INTRODUCCIÓN A JAVASCRIPT

LENGUAJES INTERPRETADOS EN EL CLIENTE

¿QUÉ ES JAVASCRIPT?

JavaScript es un lenguaje de programación usado en páginas y aplicaciones web como complemento a HTML que escribe el contenido de las páginas en el navegador y les agrega dinamismo, interactividad, animación y efectos. JavaScript funciona desde el navegador, por lo que muchas funciones están disponibles en las páginas, aun sin conexión.

VENTAJAS DE JAVASCRIPT

Interacción mínima con el servidor

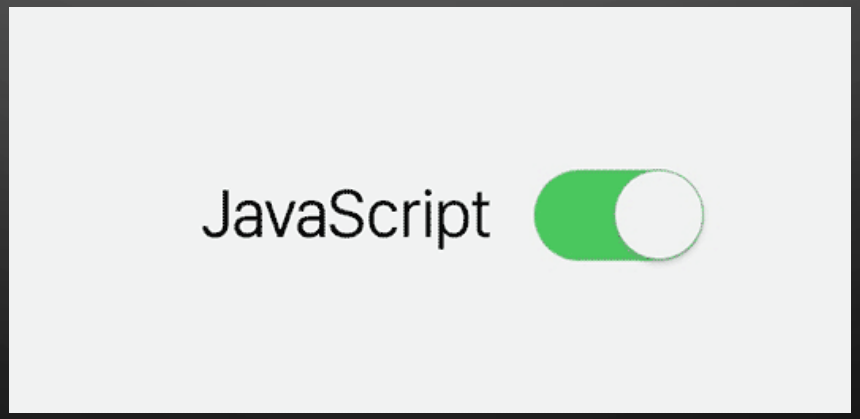
Valida la información que ingresa el usuario en el lado del cliente. Envía solicitudes al servidor después de ejecutar las comprobaciones de validación iniciales

INTERFACES MÁS COMPLETAS Y FÁCILES DE USAR

- Por ejemplo se puede agregar controles deslizantes, presentaciones de diapositivas, efectos al poner el cursor sobre objetos, funciones de arrastrar y soltar, etc

RETROALIMENTACIÓN INMEDIATA PARA EL USUARIO Y FÁCIL DEPURACIÓN

- Manda alertas al usuario
- JavaScript es un lenguaje interpretado, lo que significa que el código escrito se descifra línea por línea. En caso de que aparezcan errores, se obtendrá el número de línea exacto donde se encuentra el problema.



FORMAS DE INSERTAR JAVASCRIPT EN HTML

1. **Agregar JavaScript directamente a un archivo HTML, utilizando la etiqueta <script> </script>**
    - **entre las etiquetas <head>**
    - **entre las etiquetas <body>**
2. **Utilizando los eventos de las etiquetas HTML**
3. **Agregar código JavaScript a un archivo separado**

```
<head>
...
...
<script type="text/javascript" src="/rutaDelArchivo1.js">
</script>
<script type="text/javascript" src="/rutaDelArchivo2.js">
</script>
<script type="text/javascript" src="/rutaDelArchivo3.js">
</script>
...
...
</head>
```

AGREGAR JAVASCRIPT DIRECTAMENTE A UN ARCHIVO HTML

- Las etiquetas a utilizar son: <script> </script> que debe envolver todo el código JS. Estas pueden estar entre las etiquetas <head> y <body>

```
<script>
 alert('Mensaje');
</script>
```

EJEMPLO 1 ENTRE LAS ETIQUETAS BODY

java.html

```
1  <!DOCTYPE html>
2  <html lang="en-US">
3  <head>
4  <meta charset="UTF-8">
5  <meta name="viewport" content="width=device-width, initial-scale=1">
6  <title>Fecha actual </title>
7  </head>
8  <body>
9  <script>
10 var d=new Date();
11 document.write('Hoy es:');
12 document.write(d.getDate());
13 </script>
14 </body>
15 </html>
```

```
<script language="Javascript">

</script>
```

