

# Taller de Javascript



Manual para imprimir

## Autores del manual

Este manual ha sido realizado por los siguientes colaboradores de DesarrolloWeb.com:

**Miguel Angel Alvarez**

Director de DesarrolloWeb.com  
<http://www.desarrolloweb.com>  
(60 capítulos)

**Fabio Núñez**

Webmaster de Dreamweaving  
<http://www.dreamweaving.com>  
(1 capítulo)

**David**

(1 capítulo)

**Manuel Estévez Simonet**

(1 capítulo)

**Carlos Luis Cuenca**

<http://www.helloworldsolutions.com/>  
(6 capítulos)

**Richar Mendoza**

Webmaster de AyacuchoEnLinea,  
Universidad San Cristóbal  
<http://www.ayacuchoenlinea.com>  
(1 capítulo)

**José Alberto Torres Arroyo**

Ldo en Publicidad y RR.PP  
(1 capítulo)

**Juan Carlos Aleman Paez**

(1 capítulo)

**Ignacio Rodriguez**

(1 capítulo)

**Juan Carlos Gámez**

Espacio Latino  
<http://www.espaciolatino.com>  
(1 capítulo)

**Bruno Suárez Laffargue**

Analista/Programador  
<http://www.helloworldsolutions.com/>  
(1 capítulo)

**Arturo Ramo García**

<http://www.aplicaciones.info/>

**César Pietri**  
<http://www.tips-web.com>  
(1 capítulo)

**Alex Sancho**  
<http://alexsancho.name/>  
(1 capítulo)

**José Antonio Jiménez  
Garelli**  
(2 capítulos)

**Ismael Zori**  
<http://www.telefonica.net/web2/izori/>  
(1 capítulo)

(1 capítulo)  
**Javier Bernal Lérica**  
(1 capítulo)

**Gema Maria Molina Prados**  
Equipo DesarrolloWeb.com  
<http://www.desarrolloweb.com/>  
(1 capítulo)

## Efectos rápidos con Javascript

Antes de meternos en materia podemos ver una serie de efectos rápidos que se pueden programar con Javascript. Esto nos puede hacer una idea más clara de las capacidades y potencia del lenguaje que nos vendrán bien para tener una idea más exacta de lo que es Javascript a la hora de recorrer los siguientes capítulos.

### Abrir una ventana secundaria

Primero vamos a ver que con una línea de Javascript podemos hacer cosas bastante atractivas. Por ejemplo podemos ver cómo abrir una ventana secundaria sin barras de menús que muestre el buscador Google. El código sería el siguiente.

```
<script>
window.open("http://www.google.com","", "width=550,height=420,menubar=no")
</script>
```

Podemos [ver el ejemplo en marcha aquí](#).

### Un mensaje de bienvenida

Podemos mostrar una caja de texto emergente al terminarse de cargar la portada de nuestro sitio web, que podría dar la bienvenida a los visitantes.

```
<script>
window.alert("Bienvenido a mi sitio web. Gracias...")
</script>
```

Puedes [ver el ejemplo en una página a parte](#).

### Fecha actual

Veamos ahora un sencillo script para mostrar la fecha de hoy. A veces es muy interesante mostrarla en las webs para dar un efecto de que la página está al "al día", es decir, está actualizada.

```
<script> document.write(new Date()) </script>
```

Estas líneas deberían introducirse dentro del cuerpo de la página en el lugar donde queramos que aparezca la fecha de última actualización. Podemos [ver el ejemplo en marcha aquí](#).

**Nota:** Un detalle a destacar es que la fecha aparece en un formato un poco raro, indicando también la hora y otros atributos de la misma, pero ya aprenderemos a obtener exactamente lo que deseemos en el

formato correcto.

## Botón de volver

Otro ejemplo rápido se puede ver a continuación. Se trata de un botón para volver hacia atrás, como el que tenemos en la barra de herramientas del navegador. Ahora veremos una línea de código que mezcla HTML y Javascript para crear este botón que muestra la página anterior en el historial, si es que la hubiera.

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

El botón sería parecido al siguiente. Podemos pulsarlo para ver su funcionamiento (debería llevarnos a la página anterior).

Atrás

Como diferencia con los ejemplos anteriores, hay que destacar que en este caso la instrucción Javascript se encuentra dentro de un atributo de HTML, onclick, que indica que esa instrucción se tiene que ejecutar como respuesta a la pulsación del botón.

Se ha podido comprobar la facilidad con la que se pueden realizar algunas acciones interesantes, existirían muchas otras muestras que nos reservamos para capítulos posteriores.

*Artículo por **Miguel Angel Alvarez***

## Apertura y configuración de popups con javascript

En determinadas ocasiones es muy útil abrir un enlace en una ventana secundaria, es decir, una ventana aparte que se abre para mostrar una información específica. Algunas ventajas de abrir un enlace en una ventana secundaria pueden ser que:

- El usuario no se marcha de la página donde estaba el enlace.
- La ventana secundaria se puede configurar libremente con lo que se pueden hacer ventanas más grandes o pequeñas y con más o menos menús.
- En general, el grado de control de la ventana secundaria utilizando Javascript aumenta.

### [Pincha aquí para ver lo que es una ventana secundaria](#)

Para abrir una ventana secundaria podemos hacerlo de dos maneras, con HTML y con Javascript. Veamos cada una de ellas

#### Abrir una ventana con HTML

Se puede conseguir abrir una ventana secundaria muy fácilmente con tan solo HTML. Para ello podemos utilizar el atributo TARGET de las etiquetas HREF. Si ponemos target="\_blank" en el enlace, la página se abrirá en una ventana secundaria. También podemos poner target="xxx" para que el enlace se presente en la ventana llamada xxx o en el frame xxx.

El enlace tendría que tener esta forma:

```
<a href="mipagina.html" target="_blank">
```

El problema de abrir una página secundaria con HTML consiste en que no podemos definir la forma de ésta ni podremos ejercer mayor control sobre ella, tal como comentábamos entre las ventajas de abrir una ventana secundaria con Javascript. La ventana que se abre siempre será como el usuario tenga definido por defecto en su navegador.

