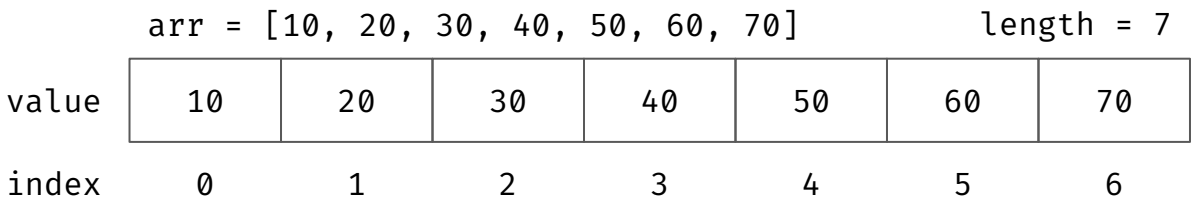
Array

Array - mảng là một cấu trúc dữ liệu, cho phép lưu trữ bộ sưu tập dữ liệu có thứ tự, thực tế, mảng cũng là **object**, nó cung cấp các phương thức để xử lý dữ liệu trong mảng (**typeof array == "object"**). Mỗi mục trong mảng được gọi là một phần tử, được "đánh số" chỉ mục (**index**) theo thứ tự bắt đầu từ 0

Cú pháp khai báo mảng

```
let arr = [];
```

```
let arr = new Array();
```



Array

Ví dụ:

```
let students = ["ba", "Béo"]; // length = 2
// truy cập phần tử mảng
students[0]; // "ba"
students[1]; // "Béo"
students[-1]; // undefined
// thay đổi giá trị
students[0] = "Ba";
students[1] = "Đẹp";
// thêm phần tử
students[2] = "Trai";
students[3] = "😅"; // students = [“Ba”, “Đẹp”, “Trai”, “😂]
```

Array

Mảng cho phép lưu dữ liệu bất kỳ, các phần tử không nhất thiết phải có cùng kiểu dữ liệu

```
let arr = ["Ba", 0, { name: "Béo Ú", age: 28 }, true, null, [1, 2]];
arr[2].name; // "Béo Ú"
arr[5][0]; // 1
arr.length; // 5
arr[5].length; // 2
typeof arr[0]; // "string"
typeof arr[2]; // "object"
typeof arr[4]; // "null"
typeof arr[5]; // "object"
```

Array

length là thuộc tính xác định "độ dài" của mảng, được cập nhật tự động khi mảng được thêm/xóa phần tử.

Tuy nhiên, **length** *không phải "độ dài" thực sự* (số lượng phần tử) của mảng, mà là **chỉ mục lớn nhất trong mảng + 1.**

Thuộc tính **length** có thể thay đổi được, khi **length** giảm, phần tử mảng sẽ bị cắt bớt.

```
let a = [1, 2, 3];
a[999] = 0;
a.length; // 1000;
a[998]; // undefined
a.length = 0; // clear array
```

Array

Lặp qua tất cả phần tử trong mảng

```
let arr = [1, 2, 3, 4, 5, 6, 7];

for (let i = 0; i < arr.length; i++) {
    arr[i] = arr[i] * 2;
}

alert(arr); // “2,4,6,8,10,12,14”
```

💡 Mảng mặc định khi chuyển về kiểu **string** sẽ có dạng

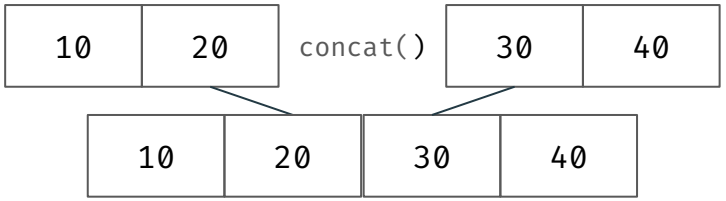
💡 Vòng lặp **for in** cũng có thể duyệt qua mảng, tuy nhiên nên sử dụng **for** thông thường

