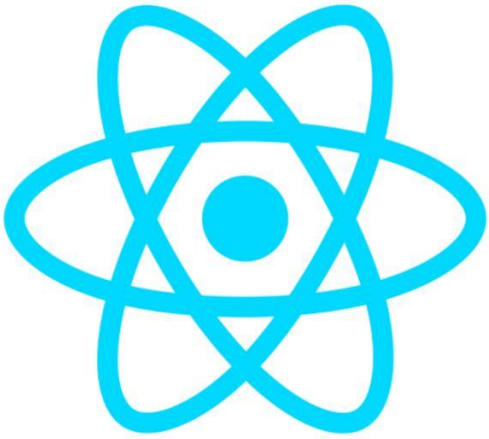
Semester: V **Subject:** Web Computing **Academic Year:** 2023 – 2024



React.Js

React Conditional Rendering:-

There is more than one way to do conditional rendering in React.

- if
- It is the easiest way to have conditional rendering in React in the render method.

```
function UserLogin (props){
    return <h1> Welcome Back </h1>;
}
function GuestLogin(Props ) {
    return <h1> Please Sign Up </h1>
}
function SignUp (props){
  const isLoggedIn = props. isLoggedIn;
  if (isLoggedIn ){
      return <User Login />;
  }
 return <GuestLogin />;
  }
ReactDom. render {
  <signUp isLoggedIn = {false} />
  document. get Element By Id ('root')
);
```

Logical && operator.

The operator is used to checking the condition.

```
import React from 'react';
import ReactDOM from 'react-Dom';
function Example ()
```

```
return (<div>
 { (10>5) && alert ('This alert will be
        Shown!")
 }
 </div>
);
}
```

Ternary Operator :-

```
render ( ) {
 const isLoggedIn = this.state.isLoggedIn;
 return {
  <div>
Welcome {isLoggedIn ? 'Back ; Please login}
 </div>
 };
}
```

Switch Case Operator

```
functional NotificationMsg ({text}) {
  Switch (text) {
   case 'Hi all';
       return <Message: text ={text}/>;
   case "Hellow JavaTpoint ";
       return <Message text ={text}/>;
   default
    return null;
  }
 }
```

Conditional Rendering with enums

An enum is a great way to have a multiple Conditional rendering. It is more readable as compared to switch case operator.

```
function NotificationMsg({text, state}) {
  return(
    <div>
    {{
      info: <Message text={text} />
          warning : <Message text={text} />
      }[state]}
    </div>
  );
}
```

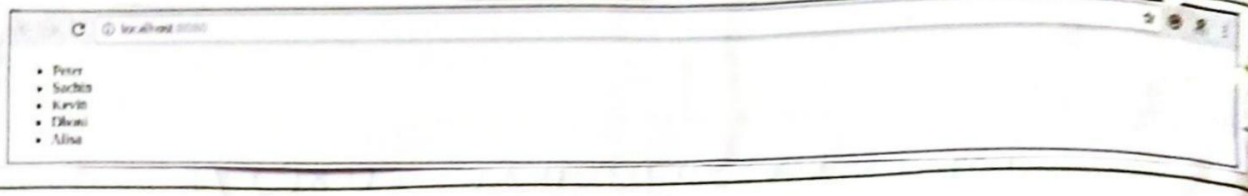
React Lists:-

Lists are used to display in an ordered forma and mainly used to display menus on websit The map() function is used to traversing the list. Include the new list <ul> elements and render it to dom.

```
import React from 'react';
import ReactDOM from 'react-dom';
const myList = ['Peter', 'sachin', 'kelvin', 'Dhoni
  'Alisa'];
Const myListItems = myList.map((myList)=> {
    return <li> {mylist} </li>
});
ReactDOM.render (
    <ul> {listItems} </ul>
```

```
document.getElement By Id ('app')
);
Export default App;
```

Output:



React Keys :-

A key is unique identifier. In React, It is used to identify which items have changed, updated or deleted from the lists. It is useful when we dynamically created component or when the user alert the Lists. It also helps to determine which component in a collection need to be rendered instead of re-rendering the entire set of components every time.

```
Const StringList = ['Peter', 'Sachin, 'Kevin',
        'Dhoni', 'Alisa'];
Const updatedLists = StringLists.map ((StrList
    <li key ={strList.id} > {strList} </li>
  } );
```

If there are no stable IDs for rendered items, you can assign the item index as a key of the lists. It can be shown in the below,

```
const stringLists = ['Peter', 'Sachin', 'Kevin', 'Dhoni',
    'Alisa'];
const updatedLists = stringLists.map((strList, index)=>
    {
  <li key={index} > {strList} </li>
  });
```

React Refs:-

Refs is the shorthand used for referances in Reac
It is an attribute which makes it possible to store a referance to particular DOM nodes or React elements. It provides a way to access React DOM nodes or React elements and how to interac with it.

When to Use Refs:-

- When we need DOM measurements such as managing focus, text selection, or media playback.
- It is used triggring imperative animations.
- When integrating with third-party DOM libraries.
- It can also use as in callbacks.

When to not use Refs:-

- It is use should be avoided for identify anything that can be done, declaratively. insted of using open() and close() methods on a Dialog component, you need to pass an isOpen prop to it.
- You should have to avoid overuse of the Refs.

