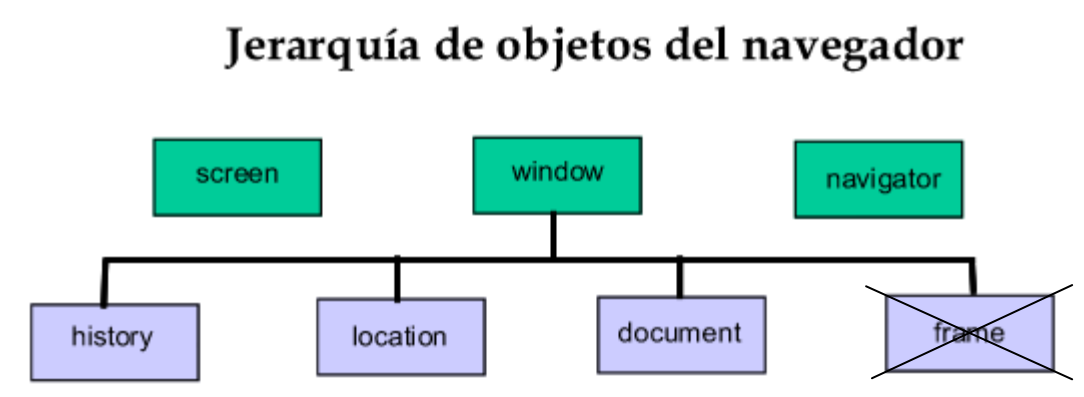
Desarrollo Web en entorno Cliente: JAVASCRIPT

UNIDAD 7. Objetos del navegador

UNIDAD 7. OBJETOS DEL NAVEGADOR

7.1 INTRODUCCIÓN

Vamos a recordar la jerarquía que presentan los objetos del navegador, en la siguiente figura podemos observar gráficamente dicha jerarquía, en lo que se refiere a los objetos del navegador.



Vamos a estudiar, a continuación, todos los objetos representados excepto document que será objeto del siguiente capítulo.

7.2 OBJETO SCREEN

El objeto **screen** permite determinar la resolución del usuario para después modificar el código de la página en base a esa resolución.

El subobjeto **screen** hace referencia a los ajustes del monitor del usuario. La tabla siguiente lista las propiedades disponibles para este objeto (no tiene métodos).

Propiedades del objeto screen

Propiedad	Significado
screen.availHeight	Altura máxima disponible en el monitor del usuario, en píxeles.
screen.availWidth	Anchura máxima disponible en el monitor del usuario, en píxeles.
screen.colorDepth	Bits por píxel disponibles en el monitor del usuario.
screen.height	Altura actual del monitor del usuario, en píxeles.
screen.width	Anchura actual del monitor del usuario, en píxeles.

El siguiente ejemplo prueba las propiedades de screen

```
<body>
  <script language="javascript">
    <!--
    document.write("<b>La anchura máxima es " + screen.width +
           " píxeles.</b><br>")
    document.write("<b> La anchura disponible es " + screen.availWidth +
           " píxeles.</b><br>")
    document.write("<p>")
    document.write("<b> La altura máxima es " + screen.height +
           " píxeles.</b><br>")
    document.write("<b> La altura disponible es " + screen.availHeight +
           " píxeles.</b><br>")
    document.write("<p>")
    document.write("<b>La profundidad de color del píxel es de " +
           screen.pixelDepth + " bits por píxel.</b><br>")
    document.write("<b> La profundidad de color es de " +
           screen.colorDepth + " bits por píxel.</b><br>")

    if (screen.width == 640) {
       /* Cargar el código, a la resolución de 640 x 480 */ }
    else if (screen.width <= 800) {
       /* Cargar el código, a la resolución de 800 x 600 */ }
    else if (screen.width <= 848) {
       /* Cargar el código, a la resolución de 848 x 480 */ }
    else if (screen.width <= 1024) {
       /* Cargar el código, a la resolución de 1024 x 768 */ }
    else if (screen.width <= 1152) {
       /* Cargar el código, a la resolución de 1152 x 864 */ }
    else if (screen.width <= 1200) {
       /* Cargar el código, a la resolución de 1200 x 800 */ }
    else {
       /* Cargar el código, o la página, a la resolución de 1600 x 1200 */ }
    -->
  </script>
</body>
```

Ejemplo anterior (Probar las propiedades del objeto screen**):** **Screen.html**

7.3 OBJETO WINDOW

El objeto window es el objeto más alto en la jerarquía del navegador (navigator es un objeto independiente de todos en la jerarquía), pues todos los componentes de una página web están situados dentro de una ventana.

El objeto window hace referencia a la ventana actual.

A continuación veremos las **Propiedades** y **Métodos** de éste objeto.

Propiedades del objeto window

Propiedad	Significado
closed	Indica la posibilidad de que se haya cerrado la ventana.
defaultStatus	Texto que se escribe por defecto en la barra de estado del navegador.
document	Objeto que contiene la página web que se está mostrando.
history	Objeto historial de páginas visitadas.
innerHeight	Tamaño en pixels del espacio donde se visualiza la página, en vertical.
innerWidth	Tamaño en pixels del espacio donde se visualiza la página, en horizontal.
location	La URL del documento que se está visualizando. Podemos cambiar el valor de esta propiedad para movernos a otra página. Ver también la propiedad location del objeto document.
locationbar	Objeto barra de direcciones de la ventana.
menubar	Objeto barra de menús de la ventana.
name	Nombre de la ventana. Lo asignamos cuando abrimos una nueva ventana.
opener	Hace referencia a la ventana de navegador que abrió la ventana donde estamos trabajando. Se verá con detenimiento en el tratamiento de ventanas con Javascript.
outherHeight	Tamaño en pixels del espacio de toda la ventana, en vertical. Esto incluye las barras de desplazamiento, botones, etc.
outherWidth	Tamaño en pixels del espacio de toda la ventana, en horizontal. Esto incluye las barras de desplazamiento.
personalbar	Objeto barra personal del navegador.
self	Ventana actual.
scrollbars	Objeto de las barras de desplazamiento de la ventana.
status	Texto de la barra de estado.
statusbar	Objeto barra de estado del navegador.
toolbar	Objeto barra de herramientas.
top	Hace referencia a la ventana donde estamos trabajando.
window	Hace referencia a la ventana actual, igual que la propiedad self.