## Abrir una ventana con Javascript

Para abrir una ventana con Javascript podemos utilizar la sentencia `window.open()`. No pasa nada por que no conozcas Javascript, puesto que es muy sencillo utilizarlo para este caso. Vamos a ver paso a paso cómo abrir una ventana secundaria utilizando Javascript.

### 1. Sentencia Javascript para abrir una ventana

La sentencia es simplemente la función `window.open()`, lo más complicado es saber cómo utilizar esa función, pero ahora veremos que no reviste ninguna complicación.

La función `window.open()` recibe tres parámetros, que se colocan dentro de los paréntesis, de este modo:

**`window.open(URL,nombre_de_la_ventana,forma_de_la_ventana)`**

Veamos rápidamente cada uno de estos parámetros por separado.

**URL:** representa el URL que deseamos abrir en la ventana secundaria, por ejemplo `http://www.desarrolloweb.com`

**nombre\_de\_la\_ventana:** es el nombre que se le asigna a esta ventana para dirigir enlaces con el atributo `target` del HTML

**forma\_de\_la\_ventana:** se indica el aspecto que va a tener la ventana secundaria. Por ejemplo se puede definir su altura, anchura, si tiene barras de desplazamiento, etc

Veamos un ejemplo de sentencia Javascript completa para abrir una ventana secundaria:

```
window.open("http://www.desarrolloweb.com" , "ventana1" , "width=120,height=300,scrollbars=NO")
```

Esto quiere decir que abra la página inicial de desarrolloweb.com en una ventana secundaria a la que vamos a llamar ventana1. Además, la ventana será de 120 pixels de ancho, 300 de alto y no tendrá barras de desplazamiento.

Una aclaración adicional, si después de abrir esa ventana colocamos otro enlace en la página que abría la ventana cuyo atributo `target` está dirigido hacia el `nombre_de_la_ventana` (en este caso `ventana1`), este enlace se mostrará en la ventana secundaria.

### 2. Función que abre una ventana

Lo más cómodo para abrir una ventana es colocar una función Javascript que se encargue de las tareas de abrirla y que reciba por parámetro la URL que se desea abrir.

El script es sencillo, veámoslo a continuación:

```
<script language=javascript>
function ventanaSecundaria (URL){
    window.open(URL,"ventana1","width=120,height=300,scrollbars=NO")
}
```

```
}  
</script>
```

### 3. Colocamos un enlace

Este enlace no debe estar dirigido directamente a la página que queramos abrir, sino a la sentencia Javascript necesaria para abrir la ventana secundaria. Para ejecutar una sentencia Javascript con la pulsación de un enlace lo hacemos así:

```
<a href="javascript:sentencia_javascript_para_abrir_la_ventana">
```

### 4. El enlace llama a la función que abre la ventana

Ahora Veamos cómo quedaría todo ese enlace en la página.

```
<a href="javascript:ventanaSecundaria('http://www.desarrolloweb.com')"> Pincha en este enlace para abrir la  
ventana secundaria</a>
```

Que da como resultado:

[Pincha en este enlace para abrir la ventana secundaria](#)

(En la página que vayamos a colocar este enlace deberíamos tener el script que hemos hecho anteriormente que contenía la función para abrir la ventana.)

Hay que fijarse que las comillas simples que se colocan para definir el URL que se pasa como parámetro de la función `ventanaSecundaria()`. Son comillas simples porque el href del enlace ya tiene unas comillas dobles, y dentro de comillas dobles siempre se han de utilizar comillas simples a no ser que deseemos cerrar las comillas dobles.

### Parámetros para dar forma a una ventana

Estos atributos los puedes utilizar en la función `window.open()` para definir la forma que desees que tenga tu ventana secundaria.

<b>Width</b>	Ajusta el ancho de la ventana. En pixels
<b>Height</b>	Ajusta el alto de la ventana
<b>Top</b>	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.
<b>Left</b>	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.
<b>Scrollbars</b>	Para definir de forma exacta si salen o no las barras de desplazamiento. <code>scrollbars=NO</code> hace que nunca salgan. <code>Scrollbars=YES</code> hace que salgan (siempre en ie y solo si son necesarias en NTS).
<b>Resizable</b>	Establece si se puede o no modificar el tamaño de la ventana. Con <code>resizable=YES</code> se puede modificar el tamaño y con <code>resizable=NO</code> se consigue un tamaño fijo.
<b>Directories</b> (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

**Location** de estado.

(barra

direcciones)

**Menubar**

(barra de menús)

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.

**Status**

(barra de estado)

**Titlebar**

(la barra del título)

**Toolbar**

(barra de herramientas)

## Abrir una ventana sin un enlace

En otras ocasiones desearemos abrir una ventana secundaria automáticamente, es decir, sin necesidad de que el usuario pulse sobre ningún enlace. En este caso, el código de la función ventanaSecundaria nos sirve también y habrá que añadir una línea de código Javascript a continuación de la función ventanaSecundaria. Esta línea a añadir simplemente será una llamada a la función que se ejecutará según se está cargando la página. Veamos como quedaría este código:

```
<script language=javascript>
function ventanaSecundaria (URL){
    window.open(URL,"ventana1","width=120,height=300,scrollbars=NO")
}
```

```
ventanaSecundaria("http://www.desarrolloweb.com");
</script>
```

Queda en negrita lo que sería la llamada a la función que abre la ventana secundaria, como está fuera de una función se ejecuta según se está cargando la página.

**Referencias:** Hemos publicado un manual en DesarrolloWeb.com que explica en detalle cómo controlar ventanas secundarias. No solo abrirlas, también aprendemos a cerrarlas, comunicar entre ventanas, etc. [Control de ventanas en Javascript](#)  
También hemos propuesto [crear un popup con Javascript y capas \(DHTML\)](#), que evita los sistemas que bloquean ventanas secundarias, ya que no es exactamente una ventana.

Si quieres, puedes descargarte el texto de esta ayuda técnica y los ejemplos.  [ventanassecundarias.zip](#) 6Kb

Artículo por **Miguel Angel Alvarez**

## Acceso por clave javascript

Lamentablemente, javascript no es un lenguaje con el que se pueda realizar un método interesante para hacer que algunas páginas de nuestro sitio solo sean accesibles si se introduce una clave correcta. Aun así, existe un mecanismo para poder realizar esto, que no es muy avanzado ni tampoco muy seguro, pero que puede dar el efecto en nuestras páginas que estamos deseando.

