

Javascript

The logo for JavaScript, featuring the letters 'JS' in a bold, black, sans-serif font inside a yellow square, followed by the word 'JavaScript' in a bold, black, sans-serif font on a white background.

JS JavaScript

Historia

- Junto con HTML y CSS, es una de las tres tecnologías pilares de la WWW
- Creado por Netscape Communications en 1995
- Aunque su nombre se parece, no estaba relacionado originalmente al lenguaje Java sino a C (pero Java era más marketinero en ese momento)
- Se utiliza para escribir código que se ejecuta en el lado del cliente
 - En épocas de baja velocidad de conexión buscaba evitar que el usuario tenga que volver a enviar/recibir más información
 - Ejemplo de uso: verificar datos de un formulario localmente, antes de que el cliente los envíe

Limitaciones

- JavaScript fue diseñado para ejecutar en un entorno muy limitado que permitiera a los usuarios confiar en su ejecución
 - Scripts Javascripts no pueden cerrar ventanas que no hayan abierto ellos mismos.
 - Las ventanas creadas no pueden ser demasiado pequeñas ni demasiado grandes ni colocarse fuera de la vista del usuario
 - No puede acceder a archivos del ordenador del usuario ni leer o modificar las preferencias del navegador.
 - Si la ejecución de un script dura demasiado tiempo, el navegador le informa al usuario y le da la posibilidad de detener su ejecución.
- Existen alternativas para poder saltar algunas de las limitaciones anteriores, pero quedan fuera del alcance de este curso

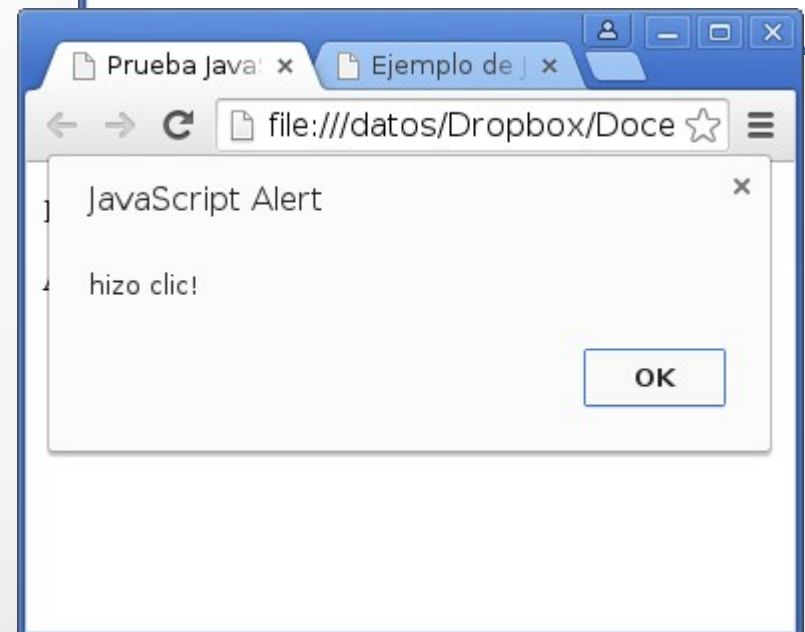
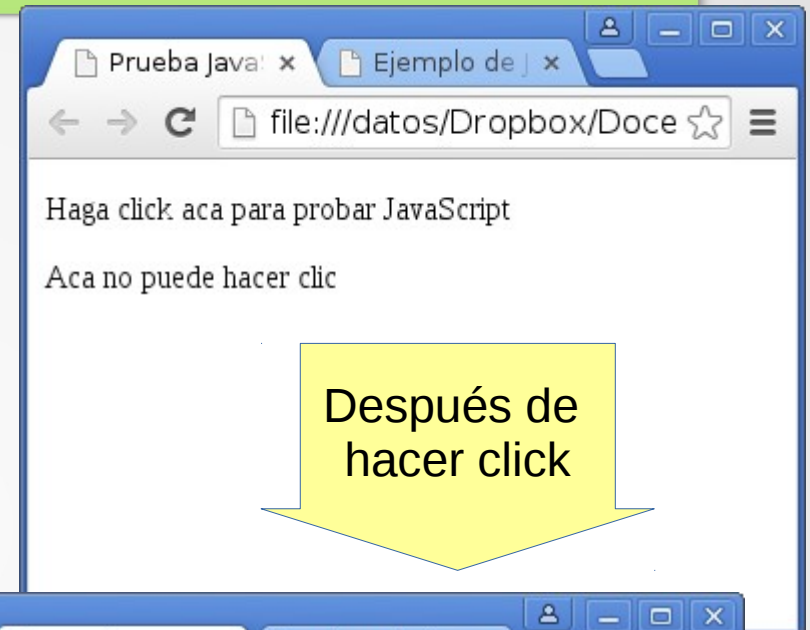
Dónde y cómo incluir JavaScript?