How to create Refs.

In React, Refs can be created by using React.createRef(). It can be assigned to React Element via the ref attribute. It is commonly assigned to an instance property when a component is created and them can be referenced throughout the component.

```
class MyComponent extends React.Component {
  constructor(props) {
  super(props);
  this.callRef = React.createRef();
  }
  render() {
    return <div ref={this.callRef} />
  } }
```

How to access Ref's

When a ref is passed to an element inside render method. a referance to the node can be accessed via the current attribute of the ref.

```
const node = this.callRef.current;
```

Ref current Properties

- When the ref attribute is used in HTML element, the ref created with React.createRef() receives, the underlying Dom element as its current property.

- If the ref attribute is used on a custom class component then ref object receives the mounted instance of the component as its current property.
- The ref attribute cannot be used to on function component because they dont have instances.

Callback Refs :-

There is another way to use refs that is called 'callback refs' and it gives more control when the refs are set and unset. Insted of creating refs by passing a callback function to the ref attribute of a component.

```
<input type ="text" ref ={element => this.callRefInput =element}/ >
```

The callback function is used to store a referance to the Dom node in an instance property and can be accessed elsewhere.

```
this.callRefInput.value
```

React with useRef ()

It is introduced in React 16.7 and above version It helps to get access the Dom node element and the we can interact with the Dom node or element such as focusing the element or accessing the input element value It returns the ref object whose .current property initiallized to the passed argument.

```
const refContainer = useRef(initialValue);

function useRefExample (){
   const inputRef = useRef (null);
   const onButtonClick =()=>{
   inputRef.current.focus ();
   };
   return (
    <>
<input ref={inputRef} type ="text"/>
<button onClick ={onButtonClick}> Submit
                                  </button>

    </>
  );
}
```

React Fragments:-

The render method can return single elements or multiple element. The render method will only render a single node inside at a time.

```
class App extends React.Component {
   render (){
      return (
   <div>
   <h2> Hellow World </h2>
   <P> Welcome to the JavaTpoint </P>
   </div>
   );
  }
}
```

Syntax :-

```
<React Fragment>
  <h2> Child 1 </h2>
  <p> child 2 </p>
  ..... ...
</React.Fragment>
```

Why we use Fragments?

① It makes the execution of code faster as compared to the div tag.

② It take less Memory.

Fragments Short Syntax:-

There is also another shorthand exists for declaring fragments for the above method. It looks like empty tag in which we can use '<>' and insted of the 'React.Fragment'

```
Class Colums extend React.Component {
  render() {
    return (
    <>
    <h2> Hello World! </h2>
    <p> Welcome to the JavaTPoint </p>
    </>
    );
  }
}
```

Key Fragments:-

The shorthand syntax does not accept key

key attribute. You need a key for mapp
a collection to an array of fragments s
as to create a description list.

```
Function = (props) {
  return (
    <Fragment>
  {props.item.data.map(item => (
    <React.Fragment key = {item.id}>
      <h2> {item.name} </h2>
      <p> {item.url} </p>
      <p> {item.description} </p>
    </React.Fragment>
  ))}
  </Fragment>
  )
}
```

React Router :-

Routing is a process in which a user is dire
to diffrent pages based on their action
or request. ReactJs Router mainly use
for developing Single Page application.
React Router is used to define multi
routes in the application.
React Route is a standard library syst
built on the Top of the React and use
to create routing in the React applic
using React Router package. It provi
the synchronous URL on the browser
with data that will be displayed o
the web page.

React contain 3 diffrent package of routing

① react-router :- It provides the core routing Componants and function for the React Router application.

② react-router-native:-
It is used for mobile application.

③ react-router-dom :-
It is used for web applicatron design.

$npm install react-router-dom --save

Components in React Router.

<BrowserRouter>
- It is used for handling URL

<HashRouter>
- It is used to handling static request.

Benefits of React Router:-

① It is not necessary to set the browser history manually.

② Link uses a navigate the internal links in the application. It is similar to anchor tag.

③ It uses switch feature for rendering

④ If the router needs only a single child element.

React CSS

CSS in React is used to style the React App or component. The style attribute is the most attribute for styling in React applications which adds dynamically-computed styles at render time. It accepts a javascript object in camelcased properties rather than CSS String

① Inline Styling

The inline styles are specified with a javascript object in camelcase version of the style name. Its value is the styles value which we usually take in a string.

App.Js

```
import React from 'react';
import ReactDOM from 'react-dom';
class App extends React Component{
  render() {
    return (
    <div>
    <h1 style={{ color:"Green" }} Hello JavaT</h1>
    <p> Here you can find all Cs Tutorials </p>
    </div>
  );
  }
}
export default App;
```

② CSS Stylesheet :-

You can write styling in a seperate file for your React application, and save the file with a css extension. You can import the file in your application.