Se trata de colocar páginas web en nuestro espacio huérfanas de enlaces, para que nadie pueda acceder a ellas. Esta es toda la seguridad que les podemos dar a nuestras páginas: como no existen enlaces dirigidos hacia ellas, nadie podrá accederlas. La única manera de acceder a las páginas sería conocer el nombre de archivo y escribir la URL del mismo, pero como tampoco vamos a publicar el nombre de archivo, podremos estar casi seguros de que nadie acertará a construir la dirección de la página que queremos ocultar. Luego crearemos un formulario muy sencillo, que incluirá un campo de texto y un botón. En el campo de texto habrá que escribir el nombre del archivo que se desea ver y al pulsar el botón javascript seremos conducidos hacia la página que tenga ese nombre de archivo. En este punto pueden pasar dos cosas.

1. Que el nombre de archivo sea incorrecto, es decir, nos hemos inventado la clave y casi seguro que no hemos acertado con el nombre de la página escondida. En este caso, se mostraría una página de error típica, de esas que muestra el servidor cuando intentamos acceder a una página que no existe.
2. Que el nombre de la página sea correcto, es decir, que la clave que hemos introducido sea igual al nombre del archivo que queremos acceder. En este caso, javascript nos conducirá al lugar correcto y podremos ver la página oculta.

Veamos paso a paso cómo construir este sistema de acceso por clave

#### 1.- Las páginas que desarrollar

Tenemos que trabajar con 2 páginas web por lo menos, una para colocar el formulario y otra que sería la página oculta. Estas páginas las tendremos colocadas en el mismo directorio, con lo que simplificaremos un poco el problema.

#### 2.- Formulario para la clave

En la página que queremos poner el acceso por clave debemos colocar el siguiente formulario.

```
<FORM name=formclave>
<INPUT type=password name=clave>
<INPUT type=button value=Acceder>
</FORM>
```

#### 3.- Función que nos envía a la página oculta

Como la página oculta tiene como nombre de archivo la lo que se haya escrito en el campo de texto podremos acceder a ella de la siguiente forma.

```
<SCRIPT>
function acceso(){
    window.location = document.formclave.clave.value + ".html"
}
</SCRIPT>
```

La función es muy sencilla, lo único que hace es concatenar el nombre que se ha escrito en el campo de texto con ".html" y nos manda, utilizando window.location, directamente a la página cuyo nombre se acaba de construir.

Como hemos concatenado con ".html" el nombre del archivo escrito en el formulario, el nombre que escribamos deberá ir sin ".html".

#### 4.- Incluir en el botón la llamada a la función

Con el objetivo de que al pulsar el botón el navegador nos lleve a la página oculta, se ha de hacer que al pulsarlo, se llame a la función acceso definida en el punto anterior. Esto se consigue mediante el atributo onclick, que debemos insertar en la etiqueta del botón.

```
<INPUT type=button value=Acceder onclick="acceso()">
```

#### 5.- Código entero de la página

Podemos ver a continuación el código de la página entera. Solo enseñamos el código de la página que tiene el formulario, porque la página oculta podrá ser como cada uno desee.

```
<html>
<head>
  <title>clave acceso</title>
</head>

<body>
<SCRIPT>
function acceso(){
  window.location = document.formclave.clave.value + ".html"
}
</SCRIPT>

<FORM name=formclave>
<INPUT type=password name=clave>
<INPUT type=button value=Acceder onclick="acceso()">
</FORM>

</body>
</html>
```

#### 6.- Últimos apuntes y demo

Una cosa importante a la hora de conseguir que el script sea más seguro consiste en crear páginas con un nombre de archivo difícil de inventarse. Como el nombre de la página es la clave con la que se va a acceder a esa página necesitaremos que dicho nombre sea un poco complejo, por ejemplo, fks12dmxc53.html. Si la página clave se llamase por ejemplo, index.html cualquiera podría con un poco de imaginación inventarsela.

Antes de terminar, cabe repetir que este no es el método más seguro que existe para crear scripts para realizar accesos restringidos, solo es una pequeña astucia que "funciona". Para realizar este objetivo con mejores resultados tenemos lenguajes como ASP, PHP o CGI. También podemos restringir el acceso a las páginas utilizando el propio sistema operativo y la autenticación que implementa este, tal vez sea la opción más cómoda, aunque no es del todo probable que nuestro proveedor de alojamiento nos la permita.

Podemos ver [el ejemplo entero funcionando](#) en esta misma web.

*Artículo por **Miguel Angel Alvarez***



## Rollover con javascript

Hacer que cambie una imagen al pasar el ratón por encima, como invitando a pulsar le llamamos rollover. Es uno de los efectos más vistosos que podemos incluir en una página web con unas pocas líneas de Javascript, y sin necesidad de saber programar.



Ejemplo de rollover

Vamos a ver la técnica por la **práctica y con una elemental receta:**

### 1. Nombramos la imagen

Ponemos en la página la imagen que tiene el dibujo apagado. Además le asignamos un nombre, para poder referirnos a ella mediante JavaScript.

```

```

### 2. Ponemos un enlace en la imagen

Ahora ponemos el enlace al que queremos navegar en el momento en el que el usuario pinche en el.

```
<a href="ir_a.html">
```

### 3. Empezamos con JavaScript

Debemos colocar dos atributos HTML al enlace que nos van a servir para realizar el efecto buscado.

**OnMouseOver**, col él indicaremos, mediante JavaScript, la acción a realizar cuando se pose el ratón encima de la imagen.

**OnMouseOut** nos sirve para definir el evento de retirar el ratón de la imagen,

```
<a href="ir_a.html" onmouseover="-Código JavaScript-" onmouseout="-Código JavaScript-">
```

### 4. Tomamos las imágenes con JavaScript

Vamos a declarar dos variables con JavaScript para guardar las imágenes iluminada y apagada. Para ello vamos a utilizar la etiqueta <SCRIPT> de HTML. El script lo podemos colocar en cualquier sitio, pero sería adecuado colocarlo antes de la imagen.

