

ÔN LẠI JAVASCRIPT

Contents

1. Phân biệt sự khác nhau giữa var, let và const trong lập trình ES6	2
a. Ví dụ 1: biến var	2
b. Ví dụ 2: biến let	2
c. Ví dụ 3: biến const.....	3
d. " use strict" hay strict mode (chế độ nghiêm ngặt), hay hiểu đơn giản thì "use strict" là chế độ giúp chúng ta viết code Javascript trở nên an toàn hơn.....	4
e. Bài tập	4
2. Array (mảng)	4
a. Khai báo có thể khởi tạo array:	4
b. Một số phương thức cơ bản của array	5
c. Phương thức của mảng trong ES6	7
d. Mảng nhiều chiều	8
e. Duyệt các phần tử trong mảng	8
3. Objects	8
4. Arrow Function	10
a. Cú pháp nhiều tham số	10
b. Không có tham số.....	11
c. Cú pháp Object literals.....	11
d. Callback	12
5. Rest parameters và spread-operator	12
a. Destructuring	12
b. Rest parameter.....	12
c. Spread	13
6. Bài tập 1: tạo 1 đối tượng với các thuộc tính (person)	13
7. Bài tập 2:	15
8. Bài tập 3:	16

1. Phân biệt sự khác nhau giữa var, let và const trong lập trình ES6

- Khai báo var được định phạm vi toàn cục (global) hay hàm (function) trong khi let và const được định phạm vi là khối mã (block)
- Biến var có thể được cập nhật và khai báo lại trong phạm vi tồn tại. Biến let có thể được cập nhật nhưng không thể khai báo lại. Biến const không thể cập nhật và không thể khai báo lại.
- Khai báo của var, let, const đều được dịch chuyển lên đầu của phạm vi (**hoisting**). Nhưng trong khi biến var được khởi gán giá trị với undefined, biến let và const không được khởi gán giá trị.
- Trong khi var và let có thể được khai báo không khởi gán giá trị, const phải khởi gán giá trị khi khai báo.

a. Ví dụ 1: biến var

Khai báo	<pre>console.log (greeting); var greeting = "say hello";</pre>
Sẽ được biên dịch là	<pre>var greeting; console.log(greeting); // greeting is undefined greeting = "say hello";</pre>
Có một vấn đề với var (vẫn giữ nguyên giá trị sau khi thoát khỏi block)	<pre>var greeting = "say hi"; var times = 4; if (times > 3) { var greeting = "say Hello instead"; } console.log(greeting); //"say Hello instead"</pre>
??	<pre>console.log(fname); var fname='to an an'; console.log("sum:",sum(2,6)); function sum(a,b){ return (a+b); }</pre>

b. Ví dụ 2: biến let

<ul style="list-style-type: none"> - Biến let được khai báo sẽ có scope là block scoped - var được khởi tạo với giá trị là undefined, let không khởi tạo giá trị 	<pre>let greeting = "say Hi"; let times = 4; if (times > 3) { let hello = "say Hello instead"; console.log(hello); // "say Hello instead" } console.log(hello); // hello is not defined</pre>
let cho phép cập nhật giá trị của biến chứ không cho phép tái khai báo lại biến đó.	<pre>let greeting = "say Hi"; console.log(greeting); //"say Hi" greeting = "say Hello instead"; console.log(greeting); //"say Hello instead" let greeting = "say Hi";//error:Identifier //'greeting' has already been declared</pre>

đối với từng scope, mỗi biến sẽ được xem xét là biến riêng khác biệt	<pre>let greeting = "say Hi"; if (true) { let greeting = "say Hello instead"; console.log(greeting); //"say Hello instead" } console.log(greeting); // "say Hi"</pre>
--	---

