Somos un ecosistema de desarrolladores de software

</Riwi>

Introducción a Java Script

Number

Propiedades

En Javascript, los **números** son uno de los tipos de datos básicos (tipos primitivos), que, para crearlos, simplemente basta con escribirlos literalmente.

Number

Propiedades

Constructor	Descripción
new Number(number)	Crea un objeto numérico a partir del número number pasado por parámetro.
number	Simplemente, el número en cuestión. **Notación preferida**.

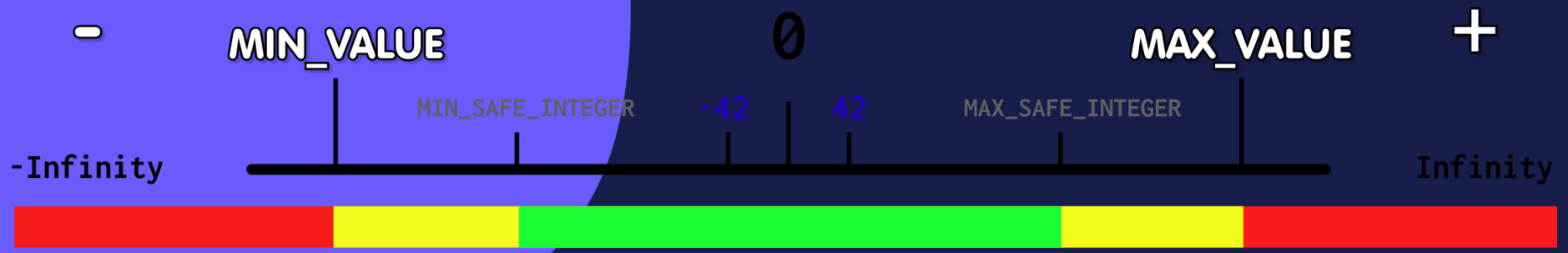
```
//Cualquier parámetro pasado al new Number() que no sea un número (por ejemplo, la "A"), dará como resultado un valor NaN (Not A Number)
```

```
// Notación literal (preferida)
const number = 4;
const decimal = 15.8;
const legibleNumber = 5_000_000;

// Notación con objetos (evitar)
const number = new Number(4);
const decimal = new Number(15.8);
const letter = new Number("A");
```

Number

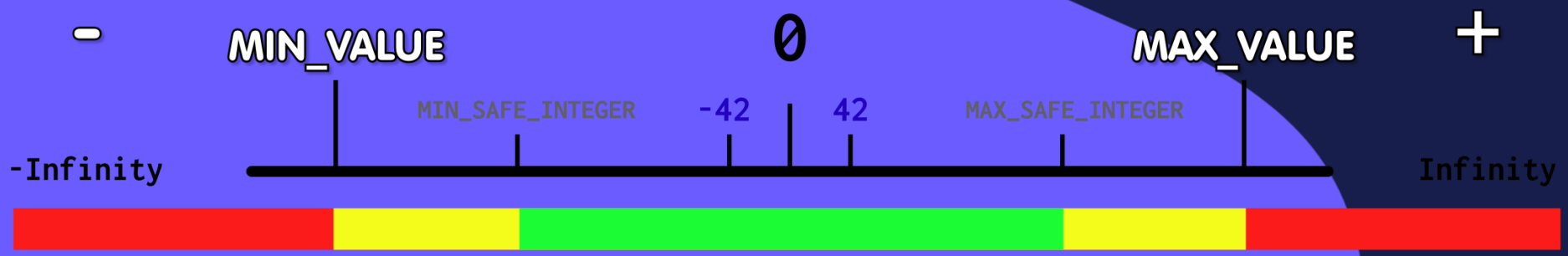
Rangos Numéricos

Algo que hay que tener muy claro en Javascript (y en general, en programación, ya que no es algo propio de Javascript) es que cuando trabajamos con datos numéricos, es posible que ciertos números no se puedan representar exactamente, y no sean tan precisos como nos gustaría.



Number

Rangos Numéricos



Constante	Valor en Javascript	Descripción
Number.MAX_VALUE	~ 2^{1024}	Valor más grande
Number.MIN_VALUE	~ 5×10^{-324}	Valor más pequeño
Number.MAX_SAFE_INTEGER	$2^{53}-1$	Valor seguro más grande
Number.MIN_SAFE_INTEGER	$-(2^{53}-1)$	Valor seguro más pequeño
Number.EPSILON	2^{-52}	Número muy pequeño: ε
Number.POSITIVE_INFINITY	Infinity	Infinito positivo: +∞
Number.NEGATIVE_INFINITY	-Infinity	Infinito negativo: -∞

Number

Comprobaciones numéricas

Método	Descripción
Number.isFinite(number)	Comprueba si number es un número finito.
Number.isInteger(number)	Comprueba si number es un número entero.
Number.isSafeInteger (number)	Comprueba si number es un número seguro.

