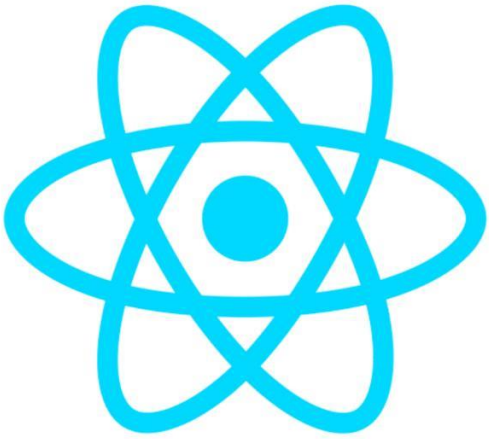
Semester: V **Subject:** Web Computing **Academic Year:** 2023 – 2024



React.Js

INDEX

14 React Component API

15 React Component Life Cycle

16 React Forms

17 Controlled Vs UnControlled

18 Conditional Rendering

19 React Lists

20 React Keys

21 React Refs

22 React Fragments

23 React Router

24 React CSS

25 React Animation

26 React Map

27 React Table

28 Higher Order Component

29 React Code Spliting

30 React Context

31 React Hooks

32 React Flux vs MVC

33 React Redux

34 React Portal

React.Js

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. It was initially developed and maintained by Facebook

Why we use ReactJs :-

The main objective of ReactJs is to develop User Interface (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. We can use ReactJs on the client and server-side as well as with other frameworks. It uses component and data patterns that improves readability and maintain larger apps.

React create-react-app

The create-react-app is an excellent tool for beginners, which allows you to create and run React project very quickly. It does not take any configuration manually. This tool is wrapping all of the required dependencies like Webpack, Babel for React project itself and then you need to focus on writing code.

Requirement :-

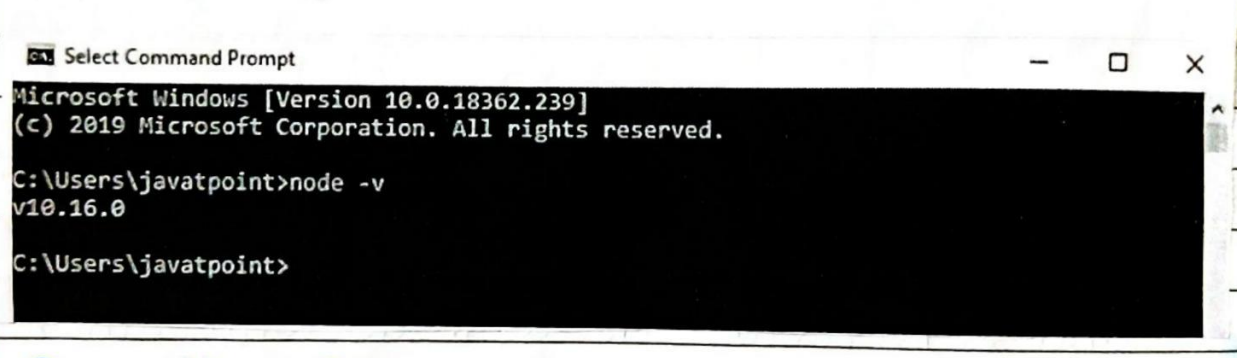
The create React App is maintained by Facebook and can works on any platform for ex- macOS, Windows, Linux.

To create a React project using create react-app. you need to have installed the following things.

1) Node version >= 8.10
2) NPM version >= 5.6

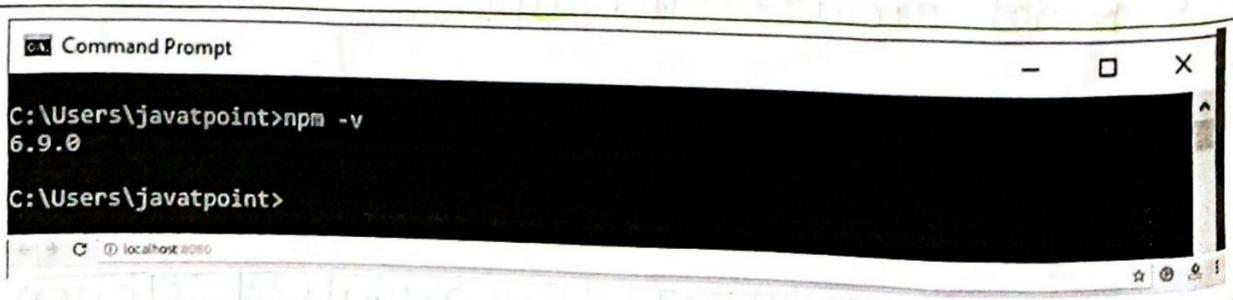
- Run the following code to check the Node version in command prompt.

```
$ node -v
```



- Run the following command to check the NPM version in command prompt.

```
$ npm -v
```



In react application, there are several files and folders in the root directory, some of them are as follows.

1] node-module :- It contains the React library and any other third party libraries needed

2] public :- It holds the public assets of the application. It contains the index.html where React will mount the application by default on the   element.

3] Src :- It contains the App.css, App.js, index.js and serviceWorker.js, Here the App.js file always responsible for displaying the output screen react.

4] package-lock.json :- It is generated automatically for any operations where npm package modifiers either the node-module tree or package.json. It cannot be published. It will be ignored if it finds any other place rather than the top-level package.

5] 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 projects.

6] Readme.md :- It provides the documentation to read about React topics

React Features :-

• JSX :-

JSX stands for Javascript XML. It is a java-Script syntax extension. Its an XML or HTML like syntax used by ReactJs. This syntax is proceed into javascript calls of React Framework. It extends the ES6 so that HTML like text can co-exit with Javascript react code. It is not necessary to use JSX, but it is recommended to use in ReactJs.

