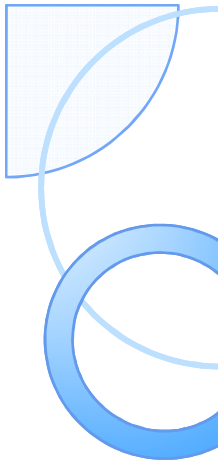
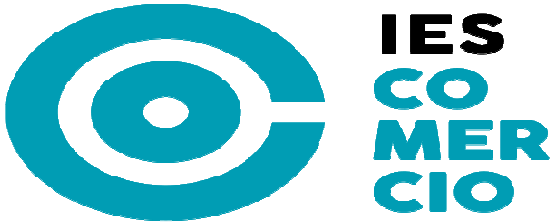




Tema 3:
Características del lenguaje JavaScript.

Desarrollo Web en Entorno Cliente

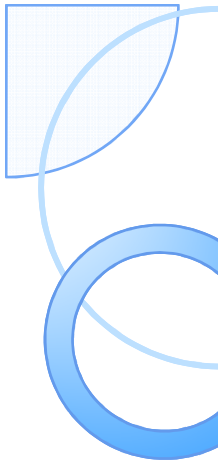
(Curso 2016 - 2017)

Objetivos

- Conocer las principales características del lenguaje JavaScript.
- Dominar la sintaxis básica del lenguaje.
- Comprender y utilizar los distintos tipos de variables y operadores presentes en el lenguaje JavaScript.
- Conocer las diferentes sentencias condicionales de JavaScript y saber realizar operaciones complejas con ellas.

Contenidos

1. Características de JavaScript.
2. El lenguaje JavaScript: Sintaxis.
3. Tipos de datos.
4. Variables.
5. Operadores.
6. Sentencias condicionales
    1. IF.
    2. SWITCH.
    3. WHILE
    4. FOR



1.- Características de JavaScript

- ¿Qué es 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.
- Sus objetos utilizan herencia basada en prototipos.
- Es un lenguaje débilmente tipado.
- Todas sus variables son globales.

2.- El lenguaje JavaScript: Sintaxis

- **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 en el código:**
  - 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.

2.- El lenguaje JavaScript: Sintaxis

- **Comentarios en el código –Ejemplo:**

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

2.- El lenguaje JavaScript: Sintaxis

- **Tabulación y saltos de línea:**
  - JavaScript ignora los espacios, las tabulaciones y los saltos de línea con algunas excepciones.
  - Emplear la tabulación y los saltos de línea mejora la presentación y la legibilidad del código.

2.- El lenguaje JavaScript: Sintaxis

- Tabulación y saltos de línea – Diferencias:

```
<script
type="text/javascript">va
r 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>
```

```
<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>
```

2.- El lenguaje JavaScript: Sintaxis

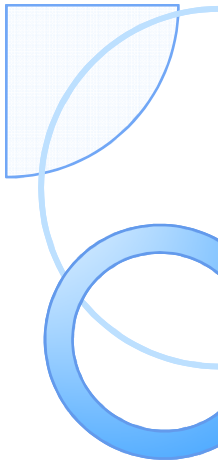
- **El punto y coma:**
  - Se suele insertar un signo de punto y coma (;) al final de cada instrucción de JavaScript.
  - Su utilidad es separar y diferenciar cada instrucción.
  - Se puede omitir si cada instrucción se encuentra en una línea independiente (la omisión del punto y coma no es una buena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

2.- El lenguaje JavaScript: Sintaxis

- **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.

3.- 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.



3.- Tipos de datos

- **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++.

3.- Tipos de datos

- **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.

3. -Tipos de datos

- **Cadenas de texto – Secuencias de escape:**

Secuencia de escape	Resultado
\\	Barra invertida
\'	Comilla simple
\"	Comillas dobles
\n	Salto de línea
\t	Tabulación horizontal
\v	Tabulación vertical
\f	Salto de página
\r	Retorno de carro
\b	Retroceso

3. -Tipos de datos

- **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.

