Tipos de Datos Javascript

Una manera de clasificar los lenguajes de programacion es si son de tipado estático o dinámico.

Lenguajes con Tipado Estático `(Fuertemente Tipados)`
Cuando creamos las variables deberemos indicar el tipo de dato que van a contener.

Lenguajes con Tipado Dinámico `(Débilmente Tipados)`
Cuando creamos las variables no necesitamos indicar el tipo de datos que van a contener.

Javascript pertenece a los lenguajes de tipado dinámicos.
Es decir, si tiene tipos de datos, pero las variables no están atadas a un tipo en específico. Javascript resuelve el tipo de las variables sobre la marcha y hace conversiones de tipo mediante un mecanismo llamado coerción.
En Javascript una variable puede cambiar de tipo:

```
let variable = "Juan";    // Declaración de variable y asignación de un string
console.log(variable);

variable = 5000;          // Cabiamos su valor (su tipo) a number
console.log(variable);
```

Tipos de Datos Javascript

Javascript divide los tipos de datos en dos categorías: Primitivos y Objetos.

Primitivos

Se llaman primitivos porque solo pueden contener una cosa (cadena, número etc).
Números, cadenas de texto, booleanos, null, undefined, symbol.

Tipo	Descripción
number	Enteros y de punto flotantes
Bigint	Entero grande (ES6)
String	Cadenas de caracteres (En Javascript no hay tipo char(n))
Boolean	True y False
null	Significa vacío o inexistente *algo que esta vacio*
undefined	Significa valor no asignado *algo a lo cual no le hemos asignado nada*
Symbol	Creación de identificadores únicos para objetos (ES6)

Objetos

Cualquier *valor* que no sea ninguno de los anteriores es un objeto
por ejemplo arreglos, funciones, objetos literales para almacenar colecciones de datos y estructuras mas complejas.

Tipos de Datos Javascript

Tipo number

Javascript no distingue entre enteros y flotantes, ambos serán tipo number:

```
let a = 10;             // number (entero)
let b = 3.11416;        // number (flotante)
let c = 0xAF;           // Hexadecimal (base 16) si el número empieza con 0x
let d = 0o47;           // Octal       (base 8)  si el numero empieza con 0o
let e = 0b1010;         // Binario     (base 2)  si el numero empieza con 0b
let f = 1e6;            // Notación exponencial 1000000 (1*10^6)
let g = 2E3;            // Notación exponencial 2000    (2*10^3)
```

Tipos number que representan error:

Valor	Ocurrirá cuando...
NaN	Una operación falla o el navegador no puede interpretar un número
Infinity	El número no puede ser representado debido a su magnitud Dividir entre cero también regresa Inifnity

```
alert(1/0);         //Infinity
alert("cadena"/2);  //Nan
```

Tipos de Datos Javascript

Verificación del tipo

Para saber el tipo de dato de una variable podemos usar el operador `typeof`

```
let nom;                                          // Declaramos la variable nom
console.log("El tipo de variable es "+typeof(nom)); // Undefined=Ya esta declarada pero No esta definido su valor

nom = "Juan";                                     // Le asignamos un string
console.log("El tipo de variable es "+typeof nom);  // Su tipo es String
```

Para verificar si un valor es un número válido:

```
Number.isFinite(1/0);       // false
Number.isFinite(42);        // true
```

Para verificar si un valor es un entero:

```
let a = 10;                 // number
let b = 3.11416;            // number
console.log(a.isInteger()); // true   (ES6)
console.log(b.isInteger()); // false  (ES6)
```

Tipos de Datos Javascript Asignación por valor y por referencia

Primitivos: Asignación por valor

Cuando asignamos valores primitivos: `Number, String, Boolean, Null, Undefined`
Javascript copia el valor.

```
let a = 1;
let b = a;
console.log(a,b);   // 1 1

b = 2;
console.log(a,b);   // 1 2
```

Objetos: Asignación por referencia

Cuando asignamos valores no-primitivos:`Array, Function, Object`
Javascript copia la referencia.

```
let a = [1,2,3];
let b = a;
console.log(a,b);   // [1,2,3] [1,2,3]

b.push(4);
console.log(a,b);   // [1,2,3,4] [1,2,3,4]
```

Tipos de Datos Javascript Pasar primitivos y objetos a una función

Pasando primitivos a una función

Cuando en el argumento de una función pasamos un tipo primitivo. El valor que recibimos en la función es una copia.

```
function sumaUno(arg) {
    arg = arg + 1;
    return arg;
}
let a = 1;
let b = sumaUno(a);
console.log(a);   // 1
console.log(b);   // 2
```

Pasando objetos a una función

Cuando en el argumento de una función le pasamos un objeto y dentro de la función modificamos ese argumento.
Estamos modificando el mismo valor al que hace referencia ese argumento

```
function agregaElemento(arg) {       // La funcion recibe una referencia al objeto
    arg.push(4);                     // En la funcion estaremos modificando el mismo objeto
    return arg;
}
let a = [1,2,3];                     // Declaramos un array (los array son objetos)
let b = agregaElemento(a);           // Copiamos el array a otra variable y Se lo pasamos a la funcion
console.log(a);   // [1,2,3,4]
console.log(b);   // [1,2,3,4]
```

Tipos de Datos Javascript Coerción de Tipos

Como Javascript es un lenguaje débilmente tipado, va inferir(deducir) el tipo de manera automática.
La coerción es una característica de Javascript que hace que una variable de un tipo tenga el comportamiento de otro tipo diferente.
La coerción ocurrirá cuando los operandos de una operación son de diferente tipo.
Pero como podemos ver en el siguiente ejemplo la coerción no siempre es lógica o consistente:

```
console.log('2' + 8);  // Sumar un string y un number resultará en string "28"
console.log('2' * 8);  // Mupltiplicar un string y un number resultará en number 16
```

s

> Lo recomendable es ser lo mas específico que se pueda en los valores que usemos
> y si necesitamos cambiar un tipo, hacerlo manualmente en lugar de *confiar*
> en el mecanismo de coerción de Javascript

Tipos de Datos Javascript Conversión entre number y string

De string a number

```
Number('23');               // Si no es posible la conversion regresara NaN
parseInt('23',10);          // Usando la función parseInt(cadena,[base])
parseFloat('2.9');          // Si omitimos la base, por default será 10
```

De number a string

```
String(3);
```

> Podemos encontrar código donde se convierta de string a number y viceversa
> usando el mecanismo de coerción de Javascipt pero esto es una mala práctica:
>
> ```
> '5'* 1          // Convertirá el 5 (string) en number
> 3 + ''         // Convertirá el 3 (number) en string
> ```