**Create a modal that can be opened from anywhere in Next.js 13**

[[Anastasiia Pirus](https://medium.com/@a.pirus?source=post_page-----36f6d0ce1bcf--------------------------------)](https://medium.com/@a.pirus?source=post_page-----36f6d0ce1bcf--------------------------------)

[Anastasiia Pirus](https://medium.com/@a.pirus?source=post_page-----36f6d0ce1bcf--------------------------------)

·

Follow

3 min read

·

6 days ago

24

1

Discover how to seamlessly integrate a customizable modal component into your Next.js application



**1. Create new Next.js application.**

Begin by tidying up the ‘page.tsx’ and ‘globals.css’.

**2. Create the Modal Component.**

In the ‘app’ folder, establish a ‘components’ folder and add a ‘Modal.tsx’ file.

function Modal() {  
 return (  
 <>  
 <div  
 className="fixed left-0 top-0 w-full h-full bg-black bg-opacity-50 z-50 overflow-auto backdrop-blur flex justify-center items-center">  
 <div className="bg-white m-auto p-8">  
 <div className="flex flex-col items-center">  
 <h3>Modal content</h3>  
 <br/>  
 <button type="button" className="bg-red-500 text-white p-2 ">Close Modal</button>  
 </div>  
  
 </div>  
 </div>  
 </>  
 );  
}  
  
export default Modal;

This is a simple modal component. We’ve added ‘Close Modal’ button, but it’s not doing anything yet.

**3. Implement the Modal in the Layout**

Integrate the Modal component into the ‘layout.tsx’ to visualize its appearance across the application.

import './globals.css'  
import type {Metadata} from 'next'  
import {Inter} from 'next/font/google'  
import Modal from "./components/Modal";  
  
const inter = Inter({subsets: ['latin']})  
  
export const metadata: Metadata = {  
 title: 'Create Next App',  
 description: 'Generated by create next app',  
}  
  
export default function RootLayout({  
 children,  
 }: {  
 children: React.ReactNode  
}) {  
 return (  
 <html lang="en">  
 <body className={inter.className}>  
 {children}  
 <Modal/>  
 </body>  
 </html>  
 )  
}

**4. Now, let’s get to the part for which you clicked on the tutorial: how to implement a logic for opening/closing the**

modal.

I suggest creating a query parameter, that will indicate that modal should be open.

In our modal functional component we use ‘useSearchParams()’ hook to check there is a query in out pat and then create a condition to show a modal only when it is true. Don’t forget to add “use client” at the top of the page to be able to use the hook.

To close modal we need to delete ‘modal’ query parameter from url. Let’s do it by using ‘usePathname’ hook.

"use client";  
import {useSearchParams, usePathname} from "next/navigation";  
import Link from "next/link";  
  
function Modal() {  
 const searchParams = useSearchParams();  
 const modal = searchParams.get("modal");  
 const pathname = usePathname();  
  
 return (  
 <>  
 {modal &&  
 <div  
 className="fixed left-0 top-0 w-full h-full bg-black bg-opacity-50 z-50 overflow-auto backdrop-blur flex justify-center items-center">  
 <div className="bg-white m-auto p-8">  
 <div className="flex flex-col items-center">  
 <p>Modal content</p>  
 <br/>  
 <Link href={pathname}>  
 <button type="button" className="bg-red-500 text-white p-2">Close Modal</button>  
 </Link>  
 </div>  
 </div>  
 </div>  
 }  
 </>  
 );  
}  
  
export default Modal;

**5. Opening the Modal**

To open the modal, append modal=true to the query parameters. Use a Link component with href=”?modal=true” to wrap the triggering element (button or other components).

import Link from "next/link";  
  
export default function Home() {  
 return (  
 <>  
 <div className="p-4">  
 <p>Homepage</p>  
 <Link href="?modal=true">  
 <button type="button" className="bg-blue-500 text-white p-2">Open Modal</button>  
 </Link>  
 </div>  
 </>  
 )  
}

Now you can add this query parameters at any page or component on your website! If you want it to work only on a certain page — add ‘’ on the desired page instead of global ‘layout.tsx’.

**P.S.** If you get a warning “Entire page / deopted into client-side rendering” — here is a simple solution for you: wrap your <Modal /> component in Suspense boundary.

import {Suspense} from "react";  
  
...  
<Suspense fallback={<>Loading...</>}>  
 <Modal/>  
</Suspense>  
...



Thank you for going through my tutorial. I really hope it was helpful for you!

You can find the source code for this project on [GitHub](https://github.com/AnastasiiaPirus/source-code-for-blogs/tree/main/modal-tutorial).

Feel free to connect with me on [LinkedIn](https://www.linkedin.com/in/anastasia-pirus/) or [GitHub](https://github.com/AnastasiiaPirus)! Check out my website, I’m going to post more tutorials [here](https://www.pirus.dev/).