Next.js — "use client" এবং "use server"

সংক্ষেপে: Next.js (app/ রাউটারে) **Server Components ডিফল্ট** — অর্থাৎ কোডটা সার্ভারে রান করে HTML রেন্ডার করে পাঠাবে। যখন ব্রাউজারে **ইন্টারঅ্যাকশন** লাগে (state, event handler, DOM API ইত্যাদি) তখন সেই ফাইলকে ক্লায়েন্ট-কম্পোনেন্ট হিসেবে চিহ্নিত করতে হয় "use client" দিয়ে। আর **সার্ভারে রান করা বিশেষ ফাংশন/মিউটেশন** (Server Actions) তৈরি করতে হয় "use server" দিয়ে। Next.js

1) "use client" — কি, কোথায় রাখবেন, কেন দরকার

- **কি**: ফাইলের সবচেয়ে উপরে (import এর আগেই) 'use client' লিখলে সেটি সেই ফাইলকে **Client Component** হিসেবে দেখায়। এর মানে: এই ফাইল এবং এর সব import/child কম্পোনেন্টগুলো ক্লায়েন্ট(bundle)+ব্রাউজারে যাবে। Next.js
- **কেন**: ক্লায়েন্ট-সাইড ফিচার দরকার হলে — useState, useEffect, ব্রাউজার API (window, localStorage), ইন্টারঅ্যাকশন (onClick), কোন থার্ড-পার্টি লাইব্রেরি যা DOM প্রয়োজন — এসবের জন্য। Next.js
- **নিয়ম**: 'use client' *সবসময়* ফাইলের টপ-লাইনে থাকবে, import-এর আগে। একবার ফাইল ক্লায়েন্ট হলে তার সব child import-ও ক্লায়েন্ট হিসেবে বানানো হয় — তাই অযাচিতভাবে বড় অংশ ক্লায়েন্ট বানাবেন না (bundle বড়ে যায়)। Next.js

ছোট উদাহরণ (Client)

```
// app/components/Counter.jsx
'use client'            // <-- অবশ্যই ফাইলের প্রথম লাইন
import { useState } from 'react'

export default function Counter() {
  const [count, setCount] = useState(0) // ক্লায়েন্ট-সাইড state
  return <button onClick={() => setCount(c => c + 1)}>Count: {count}</button>
}
```

লাইন-বাই-লাইন:

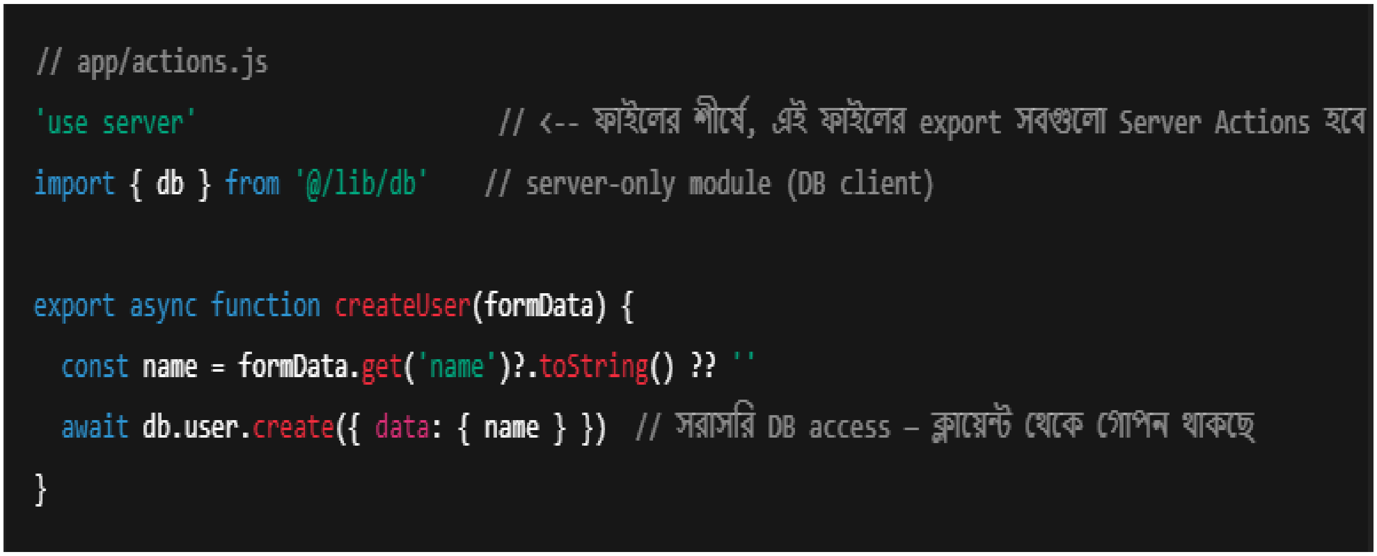
- **'use client' → এই ফাইলের JS ব্রাউজারে যাবে।**
- **useState ব্যবহার করা যায় (Server Component-এ কাজ করবে না)।**
- **onClick ইভেন্ট ব্রাউজারে চলে — তাই ক্লায়েন্ট কম্পোনেন্ট লাগবে।**

পরামর্শ: যতটা সম্ভব *server components* রাখুন (কম JS, দ্রুত initial render)। ক্লায়েন্ট-কম্পোনেন্ট শুধু যেখানে দরকার তাই ব্যবহার করুন। Next.js

