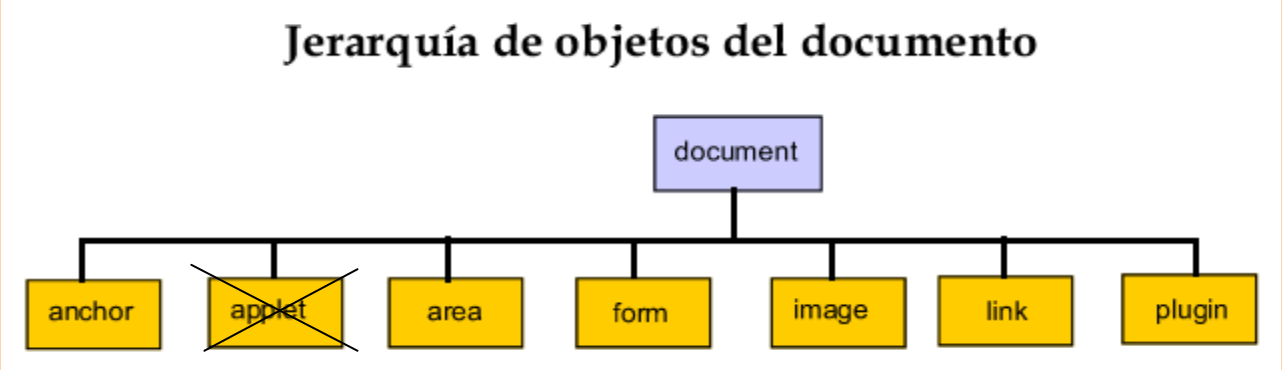
Desarrollo Web en entorno Cliente: JAVASCRIPT

UNIDAD 8. OBJETOS DE DOCUMENTO

UNIDAD 8. OBJETOS DE DOCUMENTO

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 documento.



Vamos a estudiar, a continuación, los objetos representados más utilizados, es decir document, link, anchor e image.

El objeto form ya se estudió en el capítulo de objetos del formulario.

El tag <applet> ya no existe en HTML 5, obsoleto. Se usa el tag <object> en su lugar.

El objeto area no lo veremos [La etiqueta <area> define un área dentro de una imagen de mapa con zonas seleccionables. El elemento <area> siempre está anidado dentro de una etiqueta <map>. Nota: El atributo usemap en el <img> etiqueta se asocia con la <map> atributo de nombre del elemento, y crea una relación entre la imagen y el mapa.]

El objeto plugins ya lo hemos visto en la unidad anterior (con el objeto navigator), también existe para document.

De forma esquemática, la jerarquía del navegador la podemos representar de la manera siguiente (al lado está la directiva HTML con que se incluyen en el documento objetos de este tipo, cuando exista esta directiva):

- window
  - history
  - location
  - document <BODY> ... </BODY>
    - anchor <A NAME="..."> ... </A>
    - ~~applet <APPLET> ... </APPLET>~~
    - area <MAP> ... </MAP>
    - form <FORM> ... </FORM>
      - button <INPUT TYPE="button">
      - checkbox <INPUT TYPE="checkbox">
      - fileUpload <INPUT TYPE="file">
      - hidden <INPUT TYPE="hidden">
      - password <INPUT TYPE="password">
      - radio <INPUT TYPE="radio">
      - reset <INPUT TYPE="reset">
      - select <SELECT> ... </SELECT>
        - options <INPUT TYPE="option">
      - submit <INPUT TYPE="submit">
      - text <INPUT TYPE="text">
      - textarea <TEXTAREA> ... </TEXTAREA>
    - image <IMG SRC="...">
    - link <A HREF="..."> ... </A>
    - plugin <EMBED SRC="...">
  - ~~frame <FRAME>~~
- navigator

8.1 OBJETO document

El objeto document se construye a partir de la marca BODY de HTML. Sobre él pueden actuar los eventos: onClick, onDblClick, onKeyDown, onKeyPress, onKeyUp, onMouseDown, onMouseUp.

Cuando un documento HTML se carga en un navegador web, se convierte en un **objeto de documento** .

El objeto del documento proporciona propiedades y métodos para acceder a todos los objetos de nodo, desde dentro JavaScript.

Consejo: El documento es una parte del objeto de la ventana y se puede acceder como window.document.

El objeto del documento es compatible con todos los navegadores

Propiedades del objeto document

Arrays de elementos *contenidos en el documento actual:*

plugins array de plugins del documento
layers array que contiene las capas del documento
images array que contiene las imágenes cargadas en el documento
links array que contiene los enlaces del documento
forms array que contiene los formularios definidos en el documento
(ya visto unidades anteriores)
anchors array que contiene los puntos de anclaje (anchor) del documento

Propiedad	Significado
alinkColor	Color de los enlaces activos
anchor	Un ancla de la página. Se consigue con la etiqueta <a name="nombre_del_ancla">. Se accede por su nombre.
anchors array	Un array de las anclas del documento.
bgColor	El color de fondo del documento.
classes	Las clases definidas en la declaración de estilos CSS.
cookie	Una cookie
domain	Nombre del dominio del servidor de la página.
fgColor	El color del texto. Para ver los cambios hay que reescribir la página.
form	Un formulario de la página. Se accede por su nombre.
forms array	Un array con todos los formularios de la página.
ids	Para acceder a estilos CSS.
Image	Una imagen de la página web. Se accede por su nombre.
images array	Cada una de las imágenes de la página introducidas en un array.
lastModified	La fecha de última modificación del documento.
linkColor	El color de los enlaces.

Link	Un enlace de los de la página. Se accede por su nombre.
links array	Un array con cada uno de los enlaces de la página.
location	La URL del documento que se está visualizando. Es de solo lectura.
referrer	URL del documento desde el que se llamó al documento actual (si el usuario utilizó un enlace para llegar al documento actual).
tags	Estilos definidos a las etiquetas de HTML en la página web.
title	El titulo de la página.
embed	Un elemento de la página incrustado con la etiqueta <embed>. Se accede por su nombre.
embeds array	Todos los elementos de la página incrustados con <embed>. Define un contenedor para una aplicación externa o contenido interactivo (un plug-in)
URL	Lo mismo que location, pero es aconsejable utilizar location ya que URL no existe en todos los navegadores.
vlinkColor	El color de los enlaces visitados.

