Scripting en el cliente: Javascript

Tecnologías Web

Motivación

- ¿Por qué usar JavaScript?
  - Permite crear efectos atractivos visualmente
  - Permite crear sitios WEB que se visualicen de la misma manera en distintos navegadores
  - Permite suplir las carencias de implementación de especificaciones entre los distintos navegadores
  - Permite interactuar con el usuario sin los problemas de la latencia de la red.
- ¿Por qué aprender JavaScript?
  - Habitualmente se copian scripts genéricos disponibles en la WEB.
  - Están pobremente escritos
  - En la mayor parte de casos es necesario particularizarlos para adaptarlo a nuestro sitio WEB

¿ Qué es Javascript ?

- Características
  - Es un lenguaje de **script** (guión ⇔ secuencia de instrucciones)
  - Tiene algunas características de orientación a objetos
  - Es un lenguaje **interpretado** → No se compila
- JavaScript **NO** es Java
- Se ejecuta en el **cliente**
  - El navegador del cliente es el que se encarga de interpretar y ejecutar los comandos de JavaScript

¿Qué podemos hacer con Javacript?

- Crear páginas WEB más atractivas
  - Crear etiquetas dinámicamente
  - Intercambiar imágenes
- Controlar el navegador
  - Mostrar mensajes al usuario
  - Crear ventanas emergentes
- Interactuar con formularios
  - Validar formularios
- Interactuar con el usuario
  - Responder a sucesos generados al interactuar el usuario con el navegador

¿Qué no podemos hacer con Javascript?

- No podemos generar gráficos
- No soporta conexiones por red de ningún tipo
  - Hasta la aparición de AJAX
- No podemos leer ni escribir archivos

Conceptos básicos de Javascript

- Objetos
  - Elementos que se pueden manipular: ventanas, formularios, navegador, …
- Propiedades (atributos)
- Métodos
- Eventos
- Funciones

```
<html>
  <head>
    <title>Ejemplo de JavaScript</title>
    <script language="javascript" type="text/javascript">
      function mensaje(){ alert("Hola Mundo !!"); }
      window.onload=mensaje;
    </script>
  </head>
  <body>
    ...
  </body>
</html>
```

¿Dónde se incluye el código?

- El código Javascript se incluye dentro del documento HTML, dentro del elemento `<script>` ... `</script>`
- Generalmente al elemento se le añade un atributo "`language`" y otro "`type`" para especificar que el script es JavaScript

```
<script language="JavaScript" type="text/javascript">...</script>
```

- Referencia a archivos externos
  - Útil para guiones largos y para no repetir código
  - El archivo externo es un archivo de texto que sólo contiene instrucciones JavaScript

```
<script language="JavaScript" type="text/javascript" src="fiochero.js">...</script>
```

- Dentro de un manejador de eventos

```
<input type="button"
              value="Haz click aqui"
              onclick="alert('Has hecho click sobre mi');">
```

Ocultar los guiones para los navegadores antiguos

- Puesto que los scripts se ejecutan en el navegador, existe la posibilidad (cada vez más remota) de que el navegador de un cliente sea antiguo y no admita javascript
  - Sin embargo, el usuario puede tenerlo desactivado
- Para ocultar el script a esta clase de navegadores la sentencias JavaScript se suelen colocar dentro de un comentario de HTML

HTML

```
<script language="JavaScript" type="text/javascript">
<!--
...
//-->
</script>
```

XHTML

```
<script language="JavaScript" type="text/javascript"><![CDATA[
<!--
...
//-->]]></script>
```

Ocultar los guiones a los navegadores antiguos (cont)

- También es posible incluir código HTML en el documento para que se muestre únicamente en navegadores que no interpreten JavaScript
  - Este código se incluye en el elemento `<noscript>` de HTML
  - Los navegadores antiguos no entenderán el elemento e interpretarán el código que contiene

```
Navegador sin soporte de JavaScript
```

```
<noscript>
Su navegador no admite JavaScript, o JavaScript ha sido deshabilitado.
<p>Pruebe con la <a href="no-js.html">versión sin JavaScript</a>
de esta página</p>
</noscript>
```

