Lenguaje de Marcas: Javascript

Iván Nieto Ruiz

il.nietoruiz@edu.gva.es

Numeros

```
let age = 35
const GRAVITY = 9.81
let mass = 72
const PI = 3.14
```

Class Math

```
// Redondear al número más cercano
console.log(Math.PI); // 3.141592653589793
console.log(Math.round(Math.PI)); // 3
console.log(Math.round(9.81)); // 10
// Redondeo hacia abajo
console.log(Math.floor(PI)); // 3
// Redondeo hacia arriba
console.log(Math.ceil(PI)); // 4
// Mínimo valor en una lista de números
console.log(Math.min(-5, 3, 20, 4, 5, 10)); // -5
// Máximo valor en una lista de números
console.log(Math.max(-5, 3, 20, 4, 5, 10)); // 20
```

Numeros

Class Math

```
// Crear un número aleatorio entre 0 y 0.999999
const randNum = Math.random();
console.log(randNum);
// Crear un número aleatorio entre 0 y 10
const num = Math.floor(Math.random() * 11);
console.log(num);
// Valor absoluto
console.log(Math.abs(-10)); // 10
// Raíz cuadrada
console.log(Math.sqrt(25)); // 5
console.log(Math.sqrt(2)); // 1.4142135623730951
// Potencia
console.log(Math.pow(3, 2)); // 9
```

Strings

Textos definidos entre comillas.

```
let oneSpace = ' ' // string vacio
let city = 'Benidorm'
let cita= "The saying,'Seeing is Believing' is not correct in 2021."
let citaPlantilla= `The saying,'Seeing is Believing' is not correct in 2021.`
```

Concatenación de strings

```
let pais = 'España';
let edad = 25;
let informacionPersona = 'Tengo ' + edad + ' años y vivo en ' + pais;
console.log(informacionPersona); // Tengo 25 años y vivo en España
```

Strings multilinea, utlizamos (\)

```
const parrafo = "Me llamo John Doe. Vivo en Benidorm, España.\
Soy profesor y me encanta enseñar. Enseño HTML, CSS, JavaScript, \
a cualquiera que esté interesado en aprender. \
Espero que tú también lo estés disfrutando.";
```

Strings

Secuencias de escape en cadenas de texto

```
console.log('Espero que todos lo estén disfrutando.\n¿Y tú?'); // Salto de línea
console.log('Días\tTemas\tEjercicios'); // Tabulación (equivale a 8 espacios)
console.log('Día 1\t3\t5'); // Tabulación
console.log('Este es un símbolo de barra invertida (\\)'); // Para escribir una barra invertida
console.log('En todos los lenguajes de programación comienza con \"¡Hola, Mundo!\"'); // Comillas dobles escapadas
console.log("En todos los lenguajes de programación comienza con \'¡Hola, Mundo!\'"); // Comillas simples escapadas
```

```
Espero que todos lo estén disfrutando.
¿Y tú?
Días Temas Ejercicios
Día 1 3 5
Este es un símbolo de barra invertida (\)
En todos los lenguajes de programación comienza con "¡Hola, Mundo!"
En todos los lenguajes de programación comienza con '¡Hola, Mundo!'
```

Strings

Métodos Comunes de Cadenas en JavaScript

Todos estos métodos no modifican el valor de la variable a menos que la reasignes.

```
let s1 = "Esto es un string";
// Obtener la longitud del string
console.log(s1.length); // Imprime 17
// Obtener el carácter de una cierta posición del string (Empieza en 0)
console.log(s1.charAt(0)); // Imprime "E"
// Obtiene el índice de la primera ocurrencia
console.log(s1.indexOf("s")); // Imprime 1
// Obtiene el índice de su última ocurrencia
console.log(s1.lastIndexOf("s")); // Imprime 11
// Devuelve un array con todas las coincidencias en de una expresión regular
console.log(s1.match(/.s/g)); // Imprime ["Es", "es", " s"]
// Obtiene la posición de la primera ocurrencia de una expresión regular
console.log(s1.search(/[aeiou]/)); // Imprime 3
// Reemplaza la coincidencia de una expresión regular (o string) con un string (/g opcionalmente reemplaza todas)
```