Métodos del objeto window

Método	Significado
alert(texto)	Presenta una ventana de alerta donde se puede leer el texto que recibe por parámetro.
back()	Ir una página atrás en el historial de páginas visitadas. Funciona como el botón de volver de la barra de herramientas.
blur()	Quitar el foco de la ventana actual.
captureEvents(eventos)	Captura los eventos que se indiquen por parámetro.
clearInterval()	Elimina la ejecución de sentencias asociadas a un intervalo indicadas con el método setInterval().
clearTimeout()	Elimina la ejecución de sentencias asociadas a un tiempo de espera indicadas con el método setTimeout().
close()	Cierra la ventana.
confirm(texto)	Muestra una ventana de confirmación y permite aceptar o rechazar.
find()	Muestra una ventanita de búsqueda. (Javascript 1.2 para Netscape)
focus()	Coloca el foco de la aplicación en la ventana.
forward()	Ir una página adelante en el historial de páginas visitadas. Como si pulsásemos el botón de adelante del navegador.
home()	Ir a la página de inicio que haya configurada en el explorador.
moveBy(pixelsX, pixelsY)	Mueve la ventana del navegador los pixels que se indican por parámetro hacia la derecha y abajo.
moveTo(coordenX, coordenY)	Mueve la ventana del navegador a la posición indicada en las coordenadas que recibe por parámetro.
open()	Abre una ventana secundaria del navegador.
print()	Como si pulsásemos el botón de imprimir del navegador.
prompt(pregunta,inicializacion_de_la_respuesta)	Muestra una caja de diálogo para pedir un dato. Devuelve el dato que se ha escrito.
releaseEvents(eventos)	Deja de capturar eventos del tipo que se indique por parámetro.
resizeBy(pixelsAncho,pixelsAlto)	Redimensiona el tamaño de la ventana, añadiendo a su tamaño actual los valores indicados en los parámetros. El primero para la altura y el segundo para la anchura. Admite valores negativos si se desea reducir la ventana.
resizeTo(pixelsAncho,pixelsAlto)	Redimensiona la ventana del navegador para que ocupe el espacio en pixels que se indica por parámetro
routeEvent()	Enruta un evento por la jerarquía de eventos.
scroll(pixelsX,pixelsY)	Hace un scroll de la ventana hacia la coordenada indicada por parámetro. Este método está desaconsejado, pues ahora se debería utilizar scrollTo()
scrollBy(pixelsX,pixelsY)	Hace un scroll del contenido de la ventana relativo a la posición actual.

scrollTo(pixelsX,pixelsY)	Hace un scroll de la ventana a la posición indicada por el parámetro. Este método se tiene que utilizar en lugar de scroll.
setInterval()	Define un script para que sea ejecutado indefinidamente en cada intervalo de tiempo.
setTimeout(sentencia,milisegundos)	Define un script para que sea ejecutado una vez después de un tiempo de espera determinado.
stop()	Como pulsar el botón de stop de la ventana del navegador.

A continuación veremos ejemplos de como utilizar las Propiedades y Métodos anteriores:

- ***Como hacer referencia al objeto window***
- ***Temporizadores setTimeout y setInterval***
- ***El método open().- Abrir ventanas popup***
- ***Como hacer referencia a la página que ha abierto una ventana***
- ***Cierre de una ventana***
- ***Otros métodos***

Como hacer referencia al objeto window

Cuando necesite hacer referencia al objeto window, el hecho de que sea el elemento que se encuentra más arriba en la jerarquía de objetos le proporciona más flexibilidad que otros situados en niveles inferiores.

Para comenzar, puede combinar la palabra clave **window** con las propiedades y métodos:

```
window.Propiedad
window.Metodo()
```

No obstante, el objeto **window** es el objeto predeterminado en Javascript y por tanto puede omitirse; es decir las dos instrucciones superiores equivalen a:

```
Propiedad
Metodo()
```

Javascript ofrece también la propiedad **self**, que también hace referencia a **window** (ventana actual). Esto quiere decir que puede aplicar las propiedades y métodos del objeto **window** al "sub-objeto" **self**:

```
window.self.Propiedad
window.self.Metodo()
```

O también puede usar:

```
self.Propiedad
self.Metodo()
```

Por ejemplo, es lo mismo ***window.close()*** que ***self.close()***.

Temporizadores setTimeout y setInterval

Probar setTimeout y clearTimeout.

```
<html>
<head>
  <script language="javascript">
    <!--
    // Visualizar el mensaje
    status="I.E.S. El Rincón"

    // Fijar la demora
    var id_temporizador = setTimeout("parar_estado()", 5000)

    // Función llamada cuando el tiempo de demora transcurre
    function parar_estado()
      {
      status = ""
      }

    // Fijar el manejador del evento click
    document.onclick = cancelar_temporizador

    // Ejecutar la función cuando el usuario provoque el click
    function cancelar_temporizador()
      {
      clearTimeout(id_temporizador)
      alert("El temporizador timeout ha sido cancelado.")
       document.onclick=null
      }
    -->
  </script>
</head>
<body>
<b>Haga click dentro de la página antes de 5 segundos
        para cancelar el temporizador</b>
</body>
</html>
```

Ejemplo anterior (Probar setTimeout y clearTimeout**):** **CancelarTimeout.html**

Como poner un mensaje móvil en la barra de estado con setInterval

```
  <script language="javascript">
    <!--
    // Mensaje en la barra de estado
    var mensaje_barra_estado = "Instituto de E.S. El Rincón"
    // Retraso
```

```
    var tiempo_demora = 50
    // Donde comienza el mensaje
    var posicion_inicio = 100
    // Máximo número de scroll
    var maximo_numero_scrolls = 2
    // Rellenar el mensaje con espacios en blanco para obtener
    // la posición de inicio
    for (var contador = 1; contador <= posicion_inicio; contador++)
      {
      mensaje_barra_estado = " " + mensaje_barra_estado
      }
    var posicion_actual = 0
    var total_scrolls = 0
    // Lanzar el temporizador
    var id_temporizador = setInterval("mensaje_scroll()", tiempo_demora)
    // Función llamada después de cada intervalo

    function mensaje_scroll()
      {
      // Incrementar la posición actual
      posicion_actual++
      // Podemos avanzar todavía?
      if (posicion_actual < mensaje_barra_estado.length)
         // En caso afirmativo, visualizar el mensaje
         status = mensaje_barra_estado.substring(posicion_actual)
      else
        {
         // Si no es así actualizar los valores de las variables de control
         posicion_actual = 0
         total_scrolls++
         // Hemos alcanzado el máximo número de scrolls?
         if (total_scrolls == maximo_numero_scrolls)
            {
            // Si la respuesta es afirmativa, parar el temporizador
            clearInterval(id_temporizador)
            status = ""
            return
            }
        }
      }
    -->
  </script>
```

Ejemplo anterior (Poner mensaje móvil en la barra de estado**)**: **MensajeMovil.html**
(Sólo se ve en Internet Explorer, activar barra de estado)
(Recordar que en chrome esto no funciona no tiene barra de estado)

El método open().- Abrir ventanas popup

El método **open(URL, nombre, características)** abre la URL que le pasemos como primer parámetro en una ventana de nombre **nombre**. Si esta ventana no existe, abrirá una ventana nueva en la que mostrará el contenido con las características especificadas.

