7. 훅스로 컴포넌트 개선하기 (실습)

Prof. Seunghyun Park (sp@hansung.ac.kr)

Division of Computer Engineering

한성대학교 HANSUNG UNIVERSITY

학습 목표: 7. 훅스로 컴포넌트 개선하기

- 실습준비: ex-sample 활용하여 프로젝트 구조 생성

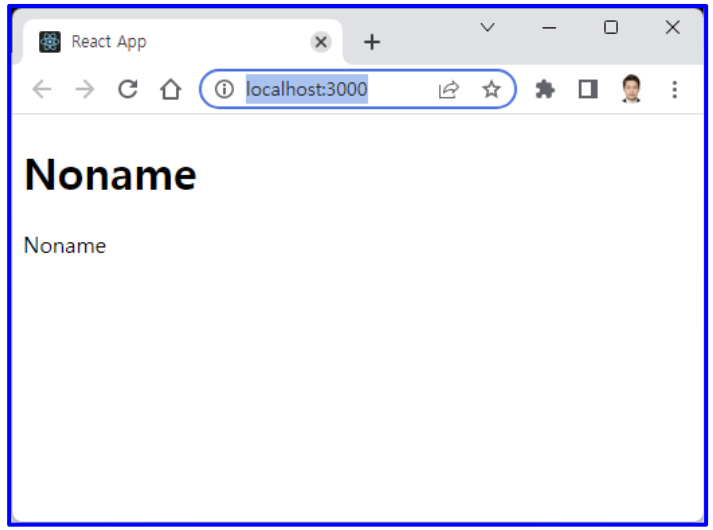
- 실습33: useState()
- 실습34: useEffect()
- 실습35-1: useState(), useReducer()
- 실습35-2: custom hooks: useCounter()

실습준비: 빈 프로젝트 생성 – ex-sample 활용

새로운 프로젝트 생성

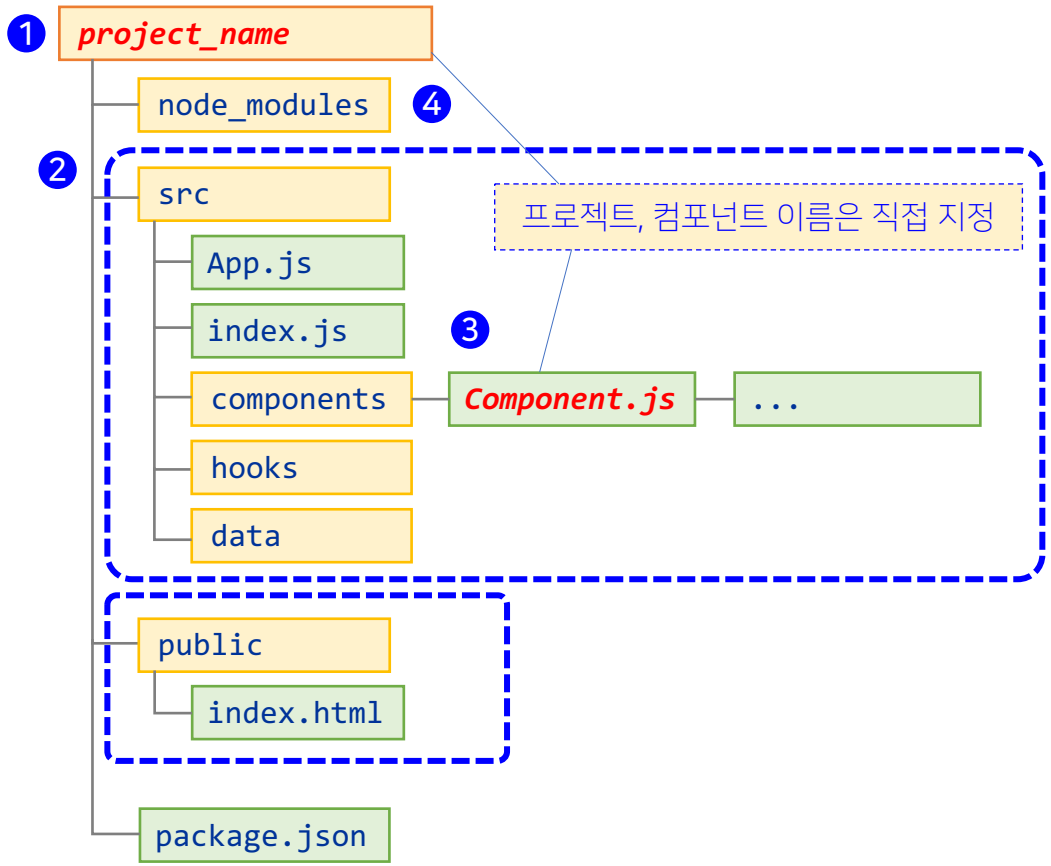
1. > mkdir *project_name*
2. > *copy files from the ex-sample to your project folders*
3. > *edit source codes including components*
4. > npm install
   > npm start

