Class Component and State | Cheat Sheet

Concepts in Focus

- Components
 - Functional Components
 - Class Components
- React Events
- State
 - Updating State
 - State Updates are Merged
 - o Functional Components vs Class Components
- Counter Application

1. Components

There are two ways to write React Components.

They are:

- Functional Components
- Class Components

1.1 Functional Components

These are JavaScript functions that take props as a parameter if necessary and return react element (JSX).

```
const Welcome = () => <h1>Hello, User</h1>;

export default Welcome;
```

1.2 Class Components

These components are built using an ES6 class.

To define a React Class Component,

- Create an ES6 class that extends React.Component .
- Add a single empty method to it called render() .

1.2.1 extends

The

extends keyword is used to inherit methods and properties from the React.Component .

1.2.2 render()

The

render() method is the only required method in a class component. It returns the JSX element.

Syntax:

```
import { Component } from "react";

class MyComponent extends Component {
   render() {
      return JSX;
   }
}
```

Use

this.props in the render() body to access the props in a class component.

```
1 - class Welcome extends Component {
2 - render() {
3     const { name } = this.props
4     return <h1>Hello, {name}</h1>
5  }
6 }
```



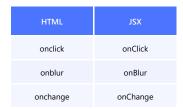
The component name should always be in the pascal case.

2. React Events

Handling events with React elements is very similar to handling events on DOM elements. There are some syntax differences:

 $1. \ React \ events \ are \ named \ using \ {\color{blue} camel Case}, \ rather \ than \ {\color{blue} lower case}.$

Example:



2. With JSX, you pass a **function** as the event handler rather than a **string**.

Example:

```
i 1 <button onclick="activateLasers()">Activate Lasers</button>
```

We should not call the function when we add an event in JSX.

In the above function, the

handleClick is called instead of passed as a reference.

In the above function, the

handleClick is passed as a reference. So, the function is not being called every time the component renders.

Providing Arrow Functions

To not change the context of

this , we have to pass an arrow function to the event.

```
1  class MyComponent extends Component {|
2     handleClick = () => {
3         console.log(this) // MyComponent {...}
4     }
5     render() {
```

3. State

The state is a JS object in which we store the component's data that changes over time.

When the state object changes, the component re-renders.

Intialising State:

```
1  class Counter extends Component {
2   state = { count: 0 }
3  render() {
4    const { count } = this.state;
5    return {count};
6  }
7 }
```

3.1 Updating State

We can update the state by using

setState() . We can provide function/object as an argument to set the state.

Providing Function as an Argument:

Syntax:

```
this.setState( prevState => ({... }) )
```

Here the previous state is sent as a parameter to the callback function.

```
1 v onIncrement = () => {
2    this.setState((prevState) =>
3         console.log(`previous state value ${prevState.count}`)
i 4    )
i 5  }
```

3.2 State Updates are Merged

State updates are merged. It means that when you update only one key-value pair in the state object, it will not affect the other key-value pai object.

```
1  // For example let's say your state is as followed:
2  state = { key1 : value1, key2 : value2 }
3
4  // if you use this.setState such as :
5  this.setState((prevState) => ({ prevState.key1 : value3 }))
6
7  // your new state will be :
8  state = { key1 : value3, key2 : value2 }
```

3.3 Functional Components vs Class Components

Functional Components	Class Components
Renders the UI based on props	Renders the UI based on props and state

Use Class Components whenever the state is required. Otherwise, use the Functional components.

4. Counter Application

File: src/App.js

```
import Counter from "./components/Counter";

const App = () => {
   return <Counter />
}

export default App;
```

File: src/components/Counter/index.js

```
Submit Feedback

Notes Discussions
```