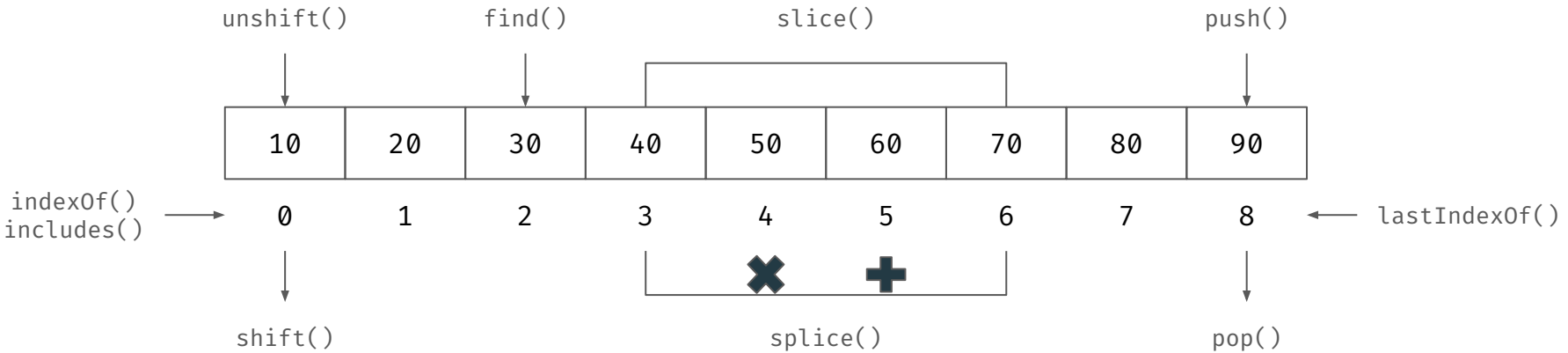
Array Methods

Một số phương thức mảng thường dùng

```
arr.reverse(); // đảo ngược mảng
arr.slice();                          arr.sort();
arr.join(); // gộp mảng thành chuỗi   arr.splice();
arr.push();                           arr.indexOf();
arr.concat();                         arr.lastIndexOf();
arr.pop();                            arr.includes();
arr.shift();                          arr.map();
arr.unshift();                        arr.find();
Array.isArray(); // kiểm tra có phải mảng hay không
Array.from(); // chuyển đổi một collection thành mảng
```

Array Methods





Exercise

1. Viết hàm **arr._concat(arr2)** gộp các phần tử của mảng **arr2** vào **arr1**
2. Viết hàm **arr._push(value)** thêm giá trị vào cuối mảng
3. Viết hàm **arr._pop()** xóa phần tử cuối mảng, đồng thời trả về giá trị của phần tử bị xóa
4. Viết hàm **arr._indexOf(value)** tìm và trả về index của phần tử, nếu không có trả về -1
5. Viết hàm **arr._reverse()** đảo ngược giá trị mảng

Array Methods

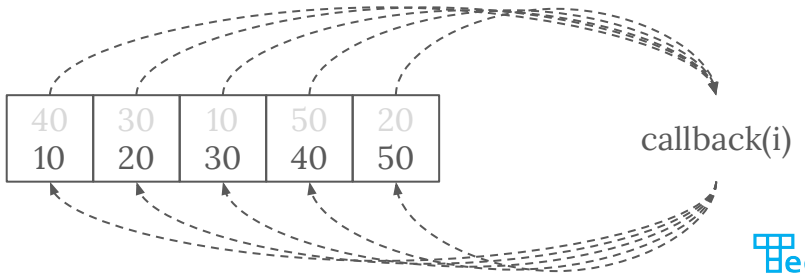
Hàm `sort()` được sử dụng để sắp xếp mảng, nó cập nhật trực tiếp giá trị trong mảng

Hàm `sort()` có thể nhận vào một tham số là hàm **callback** để so sánh 2 phần tử của mảng. Mặc định `sort()` so sánh phần tử theo kiểu dữ liệu **string**

```
[1, 2, 15].sort(); // ⇒ "1", "2", "15" ⇒ [1, 15, 2]
```

Để sắp xếp một mảng theo giá trị kiểu number

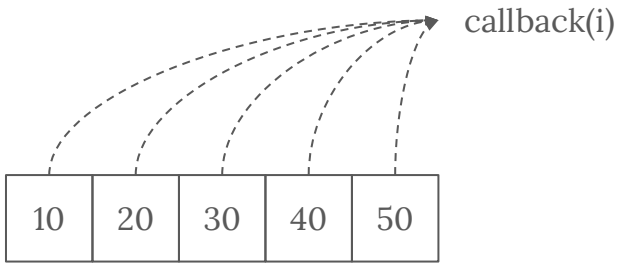
```
[15, 1, 2].sort((a, b) ⇒ a - b);
// [1, 2, 15]

[15, 1, 2].sort((a, b) ⇒ b - a);
// [15, 2, 1]
```



Array Methods

Hàm `forEach()` nhận vào 1 tham số là hàm **callback**, nó lặp qua mảng và với mỗi phần tử, nó gọi hàm **callback** với giá trị của phần tử đó

```
[1, 2, 3].forEach((i) => console.log(i * i)); // 1, 4, 9
[1, 2, 3].forEach((value, index, array) =>
    console.log(`${value} has index ${index} in array [${array}]`)
);
```

