Array.prototype.length

The **`length`** property of an object which is an instance of type `Array` sets or returns the number of elements in that array. The value is an unsigned, 32-bit integer that is always numerically greater than the highest index in the array.

Try it

JavaScript Demo: Array.length

```
1 const clothing = ['shoes', 'shirts', 'socks', 'sweaters'];
2
3 console.log(clothing.length);
4 // expected output: 4
5
```

Run ›

Reset

Description

The value of the `length` property is an integer with a positive sign and a value less than 2 to the 32nd power (2^32).

```
const listA = [1,2,3];
const listB = new Array(6);

console.log(listA.length);
// 3

console.log(listB.length);
// 6

listB.length = 4294967296; //2 to the 32nd power = 4294967296
// RangeError: Invalid array length

const listC = new Array(-100) //negative sign
// RangeError: Invalid array length
```

You can set the `length` property to truncate an array at any time. When you extend an array by changing its `length` property, the number of actual elements increases; for example, if you set `length` to 3 when it is currently 2, the array now contains 3 elements, which causes the third element to be a non-iterable empty slot.

```
const arr = [1, 2];
console.log(arr);
// [ 1, 2 ]

arr.length = 5; // set array length to 5 while currently 2.
console.log(arr);
// [ 1, 2, <3 empty items> ]

arr.forEach((element) => console.log(element));
// 1
// 2
```

As you can see, the `length` property does not necessarily indicate the number of defined values in the array. See also Relationship between `length` and numerical properties.

Property attributes of `Array.prototype.length`	
Writable	yes
Enumerable	no
Configurable	no

- `Writable` : If this attribute set to `false` , the value of the property cannot be changed.
- `Configurable` : If this attribute set to `false` , any attempts to delete the property or change its attributes (`Writable` , `Configurable` , or `Enumerable`) will fail.
- `Enumerable` : If this attribute set to `true` , the property will be iterated over during `for` or `for...in` loops.

Examples

Iterating over an array

In the following example, the array `numbers` is iterated through by looking at the `length` property. The value in each element is then doubled.

```
const numbers = [1, 2, 3, 4, 5];
const length = numbers.length;
for (let i = 0; i < length; i++) {
  numbers[i] *= 2;
}
// numbers is now [2, 4, 6, 8, 10]
```

Shortening an array

The following example shortens the array `numbers` to a length of 3 if the current length is greater than 3.

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

```
if (numbers.length > 3) {
  numbers.length = 3;
}

console.log(numbers); // [1, 2, 3]
console.log(numbers.length); // 3
```

Create empty array of fixed length

```
const numbers = [];
numbers.length = 3;
console.log(numbers); // [empty x 3]
```

Specifications

Specification
ECMAScript Language Specification # sec-properties-of-array-instances-length

Browser compatibility

Report problems with this compatibility data on GitHub

	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	Chrome Android	Firefox for Android	Opera Android
length	Chrome 1	Edge 12	Firefox 1	Internet Explorer 4	Opera 4	Safari 1	Chrome Android 18	Firefox for Android 4	Opera Android 10.1

Full support

See also

- Array
- RangeError: invalid array length

Last modified: Jul 23, 2022, by MDN contributors