Array Method - JS

Array concat() Method

Description - Javascript array **concat()** method returns a new array comprised of this array joined with two or more arrays.

```
var alpha = ['a', 'b', 'c'];
var numeric = [1, 2, 3];

alpha.concat(numeric);
// result in ['a', 'b', 'c', 1, 2, 3]
```

Array find() Method

Description - The **find()** method returns the **value** of the **first element** in the array that satisfies the provided testing function. Otherwise Undefined is returned

```
var array1 = [5, 12, 8, 130, 44];
var found = array1.find(function(element) {
  return element > 10;
});
console.log(found); // expected output: 12
```

Array findIndex() Method

Description - The **findIndex()** method returns the **index** of the **first element** in the array that satisfies the provided testing function. Otherwise -1 is returned.

```
var array1 = [5, 12, 8, 130, 44];
function findFirstLargeNumber(element) {
  return element > 13;
}
console.log(array1.findIndex(findFirstLargeNumber)); // expected output: 3
```

Array Method - JS

Array keys() Method

Description - The **`keys()`** method returns a new **`Array Iterator`** object that contains the keys for each index in the array.

```
var array1 = ['a', 'b', 'c'];
var iterator = array1.keys();
for (let key of iterator) {
  console.log(key); // expected output: 0 1 2
}
```

Array every() Method

Description - Javascript array **every()** method tests whether all the elements in an array passes the test implemented by the provided function.

```
function isBigEnough(element, index, array) {
  return element >= 10;
}
[12, 5, 8, 130, 44].every(isBigEnough);   // false
[12, 54, 18, 130, 44].every(isBigEnough); // true
```

Array entries () Method

Description - The **`entries()`** method returns a new **`Array Iterator`** object that contains the key/value pairs for each index in the array.

```
var a = ['a', 'b'];
var iterator = a.entries();

for (let e of iterator) {
  console.log(e);
}
// [0, 'a']
// [1, 'b']
```

Array Method - JS

Array filter() Method

Description - Javascript array **filter()** method creates a new array with all elements that pass the test implemented by the provided function.

```
function isBigEnough(value) {
  return value >= 10;
}

var filtered = [12, 5, 8, 130, 44].filter(isBigEnough);
// filtered is [12, 130, 44]
```

Array forEach() Method

Description - Javascript array **forEach()** method calls a function for each element in the array.

```
const items = ['item1', 'item2', 'item3'];
const copy = [];
items.forEach(function(item){
  copy.push(item)
});
```

Array indexOf() Method

Description - Javascript array **indexOf()** method returns the first index at which a given element can be found in the array, or -1 if it is not present.

```
var array = [2, 9, 9];
array.indexOf(2);     // 0
array.indexOf(7);     // -1
array.indexOf(9, 2);  // 2 , start from index 2
array.indexOf(2, -1); // -1
array.indexOf(2, -3); // 0
```

Array Method - JS

Array map() Method

Description - Javascript array **map()** method creates a new array with the results of calling a provided function on every element in this array.

```
var numbers = [1, 4, 9];
var roots = numbers.map(Math.sqrt);
// roots is now [1, 2, 3]
// numbers is still [1, 4, 9]
```

Array pop() Method

Description - Javascript array **pop()** method removes the last element from an array and returns that element.

```
var myFish = ['angel', 'clown', 'mandarin', 'sturgeon'];

var popped = myFish.pop();

console.log(myFish); // ['angel', 'clown', 'mandarin' ]

console.log(popped); // 'sturgeon'
```

Array reduce() Method

Description - Javascript array **reduce()** method applies a function simultaneously against two values of the array (from left-to-right) as to reduce it to a single value.

```
var sum = [0, 1, 2, 3].reduce(function (accumulator, currentValue) {
  return accumulator + currentValue;
}, 0); // 0 is Initial value
// sum is 6
```

