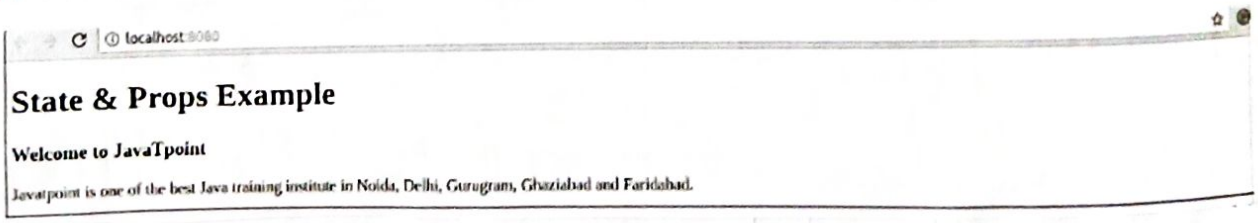
```
</div>
);  } }
class JTP extends React.Component {
 render () {
  return (
   <div>
   <h1> State & Prope example </h1>
   <h3> Welcome to {this.props.jtpProp} </h3>
   <p> JavaTpoint is one of the best Java
     training institute in Noida </p>
  </div>)
 );   }}
export default App;
```

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 child components
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.

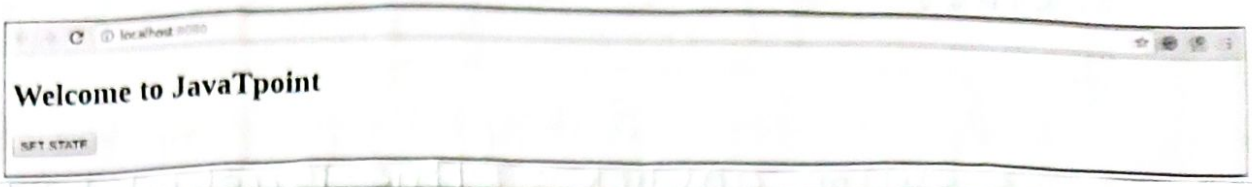
```
this.state.State (object newState [, function
   callback ]);
```

```
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;
```

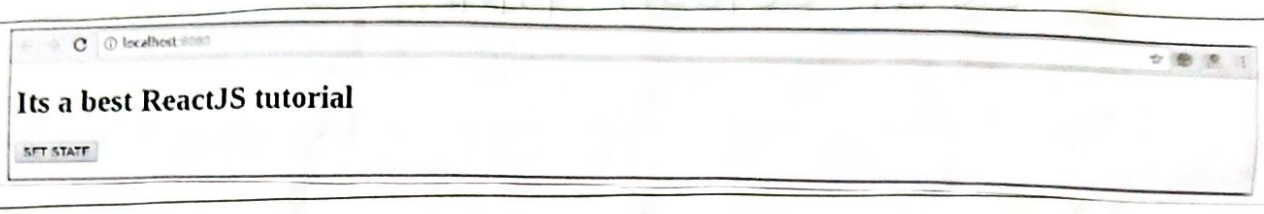
Main.Js

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

ReactDom.render(<App/>, document.getElem
nt By Id ('app'));
```



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



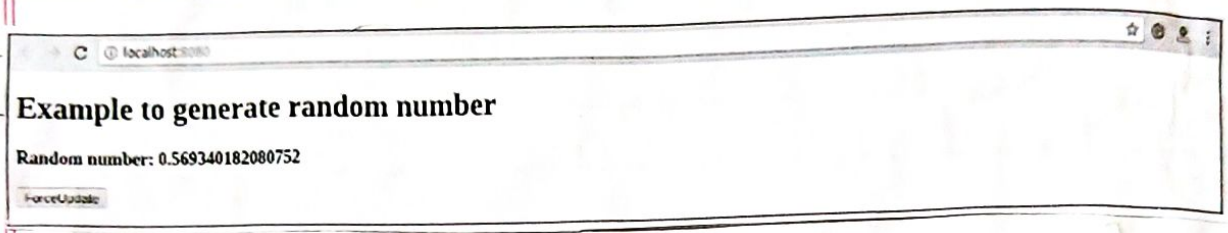
forceUpdate()
This method allows us to update the component manually.

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

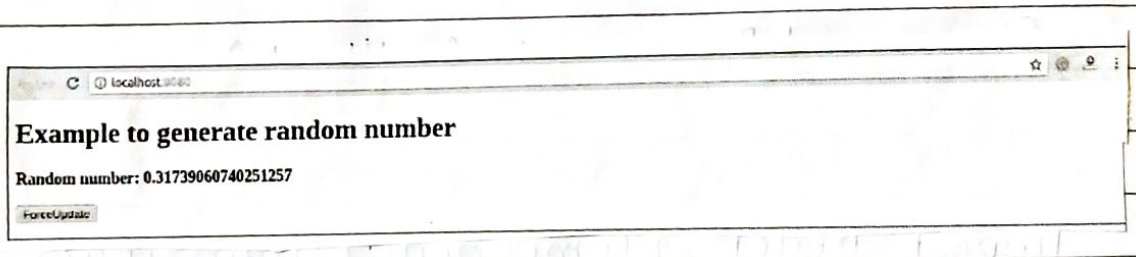
Example. App.Js.

```
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 Numbers</
 <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.

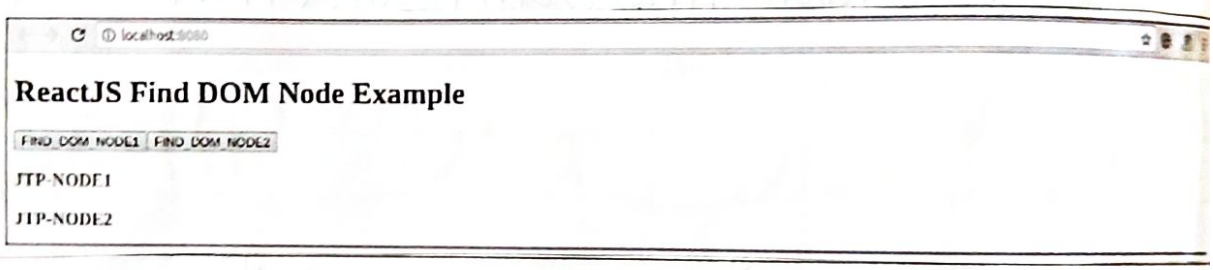
```
ReactDOM.findDOMNode(component);
```

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 </h1
 <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.forceUpdateState}
    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 perfo 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