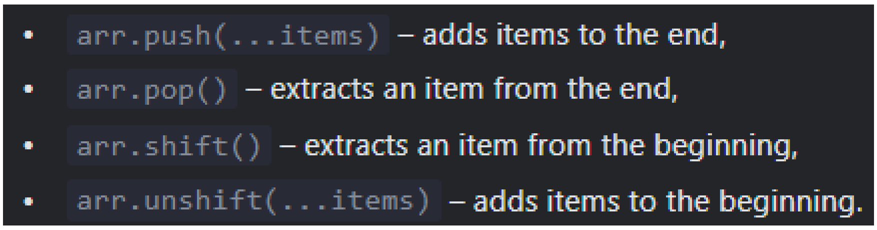
ARRAY METHODS

Another full reference: W3Schools
Note: sort(), reverese(), and splice() modify the array itself.

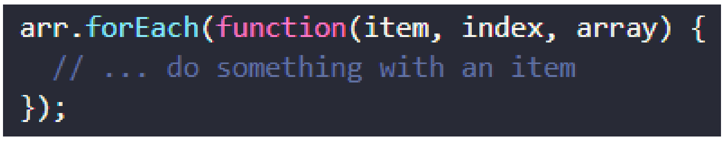
A. Add/Remove Items



- `arr.push(...items)` – adds items to the end,
- `arr.pop()` – extracts an item from the end,
- `arr.shift()` – extracts an item from the beginning,
- `arr.unshift(...items)` – adds items to the beginning.

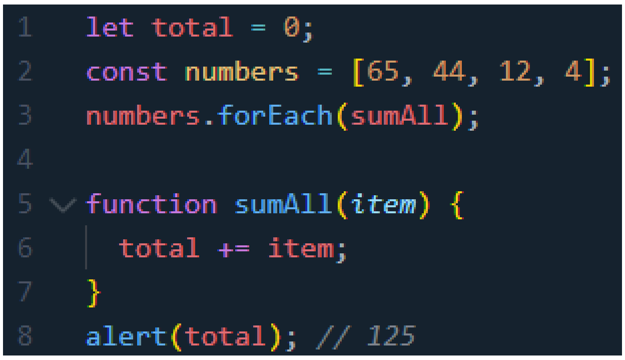
- Arr.splice([start], [number of elements to delete])
- Arr.slice([start], [end])
- Arr.concat(arg1, arg2…)

B. Iterate



```
arr.forEach(function(item, index, array) {
  // ... do something with an item
});
```

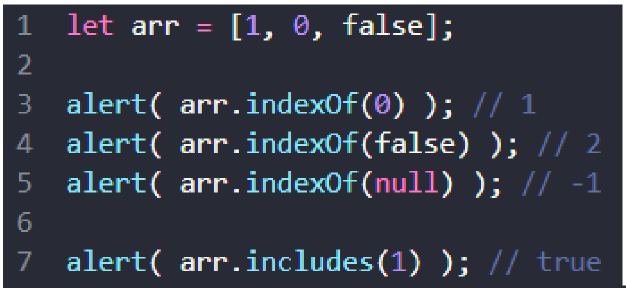
.foreach() sama seperti .map(), bedanya
.foreach() tidak mereturn apapun(hanya perform function pada tiap elemen. biasanya hasil perform functionnya ada dalam function tsb.
.map() mereturn array baru(hasil perform function pada tiap elemen)



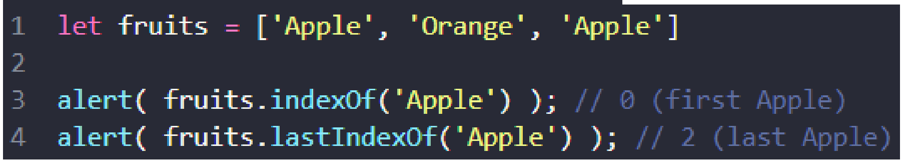
```
let total = 0;
const numbers = [65, 44, 12, 4];
numbers.forEach(sumAll);

function sumAll(item) {
  total += item;
}
alert(total); // 125
```

