20. 데이터 정렬

1. Number 정렬

> JavaScript의 `.sort()` 메서드

```
const data = [4, 3, 2, 1];
data.sort(); // 1, 2, 3, 4
const data2 = [1, 10, 2, 3];
data2.sort(); // 1, 10, 2, 3
```

> `.sort()` 메서드 안에서 정렬 기준을 `comparator` 함수로 설정

```
const data2 = [1, 2, 3, 4 ,10];
data2.sort(); //1, 10, 2, 3, 4
data2.sort((a, b) => a - b);
//1, 2, 3, 4, 10
```

> 앞뒤 값을 계속 비교하면서 정렬하기
>
> - 내림 차순일 경우
>
> ```
> data2.sort((a, b) => b - a);
> ```

2. string 정렬

> `localeCompar()` 메서드를 통해 문자의 순서를 정렬 할 수 있다.

```
const data = ['a', 'b', 'A', 'B', 't'];
data.sort();
// 'A', 'B', 'a', 'b', 't'
data.sort((a, b) => a.localCompare(b));
// 'a', 'A', 'b', 'B', 't'
```

3. object 정렬

> 여러가지 속성이 있는 객체들을 정렬하려면 어떻게 해야하는가?

```
const vegitables = [
    { weight: 10, cost: 10, name: "양파" },
    { weight: 5, cost: 20, name: "토마토"},
    { weight: 20, cost: 5, name: "당근" },
]
```

4. 테이블 정렬

> 테이블에서 여러가지 기준에 따라 정렬하기

- TODO 정렬 화살표 클릭 시 정렬을 해주는 함수

```
const handleClick = (label) => {
    //TODO 정렬 화살표 클릭 시 정렬을 해주는 함수
    if(sortBy && label !== sortBy){
        setSortOrder("asc");
        setSortBy(label);
        return;
    }
    if(sortOrder === null){
        setSortOrder('asc');
        setSortBy(label);
    } else if(sortOrder === 'asc'){
        setSortOrder('desc')
        setSortBy(label);
    } else if(sortOrder === 'desc'){
        setSortOrder(null);
        setSortBy(null);
    }
};
```

- TODO 정렬된 데이터로 바꿔 끼우는 부분 들어갈 comparator 함수

```
let sortedData = data;
if(sortOrder && sortBy) {
    const { sortValue } = config.find((column) => column.label === sortBy);
    // sort 데이터는 새롭게 정렬된 데이터 이다.
    sortedData = [...data].sort((a, b) => {
        //TODO - 정렬된 데이터로 바꿔 끼우는 부분 들어갈 COMPARATOR 함수
        const valueA = sortValue(a);
        const valueB = sortValue(b);

        const reverseOrder = sortOrder === 'asc' ? 1 : -1;

        if(typeof valueA === 'string') {
            return valueA.localCompare(valueB) * reverseOrder;
        }else{
            return (valueA - valueB) * reverseOrder;
        }
    });

    return <Tabel {...props} data={sortedData} config={updatedConfig} />
}
```

- number 타입을 정렬 할 때는 comparator 함수를 사용해주어야 한다.

- string 타입을 정렬할 때는 localCompare() 메서드를 사용해 주어야 한다.
- object 타입의 정렬을 할 때는 정렬 기준에 맞는 로직을 작성해 주어야 한다.