Module-5

ReactJS

Introduction - JSX - React Components - Classes, Passing Data using Properties and Children - React State - Initial State, Updating State, Event Handling, Stateless Components, Designing Components -React Forms - Controlled Components, More Filters, Specialized Input Components, Server Rendering - Basic Server Rendering, Webpack for the Server

React JS-Introduction

- Currently, ReactJS is one of the most popular JavaScript front-end libraries which has a strong foundation and a large community.
- ReactJS is a **declarative**, **efficient**, and flexible **JavaScript library** for building reusable UI components.
- It is an open-source, component-based front end library which is responsible only for the view layer of the application.
- The main objective of ReactJS is to develop User Interfaces (UI) that improves the speed of the apps
- It uses virtual DOM (JavaScript object), which improves the performance of the app. The JavaScript virtual DOM is faster than the regular DOM.

- **Declarative**
- React allows you to describe **what** you want the UI to look like, not **how** to achieve it. Instead of manually manipulating the DOM (like with vanilla JavaScript), you just update the state, and React handles updating the UI efficiently. Example:
- const Greeting = ({ name }) => <h1>Hello, {name}!</h1>;
- You simply declare that you want to display a greeting, and React figures out how to update the UI.
- **Efficient**
- React uses a **virtual DOM**—an in-memory representation of the real DOM. When changes occur, React compares the new virtual DOM with the previous version (this process is called *reconciliation*) and only updates the parts of the real DOM that actually changed.
  This minimizes costly DOM operations, making updates faster and more efficient.
- **Flexible**
- React can be used with other libraries and frameworks (like Redux for state management or Next.js for server-side rendering). You can build anything from simple components to full-blown applications, and it's adaptable for mobile (via **React Native**) or even desktop apps.
- React promotes breaking your UI into independent, reusable components. Each component manages its own state and logic, making your code modular and easier to maintain.
  Example:const Button = ({ label, onClick }) => (<button onClick={onClick}>{label}</button>);
- You can use this <Button /> component multiple times with different props throughout your app.

- **Open-source**
- React is free to use, and its source code is available for anyone to view, modify, and contribute to. Maintained primarily by **Meta (formerly Facebook)** and a community of developers, it evolves through community contributions.
- **Component-based**
- In React, the UI is built using **components**—independent, reusable pieces of code that represent parts of the user interface (like buttons, forms, or entire sections). Each component can manage its own state and logic, making development modular and organized.
  Example:
- const Header = () => <header><h1>Welcome to My App</h1></header>;
- You can combine multiple small components to build complex interfaces.
- **Front-end Library (View Layer Only)**
- React focuses solely on the **view layer** of an application in the **Model-View-Controller (MVC)** architecture. It handles how the UI looks and updates based on changes in data but doesn't handle things like routing or data fetching directly. You can use libraries like **React Router** for navigation or **Redux** for state management.

- It was created by **Jordan Walke,** who was a software engineer at **Facebook.** It was initially developed and maintained by Facebook and was later used in its products like **WhatsApp** & **Instagram.**
- Facebook developed ReactJS in **2011** in its newsfeed section, but it was released to the public in the month of **May 2013.**
- A ReactJS application is made up of multiple components, each component responsible for outputting a small, reusable piece of HTML code.
  - **The components are the heart of all React applications.** These Components can be nested with other components to allow complex applications to be built of simple building blocks.
- ReactJS uses virtual DOM based mechanism to fill data in HTML DOM.
  - The virtual DOM works fast as it only changes individual DOM elements instead of reloading complete DOM every time.
- To create React app, we write React components that correspond to various elements.
  - We organize these components inside higher level components which define the application structure.
- For example, we take a form that consists of many elements like input fields, labels, or buttons. We can write each element of the form as React components, and then we combine it into a higher-level component, i.e., the form component itself.

The React Virtual DOM exists entirely in memory and is a representation of the web browser's DOM.

Why ReactJS?