Los valores que se incluyen en el atributo type están estandarizados y para el caso de JavaScript, el valor correcto es text/javascript

EJEMPLO 2

```
<html>
    <head>
    </head>
    <body>
    <a href="http://www.udb.edu.sv/" onclick="alert('Página de la UDB')">UDB</a>
    </body>
    </html>
```

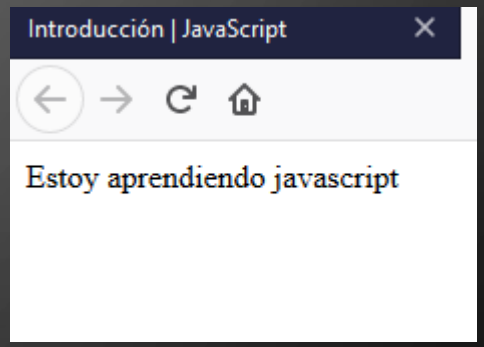
Este método es el menos utilizado, ya que consiste en incluir instrucciones JavaScript dentro del código HTML .

Este método sólo se utiliza para definir algunos eventos y en algunos otros casos especiales

EJEMPLO ARCHIVO EXTERNO

java.html

```
1
2
3   <!DOCTYPE html">
4   <head>
5       <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
6       <title>Ejemplo de código JavaScript </title>
7       <script type="text/javascript">
8         console.log("Logica de programacion");
9       </script>
10  </head>
11  <body>
12      <p>Estoy aprendiendo javascript</p>
13  </body>
14  </html>
15
16
```



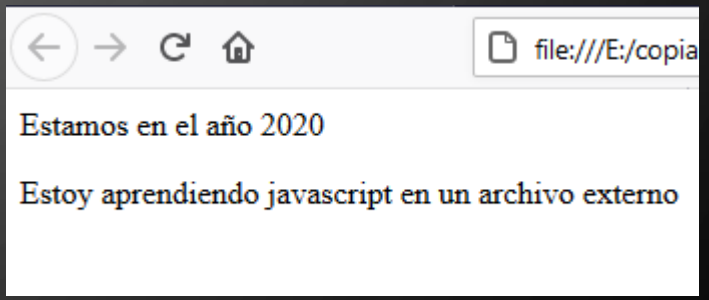
DEFINIR JAVASCRIPT EN UN ARCHIVO EXTERNO

go.js | java1.html

```
<!DOCTYPE html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Ejemplo de código JavaScript </title>
    <script type="text/javascript" src="codigo.js">
    </script>
</head>
<body>
    <p>Ejemplo de javascript en un archivo externo</p>
</body>
</html>
```

codigo.js

```
1  var d=new Date();
2  document.write('Estamos en el año ');
3  document.write(d.getFullYear());
4  console.log("Estoy en clase de Lenguajes interpretados en el cliente");
```



FUNCIONES DE JAVASCRIPT PARA FECHA Y HORA

getDate()	Devuelve el dia del mes (entre el 1 y el 31)
getDay()	Devuelve el dia de la semana (entre el 0 y el 6)
getMonth()	Devuelve el mes (entre el 0 y el 11)
getFullYear()	Devuelve el año (en formato de 4 digitos)
getHours()	Devuelve la hora (entre el 0 y el 24)
getMinutes()	Devuelve los minutos (desde 0 a 59)
getSeconds()	Devuelve los segundos (desde 0 a 59)
getTime()	Devuelve el número de milisegundos desde el 1ro de Enero de 1970
getTimezoneOffset()	Devuelve la diferencia de horario en minutos entre la hora local y GMT (Meridiano de Greenwich)
getUTCHours()	Devuelve la hora de acuerdo a UTC (Tiempo Universal Coordinado)

Los archivos de tipo JavaScript son documentos normales de texto con la extensión .js, que se pueden crear usando un editor de texto.
La principal ventaja de enlazar un archivo JavaScript externo es que se simplifica el código HTML de la página, que se puede reutilizar el mismo código JavaScript en todas las páginas del sitio web y que cualquier modificación realizada en el archivo JavaScript se ve reflejada inmediatamente en todas las páginas HTML que lo enlazan.

