

DWEC – Javascript Web Cliente.

JavaScript 01 – Sintaxis (III)

JavaScript 01 – Sintaxis (III)	1
Tipos de datos básicos	1
Casting de variables	1
Number	2
Veamos algunos ejemplos de código y la salida en consola:	2
Otras funciones útiles son:	3
String	4
Template literals	5
Tratamiento de una cadena como un array	5
Boolean	5

Tipos de datos básicos

Para saber de qué tipo es el valor de una variable tenemos el operador **typeof**. Ej.:

- `typeof 3` devuelve *number*
- `typeof 'Hola'` devuelve *string*

En Javascript hay 2 valores especiales:

- **undefined**: es lo que vale una variable a la que no se ha asignado ningún valor.
- **null**: es un tipo de valor especial que podemos asignar a una variable. Es como un objeto vacío (`typeof null` devuelve *object*)

También hay otros valores especiales relacionados con operaciones numéricas (o con números):

- **NaN** (*Not a Number*): indica que el resultado de la operación no puede ser convertido a un número (ej. `'Hola'*2`, aunque `'2'*2` daría 4 ya que se convierte la cadena '2' al número 2)
- **Infinity** y **-Infinity**: indica que el resultado es demasiado grande o demasiado pequeño (ej. `1/0` o `-1/0`)

Casting de variables

Como hemos dicho, las variables pueden contener cualquier tipo de valor.

En las operaciones, Javascript realiza **automáticamente** las conversiones necesarias para, si es posible, realizar la operación.

Ejemplos:

- `'4' / 2` devuelve 2 (convierte '4' en 4 y realiza la operación)
- `'23' - null` devuelve 23 (hace 23 - 0)

- '23' - undefined devuelve *NaN* (no puede convertir undefined a nada así que no puede hacer la operación)
- '23' * true devuelve 23 (23 * 1)
- '23' * 'Hello' devuelve *NaN* (no puede convertir 'Hello')
- 23 + 'Hello' devuelve '23Hello' (+ es el operador de concatenación así que convierte 23 a '23' y los concatena)
- 23 + '23' devuelve 2323 (OJO, convierte 23 a '23', no al revés)

Ten en cuenta que en Javascript todo son objetos, por lo que todo tiene métodos y propiedades. Veamos brevemente los tipos de datos básicos.

EJERCICIO: Prueba en la consola las operaciones anteriores y alguna más con la que tengas dudas de qué devolverá.

Number

Sólo hay 1 tipo de números, no existen enteros y decimales. El tipo de dato para cualquier número es **number**. El carácter para la coma decimal es el . (como en inglés, así que 23,12 debemos escribirlo como 23.12).

Tenemos los operadores aritméticos +, -, *, / y % y los unarios ++ y --

Existen los valores especiales **Infinity** y **-Infinity** (23/0 no produce un error sino que devuelve *Infinity*).

Podemos usar los operadores aritméticos junto al operador de asignación = (+=, -=, *=, /= y %=).

Algunos métodos útiles de los números son:

- **.toFixed(num)**: redondea el número a los decimales indicados. Ej. 23.2376.toFixed(2) devuelve 23.24
- **.toLocaleString()**: devuelve el número convertido al formato local. Ej. 23.76.toLocaleString() devuelve '23,76' (Los navegadores en español convierten el punto decimal en coma y ponen el punto separador de miles)

Podemos forzar la conversión a número con la función **Number(valor)**. Ejemplo Number('23.12') devuelve 23.12

Veamos algunos ejemplos de código y la salida en consola:

```
let num=23.3333333333;
console.log(typeof(num));
console.log(num);
```

number
23.3333333333

```
num= 45212.444444
console.log(typeof(num));
console.log(num.toLocaleString());
```

number
45.212,444

```
let num2 = 36.2222;
console.log(typeof(num2));
num2=num2.toLocaleString();
console.log(typeof(num2));
console.log(num2);
```

number
string
36,222

solo 3 decimales

```

num2=888; //ojo
console.log(num2);
num2=num2.toLocaleString();
console.log(num2);
console.log(typeof(num2));
num2 = Number(num2);
console.log(typeof(num2));
console.log(num2);

```

```

num2=888.8888; //ojo
console.log(num2);
num2=num2.toLocaleString();
console.log(num2);
console.log(typeof(num2));
num2 = Number(num2);
console.log(typeof(num2));
console.log(num2);

```

En el caso anterior, no es numérico porque después de convertir `toLocaleString()`, no puede volver a numérico.

