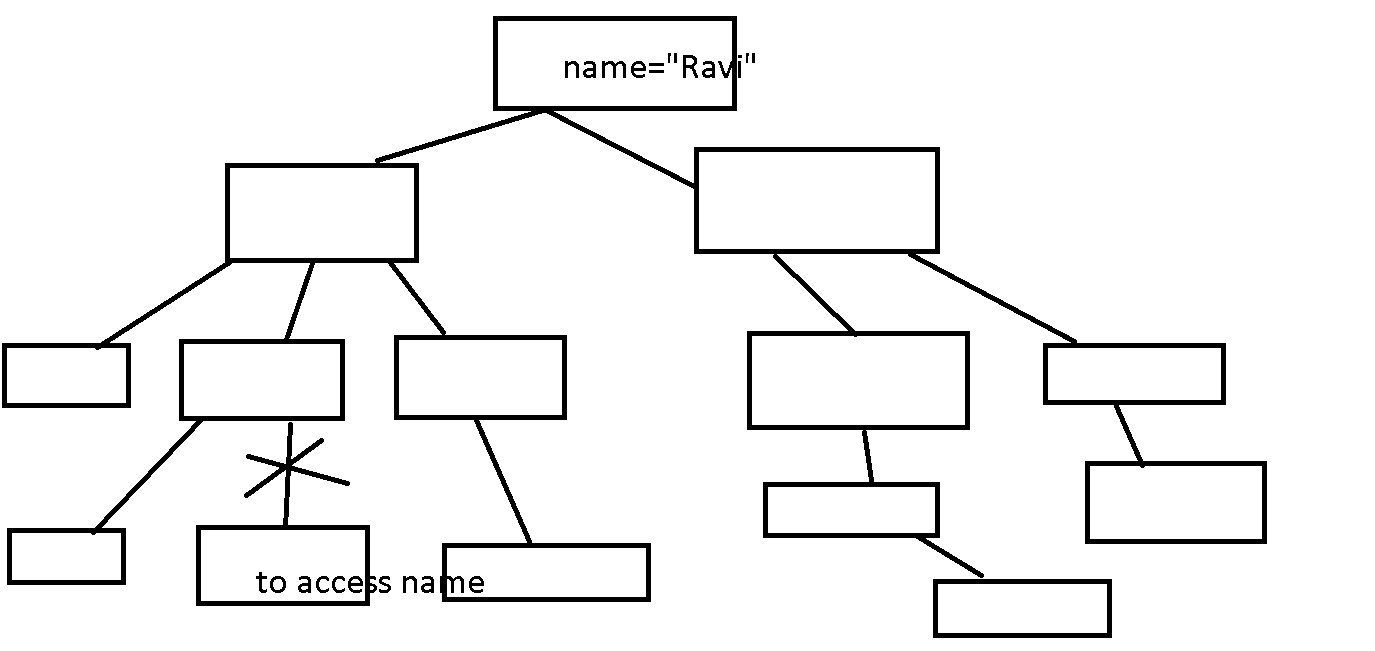
**Hook and Redux**

**Phase 2 : Day 5**

**09-06-2021**

**React Context API**

**React Context API provides the programmer a way to pass the data (state) from one component to another component without taking the help of props. In Context API it doesn’t matter that is relationship between two component.**

****

**Using React Context API helps use to pass to value from one component another simplest way.**

**Context is designed to share data that be consider global for a tree on React Component. Current authentication, theme etc.**

**React JS Context API provide the concept called Provider and Consumer**

**Provider is use to share the value of any types. And consumer is use to receive the value from provider. Like a broadcasting. One to many relationship.**

**First we will configure the context and make the reference of Provider and Consumer**

**React JS with Redux**

**Redux is a state management tool or library for JavaScript as well as React JS.**

**Redux manage the State of any type number, string, Boolean, array type or object type.**

**In React We will create the project using component. Every component hold set of state of different type. Those state are local to that component.**

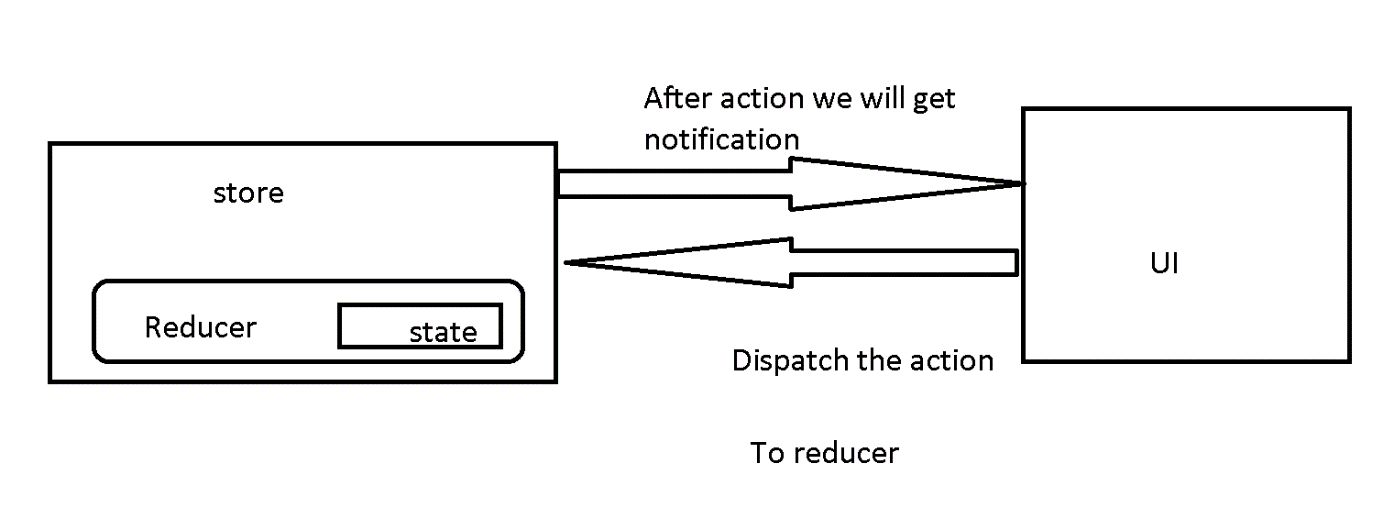
**But if you want to make global for all component in enterprise application where we need 100 or 1000, 10000 components.**