GLOSARIO BASICO

•**Script**: programas, aplicaciones o trozos de código creados con el lenguaje de programación JavaScript.
•**Sentencia**: cada una de las instrucciones que forman un script.
•**Palabras reservadas**: son las palabras (en inglés) que se utilizan para construir las sentencias de JavaScript .

Las palabras actualmente reservadas por JavaScript son: break, case, catch, continue, default, delete, do, else, finally, for, function, if, in, instanceof, new, return, switch, this, throw, try, typeof, var, void, while, with.

NORMAS QUE SIGUE JAVASCRIPT

- No se tienen en cuenta los espacios en blanco y las nuevas líneas
- Se distinguen las mayúsculas y minúsculas
- No se define el tipo de las variables: al crear una variable, no es necesario indicar el tipo de dato que almacenará. De esta forma, una misma variable puede almacenar diferentes tipos de datos durante la ejecución del script.
- No es necesario terminar cada sentencia con el carácter de punto y coma Aunque JavaScript no obliga a hacerlo, es conveniente seguir la tradición de terminar cada sentencia con el carácter del punto y coma (;).

•Se pueden incluir comentarios: Aunque el contenido de los comentarios no se visualiza por pantalla, si que se envía al navegador del usuario junto con el resto del script, por lo que es necesario extremar las precauciones sobre la información incluida en los comentarios.
JavaScript define dos tipos de comentarios: los de una sola línea y los que ocupan varias líneas.

// un mensaje

/* varios comentarios*/

- **Palabra clave**

  Son palabras que son parte del lenguaje en sí. Las palabras clave no se pueden utilizar como identificadores. Ejemplos: *if*, *while*, *function*, *var*, etc.

- **Palabra reservada**

  Son palabras que podrían convertirse en parte del lenguaje en sí. Las palabras reservadas tampoco deberían ser utilizadas como identificadores. A veces esta restricción no es total como en el caso de las palabras claves.

CONVERSIÓN DE TIPOS

- **JavaScript**, a diferencia de lenguajes de programación como Java, C++ o Perl, es un **lenguaje de tipología débil**.
- Esto significa que el tipo de dato de las variables en JavaScript se infiere del valor almacenado y no de una declaración de tipo preestablecida.
- La consecuencia de esta inferencia es que el tipo de dato de una variable puede cambiar de forma dinámica.
- Esto es tanto una ventaja como un aspecto que debe llamar a la precaución de los programadores porque en cierta forma introduce ambigüedad en algunos cálculos matemáticos sobre todo.

CONVERSIÓN DE TIPOS

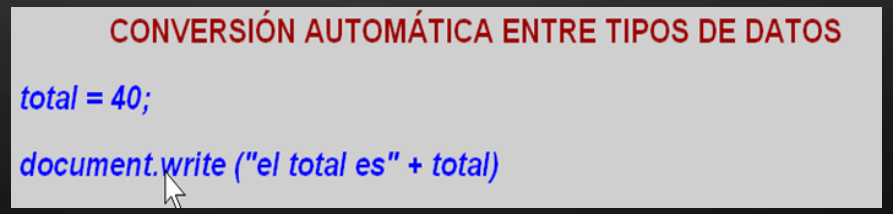
- La **conversión automática de tipos** es una característica de JavaScript que permite que el tipo de dato de una variable cambie de forma automática cuando cambia el tipo de valor almacenado en esta.