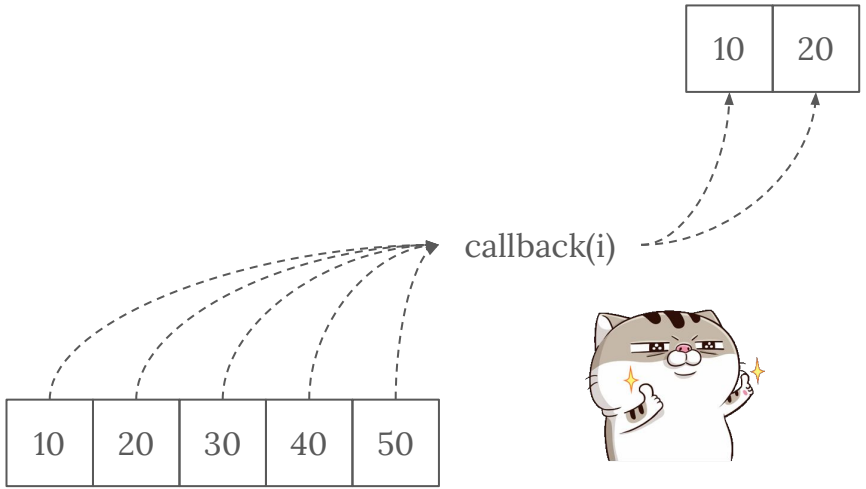


💡 Không thể ngắt `forEach()` với **`break`** hoặc **`continue`** và câu lệnh **`return`** bị bỏ qua

Array Methods

Hàm `filter()` trả về một mảng các phần tử "khớp" với điều kiện chỉ định, nó nhận vào một hàm **callback** để so sánh giá trị, hàm **callback** phải trả về giá trị **`true`** hoặc **`false`**

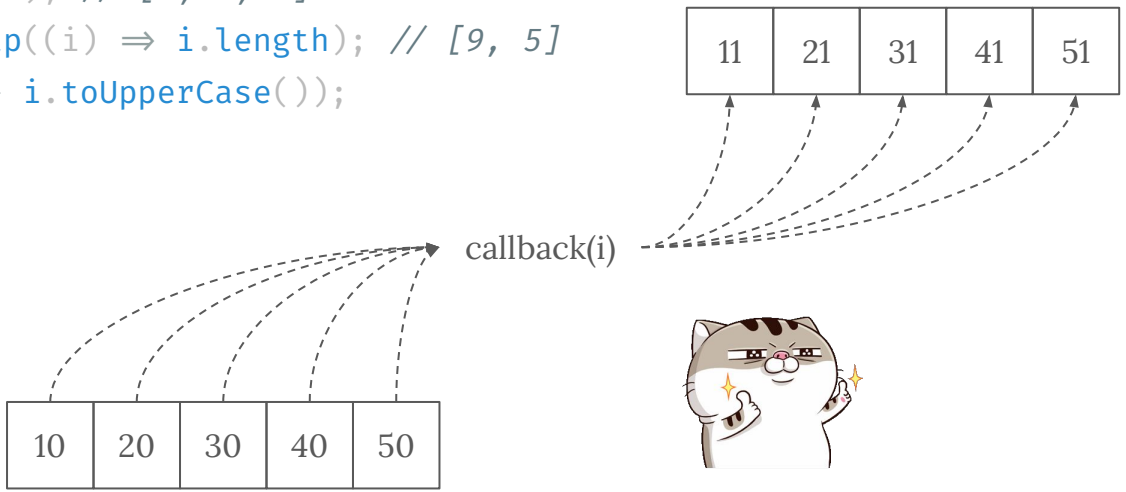
```
let arr = [
    { name: "Ba", age: 29 },
    { name: "Bon", age: 3 },
];

arr.filter((i) => i.age > 20);
// [ {name: "Ba", age: 29} ]
```



Array Methods

Hàm `map()` nhận một tham số là hàm **callback**, với mỗi phần tử, nó gọi hàm **callback** và trả về một mảng mới với giá trị của phần tử là giá trị trả về từ hàm **callback**

```
[1, 2, 3].map((i) => i * i); // [1, 4, 9]
["Ba Nguyen", "Béo Ú"].map((i) => i.length); // [9, 5]
["abc", "def"].map((i) => i.toUpperCase());
// ["ABC", "DEF"]
```



