ChatGPT

React 19 Migration and Adoption Guide

Overview

React 19 introduces **Server Components** and **Actions** to simplify data fetching and user interactions. It also adds a suite of new hooks (`useActionState`, `useFormStatus`, `useOptimistic`, `use`, `useDeferredValue`) that improve asynchronous state management and user experience. Server Components allow code to run on the server and transmit rendered HTML to the client, improving initial load times, SEO and security [1]. Actions enable functions to be declared alongside components and executed on the server or client, passing `FormData` directly without custom API routes [2].

Step-by-Step Migration

1. **Upgrade dependencies**:
2. Update `react` and `react-dom` to `^19.0.0`.
3. If using Next.js, upgrade to **Next.js 15** and run the codemod: `npx next-codemod update-react-19` [3].
4. **Refactor route handlers to async** (Next.js 15):
5. Convert GET/PUT/HEAD/PATCH handlers into `async` functions to enable streaming and caching control.
6. **Introduce Server Components**:
7. Rename client components to `*.client.tsx` and server components to `*.server.tsx`.
8. Move data fetching and heavy logic into server components; export them from your `app` directory.
9. **Implement Actions**:
10. Define an async function in a server component and annotate with `export async function actionName(formData: FormData)`.
11. Use `<form action={actionName}>` or the `next/form` component to tie form submission to the action [2].
12. **Adopt new hooks**:
13. Use `useActionState` to handle pending state and result of an action [4].
14. Use `useFormStatus` within nested components to access a parent form's pending state [5].
15. Use `useOptimistic` to provide immediate UI feedback while awaiting server response [6].
16. Use `useDeferredValue` to defer non-urgent updates for smoother rendering [7].

17. **Define Suspense boundaries**:

18. Wrap components that read promises with `<Suspense>` and provide fallback UI to handle loading states properly.

Common Pitfalls and Solutions

- **Missing Suspense boundaries**: Without `Suspense`, your app may throw hydration errors. Wrap asynchronous components appropriately.
- **Server/Client mismatch**: Keep browser-only APIs in client components. Server components cannot access `window` or `localStorage`.
- **Overusing `use`**: The experimental `use` function allows reading promises inside render but is intended for library authors [8]. Prefer `await` in server components.

Code Example

```
// app/profile/page.tsx – Server Component
import { getUser } from '../lib/actions';
export default async function Profile() {
  const user = await getUser();
  return <ProfileView user={user} />;
}

// app/profile/ProfileView.client.tsx – Client Component
export function ProfileView({ user }) {
  const [updateUser, { pending, data }] = useActionState(updateUserAction);
  return (
    <form action={updateUser}>
      <input name="name" defaultValue={user.name} />
      <button disabled={pending}>Save</button>
    </form>
  );
}
```

Success Metrics

- Reduced bundle size and faster Time-to-First-Byte (TTFB).
- Fewer API routes and simplified data fetching logic.
- Improved SEO scores due to server-rendered HTML.

Sources

- GeeksforGeeks summary of React 19 features [1] [2].
- Next.js 15 release notes [3].

---

[1] [2] [4] [5] [6] [7] [8] React 19 : New Features and Updates - GeeksforGeeks
https://www.geeksforgeeks.org/reactjs/react-19-new-features-and-updates/

3 Next.js 15 | Next.js
https://nextjs.org/blog/next-15