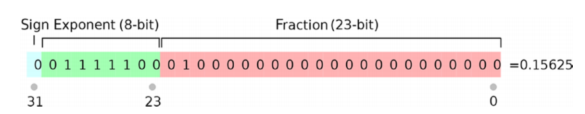
**Number System**

JavaScript uses a 32-bit floating-point representation for numbers.

The sign bit (the 31st bit) indicates that the number is negative if the sign bit is 1. The next 8 bits (the 30th to 23rd bits) indicate the exponent value, e. Finally, the remaining 23 bits represent the fraction value



**JavaScript Number Object**

**Integer Rounding**

Since JavaScript uses floating point to represent all numbers, integer division does not work.

Integer division in programming languages like Java simply evaluates division expressions to their quotient.

For example, 5/4 is 1 in Java because the quotient is 1 (although there is a remainder of 1 left).

However, in JavaScript, it is a floating point. 1 5/4; // 1.25

This is because Java requires you to explicitly type the integer as an integer. Hence, the result cannot be a floating point.

However, if JavaScript developers want to implement integer division, they can do one of the following:

Math.floor - rounds down to nearest integer

Math.round - rounds to nearest integer

Math.ceil - rounds up to nearest integer

*Math.floor(0.9); // 0*

*Math.floor(1.1); // 1*

*Math.round(0.49); // 0*

*Math.round(0.5); // 1*

*Math.round(2.9); // 3*

*Math.ceil(0.1); // 1*

*Math.ceil(0.9); // 1*

*Math.ceil(21); // 21*

*Math.ceil(21.01); // 22*

**Number.EPSILON**

Number.EPSILON returns the smallest interval between two representable numbers.

This is useful for the problem with floating-point approximation. 1

function **numberEquals**(x, y)

{

return Math.abs(x - y) < Number.EPSILON;

}

**numberEquals(**0.1 + 0.2, 0.3); // true

This function works by checking whether the difference between the two numbers are smaller than Number.EPSILON. *Remember that Number.EPSILON is the smallest difference between two representable numbers*. The difference between 0.1+0.2 and 0.3 will be smaller than Number.EPSILON.

**Maximums**

**Number.MAX\_SAFE\_INTEGER** returns the largest integer.

Example,

1. Number.MAX\_SAFE\_INTEGER + 1 === Number.MAX\_SAFE\_INTEGER + 2; // true
2. Number.MAX\_SAFE\_INTEGER + 1.111 === Number.MAX\_SAFE\_INTEGER + 2.022; // false

**Number.MAX\_VALUE** returns the largest floating-point number possible.

**Minimums**

**Number.MIN\_SAFE\_INTEGER** returns the smallest integer.

*Number.MIN\_SAFE\_INTEGER is equal to -9007199254740991*

**Number.MIN\_VALUE** is also the closest floating point to zero.

**Infinity**

The only thing greater than *Number.MAX\_VALUE is Infinity*, and the only thing smaller than *Number.MAX\_SAFE\_INTEGER is -Infinity.*

This inequality summarizes the size of JavaScript numbers from smallest (left) to largest (right): -Infinity < Number.MIN\_SAFE\_INTEGER < Number.MIN\_VALUE < 0 < Number.MAX\_ SAFE\_IN- TEGER < Number.MAX\_VALUE < Infinity