Array

Os Arrays são apenas objetos que oferecem operações para acessar e manipular suas propriedades por meio de índices.

```
1.  [];
2.  new Array();
```

Criando um Array

```
1.    let cars = ["Ka", "Corsa", "Palio"];
```

Inserindo elementos no Array

```
1.    let cars = [];
2.    cars[0] = "Ka";
3.    cars[1] = "Corsa";
4.    cars[2] = "Palio";
```

Inserindo elementos no Array

```
1.    var cars = [];
2.
3.    cars[0] = "Ka";
4.    cars[1] = "Corsa";
5.    cars[2] = "Palio";
6.
7.    cars.length; // 3
```

Consultando o tamanho do Array com length

```
1.    var cars = [];
2.
3.    cars[0] = "Ka";
4.    cars[1] = "Corsa";
5.    cars[2] = "Palio";
6.
7.    cars.indexOf("Corsa"); // 1
```

Localizando o índice do elemento com indexOf

```
1.   var cars = [];
2.
3.   cars[0] = "Ka";
4.   cars[1] = "Corsa";
5.   cars[2] = "Palio";
6.
7.   cars.forEach(function (car) {
8.     console.log(car);
9.   });
```

Percorrendo os elementos de um Array com forEach

```
1.   var cars = [];
2.
3.   cars[0] = "Ka";
4.   cars[1] = "Corsa";
5.   cars[2] = "Palio";
6.
7.   for(var car of cars) {
8.     console.log(cars);
9.   }
```

Percorrendo os elementos de um Array com for/of

```
1.  var cars = [];
2.
3.  cars[0] = "Ka";
4.  cars[1] = "Corsa";
5.  cars[2] = "Palio";
6.
7.  cars.push("Gol"); // 4
8.
9.  cars.toString(); // ["Ka", "Corsa", "Palio",
    "Gol"]
```

Inserindo novos elementos no final com push

```
1.  var cars = [];
2.
3.  cars[0] = "Ka";
4.  cars[1] = "Corsa";
5.  cars[2] = "Palio";
6.
7.  cars.pop(); // "Palio"
8.
9.  cars.toString(); // ["Ka", "Corsa"]
```

Retirando elementos do final com pop

```
1.  var cars = [];
2.
3.  cars[0] = "Ka";
4.  cars[1] = "Corsa";
5.  cars[2] = "Palio";
6.
7.  cars.unshift("Gol"); // 4
8.
9.  cars.toString(); // ["Gol", "Ka", "Corsa",
    "Palio"]
```

Inserindo novos elementos no início com unshift

```
1.  var cars = [];
2.
3.  cars[0] = "Ka";
4.  cars[1] = "Corsa";
5.  cars[2] = "Palio";
6.
7.  cars.shift(); // "Ka"
8.
9.  cars.toString(); // ["Corsa", "Palio"]
```

Retirando elementos do início com shift

```
1.    var cars = [];
2.
3.    cars[0] = "Ka";
4.    cars[1] = "Corsa";
5.    cars[2] = "Palio";
6.
7.    var pos = cars.indexOf("Corsa"); // 1
8.
9.    cars.splice(pos, 1); // ["Corsa"]
10.
11.   cars.toString(); // ["Ka", "Palio"]
```

Removendo elementos em uma posição com splice

```
1.    var cars = [];
2.
3.    cars[0] = "Ka";
4.    cars[1] = "Corsa";
5.    cars[2] = "Palio";
6.
7.    var pos = cars.indexOf("Corsa"); // 1
8.
9.    cars.splice(pos, 1, "Sonic"); // ['Corsa']
10.
11.   // ['Ka', 'Sonic', 'Palio']
```

Substituindo elementos em uma posição com splice

```
1.    var cars = [];
2.
3.    cars[0] = "Ka";
4.    cars[1] = "Corsa";
5.    cars[2] = "Palio";
6.
7.    var pos = cars.indexOf("Corsa"); // 1
8.
9.    cars.splice(pos, 0, "Sonic"); // []
10.
11.   // ['Ka', 'Sonic', 'Corsa', 'Palio']
```

Adicionando elementos em uma posição com splice

```
1.  var cars = [];
2.
3.  cars[0] = {brand: "Ford", model: "Ka"};
4.  cars[1] = {brand: "Ford", model: "Edge"};
5.  cars[2] = {brand: "Fiat", model: "Palio"};
6.
7.  cars.filter(function (elemento) {
8.    return elemento.brand === "Ford";
9.  });
10.
11. // [{brand: 'Ford', model: 'Ka'}, {brand:
    'Ford', model: 'Edge'}]
```