Ten en cuenta que la notación 1e5 significa «5 ceros seguidos de un 1», es decir, 100000.

```
// ¿Número finito?
Number.isFinite(42);             // true
Number.isFinite(551.3);          // true
Number.isFinite(Infinity);       // false, es infinito

// ¿Número entero?
Number.isInteger(5);             // true
Number.isInteger(4.6);           // false, es decimal

// ¿Número seguro?
Number.isSafeInteger(1e15);      // true (valor en la franja verde)
Number.isSafeInteger(1e16);      // false (valor en la franja
amarilla)

1e309 === Infinity;              // true
Number.isSafeInteger(1e309);     // false (valor en la franja roja)
```

Number

Representación numérica

Método	Descripción
.toExponential(digits)	Convierte el número a **notación exponencial** con digits decimales.
.toFixed(digits)	Convierte el número a **notación de punto fijo** con digits decimales.
.toPrecision(size)	Utiliza size dígitos de precisión en el número.

```
No olvides que en todos los casos, el número se convierte a tipo de dato STRING .
```

```
// Notación exponencial
(1.25).toExponential(0);      // "1e+0"
(1.25).toExponential(1);      // "1.3e+0"
(1.25).toExponential(2);      // "1.25e+0"
(1.25).toExponential(3);      // "1.250e+0"

// Notación punto fijo
(1.25).toFixed(0);            // "1"
(1.25).toFixed(1);            // "1.3"
(1.25).toFixed(2);            // "1.25"
(1.25).toFixed(3);            // "1.250"
// Cambiando precisión
(523.75).toPrecision(1);      // "5e+2"
(523.75).toPrecision(2);      // "5.2e+2"
(523.75).toPrecision(3);      // "524"
(523.75).toPrecision(4);      // "523.8"
(523.75).toPrecision(5);      // "523.75"
```

Number

Convertir texto a números

Método	Descripción
Number.parseInt(text)	Convierte un STRING text en un NUMBER entero.
Number.parseInt(text, radix)	Idem, pero el STRING tiene un número en base NUMBER radix.
Number.parseFloat(text)	Convierte un STRING text en un NUMBER decimal.
Number.parseFloat(text, radix)	Idem, pero el STRING tiene un número en base NUMBER radix.

Number

Convertir texto a números

```
Number.parseInt("42");       // 42
Number.parseInt("42€");      // 42 (descarta todo desde un carácter no numérico)
Number.parseInt("Núm. 42"); // NaN (empieza a descartar en Núm, descarta también 42)
Number.parseInt("A");        // NaN (No se puede representar como un número)
Number.parseInt("");         // NaN (No se puede representar como un número)
Number.parseInt("42.5");       // 42 (descarta los decimales)
Number.parseInt("88.99€");     // 88 (descarta decimales y resto de caracteres)
Number.parseInt("Núm. 33.5"); // NaN (empieza a descartar en Núm, descarta todo)
```

```
Number.parseFloat("42.5");       // 42.5 (Conserva decimales)
Number.parseFloat("42");         // 42 (El número es entero, convierte a entero)
Number.parseFloat("88.99€");     // 88.99 (Conserva decimales)
Number.parseFloat("42€");        // 42 (El número es entero, convierte a entero)
Number.parseFloat("Núm. 33.5"); // NaN (empieza a descartar en Núm, descarta todo)
```

Number

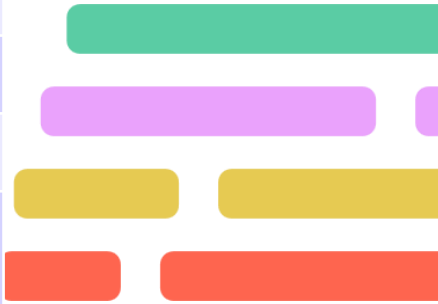
De Numer a String

```
(16).toString();         // "16"     (lo convierte a string)
(42.5).toString();       // "42.5"   (lo convierte a string)
(26).toString(2);        // "11010"  (26 en decimal, es 11010 en binario)
(80).toString(8);        // "120"    (80 en decimal, es 120 en octal)
(245123).toString(16);   // "3bd83"  (245123 en decimal, es 3bd83 en hexadecimal)
```

Number

Métodos Matemáticos

Método	Descripción	Ejemplo
Math.abs(x)	Devuelve el valor absoluto de x.	|x|
Math.sign(x)	Devuelve el signo del número: 1 positivo, -1 negativo	
Math.exp(x)	Exponenciación. Devuelve el número e elevado a x.	e^x
Math.expm1(x)	Equivalente a Math.exp(x) - 1.	e^x-1
Math.max(a, b, c...)	Devuelve el número más grande de los indicados por parámetro.	
Math.min(a, b, c...)	Devuelve el número más pequeño de los indicados por parámetro.	
Math.pow(base, exp)	Potenciación. Devuelve el número base elevado a exp.	$base^{exp}$
Math.sqrt(x)	Devuelve la raíz cuadrada de x.	√x
Math.cbrt(x)	Devuelve la raíz cúbica de x.	$\sqrt[3]{x}$
Math.imul(a, b)	Equivalente a a * b, pero a nivel de bits.	
Math.clz32(x)	Devuelve el número de ceros a la izquierda de x en binario (32 bits).	
Math.random()	Devuelve un número al azar entre 0 y 1 con 16 decimales.	



