Manejo de la sintaxis del lenguaje

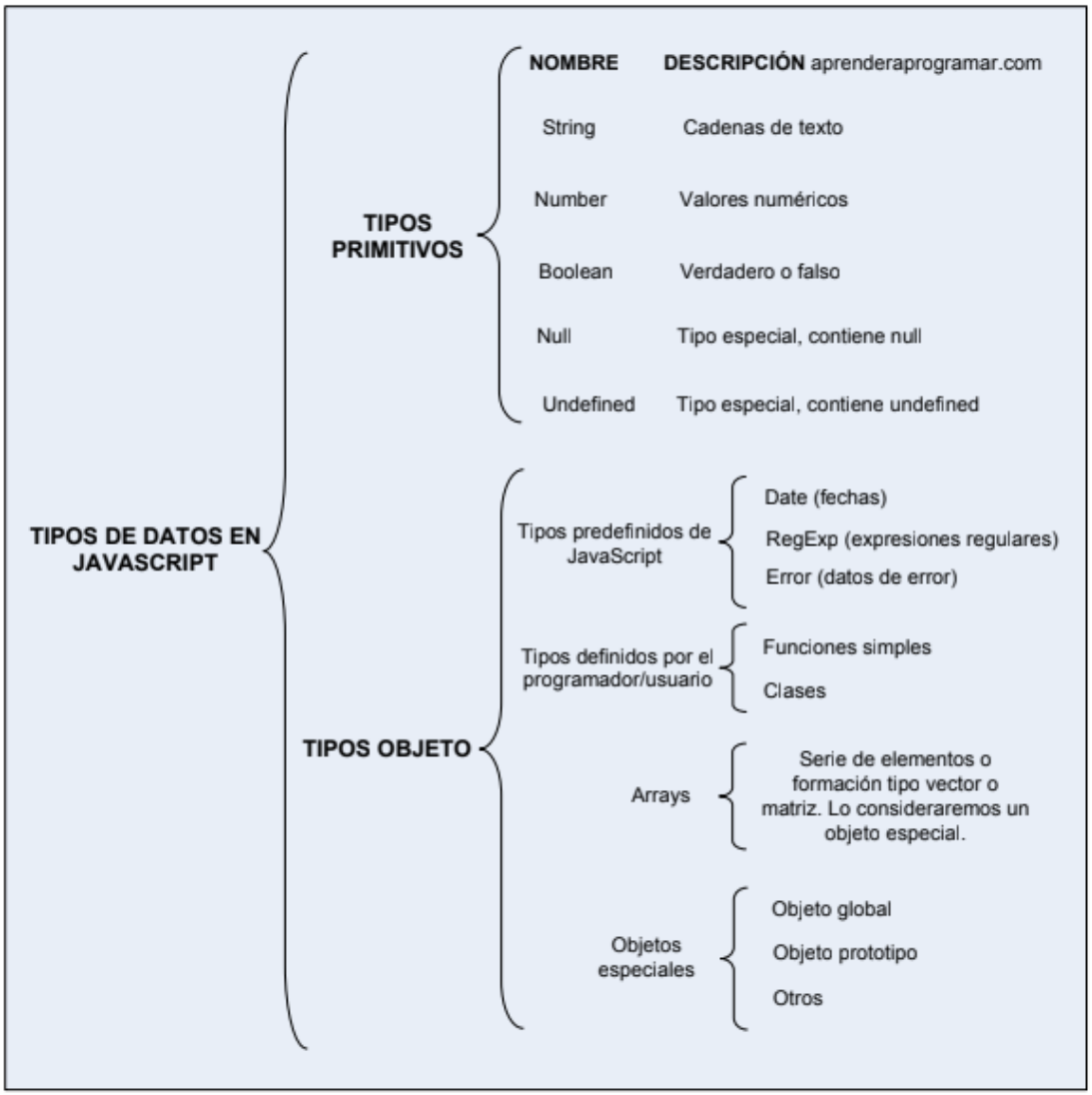
--

Tipos de datos

1. Tipos de datos

a. Introducción

El siguiente esquema es un resumen sobre tipos de datos en JavaScript.



Frente a otros lenguajes fuertemente tipados (las variables tienen un tipo declarado y no pueden
cambiar el tipo de contenido que almacenan) como Java, JavaScript es un un lenguaje débilmente tipado: las variables pueden no ser de un tipo específico y cambiar el tipo de contenido que almacenan.
JavaScript tiende a realizar conversiones automáticas siempre que le es posible.
Podemos forzar conversiones cuando sea necesario con funciones como parseInt(), Number() y String(). Las funciones de conversión Number() y String() se escriben con mayúsculas. Esto permite distinguirlos de los tipos primitivos.

