**React**

**Created By :- Himanshu Vishwakarma**

**Day = 2 26 nov 24**

**Promise static Method**

**1.Promise.all()**

* It will take one argument that is array of promises
* This method will return new promise of resolved state if all the promises is resolved .
* If **any one promise also is in rejected state** then this method will **return promise** with **rejected state.**
* If the promise is resolved then the **promise result will be the array of all the promises** and the promise is in **rejected state the promise result will be 1 rejected value**.

**2. promise.any ()**

* This also take one argument that array of promise.
* This method **will return promise** with resolved state **if any one promise also resolved** .
* This method will return **promise with rejected** state if all the promise is rejected .
* If resolved then the promise result will be first resolved value.
* If rejected then promise result will be aggregate error all promises were rejected.

**3. promise all settled()**

* This also take1 argument array of promise.
* It will always return promise with resolved state.
* The promise result will be the array of object and it hold the status and value or reason for all the promises if it is resolved also rejected also.

**4.promise.race()**

* This also take 1 argument array of promise.
* It will return the resolved promise or rejected promise Based on which function is calling first (I .E resolved or reject function ).

**ARGUMENT OBJECT**

* Arguments is a keyword which is used to fetch all the arguments Value inside the function without passing the parameter.
* It will return the argument object which is index data structure.
* We could not able to use argument keyword as identifier if the use script statement is there.

***Days = 3***   ***27 nov 24***

**Single page web application (SPA)**

* In single page web application the reloading will not happen for every functionality we are performing.
* In single web application If we send the request to the server, we will get the response & again if we are sending any new request to the server, we will get the new response and the old response will get store it the browser cache.
* single page web application must faster in performs then MTA.

**Multi page application (MTA)**

* In multi-page web application reloading will happen for every functionality we are performing
* In multi-page web application, every time both old and new request will go the server and will get new response.
* multi-page web application is better SEO then single page web application
* To create multi-page we can use technology like j query (JavaScript library)
* To create single page web application we can use technology like ( react, js library , framework , angular , veu )
* Mostly we can create e-commerce web-site any traditional website , like banking Educational website as a MPA
* We create social-media application or any streaming web application as spa EX facebook email

**Library**

* A library is collection of pre return that develops can use it preform some specific task
* The developer calls the library function method is needed
* The developer will control flow of application
* Library usually focus on particular functionalities or set of related functionalities
* Libraries are tool which you can use in your application
* Ex :- React , jQuery

**Framework**

* A Framework provides structured foundation or architecture for your application
* The framework controls the flow of program
* Developers can use component if need ( Classes & functions).
* Framework is a set of libraries or package and it includes tool for multiple aspects of development
* Ex :- (Database, UI , Routing).
* Ex for framework :- Angular, node js , D jango

**Days -4**  **28 NOV 24**

***Specifications of React***

**1.Component based architecture: -**

* React is used to create SPA architecture & application are build using reusable bits of code called component
* Each component can manage own states and logic and we can use component multiple time.

**2.Declarative in nature:-**

* Reacts allow developers to describe what the UI should like based on the application state

**3. Virtual DOM:-**

* React application are faster in performance because its follows concept virtual DOM
* When the webpage is loaded both actual DOM and virtual DOM is created for webpage
* If we make any changes in the component UI first it will create new virtual DOM then it compares the both actual DOM and virtual DOM & makes the necessary update in actual DOM

**What is virtual DOM?**

It is a light weight copy of real DOM which is used to optimize the update in DOM.

**What is reconciliation?**

* reconciliation is a process of comparing actual DOM and virtual DOM when the DOM updates or state of the component changes
* It will add only the changes which is made in virtual DOM to the actual DOM.

**Diffing Algorithm: -**

* when diffing two trees, react first compares the two root elements. The behavior is different depending on the types of the root elements.

**NPM(Node Package Manager) & NPX(Node package execute)**

NPM is the package manager for JavaScript that comes with node.js and it is used to install , uninstall & update the JavaScript packages or libraries required for development

- Manage dependencies in a project using package. JSON file

**How to install packages locally and globally in NPM?**

* + npm install package-name –g (globally)
  + npm install package-name (locally)
  + For uninstall packages we can use the comment
  + npm uninstall package-name –g (globally)
  + npm uninstall package-name (locally)

**NPX: -**

* NPX is a tool bundle with NPM version 5.2 which is used to execute the packages directly without globally installing them
* To execute the package, we can use comment
* npx package-name

**29 november 2024**

**create-react-app package**

its present in npm and its use to create new react application which involves basic project project structure and necessary.

