24.04.19

오후 4시 318호 2시간

Hook

Hook 등장 배경

- 리액트 컴포넌트는 함수형 컴포넌트(Functional Component)와 클래스형 컴포넌트(Class Component)로 나뉜다.
- 리액트 초기에는 일반적으로 함수형 컴포넌트(Functional Component)를 사용하였으나, 값의 상태를 관리(state) 혹은 Lice Cycle Method(생명 주기=컴포넌트가 생성되고 사라지는 과정이 존재 할 때)를 사용하여야 할 때에만 클래스형 컴포넌트(Class Component)를 사용하였다.
- 함수형 컴포넌트(Functional Component)가 사용된 이유는 아래와 같은 클래스형 컴포넌트(Class Component)의 대표적인 단점 때문이였다.
  - ✓ 코드의 구성이 어렵고 Component의 재사용성이 떨어진다.
  - ✓ 컴파일 단계에서 코드 최적화를 어렵게 한다.
  - ✓ 최신 기술의 적용이 효과적이지 않다.
- 이러한 클래스형 컴포넌트(Class Component)의 단점을 보완하여, 함수형 컴포넌트(Functional Component)를 사용 할 수 있도록 등장한 것이 바로 React Hook(리액트 훅)이다

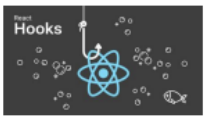
JS에서 클래스 만드는법 알기

Hook

Hook 개요

- React Hook 이란, 리액트 v16.8에 새로 도입된 기능으로, 함수형 컴포넌트(Functional Component)에서 사용되는 몇가지 기술들을 일컫는다.
- 함수형 컴포넌트(Functional Component)가 클래스형 컴포넌트(Class Component)의기능을 사용 할 수 있도록 해주며 대표적인 예로는 useState, useEffect 등이 존재한다.

- Class
  - 클래스 컴포넌트에서는 생성자에서 state를 정의하고 setState함수를 통해 state를 업데이트하게 된다.
- Function
  - 기존 함수 컴포넌트는 이러한 state를 정의해서 사용하거나 컴포넌트 생명주기에 맞춰 실행되도록 할 수 없기 때문에 나온 것이 바로 훅(Hook)이다.
  - 이러한 훅의 이름은 모두 use로 시작한다.



HOOK-내장함수

이름들이 모두 use로 시작한다

Hook

Hook 장점

- 상태 로직 단순화 :
  - ✓ Hooks를 사용하면 함수형 컴포넌트에 상태를 추가하여 전반적인 로직을 단순화하고 코드를 이해하기 쉽게 만들 수 있다.
- 코드 재사용성과 관심사 분리 :
  - ✓ Hooks를 사용하면 컴포넌트 계층 구조를 변경하지 않고도 여러 컴포넌트 간에 상태 로직을 재사용할 수 있다.
- 사이드 이펙트 감소 :
  - ✓ Hooks는 함수형 컴포넌트에 생명주기 메서드와 유사한 기능을 제공하여 사이드 이펙트를 더 효율적으로 처리할 수 있습니다.

성능 비교 - 클래스형 vs 함수형 React Hook의 어두운면

liked list - 속도가 빠른걸 원하면 사용

Hook

Hook의 규칙

- Hook은 JavaScript 함수입니다. 하지만 Hook을 사용할 때는 두 가지 규칙을 준수해야 한다.
- 우리는 이러한 규칙들을 자동으로 강제하기 위한 linter 플러그인(Create React App에 기본적으로 포함되어 있음)을 제공하고 있다.

- **최상위(at the Top Level)에서만 Hook을 호출해야 한다.**
  반복문, 조건문 혹은 중첩된 함수 내에서 Hook을 호출하면 안됨. 대신 early return이 실행되기 전에 항상 React 함수의 최상위(at the top level)에서 Hook을 호출해야 한다. 이 규칙을 따르면 컴포넌트가 렌더링 될 때마다 항상 동일한 순서로 Hook이 호출되는 것이 보장된다. 이러한 점은 React가 useState 와 useEffect 가 여러 번 호출되는 중에도 Hook의 상태를 올바르게 유지할 수 있도록 해준다.

