DESARROLLO WEB EN ENTORNO CLIENTE

2. MANEJO DE LA SINTAXIS DEL LENGUAJE

CARÁCTERÍSTICAS DE JAVASCRIPT

¿Qué es JavaScript?

- 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

CARÁCTERÍSTICAS DE 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

CARÁCTERÍSTICAS DE JAVASCRIPT

FUNCIONALIDADES

¿Qué funcionalidades tiene Javascript? Es decir, ¿qué podemos y qué no podemos hacer con él?

Busca en Internet 5 funcionalidades.

CARÁCTERÍSTICAS DE JAVASCRIPT

Lenguaje interpretado en el navegador: puede estar deshabilitado.

No puede escribir ficheros en el servidor.

Reacciona a la interacción del usuario.

Controla múltiples ventanas, marcos, plugins, applets...

Pre-procesa datos en el cliente.

Modifica estilos y contenido de navegadores.

Puede solicitar ficheros al servidor.

FUNCIONALIDADES

CARÁCTERÍSTICAS DE JAVASCRIPT

COMPATIBILIDADES

¿Es compatible Javascript en todos los dispositivos?

¿Es soportado por todos los navegadores?

¿Se puede habilitar y deshabilitar?

CARÁCTERÍSTICAS DE JAVASCRIPT

COMPATIBILIDADES

Prácticamente todos los navegadores lo soportan: debemos asegurarnos.

Hay algunas incompatibilidades entre navegadores.

Algunos dispositivos móviles no pueden ejecutar Javascript.

Puede desactivarse la ejecución de código por el usuario.

CARÁCTERÍSTICAS DE JAVASCRIPT

SEGURIDAD Y LIMITACIONES

¿Podemos, mediante Javascript, vulnerar la seguridad de un sitio web?

¿Podemos atacar un servidor mediante Javascript?

CARÁCTERÍSTICAS DE JAVASCRIPT

SEGURIDAD

Se ejecuta el código en un “espacio seguro de ejecución”: la web.

Scripts restringidos por la política del “mismo origen”.

El motor de JavaScript es quien interpreta el código en el navegador: el responsable.

CARÁCTERÍSTICAS DE JAVASCRIPT

LIMITACIONES

- No puede modificar o acceder a las preferencias del navegador, ventana principal, impresión...
- No puede acceder al sistema de ficheros del cliente.
- No puede capturar datos de un servidor para su retransmisión.
- No puede enviar e-mails de forma invisible.
- No puede interactuar directamente con los lenguajes del servidor.
- No puede acceder a páginas almacenadas en diferentes dominios.
- No puede proteger el origen de las imágenes de la página.
- Implementar multiproceso o multitarea.

INTEGRACIÓN DE CÓDIGO JAVASCRIPT EN UNA PÁGINA WEB

Etiquetas <script> en HTML

- ¿Cómo podemos integrar un código Javascript en un HTML?

Etiquetas <script> en XHTML

- ¿Cómo podemos integrar un código Javascript en un XHTML?

Navegador no soportado

- ¿Cómo podemos advertir al usuario de que su navegador no soporta Javascript?

INTEGRACIÓN DE CÓDIGO JAVASCRIPT EN UNA PÁGINA WEB

Etiquetas <script> en HTML

- <script type="text/javascript">
  Código javascript
  </script>

Etiquetas <script> en XHTML

- <script type="text/javascript">
  <!--//--><![CDATA[//><!--
  Código javascript
  //--><!]]>
  </script>
- O Encapsulando Javascript en CDATA
  <![CDATA[
  Código
  ]]>

Navegador no soportado

- <noscript>Su navegador no soporta Javascript</noscript>

INTEGRACIÓN DE CÓDIGO JAVASCRIPT EN UNA PÁGINA WEB

JavaScript en el mismo documento HTML.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Primer programa de javascript</title>
    <script type="text/javascript">alert("Hola Mundo!");</script>
</head>
<body>

</body>
</html>
```

INTEGRACIÓN DE CÓDIGO JAVASCRIPT EN UNA PÁGINA WEB

Fichero externo

- ¿Cómo podemos integrar Javascript si se encuentra en un fichero externo (.js)?

Ventajas de usar un fichero externo

- ¿Qué ventajas tiene el uso de un fichero externo?

INTEGRACIÓN DE CÓDIGO JAVASCRIPT EN UNA PÁGINA WEB

Fichero externo

- <script type="text/javascript" src="ruta/archivo.js"></script>
- <script type="text/javascript" src="../js/archivo.js"></script>
- <script type="text/javascript" src="http://www.dominio.com/archivo.js"></script>

Ventajas de usar un fichero externo

- Carga más rápida de páginas.
- Separación entre estructura y comportamiento.
- Compartición de código entre páginas.
- Facilidad para depuración de errores.
- Modularidad.
- Seguridad.

PROTECCIÓN DE CÓDIGO JAVASCRIPT

El código en Javascript no se puede proteger: está accesible y visible a través de un navegador.

¿Qué podemos hacer para protegerlo o demostrar que ha sido elaborado por nosotros?

PROTECCIÓN DE CÓDIGO JAVASCRIPT

El código en Javascript no se puede proteger:
está accesible y visible a través de un navegador.

Introducción a Javascript

Mayúsculas y minúsculas:

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

Introducción a Javascript

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.

Introducción a Javascript

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

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

Introducción a Javascript

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).

Introducción a Javascript

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

Comentarios

- Comentarios de una línea:

