Twitter Thread by Pratham

Pratham
@Prathkum

React Hooks are so powerful and especially some additional hooks.

Do you know there is an alternative to useState called useReducer hook? It is used to manage more complex states.

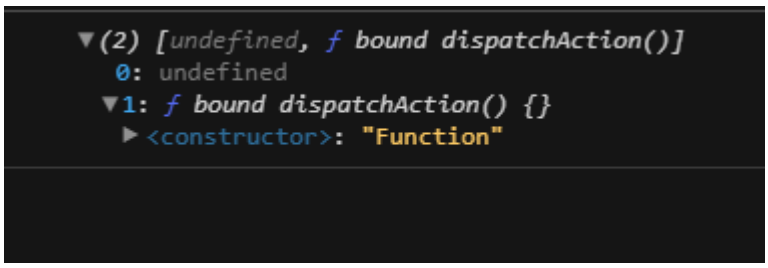
Let's look into detail ■■■



Usually, the useState hook is used to tackle states in React where you can pass the initial state and React preserves state for you between re-renders

What if you need to manage more complex states? Here useReducers comes into play

Similar to useState, the useReducer hook also returns a pair of values. Let's see what these two things are



The first value is `undefined` which is the current state (it is undefined because we didn't pass any initial state in the useReducer hook)

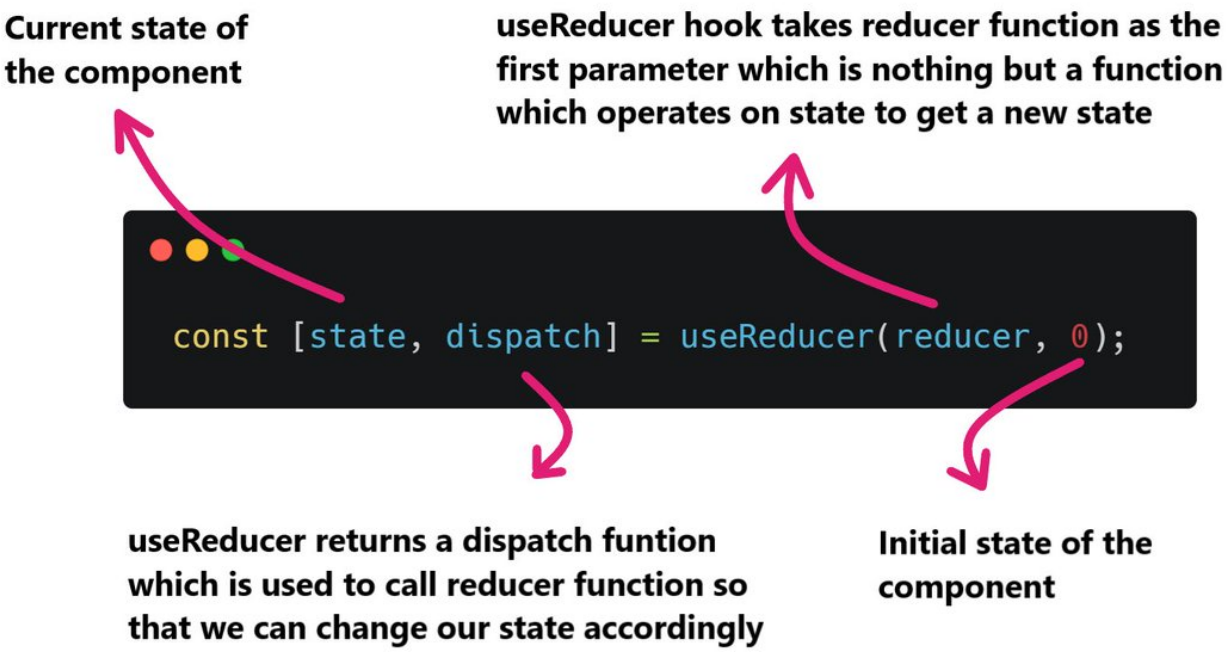
The second value is a dispatch function using which we can update our state.