4.- Variables

- Se pueden definir como zonas de la memoria de un ordenador que se identifican con un nombre y en las cuales se almacenan ciertos datos.
- El nombre puede incluir cualquier tipo de caracteres Unicode. Deben comenzar con un carácter alfabético, guion bajo o dólar.
- El desarrollo de un script conlleva:
  - Declaración de variables.
  - Inicialización de variables.

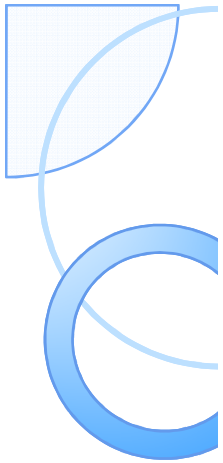
4.- Variables

- 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:");

4.- Variables

- Declaración de variables:
  - Declaración explícita: 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;
  - Declaración implícita: se asigna directamente un valor a un identificador. Declara siempre variables globales. No es recomendable declarar este tipo de variable.
    - mi_variable = “Hola”



4.- Variables

- Evaluación de variables:
  - Una variable declarada mediante var sin valor inicial tiene el valor `undefined`.
  - Acceder a una variable no declarada lanza la excepción `ReferenceError`.
  - Ámbito de variables:
    - Cualquier valor declarado de forma explicita dentro de una función tiene ámbito local.
    - Cualquier valor declarado fuera de una función tiene ámbito global.

4.- Variables

- Conversión de tipos de datos:
  - Al ser JavaScript un lenguaje dinámico:
    - No es necesario especificar el tipo de una variable.
    - El tipo del contenido de una variable puede cambiar a lo largo de la ejecución del script.
    - El tipo de las variables se puede convertir cuando sea necesario.

```
var dato = 42 //dato es numérico
dato = “Madrid” //Ahora dato es una cadena
//En una expresión con datos numéricos y de cadena
// el operador + convierte los datos a cadena.
dato = “Edad: ” + 30 //dato es “Edad: 30
//Si la expresión incluye otros operadores convierte las
//cadenas a número
dato = “28” – 2 //dato es 26
dato = “28” + 2 //dato es “282”
//Si no puede hacer la conversión a número devuelve NaN
dato = “Madrid” – 2 //Devuelve NaN (Not a Number)
```

Actividad 1.1

Realizar los ejercicios propuestos donde se trabajan los de tipos de datos y variables en JavaScript.

5.- Operadores

- JavaScript utiliza principalmente cinco tipo de operadores:
  - Aritméticos.
  - De asignación.
  - De cadenas.
  - Lógicos.
  - De comparación.
  - Condicionales.

5.- Operadores

- **Operadores aritméticos:**
  - Permiten realizar cálculos elementales entre variables numéricas.

Operador	Descripción	Ejemplo
+	Suma aritmética	3+4 devuelve 7
-	Resta	5-2 devuelve 3
*	Multiplicación	4*2 devuelve 8
/	División real (no existe la división entera)	5/2 devuelve 2.5
% (Módulo)	Operador binario. Devuelve el resto de la división entera entre sus dos operandos.	12 % 5 devuelve 2.
++ (Incremento)	Operador unitario. Suma uno a su operando. Si se usa como prefijo (++x), devuelve el valor de su operando después de la suma; si se usa como sufijo (x++), devuelve el valor de su operando antes de sumarle uno.	Si x es 3, entonces ++x establece x a 4 y devuelve 4, mientras que x++ establece x a 4 y devuelve 3.
-- (Decremento)	Operador unitario. Resta uno a su operando. Su funcionamiento es análogo al del operador de incremento.	Si x es 3, entonces --x establece x a 2 y devuelve 2, mientras que x-- establece x a 2 y devuelve 3.
- (Cambio de signo)	Operador unitario. Devuelve su operando cambiado de signo.	Si x es 3, entonces -x devuelve -3.