Las características que podemos elegir para la ventana que queramos abrir son las siguientes:

Parámetros para dar forma a una ventana

Parámetro	Significado
width	Ajusta el ancho de la ventana (no solo el área cliente). En pixels.
height	Ajusta el alto de la ventana (no solo el área cliente).
outerWidth	Nos dice el ancho *total* de la ventana en pixels.
outerHeight	Nos dice el alto *total* de la ventana el pixels.
top	Indica la posición de la ventana. En concreto es la distancia en pixels que existe entre el borde superior de la pantalla y el borde superior de la ventana.
left	Indica la posición de la ventana. En concreto es la distancia en pixels que existe entre el borde izquierdo de la pantalla y el borde izquierdo de la ventana.
scrollbars	Para definir de forma exacta si salen o no las barras de desplazamiento. scrollbars=NO hace que nunca salgan. Scrollbars=YES hace que salgan (siempre en ie y solo si son necesarias en NTS).
resizable	Establece si se puede o no modificar el tamaño de la ventana. Con resizable=YES se puede modificar el tamaño y con resizable=NO se consigue un tamaño fijo.
directories(barra directorios)	A partir de aquí se enumeran otra serie de propiedades que sirven para mostrar o no un elemento de la barra de navegación que tienen los navegadores más populares, como podría ser la barra de menús o la barra de estado. Cuando ponemos el atributo=YES estamos forzando a que ese elemento se vea. Cuando ponemos atributo=NO lo que hacemos es evitar que ese elemento se vea.
location(barra direcciones)	
menubar(barra de menus)	
status(barra de estado)	
titlebar(barra de titulo)	
toolbar(barra de herramientas)	

Abrir popup.

```
<html>
<head>
  <script language="javascript">
    <!--
    var nueva_ventana
    function abrir_ventana()
      {
      nueva ventana = window.open("ventana.htm", "MiNuevaVentana";
      alert("El nuevo nombre de la ventana es " + nueva_ventana.name);
      nueva_ventana.focus();
      }
    -->
  </script>
</head>
<body>
```

```
<a href="javascript:void(0)"  onClick="abrir_ventana()">
Abrir la página principal en una nueva ventana
</a>
</body>
</html>
```

Ejemplo anterior (Abrir popup**): AbrirPopup.html**

Abrir popup centrado.

```
<html>
<head>
  <script language="javascript">
    <!--
    function abrir_ventana_centrada()
      {
      var altura_ventana = 620
      var anchura_ventana = 990
      var abscisa_ventana = (screen.availWidth / 2) - (anchura_ventana / 2)
      var ordenada_ventana = (screen.availHeight / 2) - (altura_ventana /2)
      var dimensiones_ventana = "height=" + altura_ventana +
            ",width=" + anchura_ventana +
            ",left=" + abscisa_ventana +
            ",top=" + ordenada_ventana
      window.open("ventana.htm", "", dimensiones_ventana)
      }
    -->
  </script>
</head>
<body>
<a href="javascript:void(0)" onClick="abrir_ventana_centrada()">
Abrir página principal en una ventana centrada
</a>
</body>
</html>
```

Ejemplo anterior (Abrir popup centrado**): AbrirPopupCentrado.html**

Probar características de una ventana popup.

```
<html>
<head>
  <script language="javascript">
    <!--
    function abrir_ventana(caracteristicas_ventana)
      {
```

```
      var caracteristicas = caracteristicas_ventana +
            ",height=620,width=990,top=30,left=10"
      window.open("ventana.htm", "", caracteristicas)
      }
    -->
  </script>
</head>
<body>
<a href="javascript:void(0)" onClick="abrir_ventana('directories')">
Abrir una nueva ventana con la característica "directories" activa
</a>
<br>
<a href="javascript:void(0)" onClick="abrir_ventana('location')">
Abrir una nueva ventana con la característica "location" activa
</a>
<br>
<a href="javascript:void(0)" onClick="abrir_ventana('menubar')">
Abrir una nueva ventana con la característica "menubar" activa
</a>
<br>
<a href="javascript:void(0)" onClick="abrir_ventana('resizable')">
Abrir una nueva ventana con la característica "resizable" activa
</a>
<br>
<a href="javascript:void(0)" onClick="abrir_ventana('scrollbars')">
Abrir una nueva ventana con la característica "scrollbars" activa
</a>
<br>
<a href="javascript:void(0)" onClick="abrir_ventana('status')">
Abrir una nueva ventana con la característica "status" activa
</a>
<br>
<a href="javascript:void(0)" onClick="abrir_ventana('toolbar')">
Abrir una nueva ventana con la característica "toolbar" activa
</a>
</body>
</html>
```

Ejemplo anterior (Probar características de ventana popup, ojo con el navegador**)**:
(Funciona en Internet explorer y opera)

AbrirPopupCaracteristicas.html

NOTA: La sentencia **"javascript:void(0);"** suele ocurrir cuando un enlace no tiene una URL asignada, pero se utiliza para hacer referencia a una función JavaScript. Si no hay una URL conectada al enlace, el navegador no sabrá qué hacer y te llevará al inicio de la página. Utilizar "javascript:void(0)" en este caso puede hacer que determinados navegadores muestren tu página web de una forma extraña. En su lugar, debes utilizar la sentencia **"return false;"** para decirle al navegador que no debe buscar una URL.

Como hacer referencia a la página que ha abierto una ventana

Cuando abras una página en una nueva ventana, habrá ocasiones en las que necesites hacer referencia a algo de la ventana original. Podría ser un valor de un formulario o la dirección de la página abierta. También puede que quiera establecer de nuevo el foco sobre la ventana original.

Puedes realizar todas esas tareas usando la propiedad **opener** del objeto **window**.
window.opener
Esta propiedad devuelve una referencia al objeto **window** original.