Strings

```
let s1 = "Esto es un string";
console.log(s1.replace(/i/g, "e")); // Imprime "Esto es un streng"
// Devuelve un substring (posición inicial: incluida, posición final: no incluida)
console.log(s1.slice(5, 7)); // Imprime "es"
// Igual que slice
console.log(s1.substring(5, 7)); // Imprime "es"
// Como substring pero con una diferencia (posición inicial, número de caracteres desde la posición inicial)
console.log(s1.substr(5, 7)); // Imprime "es un s"
// Transforma en minúsculas, toLowerCase no funciona con caracteres especiales (ñ, á, é, ...)
console.log(s1.toLocaleLowerCase()); // Imprime "esto es un string"
// Transforma a mayúsculas
console.log(s1.toLocaleUpperCase()); // Imprime "ESTO ES UN STRING"
// Devuelve un string eliminando espacios, tabulaciones y saltos de línea del principio y final
console.log(" String con espacios ".trim()); // Imprime "String con espacios"
```

Strings

Conversión de tipo explicita:

Puedes convertir un dato en number usando la función **Number(value).**
- Si el valor no es un número la función devolverá NaN
- Puedes también añadir el prefijo '+' antes de la variable para conseguir el mismo resultado.

```
let s1 = "32";
let s2 = "14";
console.log(Number(s1) + Number(s2)); // Imprime 46
console.log(+s1 + +s2); // Imprime 46
```

La conversión de un dato a booleano se hace usando la función **Boolean(value).**
- **Puedes añadir !! (doble negación), antes del valor para forza la conversión.**
- Estos valores equivalen a false:
  - string vacio (""), null, undefined, 0.
- Cualquier otro valor debería devolver true.

```
let v = null;
let s = "Hello";
console.log(Boolean(v)); // Imprime false
console.log(!!s); // Imprime true
```

Strings

Operadores. Suma ‘+’

- Este operador puede usarse para sumar números o concatenar cadenas.
- Pero, ¿Qué ocurre si intentamos sumar un número con un string, o algo que no sea un número o string?
- Veamos los ejemplos:

```
console.log(4 + 6); // Imprime 10
console.log("Hello " + "world!"); // Imprime "Hello world!"
console.log("23" + 12); // Imprime "2312"
console.log("42" + true); // Imprime "42true"
console.log("42" + undefined); // Imprime "42undefined"
console.log("42" + null); // Imprime "42null"
console.log(42 + "hello"); // Imprime "42hello"
console.log(42 + true); // Imprime 43 (true => 1)
console.log(42 + false); // Imprime 42 (false => 0)
console.log(42 + undefined); // Imprime NaN (undefined no puede ser convertido a number)
console.log(42 + null); // Imprime 42 (null => 0)
console.log(13 + 10 + "12"); // Imprime "2312" (13 + 10 = 23, 23 + "12" = "2312")
```

Strings

Conversiones implícitas en la suma

- Cuando hay un string, siempre se realizará una concatenación, por tanto, si el otro valor no es un string se intentará transformar en un string.
- Si no hay strings, y algún valor no es un número, lo intentará convertir a número e intentará hacer una suma.
- Si la conversión del valor a número falla, devolverá NaN (Not a Number).

Operadores aritméticos

Operadores aritméticos son: resta (-), multiplicación (*), división (/) y módulo (%).

