Programación en JavaScript

1. Introducción a JavaScript

1.1 ¿Qué es JavaScript?

Es un lenguaje de programación interpretado, inspirado en Java, que se incluye en los documentos HTML para añadir cierta interacción a sus contenidos, evitando tener que realizar programación en el servidor. Es utilizado para crear pequeños programas encargados de realizar acciones dentro del ámbito de una página web.

Tiene múltiples propósitos, y hasta hace poco fue considerado como un complemento de HTML y CSS. Una de las innovaciones que ayudaron a cambiar la precepción de Javascript fueron los nuevos motores de navegación desarrollados para acelerar el procesamiento de códigos. Estos motores convirtieron el código Javascript en código de máquina con el fin de alcanzar velocidades de ejecución similares a las aplicaciones de escritorio.

Fue desarrollado por Netscape y Sun Microsystems, siendo una marca registrada de Oracle Corporation. Es usado con licencia por los productos creados por Netscape Communications y entidades actuales como la Fundación Mozilla.

Javascript es un lenguaje de programación, algo completamente diferente a HTML y CSS. HTML es un lenguaje de marcado, un código críptico que los navegadores interpretan para organizar la información y CSS se puede considerar como una hoja de estilo (aunque la nueva especificación lo ha convertido en una herramienta más dinámica). Javascript, por otra parte, es un lenguaje de código comparable a cualquier lenguaje de programación como C++ o Java. La única diferencia significativa entre la mayoría de los lenguajes de programación y Javascript es su naturaleza. Mientras que otros lenguajes modernos están orientados a objetos, Javascript es un lenguaje de código basado en prototipos, lo que, paradójicamente lo hace más centrado en objetos que cualquier otro lenguaje.

JavaScript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes.

A continuación se establece un cuadro comparativo entre Javascript y Java:

JavaScript	Java
Interpretado (no compilado) en un navegador.	Compilado en servidor antes de la ejecución en el cliente.
Basado en objetos. Usa objetos, pero no clases ni herencia.	Programación orientado a objetos. Los *applets* constan de clases objeto con herencia.
Código integrado en el código HTML.	*Applets* diferenciados del código HTML (accesibles desde las páginas HTML).
No es necesario declarar el tipo de las variables.	Necesario declarar los tipos.
Enlazado dinámico. Los objetos referenciados deben existir en tiempo de ejecución (lenguaje interpretado).	Enlazados estáticos. Los objetos referenciados deben existir en tiempo de compilación.

1.2 Ventajas

- Es un lenguaje sencillo
- Utiliza poca memoria
- Mejor portabilidad e integración
- Incorporación por defecto en los navegadores
- Librerías integradas en los navegadores
- Fácil manejo de datos
- Ligero de carga
- Fácil de integrar
- Validación de datos de un formulario en el lado del cliente
- Cientos de aplicaciones disponibles para uso

1.3 Características

- Lenguaje de programación interpretado
- Es un lenguaje limitado. Por ejemplo, no es posible escribir aplicaciones independientes en Javascript
- Las secuencias de comandos de Javascript sólo pueden ejecutarse con un intérprete (servidor web o navegador)
- Utiliza un gestos automático de memoria dinámica
- No necesita declarar los tipos de datos
- Distinción entre mayúsculas y minúsculas
- Ignora espacios en blanco
- Permite el uso de Comentarios.
- Permite utilizar funciones
- Las instrucciones deben terminar con un carácter de punto y coma (;)

1.4 Evolución

- 1995. Primeras versiones de JavaScript, todavía con nombres provisionales como Mocha, LiveScript.
- 1997. Definición del primer estándar JavaScript a cargo de ECMA International que fue denominado ECMA-262 first edition también denominado JavaScript 1.2.
- 1998. Aparición del segundo estándar JavaScript denominado ECMA-262 second edition también denominado JavaScript 1.3.
- 2000. Aparición de la especificación del estándar JavaScript denominado ECMA-262 third edition también denominado JavaScript 1.5.
- JavaScript 1.6, 1.7 y 1.8. A partir de la versión 1.6 prácticamente solo el navegador Firefox implementa soporte completo de todas sus características.
- 2011. Aparición de la especificación del estándar JavaScript denominado ECMA-262 fifth edition también denominado JavaScript 1.8.5.
- En Junio 2015, publicó el estándar ECMAScript 6 (última versión hasta la fecha) con un soporte irregular entre navegadores.