※ node_modules는 복사하지 않음



3. 프로젝트 폴더 구조 생성 및 모듈 배치

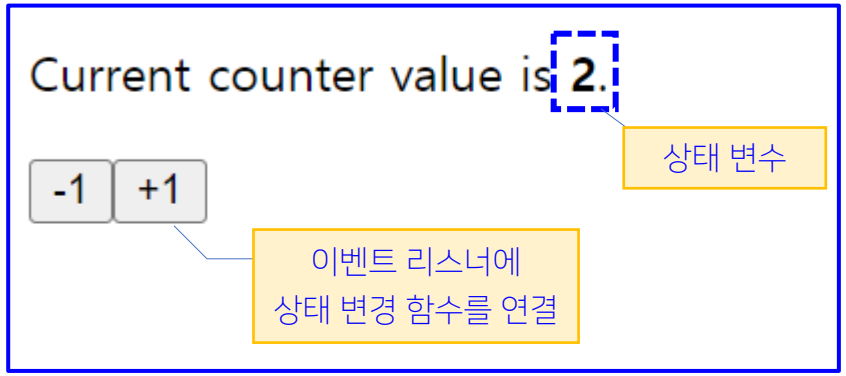
- ./src, ./src/data, ./src/components, ./public



실습33-1: Counter: useState()

❖ 컴포넌트의 상태를 관리하고 변경하는 useState() 훅을 이용하여 카운터 구현

```
/* ch07/ex/ex33-1/src/App.js */
import Counter from "./components/Counter";

function App() {
  return (
    <Counter value={0} />
  );
}

export default App;
```




```
/* ch07/ex/ex33-1/src/components/Counter.js */
import React, { useState } from 'react';

const Counter = props => {
  const [value, setValue] = useState(props.value);

  return (
    <div>
      <p>Current counter value is <b>{value}</b>.</p>
      <button onClick={() => setValue(value - 1)}>-1</button>
      <button onClick={() => setValue(value + 1)}>+1</button>
    </div>
  )
};

export default Counter;
```

실습33-2: Favorite : useState()

❖ 2개의 상태 변수를 이용하여 입력 값과 임시 저장한 값을 화면에 렌더링하는 컴포넌트 구현

```
/* ch07/ex/ex33-2/src/App.js */
import Favorite from "./components/Favorite";

function App() {
  return (
    <Favorite />
  );
}

export default App;
```