Los números que vereis corresponden con la anchura y la altura de la imagen que estais tomando.

```
<script language=javascript>
iluminada = new Image(84,34)
iluminada.src = "dibujo_iluminado.gif"
apagada = new Image(84,34)
apagada.src = "dibujo_apagado.gif"
</script>
```

### 5. Escribimos el código de los eventos

Para acceder a un elemento de JavaScript utilizamos la jerarquía de objetos de JavaScript. Puede ser complicada, pero nuestro objetivo es simple, así lo haremos:

**window.document['nombre\_de\_la\_imagen'].src = variable\_imagen\_javascript.src**

En concreto, los atributos HTML de los eventos **onmouseover** y **onmouseout** quedarán así:

**onmouseover="window.document['imagen1'].src= iluminada.src**

**onmouseout="window.document['imagen1'].src = apagada.src**

5. Este es el código completo

```
<script language=javascript>
    iluminada = new Image(84,34)
    iluminada.src = "dibujoiluminado.gif"
    apagada = new Image(84,34)
    apagada.src = "dibujoapagado.gif"
</script>
<a href="ir_a.html"
onmouseover="window.document['imagen1'].src = iluminada.src"
onmouseout="window.document['imagen1'].src = apagada.src">

</a>
```

Esto es lo único que debes hacer para iluminar una imagen, es un primer paso, ahora puedes probar con un grupo de imágenes para crear una [barra de navegación dinámica con Javascript!](#)

En este archivo zip puedes encontrar el código y las imágenes del ejemplo de arriba.

 [rollover.zip](#) 16Kb

Artículo por *Miguel Angel Alvarez*

## Navegador dinámico javascript

Vamos a ver como hacer, con Javascript y unas cuantas imágenes, **una barra de navegación para una página web**, que cambie cuando el ratón pase por encima.

Esta ayuda técnica está pensada para leerla a continuación del informe [Rollover con javascript](#), publicado en desarrolloweb, pues contiene las bases sobre las que vamos a trabajar ahora.

1. Construimos las imágenes

Tenemos que construir dos versiones de barra de navegación, una que esté iluminada, por así decirlo, y otra que esté un poco mas apagada. Al pasar el ratón cambiaremos de una a otra.

Aquí están las imágenes que proponemos para este ejercicio:

**APAGADAS**

**ILUMINADAS**

Portada
Aficiones
Curriculum
Mi tierra
Amigos
Links

Portada
Aficiones
Curriculum
Mi tierra
Amigos
Links

Portada
Aficiones
Curriculum
Mi tierra
Amigos
Links

## 2. Creamos la barra con HTML

Con una tabla de HTML vamos a hacer la barra de navegacion para la página, aun sin movimiento. Con estos detalles:

- En un principio colocamos las imágenes apagadas
- Cada imagen tiene un enlace a la página correspondiente
- Hemos dado un nombre distinto a cada imagen con el atributo **name**. desde imagen1 hasta imagen6.
- Nuestra tabla tiene cellpadding y cellspacing 0 para que no quede separacion entre las imágenes. Por esta última razón, tampoco hay que dehar espacios en blanco en el código HTML entre los TD y las imágenes.

```
<table cellspacing="0" cellpadding="0" border="0">
<tr>
  <td><a href="portada.html"></a></td>
</tr>
<tr>
  <td><a href="aficciones.html"></a></td>
</tr>
<tr>
  <td><a href="curriculum.html"></a></td>
</tr>
<tr>
  <td><a href="mitierra.html"></a></td>
</tr>
<tr>
  <td><a href="amigos.html"></a></td>
</tr>
<tr>
  <td><a href="links.html"></a></td>
</tr>
</table>
```

## 3. Precargamos las imágenes

Para tener las imágenes ya en nuestro explorador antes de que se vayan a utilizar, debemos precargarlas usando Javascript, así conseguiremos que el efecto de rollover sea rápido, y cambien las imágenes velozmente según se pase el ratón.

Utilizaremos este código, que se coloca en la cabecera del documento HTML:

```
<script>
  iluminada1 = new
Image(110,16)
iluminada1.src = "portada2.gif"
apagada1 = new Image(110,16)
apagada1.src = "portada1.gif"

  iluminada2 = new Image(110,16)
iluminada2.src = "aficciones2.gif"
apagada2 = new Image(110,16)
apagada2.src = "aficciones1.gif"
```

```

iluminada3 = new Image(110,16)
iluminada3.src =
"curriculum2.gif"
apagada3 = new Image(110,16)
apagada3.src = "curriculum1.gif"

iluminada4 = new Image(110,16)
iluminada4.src = "mitierra2.gif"
apagada4 = new Image(110,16)
apagada4.src = "mitierra1.gif"

iluminada5 = new Image(110,16)
iluminada5.src = "amigos2.gif"
apagada5 = new Image(110,16)
apagada5.src = "amigos1.gif"

iluminada6 = new Image(110,16)
iluminada6.src = "links2.gif"
apagada6 = new Image(110,16)
apagada6.src = "links1.gif"

</script>

```

#### 4. Los eventos

Tenemos que definir los eventos **onmouseover** y **onmouseout** para cada uno de los enlaces, indicando el cambio de la imagen a iluminada y a apagada respectivamente.

```

onmouseover="window.document['imagen1'].src =iluminada1.src"
onmouseout="window.document['imagen1'].src = apagada1.src"

onmouseover="window.document['imagen2'].src =iluminada2.src"
onmouseout="window.document['imagen2'].src = apagada2.src"

onmouseover="window.document['imagen3'].src =iluminada3.src"
onmouseout="window.document['imagen3'].src = apagada3.src"

onmouseover="window.document['imagen4'].src =iluminada4.src"
onmouseout="window.document['imagen4'].src = apagada4.src"

onmouseover="window.document['imagen5'].src =iluminada5.src"
onmouseout="window.document['imagen5'].src = apagada5.src"

onmouseover="window.document['imagen6'].src =iluminada6.src"
onmouseout="window.document['imagen6'].src = apagada6.src"

