TEMA 2.- INTRODUCCIÓN AL LENGUAJE JAVASCRIPT.

Objetivos

- Conocer las principales características del lenguaje Javascript.
- Dominar la sintaxis básica del lenguaje.
- Comprender y utilizar los distintos tipos de variables.
- Conocer las diferentes sentencias condicionales.

Contenidos

2.1.- Características de Javascript.

Javascript es un lenguaje de programación interpretado utilizado fundamentalmente para dotar de comportamiento dinámico a las páginas web.

Cualquier navegador web actual incorpora un intérprete para código JavaScript.

Su sintaxis se asemeja a la de C++ y Java y como curiosidad todas sus variables son globales.

La forma más inmediata de empezar a programar en Javascript es escribir secuencias de comandos es mediante un editor de texto y visualizarlo en el navegador.

2.2.- "Hola Mundo" con Javascript.

Hay dos formas de embeber el código JavaScript en una página HTML:

a) Incluirlo directamente en la página HTML mediante la etiqueta <script>.

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv= "content-type" content="text/html;
      charset=utf-8">
    <title>Hola Mundo</title>
  </head>
  <body>
    <script>
      alert('Hola mundo en JavaScript')
    </script>
  </body>
</html>
```

b) Utilizar el atributo src de la etiqueta <script> para especificar el fichero que contiene el código JavaScript.

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
      charset=utf-8">
    <title>Hola Mundo</title>
    <script type="text/javascript" src="HolaMundo.js">
    </script>
  </head>
  <body></body>
</html>
```

El fichero "HolaMundo.js" debe contener la instrucción:

```
alert('Hola Mundo en Javascript');
```

2.3.- El Lenguaje Javascript: Sintaxis.

La sintaxis de JavaScript es muy similar a la de Java o C++.

Mayúsculas y minúsculas.

El lenguaje distingue entre mayúsculas y minúsculas, a diferencia de por ejemplo HTML. No es lo mismo utilizar alert() que Alert().

Comentarios.

Los comentarios no se interpretan por el navegador. Existen dos formas de insertar comentarios:

- Doble barra (//) – Se comenta una sola línea de código.
- Barra y asterisco (/* al inicio y */ al final) – Se comentan varias líneas de código.

```
<script type="text/javascript">
  // Este modo permite comentar una sola línea
  /* Este modo permite realizar
  comentarios de
  varias líneas */
</script>
```

Mostrar información por la pantalla.

Utilizamos la instrucción:

```
document.write(variable);
document.write(función(parámetro));
document.write( "texto");
```

Tabulaciones y saltos de línea.

JavaScript ignora los espacios, las tabulaciones y los saltos de línea. Emplear la tabulación y los saltos de línea mejora la presentación y la legibilidad del código.

```
Ejemplo 1:
<script type="text/javascript">var i,j=0;
for (i=0;i<5;i++){ alert("Variable i: "+i);
for (j=0;j<5;j++){ if (i%2==0){
document.write
(i + "-" + j + "<br>");}}}</script>


Ejemplo 2:
<script type="text/javascript">
  var i,j=0;
  for (i=0;i<5;i++){
    alert("Variable i: "+i;
    for (j=0;j<5;j++){
      if (i%2==0){
        document.write(i + "-" + j + "<br>");
      }
    }
  }
</script>
```

Punto y coma.

Se suele insertar un signo de punto y coma (;) al final de cada instrucción de JavaScript. Su utilidad es separar y diferenciar cada instrucción. Se puede omitir si cada instrucción se encuentra en una línea independiente (la omisión del punto y coma no es una buena práctica de programación).

Los bloques de instrucciones se encierran entre llaves

```
      {
            bloque ;
      }
```

Palabras Reservadas.

Algunas palabras no se pueden utilizar para definir nombres de variables, funciones o etiquetas. Es aconsejable no utilizar tampoco las palabras reservadas para futuras versiones de JavaScript. En www.ecmascript.org es posible consultar todas las palabras reservadas de JavaScript.

2.4.- Tipos de datos.