Métodos del objeto document

Método	Significado
close()	Cierra el flujo de datos abierta por el método open y hace que se muestren todos los elementos
open("type MIME"))	Abre la escritura sobre un documento. Abre un búfer para recoger los resultados de los métodos write y writeln. Algunos tipos MIME son:

Tipo MIME	Descripción
text/html	Archivo de tipo ASCII cuya paginación se hace mediante datos HTML
text/plain	Archivo de tipo ASCII cuya paginación se hace mediante retornos del carro.

write()	Escribe texto / datos en el documento
writeln()	Escribe igual que el método write(), aunque coloca un salto de línea al final del texto

Empleo de colores

La lista de propiedades del objeto document incluye varios elementos que cambian los colores por defecto del documento, equivalentes en Javascript a los atributos de color de la etiqueta <body>.

El siguiente listado muestra el código que abre una nueva ventana. La idea aquí es utilizar dicha ventana para seleccionar un esquema de color para la ventana original.

```
<script language="JavaScript" type="text/javascript">
<!--
var nueva_ventana
function abrir_ventana()
  {
  if (!ventana_esta_abierta())
    nueva_ventana = window.open("SelectorColorNombre.htm", "",
                                "width=300,height=175")
  else
    nueva_ventana.focus()
  }

function ventana_esta_abierta()
  {
  if (!nueva_ventana)
    return false
  else if (nueva_ventana.closed)
    return false
  else
    return true
  }
//-->
</script>
```

El listado que selecciona el color por nombre es el que se muestra a continuación; el formulario incluye dos listas:

- **nombre_atributo** contiene un elemento para cada una de las cinco propiedades de color del objeto **document**
- **nombre_color** incluye los 139 nombres de color definidos y soportados por **Internet Explorer** (para abreviar el listado muestra sólo los tres primeros elementos y el último; consulta el listado **ListadoColoresX11.htm** para obtener la lista completa.

```
<html>
  <head>
  </head>
  <body>
  <center>
  <form>
    <b>Elija una propiedad:</b><br>
    <select name="nombre_atributo">
      <option value="alinkColor">alinkColor</option>
      <option value="linkColor">linkColor</option>
      <option value="vlinkColor">vlinkColor</option>
      <option value="bgColor">bgColor</option>
      <option value="fgColor">fgColor</option>
    </select>
    <p>
    <b>Elija un nuevo color:</b><br>
    <select name="nombre_color">
      <option value="aliceblue">aliceblue</option>
      <option value="antiquewhite"<antiquewhite</option>
      <option value="aqua">aqua</option>
      ................................
```

```
      ..................................
      <option value="yellowgreen">yellowgreen</option>
    </select>
    <p>
    <input type="button" value="Cambiar el color"
              onClick="fijar_color(this.form)">

    <input type="button" value="Cerrar" onClick="self.close()">
  </form>
  </center>
    <script language="javascript">
      <!--
      function fijar_color(formulario)
        {
        // Almacenar las dos listas en variables
        var lista_colores = formulario.nombre_color
        var lista_atributos = formulario.nombre_atributo
        // Recuperar la opción seleccionada de la lista
        var atributo_seleccionado =
             lista_atributos.options[lista_atributos.selectedIndex].value
        // Recuperar el color seleccionado
        var color_seleccionado =
             lista_colores.options[lista_colores.selectedIndex].value
        // Cambiar el color de la propiedad seleccionada
        opener.document[atributo_seleccionado] = color_seleccionado
        }
      -->
    </script>
  </body>
  </html>
```

Ejemplo anterior (Cambiar atributos de color del documento usando el nombre del color): **ColoresNombre.html**

Ejemplo que hará el alumnado en los ejercicios de la unidad: El alumnado hará el ejercicio anterior utilizando las características RGB en vez del nombre del color, para seleccionarlo.

La página siguiente define un conjunto de elementos HTML (imágenes, formularios y enlaces) e incluye el botón Características que al pulsarse hace que se ejecute la función MostrarCaracteristicasDocumento(). Ésta crea una nueva ventana en la que se incluyen alguna de las características asociadas al documento original.

```
  <html>
  <head>
    <script language="javascript">
      <!--
      function MostrarCaracteristicasDocumento(documento)
        {
        var NuevaVentana;
        var Atributos="resizable=yes, menubar=yes, left=100, top=200,";
        Atributos+=" width=500, height=200";
```

```
      var NumeroFormularios=documento.forms.length;
      var NumeroAnclas=documento.anchors.length;
      var NumeroEnlaces=documento.links.length;
      var NumeroImagenes=documento.images.length;
      NuevaVentana=window.open("","NuevaVentana",Atributos);
      NuevaVentana.document.write("<h3>CARACTERÍSTICAS DEL DOCUMENTO</h3>");
      NuevaVentana.document.write("Número de formularios: ");
      NuevaVentana.document.write(NumeroFormularios+"<br>");
      NuevaVentana.document.write("Número de anclas locales: ");
      NuevaVentana.document.write(NumeroAnclas+"<br>");
      NuevaVentana.document.write("Número de enlaces: ");
      NuevaVentana.document.write(NumeroEnlaces+"<br>");
      NuevaVentana.document.write("Número de imágenes: ");
      NuevaVentana.document.write(NumeroImagenes+"<br>");
      }
    -->
  </script>
</head>
<body>
<h4 align="center">Documento con distintos elementos</h4>
<a href="../../index.htm">Ir a página principal</a><br>
<a href="#Formulario1">Formulario 1</a><br>
<a href="#Formulario2">Formulario 2</a><hr>
<h3 align="center"><a name="Formulario1">
         <strong>Formulario 1</strong></a></h3>
<form name="Formulario_1" action="">
  <table align="center" cellspacing="25">
    <tr>
       <td rowspan="5">
             <img src="../../Imagenes/LujanPerez.gif"></td>
       <td>Nombre</td>
       <td><input type="text" name="nombre"></td>
    </tr>
    <tr>
       <td>Apellidos</td>
       <td><input type="text" name="apellidos"></td>
    </tr>
    <tr>
       <td>Direccion</td>
       <td><input type="text" name="direccion"></td>
    </tr>
    <tr>
       <td>Edad</td>
       <td><input type="text" name="edad"></td>
    </tr>
    <tr>
       <td align="center">
             <input type="submit" value="Enviar"></td>
       <td align="center">
             <input type="reset" value="Borrar"></td>
    </tr>
  </table>
</form>
<h3 align="center"><a name="Formulario2">
         <strong>Formulario 2</strong></a></h3>
<form name="Formulario_2" action="">
  <table align="center" cellspacing="25">
    <tr>
      <td rowspan="3">
```

```
          <img src="../../Imagenes/TomasMorales.gif"></td>
        <td>Profesión</td>
        <td><input type="text" name="profesion"></td>
      </tr>
      <tr>
        <td>Aficiones</td>
        <td><input type="text" name="aficiones"></td>
      </tr>
      <tr>
        <td align="center">
              <input type="submit" value="Enviar"></td>
        <td align="center">
              <input type="reset" value="Borrar"></td>
      </tr>
    </table>
  </form>
  <hr><p align="center">
  <button type="button"
            onClick="MostrarCaracteristicasDocumento(document)">
    Características
  </button>
  </body>
  </html>
```