  - Este código también es mostrado si el navegador tiene desactivado la ejecución de JavaScript

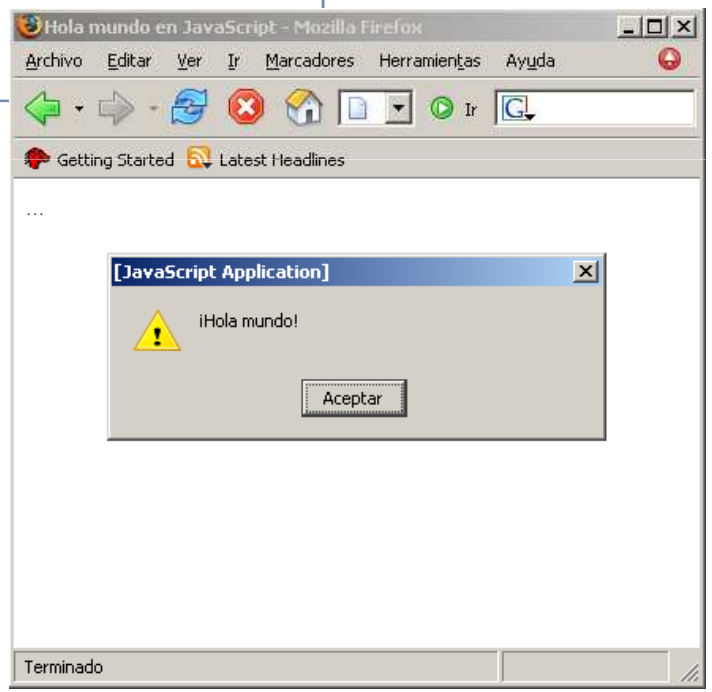
Uso de Javascript dentro de una página XHTML

- Los guiones de la págína WEB se procesan secuencialmente
  - Los guiones que hayan en la cabecera se ejecutan al abrir la página.
    - Por tanto no podemos modificar los elementos que aparecen dentro del <body>
  - Los que haya en el cuerpo se ejecutan a medida que se encuentran durante la construcción de la página web.
- Podemos usar la etiqueta <script> dentro del <head> o dentro del <body> de nuestra página WEB.
  - Lo habitual es que los scripts estén dentro de <head>
  - Excepciones a esta colocación
    - Si el script está diseñado para escribir datos en la página, debe colocarse dentro de <body>
    - Si el script se refiere a un elemento de la página, el script debe colocarse/ser llamado después del elemento
    - En muchas etiquetas HTML se puede añadir sentencias JavaScript como atributos de las mismas

Ejemplos

Ejemplo 1: Aviso por pantalla

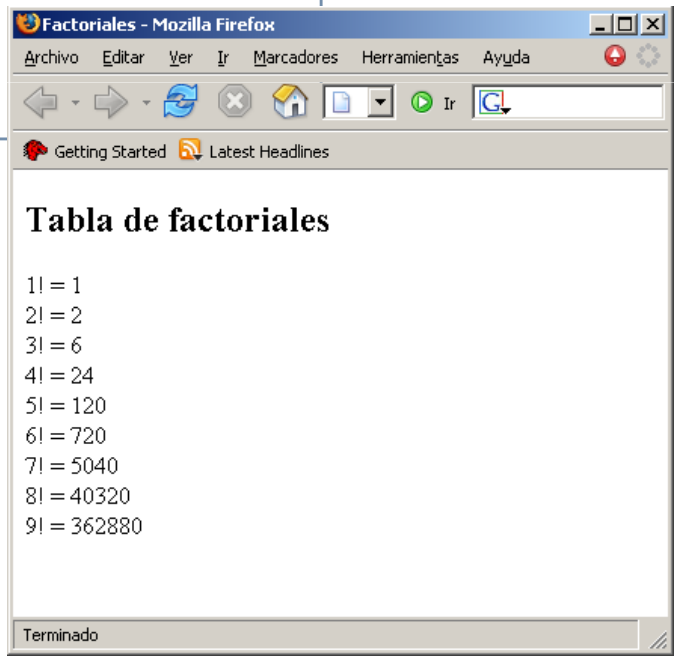
```
<html>
  <head>
    <title>Hola mundo en JavaScript</title>
    <script type="text/javascript" language="javascript"><!--
      alert("¡Hola mundo!");
    //--></script>
  </head>
  <body>...</body>
</html>
```