Los tipos de datos especifican qué tipo de valor se guardará en una determinada variable.
Los tres tipos de datos primitivos de JavaScript son:

- Números.
- Cadenas de texto.
- Valores booleanos

Números.

- En JavaScript existe sólo un tipo de dato numérico.
- Todos los números se representan a través del formato de punto flotante de 64 bits.
- Este formato es el llamado double en los lenguajes Java o C++.

Cadenas de texto.

- El tipo de datos para representar cadenas de texto se llama string.
- Se pueden representar letras, dígitos, signos de puntuación o cualquier otro carácter de Unicode.
- La cadena de caracteres se debe definir entre comillas dobles o comillas simples.

Secuencia de escape	Resultado
\\	Barra invertida
\’	Comilla simple
\”	Comillas dobles

Valores booleanos.

- También conocido como valor lógico.
- Sólo admite dos valores: true o false.
- Es muy útil a la hora de evaluar expresiones lógicas o verificar condiciones.

2.5.- Variables.

Las variables son zonas de la memoria de un ordenador que se identifican con un nombre y en las cuales se almacenan ciertos datos. Debemos tener en cuenta respecto a las variables que:

- Las variables no mantiene tipos estrictos. Se puede asignar un valor entero y luego un valor cadena.
- No es necesario utilizar la palabra reservada var.
- No es necesario inicializarlas.
- El nombre puede contener letras, números o guión bajo (_) pero no pueden empezar por número.
- Diferencia entre mayúsculas y minúsculas. No es lo mismo la variable “Edad” que “edad”.

El desarrollo de un script conlleva:

- Declaración de variables.
- Inicialización de variables.

a) Declaración de variables:

Se declaran mediante la palabra clave var seguida por el nombre que se quiera dar a la variable.

```
var mi_variable;
```

Es posible declarar más de una variable en una sola línea.

```
var mi_variable1, mi_variable2;
```

b) Inicialización de variables.

Se puede asignar un valor a una variable de tres formas:

- Asignación directa de un valor concreto.
- Asignación indirecta a través de un cálculo en el que se implican a otras variables o constantes.
- Asignación a través de la solicitud del valor al usuario del programa.

Ejemplos:

```
var mi_variable_1 = 30;
var mi_variable_2 = mi_variable_1 + 10;
var mi_variable_3 = prompt('Introduce un valor:');
```

Las variables declaradas dentro de las funciones son variables locales y existen solo dentro de la misma. Las variables globales tienen ámbito en todo el programa, incluso dentro de la función.

Ejemplos:

```
var edad = 18;
var edad = "Adulto";
var precio;
precio = 150;
var _nombre = "Luis";
var nombre_apellidos = "Luis Sanchez Lopez";
var precio = 100, cantidad = 3, código = 5;
```

Declaración de un Array.

Opción 1.

```
var nombre_array = new Array(valor1, valor2, valor3, ...);
```

Opción 2.

```
var nombre_array = [valor1, valor2, valor3, ...]
```

```
<script type="text/javascript">
      var alumnos = ["Luis", "Pedro", "Ana"];
      document.write(alumnos[1]+"<br>");
</script>
```

Declaración de un Array Asociativo.

```
  var nombre_array = {"pos1":valor1,"pos2":valor2,... "posn":valorn};
```

Ejemplo:

```
      var coche = {"marca":"Volvo","Precio":12000, "Unidades":12};
      document.write(coche['marca']+"<br>");
      // Recorre el array asociativo
      for (var pos in coche) {
            document.write(coche[pos]+"<br>");
      }
```

Declaración Array multidimensional.

Opción 1.

```
var nombre_array = new Array( new Array(valor11, valor12,..),
                              new Array(valor21, valor22,...),
                              . . .);
```

Opción 2.

```
    var nombre_array = [[valor11, valor12], [valor21,valor22],...]
```

```
<script type="text/javascript">
      var alumnos = [["Luis",7.5],["Pedro", 3.2],["Ana",8.3]];
      document.write("Alumno:"+alumnos[1][0]+"<br>");
      document.write("Nota:"+alumnos[1][1]+"<br>");
</script>
```

Operador typeof.