- **En el mismo archivo HTML**
 - **En línea:** se inserta en atributos de elementos HTML. Estos atributos son manejadores de eventos que ejecutan código de acuerdo a una acción del usuario.
 - **Embebido:** El código se encierra entre etiquetas `<script>`
 - Se puede incluir en cualquier parte del documento. Se recomienda dentro de la cabecera (entre etiquetas `<head>`)
 - En la etiqueta `<script>` hay que añadir el atributo `type=text/javascript`
 - Es útil para añadir bloques pequeños de código o instrucciones específicas en un documento HTML
 - Si está en más de un HTML, es necesario modificar todos los que incluyen el mismo bloque de código JavaScript.
- **En archivo externo:** recomendado para compartir entre varios HTML

JavaScript en línea (evento)

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Prueba JavaScript</title>
  </head>
  <body>
    <div id="principal">
      <p onclick="alert('hizo clic!')">
        Haga click aca para
        probar JavaScript</p>
      <p>Aca no puede hacer clic</p>
    </div>
  </body>
</html>
```

Si se cambia el evento **onclick** por **onMouseOver**, el alerta se disparará cada vez que el cursor del mouse pase por arriba del primer párrafo

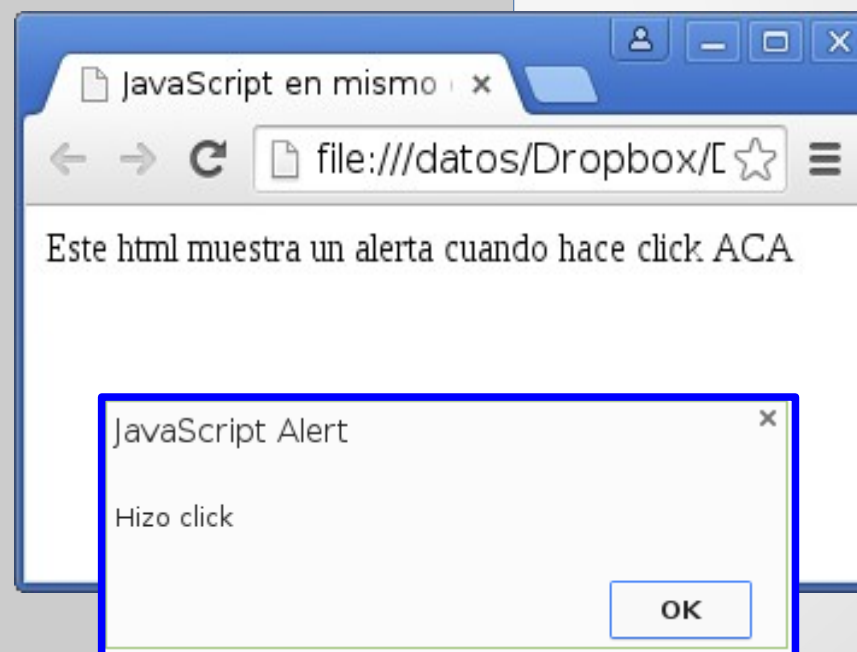


JavaScript embebido en cabecera

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title>JavaScript en mismo documento</title>

    <script type="text/javascript">
      function mensaje(){
        alert("Hizo click");
      }
    </script>
  </head>

  <body>
    Este html muestra un alerta
    cuando hace click
    <span onclick=mensaje()>
      ACA </span>
    </body>
  </html>
```



JavaScript en archivo externo

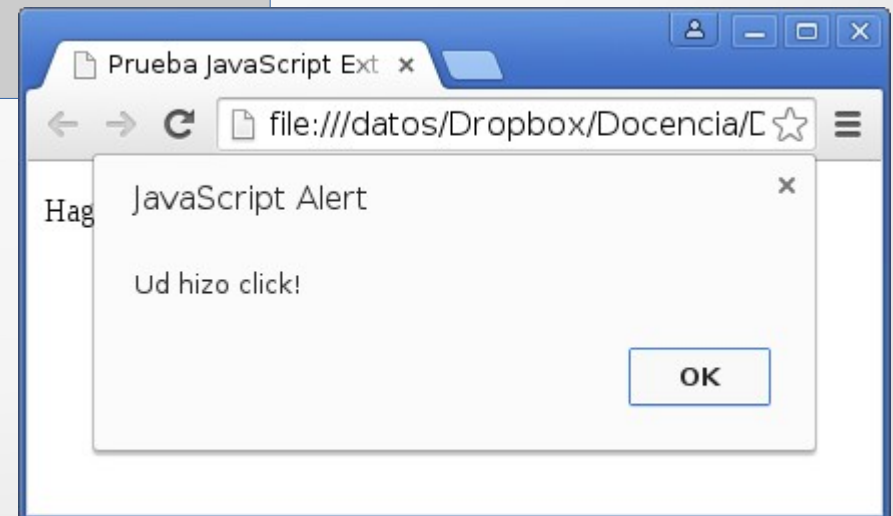
```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <title>Prueba JavaScript Externo</title>
    <script type="text/javascript"
      src="miScript.js"></script>
  </head>
  <body>
    <p onclick=mostraralerta()>Haga click acá</p>
  </body>
</html>
```

miScript.js

```
function mostraralerta(){
  alert('Ud hizo click!');
}
```



Después de
hacer click



JavaScript deshabilitado?

