Desarrollo Web en Entorno Cliente.

Unidad 6

JavaScript

Modelo BOM
(Browser Object Model)
(Modelo de Objeto Navegador)

- Objeto “window”.
- Objeto “navigator”.
- Objeto “screen”.
- Objeto “location”.
- Objeto “history”.
- Trabajo con “Ventanas”.
- Objeto “opener”.
- Objeto “frames”.

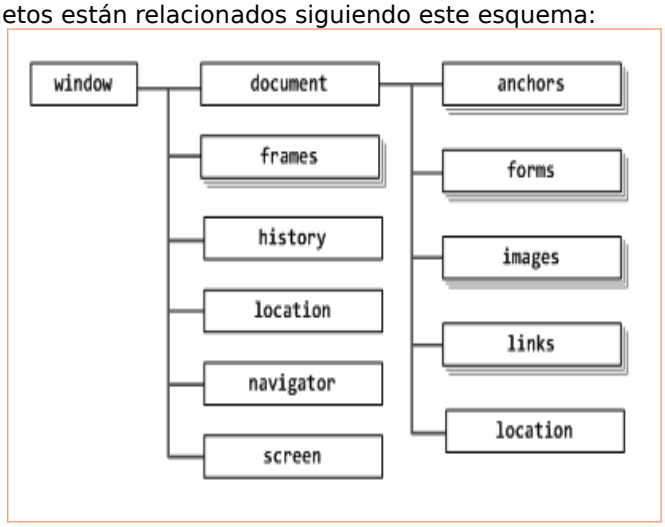
Modelo BOM.

A partir de las versiones 3.0 de Internet Explorer y Netscape se incluyó el concepto de Brower Object Model ó BOM (Modelo de Objeto Navegador), que permite acceder y modificar las propiedades de las ventanas del navegador.

Este modelo nos permite realizar diversas acciones sobre los elementos de una página HTML, ej: Redimensionar y mover la ventana del navegador, modificar el texto que se muestra en los controles HTML, cambiar el contenido de las propiedades de los objetos del modelo, etc.

El principal inconveniente de BOM es que, a diferencia de lo que sucede con DOM, no existe una estandarización del modelo entre los distintos tipos de navegadores aunque existan muchos puntos en común.

El modelo BOM es jerárquico y en la parte superior de esta jerarquía está el objeto window (ventana), el resto de los objetos están relacionados siguiendo este esquema:



Los objetos: frames, anchors, forms, images y links son arrays de objetos

Objeto window

El objeto window es el objeto principal del modelo BOM, el de Nivel Superior dentro de la jerarquía de la página HTML.

Toda la página está contenida dentro del conjunto de propiedades del objeto window.

Es un objeto que representa la ventana del navegador, por lo que cuenta con una serie de propiedades y métodos para mover y redimensionar ventanas, así como para abrir y cerrar nuevas ventanas.

Es complicado utilizar estos métodos para que funcionen igual en todos los navegadores, ya que cada uno funciona de una forma, incluso a veces los navegadores impiden algunas manipulaciones, como modificar la barra de estado o mover la pantalla.

–La notación de objetos nos permite referirnos a los subobjetos del objeto window sin necesidad de incluir el prefijo window, que se presupone: window.document.write("OK"); // es lo mismo que: document.write("OK");

Métodos del objeto window:

- **focus ()**: Permite forzar el foco de la ventana.
- **blur()**: Método inverso a focus().

  Clic y esta ventana pierde el foco
- **open()**: Abre una nueva ventana del navegador.
- **close()**: Cierra una ventana del navegador.
- **clearInterval()**: Detiene la ejecución de un proceso con temporizador definido previamente con setInterval().
- **clearTimeout()**: Suspende el temporizador iniciado con setTimeout().
- **home()**: Abre la página de inicio del visitante desde nuestro sitio web.
- **print()**: Imprime la ventana vigente, previamente abre el cuadro de diálogo de la impresora.