• Component :-

ReactJs is all about components, ReactJs application is made up of multiple compone and each component has its own logic and control These components can be reusable which helps you to maintain the code when working on larger scale proje

• One-way Data Binding :-

ReactJs is designed in such manner that follows unidirectional data flow or one-way data binding. The benifits of one-way data binding give you better co throught application. If the data flow is in another direction then it require a additional feature. It is because compon are supposed to be immutable and the data within them cannot be changed.

- **Virtual Dom :-**

A virtual Dom object is a representation of the original Dom object. It works like a one-way data binding. Whenever any modifications happen in the web application. The entire UI is re-rendered in virtual Dom representation and new Dom. Ones it has done, the real Dom will update only the things that have actually changed.

- **Simplicity :-**

ReactJs uses JSx file which makes the application simple and to code as well as understand. We know that ReactJs is component based approch which makes the code reusable as your need. This makes it simple to use and learn.

- **Performance :-**

ReactJs is known to be a great performer. This feature makes it much better than other framework out there today. The reason behind this is that it manages virtual Dom The Dom is a cross platform and programming API which deals with HTML, XML & XHTML. The Dom exits entirely in memory Due to this when we create a component. we did not write directly to Dom.

Pros and Cons ReactJs :-

Advantage of ReactJs :-

① Easy to learn and use :-

→ ReactJs is mush easier to learn and use It comes with a good supply of documentat tutorial and training resource.

② Creating Dynamic Web-Application Becomes Easier :-

To create a dynamic web application specifically with HTML Strings was tricy because it requires a complex coading, but ReactJs solved that issue and makes it easier. It provide less coding and gives more functionality. It makes use of JSX, which is particular syntax leting HTML Quotes.

③ Reusable Component :-

A ReactJs web application is made up a multiple components. and each components has its own logic and controls. These components are responsible for outputing small and reusable code helps to make your apps easier to develop and maintain

④ Performance Enhancement :-

ReactJs improves performance due to virtu Dom. The Dom is a cross platform and programming API which deals with HTML XML or XHTML. Most of the developers faced the problem when the Dom was

updated. Which slowed down the performance of the application. ReactJs solved this problem by introducing virtual Dom.

⑤ The Support of Handy Tools :-

ReactJs has also gained popularity due to the presence of handy set of tools. These tools make the task of developer understandable and easier. The React Developer Tool have been designed as Chrome and Firebox. dev extension and allow you to inspect the React component hierarchies in the virtual Dom.

⑥ Known to be SEO Friendly :-

Traditional Javascript framework have an issue in dealing with SEO. The search engins generally having trouble in reading JS application. Many web developers have often complained about this problem. ReactJs overcomes this problem that helps developers to be easily navigaated on various search engins.

⑦ The Benifits of having Javascript Library :-

ReactJs is choosing by most of the web developers. It is because it is offering very rich Javascript library. The javascript library provides more flexibility to the web developers to choose the way they want.

Disadvantage of ReactJS

① The high pace of development :-
The high pace of development has an advanta and disadvantage both. In case of disadvantage since the Envirnment continuesly change so fast, some of the developers not filling comfertable to relearn the new ways of doing things regullarly.

② Poor documentation :-
It is another cons which are common for constanly updating technology. React technolog and accelerating so fast that there is no documentation properly.

③ View Part :-
React Js cover only the UI layers of the app nothing else, so you still need to choos some other technologies to get complete tooling set for development in project

④ JSX is barrier :-
React Js uses JSX. Its syntax extension that allows HTML with Javascript mixed together. This approch has its own benefits, but some members of the developm ent community consider JSX as a barrier especially for new developers. Developers complain about its complexity in the learning curve.

ReactJs vs AngularJs :-

	AngularJs	ReactJs
Author	Google	Facebook community
Developer	Misko Hevery	Jordan Walke
Initial Release	october 2010	March 2013
Latest Version	Angular 1.7.8 on 11 March 2019	React 16.8.6 on 27 March 2019
Language	Javascript, HTML	JSX
Type	Open source MVC Framework	Open Source JS Framework.
Rendering	Client-Side	Server-Side
Packaging	Weak	Strong
Data-Binding	Bi-directional	Unidirectional
Testing	Unit and Integration Testing	Unit Testing
App Architecture	MVC	Flux

Dependencies	It manages dependencies automatically	It require additional tools to manage dependencies
Routing	It requires a template or controller to its router configuration, which has to be manged manually.	It doesn't handle routing but has a lot of module for routing.
Performance	Slow	Fast, due to virtual Dom.
Best For	It is best for single page application that update a single view at a time	It is best for single page applications that update multiple views at a time

React Js Vs React Native (Difference)

	React Js	React Native
①	The ReactJs initial release was in 2013	The react Native initial release was in 2015
②	It is used for develop	It is used for developing

	web application	mobile applications
③	It can be executed on all platform	It is not platform independant. It takes more effort to be executed on all platform
④	It uses a Javascript library and CSS for animations	It comes with built-in animation libraries.
⑤	It uses React-router for navigating web pages	It has built-in animation libraries.
⑥	It uses HTML tags	It does not use HTML tags.
⑦	It can HTML tags	It does not use HTML tags
⑧	It can use code components, which saves a lot of valuable time	It can reuse React Native UI component & modules which allow hybrid apps to render natively.
⑨	It provides high security	It provides low security in comparison to ReactJs

React Vs Vue

	React	Vue.
Defination	React is declarative efficient, flexible open-source JS library for building reusable UI Component	Vue is an open-source JS library for building reusable user interfaces and single page application
History	It was created by Jordan Walke, a software engineer at Facebook, It was initially developed and maintained by Facebook and later used in its products like Whatsapp & Insta.	Vue was created by Even You, a former employee of Google worked on many Angular project. He wanted to better version of Angular, just extracting the part which he liked about Angular and making it lighter.
Prefered language	Javascript / XML	HTML / Javascript
Size	The size of the React library is 100 kilobytes	The size of the Vue library is 60 kb.

