JavaScript Exercises

Basic

- Khai báo một mảng `arr = ["Linh", "Nhi", "Hùng", "Hà", "Mai Anh"]`
- In `arr` ra `console`
- Viết câu lệnh in ra độ dài (số lượng phần tử) của `arr`
- VIết câu lệnh in ra phần tử đầu tiên trong mảng (`index = 0`)
- Viết câu lệnh in ra phần tử thứ 3 trong mảng (`index = 2`)
- Viết câu lệnh in ra phần tử cuối cùng của mảng (`index = arr.length - 1`)
- Viết câu lệnh in ra phần tử có `index = -1`, chú ý kết quả
- Sử dụng vòng lặp in ra từng giá trị trong `arr`
- Sử dụng vòng lặp in ra giá trị và chỉ mục tương ứng trong `arr`
- Sử dụng vòng lặp in ra giá trị và chỉ mục tương ứng trong `arr` theo chiều từ cuối mảng về đầu mảng
- Viết câu lệnh thêm `"Ba"` cuối mảng sử dụng phương thưc `push()`
- Viết câu lệnh thêm `"Thảo"` vào cuối mảng **không sử dụng** phương thức `push()`
- Viết câu lệnh thêm `"Béo Ú"` vào vị trí đầu tiên trong mảng sử dụng phương thức `unshift()`
- Viết câu lệnh xóa phần tử ở vị trí cuối cùng của mảng sử dụng phương thức `pop()`
- Viết câu lệnh xóa phần tử ở vị trí đầu tiên trong mảng sử dụng phương thức `shift()`
- Viết câu lệnh sao chép 2 phần tử đầu tiên sử dụng phương thức `slice()`
- Viết câu lệnh sao chép toàn bộ phần tử của mảng sử dụng phương thức `slice()`
- Viết câu lệnh sao chép 3 phần tử cuối cùng của mảng sử dụng phương thức `slice()`
- Viết câu lệnh xóa 2 phần tử thứ 2 và 3 (`"Nhi"` và `"Hùng"`) khỏi mảng
- Viết câu lệnh thêm lại `"Nhi"` và `"Hùng"` vào vị trí thứ 2 và 3
- Viết câu lệnh tìm và in ra chỉ mục của `"Ba"` ra `console`
- Viết câu lệnh tìm và in ra chỉ mục của `"Thảo"` ra `console`, chú ý kết quả
- Viết câu lệnh kiểm tra `"Mai Anh"` có trong mảng `arr` hay không sử dụng phương thức `includes()` và in ra kết quả
- Viết câu lệnh in `arr` ra `console` dưới dạng chuỗi sử dụng phương thức `toString()`
- Viết câu lệnh nối các tên trong mảng thành một chuỗi duy nhất dạng `"Vinh-Huy-Linh-...-Hồng Anh-Bách"` sử dụng phương thức `join()` và in ra `console`
- Viết câu lệnh đảo ngược các giá trị trong mảng `arr` sử dụng phương thức `reverse()`
- Viết câu lệnh đổi chỗ 2 phần tử đầu và cuối mảng
- Viết câu lệnh xóa toàn bộ phần tử trong mảng

Intermediate

1. Viết hàm `max(arr)` nhận vào một mảng các số, tìm và trả về số lớn nhất

```
max([1, 5, 3, 4, 2]); // 5
```

---

2. Viết hàm `minMax(arr)` nhận vào một mảng các số, tìm ra số nhỏ nhất và lớn nhất trong mảng, sau đó trả về kết quả là một mảng mới chứa 2 giá trị `[min, max]`

```
minMax([1, 5, 3, 4, 2]); // [1, 5]
```

3. Viết hàm `avg(arr)` nhận vào một mảng các số, tính trung bình cộng các số và trả về kết quả

```
avg([1, 5, 3, 4, 2]); // 3
```

4. Viết hàm `swap(arr, x, y)` nhận vào một mảng các số và 2 số `x, y` tương ứng với 2 chỉ mục trong mảng, đổi chỗ vị trí 2 phần tử tương ứng, trả về kết quả là mảng `arr` đã thay đổi, lưu ý mảng `arr` phải thay đổi sau khi gọi hàm `swap()`

```
const arr = [1, 5, 3, 4, 2];

swap(arr, 0, 2); // [3, 5, 1, 4, 2]

console.log(arr); // [3, 5, 1, 4, 2]
```

5. Viết hàm `secondLargest(arr)` nhận vào một mảng `arr`, tìm và trả về kết quả là số lớn thứ 2 trong mảng, lưu ý mảng có thể chứa nhiều số trùng nhau

```
secondLargest([1, 5, 5, 3, 4, 2]); // 4
```

