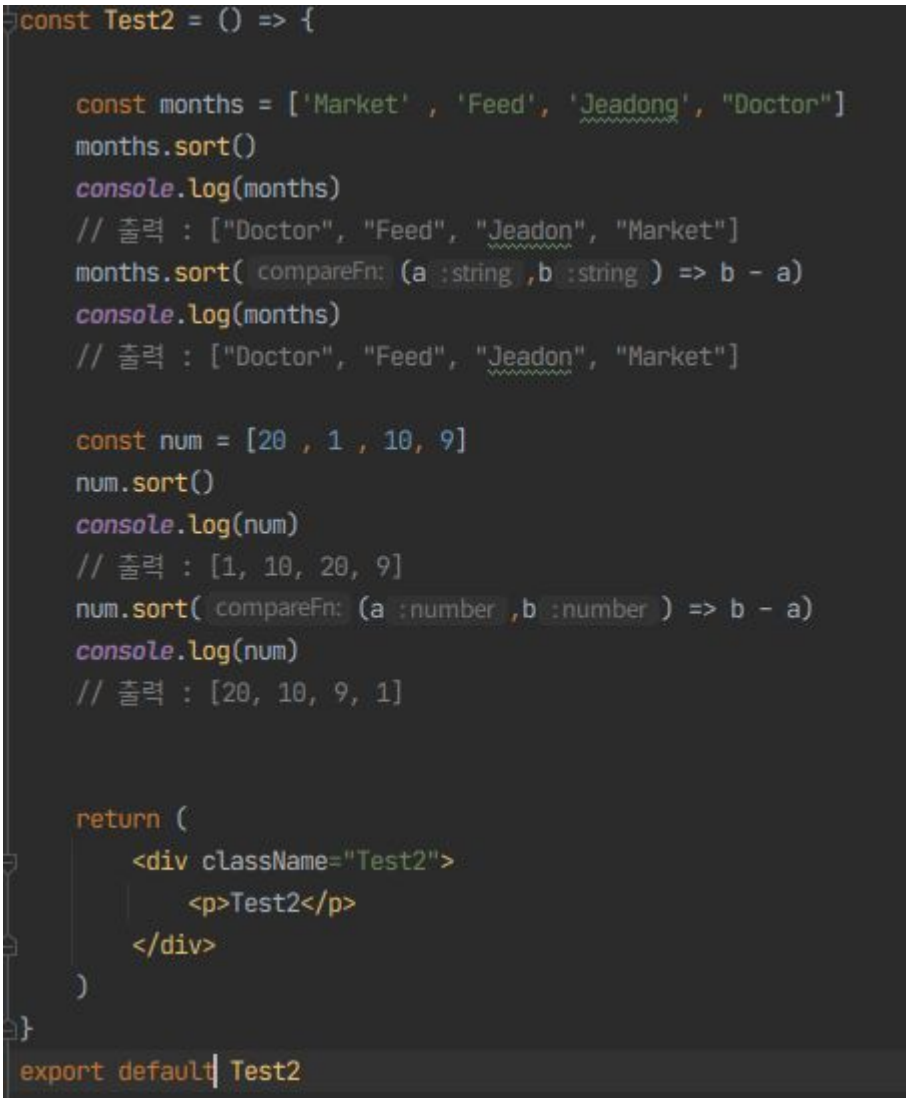
디지털컨버전스 기반 UXUI Front 전문 개발자 양성과정

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 - JungHyun LEE(이정현)
akdl911215@naver.com

테스트한 코드입니다. 위에 코드는 출력값이
["Doctor", “Feed”, “Jeadon”, “Market”]
[“Market”, “Jeadon”, “Feed”, “Doctor”] 로 출력이 될듯한데,
["Doctor", “Feed”, “Jeadon”, “Market”]
["Doctor", “Feed”, “Jeadon”, “Market”]
로 출력이 됩니다.

a b c d e f g h i j k l m n o p q r s t u v w x y z 역순으로
출력이 안됩니다.