- The previous frameworks follow the traditional data flow structure, which uses the DOM (Document Object Model).
  - DOM is an object which is created by the browser each time a web page is loaded
  - It dynamically adds or removes the data at the back end and when any modifications were done, then each time a new DOM is created for the same page.
  - This repeated creation of DOM makes unnecessary memory wastage and reduces the performance of the application.
- ReactJS allows you to divide your entire application into various components. ReactJS still used the same traditional data flow,
  - but it is not directly operating on the browser's Document Object Model (DOM) immediately; instead, it operates on a virtual DOM.
  - It means rather than manipulating the document in a browser after changes to our data, it resolves changes on a DOM built and run entirely in memory. After the virtual DOM has been updated, React determines what changes made to the actual browser's DOM.

React Environment Setup

- **Install NodeJS and NPM**
  - Node.js is a run-time environment which includes everything you need to execute a program written in JavaScript. It's used for running scripts on the server to render content before it is delivered to a web browser.
  - NPM stands for Node Package Manager, which is an application and repository for developing and sharing JavaScript code.
- Two ways to install ReactJS
  - Using the npm command
  - Using the **npm create vite@4.1.0** command (The build tool that aims to provide a faster and leaner development experience for modern web projects)

We are using Second Method

How to Install Node.js and NPM on Windows

- **Step 1:** In a web browser, navigate to https://nodejs.org/en/download/. Click the **Windows Installer** button to download the latest default version
- **Step 2: Install Node.js and NPM from Browser**

  1. Once the installer finishes downloading, launch it. Open the **downloads** link in your browser and click the file. Or, browse to the location where you have saved the file and double-click it to launch.

  2. The system will ask if you want to run the software – click **Run**.

  3. You will be welcomed to the Node.js Setup Wizard – click **Next**.

  4. On the next screen, review the license agreement. Click **Next** if you agree to the terms and install the software.

  5. The installer will prompt you for the installation location. Leave the default location, unless you have a specific need to install it somewhere else – then click **Next**.

  6. The wizard will let you select components to include or remove from the installation. Again, unless you have a specific need, accept the defaults by clicking **Next**.

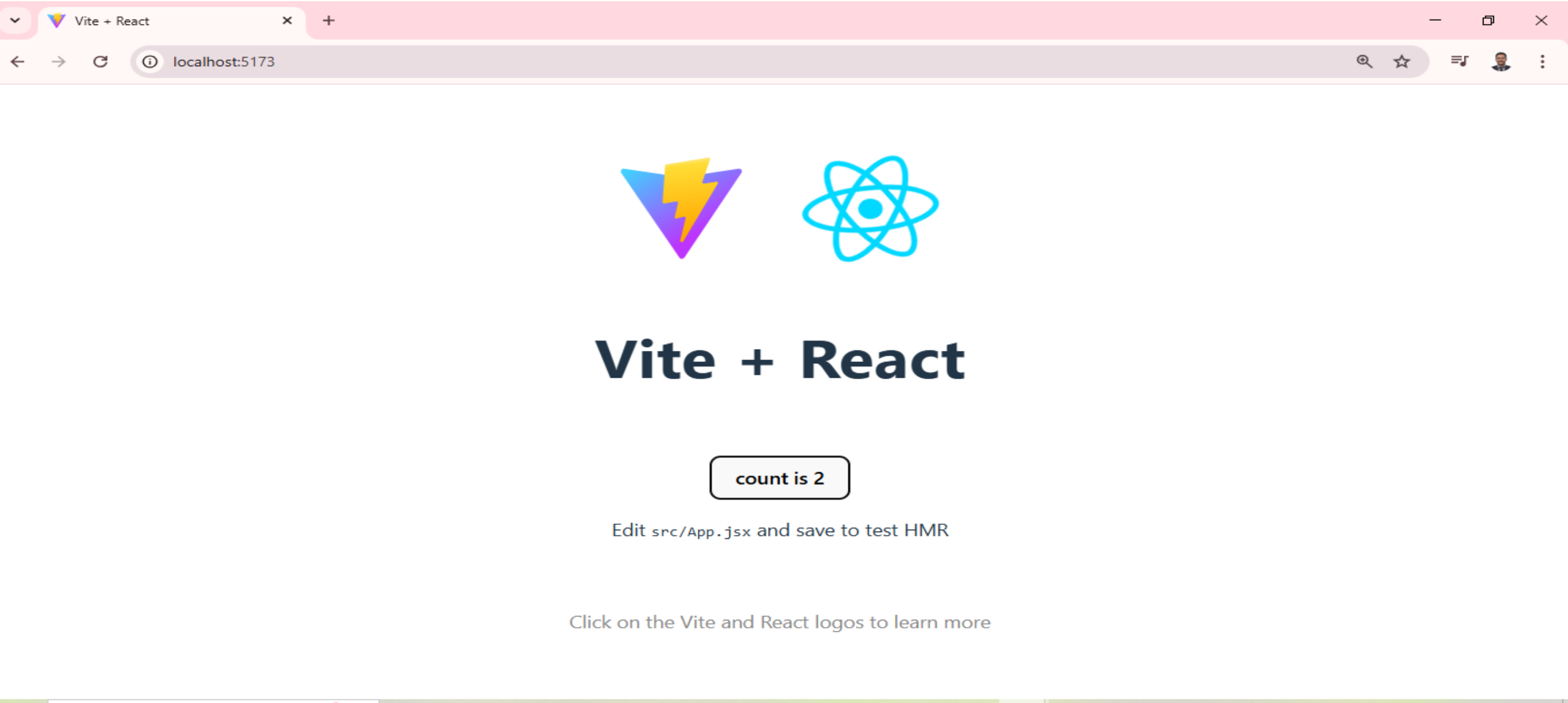
  7. Finally, click the **Install** button to run the installer. When it finishes, click **Finish**.