c. Ví dụ 3: biến const

const cũng có scope là block scoped	<pre>const greeting = "say Hi"; greeting = "say Hello instead"; // error : //Assignment to constant variable. const greeting1 = "say Hi"; const greeting1 = "say Hello instead"; // error : //Identifier 'greeting' has already been declared</pre>
Đối với trường hợp kiểu biến là reference(object, array và function) thì không thể tái khai báo hay cập nhật giá trị của biến nhưng có thể cập nhật giá trị cho thuộc tính của biến đó.	<pre>const greeting = { message : "Hello", number : "five" } greeting.message = "say Hello instead"; console.log(greeting); // {message:"say Hello //instead",number:"five"}</pre>
Nếu không muốn thay đổi các giá trị trong thuộc tính, thì đóng băng đối tượng – Hằng đối tượng	<pre>const person1 = Object.freeze({ name: 'Hung', age: 10, }) // Thay đổi giá trị cho các thuộc tính của đối tượng person person1.name = 'Son' person1.age = 20 console.log("name: " + person1.name + ", age: " + person1.age);</pre>
Vẫn thay đổi được giá trị của thuộc tính address	<pre>const person3 = Object.freeze({ name: 'TrungNQ', address: { state: 'Hai Ba Trung', city: 'Ha Noi', zipcode: 10000 } }); // Thay đổi dữ liệu address của person3 person3.address.city = 'Ha Loi' person3.name = 'Ngo Quang Trung' console.log(person3);</pre>

- d. "use strict" hay strict mode (chế độ nghiêm ngặt), hay hiểu đơn giản thì "use strict" là chế độ giúp chúng ta viết code Javascript trở nên an toàn hơn.

<pre>fname='to an an'; function test(){ age=20; } test(); console.log(fname); console.log(age);</pre>	<pre>let student=Object.freeze({ fname:'To an An' }); student.fname='nguyen van A'; console.log(student.fname);</pre>	<pre>{ function sum(a,b){ return (a+b); } } console.log(sum(3,6));</pre>
---	---	--

e. Bài tập

Bài tập 1: chuyển dãy số có "." Thành phẩy ",". Ví dụ: '555.666.9990' -> '555,666,9990'

Bài tập 2: tạo 1 hàm sinh ra đối tượng kiểu <DIV>

<div id="root"></div>	Ví dụ cho div dùng id
<div class="title"></div>	Ví dụ cho div dùng class
Dùng createElement	<div style="border: 1px solid red; padding: 10px; text-align: center;"> FPT là trường đại học ở Hoà Lạc </div>

2. Array (mảng)

Khởi tạo array: có hai cách để khởi tạo array rỗng

Sử dụng dấu ngoặc vuông []	let a1 = [];
Sử dụng [] là phổ biến hơn cả vì cách này nhanh và ngắn gọn hơn	
Sử dụng hàm khởi tạo new Array()	let a2 = new Array();

a. Khai báo có thể khởi tạo array:

Khai báo	<pre>let a; a1 = [val1, val2, ..., valn]; Hoặc: let a2 = [val1, val2, ...valn]; let a3=[23,0,56,6,8,9]; let a4=['Thu an','Tran ha','vu thu bach'];</pre>
Để truy cập các phần tử trong mảng, sử dụng arr[index] với index là chỉ số của phần tử	<pre>let a = ["a", "b", "c"]; console.log(a[0]); // a // trường hợp chỉ số ngoài phạm vi giới hạn thì kết quả là undefined console.log(a[-1]); // undefined</pre>

	<pre>console.log(a[3]); // undefined // thay đổi giá trị của mảng tại chỉ số 0 a[0] = "A"; console.log(a[0]); // A // thêm phần tử vào mảng tại chỉ số 3 a[3] = "d"; console.log(a[3]); // d</pre>
Duyệt mảng	<pre>let numbers = [10, 20, 30]; for (const n of numbers) { console.log(n+5); }</pre>
Đối tượng mảng	<pre>var arr = new Array(4) for(var i = 0; i<arr.length; i++) { arr[i] = i * 2 console.log(arr[i]) }</pre>
	<pre>let arr = new Array("JavaScript","PHP","Laravel","Vue.js"); for(let i = 0; i<arr.length; i++) { console.log(arr[i]); }</pre>
<ul style="list-style-type: none"> - Giá trị các phần tử trong mảng có thể thuộc bất kỳ kiểu dữ liệu nào - Tương tự như object, array có thể có hoặc không dùng "dấu phẩy đuôi" - dấu phẩy sau phần tử cuối cùng. 	<pre>let arr = ["a", // string 1, // number null, // null undefined, // undefined { x: 1 }, // object function () { // hàm console.log("hello"); },]; // giá trị tại chỉ số 4 là object { x: 1 } console.log(arr[4].x); // 1 // giá trị tại chỉ số 5 là hàm arr[5](); // hello</pre>