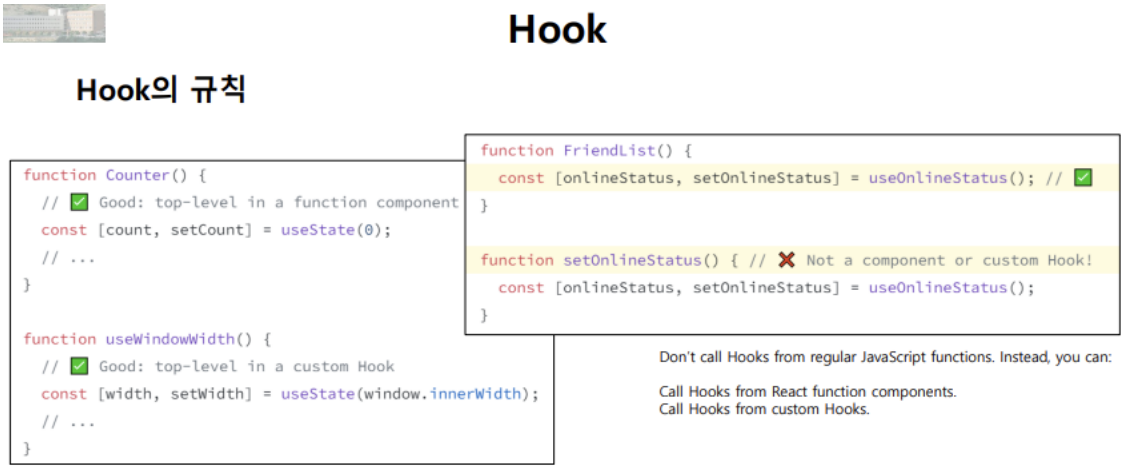
- **오직 React 함수 내에서 Hook을 호출해야 한다.**
  Hook을 일반적인 JavaScript 함수에서 호출하지 마세요. 대신 아래와 같이 호출할 수 있다.
  - ✓ React 함수 컴포넌트에서 Hook을 호출
  - ✓ Custom Hook에서 Hook을 호출

  이 규칙을 지키면 컴포넌트의 모든 상태 관련 로직을 소스코드에서 명확하게 보이도록 할 수 있습니다.

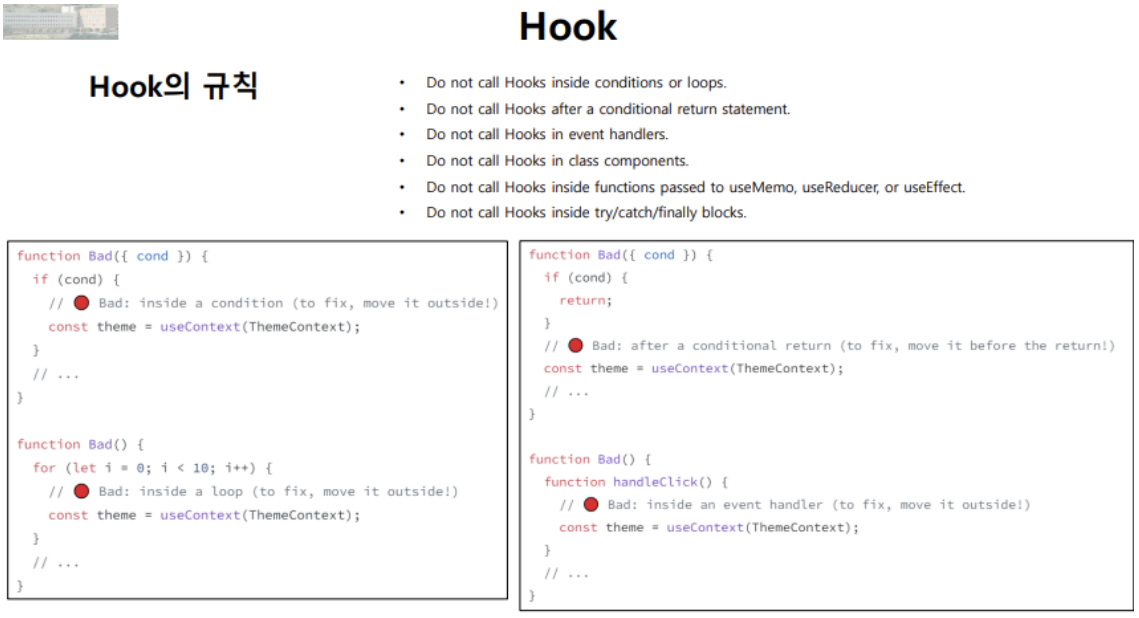
JS 합성함수 - ?