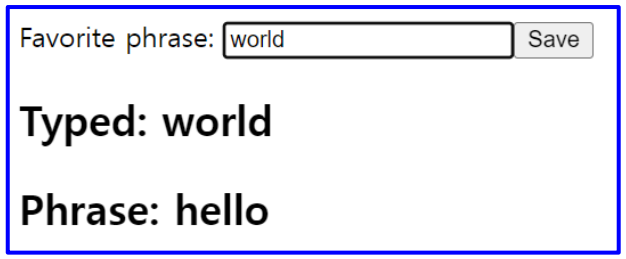

```
/* ch07/ex/ex33-2/src/components/Favorite.js */
import React, { useState } from 'react';

const Favorite = () => {
  const [typed, setTyped] = useState("");
  const [phrase, setPhrase] = useState("initial phrase");
  const createPhrase = () => {
    setPhrase(typed);
    setTyped("");
  }

  return (
    <div>
      <label>Favorite phrase: </label>
      <input value={typed} placeholder={phrase}
        onChange={e => setTyped(e.target.value)} />
      <button onClick={createPhrase}>Save</button>
      <h2>Typed: {typed}</h2>
      <h2>Phrase: {phrase}</h2>
    </div>
  )
};
export default Favorite;
```

실습34-1: Favorite : useEffect()

https://react.dev/reference/react/useEffect#specifying-reactive-dependencies

❖ 실습 1-2에 따른 렌더링 이후, 다음의 할 작업을 수행

- 입력 폼에 값 입력 또는 Save 버튼 클릭으로 상태가 변경되면 typed와 phrase 정보를 콘솔에 출력




```
typing:
saved: initial-phrase
typing: h
saved: initial-phrase
typing: he
saved: initial-phrase
typing: hel
saved: initial-phrase
typing: hell
saved: initial-phrase
typing: hello
saved: initial-phrase
typing:
saved: hello
typing: w
saved: hello
typing: wo
saved: hello
```

동작 과정을 console에서 확인

```jsx
/* ch07/ex/ex34-1/src/components/Favorite.js */
const Favorite = () => {
  const [typed, setTyped] = useState("");
  const [phrase, setPhrase] = useState("initial-phrase");
  const createPhrase = () => {
    setPhrase(typed);
    setTyped("");
  }
  const onChangeTyped = e => setTyped(e.target.value);

  useEffect(() => console.log(`typing: ${typed}`));
  useEffect(() => console.log(`saved: ${phrase}`));

  return (
    <div>
      <label>Favorite phrase: </label>
      <input value={typed} placeholder={phrase}
        onChange={onChangeTyped} />
      <button onClick={createPhrase}>Save</button>
      <h2>Typed: {typed}</h2>
      <h2>Phrase: {phrase}</h2>
    </div>
  )
};
export default Favorite;
```

실습34-2~4: Favorite : useEffect()

❖ 실습 2-1에 이어서 의존성 배열을 변경하며 결과 확인

```
typing:
saved: initial-phrase
typing: h
typing: he
typing: hel
typing: hell
typing: hello
typing:
saved: hello
typing: w
typing: wo
typing: wor
typing: worl
typing: world
```

```
typed or phrase changed: "", "initial-phrase"
typed or phrase changed: "h", "initial-phrase"
typed or phrase changed: "he", "initial-phrase"
typed or phrase changed: "hel", "initial-phrase"
typed or phrase changed: "hell", "initial-phrase"
typed or phrase changed: "hello", "initial-phrase"
typed or phrase changed: "", "hello"
typed or phrase changed: "w", "hello"
typed or phrase changed: "wo", "hello"
typed or phrase changed: "wor", "hello"
typed or phrase changed: "worl", "hello"
typed or phrase changed: "world", "hello"
```

```
only once after init-render
>
```

```
// case2: useEffect() 마다 의존관계 배열에 각각의 상태를 하나씩만 지정
useEffect(() => console.log(`typing: ${typed}`), [typed]);
useEffect(() => console.log(`saved: ${phrase}`), [phrase]);
```

typed와 phrase는 독립적으로 변경 여부 확인

```
// case3: 하나의 useEffect() 의존관계 배열에 추적할 상태를 모두 지정
useEffect(
  () => console.log(`typed or phrase changed:
                     "${typed}", "${phrase}"`),
  [typed, phrase]
);
```

