< React >

변경 사항이 있을 때 dom을 바로 생성하지 않고, virtual Dom을 생성해서 기존의 virtual Dom과 비교 후 변경된 부분만 realdom에 반영함.

< React hook >

1. React hook 이란?

- 함수 컴포넌트에서 react state 와 생명주기 기능(lifecycle features)을 연동(hook into) 할수 있게 해주는 함수

2. Hook 의 규칙이란?

- 최상위(at the Top Level) 에서만 Hook을 호출 해야함

  반복문, 조건문 혹은 중첩된 함수 내에서 Hook 을 호출하면 안됨.

- 오직 React 함수 내에서 Hook을 호출 해야함

  일반적인 Javascript 에서 호출 하면 안됨

  React 함수형 컴포넌트 또는 Custom Hook에서 호출 할 수 있음

이 두가지 규칙을 강제하는 eslint-plugin-react-hooks 라는 ESLint 플러그인이 있는데 `Create React App` 에 기본적으로 포함되어 있다.

```
// ESLint 설정 파일
{
  "plugins": [
    // ...
    "react-hooks"
  ],
  "rules": {
    // ...
    "react-hooks/rules-of-hooks": "error", // Checks rules of Hooks
```

```
    "react-hooks/exhaustive-deps": "warn" // Checks effect dependencies
  }
}
```

< useState >

1. useState 란?

- 함수형 컴포넌트에서 상태 관리를 위해 사용되며, 상태 유지값과 그 값을 갱신하는 함수를 반환한다. (=getter, setter)

- state -> 동적인 데이터를 다루기 위한 객체로 컴포넌트 내부에서 변경 될 수 있는 값.

2. useState 사용법

- 리액트 패키지에서 useState 불러오기

  Import React, {useState} from 'react';

- ES6 문법 비구조화 할당

  Const [상태 값 저장 변수, 상태 값 갱신 함수] = useState(상태 초기 값);

- State 변경 -> 리렌더링 -> state 반영

3. useState 규칙

- batch updating

  state 가 변경되면 state의 변경사항을 즉시 반영하지 않고, 대기열 (Queue)에 넣은 후 한꺼번에 적용한다.

setState는 변경된 사항을 기억하지 않기 때문에 마지막 업데이트가 상태 값에 적용되어 렌더링에 쓰이게 된다.

- State 불변성

  메모리 영역의 값을 변경할 수 없는 것.

  상태 값을 업데이트 할 때 state 값을 직접적으로 변경하지 않고, 기존의 값의 사본을 만든 후 원하는 값으로 변형 후 setState를 이용하여 새로운 값을 부여함.

- 얕은 비교 -> 객체의 참조 주소값만 변경 되었는지 확인.

  불변성을 지키지 않으면 state가 바뀌어도 그 state 객체를 참조하는 참조값은 바뀌지 않기 때문에 리액트는 값을 인지하지 못하고 리렌더링을 하지 않는 문제가 발생한다.

4. virual Dom 과 연관지어서 이해

state 값을 갱신 할 때 여러 번 렌더링 하지 않고 virtualDom 을 생성 하여 업데이트 된 부분을 비교한 후 한번에 렌더링 한다.

< useEffect >

1. useEffect 란?

- 함수형 컴포넌트에서 useEffect 를 통해 라이프사이클 메서드를 대체한다.

  useEffect는 Side-Effect를 처리하기 위해 사용한다. Side-Effect의 대표적인 예로 비동기 방식으로 api 호출하여 데이터 가져오기, 구독(subscription) 설정, 수동으로 DOM 조작, 타이머 설정, 로깅 등이 있다.

- React 에서 Side-Effect 란?

  리액트 컴포넌트가 화면에 렌더링 된 이후에 비동기적으로 처리되어야 하는 부수적인 효과들을 뜻한다. 요청 즉시 1차 렌더링을 함으로써 연동하는 API가 응답이 늦어지거나 응답이 없을 경우에도 영향을 최소화 시킬 수 있어서 사용자 경험 측면에서 유리하다.

2. useEffect 사용법

- useEffect(callBackFunc, dependencies) 두개의 인자를 넣어서 호출함
  callBackFunc: 수행하고자 하는 작업
  dependencies: 배열 형태이며, 배열 안에 검사하고자 하는 특정 값 or 빈 배열을 넣어준다.
  dependencies에 특정 값을 넣게 되면 컴포넌트가 처음 마운트 될 때 지정한 값이 바뀔 때, 언마운트될때 값이 바뀌기 직전에 모두 호출된다.

  만약 파라미터를 생략하면, 컴포넌트가 리렌더링 될 때마다 useEffect 함수가 호출된다.

- useEffect(callBackFunc, [deps1, deps2]);
  최초 렌더링 됐을 때, dep1 또는 dep2가 변경되었을 때 실행된다.

- useEffect(callBackFunc, []);
  최초 렌더링 됐을 때 한번 실행한다.

- useEffect clean-up 함수
  clean-up 함수를 활용해 컴포넌트가 Unmount 될 때 정리 해야 할 것을 처리한다.
  clean-up 함수는 Effect가 실행되기 전에 매번 호출 되며 (update), 컴포넌트가 unmount 될 때 역시 실행된다.