Performance	The performance is slow as compared to Vue	Its performance is fast as a compared to React.
Flexibility	React provides great flexibility to support third party library	Vue provides limited flexibility as compared to React.
Current Version	React 16.8.6 on March 27, 2019	Vue 2.6.10 on March 20, 2019.
Long Term Support	It is suitable for long term support.	It is not suitable for long term support.

React JSX.

JSX provides you to write HTML/XML like structure in the same file where you write javascript code, then preprocessor will transform these expression into actual Javascript code. Just like XML/HTML, JSX tags name, attributes, and childdren.

Why use JSX?

- It is faster then regular javascript because it performs optimization while translating the code to javascript.
- Insted of separating technologies by putting markup and logic in separate files, React

uses component that contain both. We will learn component

- It is type-safe, and most of the errors can be found at compilation time.
- It makes easier to create template.

Nested Elements in JSX:-

App.Jsx

```
import React {component} from 'react';
class App extends Component {
  render() {
    return (
      <div>
        <h1> Javatpoint </h1>
        <h2> Training Institute </h2>
        <p> This website contains
        the best CS tutorials </p>
        </div>
    );
  }
}
export default App;
```

JSX Attribute :-

JSX use attributes with the HTML elements same as regular HTML. JSX uses convention for attribute rather than standard naming convention of HTML

Such as a class in HTML becomes class-Name in JSX because the class is the reversed keyword in javascript. We can also user our own custom attribute, we need to use data-prefix.

Example:-

```
import React,{Component} from 'react';
class App extends Component {
  render() {
      return (
        <div>
          <h1> JavaTPoint </h1>
          <h2> Training Institute </h2>
         <p> data-demoAttribute = "demo">
    This website contains the best CS</p>
    </div>
    );
  }
}
export default App;
```

In JSX, We can specify attribute values in 2 ways;

① As String Literal:-

We can specify the values of attributes in double quotes.

```
var element = <h2 className = "firstAttribute">
  Hellow JavaTpoint </h2>
```

Example :-

```
import React, { Component } from 'react';
class App extends Component {
  render () {
    return (
      <div>
      <h1 className = "hello" > JavaTpoint</
      <p data-demoAttribute = "demo" >
      This website contains the best cs
       tutorials </p>
    </div>
  );
  }
}
Export default App;
```

2] As Expressions :-

We can specify the values of attributes as expression using curly braces {}

Example.

```
import React, { Component } from 'react';
 class App extends Component {
    render () {
      return (
        <div>
          <h1 class Name = "hello" > (25+20)
                                    </h1>
      </div>
    );
    }
}
Export default App;
```

JSx Comments

JSx allows us to comments that begins with /* and ends with */ and wrapping them in curly braces {} just like in the case of JSX expressions

Example

```
import React, {Component} from 'react';
class App extends Component {
    render (){
        return (
            <div>
        <h1 className="hello"> Hello JavaPoint </h
        </div>
    );
    }
}
export default App;
```

JSx Styling.

React allways recommended to use inline style To set inline styles, you need to use camel-case syntax. React automatically allows appending px after the number value on specific elements

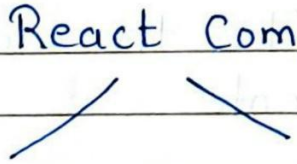
Example :-

```
import React, { Component } from 'react';
class App extends Component {
 render (){
   Var myStyle ={
     fontSize : 80
```

```
font family: 'Courier',
 Color : '#003300'
}
return (
    <div>
      <h1 style = {mystyle}>
            www.javatpoint.com </h1>
      </div>
  );
 }
}
export default App;
```

Output:-

React Components



Functional Component

Class Component

Functional Component:-

In React, function component are a way to write component that only cont

a render method and don't have their own state. They are simply JavaScript function that may or may not receive data as a parameter. We can create function that takes props as input and returns what should we rendered.

```
function WelcomeMessage(props) {
  return <h1> Welcome to the ; {Props.name}
                                </h1>
  }
```

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

Example.

```
import React, {Component} from 'react';
class App extends React.Component {
   render () {
      return(
        <div>
          <First/>
            <second/>
             </div>
      );
    }
  }
class First extends React.Component {
    render () {
       return (
      <div>
        <h1> JavaTPoint </h1>
      </div>
```

```
    };
}
class Second extends React.Component{
    render(
        <div>
        <h2> www.JavaTpoint </h2>
        <p> This website contains the
            great cs tuto </p>
        </div>
    );
  }
 }
}
export default App;
```

Class Components.

Class components are more complex than functional components. It requires you to extend from React.Component and create a render function which returns a React element. You can pass data from one class to other class component.

Example:-

```
import React {Component} from 'react';
class App extends React.Component {
constructor() {
```

```
Super ();
this.state ={
  data:
    [  {
          "name":"Abhishek"
       }
       {  "name" : "Saharsh"
       }
       {
          "name": "Ajay"
       }
    ]
  }
} render(){
    return (
        <div>
<h1> Student Name Detail </h1>
</div>
 );
} }
Class List extends React.Component {
 render (){
  return (
     <ul>
       <li> {this.props.data.name} </li>
     </ul>
  };
} }
export default App;
```

React State :-

The state is an updated structure that is use to contain data or information about the component. The state in component can change over time. The state in change over time happen as a response to user action or system event.

A state must be kept as simple as possible It can be state by using the setState() method and calling setState() method trigger UI Updates. A state represents the component's local state or information. To state an initial state before any interaction occure we need to use getInitialstate.. method.