**React project without redux to do change on employee salary**

**Id,name, salary**

**Increment and decrement.**

**create-react-app employee-inc-dec-without-redux**

****

**With redux, the Global state of our application is kept in store and each component can access those global state and do change with help of reducer with specific actions.**

**Store : According to Redux every react application contains one store. A store is a container that hold the application global state.**

**Redux can have only one single store in whole application.**

**Store is like data database.**

**Store is a container for state (global state).**

**Reducer : If store is as database reducer is table for that database.**

**Reducer is a plain JavaScript function. Which takes two parameter first one is initial state and second one is action which help to do some changes on that state which is part of store.**

**State : State is a single immutable. If reducer is like a table then state are fields.**

**Dispatch : dispatch is use to pass the action to reducer to do the change on state. Dispatch is like a query in database.**

**Store -🡪 Database**

**Reducer 🡪 table**

**State -🡪 fields**

**Action (dispatch)🡪 Query on table (create, delete, update, retrieve or more etc).**

**First Step : Create new project**

**Install two external module ie**

**npm install redux : this module help to create store**

**npm install react-redux : This module help to integrate react application to store.**

**First we will create the reducer**

// state may be number, string, boolean, array type or object.

const initialState = {

    id:100,

    name:"Ravi",

    salary:12000

}

export default function reducer(state = initialState,action){

    return state;

}

**Now open the index.js file**

**Create the store**

First import the store from redux

import {createStore} from 'redux';

**Then import reducer file which require to connect reducer to store.**

import reducer from './reducer';

Then connect reducer with store

const myStore = createStore(reducer);

**After created store now we have to take the help of Pre-defined component ie Provider part react-redux which help to access store in all react components.**

import {Provider} from 'react-redux';

**Now link store with provider with React Root component we have to write the code as**

ReactDOM.render(

  <Provider store={myStore}>

  <React.StrictMode>

    <App />

  </React.StrictMode>

  </Provider>,

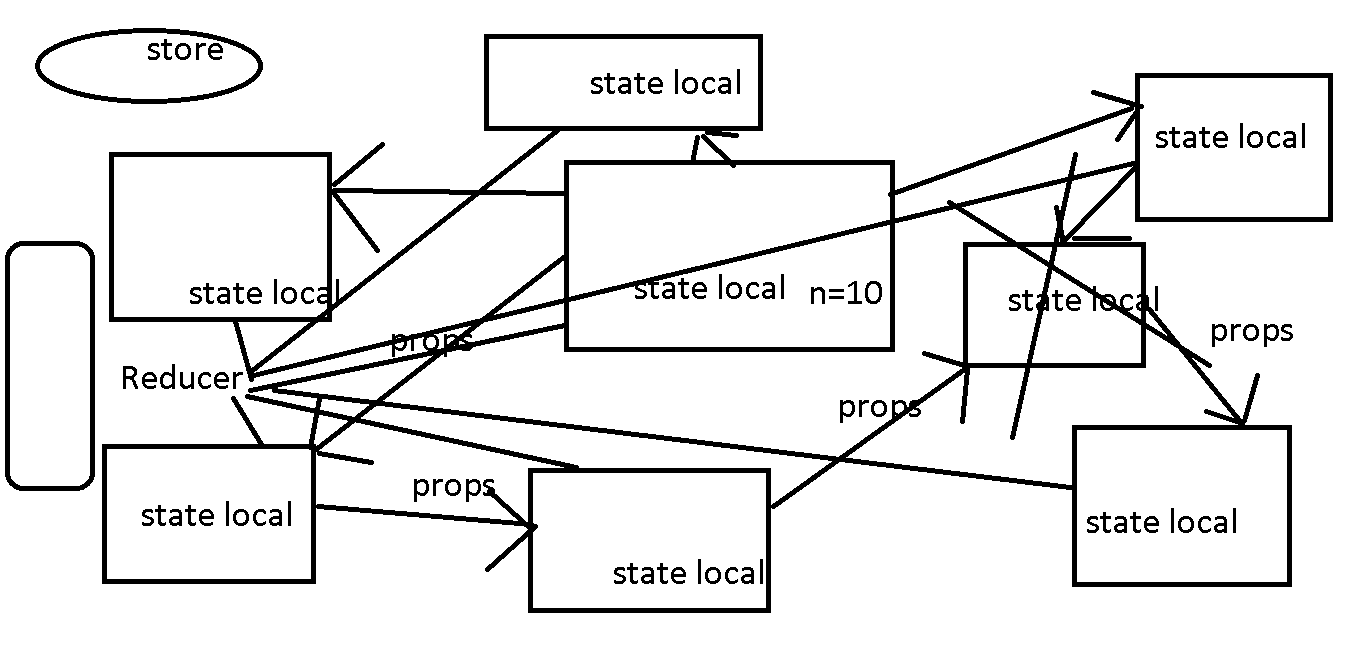
  document.getElementById('root')