- Por ejemplo, en las siguientes instrucciones:

```
x = "7";    //literal de cadena "7"

frase = x + " caballos";  //frase contendrá 7 caballos

resta = x – 5;  //resta contendrá 2
```



OPERADORES DE JAVASCRIPT

- Operadores de comparación
- Operadores aritméticos
- Operadores de asignación
- Operadores de incremento/decremento
- Operadores lógicos y a nivel de bits
- Operadores de cadena de caracteres
- Operador condicional ternario

OPERADORES DE COMPARACIÓN

Operador	Descripción
==	Igualdad de comparación. La comparación se realiza con ajuste de tipos si es necesario.
!=	Desigualdad (no igualdad) de comparación. Ajustando tipos si es necesario.
===	Igualdad de comparación absoluta. Devuelve true si los elementos representan el mismo valor y además son del mismo tipo de dato.
!==	No igualdad absoluta.
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que

OPERADORES ARITMÉTICOS

Operador	Descripción
+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo. Resto de una división de dos enteros.
++ (v)	Operador incremento. Incrementa en una unidad el valor de una variable.
-- (v)	Operador decremento. Decrementa en una unidad el valor de una variable.
- (v)	Negación. Cambia el signo del valor al que acompaña.

OPERADORES DE ASIGNACIÓN

Operador	Descripción
x = y	Asigna a 'x' el valor de 'y'
x += y	Suma con asignación. x = x + y
x -= y	Resta con asignación. x = x - y
x *= y	Multiplicación con asignación. x = x * y
x /= y	División con asignación. x = x / y
x %= y	Módulo con asignación. x = x % y

OPERADORES DE INCREMENTO Y DECREMENTO

Operador	Descripcion
++	Operador de incremento que se utiliza para incrementar el valor de una variable en una unidad
--	Operador de decremento utilizado para decrementar el valor de una variable en una unidad
var++	Operador de post-incremento. El valor de la variable var se asignará y luego se incrementará.
var--	Operador de post-decremento. El valor de la variable var se asignará y luego se decrementará
++var	Operador de pre-incremento. El valor de la variable se incrementa y luego se asigna
--var	Operador de pre-decremento. El valor de la variable se decrementa y luego se asigna

OPERADORES LÓGICOS Y A NIVEL DE BITS

Operador	Descripcion
x && y	Operador 'Y' (AND). Devuelve 'true' si son 'true' 'x' y 'y‘
x || y	Operador 'O' (OR). Devuelve 'true' si alguno de los operandos es ‘true'.
!x	Operador 'NO' (NOT). Negación del operando.

Operador	Descripcion
x & y	Operación AND a nivel de bits
x | y	Operación OR a nivel de bits
x ^ y	Operación XOR a nivel de bits
~ x	Operación NOT. Negación (inversión) a nivel de bits
x << y	Desplazamiento a la izquierda de los bits de 'x' tantas posiciones como indique 'y'. Se rellena por la derecha con ceros.
x >> y	Desplazamiento a la derecha de los bits de 'x' tantas posiciones como indique 'y'. Se conserva el signo.
x >>> y	Desplazamiento a la derecha de los bits de 'x' tantas posiciones como indique 'y'. Se rellena por la izquierda con ceros.

OPERADORES DE CADENA DE CARACTERES

Operador	Descripción
+	Concatena dos cadenas uniéndolas. Tiene mayor precedencia que el operador + utilizado para realizar sumas aritméticas
+=	Concatenación y asignación. Primero se concatena la cadena de la derecha a la variable actual (en la izquierda) y luego se asigna el resultado a esta misma variable.

Operador condicional ternario

- Es un operador con tres operandos. Las sintaxis es la siguiente:

```
condicion ? expresion1 : expresion2
```

PRECEDENCIA DE OPERADORES

Tipo de operador	Operadores individuales
miembro	. []
llamada/crear instancia	() new
negación/incremento	! ~ - + ++ -- typeof void delete
multiplicación/división	* / %
suma/resta	+ -
desplazamiento de bits	<< >> >>>
relacionales	< <= > >= in instanceof
igualdad	== != === !==
and a nivel de bits	&
xor a nivel de bits	^
or a nivel de bits	|
and lógico	&&
or lógico	||
condicional	?:
asignación	= += -= *= /= %= <<= >>= >>>= &= ^= |=
coma	,

