JavaScript 정렬 알고리즘

5) JavaScript 정렬 라이브러리

JavaScript 정렬 라이브러리 | 알고리즘의 기본이 되는 정렬 알고리즘 이해하기

강사 나동빈

JavaScript
핵심 자료구조 알아보기

5) JavaScript 정렬 라이브러리

JavaScript 정렬 라이브러리

JavaScript
정렬
정렬 라이브러리

- JavaScript에서는 배열에 포함된 데이터를 정렬하는 $sort()$ 함수를 제공한다.
- 최악의 경우 시간 복잡도 $O(NlogN)$을 보장한다.
- **알고리즘 및 코딩 테스트 문제**를 해결할 때 정렬 기능이 필요하다면, 단순히 $sort()$ 함수를 사용하는 것을 권장한다.
- 만약, $sort()$ 함수의 사용이 제한된다면, 병합 정렬과 같은 알고리즘을 직접 구현하여 사용하자.

JavaScript 정렬 라이브러리

JavaScript
정렬
정렬 라이브러리

- 다음과 같은 형태로 사용할 수 있다.
- 이때, compareFunction은 **정렬 기준**을 정해주는 함수다.
- 내림차순, 오름차순 등 구체적인 정렬 기준을 설정할 수 있다.

```
arr.sort(compareFunction);
```

JavaScript 정렬 기준 함수(Compare Function)

- JavaScript의 정렬 함수에서는 **정렬 기준 함수**가 사용된다.
- 두 개의 원소 a, b를 입력으로 받는다.

1. 반환 값이 0보다 작은 경우 → a가 우선순위가 높아, 앞에 위치한다.
2. 반환 값이 0보다 큰 경우 → b가 우선순위가 높아, 앞에 위치한다.
3. 반환 값이 0인 경우 → a와 b의 순서를 변경하지 않는다.

정렬 기준 함수(Compare Function) 참고 사항

JavaScript
정렬
정렬 라이브러리

- 정렬 기준 함수를 사용하지 않으면 각 원소는 문자열로 취급된다.
- **유니코드 값 순서대로 정렬**된다.
- 따라서, 항상 정렬 기준 함수를 명시하는 습관을 들일 필요가 있다.

정수에 대한 오름차순 정렬 예시 1)

JavaScript 정렬
정렬 라이브러리

- 정수에 대하여 오름차순 정렬하는 코드 예시는 다음과 같다.

```
let arr = [1, 8, 5, 9, 21, 3, 7, 2, 15];

function compare(a, b) {
  if (a < b) return -1;
  else if (a > b) return 1;
  else return 0;
}

arr.sort(compare);

console.log(arr);
```

[실행 결과]

```
[
  1, 2,  3,  5, 7,
  8, 9, 15, 21
]
```

정수에 대한 오름차순 정렬 예시 2)

JavaScript 정렬
정렬 라이브러리

- a가 b보다 작을 때, 반환 값이 음수가 되어 a가 앞에 위치한다.

```
let arr = [1, 8, 5, 9, 21, 3, 7, 2, 15];

function compare(a, b) {
  return a - b;
}

arr.sort(compare);

console.log(arr);
```

[실행 결과]

```
[
  1, 2,  3,  5, 7,
  8, 9, 15, 21
]
```

정수에 대한 오름차순 정렬 예시 3)

JavaScript 정렬
정렬 라이브러리

- 비교 함수를 한 줄에 정의하여 곧 바로 적용할 수 있다.

```
let arr = [1, 8, 5, 9, 21, 3, 7, 2, 15];

arr.sort(function(a, b) {
  return a - b;
});

console.log(arr);
```

[실행 결과]

```
[
  1, 2,  3,  5, 7,
  8, 9, 15, 21
]
```

정수에 대한 내림차순 정렬 예시

JavaScript 정렬
정렬 라이브러리

- a가 b보다 클 때, 반환 값이 음수가 되어 a가 앞에 위치한다.

```
let arr = [1, 8, 5, 9, 21, 3, 7, 2, 15];

function compare(a, b) {
  return b - a;
}

arr.sort(compare);

console.log(arr);
```

[실행 결과]

```
[
  21, 15, 9, 8, 7,
   5,  3, 2, 1
]
```

문자열 대한 오름차순 정렬 예시

- 별도로 비교 함수(compare function)을 사용하지 않으면, 유니코드 순으로 정렬된다.
- 따라서 함수를 적용하지 않음으로써, 간단히 **문자열 정렬을 수행**할 수 있다.

```
let arr = [
  "fineapple",
  "banana",
  "durian",
  "apple",
  "carrot"
];

arr.sort();
console.log(arr);
```

[실행 결과]

```
[ 'apple', 'banana', 'carrot', 'durian', 'fineapple' ]
```

문자열 대한 내림차순 정렬 예시

- 문자열에 대하여 내림차순 정렬이 가능하다.

```
let arr = [
  "fineapple",
  "banana",
  "durian",
  "apple",
  "carrot"
];

function compare(a, b) {
  if (a > b) return -1;
  else if (a < b) return 1;
  else return 0;
}

arr.sort(compare);
console.log(arr);
```

[실행 결과]

```
[ 'fineapple', 'durian', 'carrot', 'banana', 'apple' ]
```

문자열 대한 오름차순 정렬 예시 (대소문자 구분 X)

- 대소문자를 구분하지 않도록 toUpperCase() 메서드를 사용할 수 있다.

```
let arr = ["fineapple", "Banana", "durian", "Apple", "carrot"];

function compare(a, b) {
  let upperCaseA = a.toUpperCase();
  let upperCaseB = b.toUpperCase();
  if (upperCaseA < upperCaseB) return -1;
  else if (upperCaseA > upperCaseB) return 1;
  else return 0;
}

arr.sort(compare);

console.log(arr);
```

[실행 결과]

```
[ 'Apple', 'Banana', 'carrot', 'durian', 'fineapple' ]
```

객체에 대하여 원하는 기준으로 오름차순 정렬 예시

- 성적 점수가 높은 순으로 학생 데이터를 나열할 수 있다.

```
let arr = [
  { name: "홍길동", score: 90 },
  { name: "김철수", score: 85 },
  { name: "박영희", score: 97 }
];

function compare(a, b) {
  return b.score - a.score;
}

arr.sort(compare);

console.log(arr);
```

[실행 결과]

```
[
  { name: '박영희', score: 97 },
  { name: '홍길동', score: 90 },
  { name: '김철수', score: 85 }
]
```