- HTML define la etiqueta `<noscript>` para mostrar un mensaje cuando el navegador no puede ejecutar JavaScript
- La etiqueta `<noscript>` se debe incluir en el interior de la etiqueta `<body>` (normalmente se incluye al principio)

```
<body>
  <noscript>
    <p>Bienvenido a Mi Sitio</p>
    <p>La página que está viendo requiere
      JavaScript para funcionar. <br />
      Si lo ha deshabilitado intencionadamente,
      por favor vuelve a activarlo.</p>
  </noscript>
  ...
</body>
```


Sobre el lenguaje JavaScript

- Sintaxis similar a otros lenguajes como Java y C
- Normas básicas
 - Es *case sensitive*: Si no se respeta, el script no funciona.
 - *No se declara el tipo de las variables*. La misma variable puede almacenar diferentes tipos de datos durante la ejecución del script.
 - No es necesario terminar cada sentencia con punto y coma (;)
 - Permite comentarios de una línea (*//...*) o más (*/* ... */*)
- Palabras actualmente reservadas por JavaScript
 - *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*

Resumen lenguaje: tipos y variables

- Las variables se declaran con la palabra var por delante (sin indicar el tipo, que ya se dijo que puede variar)
 - `var iva = 16;`
 - `var total = 234.65;`
 - `var mensaje = "Bienvenido a nuestro sitio";`
 - `var nombreProducto = 'Producto ABC';`
 - `var letraSeleccionada = 'c';`
- Uso indistinto de comillas simples y dobles para texto. Sirve para generar fácilmente texto que incluye a alguna de ellas:
 - `var texto1 = "Una frase con 'comillas simples' dentro";`
 - `var texto2 = 'Una frase con "comillas dobles" dentro';`
- También caracteres de escape iguales en Java: `\n, \t, \', \", \\`

Resumen lenguaje: sentencias

- Operadores matemáticos, lógicos y relacionales igual que en Java/C
- Sentencia de asignación con símbolo =
- Estructuras de control de flujo:
 - `if(condicion) { ... }`
 - `if(condicion) { ... } else { ... }`
 - `switch (expresión) { case X : ... break; ... }`
 - `for(inicializacion; condicion; actualizacion) { ... }`
 - `for(indice in array) { ... }`
 - `while (condición) { ... }`
 - `do { ... } while (condición)`

Resumen lenguaje: funciones sobre texto

- `var s1 = "Hola"; var s2 = " Mundo";`
- `var s = s1 + s2;` // `s = "Hola Mundo"`
- `s = s1.concat(s2);` // idem `s = s1 + s2`
- `var cantLetras = s.length;` // `cantLetras = 10`
- `s1 = s.toUpperCase();` // `s1 = "HOLA MUNDO"`
- `s2 = s.toLowerCase();` // `s2 = "hola mundo"`
- `var letra = s.charAt(5);` // `letra = "M"`
- `pos = s.indexOf("o");` // `pos = 1`
- `pos = s.lastIndexOf("o");` // `pos = 9`
- `s1 = s.substring(2, 7);` // `s1 = "la Mu"`
- `var palabras = s.split(" ");` // `palabras = ["Hola", "Mundo"];`
- `var letras = s.split("");` // `letras = ["H", "o", "l", "a", " ", "M", "u", "n", "d", "o"]`

Resumen lenguaje: operaciones array

- `var array1 = [1, 2, 3];`
- `var cant = array1.length; // cant = 3`
- `var array2 = array1.concat(14, 15, 16); // array2 = [1, 2, 3, 14, 15, 16]`
- `var array3 = array1.concat([4, 5, 6]); // array3 = [1, 2, 3, 4, 5, 6]`
- `var mensaje = array2.join(""); // mensaje = "123141516"`
- `mensaje = array2.join(" "); // mensaje = "1 2 3 14 15 16"`
- `var ult = array1.pop(); // quita al final → array1 = [1, 2], ult = 3`
- `array1.push(4); // pone al final → array1 = [1, 2, 4]`
- `var prim = array1.shift(); // quita primero → array1 = [2, 4], prim = 1`
- `array1.unshift(0); // agrega al principio → array1 = [0, 2, 4]`
- `array1.reverse(); // invierte → array1 = [4, 2, 0]`