b. Một số phương thức cơ bản của array

Phương thức arr.pop() Lấy ra và trả về phần tử cuối cùng của mảng	<pre>let a = ['a', 'b', 'c']; let item = a.pop(); console.log(item); // c console.log(a); // (2) ['a', 'b']</pre>
Phương thức arr.push() Thêm một hoặc nhiều phần tử vào cuối mảng và trả về độ dài mới của mảng	<pre>let a = ['a', 'b', 'c']; // thêm một phần tử vào cuối mảng let length = a.push('d'); console.log(length); // 4 console.log(a); // (4) ['a', 'b', 'c', 'd'] // thêm nhiều phần tử vào cuối mảng</pre>

	<pre>length = a.push('e', 'f'); console.log(length); // 6 console.log(a); // (6) ['a', 'b', 'c', 'd', 'e', 'f']</pre>
Phương thức arr.shift() Lấy ra và trả về phần tử đầu tiên của mảng	<pre>let a = ['a', 'b', 'c']; let item = a.shift(); console.log(item); // a console.log(a); // (2) ['b', 'c']</pre>
Phương thức arr.unshift() Thêm một hoặc nhiều phần tử vào đầu mảng	<pre>let a = ['a', 'b', 'c']; // thêm một phần tử vào đầu mảng let length = a.unshift('d'); console.log(length); // 4 console.log(a); // (4) ['d', 'a', 'b', 'c'] // thêm nhiều phần tử vào đầu mảng length = a.unshift('e', 'f'); console.log(length); // 6 console.log(a); // (6) ['e', 'f', 'd', 'a', 'b', 'c']</pre>
concat()	<pre>let a = ['a', 'b', 'c']; let b = [1, 2, 3]; let con = b.concat(a); console.log(con); //1,2,3,a,b,c</pre>
every()	<pre>function check(x) { return (x >= 10); } let result = [12, 5, 8, 130, 44].every(check); console.log(result); //false</pre>
filter()	<pre>//Lọc những phần tử >=10 function filt(element) { return (element >= 10); } let passed = [12, 5, 8, 130, 44].filter(filt); console.log("Result : " + passed); // Result : 12,130,44</pre>
forEach() : gọi mỗi hàm cho mỗi phần tử	<pre>let arr = new Array(12,13,14,15); console.log("In mảng gốc....."); arr.forEach(function(val,index) { console.log(val) }); arr.reverse(); //đảo ngược mảng console.log("In mảng sau khi đảo ngược...."); arr.forEach(function(val,index){ console.log(val) });</pre>
indexOf()	<pre>let in = [12, 5, 8, 130, 44, 8].indexOf(8); console.log("Kết quả : " + in); //Kết quả : 2</pre>
join()	<pre>let arr = new Array("FPT","la","truong toi"); let str = arr.join("-"); console.log(str); //FPT-la-truong toi</pre>
lastIndexOf()	<pre>let in = [12, 5, 8, 130, 44, 8].lastIndexOf(8);</pre>

