

# Fundamentos de Programación con JavaScript

## Tipos de Datos

04

JS



# Datos básicos

**JS**

MÉTODO: todo aquello que un objeto puede hacer (acción)

PROPIEDAD: es una característica (atributo) que un objeto tiene

Se utilizan usando la nomenclatura del punto

- Las propiedades no usan paréntesis
- Los métodos usan paréntesis y necesitan o no parámetros

A yellow square logo with the letters 'JS' in a bold, black, sans-serif font, representing JavaScript.

# String

En cualquier lenguaje de programación, un **string** es una secuencia de caracteres usado para representar texto.

En JavaScript, un **string** es uno de los valores primitivos

El objeto `String` se utiliza para representar y manipular una secuencia de caracteres.

Interactuar con cadenas:

- **Concatenar**
- **Interpolar**

Las cadenas son útiles para almacenar datos que se pueden representar en forma de texto. Algunas de las operaciones más utilizadas en cadenas son verificar su `length`, para construirlas y concatenarlas usando operadores de cadena `+` y `+=`, verificando la existencia o ubicación de subcadenas con `indexOf()` o extraer subcadenas con el método `substring()`.

Los cadena literales se pueden especificar usando comillas simples o dobles, que se tratan de manera idéntica, o usando el carácter de comilla invertida ```. Esta última forma especifica una **Plantilla literal**: con esta forma puedes interpolar expresiones.

*`Template  
Strings`*

# String [ MÉTODOS y PROPIEDAD ]

**MÉTODOS** Todos los métodos devuelven una cadena nueva, la cadena original no se modificada.

```
// console.log(cadena.length);  
// console.log(cadena.toUpperCase());
```

`toUpperCase()` --> Devuelve la cadena a mayúsculas

`toLowerCase()` --> Devuelve la cadena a minúsculas

`indexOf(cadena)` --> Devuelve la posición en la que se encuentra la cadena, si no lo encuentra devuelve -1

`replace(valor a buscar, valor nuevo)` --> Reemplaza el fragmento de la cadena que le digamos y pone el valor nuevo

`substring(inicio [,fin])` --> Extrae los caracteres desde inicio hasta fin (el final no se incluye)  
Si no se incluye el fin extrae hasta el final.

`slice(inicio [,fin])` --> Igual que substring pero admite valores negativos,  
si ponemos valores negativos empezará desde atrás.  
Si no se incluye el final extrae hasta el final  
(2,-4) --> Empieza a contar en el tercer caracter y los 4 últimos no los considera

`trim()` --> Elimina los espacios al inicio y al final de la cadena

# String [ MÉTODOS ]

--- ES6 ---

`startsWith(valor [,inicio])` --> Sirve para saber si la cadena empieza con ese valor. Devuelve true o false

`endsWith(valor [,longitud])` --> Sirve para saber si la cadena termina con ese valor. Devuelve true o false

`includes(valor[,inicio])` --> Igual que `indexOf` pero devuelve true o false

`repeat(valor)` --> Repite el string el número de veces que le indiquemos.

Template Strings

## PROPIEDADES

`length` --> Devuelve la longitud de la cadena



# Number

Es un objeto primitivo envolvente que permite representar y manipular valores numéricos de cualquier tipo: naturales, enteros y reales.

En JavaScript, los números se implementan en [Formato en coma flotante de doble precisión](#) de 64 bits IEEE 754 (es decir, un número entre  $\pm 2^{-1022}$  y  $\pm 2^{+1023}$ , o aproximadamente  $\pm 10^{-308}$  a  $\pm 10^{+308}$ , con una precisión numérica de 53 bits). Los valores enteros hasta  $\pm 2^{53} - 1$  se pueden representar con exactitud.

Además de poder representar números de punto flotante, el tipo `number` tiene tres valores simbólicos: `+Infinity`, `-Infinity` y `NaN` (*Not-a-Number*, no es un número).

Una adición más reciente a JavaScript es el `BigInt` que te permite representar números enteros que pueden ser muy grandes. Sin embargo, existen advertencias para usar `BigInt`.