1.5 Compatibilidad con navegadores web

Navegadores de escritorio	
Internet Explorer	8.0*, 9.0†
Firefox	Actual estable, Beta, Aurora, Nightly, y versiones ESR Versión anterior estable
Chrome	Última versión estable
Safari	Última versión estable
Opera	Última versión estable‡
Navegadores en iOS	
Mobile Safari	iOS 5.x — 6.x
Navegadores en Android	
Navegador predeterminado	2.x — 4.x
Firefox	Actual estable, Beta, Aurora, y versiones Versión anterior estable
Chrome	Última versión estable

2. Fundamentos de JavaScript

Sintaxis Básica de JavaScript

Delimitar los bloques de código Javascript

```
<script>

...

</script>
```

ó

```
<script type = 'text/javascript'>

 ...

</script>

<script language="javascript">

...

 </script>  (obsoleto)
```

- Se puede utilizar cualquier combinación de los formatos de etiquetas.
- Las instrucciones finalizan con punto y coma.

¿Cómo utilizar Javascript?

- El código Javascript puede ser escrito en cualquier editor de texto.
- Los archivos creados deben ser guardados con la extensión .html.
- Se deben colocar unas etiquetas delimitadoras de inicio y fin de código Javascript: <script> </script>.
- Guarde el archivo .html, en el directorio raíz del servidor web o en un directorio dentro del mismo.
- Acceda al archivo creado utilizando el navegador.

Método Alert( )

alert( ) es un método predefinido de Javascript que genera una ventana emergente para mostrar determinada información en la pantalla. El método muestra en la ventana cualquier valor declarado entre paréntesis.

sintaxis:

```
alert ( "Mensaje al Usuario usando Javascript" );
```

Ejemplo de Primer Programa en Javascript

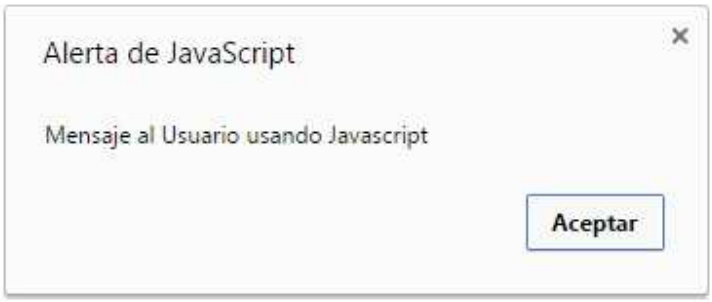
nombre: primerjs.html

sintaxis:

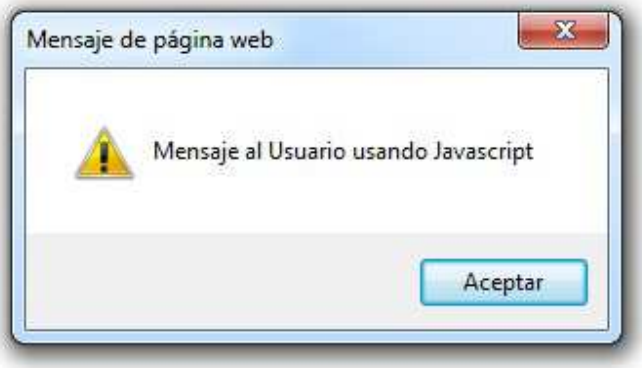
```
<!DOCTYPE html>
<html lang="es">
<head><title>Primer Programa Javascript</title>
	<meta charset='UTF-8'>
	<link rel="stylesheet" href="estilos.css">
	<link rel="icon" href="image/favicon.ico" type="image/x-icon">

	<script>
		alert ( "Mensaje al Usuario usando Javascript" );
	</script>
</head>
<body>
	<div id="caja_principal">
	</div>
</body>
</html>
```

Ventanas Javascript en Navegadores

Chrome:



Internet Explorer:



Aurora:

Safari:

2.1. Tipos de datos y operadores

Tipos de Datos

Javascript reconoce seis tipos de valores diferentes: numéricos, lógicos, objetos, cadenas, nulos e indefinidos.

TIPOS DE DATOS

Numéricos	Los enteros pueden ser expresados en base decimal, hexadecimal u octal. Ejemplos de tipo de datos numéricos: 45, 34322, -4343 (decimal) 04343, 03234, -0543 (octal) 0x32fd4 0xdd (hexadecimal)
Boleanos	Verdadero o Falso.
Cadenas	Los tipos de datos cadena deben ir delimitados por comillas simples Ejemplo: **"Un_dia";**
Objetos	objeto = **new** objeto();
Nulos	null
Indefinidos	Un valor indefindo es el que corresponde a una variable que ha sido creada pero no le ha sido asignado un valor.

Operadores

Pueden usarse para calcular o comparar valores, toman una o más variables o valores y devuelven un nuevo valor.

Los operadores están clasificados en varias clases dependiendo de la relación que determinan, éstos son: operadores aritméticos, de comparación, lógicos, de cadena, de asignación.

