Florida Universitària

Bloque1: Programación básica con JavaScript (JS)

Sintaxis Básica

Nombrar variables y constantes (I)

Florida Universitària

Reglas sobre los nombres de variables y constantes.

1 No podemos darle el nombre a una variable o constante, igual a una palabra reservada del lenguaje de programación

```
let let = 20; X
```

if, const, for, while, case, switch...

```
let letra = 20;
```

RECOMENDACIÓN

LOS NOMBRES DE LAS VARIABLES Y CONSTANTES, DEBEN DE TENER UN NOMBRE SIGNIFICATIVO A LO QUE ALMACENARÁN

```
nombre = 900;
```

Nombrar variables y constantes (II)

Florida Universitària

Reglas sobre los nombres de variables y constantes.

2 Puedes usar letras, números y guión bajo para definir tus variables y constantes

NO PUEDES INICIAR EL NOMBRE CON UN NÚMERO

```
let nombre1 ;   ✓        let 1nombre ;   ✗
let nombre_1 ;  ✓        let nombre@ ;   ✗
let _nombre1 ;  ✓
```

Ejemplos de variables y constantes:

```
let salario_base = 100;          let apellido1 = 'García';

const iva = 0.21;          const pi = 3,14159;
```

Nombrar variables y constantes (III)

Florida Universitària

Reglas sobre los nombres de variables y constantes.

3 No pueden contener caracteres especiales como:

@ (^ % ' # ! á é í ó ú ñ

EL SIGNO DE $ (DÓLAR) ES PERMITIDO TAMBIÉN,
PERO DEBES USARLO CON DISCRECIÓN Y SABER QUE ES LO QUE HARÁS

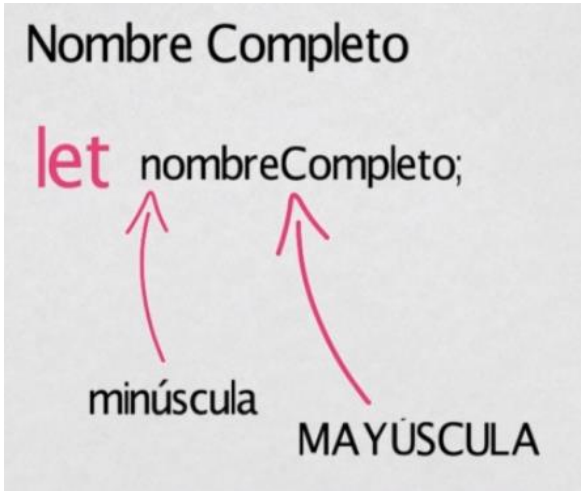
EJEMPLOS

```
let $hola;
let hol@;
let adiós;
let o'conor;
let ca$a;
let año;
```

¿Cuáles son correctas y no correctas?

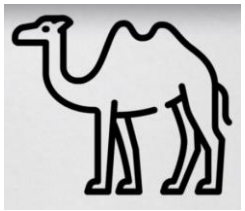
Uso del CamelCase (Recomendación)

Reglas sobre los nombres de variables y constantes.



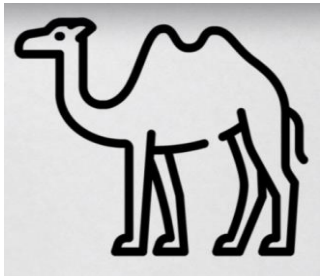
Ejemplos: Definir variables para

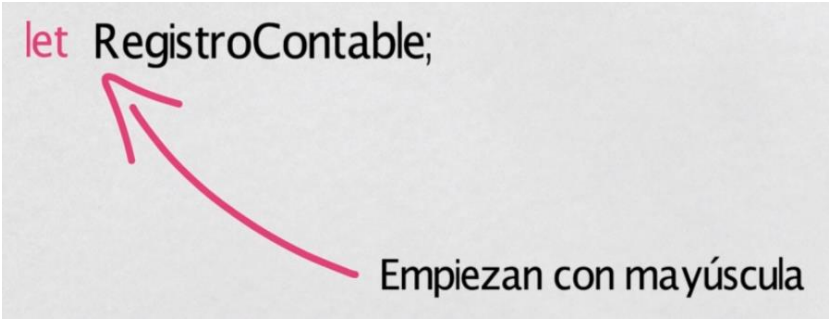
- Dirección de una casa
- Salario Bruto
- Tamaño de envase



lowerCamelCase

UpperCamelCase





Tipos de Datos (I)

TIPOS DE DATOS

EN DECIRLE A LA COMPUTADORA
QUE TIPO DE CONTENIDO TIENE UNA VARIABLE O CONSTANTE

STRING - CADENAS DE CARACTERES

Almacenan palabras, frases, cadenas de letras, símbolos...

```
let nombre = "Claudia Soto";
```

```
let nombre :string = "Claudia Soto";
```

```
let salario = 1500;
```

