Validating Props

In this example we are creating **App** component with all the **props** that we need. **App.propTypes** is used for props validation. If some of the props aren't using correct type that we assigned, we will get console warning. After we specified validation patterns, we are setting **App.defaultProps**.

### App.jsx

import React from 'react';

class App extends React.Component {

render() {

return (

<div>

<h3>Array: {this.props.propArray}</h3>

<h3>Bool: {this.props.propBool ? "True..." : "False..."}</h3>

<h3>Func: {this.props.propFunc(3)}</h3>

<h3>Number: {this.props.propNumber}</h3>

<h3>String: {this.props.propString}</h3>

<h3>Object: {this.props.propObject.objectName1}</h3>

<h3>Object: {this.props.propObject.objectName2}</h3>

<h3>Object: {this.props.propObject.objectName3}</h3>

</div>

);

}

App.propTypes = {

propArray: React.PropTypes.array.isRequired,

propBool: React.PropTypes.bool.isRequired,

propFunc: React.PropTypes.func,

propNumber: React.PropTypes.number,

propString: React.PropTypes.string,

propObject: React.PropTypes.object

}

App.defaultProps = {

propArray: [1,2,3,4,5],

propBool: true,

propFunc: function(e){return e},

propNumber: 1,

propString: "String value...",

propObject: {

objectName1:"objectValue1",

objectName2: "objectValue2",

objectName3: "objectValue3"

}

}

export default App;

### main.js

import React from 'react';

import ReactDOM from 'react-dom';

import App from './App.jsx';

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

**setState()** method is used for updating the state of the component. This method will not replace the state but only add changes to original state.

import React from 'react';

class App extends React.Component {

constructor() {

super();

this.state = {

data: []

}

this.setStateHandler = this.setStateHandler.bind(this);

};

setStateHandler() {

var item = "setState..."

var myArray = this.state.data;

myArray.push(item)

this.setState({data: myArray})

};

render() {

return (

<div>

<button onClick = {this.setStateHandler}>SET STATE</button>

<h4>State Array: {this.state.data}</h4>

</div>

);

}

}

export default App;

Sometimes you want to update the component manually. You can achieve this by using **forceUpdate()** method.

import React from 'react';

class App extends React.Component {

constructor() {

super();

this.forceUpdateHandler = this.forceUpdateHandler.bind(this);

};

forceUpdateHandler() {

this.forceUpdate();

};

render() {

return (

<div>

<button onClick = {this.forceUpdateHandler}>FORCE UPDATE</button>

<h4>Random number: {Math.random()}</h4>

</div>

);

}

}

export default App;

For DOM manipulation, we can use **ReactDOM.findDOMNode()** method. First we need to import **react-dom**.

import React from 'react';

import ReactDOM from 'react-dom';

class App extends React.Component {

constructor() {

super();

this.findDomNodeHandler = this.findDomNodeHandler.bind(this);

};

findDomNodeHandler() {

var myDiv = document.getElementById('myDiv');

ReactDOM.findDOMNode(myDiv).style.color = 'green';

}

render() {

return (

<div>

<button onClick = {this.findDomNodeHandler}>FIND DOME NODE</button>

<div id = "myDiv">NODE</div>

</div>

);

}

}

export default App;