JavaScript: Un Curso Rápido

Parte III: Características Específicas del Navegador

Traducción de Dr. Roberto Solís Robles

Temas

- **Manipulación del DOM**
  - Encontrando elementos de interés
  - Insertando XHTML en elementos existentes
- **Lectura de valores de campos de texto**
  - Y otros elementos de entrada
- **Asignación de Manejadores de Eventos**
  - Directamente en XHTML
  - Desde JavaScript, via window.onload
- **Clases específicas a XHTML**

Intro

Puntos Principales

- **Temas más importantes**
  - Como insertar XHTML en una página
    - document.getElementById(…).innerHTML = …;
  - Como leer un valor de un campo de texto
    - escape(document.getElementById(…).value);
  - Como asignar un manejador de eventos
    - <input … onclick="algunaLlamadaFuncion()"/>
      - Esto va en la página XHTML
    - algunElemento.onclick = algunaFuncion;
      - Esto es hecho directamente en JavaScript, probablemente desde el manejador window.onload.

» Esta sección cubre muchos otros temas, y a menos que ya tenga experiencia con JavaScript, no podrá seguirlos todos. Pero, para la mayoría de las aplicaciones Ajax, los temas mencionados arriba son por mucho los más importantes.

Clases Específicas al Navegador

- **Anteriormente**
  - Describimos la sintáxis y capacidades generales de JavaScript
- **En estas diapositivas**
  - Describimos JavaScript especificamente para aplicaciones del navegador
- **En las siguientes diapositivas**
  - Describimos las capacidades de JavaScript para analizar XML
- **Referencias**
  - En línea
    - http://www.w3schools.com/js/
    - http://www.w3schools.com/htmldom/dom_reference.asp
    - http://www.devguru.com/technologies/ecmascript/QuickRef/javascript_intro.html
  - Libros
    - *JavaScript, the Definitive Guide* de David Flanagan

Manipulación del DOM

Resumen

- **Temas más importantes**
  - Como encontrar un elemento con un id dado
    - var elemento = document.getElementById("...");
  - Como insertar XHTML en una página
    - elemento.innerHTML = "…";
      - Puede insertar en todo tipo de elementos (div, span, p, h1, td, etc.).
      - Pero, asegurese de que lo quiere insertar es legal. Por tanto, no intente insertar elementos a nivel de bloque (como h1) en elementos en línea (como span).
- **Temas auxiliares**
  - Encontrar elementos XHTML por nombre
  - Encontrar todos los elementos XHTML de un cierto tipo
  - Info general que representa la pagina en su totalidad (URL, titulo, etc.)

Clase HTMLDocument

- **Visión General**
  - Se obtiene una referencia con la variable predefinida “document”
  - Subclase especializada de la clase Document que será discutida posteriormente
- **Propiedades**
  - title, domain, URL
    - Info sobre la página.
    - document.URL es lo mismo que window.location.href a menos que ocurra una redirección
  - body
    - El elemento body
  - anchors, applets, forms, images, links
    - Arreglos de subelementos, en el orden en que aparecen en el documento. Usualmente es mejor encontrar los elementos por id.
  - cookie, lastModified, referrer
    - En Ajax, es usualmente mejor manipular estos en el servidor. Note que es document.referrer, aún y cuando en el encabezado HTTP es Referer.
  - blah
    - Elemento con nombre="blah" (el primero si es que esta repetido)

HTMLDocument: Métodos

- **getElementById**
  - Encuentra el elemento con el atributo id especificado
- **write, writeln**
  - Dinámicamente inserta texto en el documento
  - Usado en la etiqueta <script> que tenga contenido en body
  - No usados por los manejadores de respuesta de Ajax
    - Usa la propiedad HTMLElement.innerHTML
- **getElementsByName**
  - Regresa arreglo de elementos que tienen un nombre dado
- **getElementsByTagName**
  - Regresa arreglo de elementos que tienen la misma etiqueta dada
    - Insensitivo a mayúsculas/minúsculas
    - Heredado de la clase Document