Para ver como funciona consulte el siguiente listado

```
<a href="javascript:void(0)"
       onClick="window.open('Opener2.htm','')">
Abrir Opener2.htm
</a>
```

El archivo **opener2.htm** contiene el script que se muestra a continuación:

```
<body>
<b>La dirección de la página que ha abierto esta ventana es:
    </b><br>
<script language="JavaScript" type="text/javascript">
<!--
  document.write(opener.location)
//-->
</script>
<p>
<a href="javascript:void(0)" onClick="opener.focus()">
  Devolver el foco a la ventana original
</a>
</body>
```

Ejemplo anterior (Como hacer referencia a la página que ha abierto una ventana, ver también Opener1-copia y Opener2-copia)**):** **Opener1.html**

Nota importante: en Chrome ***opener*** funciona usándolo con ***onLoad=...***
Y cuando lo ejecutamos desde servidor

Cierre de una ventana

Cuando su código ya ha sido usado para una ventana que ya está abierta, se debería cerrar para evitar que desordene la pantalla del usuario de manera innecesaria.

Para este tipo de situaciones, puede cerrar la ventana ejecutando el método **close()** del objeto **window**.

Cuando la ventana no ha sido abierta desde programa, sino que es una ventana original, llamada a través de su dirección, no es posible cerrarla sin que salga la petición de confirmación de dicho cierre en un cuadro de diálogo tipo **confirm()**. Esta situación ha sido normalizada en Javascript para proteger al usuario y evitar que se le presenten formatos de ventana, desde el principio, que él no ha elegido.

Probar características de una ventana popup.

```
<html>
<head>
  <script language="javascript">
    <!--
    var nueva_ventana
    function abrir_ventana()
      {
      nueva_ventana = window.open("Opener2.htm", "")
      }

    function cerrar_ventana()
      {
      if (!nueva_ventana)
        alert("La ventana nunca ha sido abierta.")
      else if (nueva_ventana.closed)
        alert("La ventana ya está cerrada.")
      else
        nueva_ventana.close()
      }
    -->
  </script>
</head>
<body>
<a href="javascript:void(0)" onClick="abrir_ventana()">
    Abrir Opener2.htm
</a>
<p>
<a href="javascript:void(0)" onClick="cerrar_ventana()">
    Cerrar Opener2.htm
</a>
</body>
</html>
```

Ejemplo anterior (Cierre de una ventana**)**: **CerrarVentana.html**

Otros métodos

En el siguiente ejemplo, estudiaremos los métodos que sirven para mover la ventana, redimensionarla y hacer scroll en su interior.

Ejemplo anterior (Cierre de una ventana**)**: **MoverScrollRedimensionar.html**

7.4 OBJETO FRAME (obsoleto para HTML 5)

Con la llegada de HTML 5 los frame/frameset quedan obsoletos utilizándose en su lugar ** .** Algunos usan los marcos flotantes <**iframes>** o también los **<object>**.

Ver en el siguiente enlace: **Como abandonar los frames de html en 3 simples pasos**
http://cucarachasracing.blogspot.com.es/2012/02/como-abandonar-los-frames-de-html-en-3.html

Ver en siguiente enlace: **Marcos flotantes <iframe> en HTML 5**
http://www.w3schools.com/tags/tag_iframe.asp

Ver en siguiente enlace: **HTML5 nuevos atributos del elemento IFRAME**
http://webdesign.about.com/od/iframes/a/html5-iframe-attributes.htm

Ver en siguiente enlace : **Ejemplo de <iframe></iframe>**
http://www.hscripts.com/tutorials/html5/iframe-tag.html

<iframe src="http://www.jqslider.com/" name="JQuery Slide Show Effects" height="125" width="400"></iframe>

Ver en siguiente enlace : **HTML5 ejemplo <iframe>, Cómo utilizar <iframe> etiqueta de HTML5**
http://www.roseindia.net/tutorial/html/html5/HTML5iFrameTag.html

Ver en el siguiente enlace: **El sustituto del 'iframe' en HTML5**

```
<object data =" " type="text/html" height=" ">
    <param …. />
</object>
```

http://es.globedia.com/sustituto-iframe-html5
http://codexexempla.org/articulos/2007/object_alternativa_iframe.php

Ejemplos:

IFRAME

```
<iframe allowFullScreen allowTransparency="true" class="vzaar-video-player"
frameborder="0" height="216" id="vzvd-401491" mozallowfullscreen
name="vzvd-401491" src="http://view.vzaar.com/401491/player"
title="vzaar video player&quot type="text/html" webkitAllowFullScreen
width="384"></iframe>
```

OBJECT Tag

```
<div class="vzaar_media_player">
<object data="http://view.vzaar.com/401491/flashplayer" height="216" id="vzvd-401491"
type="application/x-shockwave-flash" width="384">
```

```
<param name="wmode" value="transparent" />
<param name="allowFullScreen" value="true" />
<param name="movie" value="http://view.vzaar.com/401491/flashplayer" />
<param name="flashvars"
value="endLink=vzaar.com%2Fsignup&showplaybutton=true&border=none&endText=Sign+Up+For+A+FREE+TRIAL+or+call+our+sales+team+on+1-877-831-7110" />
<param name="allowScriptAccess" value="always" />
<video controls height="216" id="vzvid" onclick="this.play();"
poster="http://view.vzaar.com/401491/image" preload="none" src="http://view.vzaar.com/401491/video"
width="384">
</video>
</object>
</div>
```

Objeto <iframe> en javascript

Como actualizar dos iframes al mismo tiempo
Actualizar_dos_iframes.htm

Accediendo a los objetos iframe con Javascript
http://sviudes.blogspot.com.es/2010/01/accediendo-los-iframes-con-javascript.html

Como acceder a un elemento de un iframe con JavaScript
http://www.antisacsor.com/articulo/10_89_como-acceder-a-un-elemento-de-un-iframe-con-javascript

Acceder a un elemento html dentro de un iframe con javascript
http://www.codigogratis.com.ar/acceder-a-un-elemento-html-dentro-de-un-frame-con-javascript/

Ver ejemplos en: Unidad 7 → Ejemplos Unidad 7 → carpeta iframes

INICIO OBJETO FRAME (obsoleto para HTML 5)

Antes de nada, vamos a ver las ventajas e inconvenientes del uso de los marcos (frames):
http://www.desarrolloweb.com/articulos/936.php

Ventajas de usar frames

- La navegación de la página será más rápida. Aunque la primera carga de la página sería igual, en sucesivas impresiones de páginas ya tendremos algunos marcos guardados , que no tendrían que volverse a descargar.
- Crear páginas del sitio sería más rápido. Como no tenemos que incluir partes de código como la barra de navegación, título, etc. crear nuevas páginas sería un proceso mucho más rápido.

- Partes de la página (como la barra de navegación) se mantienen fijas y eso puede ser bueno, para que el usuario no las pierda nunca de vista.
- Estas mismas partes visibles constantemente, si contienen enlaces, pueden servir muy bien para mejorar la navegación por el sitio.
- Mantienen una identidad del sitio donde se navega, pues los elementos fijos conservan la imagen siempre visible.

