



SERVER ACTIONS IN NEXT.JS

معلومات مقدمه و نصائح !

```
app/page.tsx TypeScript
1 export default async function Page() {
2   const publishVersion = await getLatestVersion();
3
4   async function publish() {
5     "use server";
6     if (publishVersion !== await getLatestVersion()) {
7       throw new Error('The version has changed since pressing publish');
8     }
9     ...
10  }
11
12  return (
13    <form>
14      <button formAction={publish}>Publish</button>
15    </form>
16  );
17 }
```

CLOSURE

- **Server Actions** بتستخدم مفهوم الـ **Closure** لما تستدعي متغيرات معرفة في **نفس** الـ **Server File**
- الـ **closure** هو ان **scope** الـ **function** يكون قادر يشوف **variables** الـ **Scope** الي فوقيه
- هنا **function** الـ **public** قدرت انها تشوف **constant** الـ **publishVersion** مع انها مش في الـ **Scope** بتاعها، هو ده الـ **Closure**

SECURITY

- احرص دائما انك تعمل **validation** علي ال **data** الي رجعالك من ال **inputs**, ممكن تستخدم مكتبه زي **ZOD**
- و اضافته **authentication** يبقا افضل

TS app/actions.ts

VALIDATION TypeScript



```
1  'use server'
2
3  import { z } from 'zod'
4
5  const schema = z.object({
6    email: z.string({
7      invalid_type_error: 'Invalid Email',
8    }),
9  })
10
11 export default async function createUser(formData: FormData) {
12   const validatedFields = schema.safeParse({
13     email: formData.get('email'),
14   })
15
16   // Return early if the form data is invalid
17   if (!validatedFields.success) {
18     return {
19       errors: validatedFields.error.flatten().fieldErrors,
20     }
21   }
22
23   // Mutate data
24 }
```

TS app/actions.ts

AUTHENTICATION

```
1  'use server'
2
3  import { auth } from './lib'
4
5  export function addItem() {
6    const { user } = auth()
7    if (!user) {
8      throw new Error('You must be signed in to perform this action')
9    }
10
11    // ...
12 }
```

```
'use client'
```

```
import { updateUser } from '../actions'
```

```
export function UserProfile({ userId }: { userId: string }) {  
  const updateUserWithId = updateUser.bind(null, userId)
```

```
  return (  
    <form action={updateUserWithId}>  
      <input type="text" name="name" />  
      <button type="submit">Update User Name</button>  
    </form>  
  )  
}
```

BIND method

- لو عايز تبعث **parameter** لل **server action** زي مثلا **userId**
- ف لازم تستخدم **bind** عشان لو بعث **parameter** من غير **bind** بالشكل ده :
- ❌ **const user = updateUser(null,userId)**
- ف انت بت **invoke** ال **function** علي طول
- ف وظيفه **bind** انها تنشئ نسخه من ال **function** وتنفذها وقت استدعائها
- ال **null** هنا قيمتها **this** و مش مهمه في ال **Server action**

```
1  'use server'
2
3  import { revalidatePath } from 'next/cache'
4
5  export async function createPost() {
6    try {
7      // ...
8    } catch (error) {
9      // ...
10   }
11
12   revalidatePath('/posts')
13 }
```

RevalidatePath

- لو عايز تعيد تحميل صفحه بعد اي **server action** تقدر تستخدم **revalidatepath** و تحط **path** الصفحة في ال **param**

RevalidateTag

- اما لو عايز تحدث **requests** معينه بعد اي **server action**, ف تقدر تبعث **next param** مع ال **fetch** و جوه ال **object** تبعث ال **tags** الي انت حددتها في ال **Server action**

TS app/actions.ts

TypeScript

```
1  'use server'
2
3  import { revalidateTag } from 'next/cache'
4
5  export async function createPost() {
6    try {
7      // ...
8    } catch (error) {
9      // ...
10   }
11
12   revalidateTag('posts')
13 }
```

```
const posts = await fetch('https://api.example.com/posts', {
  next: { tags: ['posts'] }, // على هذه البيانات 'posts' وضع وسم
});
```

useActionState

- تقدر تستخدم **hook** زي **useActionState** بتديك ال **server action** , **pending state** و **state** انت بتعرفها في ال **action**

```
1 'use server'
2
3 import { redirect } from 'next/navigation'
4
5 export async function createUser(prevState: any, formData: FormData) {
6   const res = await fetch('https://...')
7   const json = await res.json()
8
9   if (!res.ok) {
10     return { message: 'Please enter a valid email' }
11   }
12
13   redirect('/dashboard')
14 }
```

ال **state** الي راجعه

```
app/ui/signup.tsx
TypeScript

'use client'

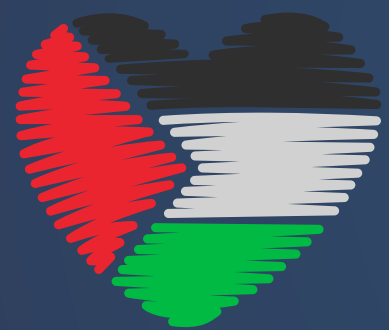
import { useActionState } from 'react'
import { createUser } from '@app/actions'

const initialState = {
  message: '',
}

export function Signup() {
  const [state, formAction, pending] = useActionState(createUser, initialState)

  return (
    <form action={formAction}>
      <label htmlFor="email">Email</label>
      <input type="text" id="email" name="email" required />
      { /* ... */ }
      <p aria-live="polite">{state?.message}</p>
      <button disabled={pending}>Sign up</button>
    </form>
  )
}
```

اللهم انصر اخواننا المستضعفين في فلسطين وفي جميع



بلاد المسلمين