**to install package** we can use command = npm install create -react-app & -g

**to create the react application** we use the command create -react -app appName

**(**name should not in capital letters**)**

**app name should be in lower case it didn’t accept upper case**

**npm start**

Starts the development server.

**npm run build**

Bundles the app into static files for production

**npm test**

starts the test runner.

**npm run eject**

removes this tool and copies build dependencies configuration files and scripts into the app directory . if you do this . you can’t goback!

after create the project to navigate app directory we can use the command CD appName.

**Folder and file details of react app when we create using create-react-app package**

1. node models it contents all the dependences and libraries install for the project.

We generally don’t modified anything in node module folder and it handle by npm & yarn

1. public folder it contains static files and assists that will not process by webPack . and this static files we can use for our application . Inside this public folder we have one key file called index.html which in running on the browser when we start the folder

and it contain one element call div or rendering the react components.

1. src folder : it’s a main directory where we can write react code ( components , styles, etc) within the src mandatory we have to create one index.js file which is the entry point for the react app. The file renders the react app into the dom by targeting the div tag with id root in the index.html file.

Package.json file: it contains the metadata about project which include project name , description and version dependencies .

**React package:**

* It is used to create the root element for the react application so that we can make every component as a child of this root element using render()
* It is present inside ReactDOM package

React.createElement():

* This method is present within the react package and it is used to create new react element and it will take 3 arguments
* 1 element-type
* 2 attribute in the form of obj {},null
* 3 content “ ”,child

Code:

import React ,{createElement}from "react";

import ReactDOM from "react-dom/client";

let div=document.querySelector("#root");

let x=ReactDOM.createRoot(div);

let form=createElement(

    "form",{className:"form-control"},

    createElement("h1",null,"form-details"),

    createElement("input",{type:"text",name:"username",placeholder:"username"},),

    createElement("br"),

    createElement("input",{type:"text",name:"password",placeholder:"password"}),

    createElement("br"),

    createElement("button",{type :"submit"},"submit")

)

x.render(form)

**02 dec 2024**

**JSX**

JSX stands for JavaScript xml, and it is used to write html code in react.

JSX allows us to write the html elements in JavaScript and place them in the DOM without any create element and append child ()

JSX converts html tags into react elements

**RULES OF JSX:**

1. If we define multiple react elements mandatorily it should wrapped within the parent element i.e. one top level element should be present. Alternatively, we can use fragment also.
2. Each and every tag must be closed.
3. If multiple elements are there, it should be within the parenthesis {} .

**CHANGES IN HTML ATTRIBUTES:**

1. In react, class attribute should be named as className property because, class is the reserved word (key word) in JavaScript.
2. For attribute in label tag should named as html for property in react because for is also a keyword.

**EXPRESSIONS IN JSX:**

* With JSX, you can write the expressions within the {}.
* The expression can be a react variable or a property or any validation or any other JavaScript expression. JSX will execute the expression, and it will return the result.

**FRAGMENTS:**

* A fragment looks like and empty html tag (<> </>)
* Instead of taking one top level elements for JSX alternatively we can use fragments to wrap multiple lines this will prevent unnecessarily adding extra nodes to the DOM.

**CONDITIONS IN JSX:**

* React supports if statements but not inside JSX.
* We can use if statements outside JSX and we can take the result inside the JSX or we can use ternary expression inside the JSX.

**COMPONENT IN REACT:**

Components are the independent reuse able bits of code and returns JSX.