HOOK은 super처럼 제일 위에 적어야한다

리엑트에서 호출해야함 JS에서 호출하면 리엑트 엔진에서 JS로 변환하는 과정을 거쳐야하는데 그게 불가능함



밑에꺼는 컴포넌트가 X 일반함수기 때문에 작성 불가



클래스 컴포넌트에서 사용 X

only function 컴포넌트에서만 사용가능

Hook

Hook의 규칙

```jsx
function Bad() {
  const style = useMemo(() => {
    // 🔴 Bad: inside useMemo (to fix, move it outside!)
    const theme = useContext(ThemeContext);
    return createStyle(theme);
  });
  // ...
}

class Bad extends React.Component {
  render() {
    // 🔴 Bad: inside a class component (to fix, write a function component instead of a class!)
    useEffect(() => {})
    // ...
  }
}
```

```jsx
function Bad() {
  try {
    // 🔴 Bad: inside try/catch/finally block (to fix, move it outside!)
    const [x, setX] = useState(0);
  } catch {
    const [x, setX] = useState(1);
  }
}
```

Hook

Built-in React Hooks

Hook	종류			
State Hooks	useState	useReducer		
Context Hooks	useContext			
Ref Hooks	useRef	useImperativeHandle		
Effect Hooks	useEffect	useLayoutEffect	useInsertionEffect	
Performance Hooks	useMemo	useCallback	useTransition	useDeferredValue
Resource Hooks	use			
Other Hooks	useDebugValue	useId	useSyncExternalStore	
Your own Hooks	custom Hooks			

useState

useState

다시 작업 예정

- 컴포넌트에 state variable를 추가할 수 있다.

```
const [state, setState] = useState(initialState)
```

Reference

- useState(initialState)
  - 상태 변수를 선언하려면 구성 요소의 최상위 수준에서 useState를 호출한다.

```
import { useState } from 'react';

function MyComponent() {
  const [age, setAge] = useState(28);
  const [name, setName] = useState('Taylor');
  const [todos, setTodos] = useState(() => createTodos());
  // ...
```

  - [something, setSomething]과 같은 상태 변수의 이름은 array destructuring을 이용한다.

컴포넌트 내에 있는 변수

컴포넌트 안의 값이 바뀌면 화면도 같이 바뀜 (state)

변수이름 주고 변수를 바꿀 set값도 줘야함

initial State - 초기의 변수값

State value 바뀌는건데 변수가 바뀌면 화면도 같이 바뀌는 것

useState

Reference

- Parameters
  - initialState: 상태의 초기값. 모든 유형의 값이 될 수 있지만 함수에는 특별한 동작이 있다. 이 인수는 초기 렌더링 후에는 무시된다.
  - initialState는 초기화 함수처럼 처리된다. 순수해야 하고, 인수를 취하지 않아야 하며, 모든 유형의 값을 반환해야 한다. React는 컴포넌트를 초기화할 때 초기화 함수를 호출하고 반환 값을 초기 상태로 저장한다.
- Returns
  - useState 두 개의 값이 있는 배열을 반환한다.
    - 현재 상태. 첫 번째 렌더링 중에는 initialState 값이다.
    - Set 함수. 상태를 다른 값으로 다시 업데이트하고, 리랜더링을 트리거할 수 있는 함수이다.
- Caveats
  - useState는 Hook이므로 컴포넌트의 최상위 수준이나 자체 Hook에서만 호출할 수 있다. 루프나 조건 내에서는 호출할 수 없다. 필요한 경우 새 구성 요소를 추출하고 상태를 해당 구성 요소로 옮긴다.
  - Strict 모드에서 React는 실수로 발생한 불순물을 찾는 데 도움을 주기 위해 초기화 함수를 두 번 호출 한다. 이는 개발 전용 동작이며 프로덕션에는 영향을 주지 않는다. 초기화 함수가 순수(순수해야 함)인 경우 이는 동작에 영향을 주지 않는다. 호출 중 하나의 결과는 무시된다.

useState

Reference

- set functions, like setSomething(nextState)
  - useState에 의해 반환되는 set 함수는 상태를 다른 값으로 업데이트하고 리렌더링을 트리거할 수 있다. 다음 상태를 직접 전달하거나 이전 상태에서 이를 계산하는 함수를 전달할 수 있다.

```
const [name, setName] = useState('Edward');

function handleClick() {
  setName('Taylor');
  setAge(a => a + 1);
  // ...
```

- Parameters
  - useState에 의해 반환되는 set 함수는 상태를 다른 값으로 업데이트하고 리렌더링을 트리거할 수 있다. 다음 상태를 직접 전달하거나 이전 상태에서 이를 계산하는 함수를 전달할 수 있다.

에드워드 값이 테일러로 변경됨

useState

Reference

- Parameters
  - nextState: 원하는 상태 값. 모든 유형의 값이 될 수 있지만 함수에는 특별한 동작을 수행한다.
  - nextState를 실행하면 updater function처럼 처리된다. 순수해야 하고 보류중인(pendding) 상태를 유일한 인수로 사용해야 하며 다음 상태를 반환해야 한다. React는 업데이트 기능을 대기열에 넣고 구성 요소를 다시 렌더링한다. 다음 렌더링 동안 React는 대기 중인 모든 업데이트를 이전 상태에 적용하여 다음 상태를 계산한다.
- Returns
  - Set 함수는 리턴 값이 없다.

