

## 6. 리액트 상태 관리 (실습)

Prof. Seunghyun Park ([sp@hansung.ac.kr](mailto:sp@hansung.ac.kr))

Division of Computer Engineering

- 실습32-1: 별점 프로젝트

- 프로젝트 생성
- 별 1개 렌더링 (Star 컴포넌트)
- 별 5개 렌더링 (StarRating 컴포넌트)
- 상태변수 selectedStars 활용: useState() 혹
- click 이벤트 처리, 상태변경 함수 setSelectedStars()

- 실습32-2: 별점 프로젝트 (계속)

- 데이터를 파일로 입력 받아 화면에 렌더링 (Color 컴포넌트)
- 데이터를 배열로 확장, 화면에 렌더링 (ColorList 컴포넌트)
- 아이템 삭제 (click 이벤트, colors 상태, onRemoveColor)
- 별점변경 (click 이벤트, colors 상태, onRateColor)

# 실습32-1: 별점 프로젝트 (단계1)

단계 1: ex-sample의 소스를 활용하여 ex-32-1 프로젝트 생성

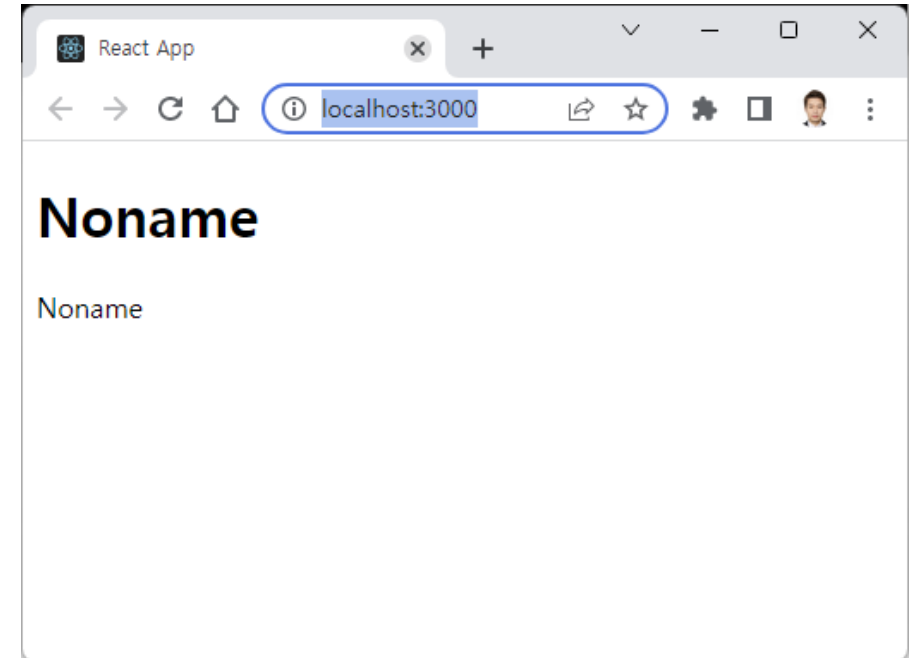
```
> mkdir ex32-1
> copy files from ex-sample/. to ex32-1/.
> cd ex32-1
> npm install
> npm start
```

node\_modules 폴더 제외



File explorer view of the ex-sample directory:

- ex-sample
  - node\_modules
  - public
    - index.html
  - src
    - components
    - Noname.js
    - App.js
    - index.js
  - .gitignore
  - package-lock.json
  - package.json



# 실습32-1: 별점 프로젝트 (단계2)

## 단계 2: Star 컴포넌트를 생성하고, FaStar 라이브러리를 이용하여 별 1개 렌더링

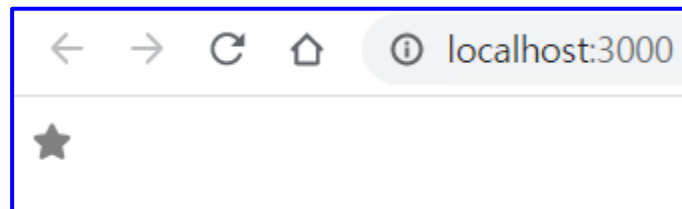
- App 컴포넌트에서 Star 컴포넌트로 전달하는 selected 프로퍼티가 true이면 붉은 색, false이면 회색의 별을 렌더링
- App 컴포넌트에서 호출하는 Noname 컴포넌트는 더 이상 사용하지 않으므로, 이 컴포넌트 이름을 변경하여 사용

```
/* ./src/components/Star.js */  
  
// import module FaStar from react-icons/fa  
import { FaStar } from "react-icons/fa";  
  
const Star = ({ selected = false }) =>  
  <FaStar color={selected ? "red" : "grey"} />  
  
// export module  
export default Star;
```