Ejemplo anterior (Mostrar características del documento):
CaracteristicasDocumento.html

8.2 OBJETO link

Un objeto link es un objeto que enlaza a otra URL mediante hipertexto.

Cada enlace presente en un documento corresponde a un objeto que se coloca en la **matriz links**. Así, se accederá al primer enlace mediante document.links[0], al segundo mediante document.links[1], y así sucesivamente.

El objeto link carece de métodos pero no de propiedades como veremos a continuación.

A continuación analizaremos las propiedades de un link. Para ello, utilizaremos el primer link que se encuentra en esta página que es:

```
<a name="enlace1" href="cap6.htm" target="window">capítulo anterior</a>
```

Este objeto engloba todas las propiedades que tienen los enlaces externos al documento actual.

Es capaz de responder a tres eventos: onClick, onMouseOver y onMouseOut.

Propiedades del objeto link

Propiedad	Significado
length	Tamaño del objeto link
hash	Es una cadena con el nombre del enlace, dentro de la URL.
host	Es una cadena con el nombre del servidor y número de puerto, dentro de la URL.
hostname	Es una cadena con el nombre de dominio del servidor (o la dirección IP) dentro de la URL.
href	Es una cadena con la URL completa.
pathname	Es una cadena con el camino al recurso, dentro de la URL.
port	Es una cadena con el número de puerto, dentro de la URL.
protocol	Es una cadena con el protocolo usado, incluyendo los : (los dos puntos), dentro de la URL.
search	Es una cadena que tiene la información pasada en una llamada a un script, dentro de la URL. Parte del URL que caracteriza una consulta (a continuación de ?)

8.3 OBJETO anchor

Una anchor o ancla es una parte de texto que define una etiqueta para ser referenciada desde otro lugar por un enlace hipertexto.

Cada ancla presente en un documento corresponde a un objeto que se coloca en la matriz anchors. Así, se accederá al primer enlace mediante document.anchors[0], al segundo mediante document.anchors[1], y así sucesivamente.

Este objeto carece de métodos y la única propiedad que tiene es length que indica el tamaño del objeto anchor. También carece de eventos asociados al mismo.

Este objeto engloba todas las propiedades que tienen los enlaces internos al documento actual.

Propiedades del objeto anchor

Propiedad	Significado
length	Número de anclas existentes en el documento
name	Nombre del ancla.

En la página CaracteristicasEnlaces.htm se definen dos enlaces, uno externo y uno local. Cuando se pulsa el botón Características enlaces se ejecuta la función MostrarCaracteristicasenlaces() que crea una nueva ventana en la que se incluye determinada información sobre los dos enlaces que contiene el documento original:

```
<html>
<head>
  <script language="javascript">
    <!--
    function MostrarCaracteristicasEnlaces(documento)
      {
      var NuevaVentana;
      var Atributos="resizable=yes, menubar=yes, left=100, top=50,";
      Atributos+=" width=700, height=540";
      var NumeroEnlaces=documento.links.length;
      NuevaVentana=window.open("","NuevaVentana",Atributos);
      NuevaVentana.document.write
                 ("<h3>CARACTERÍSTICAS DE LOS ENLACES</h3>");
      NuevaVentana.document.write("Número de enlaces: ");
      NuevaVentana.document.write(NumeroEnlaces+"<br>");
      for (var n=0;n<documento.links.length;n++)
        {
     NuevaVentana.document.write("<strong>hash: </strong>");
     NuevaVentana.document.write(documento.links[n].hash +"<br>");
     NuevaVentana.document.write("<strong>host: </strong>");
     NuevaVentana.document.write(documento.links[n].host +"<br>");
     NuevaVentana.document.write("<strong>hostname: </strong>");
     NuevaVentana.document.write(documento.links[n].hostname +"<br>");
     NuevaVentana.document.write("<strong>href: </strong>");
     NuevaVentana.document.write(documento.links[n].href +"<br>");
     NuevaVentana.document.write("<strong>pathname: </strong>");
     NuevaVentana.document.write(documento.links[n].pathname +"<br>");
     NuevaVentana.document.write("<strong>port: </strong>");
     NuevaVentana.document.write(documento.links[n].port +"<br>");
     NuevaVentana.document.write("<strong>protocol: </strong>");
     NuevaVentana.document.write(documento.links[n].protocol +"<br>");
     NuevaVentana.document.write("<strong>search: </strong>");
     NuevaVentana.document.write(documento.links[n].search +"<br>");
     NuevaVentana.document.write("<strong>target: </strong>");
     NuevaVentana.document.write(documento.links[n].target +"<br>");
     NuevaVentana.document.write("<hr>");
        }
      }
    -->
  </script>
</head>
<body>
<h4 align="center">Documento con dos enlaces</h4>
<a href="../../index.htm">Ir a página principal</a><br>
<a href="#EnlaceLocal1">Enlace local 1</a><br>
<h3><a name="EnlaceLocal1">Enlace local 1</a><hr3>
<hr><p align="center">
<button type="button"
            onClick="MostrarCaracteristicasEnlaces(document)">
  Características enlaces
</button>
</body>
</html>
```

Ejemplo anterior (Mostrar características de los enlaces links y anchors):
CaracteristicasEnlaces.html

8.4 OBJETO image

Gracias a este objeto (disponible a partir de la versión 1.1 de JavaScript, aunque Microsoft lo adoptó en la versión 4 de su navegador) vamos a poder manipular las imágenes del documento, pudiendo conseguir efectos como el conocido rollover, o cambio de imágenes, por ejemplo, al pasar el ratón sobre la imagen.