Inconvenientes de usar frames

- Quitan espacio en la pantalla. El espacio ocupado por los frames fijos se pierde a la hora de hacer páginas nuevas, porque ya está utilizado. En definiciones de pantalla pequeña o dispositivos como Palms, este problema se hace más patente.
- Fuerzan al visitante a entrar por la declaración de frames. Si no lo hacen así, sólo se vería una página interior sin los recudros. Estos recuadros podrían ser insuficientes para una buena navegación por los contenidos y podrían no conservar una buena imagen corporativa.
- La promoción de la página sería, en principio, más limitada. Esto es debido a que sólo se debería promocionar la portada, pues si se promocionan páginas interiores, podría darse en caso de que los visitantes entrasen por ellas en lugar de por la portada, creandose el problema descrito en el punto anterior.
- A mucha gente les disgustan pues no se sienten libres en la navegación, pues entienden que esas partes fijas están limitando su movilidad por la web. Este efecto se hace más patente si la página con frames tiene enlaces a otras páginas web fuera del sitio y, al pulsar un enlace, se muestra la página nueva con los marcos de la página que tiene frames.
- Algunos navegadores no los soportan. Esto no es muy habitual, pero si estamos haciendo una página que queramos que sea totalmente accesible deberíamos considerarlo importante.
- Los bookmarks o favoritos no funcionan correctamente en muchos casos. Si queremos incluir un favorito a una página de un frame que no sea la portada podemos encontrar problemas.
- Puede que el botón de atrás del navegador no se comporte como deseamos.
- Si quieres actualizar más de un frame con la pulsación de un enlace deberás utilizar Javascript. Además los scripts se pueden complicar bastante cuando se tienen que comunicar varios frames entre si.

Todos sabemos que la ventana del navegador puede ser dividida en varios frames que contengan cada uno de ellos un documento en el que mostrar contenidos diferentes. Al igual que con las ventanas, cada uno de estos frames puede ser nombrado y referenciado, lo que nos permite cargar documentos en un marco sin que esto afecte al resto.

Las **propiedades** y **métodos** de este objeto son los mismos que las del objeto window.

Propiedades del objeto frame

Propiedad	Significado
closed	Indica la posibilidad de que se haya cerrado la ventana.
defaultStatus	Texto que se escribe por defecto en la barra de estado del navegador.
document	Objeto que contiene el la página web que se está mostrando.
frame	Un objeto frame de una página web. Se accede por su nombre.
frames array	El vector que contiene todos los frames de la página. Se accede por su índice a partir

	de 0.
history	Objeto historial de páginas visitadas.
innerHeight	Tamaño en pixels del espacio donde se visualiza la página, en vertical.
innerWidth	Tamaño en pixels del espacio donde se visualiza la página, en horizontal.
length	Numero de frames de la ventana.
location	La URL del documento que se está visualizando. Podemos cambiar el valor de esta propiedad para movernos a otra página. Ver también la propiedad location del objeto document.
locationbar	Objeto barra de direcciones de la ventana.
menubar	Objeto barra de menús de la ventana.
name	Nombre de la ventana. Lo asignamos cuando abrimos una nueva ventana.
opener	Hace referencia a la ventana de navegador que abrió la ventana donde estamos trabajando. Se verá con detenimiento en el tratamiento de ventanas con Javascript.
outherHeight	Tamaño en pixels del espacio de toda la ventana, en vertical. Esto incluye las barras de desplazamiento, botones, etc.
outherWidth	Tamaño en pixels del espacio de toda la ventana, en horizontal. Esto incluye las barras de desplazamiento.
parent	Hace referencia a la ventana donde está situada el frame donde estamos trabajando. La veremos con detenimiento al estudiar el control de frames con Javascript.
personalbar	Objeto barra personal del navegador.
self	Ventana o frame actual.
scrollbars	Objeto de las barras de desplazamiento de la ventana.
status	Texto de la barra de estado.
statusbar	Objeto barra de estado del navegador.
toolbar	Objeto barra de herramientas.
top	Hace referencia a la ventana donde está situada el frame donde estamos trabajando. Como la propiedad parent.
window	Hace referencia a la ventana actual, igual que la propiedad self.

Métodos del objeto frame

Método	Significado
alert(texto)	Presenta una ventana de alerta donde se puede leer el texto que recibe por parámetro.
back()	Ir una página atrás en el historial de páginas visitadas. Funciona como el botón de volver de la barra de herramientas.
blur()	Quitar el foco de la ventana actual.
captureEvents(eventos)	Captura los eventos que se indiquen por parámetro.
clearInterval()	Elimina la ejecución de sentencias asociadas a un intervalo indicadas con el método setInterval().

clearTimeout()	Elimina la ejecución de sentencias asociadas a un tiempo de espera indicadas con el método setTimeout().
close()	Cierra la ventana.
confirm(texto)	Muestra una ventana de confirmación y permite aceptar o rechazar.
find()	Muestra una ventanita de búsqueda. (Javascript 1.2 para Netscape)
focus()	Coloca el foco de la aplicación en la ventana.
forward()	Ir una página adelante en el historial de páginas visitadas. Como si pulsásemos el botón de adelante del navegador.
home()	Ir a la página de inicio que haya configurada en el explorador.
moveBy(pixelsX, pixelsY)	Mueve la ventana del navegador los pixels que se indican por parámetro hacia la derecha y abajo.
moveTo(pixelsX, pixelsY)	Mueve la ventana del navegador a la posición indicada en las coordenadas que recibe por parámetro.
open()	Abre una ventana secundaria del navegador.
print()	Como si pulsásemos el botón de imprimir del navegador.
prompt(pregunta,inicializacion_de_la_respuesta)	Muestra una caja de diálogo para pedir un dato. Devuelve el dato que se ha escrito.
releaseEvents(eventos)	Deja de capturar eventos del tipo que se indique por parámetro.
resizeBy(pixelsAncho,pixelsAlto)	Redimensiona el tamaño de la ventana, añadiendo a su tamaño actual los valores indicados en los parámetros. El primero para la altura y el segundo para la anchura. Admite valores negativos si se desea reducir la ventana.
resizeTo(pixelsAncho,pixelsAlto)	Redimensiona la ventana del navegador para que ocupe el espacio en pixels que se indica por parámetro
routeEvent()	Enruta un evento por la jerarquía de eventos.
scroll(pixelsX,pixelsY)	Hace un scroll de la ventana hacia la coordenada indicada por parámetro. Este método está desaconsejado, pues ahora se debería utilizar scrollTo()
scrollBy(pixelsX,pixelsY)	Hace un scroll del contenido de la ventana relativo a la posición actual.
scrollTo(pixelsX,pixelsY)	Hace un scroll de la ventana a la posición indicada por el parámetro. Este método se tiene que utilizar en lugar de scroll.
setInterval()	Define un script para que sea ejecutado indefinidamente en cada intervalo de tiempo.
setTimeout(sentencia,milisegundos)	Define un script para que sea ejecutado una vez después de un tiempo de espera determinado.
stop()	Como pulsar el botón de stop de la ventana del navegador.