- **Step 3: Verify Installation**
  - Open a command prompt and enter **node –v** to display the Node.js installed in your system. Also check for **npm –v** to know the version of NPM

Using Vite(Pronounced as veet, which means fast in French

- **Step 4:** To get started, we need to open a command prompt using Win+R and type '*cmd*'. Then, type in the following command.
  - npm create vite@4.1.0
  - Enter ‘y’ to proceed
  - Enter Project Name: say ‘*reactpro*’
  - Select React as Framework
  - Select ‘Javascript’ as variant
- Step 5: Switch to ‘reactpro’ directory
  - Execute the command ‘***npm install***’ – it takes all dependencies mentioned in package.json
- Step 6: To run our new project, we need to use the command
  - Exceute ‘***npm run dev***’ command will run the project application.

Successful Execution-Output



Anatomy of React Application

- In React application, there are several files and folders in the root directory. Some of them are as follows:
  - **node_modules:** It contains the React library and any other third party libraries needed.
  - **public:** It holds the public assets of the application.
  - **Index.html:** where React will mount the application by default on the element.
  - **src:** It contains the App.css, App.jsx, index.css, main.jsx (javascript file corresponding to index.html). Here, the **App.jsx file always responsible for displaying the output screen in React**.
  - **package-lock.json:** It is generated automatically for any operations where npm package modifies either the node_modules tree or package.json. It cannot be published. It will be ignored if it finds any other place rather than the top-level package.
  - **package.json:** It holds various metadata required for the project. It gives information to npm, which allows to identify the project as well as handle the project's dependencies.

- Open the **src >> App.jsx** file and make changes which you want to display on the screen. After making desired changes, **save** the file.
- As soon as we save the file, Webpack recompiles the code, and the page will refresh automatically, and changes are reflected on the browser screen.