HTMLElement

- **Subclase de la clase Element. Se obtiene con**
  - document.body, document.getElementsByTagName, document.getElementsByName, document.images. etc.
  - otroElemento.getElementsByTagName, otroElemento.childNodes, otroElemento.firstChild, etc.
- **Propiedades más importantes**
  - id
    - Atributo id
  - nodeName
    - Nombre del elemento (heredado de la clase Node).
  - name
    - El atributo name (para elementos XHTML que lo tengan)
  - innerHTML
    - Propiedad de lectura/escritura que da/genera el texto XHTML que se encuentra dentro del elemento
  - style
    - Objeto de clase CSS2Properties que representa el estilo del elemento
  - className
    - Lista separada por espacios de nombres de clase CSS
- **Metodo**
  - scrollIntoView
    - Desliza el navegador para que el elemento este visible

API de Selectores

- **API definida por W3C para hacer búsquedas basadas en CSS**
  - Encuentra un elemento que concuerde con un selector CSS
  - Encuentra todos los elementos que concuerdan con un selector o varios selectores CSS
  - Detalles en http://www.w3.org/TR/selectors-api/
  - Soportado por Firefox y Chrome
- **Principales dos métodos**
  - document.querySelector
  - document.querySelectorAll

Lectura de Valores de Elementos de Entrada

Clase Form

- **Obteniendo una referencia**
  - Arreglo document.forms
  - Cualquier método o propiedad que regrese un objeto de clase Node (getElementById, childNodes, etc.)
- **Propiedades**
  - elements
    - Arreglo de todos los elementos de entrada en la forma
  - action, encType, method, name, target
    - Corresponde a los atributos XHTML
- **Métodos**
  - submit, reset

Clase Image

- **Obteniendo una referencia**
  - Arreglo document.images, document.nameOfImage
  - Cualquier método o propiedad que regrese un elemento de clase Node (getElementById, childNodes, etc.)
- **Propiedades**
  - src
    - Propiedad de lectura/escritura. Cambiar esta propiedad cambia la imagen.
    - Se precargan las imagenes primero con new Image(...) y se establece su href, para que el navegador tenga las imagenes en cache.
  - name, align, alt, border, height, hspace, ismap, usemap, vspace, width
    - Corresponden a los atributos XHTML

Clase Input

- **Obteniendo una referencia**
  - Arreglo forma.elements
  - Cualquier método o propiedad que regrese un elemento de clase Node (getElementById, childNodes, etc.)
- **Propiedades**
  - name, id, value, type, disabled, form (enclosing form)
    - Para todos los elementos de entrada
  - defaultValue
    - Valor inicial como esta dado en XHTML
  - Propiedades específicas al tipo (consulte API en línea)
    - checked, maxLength, useMap, etc.
- **Metodos**
  - blur/focus (all), click (buttons, checkboxes, radio buttons), select (file, password, text)

La Clase Window

Idea

- **Información general sobre la página**
  - Historia, URL, línea de status
  - No se refiere al DOM (use document para ello)
- **Lo más importante para Ajax**
  - window.onload = algunaFuncion;
    - Codigo a ejecutarse al inicio
  - window.setInterval(algunaFuncion, milisegundos);
    - Codigo a ejecutarse periodicamente
  - window.alert(“algun mensaje")
    - Mensaje a mostrarse en una caja de dialogo en popup. Puede usar solo alert en vez de window.alert.

Clase Window

- **Obteniendo una referencia**
  - Use “window” (o “self”)
- **Propiedades**
  - history
    - Objeto History. No escribible
  - location
    - Objeto Location.
    - Para redirigir el navegador, use
      - location.href = “nueva direccion";
  - status
    - Valor de la línea de status. Lectura/escritura.
  - Tamaño y posición
    - innerWidth, innerHeight, outerWidth, outerHeight, screenX (o screenLeft), screenY (o screenTop)

Clase Window : Métodos

- **alert, confirm, prompt**
  - Muestran cajas de diálogo de diversos tipos
- **print**
  - Invoca un cuadro de dialogo de impresión
- **setInterval, clearInterval [acciones repetitivas]**
  - setInterval(algunaFuncion, milisegundos)
- **setTimeout, clearTimeout [acciones que se ejecutan una sola vez]**
  - setTimeout(algunaFuncion, milisegundos)
- **getComputedStyle**
  - Obtiene información de estilo para un elemento especificado
- **Movimiento**
  - Muchos métods para abrir, cerrar, redimensionar ventanas

