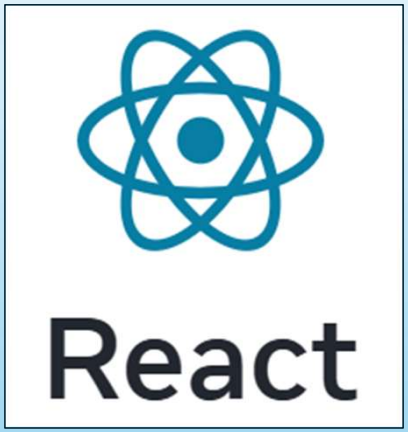

Lesson - useActionState

Luis Coco Enríquez

1. Introduction

useActionState is a React Hook that helps you manage the state of an action, such as a form submission or any user-triggered process. It runs a function you provide (fn), stores its returned value as state, and gives you a flag to know if the action is still running. It's especially useful when working with server actions or any async operations that update your UI after completion.

import { **useActionState** } from 'react';

const [**state**, **formAction**, **isPending**] = **useActionState**(fn, initialState, permalink?);

Parameters

fn (function): The action function that runs when the form is submitted or the action is triggered.

It receives two **arguments** automatically:

The **previous state** (from the last result of the action).

The **form data** or arguments passed from the form submission.

It should **return** the **new state value** (can be synchronous or asynchronous).

Return Values

The hook **returns** three items:

state – the current value of your action's state (updated after each action).

formAction – a function you attach to your form's action attribute to trigger fn.

isPending – a boolean flag that is true while the action is running, and false once it completes.

2. What is useActionState and when do we use it?

You use it when you want a form or a user interaction to trigger an async function and then update the component with the result automatically.

```
const [state, formAction, isPending] = useActionState(async (prevState, formData) => {
  const result = await saveData(formData);
  return result.message;
}, "Initial message");
```

The hook calls **saveData** when the action runs.

The returned value becomes the new state.

isPending tells you if the action is still running.

3. What the hook returns and how to think of it

const [**message**, **action**, **isPending**] = **useActionState**(sendMessage, "");

message: current value of the **state**.

action: function you use in your form or button.

isPending: true while the action is running.

```
<form action={action}>
  <button disabled={isPending}>Send</button>
</form>
<p>{message}</p>
```

This automatically connects the form to your action and updates the UI once it completes.

4. Useful scenarios and benefits

Great for server actions that need to send data and show results.

Eliminates extra **useState** + **useEffect** logic.

Makes loading indicators simple.

const [**result**, **updateUser**, **pending**] = **useActionState**(updateUserAction, null);

```
<form action={updateUser}>
  <button disabled={pending}>Update</button>
</form>
```

5. Parameter details and what to watch

```
const [state, action] = useActionState(
  async (previousState, formData) => {
    // previousState is the old state
    // formData is the form input data
    return await doSomething(formData);
  },
  "Initial"
);
```

Parameter 1: **action** function (receives previous state and form data).
Parameter 2: initial **state**.
Parameter 3 (optional): permalink (used with navigation or progressive enhancement).

6. Caveats and gotchas

The first argument of your action is the previous state, not the form data.

Works best with asynchronous actions.

If you use permalink, ensure the same form exists on both pages.

const [**data**, **submit**, **pending**] = **useActionState**(handleSubmit, "Init", "my-form");

If you change the action or remove the form, React may lose the link to its state.

7. Summary and key take-aways

useActionState lets you:

Run actions (like form submissions) declaratively.

Automatically store returned values as state.

Easily manage loading with isPending.

const [**message**, **formAction**, **pending**] = **useActionState**(sendMessage, "");

Use it whenever your UI depends on the result of an action — especially in server-action environments or progressive web apps built with modern React.