



Math.max()

English ▼

The **Math.max()** function returns the largest of zero or more numbers.

JavaScript Demo: Math.max()

```
1 console.log(Math.max(1, 3, 2));
2 // expected output: 3
3
4 console.log(Math.max(-1, -3, -2));
5 // expected output: -1
6
7 const array1 = [1, 3, 2];
8
9 console.log(Math.max(...array1));
10 // expected output: 3
11
```

Run >

Reset

Syntax

```
Math.max([value1[, value2[, ...]]])
```



Parameters

`value1, value2, ...`

Numbers.

Return value

The largest of the given numbers. If at least one of the arguments cannot be converted to a number, `NaN` is returned.

Description

Because `Math` is not a constructor, `max()` is a static method of `Math` (You always use it as `Math.max()`, rather than as a method of an instanced `Math` object).

`-Infinity` is the initial comparant because almost every other value is bigger, that's why when no arguments are given, `-Infinity` is returned.

If at least one of arguments cannot be converted to a number, the result is `NaN`.

Examples

Using `Math.max()`

```
1 | Math.max(10, 20);    // 20
2 | Math.max(-10, -20); // -10
3 | Math.max(-10, 20);   // 20
```

Getting the maximum element of an array

`Array.reduce()` can be used to find the maximum element in a numeric array, by comparing each value:

```
1 | var arr = [1,2,3];
2 | var max = arr.reduce(function(a, b) {
```



```
3 |   return Math.max(a, b);
4 | });
```

The following function uses `Function.prototype.apply()` to get the maximum of an array. `getMaxOfArray([1, 2, 3])` is equivalent to `Math.max(1, 2, 3)`, but you can use `getMaxOfArray()` on programmatically constructed arrays. This should only be used for arrays with relatively few elements.

```
1 | function getMaxOfArray(numArray) {
2 |   return Math.max.apply(null, numArray);
3 | }
```

The new [spread operator](#) is a shorter way of writing the `apply` solution to get the maximum of an array:

```
1 | var arr = [1, 2, 3];
2 | var max = Math.max(...arr);
```

However, both spread (`...`) and `apply` will either fail or return the wrong result if the array has too many elements, because they try to pass the array elements as function parameters. See [Using `apply` and built-in functions](#) for more details. The `reduce` solution does not have this problem.

Specifications

Specification

ECMAScript Latest Draft (ECMA-262)

The definition of 'Math.max' in that specification.

Browser compatibility

[Update compatibility data on GitHub](#)

max

Chrome

1

What are we missing?



See also

- `Math.min()`

🕒 **Last modified:** Jan 15, 2020, by [MDN contributors](#)

- Syntax
- Description
- Examples
- Specifications
- Browser compatibility
- See also







Related Topics

Standard built-in objects

Math

Properties

`Math.E`

`Math.LN10`

`Math.LN2`

`Math.LOG10E`

`Math.LOG2E`

`Math.PI`

`Math.SQRT1_2`

`Math.SQRT2`

Methods

`Math.abs()`

`Math.acos()`

`Math.acosh()`

`Math.asin()`

`Math.asinh()`

`Math.atan()`

`Math.atan2()`

`Math.atanh()`

`Math.cbrt()`

`Math.ceil()`

`Math.clz32()`

`Math.cos()`

`Math.cosh()`

`Math.exp()`

`Math.expm1()`

`Math.floor()`

`Math.fround()`

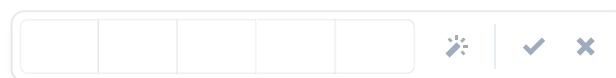
`Math.hypot()`

`Math.imul()`

`Math.log()`

`Math.log10()`

`Math.log1p()`







```
Math.log2()  
  
Math.max()  
  
Math.min()  
  
Math.pow()  
  
Math.random()  
  
Math.round()  
  
Math.sign()  
  
Math.sin()  
  
Math.sinh()  
  
Math.sqrt()  
  
Math.tan()  
  
Math.tanh()  
  
Math.trunc()
```



Inheritance:


Object

Properties

  Object.prototype.__count__

  Object.prototype.__noSuchMethod__

  Object.prototype.__parent__

 Object.prototype.__proto__

Object.prototype.constructor

Methods

 Object.prototype.__defineGetter__()

 Object.prototype.__defineSetter__()

 Object.prototype.__lookupGetter__()

 Object.prototype.__lookupSetter__()

Object.prototype.hasOwnProperty()

Object.prototype.isPrototypeOf()

Object.prototype.propertyIsEnumerable()

Object.prototype.toLocaleString()

 Object.prototype.toSource()

Object.prototype.toString()

  Object.prototype.unwatch()

Object.prototype.valueOf()



 `Object.prototype.watch()``Object.setPrototypeOf()`

Learn the best of web development

Get the latest and greatest from MDN delivered straight to your inbox.

Sign up now