App.Js.

```
import React from 'react';
import ReactDom from 'react-dom';
import '/App.css';
class App extends React Component {
  render (
   <div>
   <h1> Hello JavaTpoint </h1>
   <p> You can find all cs tutorials </p>
  </div>
 );
} }
export default App;
```

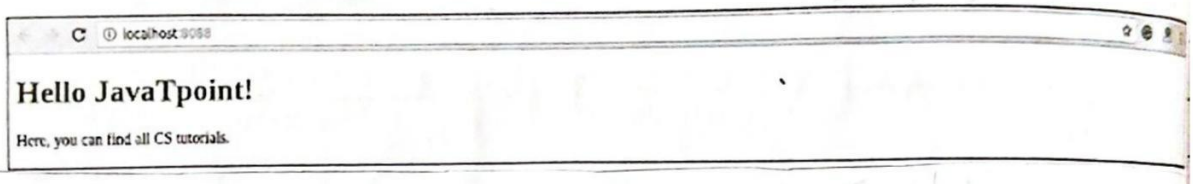
App.css

```
body {
 background color : #008080;
  color : yellow;
  padding : 40 px;
   font-family : Arial;
    text-align : center;
  }
```

Index.html

```
<!DocTYPE html>
  <html lang = "en" >
```

```
<head>
<meta charset = "utf-8" />
 <meta name = "viewport"
  content = "width-device-width,
            initial-scale=1" />
 <title> React App </title>
</head>
<body>
  <div id = "app" ></div>
 </body>
</html>
```



③ CSS Module

CSS Module is another way of adding styles to your application. In a css file where all class names and animation names are scoped locally by default. It is available only for the component which import means any styling you add can never be applied to other components without your permission, and you never need to worry about name conflicts.

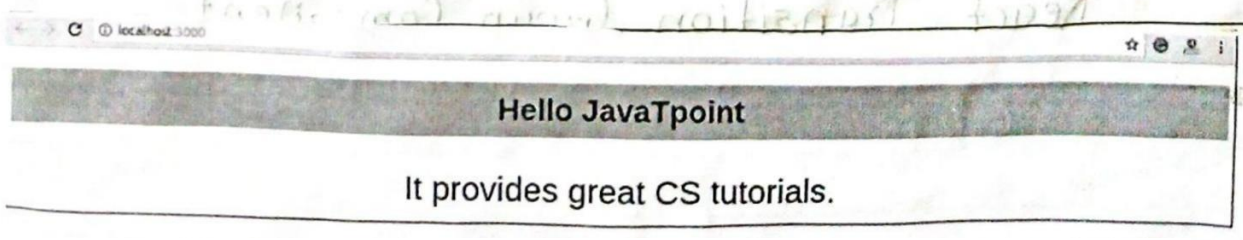
App.Js.

```
import React from 'react';
import ReactDom from 'react-dom';
import styles from '/myStyles.modules.css
```

```
class App extends React.component {
render() {
  return(
  <div>
  <h1 className={style.mystyle}> Hellow JavaT</h1>
  <p className={style.parastyle}> It provides great
   cs tutorials </p>
  </div>
  );
   }
}
export default App;
```

mystyles.module.css

```
.mystyle {
 background-color : #cdc0b0;
  color : Red;
  padding : 10 px;
   font-family : Arial
    text-align : center
  }
  .parastyle {
     Color : Green;
     font-family : Arial;
     font-size : 35 px;
     text-align : center;
  }
```



4. Styled Component:-

Styled component is a library for React. It uses enhance css for styling React component System in your application. which is written with mixture of Javascript and CSS.

The styled component provides.

- Automatic Critical css.
- No class name bugs.
- Easier deletion of css.
- Simple dynamic styling.
- painless maintenance.

React Animation.

The animation is a technique in which images are manipulated to appear as moving images. It is one of the most used technique to make interactive web application

React Transition group has mainly 2 API Create Transition.

① React Transition Group:-
It uses as low-level API for animation.

② React CSS Transition Group:-
It uses as a high level API for implementing basic CSS transitions and animations.

React Transition Group Component.
Provides 3 main component.

① Transition
② CSS Transition

Transition Group.

Transition:- It has a simple component API to declare a transition from one component state to another over time. It is mainly used to animate the mounting and unmounting of a component. It can also be used fro in-place transition state as well.

① entering ② entered.
③ existing ④ exited.

CSS Transition:-
The CSS transition component uses stylesheet classes to write the transition and create animations. It is inspired by the ng-animate library. It can also inherit all the props of the transition component.

- Appear
- Enter
- Exist.

Transition Group:-
This component is used to manage a set of transition component in a list. It is a state machine that controls the mounting and unmounting components overtime. The transition component does not define any animation directly. 'Transition Group' component can have diffrent animation with a component.

App.Js

```
import React, {Component } from 'react';
import {CSS TransitionGroup } from 'react-transiti
                                    group';
class App extends React.Component {
 Constructor (props ) {
 Super (props );
  this.state ={items ['Blockchain', 'React Js',
      'TypeScript', 'JavaTpoint' ]};
  this.handle Add = this. handle Add . bind (this );
  }
 handle fAdd ( ){
  Const newsItems = this. state. Items. Concat ([
   prompt ('Enter Item Name')
  ]);
  this.setState ({items: newsItems });
 }
 handleRemove (i){
  let newItems =this.state. items. slice ( );
  newsItems. splice (i, 1 );
  this.set State ({items : newsItems });
  }
 render ( ){
  Const items =this.state. items. map ((item, i)=>
  <div key ={item } onClick ={( )=>
                    this.handle Remove (i)}>
    {item}
  </div>
 ));
 return (
   <div>
 <h1> Animation Examples </h1>
```

```
<button onClick={this.handleAdd}> Insert them
        </button>
<CSSTransitionGroup
    transitionName="example"
    transitionEnterTimeOut={800}
    transitionLeaveTimeout={600}>
    {items}
</CSSTransitionGroup>
</div>
);
}
}
export default App;
```