  Imprimir
- **scrollTo()**: Método para desplazar la posición horizontal y vertical (valores absolutos).
- **scrollBy()**: Método para desplazar la posición horizontal y vertical (valores relativos a la posición actual).
- **moveBy(x,y)**: Desplaza la ventana x píxeles a la derecha y y píxeles hacia abajo. Si los números son negativos se desplaza hacia la izquierda y hacia arriba.
- **moveTo(x,y)**: Desplaza la ventana del navegador a la posición x,y de la pantalla.
- **resizeBy(x,y)**: Redimensiona la ventana del navegador sumando a la anchura anterior el valor x y a la altura anterior el valor y. Si los números son negativos se reducen la anchura y la altura.
- **resizeTo(x,y)**: Redimensiona la ventana del navegador a una anchura x y altura y.
- **setInterval("función", milisegundos)**: Inicia un proceso que se repite a intervalos regulares definidos en milisegundos.
- **setTimeout("función", milisegundos)**: Inicia un proceso que se ejecuta una única vez después de un período definido en milisegundos. Para que se repita el ciclo hay que volver a utilizar el método.
- **stop()**: Detiene la carga de la página.

Métodos Auxiliares para Cuadros de Diálogo:

- **window.alert():** Permite generar un cuadro de mensaje sin ningún valor de retorno.

  Alert("Aviso de alerta con alert()");
- **window.cofirm():** Permite generar un cuadro de diálogo con un icono de pregunta y con dos botones que dan la opción de aceptar y cancelar. Si devuelve True es porque el usuario ha aceptado el cuadro.
- **window.prompt():** Presenta un cuadro de diálogo de entrada de datos. El método devuelve el valor introducido o null si se pulsó el botón Cancelar ó si se cerró el cuadro de diálogo.

bjeto navigator

El objeto window contiene varios objetos, entre ellos el objeto navigator.
El objeto navigator nos proporciona información sobre el navegador actualmente en uso.
Las propiedades del objeto navigator son de sólo lectura, y contienen la información:

Propiedades

- **appCodeName**. Cadena que contiene el nombre del código de la aplicación cliente.
  Tanto IE como Firefox/Netscape le asignan el mismo valor, por lo que no sirve para identificarlos.
- **appName**. Cadena que contiene el nombre del Navegador cliente.
- **appVersion**: Cadena con la versión del navegador cliente.
- **appMinorVersion**: Versión menor del navegador.

- **language**: Cadena de dos caracteres que contiene información sobre el idioma de la versión del cliente.
- **mimeTypes**: Array que contiene todos los tipos MIME, es decir todos los tipos de archivos soportados por el navegador cliente. IE devuelve un elemento vacío. Al ser una matriz de valores se puede obtener la lista recorriendo sus elementos hasta window.navigator.mimeTypes.length.
- **platform**: Cadena con la plataforma sobre la que se está ejecutando el programa cliente.
- **cpuClass**: Tipo del procesador.
- **plugins**: Array que contiene todos los plug-ins soportados por el cliente. Lista de complementos instalados. IE devuelve una lista vacía. Al ser una matriz de valores se puede obtener la lista recorriendo sus elementos hasta window.navigator.plugins.length.

- **userProfile:** Perfil del usuario.
- **userAgent:** Cadena que contiene la cabecera completa del agente enviada en una petición HTTP. Contiene la información de las propiedades appCodeName y appVersion.
- **userLanguage**: En IE, idioma del navegador. En otros navegadores, la propiedad es language.
- **systemLanguage**: En IE, idioma del sistema. En otros navegadores se utiliza la propiedad language.
- **onLine**: Valor booleano que indica el estado de conexión del navegador.
- **cookieEnabled**: Valor booleano que indica si se aceptan ó no archivos cookie.

Nota: Los distintos navegadores no se comportan de modo coherente cuando llega el momento de asignar valores a cada una de las propiedades. Así, alguna propiedad tenga el valor vacío en algún caso y un valor concreto en otro.
Las propiedades con comportamiento homogéneo en los diferentes navegadores son:
appCodeName, appName, appVersion, userAgent, language, platform.