Clase Window : El manejador de eventos onload

- **Idea**
  - Codigo a ejecutarse después de haberse cargado la página
    - No puede directamente hacer lo siguiente en el archivo JavaScript que se carga en la sección head:
      - document.getElementById(“algunID").innerHTML = “XHTML";
      - document.getElementById(“algunID").onclick = algunaFuncion;
    - La razón es que el DOM no existe aún, por tanto no puede encontrar los elementos y manipularlos.
  - Forma general
    - window.onload = function() { codigoAEjecutarDespuesDeCargarPagina (); }
- **Detalles**
  - En la siguiente sección (Manejo de Eventos)

Manejo de Eventos

Idea

- **La mayoría de los elementos XHTML tienen atributos onxxx**
  - Designan una función a ejecutarse cuando un evento especificado ocurre
- **Manejadores de eventos generales**
  - onclick, ondblclick
    - Usuario hace un solo click o doble click en elemento
  - onkeydown, onkeypress, onkeyup
    - Usuario presiona una tecla cuando elemento esta activo
  - onmousedown, onmouseup
    - Usuario presiona el mouse sobre el elemento
  - onmouseover, onmouseout
    - Mouse se mueve encima o sale del elemento
  - onmousemove
    - Mouse se mueve mientras esta sobre el elemento

Enfoques para Asignar Manejadores de Eventos

- **Asignar directamente a la propiedad en XHTML**
  - <input type="button" onclick="algunaFuncion()"/>
    - Note los parentensis: esto es una *llamada* a función
  - Dentro de la función, “this” se refiere a la ventana
- **Asignar a la propiedad indirectamente**
  - var elemento = document.getElementById("blah");
  - elemento.onclick = algunaFuncion;
    - Note: no hay parentesis: este es un *valor* de función. Dentro de la función, “this” se refiere al elemento.
    - Este proceso a menudo se hace en window.onload
- **Use addEventHandler o attachEvent**
  - DOM 2: addEventHandler; IE: attachEvent
    - Falta de portabilidad hace este enfoque difícil de manejar

Pros y Cons de los Enfoques de Manejo de Eventos

- **Asignar directamente a la propiedad en XHTML**
  - <input … onclick="algunaFuncion()"/>
    - Mas simple, especialmente para principiantes
    - Un poco mas fácil pasar argumentos al manejador
- **Asignar a la propiedad indirectamente**
  - algunElemento.onclick = algunaFuncion;
    - Mejor separación: todo JavaScript en archivos JS. Archivo XHTML contiene solo XHTML.
      - Este enfoque es a veces llamado “JavaScript no intrusivo”.
    - Mas trabajo de inicialización a menos que se use alguna librería como Prototype o jQuery. Usar window.onload tiene sus problemas (que veremos mas adelante).
    - Un poco mas de trabajo para pasar argumentos la función (hay que usar funciones anónimas)