There are two types of components:

1. Class based component (stateful)
2. Function based component (stateless)

**Code:**

import React from "react";

import ReactDOM, {createRoot } from "react-dom/client";

import "./global.css";

let h1 =(

    <div>

        <h1>from</h1>

        <br/>

        <label htmlFor="username"> Enter username </label>

      <input type="text" id ="username" name="username" placeholder="username"> </input>

        <br/><br/>

        <label htmlFor="dob">Date of birth</label>

        <input type="date" id="date" name="date"></input>

        <br/><br/>

        <label htmlFor="password">password</label>

        <input type="password" id="password" pattern="password"></input>

    </div>

);

createRoot(document.querySelector("#root")).render(h1);

**global.css**

\*{

    padding: 0px;

    background-color: bisque;

}

h1 {

    color: rgb(160, 201, 233);

    font-size: 36px;

    text-align: center;

    background-color: green;

    border: 2px solid black;

  }

Output:



Code:

import React from "react";

import ReactDOM, {createRoot } from "react-dom/client";

let str = "Javascrpit";

var res = false;

let r1;

if (res) r1 = "react";

else r1 ="node";

let h1 = (

    <>

    <h1>{10+25}</h1>

    <h2>{str}</h2>

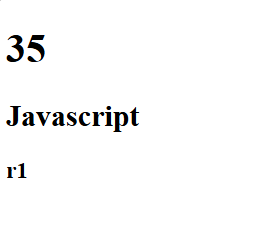
    <h3>r1</h3>

    </>

);

createRoot(document.querySelector("#root")).render(h1);

OutPut:

**Index.js**

import React from "react";

import { createRoot } from "react-dom/client";

import App from "./App.jsx";

// Create a root element and render the App component

createRoot(document.querySelector("#root")).render(<App />);

App.jsx

// class components

import React from "react";

class App extends React.Component {

  render() {

    return <h1>Hello, World!</h1>;

  }

}

export default App;

//! Function Component

import React from "react";

function App(){

    return<h1>hello markus</h1>

}

export default App;

**03-12-2024**

**Class Based Components: (CBC)**

* These components are simple JavaScript classes made-up of multiple function that add

functionality to the application.

* Every class components are **stateful** i.e. CBC we can define state with the help of state property (inbuilt property for every class based components).
* Every class component must inherit react. Components class and it should have render Method which returns JSX

**Function Based Components (FBC)**

* These components are simple JavaScript functions (arrow function , named function, function with expression) that return JSX.
* Function based components are **stateless** i.e we don’t have any inbuilt property called as state to define the data for that particular components.
* But after introduction of react hooks , we can able define the state in function based components also we can use usestate()hook

**State in react**

* State holds the information or data about that particular component.
* We could not able to transfer the state from one components to another components.
* State are mutable (we can change)
* Whenever there is any changes in the state value the components will re-render.

**How to define state in CBC?**

* We can define state in two ways in CBC
* using state property inside class and outside the constructor
* using this.state inside the constructor
* In CBC state value should be obj or null otherwise it will throws warning App.state : must be set to an object or null
* Inside the constructor 1 statement should be the super calling statement which calls the parent class constructor
* In CBC 1 constructor will excute after that render() will execute.

**by using state property**

//^HOW TO DEFINE THE STATE IN CLASS BASED COMPONENT(CBC)

import React, {Component} from "react";

class App extends Component{

   //& by using state property

    state = {

        name:"pain",

        id: 101,

        skills: ["js", "java"],

    };

    render(){

        return(

            <div>

                <h1>USERNAME: {this.state.name}</h1>

                <h2>ID:{this.state.id}</h2>

                <h3>Skills:{this.state.skills}</h3>

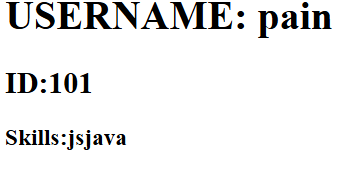
            </div>

        );

    }

}

export default App;



//^HOW TO DEFINE THE STATE IN CLASS BASED COMPONENT(CBC)

import React, {Component} from "react";

class App extends Component{

      //& by using constuctor

    constructor(){

        super();

        this.state = {

            name:"MARkus",

            id: 102,

            skills :["js", "reacct"],

        };

        console.log(this);//!app component

    }

    render(){

        return(

            <div>

                <h1>USERNAME: {this.state.name}</h1>

                <h2>ID:{this.state.id}</h2>

                <h3>Skills:

