6. 리액트 상태 관리 (1)

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

Division of Computer Engineering

한성대학교 HANSUNG UNIVERSITY

학습 목표: 6장. 리액트 상태 관리

- 리액트 상태
  - 예제 (별점 프로젝트: `StarRating`, `Star` 컴포넌트 구성)
- 컴포넌트 상태와 상태 변경을 위한 `useState()`
- 리팩토링
- 컴포넌트 트리
- 폼만들기
- 리액트 컨텍스트

리액트 상태

- 상태 (state)
  - 리액트 컴포넌트의 데이터를 표현하는 객체
  - 컴포넌트 내부에서 변경될 수 있는 값

※ props와의 차이

- 리액트 컴포넌트에서 속성 (property)을 나타내는 데이터
- 컴포넌트의 매개변수로 전달
- 읽기 전용
  - 컴포넌트 (함수) 는 컴포넌트 외부의 데이터인 props를 수정하지 않음 (순수함수의 개념 참조)

예제: 별점 프로젝트

새로운 프로젝트 생성: color-org

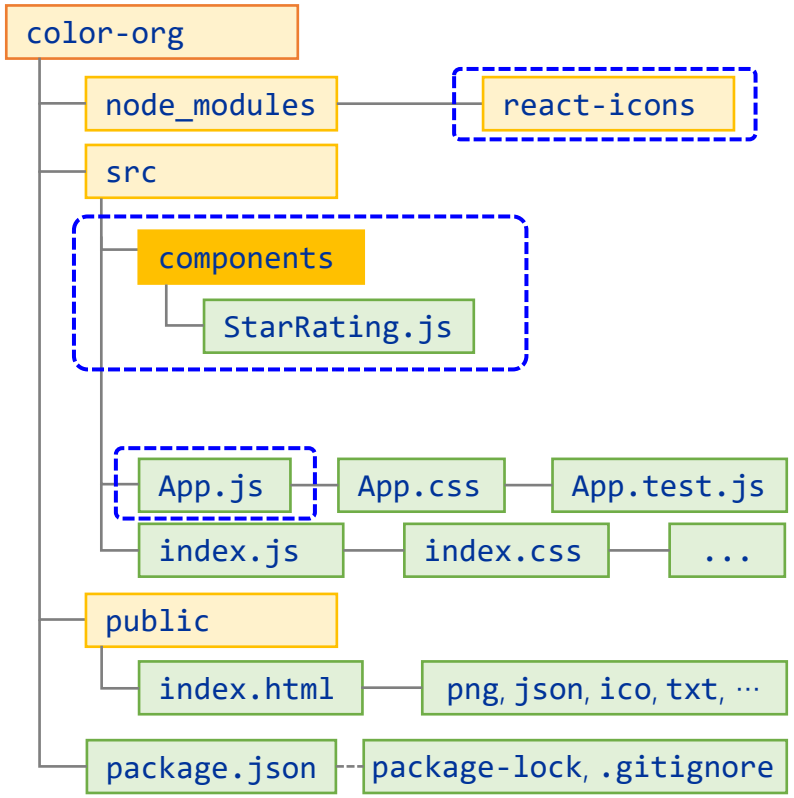
```
> create-react-app color-org
```

프로젝트에 필요한 패키지 설치: react-icons

```
> cd color-org
color-org> npm install react-icons
```

소스코드: 컴포넌트 구성

```
color-org-01> mkdir ./src/components
color-org-01> vi ./src/components/StarRating.js
color-org-01> vi ./src/App.js
```



