Sentencias

El código se divide en sentencias o instrucciones. Un programa es un conjunto de sentencias.
Ya no es necesario que finalicen las lineas con el punto y coma (;) Aunque es una buena práctica.

Ejemplo: *var nombre = "Mario";*
(Asigna el texto Mario a una variable llamada nombre)

Identificadores y palabras reservadas

Las *palabras reservadas* son términos que ofrece el lenguaje para poder programar, ejemplos: var, if, try, return, enum, else, new, class …

Los *indentificadores* son los nombres que damos a las variables y otros elementos del lenguaje, ejemplo: **nombre**

✔ Deben de comenzar por una letra, guion bajo (_) o el símbolo dolar ($)

✔ Nunca deben empezar por números o caracteres especiales

✔ Después se pueden usar letras o números

✔ Se distinguen entre mayúsculas y minúsculas, por lo tanto: **mes y Mes son identificadores diferentes.**

Recomendaciones para la nomenclatura:

- Usa snake_case para archivos :

  mi_archivo_javascript.js

- Usa UPPER_CASE para constantes:

  const PI =3.1416

  const UNA_CONSTANTE =’Soy una constante’;

- Usa UpperCamelCase para clases

  class SerHumano

- Usa LowerCamelCase para objetos, funciones, variables …

  var unaCadena;

  let miNombre = ‘Belén’ ;

Comentarios

Son textos que no se tienen en cuenta al interpretar el código, se usan para documentar. Hay dos formas:

● De una sola linea: comienzan por los símbolos **//**

✔ Ejemplo:

```
var tam=nombre.length();    //tam guarda el tamano del nombre del usuario
```

● De varias lineas, comienzan y terminan con los símbolos **/* y */**

✔ Ejemplo:

```
/* Comienzo de la funcion principal
que muestra los datos */
```

Bloques de sentencias

Las sentencias se pueden agrupar en bloques e irán agrupadas dentro de llaves **{ }**

✔ Ejemplo:

```
var x=9;
if( x==9 ){
        x++;
        console.log(x);
}
```

Por consola se escribe el número 10 ya que se cumple la condición x igual a 9.

Tipos de Datos y operador typeof

- Primitivos
  - Boolean : valores verdadero/falso (true/false)
  - Number: números sobre los que se pueden hacer operaciones matemáticas
  - Undefined: valor no definido, la variables o constante existe pero no tiene valor
  - String: conjunto de caracteres alfanuméricos
  - NaN: not a number
  - BigInt: para representar numéros enteros de tamaño arbitrario. (ECMAScript 2020)
  - Symbol: para añadir claves de propiedades únicas a un objeto. (Nuevo en ECMAScript 2015)
- Otros
  - Object: objetos, arrays, null: contiene el valor null
  - Function

1. Para conocer el tipo de datos se utiliza el operador

typeof

2. **Ej: typeof 1+1**

✔ Ejercicio:

Escribir en la consola qué tipos corresponden a los siguientes valores. Crear un fichero index.html.

12, 12.3, (12*3), 'Hola', "Hola", Hola, (1/0), null , [1,2,3,4], true, (12>3)

Variables

- Las variables deben ser declaradas antes de poderse usar. Declarar una variable es asignarla a un identificador
- Se usa la palabra reservada **var** → ámbito global o de función, **NO POR BLOQUE**

```
var x; //a partir de este momento podemos usar en el código la variable x
```

- Es muy habitual asignar un valor al declarar una variable

```
var x=10;
```

- Las variables sólo se declaran una vez

```
var x=9;
x=25;
console.log(x);  // escribe el valor 25
```

- ¿Qué **muestra** el siguiente código? Probar el siguiente código y ver el resultado

```
<script>
        {
           var x = 9;
        }
        console.log(x);
</script>
```

¿ Aparece el número 9 ?

Diferencias entre let y var

A partir de 2015:

let se utiliza para declarar variables de forma más local que var, tiene ámbito en bloque. Es mejor USAR LET.