                    <ul>

                        {this.state.skills.map((v) => {

                            return <li>{v}</li>

                        })}

                    </ul>

                </h3>

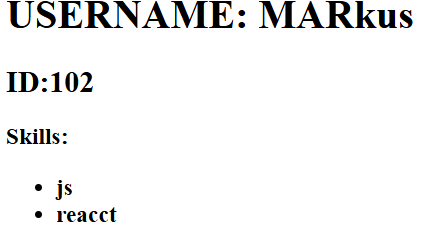
            </div>

        );

    }

}

export default App;

****

**How to update the state value in CBC?**

* Every class component have one inbuilt method i.e setState() which is used to update state value in CBC and it will take one argument i.e value we are going to update (obj or null)
* //^HOW TO UPDATE SATATE VALUE IN CBC
* //? The state is directly initialized as a class property,
* import React, { Component } from "react";
* export default class App extends Component {
* state = { name: "Markus", id: 10 };
* componentDidMount() {
* setTimeout(() => this.setState({ name: "Sachin", id: 200 }), 3000);
* }
* render() {
* const { name, id } = this.state;
* return (
* <div>
* <h1>Username: {name}</h1>
* <h2>ID: {id}</h2>
* </div>
* );
* }
* }

****

**Task:**

**App.jsx**

import React from "react";

import shoping from "./data.json"; // Importing the data

import "./App.css"; // Import the CSS file

        const App = () => {

            return (

              <div>

                <h1>Product List</h1>

                <div className="product-container" >

                {shoping.map((pr) => (

                  <div key={pr.id} className="product-card">

                    <img className="product-image" src={pr.image} alt={pr.title} />

                    <p>Product ID: {pr.id}</p>

                    <h3>{pr.title}</h3>

                    <p>{pr.description}</p>

                    <p>Category: {pr.category}</p>

                    <p>Price: ${pr.price}</p>

                    <p>Rating: {pr.rating.rate} ({pr.rating.count} reviews)</p>

                  </div>

                ))}

              </div>

            //   </div>

            );

          };

export default App;

**App.css**

.product-container {

    display: flex;

    flex-wrap: wrap;

    gap: 20px;

    justify-content: center;

    background-color: antiquewhite;

  }

  h1{

    background-color: #e7f30d;

    text-align: center;

  }

  .product-card {

    width: 300px;

    border: 1px solid black;

    border-radius: 8px;

    padding: 16px;

    color: white;

    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

    background-color: rgb(28, 151, 135);

    text-align: center;

  }

  .product-image {

    width: 100%;

    height: auto;

    max-height: 300px;

    object-fit: cover;

    border-radius: 4px;

  }

****

**4 dec 2024**

**React Hooks:**

* React Hooks were introduced from the version react 16.8.
* Hooks allows function component to have access to state and other react features so that class components are generally not needed.
* Hooks allow us to use Lifecyle methods and state.

**RULES OF HOOKS:**

There are three rules:

1) Hooks can only be called inside function components.

2) Hooks can only call at the top-level of the component.

3) Hooks cannot be conditional.

NOTE: Hooks will not work in react class based components.

We can also create custom hooks in react but it should start with use.

Hooks are simple JavaScript functions which is used to perform some functionality in function based component.

**How to define the state in Function Based Component?**

• Function Based Components are stateless i.e. we don’t have any inbuilt property to define the state but we can use useState hook to define the state value.

• useState hook will take only one argument i.e. state value which can be any datatype and it will return one array with two elements. 1st is Statevalue and 2nd is updater function.

• This updater function is used to update the state value and it will take one argument i.e. the value we are going to update.

**Events in React:**

• Events in React are triggered by user action such as clicking the button or handle by eventhandlers.

• We have to pass the eventhandler property such as onClick, onSubmit in the opening tag of the react element.

• This property will take one function as the value and this function will execute when the use triggers the event.

**How to use Events in React ?**

Code :- App.jsx

//! How to use MouseOver Event in React:

import React,{Component} from "react";

import "./App.css";

*const* App = () *=>* {

*let* handleMouseOver = (*e*) *=>*{

*e*.target.style.color = "blue"

*e*.target.style.backgroundColor = "red"

    }

*let* handleMouseOut = (*e*) *=>*{

*e*.target.style = "none"

    }

    return(

        <div>

            <h1 onMouseOver={handleMouseOver} onMouseOut={handleMouseOut}>

                Himanshu Vishwakarma

            </h1>

        </div>

    )

}

export default App;