C. Searching

- indexOf() / lastIndexOf() and includes()
  indexOf() : return index of element(first match)
  lastIndexOf() : return index of element(last match)
  includes() : return true if array includes(menyertakan) the element



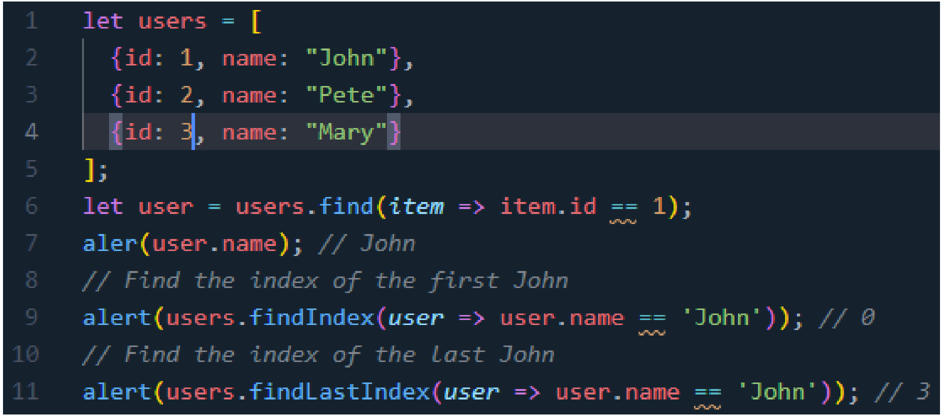
```
let arr = [1, 0, false];

alert( arr.indexOf(0) ); // 1
alert( arr.indexOf(false) ); // 2
alert( arr.indexOf(null) ); // -1

alert( arr.includes(1) ); // true
```



```
let fruits = ['Apple', 'Orange', 'Apple']

alert( fruits.indexOf('Apple') ); // 0 (first Apple)
alert( fruits.lastIndexOf('Apple') ); // 2 (last Apple)
```

- find() and findIndex() / findLastIndex()
  find() : return first element that match the function.
  findIndex() : return index of first element that match the function.

findLastIndex() : return index of lasat element that match the function.



```
let users = [
  {id: 1, name: "John"},
  {id: 2, name: "Pete"},
  {id: 3, name: "Mary"}
];
let user = users.find(item => item.id == 1);
aler(user.name); // John
// Find the index of the first John
alert(users.findIndex(user => user.name == 'John')); // 0
// Find the index of the last John
alert(users.findLastIndex(user => user.name == 'John')); // 3
```

- filter() : return (all) element that match the function

D. Check isSome(true) & isEvery(true)

- some() : to check atleast 1 element match the function
- every() : to check if function return true for all elements. in other words, executes a function for each array element. returns true if the function returns true for all elements.

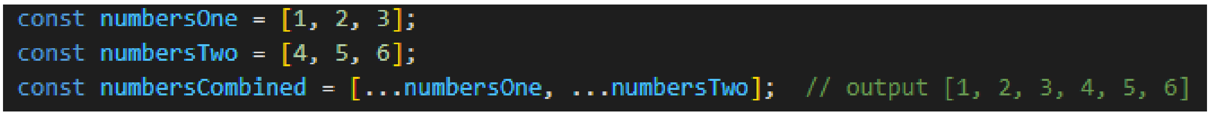
E. Transform an array

- map(function(currentValue, index, arr), thisValue) : (creates new array)performing function to every array element.
  Example at **Rangkuman Object Basics sub-bab E**
- sort() : mengurutkan elemen-elemen dalam sebuah array(sesuai urutan karakter Unicode).
  Example at **Rangkuman Object Basics sub-bab E**
- reverse() : reverses the order of elements (in array).
- split() & join()
  - split(*separator*, *limit*) : splits a string into an array of substrings. If *separator* omitted, an array with the original string is returned. *limit* means An integer that limits the number of splits.
  - join(*separator*) : joins all array elements into a string(you can specify the separator).
- reduce() & reduceRight()
  - reduce() : (creates new array) returns a single value(the function's accumulated result). This method is very useful for various operations such as adding values, multiplying values, combining objects, and so on.
    Example at **Rangkuman Object Basics sub-bab E**
  - reduceRight() : does the same but goes from right to left.