	<code>console.log("Kết quả : " + in) //Kết quả : 5</code>
map(): Tạo 1 mảng mới với kết quả là lời gọi 1 hàm được cung cấp trên mỗi PT mảng này	<code>let arr = [1, 4, 9]; let newArr = arr.map(Math.sqrt); console.log("Result : " + newArr); //Result : 1,2,3</code>
reduce(): áp dụng đồng thời 1 hàm đối với 2 giá trị của mảng (từ trái sang phải) thành 1 giá trị	<code>let sum = [0, 1, 2, 3].reduce(function(x, y){return x + y;}); console.log("Sum : " + total); //Sum : 6</code>
reduceRight()	<code>let sum = [0, 1, 2, 3].reduceRight(function(x, y){return x + y;}); console.log("Sum : " + total); //Sum : 6</code>
slice(): trích xuất 1 PT của mảng và trả về 1 mảng mới	<code>let arr = ["FPT","la","truong toi", "hoc", "khi thanh xuan"]; console.log(arr.slice(0, 2)); //FPT,la console.log(arr.slice(1, 3)); //la,truong toi</code>
some(): trả về true nếu có ít nhất 1 PT trong mảng thỏa mãn hàm kiểm tra	<code>function check(element) { return (element >= 10); } let result1 = [1, 2, 3, 5, 8].some(check); console.log(result1); //false let result2 = [1, 1, 2, 3, 5, 8, 13].some(check); console.log(result2); //true</code>
toString(): Phương thức này trả về một string chứa giá trị của các phần tử mảng - được ngăn cách nhau bởi dấu phẩy	<code>let arr = new Array("đỏ", "da cam", "vàng", "lục"); let str = arr.toString(); console.log(str); //đỏ,da cam,vàng,lục</code>

c. Phương thức của mảng trong ES6

find(): lặp qua 1 mảng và lấy phần tử đầu tiên phù hợp	<code>let arr = [1, 2, 3]; let firstOdd = arr.find((x) => x % 2 == 1); console.log(firstOdd); // 1</code>
findIndex(): chỉ số	<code>let numbers = [4, 6, 3, 5]; let oddNumber = numbers.findIndex((x) => x % 2 == 1); console.log(oddNumber); // 2</code>
entries(): trả về bộ lặp mảng, lặp qua khóa (key) và giá trị (value)	<div> <code>let arr = [2, 4, 6]; let val = arr.entries(); console.log('Value 0:',val.next().value); for (let i of val) { console.log(i); }</code> </div> <div>Value 0:[0,2] [1,4] [2,6]</div>
Sử dụng toán tử spread	<div> <code>var numbers = [1, 2, 3]; var val= numbers.entries(); console.log(...val);</code> </div> <div>[0,1],[1,2],[2,3]</div>

Array.from() : cho phép tạo mảng mới từ đối tượng mảng	//không nên dùng for (let x of Array.from('VlStudy')) { console.log(x) }	V 1
keys() : trả về chỉ mục của mảng	console.log(Array.from(['x', 'y', 'z'].keys()))	[0, 1, 2]

d. Mảng nhiều chiều

mảng hai chiều để lưu ma trận 2x2	<pre> let matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9],]; // truy cập phần tử ở hàng 1, cột 1 console.log(matrix[1][1]); // 5 let sum=0; for (let i = 0; i < matrix.length; i++) { for (let j = 0; j < matrix[0].length; j++) { sum+=matrix[i][j]; } } console.log(sum); </pre>
-----------------------------------	--

e. Duyệt các phần tử trong mảng

Cách cơ bản nhất để duyệt tất cả các phần tử của mảng	<pre> let letters = ["a", "b", "c"]; for (let i = 0; i < letters.length; i++) { console.log(letters[i]); } </pre>
Nếu không quan tâm đến chỉ số, bạn có thể dùng vòng lặp for...of	<pre> let letters = ["a", "b", "c"]; for (let value of letters) { console.log(value); } </pre>

3. Objects

JavaScript có 5 kiểu dữ liệu cơ bản: Number, String, Boolean, Undefined và Null và còn 1 kiểu khác nữa là Object (hay còn gọi là kiểu phức hợp)

Creating Objects: There are 3 ways to create objects.

