REACT Cheat Sheet

This cheat sheet is for the course **React 18 Course - Learn React JS the fast way** by ***Jannick Leismann***.

useState & useEffects

Hooks

Are functions that let you use **state** and other React features in functional components. They provide a way to "**hook into**" React's state and lifecycle features from function components, which were previously only available in class components.

'useState'

Used to add state variables to functional components. Allows you to declare **state** in a **functional component**, and provides a way to update the state.

Usage of **useState:**

```
import React, { useState } from "react";
import "../App.css";

export default function MoviesGrid() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>Click me</button>
    </div>
  );
}
```

useState returns an array with two elements: the current state value and a function to update it**.**

You can initialize with an initial value (e.g., 0 in the example)

When to use: When you need to manage **local state** within a **component**, such as **form inputs**, **counters**, **toggles**, etc.

‘useEffect’

Is used to perform **side effects** in **functional components**. It can handle tasks such as **data fetching**, **manually changing the DOM,** etc. which are side effects and should not be done during render.

Usage of **useEffect**

```
import React, { useState, useEffect } from "react";
import "../App.css";

export default function MoviesGrid() {
  const [count, setCount] = useState(0);

  // Use useEffect to update the document title whenever count changes
  useEffect(() => {
    document.title = `You clicked ${count} times`;
  }, [count]); // Only re-run the effect if count changes

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>Click me</button>
    </div>
  );
}
```

useEffect takes two arguments: a function (the effect) and an optional array of dependencies.

The effect runs after every **render** by default.

When to use: When you need to perform operations after the **component renders**, such as **data fetching**, setting up subscriptions, or manually updating the DOM.