Métodos

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

Ejemplo:

```
<!-- Ejemplo del objeto navigator -->
<HTML>
   <HEAD>
        <title>Ejemplo de JavaScript</title>
   </HEAD>
   <BODY>
        <script LANGUAGE="JavaScript">
        <!--
        document.write("Navigator <b>appCodeName</b>: " + navigator.appCodeName + "<br>");
        document.write("Navigator <b>appName</b>: " + navigator.appName + "<br>");
        document.write("Navigator <b>appVersion</b>: " + navigator.appVersion + "<br>");
        document.write("Navigator <b>language</b>: " + navigator.language + "<br>");
        document.write("Navigator <b>platform</b>: " + navigator.platform + "<br>");
        document.write("Navigator <b>userAgent</b>: " + navigator.userAgent + "<br>");
        //-->
        </script>
   </BODY>
</HTML>
```

Objeto screen

El objeto screen es otro subobjeto del objeto window.
El objeto screen contiene la información relacionada con la capacidad física de la pantalla (resolución y color), es decir, información acerca del monitor donde se está ejecutando el navegador.

Nota: El objeto **screen** es una propiedad del objeto window, por lo que haber dispuesto la sintaxis: window.screen.width etc. es la forma más completa, pero más larga de escribir (recordar que el objeto window es el principal y lo podemos obviar cuando accedemos a sus propiedades o métodos).

Propiedades

- screen.**width**. Ancho total de la pantalla en píxeles.
- screen.**height**. Altura total de la pantalla en píxeles.
- screen.**availWidth**: Ancho disponible en píxeles para el navegador. No se toman en cuenta las barras.
- screen.**availHeight**: Altura disponible en píxeles para el navegador. No se toman en cuenta las barras.
- screen.**colorDepth**: Profundidad de color. Número de bits usados para representar los colores.
- **Cantidad de colores**: Se calcula de manera indirecta, es el resultado de 2 elevado al valor de la profundidad de color. **Math.pow(2, screen.colorDepth)**

Ejemplo:
Mostrar el valor de las cinco propiedades que tiene el objeto screen:

```
<html>
      <head>
            <title>Propiedades de screen.</title>
      </head>
      <body>
        <script language="javascript">
          // propiedades del objeto screen
          document.write('Valores de las propiedades del objeto screen:<br>');
          document.write('Ancho Total de la Pantalla:' + screen.width + 'píxeles'+'<br>');
          document.write('Altura Total de Pantalla:' + screen.height + 'píxles' + '<br>');

          document.write("Ancho Disponible de Pantalla:", screen.availWidth,"píxeles<br />");
          document.write("Alto Disponible de Pantalla:", screen.availHeight,"píxeles<br />");

          document.write('Profundidad de Color:' + screen.colorDepth + 'bits' + '<br>');
          document.write('Cantidad de colores:' + Math.pow(2,screen.colorDepth) + '<br>');
        </script>
      </body>
</html>
```

Objeto location

El objeto location es otro subobjeto del objeto window.
El objeto location contiene la información relacionada con la dirección de la página vigente (la que se está visualizando en el navegador).

Propiedades

- location.**hostname:** Nombre del Dominio.
- location.**href:** Dirección completa de la página.
- location.**hash**: Referencia que recibió el clic en la página.
- location.**pathname**: Ruta de acceso a la página, sin el nombre del dominio.
- location.**port**: Número de Puerto.
- location.**protocol**: Tipo de Protocolo de la conexión.
- location.**search**: Cadena compuesta por los parámetros de la página.

Métodos

- location.**reload():** Recargar la página vigente. Equivale a usar la tecla "Actualizar" del Navegador.

```
<a href="javascript:window.location.reload()"> Recargar esta misma página.</a>
```

- location.**replace()**: Carga otra página en el parámetro del método.

```
window.location.replace("xxxx.html")
```

Ejemplo:

Extraer todas las propiedades del objeto location:

```
<html>
        <head>
                <title>Propiedades de location.</title>
        </head>
        <body>
            <script type="text/javascript">
              // propiedades del objeto location
              document.write("Propiedades del objeto location <br /><br />");
              for (x in window.location){
                        document.write(x + " : " + window.location[x] + "<br />");
              }
            </script>
        </body>
</html>
```

