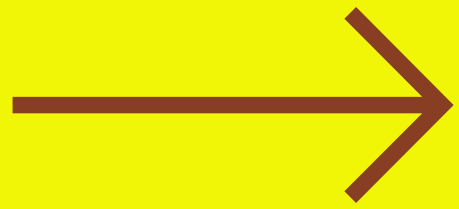
Beyond useState

The anatomy of useReducer hook in React



General Syntax

```
const [state, dispatch] = useReducer(reducer,
initialState);
```

In one senetnce , if anyone has to understand what useReducer is than memorize this

To Update the state , useReducer “**dispatches**” an “**action**” to a “**reducer**” function & the action is a javascript object

or even simpler terms .

To Update the state , useReducer calls a method(dispatch) that takes an object as parameters(action). Actual logic to update the state in written inside reducer function

Lets Dissect

This is the current state managed by the reducer. You can read from it to access the current state values.

```
const [state, dispatch] = useReducer(reducer, initialState);
```

This is a function you use to send actions to the reducer, which in turn updates the state

This function is responsible for handling state updates. It takes two arguments: the current state and an action object that describes what kind of update to perform. The reducer function must return the new state.

Usage

Importing the Hook

To use the useReducer hook, you first need to import it from the React library at the top of your JavaScript or TypeScript file:

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

Reducer Function

At the core of **useReducer** is a reducer function. This function is responsible for handling state updates. It takes two arguments: the current state and an action object that describes what kind of update to perform**. The reducer function must return the new state.**

```
 const reducer = (state, action) => {
// Update state based on the action
// Return the new state
   };
```

Dispatching Actions

To update the state managed by the reducer, you dispatch actions using the dispatch function. An action is a plain JavaScript object that typically has a type property that describes the action and additional data as needed:

```
dispatch({ type: 'INCREMENT', payload: 1 });
```

In this example, we're dispatching an action of type 'INCREMENT' with a payload of 1.

Handling Actions in the Reducer

Inside the reducer function, you use a switch statement (or another branching mechanism) to handle different action types and update the state accordingly:

```
const reducer = (state, action) => {
  switch (action.type) {
    case 'INCREMENT':
      return { count: state.count + action.payload };
    case 'DECREMENT':
      return { count: state.count - action.payload };
    default:
      return state; // Return the current state for unknown actions
  }
};
```

This code handles actions of types 'INCREMENT' and 'DECREMENT' by updating the state based on the action's payload. For unknown actions, it returns the current state unchanged.

Stay tuned

🙏

to know when to choose
useReducer over useEffect