React Props

Props stand for "**Properties**." They are **read-only** components. It is an object which stores the value of attributes of a tag and work similar to the HTML attributes. It gives a way to pass data from one component to other components. It is similar to function arguments. Props are passed to the component in the same way as arguments passed in a function.

Props are **immutable** so we cannot modify the props from inside the component. Inside the components, we can add attributes called props. These attributes are available in the component as **this.props** and can be used to render dynamic data in our render method.

When you need immutable data in the component, you have to add props to **reactDom.render()** method in the **main.js** file of your ReactJS project and used it inside the component in which you need. It can be explained in the below example.

Example

App.js

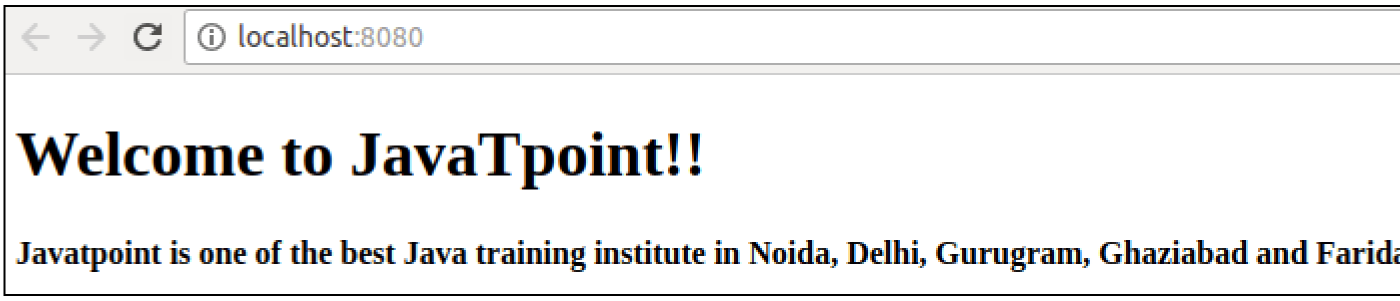
```
import React, { Component } from 'react';
class App extends React.Component {
   render() {
      return (
          <div>
            <h1> Welcome to { this.props.name } </h1>
            <p> <h4> Javatpoint is one of the best Java training institute in Noida, Delhi, Gurugr
am, Ghaziabad and Faridabad. </h4> </p>
          </div>
      );
   }
}
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'));
```

Output



Welcome to JavaTpoint!!

Javatpoint is one of the best Java training institute in Noida, Delhi, Gurugram, Ghaziabad and Farida

Default Props

It is not necessary to always add props in the reactDom.render() element. You can also set **default** props directly on the component constructor. It can be explained in the below example.

Example

App.js

```
import React, { Component } from 'react';
class App extends React.Component {
  render() {
    return (
      <div>
          <h1>Default Props Example</h1>
        <h3>Welcome to {this.props.name}</h3>
        <p>Javatpoint is one of the best Java training institute in Noida, Delhi, Gurugram, Ghaziabad and Faridabad.</p>
      </div>
    );
  }
}
App.defaultProps = {
  name: "JavaTpoint"
}
```

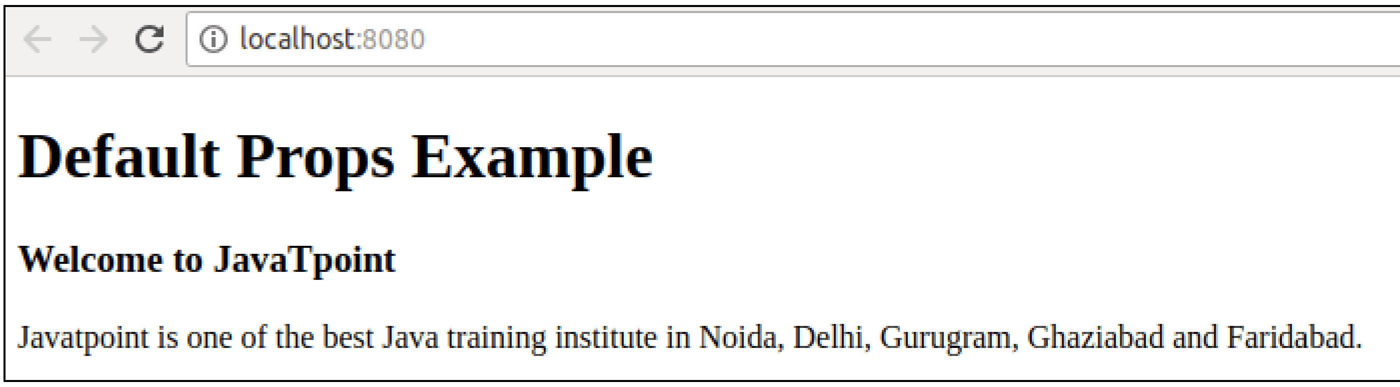
```
export default App;
```

Main.js

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

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

Output



Default Props Example

Welcome to JavaTpoint

Javatpoint is one of the best Java training institute in Noida, Delhi, Gurugram, Ghaziabad and Faridabad.

