1. Data Fetching

- 기존 fetch와 차이점

- 기존 fetch
    - 브라우저에서만 동작
    - 브라우저 HTTP 캐시 사용
    - 네트워크 요청은 항상 클라이언트에서 실행
- Next.js fetch
    - 서버 환경에서 동작
    - Next.js가 요청을 처리해 데이터 캐싱, 성능 최적화, 재검증 기능을 제공한
    - 같은 fetch 코드가 클라이언트 서버 모두 동작

* Next.js 15버전 이후부터는 cache 기본값이 'force-cache'에서 'no-sotre' 변경

- Data fetching pattern

구 분	특 징
Sequential Fetching	데이터간 의존성이 있을 때 사용 (순차적으로 진행)
Parallel Fetching	독립적인 데이터는 동시에 가져오기 (속도 빠름)

2. API 캐싱 전략

옵 션	설 명
cache: 'no-store'	매 요청마다 새 데이터를 가져옴 (캐시 사용 X)
cache: 'force-cache'	가능하면 캐시된 데이터를 사용하고, 없으면 새로 패칭
next: { revalidate: N }	ISR(Incremental Static Regeneration)처럼, 특정 초마다 새로 패칭
next: { tags: [' T '] }	특정 태그로 캐시 제어

1. cache : `fetch(https://..., { cache: 'force-cache' | 'no-store' })`

- auto no cache(default) : cache에 아무 값도 입력하지 않았을 때 적용되는 기본 값
- no-store : Dynamic API를 사용하지 않더라도 모든 요청에 대해 fetch를 진행
- force-cache : Next.js가 데이터 캐시와 일치하는 요청을 탐색
    - 일치하는 데이터 캐시가 있고 fresh하다면 캐시된 데이터를 탐색
    - 일치하는 항목이 없거나, 데이터가 오래되었다면 데이터를 새로 업로드

* 주의 사항 : 개발 환경과 배포 환경에서의 차이 *

(개발 환경에서는 캐시를 사용하지 않고 매번 요청. 단 배포 환경에서는 빌드 시 한 번만 요청을 보내고, 결과를 캐싱. 하지만 Dynamic API가 감지된다면 모든 요청에 대해 fetch를 진행*

2. next.revlidate : `fetch("https://...", { next: { revalidate: false | 0 | number } })`

(revalidate는 캐시의 지속시간을 설정)

- false : 무기한으로 캐시, 오래된 HTTP 캐시는 오래된 리소스 제거 가능
- 0 : 리소스가 캐시되지 않음
- number : 단위는 초 단위로 리소스를 캐시함

3. next.tags : `fetch("http://...", {next: {tags [ ' ' ]}})`

- ' '에 태그가 붙어 해당 태그가 달린 캐시만 갱신 가능

3. Sever Actions

- 기본 개념

- 서버에서 실행되는 비동기 함수
- 서버 컴포넌트와 클라이언트 컴포넌트에서 호출하여 사용 가능
- Form 처리와 data mutation에 사용 가능
- API 호출, 요청 처리, 데이터 베이스 접근 과 같은 과정을 생략
- 클라이언트에서 API가 노출되는 위험 방어(서버 처리후 사용자에게 노출)

- 규칙

- Async 함수로 작성(비동기 처리가 기본이므로)
- 반환 값은 serializable objects(직렬화가 가능해야함)
- 클라이언트를 직접 호출 하는 것이 아닌 Next.js에서 관리하는 방식으로 연결

- 사용방법

1. 'use server' 티렉티브를 함수 파일의 최상단에 추가

2. 아래 두 가지의 방법중 하나의 클라이언트에서 Server Action에 연결

구 분	특 징
Form action	From 전체를 특정 Server Action에 연결
Button formAction	버튼 단위를 각각 다른 Server Action에 戀結

- Form action : `<form action={ServerAction}>...</form>`
- Button formAction : `<button formAction={ServerAction}>버튼</button>`

3. 클라이언트 컴포턴트에서 해당 결과값을 받아 반환

구 분	특 징
useActionState	실시간으로 상태 업데이트
직접 리디렉션 or 캐시 무효화	Server Action내 페이지 이동 혹은 캐시 갱신