Number

Métodos Matemáticos

```
Math.abs(-5);               // 5
Math.sign(-5);              // -1
Math.exp(1);                // e, o sea, 2.718281828459045
Math.expm1(1);              // 1.718281828459045
Math.max(1, 40, 5, 15);     // 40
Math.min(5, 10, -2, 0);     // -2
Math.pow(2, 10);            // 1024 (Equivale a 2**10)
Math.sqrt(2);               // 1.4142135623730951 (Equivale a Math.SQRT2)
Math.cbrt(2);               // 1.2599210498948732
Math.imul(0xffffffff, 7); // -7

// Ejemplo de clz32 (count leading zeros)
const x = 1;
"0".repeat(Math.clz32(x)) + x.toString(2);
// Devuelve "00000000000000000000000000000001"
```

Number

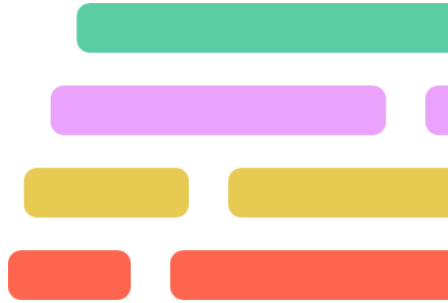
.random()

```
// Obtenemos un número al azar entre [0, 1) con 16 decimales
let x = Math.random();

// Multiplicamos dicho número por el valor máximo que buscamos (5)
x = x * 5;

// Redondeamos inferiormente, quedándonos sólo con la parte entera
x = Math.floor(x);

// Número al azar entre 0 y 5 (no incluido)
const x = Math.floor(Math.random() * 5);

// Equivalente al anterior
const x = ~(Math.random() * 5);
```

Number

Métodos de redondeo

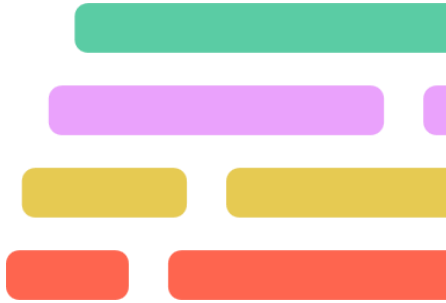
Método	Descripción
Math.round(x)	Devuelve x con **redondeo** (el entero más cercano)
Math.ceil(x)	Devuelve x con **redondeo superior** (el entero más alto)
Math.floor(x)	Devuelve x con **redondeo inferior** (el entero más bajo)
Math.fround(x)	Devuelve x con **redondeo** (flotante con precisión simple)
Math.trunc(x)	Trunca el número x (devuelve sólo la parte entera)



Number

Métodos de redondeo

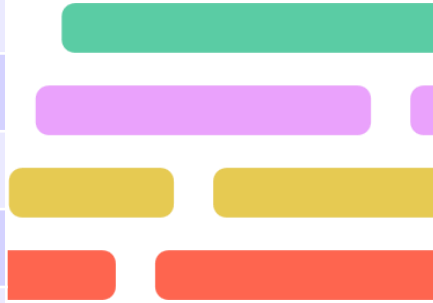
```
// Redondeo natural, el más cercano
Math.round(3.75);           // 4
Math.round(3.25);           // 3

// Redondeo superior (el más alto)
Math.ceil(3.75);            // 4
Math.ceil(3.25);            // 4

// Redondeo inferior (el más bajo)
Math.floor(3.75);           // 3
Math.floor(3.25);           // 3

// Redondeo con precisión
Math.round(3.123456789);    // 3
Math.fround(3.123456789);   // 3.1234567165374756

// Truncado (sólo parte entera)
Math.trunc(3.75);           // 3
Math.round(-3.75);          // -4
Math.trunc(-3.75);          // -3
```