Filtrando o Array com filter

```
1.  var cars = [];
2.
3.  cars[0] = {brand: "Ford", model: "Ka"};
4.  cars[1] = {brand: "Chevrolet", model:
    "Corsa"};
5.  cars[2] = {brand: "Fiat", model: "Palio"};
6.
7.  cars.find(function (car) {
8.    return car.model === "Ka";
9.  });
10.
11. // {brand: 'Ford', model: 'Ka'}
```

Buscando um elemento mo Array com find

```
1.  var cars = [];
2.
3.  cars[0] = {brand: "Ford", model: "Ka"};
4.  cars[1] = {brand: "Chevrolet", model:
    "Corsa"};
5.  cars[2] = {brand: "Fiat", model: "Palio"};
6.
7.  cars.every(function (car) {
8.    return car.brand === "Ford";
9.  });
10.
11. // false
```

Verificando os elementos do Array com every

```
1.  var cars = [];
2.
3.  cars[0] = {brand: "Ford", model: "Ka"};
4.  cars[1] = {brand: "Chevrolet", model:
    "Corsa"};
5.  cars[2] = {brand: "Fiat", model: "Palio"};
6.
7.  cars.some(function (elemento) {
8.    return elemento.brand === "Ford";
9.  });
10.
11. // true
```

Verificando os elementos do Array com some

```
1.    var cars = [];
2.
3.    cars[0] = {brand: "Ford", model: "Ka"};
4.    cars[1] = {brand: "Chevrolet", model: "Corsa"};
5.    cars[2] = {brand: "Fiat", model: "Palio"};
6.
7.    cars.map(function (car) {
8.      return car.brand;
9.    });
10.
11.   // ['Ford', 'Chevrolet', 'Fiat']
```

Mapeando os elementos do Array com map

```
1.    var cars = [];
2.
3.    cars[0] = {model: "Ka", price: 28800};
4.    cars[1] = {model: "Corsa", price: 34750};
5.    cars[2] = {model: "Palio", price: 32000};
6.
7.    cars.reduce(function (prev, cur) {
8.      return prev + cur.price;
9.    }, 0);
10.
11.   // 95550
```

Processando os elementos do Array com reduce

```
1.  var cars = ["Ka", "Corsa", "Palio"];
2.  var bikes = ["Honda", "Yamaha"];
3.
4.  var vehicles = cars.concat(bikes);
5.
6.  // ['Ka', 'Corsa', 'Palio', 'Honda', 'Yamaha']
```

Concatenando dois Arrays com concat

```
1.  var cars = ["Ka", "Corsa", "Palio"];
2.  var bikes = ["Honda", "Yamaha"];
3.
4.  var vehicles = [...cars, ...bikes];
5.
6.  // ['Ka', 'Corsa', 'Palio', 'Honda', 'Yamaha']
```

Concatenando dois Arrays com o spread operator

```
1.  var cars = [];
2.
3.  cars[0] = "Ka";
4.  cars[1] = "Corsa";
5.  cars[2] = "Palio";
6.  cars[3] = "Gol";
7.
8.  cars.slice(0,2); // ['Ka', 'Corsa']
9.  cars.slice(1,3); // ['Corsa', 'Palio']
10. cars.slice(2); // ['Palio', 'Gol']
```

Extraindo partes de um Array com slice

```
1.  var cars = [];
2.
3.  cars[0] = "Ka";
4.  cars[1] = "Corsa";
5.  cars[2] = "Palio";
6.  cars[3] = "Gol";
7.
8.  cars.reverse();
9.
10. // ['Gol', 'Palio', 'Corsa', 'Ka']
```

Invertendo a ordem de um Array com reverse

```
1.  var cars = [];
2.
3.  cars[0] = {model: "Ka", price: 28800};
4.  cars[1] = {model: "Corsa", price: 34750};
5.  cars[2] = {model: "Palio", price: 32000};
6.
7.  cars.sort(function (a, b) {
8.    return a.price - b.price;
9.  });
```

Ordenando os elementos de um Array com sort

```
1.    var cars = [];
2.
3.    cars[0] = {model: "Ka", price: 28800};
4.    cars[1] = {model: "Corsa", price: 34750};
5.    cars[2] = {model: "Palio", price: 32000};
6.    cars[3] = {model: "Fusion", price: 70000};

8.    cars.sort(function (a, b) {
9.      if (a.model < b.model) return -1;
10.     if (a.model > b.model) return 1;
11.     return 0;
12.   });
```

