




React Unpacked

Demo

page.tsx M X

```
13
14
15 async function handleAddToCart(e: FormEvent<HTMLFormElement>) {
16   startTransition(async () => {
17     e.preventDefault();
18     setIsPending(true);
19
20     let res = await fetch("/api/cart/add-to-cart", { method: "POST"
21     let json = await res.json();
22
23     if (json.ok) {
24       await revalidate();
25       setError(null);
26     } else {
27       setError(json.error);
28     }
29
30     setIsPending(false);
31   });
32 }
33
34 if (!data) {
35   return <Spinner fullPage />;
36 }
37
```



Add to cart

Your cart: 9

Classic T-Shirt


Details +

0 9 Refresh

Stock Cart

page.tsx M X

```
8
9
10 export default function Page() {
11   let { data, revalidate } = useQuery("/api/cart");
12   let { isPending, startTransition } = useTransition();
13   let { error, setError } = useState(null);
14
15   async function handleAddToCart(e: FormEvent<HTMLFormElement>) {
16     startTransition(async () => {
17       e.preventDefault();
18
19       let res = await fetch("/api/cart/add-to-cart", { method: "POST"
20       let json = await res.json();
21
22       if (json.ok) {
23         await revalidate();
24         setError(null);
25       } else {
26         setError(json.error);
27       }
28     });
29   }
30
31   if (!data) {
32     return <Spinner fullPage />;
33   }
34 }
35
```



Add to cart

Your cart: 14

Classic T-Shirt

Details +

1 14 Refresh


Stock Cart

The screenshot displays a web application interface for a product page. On the left, a code editor shows the implementation of the `addToCart` function and the `render` method. The function uses `useState` to manage `error` and `setError`, and `startTransition` to wrap an `async` fetch call to the `/api/cart/add-to-cart` endpoint. The `render` method uses `if (data)` to show a spinner or the `<ProductDetails />` component, and a `<Form>` with an `addToCartAction` and a submit button. The cart summary at the bottom shows 0 items in stock and 17 in the cart, with a 'Refresh' button.

The screenshot displays a web application interface for a product page. On the left, the code editor shows the implementation of the 'Add to cart' button and the 'Pending' state handling using the 'useFormStatus' hook. The main content area shows a product image of a man in a white t-shirt, the product name 'Classic T-Shirt', and a 'Details' link. Below the product image is a form with an 'Add to cart' button. The form is currently in a 'Pending' state, indicated by a spinner. The cart summary at the bottom shows 0 items in the cart and 23 items in the stock, with a 'Refresh' button.

page.tsx M X

```
13 let [error, setError] = useState<null>;
14 let [state, addToCartAction, isPending] = useActionState<async () => {
15   let res = await fetch("/api/cart/add-to-cart", { method: "POST" });
16   let json = await res.json();
17
18   if (json.ok) {
19     await revalidate();
20     setError<null>;
21   } else {
22     setError<json.error>;
23   }
24 };
25
26 if (!data) {
27   return <Spinner fullPage />;
28 }
29
30 return (
31   <
32     <ProductDetails />
33
34     <form
35       action={addToCartAction}
36       className="flex flex-col gap-4 items-center"
37     >
38       <Button type="submit">Add to cart</Button>
39
40       <>Your cart: {data.items}</>
41
42       {error && < className="text-red-500">{error}</>}
43     </>
44   </>
45 );
```



Add to cart

Your cart: 25

Classic T-Shirt

Details +


0 + 25

Stock Cart

Refresh

page.tsx M X

```
12 let [data, revalidate] = useQuery< /api/cart />;
13 let [error, addToCartAction, isPending] = useActionState<async () => {
14   let res = await fetch("/api/cart/add-to-cart", { method: "POST" });
15   let json = await res.json();
16
17   if (json.ok) {
18     await revalidate();
19   } else {
20     return json.error;
21   }
22 };
23
24 if (!data) {
25   return <Spinner fullPage />;
26 }
27
28 return (
29   <
30     <ProductDetails />
31
32     <form
33       action={addToCartAction}
34       className="flex flex-col gap-4 items-center"
35     >
36       <Button type="submit">Add to cart</Button>
37
38       <>Your cart: {data.items}</>
39
40       {error && < className="text-red-500">{error}</>}
41
42       {isPending && <Spinner />}
43     </>
44   </>
45 );
```



Add to cart

Your cart: 27

Classic T-Shirt

Details +


0 + 27

Stock Cart

Refresh

page.tsx M

```
1 "use client";
2
3 import { Spinner } from "@components/spinner";
4 import Button from "@components/ui/button";
5 import ProductDetails from "@components/ui/product-details";
6 import useActionState from "@use-action-state";
7 import useQuery from "@use-query";
8 import { useOptimistic } from "react";
9
10 export default function Page() {
11   let { data, revalidate } = useQuery("/api/cart");
12   let [optimisticData, setOptimisticData] = useOptimistic(data);
13
14   let { error, addToCartAction, isPending } = useActionState(async () => {
15     let res = await fetch("/api/cart/add-to-cart", { method: "POST" });
16     let json = await res.json();
17
18     if (json.ok) {
19       await revalidate();
20     } else {
21       return json.error;
22     }
23   });
24
25   if (!data) {
26     return <Spinner fullPage />;
27   }
28
29   return (
30     <ProductDetails />
31   );
32 }
```



Add to cart

Your cart: 31

Classic T-Shirt

Details +

2

+

30


Refresh

Stock

Cart

page.tsx M X

```
17
18   if (json.ok) {
19     await revalidate();
20   } else {
21     return json.error;
22   }
23 };
24
25 if (!data) {
26   return <Spinner fullPage />;
27 }
28
29 return (
30   <ProductDetails />
31   <form
32     action={() => {
33       setOptimisticData((d) => ({ ...d, items: d.items + 1 }));
34       addToCartAction();
35     }}
36     className="flex flex-col gap-4 items-center"
37   >
38     <Button type="submit">Add to cart</Button>
39
40     <div>Your cart: {optimisticData.items}</div>
41
42     {error && <div className="text-red-500">{error}</div>}
43
44     {isPending && <Spinner />}
45   </form>
46 );
47 }
```



Add to cart

Your cart: 33

Classic T-Shirt

Details +

0

+

33

Refresh

Stock

Cart