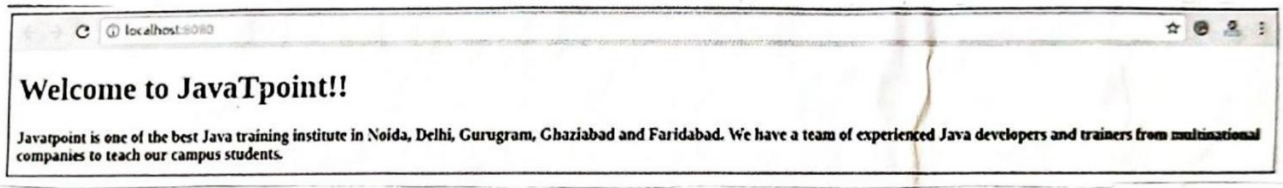
Defining State:-

To define a state, you have to first declare a default set of values for the defining the components initial state.

Example:-

```
import React,{componet} from 'react';
Class App extends React Component {
  Constructor (){
    Super ();
    this.state = {displayBio:true};
  }
  render (){
    const bio = this.state.displayBio!(
    <div>
<h3> Javatpoint is one of the best traini
```

```
institute In Noida ,Delhi, Gurugram, Ghaziabad
</h3> </p>
    </div>
    ): null;
     return (
    <div>
       <h1> Welcome to javaTpoint </h1>
       {bio}
    </div>
    );
  } }
export default App ;
```



Changing the state:-

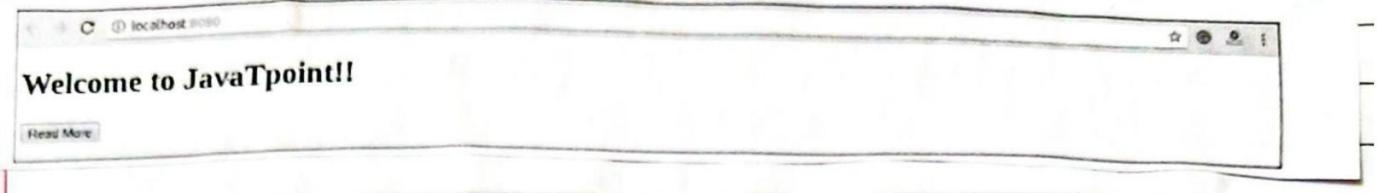
We can change the component state by using setState() method and passing a new state object as the argument Create a new method toggle-DisplayBio() in the above example.

```
this.toggleDisplay Bio = this.toggleDisplayBio.bind
                                        (this);
```

```
import React {Component } from 'react';
class App extends React.Component {
  Constructor (){
  Super ();
  this.state ={displayBio : false };
```

```
console.log('component this', this);
this.toggleDisplayBio = this.toggleDisplayBio
                        .bind(this);
}
this.toggleDisplayBio() {
    this.setState({displayBio: !this.state
                   .displayBio});
}
render(){
    return (
      <div>
<h1> Welcome to Javatpoint </h1>
{
    this.state.displayBio? (
      <div>
<h4> Javatpoint is one of the best Java
institute in Noida</h4> </p>
<button onClick={this.toggleDispla
    Bio}> Show Less </button>
</div>
): (
    <div>
<button onClick={this.toggleDisplayBio}>
    Read more </button>
</div>
) }
</div>
) } } }
export default App;
```

when you click on Read More



React Props :-

Props stand for Properties. They are read-only components. It is an object which store the value of attribute of a tag and work simillar to the HTML attribute.

Props are immutable so we cannot modify this props from inside the component.

Example :-

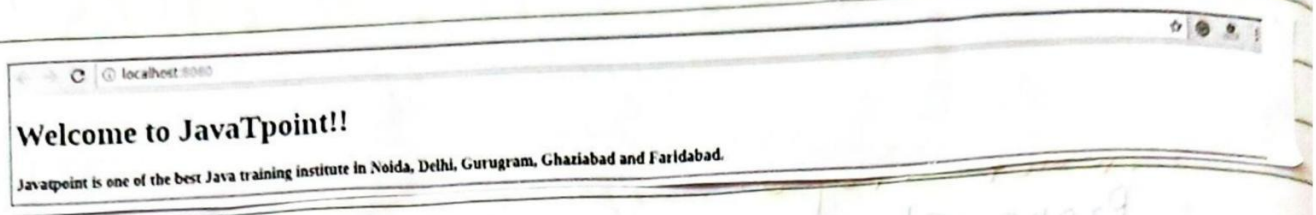
App.Js

```
import React ,{ componeht } from 'react';
class App extends React.Component {
  render ( ){
    return (
      <div>
        <h1> Welcome to { this.props.name } </h
      <P> <h4> Javatpoint is one of the best
        training institute in India </h4> </di
    );
  } }
export default App;
```

Main.Js

```
import React from 'react'
import ReactDom from 'react-dom';
import App from './App.Js';
```

```
ReactDom.render (<App name = 'JavaTpoint" />,
document.getElementById('app'));
```



Default Props :-

It is not necessary to always add props in the ReactDom.render() element. You can also set defaults props directly

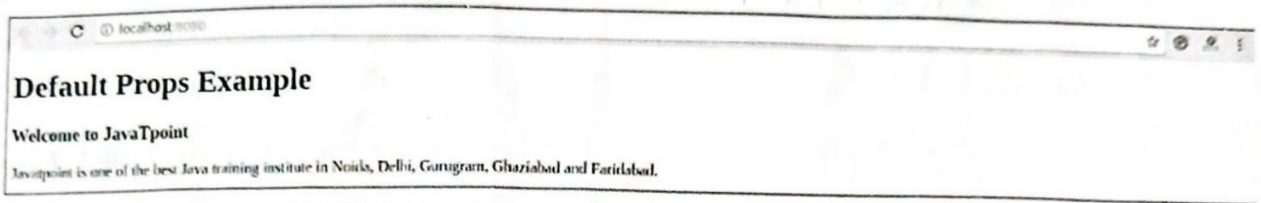
Example

App.Js

```
import React, {Component} from 'react';
 class App extends React.Component {
   render() {
      return (
       <div>
       <h1> default Props Example </h1>
       <h3> Welecome to {this.props.name </h
       <P> JavaTpoint is one of the best Java
       training </P>
       </div>
    ); } }
  App.defaultProps = {
      name : "JavaTpoint"
    }
  export default App;
```

Main.Js

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



State and Props.