Resumen lenguaje: operaciones útiles

- **NaN**, (del inglés, "Not a Number") indica un valor numérico no definido (por ejemplo, la división 0/0).
 - `var numero1 = 0; var numero2 = 0;`
`alert(numero1/numero2); // muestra el valor NaN`
- **isNaN()**, permite proteger a la aplicación de posibles valores numéricos no definidos
 - `var numero1 = 0;`
`var numero2 = 0;`
`if(isNaN(numero1/numero2)) {`
`alert("La división no está definida para los números indicados");`
`} else {`
`alert("La división es igual a => " + numero1/numero2);`
`}`
- **Infinity**, valor numérico infinito y positivo (también existe el valor `-Infinity` para los infinitos negativos)
 - `var numero1 = 10; var numero2 = 0;`
`alert(numero1/numero2); // muestra el valor Infinity`
- **toFixed(digitos)**, devuelve número original con los decimales indicados por parámetro y redondeado
 - `var numero1 = 4564.34567;`
`numero1.toFixed(2); // 4564.35`
`numero1.toFixed(6); // 4564.345670`
`numero1.toFixed(); // 4564`

Definiendo funciones

- Para declarar una función, se usa la palabra clave “function”, el nombre deseado y la lista de parámetros formales
- Para retornar un resultado se usa la sentencia “return”
- No se indica tipo de los parámetros ni del resultado

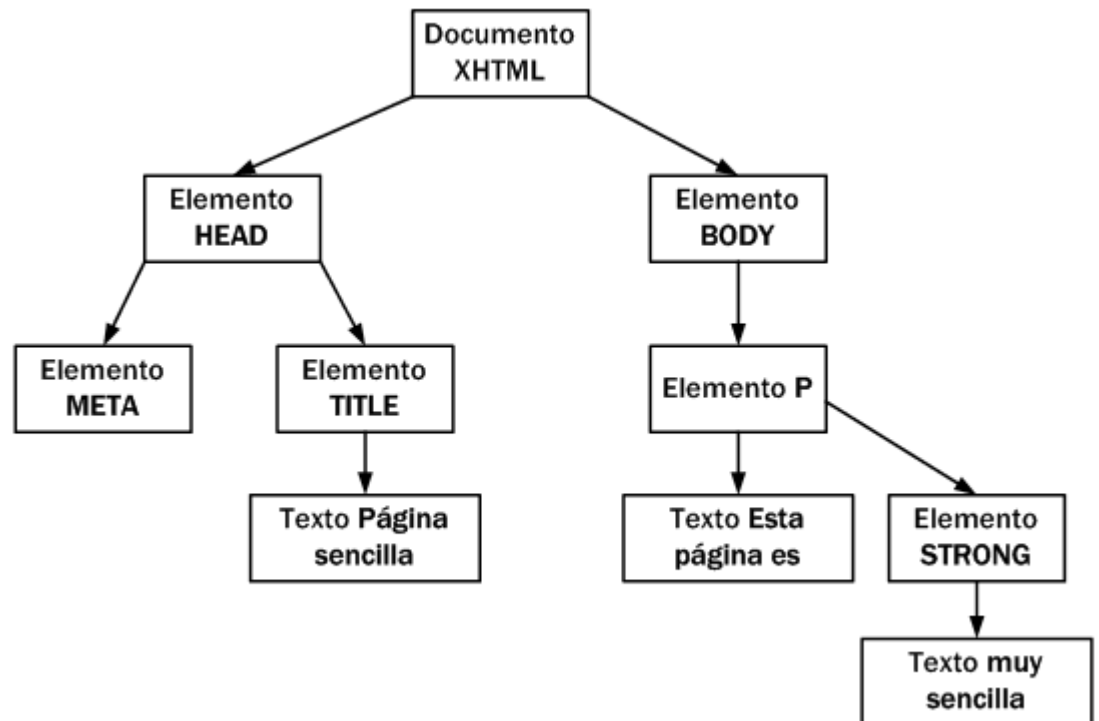
```
function calculaPrecioTotal(precio, porcentajeImp) {  
    var gastosEnvio = 10;  
    var precioConImp = (1 + porcentajeImp/100) * precio;  
    var precioTotal = precioConImp + gastosEnvio;  
    return precioTotal.toFixed(2);  
}
```

DOM: Document Object Model

- El DOM o del Document Object Model es un modelo que permite a los programadores web acceder y manipular las páginas web como si fueran documentos XML
- En el modelo DOM, el archivo HTML de la izquierda se transforma en el árbol de nodos de la derecha

```
<!DOCTYPE html ...>
<html>
<head>
<meta ... />
<title>Página sencilla</title>
</head>

<body>
<p>Esta página es
    <strong>muy sencilla</strong>
</p>
</body>
</html>
```