F. Spread Operator ('…')

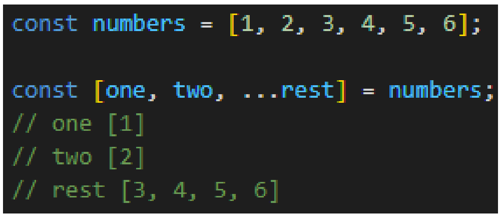
(…) expands an iterable (like an array) into more elements. This handy **to copy** all or parts of an existing **array** into another array.

Example: copy entire array



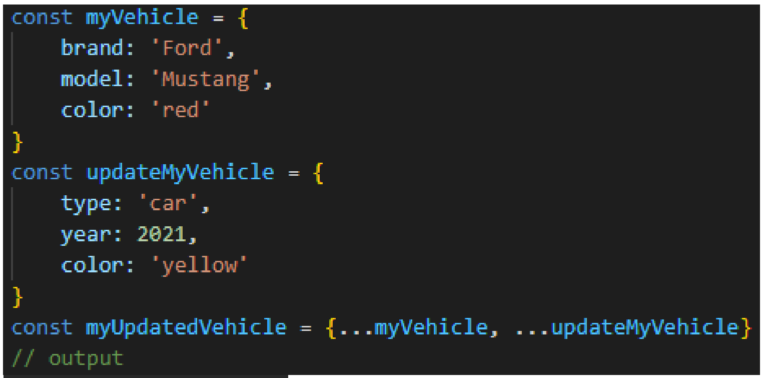
```
const numbersOne = [1, 2, 3];
const numbersTwo = [4, 5, 6];
const numbersCombined = [...numbersOne, ...numbersTwo];  // output [1, 2, 3, 4, 5, 6]
```

Example: copy parts of array

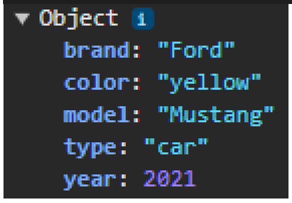


```
const numbers = [1, 2, 3, 4, 5, 6];

const [one, two, ...rest] = numbers;
// one [1]
// two [2]
// rest [3, 4, 5, 6]
```

Example: in object

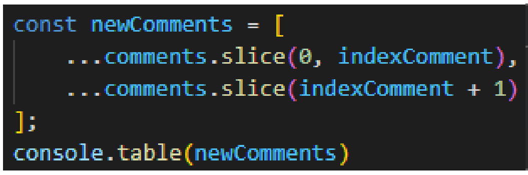


```
const myVehicle = {
    brand: 'Ford',
    model: 'Mustang',
    color: 'red'
}
const updateMyVehicle = {
    type: 'car',
    year: 2021,
    color: 'yellow'
}
const myUpdatedVehicle = {...myVehicle, ...updateMyVehicle}
// output
```

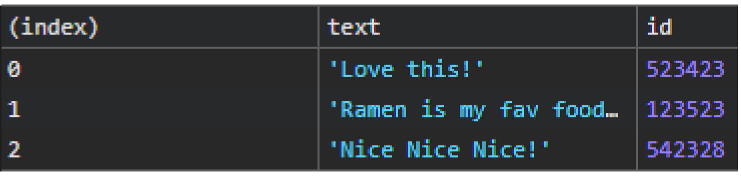


```
▼ Object i
    brand: "Ford"
    color: "yellow"
    model: "Mustang"
    type: "car"
    year: 2021
```

Spread operator (...) menyebarkan elemen-elemen dari array menjadi elemen-elemen individu, sedangkan tanpa operator spread, array tersebut dimasukkan sebagai elemen tunggal dalam array baru.