NUMBER - NÚMERO

QUIERE DECIR, QUE ESTA VARIABLE CONTENDRÁ ÚNICAMENTE NÚMEROS
QUE PUEDEN SER ASIGNADOS O PRODUCTO DE CÁLCULOS MATEMÁTICOS

```
let promedioNotas = 85;
```

```
let promedioNotas :number = 85;
```

BOOLEANS - BOOLEANOS

SON VARIABLES QUE ALMACENAN ÚNICAMENTE 2 ESTADOS.
VERDADERO Y FALSO

```
let activo = true;
```

PALABRA RESERVADA

```
let feo:boolean = false;
```

RECOMENDACIÓN

El nombre para una variable de tipo booleana debe de leerse en positivo

```
let noActivo = false;
```

Hagamos ejemplos prácticos de tipos de datos…

A practicar !!!

Tipos de Datos (II)

Lenguajes débilmente tipados:

– No se suele indicar el tipo de variable al declararla.

VENTAJAS

- Nos olvidamos de declarar el tipo
- Podemos cambiar el tipo de la variable sobre la marcha. Por ejemplo, asignarle un **string** a un **int**
- Escribimos menos código

DESVENTAJAS ⟶ *¿Analizar cuales?*

```
> let resultado = "x" + 5;
<· undefined
> resultado
<· "x5"
```

En un lenguaje fuertemente tipado daría un error, pero en JavaScript no pasa nada:

Tipos de Datos (III)

Lenguajes fuertemente tipados:

– Se debe indicar el tipo de dato al declarar la variable no pudiendo operar entre distintos tipos directamente.




VENTAJAS

- Código expresivo: ahora sí sabremos de qué tipo espera un argumento una función
- Menos errores: Nos olvidaremos de ver el tipo de variable antes de hacer operaciones con ésta

DESVENTAJAS ⟶ *¿Analizar cuales?*

```
>>> resultado = "x" + 5
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: must be str, not int
>>>
```

la operación de “x” + 5 en Python

Tipos de Datos (IV)

Tabla de tipos de datos de C#

Estructura	Detalles	Bits	Intervalo de valores	Tipo C#
Byte	Bytes sin signo	8	[0, 255]	byte
Int16	Enteros cortos con signo	16	[-32.768, 32.767]	short
UInt16	Enteros cortos sin signo	16	[0, 65.535]	ushort
Int32	Enteros normales	32	[-2.147.483.648, 2.147.483.647]	int
UInt32	Enteros normales sin signo	32	[0, 4.294.967.295]	uint
Int64	Enteros largos	64	[-9.223.372.036.854.775.808, 9.223.372.036.854.775.807]	long
UInt64	Enteros largos sin signo	64	[0, 18.446.744.073.709.551.615]	ulong
Single	Reales con 7 dígitos de precisión	32	$[1{,}5\times10^{-45}$, $3{,}4\times10^{38}]$	float
Double	Reales de 15-16 dígitos de precisión	64	$[5{,}0\times10^{-324}$, $1{,}7\times10^{308}]$	double
Decimal	Reales de 28-29 dígitos de precisión	128	$[1{,}0\times10^{-28}$, $7{,}9\times10^{28}]$	decimal
Boolean	Valores lógicos	32	true, false	bool
Char	Caracteres unicode	16	['\u0000', '\uFFFF']	char
String	Cadenas de caracteres	Variable	El permitido por la memoria	string
Object	Cualquier objeto	Variable	Cualquier objeto	object

Ejemplos de declaraciones:

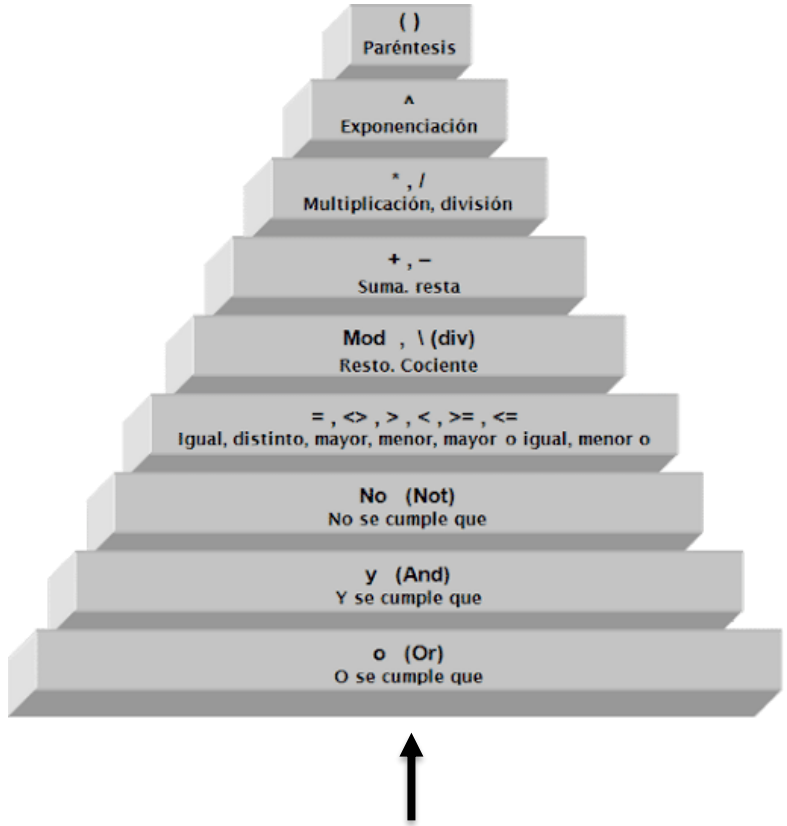
```
int entero = -2; //Declarado e inicializado
uint sinsigno; //Declarado
float concoma = 3.7f; //Declarado e inicializado
char caracter; //Declarado
string cadena; //Declarado
bool nuevo = false; //Declarado e inicializado

//Inicializando los que declaramos
sinsigno = 5;
caracter = 'k'; //Entre comillas simples (' ')
cadena = "palabra"; //Entre comillas dobles (" ")
```

Operadores aritméticos

Operadores aritméticos

Operadores Aritméticos		
	Significado	Ejemplo
+	Suma	a + b
-	Resta	a - b
-	Niega	-a
*	Producto	a * b
/	Cociente	a / b
%	Módulo	a % b



Prioridades en expresiones aritméticas: Uso de paréntesis ()

Expresiones y operadores - JavaScript | MDN (mozilla.org)

Florida Universitària

Operadores aritméticos

Operadores Aritméticos

	Significado	Ejemplo
+	Suma	a + b
-	Resta	a - b
-	Niega	-a
*	Producto	a * b
/	Cociente	a / b
%	Módulo	a % b

OPERADORES ARITMETICOS			
Operador	Descripción	Ejemplo	Resultado
+	Suma	c = 3 + 5	c = 8
-	Resta	c = 4 - 2	c = 2
-	Negación	c = -7	c = -7
*	Multiplicación	c = 3 * 6	c = 18
**	Potenciación	c = 2 ** 3	c = 8
/	División	c = 7.5 / 2	c = 3.75
//	División entera	c = 7.5 // 2	c = 3.0
%	Módulo	c = 8 % 3	c = 2

Prioridades en expresiones aritméticas: Uso de paréntesis ()

Expresiones y operadores - JavaScript | MDN (mozilla.org)

MDN web docs moz://a

Operadores aritméticos

Programa básico que calcula los valores de la ecuación de 2º grado

$$ax^2 + bx + c = 0$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$3 \cdot x^2 + -5 \cdot x + 1 = 0$$

ecu2grado.js > ...

```
1  let a, b, c;
2  let primeraSol, segundaSol;
3
4  a = 3;
5  b = -5;
6  c = 1;
7
8  primeraSol = (-b + Math.sqrt(b*b-4*a*c))/(2*a);
9  segundaSol = (-b - Math.sqrt(b*b-4*a*c))/(2*a);
10
11 console.log("Pimera solución:",primeraSol);
12 console.log("Segunda solución:",segundaSol);
```

```
Pimera solución: 1.4342585459106665
Segunda solución: 0.23240812075600018
```

¡¡¡ Muy importante saber gestionar las prioridades aritméticas !!!

Florida Universitària

Operadores aritméticos abreviados

Operadores Aritméticos abreviados		
+=	b += 3	b =b + 3
-=	b -= 3	b =b - 3
*=	b *= 3	b =b * 3
/=	b /= 3	b =b / 3
%=	b %= 3	b =b % 3
++	++b, b++	b=b + 1
--	--b, b--	b=b - 1

++	op ++	Incrementa op en 1; evalúa el valor antes de incrementar
++	++ op	Incrementa op en 1; evalúa el valor después de incrementar
--	op --	Decrementa op en 1; evalúa el valor antes de decrementar
--	-- op	Decrementa op en 1; evalúa el valor después de decrementar

¿Cuál es la diferencia entre?

```
let b = 10, a;

a = b++;
a = ++b;
```

Expresiones y operadores - JavaScript | MDN (mozilla.org)

MDN web docs moz://a

Operadores relacionales (comparación)

Operadores relacionales (de comparación)