OPERADORES ESPECIALES

- **new**: este operador crea una nueva instancia de un objeto
- **delete**: este operador permite borrar un objeto, una propiedad de un objeto, o un elemento especificado de un arreglo
- **void**: especifica una expresión que será evaluada sin devolver ningún valor. Se utiliza mucho en pseudo-enlaces
- **this**: se utiliza principalmente dentro de la definición de un objeto para referirse al objeto actual
- **typeof**: devuelve una cadena con el tipo de dato del operando que se envía como argumento

CONTROL DE FLUJO DE PROGRAMA

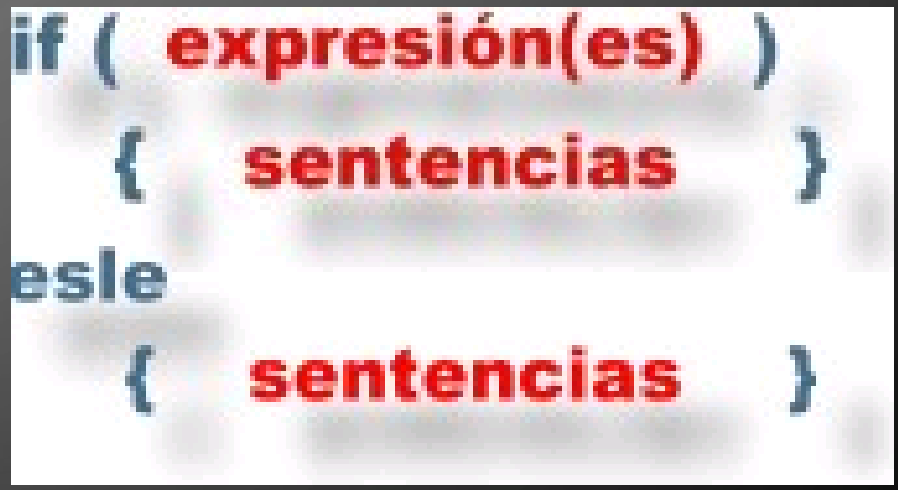
SENTENCIAS CONDICIONALES

CONTENIDO A DESARROLLAR

1. Sentencias condicionales.
    1. Sentencia if simple
    2. Sentencia if-else
    3. Sentencia if-else if-else
2. Consideraciones sobre la sentencia if.
3. Utilización de la sentencia switch.
4. El operador condicional.

SENTENCIAS CONDICIONALES

- Son estructuras de control utilizadas para poder decidir cuáles instrucciones del script se deben ejecutar dependiendo del valor devuelto por una expresión condicional.



SENTENCIAS CONDICIONALES

- Existen dos sentencias condicionales en JavaScript que son el ***if*** y el ***switch***. La sentencia ***if*** más simple únicamente lleva la palabra reservada ***if***, una sentencia ***if*** más compleja se utiliza junto con el ***else***.
- Adicionalmente, se tiene una sentencia ***if*** que puede incluir, uno o varios ***else if***, y terminar, opcionalmente, con un ***else***.

SENTENCIA *IF*

- Una sentencia if simple es una sentencia con la siguiente estructura sintáctica:

```
if(condicion) sentencia;
```

O bien,

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

La primera forma puede utilizarse únicamente si el bloque de instrucciones *if* posee una sola instrucción. En el caso de la segunda, puede utilizarse sin importar cuántas sentencias componen el bloque de instrucciones.

