REACT Cheat Sheet

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

useReducer

A react hook that allows **complex state management logic**. Alternate to `**useState**` hook and useful when handling complex state transitions.

Reducer function

This reducer function takes **two arguments**: the **current state** and an **action**. Determines how the state should change based on the action received and uses a **switch statement** to handle **action types**.

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

const reducer = (state, action) => {
  switch (action.type) {
    case 'increment':
      return { count: state.count + 1 };
    case 'decrement':
      return { count: state.count - 1 };
    default:
      return state;
  }};
```

Initialize state and set up reducer

Use the **useReducer** hook to initialize the state and dispatch function. The **initialState** is an object that represents the **initial state** of the **component**.

The useReducer hook takes the **reducer function** and **initial state** as arguments and returns the array with the **current state** and the **dispatch** function.

```
const initialState = { count: 0 };

export default function Counter() {
  const [state, dispatch] = useReducer(reducer, initialState);
```

```
  return (
    <div>
      <p>Count: {state.count}</p>
      <button onClick={() => dispatch({ type: 'increment'
})}>Increment</button>
      <button onClick={() => dispatch({ type: 'decrement'
})}>Decrement</button>
    </div>
  );
};
```

Return Component

The **Counter** component renders the current count and buttons to **increment** and **decrement** the count.

useReducer vs useState

When state transitions are complex and have multiple values, multiple state variables depend on each other and should be updated together, **useReducer** is more effective.