OBJETOS PREDEFINIDO NUMBER EN JAVASCRIPT

Juan Rueda Morales y Rafael Delgado Peña

Índice

1. ¿Qué es?

Es un objeto envolvente que nos permite trabajar con datos numéricos. Con Number es posible declarar variables con valores números enteros y decimales.

En JavaScript se crean objetos Number con la declaración de variables de valor numérico y en raras veces es necesario declarar el objeto de forma explícita.

2. Sintaxis

Declaración de una variable de tipo numérico:

```
> var x = 5;
  x
< 5
```

Declaración e iniciación de un objeto number:

```
> var numero = new Number (5);
  numero
< ▶ Number {[[PrimitiveValue]]: 5}
```

3. Parámetros

El único parámetro que se declara es el valor numérico que se le asigna al objeto.

4. Propiedades

Number posee las propiedades heredadas de Object además de las suyas propias que son las siguientes:

- NaN (Not a Number): no es un número válido.
- MAX_VALUE: mayor número representable.
- MIN_VALUE: menor número representable.
- NEGATIVE_INFINITY: infinitos negativos.
- POSITIVE_INFINITY: infinitos positivos.

```
1  console.log("Propiedad NaN: " + Number.NaN)
2  console.log("Propiedad MAX_VALUE: " + Number.MAX_VALUE)
3  console.log("Propiedad MIN_VALUE: " + Number.MIN_VALUE)
4  console.log("Propiedad NEGATIVE_INFINITY: " + Number.NEGATIVE_INFINITY)
5  console.log("Propiedad POSITIVE_INFINITY: " + Number.POSITIVE_INFINITY)
```

```
Propiedad NaN: NaN                                          prueba.js:2
Propiedad MAX_VALUE: 1.7976931348623157e+308                prueba.js:3
Propiedad MIN_VALUE: 5e-324                                 prueba.js:4
Propiedad NEGATIVE_INFINITY: -Infinity                      prueba.js:5
Propiedad POSITIVE_INFINITY: Infinity                       prueba.js:6
>
```

5. Palabra reservada NaN

Esta palabra reservada sirve para indicar la propiedad de que el valor introducido no es un número con un formato correcto. De ahí sus iniciales en inglés, Not A Number.

6. Métodos

- **.toExponential():** Devuelve una cadena con el número en notación exponencial.
- **.toFixed():** Devuelve una cadena con el número en una notación de punto fijo.
- **.toLocaleString():** Devuelve un objeto convertido en una cadena según la configuración regional.
- **.toPrecision()**: Retorna una cadena representando el número en una notación de precisión de punto fijo.
- **.toSource():** Devuelve una cadena que contiene un número representado en notación exponencial o de punto fijo y que tiene un número de dígitos especificado.
- **.toString()**: Devuelve una cadena representando el objeto especificado.
- **.valueOf():** Devuelve el valor primitivo de un objeto especificado.

```
{
    numero = new Number(2.564);
    console.log("Método toExponential(): " + numero.toExponential(5));
    console.log("Método toFixed(): " + numero.toFixed(5));
    console.log("Método toLocaleString(): " + numero.toLocaleString());
    console.log("Método toExponential(): " + numero.toString());
    console.log("Método toExponential(): " + numero.toPrecision(2));
}
```

```
Método toExponential(): 2.56400e+0                    prueba.js:3
Método toFixed(): 2.56400                             prueba.js:4
Método toLocaleString(): 2,564                        prueba.js:5
Método toExponential(): 2.564                         prueba.js:6
Método toExponential(): 2.6                           prueba.js:7
```

7. Precisión:

Son números sin una notación exponencial que se redondean cuando llegan a 15 dígitos, pongamos unos ejemplos:

```
> var x = 999999999999999;
  var y = 9999999999999999;
< undefined
> x
< 999999999999999
> y
< 10000000000000000
>
```

8. Adición de números y cadenas

En JavaScript el operador '+' se usa tanto para la suma como para la concatenación. Esto ha de tenerse en cuenta cuando se trabaja con variables de tipo numérico y cadenas que contienen números. A continuación vemos las diferentes posibilidades que nos encontraremos a través de ejemplos:

- **Dos variables numéricas:**

```
> var x = 10;
  var y = 20;
  var z = x + y;
  z
< 30
>
```

- **Dos cadenas de texto que contienen números:**

```
> var x = "10";
  var y = "20";
  var z = x + y;
  z
< "1020"
>
```

- **Una variable numérica y una cadena de texto:**

```
> var x = 10;
  var y = "20";
  var z = x + y;
  z
< "1020"
```

- **Dos variables numéricas mostrada por pantalla junto a una cadena:**

```
> var x = 10;
  var y = 20;
  var z = "The result is: " + x + y;
  z
< "The result is: 1020"
```

- **Dos variables numéricas y una cadena de texto:**

```
> var x = 10;
  var y = 20;
  var z = "30";
  var result = x + y + z;
  result
< "3030"
```

Las cadenas numéricas

Conociendo lo explicado en el punto anterior debemos saber también que JavaScript intenta convertir cadenas a números cuando se realizan algunas operaciones matemáticas con ellas. Veamos ejemplos:

```
> var x = "100";
  var y = "10";
  var z = x / y;
  z
< 10
```

```
> var x = "100";
  var y = "10";
  var z = x * y;
  z
< 1000
```

```
> var x = "100";
  var y = "10";
  var z = x - y;
  z
< 90
```

Con estas operaciones el resultado resultante de las mismas es un valor numérico mientras que si intentamos la suma obtendremos una concatenación de las cadenas como vimos en el punto anterior.