Ejemplos

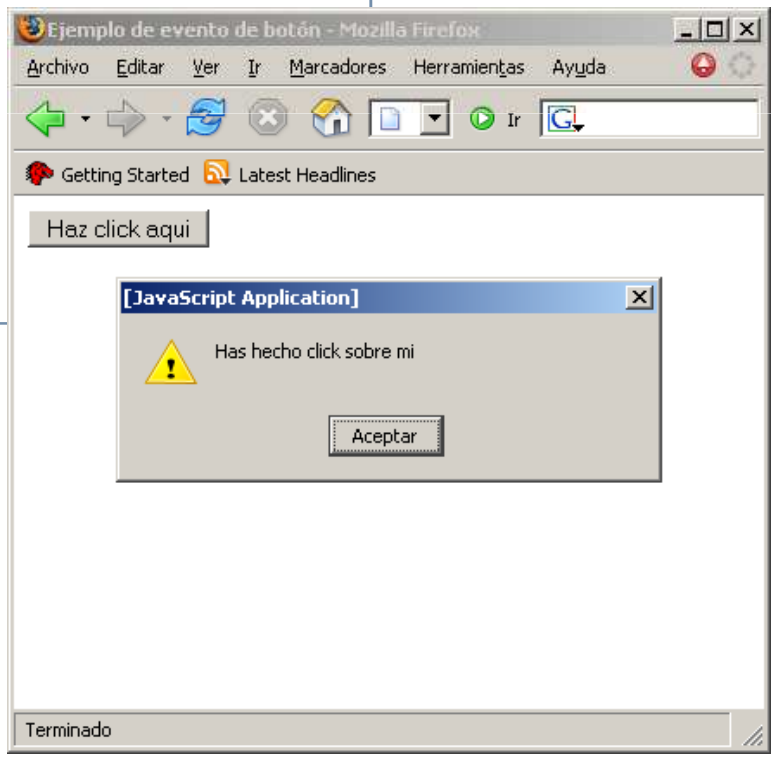
Ejemplo 2: Generación de etiquetas HTML

```
<html>
  <head><title>Factoriales</title></head>
  <body>
    <script type="text/javascript" language="javascript"><!--
       document.write("<h2>Tabla de factoriales</h2>");
       for(i = 1, fact = 1; i < 10; i++, fact *= i) {
         document.write(i + "! = " + fact);
         document.write("<br>");
       }
    //--></script>
  </body>
</html>
```



Ejemplo 3: Funciones y eventos

```
<html>
  <head><title>Ejemplo de evento de botón</title>
    <script language="JavaScript" type="text/javascript"><!—
      function generarAviso(){
        alert("Has hecho click sobre mi");
      }
    //--></script>
  </head>
  <body>
    <form>
      <input type="button"
             value="Haz click aqui"
             onclick="generarAviso();">
    </form>
  </body>
</html>
```