);

**According to redux dispatch is use to pass the action and payload(data) to reducer to change the state base upon the action.**

**dispatch({type:”actionname”,payload:”data”})**

**action is use to do the change in store and payload data which going to pass to reducer.**

****

**Phase 3 :**

**Day 1**

**15-06-2021**

**Node JS :**

**REPL tool**

**Node JS program**

**Node JS Module**

**Type of modules**

**Node JS asynchronous operation**

**Fs module**

**Util**

**http module**

**Multiprocessing concept.**

**Express module**

**Routing**

**Middleware**

**React with Express module**

**CRUD Operation**

**Socket programming**

**Mongo DB database.**

**Different types of queries**

**Node JS with mongo database**

**Mongodb**

**Mongoose db**

**React with Node with Express with mongoose with MongoDB**

**Node JS :**

**Node js goal is to provide an easy way to build scalable network application.**

**Because node js code written using JavaScript and JavaScript has already features as**

**Callback and asynchronous communication.**

**Anonymous function ES5 style**

**Arrow function ES6 style**

**JavaScript provide non block data with help of node js.**

**Node JS is a open source, cross platform runtime environment for Server side and networking application.**

**Before Node JS JavaScript is used for only client side scripting language. But after node js server we can use JavaScript for Client side as well as Server side scripting language.**

**Node JS provide even driven architecture features.**

**Node JS program are executed by V8 engine. Which use in google chrome.**

**jQuery**

**coffee js**

**Ext JS**

**Angular**

**React JS**

**Node JS**

**Node JS not a library not a framework.**

**Node JS is not good for beginner.**

**Node JS is not multi threading. It is a single threaded.**

**In Simple**

**Node Js = Runtime Environment + JavaScript library or framework.**

**Using Node js we can file handling program, creating web application, connecting database, creating rest full API etc.**

**It is command line tool, you can download external modules (using npm), compile and run the application.**

**Tsc**

**Node js provide http module which help to create client and server application using JavaScript.**

**Node js provide fs module which help to do file handling program like read, write and append the file**

**Node JS is a platform for writing JavaScript application which we can run outside web browser.**

**In Node JS we can’t access window and document objects.**

**So in node js we can’t access dom and bom hierarchy.**

**Node JS can’t run on GUI it can on terminal.**

**REPL : Read Eval Print loop**

**Node JS Provide two pre-defined global object**

**console**

**process**

**we can use these two object without import or require.**

**process**

**Node JS Modules**

**Module in Node js is a simple or complex functionality organized in single or multiple JavaScript file which can be reuse through the node js application.**

**These are module mainly use for specific purpose.**

**Types module**

**Core module**

**Third party module or external module**

**Few core modules are**

**fs**

**os**

**path**

**url**

**http**

**etc**

**how to load these modules**

**//load the modules.**

**let/var refereceName = require(“moduleName”);**

**refenceName.property**

**reference.function()**

**If we are planning to use external module provide by different vendor. We have to install those modules using command as**

**npm i –g moduleName : install those module globally**

**npm i moduleName : install locally**

**Node JS asynchronous operation**

**JavaScript is a asynchronous in nature or by default. So Node JS also asynchronous.**

**Asynchronous operation does exactly opposite of synchronous. Asynchronous code executes without having any dependencies and no order.**

**f1()**

**f2()**

**f3()**

**to alternatives to load or handle asynchronous operation**

**we can use promise with then chain and async and await**

**if we write async keyword for function it return promise objects.**

**Await keyword work inside a function if function with async keyword. Using await keyword we can handle the promise simplest way.**

**Fetch() : it is pre-defined function part of JavaScript which help to call REST API.**

**If we want to use fetch concept in node js we have to loadload external module using npm command**

**npm install node-fetch**

**node js provide fs (file system) module**

**it is a type of core module which help to do asynchronous and synchronous file IO operation.**

**Taking the value through keyboards using node js**

**readline**

**npm install readline**

**This module help us to read value through keyboards.**

**readline-sync**

**npm install readline-sync**

**util module**

**node js contains util core module.**

**Which help print formatting output on console.**

**As well as this module use to do debug the pre-defined or user-defined API.**

**URL : Uniform Resource Locator.**

**This module help use to get the URL details like protocol, port number, query string details.**

**This is also type of core modules.**

**http module**

**http module is a pre-defined module which help to create own server using JavaScript with the help of node js.**

**Tomcat**

**Weblogic**

**Jboss**

**IIS Server**

**Nxjs**

**Non Node JS server are thread base.**

**But Node JS server is single thread but use event driven architecture.**

**Client Server**

**Tomcat**

**Java Application**

**Concurrently response 100 client at time**

**100 thread.**

**101 client send req**

**Req block**

**http : using this module we can create own server using javascript.**

**http with routing : base upon the path display different message.**

**Path : signUp**

**Login creation Page : Action =”acountCreate” store in file in json format**

**Path : signIn**

**Login page : check username and password from file it correct display successfully login or failure.**

[**http://localhost:9090/signUp**](http://localhost:9090/signUp)

**create account**

**action = createAccount**

**username**

**password**

**submit**

**store data in file using fs module**

[**http://localhost:9090/signIn**](http://localhost:9090/signIn)

**login page**

**action = checkAccount**

**username**

**password**

**submit**

**check username and password from file using fs module**

**Phase 3 : Day 3**

**17-6-2021**

**Framework for Node JS**

**There are various third party open – source framework available in node JS which help to make web application using Node js faster and easy.**