5.- Operadores

- **Operadores de asignación:**
  - 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

5.- Operadores

- **De cadenas:**
  - Concatena dos cadenas.

+ concatena dos cadenas, pega la segunda cadena a continuación de la primera.

Si hacemos la operación mezclando cadenas y números, lo que hace este operador es tratar a todos como si fuesen cadenas.

5.- Operadores

- **Operadores lógicos:**
  - Combinan diferentes expresiones lógicas con el fin de evaluar si el resultado de dicha combinación es verdadero o falso.

Operador	Uso	Descripción
&&	expr1 && expr2	(AND lógico) Devuelve expr1 si la expresión puede convertirse a falso; de otro modo, devuelve expr2. Cuando se emplea con valores booleanos, && devuelve true cuando ambos operandos son verdaderos; si no, devuelve false.
||	expr1 || expr2	(OR lógico) Devuelve expr1 si puede convertirse a verdadero; de otro modo devuelve expr2. Cuando se emplea con valores booleanos, el operador || devuelve true si alguno de los operandos es verdadero; si ambos operandos son falsos devuelve false.
!	!expr	(NOT lógico) Devuelve falso si su único operando puede convertirse a verdadero; de otro modo, devuelve verdadero.

5.- Operadores

- **Operadores de comparación:**
  - Permiten comparar todo tipo de variables y devuelve un valor booleano.

Operador	Descripcion	Ejemplos que devuelven verdadero[1]
Igual (==)	Devuelve true si los operandos son iguales.	3 == var1 "3" == var1 3 == '3'
Distinto (!=)	Devuelve true si los operandos no son iguales.	var1 != 4 var2 != "3"
Igual estricto (===)	Devuelve true si los operandos son iguales y del mismo tipo.	3 === var1
Distinto estricto (!==)	Devuelve true si los operandos no son iguales y/o no son del mismo tipo.	var1 !== "3" 3 !== '3'
Mayor que (>)	Devuelve true si el operando izquierdo es mayor que el derecho.	var2 > var1 "12" > 2
Mayor o igual que (>=)	Devuelve true si el operando izquierdo es mayor o igual que el derecho.	var2 >= var1 var1 >= 3
Menor que (<)	Devuelve true si el operando izquierdo es menor que el derecho.	var1 < var2 "12" < "2"
Menor o igual que (<=)	Devuelve true si el operando izquierdo es menor o igual que el derecho.	var1 <= var2 var2 <= 5

5.- Operadores

- Operadores condicionales:
  - Permite indicar al navegador que ejecute una acción en concreto después de evaluar una expresión.

Operador	Nombre
?:	Condicional

  - Ejemplo:

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

Actividad 1.2

Realizar los ejercicios propuestos sobre los operadores.

6.- Estructuras de control.

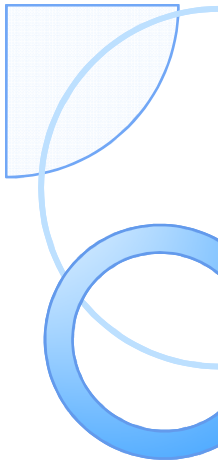
- Nos permiten controlar el flujo de nuestros programas.
- Con ellas podemos realizar toma de decisiones y bucles.
  - ***Toma de decisiones***: sirven para realizar unas acciones u otras en función del estado de las variables.
    - if.
    - switch.
  - ***Bucles***: sirven para realizar ciertas acciones repetidamente.
    - while.
    - for.
    - do while.

6.1.- Sentencias condicionales

- **if–sintaxis (1):**

```
if (expresión) {
   instrucciones
}
```

La expresión a evaluar se coloca siempre entre paréntesis y está compuesta por variables que se combinan entre sí mediante operadores condicionales.
Las expresiones condicionales se pueden combinar con las expresiones lógicas para crear expresiones más complejas.



