VARIABLES EN JAVASCRIPT

En muchos lenguajes de programación hay unas reglas a la hora de declarar las variables, pero lo cierto es que Javascript es bastante permisivo.

Javascript se salta muchas reglas por ser un lenguaje un tanto libre a la hora de programar y uno de los casos en los que otorga un poco de libertad es a la hora de declarar las variables, ya que JavaScript, no nos obliga a declarar las variables, al contrario de lo que pasa en otros lenguajes de programación como Java, C, C# y muchos otros. Otra cosa en la que JavaScript es más permisivo, es que, al ser de tipado suave, no tenemos que poner el tipo de dato de la variable a la hora de declararla, sino que al darle un valor, ahi toma el tipo de dato. **Aunque no es obligatorio declarar variables, recomendamos siempre declararlas antes de usarlas.**

VAR

Javascript cuenta con la palabra “var” que utilizaremos cuando queramos declarar una o varias variables. Como es lógico, se utiliza esa palabra para definir la variable antes de utilizarla. **Esta variable se convertirá en una propiedad del objeto global, es decir sin importar donde se declare, todos tendrán la oportunidad de llamarla y utilizarla.**

```
var variable;
```

También se puede asignar un valor a la variable cuando se está declarando

```
var precio1 = 5;
var precio2 = 6;
var total = precio1 + precio2;
```

A la hora de escribir nuestras variables mantenemos algunas buenas practicas de Java como:

- Siempre se escriben en minúsculas
- Usamos el CamelCase cuando queremos que sean dos palabras
- Podemos usar guion bajo, para dos palabras también
- Nunca usamos la Ñ, ni tildes, ni ningún carácter especial

ÁMBITO DE VARIABLES

Antes de explicar la declaración de variables con la palabra let, tenemos que explicar que es el ámbito de variables Se le llama ámbito de las variables al lugar donde estas están disponibles. Por lo general, **cuando declaramos una variable hacemos que esté disponible en el lugar donde se ha declarado.** Esto es igual a lo que hemos visto en Java, las variables globales pueden ser accedidas en cualquier lugar y las variable locales pueden ser accedidas solo donde se declararon, por ejemplo una función.

LET

Desde ECMAScript 2015 existe la declaración let. La sintaxis es la misma que var a la hora de declarar las variables, pero en el caso de let la declaración afecta al bloque.
Bloque significa cualquier espacio acotado por unas llaves, como podría ser las sentencias que hay dentro de las llaves de un bucle for. **Al declarar una variable con la palabra clave let, dentro de un bloque, hacemos que esa variable solo pueda ser accedida dentro de ese bloque concreto, por lo que solo existe dentro del bloque que la contiene**. Lo que la haría una variable local. A diferencia de la palabra clave var, la cual define una variable global o local en una función, sin importar el ámbito del bloque.

```
function varPrueba() {
var x = 31;
if (true) {
var x = 71;              // Misma variable!
console.log(x);          // Imprime el valor 71
}
console.log(x);          // Imprime el valor 71
}
function letPrueba() {
let x = 31;
if (true) {
let x = 71;              // variable diferente
console.log(x);          // Imprime el valor 71
```

```
}
console.log(x);          // Imprime el valor 31
}
```

CONST

La sentencia const sirve para declarar una constante cuyo alcance puede ser global o local para el bloque en el que se declara. Es necesario inicializar la constante, es decir, **se debe especificar su valor en la misma sentencia en la que se declara,** lo que tiene sentido, dado que no se puede cambiar posteriormente.

```
const PI = 3.141592653589793;
PI = 3.14;              // Esto dará un error
PI = PI + 10;           // Esto también dará un error
```

TEMPLATE STRINGS

Las template strings, o cadenas de texto de plantilla, son una de las herramientas de ES6 para trabajo con cadenas de caracteres que nos pueden venir muy bien para producir un código Javascript más claro. Usarlas es por tanto una recomendación dado que facilitará el mantenimiento de los programas, gracias a que su lectura será más sencilla con un simple vistazo del ojo humano.