OPERADORES LÓGICOS Y RELACIONALES	DESCRIPCIÓN	EJEMPLO
==	Es igual	a == b
===	Es estrictamente igual	a === b
!=	Es distinto	a != b
!==	Es estrictamente distinto	a !== b
<, <=, >, >=	Menor, menor o igual, mayor, mayor o igual	a <=b
&&	Operador and (y)	a && b
||	Operador or (o)	a || b
!	Operador not (no)	!a

```
1  console.log(1 == 1);
2  // expected output: true
3
4  console.log("1" == 1);
5  // expected output: true
6
7  console.log(1 === 1);
8  // expected output: true
9
10 console.log("1" === 1);
11 // expected output: false
12
```

JavaScript tiene comparaciones estrictas y de conversión de tipos. Una comparación estricta (por ejemplo, `===`) solo es verdadera si los operandos son del mismo tipo y los contenidos coinciden. La comparación abstracta más comúnmente utilizada (por ejemplo, `==`) convierte los operandos al mismo tipo antes de hacer la comparación.

Expresiones y operadores - JavaScript | MDN (mozilla.org)

Operadores lógicos

Operadores relacionales (lógicos)

Operador	Uso	Descripción
AND (&&)	*expr1* && *expr2*	&& devuelve true si ambos operandos son verdaderos; de lo contrario devuelve false.
OR (||)	*expr1* || *expr2*	|| devuelve true si cualquier operando es verdadero; pero si ambos son falsos, devuelve "false".
NOT (!)	!*expr*	Devuelve false si su único operando puede convertirse a true; de lo contrario, regresa true.

variable1	variable2	variable1 && variable2
true	true	true
true	false	false
false	true	false
false	false	false

variable1	variable2	variable1 || variable2
true	true	true
true	false	true
false	true	true
false	false	false

ejemplos del operador && (AND lógico).

```
false && true  || true      // regresa true
false && (true || true)     // regresa false
```

```
a1 = true  && true      // t && t regresa true
a2 = true  && false     // t && f regresa false
a3 = false && true      // f && t regresa false
a4 = false && (3 == 4)  // f && f regresa false
a5 = "Cat" && "Dog"     // t && t regresa "Dog"
a6 = false && "Cat"     // f && t regresa false
a7 = "Cat" && false     // t && f regresa false
```

Expresiones y operadores - JavaScript | MDN (mozilla.org)

Ejercicios con operadores (I)

Florida Universitària

Dadas las variables de tipo numérico (A=5, B=3, C=-12). Indicar si la evaluación de estas expresiones daría **TRUE o FALSE**

a) A > 3

b) A > C

c) A < C

d) B < C

e) B != C

f) A == 3

g) A * B == 15

h) A * B == -30

i) C / B < A

j) C / B == -10

k) C / B == -4

l) A + B + C == 5

m) (A+B == 8) && (A-B == 2)

n) (A+B == 8) || (A-B == 6)

o) A > 3 && B > 3 && C < 3

p) A > 3 && B >= 3 && C < -3

Puedes crear un programa donde declares las variables, les asignes los valores y muestres por consola el valor que realmente devuelve cada una de las expresiones. **¿Coinciden con lo que realmente habías pensado?...**

Ejercicios con operadores (II)

Dadas las siguientes expresiones, indicar si la evaluación daría **TRUE o FALSE**

```
5 < 4
4 >= 3
3 == 3
true === 'true'
true == 'true'
0 == false
'' == false
'' === false
'valencia' === 'valencia '
'valencia' > 'cullera' && 4>0
6>7 && 'valencia' > 'cullera'
45 < 34 || 12 !== 45
45 < 34 || 12 === 45
!(45<34) || 12 === 45
!(45 < 34)
```

Ejercicios con operadores (III)

Dadas las siguientes expresiones, indicar si la evaluación daría **TRUE o FALSE**

```
5 < 4                                   false
4 >= 3                                  true
3 == 3                                  true
true === 'true'                         false
true == 'true'                          false
0 == false                              true
'' == false                             true
'' === false                            false
'valencia' === 'valencia '              false
'valencia' > 'cullera' && 4>0           true
6>7 && 'valencia' > 'cullera'           false
45 < 34 || 12 !== 45                    true
45 < 34 || 12 === 45                    false
!(45<34) || 12 === 45                   true
!(45 < 34)                              true
```

Ejercicios con tipos de datos y operadores

Florida Universitària

JS tipodatos.js

```
1
2  // Ejemplos de tipos de datos
3  let nombre = 'Salva';
4  let apellido = 'Gutierrez';
5  let nombreCompleto = nombre + ' ' + apellido;
6  console.log(nombreCompleto);
7
8  let a = 10, b = 20;
9  console.log(a+b);
10
11 let c = '10', d = '20';
12 console.log(c+d);
13
14 let casado = true;
15 let despedido = false;
16 console.log(casado + despedido);
17
18 let casada = 'true';
19 let despedida = false;
20 console.log(casada + despedida);
21
22
```

Pruebas de tipos de datos y operadores

Probad y analizad los resultados…

A practicar !!!