Ordenando os elementos de um Array com sort

```
1.  var cars = [];
2.
3.  cars[0] = {model: "Ka", price: 28800};
4.  cars[1] = {model: "Corsa", price: 34750};
5.  cars[2] = {model: "Palio", price: 32000};
6.
7.  cars.sort(function (a, b) {
8.    return a.model.localeCompare(b.model);
9.  });
```

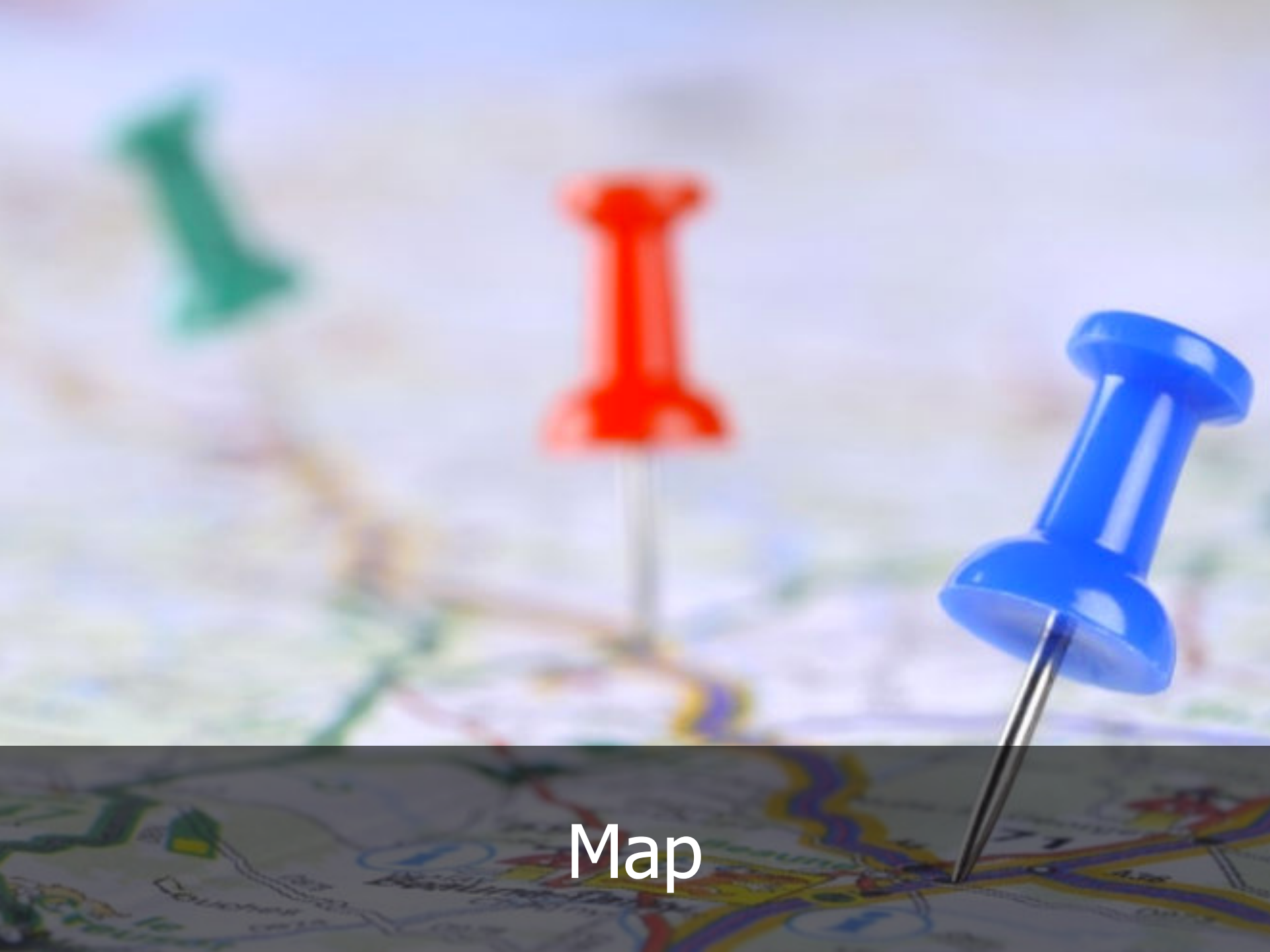
Ordenando os elementos de um Array com sort com localeCompare

```
1.  var cars = [];
2.
3.  cars[0] = "Ka";
4.  cars[1] = "Corsa";
5.  cars[2] = "Palio";
6.  cars[3] = "Gol";
7.
8.  cars.join(";"); // "Ka;Corsa;Palio;Gol"
1.  cars.join("-"); // "Ka-Corsa-Palio-Gol"
2.  cars.join(","); // "Ka,Corsa,Palio,Gol"
3.  cars.join(" "); // "Ka Corsa Palio Gol"
```