Consideraciones iniciales

- Javascript distingue entre mayúsculas y minúsculas
  - Ej: **`while`** (correcto) , `While`, `WHILE` (incorrectos)
- Javascript ignora espacios en blanco, tabuladores y saltos de línea entre "tokens" de nuestro guión
  - token: palabra reservada, número, cadenas, nombre de función, …
  - Podemos por tanto sangrar nuestros guiones para que sean más legibles.
- Los `";"` son opcionales (aunque recomendables) al final de una instrucción

```
a = 3
b = 4
```

Equivale a

```
a = 3;
b = 4;
```

```
return
true;
```

Equivale a

```
return;
true;
```

```
a = 3; b = 4
```

- Se pueden incluir comentarios al estilo de C/C++/Java

```
// Este es un comentario en línea
/* Este es un comentario
  de varias líneas */
```

Consideraciones iniciales (cont)

- Identificadores (como en C++/Java)
  - Deben comenzar por una letra o por '_'
  - Pueden contener letras, dígitos y '_'
  - No pueden coincidir con las palabras reservadas
- Palabras reservadas

break	do	if	switch	typeof
case	else	in	this	var
catch	false	instanceof	throw	void
continue	finally	new	true	while
default	for	null	try	with
delete	function	return		

abstract	double	goto	native	static
boolean	enum	implements	package	super
byte	export	import	private	synchronized
char	extends	int	protected	throws
class	final	interface	public	transient
const	float	long	short	volatile
debugger				

Variables

- JavaScript es un lenguaje débilmente tipado
  - No se especifica el tipo de la variable
  - Se deduce por el contenido de la variable y el contexto
- Para declarar una variable se usa la palabra reservada **`var`** seguida por una lista de nombres de variables a declarar separadas por `','`.

```
var variable1, variable2;
```

Tipos de datos literales

- Números
  - JavaScript permite trabajar con números enteros y con números y punto flotante.
    - Internamente las operaciones se realizan en punto flotante
  - Enteros → `0, 15, -3 ...`
  - Punto flotante → `0.07, 7.21e+5, -2.17E-3`
  - En hexadecimal, empiezan por `0x` → `0x23, 0xFA`
- Valores lógicos (Booleanos)
  - Verdadero: `true`
  - Falso: `false`

Tipos de datos literales

- Cadenas de caracteres
  - Es una secuencia de letras, números, signos de puntuación, etc. encerrados entre comillas dobles " o simples '
  - Ej: `"Creación de Sitios WEB!"`, `'Iván Martínez Ortiz'`
  - Se usan comillas simples cuando no podemos usar comillas dobles

```
<a onclick="alert('Has pulsado el enlace')">...</a>
```

  - Secuencias de escape
    - \' → Comilla simple
    - \" → Comilla doble
    - \b → Retroceso
    - \f → Salto de página
    - \n → Salto de línea
    - \t → Tabulación
    - \\ → Barra inclinada \

Funciones

- Para declarar funciones se usa la palabra **`function`**, y entre paréntesis la lista de parámetros separados por comas (,)
- Para que la función devuelva un valor se utiliza la sentencia **`return`**
  - No es necesario que una función devuelva nada
- Los parámetros de las funciones no tienen tipo, al igual que el valor que devuelven

Prototipo de declaración de funciones

```
function nombre_funcion ( arg1, arg2, ...){
  return;
}
```

Ejemplo

Ejemplo de llamada y paso de parámetros a funciones

```
<html>
  <head>
    <title>Ejemplo sobre funciones JavaScript</title>
    <style type="text/css">
      form { display: inline }
    </style>
    <script language="JavaScript" type="text/javascript"><!--
      function init(){
        cambiarFondoPantalla("red");
      }

      function cambiarFondoPantalla(colorFondo){
        document.bgColor = colorFondo;
      }
      function buttonHandler(colorFondo){
        if( confirm("¿Está seguro de querer cambiar el color?")){
          cambiarFondoPantalla(colorFondo);
        }
      }
      window.onload=init;
    //--></script>
  </head>
  <body>
    <p>Para cambiar el fondo de la pantalla de nuevo a blanco, pulsa </p>
    <form><input type="button" onClick="buttonHandler('white')" value="aquí"/></form>
  </body>
</html>
```