Si se convierte con el método `toString()`, sí que puede retornar a numérico, como vemos en el siguiente ejemplo:

```

num2=888.8888; //ojo
console.log(num2);
num2=num2.toString();
console.log(num2);
console.log(typeof(num2));
num2 = Number(num2);
console.log(typeof(num2));
console.log(num2);

```

Otras funciones útiles son:

- **isNaN(valor)**: nos dice si el valor pasado es un número (false), o no (true)
- **isFinite(valor)**: devuelve *true* si el valor es finito (no es *Infinity* ni *-Infinity*).
- **parseInt(valor)**: convierte el valor pasado a un número entero. Siempre que comience por un número la conversión se podrá hacer. Ej.:

```

parseInt(3.65)      // Devuelve 3
parseInt('3.65')    // Devuelve 3
parseInt('3 manzanas') // Devuelve 3, Number devolvería NaN

```

- **parseFloat(valor)**: como la anterior, pero conserva los decimales

OJO: al sumar números decimales (*floats*) podemos tener problemas:

```

console.log(0.1 + 0.2) // imprime 0.30000000000000004

```

Para evitarlo:

- Redondead los resultados.
- También se puede hacer una operación similar a $(0.1 \cdot 10 + 0.2 \cdot 10) / 10$.

<code>console.log((0.1 + 0.2));</code>	0.30000000000000004
<code>console.log((0.1 + 0.2).toFixed(1));</code>	0.3
<code>console.log((0.1*10 + 0.2*10) / 10);</code>	0.3

EJERCICIO: Modifica la función de calcular la nota media para que devuelva la media con 1 decimal

EJERCICIO: Modifica la función que devuelve el cubo de un número para que compruebe si el parámetro pasado es un número entero. Si no es un entero o no es un número mostrará un alert indicando cuál es el problema y devolverá false.

String

Las cadenas de texto van entre comillas simples o dobles, es indiferente. Podemos escapar un carácter con \ para poder usarlo dentro de la cadena. (Ejemplo: 'Hola \'Mundo\'' devuelve *Hola 'Mundo'*).

<code>console.log("hola \'majo\'");</code>	hola "majo"
<code>console.log('Hola "majo"');</code>	Hola "majo"

Para forzar la conversión a cadena se usa la función **String(valor)** (ej. String(23) devuelve '23')

El operador de concatenación de cadenas es +. Ojo porque si pedimos un dato con *prompt* siempre devuelve una cadena así que si le pedimos la edad al usuario (por ejemplo 20) y se sumamos 10 tendremos 2010 ('20'+10).

Algunos métodos y propiedades de las cadenas son:

- **.length**: devuelve la longitud de una cadena. Ej.: 'Hola mundo'.length devuelve 10
- **.charAt(posición)**: 'Hola mundo'.charAt(0) devuelve 'H'
- **.indexOf(carácter)**: 'Hola mundo'.indexOf('o') devuelve 1. Si no se encuentra devuelve -1
- **.lastIndexOf(carácter)**: 'Hola mundo'.lastIndexOf('o') devuelve 9
- **.substring(desde, hasta)**: 'Hola mundo'.substring(2,4) devuelve 'la'
- **.substr(desde, num caracteres)**: 'Hola mundo'.substr(2,4) devuelve 'la m'
- **.replace(busco, reemplaza)**: 'Hola mundo'.replace('Hola', 'Adiós') devuelve 'Adiós mundo'
- **.toLocaleLowerCase()**: 'Hola mundo'.toLocaleLowerCase() devuelve 'hola mundo'
- **.toLocaleUpperCase()**: 'Hola mundo'.toLocaleUpperCase() devuelve 'HOLA MUNDO'
- **.localeCompare(cadena)**: devuelve -1 si la cadena a que se aplica el método es anterior alfabéticamente a 'cadena', 1 si es posterior y 0 si ambas son iguales. Tiene en cuenta caracteres locales como acentos ñ, ç, etc
- **.trim(cadena)**: ' Hola mundo '.trim() devuelve 'Hola mundo'
- **.startsWith(cadena)**: 'Hola mundo'.startsWith('Hol') devuelve *true*
- **.endsWith(cadena)**: 'Hola mundo'.endsWith('Hol') devuelve *false*
- **.includes(cadena)**: 'Hola mundo'.includes('mun') devuelve *true*
- **.repeat(veces)**: 'Hola mundo'.repeat(3) devuelve 'Hola mundoHola mundoHola mundo'
- **.split(separador)**: 'Hola mundo'.split(' ') devuelve el array ['Hola', 'mundo']. 'Hola mundo'.split('') devuelve el array ['H', 'o', 'l', 'a', ' ', 'm', 'u', 'n', 'd', 'o']

Podemos probar los diferentes métodos en la página de [w3schools](https://www.w3schools.com/js/js_string_methods.asp).