Código de Ejemplo: Asignar Directamente el Manejador

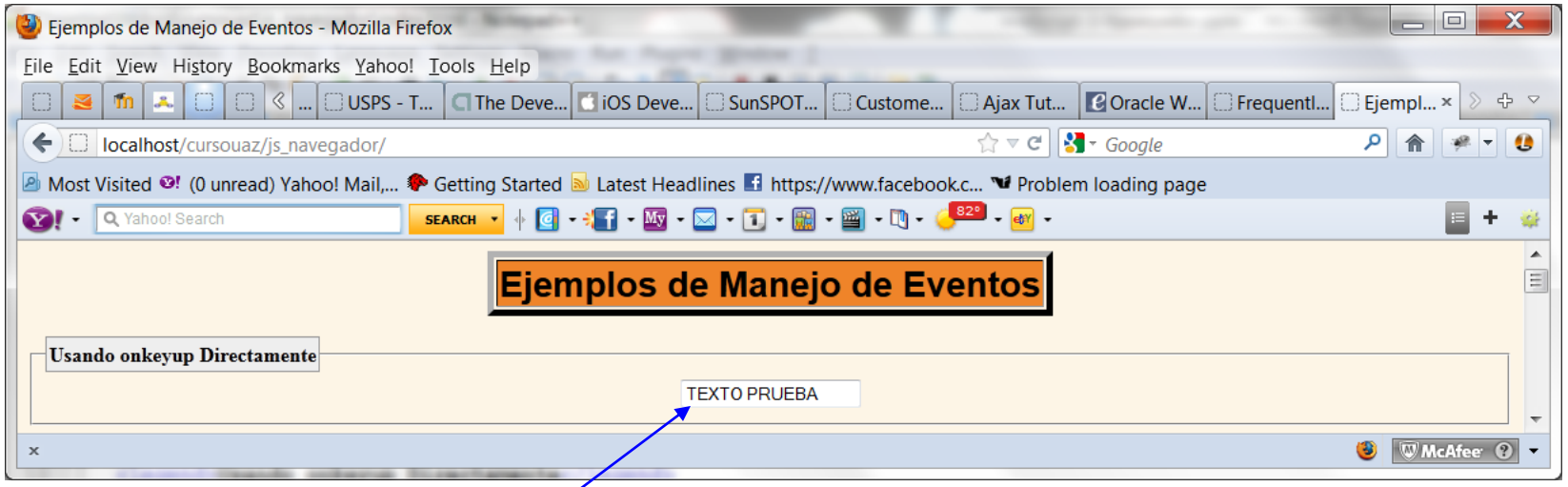
- **JavaScript**

```
function hazMayuscula(textfield) {
  textfield.value = textfield.value.toUpperCase();
}
```

- **HTML**

```
<input type="text" onkeyup="hazMayuscula(this)"/>
```

Salida del Ejemplo: Asignar Directamente el Manejador



La entrada era “Texto Prueba” (no “TEXTO PRUEBA”)

Codigo de Ejemplo: Asignar Indirectamente el Manejador

- **JavaScript**

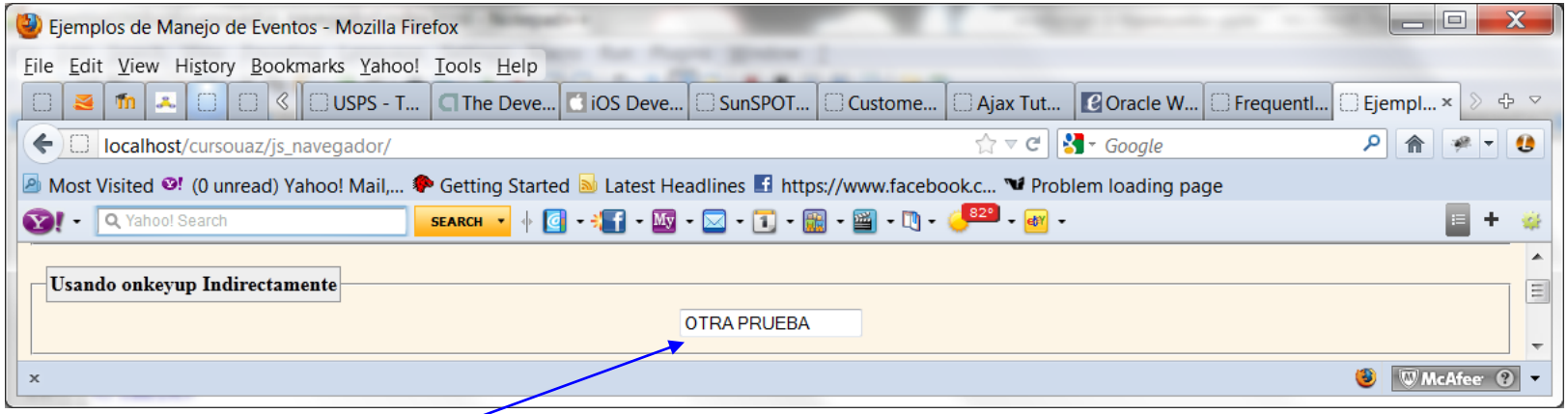
```
function hazMayuscula(textfield) {
  textfield.value = textfield.value.toUpperCase();
}

function hazmeMayuscula() {
  hazMayuscula(this);
}

window.onload = function() {
  document.getElementById("campo-mayuscula").onkeyup =
    hazmeMayuscula;
}
```

Mas adelante veremos como hacer window.onload mas seguro cuando se cargan multiples librerías de JavaScript.