Referencias entre marcos

Cuando estamos en un conjunto de marcos hay que saber referenciar un marco desde otro, a continuación exponemos diferentes casos:

Referencias padre a hijo:

```
window.nombre_marco
self.nombre_marco
nombre_marco
```

También podemos hacer referencia a los hijos a través del array frames, los siguientes ejemplos son equivalentes y hacen referencia al primer hijo:

```
window.frames[0]
self.frames[0]
frames[0]
```

Referencias hijo a padre:

```
window.parent
self.parent
parent
```

Referencias hermanos:

```
window.parent.nombre_marco
self.parent.nombre_marco
parent.nombre_marco
```

Referencia a marcos anidados: Hay que situarse en el marco padre del que hay que referenciar, por ejemplo en una relación nieto a tío:

```
window.parent.parent.nombre_marco
self.parent.parent.nombre_marco
parent.parent.nombre_marco
```

Referencia al marco de más alto nivel: La propiedad **top** devuelve una referencia al objeto **window** de más alto nivel cargado en el navegador.

FIN OBJETO FRAME (obsoleto para HTML 5)

7.5 OBJETO LOCATION

Este objeto contiene la URL actual así como algunos datos de interés respecto a esta URL. Su finalidad principal es, por una parte, modificar el objeto **location** para cambiar a una nueva URL, y extraer los componentes de dicha URL de forma separada para poder trabajar con ellos de forma individual si es el caso. Recordemos que la sintaxis de una URL era:

protocolo://maquina_host[:puerto]/camino_al_recurso/recurso#ancla?parte_search

A continuación veremos las **Propiedades** y **Métodos** de éste objeto

Propiedades del objeto location

Propiedad	Significado
hash	Cadena que contiene el nombre del enlace, dentro de la URL.
host	Cadena que contiene el nombre del servidor y el número del puerto, dentro de la URL.
hostname	Cadena que contiene el nombre de dominio del servidor (o la dirección IP), dentro de la URL.
href	Cadena que contiene la URL completa.
pathname	Cadena que contiene el camino al recurso, dentro de la URL.
port	Cadena que contiene el número de puerto del servidor, dentro de la URL.
protocol	Cadena que contiene el protocolo utilizado (incluyendo los dos puntos), dentro de la URL.
search	Cadena que contiene la información pasada en una llamada a un script CGI, dentro de la URL.

Métodos del objeto location

Método	Significado
reload()	Vuelve a cargar la URL especificada en la propiedad href del objeto location.
replace(cadenaURL)	Reemplaza el historial actual mientras carga la URL especificada en cadenaURL.

En el siguiente ejemplo comprobamos las diferentes propiedades de location: hostname, href, ... ,search, protocol.

```
<body>
  <script language="javascript">
    <!--
    for (var sPropiedad in location)
      {
      document.write('<b>'+sPropiedad+'</b> = '+
              location[sPropiedad]+'<br>');
      }
    -->
  </script>
</body>
```

Ejemplo anterior (Propiedades del objeto location**)**: **llamarLocation.html**

En el siguiente ejemplo se observa como cambiar las imagenes en miniatura de la página principal de un museo (pequeñas imágenes .gif) por su versión completa (imágen .jpg de mayor calidad). Tenga en cuenta que este es el segundo método (**search**) que hemos estudiado de pasar datos entre páginas; el primero fué a través de **input type="hidden"** y en el siguiente capítulo estudiaremos el tercer método (**cookies**).

Ejemplo que hará el alumnado en los ejercicios de la unidad (Miniaturas de cuadros en un museo**)**: **MiniaturaPintoresGrancanarios1.html**

7.6 OBJETO HISTORY

Este objeto se encarga de almacenar una lista con los sitios por los que se ha estado navegando, es decir, guarda las referencias de los lugares visitados. Se utiliza, sobre todo, para movernos hacia delante o hacia atrás en dicha lista.

A continuación veremos las **Propiedades** y **Métodos** de éste objeto

Propiedades del objeto history

Propiedad	Significado
current	Cadena que contiene la URL completa de la entrada actual en el historial.
next	Cadena que contiene la URL completa de la siguiente entrada en el historial.
length	Entero que contiene el número de entradas del historial (i.e., cuántas direcciones han sido visitadas).
previous	Cadena que contiene la URL completa de la anterior entrada en el historial.

Métodos del objeto history

Método	Significado
back()	Vuelve a cargar la URL del documento anterior dentro del historial.
forward()	Vuelve a cargar la URL del documento siguiente dentro del historial.
go(posición)	Vuelve a cargar la URL del documento especificado por **posicion** dentro del historial. **posicion** puede ser un entero, en cuyo caso indica la posición relativa del documento dentro del historial; o puede ser una cadena de caracteres, en cuyo caso representa toda o parte de una URL que esté en el historial.

En el ejemplo que sigue navegaremos por las distintas páginas de la aplicación usando **location** y los métodos de **history**.

Recuerde que estas sentencias son equivalentes:

history.back()
history.go(-1)

Al igual que las siguientes:

history.forward()
history.go(1)

Ejecutar **history.go(0)** no es lo mismo que **location.reload()**. El **método history.go(0)** simplemente refresca la página, lo que significa que no cambian los datos del formulario que ha introducido el usuario.