Juntando os elementos um Array com join

```
1.    var numeros = [1,2,3,4,5,6,7,8,9];
2.
3.    numeros.fill(7); // [7, 7, 7, 7, 7, 7, 7, 7, 7]
```

Preenchendo os elementos um Array com fill


Map

Um objeto do tipo Map, é um conjunto que relaciona chaves e valores.

```
1.    let capitals = new Map();
```

Criando um objeto do tipo Map

```
1.  let capitals = new Map();
2.  capitals.set("Florida", "Tallahassee");
3.  capitals.set("Illinois", "Springfield");
4.  capitals.set("New York", "Albany");
5.  capitals.set("New Jersey", "Trenton");
6.  capitals.set("Louisiana", "Batton Rouge");
```

Inserindo um par de chave e valor no mapa

```
1.  let capitals = new Map();
2.  capitals.set("Florida", "Tallahassee");
3.  capitals.set("Illinois", "Springfield");
4.  capitals.set("New York", "Albany");
5.  capitals.set("New Jersey", "Trenton");
6.  capitals.set("Louisiana", "Batton Rouge");
7.
8.  capitals.get("New York");
```

Obtendo o valor de uma chave do mapa

```
1.  let capitals = new Map();
2.  capitals.set("Florida", "Tallahassee");
3.  capitals.set("Illinois", "Springfield");
4.  capitals.set("New York", "Albany");
5.  capitals.set("New Jersey", "Trenton");
6.  capitals.set("Louisiana", "Batton Rouge");
7.
8.  capitals.forEach(
9.    (capital, state) => console.log(`${capital} is
    the capital of ${state}`)
10. );
```

Percorrendo o mapa

```
1.    let capitals = new Map();
2.    capitals.set("Florida", "Tallahassee");
3.    capitals.set("Illinois", "Springfield");
4.    capitals.set("New York", "Albany");
5.    capitals.set("New Jersey", "Trenton");
6.    capitals.set("Louisiana", "Batton Rouge");
7.
8.    capitals.has("Florida");
9.    capitals.has("Tennessee");
```

Verificando se existe uma chave no mapa

```
1.    let capitals = new Map();
2.    capitals.set("Florida", "Tallahassee");
3.    capitals.set("Illinois", "Springfield");
4.    capitals.set("New York", "Albany");
5.    capitals.set("New Jersey", "Trenton");
6.    capitals.set("Louisiana", "Batton Rouge");
7.
8.    capitals.delete("Louisiana");
```

Apagando uma chave do mapa

```
1.    let capitals = new Map();
2.    capitals.set("Florida", "Tallahassee");
3.    capitals.set("Illinois", "Springfield");
4.    capitals.set("New York", "Albany");
5.    capitals.set("New Jersey", "Trenton");
6.    capitals.set("Louisiana", "Batton Rouge");
7.
8.    capitals.size;
```

Obtendo o tamanho do mapa

```
1.    let capitals = new Map();
2.    capitals.set("Florida", "Tallahassee");
3.    capitals.set("Illinois", "Springfield");
4.    capitals.set("New York", "Albany");
5.    capitals.set("New Jersey", "Trenton");
6.    capitals.set("Louisiana", "Batton Rouge");
7.
8.    capitals.clear();
```

Apagando todo o mapa

```
1.    let capitals = new Map();
2.    capitals.set("Florida", "Tallahassee");
3.    capitals.set("Illinois", "Springfield");
4.    capitals.set("New York", "Albany");
5.    capitals.set("New Jersey", "Trenton");
6.    capitals.set("Louisiana", "Batton Rouge");
7.
8.    let capitalsArray = Array.from(capitals);
```

Convertendo todo o mapa para um array


Set

Um objeto do tipo Set é igual ao Map, no entanto, não permite a inserção de elementos duplicados.

Weak Map e Weak Set

Um objeto do tipo WeakMap e WeakSet, são iguais ao Map e Set, no entanto a chave deve ser uma referência para um objeto, mantida do lado de fora.