Para hacer referencia a la siguiente imagen del documento, `<img src="MiImagen.gif" name="Imagen1">`, sirven por igual las dos siguientes expresiones:

```
document.images[0]
document.images["Imagen1"]
```

Los eventos a los que responden son tres: onError, onAbort y onLoad.

Propiedades del objeto image

Propiedad	Significado
border	Contiene el valor del atributo **border** de la imagen.
complete	Es un valor booleano que nos dice si la imagen se ha descargado completamente o no.
height	Contiene el valor del atributo **height** de la imagen.
hspace	Contiene el valor del atributo **hspace** de la imagen.
lowsrc	Contiene el valor del atributo **lowsrc** de la imagen.
name	Contiene el valor del atributo **name** de la imagen.
src	Contiene el valor del atributo **src** de la imagen.
vspace	Contiene el valor del atributo **vspace** de la imagen.
width	Contiene el valor del atributo **width** de la imagen.

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

- **Recorrido a través de una serie de imágenes**
- **Creación de un reloj digital**
- **Precarga de imágenes**
- **Cambiar imágenes gif animados**
- **Creación de efectos mouseover**

Recorrido a través de una serie de imágenes

Para demostrar lo que se puede hacer cambiando la propiedad src, vamos a echar un vistazo a un script que utiliza vínculos para recorrer, hacia adelante y hacia atrás, una serie de imágenes.

```
<html>
<head>
  <script language="javascript">
    <!--
    var array_imagenes = new Array()
    // Añadir los nombres de los archivos de las imágenes al array
    array_imagenes[0] = "Imagenes/Digital0.gif"
    array_imagenes[1] = "Imagenes/Digital1.gif"
    array_imagenes[2] = "Imagenes/Digital2.gif"
    array_imagenes[3] = "Imagenes/Digital3.gif"
    array_imagenes[4] = "Imagenes/Digital4.gif"
    array_imagenes[5] = "Imagenes/Digital5.gif"
    array_imagenes[6] = "Imagenes/Digital6.gif"
    array_imagenes[7] = "Imagenes/Digital7.gif"
    array_imagenes[8] = "Imagenes/Digital8.gif"
    array_imagenes[9] = "Imagenes/Digital9.gif"
    var indice_imagen = 0

    function siguiente_imagen()
      {
      // Incrementar el índice del array
      indice_imagen++
      // ¿Hemos llegado al último elemento?
      if (indice_imagen == array_imagenes.length)
        {
        // Si es así, inicializar el índice a 0 para volver al principio
        indice_imagen = 0
        }
      // Mostrar la nueva imagen
      document.images["imagen"].src = array_imagenes[indice_imagen]
      }

    function anterior_imagen()
      {
      // Decrementar el índice del array
      indice_imagen--
      // ¿Es negativo el índice?
      if (indice_imagen < 0)
```

```
        {
        // Si es así, inicializarlo con el valor
        // del último elemento del array
        indice_imagen = array_imagenes.length - 1
        }
      // Mostrar la nueva imagen
      document.images["imagen"].src = array_imagenes[indice_imagen]
      }
    -->
  </script>
</head>
<body>
<center>
  <img src="Imagenes/Digital0.gif" name="imagen">
  <br>
  <a href="javascript:anterior_imagen()">
    Anterior
  </a>
  <a href="javascript:siguiente_imagen()">
    Siguiente
  </a>
</center>
</body>
</html>
```

Ejemplo anterior (Hacer un recorrido a través de un ciclo de imágenes):
CicloImagenes.html

Creación de un reloj digital

Las imágenes del ejemplo anterior son ideales para crear un reloj digital. La técnica que mostramos a continuación es la base de toda animación. Mostramos una imagen, esperamos un momento, mostramos otra, volvemos a esperar y así sucesivamente. Para esperar haremos uso del ya conocido **método *setInterval()***.

```
<html>
<head>
  <script language="javascript">
    <!--
    var array_imagenes = new Array()
    // Añadir los nombres de los archivos de las imágenes al array
    array_imagenes[0] = "Imagenes/Digital0.gif"
    array_imagenes[1] = "Imagenes/Digital1.gif"
    array_imagenes[2] = "Imagenes/Digital2.gif"
    array_imagenes[3] = "Imagenes/Digital3.gif"
    array_imagenes[4] = "Imagenes/Digital4.gif"
    array_imagenes[5] = "Imagenes/Digital5.gif"
    array_imagenes[6] = "Imagenes/Digital6.gif"
    array_imagenes[7] = "Imagenes/Digital7.gif"
    array_imagenes[8] = "Imagenes/Digital8.gif"
    array_imagenes[9] = "Imagenes/Digital9.gif"
    var intervalo = setInterval("actualizar_reloj()", 1000)

   function actualizar_reloj()
```

```
      {
      // Recopilar la hora actual
      var tiempo_actual = new Date()
      // Calcular y mostrar los dígitos correspondientes a la hora
      var hora_actual = tiempo_actual.getHours()
      var hora_izquierda = Math.floor(hora_actual / 10)
      var hora_derecha = hora_actual % 10
      document.images["hora1"].src = array_imagenes[hora_izquierda]
      document.images["hora2"].src = array_imagenes[hora_derecha]
      // Calcular y mostrar los dígitos correspondientes a los minutos
      var minuto_actual = tiempo_actual.getMinutes()
      var minuto_izquierdo = Math.floor(minuto_actual / 10)
      var minuto_derecho = minuto_actual % 10
      document.images["minuto1"].src = array_imagenes[minuto_izquierdo]
      document.images["minuto2"].src = array_imagenes[minuto_derecho]
      // Calcular y mostrar los dígitos correspondientes a los segundos
      var segundo_actual = tiempo_actual.getSeconds()
      var segundo_izquierdo = Math.floor(segundo_actual / 10)
      var segundo_derecho = segundo_actual % 10
      document.images["segundo1"].src = array_imagenes[segundo_izquierdo]
      document.images["segundo2"].src = array_imagenes[segundo_derecho]
      }
    -->
  </script>
</head>
<body>
<img src="Imagenes/Digital0.gif" name="hora1"><img
                src="Imagenes/Digital0.gif" name="hora2">
<img src="Imagenes/DigitalDosPuntos.gif">
<img src="Imagenes/Digital0.gif" name="minuto1"><img
                src="Imagenes/Digital0.gif" name="minuto2">
<img src="Imagenes/DigitalDosPuntos.gif">
<img src="Imagenes/Digital0.gif" name="segundo1"><img
                src="Imagenes/Digital0.gif" name="segundo2">
<p>
<a href="javascript:clearTimeout(intervalo)">
  Detener el reloj
</a>
</body>
</html>
```