**Express JS**

**Koa**

**Hapi.js**

**Geddy**

**Etc**

**Express JS framework :**

**Express JS is a third open source web application framework. It provides various features that make web application development very fast and easy. Express JS internally use or wrap http node js core module.**

**Using Express JS we can Create REST API very easily.**

**Express JS support all http protocol methods like get, post, put and delete.**

**We have to install this module using command as**

**npm install express**

**package.json file using npm command.**

**npm init –y**

**package.json file hold your project configuration details.**

**Npm install express**

**Npm install readline**

**Npm install mongodb**

**Node JS provide pre-defined global property \_\_dirname**

**This property help us to find current directory for project.**

**If you want to read the data from request body in express node JS we have to install one module**

**body-parser**

**using**

**npm install body-parser**

**This module we have to add as a middleware module in express js to unable data from request body part.**

**If we create express module with html code. Html pages(View Pages) or presentation logic are tightly coupled with express js module. In html page we can’t display dynamic value properly.**

**Web Service :**

**Giving the service for web application when both application running using different technologies.**

**Java(req)**

**Client HSBC XML/JSON SBI**

**Java JavaScript(res) Node JS**

**2 types**

1. **SOAP base web service : we can consume and product the data only in the form of xml.**
2. **REST Full Web Service : using REST full web service we can consume and product the data in any format base upon the client requirement like xml, json, text, media type etc.**

**View (HTML):**

**React JS**

**Using axios module we call get, post, put and delete.**

**Json-server is replacing by express module.**

**Using Express module I planning to expose my data as we Rest full web service with help get, post, put and delete methods.**

**Get() : This method is use to produce the data in different formats.**

**Select query**

**Using Get method we can pass the value to REST API using 2 ways**

1. **Query param : HTML forms internally uses query params concept.**

**URL?key=value&key=value&key=value;**

1. **Path param**

**url/value1/value2/value3**

**We can call Get methods through URL, as well as normal form submit button.**

**Post() : Create the resource or store the data in file or database.**

**Insert query**

**Post method we can’t call URL.**

**We can call post method through HTML form submit button with method is post.**

**Test REST API we can use third partly plugin provided by different browsers.**

**var product =[{“pid”:1,”pname”:”Tv”,”price”:145000}]**

**REST API**

**Retrieve app product details. Get**

**Retrieve specific product price using pid : Get**

**Store product details : post**

**Delete update using Id : delete**

**Update Product price using id : put method**

**Phase 3 : Day 4**

**21-6-2021**

**Backend product**

**Frontend product**

**Backend**

**First create the package.json file in backend**

**Install two dependencies.**

**npm intall express**

**npm install body-parser**

**After created backed using express module**

**Now create frontend using react js**

**Then install axios module.**

**npm install axios**

**we are running two application or domain**

**frontend – react js : 3000**

**node express – 9090**

**when two domain or application going to communicate to each other we will get the restriction.**

**CORS: Cross Origin Resource Sharing**

**It is browser mechanism which enables control to access resources or files or data located or present outside a other application or domain.**

**According to node js with express module we have to install cors external module and we have to use as middleware in express module to access resource.**

**Phase 4 : Day 4**

**22-6-2021**

**Database**

**RDBMS : MySQL /Oracle /Sql Server**

**Mongo DB**

**Input : initialization, read keyboards, network, file, database etc**

**Program Process**

**Output : display on console, on browser, send through network, store in file and database etc.**

**Persistence data.**

1. **File system using node fs module we can store the value in external file in text or json format.**

**Fs module allow to store the data in string format.**

**If we want to fire or execute any query like**

**Retrieve all records with some conditions like price > amount product start with some character, product id between range. Etc.**

**In File system to achieve CRUD Operation is not possible.**

**Limitation of file system**

1. **File system is not consistence format file, type of files**
2. **Data redundancy (we can store duplicate records in file system). We can avoid but using programming language.**
3. **CRUD Operation in file system is very complex.**
4. **Database system**

**Data : raw fact.**

**Information : processed data or meaningful data.**

**DBMS : Database management system : It is software or tool which help to store the data in table format like row and columns.**

**RDBMS : Relational Database Management system.**

**EF Codd’s Rules 12**

**Trainer\_Students**

**TId TName Tech Sid SName Age**

**1 Raj Java 100 Seeta 21**

**1 Raj Java 101 Reeta 22**

**1 Raj Java 102 Veeta 23**

**Trainer**

**PK**

**TID TName Tech**

**1 Raj Java**

**Students**

**PK FK**

**SId SName Age TS\_ID**

**100 Seeta 21 1**

**101 Reeta 22 1**

**102 Veeta 23 1**

**Limitation of RDBMS**

**In RDMBS database before the records we have to create the schema. Table name, number of columns, type of value store in each columns.**

**If we want to retrieve records from schema we have to use SQL language.**

**DDL, DML, TCL, DCL and DQL**

**Web Application**

**Data :**

**Structure format : table format.**

**Mini structure format : json, xml format, etc.**

**Non structure format :**

**No SQL**

**Key-value pairs to store : Redis, riak etc**

**Graph database : Neo4J**

**Document oriented : Mongo DB etc**

**Column family : Hbase, Cassandra**

**MongoDB :**

**MongoDB is a open source, Document based, high performance, high availability, scaling etc.**

**MongoDB doesn’t store data tables.**

**MongoDB use concept as Collection like a table.**

**These collection are comprised of JSON document instead of rows.**

**RDBMS Mongo DB**

**(Table)**

**Table Collection**

**Row or records document**

**Sql format Json format.**

**Convert json into sql query format.**

**insert into emp values(1,’Ravi’,12000);**

**Mongo DB store the data in document in the form of json format.**

**Mongo DB provide pre-defined function which help to interact with collection to do some operation on that data. Insert, delete, update and retrieve without any query language.**