Template literals

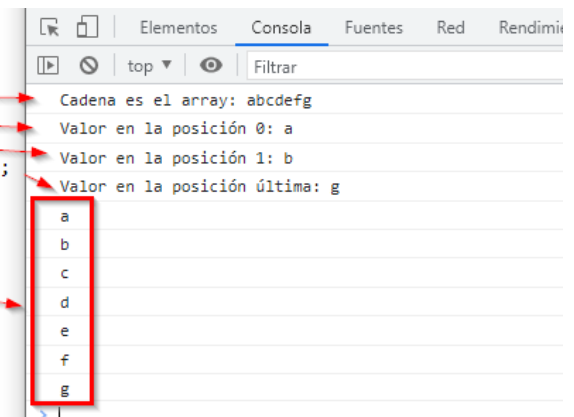
Desde ES2015 también podemos poner una cadena entre ``` (acento grave) y en ese caso podemos poner dentro variables y expresiones que serán evaluadas al ponerlas dentro de `${}`. También se respetan los saltos de línea, tabuladores, etc que haya dentro. Ejemplo:

```
let edad=25;
console.log(`El usuario tiene:
    ${edad} años`);
```

```
El usuario tiene:
  25 años
```

Tratamiento de una cadena como un array

```
let cadena = "abcdefg";
console.log(`Cadena es el array: ${cadena}`);
console.log(`Valor en la posición 0: ${cadena[0]}`);
console.log(`Valor en la posición 1: ${cadena[1]}`);
console.log(`Valor en la posición última: ${cadena[cadena.length - 1]}`);
for (const letra of cadena) {
  console.log(letra);
}
```



EJERCICIO: Haz una función a la que se le pasa un DNI (ej. 12345678w o 87654321T) y devolverá si es correcto o no. La letra que debe corresponder a un DNI correcto se obtiene dividiendo la parte numérica entre 23 y cogiendo de la cadena 'TRWAGMYFPDXBNJZSQVHLCKE' la letra correspondiente al resto de la división. Por ejemplo, si el resto es 0 la letra será la T y si es 4 será la G. Prueba la función en la consola con tu DNI

Boolean

Los valores booleanos son **true** y **false**. Para convertir algo a booleano se usar **Boolean(valor)** aunque también puede hacerse con la doble negación (**!!**). Cualquier valor se evaluará a *true* excepto 0, NaN, null, undefined o una cadena vacía (""), que se evaluarán a *false*.

```
console.log(Boolean("pepe"));
console.log(!("pepe"));
console.log(!("pepe"));
console.log(!null);
console.log(!null);
console.log(!" ");
console.log(!"");
console.log(!undefined);
```

```
true
true
false
true
false
true
false
false
```

Los operadores lógicos son **!** (negación), **&&** (and), **||** (or).

Para comparar valores tenemos **==** y **===**. La triple igualdad devuelve *true* si son igual valor y del mismo tipo. Como Javascript hace conversiones de tipos automáticas conviene usar la **===** para evitar cosas como:

- '3' == 3 true
- 3 == 3.0 true
- 0 == false true
- '' == false true
- ' ' == false true
- [] == false true
- null == false false
- undefined == false false
- undefined == null true

También tenemos 2 operadores lógicos para *diferente*: `!=` y `!==` que se comportan como hemos dicho antes.

Los operadores relacionales son `>`, `>=`, `<`, `<=`. Cuando se compara un número y una cadena ésta se convierte a número y no al revés (`23 > '5'` devuelve *true*, aunque `'23' > '5'` devuelve *false*)

<code>console.log(23 > 5);</code>	<code>true</code>
<code>console.log('23' > '5');</code>	<code>false</code>
<code>console.log('23' > 5);</code>	<code>true</code>
<code>console.log(23 > '5');</code>	<code>true</code>