Aclaración:

Propiedad **location.href**
Nos permite ordenar un redireccionamiento a otra página.

```
// Dentro de una página x hacemos un redireccionamiento a otra página distinta.
        Window.location.href = "http://www.iescamas.es";
```

Propiedad **location.search**
Esta propiedad contiene la cadena de caracteres de los parámetros de la página, es decir, lo que aparece después de la dirección de la página, a partir del símbolo ?.
Después del símbolo ? se incluyen los parámetros en la forma nombre=valor y cuando hay más de un parámetro se utiliza el ámpersan como separador.

Ejemplo:
www.misitio.com**?parm1=30&parm2=Casa&parm3=8**

En este caso, la propiedad **search** contiene los datos pasados a continuación de la dirección url:
?parm1=30&parm2=Casa&parm3=8

Ejercicio:

Puede, que nos interese trabajar con los parámetros individuales y no con la cadena completa, para ello, desarrollamos una función que nos devuelve una matriz asociativa con los parámetros.
Es decir, nuestro objetivo es obtener una matriz con este contenido:

mat.length=3
mat[parm1]=20
mat[parm2]=Casa
mat[parm3]=3

- Para ello,
1º. Generamos una matriz con los grupos clave=valor
2º. Separamos la clave del valor para crear una matriz asociativa en la que la clave es el nombre del parámetro, y su valor es el valor del parámetro.

Al ejecutar el resultado se obtiene el resultado:

Cadena de parámetros ?parm1=30&parm2=Casa&parm3=8
Parámetro : clave es parm1 y valor es 30
Parámetro : clave es parm2 y valor es Casa
Parámetro : clave es parm3 y valor es 8

Solución:

```
<!DOCTYPE HTML>
<html>
  <head>
    <script type="text/javascript">
        function obtenerParam(str1) {
            // separador por grupo clave=valor y le quitamos el primer carácter símbolo ?
                  var sep1 = new RegExp("&", "g");
                  var str2 = str1.substring(1);
             // obtenemos una matriz en la que cada elemento es clave=valor
                  var parms = str2.split(sep1);
             // creamos la matriz que almacenará el resultado
                  var mat = new Array();
             // el separador dentro de cada elemento del grupo clave=valor es el símbolo =
                  var sep2 = new RegExp("=", "g");
             // procesamos grupo clave=valor y separamos clave de valor
                  for (var i = 0; i < parms.length; i++) {
                           var parm = parms[i].split(sep2);
                      // creamos los elementos de la matriz asociativa mat[nombre de clave]=valor
                            mat[parm[0]]= parm[1];
                   }
               // se devuelve la matriz resultante
               return mat;
        }

        var strSearch = "?parm1=30&parm2=Casa&parm3=8";
        document.write("Cadena de parámetros ", strSearch, "<br />");

               // se crea una matriz para recibir los elementos resultantes de la función
        var mat = new Array();
        mat = obtenerParam(strSearch);

               // se imprime la matriz resultado
        for (ind in mat) {
               document.write("Parámetro : clave es ", ind, " y valor es ", mat[ind], "<br />");
        }
    </script>
  </head>

  <body>

  </body>


</html>
```

Objeto history

El objeto history es otro subobjeto del objeto window.
El objeto history permite acceder a la historia de las páginas visitadas.

Propiedades

- history.**current**. Cadena que contiene la URL completa de la entrada actual en el historial.
- history.**next**. Cadena que contiene la URL completa de la siguiente entrada en el historial.
- history.**previous**. Cadena que contiene la URL completa de la anterior entrada en el historial.
- history.**length**. Entero que contiene el número de entradas del historial (cuantas páginas hay registradas en la historia).

Métodos

- history.**back():** Permite acceder a la página anterior de la historia de páginas visitadas. Equivale a la tecla Atrás.

```
<a href="javascript:window.history.back()">Página anterior</a>
```

- history.**forward()**: Permite acceder a la página siguiente de la historia de páginas. Equivale a la tecla Adelante.
- history.**go()**: Permite posicionarnos en una página de la historia de páginas visitadas.

go(-2) Retrocede dos páginas

```
<input type=button  value=Atrás  onclick="history.go(-1)">
```

go(posicion)

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.
- Una cadena de caracteres, en cuyo caso representa toda o parte de una URL que esté en el historial.