El operador typeof(variable) devuelve el tipo de la variable.

```
      <script language=javascript>
            var var1;
            document.write(typeof(var1)+"<br>"); // tipo undefined
            var var1 = 5;
            document.write(typeof(var1)+"<br>"); // tipo number
            var var1 = "Curso Javascript";
            document.write(typeof(var1)+"<br>"); // tipo string
            var var1 = 1215.56;
            document.write(typeof(var1)+"<br>"); // tipo number
            var var1 = true;
            document.write(typeof(var1)+"<br>"); // tipo boolean
      </script>
```

Definir Constantes.

Una constante permite almacenar un valor que no variará a lo largo de la ejecución de la página.

```
const nombre_constante = valor;
```

Ejemplo:

```
const iva = 0.21;
```

Formatear valor numérico.

Podemos formatear valor numérico en formato euro utilizando:

```
Intl.NumberFormat("es-ES",{style:"currency",currency:"EUR"}).format(variable)
```

Ejemplo:

```
var precio = 1545.678;
document.write("Precio "+Intl.NumberFormat("es-ES",
      {style:"currency",currency:"EUR"}).format(precio)+"<br>");
```

2.6.- Operadores.

JavaScript utiliza principalmente cinco tipo de operadores:

- Aritméticos.
- Lógicos.
- De asignación.
- De comparación.
- Condicionales.

Operadores aritméticos.

Permiten realizar cálculos elementales entre variables numéricas.

Operador	Nombre
+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo
++	Incremento
--	Decremento

Ejemplos:

```
var num1 = 8, num2 = 5;
suma = num1 + num2;          // 13
resta = num1 - num2;         // 3
producto = num1 * num2; // 40
division = num1 / num2; // 1.6
resto = num1 % num2;         // 3
var num1 = 5;
var num2 = 12;
num1++;          // 6
num2--;          // 11
```

Operadores lógicos.

Combinan diferentes expresiones lógicas con el fin de evaluar si el resultado de dicha combinación es verdadero o falso.

Operador	Nombre
&&	Y
||	O
!	No

Ejemplos:

```
var sueldo = 1400;
var edad = 34;
logica = (sueldo>1000 && edad <40);  //true
logica = (sueldo>1000 && edad >40);   //false
logica = (sueldo>1000 || edad >40);      //true
logica = (sueldo<1000 || edad >40);     //false
logica = !(edad <40);   //false
```

Operadores de asignación.

El operador de asignación más importante es el = , pero existen otros operadores de asignación que permiten obtener métodos abreviados para evitar escribir dos veces la variable que se encuentra a la izquierda del operador.

Operador	Nombre
+=	Suma y asigna
-=	Resta y asigna
*=	Multiplica y asigna
/*	Divide y asigna
%=	Módulo y asigna

Ejemplo:

```
var dato1 = 10, dato2 = 2, dato;
dato = dato1;    // dato vale 10
dato2 *= dato1;  // dato2 vale 20
dato2 /= dato1;  // datos2 vale 2
dato2 += dato1;  // dato2 vale 12
dato2 -= dato1;  // dato2 vale 2
dato1 %= dato2;   // dato1 vale 0
```

Operadores de comparación.

Permiten comparar todo tipo de variables y devuelve un valor booleano.

Operador	Nombre
<	Menor que
<=	Menor o igual que
==	Igual

>	Mayor que
>=	Mayor o igual que
!=	Diferente
===	Estrictamente igual (tipo y valor)
!==	Estrictamente diferente (tipo y valor)

Ejemplos:

```
var valor1 = 10;
var valor2 = 3;

compara = valor1 > valor2;   // true
compara = valor1 < valor2;    //false
compara = valor1 >= valor2;  // true
compara = valor1 <= valor2;   //false
compara = valor1 == valor2;   //false
compara = valor1 != valor2;  // true

var valor3 = 10;
var valor4 = 10.0;
compara = valor3 === valor4;  //false
compara = valor3 !== valor4;  // true
```

Operadores condicionales.