react-icons 설치 필요

```
> npm install react-icons  
> npm start
```

```
/* ./src/App.js */  
  
import Star from "../components/Star";  
  
function App() {  
  return (  
    <Star selected={false} />  
  );  
}
```



# 실습32-1: 별점 프로젝트 (단계3)

## 단계 3: StarRating 컴포넌트를 생성하고, Star 컴포넌트를 활용하여 별 5개 그리기

- App 컴포넌트에서 StarRating 컴포넌트로 전달하는 totalStars 프로퍼티의 숫자 만큼 별을 렌더링

```
/* ./src/components/StarRating.js */
import Star from "../Star";
const createArray = length => [...Array(length)];

const StarRating = ({ totalStars = 5 }) =>
  createArray(totalStars).map((n, i) =>
    <Star key={i} />
  )

export default StarRating;
```

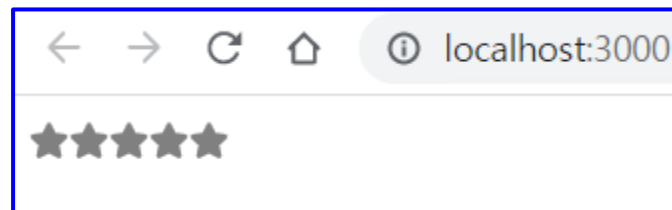
컴포넌트 생성

```
/* ./src/App.js */
import StarRating from "../components/StarRating";

function App() {
  return (
    <StarRating totalStars={5} />
  );
}

export default App;
```

컴포넌트 렌더링:  
StarRating



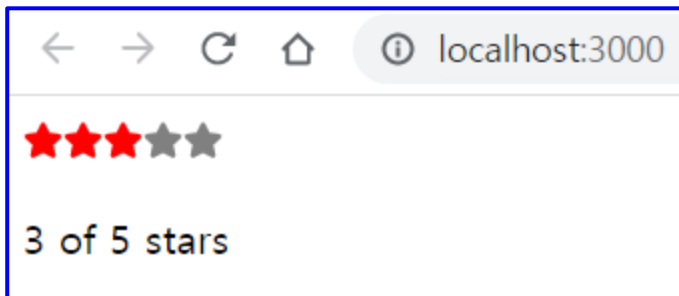
# 실습32-1: 별점 프로젝트 (단계4)

단계 4: StarRating 컴포넌트에서, 상태변수 `selectedStars`를 생성하고 초기 값 지정

- 지정된 수 만큼만 붉은 별, 나머지는 회색 별을 그리고
- 상태변수 값을 화면에 렌더링

```
/* ./src/components/StarRating.js */  
  
import { useState } from "react";  
import Star from "../Star";  
const createArray = length => [...Array(length)];  
  
const StarRating = ({ totalStars = 5 }) => {  
  const [selectedStars] = useState(3);  
}
```

상태 변수 지정



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

상태 변수 값 화면에 렌더링

# 실습32-1: 별점 프로젝트 (단계5)

## 단계 5: click 이벤트 처리: StarRating 컴포넌트에서, 상태 변경함수 setSelectedStars 등록

- FaStar 컴포넌트에 onClick 이벤트 리스너 등록하고, click 이벤트 처리 → 이벤트는 StarRating 컴포넌트에서 상태변경 함수 호출
- 상태 변경함수 setSelectedStars가 선택된 별 만큼 red로 변경함에 따라 다시 렌더링
- . 5개의 Star 중 하나를 클릭하면 선택된 Star가 몇 번째 Star인지 계산 (index+1)해서 상태변수 seletecStars의 값을 업데이트

```
/* ./src/components/StarRating.js */  
  
import { useState } from "react";  
...  
  
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>  
    </>  
  );  
}  
...  
}
```