- By object literal
- By creating instance of Object directly (using new keyword)
- By using an object constructor (using new keyword)

by object literal	<pre> let object={property1:value1,property2:value2.....propertyN:valueN}; let person = { id: 1, name: 'To an An', age: 20 }; console.log(person); console.log("ID: " + person.id + ', Name: ' + person.name + ', Age: ' + person.age); //Array-like notation ([]) console.log(person['name']); person.name = 'Jane'; console.log(person); //Adding a new property to an object person.gender = true; //Deleting a property of an object delete person.gender; //Checking if a property exists console.log('ssn' in person);//false console.log('id' in person);//true </pre>
By creating instance of Object	<pre> let objectname=new Object(); let emp=new Object(); emp.id=101; emp.name='Ravi Malik'; emp.salary=50000; </pre>
By using an Object constructor The this keyword refers to the current object.	<pre> function emp(id,name,salary){ this.id=id; this.name=name; this.salary=salary; } let e=new emp(103,"Vimal Jaiswal",30000); </pre>
Defining method in JavaScript Object	<pre> function emp(id,name,salary){ this.id=id; this.name=name; this.salary=salary; this.changeSalary=changeSalary; function changeSalary(otherSalary){ this.salary=otherSalary; } } let e=new emp(103,"Sonoo Jaiswal",30000); console.log(e); e.changeSalary(45000); console.log(e); </pre>
	<pre> let fName='an'; let age=20; let person={ fName, // fName:fName, age //age:age }; console.log(person); </pre>
	<pre> let fName='an'; let age=20; </pre>

	<pre> let person={ fName, // fName:fName, age, //age:age // getName:function(){ // return fName; // } getName(){ return fName; } }; console.log(person.getName()); </pre>
Se6 (dùng từ khoá class)	<pre> class Student { constructor(id, name, age, gender) { this.id = id; this.name = name; this.age = age; this.gender = gender; } } new Student(1, "Nam", 20, true) </pre>
Viết ra mảng đối tượng	<pre> let students = [{id: 1, name: 'To an An'}, {id: 2, name: 'Vu Thi Teo'}, {id: 3, name: 'Luu Van Bao'}]; for (const s of students) { console.log(`\${s.id}\t\${s.name}`) } for (const {id:i, name:n} of students) { console.log(`\${i}\t\${n}`) } </pre>

4. Arrow Function

Arrow functions là một trong những tính năng mới rất hay của ES6. Việc sử dụng đúng cách arrow function giúp code trở nên ngắn gọn và dễ hiểu hơn.

a. Cú pháp nhiều tham số

ES5	<pre> var testEs5 = function(a, b) { return a * b; }; </pre>
ES6	<pre> const testEs6 = (a, b) => { return a * b }; const testEs6=(a,b)=>(a*b); </pre>

Cú pháp cơ bản với một tham số

ES5	<pre> var splitterEs5 = function splitter(string) { return string.split(' '); }; </pre>
ES6	<pre> const splitterEs6 = string => string.split(" "); </pre>

	<code>console.log(splitterEs6("fpt university")); // ["fpt", "university"]</code>
ES5	<code>function Say(msg) { msg = typeof(msg) !== 'undefined' ? msg : 'Hihi' console.log(msg); } // Gọi hàm để thực thi Say(undefined) Say(undefined) Say("Hello")</code>
ES6	<code>const say1=(msg='hele')=>console.log(msg); // function say1(msg='hehe') { // console.log(msg); // } say1(); say1(undefined); say1("Hello");</code>
Giá trị mặc định của tham số là 1 hàm khác	<code>// function printDate(d = showDate()){ // console.log("current date: " + d); // } // function showDate(){ // return new Date().toLocaleDateString("en-US"); // } const printDate=(d = showDate())=>console.log("current date: " + d); const showDate=()=>new Date().toLocaleDateString("en-US"); printDate(); printDate('09/02/2023');</code>
Giá trị mặc định là một biểu thức tính toán giữa tham số đã có với 1 hàm đã có	<code>let rate = () => 0.1; // rate là 1 biến được gán cho một hàm không có //tham số và trả về giá trị là 0.1 let getPrice = (price, tax = price*rate()) => price + tax; console.log("Full Price: " + getPrice(100));</code>

b. Không có tham số

ES5	<code>var logEs5 = function logger() { console.log(document); };</code>
ES6	<code>var logEs6 = () => { console.log(document); }; logEs6();</code>

c. Cú pháp Object literals

ES5	<code>var setNameEs5 = function setName(id, name) { return { id: id, name: name</code>
-----	--