CONCATENACIÓN DE VARIABLES

En un programa realizado en JavaScript, y en cualquier lenguaje de programación en general, es normal crear cadenas en las que tenemos que juntar cadenas con los valores tomados desde las variables.

```
var sitioWeb = "DesarrolloWeb.com";
var mensaje = 'Bienvenido a ' + sitioWeb;
```

Eso es muy fácil de leer, pero a medida que el código se complica y en una cadena tenemos que concatenar el contenido de varias variables, el código comienza a ser más enrevesado.

```
var nombre = 'Miguel Angel';
var apellidos = 'Alvarez'
var profesion = 'desarrollador';
var perfil = ' ' + nombre + ' ' + apellidos + ' es ' + profesion;
```

Quizás estás acostumbrado a ver esto así. El código está bien y no tiene ningún problema, pero podría ser mucho más bonito si usas los template strings.

CREAR UN TEMPLATE STRING

Para crear un template string simplemente tienes que usar un carácter que se usa poco, como apertura y cierre de la cadena. **Es el símbolo del acento grave. (`)**

```
var cadena = `Esto es un template String`;
```

USOS DE LOS TEMPLATE STRINGS

Los template strings tienen varias características interesantes que, cómo decíamos, facilitan la sintaxis. Veremos a continuación algunos de ellos con código de ejemplo.

CONCATENACION DE VALORES

Creo que lo más interesante es el caso de la concatenación que genera un código poco claro hasta el momento. Echa un vistazo al código siguiente que haría lo mismo que el que hemos visto anteriormente del perfil.

```
var nombre = 'Miguel Angel';
var apellidos = 'Alvarez'
var profesion = 'desarrollador';
var perfil = `<b>${nombre} ${apellidos}</b> es ${profesion}`;
```

Como puedes comprobar, **dentro de un template string es posible colocar expresiones encerradas entre llaves y precediendo de un símbolo "$"**.

```
Algo como ${ expresion }.
```

En las expresiones podemos colocar código que queramos volcar, dentro de la cadena. **Las usamos generalmente para colocar valores de variables, pero también servirían para colocar operaciones matemáticas,** por ejemplo.

```
var suma = `45 + 832 = ${45 + 832}`;
```

O bien algo como esto:

```
var operando1 = 7;
var operando2 = 98;
var multiplicacion = `La multiplicación entre ${operando1} y ${operando2}
equivale a ${operando1 * operando2}`;
```

SALTOS DE LÍNEA DENTRO DE CADENAS

Hasta ahora, si queremos hacer una cadena con un salto de línea teníamos que usar el caracter de escape "contrabarra n".

```
var textoLargo = "esto es un texto\ncon varias líneas";
```

Con un template string tenemos la alternativa de colocar el salto de línea tal cual en el código y no producirá ningún problema.

```
var textoLargo = `esto es un texto
con varias líneas`;
```

TIPOS DE DATOS EN JAVASCRIPT

Los tipos de datos JavaScript se dividen en dos grupos: tipos primitivos y tipos objeto.

PRIMITIVOS	
Dato	**Descripción.**
Numérico	Números, ya sea enteros o reales.
String	Cadenas de texto.
Boolean	Valores lógicos como true o false.
null	Cuando un dato no existe.
undefined	Cuando no se le asigna un valor a la variable.

Los tipos objeto tienen sus propias subdivisiones

OBJETOS	
Tipo De Objeto	**Descripción**
Tipo predefinido de JavaScript	Date: fechas
Tipo predefinido de JavaScript	RegExp: expresiones regulares
Tipo predefinido de JavaScript	Error: datos de erros
Tipos definidos por el programador / usuario	Funciones Simples y Clases
Arrays	Serie de elementos o formación tipo vector o matriz. Lo consideramos un objeto especial
Objetos Especiales	Objeto Global
Objetos Especiales	Objeto prototipo
Objetos Especiales	Otros