- let x=25;

Probar el siguiente código y ver el resultado

```
<script>
        {
           let x = 9;
        }
        console.log(x);
</script>
```

¿Qué pasa al ejecutarse? ¿Por qué?

x is not defined

La variable x se ha definido dentro de un bloque mediante let y no se puede utilizar fuera

Sin embargo, **var** sobrevive fuera del bloque.

✔ **Ejercicio: ¿Qué muestra por consola?**

```
var nombre = 'Juan';

if(nombre) {
   var nombre = 'Pedro';

   console.log(nombre, ' dentro del IF');
}

console.log(nombre, ' fuera del IF');
```

✔ **Ejercicio: ¿Qué muestra por consola?**

```
var nombre = 'Juan';

if(nombre) {
   let nombre = 'Pedro';

   console.log(nombre, ' dentro del IF');
}

console.log(nombre, ' fuera del IF');
```

Constantes: const

- Se usan para definir constantes, valores que no pueden modificarse.
- Se usa la palabra reservada **const**
- Es recomendable usar las constantes en mayúsculas así las distinguimos de las variables
  - ✔ Ejemplo:

    const PI = 3.141592;

    PI= 23;

    La ejecución anterior daría error. No podemos modificar el valor de una variable declarada con la palabra const.

Tipos de datos

1.- String

- Son cadenas de caracteres
- Se delimitan con comillas dobles o simples

  let dia = “lunes”;

  let dia2 = ‘lunes’;
- Desde la versión 6 (2015) se permitan también las **comillas invertidas: String templates**

  let dia3 = `lunes`;

  console.log(**`Hoy es ${dia}`**);

  let saludo = new String(“Hola Mundo”);
- Operador de concatenación (+):

  console.log(“Hoy es lunes”);

  console.log(“Hoy es ” + dia);
- Métodos de String: (El primer índice es 0 y el último length-1)

  https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/String

  - **charAt**: Devuelve el carácter de la posición x. Recordad que la primera posición es 0.
    - let nombre = “leon”;
      console.log(nombre.charAt(1)); // Escribe por consola la letra e
    - Equivale a nombre[1];
  - **substring**: Devuelve la subcadena comprendida entre dos posiciones (última sin incluir)
    - let nombre = “manzana”;
      console.log(nombre.substring(1,3)); // Escribe por consola an

- **split:** Divide una cadena en un array de cadenas usando un separador especificado.
  - let fecha = ”01/01/2012”;
    let array = fecha.split(“/”); // crea un array de 3 posiciones con 01, 01 y 2012

- **toUpperCase**: Devuelve la cadena convertida a mayúsculas

- **toLowerCase**: Devuelve la cadena convertida a minúsculas

- **trim**: Elimina los espacios en blanco del inicio y fin de la cadena.
  - let cadena = “ hola”;
    console.log(cadena.trim()); //Escribe por consola: hola (sin espacios por delante)

- **indexOf**: Devuelve el primer índice donde se encuentre un carácter o cadena.
  - “jueves”.indexOf(“e”); // Devuelve 2

- **lastIndexOf**: Devuelve el último índice donde se encuentre un carácter o cadena.
  - “jueves”.lastIndexOf(“e”); // Devuelve 4

- **startsWith:** Indica si una cadena de texto comienza con un texto en concreto:
  - ‘Lunes’.startsWith(‘Lu’); // Devuelve true

- **endsWith:** Indica si una cadena de texto termina con un texto en contreto:
  - ‘Lunes’.endsWith(‘ss’); //Devuelve false

- **Secuencias de escape**

  Son caracteres especiales que van dentro de las cadenas para aportar un comportamiento determinado.