**mongod : To run the server we have to use the command as mongod**

**To open mongo database terminal we have to run the command as**

**mongno**

**commands**

**This command is use to display all the databases.**

**show dbs**

**show databases**

**database is a group of collections(tables).**

**use database : if database is available it will create and switch to that database if already exit it switch to that database.**

**If you want to check the collection present in current database.**

**Show collections**

**Show tables**

**Command to create the collection**

**db.createCollection(“Sample”)**

**RDBMS**

**Employee -- Collection**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **Name** | **Salary** | **Age** | **City** |
| **100** | **Ravi** | **12000** | **Null** | **Null** |
| **101** | **Ramesh** | **14000** | **21** | **Null** |
| **102** | **Ravi** | **Null** | **24** | **Bangalore** |
|  |  |  |  |  |

**Insert the document in collection**

**db.Sample.insert({name:”ravi”,age:21})**

**To view documents from collection we have to use the command as**

**db.Sample.find();**

**We can create the collection without createCollection functions.**

**db.Emp.insert({name:"Ravi",age:22});**

**If the collection Emp is present then new document added to existing collection or else it will create new collections.**

**In Normal RDBMS primary is not mandatory.**

**In Mongo DB database maintain the uniqueness between two document using pre-define property as \_id. \_id is property like a primary in RDBMS.**

