Weekly Webtips

React JS Examples must-do for a Beginner

Rahul Bhatt Follow
Jun 26, 2020 · 3 min read ★

If you are just starting React JS or looking for a way to start React JS, check out these examples below:

These examples/concepts are a must for a beginner to get well versed with.

Stateful Components are components depending on its state object and change its own state. The component re-renders based on changes to its state, and may pass down properties of it's state to child components as properties on a props object.

```
import React, { Component } from 'react'

export default class GameCharacter extends Component {
    constructor(props){
        super(props);
        this.state = {
            firstName: 'Rahul'
        }
    }
    render() {
        return (
            <div>
                <h1>Hello {this.state.firstName}</h1>
            </div>
        )
    }
}
```

Fig.1 Stateful Components

Stateless Components purely acts as a view, the components do not have any state. Here, we pass props (method argument) and provide the value along with it. Notice there is “this” keyword.

```
import React, { Component } from 'react'

const name = props => {
    return(
        <h1>{props.name}</h1>
    )
}
export default name;

<Name name="Rahul" />
```

Fig.2. Stateless component

Implicit Return Components are the same as above but without a return statement. This would be useful when you are creating a button or a default form that could be used elsewhere.

```
import React, { Component } from 'react'

const name = props => <h1>{props.name}</h1>;

export default name;

<Name name="Rahul" />
```

Fig.3. Implicit React Component

Fragments are used to render multiple elements without using a wrapper class. Fragments let you group a list of children without adding extra nodes to the DOM. Fragments are used in place of necessary tag so as to avoid invalid HTML DOM elements. You can use <> </> instead of <React.Fragment></React.Fragment>

```
import React, { Component } from 'react'

export default class GameCharacter extends Component {
    render() {
        return (
            <React.Fragment>
                <td>1</td>
                <td>2</td>
            </React.Fragment>
        )
    }
}
```

Fig.4. Fragment in React

Normally you would use   which makes the HTML rendering illogical and invalid due to <td> element (also applicable to any listing element like <li>, etc.).

JSX allows to write HTML elements in JS and place them in the DOM without any createElement() or appendChild() methods. It is used to convert HTML tags into react elements.

```
const name = <h1>I m Steve Rogers </h1>;

ReactDOM.render(name,document.getElementById('roo
t'));
```

Fig.5. JSX element

Only single parent tag is allowed which means that no multiple parent tags are allowed, either nest it in , <ul>, <table> or use React.Fragments.

```
import React, { Component } from 'react'

const tags = (<ul><li>1</li><li>2</li>
</ul>);
const tags = (
    <h1>Hello</h1>
    <h3>Can you hear me?</h3>
    <h4>This won't work on me Adele</h4>
);
```

Fig.6. Single parent tag allowed

One tip before we jump to our last example:

```
import React, { Component } from 'react';

export default class GameCharacter extends
Component {
    constructor(props) {
        super(props)

        this.state = {
            value: 1
        }
    }

    componentDidMount(){
        this.setState({value: this.state.value +
1});}

    render() {
        return (
            <div>
                {this.state.value}
            </div>
        );
    }
}
```

Fig.7. One last tip

You can see the value is updated to 2 after the component has been mounted, there can occur several problems with this method, use this instead:

```
...
componentDidMount(){
        this.setState(prevState => ({value: prevState.value
+ 1}));
...
```

Fig.8. Use prevState to fetch the previous state and then update it.

There exists a clearer way also:

```
...
componentDidMount(){
        this.setState(({value}) => ({value: value +
1}))}
...
```

State vs Props

State belongs to a component and is mutable which means the state of the component can be changed. The state of the component is changed using setState({}) method.

Props similar to a method argument, they are passed to a function or a constructor. The props that are passed are immutable.