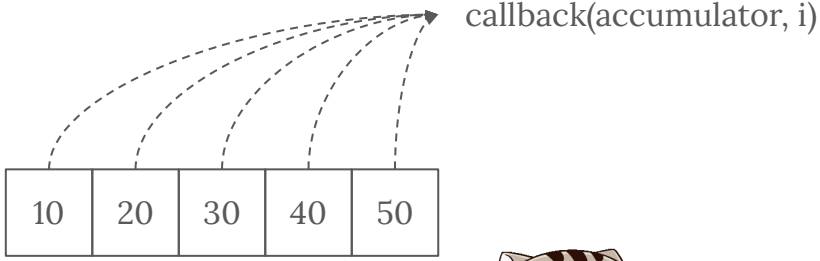
Array Methods

Hàm `reduce()` thực hiện tính toán (tổng hợp) giá trị của mảng, nó nhận vào tham số là một hàm **callback** và một giá trị khởi tạo (tùy chọn)

Cú pháp

Giá trị tích lũy sau mỗi lần lặp, ban đầu nhận giá trị **[initial]** nếu có hoặc giá trị phần tử đầu tiên trong mảng

```
arr.reduce(callback(accumulator, value, index, array) {
    // code
    // code
}, [ initial ] );
```




```
[1, 2, 3, 4, 5, 6].reduce((sum, i) => (sum += i));
// 17
```

Exercise

1. Viết hàm **arr._sort(arr, callback)** thực thi code giống như hàm **sort()**
2. Viết hàm **arr._forEach(arr, callback)** thực thi code giống như hàm **forEach()**
3. Viết hàm **arr._filter(arr, callback)** thực thi code giống như hàm **filter()**
4. Viết hàm **arr._map(arr, callback)** thực thi code giống như hàm **map()**
5. Viết hàm **arr._reduce(arr, callback)** thực thi code giống như hàm **reduce()**

Homework

1. Cho một mảng số, viết hàm tính trung bình cộng tất cả phần tử trong mảng
2. Cho một mảng số, viết hàm tìm index của một số trong mảng
3. Viết hàm sao chép một mảng số
4. Cho một mảng số, viết hàm tìm giá trị lớn nhất trong mảng
5. Viết hàm đổi chỗ vị trí 2 phần tử trong mảng
6. Cho một mảng số đã được sắp xếp tăng dần, viết hàm tìm số lớn thứ 2 trong mảng
7. Viết hàm chuyển đổi một chuỗi thành dạng capitalize. VD “hello world” => “Hello World”
8. Viết hàm tìm số lần xuất hiện lớn nhất của một phần tử trong mảng
9. Viết hàm cắt chuỗi thành một mảng có độ dài chỉ định. VD “Hello”, 2 => [“He”, “ll”, “o”]
10. Viết hàm tách chuỗi thành một mảng các chuỗi con. VD “dog” => [“d”, “do”, “dog”, “og”, “g”]
11. Cho một mảng số, viết hàm loại bỏ số trùng lặp trong mảng. VD [1,2,2,3] => [1,2,3]
12. Viết hàm trả về một mảng lưu dãy số Fibonacci từ 0 -> n. VD 8 => [0, 1, 1, 2, 3, 5, 8, 13]
13. Viết hàm trả về một mảng các số trùng nhau trong 2 mảng. VD [1,2,3], [2,3,4] => [2,3]
14. Viết hàm trả về một mảng các số không trùng nhau trong 2 mảng. VD [1,2,3], [2,3,4] => [1,4]
15. Viết hàm loại bỏ các giá trị “false” khỏi mảng. VD [null, 1, 0, NaN, “”] => [1]

Homework

1. Viết hàm sắp xếp một mảng số nguyên
2. Viết hàm sắp xếp một mảng “string”
3. Cho một mảng object user [{name: “Ba”, age: 28}, {name: “Bon”, age: 3}, ...] Viết hàm sắp xếp mảng user tăng dần theo age
4. Tương tự, viết hàm sắp xếp mảng user theo name.length
5. Viết hàm sắp xếp mảng user theo name
6. Cho một mảng số, và một số n, tìm trong mảng vị trí 2 phần tử có tổng bằng n, kết quả trả về là một mảng lưu vị trí 2 phần tử, hoặc mảng rỗng nếu không tìm thấy
7. Viết hàm lấy một phần tử ngẫu nhiên trong mảng
8. Viết hàm sắp xếp mảng với vị trí ngẫu nhiên (xáo trộn mảng)
9. Viết hàm biến một một mảng 2 chiều thành mảng 1 chiều.