# Number

Propiedad	Descripción
<a href="#">Number.MAX_VALUE</a>	El número representable más grande ( $\pm 1.7976931348623157e+308$ )
<a href="#">Number.MIN_VALUE</a>	El número representable más pequeño ( $\pm 5e-324$ )
<a href="#">Number.NaN</a>	Valor especial not a number ("no es un número")
<a href="#">Number.NEGATIVE_INFINITY</a>	Valor infinito negativo especial; devuelto por desbordamiento
<a href="#">Number.POSITIVE_INFINITY</a>	Valor infinito positivo especial; devuelto por desbordamiento
<a href="#">Number.MIN_SAFE_INTEGER</a>	Número entero seguro mínimo en JavaScript ( $-2^{53} + 1$ o $-9007199254740991$ )
<a href="#">Number.MAX_SAFE_INTEGER</a>	Máximo número entero seguro en JavaScript ( $+2^{53} - 1$ o $+9007199254740991$ )



# Number

Método	Descripción
<a href="#">Number.parseFloat()</a>	Analiza un argumento de cadena y devuelve un número de punto flotante. Igual que la función <a href="#">parseFloat()</a> global.
<a href="#">Number.parseInt()</a>	Analiza un argumento de cadena y devuelve un número entero de la base o raíz especificada. Igual que la función <a href="#">parseInt()</a> global.
<a href="#">Number.isFinite()</a>	Determina si el valor pasado es un número finito.
<a href="#">Number.isInteger()</a>	Determina si el valor pasado es un número entero.
<a href="#">Number.isNaN()</a>	Determina si el valor pasado es <b>NaN</b> . Versión más robusta del <a href="#">isNaN()</a> global
<a href="#">Number.isSafeInteger()</a>	Determina si el valor proporcionado es un número que es un <i>entero seguro</i> .

# Number

Método	Descripción
<a href="#"><u>toExponential()</u></a>	Devuelve una cadena que representa el número en notación exponencial.
<a href="#"><u>toFixed()</u></a>	Devuelve una cadena que representa el número en notación de punto fijo.
<a href="#"><u>toPrecision()</u></a>	Devuelve una cadena que representa el número con una precisión especificada en notación de punto fijo.

**parseFloat():** convierte una cadena y devuelve un número de punto flotante.

**parseInt():** Convierte una cadena y devuelve un entero.

# Math

El objeto integrado **Math** tiene propiedades y métodos para constantes y funciones matemáticas.

Método	Descripción
<a href="#">abs()</a>	Valor absoluto
<a href="#">sin()</a> , <a href="#">cos()</a> , <a href="#">tan()</a>	Funciones trigonométricas estándar; con el argumento en radianes.
<a href="#">pow()</a> , <a href="#">exp()</a> , <a href="#">expm1()</a> , <a href="#">log10()</a> , <a href="#">log2()</a>	Funciones exponenciales y logarítmicas.
<a href="#">floor()</a> , <a href="#">ceil()</a>	Devuelve el entero más grande/más pequeño menor/mayor o igual que un argumento.
<a href="#">min()</a> , <a href="#">max()</a>	Devuelven el valor mínimo o máximo (respectivamente) de una lista de números
<a href="#">random()</a>	Devuelve un número aleatorio entre 0 y 1.
<a href="#">round()</a> , <a href="#">fround()</a> , <a href="#">trunc()</a> ,	Funciones de redondeo y truncamiento.
<a href="#">sqrt()</a> , <a href="#">cbrt()</a>	Raíz cuadrada, raíz cúbica.
<a href="#">sign()</a>	El signo de un número, que indica si el número es positivo, negativo o cero.

# Especiales

## null

El valor `null` es un literal de Javascript que representa intencionalmente un valor nulo o "vacío".

Es uno de los [valores primitivos](#) de Javascript.

## undefined

Un valor [primitivo](#) automáticamente asignado a las **variables** que solo han sido declarados o a los [argumentos](#) formales para los cuales no existe argumentos reales.

## NaN

La propiedad global `NaN` es un valor que representa Not-A-Number.

## Symbol

En [JavaScript](#) , Symbol es un [valor primitivo](#) . Un valor de **símbolo** representa un identificador único.



# Boolean (booleano)

## Tipo de dato lógico

El tipo de dato **lógico** o **booleano** es en computación aquel que puede representar valores de lógica binaria, esto es 2 valores, que normalmente representan *falso* o *verdadero*. Se utiliza normalmente en la programación,

El valor pasado como primer parámetro se convierte en un valor booleano, si es necesario. Si el valor se omite o es `0`, `-0`, `null`, `false`, `NaN`, `undefined`, o la cadena vacía (`""`), el objeto tiene un valor inicial de `false`. Todos los demás valores, incluido cualquier objeto, un arreglo vacío (`[]`) o la cadena `"false"`, crean un objeto con un valor inicial de `true`.

valores del Boolean primitivo, `true` y `false`

**Cualquier** objeto cuyo valor no sea `undefined` o `null`, incluido un objeto `Boolean` cuyo valor es `false`, se evalúa como `true` cuando se pasa a una declaración condicional. Por ejemplo, la condición en la siguiente declaración `if` se evalúa como `true`.



# Fundamentos de Programación con JavaScript



# JS