별점 프로젝트: StarRating 컴포넌트

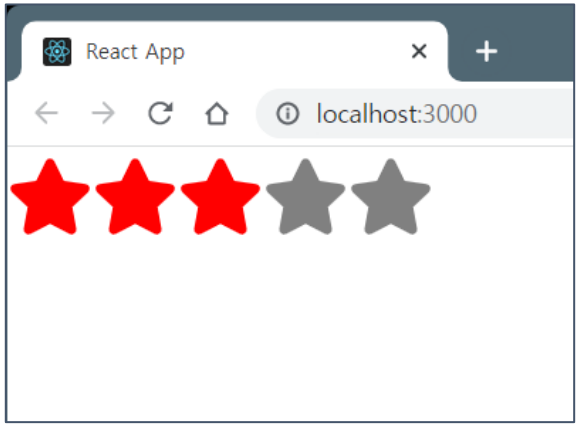
```
/* color-org/src/components/StarRating.js */
import React from "react";
import { FaStar } from "react-icons/fa";

export default function StarRating(){
    return [
        <FaStar color="red" />,
        <FaStar color="red" />,
        <FaStar color="red" />,
        <FaStar color="grey" />,
        <FaStar color="grey" />
    ];
}
```

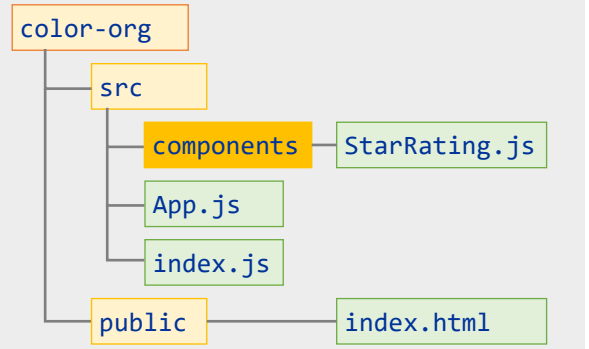
StarRating 생성
(함수형 컴포넌트)

```
/* color-org/src/App.js */
import StarRating from "./components/StarRating"

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

export default App;
```

메인 App:
StarRating 컴포넌트 반환

```
color-org> npm start
...
webpack compiled successfully
```



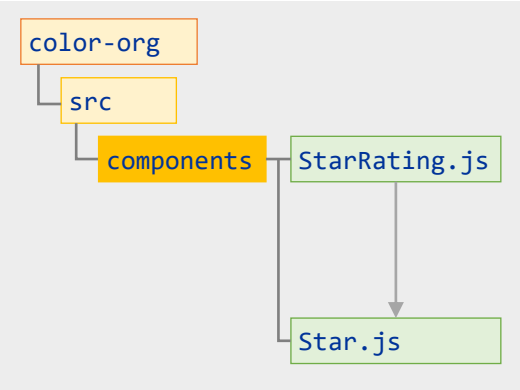


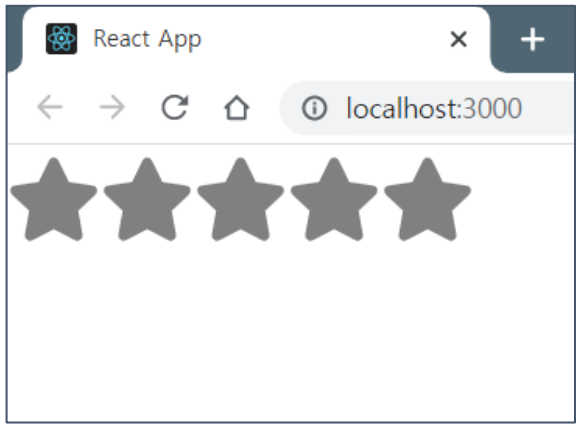

```
<!-- color-org/public/index.html */
<html>
<body>
    <div id="root"></div>
</body>
</html>
```

```
/* color-org/src/index.js */
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

별점 프로젝트: Star 컴포넌트

- StarRating 컴포넌트를 다음 조건에 따라 분리
  - StarRating: 별의 수를 전달받고 그 수만큼 객체 생성
    - 별의 개수: 기본 값 5
  - Star: 각각의 별을 생성할 때 속성에 따라 색깔을 표현
    - 별의 색깔: (true ? 빨강 : 회색)
    - 기본 값: false






```
/* color-org/src/components/StarRating.js */
import React from "react";
import Star from "./Star";