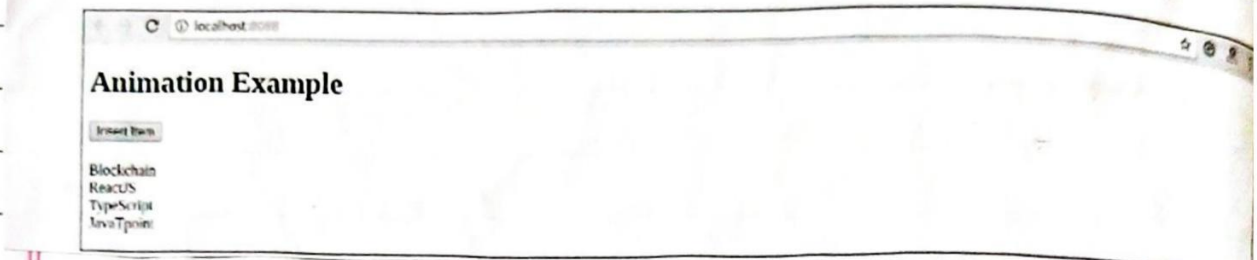
Main.Js

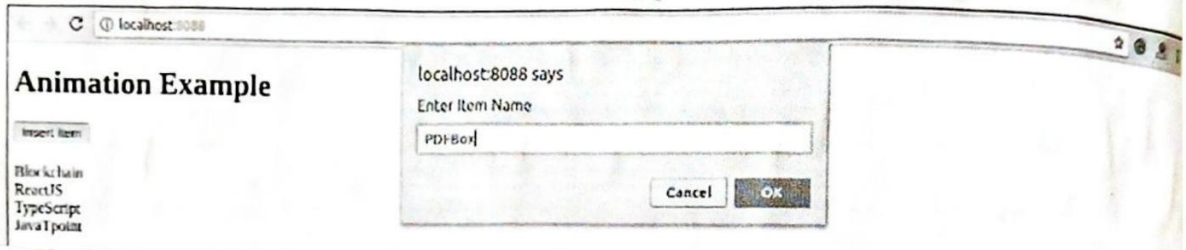
```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App.Js';
ReactDOM.render (<App/>, document.getElementById
    ('app'));
```

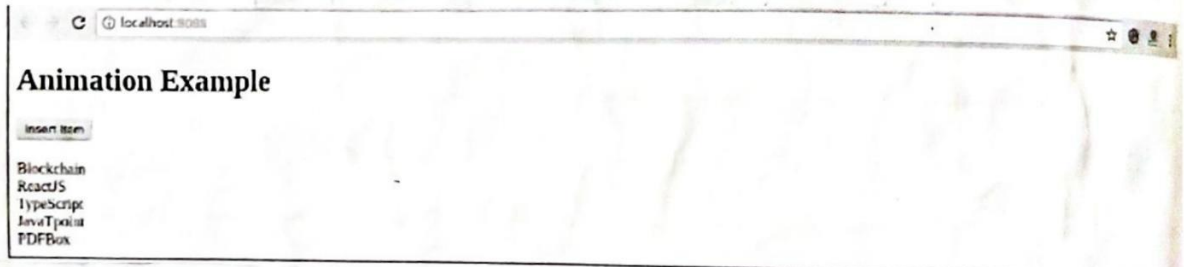
Style.css

```
.example-enter{
    opacity:0.01;
}
.example-enter.example-enter-active{
    opacity:1;
    transition: opacity 500ms ease-in;
}
.example-leave{
    opacity:1;
    }
```

```
.example-leave .example-leave-active {
  opacity : 0.01;
  transition: opacity 300ms ease-in;
}
```







React Map :-

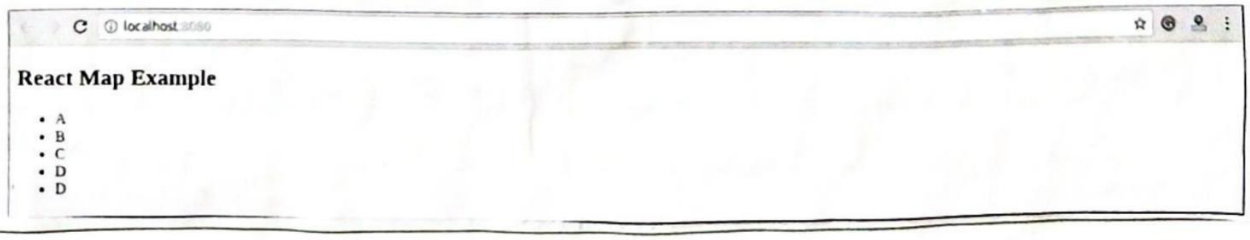
A map is a data collection type where data is stored in the form of pairs. It contains a unique key. The value stored in the map must be mapped to the key. We cannot store duplicate pair in the map. It is because of the uniqueness of each stored key. It is mainly used for fast searching and looking up data.