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. It can be shown in the below example.

Example

App.js

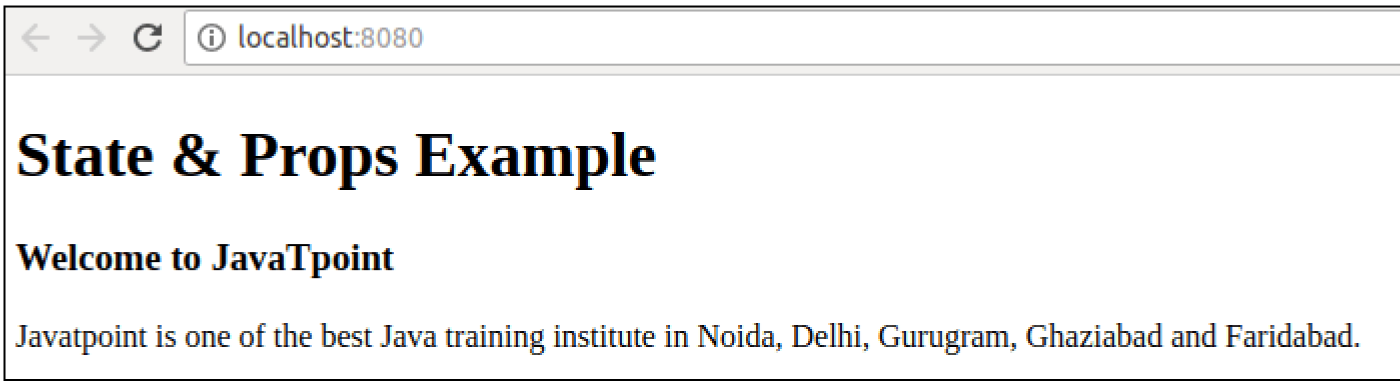
```
import React, { Component } from 'react';
class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      name: "JavaTpoint",
    }
  }
  render() {
    return (
      <div>
        <JTP jtpProp = {this.state.name}/>
      </div>
    );
  }
}
class JTP extends React.Component {
  render() {
    return (
       <div>
          <h1>State & Props Example</h1>
          <h3>Welcome to {this.props.jtpProp}</h3>
          <p>Javatpoint is one of the best Java training institute in Noida, Delhi, Gurugram, Ghaziabad and Faridabad.</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.getElementById('app'));
```

Output:



State & Props Example

Welcome to JavaTpoint

Javatpoint is one of the best Java training institute in Noida, Delhi, Gurugram, Ghaziabad and Faridabad.

React Props Validation

Props are an important mechanism for passing the **read-only** attributes to React components. The props are usually required to use correctly in the component. If it is not used correctly, the components may not behave as expected. Hence, it is required to use **props validation** in improving react components.

Props validation is a tool that will help the developers to avoid future bugs and problems. It is a useful way to force the correct usage of your components. It makes your code more readable. React components used special property **PropTypes** that help you to catch bugs by validating data types of values passed through props, although it is not necessary to define components with propTypes. However, if you use propTypes with your components, it helps you to avoid unexpected bugs.

Validating Props

App.propTypes 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, you will set the App.defaultProps.

Syntax:

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

ReactJS Props Validator

ReactJS props validator contains the following list of validators.

SN	PropsType	Description
1.	PropTypes.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 function.
5.	PropTypes.number	The props should be a number.
6.	PropTypes.object	The props should be an object.
7.	PropTypes.string	The props should be a string.
8.	PropTypes.symbol	The props should be a symbol.
9.	PropTypes.instanceOf	The props should be an instance particular JavaScript class.
10.	PropTypes.isRequired	The props must be provided.
11.	PropTypes.element	The props must be an element.
12.	PropTypes.node	The props can render anything: num strings, elements or an array (or fragr containing these types.
13.	PropTypes.oneOf()	The props should be one of several of specific values.
14.	PropTypes.oneOfType([PropTypes.string,PropTypes.number])	The props should be an object that be one of many types.

Example

Here, we are creating an App component which contains all the props that we need. In this example, **App.propTypes** is used for props validation. For props validation, you must have to add this line: **import PropTypes from 'prop-types'** in **App.js file**.