- **HTML**

```
<input type="text" id="campo-mayuscula"/>
```

Salida del Ejemplo: Asignar Indirectamente el Manejador



Entrada era “otra prueba” (no “OTRA PRUEBA”)

Pasando Eventos a los Manejadores de Eventos

- **Idea**
  - JavaScript automáticamente pasa un objeto de evento como último argumento a los manejadores de eventos
    - En versiones viejas de IE se tenía que usar window.event
  - Algunas veces necesita el objeto de evento para obtener información extra
    - General
      - event.type (“click”, “mouseover”, etc.)
      - event.target (elemento sobre el cual ocurrió el evento)
    - Eventos de Mouse
      - event.button (0 = izq, 1 = medio, 2 = der)
      - event.altKey, event.ctrlKey, event.metaKey, event.shiftKey
        - » Booleanos que indican si estaban teclas presionadas cuando ocurrio el evento
      - event.clientX, event.clientY, event.screenX, event.screenY
    - Eventos de teclado
      - event.charCode, event.keyCode (ver ejemplo mas adelante)

Eventos de Teclado (onkeypress)

- **Internet Explorer**
  - Se usa argumento “event” en versiones recientes
  - Se usa “window.event” en versiones anteriores
  - event.charCode es un código numérico de caracter
    - Para caracteres imprimibles, convierta a caracter con String.fromCharCode
- **Otros navegadores**
  - Se usa “event” en todas las versiones
  - event.charCode es un código numérico de caracter si era imprimible
    - Convierta a caracter con String.fromCharCode
  - event.keyCode es un código numérico de caracter si no era imprimible (flechitas, ENTER, Control, etc.)
    - Debe comparar con valores numéricos

Ejemplo: Verificación Portable de Caracteres

- **Meta**
  - Reconocer cuando se presiona la flecha hacia abajo
  - Cuando se presione, haga lo que tuviera que hacer si se diera click en el boton
- **Ideas principales**
  - Defina el manejador para tomar “event” como argumento
  - Use “event” si esta definido, de otra manera, use “window.event”
  - Use charCode si esta definido, de otra manera use keyCode
  - No repita codigo: programaticamente busque en manejador onclick del boton y llamelo

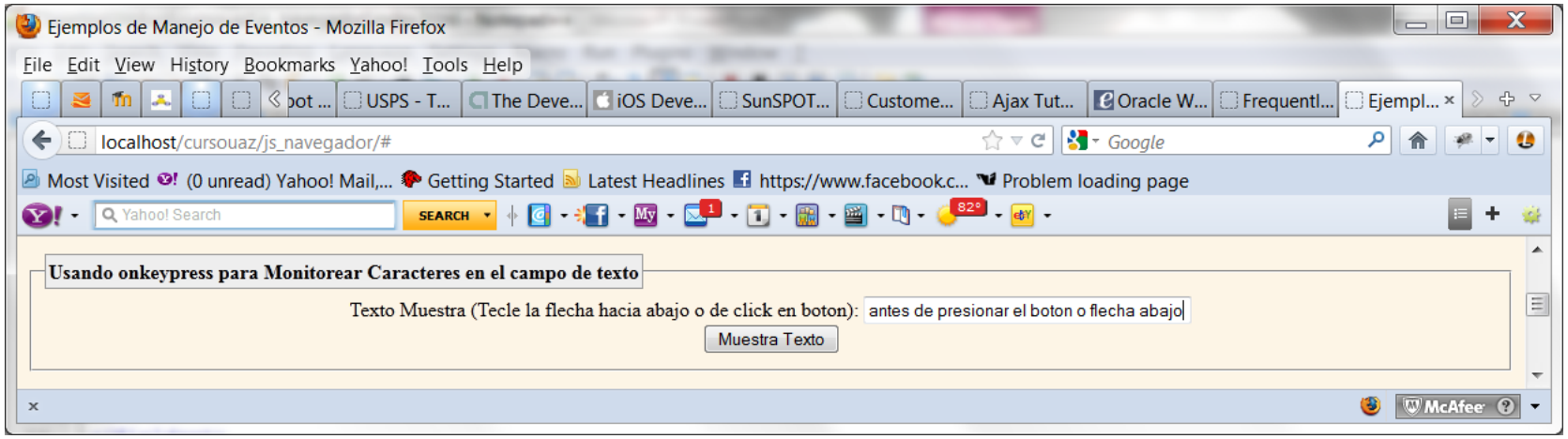
JavaScript de Ejemplo: Verificación Portable de Caracteres