In React, the map() method used for:

① Traversing the list element.

```
import React from 'react';
import ReactDOM from 'react-dom';
function NameList(props){
  const myList(props) {
    Const listItems = myList.map((myList) =>
  <li> {myList} </li>
);
return (
  <div>
    <h2> React Map example </h2>
      <ul> {listItems} </ul>
    </div>
  );
}
const myLists = ['A','B','C','D'];
ReactDOM.render (
  <Namelist myList = {myLists} />
  document.getElementById ('app')
);
export default App;
```

Output-



React Table :-

A table is an arrangement which organizes information into 'rows and columns. It is used to store and display data in a struct format

The react-table is light weight, fast, fully customizable and extendable Datagrid built for React. It is fully controllable via optional and callbacks.

Features:-

① It is lightweight at 11 kb.
② It is fully customizable.
③ It is fully controllable via optional props and callbacks.
④ It has client-side & servr-side pagination
⑤ It has filters
⑥ Pivoting & Aggregation.
⑦ Minimal design & easily themeble

```
import React, { Component } from 'react';
import ReactTable from "react-table";
 import "react-table/react-table.css";
class App extends Component {
 render() {
  const data = [{
       name : "Ityaan";
       age : 26;
     }.{
     name : 'Ahana',
      age : 22
     }, {
```

```
name: 'Peter',
age: 40
},{
name: 'Dhoni',
age: 37
}]
Const columns =[{
Header: 'Name',
accessor: 'name'
},{
Header: 'Age',
accessor: 'age'
}]
return (
<div>
<ReactTable
data={data}
columns={columns}
defaultPageSize={2}
pageSizeOptions={[2,4,6]}
/>
</div>
) } }
export default App;
```

React Higher-Order Component

It also known as HOC. In React, HOC is an advanced technique for reusing component logic It is a function that takes a component and returns a new component

```
const NewComponent = higherOrderComponent
                     (wrappedComponent)
```

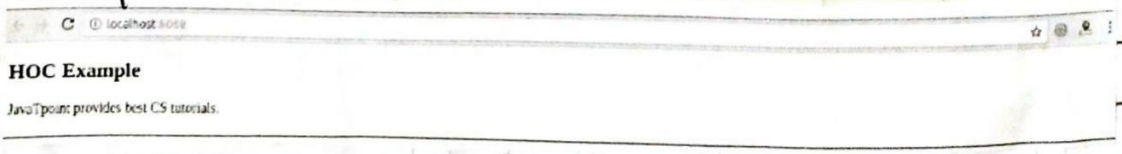
HOC.JS

```
import React, {Component} from 'react';
export default function HOC(HOCComponent){
 return class extends Component {
  render(){
   return (
   <div>
   <HOCComponent> </HOCComponent>
  </div>
 ); } }}
```

App.Js

```
import React {Component} from 'react';
import HOC from '/HOC';
Class App extends Component {
render (
 <div>
 <h2> HOC Example </h2>
  JavaTPoint provides best CS Tutorial
  </div>
 ) } } }
 App = HOC(App);
  export default App;
```

Output -



React Code Splitting :-

The react app budled their files using tools like Webpack or Browsefy. Bundling is a process which takes multiple files and merge them into a single file, which is called bundle. The bundle is responsible for loading an entire app at ones on the webpage.

Code splitting improves:-

1. The performance of the app.
2. The impact on memory
3. The downloadable killobytes size.

React.lazy:-

The best way for code splitting into the app is through the dynamic import () syntax. The React.lazy function allows us to render a dynamic import as a regular component.

Before:-

```
Import ExampleComponent from '/ExampleComponent';
function MyComponent (){
 return (
   <div>
   <ExampleComponent />
   </div>
 );
}
```

After:-

```
Const ExampleComponent = React.lazy (() =>
          import ('/ExampleComponent'));
```

```
function MyComponent () {
  return (
    <div>
    <ExampleComponent/>
    </div>
  );
}
```

Error Boundaries:-

If any module fails to load, for example, due to network failure, we will get an error. We can handle these errors with Error Boundaries. Ones we have created the Error Boundary, we can use it anywhere

```
import MyErrorBoundary from '/MyErrorBoundary
const ExampleComponent = React.lazy(() =>
                    import ('/Example Component'
Const ExampleComponent = React.lazy ())
 Const MyComponent = () => (
   <div>
   <MyErrorBoundary >
  Suspence fallback = {<div >Loading </div>}
   <section>
    <ExampleComponent/>
    <ExampleComponent />
    </section>
   </suspense>
  </MyErrorBoundary>
  </div>
  );
```

React Context :-

Context allows passing data through the component tree without passing props down manually at every level.

React Context API :-

1. React.createContext
2. Context.provider.
3. Context.Consumer.
4. class.contextType.

① React.createContext :-

It creates a context object

```
const MyContext = React.createContext(default.value
```

② Context.Provider :-

Every Context object has a provider React component which allows consuming components to subscribe to context changes.

```
<MyContext.Provider value = {/* Some value */}>
```

③ Context.Consumer.

It is an React Component which subscribes to the context changes. It allows us to the context within an function.

