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 chilldren.

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

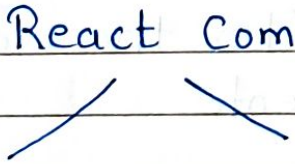
Example :-

```
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:-

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.

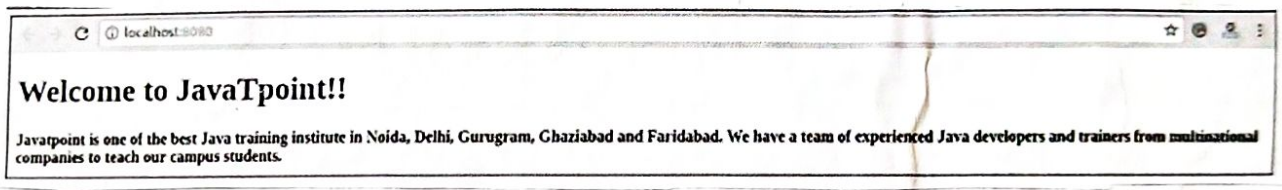
Defining State:-

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 = { display Bio : true };
  }
  render (){
    const bio = this.state.displayBio ? (
    <div>
<h3> Javatpoint is one of the best train
```

```
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:-

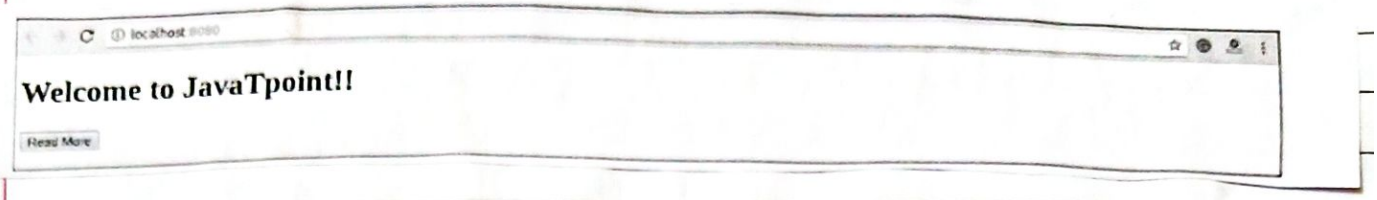
We can change the component state by using setState() method and passing a new state object as the argument Create a new method toggle-DisplayBio() in the above example.

```
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 valu 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 :-

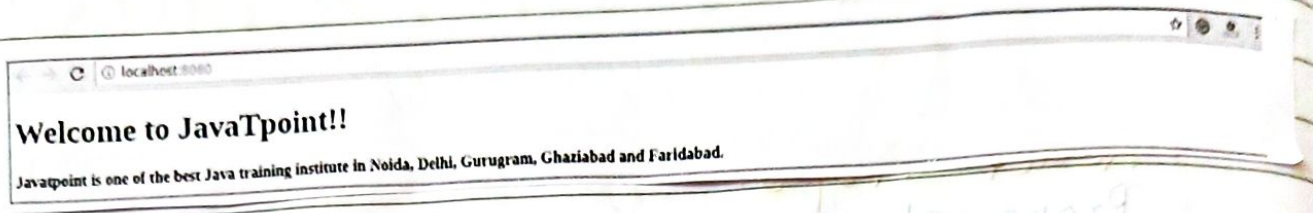
App.Js

```
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';
```

```
React Dom. render (<App name = 'Java Tpoint" />,
 document. get Element By Id ('app'));
```



Default Props :-

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

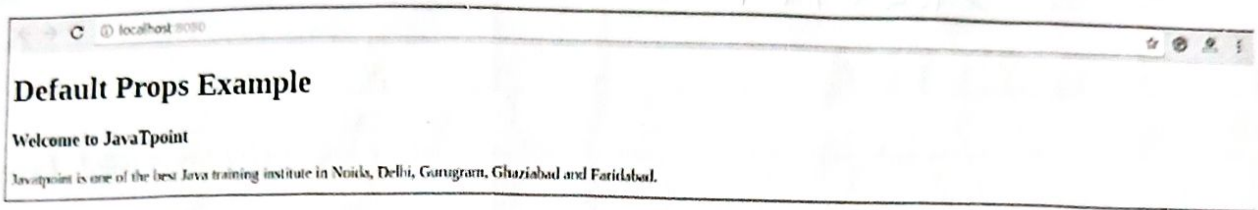
Example

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}/>
```