둘 중 하나라도 변경되면 수행하도록 의존성 배열 구성

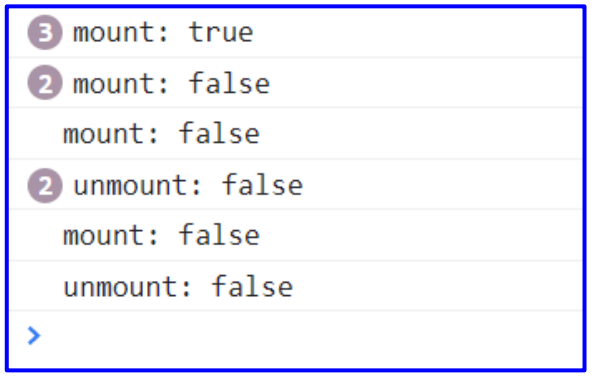
```
// case4: useEffect() 의존관계 배열에 아무 상태도 지정하지 않음
// ex34-1과 비교
useEffect(
  () => console.log(`only once after init-render`),
  []
);
```

렌더링 직후 한 번만 수행

실습34-5: color-org : useEffect()

❖ useEffect() 훅의 return 으로 clean-up 함수 반환을 구현

- 반환할 때 FaStar 컴포넌트의 색상을 결정하는 변수의 값을 출력




```jsx
/* ch07/ex/ex34-5/src/App.js */
import React, { useState } from 'react';
import StarRating from "./components/StarRating";

const App = () => {
  const [counter, setCounter] = useState(5);

  const removeStar = () => setCounter(counter - 1);
  const addStar = () => setCounter(counter + 1);

  return (
    <>
      <div>
        <button onClick={removeStar}>-1</button>
        <button onClick={addStar}>+1</button>
      </div>
      <StarRating totalStars={counter} />
    </>
  );
}

export default App;
```

전체 별의 개수를 App 컴포넌트의 상태로 관리

버튼의 클릭 이벤트 처리: -/+으로 별의 개수 증감

실습34-5: color-org : useEffect()

```
/* ch07/ex/ex34-5/src/components/StarRating.js */
import { useState } from "react";
import Star from "./Star";

const createArray = length => [...Array(length)];

const StarRating = ({ totalStars = 5 }) => {
  const [selectedStars, setSelectedStars] = useState(3);

  return (
    <>
      {createArray(totalStars).map((n, i) =>
        <Star key={i}  selected={selectedStars > i}
          onSelect={() => setSelectedStars(i+1)} />)}
      <p>{selectedStars} of {totalStars} stars</p>
    </>
  );
}

export default StarRating;
```

App으로부터 전달된 수만큼 Star를 그리도록 Star 목록 관리

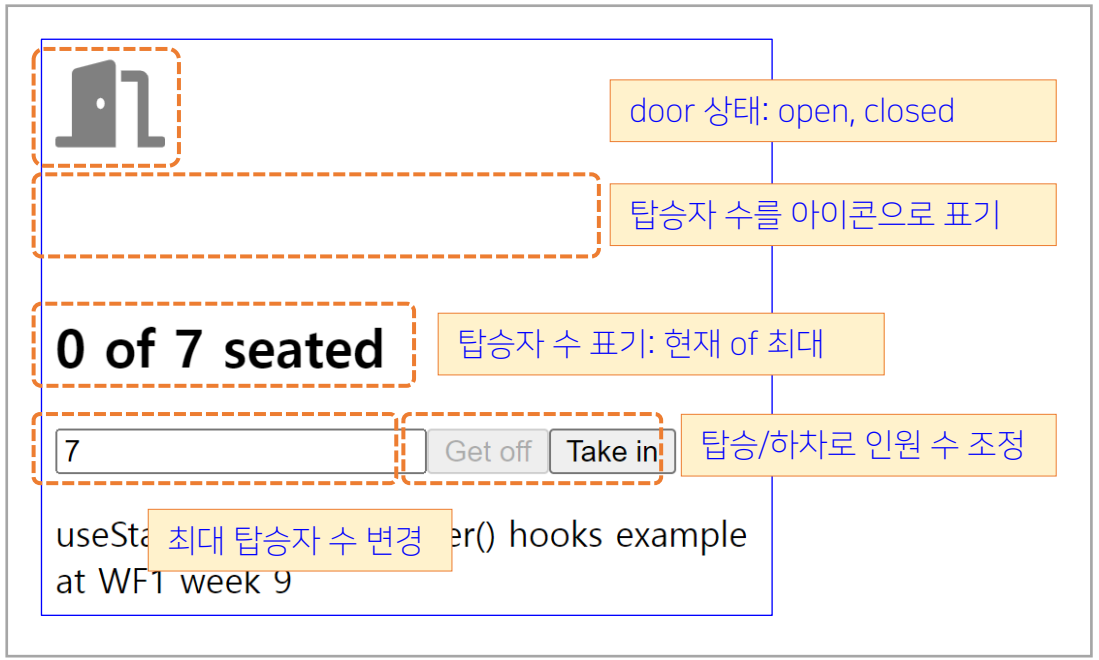
선택된 Star까지는 red, 이후는 grey를 지정하도록 Star 선택에 대한 상태 관리