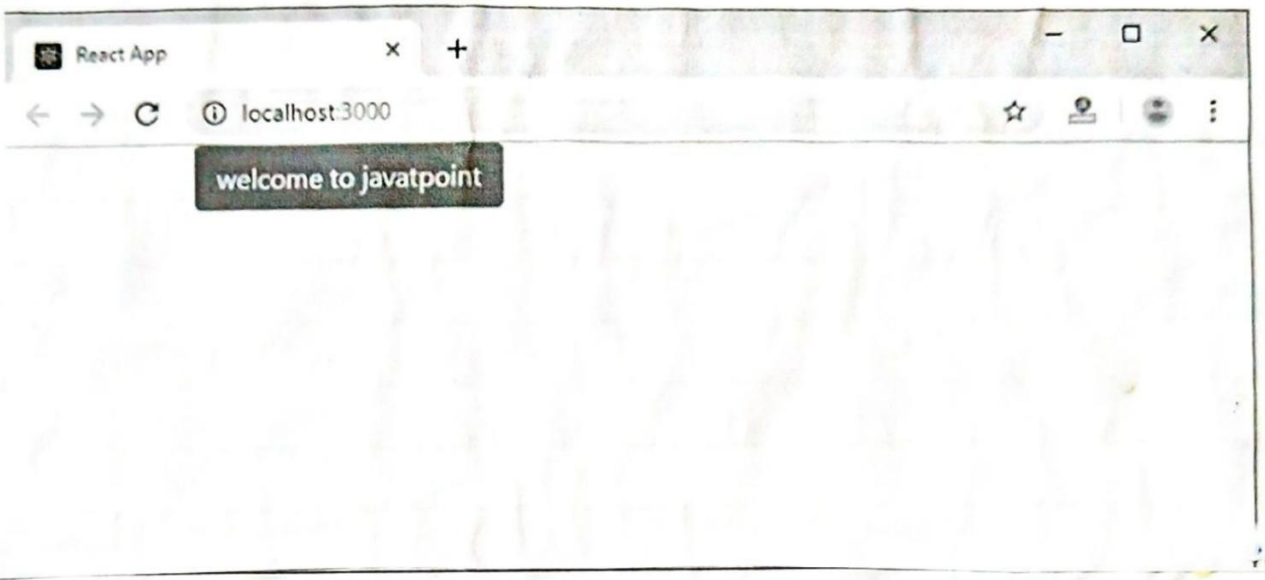
```
<MyContext.Consumer>
    {value => /* render something which is
        based on the context value */}
```

Class contextType:-

The contextType property on a class used to asign a context object which is created by React.createContext(). It allows you to consume the closet current value of that context using this.context.

```
import React, {Component} from 'react';
import 'bootstrap/dist/css/bootstrap.min.css';
Const BtnColorContext = React.createContext('
  btn btn-darkyellow");
class App extends Component {
  render() {
    return (
<BtnColorContext.Provider value='btn btn-info
</Button>
</BtnColorContext.Provider>
  );
  }
}
function Button(props) {
  return (
  <div className = "container">
  </ThemeButton />
</div>
  };
}
class ThemedButton extends Component {
  static contextType = BtnColorContext;
  render() {
  return < button className = {this.context}>
  Welcome to JavaTpoint
```

```
</button>
}
}
export default App;
```



React Hooks:-

Hooks are the new feature introduced in React 16.8 version. It allows you to use state and other React feature without writing a class.

Rules of Hooks:-

1. Only call Hooks at the top level
2. Only call Hooks from React functions.

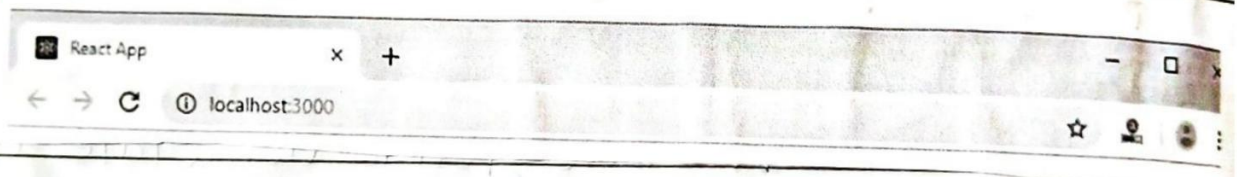
Hooks State :-

Hook state is the new way of declaring a state in React app. Hook uses useState() functional component for setting and retrivi g state.

App.Js

```
import React, {useState} from 'react';
```

```
function CountApp() {
  const [count, setCount] = useState(0);
  return (
  <div>
  <p> You clicked {count} times </p>
  <button onClick={() => setCount(count+1
   Click me
  </button>
  </div>
  );
}
export default CountApp
```



React Flux Vs MVC

MVC

MVC stands for Model View Controller. It is an architectural pattern used for developing the user interface