Mucha gente piensa que el historial retiene cada página visitada en la sesión actual del navegador, pero esto no es cierto. Tenga en cuenta que cuando vuelve a una página anterior en el historial y navega después a otra diferente, la pila hacia atrás estará vacía.

```
<html>
<head>
  <script language="javascript">
    <!--
    function Ir()
       {
       var indice=document.getElementById('slcPaginas').selectedIndex;
       var location=document.getElementById('slcPaginas').options[indice].text;
       window.location.href=location;
       }
     -->
  </script>
</head>
<body onload="divHistorial.innerText=history.length">
<table border="1">
  <caption><h2>Página 1</h2></caption>
  <tr height="400">
    <td width="600">
      <h3>Contenido de Página 1</h3>
      <select id="slcPaginas" onchange="Ir()">
        <option> Elija la página a la que quiere ir</option>
```

```
        <option> Pagina1.htm </option>
        <option> Pagina2.htm </option>
        <option> Pagina3.htm </option>
      </select>
    </td>
  </tr>
  <tr>
    <td>
     <table border="0">
       <tr>
         <td>Páginas en el historial </td>
         <td>
           <div id="divHistorial" style="width=30px;text-align:center;">
         </td>
         <td width="50"></td>
         <td>
           <input type="button" value="<" onclick="history.back()"
                                style="width:30px;">
         </td>
         <td width="20"></td>
         <td>
           <input type="button" value=">" onclick="history.forward()"
                                style="width:30px;">
         </td>
         <td width="50"></td>
         <td>Movimiento relativo a la página +-</td>
         <td width="20"></td>
         <td><input type="text" id="txtPagina" style="width:30px;"></td>
         <td width="20"></td>
         <td><input type="button" value="IR"
                    onclick="eval('history.go('+txtPagina.value+')')"
                    style="width:30px;"></td>
       </tr>
     </table>
    </td>
  </tr>
</table>
</body>
 </html>
```

Ejemplo anterior (Navegación a través del historial de páginas visitadas**)**: **Pagina1.html**

7.7 OBJETO NAVIGATOR

Este objeto simplemente nos da información relativa al navegador que esté utilizando el usuario.

A continuación veremos las **Propiedades** y **Métodos** de éste objeto.

Propiedades del objeto navigator

Propiedad	Significado
appCodeName	Cadena que contiene el nombre del código del cliente.
appName	Cadena que contiene el nombre del cliente.
appVersion	Cadena que contiene información sobre la versión del cliente.
cookieEnabled	Determina si las cookies están habilitadas en el navegador
language	Devuelve el idioma del navegador
onLine	Determina si el navegador está online
platform	Cadena con la plataforma sobre la que se está ejecutando el programa cliente.
product	Devuelve el nombre del motor del navegador
userAgent	Cadena que contiene la cabecera completa del agente enviada en una petición HTTP. Contiene la información de las propiedades appCodeName y appVersion.
plugins	array de plugins del navegador

Métodos del objeto navigator

Método	Significado
javaEnabled()	Devuelve **true** si el cliente permite la utilización de Java, en caso contrario, devuelve **false**.

Advertencia!

La información del objeto navigator a menudo puede ser engañosa, y no debe ser utilizado para detectar las versiones del navegador porque:

- Los datos del navegador pueden ser cambiados por el propietario navegador
- Algunos navegadores se identifican incorrectamente a sí mismos
- Los navegadores no pueden informar de los nuevos sistemas operativos

Detección del navegador

Desde el objeto navegator puede ser engañosa la detección del navegador.

Dado que los diferentes navegadores soportan diferentes objetos, se puede utilizar objetos para detectar navegadores. Por ejemplo, ya que sólo Opera soporta la propiedad "window.opera", podemos utilizar esto para identificar Opera.

Ejemplo: if (window.opera) {... alguna acción ...}

A continuación veremos ejemplos de como utilizar las Propiedades y Métodos anteriores:

- **Propiedades generales del navegador**
- **Cómo listar los plugins que tenemos instalados en el navegador**
- **Cómo determinar el nombre del navegador**
- **Cómo determinar la versión del navegador**
- **La propiedad userAgent**
- **Cómo determinar el sistema operativo.**
- **El husmeador.**

Propiedades generales del navegador

```
<script language="javascript">

        txt = "<p>Browser CodeName: " + navigator.appCodeName + "</p>";
        txt+= "<p>Browser Name: " + navigator.appName + "</p>";
        txt+= "<p>Browser Version: " + navigator.appVersion + "</p>";
        txt+= "<p>Cookies Enabled: " + navigator.cookieEnabled + "</p>";
        txt+= "<p>Browser Language: " + navigator.language + "</p>";
        txt+= "<p>Browser Online: " + navigator.onLine + "</p>";
        txt+= "<p>Platform: " + navigator.platform + "</p>";
        txt+= "<p>User-agent header: " + navigator.userAgent + "</p>";
        txt+= "<p>User-agent language: " + navigator.systemLanguage + "</p>";

        document.getElementById("example").innerHTML=txt;

</script>
```

Ejemplo anterior: **propiedades_navigator.html**

Cómo listar los plugins que tenemos instalados en el navegador

```
<script type="text/javascript">
  for (var i=0; i < navigator.plugins.length; i++) {
        document.write("<br/>"+navigator.plugins[i].name+"<br/>");

        if (navigator.plugins[i].description) {
               document.write("<p style='font-weight:bold;'>"+
        navigator.plugins[i].description+ "</p>");
        }
  }
</script>
```

Ejemplo anterior: **plugins-navegador.html**

Cómo determinar el nombre del navegador

El camino más directo es la propiedad **appname** del objeto navigator, que devuelve lo siguiente:

- Para Internet Explorer, la cadena **Microsoft Internet Explorer**.
- Para otros navegadores, una cadena que contiene el nombre del navegador.

Es, en este último punto, donde las cosas tieneden a complicarse. Por ejemplo Opera según la configuración de la opción **Preferences** puede devolver **Microsoft Internet Explorer**, **Netscape** e incluso **Opera**.

Debido a estas ambigüedades, les recomendamos que no se fíe de la propiedad **appName**. En vez de ello use la propiedad **userAgent**.

Cómo determinar la versión del navegador

El camino directo es **appVersion**, pero no siempre conduce al camino deseado.

Para saber porque ocurre esto, observe el siguiente script:

```
  <body>
     <script language="javascript">
       <!--
       document.write("<b>La cadena appName completa es:
                    </b><br>")
       document.write(navigator.appName)
       document.write("<p>")
       var version_mayor = parseInt(navigator.appVersion)
       var version_completa = parseFloat(navigator.appVersion)
       document.write("<b>La cadena appVersion completa es:
                    </b><br>")
       document.write(navigator.appVersion)
       document.write("<p>")
       document.write("<b>El número principal de la versión es:
                    </b><br>")
       document.write(version_mayor)
```

```
      document.write("<p>")
      document.write("<b>El número completo de versión es:
                    </b><br>")
      document.write(version_completa)
      -->
    </script>
  </body>
```

Ejemplo anterior (Visualizar appName y appVersion**)**: **appnameYappVersion.html**

Esta eficaz técnica no siempre funciona con Internet Explorer, pues las versiones 5 y 6 devuelven un 4.

