훅(hooks)

안 화 수

hooks

❖ 훅(hooks) ?

1. 클래스 컴포넌트에서는 생성자(constructor)에서 state를 정의하고 setState() 함수를 통해서 state를 업데이트 합니다. 이처럼 클래스 컴포넌트에서 사용하는 기능을 함수 컴포넌트에서 동일하게 구현할 수 있도록 해주는 기능이 훅(hook)이다.
2. 훅(hooks)은 리액트 v16.8 이후부터 지원하는 기능이다.
3. Hooks 이라는 영단어는 갈고리라는 뜻을 갖고 있는데, 리액트의 훅도 리액트의 생명주기 기능에 갈고리를 걸어 원하는 시점에 정해진 함수를 실행되도록 만든 것이다.
4. 리액트의 훅(hooks)의 이름은 use로 시작한다.

state

❖ state

1. 일반적으로 컴포넌트의 내부에서 변경 가능한 데이터를 관리해야 할 경우에 사용 한다.
2. 프로퍼티(props)의 특징은 컴포넌트 내부에서 값을 바꿀 수 없다는 것이었다.
3. 변수의 값을 바꿔야 하는 경우에 state라는 것을 사용한다.
4. 변수의 값을 저장하거나 변경할 수 있는 객체로 보통 이벤트와 함께 사용된다.
5. 컴포넌트에서 동적인 값을 상태(state)라고 부르며, 동적인 데이터를 다룰 때 사용된다.

hooks

훅의 종류

- useState
- useEffect
- useMemo
- useCallback
- useRef
- 커스텀 훅

useState

- **useState**

  함수 컴포넌트에서 가변적인 상태를 처리해주는 훅이다.

  const [변수명, set함수명] = useState(초기값)

  1. useState()에 초기값을 넣어서 호출하면, 배열이 리턴 되어서 나온다.
  2. 리턴되는 배열의 첫 번째 항목은 state로 선언된 변수이고, 두 번째 항목은 해당 state의 set함수이다.

hooks

- **프로젝트 생성**

npx create-react-app hooks01

useState

- **useState 예제1 : src/App.js**

  함수 컴포넌트에서 가변적인 상태를 처리해주는 훅이다.

```
import Counter from "./Counter";

//function App() {
const App=()=>{
  return <Counter />
}

export default App;
```

useState

- **useState 예제1 : src/Counter.js**

  함수 컴포넌트에서 가변적인 상태를 처리해주는 훅이다.

```
import React, {useState} from "react";

//function Counter(){
const Counter=()=>{
   // const [변수, set함수] = useState(초기값);
    const [count, setCount] = useState(0)
    return(
        <div>
            <p>총 {count}번 클릭했습니다.</p>
            <button onClick={ ()=> setCount(count+1) }>클릭</button>
        </div>
    )
}
export default Counter;
```

useState

- **useState 예제2 : src/App.js**

  useState를 여러 번 사용하기

```
import Counter from "./Counter";
import Info from "./Info";

//function App() {
const App=()=>{
//  return <Counter />
    return <Info />
}

export default App;
```

useState

- **useState 예제2 : src/Info.js (1/2)**
  useState를 여러 번 사용하기

```
import React, { useState } from "react";

//function Info(){
const Info=()=>{
    const [name, setName] = useState('');
    const [nickname, setNickname] = useState('');

    const onChangeName=(e)=>{                    // onChangeName()함수 정의
        setName(e.target.value);
    }
    const onChangeNickname=(e)=>{               // onChangeNickname()함수 정의
        setNickname(e.target.value);
    }
```

useState

- **useState 예제2 : src/Info.js (2/2)**

```
 return (
        <div>
            <div>
                <input  onChange={onChangeName}/>
                <input  onChange={onChangeNickname}/>
            </div>
            <div>이름 : {name} </div>
            <div>닉네임 : {nickname} </div>
        </div>
    );
};

export default Info;
```

useState

- **useState 예제3 : src/App.js**

```
import Counter from "./Counter";
import Info from "./Info";
import Say from "./Say";

//function App() {
const App=()=>{
//    return <Counter />
//    return <Info />
      return <Say />
}

export default App;
```

useState

useState 예제3 : src/Say.js (1/2)

```
import React, {useState} from "react";

//function Say(){
const Say=()=>{
    const [message, setMessage] = useState('');
    const onClickEnter = () => {setMessage('안녕 하세요?');}      // onClickEnter()함수 정의
    const onClickLeave = () => {setMessage('안녕히 가세요!');}   // onClickLeave()함수 정의

    const [color, setColor] = useState('black');
```

useState

- **useState 예제3 : src/Say.js (2/2)**

```
return (
      <div>
        <button onClick={()=>setMessage('안녕 하세요?')}>입장1</button>
        <button onClick={onClickEnter}>입장2</button>
        <button onClick={onClickLeave}>퇴장</button>
        <h1 style={{ color }}>{message}</h1>

        <button style={{ color: 'red' }} onClick={() => setColor('red')}>빨간색</button>
        <button style={{ color: 'green' }} onClick={() => setColor('green')}>초록색</button>
        <button style={{ color: 'blue' }} onClick={() => setColor('blue')}>파란색</button>
      </div>
    );
};

export default Say;
```

useState

- **useState 예제4 : src/App.js**

```
import Counter from "./Counter";
import Info from "./Info";
import Say from "./Say";
import Average from './Average';

//function App() {
const App=()=>{
//    return <Counter />
//    return <Info />
//    return <Say />
      return <Average />
}

export default App;
```

useState

useState 예제4 : src/Average.js (1/3)

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

// 평균을 구해주는 함수
const getAverage = (numbers) => {
  console.log('평균값 계산중..');
  if (numbers.length === 0) return 0;