Important Features of ReactJS

- The important features of ReactJS are as following.
  - JSX - JavaScript XML
  - Components
  - One-way Data Binding
  - Virtual DOM
  - Simplicity
  - Performance

Pros & Cons

- Advantages
  - Easy to learn and Use- has a simple API and focuses mainly on the view layer, making it easier for developers
  - Creating Dynamic Web Applications Becomes Easier-to build interactive and dynamic UIs more efficiently with its component-driven architecture and state management features
  - Reusable Components-can be reused across different parts of the application, promoting cleaner, modular code and reducing development time
  - Performance Enhancement-**Virtual DOM** allows React to minimize direct manipulation of the real DOM, making applications faster and more efficient
  - The Support of Handy Tools-powerful developer tools like **React Developer Tools** for debugging and performance monitoring, making development smoother
  - Known to be SEO Friendly-can render on the server side using tools like **Next.js**, improving SEO by allowing search engines to index content more effectively
  - The Benefit of Having JavaScript Library-it allows seamless integration with other JavaScript libraries, offering flexibility for developers
  - Scope for Testing the Codes-React applications can be easily tested using tools like **Jest** or **Enzyme**, making it easier to ensure app reliability and performance

Pros & Cons

- Disadvantages
  - The high pace of development-React is frequently updated, which can make it challenging for developers to keep up with the latest changes and best practices.
  - Poor Documentation-Rapid updates often mean that official documentation can lag behind, making it harder to find up-to-date resources for newer features.
  - View Part-React only handles the UI (View) part of the application, so developers often need to rely on other libraries for routing, state management, or backend integration.
  - JSX as a barrier-can be confusing for new developers, especially those not familiar with JavaScript or modern front-end development.

React Architecture

- React's primary purpose is to enable the developer to create user interface using pure JavaScript.
- React introduces three simple concepts
  - React elements - React.createElement to create React Element.
  - JSX - JSX is an XML based, extensible language supporting HTML syntax with little modification. JSX allows us to write HTML directly within the JavaScript code.
  - React Component- React component is the primary building block of the React application.
    - It uses React elements and JSX to design its user interface.
    - React component is basically a JavaScript class (extends the ***React.component*** class) or pure JavaScript function.
    - React component has properties, state management, life cycle and event handle

Simple React Application

- Adding React to a Website
  - Step 1: Add a DOM Container to the HTML
    - First, open the HTML page you want to edit. Add an empty tag to mark the spot where you want to display something with React.
    - ** **
    - We gave this a unique id HTML attribute.
  - Step 2: Add the Script Tags
    - <script type="module" src="/src/main.jsx"></script>
  - Step 3: Create a React Component- App.jsx

```
function App() {
  return (
    <div>
      <b>First REACT App!!!</b>
    </div>
  )
}
export default App
```

  - Step 4: Import the React Component- ‘App’ in main.jsx

```
import App from './App'
ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
);
```

React JSX

- React JSX is an extension to JavaScript.
- It enables developer to create virtual DOM using XML syntax.
- It compiles down to pure JavaScript (*React.createElement function calls*). Since it compiles to JavaScript, it can be used inside any valid JavaScript code.
- JSX supports expression in pure JavaScript syntax. Expression has to be enclosed inside the curly braces, ***{ }***.

- **Variable Assignment**
  - var greeting = <h1>Hello React!</h1>

- **Assign to a variable based on a condition.**
  - var canGreet = true;
  - if(canGreet) {
  - greeting = <h1>Hello React!</h1>
  - }

- **Can be used as return value of a function.**
  - function Greeting() {
  - return <h1>Hello React!</h1>
  - }

    export default Greeting;
  - greeting = Greeting()

- **Can be used as argument of a function.**
  - function Greet(message) {
  - ReactDOM.render(message, document.getElementById('react-app')
  - }
  - Greet(<h1>Hello React!</h1>)

React Component

- React component is the building block of a React application.
- A React component represents a small chunk of user interface in a webpage.
- The primary job of a React component is to render its user interface and update it whenever its internal state is changed.
- React component provides below functionalities.
  - Initial rendering of the user interface.
  - Management and handling of events.
  - Updating the user interface whenever the internal state is changed.
- React library has two component types.
  - Function component – Uses plain JavaScript function.
  - ES6 class component – Uses ES6 class.