**We can’t change the \_id property name.**

**Employee**

**\_id, name, age, address :{city , state}**

**5 documents.**

**db.Employee.find().pretty();**

**To view particular documents.**

**db.Employee.find()[indexPosition];**

**db.Employee.find()[1];**

**To view particular index position specific attribute value.**

**db.Employee.find()[1].name;**

**To view particular specific fields of all documents.**

**db.Employee.find({},{fieldname:1})**

**It display name and pre-defined property \_id**

**db.Employee.find({},{name:1});**

**To display only name fields**

**db.Employee.find({},{name:1,\_id:0});**

**To display two user-defined fields**

**db.Employee.find({},{name:1,\_id:0,age:1});**

**To display particular position two fields details**

**db.Employee.find({},{\_id:0,name:1,age:1})[2];**

**To display first document in collection**

**db.Employee.findOne();**

**If we want to retrieve top most n number of documents.**

**db.Employee.find().limit(2);**

**If we want to skip n number of documents**

**db.Employee.find().skip(2)**

**db.Employee.find().skip(2).limit(2);**

**if we want to display the document with conditions.**

**db.collectonName.find({condition})**

**db.Employee.find({\_id:101});**

**db.Employee.find({name:'Nandu'});**

**db.Employee.find({age:28});**

**db.Employee.find({"address.city":"Bangalore"});**

**conditional operator**

**> $gt**

**>= &gte**

**< &lt**

**<= &lte**

**= &eq**

**db.Employee.find({age:{$gt:30}});**

**db.Employee.find({age:{$gte:30}});**

**db.Employee.find({age:{$lt:30}});**

**db.Employee.find({age:{$lte:30}});**

**db.Employee.find({age:{$eq:30}});**

**db.Employee.find({age:{$ne:30}});**

**and / or**

**Display those document details when both condition satisfies or any one**

**db.Employee.find({$and:[{\_id:100},{name:"Ajay"}]});**

**db.Employee.find({$or:[{\_id:101},{name:"Ajay"}]});**

**db.Employee.find({$and:[{name:"Vijay"},{age:{$gt:40}}]});**

**conditions with specific fields**

**Display employee name whose age is > 30**

**db.Employee.find({age:{$gt:30}},{\_id:0,name:1});**

**sorting by some specific fields.**

**Backend(Express with Array) and frontend (React JS )**

**express with fs module**

**Phase 3 : Day 5**

**23-6-2021**

**Update documents**

**db.Emp.update({condition},{updatethevalue});**

**without condition it insert the records.**

**Insert new records**

**db.Emp.update({},{name:"Raj"});**

**name field replace by age fields.**

**db.Emp.update({name:’Ravi’},{age:55});**

**update the records using conditions**

**db.Employee.update({name:'Ajay'},{$set:{age:35}});**

**update the two fields records using one conditions.**

**db.Employee.update({age:24},{$set:{name:'Lokesh Kumar',"address.city":"Bangalore"}});**

**update only record if conditions satisfies doesn’t update other records**

**db.Emp.update({name:"Ravi"},{$set:{age:35}});**

**if we wan to update multiple records then**

**db.Emp.update({name:"Ravi"},{$set:{age:40}},{multi:true});**

**db.Emp.updateMany({name:"Ravi"},{$set:{age:55}});**

**Remove documents**

**All document removed**

**db.Emp.remove({})**

**remove document using conditions**

**db.Emp.remove({age:25})**

**if we want to remove only one document if more than o one condition also then run the command as**

**db.Emp.remove({age:25},1)**

**Remove fields from existing documents.**

**db.Emp.update({name:"Vijay"},{$unset:{salary:1}});**

**Remove first documents where age fields is present in document remove from that documents.**

**db.Emp.update({},{$unset:{age:1}});**

**remove all document specific fields**

**db.Emp.updateMany({},{$unset:{name:1}});**

**more than one collection**

**Collection relationship**

**One to one relationship : Person – passport**

**1 1**

**Person – address**

**1 1**

**One to many relationship**

**Departments more than one employees are working**

**Department (1) Employee (many)**

**Trainer (1) Students(many)**

**Using mongo DB we can achieve this relations using 2 ways**

**Entity : employee, address, login, customer, account etc**

**Employee –id,name,age,**

**Address – city,state,pincode,**

**Login –username,password**

**Customer –cid,cnamem,age, etc**

**Account –accno,typeaccount,amount,etc**

**RDBMS doesn’t support embedded table concepts.**

**Employee ->table**

**Id,name,age**

**Address 🡪Table**

**City,state,pincode**

**Employee**

**Id,name,age,address**

**city,state,pincode**

1. **Embedding collections (using only one collection**

**Person one entity and address another entity**

**Person**

**pid**

**pname**

**age**

**address**

**city**

**state**

**pin code**

**db.person.insert({\_id:100,pname:”Raj”,age:21,**

**address:{city,”Bangalore”,state:”Kar”}})**

**one to one**

**db.Person.insert({\_id:100,pname:"Raj",age:21,address:{city:"Bangalore",state:"Kar"}})**

**its own property**

**db.Person.insert({\_id:101,pname:"Ram",age:22,city:"Mumbai",state:"Mh"})**

**db.Person.insert({\_id:102,pname:"Rajesh",age:23,address:[{city:"Bangalore",state:"Kar"},{city:"Mumbai",state:"Mh"}]})**

**Trainer -- Student -- one to many relationship**

**{\_id:1,sname:"Seeta",age:21}**

**{\_id:2,sname:"Reeta",age:22}**

**{\_id:3,sname:"Veeta",age:23}**

**db.Trainer.insert({\_id:100,tname:"Ravi",tech:"Java",**

**students:[**

**{\_id:1,sname:"Seeta",age:21},**

**{\_id:2,sname:"Reeta",age:22},**

**{\_id:3,sname:"Veeta",age:23}]})**

**Remove whole collection**

**db.Trainer.drop();**

**it remove collection as well as all documents from that collections.**

**db.Trainer.remove({}): all document removed from that collection but empty collection present in database.**

**Linking collections ( more than one collection)**

**Students**

**Sid Name Age**

**100 Reeta 21 java**

**101 Meeta 22 java**

**102 Keeta 23 python**

**103 Veeta 24 Angular**

**Trainers**

**TId Name Tech TSId**

**1 Raj Java 100,101**

**2 Ravi Python 102**

**3 Ramesh C**

**db.Trainer.insertMany([**

**{\_id:1,tname:"Raj",tech:"Java"},**

**{\_id:2,tname:"Ravi",tech:"Python"},**

**{\_id:3,tname:"Ramesh",tech:"C"},**

**])**

**using this concept we can retrieve student and trainer records from Student collection only.**

**db.Student.insertMany([**

**{\_id:100,sname:"Reeta",age:21,TSId:db.Trainer.find()[0]},**

**{\_id:101,sname:"Meeta",age:22,TSId:db.Trainer.find()[0]},**

**{\_id:102,sname:"Keeta",age:23,TSId:db.Trainer.find()[1]},**

**{\_id:103,sname:"Veeta",age:24}**

**])**

**using this concept if we want to retrieve trainer and student records we have to use both collection**

**with aggegate concept.**

**db.StudentDet**

**ails.insertMany([**

**{\_id:100,sname:"Reeta",age:21,TSId:db.Trainer.find()[0].\_id},**

**{\_id:101,sname:"Meeta",age:22,TSId:db.Trainer.find()[0].\_id},**

**{\_id:102,sname:"Keeta",age:23,TSId:db.Trainer.find()[1].\_id},**

**{\_id:103,sname:"Veeta",age:24}**

**])**

**Aggregate function**

**MongoDB process the data and return single result using aggregate functions.**

**db.Employee.find();**

**db.EmployeeDetails.insertMany([**

**{\_id:1,name:"Raj",salary:45000,deptId:10,city:"Bangalore",mgrId:null},**

**{\_id:2,name:"Ravi",salary:30000,deptId:20,city:"Bangalore",mgrId:1},**

**{\_id:3,name:"Ramesh",salary:25000,deptId:20,city:"Bangalore",mgrId:1},**

**{\_id:4,name:"Ajay",salary:22000,deptId:20,city:"Mumbia",mgrId:2},**

**{\_id:5,name:"Vijay",salary:20000,deptId:30,city:"Mumbia",mgrId:2},**

**{\_id:6,name:"Balaji",salary:18000,deptId:30,city:"Mumbia",mgrId:3},**

**{\_id:7,name:"Dinesh",salary:15000,deptId:40,city:"Pune",mgrId:3},**

**{\_id:8,name:"Chinu",salary:12000,deptId:40,city:"Pune",mgrId:4}**

**]);**

**Group by function**

**db.EmployeeDetails.aggregate([{$group:{\_id:"$city"}}]);**

**db.EmployeeDetails.aggregate([{$group:{\_id:"$deptId"}}]);**

**db.EmployeeDetails.aggregate([{$group:{\_id:"$deptId",maxSalary:{$max:"$salary"}}}]);**

**db.EmployeeDetails.aggregate([{$group:{\_id:"$deptId",totalSalary:{$sum:"$salary"}}}]);**

**db.EmployeeDetails.aggregate([{$group:{\_id:"$deptId",minSalary:{$min:"$salary"}}}]);**

**db.EmployeeDetails.aggregate([{$group:{\_id:"$deptId",avgSalary:{$avg:"$salary"}}}]);**

**db.EmployeeDetails.aggregate([{$group:{\_id:"$deptId",numberOfRec:{$sum:1 }}}]);**

**select sum(salary) from employeeDetails group by deptid**

**retrieve documents from Trainer and StudentDetails**

**db.Trainer.aggregate(**

**[**

**{$lookup:**

**{**

**from:"StudentDetails",**

**localField:"\_id",**

**foreignField:"TSId",**

**as:"TrainerStudent"**

**}**

**}**

**]**

**);**

**Phase 3 : Day 6**

**24-6-2021**

**Connecting Node JS program to Mongo DB database.**

**Using mongodb module**

**mongodb pre-defined native external module provide by node js which help to connect node js program to Mongo DB database to do CRUD Operation on collections.**

