> Menu

App Router > ... > Components > <Form>

<Form> 2

 \mathcal{A} This API is currently in the canary channel and not yet available in a stable version.

The <Form> component extends the HTML <form> element to provide **prefetching** of loading UI, **client-side navigation** on submission, and **progressive enhancement**.

It's useful for forms that update URL search params as it reduces the boilerplate code needed to achieve the above.

Basic usage:

```
Ts /app/page.tsx
                                                                      TypeScript ∨
 1
     import Form from 'next/form'
 2
 3
    export default function Page() {
       return (
 4
         <Form action="/search">
 5
           \{/*\ On submission, the input value will be appended to
 6
 7
               the URL, e.g. /search?query=abc */}
           <input name="query" />
 8
 9
           <button type="submit">Submit
         </Form>
10
11
       )
12
     }
```

Reference

The behavior of the <Form> component depends on whether the action prop is passed a string or function.

- When action is a **string**, the <Form> behaves like a native HTML form that uses a GET method. The form data is encoded into the URL as search params, and when the form is submitted, it navigates to the specified URL. In addition, Next.js:
 - Prefetches the path when the form becomes visible, this preloads shared UI (e.g. layout.js and loading.js), resulting in faster navigation.
 - Performs a client-side navigation instead of a full page reload when the form is submitted. This retains shared UI and client-side state.
- When action is a **function** (Server Action), <Form> behaves like a React form →, executing the action when the form is submitted.

action (string) Props

When action is a string, the <Form> component supports the following props:

Prop	Example	Туре	Required
action	action="/search"	string (URL or relative path)	Yes
replace	replace={false}	boolean	-
scroll	scroll={true}	boolean	-

- action: The URL or path to navigate to when the form is submitted.
 - An empty string "" will navigate to the same route with updated search params.
- (scroll): Controls the scroll behavior during navigation. Defaults to (true), this means it will scroll to the top of the new route, and maintain the scroll position for backwards and forwards navigation.

- replace: Replaces the current history state instead of pushing a new one to the browser's history stack. Default is false.

action (function) Props

When action is a function, the <Form> component supports the following prop:

Prop	Example	Туре	Required
action	action={myAction}	function (Server Action)	Yes

- action: The Server Action to be called when the form is submitted. See the React docs of for more.

Good to know: When action is a function, the replace and scroll props are ignored.

Caveats

- onSubmit: Can be used to handle form submission logic. However, calling event.preventDefault() will override <Form> behavior such as navigating to the specified URL.
- **formAction**: Can be used in a fields to override the action prop. Next.js will perform a client-side navigation, however, this approach doesn't support prefetching.
 - When using basePath, you must also include it in the formAction path. e.g. formAction="/base-path/search".
- method 7, encType 7, target 7: Are not supported as they override <Form> behavior.
 - Similarly, formMethod, formEncType, and formTarget can be used to override the method, encType, and target props respectively, and using them will fallback to native browser behavior.
 - If you need to use these props, use the HTML <form> element instead.

- key: Passing a key prop to a string action is not supported. If you'd like to trigger a rerender or perform a mutation, consider using a function action instead.
- <input type="file">: Using this input type when the action is a string will match browser behavior by submitting the filename instead of the file object.

Examples

Search form that leads to a search result page

You can create a search form that navigates to a search results page by passing the path as an action:

```
/app/page.tsx
                                                                   TypeScript ∨
                                                                                 import Form from 'next/form'
 1
 2
    export default function Page() {
 4
      return (
        <Form action="/search">
 5
          <input name="query" />
 7
          <button type="submit">Submit
 8
        </Form>
      )
 9
10
```

When the user updates the query input field and submits the form, the form data will be encoded into the URL as search params, e.g. /search?query=abc.

```
Good to know: If you pass an empty string to action, the form will navigate to the same route with updated search params.
```

On the results page, you can access the query using the searchParams page.js prop and use it to fetch data from an external source.



```
import { getSearchResults } from '@/lib/search'

2
```