Component Lifecycle

- Every React Component has a lifecycle of its own, lifecycle of a component can be defined as the series of methods that are invoked in different stages of the component's existence.
- A React Component can go through four stages of its life as follows.
  - **Initialization:** This is the stage where the component is constructed with the given Props and default state. This is done in the constructor of a Component Class.
  - **Mounting:** Mounting is the stage of rendering the JSX returned by the render method itself.
  - **Updating:** Updating is the stage when the state of a component is updated and the application is repainted.
  - **Unmounting:** As the name suggests Unmounting is the final step of the component lifecycle where the component is removed from the page.

Functional Components(Stateless)

- Function components are a way to write components that only contain a render method and don't have their own state.

```
function MyFunction() {
    var cTime = new Date().toTimeString();
    return (
      <div><p>The current time is {cTime}</p></div>
    );
  }
```

- The functional component is also known as a stateless component because they do not hold or manage state

Refer MyFunction.jsx &MyFunction.css

Passing Data using Properties

- React components are designed to be reusable units. To enable reuse, components allow for data to be passed into them through properties or props.
- **props** are read-only values available to a single instance of a component. They can be set programmatically or with the same syntax you use when setting attributes for HTML elements

JavaScript

```
// TODO: Create RecipeTitle component
function RecipeTitle(props) {
    return (
        <section>
            <h2>{ props.title }</h2>
        </section>
    )
};

export default RecipeTitle;
```

First, it accepts a parameter named props. This will automatically contain all attributes or properties passed into the component when it's used.

Second, you're able to use the values contained in props like you would any normal parameter.

Class Components

- It requires you to extend from React. Component and create a render function which returns a React element

```
class MyComponent extends React.Component {
  render() {
    return (
      <div>This is main component.</div>
    );
  }
}
```

Refer MyComponentApp.jsx

- The class component is also known as a stateful component because they can hold or manage local state.

React JS Styling

- React allows component to be styled using CSS class through className attribute.
- Three important methodology to style our component
  - CSS Stylesheet
  - Inline Styling
  - CSS Modules
- CSS Stylesheet- refer myfunction.jsx
  - Simply create a CSS stylesheet for a component and enter all your styles for that particular component. Then, in the component, use className to refer the styles.
- Inline Styling- refer main.jsx
  - Style can be directly set inside the component
- CSS Modules
  - While importing the styles, CSS modules converts all the styles into locally scoped styles so that the name conflicts will not happen.

- **Create a CSS module file**
  Name your CSS file with .module.css, for example:
- Css file
- /* styles.module.css */
- .heading { color: blue; font-size: 24px; text-align: center; }
- **Import the CSS module in a React component**
- Jsx file
- import styles from './styles.module.css';
- function App()
- { return <h1 className={styles.heading}>Hello, CSS Modules!</h1>; }
- export default App;
- **On the set**
- CSS Modules **automatically generate unique class names** (e.g., .heading_abc123).
- This prevents global scope issues where styles override each other.

Properties

- React enables developers to create dynamic and advanced component using properties. Every component can have attributes similar to HTML attributes and each attribute's value can be accessed inside the component using properties (props).
- <Hello name="React" />

// value of name will be "Hello* const name = this.props.name

- React properties supports attribute's value of different types. They are as follows,
  - String
  - Number
  - Datetime
  - Array
  - List
  - Objects

Refer PassingDataUsingProps.js

Children Prop

- In React, children is a special prop that allows components to be passed as data to other components.
- The content passed to a component through props.children can include **undefined, null, a Boolean, a number, a string, a React element, or an array** of any of these types recursively.
- Whenever this component is invoked, {props.children} will also be displayed and this is just a reference to what is between the opening and closing tags of the component.

