Reactivity

Fine & Coarse Grained

Reactivity 란?

Reactivity

Reactivity 란?

데이터가 변경되면 UI또한 자동으로 갱신되는 구조

핵심은
선언적 UI 와 의존성 추적

Reactivity

선언적 UI 와 의존성 추적?

- 선언적 UI
 {count}

- 의존성 추적
<Child count={count} />

그렇다면 UI 를 갱신하는 단위는 어떻게 나눌 수 있을까요?

Reactivity

그렇다면 UI 를 갱신하는 단위는 어떻게 나눌 수 있을까요?

- 컴포넌트단위로 갱신한다

- 값 단위로 갱신한다

Reactivity

그렇다면 UI 를 갱신하는 단위는 어떻게 나눌 수 있을까요?

- 컴포넌트단위로 갱신한다 - Coarse Grained

- 값 단위로 갱신한다 - Fine Grained

Dependencies

Fine Grained & Coarse Grained - Reactivity

왜 Grained 인가?

Dependencies

Fine Grained & Coarse Grained - Reactivity

왜 Grained 인가?

입자의 크기를 비유로 사용

Coarse Grained 는 큰 입자
Fine Grained 는 작은 입자

Dependencies

Fine Grained & Coarse Grained - Reactivity

하나의 캔버스를 전부 덮어씌워 다시 그리느냐

필요한 부분만 섬세하게 수정하느냐의 차이

Dependencies

구분	Coarse-grained	Fine-grained
대표 기술	React, Redux, Zustand	SolidJS, MobX
추적 단위	컴포넌트(함수 호출)	시그널(값)
업데이트 방식	State 변경 → 컴포넌트 전체 재실행	변경된 값이 사용된 DOM만 갱신
장점	구조화된 흐름, 예측 가능성 높음	오버렌더링 최소화 → 최적화 용이
단점	불필요한 렌더링 발생 가능	추적 로직 복잡 → 디버깅 어려움

Dependencies

간단한 예제코드 참고해보기

Dependencies

MobX

Fine Grained Reactivity 의 대표적인 라이브러리

앞선 코드에서 복잡한 시그널 업데이트를 매우 효율적이고
비교적 간단하게 & 객체지향적으로 제공해주는 라이브러리

어떻게보면 React 와는 다른 철학을 가진 라이브러리

Dependencies

- Observable
- Action
- Computed
- autoRun & reaction
- observer

Dependencies

결론

- 거의 대부분의 경우 React 만으로도 충분하다

- 하지만 만약 대규모 + 실시간 + 고성능 UI 렌더링이 필요하다면?
  Fine Grained 는 좋은 선택