Estos operadores operan siempre con números, por tanto, cualquier operando que no sea un número debe ser convertido a número.

```
console.log(4 * 6); // Imprime 24
console.log("Hello " * "world!"); // Imprime NaN
console.log("24" / 12); // Imprime 2 (24 / 12)
console.log("42" * true); // Imprime 42 (42 * 1)
console.log("42" * false); // Imprime 0 (42 * 0)
console.log("42" * undefined); // Imprime NaN
console.log("42" - null); // Imprime 42 (42 - 0)
console.log(12 * "hello"); // Imprime NaN
```

Operadores incremento y decremento

En JavaScript podemos preincrementar (++variable), postincrementar (variable++), predecrementar (--variable) y postdecrementar (variable--).

```
let a = 1;
let b = 5;
console.log(a++); // Imprime 1 y incrementa a (2)
console.log(++a); // Incrementa a (3), e imprime 3
console.log(++a + ++b); // Incrementa a (4) y b (6). Suma (4+6), e imprime 10
console.log(a-- + --b); // Decrementa b (5). Suma (4+5). Imprime 9. Decrementa a (3)
```

Operador cambio de signo

Podemos usar los signos – y + delante de un número para cambiar o mantener su signo.

Si aplicamos estos operadores con un dato que no es un número, este será convertido a número primero.

Por eso, es una buena opción usar **+value para convertir a número,** lo cual equivale a usar Number(value).

```
let a = "12";
let b = "13";
let c = true;
console.log(a + b); // Imprime "1213"
console.log(+a + +b); // Imprime 25 (12 + 13)
console.log(+b + +c); // Imprime 14 (13 + 1). True -> 1
```

Operadores relacionales

- El operador de comparación, compara dos valores y devuelve un booleano (true o false)
- Estos operadores son prácticamente los mismos que en la mayoría de lenguajes de programación, a excepción de algunos, que veremos a continuación.
- Podemos usar == o === para comparar la igualdad (o lo contrario !=, !==).
- La principal diferencia es que el primero, no tiene en cuenta los tipos de datos que están siendo comparados, compara si los valores son equivalentes.
- Cuando usamos ===, los valores además deben ser del mismo tipo.
- Si el tipo de dato o el valor son diferentes devolverá falso.
- Devolverá true cuando ambos valores son idénticos y del mismo tipo.

Operadores relacionales

```
console.log(3 == "3"); // true
console.log(3 === "3"); // false
console.log(3 != "3"); // false
console.log(3 !== "3"); // true
// Equivalente a falso (todo lo demás es equivalente a cierto)
console.log("" == false); // true
console.log(false == null); // false (null no es equivalente a cualquier boolean).
console.log(false == undefined); // false (undefined no es equivalente a
cualquier boolean).
console.log(null == undefined); // true (regla especial de JavaScript)
console.log(0 == false); // true
console.log({} >= false); // Object vacío -> false
console.log([] >= false); // Array vacío -> true
```

Otros operadores relacionales

Otros operadores relaciones para números o strings son: menor que (<), mayor que (>), menor o igual que (<=) y mayor o igual que (>=)

Cuando comparamos un string con estos operadores, se va comparando carácter a carácter y se compara su posición en la codificación Unicode para determinar si es menor (situado antes) o mayor (situado después).

A diferencia del operador de suma (+), cuando uno de los dos operandos es un número, el otro será transformado en número para comparar.

Para poder comparar como string, ambos operandos deben ser string.

```
console.log(6 >= 6); // true
console.log(3 < "5"); // true ("5" → 5)
console.log("adiós" < "bye"); // true
console.log("Bye" > "Adiós"); // true
console.log("Bye" > "adiós"); // false. Las letras mayúsculas van siempre antes
console.log("ad" < "adiós"); // true
```

Operadores booleanos

Los operadores booleanos son negación (!), y (&&), o (||).

Estos operadores, normalmente, son usados de forma combinada con los operadores relacionales formando una condición más compleja, la cual devuelve true o false.

```
console.log(!true); // Imprime false
console.log(!(5 < 3)); // Imprime true (!false)
console.log(4 < 5 && 4 < 2); // Imprime false (ambas condiciones deben ser ciertas)
console.log(4 < 5 || 4 < 2); // Imprime true (en cuanto una condición sea cierta, devuelve cierta y deja de comparar)
```

¿Preguntas?