**First create the folder**

**Create the package.json**

**npm init –y**

**install mongodb module**

**npm install mongodb**

**mongoose : It is a open source third party modules which use to connect the MongoDb database.**

**Mongoose use ODM (Object data modelling) library for mongodb database.**

**Using Mongoose library we can write the rules for document what is name of the field, field contains what type of values as well as optional fields.**

**Customer : cid,cname,age**

**Java -🡪JDBC mongodb**

**ORM (Hibernate or JPA) mongoose (ORM)**

**Mongoose internally use mongodb module and extra features.**

**Mongoose module**

**Create the separate project**

**Create the package.json file using command as**

**npm init –y**

**then install the external module as**

**npm install mongoose**

**Schema is use to create the rules for the documents.**

**Express JS with Mongoose**

**Create the folder**

**npm init –y package.json**

**npm install express**

**npm install body-parser**

**npm install mongoose**

**Product CRUD Operation**

**React JS : Insert, Delete, Update and Retrieve**

**Express with Mongoose with MongoDB Database.**

**(Array replace by Mongoose modules and store in database).**

**MERN Stack :**

**MongoDB Express JS React and Node JS**

**React JS 🡪Express -🡪Mongodb/Mongoose -🡪MongoDB Database**

**Phase 3 : Day 8**

**28-6-2021**

**Express with mongoose module**

**MVC : Model View Controller**

**View -🡪React JS**

**From client request pass from post(), put(), delete() or get() method**

**Server.js / app.js (create the reference of express and database connection run on port number)**

**From here base upon path we will redirect to router**

**Controller -🡪**

**router : we will verify sub path and methods(get, post, put delete).**

**Router will redirect to controller to the task.**

**Controller**

**Controller take the help of model and do the operation base upon the router and return result.**

**Model layer -🡪 model which define the schema**

**First create the folder Express MVC**

**Create package.json file using command as**

**npm init –y**

**npm install express**

**npm install body-parser**

**npm install mongoose**

**or**

**npm install express body-parser mongoose**

**View (Browser)-------🡪URL**

**app.js ( find the main path)**

**Phase 3**

**29-06-2021**

**Socket programming concept.**

**Client and Server architecture**

**Using http or https protocol client send the request to server and server receive the request from send response back to client.**

**Using http protocol we can access the web pages. Using REST API we share the data between client and server using json or xml format.**

**TCP : Transfer Control Protocol as well as UDP (User Data protocol).**

**It allow you to make TCP or UDP connection so some endpoint and then we can send as well as receive the data from one application to another application in the form data packet.**

**To achieve TCP concept in node js. NodeJS provide pre-defined module as net module.**

**npm install readline –g**

**Socket : it is a endpoint which help make the connection between two device or application.**

**WebSocket : Socket programming on web application using http protocol.**

**WebSocket persistence connection between client and server. Web socket use full duplex and bidirection communication channel between client to server.**

**In net module client as well as server must be node program or console program.**

**In WebSocket client is browser and server is node program.**

**If we want to achieve web socket we have to install the external module**

**express**

**express-ws (websocket)**

**first create empty folder**

**then create package.json file using command as**

**npm init –y**

**npm install express**

**npm install express-ws**

**socket.io : this is third party library base upon the websocket. Which help to do socket programming simplest way.**

**Create package.json file**

**install socket.io third party library**

**npm install socket.io**

**Phase 4 : testing and deployment**

**Testing Jasmine and Karma : Front end as well as backend testing**

**Mocha and Chai ---🡪 Testing for Express**

**JEST --🡪 React JS Testing**

**GraphQL : self learning**

**Grunt**

**Docker**

**Jenkins**

**Deploy the application in AWS**

**AWS Overview :**

**EC2, S3 and deploy the MERN Stack Application**

**4 and 4 days.**

**Testing**

**Testing is use to find the defect or error or bugs in application.**

**JavaScript**

**function add(a,b){**

**// coding then result**

**return 0;**

**}**

**Testing : types of Testing**

1. **Black box testing: we are not checking internal logic**

**Input -------🡪 Process -------🡪Output**

**Java**

**A, B**

**10, 20**

**Result 30**

**Expectation and result match then test is pass**

**Else fail.**

1. **White box testing : we are checking the coding.**

**Input -------🡪Process ---------🡪Output**

**Unit Testing : unit is where we write the code like function, module, class, interface etc. It is smallest block to perform a specific task using respective programming language.**

**function, method, package, module, classes, etc.**

**Unit testing is a type of white box testing.**

**Unit testing is a kind of software testing method or function which each individual and independent part of the source code is tested.**

**Unit testing we will do early stages.**

**When we perform unit testing on the front end (client side) of software. Then is called as front end unit testing.**

**Jasmine : Jasmine is a open source framework which provide set of pre-defined function which help to do the unit testing for front end technologies.**

**BDD and TDD**

**Behaviour Driven Development : BDD is a development technique which focuses more on the software application behaviour. In BDD involves product manager, developer and tester.**