2) "use server" — কি এবং কখন ব্যবহার করবেন

- **কি**: React/Next-এর Server Actions (server-side functions) চিহ্নিত করতে 'use server' ব্যবহার করা হয়। এটা ফাংশনের ভিতরে (inline) বা ফাইলের উপরে (module-level) রাখা যায় — module-level এ রাখলে ওই ফাইলের export গুলো সার্ভার-অ্যাকশন হিসেবে গণ্য হবে। Next.js
- **কেন**: সার্ভার-সাইড কাজ (DB access, secrets, ফাইল সিস্টেম, সেনসিটিভ লজিক) ক্লায়েন্টে না পাঠিয়ে সরাসরি সার্ভারে রেখে কল করতে চান — তখন server action বানান। ক্লায়েন্ট থেকে এই action গুলো form action={myAction} বা bind/startTransition ইত্যাদি দিয়ে কল করা যায়। Next.js+1

উদাহরণ — module-level server action (actions.js / actions.ts)



```
// app/actions.js
'use server'                    // <-- ফাইলের শীর্ষে, এই ফাইলের export সবগুলো Server Actions হবে
import { db } from '@/lib/db'    // server-only module (DB client)

export async function createUser(formData) {
  const name = formData.get('name')?.toString() ?? ''
  await db.user.create({ data: { name } })  // সরাসরি DB access – ক্লায়েন্ট থেকে গোপন থাকছে
}
```

এবং ক্লায়েন্ট কম্পোনেন্ট থেকে কল:

```
// app/components/CreateUserForm.jsx
'use client'
import { createUser } from '@/app/actions'

export default function CreateUserForm() {
  return (
    <form action={createUser}>
      <input name="name" />
      <button type="submit">Create</button>
    </form>
  )
}
```

নোট:

- Client Component থেকে module-level server action import করে ব্যবহার করা যায় — bundler সেটাকে সার্ভার-এ অনুবাদ করে POST-এন্ডপয়েন্ট বানায়। Next.js+1
- Server action-এ যেসব আর্গুমেন্ট/রিটার্ন পাঠাবেন/পাবেন তা **serializable** হতে হবে (JSON-র মতো)। ফাংশন বা DOM nodes ইত্যাদি নয়। react.dev

3) সচেতনতা / সীমাবদ্ধতা (important practical rules)

1. **Props serializable থাকতে হবে**: Server → Client কোনো প্রপস পাঠালে সেটা সিম্পল JSON-serializable হওয়া উচিত। ফাংশন/বড় ক্লাস/Date/DOM nodes পাস করলে সমস্যা হবে। (Server Actions হলো কিভাবে ফাংশন-টাই পাস করা যায় তার বিশেষ কৌশল)। plasmic.app+1
2. **Functions cannot be passed normally**: সার্ভার-কম্পোনেন্ট থেকে ক্লায়েন্ট-কম্পোনেন্টে সরাসরি ফাংশন পাস করলে error। যদি সেটা সার্ভার-অ্যাকশন হয়

এবং আপনি সেটা use server দিয়ে এক্সপোজ করেন, তখন bundler সেই ফাংশনকে বিশেষভাবে হ্যান্ডল করে। GitHub+1

3. **Performance**: একটি ফাইলে 'use client' দিলে পুরো তার import graph ক্লায়েন্টে যাবে — ফলে bundle বড় হতে পারে। শুধুমাত্র যেখানে দরকার সেখানে ছোট ছোট client components বানান। Next.js
4. **Secrets & DB**: API keys, DB clients কখনই ক্লায়েন্টে পাঠাবেন না — সার্ভার-কম্পোনেন্ট বা server action এ রাখুন। Client থেকে সেই কাজ করতে চাইলে server action ব্যবহার করুন। react.dev
5. **Progressive enhancement**: Server Action ব্যবহার করা হলে <form action={myAction}> দিয়ে progressive enhancement পাওয়া যায় — JS না থাকলেও ফর্ম সাবমিট হবে, JS থাকলে আরও উন্নত UX পাওয়া যায়। Next.js

4) ছোট একটি অনুশীলনী — কবে কোনটা দেবেন?

- পেজ/লেআউট/ডেটা-ফেচিং: **Server Component** (ডিফল্ট)। SEO ও দ্রুত initial HTML এর জন্য। Next.js
- যে অংশটা ইউজারের ক্লিক/টাইপ/রিয়েক্টিভ state-নির্ভর: **'use client'**।
- যদি ইউজার-অ্যাকশনের পরে DB update/সিক্রেট-ব্যবহার/redirect করতে চান: **Server Action** ('use server') বানান, ক্লায়েন্টে form/action দিয়ে কল করুন। Next.js

5) দ্রুত রেফারেন্স (বাংলায় — টীকি)

- 'use client' = ফাইলকে ক্লায়েন্ট-কম্পোনেন্ট বানায়; imports/children ক্লায়েন্টে যাবে; প্রয়োজন শুধু interactive UI তে। Next.js+1
- 'use server' = Server Actions / server-only ফাংশন চিহ্নিত করে; module-level করলে client থেকে import করে ব্যবহার করা যায়; আর্গুমেন্ট/রিটার্ন serializable হতে হবে। Next.js+1