Ejemplo anterior (Diseñar un reloj digital): **RelojDigital.html**

Precarga de imágenes

Cuando se cambia la propiedad src de un objeto image, el navegador debe solicitar al servidor el nuevo archivo (asumiendo que éste no se encuentra ya en la caché del navegador) antes de poder mostrarlo. Esto no supone ningún problema para las pequeñas imágenes. Sin embargo, con archivos mayores, puede transcurrir bastante tiempo entre el momento en que el script cambia el valor de src y cuándo aparece realmente dicha imagen. El tamaño de la imagen, o la lentitud de la conexión del usuario, son factores que influirán en el proceso.

Sin embargo, puede reducir este tiempo de espera a cero precargando las imágenes. Esto significa que el navegador cargará las imágenes mientras se está recibiendo el resto de la página, aunque ello

implica que dicha imagen no podrá ser mostrada. En cambio se almacenará en la caché del navegador y permanecerá ahí hasta que el script la asigne a un objeto ***image***.

Para precargar una imagen, use el constructor de la función **image(ancho,alto)**:

var digital_uno=new image(16,21)

una vez realizada esta operación, debe asignar la dirección del nuevo archivo de imagen a la propiedad src del nuevo objeto **image**:

digital_uno.src="digital1.gif"

para mostrar una de estas imágenes precargadas, debe asignar su **src** y asignárselo a cualquier etiqueta **<image>** de la página. Por ejemplo si existe **<image src="digital_cero.gif" name="hora1">**, pondremos:

document.images["hora1"].src=digital_uno.src

Para demostrar lo que puede ofrecer la precarga de imágenes, vamos a mostrar un ejemplo que simula una animación por medio de 47 imágenes que representan un coche circulando sobre un fondo. Las imágenes han sido nombradas secuencialmente para facilitar el código de precarga.

```
<html>
<head>
  <script language="javascript">
    <!--
    var array_imagenes = new Array(47)
    // Precarga de las imágenes de la animación en un array
    for (var contador = 0; contador < array_imagenes.length; contador++)
      {
      // Construcción del objeto Image
      array_imagenes[contador] = new Image(320, 240)
      // Construcción del nombre del archivo
      // (desde car00.jpg hasta car46.jpg)
      nombre_archivo_imagen = "Imagenes/car" +
              (contador < 10 ? "0" : "") + contador + ".jpg"
      // Asignación del nombre del archivo anterior al objeto Image
      array_imagenes[contador].src = nombre_archivo_imagen
      }
    var intervalo_on = false
    var intervalo
    var numero_secuencia = 0

    function comenzar_animacion()
      {
      if (intervalo_on)
        parar_animacion()
      intervalo = setInterval("siguiente_secuencia()", 50)
      intervalo_on = true
      }

    function parar_animacion()
      {
      clearInterval(intervalo)
      intervalo_on = false
      }
```

```
      function siguiente_secuencia()
        {
        // Mostrar la siguiente imagen
        document.images["imagen"].src = array_imagenes[numero_secuencia].src
        // Incrementar el número de fotograma
        numero_secuencia++
        // Si hemos llegado al final, reiniciar el número de
        // fotograma para comenzar de nuevo
        if (numero_secuencia == array_imagenes.length)
          numero_secuencia = 0
        }
      -->
    </script>
  </head>
  <body>
  <center>
    <img src="Imagenes/car01.jpg" name="imagen">
    <p>
    <a href="javascript:comenzar_animacion()">
      Iniciar animación
    </a>
    <a href="javascript:parar_animacion()">
      Detener animación
    </a>
  </center>
  </body>
  </html>
```

Ejemplo anterior (Animación con precarga de imágenes): **Animacion.html**

Cambiar imágenes gif animados

```
        <SCRIPT language="JavaScript">
        <!--
                if (document.images) {
                        var activado = new Image();
                        activado.src="Boeing-Sikorsky-Rah-66-Comanche-Inclinado-59145.gif";
                        var desactivado= new Image();
                        desactivado.src="helicoptaire009.gif";
                }
                function activar(nombreImagen) {
                        if (document.images) {
                                document[nombreImagen].src=activado.src;
                        }
                }
                function desactivar(nombreImagen) {
                        if (document.images) {
                        document[nombreImagen].src=desactivado.src;
                        }
                }
        //-->
        </SCRIPT>
</HEAD>
<BODY>
...
```

```
<A NAME="#alli"></A>

...
<center>
<A HREF="#alli" onMouseOver="activar('prueba');"
                        onMouseOut="desactivar('prueba');">
         <IMG NAME="prueba" SRC="Boeing-Sikorsky-Rah-66-Comanche-Inclinado-59145.gif"
                                      BORDER=0></A>
</center>

...

</BODY>
```

Ejemplo anterior (Cambio imágenes animación gif): **ejercicio_image.html**

Creación de efectos mouseover

Con seguridad, el efecto especial más popular en la web es el mouseover (también conocido como rollover o imagen de sustitución). Consiste en que cuando el usuario sitúa el puntero del ratón sobre una imagen, ésta es sustituída por otra (el efecto lógico usual es recuperar la imagen original una vez que el puntero del ratón se mueva fuera de esta nueva imagen). Por ejemplo, la imagen mouseover podría ser una versión resaltada de la original, o bien una flecha que apareciera a lo largo de la misma. También puede utilizar mouseover para suministrar información adicional sobre el propósito del vínculo de la imagen, como, por ejemplo, un imagen de texto que ofreciera una descripción de dicho vínculo.