  // reduce() : numbers배열 원소들의 합을 구해준다.
  const sum = numbers.reduce((a, b) => a + b);
  return sum / numbers.length;
};
```

useState

useState 예제4 : src/Average.js (2/3)

```
// Average 함수 컴포넌트
const Average = () => {
  const [list, setList] = useState([]);      // 배열로 초기화
  const [number, setNumber] = useState('');

  const onChange = (e) =>{                   // onChange()함수 정의
    setNumber(e.target.value);
  }
  const onInsert = (e) => {                  // onInsert()함수 정의
    // concat(): 배열을 결합하여 새로운 배열을 생성하는 함수
    const nextList = list.concat(parseInt(number));
    setList(nextList);
    setNumber('');
  }
```

useState

- **useState 예제4 : src/Average.js (3/3)**

```
return (
    <div>
        <input value={number}  onChange={onChange}  />
        <button onClick={onInsert}>등록</button>
        <ul>
              { list.map((value, index) => (
                    <li key={index}>{value}</li>
               )) }
        </ul>
        <div>
          <b>평균값:</b> {getAverage(list)}
        </div>
    </div>
  );
};
export default Average;
```

useState

- **프로젝트 생성**

```
npx    create-react-app    hooks02
```

useEffect

useEffect

useEffect는 리액트 컴포넌트가 렌더링될 때마다 특정 작업이 수행하도록 설정할 수 있는 훅(hooks)이다.

useEffect (이펙트 함수, 의존성 배열)

1. 첫 번째 파라미터는 이펙트 함수가 들어가고, 두 번째 파라미터는 의존성 배열이 들어간다. 이때 배열안에 들어 있는 변수 중에서 하나라도 값이 변경되었을때 이펙트 함수가 실행된다.
2. 이펙트 함수는 처음 컴포넌트가 렌드링된 이후와 업데이트로 인한 재렌더링 이후에 실행된다.
3. 이펙트 함수를 마운트와 언마운트시에 단 한번만 실행되게 하고 싶으면, 의존성 배열에 빈 배열([ ])을 넣으면 된다.
4. 의존성 배열은 생략할 수도 있는데, 생략하게 되면 컴포넌트가 업데이트 될 때 마다 호출된다.

useEffect

useEffect

1. useEffect는 리액트 컴포넌트가 렌더링될 때마다 특정 작업이 수행하도록 설정 할 수 있는 훅(hooks)이다.
2. 컴포넌트가 마운트 됐을 때(처음 나타났을 때), 언마운트 됐을 때(사라 질때) 업데이트 될 때(특정 props가 바뀔 때) 처리할 수 있는 훅이다.

```
useEffect ( () => {
    //1.컴포넌트가 마운트 된 이후,
    //  의존성 배열에 있는 변수들 중 하나라도 값이 변경 되었을때 실행됨
    //2.의존성 배열에 빈 배열([])을 넣으면, 마운트와 언마운트시에 단 한번씩만 실행됨
    //3.의존성 배열 생략 시 컴포넌트 업데이트 시마다 실행됨

} ,  [의존성 변수1, 변수2, ...]  ) ;
```

useEffect

- **useEffect 예제1 : src/App.js**

```
import React from "react";
import Counter2 from "./Counter2";

//function App() {
const App=()=>{
        return <Counter2 />;
}

export default App;
```

useEffect

- **useEffect 예제1 : src/Counter2.js**

```
import React, {useState, useEffect} from "react";

//function Counter2(){
const Counter2=()=>{
    const [count, setCount] = useState(0)
    useEffect(() => {
                document.title = count +'번 클릭';
                 console.log(count +'번 클릭');
    })
    return(
        <div>
            <p>총 {count}번 클릭했습니다.</p>
            <button onClick={ ()=> setCount(count+1) }>클릭</button>
        </div>
    )
}
export default Counter2;
```

useEffect

- **useEffect 예제2 : src/App.js**

```
import React from "react";
import Counter2 from "./Counter2";
import Info2 from "./Info2";

//function App() {
const App=()=>{

//  return <Counter2 />;
    return <Info2 />;

}

export default App;
```

useEffect

- **useEffect 예제2 : src/Info2.js (1/3)**

```
import React, { useState, useEffect } from "react";

//function Info2(){
const Info2=()=>{
    const [name, setName] = useState('');
    const [nickname, setNickname] = useState('');

    const onChangeName=(e)=>{                    // onChangeName()함수 정의
        setName(e.target.value);
    }
    const onChangeNickname=(e)=>{                // onChangeNickname()함수 정의
        setNickname(e.target.value);
    }
```

useEffect

- **useEffect 예제2 : src/Info2.js (2/3)**

```
 // useEffect는 리액트 컴포넌트가 렌더링될 때마다 특정 작업을 수행하도록
// 설정할 수 있는 훅(hooks)이다.
useEffect(()=>{
    console.log('렌더링이 완료 되었습니다.');
    console.log({name, nickname});
});

// useEffect에서 설정한 함수를 컴포넌트가 화면에 맨 처음 렌더링될 때만 실행하고,
// 업데이트될 때는 실행하지 않으려면 함수의 두번째 파라미터를 비어있는
//  배열로 설정하면 된다.
// useEffect(()=>{
//     console.log('마운트될 때만 실행됩니다.');
// }, []);
```

useEffect

- **useEffect 예제2 : src/Info2.js (3/3)**

```
 return (
        <div>
            <div>
                <input  onChange={onChangeName}/>
                <input  onChange={onChangeNickname}/>
            </div>
            <div>이름 : {name} </div>
            <div>닉네임 : {nickname} </div>
        </div>
    );
};

export default Info2;
```