Tipo de dato JS

Aunque JavaScript **no** es un **lenguaje fuertemente tipado**, esto no quiere decir que no existan tipos de datos. Vamos a ver los **tipos** que existen en JavaScript:

- **String**: Nos permite almacenar cadenas de texto
- **Number**: Nos permite almacenar valores numéricos (tanto enteros como con decimales).
- **Boolean**: Representa un valor lógico: verdadero o falso (true ó false).
- **Null**: Es un tipo especial. Su contenido es “null”.
- **Undefined**: Es una variable cuyo valor aún no se ha definido.
- **Object**: Contiene una referencia a un espacio en la memoria en el que encontramos una colección de propiedades. Lo veremos en detalle más adelante.

JavaScript es un lenguaje dinámico o de tipado débil, ya que no es obligatorio indicar el tipo de una variable al definirlo. De hecho, una misma variable puede cambiar de tipo sin problema alguno durante el proceso de ejecución de nuestro código. Podemos realizar pruebas haciendo uso de la función typeof, la cual nos indicará el tipo de una variable en cada momento.

```
let txt = typeof "texto"; // string
let num = typeof 10;      // number
let bool = typeof true;   // boolean
```

Datos dinámicos

JavaScript tiene tipos de datos dinámicos, lo que significa que **la comprobación de tipo se realiza en tiempo de ejecución en lugar de tiempo de compilación**. Con los lenguajes de tipo dinámico, se puede utilizar una variable del mismo nombre para contener diferentes tipos de datos.

Por ejemplo, la variable t , definida como una variable por la palabra clave let , se puede asignar para contener diferentes tipos de datos o puede inicializarse pero no se define.

```
// t is undefined
let t;
// t is a number
t = 6;
// t is a Boolean
t = true;
// t is undefined
t = "cadena";
```

Cada una de las variables t anteriores puede establecerse en cualquier tipo de datos disponible en JavaScript; No es necesario que se declaren explícitamente con un tipo de datos antes de que se utilicen.

String

Uno de los tipos que más vamos a utilizar, serán las cadenas de texto o string. Sencillamente son un grupo de caracteres o palabras, entre comillas (dobles o simples).

```
let texto = "Hola equipo";
let str = 'otro ejemplo';
```

Number

Otro tipo bastante intuitivo serán los números.

```
let numeroPi = 3.14;
let edad = 17;
```

Booleans

En programación, existe un tipo de dato que identifica los valores binarios 0 y 1. Una forma más intuitiva de trabajar con ello, es mediante booleanos. Un valor booleano, solo puede tomar verdadero o falso.

```
let isActive = true;
let isDisabled = false;
```

Null

Su valor es un objeto de tipo Null.

```
let error = null;
```

Undefined

Su valor no está definido.

```
let notDefined;
let defineUndefined = undefined;
```

Object

Es una estructura de datos organizada, es decir podemos agrupar datos en base a un único elemento.

```
let student = {
    name:'Alberto',
    surname: 'Rivera',
    age: 34
}
```

Array

Pese a que son de tipo Object tienen la particularidad de ser una manera de agrupar información sin tener la necesidad de relación.

```
let elements = [1, 'Alberto', true, 95, {name:'Pepe'}];
```

Mezclando tipos de datos

Aunque cada aplicación que creemos contendrá varios tipos de datos, es importante tener en cuenta que generalmente realizará operaciones dentro del mismo tipo de datos. Es decir, estaremos realizando matemáticas en números, o rebelando cadenas.

Por ejemplo, al usar el operador + con números y cadenas juntas, los números se tratarán como una cadena (por lo tanto se concatenarán), pero el orden de los tipos de datos influirá en la concatenación.

Por lo tanto, si creamos una variable que realiza la siguiente concatenación, JavaScript interpretará cada elemento a continuación como una cadena:

```
let concatena = "school" + 5 + 5;

//school55
```

Sin embargo, si entramos con números, los dos números se agregarán antes de que se interpreten como una cadena cuando el tiempo de ejecución del programa llegue a "school" , por lo que el valor devuelto será la suma de los dos números concatenados con la cadena:

```
let sumaConcatena = 5 + 3 + "school";

//8school
```

Debido a estos resultados inesperados, es probable que realice operaciones y métodos dentro de un tipo de datos en lugar de a través de ellos. **JavaScript, no devuelve errores al mezclar tipos de datos, como hacen algunos otros lenguajes de programación**.

Comprobación de tipo

El operador typeof devuelve los siguientes resultados para estas variables.

```
let myFun = new Function('5 + 2');
let shape = 'round';
let size = 1;
let foo = ['Apple', 'Mango', 'Orange'];
let today = new Date();

typeof myFun;       // devuelve "function"
typeof shape;       // devuelve "string"
typeof size;        // devuelve "number"
typeof foo;         // devuelve "object"
typeof today;       // devuelve "object"
typeof doesntExist; // devuelve "undefined"
```

```
typeof true; // devuelve "boolean"
typeof null; // devuelve "object"
```