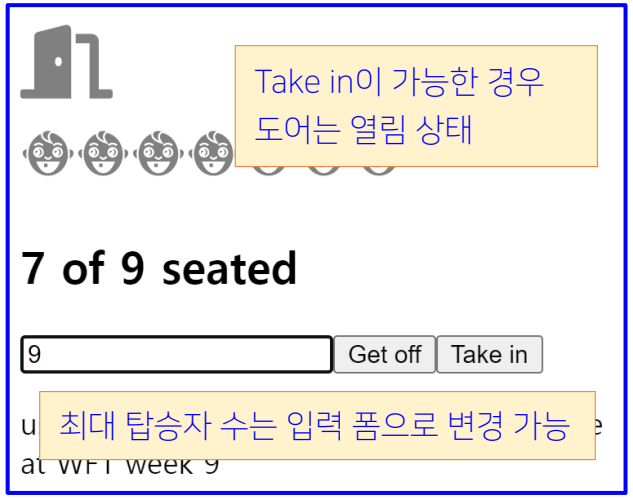
```
/* ch07/ex/ex34-5/src/components/Star.js */
import { useEffect } from "react";
import { FaStar } from "react-icons/fa";

const Star = ({ selected = false, onSelect = f => f }) => {

  1 useEffect(() => {
      console.log(`mount: ${selected}`);

    2 return () => {
        console.log(`unmount: ${selected}`);
      }
    }, []);

  return (
    <FaStar color={selected ? "red" : "grey"}
      onClick={onSelect} />
  );
}

export default Star;
```

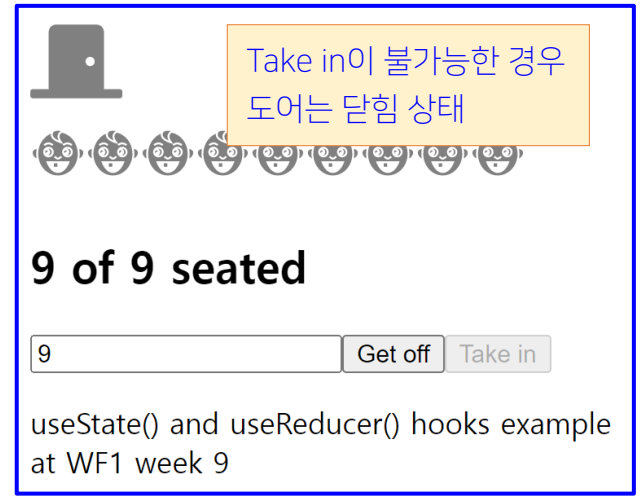
① Star가 렌더링 될 때와 ② Star가 소멸될 때 수행할 동작을 useEffect()으로 구현