```

#### 5. Código entero

Eso es todo. A continuación podemos ver el código entero de este ejemplo:

```

<html>
<head>
<title>Navegador</title>
<body>
<input type="button" value="iluminada1" onmouseover="window.document['imagen1'].src =iluminada1.src" onmouseout="window.document['imagen1'].src = apagada1.src" />
<input type="button" value="iluminada2" onmouseover="window.document['imagen2'].src =iluminada2.src" onmouseout="window.document['imagen2'].src = apagada2.src" />
<input type="button" value="iluminada3" onmouseover="window.document['imagen3'].src =iluminada3.src" onmouseout="window.document['imagen3'].src = apagada3.src" />
<input type="button" value="iluminada4" onmouseover="window.document['imagen4'].src =iluminada4.src" onmouseout="window.document['imagen4'].src = apagada4.src" />
<input type="button" value="iluminada5" onmouseover="window.document['imagen5'].src =iluminada5.src" onmouseout="window.document['imagen5'].src = apagada5.src" />
<input type="button" value="iluminada6" onmouseover="window.document['imagen6'].src =iluminada6.src" onmouseout="window.document['imagen6'].src = apagada6.src" />

```

```

minada4 = new Image(110,16)
minada4.src = "mitierra2.gif"
agada4 = new Image(110,16)
agada4.src = "mitierra1.gif"

minada5 = new Image(110,16)
minada5.src = "amigos2.gif"
agada5 = new Image(110,16)
agada5.src = "amigos1.gif"

minada6 = new Image(110,16)
minada6.src = "links2.gif"
agada6 = new Image(110,16)
agada6.src = "links1.gif"

<script>
</script>
<table border="1" style="background-color:#6e6d52; color:white; text-align:center; width:100%; border-collapse: collapse;">
|  |
| --- |
| <a href="portada.html" onmouseover="window.document['imagen1'].src =iluminada1.src" onmouseout="window.document['imagen1'].src = iluminada1.src"></a></td> |
| <a href="aficciones.html" onmouseover="window.document['imagen2'].src =iluminada2.src" onmouseout="window.document['imagen2'].src = iluminada2.src"></a></td> |
| <a href="curriculum.html" onmouseover="window.document['imagen3'].src =iluminada3.src" onmouseout="window.document['imagen3'].src = iluminada3.src"></a></td> |
| <a href="mitierra.html" onmouseover="window.document['imagen4'].src =iluminada4.src" onmouseout="window.document['imagen4'].src = iluminada4.src"></a></td> |
| <a href="amigos.html" onmouseover="window.document['imagen5'].src =iluminada5.src" onmouseout="window.document['imagen5'].src = iluminada5.src"></a></td> |
| <a href="links.html" onmouseover="window.document['imagen6'].src =iluminada6.src" onmouseout="window.document['imagen6'].src = iluminada6.src"></a></td> |

```

Ejemplo funcionando



Evidentemente, hay muchas otras maneras de hacer una barra de navegación, pero esta es una buena forma. Con un poco de creatividad podrás crear unas bonitas imágenes que hagan unos bonitos efectos al pasar el ratón por encima.

En este archivo zip puedes encontrar el código y las imágenes del ejemplo.

 [navegador.zip](#) 11Kb

Artículo por **Miguel Angel Alvarez**

## Navegador desplegable

Esta muy de moda incluir un navegador de un sitio web consistente en un campo desplegable de formulario o caja de selección que, al seleccionar uno de sus elementos, que se corresponden con secciones del sitio, nos lleve a la sección que pretendemos visitar de manera automática.

Se puede ver un ejemplo de estos en la cabecera en uso en desarrolloweb en el momento de escribir este artículo. También se puede ver a continuación.

Este tipo de control posee varias ventajas, entre las que podemos destacar:

- Ocupa poco espacio en la página
- Se pueden poner tantas opciones como se desee
- Es una herramienta muy visual y práctica
- Es muy fácil de implementar

Pasos a realizar

Para conseguir un navegador así en nuestra página web debemos enfrentarnos a una tarea que se podría dividir en dos partes. Por un lado, debemos crear un formulario que contenga la caja de selección desplegable y por otro, un sencillo script que le de vida al recuadro, es decir, que prepare al recuadro para que el navegador se dirija a la página seleccionada. Vamos a ver por separado cada uno de estos elementos.

Formulario

Costará de la etiqueta `<form>` para abrir y cerrar el formulario. Le daremos un nombre ("navegador") al formulario para poder acceder a él más tarde usando Javascript.

Dentro del formulario colocaremos un único elemento: el campo de selección multiple desplegable. Daremos un nombre también al campo ("secciones") para poder acceder a él usando Javascript. Este campo contendrá las distintas opciones que queremos que se presenten en el menú desplegable. Éstas podrán ser secciones de tu web o también páginas que estén fuera del sitio.

Vamos a ver el código del formulario antes de continuar con la explicación:

```
<form name=navegador>
<select name="secciones">
<option value="no">Secciones y servicios de desarrolloweb
<option value="no">-----
<option value="http://www.desarrolloweb.com/">Portada
<option value="http://www.desarrolloweb.com/reportajes/manuales.asp">Manuales
<option value="http://www.desarrolloweb.com/promocion">Promoción de páginas
<option value="http://www.desarrolloweb.com/programas">Programas
<option value="http://www.desarrolloweb.com/reportajes/ayudastecnicas.asp">Ayudas técnicas
<option value="http://www.desarrolloweb.com/noticias">Noticias
<option value="http://www.desarrolloweb.com/reportajes/cosecha2000.asp">Cosecha de 2000
<option value="http://www.desarrolloweb.com/favoritas">Favorita del mes
<option value="http://www.desarrolloweb.com/vuestraspaginas/">Vuestras páginas
<option value="http://www.desarrolloweb.com/colabora/">Colaborar
<option value="no">-----
<option value="http://www.desarrolloweb.com/listacorreo/">Lista de correo de ayuda
<option value="http://www.desarrolloweb.com/mailnovedades.asp">Boletín de novedades
```

```
<option value="http://www.desarrolloweb.com/librovisitas/">Libro de visitas
</select>
</form>
```

Si te fijas, cada opción se corresponde con una de las secciones o servicios de desarrolloweb y está compuesta de dos partes importantes, la primera corresponde con el atributo **value** de la etiqueta `<option>`, que es igual a la URL absoluta de la página a la que nos queremos dirigir. La segunda parte destacable corresponde con el nombre que deseamos que se vea en la caja de selección.