Array Method - JS

Array push() Method

Description - Javascript array **push()** method appends the given element(s) in the last of the array and returns the length of the new array.

```
var sports = ['soccer', 'baseball'];
var total = sports.push('football', 'swimming');

console.log(sports); // ['soccer', 'baseball', 'football', 'swimming']
console.log(total);  // 4
```

Array reduceRight() Method

Description – Javascript array **reduceRight()** method applies a function simultaneously against two values of the array (from right-to-left) as to reduce it to a single value

```
var flattened = [[0, 1], [2, 3], [4, 5]].reduceRight(function(a, b) {
    return a.concat(b);
}, []);
// flattened is [4, 5, 2, 3, 0, 1]
```

Array reverse() Method

Description - Javascript array **reverse()** method reverses the element of an array. The first array element becomes the last and the last becomes the first.

```
const a = [1, 2, 3];
a.reverse();
console.log(a); // [3, 2, 1]
```

Array Method - JS

Array shift() Method

Description - Javascript array **shift()**method removes the first element from an array and returns that element.

```
var array1 = [1, 2, 3];
var firstElement = array1.shift();
console.log(array1); // output: Array [2, 3]
console.log(firstElement); // output: 1
```

Array slice() Method

Description - Javascript array **slice()** method extracts a section of an array and returns a new array.

```
var fruits = ['Banana', 'Orange', 'Lemon', 'Apple', 'Mango'];
var citrus = fruits.slice(1, 3);
// citrus contains ['Orange','Lemon']
```

Array some() Method

Description - Javascript array **some()** method tests whether some element in the array passes the test implemented by the provided function.

```
function isBiggerThan10(element, index, array) {
  return element > 10;
}

[2, 5, 8, 1, 4].some(isBiggerThan10);  // false
[12, 5, 8, 1, 4].some(isBiggerThan10); // true
```

Array unshift() Method

Description - Javascript array **unshift()** method adds one or more elements to the beginning of an array and returns the new length of the array.

```
var arr = [1, 2];
arr.unshift(-2, -1); // = 5 // arr is [-2, -1, 0, 1, 2]
```

Array Method - JS

Array sort() Method

Description - Javascript array **sort()** method sorts the elements of an array.

```
var numbers = [4, 2, 5, 1, 3];
numbers.sort(function(a, b) {
  return a - b;
});
console.log(numbers);

// [1, 2, 3, 4, 5]
```

- If `compareFunction(a, b)` is less than 0, sort a to an index lower than b, i.e. acomes first.
- If `compareFunction(a, b)` returns 0, leave a and b unchanged with respect to each other, but sorted with respect to all different elements. Note: the ECMAscript

  standard does not guarantee this behaviour, and thus not all browsers (e.g. Mozilla versions dating back to at least 2003) respect this.
- If `compareFunction(a, b)` is greater than 0, sort b to an index lower than a, i.e. bcomes first.
- `compareFunction(a, b)` must always return the same value when given a specific pair of elements a and b as its two arguments. If inconsistent results are returned then the sort order is undefined.

```
function compare(a, b) {
  if (a is less than b by some ordering criterion) {
    return -1;
  }
  if (a is greater than b by the ordering criterion) {
    return 1;
  }
  // a must be equal to b
  return 0;
}
```

Array Method - JS

Array splice() Method

Description - Javascript array **splice()** method changes the content of an array, adding new elements while removing old elements.

```
var myFish = ['angel', 'clown', 'mandarin', 'sturgeon'];
var removed = myFish.splice(2, 0, 'drum');
// myFish is ["angel", "clown", "drum", "mandarin", "sturgeon"]
// removed is [], no elements removed

var removed2 = myFish.splice(2, 1, 'trumpet');

// myFish is ["angel", "clown", "trumpet", "sturgeon"]
// removed2 is ["drum"]
```

Supported Resources –

- https://www.tutorialspoint.com/javascript/javascript_arrays_object.htm
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array