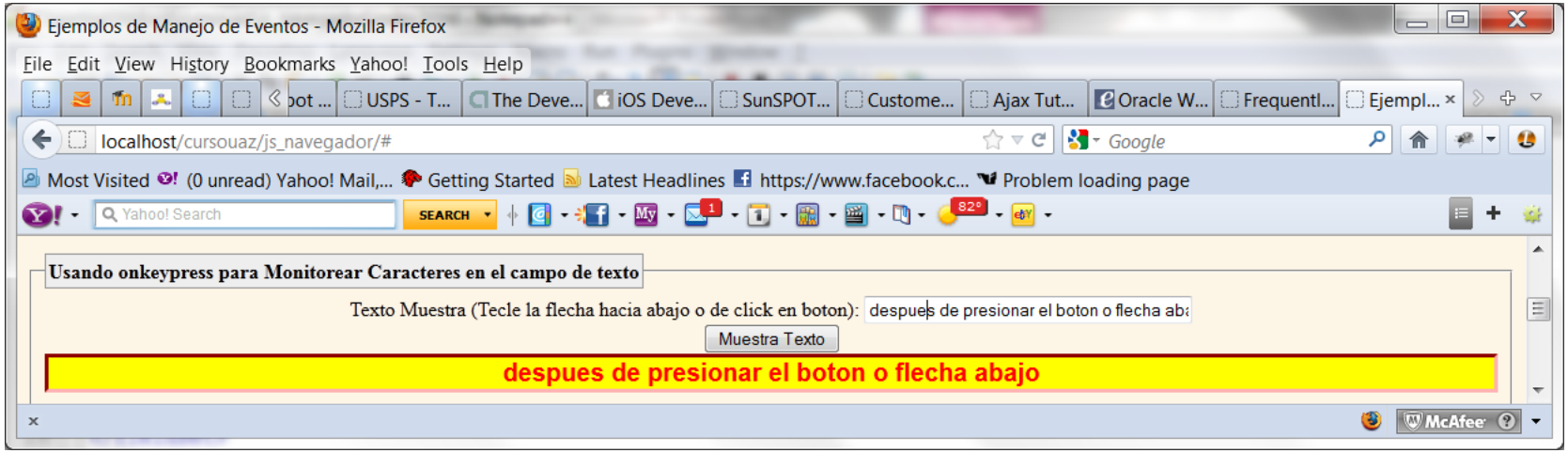
```
function muestraValor(IDEntrada, IDResultado) {
  var html =
    "<div class='sample'>" +
    document.getElementById(IDEntrada).value +
    "</div>"
  document.getElementById(IDResultado).innerHTML =
    html;
}

function hacerAlDarClickEn (IDboton, event) {
  var e = event || window.event;
  var codigo = e.charCode || e.keyCode;
  if (codigo == 40) {  // 40 es Flecha Hacia Abajo
    var f = document.getElementById(IDboton).onclick;
    f();
  }
}
```

JavaScript de Ejemplo: Verificación Portable de Caracteres

```
<form action="#">
   <label for="campotexto-1">
Texto Muestra (Tecle la flecha hacia abajo o de click en
 boton):</label>
   <input type="text" id="campotexto-1" size="40"
        onkeyup="hacerAlDarClickEn('boton-1', event)"/><br/>
   <input type="button" id="boton-1" value="Muestra Texto"
      onclick="muestraValor('campotexto-1', 'div-1')"/><br/>
   <div id="div-1"></div>
 </form>
```