DOM: tipos de nodos

- 12 tipos de nodos
- La mayoría de las páginas se pueden manipular usando los sig:
 - **Document**: nodo raíz del que derivan todos los demás nodos del árbol.
 - **Element**: representa cada etiqueta XHTML. Puede contener atributos y puede derivar otros nodos.
 - **Attr**: para representar atributos de las etiquetas (uno por cada par atributo=valor)
 - **Text**: contiene el texto encerrado por una etiqueta XHTML
 - **Comment**: comentarios incluidos en la página
- Otros tipos de nodos más específicos: DocumentType, CDataSection, DocumentFragment, Entity, EntityReference, ProcessingInstruction y Notation

Acceso a los nodos

- Una vez construido el árbol de nodos DOM se puede **acceder a cualquier nodo del mismo para manipularlo** (leer o modificar el valor de un elemento, mover un elemento de la página, añadir nuevos elementos, etc)
- Para acceder a un nodo se puede bajar por la jerarquía, comenzado por la raíz del árbol (nodo Documento) o hacerlo **por acceso directo, que es la forma más rápida**
- El acceso a los nodos, para su modificación o eliminación **solamente es posible cuando el árbol DOM ha sido construido completamente** (es decir, después de que la página XHTML se ha cargado por completo)

Crear un elemento nuevo

- 4 pasos para añadir un elemento nuevo a una página

```
// paso 1: Crear nodo de tipo Element
```

```
var parrafo = document.createElement("p");
```

```
// paso 2: Crear nodo de tipo Text
```

```
var contenido = document.createTextNode("Hola Mundo!");
```

```
// paso 3: Añadir el nodo Text como hijo del nodo Element
```

```
parrafo.appendChild(contenido);
```

```
// paso 4: Añadir el nodo Element como hijo de la pagina
```

```
document.body.appendChild(parrafo);
```

Eliminar un elemento

- Para eliminar un nodo del árbol DOM es necesario utilizar la función `removeChild()`:

```
var parrafo =  
document.getElementById("provisional");  
parrafo.parentNode.removeChild(parrafo);
```

- Donde en la página HTML, se habrá definido dicho id:
`<p id="provisional">...</p>`
- `removeChild()` requiere como parámetro el nodo del elemento a eliminar (se obtiene mediante `getElementById()`)
- Luego, `removeChild()` debe ser invocada desde el nodo padre. Se obtiene mediante `nodoHijo.parentNode`

Acceder a un atributo en HTML

- Una vez se accede a un nodo, se puede conocer o modificar sus atributos y propiedades.
- Los atributos XHTML son propiedad de los nodos. Para acceder a su valor, se indica el nombre del atributo
- Ejemplo: obtener dirección a la que enlaza un enlace
 - En Javascript

```
var enlace = document.getElementById("enlace");  
alert(enlace.href); // muestra http://www...com
```

- En HTML

```
<a id="enlace" href="http://www...com">Mi  
Enlace</a>
```

Acceder a un atributo en CSS

- Para obtener las propiedades CSS se debe utilizar el atributo style
- Ejemplo: obtener el valor de la propiedad margin de una imagen
 - En JavaScript

```
var imagen = document.getElementById("imagen");  
alert(imagen.style.margin);
```

- En HTML

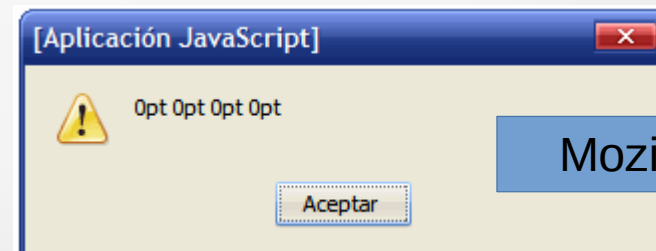
```

```

- Atención: El resultado puede diferir en distintos navegadores:



iexplore



Mozilla

Acceder a propiedades CSS compuestas

- Si el nombre de una propiedad CSS es compuesto, se en JavaScript se modifica ligeramente su nombre:
 - En Javascript

```
var parrafo = document.getElementById("parrafo");  
alert(parrafo.style.fontWeight); // muestra "bold"
```
 - En HTML

```
<p id="parrafo" style="font-weight: bold;">...</p>
```
- Para nombre de propiedades CSS compuestos se eliminan los guiones medios (-) y escribe en mayúscula la letra siguiente:
 - line-height se transforma en lineHeight
 - border-top-style se transforma en borderTopStyle
 - etc

Modificar contenido de un elemento

- Acceder al contenido de un párrafo y lo modifica

- En HTML

```
<p id="modificable"> Texto original </p>
```

- En JavaScript

```
parrafo = document.getElementById("modificable");  
alert("Contenido anterior " + parrafo.innerHTML);  
parrafo.innerHTML = "Texto modificado";  
parrafo.style.fontWeight="bold";
```

innerHTML sirve para acceder a elementos que tienen texto (<p>, <h1>, <a>, etc)

Modelos de Eventos

- Es la principal incompatibilidad entre navegadores
- Hay 3 modelos para manejar eventos
 - Modelo básico: Es el único modelo compatible en todos los navegadores. Permite crear aplicaciones que funcionan de la misma manera en todos los navegadores.
 - Modelo estándar: más poderoso que el modelo básico. Definido en DOM nivel 2. Todos los navegadores modernos lo incluyen, salvo Internet Explorer.
 - Modelo de Internet Explorer: es similar pero incompatible con el modelo estándar.

