Hook 기본 다지기#1

useEffect#0

# Side Effect

‘Side Effect’ 란 무엇인가 ? 바로 함수 호출 시, 함수 내부가 아닌 외부에 영향을 미치는 것을 말한다. 다음의 코드를 보도록 하자

let a = 0;

function sideEffect(){

a++;

}

sideEffect()

sideEffect는 매개변수로 들어온 a를 변경하거나, a를 1을 증가시켜 return하지 않는다. 단순히 어떤 함수를 이용하다가 외부 스코프에 있는 변수 a를 변경하였다. 이처럼 외부 상태를 변경시키는 것을 ‘side Effect’라고 한다. ‘부수 효과’라는 어감 때문에, 무조건 안 좋은 느낌이 들 수 있지만 절대 나쁜 용도로 사용하는 것이 아니다.

# Use effect

클래스 형 컴포넌트에서는 부수효과를 발생 시키기 위해, ‘componentDidMount’함수를 사용하거나, ‘componetWillUpdate’, ‘componentWillUnmount’함수를 사용 했을 것이다. 이렇게 말하면 약간 와 닿지 않을 것 같아서, 다음의 예를 클래스형 컴포넌트로 예제를 보자.

export default class App extends Component {

state = {

name : "",

id : ""

}

async componentDidMount(){

const getUserInfo = await fetch("http://www.naver.com");

const userInfo = await getUserInfo.json();

this.setState({

name : userInfo.name,

id : userInfo.id

})

}

render() {

return (

<div>

</div>

)

}

}

보통 페이지가 마운트 되고, 유저의 정보를 외부 URL에서 가져오거나, 페이지에 필요한 API를 호출 하기 위해서 ‘componentDidMount’ 함수를 사용한다. 이처럼 클래스형 컴포넌트에서는 ‘react lift cycle’ 함수를 직접 호출 할 수 있기 때문에, ‘life cycle flow chart’만 보고 쉽게 사이드 이펙트 처리가 가능하다.

당연한 이야기지만, 함수형 컴포넌트는 Component를 상속 받지 않기 때문에, ‘react life cycle’ 함수를 호출 할 수 없다. 그렇기 때문에, 이에 대응하는 함수인 ‘useEffect’ 함수가 존재한다.

useEffect 함수는 두개의 전달인자를 받는데, 첫번째로 부수효과에 관련한 콜백 함수이며, 두번째인자는 이벤트 콜백이 발생하는 시기를 결정할 의존성 리스트 이다. 천천히 ‘life cycle’을 구현해보면서 이해해보도록 하자 .

# ComponentDidMount 구현

‘componentDidMount’ 함수는 컴포넌트가 최초 마운트 된 후 발생하지 않는 함수이다. 주로 user의 정보를 가져오거나 필요한 API를 호출하는 작업을 담당 한다 .

다음의 예제를 보자.

import logo from './logo.svg';

import './App.css';

import React, {useState,useEffect} from 'react'

function App() {

const [counter,setCounter] = useState(0);

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

const [obj,setobj] = useState({id : "hokidoki", pw : "0000"});

useEffect(()=>{

console.log("component Did Mount !!!")

},[])

return (

<div className="App">

{counter}

<button onClick={()=> setCounter(counter + 1)}>+</button>

</div>

);

}

export default App;

첫번째 전달인자로 콜백을 전달하고, 2번째 전달 인자로 빈 배열을 전달 하였다. 이 결과는 어떻게 이루어질까 ?

텍스트이(가) 표시된 사진

자동 생성된 설명

최초 함수형 컴포넌트가 마운트 될 때 단 한번만 실행되고 더 이상 실행 되지 않는다. 이를 위해 버튼을 눌러서 count를 마구 클릭해도 해당 콜백은 더 이상 작동 하지 않는다.