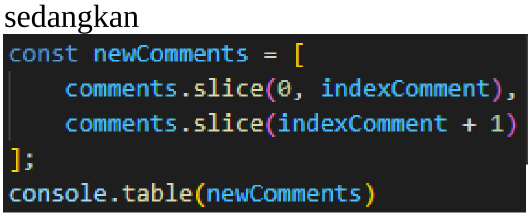


```
const newComments = [
    ...comments.slice(0, indexComment),
    ...comments.slice(indexComment + 1)
];
console.table(newComments)
```

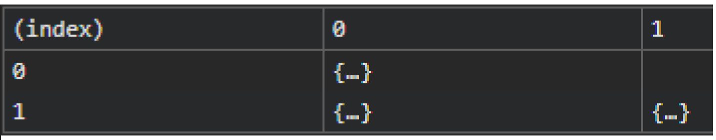


(index)	text	id
0	'Love this!'	523423
1	'Ramen is my fav food…	123523
2	'Nice Nice Nice!'	542328

sedangkan



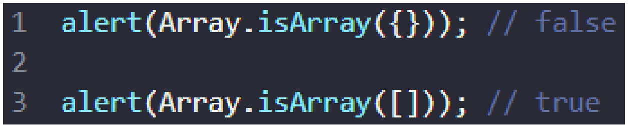
```
const newComments = [
    comments.slice(0, indexComment),
    comments.slice(indexComment + 1)
];
console.table(newComments)
```



(index)	0	1
0	{…}	
1	{…}	{…}

G. Additional

- ➢ isArray() : to check if it is array



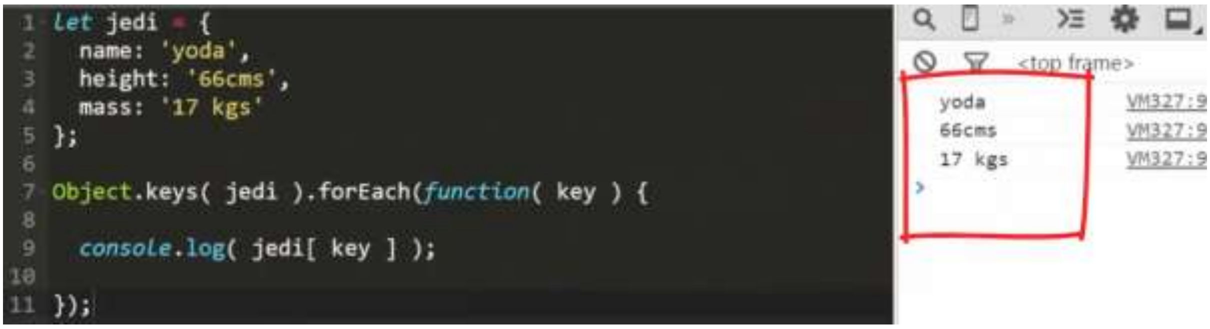
```
alert(Array.isArray({})); // false

alert(Array.isArray([])); // true
```

- ➢ thisArgs / thisValue

- Array.from (arrayLike [, mapfn [, thisArg]])
- Array.prototype.every (callbackfn [, thisArg])
- Array.prototype.filter (callbackfn [, thisArg])
- Array.prototype.find (predicate [, thisArg])
- Array.prototype.findIndex (predicate [, thisArg])
- Array.prototype.forEach (callbackfn [, thisArg])
- Array.prototype.map (callbackfn [, thisArg])
- Array.prototype.some (callbackfn [, thisArg])

Banyak array method yang menggunakan thisArg sebagai parameter keduanya. This is what the second argument does, thisArg, assigns value for ‘this’ object.