Salida de Ejemplo: Verificación Portable de Caracteres





Eventos del Mouse

- **Se pueden capturar eventos del mouse sobre *cualquier* elemento**
  - Imagenes, enlaces, elementos de entrada, incluso texto normal
- **Propiedades del evento**
  - Para accesar las propiedades, use manejadores de eventos que tomen event como argumento final.
    - Para soportar navegadores viejitos, use window.event si event es undefined, como en el ejemplo anterior
  - altKey, ctrlKey, metaKey, shiftKey
    - Booleanos que indican si la tecla estaba presionada cuando ocurrió el evento
  - button
    - 0 boton izq, 1 medio, 2 der
  - clientX, clientY
    - Coordenadas x e y relativas a la ventana del navegador
  - screenX, screenY
    - Coordenadas x e y relativas al monitor del usuario

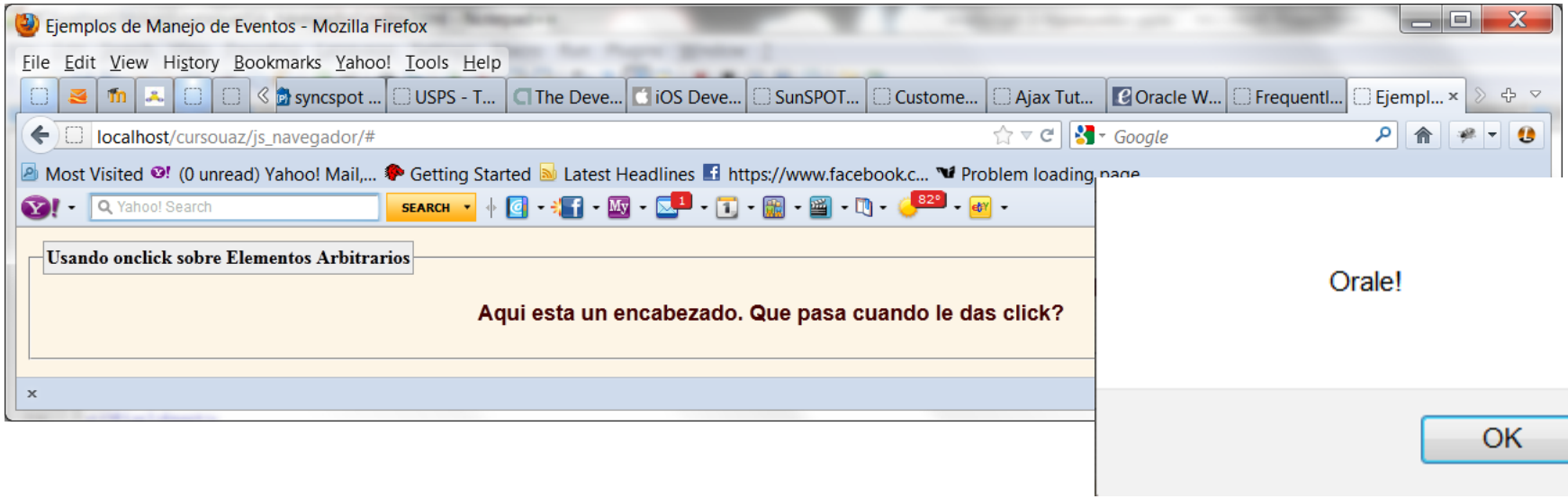
Manejadores de Eventos de Mouse: Ejemplo (JavaScript)

```
function dentro(event) {
  var e = event || window.event;
  var mensaje =
    "<ul class='sample'>" +
    "  <li>Xcliente: " + e.clientX + "</li>" +
    "  <li>Ycliente: " + e.clientY + "</li>" +
    "  <li>Xpantalla: " + e.screenX + "</li>" +
    "  <li>Ypantalla: " + e.screenY + "</li>" +
    "</ul>";
  var region = document.getElementById("regionMensaje");
  region.innerHTML = mensaje;
}

function off() {
  var region = document.getElementById("regionMensaje");
  region.innerHTML = "";
}
```