Expresiones

- Expresiones numéricas
  - Operadores disponibles: `+` , `++`, `-`, `--`, `*`, `/`, % (módulo).
  - Operadores adicionales: +=, `-=`, `*=`, `/=`, `^=` (exponenciación), %=
- Expresiones de cadenas
  - El operador `'+'` concatena cadenas
  - Ej: `var cadena = "Java"+"Script"`
  - NOTA: Si en una expresión el primer operando es una cadena, JavaScript convierte cualquier número de dicha expresión a cadena.
    - Ej: `var cad ="2"+2+2` ('cad' tiene almacenada la cadena "222")

Expresiones

- Expresiones de comparación
  - Operadores relacionales: `==`, `!=`, `>`, `<`, `>=`, `<=`, `===` (identidad), `!==` (no identidad)
  - Conversión automática de tipos en las comparaciones
    - Si mezclamos en una expresión distintos tipos de datos, JavaScript realiza conversiones automáticas entre tipos para llevar a cabo la comparación. Las reglas son
    - Si un operando es una cadena y el otro un número, JavaScript intenta convertir la cadena a número. Si no se puede convertir la comparación devuelve `false`

```
var oper1 = "5"; var oper2 = 5; var resultado = oper1 == oper2
```

    - Si uno de los operandos es un booleano y el otro un número se convierte el booleano a número (`true` → 1, `false` → 0)

```
var oper1 = true; var oper2 = 1; var resultado = oper1 == oper2
```

  - Los operadores ===, !==, realizan una comparación estricta, es decir no se realiza ninguna conversión de tipos para realizar la comparación

- Expresiones lógicas
  - Operadores disponibles: && (and), || (or), ! (not)

Ejemplo de expresiones

```
<html>
  <head><title>Pruebas con expresiones booleanas</title></head>
  <body>
  <script language="JavaScript" type="text/javascript"><!--
    var val1 = true;
    var val2 = 10;
    var val3 = "comprobado";
    var val4 = false;
    var val5 = 0;
    var val6 = "";
    var oper1 = eval(prompt("Introduzca el primer operando", true));
    var oper2 = eval(prompt("Introduzca del segundo operando", true));
    var salida = oper1 && oper2;
    alert(salida);
  //--></script>
  </body>
</html>
```

Evalúa la expresión antes de asignarla a la variable

¿ false && 10 ? ¿ true && 10 ?

Estructuras de control

- Sentencias condicionales: **`if`**, **`switch`**

Sentencia if

```
if( expresion ){
  // Sentencias
}else{
  // Sentencias
}
```

NOTA: 0, "", null, undefined son evaluados como false

Sentencia switch

```
switch( expresion ){
  case Caso1:
    // Sentencias caso 1
 case Caso2:
    // Sentencias caso 2
 case Caso3:
    // Sentencias caso 3
  default:
    // Sentencias pre
}
```

La expresión es evaluada al comienzo de la estructura y debe devolver un valor ya sea número, lógico o cadena

- Bucles: **`while`**, **`for`**, **`do .. while`**

Sentencia while

```
while( expresion ){
  // Sentencias
}
```

Sentencia for

```
for( var contador = valor_inicial; Expresión; contador++){
  // Sentencias
}
```

Sentencia do ... while

```
do {
  // Sentencias
}while(expresion)
```

- Sentencias para control de bucles
  - Salir del bucle → **`break`**
  - Saltar a la siguiente iteración → **`continue`**

Referencias

- http://www.w3schools.com/js/default.asp
- http://www.mozilla.org/catalog/web-developer/
- http://www.mozilla.org/js/
- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/anch_webdev.asp

Críticas, dudas y sugerencias...

Federico Peinado
www.federicopeinado.es