기본값 매개변수 (default parameter)

기본값 함수 매개 변수(default function parameter)를 사용하면 값이 없거나 undefined가 전달 될 경우 이름붙은 매개변수를 초기화 할 수 있습니다.

```
const Test = () => {

    function multiply(a, b = 1) {
        return a * b;
    }

    console.log(multiply(5,2))
    // 값 = 10

    console.log(multiply(5))
    // 값 = 5

    return (
        <div className="Test">
            <p>Test</p>
        </div>
    )
}
export default Test
```

```
10                                  Test.js:7
5                                   Test.js:10
```

출력

구문작성법

```
function [name]([param1[ = defaultValue1 ][, ..., paramN[ = defaultValueN]]]) {
        statements
    }
```

Colored by Color Scripter

예시

```
const Test2 = () => {

    function multiply(a, b) {
        return a * b
    }

    console.log(multiply(5 ,2))
    // 10 출력
    console.log(multiply(5))
    // NaN 출력

    return (
        <div className="Test2">
            <p>Test2</p>
        </div>
    )
}
export default Test2
```

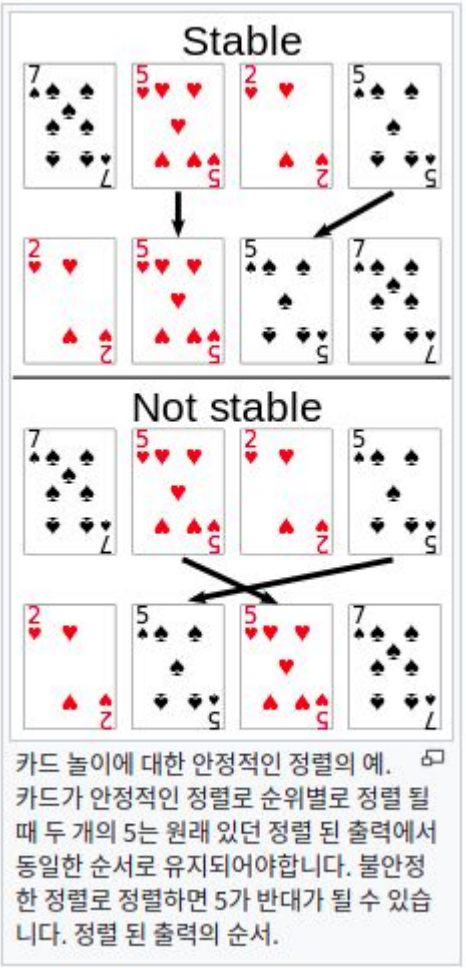
```jsx
const Test2 = () => {

    function multiply(a, b) {
        b = (typeof b === 'undefined' ? 1 : b)
        return a * b
    }

    console.log(multiply(5 ,2))
    // 10 출력
    console.log(multiply(5))
    // 5 출력

    return (
        <div className="Test2">
            <p>Test2</p>
        </div>
    )
}
export default Test2
```

```jsx
const Test2 = () => {

    function multiply(a, b = 1) {
        return a * b
    }

    console.log(multiply(5 ,2))
    // 10 출력
    console.log(multiply(5))
    // 5 출력
    console.log(multiply(5, undefined))
    // 5출력

    return (
        <div className="Test2">
            <p>Test2</p>
        </div>
    )
}
export default Test2
```

Stability(안정성)

안정적인 정렬 알고리즘은 반복되는 요소를 입력에 나타나는 것과 동일한 순서로 정렬합니다. 일부 데이터를 정렬 할 때 정렬 순서를 결정할 때 데이터의 일부만 검사합니다. 예를 들어 오른쪽의 카드 정렬 예에서 카드는 순위별로 정렬되고 해당 카드의 수트는 무시됩니다. 이렇게하면 원본 목록의 여러가지 하면 원본 목록의 여러가지로 올바르게 정렬 된 버전이 가능합니다. 안정적인 정렬 알고리즘은 다음 규칙에 따라 이 중 하나를 선택합니다. 두 개의 5개 카드처럼 두 항목이 동일하게 비교되면 상대적 순서가 유지되므로 하나가 입력에서 다른 항목보다 먼저 오면 출력에서 다른것보다 먼저 옵니다.