Trabajo con Ventanas.

1. Cambiar el Tamaño de la ventana.

Existen 2 métodos para modificar el tamaño de las ventanas.

– **resizeTo()**: Cambia el tamaño de la ventana, ancho y alto. (valores en píxeles)

```
resizeTo(pAncho, pAlto);
```

– **resizeBy()**: Modifica los valores actuales del tamaño de la pantalla.

- Si los parámetros son negativos, se reduce el tamaño.
- Si en algún parámetro el valor es 0, no se modifica esa dimensión de la ventana.

```
resizeBy(pAncho, pAlto);
```

Nota: Si queremos modificar el tamaño de la ventana para que ocupe toda la pantalla se debe obtener el tamaño máximo de la pantalla a partir del objeto screen y después utilizar esos valores como parámetros del método **resizeTo()**.

```
resizeTo(screen.availWidth, screen.availHeight);
```

2. Cambiar la posición de la ventana.

Podemos mover la ventana de forma absoluta ó relativa a la posición actual.
Los valores de los parámetros van en píxeles. El valor 0,0 es la esquina superior izquierda de la ventana.

- Movimiento Absoluto: `moveTo(pX,pY);` pX y pY son las coordenadas en el eje x e y.
- Movimiento Relativo: `moveBy(desplazX, desplazY);`

desplazX y desplazY son los píxeles que se desplazarán respecto a la posición vigente.
Estos valores pueden ser positivos, ceros ó negativos que se suman al valor actual.

3. Impresión de la Pantalla.

El método print() del objeto window abre un cuadro de diálogo de impresión para seleccionar la impresora, para ajustar la configuración y la cantidad de páginas y copias.

```
<a href="javascript:window.print()"> Impresión de la página </a>
```

4. Ventanas Emergentes (popup).

Son ventanas de tamaño reducido que se abre a partir de alguna acción en la página.
El método open() del objeto window nos permite abrir una ventana popup:

var winpopup = window.open(popUrl, popNombre, popOpción);

popUrl: Es la dirección Url de la ventana popup.

popNombre: Es el nombre de la ventana.

popOpción: Es una cadena de caracteres que contiene una lista de opciones de apertura de la ventana.

Las opciones se codifican siguiendo el formato:

opcion1=valor1, **opcion2=valor2** La coma actúa como separador entre una opción y otra.

Opciones de Apertura:

- **directories**: Valor yes/no. Añade o no una barra de links.
- **menubar**: Valor yes/no. Añade o no una barra de menús.
- **status**: Valor yes/no. Añade o no una barra de estado.
- **location**: Valor yes/no. Añade o no una barra de direcciones.
- **scrollbars**: Valor yes/no/auto. Añade o no las barras de desplazamientos.
- **resizable**: Valor yes/no. Permite o no el redireccionamiento.
- **height**: Altura en píxeles.
- **width**: Ancho en píxeles.
- **left**: Posicionamiento del margen izquierdo.
- **top**: Posicionamiento del margen superior.
- **fullscreen**: Valor yes/no. Pantalla completa o no, solo en IE.

Ejercicio: Ventanas Emergentes "popup".

Realiza una página que abre una ventana popup en el centro de la pantalla y también se verifica si existe bloqueo de popups (algo que ocurre habitualmente según la configuración del navegador.

Solución:

```
<!DOCTYPE HTML>
<!-- Ejercicio Ventanas Emergentes "popup" -->
<html>
     <head>
       <script type="text/javascript">
          function miPopup(txtPag, iAncho, iAltura, txtOpt) {
             var iTop=(screen.height - iAltura) / 2;
             var iLeft=(screen.width - iAncho) / 2;
             var miPopup=window.open(txtPag,"","top="+iTop+",left="+iLeft+",width="+iAncho+",
                                      height="+iAltura+"," + txtOpt);
              if (miPopup) {
                        miPopup.focus();
                    } else {
                        alert("Popup bloqueado")
                    }
              }
        </script>
     </head>
     <body>
        <a href="javascript:miPopup('popup.html', 200, 100, 'resizable=no, location=no, menubar=no,
          status=no, scrollbars=no, menubar=no')">Abrir ventana popup</a>
     </body>
</html>
```

Ejercicio: Comunicación entre Ventanas.