App.js

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';
class App extends React.Component {
   render() {
        return (
            <div>
                <h1>ReactJS Props validation example</h1>
                <table>
                    <tr>
                        <th>Type</th>
                        <th>Value</th>
                        <th>Valid</th>
                    </tr>
                  <tr>
                        <td>Array</td>
                        <td>{this.props.propArray}</td>
                        <td>{this.props.propArray ? "true" : "False"}</td>
                    </tr>
                    <tr>
                        <td>Boolean</td>
                        <td>{this.props.propBool ? "true" : "False"}</td>
                        <td>{this.props.propBool ? "true" : "False"}</td>
                    </tr>
                    <tr>
                        <td>Function</td>
                        <td>{this.props.propFunc(5)}</td>
                        <td>{this.props.propFunc(5) ? "true" : "False"}</td>
                    </tr>
                    <tr>
                        <td>String</td>
                        <td>{this.props.propString}</td>
                        <td>{this.props.propString ? "true" : "False"}</td>
                    </tr>
                    <tr>
                        <td>Number</td>
                        <td>{this.props.propNumber}</td>
                        <td>{this.props.propNumber ? "true" : "False"}</td>
                    </tr>
               </table>
          </div>
         );
```

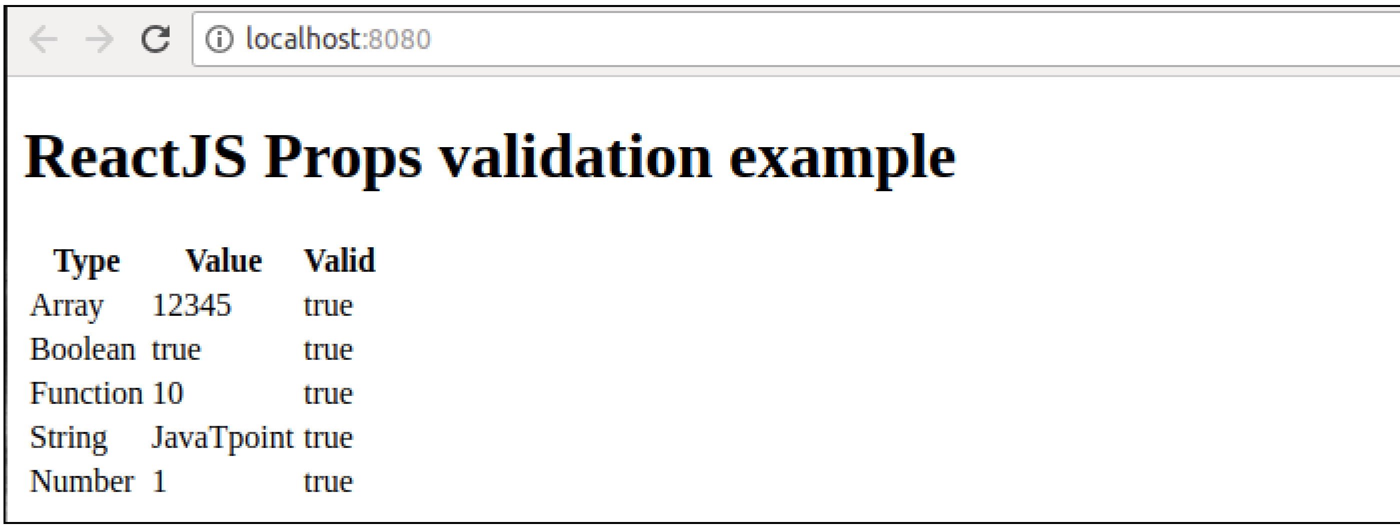
```
   }
}
App.propTypes = {
    propArray: PropTypes.array.isRequired,
    propBool: PropTypes.bool.isRequired,
    propFunc: PropTypes.func,
    propNumber: PropTypes.number,
    propString: PropTypes.string,
}
App.defaultProps = {
    propArray: [1,2,3,4,5],
    propBool: true,
    propFunc: function(x){return x+5},
    propNumber: 1,
    propString: "JavaTpoint",
}
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'));
```

Output:


localhost:8080

```
ReactJS Props validation example

  Type     Value      Valid
Array     12345      true
Boolean  true        true
Function 10          true
String    JavaTpoint true
Number   1           true
```

ReactJS Custom Validators

ReactJS allows creating a custom validation function to perform custom validation. The following argument is used to create a custom validation function.

- **props:** It should be the first argument in the component.
- **propName:** It is the propName that is going to validate.
- **componentName:** It is the componentName that are going to validated again.

Example

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