La propiedad userAgent

La propiedad **userAgent** es, por ahora, la más utilizada del objeto **navigator** para devolver cadenas, ya que contiene abundante información acerca del navegador, versión y sistema operativo:

En la siguiente lista les mostramos algunos de los valores devueltos:

Mozilla/2.0 (compatible; MSIE 3.0b; Windows 3.1) en Internet Explorer 3.0 bajo Windows 3.11
Mozilla/4.0 (compatible; MSIE 5.0; Mac_PowerPC) en Internet Explorer 5.0 bajo Macintosh PowerPC
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727) en Internet Explorer 6.0 bajo Windows XP.
Mozilla/5.0 (X11; U; Linux 2.2.16-3 i686; en-US; m18) Gecko/20010129 en Netscape 6.0, bajo Linux.
Mozilla/5.0 (Windows; U; Windows NT 5.1; es-ES; rv:1.8.0.7) Gecko/20060909 Firefox/1.5.0.7 en Firefox 1.5.0.7 bajo Windows XP.
Mozilla/5.0 (Windows NT 4.0; U) Opera 5.0 [en] en Opera 5.02 bajo Windows NT 4.0.
Mozilla/3.01 (compatible;) en un navegador desconocido compatible con Netscape 3.01.

Aunque existen situaciones en las que la propiedad userAgent no ofrece el nombre del navegador (véase, por ejemplo, la última entrada de la tabla anterior), éstas son bastantes raras.

En el siguiente script usamos **navigator.userAgent.indexOf("MSIE")** para detectar la cadena **MSIE**. En la práctica no nos preocuparemos de la cadena exacta sino que configuraremos una variable lógica que nos da verdadero para los navegadores más usados.

```
  <body>
    <script language="javascript">
      <!--
      // Una de estas variables lógicas
      // tendrá un valor verdadero basándose en el nombre del navegador
      var es_ie = false
      var es_ns = false
      var es_opera = false
      var es_webtv = false
      var es_compatible = false
      var es_firefox = false
      // Trabajaremos con minúsculas para facilitar las cosas
      var user_agent = navigator.userAgent.toLowerCase()
      // Utilizamos indexOf() para examinar la cadena userAgent
      // y obtener signos reveladores del nombre del navegador
```

```
      if (user_agent.indexOf("opera") != -1) { es_opera = true }
      else if (user_agent.indexOf("webtv") != -1) { es_webtv = true }
      else if (user_agent.indexOf("msie") != -1) { es_ie = true }
      else if (user_agent.indexOf("firefox") != -1) { es_firefox = true }
      else if (user_agent.indexOf("mozilla") != -1) {
      // Para "mozilla", necesitamos barajar algunas posibilidades; primero
      if ((user_agent.indexOf("compatible") == -1) &&
          (user_agent.indexOf("spoofer") == -1) &&
          (user_agent.indexOf("hotjava") == -1)) {
          es_ns = true
      }
      else { es_compatible = true }
      }
      document.write("<b>La cadena userAgent completa es:</b><br>")
      document.write(navigator.userAgent)
      document.write("<p>")
      document.write("<b>El nombre del navegador es:</b><br>")
      if (es_ie) { document.write("Internet Explorer") }
      else if (es_ns) { document.write("Netscape") }
      else if (es_opera) { document.write("Opera") }
      else if (es_webtv) { document.write("WebTV") }
      else if (es_firefox) { document.write("Firefox") }
      else if (es_compatible) { document.write("Compatible Netscape") }
      -->
   </script>
</body>
```

Ejemplo anterior (Detectar navegador a partir de userAgent**)**: **userAgent.html**
(Otra versión de este programa lo podemos ver en
Unidad 7→Ejemplos Unidad 7→userAgent→Desarrollado-Moisés)

Cómo determinar el sistema operativo.

Otro elemento importante que podemos determinar con la propiedad userAgent es el sistema operativo. El siguiente listado es una sección parcial de un script que determina el sistema operativo.

```
<body>
   <script language="javascript">
     <!--
     // Una de estas variables lógicas tendrá un valor
     // verdadero, basándose en el sistema operativo
     var es_win31 = false
     var es_win95 = false
     var es_win98 = false
     var es_winme = false
     var es_winnt = false
     var es_win2000 = false
     var es_winxp = false
     var es_mac68k = false
     var es_macppc = false
     var es_linux = false
     var es_other_os = false
     // Trabajaremos con minúsculas para facilitar las cosas
     var user_agent = navigator.userAgent.toLowerCase()
```

```
      // Utilizamos indexOf() para examinar la cadena userAgent
      // y obtener signos reveladores del sistema operativo
      // WINDOWS 3.1
      if ((user_agent.indexOf("windows 3.1") != -1) ||
            (user_agent.indexOf("win16") != -1) ||
            (user_agent.indexOf("16bit") != -1) ||
            (user_agent.indexOf("16-bit") != -1)) { es_win31 = true }
    ..............................
    ..............................
      // WINDOWS XP
      else if ((user_agent.indexOf("windows nt 5.1") != -1) ||
            (user_agent.indexOf("winnt 5.1") != -1)) { es_winxp = true }
      // WINDOWS 2000
      else if ((user_agent.indexOf("windows nt 5.0") != -1) ||
            (user_agent.indexOf("winnt 5.0") != -1)) { es_win2000 = true }
    ..............................
    ..............................
      // LINUX
      else if (user_agent.indexOf("linux") != -1) { es_linux = true }
      // OTROS SISTEMAS OPERATIVOS
      else { es_other_os = true }
      document.write("<b>La cadena userAgent completa es:</b><br>")
      document.write(navigator.userAgent)
      document.write("<p>")
      document.write("<b>El sistema operativo es:</b><br>")
      if (es_win31) { document.write("Windows 3.x") }
      else if (es_win95) { document.write("Windows 95") }
      else if (es_win98) { document.write("Windows 98") }
      else if (es_winme) { document.write("Windows Me") }
      else if (es_winnt) { document.write("Windows NT") }
      else if (es_win2000) { document.write("Windows 2000") }
      else if (es_winxp) { document.write("Windows XP") }
      else if (es_mac68k) { document.write("Mac 680x0") }
      else if (es_macppc) { document.write("Mac PPC") }
      else if (es_linux) { document.write("Linux") }
      else if (es_webtv) { document.write("WebTV") }
      else if (es_other_os) { document.write("Otro sistema operativo") }
      -->
    </script>
  </body>
```

Ejemplo anterior (Detectar el sistema operativo a partir de userAgent**)**:
SistemaOperativo.html

El husmeador.

Todo eso junto nos servirá para averiguarlo todo, como se observa en el ejemplo siguiente:

Ejemplo anterior (Detectar navegador, sistema, plataforma, … etc**)**: **Husmeador.html**

Lo correcto sería unir todo en un archivo denominado **husmeador.js** que adjuntaríamos en nuestros scripts.