Cuando se utiliza el método open() para abrir una ventana se obtiene, como retorno de la función, un objeto de tipo window, esta variable nos permite posteriormente la comunicación entre las ventanas (por ejemplo, con la ventana popup abierta).

→ Realiza una página donde se permita la comunicación entre dos ventanas, desde la ventana principal se abre la ventana popup. (Lo que interesa aquí, está dentro de la codificación de la ventanta popup (pop2.html)).

Solución:

```
<!DOCTYPE HTML>
<!-- Comunicación entre ventanas -->
<html>
    <head>
       <script type="text/javascript">
          var winPopup;
          function cerrarPopup() {
              if (winPopup) {
                 if (!winPopup.closed) {
                      winPopup.close();
                 }
              }
          }
          function miPopup(txtPag, iAncho, iAltura, txtOpt) {
               var iTop=(screen.height - iAltura) / 2;
               var iLeft=(screen.width - iAncho) / 2;
               winPopup = window.open(txtPag,"","top="+iTop+", left="+iLeft+
                  ", width="+iAncho+", height="+iAltura+","+txtOpt);
               if (winPopup) {
                        winPopup.focus();
               } else {
                        alert("Popup bloqueado")
               }
          }
       </script>
    </head>
    <body>
       <form name="form1" action="" method="post">
          País: <input type="text" name="campo1" value=""><br />
          <a href="javascript:miPopup('pop2.html', 200, 200, 'resizable=no,location=no,menubar=no,
                                      status=no, scrollbars=no, menubar=no')">Cambiar país</a><BR />
       </form>
       <br /><br />
       <a href="javascript:cerrarPopup()">Cerrar ventana popup</a>
    </body>
</html>
```

Objeto opener

El objeto opener es otro subobjeto del objeto window.
El objeto opener nos da acceso a la ventana padre (la ventana desde la que se abrió la ventana vigente).
Al acceder a ese objeto podemos trabajar sobre el objeto document, que nos permite acceder a toda la página.
→ Ahora vemos el código de la ventana popup (pop2.html).

```
<html>
      <head>
          <title>(popup auxiliar) Selección de un valor para el formulario principal</title>
          <script type="text/javascript">
             function asignarPaís(txtCode) {
                 if (window.opener) {
                        if (window.opener.document.form1){
                               window.opener.document.form1.campo1.value=txtCode;
                        }
                         return;
                 }
                 alert("La ventana principal ya no existe");
              }
          </script>
      </head>
      <body>
             <ul>
               <center>Elija un país: </center>
               <li><a href="javascript:asignarPaís('España')">España</a></li>
               <li><a href="javascript:asignarPaís('Francia)">Francia</a></li>
               <li><a href="javascript:asignarPaís('Portugal')">Portugal</a></li>
               <li><a href="javascript:asignarPaís('Italia')">Italia</a></li>
               <li><a href="javascript:asignarPaís('UK')">UK</a></li>
             </ul><br>
             <a href="javascript:window.close()">Cerrar popup</a>
      </body>
</html>
```

Cada referencia ejecuta la función asignarPaís() con el parámetro correspondiente al país seleccionado. Si existe el objeto window.opener (la ventana desde donde se abrió la ventana popup) y si existe el formulario que se quiere actualizar, se asigna un texto a uno de los campos de entrada.
De esta manera al seleccionar un país, el dato elegido se traslada al formulario de la ventana principal.
La ventana popup o emergente se cierra utilizando el método close() sobre el objeto window, del mismo modo que se cierra cualquier tipo de ventana.

Precaución al usar ventanas "popups".