En JavaScript todo lo que no sea de tipo primitivo es un objeto, incluidas las funciones.

b. Tipos primitivos

Javascript tiene seis tipos primitivos: Sin definir (undefined), Nulo (null), Lógicos (boolean), Numérico (number), Cadena (string), Símbolo (symbol). Todos los demás tipos son objetos (Object): Array, Date, Promise, etc.

i. undefined

Es cuando javascript no puede evaluar una expresión, por ejemplo, typeof(hola), devuelve undefined si la variable hola no está definida.
Cualquier variable no definida tiene como valor un tipo indefinido. Este valor no es un objeto. Existe una variable global con el nombre undefined.

ii. null

```
var testNull = null;
alert(typeof(testNull);
```

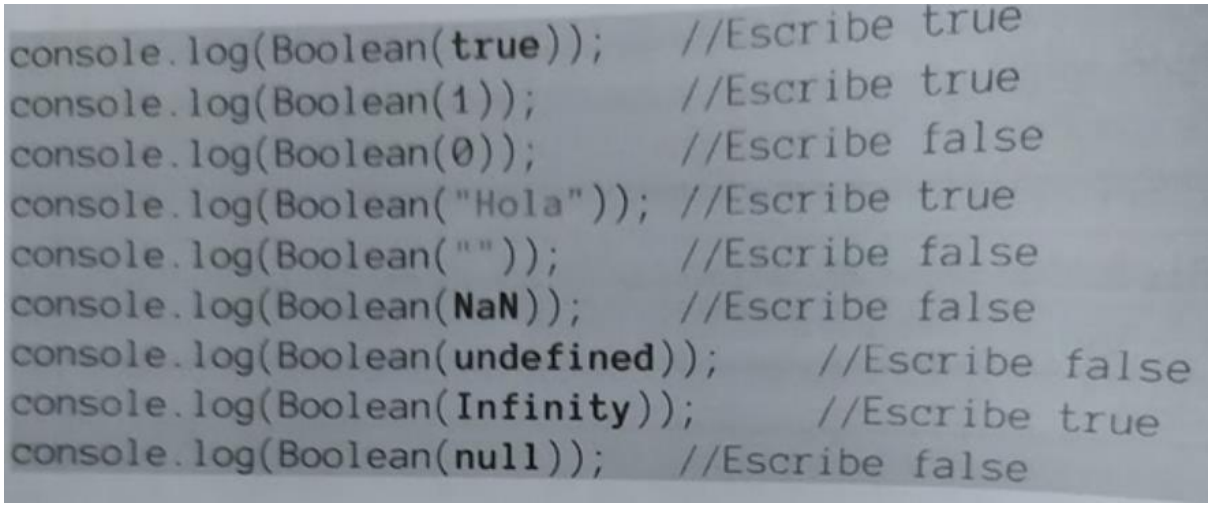
Como se puede comprobar null es un objeto según typeof, pero por otra parte no es una instancia de Object. Esta es una circunstancia peculiar que nos obliga a un tratamiento específico de null, ya que la única forma de comprobar si un valor es nulo, es compararlo con el literal null. (alert((testNull === null));)

En ES6 disponemos de un nuevo tipo primitivo, los símbolos (symbol). Estos tipos de datos son utilizados como claves en objetos y mapas de datos.
Cada vez que creamos un símbolo obtenemos un valor único, por lo que son de mucha utilidad a la hora de crear constantes con valores únicos y asegurarnos que no se va a producir confusiones por valores duplicados.
En todos los casos, la mejor forma de saber si un valor es de tipo symbol es utilizar typeof.

iii. symbol

iv. Lógico (boolean)

Los valores booleanos sólo pueden ser true o false. Es bien sabido que se producen al evaluar expresiones lógicas (condiciones de if, while...).
Es interesante el siguiente cuadro creado con el uso de la función Boolean(), a la hora de construir expresiones lógicas.



```
console.log(Boolean(true));    //Escribe true
console.log(Boolean(1));       //Escribe true
console.log(Boolean(0));       //Escribe false
console.log(Boolean("Hola")); //Escribe true
console.log(Boolean(""));      //Escribe false
console.log(Boolean(NaN));     //Escribe false
console.log(Boolean(undefined));    //Escribe false
console.log(Boolean(Infinity));     //Escribe true
console.log(Boolean(null));    //Escribe false
```

v. Numérico (number) .

En javascript solamente existe un único tipo numérico, number. Los decimales se indican mediante el punto (.) Se permite trabajar con números en base diferente a la base 10. (let binario=0b1111001; let octal=0o34106; let hexa=0xA490D). También disponemos del número especial infinity y del acrónimo NaN (Not a Number) que se produce cuando se genera un número ilegal y es muy útil para averiguar si un valor es numérico mediante el uso de la función isNaN().

vi. Cadenas (string)

Son cadenas de caracteres. Se pueden delimitar indistintamente con comillas dobles o simples. De esta manera pueden formar parte del texto comillas:

texto=”Calle o’Donell”;
texto=’El niño dijo “Hola”’;

También se permite el uso de las comillas invertidas:

console.log(`Me llamo ${nombre}`);
console.log(“Me llamo” + nombre);

Podemos incluir secuencias de escape y códigos unicode:
console.log(“\u{1F488}”);

Tabla 2.2 Secuencias de escape

Secuencia de escape	Resultado
\\	Barra invertida
\'	Comilla simple
\"	Comillas dobles
\n	Salto de línea
\t	Tabulación horizontal
\v	Tabulación vertical
\f	Salto de página
\r	Retorno de carro
\b	Retroceso

Tabla unicode
console.log("/u{1F48B}");
https://unicode-table.com/es/search/?q=1f48c

c. Los objetos

Todo aquello que no es algún tipo primitivo es un objeto. Se verán más adelante.

2. Estrategias para la identificación de tipos de datos

a. typeof

Podemos utilizar typeof para conocer el tipo del valor de una variable. Funciona perfectamente con los tipos primitivos y con las funciones, pero en el caso de los objetos ofrece muy poca información, ya que para todos ellos devuelve object sin mayor detalle.

b. instanceof

Para los objetos existe una instrucción bastante práctica que permite conocer si un objeto determinado es una instancia de una clase superior. De esta forma podemos comprobar si un determinado dato es de tipo Date, Array, etc.