  \n Salto de línea: let texto = “Hoy hace sol pero **\n**está nublado”;

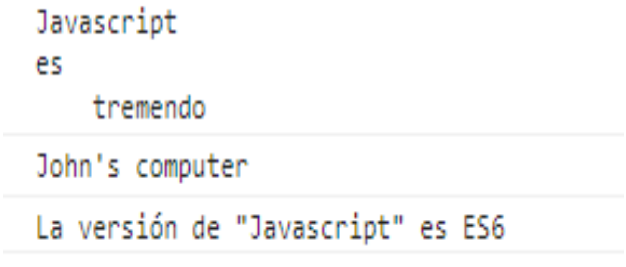
  \” Comillas dobles: let texto =”Emilio dijo \”¡qué calor! \” ”;

  \t Tabulador: let texto =”Emilio dijo: \tBuenas tardes”;

  \’ Comillas simples let texto =”Emilio dijo: \’Buenas tardes\’”;

  \\ El carácter backslash: let texto =”Emilio dijo: \\Buenas tardes”;

Vamos a practicar. Crear dentro de una etiqueta <script> código JavaScript para mostrar lo siguiente:



2.- Number: https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/Number

- No importa si tiene o no decimales.
- Se escriben tal cual, sin comillas ni símbolos extra.
- Si tiene decimal, se usa el **punto** como separador decimal.

  ✔ Ejemplo:

```
let entero = 2000;
let decimal = 0.21;
```

- Formatos

```
const CIENTIFICO = 6.022e+23; // significa 6,022*10^23
let hexadecimal = 0xAB12;  //El resultado es 43794 ( AB12 en decimal)
let octal = 0o27652; // El resultado es 12202 ( 27652 en decimal)
let binario = 0b10111011; // El resultado es 187
```

- Métodos de Number
  - **toFixed**: formatea un número con tantos decimales como se le indique. Si no se indica el número de decimales redondea el número.

```
console.log((12.89).toFixed());       // Escribe  13
console.log((12.123).toFixed(2));   // Escribe 12.12
```

  - **isNaN**: Determina si el valor es NaN (Not a Number), es decir si una variable es un número o no. Si no es un número devuelve true y si lo es devuelve false.

```
isNaN(“Hola”);   // true
isNaN(“15”);      // false
```

  - **parseInt**: Convierte un string en número.

```
console.log(parseInt("hola"));       // NaN
console.log(parseInt("37"));          // 37
console.log(parseInt("37hola"));    // 37
console.log(parseInt("hola37"));    // NaN
```

3.- Boolean

- Sólo puede tomar los valores: true o false.
- Ejemplo:

  let b = true;

  let b = (2>3); // b sería false

4.- Null y undefined: ¿Qué diferencia hay entre ambas?

- Undefined: variable que no ha sido inicializada:

  let a;

- Null: la variable explícitamente ha sido asignada con null.

Operadores

1.- Operadores Aritméticos

Operador	Descripción	Ejemplo
Resto(%)	Operador binario. Devuelve el resto entero de dividir los dos operandos.	12 % 5 devuelve 2.
Incremento(++)	Operador unario. Agrega uno a su operando. Si se usa como operador prefijo (++x), devuelve el valor de su operando después de agregar uno; si se usa como operador sufijo (x++), devuelve el valor de su operando antes de agregar uno.	Si x es 3, ++x establece x en 4 y devuelve 4, mientras que x++ devuelve 3 y , solo entonces, establece x en 4.
Decremento(--)	Operador unario. Resta uno de su operando. El valor de retorno es análogo al del operador de incremento.	Si x es 3, entonces –x establece x en 2 y devuelve 2, mientras que x-- devuelve 3 y, solo entonces, establece x en 2.
Negación unaria(-)	Operador unario. Devuelve la negación de su operando.	Si x es 3, entonces -x devuelve -3.
Positivo unario(+)	Operador unario. Intenta convertir el operando en un número, si aún no lo es.	+"3" devuelve 3. +true devuelve1.
Operador de exponenciación(**)	Calcula la base a la potencia de exponente	2 ** 3 devuelve 8 10 ** -1 devuelve 0.1

Comparar estos comportamientos:

```
let valor =50, incremento, decremento;
incremento = valor ++;    //  ¿cuanto vale incremento y valor?
decremento = valor--;     //   ¿cuanto vale incremento y valor?

valor =50;
incremento = ++valor; //  ¿cuanto vale incremento y valor?
decremento = --valor;  //   ¿cuanto vale incremento y valor?
```