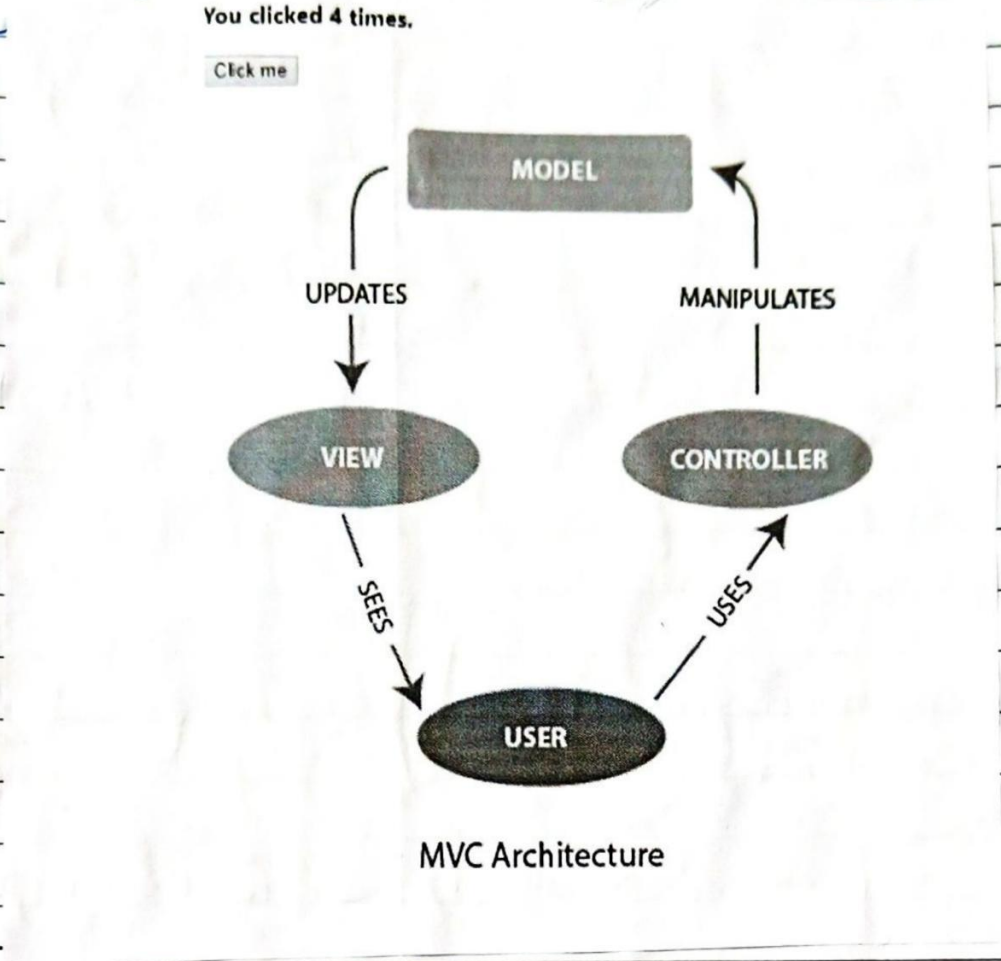
MVC Architecture :-

Model :- It is responsible for maintaining th behaviour and data of an application

View :- It is used to display the model in the user interface.

Controller :- It acts as an interface between

the Model and the view Components. It takes user input, manipulates the data and causes view

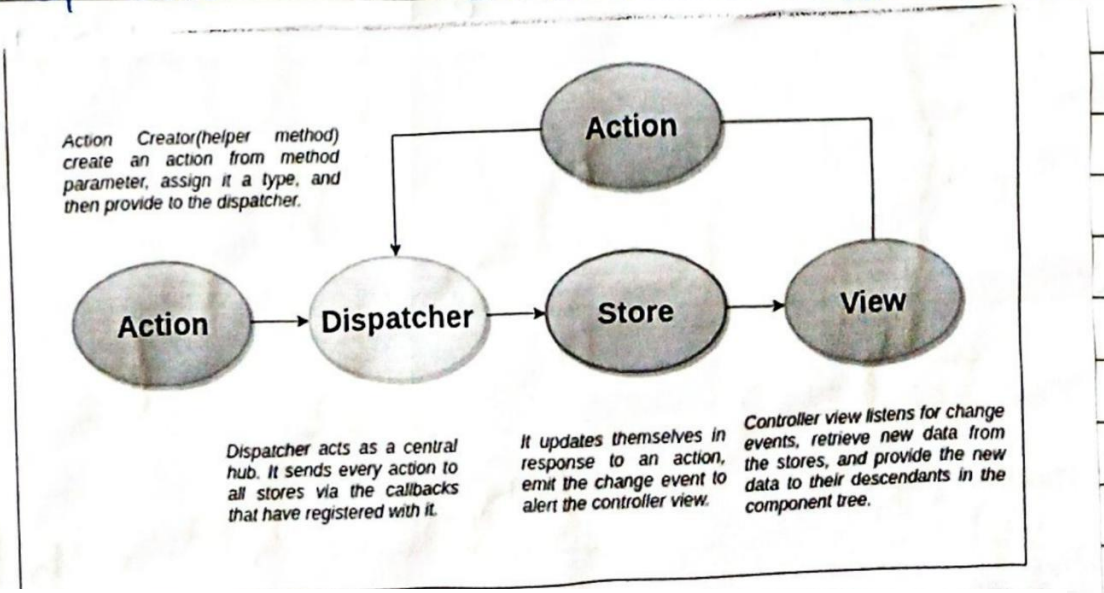


MVC Architecture

Flux :-

Flux is an application architecture that Facebook uses for building client-side web application Flux architecture has 3 Major rules in dealing with data :-

① Dispatcher ② Store ③ views.