Permite indicar al navegador que ejecute una acción en concreto después de evaluar una expresión. También llamado if compacto.

Operador	Nombre
?:	Condicional

Formato:

condicion ? caso_verdad : caso_falso

Ejemplo:

```
<script type="text/javascript">
  var dividendo = prompt("Introduce el dividendo: ");
  var divisor = prompt("Introduce el divisor: ");
  var resultado;
  divisor != 0 ? resultado = dividendo/divisor :
      resultado = "No es posible la división por cero";
  alert("El resultado es: " + resultado);
</script>
```

2.7.- Sentencias Condicionales.

Permiten evaluar condiciones y ejecutar ciertas instrucciones si la condición es verdadera y ejecutar otras instrucciones si la condición es falsa.

Existen dos tipos de sentencias condicionales: sentencia if, sentencia switch.

Sentencias if.

Indica al navegador si debe ejecutar una parte de código en base al valor lógico de una expresión condicional.

En Javascript hay tres tipos de estructuras if.

a) if.

Si se cumple una condición ejecuta unas sentencias.

```
if (condicion) {
      sentencias;
}
```

Las llaves no son obligatorias en caso de que sólo se ejecute una sentencia.

b) if else.

Si se cumple una condición ejecuta unas sentencias y el caso contrario ejecuta otras.

```
if (condicion) {
      sentencias;
} else {
      sentencias;
}
```

c) if else if.

Si se cumple una condición ejecuta unas sentencias y el caso contrario comprueba otra condición ejecutando unas sentencias si se cumple y así sucesivamente.

```
if (condicion) {
      sentencias;
} else if (condicion) {
      sentencias;
} else if (condicion) {
      sentencias;
} else {
      sentencias;
}
```

Ejemplos:
Si el sueldo del empleado es mayor de 1500 euros calcular una retención del 12% sobre el sueldo.

```
var sueldo = 1600;
if (sueldo > 1500) {
      retencion = sueldo * 0.12;
      document.write("La retención es de: " + retencion);
}
```

Si el sueldo del empleado es mayor de 1500 euros calcular una retención del 12% sobre el sueldo en caso contrario calcular una retención del 5%.

```
var sueldo = 1200;
if (sueldo > 1500) {
      retencion = sueldo * 0.12;
} else {
      retencion = sueldo * 0.05;
}
document.write("La retención es de: " + retencion);
```

Si el sueldo del empleado es menor de 1200 euros calcular una retención del 12% sobre el sueldo en caso contrario si el sueldo es mayor o igual a 1200 y menor o igual a 2500 calcular una retención del 18% sobre el sueldo y si no calcular una retención del 25% sobre el sueldo.

```
var sueldo = 1800;
if (sueldo < 1200) {
      retencion = sueldo * 0.12;
} else if (sueldo >= 1200 && sueldo <= 2500){
```

```
        retencion = sueldo * 0.18;
    } else {
        retencion = sueldo * 0.25;
    }
    document.write("La retención es de: " + retencion);
```

Sentencias switch.

Compara el valor de una variable con una serie de valores conocidos. Si uno de los valores coincide con el valor de la variable se ejecuta el código asociado a dicho valor conocido.

```
switch (expresion){
    case valor1: sentencias1; break;
    case valor1: sentencias1; break;
          :     :     :     :
    case valorn: sentenciasn; break;
    [default: sentenciasdef;]
}
```

Ejemplo:
Si un empleado tiene categoría laboral 1 se le aplica un 10% de retención sobre el sueldo, si tiene categoría laboral 2 se le aplica un 15% de retención sobre el sueldo, si tiene categoría laboral 3 se le aplica un 20% de retención sobre el sueldo, si tiene categoría laboral 4 se le aplica un 25% de retención sobre el sueldo el cualquier otro caso se le aplica un 30% de retención sobre el sueldo.

```
var sueldo = 1800;
var catLaboral = 3;
var retencion;
switch (catLaboral){
    case 1: retencion = sueldo * 0.10; break;
    case 2: retencion = sueldo * 0.15; break;
    case 3: retencion = sueldo * 0.20; break;
    case 4: retencion = sueldo * 0.25; break;
    default: retencion = sueldo * 0.30;
}
document.write("La retención es de: " + retencion);
```

2.8.- Sentencias bucle.