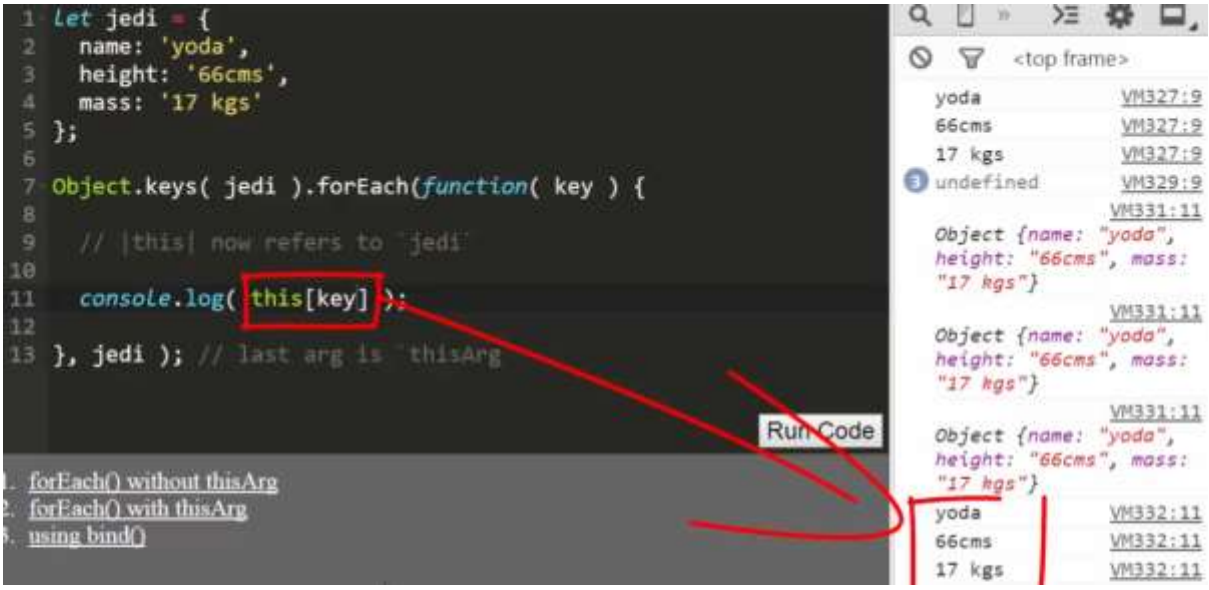
jadi parameter kedua merupakan nilai dari ‘this’. Berikut contohnya:



```
let jedi = {
  name: 'yoda',
  height: '66cms',
  mass: '17 kgs'
};

Object.keys( jedi ).forEach(function( key ) {

  console.log( jedi[ key ] );

});
```

```
yoda        VM327:9
66cms       VM327:9
17 kgs      VM327:9
>
```

kode di bawah ini sama artinya dengan kode yang di atas^



```
let jedi = {
  name: 'yoda',
  height: '66cms',
  mass: '17 kgs'
};

Object.keys( jedi ).forEach(function( key ) {

  // |this| now refers to `jedi`

  console.log( this[key] );

}, jedi ); // last arg is `thisArg`
```

1. forEach() without thisArg
2. forEach() with thisArg
3. using bind()

```
yoda                                          VM327:9
66cms                                         VM327:9
17 kgs                                        VM327:9
3 undefined                                   VM329:9
                                             VM331:11
Object {name: "yoda",
height: "66cms", mass:
"17 kgs"}
                                             VM331:11
Object {name: "yoda",
height: "66cms", mass:
"17 kgs"}
                                             VM331:11
Object {name: "yoda",
height: "66cms", mass:
"17 kgs"}
yoda                                         VM332:11
66cms                                        VM332:11
17 kgs                                       VM332:11
```