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

const StarRating = function( { totalStars = 5 }){
    return createArray(totalStars).map(
                                (n, i) => <Star key={i} /> );
}
export default StarRating;
```

길이가 length인 배열 생성하여 반환

전달된 길이의 배열을 생성하고
배열의 모든 요소를 <Star />로 변환

데이터 흐름의 방향 확인

```
/* color-org/src/components/Star.js */
import { FaStar } from "react-icons/fa";

const Star = ({ selected = false }) => (
    <FaStar color={selected ? "red" : "grey"} />
);

export default Star;
```

Star 컴포넌트 생성:
기본 값이 false인 <FaStar /> 반환

컴포넌트 상태: state

```
/* ./component/StarRating.js */
const StarRating = function( { total = 5 }){
  return createArray(total).map(
    (n, i) => <Star key={i} sel={true} /> );
}
```

```
/* ./component/Star.js */
const Star = ({ sel = false }) => (
  <FaStar color={sel ? "red" : "grey"} />
);
```

props로 데이터 전달

- 리액트 컴포넌트의 상태를 다루는 state
  - state는 컴포넌트 내부에서 상태를 저장하고, 변경하기 위해 사용

    ※ 비교: props는 값을 전달하는 부모 컴포넌트에서 설정하고, 컴포넌트 자신은 읽기전용으로만 사용
  - state 활용 방법
    - 클래스형 컴포넌트: 클래스 내부에 state 객체가 존재하므로, this.state로 참조
    - 함수형 컴포넌트: 상태 변수를 별도로 생성하고, useState() 를 활용하여 상태를 처리

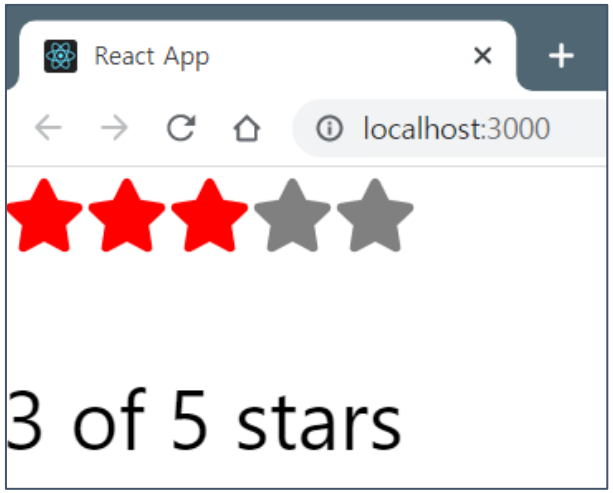
```
[ currentState, setterFunction ] = useState( initState );
```

현재 상태　　상태를 변경하는 함수　　초기 값

useState()

```
/* color-org/src/components/StarRating.js */
import React, {useState} from "react";
import Star from "./Star";

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

const StarRating = function( { totalStars = 5 }){
  const [selectedStars] = useState(3);

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

export default StarRating;
```

현재 상태를 selectedStars에 저장 (초기값: 3)
※ useState()는 배열을 반환 → 배열의 구조분해 할당 참조



i가 selectedStars인 (3)보다 작은 경우 (0, 1, 2) 만 true
해당 결과를 Star 컴포넌트의 매개변수로 전달

```
/* ./component/Star.js */
const Star = ({ selected = false }) => (
  <FaStar color={selected ? "red" : "grey"} />
);
```

컴포넌트 state의 변경

```
/* color-org/src/components/StarRating.js */
...
const StarRating = function( { 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>
    </>
  );
}
```

상태변경 함수에 의해 상태 값이 변경되는 경우,
함수 컴포넌트가 훅에 의해 다시 호출되면서 다시 렌더링이 발생함

<Star />에 클릭 이벤트가 발생하면,
onSelect 프로퍼티는 상태변경 함수 setSelectedStars()를 통해
selectedStars 상태의 값을 인덱스(i)+1로 변경

```
/* color-org/src/components/Star.js */
...
const Star = ({ selected = false, onSelect = f => f }) => (
  <FaStar
    color={selected ? "red" : "grey"}
    onClick={onSelect}
  />
);
```