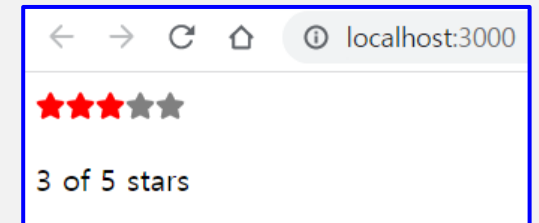
상태 변경함수

이벤트 처리

```
/* ./src/components/Star.js */  
  
import { FaStar } from "react-icons/fa";  
  
const Star = ({  
  selected = false,  
  onSelect = f => f  
}) =>  
  <FaStar  
    color={selected ? "red" : "grey"}  
    onClick={onSelect}  
  />  
  
export default Star;
```

이벤트 리스너

클릭: FaStar 컴포넌트에서 발생  
별점 변경: StarRating 컴포넌트에서 처리  
> 값 변경 후 다시 렌더링



## 실습32-2: 별점 프로젝트 (단계1)

단계 1: 데이터를 json 파일로 import 하고, 데이터의 구성에 따라 정보를 렌더링하는 Color 컴포넌트 생성

- 화면에 표시할 정보는 title, color, 별점 정보 (실습2: StarRating + Star 컴포넌트)

```
/* ./src/components/Color.js */  
  
import StarRating from "../StarRating";  
  
const Color = ({ colors }) =>  
  <section>  
    <h1>{colors.title}</h1>  
    <div style={{ backgroundColor: colors.color, height: 50 }} />  
    <StarRating selectedStars={colors.rating} />  
  </section>  
  
export default Color;
```

컴포넌트 생성:  
Color

```
/* ./src/App.js */  
import Color from "../components/Color";  
import colorData from "../data/color-data.json";  
  
function App() {  
  return (  
    <Color colors={colorData} />  
  );  
}  
  
export default App;
```

새로운 컴포넌트 Color 렌더링

```
{  
  "id": "0175d1f0-a8c6-41bf-8d02-df5734d829a4",  
  "title": "ocean at dusk",  
  "color": "#00c4e2",  
  "rating": 5  
}
```

데이터 파일  
color-data.json



## 실습32-2: 별점 프로젝트 (단계2)

단계 2: 3개의 데이터를 array를 json 파일로 import 하고, 데이터의 구성을 표현하는 ColorList 컴포넌트 생성

```
/* ./src/App.js */
import ColorList from "../components/ColorList";
import colorData from "../data/color-data.json";

function App() {
  return (
    <ColorList colors={colorData} />
  );
}

export default App;
```

새로운 컴포넌트 렌더링

```
[
  { "id": "...", "title": "ocean at dusk",
    "color": "#00c4e2", "rating": 5 },
  { "id": "...", "title": "lawn",
    "color": "#26ac56", "rating": 3 },
  { "id": "...", "title": "bright red",
    "color": "#ff0000", "rating": 0 }
]
```

데이터 확장

```
/* ./src/components/ColorList.js */
import Color from "../Color";

const ColorList = ({ colors = [] }) =>
  <div>
    {colors.map( color =>
      <Color key={color.id} {...color} />)}
  </div>
export default ColorList;
```

컴포넌트 생성

```
/* ./src/components/Color.js */
import StarRating from "../StarRating";

const Color = ({ title, color, rating }) =>
  <section>
    <h1>{title}</h1>
    <div style={{ backgroundColor: color, height: 50 }} />
    <StarRating selectedStars={rating} />
  </section>
export default Color;
```

ocean at dusk



★★★★★

5 of 5 stars

lawn



★★★☆☆

3 of 5 stars

bright red



☆☆☆☆☆

0 of 5 stars

## 실습32-2: 별점 프로젝트 (단계3)

단계 3: 아이템 삭제 - Color 컴포넌트 내부에 버튼을 추가하고, 버튼이 클릭되면 해당 아이템을 삭제하도록 구현

- 버튼과 이벤트 (Color 컴포넌트), 리스트 삭제 (ColorList 컴포넌트), 데이터 반영 (App 컴포넌트)

```
/* ./src/components/Color.js */
import StarRating from "../StarRating";
import { FaTrash } from "react-icons/fa";

const Color = ({ id, title, color, rating,
  onRemove = f => f }) =>
  <section>
    <h1>{title}</h1>
    <button onClick={() => onRemove(id)}>
      <FaTrash />
    </button>
    <div style={{ backgroundColor: color, height: 50 }} />
    <StarRating selectedStars={rating} />
  </section>

export default Color;
```