- Example: Parent Component

```jsx
import React from 'react';
import ChildComponent from './ChildComponent'

const ParentComponent=()=>{

    return(
        <div style={{ backgroundColor: 'blue', padding: '10px', width: '50vw' }} >
            <h3>This is ParentComponent</h3>
            <ChildComponent>
                <p style={{ backgroundColor: 'white', padding: '10px' }} >
                    This paragraph will show up in the child, but was sent from the parent.
                </p>
            </ChildComponent>

        </div>
    );
}

export default ParentComponent;
```

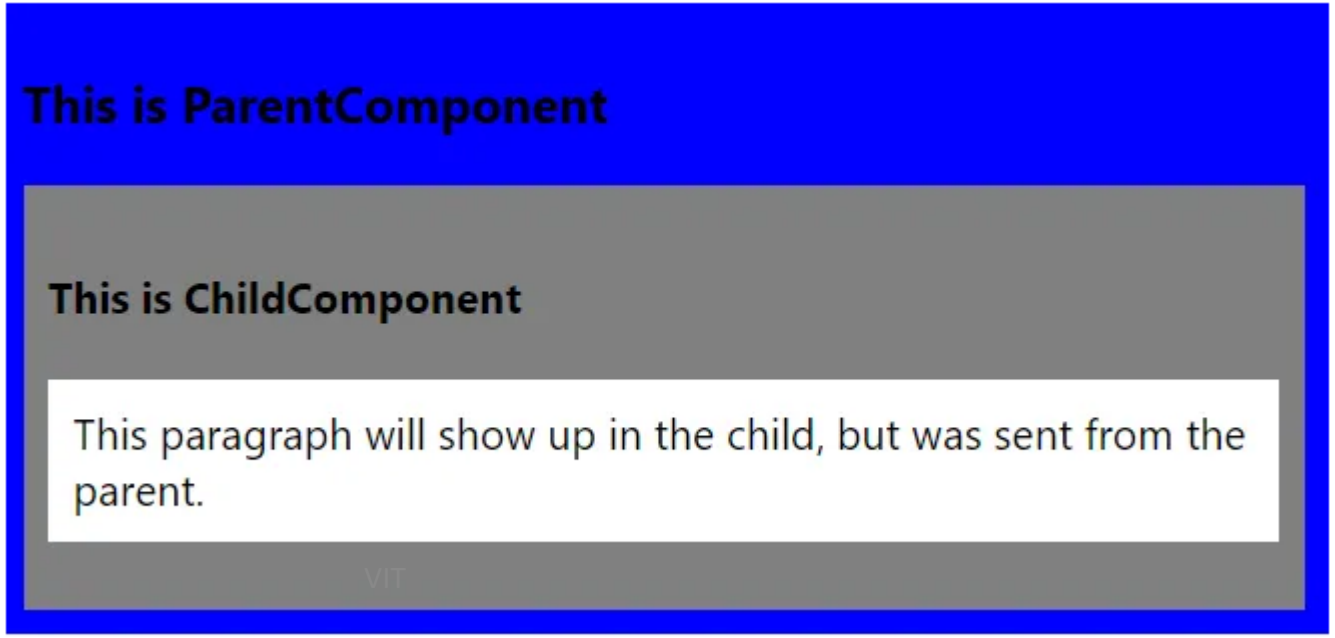
Implementing <ParentComponent />: The <ParentComponent /> defines a paragraph that will be passed down to the <ChildComponent /> as props.

- Child Component

```jsx
import React from 'react';

const ChildComponent=(props)⇒{

    return(
        <div style={{ backgroundColor: 'grey', padding: '10px' }} >
            <h4> This is ChildComponent </h4>
            { props.children }
        </div>

    );
}

export default ChildComponent;
```

Implementing <ChildComponent />: In the ChildComponent. { props.children } will render any data that has been passed down from the parent, allowing the ParentComponent to customize the contents of the ChildComponent.

- Output



Refer ListItemMethod component in MyFunction.jsx for the usage of props.children

Constructors

- A constructor is a method that is automatically called when an object of that class is created.
- It is not necessary to have a constructor in every component.
- It is necessary to call super() within the constructor. To set property or use 'this' inside the constructor it is mandatory to call super().
- It used for initializing the local state of the component by assigning an object to this.state.
- It used for binding event handler methods that occur in your component.