The useReducer hook accepts 3 parameter

1. reducer
2. Initial state
3. init

Now we know that useReducer takes three parameters and return two values

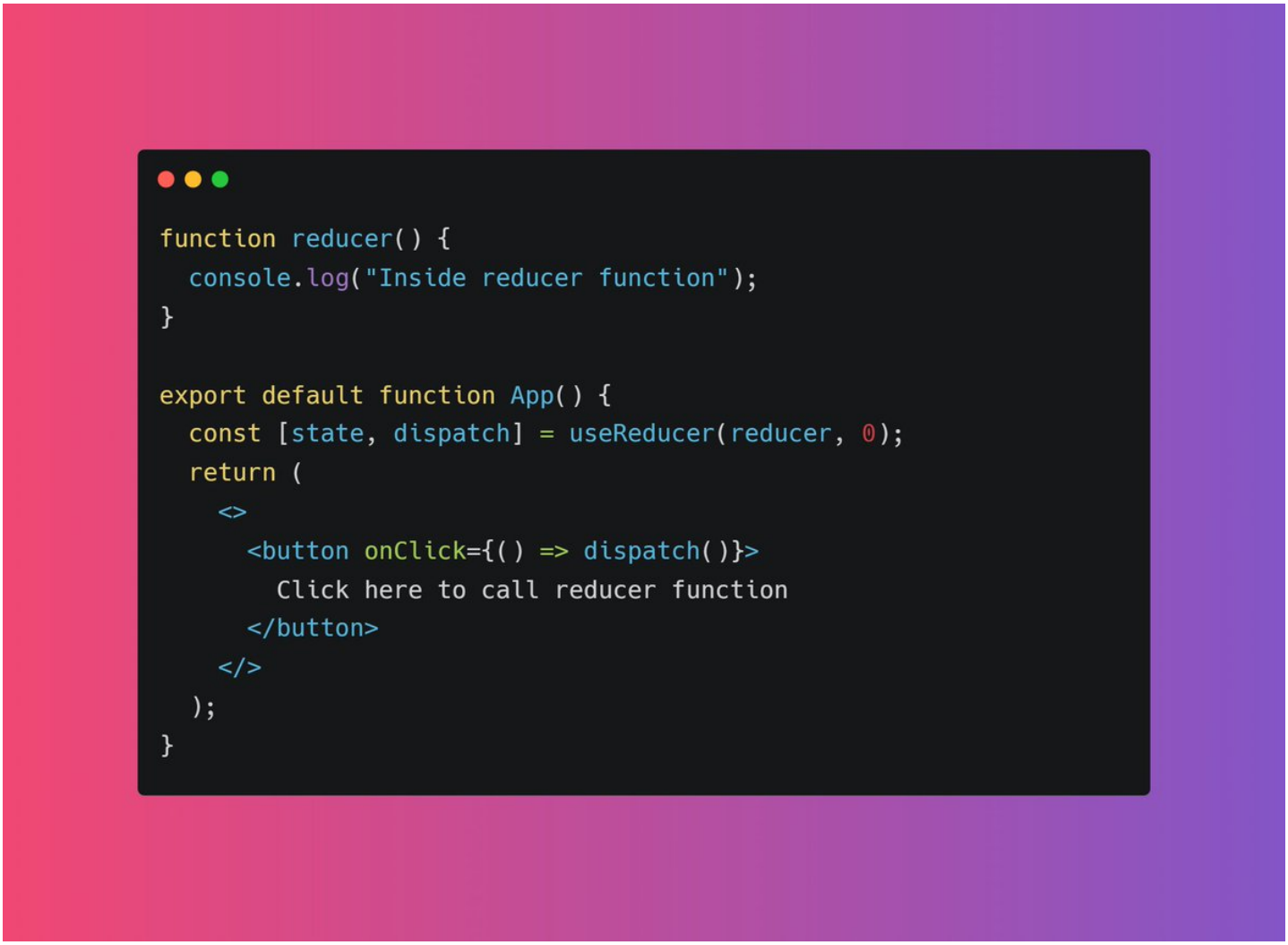
Here is the complete syntax ■■



@prathkum

Let's try to see this into action.

I just implemented a basic code here. Basically, I am calling dispatch function on button click which will eventually call reducer function



```
function reducer() {
  console.log("Inside reducer function");
}

export default function App() {
  const [state, dispatch] = useReducer(reducer, 0);
  return (
    <>
      <button onClick={() => dispatch()}>
        Click here to call reducer function
      </button>
    </>
  );
}
```

So far we haven't seen how the state can be managed using this.

Before that one thing to note here is that useReducer is for handling more complex states hence it is recommended to passing the current state as an object



Now we can simply update our state by passing the state parameter in the reducer function.