6.1.- Sentencias condicionales

- **if–sintaxis (2):**

```
if (expresión) {
        instrucciones_si_true
} else {
        instrucciones_si_false
}
```

6.1.- Sentencias condicionales

```
if (credito >= precio)
 {
    //enseño compra
     document.write("has comprado el artículo " + nuevoArtículo)
    //introduzco el artículo en el carrito de la compra
     carrito += nuevoArticulo
    //disminuyo el crédito según el precio del artículo
     credito -= precio
 }
  else
 {
    //informo que te falta dinero
     document.write("se te ha acabado el crédito")
    //voy a la página del carrito
     window.location = "carritodelacompra.html“
 }
```

6.1.- Sentencias condicionales

- **if–sintaxis (3):**

```
if (expresión_1) {
        instrucciones_1
} else if (expresión_2) {
        instrucciones_2
} else {
        instrucciones_3
}
```

6.1.- Sentencias condicionales

```
var numero1=23
var numero2=63
if (numero1 == numero2)
{
    document.write("Los dos números son iguales")
}
  else
   {
    if (numero1 > numero2)
     {
       document.write("El primer número es mayor que
       el segundo")
     }
       else
        {
           document.write("El primer número es menor que
            el segundo”)
        }
    }
```

6.1.- Sentencias condicionales

- **switch–sintaxis:**

```
switch (expresión) {
 case valor1:
        instrucciones a ejecutar si expresión = valor1
 break;
 case valor2:
        instrucciones a ejecutar si expresión = valor2
 break;
 case valor3:
        instrucciones a ejecutar si expresión = valor3
 break
 default:
        instrucciones a ejecutar si expresión es diferente a
        los valores anteriores
}
```

6.1.- Sentencias condicionales

```
switch (dia_de_la_semana) {
   case 1:
      document.write("Es Lunes")
   break
   case 2:
      document.write("Es Martes")
   break
   case 3:
      document.write("Es Miércoles")
   break
   case 4:
      document.write("Es Jueves")
   break
   case 5:
      document.write("Es viernes")
   break
   case 6:
   case 7:
      document.write("Es fin de semana")
   break
   default:
      document.write("Ese día no existe")
}
```

6.1-Sentencias condicionales

- **for–sintaxis:**

  for (valor_inicial_variable;
  expresión_condicional;
  incremento_o_decremento_de_la_variable)
  {
  cuerpo_del_bucle
  }

```
for (i=1;i<=1000;i+=2)
   document.write(i)
```

Si queremos contar descendentemente del 343 al 10 utilizaríamos este bucle.

```
for (i=343;i>=10;i--)
   document.write(i)
```

6.1.- Sentencias condicionales

- **while–sintaxis:**

(1)

```
while (expresión) {
        instrucciones
}
```

```
var color = ""
while (color != "rojo")
{
     color = prompt("dame un
         color (escribe rojo
         para salir)","")
}
```

(2)

```
do {
        instrucciones
} while (expresión)
```

```
var color
do {
    color   =   prompt("dame   un
                color(escribe
             rojo para salir)","")
} while (color != "rojo")
```

6.1.- Sentencias condicionales

- **break**

  Detiene la ejecución de un bucle y se sale de el.

```
for (i=0;i<10;i++)
{
    document.write (i)
    escribe = prompt("dime si
              continuo
          preguntando...", "si");
    if (escribe == "no")
        break;
}
```

- **continue**

  Sirve para volver al principio de un bucle en cualquier momento sin ejecutar las líneas que hay debajo de continue.

```
var i=0
while (i<7)
{
    incrementar   =   prompt("La
          cuenta está en " + i
          +     ",     dime     si
        incremento", "si");
    if (incrementar == "no")
       continue;
    i++;
}
```

Actividad 1.3

Realizar los ejercicios de estructuras de control propuestos.