실습35-1: Counter

❖ 자동차 탑승자 수와 도어 상태를 표시 및 제어하는 어플리케이션







실습35-1: Counter : useState(), useReducer()

```jsx
/* ch07/ex/ex35-1/src/App.js */
import Counter from "./components/Counter";

const App = () => (
  <Counter max={7} />
);

export default App;
```

```jsx
/* ch07/ex/ex35-1/src/components/Counter.js */
...

const Counter = ({ max }) => {
  <Door ... />
  <Passenger ... />
}

export default Counter;
```

```jsx
/* ch07/ex/ex35-1/src/components/Passenger.js */
import { GiBabyFace } from "react-icons/gi";

const Passenger = ({ count = 0 }) => {
  return (
    <div style={{ height: 40 }}>
      {[...Array(count)].map((e, i) =>
         <GiBabyFace size={30} key={i} color="grey" />)}
    </div>
  );
};

export default Passenger;
```

```jsx
/* ch07/ex/ex35-1/src/components/Door.js */
import { FaDoorOpen, FaDoorClosed } from "react-icons/fa";

const Door = ({ full = false }) => {
  return (
    <div style={{ height: 60 }}>
      {!full && <FaDoorOpen color="grey" size={50} />}
      {full && <FaDoorClosed color="grey" size={50} />}
    </div>
  )
};

export default Door;
```

실습35-1: Counter : useState(), useReducer()

```jsx
/* ch07/ex/ex35-1/src/components/Counter.js */
import React, { useReducer, useState } from 'react';
import Door from './Door';
import Passenger from './Passenger';

const Counter = ({ max }) => {
  const [maxSeat, setMaxSeat] = useState(max);
  const reducer = (state, action) => {
    switch (action.type){
      case 'INC':
        return state >= maxSeat ? state : state + 1;
      case 'DEC':
        return state == 0 ? state : state - 1;
      default:
        return state;
    }
  }
  const [counter, dispatch] = useReducer(reducer, 0);
  const increaseCount = () => dispatch({type:'INC'});
  const decreaseCount = () => dispatch({type:'DEC'});
  const onChangeCapacity = e => setMaxSeat(e.target.value);
  const isFull = counter >= maxSeat;
  const isEmpty = counter == 0;

  return (
    <>
      <Door full={isFull} />
      <Passenger count={counter} />
      <h2>{counter} of {maxSeat} seated</h2>
      <div>
        <input value={maxSeat} placeholder={maxSeat}
          onChange={onChangeCapacity} />
        <button onClick={decreaseCount}
          disabled={isEmpty}>Get out</button>
        <button onClick={increaseCount}
          disabled={isFull}>Get in</button>
      </div>
      <p>useState(), useReducer() hooks example</p>
    </>
  )
};

export default Counter;
```

실습35-2: Counter : useCounter() 커스텀 훅

```
/* ch07/ex/ex35-2/src/components/Counter.js */
import React, { useState } from 'react';
import { useCounter } from '../hooks/hooks';
import Door from './Door';
import Passenger from './Passenger';

const Counter = ({ max }) => {
  const [maxSeat, setMaxSeat] = useState(max);
  const [count, empty, full, increaseCount, decreaseCount]
    = useCounter(maxSeat);
  const onChangeMaxSeat = e => setMaxSeat(e.target.value);

  return (
   <>
    <Door full={full} />
    <Passenger count={count} />
    <h2>{count} of {maxSeat} seated</h2>
    <div> ... </div>
    <p>Custom hooks example at WF1 week 9</p>
   </>
  )
};

export default Counter;
```

```
/* ch07/ex/ex35-2/src/hooks/hooks.js */

import { useState } from 'react';

export const useCounter = capacity => {
```

관리할 상태 정보: count

반환할 정보: count, empty, full

. count: 현재 탑승자 수

. empty: 탑승자가 0인 경우

. full: 탑승자가 가득찬 경우

반환할 함수: incCount, decCount

. incCount(): 탑승자 1 증가

. decCount(): 탑승자 1 감소

반환할 정보와 함수를 array로 반환

```
};
```

```
<input value={maxSeat} placeholder={max} onChange={onChangeMaxSeat} />
<button onClick={decreaseCount} disabled={empty}>Get off</button>
<button onClick={increaseCount} disabled={full}>Take in</button>
```