Modelo básico de eventos

- Cada elemento define su propia lista de eventos
 - Un tipo de evento (ej. click) puede estar definido para varios elementos diferentes y un elemento puede tener asociados varios eventos.
 - Nombre del evento = prefijo on + nombre en inglés de la acción asociada al evento. Ej: onclick (hacer click con el mouse), onmousemove (mover el ratón sobre un objeto)
- Más utilizados en aplicaciones web tradicionales:
 - **onload**: cuando se carga la página por completo
 - **onclick**, **onmouseover**, **onmouseout** para controlar el ratón
 - **onsubmit**: para controlar el envío de los formularios.
 - Lista completa: http://www.w3schools.com/jsref/dom_obj_event.asp

Uso de la variable this

- La variable **this** se utiliza para referirse al elemento XHTML que ha provocado el evento.
- Ejemplo: Dado un div con borde gris
 - Cuando el usuario pase el mouse por encima del <div>, que el color del borde sea rojo.
 - Cuando el ratón salga del <div>, que se vuelva a mostrar el borde con el color gris claro original

```
<div style="border:thin solid silver"  
    onmouseover="this.style.borderColor='red';"  
    onmouseout="this.style.borderColor='silver';">  
texto dentro de div </div>
```

Como trabajar en funciones

- También se puede usar funciones externas pasando la variable this como parámetro

```
function resalta(elemento) {  
  switch(elemento.style.borderColor) {  
    case 'silver':  
    case 'silver silver silver silver':  
    case '#c0c0c0':  
      elemento.style.borderColor = 'red';  
      break;  
  
    case 'red':  
    case 'red red red red':  
    case '#FF0000':  
      elemento.style.borderColor = 'silver';  
      break;  
  }  
}
```

La complejidad de la función se debe a la forma en la que los distintos navegadores almacenan el valor de la propiedad `borderColor`

Firefox, Chrome: black,
IE Explorer: black black black
black
Opera: #000000.

Eventos del Mouse

- **onclick**: Hacer click y soltar el botón del mouse. Todos los elementos.
- **ondblclick**: Hacer dos veces click. Todos los elementos.
- **onmousedown**: Pulsar (sin soltar) un botón del mouse. Todos los elementos.
- **onmousemove**: Mover el mouse. Todos los elementos.
- **onmouseout**: El mouse sale del elemento. Todos los elementos.
- **onmouseover**: El mouse entra en el elemento. Todos los elementos.
- **onmouseup**: Soltar el botón del mouse. Todos los elementos.
- **onmousedown**: Pulsar (sin soltar) un botón del mouse. Todos los elementos.

Propiedades de eventos de mouse

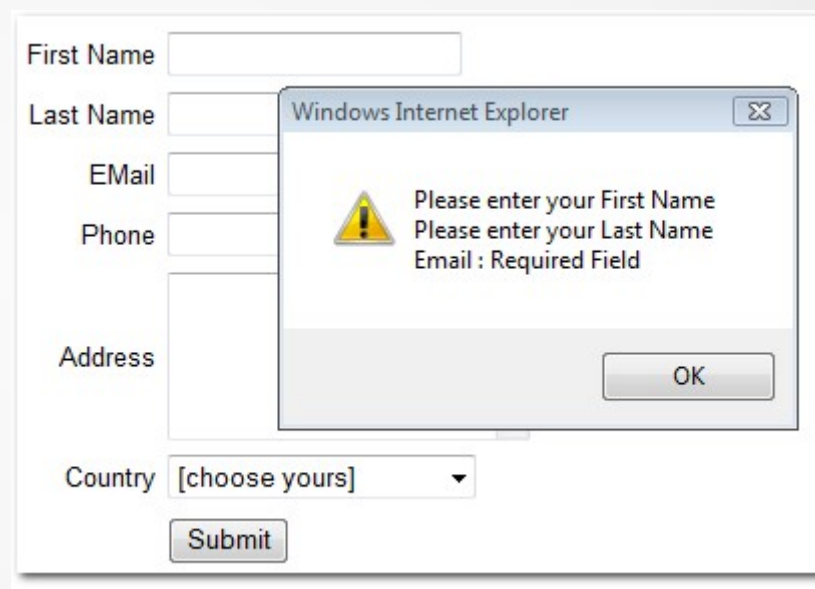
- Propiedades de los eventos del mouse
 - Posición del puntero en la ventana del navegador: clientX y clientY
 - Posición del puntero respecto al monitor: screenX y screenY
 - Posición respecto al inicio de la página (en caso que se haya hecho scroll, no coincide con posición en la ventana): pageX y pageY
(Esto no funciona en iexplore)

