|  |  |
| --- | --- |
| **ch12.ReactJS Props Validation** | **Date: 22-02-2022** |

**Topics**

React Props Validation, Validating Props, Syntax, ReactJS Props Validator, ReactJS Custom Props Validators,

# 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 of a 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: numbers, strings, elements or an array (or fragment) containing these types. |
| **13.** | PropTypes.oneOf() | The props should be one of several types of specific values. |
| **14.** | PropTypes.oneOfType([PropTypes.string,PropTypes.number]) | The props should be an object that could 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**

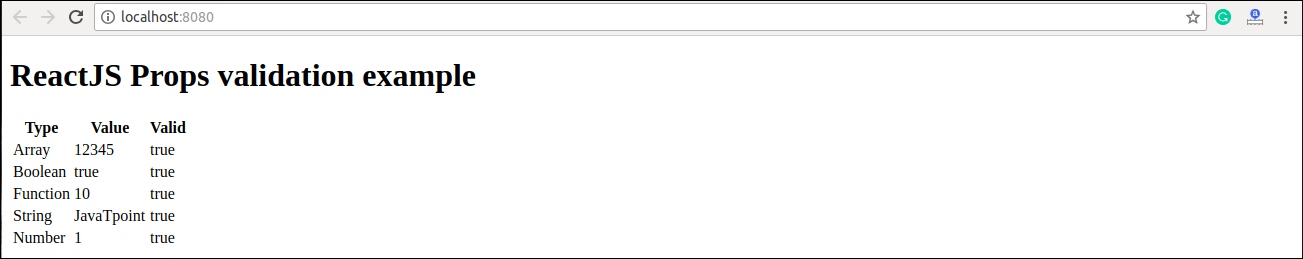
import React from 'react';

import ReactDOM from 'react-dom';

import App from './App.js';

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

**Output:**



# 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!');

            }

          }

       }

    })