2.- Operadores Relacionales. Devuelven un valor booleano

Operador	Descripción	Ejemplo
Igual **==**	Compara si los dos valores son iguales	2==3
Distinto **!=**	Compara si los dos valores son distintos	2!=3
Mayor que **>**	Comprueba si el primer valor es mayor que el segundo	3>2
Menor que **<**	Comprueba si el primer valor es menor que el segundo	2<4
Mayor o igual que **>=**	Comprueba si el primer valor es mayor o igual que el segundo	2>=2
Menor o igual que **<=**	Comprueba si el primer valor es menor o igual que el segundo	3<=2
Estrictamente igual **===**	Compara si son iguales en **valor y tipo**	a===b
Estrictamente distinto **!==**	Compara si son distintos en **valor y tipo**	a!==b

¿Qué diferencia hay entre:
=
==
===
?

3.- Operadores Lógicos. Devuelven un valor booleano

Operador	Descripción	Ejemplo
OR **||**	Devuelve true si una o las dos condiciones son true	x==1 || x==2
AND **&&**	Devuelve true si las dos condiciones son true	x==1 && y==2
NOT **!**	Devuelve el opuesto valor booleano.	!x

¿Qué devuelven las siguientes expresiones?

```
let x=9;
let y=10;
console.log(x==9 && y ==10)

let x=9;
let y=12;
console.log(x==9 || y ==10)
```

```
let x=11;
let y=12;
console.log(!x==9 || y ==12)

let x=11;
let y=12;
console.log(!(x==9 || y ==12))
```

NOTA: EL OPERADOR AND SOLO EVALUA LA SEGUNDA EXPRESION SI LA PRIMERA ES VERDADERA.

Ejemplo: ¿Qué devuelve la variable **conduce**?

```
let edad=17;
let conduce =(edad >=18 && carnet==true);

console.log(conduce);
```

La variable **conduce** será falsa porque la edad es 17.

Aunque no hayamos declarado la variable carnet y provocaría un error por indefinición, la realidad es que no se va a evaluar la segunda expresión porque es falsa la primera.

4.- Operadores de Asignación

Operador	Descripción	Ejemplo
Asignación **=**	Asigna un valor a una variable	x=2
Asignación de adición **+=**	Suma a la variable un valor y lo asigna de nuevo	x+=2 → x=x+2;
Asignación de resta **+=**	Resta a la variable un valor y lo asigna de nuevo	x-=2 → x= x-2;
Asignación de multiplicación ***=**	Multiplica a la variable un valor y lo asigna de nuevo	x*=2 → x= x*2;
Asignación de división **/=**	Divide a la variable un valor y lo asigna de nuevo	x/=2 → x= x/2;
Asignación de resto **%=**	Aplica el resto de la variable x entre un número y lo asigna de nuevo	x%=2 → x= x%2;
Asignación de exponenciacion ****=**	Aplica el exponente de la variable x elevado a un número y lo asigna de nuevo	x**=2 → x= x**;

5.- Operador Condicional

Condición ? ValorSiVerdadera : valorSiFalsa

Si se cumple una condición, devuelve el primer valor, en caso contrario devuelve el segundo.

Ejemplo:

let tipo = (numero%2==0) ? “par” : “impar”;

Conversión automática

Javascript no es un lenguaje tipado, esto significa que no tiene tipos declarados, por lo que hace conversiones automáticas:

Ejemplos:

- El cuadro de mensajes alert convierte automaticamente cualquier valor a string para mostrarlo
- Las operaciones matemáticas convierten los valores a números.

Reglas de conversión numéricas:

Valor	Se convierte a …
Undefined	NaN
Null	0
True	1
False	0
String	NaN

Ejemplos:

console.log(‘2’ * 3); → ¿Error?

console.log(‘2’ + 3); → Ojo ¿Qué devuelve?

console.log(2 + ‘3’); → Ojo ¿Qué devuelve?

console.log(‘Hola’ * 3); → NaN

console.log(9+4+"10"); Ojo, ¿Qué obtenemos?