It is possible to combine both state and props in your app. You can set the state in the parent Component and pass it in the child component using props.

Example.

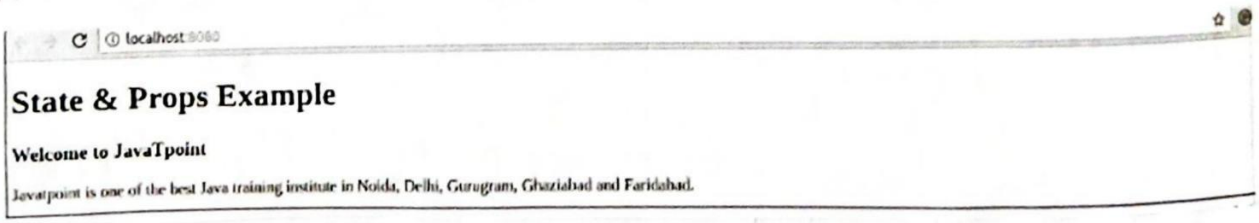
App.Js

```
import React {Component} from 'react';
class App extends React.Component {
  Constructor (props) {
    Super (props);
     this.state ={
        name :"JavaTpoint"
     }
    }
  render() {
    return {
  <div>
    <JTP jtpProps={this.state.name}/>
```

```
</div>
); } }
class JTP extends React.Component {
 render() {
  return (
   <div>
   <h1> State & Prope example </h1>
   <h3> Welcome to {this.props.jtpProp} </h3>
    <p> JavaTpoint is one of the best Java
      training institute in Noida </p>
  </div>
  );  }}
export default App;
```

Main.Js

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



React Props Validation:-

Props are an important mechanism for passing the read-only attribute to React component. The props are usually requi to use correctly in the component.

Validating Props :-

App.propsType is used for props validation in react component. When some of the props are passed with an invalid type, you will get the warnings on Javascript console. After specifying the validation patterns.

Syntax:-

```
class App extends React.Component {
    render() {}
}
Component.propTypes = {/*    */}
```

ReactJs Props Validator

	Props Type	Description.
①	PropType.any	The props can be of any data type.
②	PropTypes.array	The props should be an array.
③	PropTypes.bool	The props should be a boolean.
④	PropTypes.func	The props should be a fun
⑤	PropTypes.number	The props should be a number
⑥	PropFypes.Object.	The props should be a object

⑦	PropTypes.string	The props should be String
⑧	PropTypes.Symbol	The props should be Symbol
⑨	PropTypes.instanceOf	The props should be an instance of particular Javascript class.
⑩	PropTypes.isRequired	The props must be provided
⑪	PropTypes.element	The props must be element
⑫	PropTypes.node	The props can render anything : number, string, element or an array containging these types.
⑬	PropTypes.oneOf()	The props should be one of several types of specific values.
⑭	PoopTypes.oneOfType ([PropTypes.string, Prop.Types.number])	The props should be an object that could be one of many type

- **React Js Custom Validators.**

React Js allows creating a custom validation function to perform custom validation. The following arguments is used to create a custom validation function.

props :- It should be first argument in the Component.

propName :- It is the propName that going to validate.

ComponentName :- It is the componentName that are going to validate again.

Example :-

```
var Component = React.createClass({
App.propTypes = {
 customProp : function (props, propName, compo-
  nentName) {
 if(!item.isValid (props [propName])) {
   return new Error ('Validation failed');
  } } } } )
```

Difference between State and Props.

Props	State.
Props are read-only	State changes can be asynchronous.
Props are immutable	State is mutable
Props allow you to pass data from one component to other component as an argument.	State holds information about the components
Props can be accessed	State cannot be accessed

by the child component	by ...
Props are used to communicate between component	State can be used fo rendering dynamic ch- anges with the compone
Stateless component can have Props	Stateless component cannot have State.
Props make component reusable	State cannot make components reusable
Props are external and controlled by whatever render the component	The State is internal and controlled by the React Component itself.

SN	State and Props
①	Both are plain JS object
②	Both can contain default values.
③	Both are read-only when they are using by this.

What is Constructor:-

The constructor is a method used to initialize an object State in class. It automatically called during the creation of an object in

class.

```
Constructor (props) {
    super (props);
  }
```

In react, constructor are mainly used.

① It is used for initializing the local state of the component by assigning an object in this.state.

② It is used for binding event handler methods that occure in your component.

Example :-

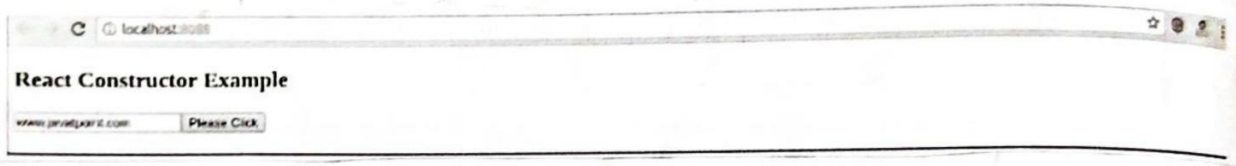
App.JS

```
import React, { Component } from 'react';
class App extends Component {
  constructor (props)
   Super(props);
   this.state = {
     data : " www.javatpoint.com")
      }
   this.handleEvent = this.handleEvent.bind (this);
  } handleEvent () {
      Console.log (this.props);
  } render() {
  return (
    <div className = "App" >
    <h2> React Constructor </h2>
<input type = "text" value = {this.state.data}
/>
<button onClick = {this.handleEvent} >
        Please Click </button>
```

```
</div>
); }}
 export default App;
```

Main-Js

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



Arrow Function:-

The arrow function is the new feature ES6 standard. If you need to use arrow functions. It is not necessary to bind any event to 'this'.

```
import React, {Component} from 'react';
class App extends Component {
  Constructor ( props ) {
      Super (props);
   this.state = {
      data : 'www.javatpoint.com'
   } }
  handleEvent = ( ) => {
   console.log (this.props );
  }
```

```
render (){
 return (
   <div className='App'>
 <h2> React Constructor </h2>
  <input type = "text" value={this.state.data}/>
  <button onClick={this.handleEvent} >
              Please Click </button>
   </div>
 ); }
 export default App;
```

React Component API.

