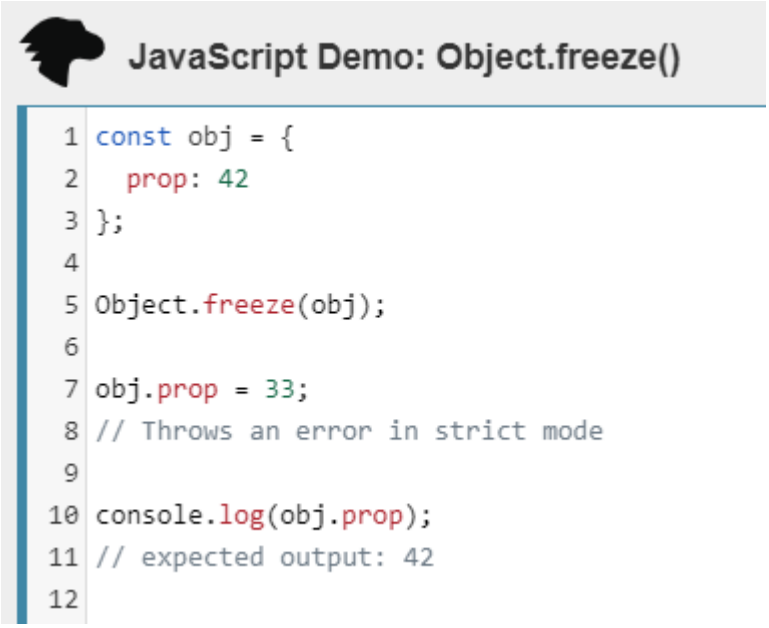
The `Object.freeze()` method **freezes** an object. A frozen object can no longer be changed; freezing an object prevents new properties from being added to it, existing properties from being removed, prevents changing the enumerability, configurability, or writability of existing properties, and prevents the values of existing properties from being changed. In addition, freezing an object also prevents its prototype from being changed. `freeze()` returns the same object that was passed in.



JavaScript Array reduce() Method

https://www.w3schools.com/jsref/jsref_reduce.asp

Definition and Usage

The `reduce()` method reduces the array to a single value.

The `reduce()` method executes a provided function for each value of the array (from left-to-right).

The return value of the function is stored in an accumulator (result/total).

Note: `reduce()` does not execute the function for array elements without values.

Note: This method does not change the original array.

Example

Subtract the numbers in the array, starting from the beginning:

```
var numbers = [175, 50, 25];

document.getElementById("demo").innerHTML = numbers.reduce(myFunc);

function myFunc(total, num) {
  return total - num;
}
```

```
var testArray = [3, 2, 5, 15];
testArray.reduce(function(total, current,) {
  return total + current
});
25
```

JavaScript String charAt() Method

Example

Return the first character of a string:

```
var str = "HELLO WORLD";
var res = str.charAt(0);
```

Definition and Usage

The charAt() method returns the character at the specified index in a string.

The index of the first character is 0, the second character is 1, and so on.

Tip: The index of the last character in a string is *string*.length-1, the second last character is *string*.length-2, and so on (See "More Examples").

JavaScript Array sort() Method

Example

Sort an array:

```
> var fruits = ["Banana", "Orange", "Apple", "Mango"];
  fruits.sort();
< ▶ (4) ["Apple", "Banana", "Mango", "Orange"]
```

The sort() method sorts the items of an array.

The sort order can be either alphabetic or numeric, and either ascending (up) or descending (down).

```
let testArr = [2, 1, 33, 4, 3, 5, 7, 6, 76, 25, 14, 4, 17, 12];
function compareDown(a, b) {
  return b - a;
}
testArr.sort(compareDown);
▶ (14) [76, 33, 25, 17, 14, 12, 7, 6, 5, 4, 4, 3, 2, 1]
```

By default, the sort() method sorts the values as strings in alphabetical and ascending order.

This works well for strings ("Apple" comes before "Banana"). However, if numbers are sorted as strings, "25" is bigger than "100", because "2" is bigger than "1".