Ejemplo de uso

```
function mostrarMensaje(elEvento){  
    var evento = elEvento || window.event;  
    switch (evento.type){  
        case 'click':  
            alert("Evento Click en pos " + evento.clientX  
                + ", " + evento.clientY  
            break;  
        case 'keypress':  
            alert("Presionó tecla "  
                + "\n key code: " + evento.keyCode  
                + "\n char code: " + evento.charCode);  
            break;  
    }  
}
```

Manipulación de Formularios

- La programación de aplicaciones que contienen formularios web ha sido una de las tareas fundamentales de JavaScript
 - JavaScript surge por necesidad de validar datos de formularios del lado del usuario para evitar recargar la red
- Aplicaciones AJAX han cambiado un poco la historia, pero el manejo de formularios sigue siendo un requerimiento imprescindible para un programador JavaScript.



The image shows a web form with the following fields: First Name, Last Name, EMail, Phone, Address, and Country (a dropdown menu with the text "[choose yours]"). A "Submit" button is at the bottom. A "Windows Internet Explorer" error dialog box is overlaid on the form. The dialog box contains a yellow warning icon and the text: "Please enter your First Name", "Please enter your Last Name", and "Email : Required Field". An "OK" button is at the bottom right of the dialog box.

Acceso directo a los formularios

- Conviene acceder directamente a través de su nombre (atributo name) o de su atributo id.

- En JavaScript

```
var formularioPrincipal = document.form1;  
var formularioSecundario = document.form2;
```

- En HTML

```
<form name="form1" >  
    ...  
</form>  
<form name="form2" >  
    ...  
</form>
```

Acceso directo a elemento de formulario

- Los elementos de los formularios también se pueden acceder directamente mediante su atributo name
- En JavaScript

```
var formularioPrincipal = document.formulario;  
var primerElemento = document.formulario.elemento;
```
- En HTML

```
<form name="formulario">  
    <input type="text" name="elemento" />  
</form>
```

Elementos de un formulario: Texto

- **Cuadro de texto y textarea:** permiten ingreso de texto.
 - La diferencia es que textarea permite ingreso multi-línea
 - El contenido se obtiene y establece mediante la propiedad value
 - `<input type="text" id="texto" />`
`var valor = document.getElementById("texto").value;`
 - `<textarea id="parrafo"></textarea>`
`var valor = document.getElementById("parrafo").value;`

Ingrese su nombre

Ingrese sus comentarios

Limitar cant de caracteres de textarea

- Text tiene propiedad maxlength, pero textarea no tiene algo parecido. Cómo hacerlo con JavaScript?

- En JavaScript

```
function limita(maximoCaracteres) {  
    var elemento = document.getElementById("texto");  
    if(elemento.value.length >= maximoCaracteres ) {  
        return false;  
    }  
    else {  
        return true;  
    }  
}
```

- En HTML

```
<textarea id="texto" onkeypress="return limita(100);"></textarea>
```

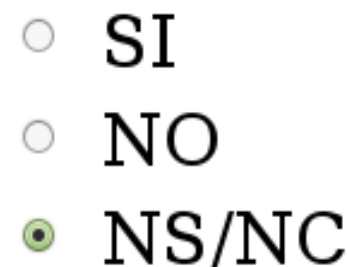
Elementos de un formulario: Radio button

- **RadioButton:** Permite elegir una de varias opciones

```
<input type="radio" value="si" name="pregunta" id="preg_si"/> SI  
<input type="radio" value="no" name="pregunta" id="preg_no"/> NO  
<input type="radio" value="nsnc" name="pregunta" id="preg_nsnc"/>  
NS/NC
```

- Para saber cual fue seleccionado: La propiedad checked devuelve true para el radiobutton seleccionado y false en cualquier otro caso

```
var elementos = document.getElementsByName("pregunta");  
var i=0;  
var seleccionado = "";  
while (i<elementos.length && seleccionado=="") {  
    if (elementos[i].checked)  
        seleccionado = elementos[i].value;  
    i++;  
}
```