Refer MultiGrocery.jsx

State

like a storage for data that can change over time, & React uses it to make sure the UI updates properly when data changes.

- State of a component is an object that holds some information that may change over the lifetime of the component.
- The state is an updatable structure that is used to contain data or information about the component.
- The change in state over time can happen as a response to user action or system event.
- A component with the state is known as stateful components. It is the heart of the react component which determines the behavior of the component and how it will render(how it looks on the screen).
- To define a state, you have to first declare a default set of values for defining the component's initial state.
- Add a class constructor which assigns an initial state using this.state. The '**this.state**' property can be rendered inside **render()** method.
- The **bind()** is **an inbuilt method in React that is used to pass the data as an argument to the function of a class based component**. It ensures they work properly in a class-based component.

Refer CounterClass.jsx

useState() hooks

- The useState hook in React is a function that lets you add state variables to functional components.(Earlier, state was only available in **class-based components**, but with hooks, we can now manage state in functional components as well.)
- What can you do with useState?
  - Add state to a component
  - Update state based on the previous state
  - Update objects and arrays in state
  - Avoid recreating the initial state
  - Keep track of strings, numbers, booleans, arrays, and objects
- useState returns an array with two values: the current state and a function that updates it
- You can use the useEffect hook to perform actions based on the updated state

Refer Counter.jsx

Props Validation

- used to pass data from a parent component to a child component. Since props are essential for communication between components, it's important to make sure that the data being passed is of the correct type and structure. This is where **props validation** comes in.
- Props validation is a tool that will help the developers to ensure the correct usage of your components.
- React components used special property **PropTypes** that help you to catch bugs by validating data types of values passed through props.
- **App.propTypes** is used for props validation in react component.
- PropTypes exports a range of validators that can be used to make sure the data you receive is valid. (process of ensuring that the data passed to a component via props matches the expected type, and meets certain requirements (like being required or having a default value). This helps to catch potential bugs and improve the overall quality of your application)
- - npm install prop-types -import PropTypes from 'prop-types';
- States Vs Props
  - Props are read-only components. State is mutable.
  - State cannot be accessed by child components
  - State cannot make components reusable.

Range of Validators and Custom Validation

- String
- Function
- Objects
- Arrays
- Boolean
- InstanceOf…

Types of Prop Validations:

- `PropTypes.string` : Validates that the prop is a string.
- `PropTypes.number` : Validates that the prop is a number.
- `PropTypes.bool` : Validates that the prop is a boolean.
- `PropTypes.object` : Validates that the prop is an object.
- `PropTypes.array` : Validates that the prop is an array.
- `PropTypes.func` : Validates that the prop is a function.
- `PropTypes.node` : Validates that the prop is anything that can be rendered (strings, numbers, elements, etc.).

Validators- Example

```jsx
import PropTypes from 'prop-types';

function MyComponent({ name, age }) {
  return (
    <div>
      <h1>{name}</h1>
      <p>Age: {age}</p>
    </div>
  );
}

// Prop validation
MyComponent.propTypes = {
  name: PropTypes.string.isRequired,  // 'name' should be a string and is required
  age: PropTypes.number.isRequired   // 'age' should be a number and is required
};
```

Custom Validation

- ReactJS allows creating a custom validation function to perform custom validation.
- (allows you to define your own logic for validating props passed to a component, beyond the standard built-in prop types. This is useful when you need to impose more complex validation rules, such as checking the range of a number, validating a specific pattern for a string, or checking if a prop meets multiple conditions)
- The following argument is used to create a custom validation function.
  - **props:** It should be the first argument in the component. The object containing all the props passed to the component.
  - **propName:** It is the propName that is going to validate.
  - **componentName:** It is the componentName, where the validation is being done