	}; };
ES6	let setNameEs6 = (id, name) => ({ id: id, name: name }); console.log(setNameEs6 (4, "Van Quy")); // Object {id: 4, name: "Van Quy"}

d. Callback

- Là hàm (Function) được truyền qua đối số
- Khi gọi hàm khác
- Gọi lại (trong hàm nhận đối số)

```
function myFun(param) {
    if(typeof param==='function' ){
        param('yeu fpt');
    }
}
function myCallBack(value){
    console.log("say:",value);
}
myFun(myCallBack);
```

5. Rest parameters và spread-operator

a. Destructuring

let arr=['van','anh','thu']; let a=arr[0]; let b=arr[1]; let c=arr[2]; console.log(a,b,c);	let arr=['van','anh','thu']; let [a,b,c]=arr; console.log(a,b,c);	let [a,,c]=arr; console.log(a,b,c);//error console.log(a,c);//ok
let colors = ['red', 'pink', 'yellow']; for (const [index, value] of colors.entries()) { console.log(`\${value} at index \${index}`); }		

b. Rest parameter

	let arr=['van','anh','thu']; let [a,...rest]=arr; console.log(rest);	
let o={ fname:'anh', age:20	let o={ fname:'anh', age:20,	let o={ fname:'anh', age:20,

<pre> } let {fname,age}=o; console.log(fname,age); </pre>	<pre> gender:true } let {fname,...rest}=o; console.log(fname); console.log(rest); </pre>	<pre> gender:true } let {fname,...newObject}=o; console.log(newObject); //xoa fname </pre>
---	--	--

Chú ý: rest là tham số của hàm (phần còn lại), còn lại là spread

Ví dụ:

<pre> function logger(...params){ console.log(params); } logger(1,2,3,4,5); </pre>	<pre> function logger(a,...params){ console.log(params); } logger(1,2,3,4,5); //bo 1 </pre>
--	---

Bài tập : Tính tổng dãy số, ví dụ: (1, 2, 3, 4) – 10; ("fpt", true, 3, 1, "keke",10) – 14

c. Spread

<pre> let a1=[1,2,3]; let a2=[...a1,8,9]; //1,2,3,8,9 let a3=[...a1,20,...a2]; //1,2,3,20,1,2,3,8,9 </pre>	<pre> let o1={ fname:'an' } let o2={ age:20 } let o3={ ...o1,...o2 } </pre>	<pre> let defaultConfig={ api:'https://fpt.edu.vn/', version:'v1', other:'other' } let hnConfig={ ...defaultConfig, api:'https://hanoi.fpt.edu.vn/' } console.log(hnConfig) </pre>
--	---	--

6. Bài tập 1: tạo 1 đối tượng với các thuộc tính (person)

Định nghĩa đối tượng person (id,name,age)

<pre> let person = { id: 1, name: 'To an An', age: 20 }; console.log("ID: " + person.id + ', Name: ' + person.name + ', Age: ' + person.age); </pre>
--

Tạo 1 mảng chứa 3 đối tượng với các thuộc tính giống person, bổ sung thêm gender (boolean), sử dụng vòng lặp viết ra danh sách có định dạng như sau:

Id	Name	Age	Gender
1	Dung	20	Male
2	Nam	19	Male
3	Hong	21	Female

Sử dụng unshift để đẩy đối tượng vào vị trí đầu tiên

Lọc và hiển thị với các person có name bắt đầu bởi chuỗi 'a'

```

let personsArray = [
  { id: 1, name: 'An', age: 21, gender: true },
  { id: 2, name: 'Van', age: 20, gender: true },
  { id: 3, name: 'Tuan', age: 22, gender: false }
];

let printArray = (arr) => {
  console.log('ID\tName\tAge\tGender');
  arr.forEach(person => {
    console.log(person.id + '\t' + person.name + '\t' +
person.age + '\t' + (person.gender ? "Male" : "Female"));
  });
};

let printArray2 = (arr) => {
  console.log('ID\tName\tAge\tGender');
  arr.forEach((e) => {
    console.log(`${e.id} \t ${e.name} \t ${e.age} \t
${e.gender?"Male":"Female"}`);
  });
};

printArray(personsArray);

let newPerson = { id: 4, name: 'anh', age: 19, gender: false }
personsArray.push(newPerson); // Su dung unshift de day doi tuong vao
vi tri dau tien
// Loc va hien thi voi cac person co name bat dau boi chuoi 'a'
let resultFilter = personsArray.filter(person =>
  person.name.toLocaleLowerCase().startsWith('a')
)
printArray(resultFilter)

