Bài 1: Viết hàm kiểm tra object thứ 2 có phải là con của object thứ nhất hay không?

```
console.log(matches({ age: 25, hair: 'long', beard: true }, { hair: 'long', beard: true })); // true
console.log(matches({ hair: 'long', beard: true }, { age: 25, hair: 'long', beard: true })); // false
console.log(matches({ hair: 'long', beard: true }, { age: 26, hair: 'long', beard: true })); // false
```

Bài 2: Viết hàm chuyển 1 số thành mảng của các chữ số?

```
console.log(digitize(123));
console.log(digitize(1230));

// Sample output

[1,2,3]
[1,2,3,0]
```

Bài 3: Viết hàm lọc các phần tử trong mảng theo điều kiện cho trước, nếu không có điều kiện lọc thì trả về mảng ban đầu.

```
const pull = (arr, ...args) => {
};


let arra1 = ["a", "b", "c", "a", "b", "c"];

let arra2 = ["a", "b", "c", "a", "b", "c"];


console.log(pull(arra1, "a", "c"));
console.log(pull(arra2, "b"));

// Sample output

["b","b"]
["a","c","a","c"]
```

Bài 4: Viết chương trình lấy giá trị của mảng được cho bởi một khoảng cho trước:

```
const pull_at_Index = (arr, pullArr) => {

};


let arra1 = ["a", "b", "c", "d", "e", "f"];
let arra2 = [1, 2, 3, 4, 5, 6, 7];


console.log(pull_at_Index(arra1, [1, 3]));
console.log(pull_at_Index(arra2, [4]));

// Sample output
["b", "d"]
[5];
```

Bài 5: Viết hàm xoá n phần tử từ bên trái của một mảng cho trước:

```
function remove_from_left(arr, n = 1){
    return arr.slice(n);
 }
 console.log(remove_from_left([1, 2, 3]));
 console.log(remove_from_left([1, 2, 3], 1));
 console.log(remove_from_left([1, 2, 3], 2));
 console.log(remove_from_left([1, 2, 3], 4));
```

Bài 6: Viết hàm giữ lại n phần tử từ bên phải của một mảng cho trước:

```
function remove_from_right(arr, n = -1){
    return arr.slice(n);
 }

 console.log(remove_from_right([1, 2, 3]));
 console.log(remove_from_right([1, 2, 3], -1));
 console.log(remove_from_right([1, 2, 3], -2));
 console.log(remove_from_right([1, 2, 3], -4));
```

Bài 7: Viết hàm lọc các phần tử chia hết cho n cho trước trong mảng:

```
const every_nth = (arr, nth) => arr.filter((e, i) => i % nth === nth - 1);
console.log(every_nth([1, 2, 3, 4, 5, 6], 1));
console.log(every_nth([1, 2, 3, 4, 5, 6], 2));
console.log(every_nth([1, 2, 3, 4, 5, 6], 3));
console.log(every_nth([1, 2, 3, 4, 5, 6], 4));
```

Bài 17: Viết hàm lọc ra các phần tử không xuất hiện 2 lần trong mảng:

```
const filter_Non_Unique = arr =>
  arr.filter(i => arr.indexOf(i) === arr.lastIndexOf(i));

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

Bài 20: Viết hàm trả về true nếu string là y hoặc yes, trả về false nếu string là n hoặc no?

```
const yes_No = (val, def = false) =>
  /^(y|yes)$/i.test(val) ? true : /^(n|no)$/i.test(val) ? false : def;
console.log(yes_No("Y"));
console.log(yes_No("yes"));
console.log(yes_No("No"));
console.log(yes_No("Foo", true));
```

Bài 21: Viết hàm tính thời gian thực thi của một hàm:

```
const time_taken = callback => {
  const result = callback();
  return result;
};
console.log("Time taken: " + time_taken(() => Math.pow(2, 10)) + " ms");
console.log("Time taken: " + time_taken(() => Math.sqrt(225)) + " ms");
console.log(
  "Time taken: " + time_taken(() => Math.sqrt(5 * 5 + 6 * 6)) + " ms"
);
```

Bài 22: Viết hàm làm tròn số thực đến 0.5:

```
const to_Safe_Integer = num =>
  Math.round(
    Math.max(Math.min(num, Number.MAX_SAFE_INTEGER), Number.MIN_SAFE_INTEGER)
  );
console.log(to_Safe_Integer("5.2"));
console.log(to_Safe_Integer("5.52"));
console.log(to_Safe_Integer(Infinity));
```

Bài 23: Viết hàm lọc mảng từ mảng cho trước (lọc ra các số không xuất hiện trong mảng điều kiện):

```
const without = (arr, ...args) => arr.filter(v => !args.includes(v));

console.log(without([2, 1, 2, 3], 1, 2));
console.log(without([2, 1, 2, 3], 3));
```

Bài 24: Viết hàm bỏ đi phần tử đầu tiên của mảng, nếu mảng chỉ có 1 phần tử thì lấy phần tử đầu tiên:

```
const tail = arr => (arr.length > 1 ? arr.slice(1) : arr);
console.log(tail([1, 2, 3]));
console.log(tail([1]));
```

Bài 25: Viết hàm trả về một số ngẫu nhiên trong khoảng cho trước:

```
const random_Number_In_Range = (min, max) => Math.random() * (max - min) + min;
console.log(random_Number_In_Range(2, 10));
console.log(random_Number_In_Range(1, 5));
console.log(random_Number_In_Range(-5, -2));
console.log(random_Number_In_Range(0, 1));
```

Bài 26: Viết hàm trả về một số nguyên ngẫu nhiên trong khoảng cho trước:

```
const randomIntegerInRange = (min, max) =>
  Math.floor(Math.random() * (max - min + 1)) + min;
console.log(randomIntegerInRange(0, 5));
console.log(randomIntegerInRange(2, 5));
console.log(randomIntegerInRange(5, -5));
console.log(randomIntegerInRange(-2, -7));
```

Bài 27: Viết hàm trả về một mảng n phần tử nguyên được random ngẫu nhiên trong một khoảng cho trước:

```
const random_intArray_In_Range = (min, max, n = 1) =>
  Array.from(
    { length: n },
    () => Math.floor(Math.random() * (max - min + 1)) + min
  );
console.log(random_intArray_In_Range(1, 20, 10));
console.log(random_intArray_In_Range(-10, 10, 5));
```

Bài 28: Viết hàm xoá các cặp key-value từ một mảng cho trước:

```
const omit = (obj, arr) =>
  Object.keys(obj)
    .filter(k => !arr.includes(k))
    .reduce((acc, key) => ((acc[key] = obj[key]), acc), {});

console.log(omit({ a: 1, b: "2", c: 3 }, ["b"]));
console.log(omit({ a: 1, b: 2, c: 3 }, ["c"]));
```

Bài 29: Viết hàm trả về một mảng với các phần tử là mảng chứa key-value của object:

```
const object_to_pairs = obj => Object.keys(obj).map(k => [k, obj[k]]);
console.log(object_to_pairs({ a: 1, b: 2 }));
console.log(object_to_pairs({ a: 1, b: 2, c: 3 }));
```

Bài 30: Viết hàm chuyển ngược kết quả của câu trên thành object:

```
const object_From_Pairs = arr =>
  arr.reduce((a, v) => ((a[v[0]] = v[1]), a), {});
console.log(object_From_Pairs([["a", 1], ["b", 2]]));
console.log(object_From_Pairs([[1, 10], [2, 20], [3, 30]]));
```

Bài 31: Viết hàm xoá đi các falsy value từ mảng cho trước:

```
const compact = arr => arr.filter(Boolean);
console.log(compact([0, 1, false, 2, "", 3, "a", "e" * 23, NaN, "s", 34]));
console.log(compact([false, NaN, "e" * 23]));
```

Bài 32: Viết hàm nhập vào 1 số bé hơn 25 và chuyển đổi thành giờ dạng am hoặc pm:

```
const get_Meridiem_Suffix_Of_Integer = num =>
  num === 0 || num === 24
    ? 12 + "am"
    : num === 12
    ? 12 + "pm"
    : num < 12
    ? (num % 12) + "am"
    : (num % 12) + "pm";

console.log(get_Meridiem_Suffix_Of_Integer(0));
console.log(get_Meridiem_Suffix_Of_Integer(11));
console.log(get_Meridiem_Suffix_Of_Integer(13));
console.log(get_Meridiem_Suffix_Of_Integer(25));
```

Bài 33: Viết hàm nhận vào một url và chuyển các url parameter thành object:

```
const get_URL_Parameters = url =>
  (url.match(/([^?=&]+)(=([^&]*))/g) || []).reduce(
    (a, v) => (
      (a[v.slice(0, v.indexOf("="))] = v.slice(v.indexOf("=") + 1)), a
    ),
    {}
  );
console.log(get_URL_Parameters("http://url.com/page?name=Adam&surname=Smith"));
console.log(get_URL_Parameters("google.com"));
console.log(get_URL_Parameters("https://www.w3resource.com"));
```

Bài 34: Viết hàm khởi tạo 1 mảng 2 chiều từ 3 tham số, chiều dài, chiều rộng và giá trị của mỗi phần tử:

```
const initialize_2D_Array = (w, h, val = null) =>
  Array.from({ length: h }).map(() => Array.from({ length: w }).fill(val));

console.log(initialize_2D_Array(2, 2, 0));
console.log(initialize_2D_Array(3, 3, 3));
```

Bài 35: Viết hàm khởi tạo mảng theo các tham số: bắt đầu, kết thúc, và bước nhảy, trong đó bước nhảy là khoảng cách giữa các phần tử:

```
const initialize_Array_With_Range = (end, start = 0, step = 1) =>
  Array.from({ length: Math.ceil((end + 1 - start) / step) }).map(
    (v, i) => i * step + start
  );

console.log(initialize_Array_With_Range(5));
console.log(initialize_Array_With_Range(8, 3));
console.log(initialize_Array_With_Range(6, 0, 2));
```

Bài 36: Viết hàm kiểm tra xem các phần tử trong mảng có bằng nhau hay không?

```
const allEqual = arr => arr.every(val => val === arr[0]);
console.log(allEqual([1, 2, 3, 4, 5, 6]));
console.log(allEqual([12, 12, 12, 12]));
```

Bài 37: Viết hàm gom các tham số đưa vào thành mảng:

```
const castArray = val => (Array.isArray(val) ? val : [val]);
console.log(castArray("w3r"));
console.log(castArray([100]));
```

Bài 38: Viết hàm chuyển các phần tử trong mảng theo một điều kiện cho trước và đếm số phần tử sau khi nhóm của từng loại:

```
const countBy = (arr, fn) =>
  arr
    .map(typeof fn === "function" ? fn : val => val[fn])
    .reduce((acc, val, i) => {
      acc[val] = (acc[val] || 0) + 1;
      return acc;
    }, {});
console.log(countBy([6, 10, 100, 10], Math.sqrt));
console.log(countBy([6.1, 4.2, 6.3], Math.floor));
console.log(countBy(["one", "two", "three"], "length"));
```

Bài 39: Viết hàm đếm số lần xuất hiện của một phần tử cho trước trong mảng:

```
const countOccurrences = (arr, val) =>
  arr.reduce((a, v) => (v === val ? a + 1 : a), 0);
console.log(countOccurrences([1, 1, 2, 1, 2, 3], 1));
console.log(countOccurrences([1, 1, 2, 1, 2, 3], 2));
console.log(countOccurrences([1, 1, 2, 1, 2, 3], 3));
```

Bài 40: Viết hàm thực hiện deepClone object: deep clone là sao chép giá trị của object đã cho và gán vào một object mới với địa chỉ mới trên RAM:

```
const deepClone = obj => {
  let clone = Object.assign({}, obj);
  Object.keys(clone).forEach(
    key =>
      (clone[key] =
        typeof obj[key] === "object" ? deepClone(obj[key]) : obj[key])
  );
  return Array.isArray(obj)
    ? (clone.length = obj.length) && Array.from(clone)
    : clone;
};
const a = { foo: "bar", obj: { a: 1, b: 2 } };
const b = deepClone(a); // a !== b, a.obj !== b.obj
console.log(b);
```

Bài 41:

Viết hàm chạy trên browser kiểm tra xem web đang được chạy trên môi trường nào:

```
const detectDeviceType = () =>
  /Android|webOS|iPhone|iPad|iPod|BlackBerry|IEMobile|Opera Mini/i.test(
    navigator.userAgent
  )
    ? "Mobile"
    : "Desktop";
console.log(detectDeviceType()); // "Mobile" or "Desktop"
```

Bài 42: Viết hàm so sánh sự khác nhau giữa 2 mảng và thực hiện một function ứng với từng phần tử của mảng kết quả:

```
const differenceBy = (a, b, fn) => {
  const s = new Set(b.map(v => fn(v)));
  return a.filter(x => !s.has(fn(x)));
};
console.log(differenceBy([2.1, 1.2], [2.3, 3.4], Math.floor));
console.log(differenceBy([{ x: 2 }, { x: 1 }], [{ x: 1 }], v => v.x));
```

Bài 42: Viết hàm forEach nhưng duyệt các phần tử từ phải sang trái:

```
const forEachRight = (arr, callback) =>
  arr
    .slice(0)
    .reverse()
    .forEach(callback);
forEachRight([1, 2, 3, 4], val => console.log(val));
```

Bài 43: Viết chương trình trả về mảng có độ dài lớn nhất từ một danh sách các mảng được truyền vào:

```
const longestItem = (...vals) =>
  [...vals].sort((a, b) => b.length - a.length)[0];
console.log(longestItem("this", "is", "a", "testcase"));
console.log(longestItem(...["a", "ab", "abc"]));
console.log(longestItem(...["a", "ab", "abc"], "abcd"));
console.log(longestItem([1, 2, 3], [1, 2], [1, 2, 3, 4, 5]));
console.log(longestItem([1, 2, 3], "foobar"));
```

Bài 44: Viết hàm thực hiện chuyển thành chữ hoa từng từ trong chuỗi:

```
const mapString = (str, fn) =>
  str
    .split("")
    .map((c, i) => fn(c, i, str))
    .join("");

console.log(mapString("Javascript exercises", c => c.toUpperCase()));
```

Bài 45: Viết hàm giữ lại n kí tự từ chuỗi cho trước, các kí tự còn lại thay bằng kí tự mask cho trước

```
const mask = (cc, num = 4, mask = "*") =>
  ("" + cc).slice(0, -num).replace(/./g, mask) + ("" + cc).slice(-num);

console.log(mask(1234567890));
console.log(mask(1234567890, 3));
console.log(mask(1234567890, -4, "$"));
```

;

Bài 46: Viết hàm chuyển n phần tử của mảng về cuối mảng:

```
const offset = (arr, offset) => [...arr.slice(offset), ...arr.slice(0, offset)];
console.log(offset([1, 2, 3, 4, 5], 2));
console.log(offset([1, 2, 3, 4, 5], -2));
```

Bài 47: Viết hàm truncate string: Truncate là viết string cho sẵn thành một string mới có độ dài n kí tự trong đó có chứa 3 dấu chấm ví dụ kết quả của hàm đáp án sau khi thực thi là: boom...

```
const truncateString = (str, num) =>
  str.length > num ? str.slice(0, num > 3 ? num - 3 : num) + "..." : str;

console.log(truncateString("boomerang", 7));
```

Bài 48: Viết hàm chuyển số cho trước thành đơn vị tiền tệ của quốc gia cho trước:

```
const toCurrency = (n, curr, LanguageFormat = undefined) =>
  Intl.NumberFormat(LanguageFormat, {
    style: "currency",
    currency: curr
  }).format(n);

console.log(toCurrency(123456.789, "EUR")); // currency: Euro | currencyLangFormat
console.log(toCurrency(123456.789, "USD", "en-us")); // currency: US Dollar | curr
console.log(toCurrency(123456.789, "USD", "fa")); //currency: US Dollar | currency
console.log(toCurrency(322342436423.2435, "JPY")); //currency: Japanese Yen | curr
console.log(toCurrency(322342436423.2435, "JPY", "fi")); //currency: Japanese Yen
```

Bài 49: Viết hàm trả về một mảng có n phần tử cho trước, nếu n nhập vào lớn hơn số phần tử thì mảng giữ nguyê, nếu n bé hơn số phần tử thì mảng sẽ được xoá từ bên trái các phần tử cho đến khi số lượng phần tử của mảng bằng với n:

```
const take = (arr, n = 1) => arr.slice(0, n);
console.log(take([1, 2, 3], 5));
console.log(take([1, 2, 3], 0));
```

Bài 50: Viết hàm chuyển văn bản text có kí tự xuống hàng thành mảng các phần tử:

```
const splitLines = str => str.split(/\r?\n/);
console.log('Original string:');
console.log('This\nis a\nmultiline\nstring.\n');
console.log(splitLines('This\nis a\nmultiline\nstring.\n'));
```