6. Viết hàm `mix(arr1, arr2)` nhận vào 2 mảng bất kỳ, thực hiện trộn (kết hợp) 2 mảng vào nhau và trả về kết quả là một mảng mới chứa các phần tử đã trộn

```
mix([1, 2, 3], [4, 5, 6]); // [1, 4, 2, 5, 3, 6]
```

7. Viết hàm `shuffle(arr)` nhận vào một mảng chứa các giá trị bất kỳ, thực hiện xáo trộn ngẫu nhiên vị trí các phần tử trong mảng và trả về kết quả là mảng đã xáo trộn, lưu ý mảng `arr` phải thay đổi sau khi gọi hàm

```
const arr = [10, 12, 15];

shuffle(arr); // [12, 15, 10] => kết quả có thể khác nhau

console.log(arr); // [12, 15, 10]
```

8. Viết hàm `intersection(arr1, arr2)` nhận vào 2 mảng bất kỳ, trả về kết quả là một mảng mới chứa toàn bộ phần tử xuất hiện trong cả 2 mảng đó

```
intersection([1, 2, 3], [3, 4, 5]); // [3]
```

9. Viết hàm `difference(arr1, arr2)` nhận vào 2 mảng bất kỳ, trả về kết quả là một mảng mới chứa toàn bộ phần tử chỉ xuất hiện ở 1 trong 2 mảng

```
difference([1, 2, 3], [2, 3, 4]); // [1, 4]
```

0. Viết hàm `removeDuplicate(arr)` nhận vào 1 mảng bất kỳ, trả về kết quả là một mảng mới chứa các giá trị duy nhất (*unique* - không chứa các giá trị trung lặp) của mảng

```
removeDuplicate([1, 2, 5, 2, 3, 1, 3]); // [1, 2, 5, 3]
```

1. Viết hàm `filterRange(arr, a, b)` nhận vào một mảng số, trả về kết quả là một mảng mới chỉ chứa các số lớn hơn hoặc bằng `a` và nhỏ hơn hoặc bằng `b` (`a` nhỏ hơn `b`), gợi ý sử dụng phương thức `filter()`

```
const arr = [5, 3, 8, 1];

filterRange(arr, 1, 4); // [5, 3]
```

2. Viết hàm `getNames(users)` nhận vào một mảng các object, mỗi object có thông tin tên và tuổi, trả về một mảng mới chỉ chứa tên, gợi ý sử dụng phương thức `map()`

```
const users = [
  { name: "John", age: 25 },
  { name: "Pete", age: 30 },
  { name: "Mary", age: 28 }
];

getNames(users); // [ "John", "Pete", "Mary" ]
```

3. Viết hàm `mapFullname(users)` nhận vào một mảng các object, mỗi object có thông tin tên, tên đệm và id , trả về một mảng mới chứa các object tương tự, thay thế tên và tên đệm thành tên đây đủ (tên + tên đệm), gợi ý sử dụng phương thức `map()`

```
const users = [
  { name: "John", surname: "Smith", id: 1 },
  { name: "Pete", surname: "Hunt", id: 2 },
  { name: "Mary", surname: "Key", id: 3 }
];

mapFullname(users);
// kết quả
// [
//   { fullName: "John Smith", id: 1 },
//   { fullName: "Pete Hunt", id: 2 },
//   { fullName: "Mary Key", id: 3 }
// ]
```

4. Viết hàm `greaterThan(users, age)` nhận vào một mảng các object, mỗi object có thông tin tên và tuổi, và `age` là một số nguyên dương bất kỳ, trả về một mảng mới chỉ chứa các object có tuổi lớn hơn hoặc bằng `age`, gợi ý sử dụng phương thức `filter()`

```
const users = [
  { name: "John", age: 25 },
  { name: "Pete", age: 30 },
  { name: "Mary", age: 28 }
];

greaterThan(users, 29);
// kết quả
// [
//   { name: "Pete", age: 30 },
// ]
```

5. Viết hàm `avgAge(users)` nhận vào một mảng các object, mỗi object có thông tin tên và tuổi, tính tuổi trung bình và trả về kết quả, gợi ý sử dụng phương thức `reduce()`

```
const users = [
  { name: "John", age: 25 },
  { name: "Pete", age: 30 },
  { name: "Mary", age: 28 }
];

avgAge(users); // (25 + 30 + 29) / 3 = 28
```

6. Viết hàm `sortUsersByAge(users)` nhận vào một mảng các object, mỗi object có thông tin tên và tuổi, sắp xếp `users` theo độ tuổi giảm dần và trả về kết quả, lưu ý mảng `users` phải thay đổi sau khi gọi hàm, gợi ý sử dụng phương thức `sort()`

```
const users = [
  { name: "John", age: 25 },
  { name: "Pete", age: 30 },
  { name: "Mary", age: 28 }
];

sortUsersByAge(users);
// kết quả
// [
//   { name: "Pete", age: 30 },
//   { name: "Mary", age: 28 },
//   { name: "John", age: 25 }
// ]
```