   VD [[1,2,3],[3,4,5] => [1,2,3,3,4,5]

10. Viết hàm biến một mảng nhiều chiều (3 hoặc nhiều hơn) thành mảng một chiều

Homework

1. Viết hàm biến đổi các phần tử của mảng số nguyên thành bình phương của chính nó
2. Viết hàm biến đổi các phần tử của mảng chuỗi thành dạng uppercase()
3. Viết hàm lọc ra các phần tử có kiểu "number" trong một mảng hỗn hợp
4. Tạo một mảng object với các thông tin name, age, ...
5. Viết hàm lọc ra các object với age > 20
6. Viết hàm chuyển đổi name của object thành dạng capitalize
7. Viết hàm chuyển đổi name của object thành dạng viết tắt. VD "Ba Nguyen" => "Ba N."
8. Viết hàm để chuyển mảng object thành một mảng chỉ chứa name.

   VD [{name: "Ba", age: 28}, {name: "Béo Ú", age: 82}] => ["Ba", "Béo Ú"]

Datetime Methods

Đối tượng **Date** trong JavaScript cung cấp các phương thức xử lý dữ liệu về thời gian

Khởi tạo một object **Date**

```
new Date(); // current date time
new Date(millisecond); // from 1970-01-01 00:00:00:000
new Date(string); // parse string to date “YYYY-MM-DD”
new Date(year, month, date, hours, minutes, seconds, ms);
```

Nếu một giá trị vượt quá phạm vi, **Date** có thể tự động tính toán và phân phối phần vượt quá cho những thành phần khác VD: `new Date("2020-02-30"); // "2020-03-01"`

Datetime Methods

Một số phương thức thường dùng của **Date**:

```
getFullYear(); // 2020
getMonth(); // 0 → 11
getDate(); // 1 → 31
getHours();
getMinutes();
getSeconds();
getMilliseconds();
setFullYear(year [, month] [, date]);
setHours(hour [, min] [, sec] [, ms]);
setMinutes(min [, sec] [, ms]);
setMilliseconds(ms);
```

```
getDay(); // 0 (sunday) → 6 (saturday)
getTime(); // timestamp
getTimezoneOffset(); // UTC+7 ⇒ -420
toLocaleString(); // “vi-VN”
toLocaleTimeString(); // “vi-VN”
toLocaleDateString(); // “vi-VN”
Date.now(); // current timestamp
    setMonth(month [, date]);
    setDate(date);
    setSeconds(sec [, ms]);
    setTime(ms);
```

💡 So sánh mốc thời gian: `date1.getTime() - date2.getTime();`

Homework

1. Viết hàm kiểm tra xem một giá trị có phải giá trị thời gian hợp lệ hay không
2. Viết hàm in ra thứ viết tắt, tương ứng với ngày hiện tại. VD “T2”, “T3”, “CN”
3. Viết hàm trả về ngày trước ngày hiện tại n ngày
4. Viết hàm trả về số ngày trong tháng bất kỳ, năm và tháng là tham số truyền vào
5. Viết hàm tính số giây hiện tại trong ngày
6. Viết hàm tính số ngày còn lại đến tết dương lịch năm sau
7. Viết hàm kiểm tra một ngày có phải cuối tuần hay không, ngày là tham số truyền vào
8. Viết hàm trả về số quý tương ứng với giá trị ngày tháng truyền vào (quý 1 -> 4)
9. Viết hàm tính tổng số ngày đã qua trong năm
10. Viết hàm tính tuổi theo ngày tháng truyền vào
11. Viết hàm trả về chuỗi ngày tháng hiện tại có dạng “10:01:30 CN 20/01/2020”
12. Viết hàm trả về thời gian chênh lệnh giữa 2 quốc gia, giá trị trả về dạng “01h 10m 30s”
13. Viết hàm trả về ngày sau ngày hiện tại n ngày
14. Viết hàm trả về số giờ chênh lệnh giữa 2 ngày
15. Viết hàm trả về ngày tương ứng với ngày đầu tuần

JSON Methods

JSON (JavaScript Object Notation) là một định dạng chung mô tả các đối tượng, server và client thường trao đổi dữ liệu dưới dạng **JSON**.

JSON.stringify(obj) nhận vào một đối tượng và chuyển đổi nó thành chuỗi được gọi là *chuỗi mã hóa JSON*. **JSON.parse(str)** nhận vào một *chuỗi mã hóa JSON* và chuyển đổi nó thành đối tượng JavaScript thông thường

```
let user = {
    name: "Ba",
    age: 28,
};

JSON.stringify(user);
// "{"name":"Ba","age":28}"
```

```
let str = '{ "name": "Ba", "age": 28 }';

JSON.parse(str);
// { name: "Ba", age: 28}
```