	MVC	Flux
①	It is introduced in 1976	It was introduced just a few years ago
②	It supports Bi-directional data Flow model	It support Uni-directional data flow model
③	In this, data binding is the key	In this events or actio are key
④	It is synchronous	It is asynchronous.
⑤	Controllers handle everything	Store handle all logi
⑥	It is hard to debug	It is easy to debug.
⑦	It is difficult to understand as project size increases	It is easy to the understand.
⑧	Its maintainability is difficult as the project scope goes huge.	It maintainability is easy and reduces run-time errors.
⑨	Testing of application is difficult.	Testing of application is easy.
⑩	Scalability is complex	It can be easily sca lable.

React Redux :-

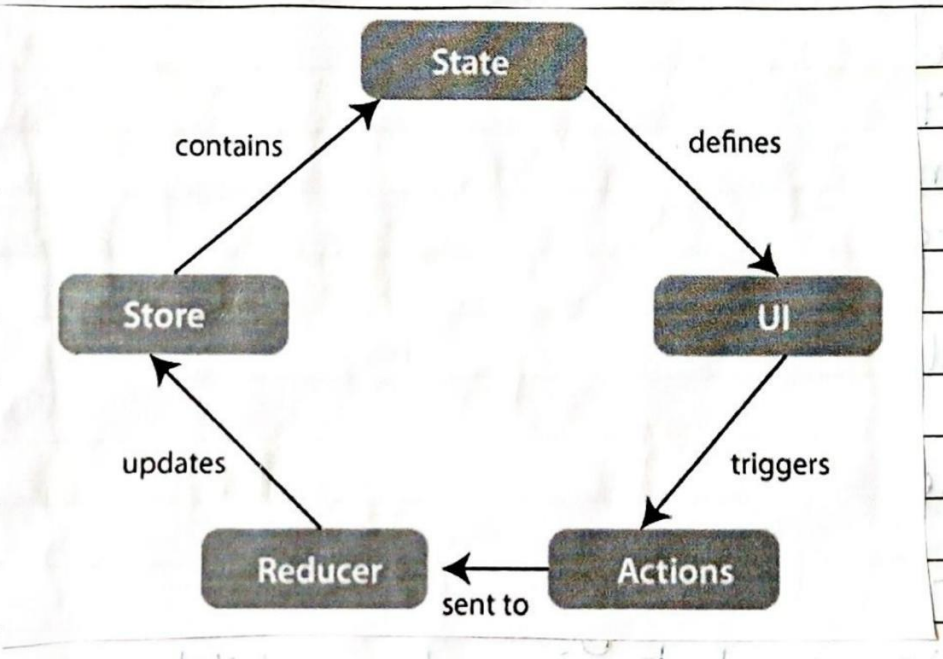
Redux was inspired by flux. Redux studied the Flux architecture.

① Redux does not have dispatcher concept.

② Redux has an only store whereas Flux has many stores.

③ The Action objects will be received and handled directly by Store.

React Architecture :-



Store :- A store is a place where the entire state of your application lists. It manages the status of the application and has a dispatch function. It is like a brain responsible for all moving parts in Redux.

Action:- Action is sent or dispatched from the view has which are payloads that can be read by Reducers. It is pure object created

to the store the information of users event. It includes information such as type of action, time of occurances, location of occurances, its coordinates and which state it aims to change.

Reducer:- Reducer reads the payload from the action and then updates the store via the state accordingly. It is pure function to return a new state from the initial state.

React Portals:-

It is very tricky to render the child component outside of its parent component hierarchy. It breaks the convention where a component needs to render as a new element and follow a parent-child hierarc

```
ReactDOM.createPortal(child, container)
```

Example using Portal:-
We want to insert a child component in different location in the DOM

```
render() {
  return ReactDOM.createPortal(
    this.props.children,
    myNode,
  );
}
```

Features :-

- It uses React version 16 and its official API for creating portal
- It has a fallback for React version 15.
- It transports its children component into a new React portal which is appended by default to document.body.
- It can also target user specified DOM element.
- It supports server-side rendering
- It supports returning arrays.
- It doesn't produce any DOM mess.
- It does no dependancies, minimalistics.

When to use?

1. Modals.
2. Tooltips
3. Floating menus.
4. Widgets

Explanation of React Portal.

App.Js.

```
import React, {Component} from 'react';
import './App.css'
import PortalDemo from './PortalDemo.Js';
Class App extends Component {
```

```
    return (
    <div className = "App">
      <PortalDemo />
    </div>
      );
    }
  }
export default App;
```

PortalDemo.Js

```
import React from 'react'
import ReactDOM from 'react-dom'
function PortalDemo(){
   return ReactDOM.createPortal(
    <h1> Portals Demo </h1>
   document.getElementById('portal-root')
  )
}
export default PortalDemo
```

index.html.

```
<! DOCTYPE html >
 <html lang = "en">
  <head>
  <meta charset = "utf-8" />
  <link rel = "shortcut icon" href =
                "%.PUBLIC-URL %.favicon.ico"
< meta name = "viewport" content =
        "width = device-width, initial-scale=1
<meta name = "theme-color" content =
                "#000000" />
```

```
<title> React App </title>
</head>
<body>
<noscript> It required to enable Javascript
    to run this app </noscript>
<div id = "root "> </div>
</body>
</html>
```