Operadores aritméticos

Operador	Nombre	Ejemplo	Descripción
+	Suma	5 + 6	Suma dos números.
-	Substracción	7 - 9	Resta dos números.
*	Multiplicación	6 * 3	Multiplica dos números.
/	División	4 / 8	Divide dos números.
%	Módulo: el resto después de la división	7 % 2	Devuelve el resto de dividir ambos números, en este ejemplo el resultado es 1.
++	Incremento	a++	Suma 1 al contenido de una variable.
--	Decremento	a--	Resta 1 al contenido de una variable.
-	Invierte el signo de un operando	-a	Invierte el signo de un operando.

Operadores de comparación

Operador	Descripción
==	" Igual a" devuelve *true* si los operandos son iguales.
===	Estrictamente "igual a" (JavaScript 1.3).
!=	" No igual a" devuelve *true* si los operandos no son iguales.
!==	Estrictamente " No igual a" (JavaScript 1.3).
>	" Mayor que" devuelve *true* si el operador de la izquierda es mayor que el de la derecha.
>=	" Mayor o igual que " devuelve *true* si el operador de la izquierda es mayor o igual que el de la derecha.
<	" Menor que" devuelve *true* si el operador de la izquierda es menor que el de la derecha.
<=	"Menor o igual que" devuelve *true* si el operador de la izquierda es menor o igual que el de la derecha.

Operadores lógicos

Operador	Descripción
&&	" Y " devuelve verdadero si ambos operandos son verdaderos.
||	" O " devuelve verdadero si uno de los operandos es verdadero.
!	"No" devuelve verdadero si la negación del operando es verdadero.

Operadores de cadena

Se pueden concatenar cadenas usando el operador +

Ejm:

"Programación Web " + "en Javascript"

Operadores de asignación

Operador	Descripción
=	Asigna el valor del operando de la derecha a la variable de la izquierda. Ejemplo: total=100;
+= (también - =, * =, / =)	Añade el valor del operando de la derecha a la variable de la izquierda. Ejemplo: total +=100, es lo mismo que total=total+100
&= (también |=)	Asigna el resultado de (operando de la izquierda & operando de la derecha) al operando de la izquierda.

2.2 Variables

Una variable es una posición de memoria a la que se le ha asignado un nombre para poder manejarla como un "almacén" de información.

En Javascript todos los nombre de las variables deben cumplir con las siguientes reglas:

1.- No debe empezar por Números o Símbolos.
2.- No debe contener espacios en blanco.
3.- No debe contener caracteres especiales.

Las variables permiten el uso interno únicamente del carácter guión bajo o underscore (_).

El uso de mayúsculas y minúsculas está permitido, pero se debe tener en cuenta que el intérprete las diferencia, asumiendo que son distintas variables.

Identificador	Válido
MiVariable !MiVariable Mi_Variable 3MiVariable MiVariable3 Mi,Variable	Si No Si No Si No

Ejemplo de Variables en Javascript

nombre: variables.html

sintaxis:

```
<script>
        var minumero = 2;
</script>
```

nombre: variables2.html

sintaxis:

```
<script>
        var minumero = 2;
        alert(minumero);
</script>
```

nombre: variables3.html

sintaxis:

```
<script>
    var minumero = 2;
    minumero = 3;
    alert(minumero);
</script>
```

nombre: variables4.html

sintaxis:

```
<script>
    var minumero = 2;
    minumero = minumero + 1;
    alert(minumero);
</script>
```

nombre: variables5.html

sintaxis:

```
<script>
    var mitexto = "¡Programación en Javascript!";
    alert(mitexto);
</script>
```

nombre: variables6.html

sintaxis:

```
<script>
    var mimatriz = ["rojo", "verde", "azul"];
    alert(mimatriz[0]);
</script>
```

nombre: variables7.html

sintaxis:

```
<script>
var mimatriz = ["red", , 32, null, "¡HTML5 es increible!"];
alert(mimatriz[1]);
```

```
</script>
```

nombre: variables8.html

sintaxis:

```
<script>
    var mimatriz = ["rojo", 32];
    alert(mimatriz[0]);
    mimatriz[0] = "color " + mimatriz[0];
    mimatriz[1] = mimatriz[1] + 10;
    alert(mimatriz[1] + " " + mimatriz[0]);
</script>
```

nombre: variables9.html

sintaxis:

```
<script>
    var mimatriz = ["rojo", 32];
    mimatriz.push('coche'); // Este método que agrega un nuevo valor al final de la matriz.
    alert(mimatriz[2]);
    alert(mimatriz.shift()); // Este método elimina y devuelve el último valor de la matriz.
</script>
```