**1. 서버사이드 렌더링이란 무엇인가**

- SSR (Server Side Rendering) : 서버에서 렌더링을 작업합니다. 기존에 존재하던 방식으로 사용자가 웹페이지에 접근할 때, 서버에 페이지에 대한 요청을 하며, 서버에서는 Html, View 와 같은 리소스들을 어떻게 보여질지 해석하고 렌더링하여 사용자에게 반환 합니다.

웹에서 제공하는 정보량이 많아지고, 데스크탑보다 성능이 다소 떨어지는 스마트폰의 웹에 대한 요구가 커지면서 새로운 기법이 탄생했습니다.

**2. SPA란 무엇이고 어떤 장점이 있는가.**

Single Page Appilcation : 서버로부터 완전한 새로운 페이지를 불러오지 않고 현재의 페이지를 동적으로 다시 작성함으로써 사용자와 소통하는 웹 애플리케이션이나 웹사이트를 말한다. 이러한 접근은 연속되는 페이지들 간의 사용자 경험의 간섭을 막아주고 애플리케이션이 더 데스크톱 애플리케이션처럼 동작하도록 만들어준다.

SPA에서 HTML, 자바스크립트, CSS 등 필요한 모든 코드는 하나의 페이지로 불러오거나,

적절한 자원들을 동적으로 불러들여서 필요하면 문서에 추가하는데, 보통 사용자의 동작에 응답하게 되는 방식이다.

**3. 리액트에 사용되는 디자인패턴은 어떤 것들이 있는가. 각각 설명하시오.**

- 디자인패턴 이란: 디자인 패턴은 다양한 상황에서 코드에 적용할 수 있는 잘 정의된 솔루션이다.

디자인 패턴은 모든 언어에 적용된다. 즉 작성해야 할 코드가 아니라 어떤 클래스/객체를 사용할 지와 서로가 영향을 어떻게 주는 지에 대한 일종의 개요 같은 것이다.

디자인 패턴은 다음 세 가지 유형이 있다.

창조적 패턴: 객체 생성에 대한 유연성을 제공한다.

구조적 패턴: 클래스와 객체를 구조화 하는데 도움을 준다. (어디에 속하는 지, 객체가 어떻게 다른 객체에 접근할 수 있는지)

행위적 패턴: 객체 간의 커뮤니케이션과 관련되어 있다.

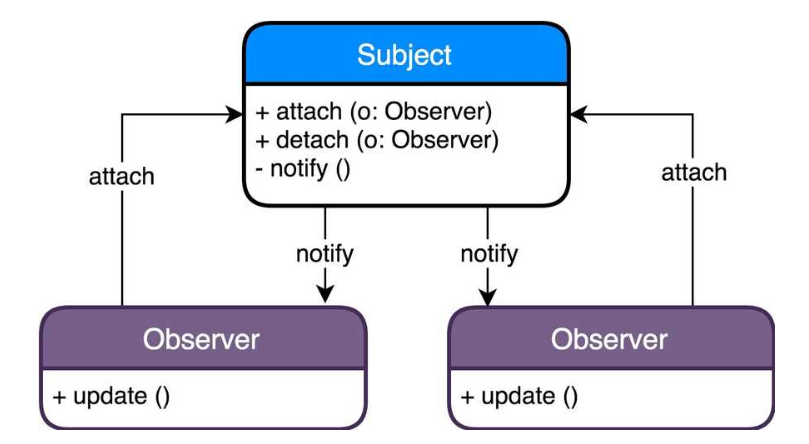
**- 리액트에 사용되는 디자인패턴**

Observer 패턴 : 하나의 상태에 의존하는 여러 개의 객체들이 존재하는 케이스에서 유용하다.

React에서는, 서로 연결되진 않았지만 데이터가 동기화 되어 있으면 하는 두 컴포넌트가 존재할 때 유용하다.

동작 방식은 먼저 Observer라고 불리는 것이 자신을 Subject에 attach 한 후 계속 기다린다.

그리고 Subject가 변경되면 자신의 최신 데이터를 attach 된 Observer 들을 한테 notify 한다.



Provider 패턴 : .Net Microsoft 앱 내에서 클래스의 초기화에 사용되는 다양한 조미료(React에선 props)를 제공하기 위해 처음 사용되었다. 이 패턴은 컴포넌트들이 ‘전역’ 상태를 가질 수 있게 하는 React Context API에 유용하게 사용되었다.

다수의 컴포넌트에서 사용가능한 일반적인 객체를 만들기를 원하고 그 객체가 변할때 마다 자식 컴포넌트들이 업데이트 되길 원할 때 유용하다.

이 패턴의 주요한 이점은 트리 내의 컴포넌트마다 각각 다 props로 값을 넘겨야 하는 상황을 피할 수 있다는 것이다. 이 문제는 보통 “Prop Drilling”으로 잘 알려져 있다.

동작 방식은 먼저 Provider가 컴포넌트 트리의 상위 레벨에 설정된 Context 객체에 일부 값들을 설정한다. Consumer라고 불리는 자식 컴포넌트들은 props로 값들을 전달받는 것 대신, context로부터 값들을 직접 가져올 수 있다.