React JS Forms

- It allows the users to interact with the application as well as gather information from the users. Forms can perform many tasks that depend on the nature of your business requirements and logic such as authentication of the user, adding user, searching, filtering, booking, ordering, etc.
- The component rather than the DOM usually handles the React form.
- In React, the form is usually implemented by using controlled components.
- There are mainly two types of form input in React.
  - Uncontrolled component-Uses **refs** to access values **directly** from the DOM
  - Controlled component-Uses useState or this.state to store form values

Uncontrolled Component

- Uncontrolled component does not support React based form programming. (**because it does not track input values in state—it directly accesses them using React.createRef() instead**)
- To get the content of the react component, we use React *ref* feature.
- Steps
  - 1. React.createRef() to create a new reference **(this.ref)**.
    - **this.inputRef = React.createRef();**
  - 2. The newly created reference can be attached to the form element
    - **<input type="text" name="username" ref={this.inputRef} />**
  - 3.Attached form element's value can be accessed using **this.ref.current.value** whenever necessary
    - **alert(this.inputRef.current.value);**
- To set default value of an input element, use *defaultValue* attribute instead of *value* attribute. If value is used, it will get updated during rendering phase of the component.

Refer UncontrolledForm.jsx

Controlled component

- In the controlled component, the input form element is handled by the component rather than the DOM.
- Controlled components have functions that govern the data passing into them on every **onChange event**
- This data is then saved to state and updated with setState() method. This makes component have better control over the form elements and data.
- A controlled component takes its current value through **props** and notifies the changes through **callbacks** like an onChange event.
- Steps
  - 1. Create a form element. Add a onChange attribute and assign a handler method.
    - <input type="text" name="username" onChange={this.handleUsernameChange} />
  - 2. Write the handler method and update the state whenever the event is fired. Bind the event handler in the constructor of the component.
  - 3. Get the input value using **username** from **this.state** during validation and submission.

Refer ControlledForm.jsx

Differences

Feature	Controlled Component	Uncontrolled Component
Data Storage	Stored in `state`	Stored in the DOM (`ref`)
Input Handling	`onChange` updates state	Uses `React.createRef()`
Submission	Reads from state	Reads from `ref.current.value`
Use Case	Recommended for form handling	Simple, non-tracked input fields

Events

- React has its own event handling system which is very similar to handling events on DOM elements. The react event handling system is known as Synthetic Events.
- The synthetic event is a cross-browser wrapper of the browser's native event.

Refer ControlledForm.jsx

- Handling events with react have some syntactic differences from handling events on DOM. These are:
  - React events are named as **camelCase** instead of **lowercase**.
  - With JSX, a function is passed as the **event handler** instead of a **string.**
  - In react, we cannot return **false** to prevent the **default** behavior. We must call **preventDefault** event explicitly to prevent the default behavior.

React List

- Lists are used to display data in an ordered format and mainly used to display menus on websites. In React, Lists can be created in a similar way as we create lists in JavaScript.
- Generate a list of items from an array:
  - Move the data into an array: const myList = ['ONE', 'TWO', 'THREE', 'Four', 'FIVE'];
  - Map the myList members into a new array of JSX nodes, listItems:
    - const listItems = myList.map((numeral)=>{ return **
{numeral}
**});
  - Return listItems from your component wrapped in a
    - return {listItems};
    -
      <li ONE
      TWO
    -

Refer ListProc component in MyFunction.jsx

Keys

- A key is a unique identifier. In React, it is used to identify which items have changed, updated, or deleted from the Lists.
- Keys should be given to the elements inside the array to give the elements a stable identity
- The best way to pick a key is to use a string that uniquely identifies a list item among its siblings.
- When you don't have stable IDs for rendered items, you may use the item index as a key as a last resort **(Not Recommended)**
- Keys are internal to React and can not be accessed from inside of the component like props
- **Const arrayelt = ['ab', 'bc', 'kd', 'lk', 'kj']**
- **<li key=0 value=ab>**
- **<li key=1 value=bc>**
- **<li key=2 value=kd>**
- **Const arrayelt=['ab', 'kj', 'kd', 'lk', 'bc']**

Refer NumberListMethod component in MyFunction.jsx