Ejemplo que hará el alumnado en los ejercicios de la unidad (Efecto rollover con precarga de imágenes)

8.5 COOKIES

Manejo de cookies

A lo largo de este tutorial hemos podido comprobar los beneficios de las variables globales, puesto que permiten que cualquier sentencia de una página acceda a su valor.

Por desgracia, cuando los usuarios navegan de una página a otra del sitio, o cuando lo abandonan y regresan más tarde a él, los valores de estas variables se pierden definitivamente.

La solución es crear una cookie. Según el navegador con el que se trabaje una cookie puede ser un archivo de texto, o una línea de un archivo de texto concreto. De cualquier forma dicho archivo se guarda en el ordenador del usuario, lo que significa que los datos de la cookie están disponibles para cualquier script del sitio que la creó. En lenguaje de Javascript, esto se conoce como guardar el estado.

La palabra cookie viene del antiguo término de programación magic cookie que se define como algo que se pasa entre rutinas o programas. Esta idea de pasar datos de una cosa a otra (en nuestro caso de una página web a otra) es en lo que se basaron los programadores para usar el nombre de cookie.

Hay que comentar que el navegador Google Chrome ignora las cookies de páginas locales, para que funcionen hay que ubicar nuestras páginas en un servidor remoto y comenzarán a funcionar. Con el resto de los navegadores no tendremos problemas. Sería interesante echarle un vistazo a las webs siguientes:

http://stackoverflow.com/questions/8105135/cannot-set-cookies-in-javascript

http://travel.michelin.com/web/faq?section=FAQ_WEB_Cookies_JS_Chrome_GBR

¿Qué es una cookie?

En términos sencillos, una cookie no es más que una parte de texto almacenado en el disco duro del usuario. El lugar de dicho almacenamiento depende del navegador y del sistema operativo:

http://galimundi.com/2012/04/04/como-localizar-borrar-gestionar-cookies-del-navegador-web/

Una cookie funciona de forma similar a una variable Javascript. Consiste en un par ***nombre=valor***. Por ejemplo si necesitamos una cookie para almacenar el nombre de usuario de una persona, podríamos usar: ***nombre_usuario=Juan***.

Otro aspecto importante que siempre se ha de tener en cuenta es que todas las cookies pertenecen a un sitio web específico; es decir, los scripts solo podrán leer aquellas que hayan sido creadas por las páginas de ese sitio web. No es posible para un sitio web crear, leer, cambiar o borrar las cookies procedentes de otro sitio.

Ventajas de las cookies

Las cookies resultan útiles, ya que permiten compartir datos entre varias páginas, sesiones del navegador sin necesidad de un servidor. A continuación tenemos una lista de las situaciones ventajosas más comunes:

- Personalizar una página, como por ejemplo guardar los colores preferidos de tu diseño
- Guardar otras preferencias como visualizar imágenes de alta o baja resolución, etc.
- Seguimiento de las visitas del usuario
- Controlar la visualización de banners para continuar con el siguiente del carrusel
- Pasar datos entre las páginas de las aplicaciones. Por ejemplo en un carrito de la compra.

Desventajas de las cookies

- Pueden ser rechazadas o desactivadas.
- Son específicas de un navegador.
- Son específicas de un ordenador.
- Son específicas de un usuario.
- Pueden ser borradas.

JavaScript puede crear cookies, leer las cookies y borrar las cookies con la propiedad ***document.cookie***

Establecer el nombre y el valor de una cookie

La cookie más básica solo contiene el nombre y el valor. Ésta es la sintaxis general: ***document.cookie="nombre_cookie=valor_cookie".***

Por ejemplo, si se desea una cookie llamada usuario con valor Pablo: ***document.cookie="usuario=Pablo".***

Si el valor contiene espacios en blanco o cualquier otro carácter especial usaremos la función ***escape(): document.cookie="nombre_libro="+escape("Edición especial Javascript").***

Recuperación del valor de una cookie

Recuperar el valor de una cookie lleva algo más de trabajo. Lo primero es almacenarla en una ***variable cadena_cookie=document.cookie***

Esto devolvería una cadena con el siguiente formato: ***nombre_cookie=valor_cookie***

Existen varias formas de extraer la parte valor_cookie. Un método consiste en localizar la posición del signo igual: ***posicion_signo_igual=cadena_cookie.indexOf("=")***

Tras esto, podemos usar el ***método substring()*** del objeto String para extraer el valor: ***valor_cookie=cadena_cookie.substring(posicion_signo_igual + 1)***

Si el valor fue codificado con ***escape() o encodeURI()*** (para HTML5), deberá decodificarlo mediante el método ***unescape() o decodeURI()*** (para HTML5) :

unescape(valor_cookie) o decodeURI(valor_cookie)

El siguiente es un sencillo ejemplo que implementa un contador de visitas por usuario:

```
<script language="javascript">
  <!--
  function LeerCookie()
    {
   //recordar que la function unescape() ha sido sustituida por decodeURI()
   //var sCookie=unescape(document.cookie);

    var sCookie=decodeURI(document.cookie);
    var n=sCookie.indexOf("=");
    var SValorCookie=sCookie.substring(n+1);
    return SValorCookie;
    }

    var nNumVisita=parseInt(LeerCookie());
    if(!nNumVisita)
      {
      nNumVisita=1;
      document.write("Bienvenido a la página de las cookies");
      }
    else
      {
      nNumVisita++;
      document.write("Esta es su visita "+nNumVisita+".");
      }
    document.cookie="Contador_visitas="+nNumVisita;
  -->
</script>
```

Ejemplo anterior (Implementar un contador de visitas con cookies):
ContadorVisitas1.html

Manipulación de varias cookies

Si en las páginas sólo es necesaria una cookie, el listado anterior servirá perfectamente. Sin embargo lo normal es que necesite dos o más. En este caso se siguen almacenando de igual manera, es decir ***document.cookie="nombre_cookie=valor_cookie"***, aunque su recuperación requiere una forma especial de hacerlo. Específicamente, debe examinar la propiedad cookie y extraer aquella con la que quiere trabajar.