☐ SI
☐ NO
☒ NS/NC

Elementos de un formulario: Checkbox

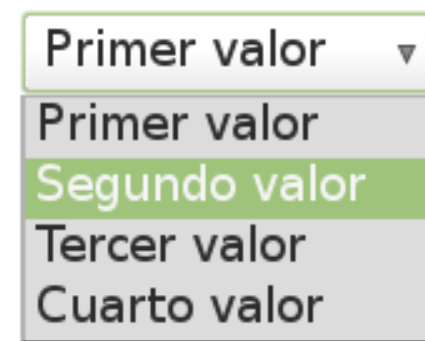
- Se seleccionan de manera independiente
 - `<input type="checkbox" value="condiciones" name="condiciones" id="condiciones"/>` He leído y acepto las condiciones
`<input type="checkbox" value="privacidad" name="privacidad" id="privacidad"/>` He leído la política de privacidad
- Utilizando la propiedad `checked`, es posible comprobar si cada uno ha sido seleccionado:
 - `var elemento = document.getElementById("condiciones");`
`alert(" Elemento: " + elemento.value + "\n Seleccionado: " + elemento.checked);`
`elemento = document.getElementById("privacidad");`
`alert(" Elemento: " + elemento.value + "\n Seleccionado: " + elemento.checked);`

<input type="checkbox"/> He leído y acepto las condiciones
<input type="checkbox"/> He leído la política de privacidad

Elementos de un formulario: Select

- Lista desplegable (<select>) de varios elementos. Permite seleccionar uno solo de ellos

- `<select id="opciones" name="opciones">`
 `<option value="1">Primer valor</option>`
 `<option value="2">Segundo valor</option>`
 `<option value="3">Tercer valor</option>`
 `<option value="4">Cuarto valor</option>`
 `</select>`



- Para saber cuál es la seleccionada
 - `lista.options[lista.selectedIndex].value;`
 devuelve el valor asignado a la opción, es decir 2
 - `lista.options[lista.selectedIndex].text;`
 devuelve el texto asignado a la op. 2: "Segundo valor"

Establecer el foco en un elemento

- Establecer el foco en un elemento
 - Si un cuadro de texto de un formulario tiene el foco, se puede escribir en él sin seleccionarlo con el mouse. Luego se avanza al siguiente con la tecla TAB
- Se suele necesitar poner el foco en el primer elemento del formulario cuando se carga la página. Ej: Para asignarlo al elemento cuyo id es “primero”
 - En JavaScript
`document.getElementById("primero").focus();`
 - En HTML

```
<form id="formulario" action="#">  
  <input type="text" id="primero" />  
</form>
```


Eventos de un formulario

- **onblur**: Deseleccionar un elemento. <button>, <input>, <label>, <select>, <textarea>, <body>.
- **onchange**: Deseleccionar un elemento que se ha modificado. <input>, <select>, <textarea>
- **onfocus**: Seleccionar un elemento. <button>, <input>, <label>, <select>, <textarea>, <body>.
- **onreset**: Inicializar el formulario (borrar todos sus datos). <form>.
- **onsubmit**: Enviar el formulario. <form>
- **onselect**: Selección de Texto. <input>, <textarea>

Validar un formulario antes del submit

- Validar = Llamar a una función cuando el usuario pulsa el botón de envío.
- Comprueba si los valores introducidos cumplen las restricciones impuestas (campos obligatorios con datos, hay valor seleccionado en cada lista desplegable, dirección de mail correcta, fecha apropiada, que se haya introducido un número donde así se requiere, etc.)
- Código en form

```
<form action="" method="" id="" name=""  
  onsubmit="return validacion()"> ... </form>
```
- ¿Cómo funciona? el evento “onsubmit” varía el comportamiento de la acción a realizar: Si onsubmit devuelve true, el formulario se envía; en caso contrario no se envía

Validaciones comunes

- **Validar un campo de texto obligatorio:** chequea cantidad de caracteres mayor que cero y que no se hayan introducido sólo espacios en blanco.

```
valor = document.getElementById("campo").value;  
if( valor == null || valor.length == 0 || /^s+$/ .test(valor) )  
    return false;
```

- **Validar un campo de texto con valores numéricos**

```
valor = document.getElementById("campo").value;  
if( isNaN(valor) )  
    return false;
```

- **Validar que se ha seleccionado una opción de una lista**

```
indice = document.getElementById("opciones").selectedIndex;  
if( indice == null || indice == 0 )  
    return false;
```

Validaciones comunes (2)

- **Validar dirección de email**

```
valor = document.getElementById("campo").value;  
if( !(/^w+([-+.']\w+)*@\w+([-.] \w+)*\.\w+  
([-.] \w+)/.test(valor)) )  
    return false;
```

- **Validar una fecha**

```
- var anio = document.getElementById("anio").value;  
  var mes = document.getElementById("mes").value;  
  var dia = document.getElementById("dia").value;  
  valor = new Date(anio, mes, dia);  
  if( !isNaN(valor) )  
      return false;
```

Referencias

- **Introducción a JavaScript**
Javier Eguiluz. Licencia Creative Commons
<http://librosweb.es/libro/javascript/>
- **El gran libro de HTML5, CSS3 y Javascript**
J. D. Gauchat, 2013, ISBN 9788426719959