React Js Component is a top-level API. It makes the code completly individual and reusable in the application. It includes various method

- creating element.
- Transforming element.
- Fragments

① setState ()

This method is used to update state of the component. This method does not always replace the state immediatly. It only adds change to the original state. It is primary method that is used to update the user interface (UI) in response (UI) in event handler and server response.

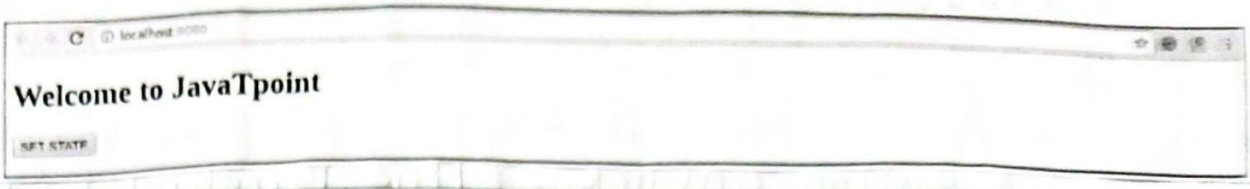
```
this.state.State (object newState [, function
   callback ]);
```

```
import React, {Component} from 'react';
import PropTypes from 'prop-types';
Class App extends React.Component {
  Constructor() {
   Super();
  this.state ={
    msg : "Welcome to JavaTpoint"
  };
  this.updateSetState = this.updateSetState
          .bind(this);
}
updateSetState(){
    this.setState({
     msg."Its best tutorial".
    }); }
render() {
    return (
      <div>
   <h1> {this.state.msg} </h1>
  <button onClick={this.updateState}>
    SetSTATE </button>
</div>
); }}
export default App;
```

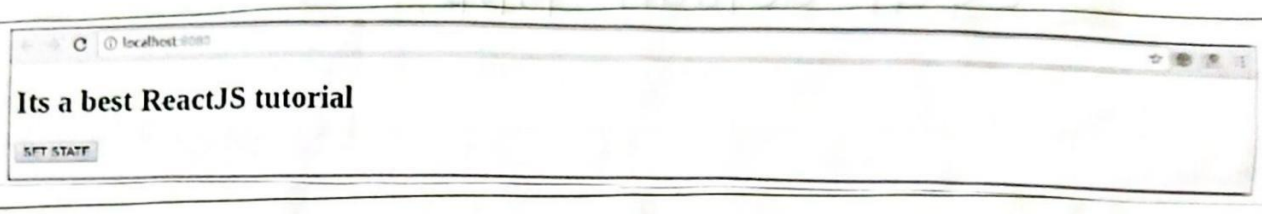
Main.Js

```
import React from 'react';
Import ReactDOM 'react-dom';
Import App from '/App.Js';

ReactDOM.render(<App/>, document.getElem
ntById('app'));
```



When you click on SET STATE button, you will see following screen with updated message.



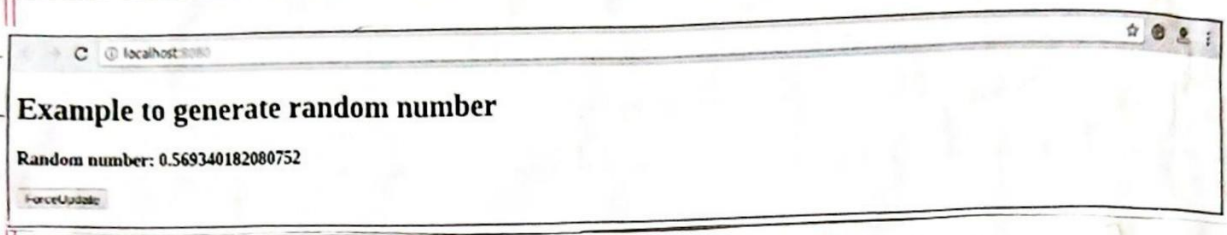
forceUpdate()
This method allows us to update the component manually.

```
Component.forceUpdate(callback);
```

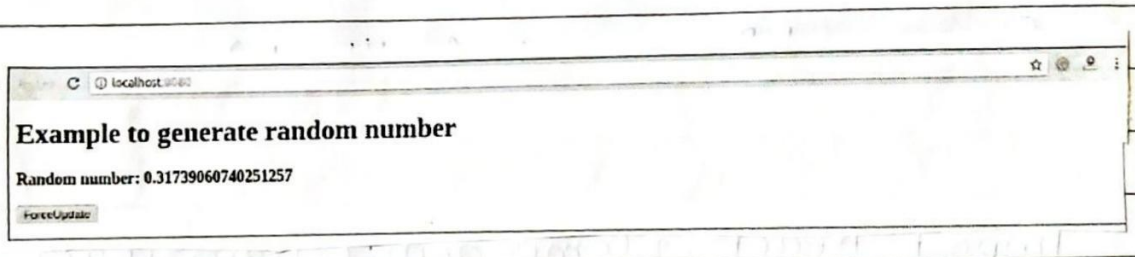
Example. App.Js.

```
import React, {Component} from 'react';
class App extends React.Component {
  constructor(){
    super();
    this.forceUpdateState = this.forceUpdateState.
    bind(this);
  }
  forceUpdateState(){
      this.forceUpdate();
  };
  render() {
    return (
```

```
<div>
<h1> Example to generate random Number</
 <h3> Random number; {Math.random()} </h
 <button onClick = {this.forceUpdateState} >
        ForceUpdate </button>
 </div>
); } }
 export default App;
```



