Introducción a JavaScript

- Un viaje por la sintaxis -

Chema Durán

¿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

Características

- 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

Características

- 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, etc.
- Preprocesa datos en el cliente
- Modifica estilos y contenido de navegadores
- Puede solicitar ficheros al servidor

Características

- ¿Es compatible JavaScript en todos los dispositivos?
- ¿Es soportado por todos los navegadores?

Hay algunas incompatibilidades entre navegadores

Algunos dispositivos móviles no pueden ejecutar JavaScript

Características

- **¿Podemos, mediante JavaScript, vulnerar la seguridad de un sitio web?**
- **¿Podemos atacar un servidor mediante JavaScript?**

Se ejecuta el código en un “espacio seguro de ejecución”

Scripts restringidos por la política del “mismo origen”

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

Características

- **No** puede modificar o acceder a las preferencias del navegador, ventana principal, impresión, etc.
- **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.

Características

- **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.

Integración de código

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>alert("Hola Mundo!");</script>
</head>
<body>

</body>
</html>
```

Integración de código

Añadir código JavaScript mediante un fichero externo.

```
<!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 src="ruta/archivo.js">alert("Hola Mundo!");</script>
    <script src="../ruta/archivo.js">alert("Hola Mundo!");</script>
    <script src="https://www.dominio.com/archivo.js">alert("Hola Mundo!");</scrip
    <script async src="ruta/archivo.js">alert("Hola Mundo!");</script>
    <script defer src="ruta/archivo.js">alert("Hola Mundo!");</script>
</head>
<body>
</body>
</html>
```

Integración de código

Ventajas de añadir código JavaScript mediante 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

Ventajas de añadir código JavaScript mediante un fichero externo.

- 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

- **No se puede proteger**

Incluir mensaje de Copyright	Ofuscar el código (www.javascriptobfuscator.com)	Promocionar el código

Sintaxis - Introducción

Mayúsculas y minúsculas

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

Tabulación y saltos de línea

- JavaScript ignora los espacio, las tabulaciones y los saltos de línea con algunas excepciones.
- Emplear la tabulación y los saltos de líneas mejora la presentación y la legibilidad del código.

Sintaxis - Introducción

Tabulación y saltos de línea

```
<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>
```

Sintaxis - Introducción

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).

Sintaxis - Introducción

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

Sintaxis - 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.

Sintaxis - Variables

- Crear variables utilizando la palabra reservada “var” o “let”:

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

- Asignar valor a una variable ya creada:

```
edad1 = 15;
```

- Crear variable y asignar valor:

```
let 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.

Sintaxis - 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:

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

Sintaxis - Tipos

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

Sintaxis - Tipos

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.*

Sintaxis - Tipos

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: \'

Sintaxis - Tipos

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.*

Sintaxis - Tipos

Booleanos:

- true
- false

Objetos:

- String
- Date
- Array
- Etc.

Null:

- null (desprovisto de valor)

Function:

- La definición de una función

Sintaxis - Tipos

Conversión entre tipos de datos:

- Entero + Float = Float
- Número + Cadena = Cadena

Conversión de cadenas a números:

- parseInt(“32”)
- parseFloat(“32.1”)

Conversión de números a cadenas:

- “” + número

Sintaxis - Operadores

Operadores de comparación

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

Sintaxis - 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.*

Sintaxis - 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

Sintaxis - 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

Sintaxis - 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

Sintaxis - Operadores

Operadores de objeto

- Punto:
  ```
  Objeto.propiedad
  Objeto.método
  ```
- Corchetes:
  - Crear un array: `let a = [“Huelva”, “Sevilla”, “Cádiz”];`
  - Enumerar un elemento de un array: `a[1] = “Cádiz”;`
  - Enumerar propiedad de un objeto: `a[“color”] = “azul”;`
- delete:
  - `Delete a[2]; // Borraría el elemento “Cádiz” 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
    ```

Sintaxis - Operadores

Operadores misceláneos

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

- Interrogación (operador condicional):
  - Es la forma reducida de `if … else`.
  - Condicion ? Expresion si es cierta : expresión si es falso;

  Ej:

```
let a = 3, b = 5;
let 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");`

Sintaxis – Estructuras de control

Construcción if:

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

Ejemplo:

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

Sintaxis – 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");
}
```

Sintaxis – 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 a
  //los valores anteriores
}
```

Sintaxis – Estructuras de control

Bucle for:

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

Ejemplo:

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

Sintaxis – Estructuras de control

Bucle while:

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

Ejemplo:

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

Sintaxis – Estructuras de control

Bucle do while:

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

Ejemplo:

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

END

prof.jduran@iesalixar.org