Es también importante destacar que hemos incluido alguna opción que no queremos que nos dirija a ningún lugar. Son opciones para separar las secciones de los servicios o la opción primera, para indicar que este menú desplegable contiene las secciones y servicios de desarrolloweb. Las opciones que no queremos que nos lleven a otra página las hemos marcado con un "no" en su atributo value.

El script

Ahora vamos a ver cuál es el script que utilizamos para animar este menú desplegable. Es muy sencillo, lo podemos ver a continuación:

```
<script language=javascript>
function destino(){
url =
document.navegador.secciones.options[document.navegador.secciones.selectedIndex].value
if (url != "no") window.location = url;
}
</script>
```

Básicamente es una función que recupera el *value* de la opción seleccionada en la caja de selección desplegable y lo almacena en una variable llamada url. Posteriormente, si la variable url no contiene la palabra "no", lleva al navegador a la posición seleccionada, es decir, al url que habíamos recogido. Recordar que si marcábamos el value de una opción a "no" es que no deseábamos que el navegador viajase a otra dirección.

El evento OnChange

Esto no funcionaría si no vinculásemos el evento onchange de la caja de selección a la función destino que hemos visto hace un momento. El evento onchange se dispara cuando ha cambiado el valor seleccionado dentro del menú desplegable y debemos hacer que llame a la función destino. Para ello, en la etiqueta `<select>` debemos incluir el siguiente atributo `onchange="destino()"`. La etiqueta quedaría así:

```
<select name="secciones" onchange="destino()">
```

Con todo esto junto ya tienes un bonito navegador desplegable y, esperamos, la capacidad para personalizarlo a tu gusto. Se puede ver también el [ejemplo completo de este navegador](#).

Una cosa más. Si tu sitio tiene frames deberías hacer unos cambios al script para que todo funcione correctamente. Si es este tu caso, lee el reportaje [cómo hacer una navegador desplegable cuando tu sitio tiene frames](#).

Artículo por **Miguel Angel Alvarez**

## Navegador desplegable con frames

Este es un reportaje que se tiene que leer a continuación del reportaje [Cómo hacer un navegador desplegable](#), publicado en desarrolloweb.com. En ese reportaje enseñábamos a crear un navegador desplegable con un elemento SELECT de un formulario.

Muchos visitantes han utilizado ya el script con éxito, pero algunos han escrito con una duda para su utilización en una página realizada con frames. La duda consiste en el navegador solo nos actualiza el frame en el que está, y lo interesante para ellos sería que actualizase un frame distinto. Es un problema muy lógico dado que a menudo se coloca el navegador de modo que esté siempre visible, en un frame donde tenemos los controles de navegación y el área que deseamos que se actualice es la correspondiente al frame principal.

Cambios en el script

El único sitio donde vamos a tener que hacer cambios es en el script que contiene la función a la que llamamos destino(). Hay que adaptar esa función para que podamos cambiar la página de un frame distinto al que estamos.

En nuestro anterior ejemplo hacíamos `window.location = url` para cambiar el contenido del frame donde estaba el navegador. Ahora debemos cambiar el `window.location` de un frame distinto a este y para acceder a location de un frame distinto se consigue a través de esta encadenación de objetos:

```
window.parent.frames[0].window.location
```

`frames[]` es un vector de frames donde el primer frame del FRAMESET sería `frames[0]`, el segundo sería `frames[1]` y así sucesivamente. Por si no ha quedado claro, veamos con un ejemplo.

### Tenemos este FRAMESET

```
<frameset rows="*,40">
<frame name="principal" src="index.html" marginwidth="10" marginheight="10" scrolling="auto"
frameborder="no">
<frame name="menudesplegable" src="despleg.html" marginwidth="10" marginheight="10" scrolling="auto"
frameborder="no">
</frameset>
```

En el segundo frame tenemos el la página que contiene menú desplegable. Como es el segundo frame accederíamos a su location de esta manera:


```
window.parent.frames[1].window.location = url
```

### El script entero quedaría así:

```
<script language=javascript>
function destino(){
    url = document.navegador.secciones.options[document.navegador.secciones.selectedIndex].value
    if (url != "no") window.parent.frames[0].window.location = url;
}
</script>
```

Eso es todo, ya no hace falta cambiar más cosas para cumplir nuestros objetivos.



Si quieres, puedes descargarte un ejemplo de esta ayuda funcionando.  [desplegableconframes.zip](#) 2Kb

Artículo por **Miguel Angel Alvarez**

## Texto en movimiento en la barra de estado

Vamos a ver en este artículo cómo hacer para que se mueva un texto por la barra de estado de nuestro navegador. Es un script bastante sencillo y también bastante corriente. Sin duda puede resultar fácil tomar el script, copiarnoslo en nuestra página y hacerlo funcionar, pero nosotros vamos a procurar una explicación para que todo quede más claro y entendamos un poco lo que estamos haciendo.

Este script va a crear un texto que se movera de derecha a izquierda por la barra de estado. Se puede ver en la [página de ejemplo](#). Ahora veamos los distintos pasos.

### 1. Defino unas variables iniciales

El script puede mover el texto que nosotros deseemos y para configurarlo se ha de crear una variable que hemos llamado texto\_estado donde introduciremos nuestro texto.

```
var texto_estado = "      Bienvenidos a mi pagina web"
```

Notemos que se han introducido unos espacios antes que el texto. Son para que se cree un espacio en la barra de estado entre la salida del texto y la entrada de este por el otro lado. El número de espacios en blanco se puede modificar libremente, así como el texto que se muestra.