Las ventanas emergentes tienen muchos usos, y se utilizan frecuentemente para incluir anuncios publicitarios. Y a los usuarios les suele molestar bastante, por ello, en los navegadores existe la opción de bloqueo de ventanas emergentes.
Si el usuario lo desea podrá ver ese popup, se pueden bloquear los que aparecen de forma automática, y también se pueden bloquear totalmente.
Esta característica de posibles bloqueos hace que tengamos que evitar usar las ventanas emergentes, para las partes importantes de nuestro sitio Web.
Las ventanas emergentes sólo las utilizaremos en funciones de menor importancia, de forma que su bloqueo potencial no haga que nuestro sitio sea inaccesible para ciertos usuarios.

Objeto frames

El objeto frames es otro subobjeto del objeto window.
Los frames permiten la subdivisión de la página.
Las etiquetas html que permiten crear frames son <frameset> y <frame>.
Desde javascript la comunicación entre los distintos frames es posible ya que dependen del objeto window.

En html, el código para crear 3 frames, 2 dentro de un frameset y otro dentro de otro frameset:

```
<frameset>
    <frameset>
        <frame name="frame11" src="frame11.html">
        <frame name="frame12" src="frame12.html">
    </frameset>
    <frame name="frame21" src="frame21.html">
</frameset>
```

Para cargar una página dentro de uno de los frames se usa:

```
<a href="xxxx.html" target="frame12">Cambiar frame</a>
```

El atributo target permite direccionar la carga de la página (subpágina) al frame deseado, en el frame12.

Los valores de este atributo pueden ser un nombre de un frame, u otros valores:

- **_parent**: Abre la página dentro del área definida por la etiqueta frameset.
- **_top**: Abre la página al nivel principal reemplazando toda la definición de frames de la página vigente.
- **_blank**: Abre una página nueva.

En JavaScript, se puede direccionar a un frame utilizando los objetos **parent** ó **top**.

```
parent.frames["frame21"].window.location.href="xxxx.html"
                    ó
top.frames["frame21"].window.location.href="xxxx.html"
```

Esto, nos permite cambiar un frame por otro.

- El objeto **parent**: Representa a la ventana definida por la etiqueta <frameset> de nivel superior y la matriz de frames contiene la lista de frames contenidos en ese frameset.
  La propiedad window.location.href permite definir la página que se incluirá en el frame referenciado, en este caso, frame21.
- El objeto **top**: Representa la ventana de nivel más alto, por lo que podemos utilizar este objeto para cargar una página en el frame21 desde frame11; frame11 pertenece a otro frameset por lo que para acceder a frame21 hay que utilizar el camino del objeto top (por el objeto parent no lo encontraríamos).
- Toda la información de un frame es accesible desde cualquier otro frame, por ejemplo, la definición de una función.

Frames y Seguridad.

La comunicación entre frames tiene limitaciones por razones de seguridad, para proteger la confidencialidad y la integridad de los datos.
Dos frames pueden comunicarse si pertenecen a un mismo dominio.
Es el navegador el que prohibe el acceso a los datos de otro frame de dominio diferente, la limitación es tanto en lectura como en escritura.
Esto hace que sea imposible obtener información de sitios ajenos, dado que en el intento aparecerá un error de JavaScript.

iframe.

En HTML se permite el uso de frames definidos mediante la etiqueta <iframe>.
Este tipo de frames se utiliza para definir áreas de anuncios, la ventaja respecto a los frames, que hemos visto anteriormente es que la carga de nuestra página no se retrasará por el contenido de la etiqueta iframe.
Si la página referenciada en la etiqueta iframe se encuentra en un sitio web que está fuera de servicio o que tiene un tiempo de respuesta muy lento, eso no afectará al resto del contenido de los frames.

```
<iframe src="http://www.mipagina.net/anuncio.html"> ...<iframe>
```

En JavaScript, la alternativa a incluir iframe sería:

```
<script type="text/javascript" src="http://www.mipagina.net/anuncio.js"></script>
```

El archivo anuncio.js contiene las instrucciones JavaScript para el anuncio publicitario, el problema de este enfoque es que un problema al cargar este archivo, afectará al resto de la página.