Permiten al navegador ejecutar un fragmento de código de forma repetida mientras la condición sea verdadera. Las estructuras de repetición o bucles son utilizadas cuando unas sentencias han de ser ejecutadas cero, una o más veces.

Hay que tener cuidado con los bucles infinitos que no terminan nunca pues el programa se seguirá ejecutando y el equipo se quedará colgado.

a) Bucle while.

El bucle **while** se utiliza cuando se tiene que ejecutar un grupo de sentencias un número determinado de veces. Las sentencias podrían no llegar a ejecutarse ya que inicialmente podría no cumplirse la condición de ejecución de bucle.

```
while (expresion){
      sentencias;
}
```

Ejemplo:

```
int contador = 1;
while (contador<=10){
      document.write(contador);
      contador++;
}
```

Ejemplo:

```
var contador = 0;
poblacion = ["Valencia", "Castellón", "Alicante"];
while (contador <=2) {
      document.write(poblacion[contador]);
      contador++;
}
```

b) Bucle do ... while.

El bucle **do ... while** se utiliza cuando se tiene que ejecutar un grupo de sentencias un número determinado de veces. Las sentencias se ejecutan al menos una vez ya que la comprobación de la condición de salida del bucle se encuentra después de las sentencias del bucle.

```
do {
      sentencias;
} while (expresion);
```

Ejemplo:

```
var contador = 1;
do {
      document.write(contador);
      contador++;
} while (contador<=10);
```

Ejemplo:

```
var contador = 0;
var poblacion = ["Valencia", "Castellón", "Alicante"];
do {
      document.write(poblacion[contador]+"<br>");
```

```
        contador++;
    } while (contador<=2);
```

c) Bucle for.

El bucle **for** se utiliza cuando se tiene que ejecutar un grupo de sentencias un número fijo y conocido de veces. La sentencia **for** se puede conseguir utilizando el bucle while.

```
for(inicializacion; condicion; incremento){
    sentencias;
}
```

Ejemplo:

```
var contador;
for(contador = 1; contador<=10; contador++){
    document.write(contador+"<br>");
}
```

Ejemplo:

```
var contador;
for(contador = 10; contador>=1; contador--){
    document.write(contador+"<br>");
}
```

Ejemplo:

```
var contador;
for(contador = 1; contador<=9; contador+=2){
    document.write(contador+"<br>");
}
```

Ejemplo:

```
var contador;
var poblacion = ["Valencia", "Castellón", "Alicante"];
for(contador = 0; contador<=2; contador++){
    document.write(poblacion[contador]+"<br>");
}
```

c) Bucle for in.

El bucle **for** se utiliza cuando se tiene que recorrer un array de elementos.

```
for(nombre_variable in nombre_arrray){
      sentencias;
}
```

Ejemplo:

```
var numeros = [1, 5, 7, 3, 2];
for (var num in numeros) {
  document.write(numeros[num]+"<br>");
}
```

2.9.- Estructuras de salto.

a) Sentencias break y continue.

La sentencia **break** se puede utilizar tanto en estructuras de selección como en estructuras de repetición y permite salir de un bloque de sentencias.
La sentencia **continue** se puede utilizar solo en estructuras de repetición y permite saltar desde el bloque de sentencias a la sentencia de evaluación de la condición.

Ejemplo:

```
var contador = 0;
while (contador < 10){
      contador++;
      if (contador==5) {
            break;
      }
      document.write(contador+"<br>");
}
```

Ejemplo:

```
var contador = 0;
while (contador < 10){
      contador++;
      if (contador==5) {
            continue;
      }
      document.write(contador+"<br>");
}
```