Each time when you click on ForceUpdate button.



findDOMNode()

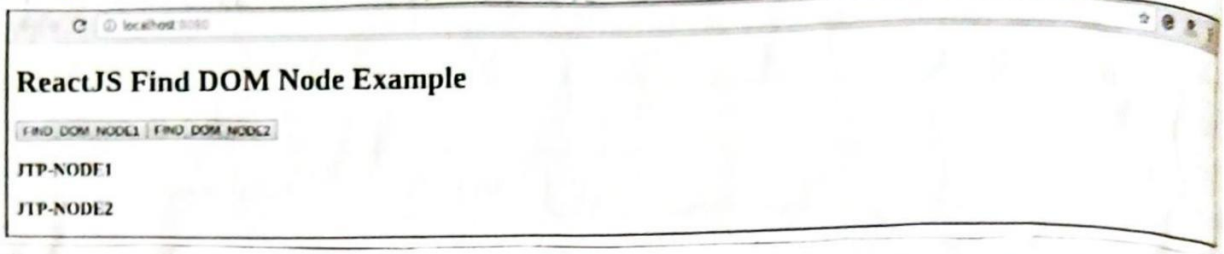
For Dom manipulation, you need to use ReactDom.findDOMNode() method. This me allows us to find or access the underl DOM node.

```
ReactDOM.findDOMNode(component);
```

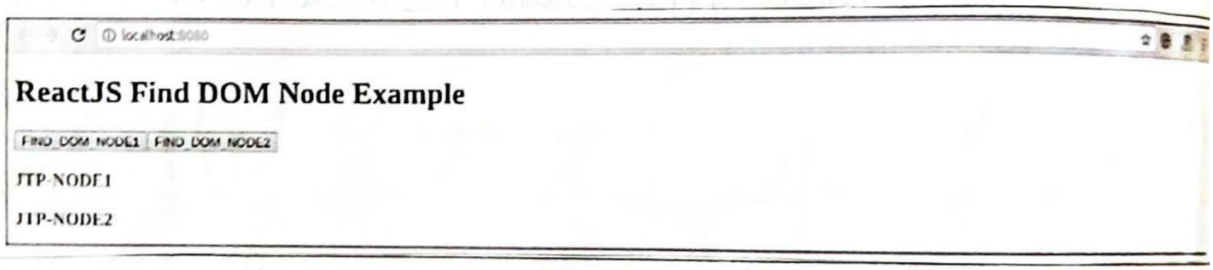
App.js

```
import React {Component} from 'react';
import ReactDom from 'react-dom';
class App extends React.Component{
 constructor (){
  Super ();
this.findDOMNodeHndler1 = this.findDomNode-
      Handler1.bind (this);
this.findDOMNodeHandler2 = this.findDOMNode-
    Handler2.bind (this);
};
findDomNodeHandler1() {
   var myDiv =document.getElementById ("
       myDivOne');
  ReactDom.findDOMNode (myDivOne).style.
    color='red';
 }
findDOMNodeHandler2(){
   var myDiv =document.getElementById
              ('myDivTwo');
   ReactDom.findDOMNode (myDivTwo).style
     .color ='blue';
 }
render (){
    return (
   <div>
  <h1> ReactJs Find DOM Node Example </hi
 <button onClick ={this.findDOMNodeHandler1}
    FIND-DOM-NODE 1 </button>
 <button onClick ={this.findDomNodeHandler2}
    FIND-Dom-NODE 2 </button>
    <h3 id = "myDivOne"> JTP-Node1 </h3>
```

```
<button onClick = {this.forceUpdate.bind(this)}>
    ForceUpdate </button>
</div>
);
} }
export default App;
```



Ones you click on the button, the color of the node gets changed.



React Component Life-Cycle.

In ReactJs, every component creation process involves various lifecycle methods. These lifecycle methods are termed as component lifecycle. These lifecycle methods are not very complicated and called at various du a componants life.

Initial Phase → Mounting phase → Updating phase → Unmounting phase.

① Initial Phase -

• getDefaultProps ()

It is used to specify the default value of this.props. It is invoked before the creation of the component or any props from the parent is passed into it.

• getInitial Phase()

It is used to specify the default value of this.state . It is invoked before the creation of the component.

② Mounting Phase

• componentWillMount ()

This is invoked immediately before a component gets renderded into the Dom. In the case when you call setState () inside this method

• ComponentDidMount ()

This is invoked immediatly after component gets rendered and placed on the Dom. You can do any Dom querying operations.

• render()

This method is defined in each and every component. It is responsibble returning a single root HTML node element.

③ Updating Phase.

• ComponentWillRecieveProps ()

It is invoked when a component receives new props. If you want to update the

state in response to prop changes, you should compare this.Props and nextprops to perform state transition by using this.setState()

- **ShouldComponentUpdate()**
  It is invoked when a component decides updating occures.

- **Component Will Update()**
It is invoked just before the component updating occurs. Here you can't change the component state by invoking this.setState() method.

- **Component Did Update()**
It is invoked immediatly after the component updating occurs. In this method, you can put any course inside this.

4] Unmounting Phase

Component Will Unmount()
This method is invoked immediatly before a component is destroyed permanantly. It performs necessary cleanup.

React Forms :-

Form are an internal part of any modern web application. It allows the users to interact with the applications as well as gather information from the user. Forms can perform many tasks that depends on the nature of your buisness requirements and

Creating Form :-

React offers a stateful, reactive approch to build a form. The component rather than Dom usually handles the React form.

