Tipos de datos en Javascript

JavaScript es un lenguaje de tipado débil o dinámico. Esto significa que no es necesario declarar el tipo de variable antes de usarla.

En los lenguajes de tipado dinámico como JavaScript o Python la comprobación de los tipos se realiza durante la ejecución del programa en vez de durante la compilación.

En los lenguajes de tipado estático o fuertemente tipados como C#, Go, Java o C++ la comprobación de los tipos se realiza durante la compilación, y no durante la ejecución.

Declaración de variables en JS

En los lenguajes con tipos dinámicos no especificamos el tipo de valor al declarar las variables y podemos utilizar el mismo nombre de variable para contener distintos tipos de datos:

```
var foo = 42; // foo es ahora un Number
```

```
var foo = "bar"; // foo es ahora un String
```

```
var foo = true; // foo es ahora un Boolean
```

En cambio en los lenguajes fuertemente tipados como en C#, especificamos el tipo de valor al declarar cada variable:

```
int edad = 38;
```

```
string nombre = "Marcel";
```

```
bool b = false;
```

En Javascript existen 7 tipos de datos

- String Cadenas de texto.
- Number Valores numéricos.
- Boolean Representa una entidad lógica y puede tener dos valores: true y false.
- null Es un valor asignado tiene el valor de “no valor”.
- undefined Una variable a la que no se le ha asignado ningún valor tiene el valor undefined.
- Symbol Nuevo en ECMAScript 2015.
- Object Un valor en memoria al que podemos acceder por un identificador.

Estos tipos se dividen en dos grupos, **Primitivos** y **de Objeto**.

Tipos Primitivos

Los valores primitivos son inmutables, no pueden ser cambiados.

String, Number, Boolean, null, undefined y Symbol son tipos primitivos.

Los tipos primitivos no tienen métodos ni propiedades, aunque en los string, numbers y booleans podemos acceder a ellas grácias a los wrappers objects que veremos enseguida.

Veámos los tipos primitivos uno por uno:

String

Un string es una cadena de caracteres. A cada carácter de una cadena se le asigna una posición, empezando por el primer carácter en la posición 0, el segundo en la posición 1 y así sucesivamente.

```
"tarta de plátano vegana con masa casera";
```

Number

JavaScript solo tiene un tipo de datos numérico. No hay un tipo específico para los números enteros y de coma flotante.

```
var peso = 54;

var pesoExacto = 54.3;
```

Declaración de variables numéricas en C#, lenguaje con diferentes tipos para para diferentes tipos de números:

```
int peso = 54;

float pesoExacto = 54.30f
```

Boolean

Boolean representa una entidad lógica y puede tener dos valores: true, y false.

```
var x = false;

if (x) {

 // este código no se ejecuta
```

```
}
```

null

El tipo Null tiene el valor: null.

undefined

Una variable a la cual no se le haya asignado valor tiene el valor undefined.

```
var saludo;

console.log(typeof saludo); // "undefined"
```

Symbol

Symbol es un tipo de datos cuyos valores son **únicos** e **inmutables**. Dichos valores pueden ser utilizados como identificadores (claves) de las propiedades de los objetos. Cada valor del tipo Symbol tiene asociado un valor del tipo String o Undefined que sirve únicamente como descripción del símbolo. *mdn*

Tipos de Objeto

Un Objeto es un valor en memoria al que podemos acceder por un identificador.

En JavaScript los objetos pueden ser vistos como una colección de propiedades.

Un Objeto en notación literal tiene este aspecto:

```
var persona = { nombre: "Marcel", edad: "38", ciudad: "Alaior" };
```

Podemos escribirlo en multiples líneas para más claridad

```
var persona = {

 nombre: "Marcel",

 edad: "38",

 ciudad: "Alaior",

};
```

Determinar tipos utilizando el operador typeof

El operador typeof devuelve una cadena indicando el tipo de valor de una variable.

Definamos unas variables:

```
var mascota = "Gato";

var hermanas = 3;

var tv = false;
```

Veamos el operador typeof en acción:

```
console.log(typeof mascota); // "string"

console.log(typeof hermanas); // "number"

console.log(typeof tv); // "boolean"
```

Primitive wrapper objects en JavaScript

Los números, las cadenas y los valores Booleanos son primitivos, se supone que no tienen métodos ni propiedades.

Por qué podemos hacer esto?

```
var nombre = "Marcel";

console.log(nombre.length); // devuelve 6
```

La variable nombre es un string primitivo, pero se comporta como si fuera un objeto, ¿qué está pasando?

Cuando se invoca un método o propiedad en un dato primitivo, Javascript crea una instancia de su versión en objeto o “wrapper object” por un corto periodo para poder devolver su valor y después la destruye.

Digamos que Javascript convierte los tipos primitivos en objetos entre bastidores sin que te des cuenta.

Excepto para null y undefined, todos los valores primitivos tienen lo que se conoce como primitive wrapper object, lo que crea la versión en Objeto de su equivalente primitivo.

Podemos crear objetos wrapper explícitamente utilizando sus constructores definidos:

- String para el primitivo string.
- Number para el primitivo number.
- Boolean para el primitivo Boolean.
- Symbol para el primitivo Symbol.

```
var a = "tarta de plátano"; // primitivo


var b = new String("tarta de plátano"); // objeto
```