안정성은 다음과 같은 이유로 중요합니다. 이름과 클래스 섹션으로 구성된 학생 기록이 웹 페이지에서 동적으로 정렬됩니다. 첫 번째는 이름별로, 그 다음에는 두 번째 작업에서 클래스 섹션별로 정렬됩니다. 두 경우 모두 안정적인 정렬 알고리즘을 사용하는 경우 클래스 별 정렬 작업은 이름 순서를 변경하지 않습니다. 불안정한 정렬의 경우 섹션별로 정렬하면 이름 순서가 섞일 수 있습니다. 안정적인 정렬을 사용하면 사용자는 먼저 이름을 사용하여 정렬하여 다시 정렬하여 섹션별로 정렬 한 다음 이름별로 정렬하도록 선택할 수 있으므로 이름 순서가 유지됩니다.(일부 스프레드 시트 프로그램은 이 동장을 따릅니다. 이름별로 정렬 한 다음 섹션별로 정렬하면 섹션별로 알파벳순 학생 목록이 생성됩니다.)



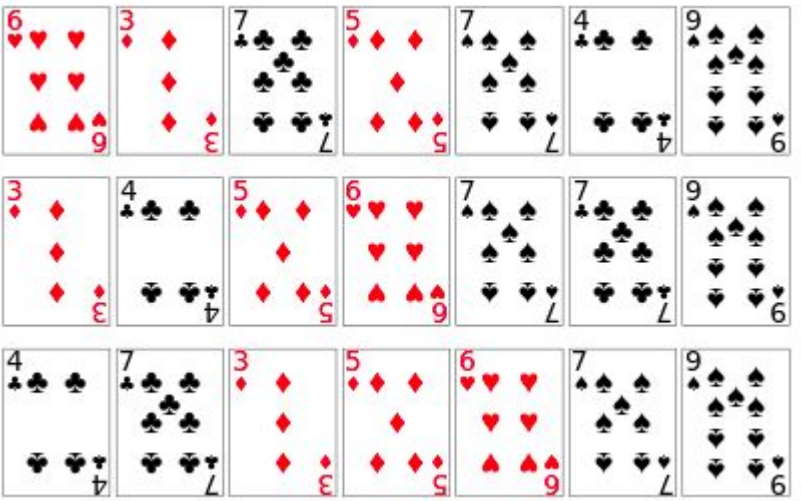
카드 놀이에 대한 안정적인 정렬의 예. 카드가 안정적인 정렬로 순위별로 정렬 될 때 두 개의 5는 원래 있던 정렬 된 출력에서 동일한 순서로 유지되어야합니다. 불안정한 정렬로 정렬하면 5가 반대가 될 수 있습니다. 정렬 된 출력의 순서.

보다 공식적으로 정렬되는 데이터는 레코드 또는 값의 튜플로 표시 될 수 있으며 정렬에 사용되는 데이터 부분을 key 라고 합니다. 카드 예에서 카드는 레코드 (랭크, 슈트)로 표시되고 키는 순위입니다. 동일한 키를 가진 두 개의 레코드는 R과 S가 있고 원래 목록에서 S 앞에 R이 나타날 때마다 R이 정렬된 목록에서 항상 S 앞에 표시되면 정렬 알고리즘은 안정적입니다.

정수 또는 보다 일반적으로 전체 요소가 핵심인 데이터와 같이 동일한 요소를 구별 할 수 없는 경우 안정성은 문제가 되지 않습니다. 모든 키가 다르면 안정성도 문제가 되지 않습니다.