También será necesario crear una variable llamada posicion donde vamos a guardar la posición del texto en la barra de estado.

```
var posicion = 0
```

### 2. Función para mover el texto

Ahora vamos a ver la función, a la que llamaremos mueve\_texto(), que se encarga de mover el texto por la barra de estado. Entender esta función puede ser un poco complejo si no se conoce un poco el lenguaje Javascript. En cualquier caso, podemos también copiarla y pegarla en nuestras páginas aunque no la consigamos entender. Realiza cuatro acciones básicas:

- Mueve la posición actual, actualizando la variable posicion. Si hemos llegado al final de la cadena se vuelve al principio  

```
if (posicion < texto_estado.length)
    posicion ++;
else
    posicion = 1;
```
- Crea una cadena a partir del texto original. La cadena creada contiene el texto que hay desde la posición actual hasta el final concatenado al que hay desde el principio hasta la posición actual. Este es el paso que realmente genera el movimiento, porque va cambiando la cadena que luego escribiremos a medida que la posición también cambia.  

```
string_actual = texto_estado.substring(posicion) + texto_estado.substring(0,posicion)
```

- Escribe la cadena resultante de la operación anterior en la barra de estado.  
`window.status = string_actual`
- La función se llama a si misma para continuar el movimiento. Para ello se utiliza una función muy socorrida, `setTimeout()`, que sirve para ejecutar una sentencia con un retardo de tiempo. La función recibe la sentencia para ejecutar (que en este caso es una llamada a esta misma función) y el número de milisegundos que tiene que esperar para ejecutarla, en este caso 50.  
`setTimeout("mueve_texto()",50)`

### La función entera tiene este código:

```
function mueve_texto(){
  if (posicion < texto_estado.length)
    posicion ++;
  else
    posicion = 1;
  string_actual = texto_estado.substring(posicion) + texto_estado.substring(0,posicion)
  window.status = string_actual
  setTimeout("mueve_texto()",50)
}
```

### 3. Llamamos a la función

Para empezar a mover el texto por la página tenemos que realizar una llamada a la función que se encarga de ello. Será más claro el código de la página si colocamos la llamada a la función después de que haya sido definida, aunque no es obligatorio.

```
mueve_texto()
```

### 4. Todo junto

Para acabar, podemos observar a continuación el código entero de una página web que mueve texto por su barra de estado. Es una página bastante sencilla después de todo.

```
<html>
<head>
  <title>Texto en la barra de estado</title>
  <script language="javascript">
    //texto del mensaje
    var texto_estado = "    Bienvenidos a mi pagina web"
    var posicion = 0

    //funcion para mover el texto de la barra de estado
    function mueve_texto(){
      if (posicion < texto_estado.length)
        posicion ++;
      else
        posicion = 1;
      string_actual = texto_estado.substring(posicion) + texto_estado.substring(0,posicion)
      window.status = string_actual
      setTimeout("mueve_texto()",50)
    }
    mueve_texto()
  </script>
</head>

<body>
<h1>Ejemplo de script con un texto en la barra de estado</h1>

</body>
</html>
```

Se puede [ver el ejemplo en funcionamiento](#).

## 5. Otro ejemplo

Dependiendo del script que utilicemos para mover el texto de la barra de estado conseguiremos unos efectos u otros. A continuación podemos ver un [segundo ejemplo sobre cómo mover un texto por la barra de estado](#) utilizando un efecto de movimiento distinto.

No vamos a comentar este segundo ejemplo porque ya se encuentra comentado en el propio código fuente, pero podremos ver que es muy parecido al anterior.

```
<html>
<head>
  <title>Segundo ejemplo de texto en movimiento</title>
</head>

<body>
<h1>Texto en movimiento en la barra de estado</h1>
<h2>Ejemplo 2</h2>

<script language="javascript">

//variable con el texto a mostrar
var texto = "Bienvenidos a mi pagina web!!!"
//variable con la posicion en el texto. poner siempre a 0
var pos = 0

//creo una funcion para cambiar el texto de la barra de estado
function textoEstado(){
  //incremento la posicion en 1 y extraigo el texto a mostrar en este momento.
  pos = pos + 1
  textoActual = texto.substring(0,pos)
  //pongo el texto que quiero mostrar en la barra de estado del navegador
  window.status = textoActual
  //Llamamos otra vez a esta funcion para que continúe mostrando texto
  if (pos == texto.length){
    //si hemos llegado al final, vuelvo al principio y hago un retardo superior
    pos = 0
    setTimeout("textoEstado()",1500)
  } else{
    //si no hemos llegado al final, sigo con la funcion con un retardo minimo.
    setTimeout("textoEstado()",100)
  }
}

//llamo a la función para poner el texto en movimiento
textoEstado()
</script>

</body>
</html>
```

Artículo por [Miguel Angel Alvarez](#)

## Mostrar capas y ocultar capas con IE 4,5 NS 4

El propósito de este artículo, es hacer que se puedan mostrar capas y ocultar capas cuando se

pasa con el ratón por encima de otra. Gracias a esta técnica, las páginas tendrán un aspecto mucho más dinámico.

**Nota:** desde que se publicó este artículo han salido muchas versiones de navegadores nuevos. Para trabajar con capas y hacer código compatible con todos los navegadores tenemos un manual en DesarrolloWeb.com que es de muy interesante lectura: [Taller de Cross-Browser DHTML](http://www.desarrolloweb.com/manuales/22/)

1. Conocer el navegador que está utilizando el usuario

Como Netscape e Internet Explorer llaman de forma distinta a las capas, lo primero que haremos, será detectar qué navegador se está utilizando. Para ello, lo que vamos a hacer es crear dos variables globales dentro de nuestro Script:

- ns4: que estará a true si el navegador utilizado es Netscape
- ie4: que estará a true si el navegador utilizado es Internet Explorer

**var ns4,ie4**

La función que utilizamos para inicializar el valor de las variables es la siguiente:

```
ns4 = (document.layers)? true:false  
ie4 = (document.all)? true:false
```

2. Crear una variable que contenga a cada una de las capas

Una vez están creadas e inicializadas las variables del navegador, tenemos que crear otra variable, que contenga a la capa que queremos mostrar u ocultar. Esta variable se inicializará de forma distinta dependiendo del navegador que esté utilizando el usuario.

La siguiente función inicializa el valor de la variable "capa" con el nombre que tiene la capa que queremos mostrar u ocultar, en este caso se llamará "menu".