Number

Métodos trigonométricos

Método	Descripción
Math.sin(x)	Seno de x
Math.asin(x)	Arcoseno de x
Math.sinh(x)	Seno hiperbólico de x
Math.asinh(x)	Arcoseno hiperbólico de x
Math.cos(x)	Coseno de x
Math.acos(x)	Arcocoseno de x
Math.cosh(x)	Coseno hiperbólico de x
Math.acosh(x)	Arcocoseno hiperbólico de x
Math.tan(x)	Tangente de x
Math.atan(x)	Arcotangente de x
Math.tanh(x)	Tangente hiperbólica de x
Math.atanh(x)	Arcotangente hiperbólica de x
Math.atan2(x, y)	Arcotangente del conciente de x/y
Math.hypot(a, b..)	Devuelve la raíz cuadrada de $a^2 + b^2 + ...$



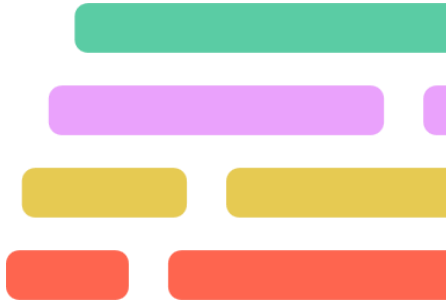
Number

BigInt

BigInt es un tipo de dato de Javascript que nace con la idea de permitir representar valores muy grandes, de una forma relativamente sencilla y compatible con lo que ya existe.

Constructor	Descripción
BigInt(number)	Devolvemos un número BigInt a partir de un number pasado por parámetro.
number+n	Simplemente, añadir una n al final del número. **Notación preferida**.

La problemática que existe actualmente con el tipo de dato Number es que valores más grandes de 253-1 no pueden ser representados, ya que superarían el límite seguro Number.MAX_SAFE_INTEGER y algunos resultados podrían perder precisión



Number

BigInt

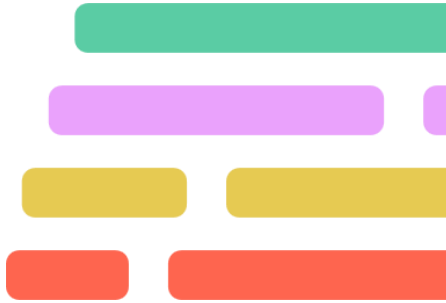
```
const number = 2 ** 53;

number.constructor.name;     // "Number" (Es de
tipo Number)

number;          // 9007199254740992
number + 1;      // 9007199254740992
number + 5;      // 9007199254740996
number + 40;     // 9007199254741032
```

Number

Mezclando tipos de dato

```
const number = 5 + 5n;          // ERROR: Cannot mix BigInt and other types,
use explicit conversions
const number = BigInt(5) + 5n   // 10n (Ok, convierte a BigInt y realiza la
operación)
const number = 5 + Number(5n);  // 10 (Ok, convierte a Number y realiza la
operación)
```

- Pasas el Number a BigInt correctamente, pero el Number ya había perdido precisión previamente
- Pasas el BigInt a Number pero pierdes precisión en el proceso, porque Number no puede representarlo



Number

Métodos de BigInt

Método	Descripción
BigInt.asIntN(bits, bigNumber)	Ajusta bigNumber a Number bits (devuelve entero con signo).
BigInt.asUintN(bits, bigNumber)	Ajusta bigNumber a Number bits (devuelve entero sin signo).

Number

Métodos de BigInt

```
// Con 2 bits con signo puedes representar desde (-2 hasta 1)
BigInt.asIntN(2, -3n);   // -3n (Representa el -3 con 2 bits) ❌
BigInt.asIntN(2, -2n);   // -2n (Representa el -2 con 2 bits) ✅ 10
BigInt.asIntN(2, -1n);   // -1n (Representa el -1 con 2 bits) ✅ 11
BigInt.asIntN(2, 0n);    // 0n (Representa el 0 con 2 bits) ✅ 00
BigInt.asIntN(2, 1n);    // 1n (Representa el 1 con 2 bits) ✅ 01
BigInt.asIntN(2, 2n);    // -2n (Representa el 2 con 2 bits) ❌
BigInt.asIntN(2, 3n);    // -1n (Representa el 3 con 2 bits) ❌
```

Number

Métodos de BigInt

```
// Con 2 bits sin signo puedes representar desde (0 hasta 4)
BigInt.asUintN(2, -2n);   // 2n (Representa el -2 con 2 bits) ❌
BigInt.asUintN(2, -1n);   // 3n (Representa el -1 con 2 bits) ❌
BigInt.asUintN(2, 0n);    // 0n (Representa el 0 con 2 bits) ✅ 00
BigInt.asUintN(2, 1n);    // 1n (Representa el 1 con 2 bits) ✅ 01
BigInt.asUintN(2, 2n);    // 2n (Representa el 2 con 2 bits) ✅ 10
BigInt.asUintN(2, 3n);    // 3n (Representa el 3 con 2 bits) ✅ 11
BigInt.asUintN(2, 4n);    // 0n (Representa el 4 con 2 bits) ❌
```

</Be a coder>