**Test Driven Development : TDD is a development technique which focuses more on the implementation of a features of application / product. In TDD involves only developers.**

**Jasmine is a type of BDD testing framework.**

**Jasmine we can do the testing two ways**

1. **Standalone without Node JS with browser**
2. **Using Node JS**

**Jasmine API**

1. **describe : describe is a pre-defined function part of jasmine framework which is known as test suite. Describe like a block which can contains more than one it function as well as another describe functions.**

**Describe syntax**

**describe(“Message”,callback)**

**describe(“Message”,function() {**

**})**

**describe(“Message”,()=> {**

**})**

1. **it : it is pre-defined function part jasmine framework which is known as test spec . Test Speciation mainly use to test specific functionality of JavaScript code.**

**it(“Message”,callback)**

**it(“message”,function(){**

**})**

**it(“message”,() => {**

**})**

**Test suit contains more than one test spec**

1. **expect : expect is a pre-defined which provide lot of property which help in testing if the expectation from the software system are fulfilled or not.**

**describe(“Aritmeticoperation”,()=> {**

**it(“addition”,()=> {**

**// call function and get result**

**expect(30).toBe(result)**

**})**

**})**

**Karma : Karma is a test runner for the unit testing framework.**

**Karma is node base test tool allow you to test JavaScript code across multiple real browser. Chrome, IE, Edge etc.**

**Jasmine testing using standalone.**

**Jasmine and Karma with Node JS**

**Create new folder : jasmin and karma with node js**

**Create the package.json file using**

**npm init –y**

**Then install the jasmine dependencies**

**npm install –D jasmine**

**npm install –g karma-cli**

**npm install –D karma**

**npm install –D karma-jasmine**

**npm install –D karma-chrome-launcher**

**now we have create karma configuration file**

**karma init**

**JavaScript divided into 2 types**

**Client Side Scripting Language ES5 and ES**

**Server Side Scripting Language (Node JS)**

**Client Side JavaScript Testing**

1. **Jasmine and With Karma.**
2. **Jasmine with Karma Node JS (source code and testing kept one file).**
3. **Angular Framework : Angular framework provide all configuration for Jasmine and Karma to unit testing for TypeScript.**
4. **React JS : React JS is a part of Facebook. They provide another testing framework ie JEST.**
5. **JEST testing framework is a part of Facebook. So using JEST we can do testing client side JavaScript program as well as React JS programs.**

**Server side scripting ie JavaScript testing.**

1. **Using jasmine we can do node Js testing.**
2. **Using mocha we can do node js testing**
3. **Mocha with chai to Express js testing.**

**Node JS program testing using Jasmine tool**

**Create the package.json file**

**npm init –y**

**npm install –D jasmine**

**npm install –D jasmine-node**

**if**

**npm install –g jasmine**

**jasmine init**

**This command is use to create the Spec folder.**

**By doing this, jasmine create a spec directory and configuration json file for you(jasmine.json). This spec folder is use to store all your test files.**

**By doing this, jasmine will know where all your tests are and then can execute them accordingly.**

**OperationSpec.js**

**Operationspec.js**

**Express JS Testing**

**If we want to do the testing for Express JS get(), post(), put() or delete() method we have to take the help of SuperTest external module.**

**SuperTest is an http assertion (assumption) library that allow you to test your Node JS with Http Server using Express Module.**

**Jasmine with SuperTest for Http protocol methods testing.**

**Create the project**

**Create the package.json file**

**npm init –y**

**npm install express**

**npm install –D jasmine**

**npm install –D jasmine-node**

**npm install –D supertest**

**npm install body-parser**

**first create app.js file with Http methods**

**then create jasmine configuration**

**jasmine init**

**Http protocol get, post, put and delete methods call are asynchronous communication**

**Mongo DB compass**

**Mocha**

**Mocha is JavaScript testing framework.**

**Mocha is a younger than Jasmine**

**Jasmine came in picture 2008.**

**Mocha cam in picture 2011**

**Mocha is not a complete test framework and doesn’t attempt to be. Instead of this mocha covers the basic and allows developers to extends it with other framework.**

**Mocha doesn’t provide assertion (type of expect) library so we have to depends upon the other assertion library like Chai, should.js, expect.js, better-assert.js**

**Mocha is very simple testing framework. Using Mocha we can do testing for browser as well as node js application.**

**Mocha with karma browser or Client side JavaScript testing.**

**Using mocha we can do asynchronous as well as http call testing very easily with other assertion library.**

**Create**

**mocha and chai folder**

**Then create node js folder**

**Then create package.json file**

**Using npm init –y**

**npm install –g mocha**

**Node JS provide core assert module which help to do sample assertion operations.**

**Mocha with node js assert module to do simple testing.**

**Mocha allow us to use any assertion library our wish.**

**Chai : expect() style, assert() style, should() style**

**Should.js**

**Expect.js**

**Chai is assertion external library which provide different types of styles which help to do the testing with mocha or other framework.**

**var expect = require(“assert”)**

**expect.equals()**

**Create the folder with name**

**Chai module**

**Then create the package.json file**

**Chai with assert style. Assert style expose through assert interface. This provide the classical assert dot notation, similar to node js assert library.**

**Chai with expect style**

**Expect style provide us chain of function together like a natural language assertions.**

**Chai with should style**

**Should style allow the same chainable assertion statement as the expect style. However it extends each object will with a should property to start your chain of functions.**

**Mocha with chai testing**

**Create package.json file**

**Npm init –y**

**npm install mocha**

**npm install chai**