불안정한 정렬 알고리즘은 안정적으로 구현할 수 있습니다. 이를 수행하는 한 가지 방법은 키 비교를 인위적으로 확장하여 원래 입력 목록의 항목 순서를 타이 브레이커로 사용하여 동일한 키가 있는 두 개체 간의 비교를 결정하는 것입니다. 그러나 이 순서를 기억하려면 추가 시간과 공간이 필요할 수 있습니다.

안정적인 정렬 알고리즘을 위한 한 가지 응용 프로그램은 기본 및 보조 키를 사용하여 목록을 정렬하는 것입니다. 예를 들어, 한 벌의 카드를 클럽, 다이아몬드, 하트, 스페이드 순서로 정렬하고 각 수트 내에서 카드를 다음과 같이 정렬한다고 가정합니다. 계급. 우선 순위별로 카드를 정렬 한 다음 (모든 종류를 사용하여) 정장별로 안정적인 정렬을 수행하면 됩니다.



각 슈트 내에서 안정된 정렬은 이미 완료된 순위 별 순서를 유지합니다. 이 아이디어는 여러 키로 확장 될 수 있으며 기수 정렬에 의해 활용됩니다. 예를 들어, 수트별로 먼저 비교 한 다음 수트가 동일한 경우 순위별로 비교하여 사전 식 키비교를 사용하여 불안정한 정렬로 동일한 효과를 얻을 수 있습니다.

배열함수

선언

let arr = new Array()

let arr = new Array(“a”)

let arr = [a]

삽입, 제거

arr.push(“b”)

arr.pop(“b”)

배열 합치기

result = arr1.concat(arr2)

배열 출력

arr -> Array[“a,b”]

arr.join(“|”) -> “a|b”

배열 역순배치

arr.reverse()

배얄 정렬

arr.sort()

배열 자르기

arr.slice(0,4) -> 0 <_ n < 4(4는 포함되지 않음을 주의)

arr.skilce(0,-2) -> 0번째부터 뒤에서 -2번째를 의미한다(0<_ m < n-2)

문자열 함수

s1.IndexOf(s2) - 문자열 s1 에서 s2를 검색하는 기본 함수. 찾은 위치를 없으면 -1 반환

s.valueOf(i) - 문자열 s의 i번째 값을 반환하는 기본함수.

까먹어서 substring()을 쓰는 일이 없도록 하자.

s.substring(i, j) - 문자열 s를 i<_n<j범위로 잘라낸다. 범위 주의!

s.substrt(i, n) - 문자열 s를 i부터 n개 잘라낸다. substring()과의 차이점에 주의

Number.prototype.valueOf(num) - Number 객체가 감싼 원시 값을 반환한다.

map(callback()) - 배열 내의 모든 요소 각각에 대하여 주어진 함수를 호출한 결과를 모아 새로운 배열을 반환한다.

reduce(callback(accumulator, currentValue))

reduce 는 reduce 함수(여기서는 그냥 callback함수)를 실행시켜 하나의 결과를 반환한다.

bind(this, arg1, ...)

this가 가리키는 것만 바꾸고 새로운 함수를 만든다. apply 와 call 이 함수를 실행하는 것과는 다르게 bind는 함수를 실행시키지 않는다. arg1 는 argynebts로 넘겨지는 인자값이다.

toFixed() - ParseFloat(String).toFixed(5) : 5자리까지만 나타낸다.

concat()

line = ‘’’’;

line.concat(“#”) 하면 결과가 바뀔까? 안바뀐다.

line = line.concat(“#”) 혹은 line += “#” 처럼 해주어야 한다.

apply(thisArg, [argArray])

apply는 함수를 호출하는 방식 중 하나다. 위에서

call(thisArg, arg1, arg2, ...)

apply와 마찬가지로 함수를 호출하는데, 차이점이 있다면 array를 넘기지 않고 각각 인자들을 풀어서 넘긴다는 점이다.