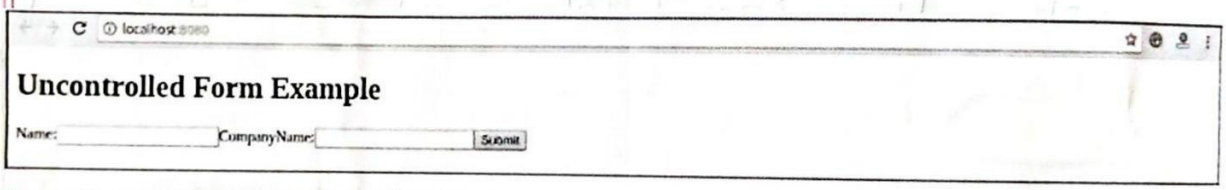
1. Uncontrolled component.
2. Controlled component.

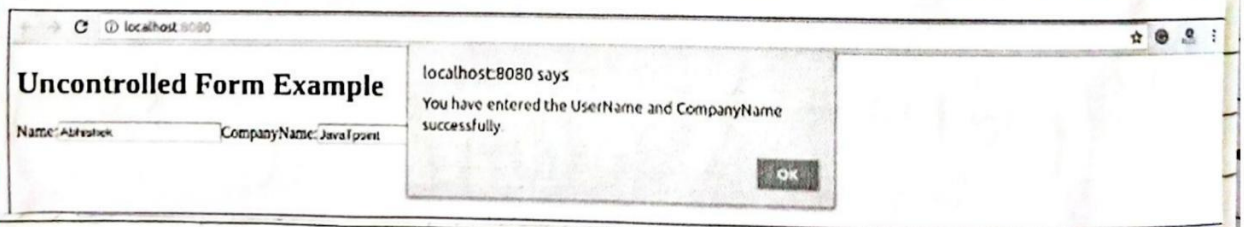
① Uncontrolled Component :-

The uncontrolled component input is similar to the traditional HTML form input. The Dom itself handles the form data. The HTML element maintain their own state that will be updated when the input values changes. To write an uncontrolled component.

```
impoit React, {Component} from 'react';
Class App extends React.Component{
    constructor (props)
        super(props);
    this.updateSubmit = this.UpdateSubmit.
                        bind (this);
    this.input = React.createRef ();
    }
UpdateSubmit (event) {
 alert ("You have entered the username and
  CompanyName successfully");
  event.preventDefault();
 }
render(){
```

```
return(
 <form onSubmit={this.updateSubmit}>
  <h1> Uncontrolled Form </h1>
  <label> Name:
    <input type = "text" ref = {this.input
  </label>
  <label>
  Company Name:
  <input type = "text" ref={this.input} />
  </label>
  </form>
 );
}}
export default App;
```





Controlled Component

Controlled component have functions that govern the data passing into them on every onChange event; rather than grabbing the data only ones.

```
import React {Component } from 'react';
class App extends React.Component {
    constructor (props) {
      super (props);
     this.state = {value :" "};
     this.handleChange = this.handleChange.bind
                                        (this);
    this.handleSubmit= this.handleSubmit.bind(this);
}
  handleChange(event) {
    this.setState({value: event.target.value})}
  }
  handleSubmit (event) {
    alert ("You have submitted the input Success-
        fully:' + this.state.value);
     event.preventDefault ();
  }
render ( ){
  return(
  <form onSubmit = { this.handleSubmit } >
   <h1> Controlled Form </h1>
  <label>
  Name:
<input type ="text" value = {this.state.value }
     onChange = {this.handlechange} />
   </label>
  <input type = "submit" value ="Submit "/>
  </form>
  );
  }
}
export Default App;
```

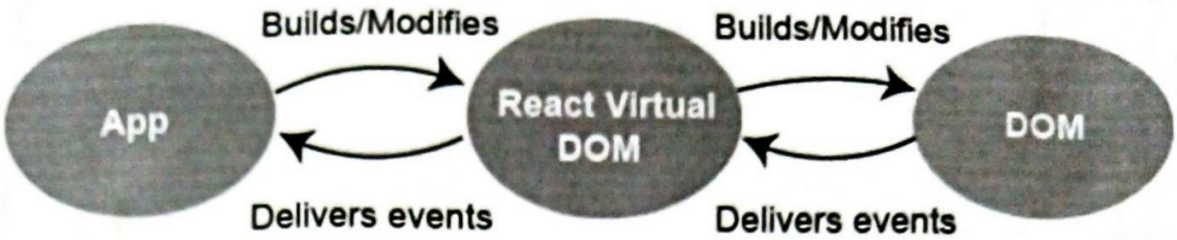
Controlled Componant Vs Uncontrolled Component.

Controlled	Uncontrolled.
It does not maintain internal state	It maintains internal state.
Data is controlled by parent component	Data is controlled by a Dom itself
It accept the current value as a prop	It uses a ref for their current values.
It allows validation control	It does not allow validation control.
It has better control over the form elements and data	It has limited control over the form and elements.

React 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 events is a a cross browser wrapper of the browser native event.

① React events are named as camelCase insted of lowercase.



② A function is passed as a event handler insted of a string.

For declaration in plain HTML :-

```
<button onclick = "showMessage()">
    Hellow JavaTpoint
</button>
```

Event declaration in React :-

```
<button onClick = {showMessage}>
    Hellow JavaTpoint
</button>
```

③ In react, we cannot return false to prevent the default behaviour we must call preventDefault event explicitly to prevent the

Example :-

```
import React, {Component} from 'react'
class App extends React Component{
  constructor (props)
    Super (props)
    this.state = {
        companyName = "
    };
```

```
};  }
changeText (event) {
    this.setState({
    companyName: event.target.value
    });
}
render (){
    return (){
        <div>
    <h2> Simple Event </h2>
    <label html For='name"> Enter Company
                Name </label>
<input type="text" id="companyName"
    onchange = {this.changeText bind (this
    <h4> You entered ; {this.state.company
    Name } </h4>
    </div>
);
}
}
export default App;
```