인수 - 보유중인것, 반환해야 할 것

useState

Reference

- Caveats(주의사항)
  - Set 함수는 다음 렌더링에 대한 상태 변수만 업데이트한다. set함수를 호출한 후 상태 변수를 읽으면 호출하기 전에 화면에 있었던 이전 값을 계속 얻을 수 있다 .
  - Object.is 비교를 통해 확인한 바와 같이 제공하는 새 값이 현재 상태와 동일한 경우 React는 구성 요소 및 구성 요소의 하위 항목을 re-rendering하는 것을 건너뛴다.. 이것은 최적화이다. 경우에 따라 React는 하위 요소를 건너뛰기 전에 구성 요소를 호출해야 할 수도 있지만 코드에 영향을 주지는 않는다.
  - React는 상태 업데이트를 일괄 처리한다. 모든 이벤트 핸들러가 실행 되고 해당 set 함수를 호출한 후에 화면을 업데이트한다. 이렇게 하면 단일 이벤트 중에 여러 번 리렌더링되는 것을 방지할 수 있다. 예를 들어 DOM에 액세스하기 위해 React를 강제로 먼저 화면을 업데이트해야 하는 경우에는 flushSync를 사용할 수 있다.
  - 렌더링 중 set 함수 호출은 현재 렌더링 구성 요소 내에서만 허용된다. React는 출력을 삭제하고 즉시 새 상태로 다시 렌더링을 시도한다. 이 패턴은 거의 필요하지 않지만 이전 렌더링의 정보를 저장 하는 데 사용할 수 있다.
  - Strict 모드에서 React는 우발적인 불순물을 찾는데 도움을 주기 위해 업데이트 기능을 두 번 호출 한다. 이는 개발 전용 동작이며 프로덕션에는 영향을 주지 않는다. 업데이터 함수가 순수하다면(있는 그대로) 동작에 영향을 주지 않아야 한다. 호출 중 하나의 결과는 무시된다.

index.js에서 `<React.StrictMode>` - 타입에 대해서 엄격하다

데이터 타입과 연관이 있다

useState

Usage

- Adding state to a component(구성요소에 상태 추가)
  - useState하나 이상의 상태 변수를 선언하려면 구성 요소의 최상위 수준에서 호출하라 .

```
import { useState } from 'react';

function MyComponent() {
  const [age, setAge] = useState(42);
  const [name, setName] = useState('Taylor');
  // ...
```

  - array destructuring를 사용하여 [Something, setSomething]과 같은 상태 변수의 이름을 지정한다.
  - useState정확히 두 개의 항목이 포함된 배열을 반환한다.
  - 이 상태 변수의 현재 상태는 처음에 사용자가 제공한 초기 상태로 설정된다 .
  - set 함수는 상호 작용에 대한 응답으로 다른 값으로 변경할 수 있다.
  - 화면의 내용을 업데이트하려면 다음 상태 값을 가진 set 함수를 호출하면 된다.

`const [agem setAge] = useState(42)` = array destructuring

age = 55; → 하면 안됨

상태변수는 assignment를 사용할 수 없다 - set함수를 사용해라

useState

Usage

- Adding state to a component

```
function handleClick() {
  setName('Robin');
}
```

- React는 다음 상태를 저장하고, 새 값으로 구성 요소를 다시 렌더링하고, UI를 업데이트한다.

Pitfall(함정)

set 함수를 호출해도 이미 실행 중인 코드의 현재 상태는 변경 되지 않는다 .

```
function handleClick() {
  setName('Robin');
  console.log(name); // Still "Taylor"!
}
```

다음 useState 렌더링부터 반환되는 항목에만 영향을 미친다 .

값을 바꾼다고 바로 변경되지 않음 → 시간이 조금 지나고 다시 불러올 때 값이 바뀐다

useState

Basic useState examples

- Counter (number)
  - count 상태 변수는 숫자를 유지한다. 버튼을 클릭하면 증가한다.

```
export default function Counter() {
  const [count, setCount] = useState(0);

  function handleClick() {
    setCount(count + 1);
  }

  return (
    <button onClick={handleClick}>
      You pressed me {count} times
    </button>
  );
}
```

You pressed me 6 times

읽는방법

HOOK의 React에있는 usestate 클릭 - 번역기사용해서 읽기(다른것도 잇다)

번역하면 일관적이지않다

usesage

Basic useState examples 공부

VS코드에서 깃허브에 올리는 것 방법 알기

내일 저녁까지 예상문제 단톡방에

문제수 10문제 내외

인의예지신, 중도