Because of this, the sort() method will produce an incorrect result when sorting numbers.

You can fix this by providing a "compare function" (See "Parameter Values" below).

Note: This method changes the original array.

Syntax

```
array.sort(compareFunction)
```

Parameter Values

Parameter	Description
compareFunction	Optional. A function that defines an alternative sort order. The function should return a negative, zero, or positive value, depending on the arguments, like: • function(a, b){return a-b} When the sort() method compares two values, it sends the values to the compare function, and sorts the values according to the returned (negative, zero, positive) value. **Example:** When comparing 40 and 100, the sort() method calls the compare function(40,100). The function calculates 40-100, and returns -60 (a negative value). The sort function will sort 40 as a value lower than 100.

```
let testArr = [2, 1, 33, 4, 3, 5, 7, 6, 76, 25, 14, 4, 17, 12];
let nextArr = [...testArr];
function compare(a, b) { // function returns true or false which works as a "filter"
  return a - b;
}
nextArr.sort(compare);
console.log(nextArr);
```

```
▶ (14) [1, 2, 3, 4, 4, 5, 6, 7, 12, 14, 17, 25, 33, 76]
```

The `Object.values()` method returns an array of a given object's own enumerable property values, in the same order as that provided by a `for...in` loop. (The only difference is that a `for...in` loop enumerates properties in the prototype chain as well.)

Syntax:

```
Object.values(obj)
```

`Object.values()` returns an array whose elements are the enumerable property values found on the object. The ordering of the properties is the same as that given by looping over the property values of the object manually.

JavaScript **Array map()** Method

https://www.w3schools.com/jsref/jsref_map.asp#:~:text=The%20map()%20method%20creates,for%20array%20elements%20without%20values.

The `map()` method creates a new array with the results of calling a function for every array element.

The `map()` method calls the provided function once for each element in an array, in order.

Syntax

```
array.map(function(currentValue, index, arr), thisValue)
```

Parameter Values

Parameter	Description
function(currentValue, index, arr)	Required. A function to be run for each element in the array. Function arguments: **Argument** / **Description** *currentValue* — Required. The value of the current element *index* — Optional. The array index of the current element *arr* — Optional. The array object the current element belongs to
thisValue	Optional. A value to be passed to the function to be used as its "this" value. If this parameter is empty, the value "undefined" will be passed as its "this" value

```
> let testArray = [1, 2, 3];
  function addNumberWord (k) {
    return "number : " + k;
  }
  testArray.map(addNumberWord);
< ▶ (3) ["number : 1", "number : 2", "number : 3"]
```

```
let testArray = [1, 2, 3, 4, 5, 6, 7, 8];
function isEven (k) {
  return k % 2 == 0;
}
testArray.map(isEven);
▶ (8) [false, true, false, true, false, true, false, true]
```

JavaScript Array join() Method

https://www.w3schools.com/jsref/jsref_join.asp

The `join()` method returns the array as a string.

The elements will be separated by a specified separator. The default separator is comma (,).

Note: this method will not change the original array.

Syntax

```
array.join(separator)
```

```
> let testArray = [1, 2, 3, 4, 5, 6, 7, 8];
  testArray.join(", potem ");
< "1, potem 2, potem 3, potem 4, potem 5, potem 6, potem 7, potem 8"
```

The `padStart()` method pads the current string with another string (multiple times, if needed) until the resulting string reaches the given length. The padding is applied from the start of the current string.

```
> "5".padStart(10, "pad");
< "padpadpad5"
```

Syntax:

```
str.padStart(targetLength [, padString])
```

JavaScript **String split()** Method

https://www.w3schools.com/jsref/jsref_split.asp

Definition and Usage

The split() method is used to split a string into an array of substrings, and returns the new array.

Tip: If an empty string ("") is used as the separator, the string is split between each character.