Something like this ■■

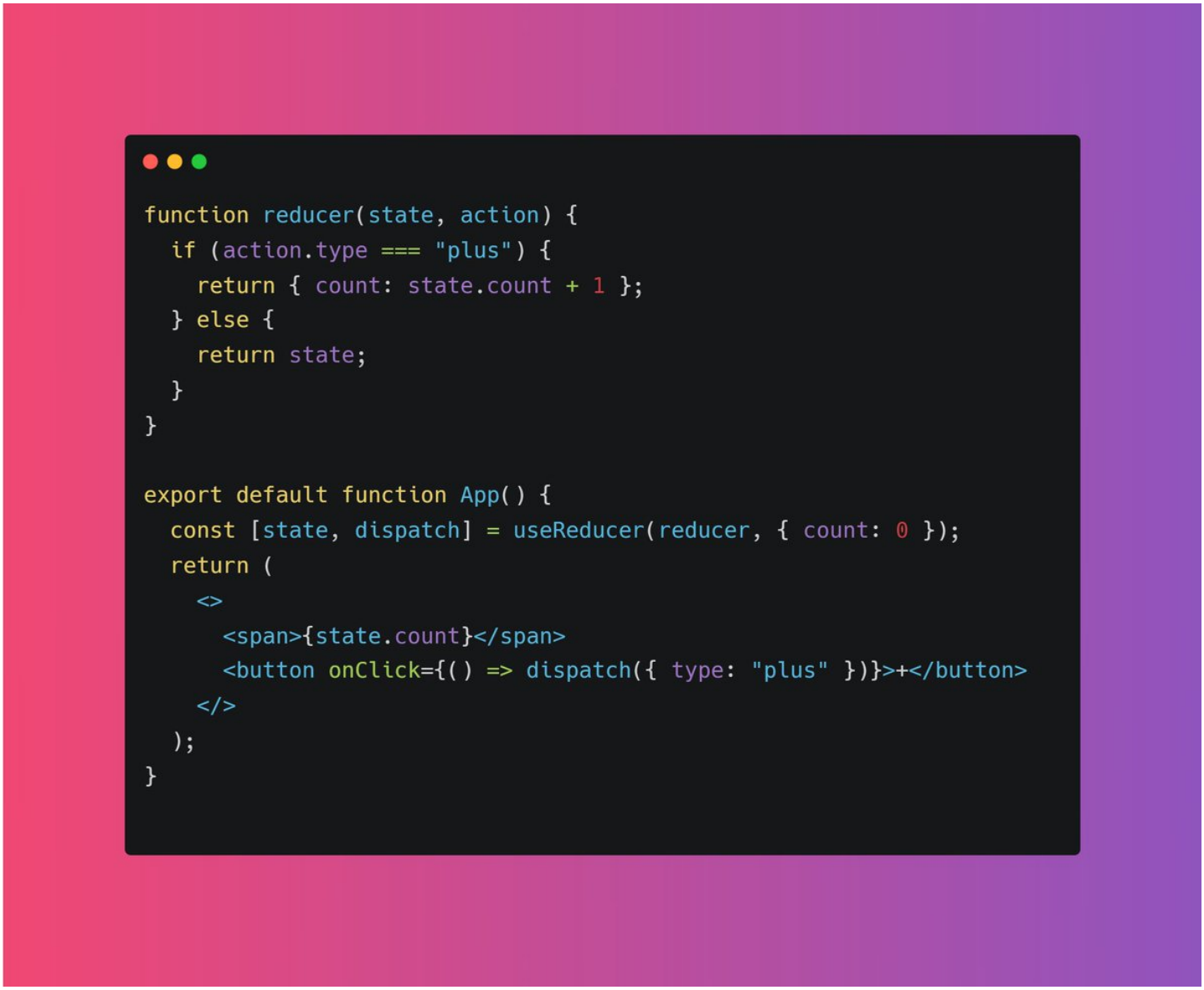


Now every time I click on the button my state will be incremented by 1 because the reducer function returning the new state by adding one to the previous state.

Play around with it here: https://t.co/6BlaBeul1U

The reducer function accepts the second argument as well known as "action" using which we can tell the reducer function that what operations need to perform on the state.

Generally, we can pass different actions through dispatch to reducer via an object.



```
function reducer(state, action) {
  if (action.type === "plus") {
    return { count: state.count + 1 };
  } else {
    return state;
  }
}

export default function App() {
  const [state, dispatch] = useReducer(reducer, { count: 0 });
  return (
    <>
      <span>{state.count}</span>
      <button onClick={() => dispatch({ type: "plus" })}>+</button>
    </>
  );
}
```

This is how you can pass different types of operations in the reducer function using the action parameter.

What if I want to pass some values through dispatch to the reducer function?

Here is "payload" comes into play. It is used to pass the value which represents the payloads of the action.

For example, suppose I want to pass "temp" variable ■■

https://t.co/6BlaBeul1U



```
function reducer(state, action) {
  if (action.type === "plus") {
    return { count: state.count + action.payload.temp };
  } else {
    return state;
  }
}

export default function App() {
  const temp = 14;
  const [state, dispatch] = useReducer(reducer, { count: 0 });
  return (
    <>
      <span>{state.count}</span>
      <button onClick={() => dispatch({ type: "plus", payload: { temp: temp } })}>
        +
      </button>
    </>
  );
}
```

This may sound a little confusing for the first time. Don't worry I created a notes app for you using the useReducer hook.

Check out the code and try to play around with it for better understanding.

https://t.co/TCPEn0W3Pz

I have a video tutorial on YouTube explaining about useReducer in more bit details.

Check it out: https://t.co/nt4hZzInIR

That's pretty much it for this thread. If you like this thread, share it with your connections it means a lot to me ■

Peace out ■