Manejadores de Eventos de Mouse: Ejemplo (XHTML)

```
<fieldset>
  <legend> Usando onclick sobre Elementos Arbitrarios
   </legend>
  <h2 onclick="alert('Orale!')">
 Aqui esta un encabezado. Que pasa cuando le das click?
  </h2>
</fieldset>
<p/>
<fieldset>
  <legend> Usando onmouseover y onmouseout </legend>
  <h2 onmouseover="dentro(event)" onmouseout="fuera()">
 Aqui esta un encabezado. Que pasa cuando se mueve encima
   de el?
  </h2>
  <div id="regionMensaje"></div>
</fieldset>
```

Manejadores de Eventos de Mouse: Ejemplo (Resultados)



Usando onmouseover y onmouseout

Aqui esta un encabezado. Que pasa cuando se mueve encima de el?

Aqui esta un encabezado. Que pasa cuando se mueve encima de el?

- **Xcliente: 832**
- **Ycliente: 113**
- **Xpantalla: 846**
- **Ypantalla: 300**

Manejadores de Eventos Especializados

- **input**
  - onclick
    - Para botones.
    - También se activa cuando el boton se invoca con el teclado.
  - onchange
    - Para cajas de texto, cuando el cambio es realizado(elemento deja de ser el activo)
      - Use onkeyup para caracteres individuales
  - onblur, onfocus
- **form**
  - onsubmit, onreset
    - Regrese false para prevenir que la forma sea enviada realmente.
    - Ampliamente usada para validaciones de campos del lado del cliente.
- **body**
  - onblur, onerror, onfocus, onload, onresize, onunload
- **img**
  - onabort, onerror, onload

Manejadores de Eventos Especializados: window.onload

- **Propósito**
  - Ejecutar código JavaScript después de cargar la página. Usado para insertar XHTML en ciertas regiones o para anexar manejadores de eventos a ciertos elementos XHTML. Ninguna de estas dos cosa puede hacerse sino hasta que se cargue la página
- **Ejemplo simple (myfile.js)**

```
window.onload = function() {
  document.getElementById("…").onclick = …;
  document.getElementById("…").innerHTML = …;
}
```

window.onload y Multiples Librerías de JavaScript

- **Problema**
  - Asignar directamente a window.onload reemplaza cualquier función window.onload existente.
  - Otra librería podría haber ya estar usando window.onload
- **Solución**
  - Ver si window.onload existe.
    - En Firefox, si no existe window.onload (typeof window.onload == "undefined") …
    - En IE, si no existe window.onload, window.onload es null
    - En cualquiera, se puede probar !window.onload
  - Si es así, tome la función
    - var oldWindowLoadFunction = window.onload;
  - Y llamela antes de trabaje su window.onload
    - oldWindowLoadFunction();

window.onload más Segura

```
if (!window.onload) {
  window.onload = function() {
    document.getElementById("…").onclick = …;
    document.getElementById("…").innerHTML = …;
  };
} else {
  var oldWindowLoadFunction = window.onload;
  window.onload = function() {
    oldWindowLoadFunction();
    document.getElementById("…").onclick = …;
    document.getElementById("…").innerHTML = …;  };
}
```

Problema adicional de window.onload

- **Problema**
  - window.onload se ejecuta después de que la página completa (incluyendo imagenes y hojas de estilo) ha sido cargada. Tiene que esperar hasta que el DOM es analizado, pero no tiene que esperar hasta que las imágenes han sido cargadas.
- **Soluciones**
  - Use window.addEventListener o window.attachEvent
    - Es difícil hacer esto portable entre navegadores
  - Muchas librerías de JavaScript (incluyendo Prototype y jQuery) tienen métodos más simples para definir código que se ejecuta después de que se carga el DOM, pero antes de que se carguen las imágenes y hojas de estilo

Resumen

- **Manipulación del DOM**

  var algunElemento = document.getElementById("…");
  - Muchas formas de encontrar elementos que no sea a través del id

  algunElemento.innerHTML = "…";
- **Lectura de valores de campos de texto**

  var val = algunElemento.value;
- **Asignación de manejadores de eventos**

  <input type="button" onclick="algunaFuncion()"/>

  window.onload = function() {
  document.getElementById("…").onclick = algunaFuncion;
  }

Preguntas?