Note: The split() method does not change the original string.

```
string.split(separator, limit)
```

Parameter Values

Parameter	Description
separator	Optional. Specifies the character, or the regular expression, to use for splitting the string. If omitted, the entire string will be returned (an array with only one item)
limit	Optional. An integer that specifies the number of splits, items after the split limit will not be included in the array

```
let testStr = "Wonder if it works.";
let yodaStr = testStr.split(" ");
console.log(yodaStr);
```

JavaScript **Array reverse()** Method

https://www.w3schools.com/jsref/jsref_reverse.asp

Definition and Usage

The `reverse()` method reverses the order of the elements in an array.

Note: this method will change the original array.

Syntax

```
array.reverse()
```

Parameters

None

```
> var fruits = ["Banana", "Orange", "Apple", "Mango"];
  fruits.reverse();
< ▶ (4) ["Mango", "Apple", "Orange", "Banana"]
```

JavaScript **String replace()** Method

https://www.w3schools.com/jsref/jsref_replace.asp

Definition and Usage

The replace() method searches a string for a specified value, or a *regular expression*, and returns a new string where the specified values are replaced.

Note: If you are replacing a value (and not a *regular expression*), only the first instance of the value will be replaced. To replace all occurrences of a specified value, use the global (g) modifier (see "More Examples" below).

```
let str = "batman brewery";
str = str.replace("b", "f");
console.log(str);
```

```
fatman brewery
```

```
let str = "batman brewery";
str = str.replace(/b/g, "f");
console.log(str);
```

```
fatman frewery
```

```
function switcheroo(x){
  return x.replace(/[ab]/g, function(c) { return c === 'a' ? 'b' : 'a'; });
}
let t2 = "aabbacacbba";
console.log(switcheroo(t2));
```

```
bbaabcbcaab
```

```
function replace(s){
  return s.replace(/[aeoiu]/ig, '!')
}
let t1 = "Batemanu";
console.log(replace(t1));
```

```
B!t!m!n!
```

JavaScript **round()** Method

https://www.w3schools.com/jsref/jsref_round.asp

Definition and Usage

The round() method rounds a number to the nearest integer.

Note: 2.49 will be rounded down (2), and 2.5 will be rounded up (3).

```
> Math.round(2.5);
< 3
```

JavaScript **Array filter()** Method

https://www.w3schools.com/jsref/jsref_filter.asp

Definition and Usage

The filter() method creates an array filled with all array elements that pass a test (provided as a function).

Note: filter() does not execute the function for array elements without values.

Note: filter() does not change the original array.

Syntax:

array.filter(function(currentValue, index, arr), thisValue)

Parameter	Description
function(currentValue, index,arr)	Required. A function to be run for each element in the array. Function arguments:

Argument	Description

	currentValue	Required. The value of the current element
	index	Optional. The array index of the current element
	arr	Optional. The array object the current element belongs to
thisValue	Optional. A value to be passed to the function to be used as its "this" value. If this parameter is empty, the value "undefined" will be passed as its "this" value	

JavaScript **Array concat()** Method

Definition and Usage

The `concat()` method is used to join two or more arrays.

This method does not change the existing arrays, but returns a new array, containing the values of the joined arrays.

```
var hege = ["Cecilie", "Lone"];
var stale = ["Emil", "Tobias", "Linus"];
var children = hege.concat(stale);
console.log(children);
▶ (5) ["Cecilie", "Lone", "Emil", "Tobias", "Linus"]
```

Syntax:

array1.concat(array2, array3, ..., arrayX)

Parameter	Description
array2, *array3*, ..., *arrayX*	Required. The arrays to be joined

JavaScript **Array every()** Method

https://www.w3schools.com/jsref/jsref_every.asp

Definition and Usage

The `every()` method checks if all elements in an array pass a test (provided as a function).

The every() method executes the function once for each element present in the array:

- If it finds an array element where the function returns a *false* value, every() returns false (and does not check the remaining values)
- If no false occur, every() returns true

```
> var ages = [32, 33, 16, 40];

  function checkAdult(age) {
    return age >= 18;
  }
  function myFunction(){
    return ages.every(checkAdult)
  };
  myFunction();
< false
```

Array.prototype.flat()