배열

Array.prototype.sort 메서드

1__ mutatable method

```
// 영어 문자
const fruits = ['Banana', 'Orange', 'Apple'];
fruits.sort();
console.log(fruits); // ['Apple', 'Banana', 'Orange']

// 한글 문자
const fruits = ['바나나', '오렌지', '사과'];
fruits.sort();
console.log(fruits); // ['바나나', '사과', '오렌지']
```

2__ 배열 요소를 문자열로 변환한 후, 유니코드 코드 포인트 순서에 따라 정렬

```
['2','1'].sort(); // ['1', '2']

const points = [40, 100, 1, 5, 2, 25, 10];
points.sort();
console.log(points); // [1, 10, 100, 2, 25, 40, 5]
```

2의 유니코드 코드 포인트는 U+0032

10의 유니코드 코드 포인트는 U+0031U+0030

따라서, 10의 유니코드 코드 포인트가 앞서기 때문에, 10, 2 순으로 정렬

3__ 정렬 순서를 정의하는 함수를 인자로 전달하기

(1) 위와 같은 상황을 해결하기 위해 비교 함수를 인자로 전달

(2) 양수, 0, 음수 중 하나를 반환

0이상을 반환하면 순서를 바꾸지 않으며, 음수를 반환하는 경우 순서를 바꿈

```
const points = [40, 100, 1, 5, 2, 25, 10];
points.sort((a, b) => a - b);
console.log(points); // [1, 2, 5, 10, 25, 40, 100]

points.sort((a, b) => b - a);
console.log(points); // [100, 40, 25, 10, 5, 2, 1]
```

객체를 요소로 갖는 배열도 정렬 가능

```
const todos = [
  { id: 4, content: 'JavaScript' },
  { id: 1, content: 'HTML' },
  { id: 2, content: 'CSS' },
];

function compare(key) {
  return (a, b) => (a[key] > b[key] ? 1 : (a[key] < b[key] ? -1 : 0));
}

todos.sort(compare('id'));
console.log(todos);
/*
  { id: 1, content: 'HTML' },
  { id: 2, content: 'CSS' },
  { id: 4, content: 'JavaScript' },
*/

todos.sort(compare('content'));
```

```
console.log(todos);
/*
  { id: 2, content: 'CSS' },
  { id: 1, content: 'HTML' },
  { id: 4, content: 'JavaScript' },
*/
```

브라우저별 Sort 종류

Array.sort 메서드는 브라우저 별로 사용하는 정렬 알고리즘이 다름

1__ Mozilla는 merge sort

2__ Safari는 selection sort

3__ Chrome v8은 (10 이하인 경우) insertion sort, (11이상인 경우) quick sort 사용.

현재 Chrome v8은 unstable한 quick sort 대신, stable한 timsort사용.

참조

참조

Tim Sort

1__ Tim Peters에 의하여 등장

2__ Insertion sort와 merge sort의 결합

3__ 시간 복잡도는 최선, 평균, 최악 순으로 O(n) / O(nlogn) / O(nlogn)

4__ 기존 merge sort에 비해서 적은 메모리 사용하므로, 다른 O(nlogn) 정렬 알고리즘 단점 극복

참조