Conversión Forzada

1.- Number()

let x=’2’;

let y=’3’;

console.log(**Number**(x) + **Number**(y)); ¿Qué escribe?

2.- String()

```
let x=2;
let y=3;
console.log(String(2) + String(3));     ¿Qué escribe?
```

3.- parseInt(): Convierte una cadena a número entero

Ejemplos:

```
parseInt("12");      // convierte al número 12
parseInt("Hola");  // NaN
```

4.- parseFloat():Convierte una cadena a número con decimales

Ejemplos:

```
parseFloat("3.14");      // convierte al número 3.14
parseFloat("Hola");  // NaN
```

Cuadros de diálogo

1.- Mensajes de alerta.

- Muestra una ventana con un mensaje
- Se utiliza el método: **alert**
- **Ejemplo: alert**("Buenas tardes a todos");

2.- Cuadros de confirmación

- Muestra una ventana con un mensaje y requiere contestación por parte del usuario.
- Se utiliza el método: **confirm**
- El resultado de este método es true si acepta o false si cancela.
- **Ejemplo: let resultado = confirm("¿Acepta las condiciones?");**

3.- Cuadros de entrada de texto

- Muestra una ventana con un cuadro de texto para recoger la respuesta del usuario.
- Si el usuario Acepta, se recoge la respuesta
- Si el usuario Cancela, devuelve null.
- Se utiliza el método: **prompt**
- **Ejemplo: let resultado = prompt("Escribe tu nombre");**

```
alert(`Tu nombre es ${resultado}`);
```

Ejercicios

1. Crear una página que muestre por consola:
    1. El mensaje “Bienvenido” al entrar en la página.
    2. Una frase con comillas simples en el interior
    3. Una frase con comillas dobles en el interior
    4. Al entrar muestre un mensaje alert con vuestro nombre.
    5. Repetimos el paso anterior creando una variable con vuestro nombre.
2. Crear una página que pida al usuario vuestro nombre a través de un cuadro de entrada de texto. Y que pinte en la página el resultado.
3. Crear una página que pregunte al usuario si es mayor de edad o no. Mostrar por consola el resultado
4. Crear una página tal que:
    1. Devuelva y muestre por consola la primera coincidencia de “Mundo” en la variable txt=”Hola Mundo”;
    2. Sustituya la palabra ‘Mundo’ por ‘Universo’ y muestre por consola: “Hola Universo”
    3. Muestre por consola el texto en mayúsculas
    4. Muestre por consola el texto en minúsculas
    5. Dadas dos cadenas una con vuestro nombre y otra con vuestro apellido, escribir por consola el nombre completo utilizando el operador de concatenación.
    6. Dada la palabra Cliente, recuperar la subcadena Cli y escibirla por consola.
    7. Dada la palabra Cliente, recuperar la subcadena ente y escibirla por consola.
5. La función isNaN, devuelve verdadero si la expresión no es numérica:
    1. ¿Qué devuelve console.log(isNaN(“Hola” * 5));?
    2. ¿Y console.log(isNaN(“3”*5)); ?
6. Crear una página que pida al usuario por pantalla su nombre y su correo electrónico. Mostrar los datos en la página HTML de forma que el nombre lo pinte con <h1> y el correo con <h2>
7. Crear una página que pida al usuario dos valores numéricos y escriba en la página HTML la suma y la resta de ambos de forma que muestre con string templates (no contatenando):
    1. La suma de a y b es c
    2. La resta de a y b es c
8. Crear una página que pida el precio de un artículo y la cantidad de artículos que desea comprar. Escribir en la página HTML el total que debe abonar. (El precio debe ser un numero con decimales)

    ¿Cómo podemos fijar el número decimal sólo con 2 decimales?
9. Crear una página que pida al usuario un número y escriba por pantalla el número de cifras que tiene el número. Escribir el resultado con string template en una etiqueta en cursiva y color azul.