```
// Esto es un comentario de una línea
//
// Hemos dejado una línea en blanco
```

- Comentarios de varias líneas:

```
/* Esto es
  un comentario
  de varias líneas */
```

- Utilización de comentarios:
  - Para que el código sea más claro e informar del uso de las funciones, variables, etc.
  - Para desactivar un bloque de código que no queremos que se ejecute.
  - Para dejar nuestra información de contacto para que otros desarrolladores contacten con nosotros.

Variables

- Crear variables utilizando la palabra reservada “var”:

```
var edad;
var edad1, edad2, edad3;
```

- Asignar valor a una variable ya creada:

```
edad1 = 15;
```

- Crear variable y asignar valor:

```
var edad1 = 15;
```

- Utilización de variables:
  - Formadas por caracteres alfanuméricos y _. No se utilizan signos, espacios, %, $, etc.
  - No pueden empezar por número, y no suelen empezar por mayúscula.
  - No tiene asociado un tipo. Podemos cambiar de número a cadena, a boolean, etc.

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++.

  - Entero: 726
  - Decimal: 3.75
  - Científico: 3e7 (30000000)
  - Octal: va precedido de un 0: 0327
  - Hexadecimal: ponemos delante 0x: 0xA3F2

Tipos de datos

Tarea 1: Números

- *Crea un programa en el que crees 5 variables numéricas (entero, decimal, científico, octal y hexadecimal).*
- *A las variables les asignarás los siguientes números: 1357, 1357, 135e7, 01357 y 0x1357.*
- *Muestra con 5 alerts su valor, escribiendo la siguiente sentencia:* `alert ("Número entero" + entero);`
- *Comenta el código indicando el nombre del ejercicio y tu nombre en la parte superior, y los comentarios adicionales que estimes necesarios.*

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 carácterde Unicode.
- La cadena de caracteres se debe definir entre comillas dobles o comillas simples.

- Cadenas:
  - “Texto entre comillas”
  - “7342”
  - “Cadenas” + “concatenadas”

- Utilización de cadenas (secuencias de escape):
  - Salto de línea: \n
  - Tabulador: \t
  - Comillas dobles: \”
  - Comillas simples: \’

Tipos de datos

Tarea 2: Cadenas

- *Crea un programa en el que crees 4 variables de tipo cadena con los siguientes valores: "Hola", "7", "13", y "Adios".*
- *Muestra en un alert una frase que incluya comillas simples.*
- *Muestra en un alert que ocupe una línea las variables 1ª y 4ª separadas por un salto de línea.*
- *Muestra en un alert la suma de las variables 2ª y 3ª.*
- *Muestra en un alert la suma de todas las variables.*
- *Comenta el código indicando el nombre del ejercicio y tu nombre en la parte superior, y los comentarios adicionales que estimes necesarios.*

Tipos de datos

Booleanos:

- true
- False

Objetos:

- String
- Date
- Array
- Etc.

Tipos de datos

- Conversión entre tiposde datos:
  - Entero + Float = Float
  - Número + Cadena = Cadena

- Conversión de cadenas a números:
  - parseInt(“32”)
  - parseFloat(“32.1”)

- Conversión de números a cadenas:
  - “” + número

Operadores

Operadores de comparación

Sintaxis	Nombre	Tipos de operandos	Resultados
==	Igualdad	Todos	Boolean
!=	Distinto	Todos	Boolean
===	Igualdad estricta	Todos	Boolean
!==	Desigualdad estricta	Todos	Boolean
>	Mayor que	Todos	Boolean
>=	Mayor o igual que	Todos	Boolean
<	Menor que	Todos	Boolean
<=	Menor o igual que	Todos	Boolean

Operadores

Tarea 3: Comparación

- *Crea un programa en el que muestres el resultado de varias operaciones mediante alert, mostrando el texto exacto de la operación realizada y su resultado.*
  - *Ej:*
    ```
    var operacion1 = 10 == 10;
    alert ("La operación 10 == 10 es" + operacion1)
    ```
- *Las operaciones a realizar son:*
  - *10 == 10*
  - *10 === 10*
  - *10 === 10.0*
  - *"Laura" == "laura"*
  - *"Laura" > "laura"*
  - *"Laura" < "laura"*
  - *"123" == 123*
  - *"123" === 123*
  - *parseInt("123") === 123*
- *Comenta el código indicando el nombre del ejercicio y tu nombre en la parte superior, y las conclusiones que sacas al realizar cada una de las operaciones.*

Operadores

Operadores aritméticos

Sintaxis	Nombre	Tipos de operandos	Resultados
+	Más	Entero, real, cadena	Entero, real, cadena
-	Menos	Entero, real	Entero, real
*	Multiplicación	Entero, real	Entero, real
/	División	Entero, real	Entero, real
%	Módulo	Entero, real	Entero, real
++	Incremento	Entero, real	Entero, real
- -	Decremento	Entero, real	Entero, real
+valor	Positivo	Entero, real, cadena	Entero, real
-valor	Negativo	Entero, real, cadena	Entero, real

Operadores

Operadores de asignación

Sintaxis	Nombre	Ejemplo	Significado
=	Asignación	x=y	x=y
+=, -=, *=, /=, %=	Operación y asignación	x+=y	x=x+y
<<=	Desplazar bits a la izquierda	x<<=y	x=x<<y
>=, >>=, >>>=	Desplazar bits a la derecha	x>=y	x=x>y
&=	Operación AND bit a bit	x&=y	x=x&y
|=	Operación OR bit a bit	x|=y	x=x|y
^=	Operación XOR bit a bit	x^=y	x=x^y
[]=	Desestructurar asignaciones	[a,b]=[c,d]	a=c, b=d

Operadores

Operadores booleanos

Sintaxis	Nombre	Operandos	Resultados
&&	And	Boolean	Boolean
||	Or	Boolean	Boolean
!	Not	Boolean	Boolean

La operación AND solo es true cuando todos los operadores son true.
La operación OR es true siempre que haya un operador true.
La operación NOT cambia el valor del boolean resultado

Ver otras operaciones a nivel de bits

Operadores

Operadores de objeto

- Punto:
  - Objeto.propiedad
  - Objeto.método

- Corchetes:
  - Crear un array: var a = [“Bizkaia”, “Araba”, “Gipuzkoa”]
  - Enumerar un elemento de un array: a[1] = “Araba”;
  - Enumerar propiedad de un objeto: a[“color”] = “azul”;

- Delete:
  - Delete a[2]; // Borraría el elemento “Gipuzkoa” y lo sustituiría por *undefined*

- *In:*
  - *Devuelve true si el objeto tiene la propiedad o método*
    - *Ej: “write” in document*

- *Instanceof:*
  - *Devuelve true si es una instancia de un objeto nativo Javascript:*
    - *Ej: a = new Array (1,2,3);*
      a instanceof Array; // Devuelve true

Operadores

Operadores misceláneos

- Coma:
  - Expresiones que se evaluan de izquierda a derecha: var nombre, direccion, apellidos;
  - Operación loop (repetir): for (var i=0, j=0; i<125M i++, j+10)

- Interrogación (operador condicional):
  - Es la forma reducida de if … else.
  - Condicion ? Expresion si es cierta : expresión si es falso;
  - Ej: var a=3, b=5;
    Var r = a > b ? a : b;

- Typeof:
  - Devuelve el tipo de valor de una variable o expresión
  - Los tipos son: number, string, boolean, object, function, undefined.
  - Ej: if (typeof miVariable == “number”) alert (“Mi variable es number”);

Estructuras de control

Construcción if:

```
if (condición) // entre paréntesis irá la condición que se evaluará true o false.
{
    // instrucciones a ejecutar si se cumple la condición
}
```

- Ejemplo:

```
if  (miEdad >= 18)
{
    alert ("Ya eres una persona adulta");
}
```

Estructuras de control

Construcción if…else:

```
if (condición) {
  // entre paréntesis irá la condición que se evaluará true o false.
  // instrucciones a ejecutar si se cumple la condición
} else {
  // instrucciones a ejecutar si no se cumple la condición
}
```

```
// Ejemplo:
if (miEdad >= 18) {
  alert("Ya eres una persona adulta");
} else {
  alert("Aún no eres mayor de edad");
}
```

Estructuras de control

```
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 alos valores anteriores
}
```

Estructuras de control

Bucle for:

```
for (expresion inicial; condición; incremento) {
   // instrucciones a ejecutar dentro del bucle
}
```

Ejemplo:

```
for (var i = 1; i < 20; i++) {
  // Instrucciones que se repetirán 20 veces
}
```

Estructuras de control

Bucle while:

```
while (condicion) {
// instrucciones a ejecutar dentro del bucle
}
```

Ejemplo:

```
var i = 0;
while (i <= 10) {
//Instrucciones a ejecutar hasta que i sea mayor que 10 y  i++;
}
```

Estructuras de control

Bucle do while:

```
do {
  // instrucciones a ejecutar dentro del bucle
} while (condicion);
```

Ejemplo:

```
var i = 0;
do {
  // Instrucciones a ejecutar mientras i sea menor que 3 (2 veces)
  i++;
} while (i < 3);
```