📌 React

🚀 1. React 개요

📖 React란?

- **React는 Facebook(현재 Meta)에서 개발한 오픈 소스 JavaScript 라이브러리로, 사용자 인터페이스(UI)를 구축하는 데 사용됨**
- 컴포넌트 기반(Component-Based) 아키텍처를 통해 **재사용 가능한 UI 요소를 만들고 관리할 수 있도록 설계됨**
- **SPA(Single Page Application) 개발을 위한 핵심 라이브러리**로, **클라이언트 측 렌더링(Client-Side Rendering, CSR)을 효율적으로 수행**

✅ React의 주요 특징

1. **컴포넌트 기반 아키텍처** → 재사용 가능한 UI 구성 요소
2. **Virtual DOM (VDOM) 사용** → 성능 최적화
3. **선언형(Declarative) 프로그래밍** → UI 상태 변경을 간결하게 표현 가능
4. **단방향 데이터 흐름 (One-Way Data Binding)** → 유지보수성이 뛰어남
5. **React Hooks 지원** → 클래스형 컴포넌트 없이도 상태 및 라이프사이클 관리 가능
6. **JSX(JavaScript XML) 사용** → HTML과 JavaScript를 조합하여 가독성 향상
7. **대규모 생태계와 강력한 커뮤니티 지원** → Redux, React Router, Next.js 등 다양한 라이브러리 활용 가능

💡 2. React를 왜 사용해야 하는가?

✅ React의 장점

1. **빠른 렌더링 성능** → Virtual DOM을 사용하여 최소한의 DOM 업데이트 수행
2. **재사용성과 유지보수성 향상** → 컴포넌트 기반 구조를 통해 UI 요소를 효율적으로 관리
3. **강력한 커뮤니티와 생태계** → 공식 문서 및 오픈 소스 라이브러리 지원
4. **SEO 친화적인 개발 가능** → SSR(서버 사이드 렌더링, Next.js 활용)로 검색 엔진 최적화 가능
5. **모바일 애플리케이션 개발 지원** → React Native를 활용하면 크로스 플랫폼 앱 개발 가능

✅ React vs 기존 방식 비교

비교 항목	React	jQuery	Vanilla JavaScript
UI 업데이트 방식	Virtual DOM 활용	직접 DOM 조작	직접 DOM 조작
코드 재사용성	✅ 컴포넌트 기반	❌ 코드 중복 발생	❌ 코드 중복 발생
성능 최적화	✅ 효율적인 렌더링	❌ 직접 DOM 조작으로 성능 저하	❌ 직접 DOM 조작으로 성능 저하
상태 관리	✅ useState, Redux 지원	❌ 직접 변수 관리	❌ 직접 변수 관리
확장성	✅ 대규모 프로젝트 적합	❌ 규모가 커질수록 유지보수 어려움	❌ 규모가 커질수록 유지보수 어려움

✅ React는 **SPA 개발에 최적화된 라이브러리로, 기존 방식보다 높은 성능과 유지보수성을 제공**

🔥 3. Virtual DOM (VDOM)과 성능 최적화

📖 Virtual DOM이란?

- **Virtual DOM은 실제 DOM을 조작하기 전에 메모리상에서 미리 변경 사항을 계산하여 최소한의 업데이트를 수행하는 개념**
- **React에서 가장 중요한 성능 최적화 기술 중 하나**

📌 Virtual DOM 동작 방식

1. **UI 변경 발생** → 상태(state) 또는 props 값 변경
2. **새로운 Virtual DOM 생성** → 기존 Virtual DOM과 비교(Diffing Algorithm 사용)
3. **변경된 부분만 실제 DOM에 적용(Reconciliation 과정)** → 성능 최적화

📌 예제 (Virtual DOM 적용 전후 비교)

```
function Counter() {
  const [count, setCount] = useState(0);
  return (
    <div>
      <p>현재 카운트: {count}</p>
      <button onClick={() => setCount(count + 1)}>증가</button>
    </div>
  );
}
```

✅ Virtual DOM을 활용하면 **UI 변경 시 전체 DOM을 다시 그리는 것이 아니라, 변경된 부분만 반영**

🛠 4. React의 핵심 개념

📌 1) 컴포넌트(Component)

- **컴포넌트는 React에서 UI를 구성하는 독립적인 단위**
- 두 가지 유형:
  - **함수형 컴포넌트 (Functional Component, 추천)**
  - **클래스형 컴포넌트 (Class Component, 기존 방식)**

✅ **함수형 컴포넌트 예제**

```
function Greeting({ name }) {
  return <h1>Hello, {name}!</h1>;
}
```

✅ React Hooks(`useState`, `useEffect` 등)과 함께 사용 가능

📌 2) State와 Props

- **State** → 컴포넌트 내부에서 관리하는 동적인 데이터
- **Props** → 부모에서 자식 컴포넌트로 전달하는 데이터 (읽기 전용)

📌 3) React Hooks (useState, useEffect 등)

- **React 16.8부터 클래스형 컴포넌트 없이도 상태 관리 및 라이프사이클 기능 사용 가능**
- 대표적인 Hooks:
  - `useState` → 컴포넌트의 상태 관리

- `useEffect` → 라이프사이클 관리 (데이터 패칭, 이벤트 리스너 등)
- `useContext` → 전역 상태 관리 (Redux 대체 가능)

```
function Counter() {
  const [count, setCount] = useState(0);
  useEffect(() => {
    console.log(`현재 카운트: ${count}`);
  }, [count]);
  return <button onClick={() => setCount(count + 1)}>증가</button>;
}
```

✅ React Hooks를 활용하면 코드가 간결해지고, 유지보수성이 높아짐

🎯 5. React의 발전 방향 및 활용

📌 최신 트렌드

1. **React Server Components** → 서버에서 렌더링 최적화
2. **Next.js** → SEO 최적화 및 성능 향상을 위한 프레임워크
3. **React Concurrent Mode** → 렌더링 성능 향상을 위한 최신 기술
4. **React Suspense** → 비동기 데이터 로딩 최적화

📌 React의 활용 분야

✅ **웹 애플리케이션 개발** → SPA, 대규모 프로젝트

✅ **모바일 애플리케이션 개발** → React Native 활용

✅ **SEO 최적화 프로젝트** → Next.js 기반 SSR 적용

✅ **대시보드, 실시간 애플리케이션** → 성능 최적화된 UI 구성 가능

🏁 6. 적용

- **React는 현대적인 웹 애플리케이션 개발을 위한 강력한 라이브러리**
- **컴포넌트 기반 개발, Virtual DOM, React Hooks 등의 개념을 이해하면 더 효율적인 개발 가능**
- **Next.js, React Server Components, Suspense 등 최신 기술을 학습할 수 있음**