```

Sắp xếp mảng đối tượng:

Number	<pre> let sortResult = personsArray.sort((a, b) => { return a.age - b.age; }) let sortDescending = personsArray.sort((a, b) => b.age - a.age); printArray(sortResult) </pre>
String	<pre> let sort1 = personsArray.sort((a, b) => { let fa = a.name.toLowerCase(), fb = b.name.toLowerCase(); if (fa < fb) { return -1; } if (fa > fb) { return 1; } return 0; }) printArray2(sort1) // String localeCompare(String): so sanh tra ve 0, 1, -1 let sort2 = personsArray.sort((a, b) => { return a.name.localeCompare(b.name) </pre>

	<code>})</code>
Theo 2 thuoc tinh	<code>//tang dan theo name or giam dan theo age let sort3 = personsArray.sort((a, b) => { return b.name.localeCompare(a.name) a.age - b.age })</code>
Date	<code>employees.sort((a, b) => { let da = new Date(a.dob), db = new Date(b.dob); return da - db; });</code>

Tìm kiếm theo khoảng tuổi:

```
let min = 10, max = 20;  
let findResult = personsArray.filter(p => p.age >= min && p.age <=  
max);  
printArray(findResult);
```

7. Bài tập 2:

Cho 1 danh sách các company như sau:

```
const companies = [  
    { name: "Company One", category: "Finance", start: 1981, end: 2004 },  
    { name: "Company Two", category: "Retail", start: 1992, end: 2008 },  
    { name: "Company Three", category: "Auto", start: 1999, end: 2007 },  
    { name: "Company Four", category: "Retail", start: 1989, end: 2010 },  
    { name: "Company Five", category: "Technology", start: 2009, end: 2014 },  
    { name: "Company Six", category: "Finance", start: 1987, end: 2010 },  
    { name: "Company Seven", category: "Auto", start: 1986, end: 1996 },  
    { name: "Company Eight", category: "Technology", start: 2011, end: 2016 },  
    { name: "Company Nine", category: "Retail", start: 1981, end: 1989 }  
];
```

Tạo giao diện có dạng:

Filter by Start time: From year: To year:

Trong đó button Get company names: lấy toàn bộ companies

List of company names

1. Company One
2. Company Two
3. Company Three
4. Company Four
5. Company Five
6. Company Six
7. Company Seven
8. Company Eight
9. Company Nine

Get company names

Filter by Start time: From year: To year:

Button Filter: nhập vào From year và To year. Sau đó đưa ra các companies mà có start trong khoảng thời gian đó

List of company names

1. Retail - 1992
2. Auto - 1999

Get company names

Filter by Start time: From year: To year:

8. Bài tập 3:

Tạo form gồm : id, name, age, gender. Bấm submit thì lưu thông tin về Object, hiển thị trong 1 bảng gần đó.

Ban đầu:

Id	<input type="text" value="Id..."/>	Id	Name	Age	Gender
Name	<input type="text" value="Name..."/>				
Age	<input type="text" value="Age..."/>				
Gender	Male <input checked="" type="radio"/> Female <input type="radio"/>				
<input type="button" value="Submit"/>					

Sau khi nhập dữ liệu

Id	<input type="text" value="2"/>	Id	Name	Age	Gender
Name	<input type="text" value="Vu thi Tuoi"/>	1	To an An	20	Male
Age	<input type="text" value="10"/>	2	Vu thi Tuoi	10	Female
Gender	Male <input type="radio"/> Female <input checked="" type="radio"/>				
<input type="button" value="Submit"/>					