Las diferentes cookies se almacenan: ***nombre1=valor1; nombre2=valor2***

Para leerlas y separarlas: ***var array_cookie = document.cookie.split("; ")***

El siguiente listado muestra una función que usa esta técnica y ofrece un ejemplo para el almacenamiento y recuperación de dos cookies.

```
<script language="javascript">
  <!--
  function leer_cookie(nombre)
    {
    var par_cookie
    var nombre_cookie
    var valor_cookie
    // Fraccionar todas las cookies y almacenarlas en un array
    var array_cookie = document.cookie.split("; ")
    // Recorrer todas las cookies
    for (contador = 0; contador < array_cookie.length; contador++)
      {
      // Dividir la cookie en su correspondiente par nombre/valor
      par_cookie = array_cookie[contador].split("=")
      nombre_cookie = par_cookie[0]
      valor_cookie = par_cookie[1]
      // Comparar el nombre con el que estamos buscando
      if (nombre_cookie == nombre)
        {
        // Si es éste, devolver el valor
        return unescape(valor_cookie)
          //recordar que la function unescape() ha sido sustituida por decodeURI()
        }
      }
    // SI la cookie no existe, devolver null
    return null
    }

  // Recuperar la cookie "cookie_usuario"
  var nombre_usuario = leer_cookie("cookie_usuario")
  // ¿Existe la cookie?
  if (!nombre_usuario)
    {
    // Si no existe, solicitar el nombre
    nombre_usuario = prompt("Por favor, introduzca su nombre:","")
    // Almacenar la cookie
    document.cookie = "cookie_usuario=" + escape(nombre_usuario)
   //recordar que la function escape() ha sido sustituida por encodeURI()
    }



// Recuperar la cookie "cookie_contador"
  var numero_visita = leer_cookie("cookie_contador")
  // ¿Existe la cookie?
  if (!numero_visita)
    {
    // Si no existe, ésta es la primera visita del usuario
    numero_visita = 1
    document.writeln("Bienvenido " + nombre_usuario
                + ". Ésta es su primera visita.")
    }
```

```
    else
      {
      // Si no existe, incrementar el número de visitas
      numero_visita++
      document.writeln("Bienvenido " + nombre_usuario +
                    ". Ésta es su visita número " + numero_visita + ".")
      }
    // Almacenar la cookie
    document.cookie = "cookie_contador=" + numero_visita
    -->
  </script>
```

Ejemplo anterior (Trabajar con varias cookies): **VariasCookies.html**

Adición de una caducidad a la cookie

Las **cookies con las que hemos trabajado hasta ahora son conocidas como cookies de sesión**, ya que su vida finaliza cuando el usuario cierra el navegador. En ese momento, dichas cookies son destruidas y comienzan desde el principio la siguiente vez que visita el sitio web.

Además, debe saber que las cookies de sesión nunca se almacenan en el disco duro, ya que el navegador las mantiene en la memoria.

Si necesita preservar datos debe usar la claúsula ***expires*** de la siguiente forma ***document.cookie = "nombre=valor; expires=cadena_GMT"***

La forma más corriente de especificar la cadena con la fecha de expiración es indicar una serie de días más desde la fecha actual:

```
var dias_expiracion = 30;
var fecha_expiracion = new Date();
var ms_transcurridos = dias_expiracion * 24 * 60 * 60 * 1000;
fecha_expiracion.setTime(fecha_expiracion.getTime() + ms_transcurridos);
var cadena_expiracion = fecha_expiracion.toGMTString();
```

Para incluir esta información en la cookie:

```
document.cookie = "cookie_contador=numero_visitas; expires=" + cadena_expiracion;
```

Puede utilizar expiraciones inferiores a un día.

Ejemplo: agregar una fecha de caducidad (en UTC o en hora GMT). De forma predeterminada, se elimina la cookie cuando el navegador se cierra:

```
document.cookie="username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 GMT";
```

La inclusión del parámetro ***expires*** no tiene ningún efecto sobre la cadena devuelta por la propiedad cookie, es decir que la fecha de expiración se almacena en la cookie pero la propiedad **document.cookie** sólo recupera los nombres y los valores de éstas separados por un punto y coma.

Especificación de la ruta de acceso

Una cookie puede ser manipulada, además de por la página que crea la cookie, por todas aquellas páginas que estén contenidas en subdirectorios dependientes del directorio donde está la página que creó dicha cookie.

Para obtener mayor control sobre los directorios que pueden acceder a las cookies, necesitaremos añadir el parámetro **path**. Por ejemplo si deseamos que el directorio de nivel superior de acceso a la cookie sea *tienda*, deberemos almacenarla de esta manera:

```
document.cookie = "nombre1=valor1; path=/tienda"
```

Si queremos que esté accesible para cualquier página del sitio web:

```
document.cookie = "nombre1=valor1; path=/"
```

Establecimiento de otros datos de la cookie

Para completar toda la sintaxis usada con las cookies, en esta sección se muestran dos parámetros más: **domain** y ***secure***.