# 특정 state가 변경 될 때 콜백 작동 하게 하기

이전 포스팅에서 기술 한 것과 같이 ‘useState’ 함수에는 callBack 전달 인자가 존재 하지 않는다. 그렇기 때문에, 어떤 state가 변경 됨에 따라 사이드 이펙트를 발생 시켜야 하는데 어려움이 존재한다. useEffect함수는 이런 상황을 해결 해준다.

다음의 코드를 살펴보자 .

import logo from './logo.svg';

import './App.css';

import React, {useState,useEffect} from 'react'

function App() {

const [counter,setCounter] = useState(0);

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

const [obj,setobj] = useState({id : "hokidoki", pw : "0000"});

useEffect(()=>{

console.log("component Did Mount !!!")

},[])

useEffect(()=>{

console.log(counter)

console.log("counter is Update")

},[counter])

return (

<div className="App">

{counter}

<button onClick={()=> setCounter(counter + 1)}>+</button>

</div>

);

}

export default App;

특정 state가 변경 됨에 따라 사이드 이펙트를 발생 시키고 싶다면, 의존성 배열에 해당 state를 전달 하면 된다. 혹시 componentWillUpdate처럼 렌더가 발생 하면 해당 함수가 실행되는 건지 확인 하기위해 이름을 작성하는 input을 만들어 보도록 하자 .

import logo from './logo.svg';

import './App.css';

import React, {useState,useEffect} from 'react'

function App() {

const [counter,setCounter] = useState(0);

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

useEffect(()=>{

console.log("component Did Mount !!!")

},[])

useEffect(()=>{

console.log(counter)

console.log("counter is Update")

},[counter])

return (

<div className="App">

{counter}

<button onClick={()=> setCounter(counter + 1)}>+</button>

<input value={name} onChange={(e) => setName(e.target.value)}></input>

</div>

);

}

export default App;

결과는 다음과 같이 counter 변수가 변경 될 때 마다 useEffect가 발생하고, name이 변경되더라도, 해당 콜백은 발생 하지 않는다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그렇다면 name 혹은 counter 가 변경 될 때 콜백이 실행되게 하려면 어떻게 할 까 ? 다음과 같이 useEffect의 의존성 리스트에 name을 넣어주면 된다.

useEffect(()=>{

console.log(counter)

console.log("counter is Update")

},[counter,name])

# componentWillUnmount의 구현

‘componentWillUnmount’는 컴포넌트가 마운트 해제 되었을 때 발생 하는 함수이다. 보통, 해제되는 컴포넌트의 상태를 저장하는 부수효과를 발생시킨다.

useEffect에서 mount해제될 때 함수를 발생 시키는 방법은 다음과 같다.

useEffect(()=>{

return ()=>{

console.log("component Will Unmount")

}

},[])

의존성 배열은 신경 쓰지 말자. 단순히 useEffect가 리턴하는 함수를 컴포넌트가 해제되었을 때 실행 시킨다고 인지하면 된다.

이를 확인해보기 위해 다음과 같이 새로운 컴포넌트를 만들어 보도록 하자.

import logo from './logo.svg';

import './App.css';

import React, {useState,useEffect} from 'react'

function App() {

const [counter,setCounter] = useState(0);

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

const [show,setShow] = useState(false);

useEffect(()=>{

console.log("component Did Mount !!!")

},[])

useEffect(()=>{

console.log(counter)

console.log("counter is Update")

},[counter,name])

return (

<div className="App">

{counter}

<button onClick={()=> setCounter(counter + 1)}>+</button>

<input value={name} onChange={(e) => setName(e.target.value)}></input>

<button onClick={()=> setShow(!show)}>쇼 !</button>

{show ? <ShowName name={name}/> : null}

</div>

);

}

function ShowName({name}){

useEffect(()=>{

return ()=>{

console.log("component Will Unmount")

}

},[])

return (

<div>

{name}

</div>

)

}

export default App;

이제 쇼 버튼을 눌러 컴포넌트가 해제 될 때 마다 해당 함수가 발생하는 것을 알 수 있다.