SENTENCIA *IF-ELSE*

- Con una sentencia condicional ***if-else*** puede crear dos bloques de instrucciones, de los cuales, uno se ejecutará si la condición es evaluada verdadera. El otro se ejecutará si la condición se evalúa como falsa. Pero nunca se podrá ejecutar ambos.
- En caso de tener múltiples bloques de instrucciones que dependan de condiciones diferentes puede utilizar una estructura ***if-else if-else***

SINTAXIS DE LA SENTENCIA *IF-ELSE IF-ELSE*

- La sentencia *if-else* tiene la siguiente sintaxis:

```
if (condición)[{]
    sentencia_1;
[}else{]
    sentencia_2; }]
```

- La sentencia *if-else if-else* tiene una sintaxis similar

```
if (condición 1)[{]
    sentencia_1;
[}else if (condición 2){
     sentencia_2;
 ...
 else sentencia_3;}]
```

CONSIDERACIONES SOBRE LA SENTENCIA *IF-ELSE*

- La condición en una sentencia ***if*** puede ser cualquier expresión lógica sintácticamente bien construida que devolverá únicamente verdadero (*true*) o falso (*false*).
- La sentencia ***if*** puede ir sola, no necesariamente con un ***else*** o un ***else if***. En ese caso se tendrá la sentencia ***if*** más simple posible.
- Es importante tener presente que JavaScript considera equivalentes a *false* los siguientes valores: 0, "", *null* y *undefined*. Cualquier otro valor será considerado verdadero al igual que *true*.

SENTENCIA *SWITCH*

- Es una sentencia condicional múltiple que permite evaluar una expresión y comparar el resultado de esta con las etiquetas de varios casos de prueba. Si existen coincidencia entonces se ejecuta el bloque de instrucciones correspondiente a ese caso de prueba.
- Los valores de prueba se colocan en sentencias case.
- Si no se da coincidencia con ninguno de los valores de los ***case*** se ejecuta lo que se incluya en la sentencia ***default***.

SINTAXIS DEL *SWITCH*

```
switch (expresion) {
    case etiqueta_1:
        sentencias_1;
        break;
    case etiqueta_2:
        sentencias_2;
        break; ...
  default:
        sentencias_por_defecto;
        break;
}
```

EFECTO DE CAÍDA EN CASCADA EN UN *SWITCH*

- Una sentencia ***switch*** puede producir un efecto de caída en cascada si queremos que dos valores de prueba de ***case*** diferentes ejecuten instrucciones similares.

```
case 'A':
case 'B':
     alert("Excelente nota");
     break;
```

EVALUACIÓN DE EXPRESIONES EN CORTOCIRCUITO

- Cuando tenemos la presencia de operadores lógicos como el AND (&&) y el OR (||) resulta que no siempre es necesario evaluar los dos operandos del operador.
- Si la expresión a la izquierda de un AND da falso, no importa el valor devuelto por la expresión de la derecha. Siempre será falso. Por lo tanto, no es necesario evaluarlo.
- Si la expresión a la izquierda de un OR resulta verdadera, toda la expresión será verdadera sin importar el valor devuelto por la expresión a la derecha.

OPERADOR CONDICIONAL

- El operador condicional, representado por ?: se utiliza para crear una línea condicional rápida.
- La sintaxis de este operador es la siguiente:

```
(expresion) ? Instrucción si true : instrucción si false;
```

- Donde, expresión es cualquier expresión que se pueda evaluar como verdadero (*true*) o como falso (*false*).

OPERADOR CONDICIONAL

- Si la expresión condicional es true se ejecutar la instrucción a la izquierda de los dos puntos.
- Si la expresión condicional se evalúa false, entonces se ejecutará la instrucción a la derecha de los dos puntos.
- Básicamente, el operador condicional ?: es una manera abreviada de escribir una sentencia condicional ***if-else***.

BIBLIOGRAFÍA

- David Flanagan. JavaScript La Guía Definitiva. Editorial Anaya Multimedia O'Reilly. 2007. Madrid, España.
- Terry McNavage. JavaScript Edición 2012. Editorial Anaya Multimedia /
- Tom Negrino / Dori Smith. JavaScript & AJAX Para Diseño Web. Editorial Pearson, Prentice Hall. Traducción de la 6ta Edición. Madrid España, 2007.
- Thomas Powell / Fritz Schneider. JavaScript Manual de referencia. Editorial McGraw-Hill Osborne Media. 1ra Edición. 2002. Madrid, España.