El parámetro **domain** permite especificar qué hosts del sitio web pueden acceder a la cookie. Ésta es la manera de implementarlo:

```
document.cookie = "nombre1=valor1; domain=.mi_dominio.es"
```

El parámetro final es **secure** que acepta un valor lógico. Si éste es ***true*** fuerza a que la cookie sólo pueda ser enviada a través de una conexión que use protocolo seguro https, si es ***false*** o ***se omite***, entonces la cookie puede enviarse por cualquier protocolo inseguro http.

Aquí mostramos un ejemplo: **document.cookie = "nombre1=valor1; true"**

El siguiente ejemplo permite almacenar en una cookie el color de fondo de la aplicación y prueba la manipulación de todos los parámetros:

```
<script language="javascript">
  <!--
  function fijar_color(formulario)
    {
    // Recuperar el color seleccionado actualmente
    var lista_colores = formulario.color_name
    var color_seleccionado =
          lista_colores.options[lista_colores.selectedIndex].value
    // Cambiar el color del fondo
    opener.document["bgColor"] = color_seleccionado
    // Guardarlo en una cookie
    inic_cookie("cookie_bgcolor", color_seleccionado, 365, "/")
    }

  function inic_cookie(nombre_cookie, valor_cookie, expiracion_cookie,
                       ruta_cookie,dominio_cookie, cookie_segura)
    {
    // Inicio de la cadena que contiene los parámetros de la cookie
    var cadena_cookie = nombre_cookie + "=" + valor_cookie
    // Añadir una fecha de caducidad, si es que se ha especificado
```

```
    if (expiracion_cookie)
      {
      var fecha_expiracion = new Date()
      var ms_transcurridos = expiracion_cookie * 24 * 60 * 60 * 1000
      fecha_expiracion.setTime(fecha_expiracion.getTime() +
                               ms_transcurridos)
      var expire_string = fecha_expiracion.toGMTString()
      cadena_cookie += "; expires=" + expire_string
      }
    // Añadir la ruta, si es que se ha especificado
    if (ruta_cookie)
      cadena_cookie += "; path=" + ruta_cookie
    // Añadir el dominio, si es que se ha especificado
    if (dominio_cookie)
      cadena_cookie += "; domain=" + dominio_cookie
    // Añadir el nivel de seguridad, si es que se ha especificado
    if (cookie_segura)
      cadena_cookie += "; true"
    // Establecer la cookie
    document.cookie = cadena_cookie
    }
  -->
</script>
```

Ejemplo anterior (Guardar color fondo aplicación en cookies): **ColoresCookie.html**

Eliminación de una cookie

Eliminar una cookie es sencillo: basta con asignarle una fecha de caducidad anterior a la fecha actual:

```
inic_cookie("cookie_bgcolor","",-1)
```

En este punto hay dos temas que debemos mencionar:

- El argumento valor_cookie no importa en este caso, por lo que utilizamos una cadena vacía.
- Es importante especificar los mismos parámetros tanto a la hora de borrar, como de crear, la cookie. Si al lanzarla se incluyó el **path** o **ruta** (path=/), debemos enviar también este parámetro cuando la queramos borrar.

Si deseamos borrar todas las cookies de un sitio web, el siguiente listado contiene una función que lo hace:

```
<script language="javascript">
  <!--
  function borrar_todas_las_cookies()
    {
    var par_cookie
    var nombre_cookie
    // Almacenar todas las cookies en un array
    var array_cookie = document.cookie.split("; ")
    // Iterar a través de las cookies
    for (contador = 0; contador < array_cookie.length; contador++)
```

```
          {
          // Dividir el par nombre/valor de cada cookie
          par_cookie = array_cookie[contador].split("=")
          nombre_cookie = par_cookie[0]
          inic_cookie(nombre_cookie, "", -1)
          // inic_cookie(nombre_cookie, "", -1,"/")
          }
        }
      -->
    </script>
```

Ejemplo anterior (Borrar todas las cookies): **BorrarTodasCookies.html**

Ejemplo: ejecuta la función checkCookie () cuando se carga la página

Ejemplo_completo_cookies.html

function setCookie(cname,cvalue,exdays) función que almacena el nombre del visitante en una variable de cookie

Los parámetros de la función de arriba son el nombre de la cookie (cname), el valor de la cookie (cValue), y el número de días hasta que la cookie debería expirar (exdays).

La función establece una cookie sumando el cookiename, el valor de la cookie, y la cadena de expiración.

function getCookie(cname) creamos una función que devuelve el valor de una cookie especificada

Se le pasa la cookiename como parámetro (cname).

Cree una variable (name) con el texto a buscar (cname + "=").

Document.cookie Split el punto y coma en una matriz llamada ca (ca = document.cookie.split (';')).

Bucle a través de la matriz de ca (i = 0; i <ca.length; i + +), y leer cada valor de recorte (c = ca [i] trim ().).

Si se encuentra la cookie (c.indexOf (nombre) == 0), devuelve el valor de la cookie (c.substring (name.length, c.length).

Si no se encuentra la cookie, devuelve "".

function checkCookie() creamos la función que comprueba si se establece una cookie

Si la cookie existe/está definida se mostrará un saludo.

Si la cookie no está definida, se mostrará un cuadro de diálogo, preguntando por el nombre del usuario, y almacena la cookie del nombre de usuario por 365 días, llamando a la función setCookie

```
<html>
<head>
<script>

function setCookie(cname,cvalue,exdays)
        {
        var d = new Date();
        d.setTime(d.getTime()+(exdays*24*60*60*1000));
        var expires = "expires="+d.toGMTString();
        document.cookie = cname+"="+cvalue+"; "+expires;
        }

function getCookie(cname)
        {
```

```
        var name = cname + "=";
        var ca = document.cookie.split(';');
        for(var i=0; i<ca.length; i++)
                {
                var c = ca[i].trim();
                if (c.indexOf(name)==0) return c.substring(name.length,c.length);
                }
        return "";
        }

function checkCookie()
        {
                var user=getCookie("username");
                if (user!="")
                        {
                        alert("Welcome again " + user);
                        }
                else
                {
                        user = prompt("Please enter your name:","");
                        if (user!="" && user!=null)
                                {
                                setCookie("username",user,30);
                                }
                }
        }

</script>
</head>
<body onload="checkCookie()">
</body>
</html>
```