▲ / **NEXT**.Js

Q

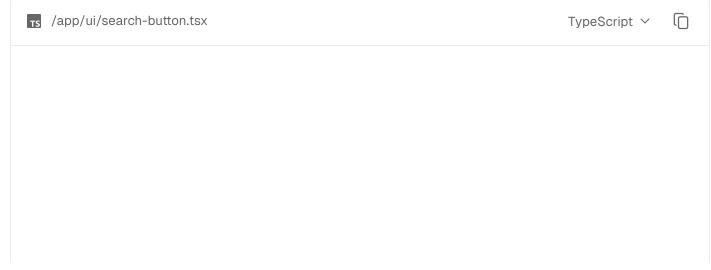
```
7  }) {
8   const results = await getSearchResults(searchParams.query)
9
10   return <div>...</div>
11 }
```

When the <Form> becomes visible in the user's viewport, shared UI (such as layout.js and loading.js) on the /search page will be prefetched. On submission, the form will immediately navigate to the new route and show loading UI while the results are being fetched. You can design the fallback UI using loading.js:

```
1 export default function Loading() {
2 return <div>Loading...</div>
3 }
```

To cover cases when shared UI hasn't yet loaded, you can show instant feedback to the user using useFormStatus.

First, create a component that displays a loading state when the form is pending:



Then, update the search form page to use the SearchButton component:

```
Ts /app/page.tsx
                                                                                  TypeScript ~
 1 import Form from 'next/form'
    import { SearchButton } from '@/ui/search-button'
 3
 4 export default function Page() {
     return (
 6
        <Form action="/search">
           <input name="query" />
 7
           <SearchButton />
 8
 9
        </Form>
10
      )
11
```

Mutations with Server Actions

You can perform mutations by passing a function to the action prop.

```
/app/posts/create/page.tsx
                                                                     TypeScript ∨
                                                                                  import Form from 'next/form'
 1
    import { createPost } from '@/posts/actions'
 4
   export default function Page() {
 5
     return (
 6
        <Form action={createPost}>
 7
          <input name="title" />
 8
          {/* ... */}
          <button type="submit">Create Post</button>
```

```
10 </Form>
11 )
12 }
```

After a mutation, it's common to redirect to the new resource. You can use the redirect function from next/navigation to navigate to the new post page.

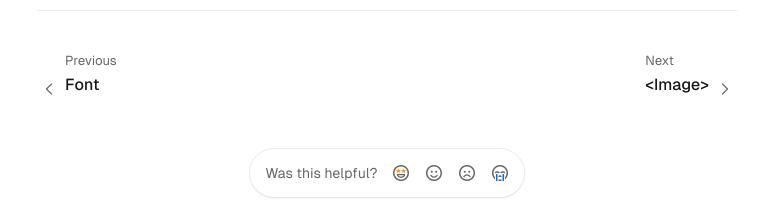
```
/app/posts/actions.ts
                                                                     TypeScript ∨
                                                                                  'use server'
    import { redirect } from 'next/navigation'
 3
 4
    export async function createPost(formData: FormData) {
 5
     // Create a new post
     // ...
 6
 7
 8
      // Redirect to the new post
 9
      redirect(`/posts/${data.id}`)
10
   }
```

Then, in the new page, you can fetch data using the params prop:



```
import { getPost } from '@/posts/data'
 1
 2
 3
    export default async function PostPage({ params }: { params: { id: string } }) {
      const data = await getPost(params.id)
 4
 6
      return (
 7
        <div>
          <h1>{data.title}</h1>
 9
          {/* ... */}
10
        </div>
      )
11
12
    }
```

See the Server Actions docs for more examples.



▲Vercel	Resources	More	About Vercel	Legal
	Docs	Next.js Commerce	Next.js + Vercel	Privacy Policy
	Learn	Contact Sales	Open Source Software	
	Showcase	GitHub	GitHub	
	Blog	Releases	Χ	
	Analytics	Telemetry		
	Next.js Conf	Governance		
	Previews			

Subscribe to our newsletter

Components: <Form> | Next.js

Stay updated on new releases and features, guides, and case studies.

you@domain.com

Subscribe

© 2024 Vercel, Inc.





