Numbers

**JavaScript NaN**

In JavaScript, NaN(Not a Number) is a keyword that indicates that the value is not a number.

Performing arithmetic operations (except + ) to numeric value with string results in NaN. For example,

const a = 4 - 'hello';

console.log(a); // NaN

The built-in function isNaN() can be used to find if a value is a number. For example,

const a = isNaN(9);

console.log(a); // false

const a = isNaN(4 - 'hello');

console.log(a); // true

When the typeof operator is used for NaN value, it gives a number output. For example,

const a = 4 - 'hello';

console.log(a); // NaN

console.log(typeof a); // "number"

**JavaScript Infinity**

In JavaScript, when calculation is done that exceeds the largest (or smallest) possible number, Infinity (or -Infinity) is returned. For example,

const a = 2 / 0;

console.log(a); // Infinity

const a = -2 / 0;

console.log(a); // -Infinity

**JavaScript BigInt**

In JavaScript, Number type can only represent numbers less than **(253** **- 1)** and more than **-(253** **- 1)**. However, if you need to use a larger number than that, you can use the BigInt data type.

A BigInt number is created by appending **n** to the end of an integer. For example,

// BigInt value

const value = 900719925124740998n;

// Adding two big integers

const value1 = value + 1n;

console.log(value1); // returns "900719925124740999n"

**Note:** BigInt was introduced in the newer version of JavaScript and is not supported by many browsers. Visit [JavaScript BigInt support](https://caniuse.com/#feat=bigint) to learn more.

**JavaScript Numbers Are Stored in 64-bit**

In JavaScript, numbers are stored in 64-bit format [IEEE-754](https://en.wikipedia.org/wiki/IEEE_754-2008_revision), also known as "double precision floating point numbers".

The numbers are stored in 64 bits (the number is stored in 0 to 51 bit positions, the exponent in 52 to 62 bit positions and the sign in 63 bit position).

|  |  |  |
| --- | --- | --- |
| Numbers | Exponent | Sign |
| 52 bits(0 - 51) | 11 bits(52- 62) | 1 bit(63) |

**Precision Problems**

Operations on floating-point numbers results in some unexpected results. For example,

const a = 0.1 + 0.2;

console.log(a); // 0.30000000000000004

[Run Code](https://www.programiz.com/javascript/online-compiler)

The result should be **0.3** instead of **0.30000000000000004**. This error occurs because in JavaScript, numbers are stored in binary form to represent decimal digits internally. And decimal numbers can't be represented in binary form exactly.

To solve the above problem, you can do something like this:

const a = (0.1 \* 10 + 0.2 \* 10) / 10;

console.log(a); // 0.3

You can also use the toFixed() method.

const a = 0.1 + 0.2;

console.log(a.toFixed(2)); // 0.30

toFixed(2) rounds up the decimal number to two decimal values.

const a = 9999999999999999

console.log(a); // 10000000000000000

**Note**: Integers are accurate up to 15 digits.

**Number Objects**

You can also create numbers using the new keyword. For example,

const a = 45;

// creating a number object

const b = new Number(45);

console.log(a); // 45

console.log(b); // 45

console.log(typeof a); // "number"

console.log(typeof b); // "object"

## JavaScript Number Methods

Here is a list of built-in number methods in JavaScript.

|  |  |
| --- | --- |
| Method | Description |
| isNaN() | determines whether the passed value is NaN |
| isFinite() | determines whether the passed value is a finite number |
| isInteger() | determines whether the passed value is an integer |
| isSafeInteger() | determines whether the passed value is a safe integer |
| parseFloat(string) | converts the numeric floating string to floating-point number |
| parseInt(string, [radix]) | converts the numeric string to integer |
| toExponential(fractionDigits) | returns a string value for a number in exponential notation |
| toFixed(digits) | returns a string value for a number in fixed-point notation |
| toPrecision() | returns a string value for a number to a specified precision |
| toString([radix]) | returns a string value in a specified radix(base) |
| valueof() | returns the numbers value |
| toLocaleString() | returns a string with a language sensitive representation of a number |

## JavaScript Number Properties

Here is a list of Number properties in JavaScript.

|  |  |
| --- | --- |
| Property | Description |
| EPSILON | returns the smallest interval between two representable numbers |
| MAX\_SAFE\_INTEGER | returns the maximum safe integer |
| MAX\_VALUE | returns the largest possible value |
| MIN\_SAFE\_INTEGER | returns the minimum safe integer |
| MIN\_VALUE | returns the smallest possible value |
| NaN | represents 'Not-a-Number' value |
| NEGATIVE\_INFINITY | represents negative infinity |
| POSITIVE\_INFINITY | represents positive infinity |
| prototype | allows the addition of properties to Number objects |