**Next.js 14+ App Router Server Actions with React useTransition. A new Era for FullStack!**

[[Mguleryuz](https://medium.com/@mguleryuz3?source=post_page-----2798e58bb793--------------------------------)](https://medium.com/@mguleryuz3?source=post_page-----2798e58bb793--------------------------------)

[Mguleryuz](https://medium.com/@mguleryuz3?source=post_page-----2798e58bb793--------------------------------)

·

Follow

3 min read

·

Dec 29, 2023

211



**Summary:**Up until now writing full-stack applications were slowed down by all the API boilerplate needed for endpoints and their handling on the client side. We will go over a new technique I came up with and how it can triple our productivity building **FullStack**Apps.

**Currently:**There is support + documentation for submitting forms using **Server Actions**but even that has many caveats.

* Handling Toasts
* Setting States
* Implementing 3rd party libraries such as @tanstack/react-query
* …

**Our Solution:**We have two main parts to it-

* serverActionWrapper.ts:

type PromiseType<T> = T extends Promise<infer U> ? U : T  
  
export type ServerActionWrapperReturn<T> = PromiseType<  
 ReturnType<typeof serverActionWrapper<T>>  
>  
  
export async function serverActionWrapper<T>(action: () => Promise<T>) {  
 try {  
 const res = await action()  
 return { success: true, res } as const  
 } catch (e: any) {  
 return {  
 success: false,  
 res: serializeError(e),  
 } as const  
 }  
}  
  
function serializeError(error: Error) {  
 if (!(error instanceof Error))  
 return {  
 name: 'No Instance of Error',  
 message: 'Unexpected: cought response is not an instance of Error',  
 }  
  
 return {  
 name: error.name,  
 message: error.message,  
 stack: error.stack,  
 cause: error.cause,  
 }  
}

* **useServerAction.ts:**

import { ServerActionWrapperReturn } from '@/lib/utils/serverActionWrapper'  
import { useTransition } from 'react'  
  
export default function useServerAction() {  
 const startTransition = useTransition()[1]  
  
 async function serverAction<T>(  
 action: () => Promise<ServerActionWrapperReturn<T>>  
 ) {  
 let promise: Promise<ServerActionWrapperReturn<T>> | undefined  
  
 startTransition(() => {  
 promise = action()  
 })  
  
 const awaited = await promise!  
  
 if (!awaited.success) {  
 const error = new Error()  
 error.stack = awaited.res.stack  
 error.name = awaited.res.name  
 error.message = awaited.res.message  
 error.cause = awaited.res.cause  
 throw error  
 }  
  
 return awaited.res  
 }  
  
 return serverAction  
}

**Context:**

* **serverActionWrapper:**We useto wrap our server action the main reason behind this is to handle the errors on our servers since we are passing the promise to our client we don’t want to throw the error on the server because it wouldn’t be handled.
* **useServerAction:**This is our react hook which utilizes useTransition to start the action and pass the promise into an empty variable, the promise then gets awaited inside the function and if there is en error it gets constructed and thrown from there, else the awaited response gets returned.

**Example of the Usage:**

* **Server Action:**

'use server'  
  
import { serverActionWrapper } from '@/lib/utils/serverActionWrapper'  
  
export async function testAction() {  
 return await serverActionWrapper(async () => {  
 if (true) throw new Error('test error')  
  
 return 'hello'  
 })  
}

* **Client Component:**

'use client'  
  
import { testAction } from '@/lib/actions/test'  
import useServerAction from '@/hooks/useServerAction'  
import { useEffect } from 'react'  
  
export default function Page() {  
 const serverAction = useServerAction()  
  
 useEffect(() => {  
 serverAction(testAction)  
 .then((res) => console.log(res))  
 .catch((e) => console.error(e))  
 }, [])  
  
 return <></>  
}

**Conclusion:**We implemented a solution where we can run **ServerActions**in our client and handle the errors, furthermore we can use 3rd party libraries such as @tanstack/react-query to handle our query flows, without writing a single line of api route.

**You can browse my work from:**[**Linktree**](https://linktr.ee/mgguleryuz)