버튼 클릭 이벤트 리스너

ocean at dusk



★★★★★

5 of 5 stars

```
/* ./src/components/ColorList.js */
const ColorList = ({ colors = [],
  onRemoveColor = f => f }) =>
  <div>
    {colors.map( color =>
      <Color key={color.id} {...color}
        onRemove={onRemoveColor} />)}
  </div>
```

```
/* ./src/App.js */
import { useState } from "react";
function App() {
  const [colors, setColors] = useState(colorData);
  return (
    <ColorList colors={colors}
      onRemoveColor={id => {
        const newColor = colors.filter(
          color => color.id !== id);
        setColors(newColor);
      }} />
  );
}
```

클릭 이벤트 처리  
→ onRemoveColor()

## 실습32-2: 별점 프로젝트 (단계4)

### 단계 4: 별점변경 이벤트 처리 - FaStar 컴포넌트 클릭에 따라 별점을 변경하도록 이벤트 처리

- 버튼과 이벤트 (FaStar 컴포넌트), 값 변경 (StarRating 컴포넌트), 데이터 반영 (App 컴포넌트)

```
/* ./src/components/Star.js */

import { FaStar } from "react-icons/fa";

const Star = ({
  selected = false,
  onSelect = f => f
}) =>
  <FaStar
    color={selected ? "red" : "grey"}
    onClick={onSelect}
  />

export default Star;
```

별점 클릭 이벤트 리스너

```
/* ./src/components/StarRating.js */

// import { useState } from "react";
import Star from "./Star";
const createArray = length => [...Array(length)];

const StarRating = ({ totalStars = 5, selectedStars,
  onRate = f => f }) => {
  // const [selectedStars, setSelectedStars] = useState(3);
  return (
    <>
      {createArray(totalStars).map((n, i) =>
        <Star key={i} selected={selectedStars > i}
          onSelect={() => onRate(i+1)}
        />)}
      <p>{selectedStars} of {totalStars} stars</p>
    </>
  );
}

export default StarRating;
```

이벤트 처리를 위한 메서드 전달

## 실습2-2: 별점 프로젝트 (단계4, 계속)

```
/* ./src/components/Color.js */
import StarRating from "../StarRating";
import { FaTrash } from "react-icons/fa";
const Color = ({ id, title, color, rating,
  onRemove = f => f, onRate = f => f }) =>
  <section>
    <h1>{title}</h1>
    <button onClick={() => onRemove(id)}><FaTrash />
  </button>
    <div style={{ backgroundColor: color, height: 50 }} />
    <StarRating selectedStars={rating}
      onRate={rating => onRate(id, rating)}
    />
  </section>
export default Color;
```

이벤트 처리를 위한 메서드 전달

```
const removeColor = id => {
  const newColor = colors.filter(color => color.id !== id);
  setColors(newColor);
}
const rateColor = (id, rating) => {
  const newColors = colors.map(color => color.id === id ?
    { ...color, rating } : color);
  setColors(newColors);
}
```

별점 변경 필요한 id이면 별점 변경  
아니면 유지

```
/* ./src/components/ColorList.js */
import Color from "../Color";
const ColorList = ({ colors = [],
  onRemoveColor = f => f, onRateColor = f => f }) =>
  <div>
    {colors.map( color =>
      <Color key={color.id} {...color}
        onRemove={onRemoveColor}
        onRate={onRateColor} />)}
  </div>
export default ColorList;
```

이벤트 처리를 위한 메서드 전달

```
/* ./src/App.js */
import { useState } from "react";
function App() {
  const [colors, setColors] = useState(colorData);
  const removeColor = ...;
  const rateColor = ...;
  return (
    <ColorList
      colors={colors} onRemoveColor={removeColor}
      onRateColor={rateColor} />
  );
}
export default App;
```