**var capa**

```
function init() {  
  if (ns4) {  
    capa = document.menu  
  }  
  if (ie4) {  
    capa = menu.style  
  }  
}
```

3. Crear una función que muestre una capa

A continuación, definimos una función que dependiendo de si se está utilizando el navegador Netscape o el Internet Explorer, cambiará el atributo de la capa que se le pasa por parámetro a visible. Para Netscape, llamará al Método Show, que será el encargado de cambiar el atributo. En Internet Explorer cambiaremos el atributo directamente.

```
function muestra(obj) {  
  if (ns4) obj.visibility = "show"  
  else if (ie4) obj.visibility = "visible"  
}
```

4. Crear una función que oculte una capa

Al igual que con la función anterior, dependiendo si el navegador es Netscape o IE, cambiaremos de una forma u otra el atributo de la capa a oculto

```
function oculta(obj) {  
  if (ns4) obj.visibility = "hide"  
  else if (ie4) obj.visibility = "hidden"  
}
```

#### 5. Crear las dos capas

Una vez tenemos definido las acciones de mostrar u ocultar capa, hay que crear el cuerpo del documento, en primer lugar, pediremos al navegador que ejecute la función Init, de forma que la variable capa esté lista para usar:

```
<body Onload="Init()">
```

Además tenemos que crear dos capas, la primera de ellas va a ser la que se va a ocultar y mostrar. El nombre de la capa deberá ser Menu, que es la que hemos decidido utilizar en el punto 2.

```
<div id="menu" style="position: ..... ">
```

Además, hay que crear la segunda capa, que mostrará la capa menu cuando el ratón esté encima de ella, y la ocultará cuando no. Cada una de estas acciones las realizará las funciones Muestra y Oculta, definidas en el punto 4.

Colocaremos además dentro de esta capa un enlace, con los manejadores de eventos necesarios para realizar acciones cuando se pose o no el ratón sobre el. Cuando se produzca el evento OnMouseOver (que el ratón entre en el enlace dentro de la capa), pediremos al navegador que ejecute la función Muestra, y cuando se produzca el evento OnMouseOut (que el ratón salga del enlace de la capa), pediremos al navegador que ejecute la función Oculta. Por lo que la definición de la capa quedará de la siguiente forma:

```
<div id="nombreCapa" Style="position:...."><a href="pagina.html"  
OnMouseOver="Muestra(capa)" OnMouseOut= "Oculta(capa)">Textode  
lacapa</a></div>
```

Y con esto ya está todo...

#### 6. Código completo:

```
<html>  
<head>  
  <title>Untitled</title>  
  
  <script language="javascript">  
    var capa  
  
    ns4 = (document.layers)? true:false  
    ie4 = (document.all)? true:false  
  
    function init() {  
      if (ns4) {  
        capa = document.menu  
      }  
      if (ie4) {  
        capa = menu.style  
      }  
    }  
  
    function muestra(obj) {  
      if (ns4) obj.visibility = "show"  
      else if (ie4) obj.visibility = "visible"  
    }  
  
    function oculta(obj) {  
      if (ns4) obj.visibility = "hide"  
      else if (ie4) obj.visibility = "hidden"  
    }  
  
  </script>  
</head>  
<body onLoad="init()">
```

```
<div id="menu" style="position:Absolute;left:50; top:60; background-color:#ff1133; visibility:hidden">hola</div>
<div id="CapaNormal" style="position:absolute;left:50; top:150;background-color:pink;"><a href="#"
OnMouseOver="muestra(capa)" OnMouseOut="oculta(capa)">Ponte encima</a>...</div>
</body>
</html>
```

Con los conocimientos aquí aprendidos puedes intentar crear una barra de navegación dinámica que tenga varias opciones que, al pasar el ratón por encima, se vean sus distintas descripciones. Puedes encontrar en este sitio un [artículo que te informa sobre cómo hacerlo](#).

**Nota:** Este artículo, aunque puede servir de ayuda e introducción al trabajo con capas, se encuentra un poco desfasado. Existen herramientas que pueden hacernos la vida más fácil. Si deseamos hacer código que funcione correctamente para el trabajo con capas, podemos apoyarnos en unas librerías que nos permitan su manejo compatible con todos los navegadores. En el artículo [Cross-Browser. DHTML compatible con todos los navegadores](#) tenemos más detalles.

Artículo por **Carlos Luis Cuenca**

## Navegador con capas

Ya hemos recibido más de una solicitud pidiendonos la realización de este reportaje que, en unos simples pasos, te permitirá crear una barra de navegación dinámica usando capas. Haremos que unas capas muestren información sobre el enlace donde esté situado el ratón.

Para comprender este ejercicio es muy recomendable que se lea después del [reportaje para ocultar y mostrar capas](#), publicado en este mismo sitio. Puede que antes de continuar quieras [ver el resultado de lo que vamos a crear](#).

### 1. Colocamos la capa donde se situarán los enlaces

La capa donde van a estar los enlaces tendría esta forma, como es código HTML la colocaremos dentro del <BODY> de la página:

```
<div align="" id="enlaces" style="font-size: 10pt; font-family: arial,verdana,Helvetica; font-weight: bold; color: #333399; position: absolute; height: 58; width: 100; background-color: #ffffcc; text-align: right;top: 10px;left: 10px">
<a href="#" OnMouseOver="muestra(capa1)" OnMouseOut="oculta(capa1)">Mis Amigos</a>
<br>
<a href="#" OnMouseOver="muestra(capa2)" OnMouseOut="oculta(capa2)">Mi pueblo</a>
<br>
<a href="#" OnMouseOver="muestra(capa3)" OnMouseOut="oculta(capa3)">Enlaces</a>
</div>
```

Lo importante de la capa, y donde tenemos que fijar la atención es en los enlaces. Cuando colocamos el ratón sobre un enlace y cuando lo retiramos de él se llama a las funciones **muestra()** y **oculta()** pasándoles el nombre de la capa que se desea mostrar o ocultar.

Si tenemos más opciones en nuestra barra de navegación nada más tendríamos que añadir más enlaces.

También hay que destacar la declaración de estilos de la capa (atributo **style** de la etiqueta <DIV>), pero esto no son más que [estilos css](#).

### 2. Colocamos las capas de explicación de cada enlace

Cada capa es una etiqueta <DIV>, se muestran a continuación:

```
<div id="descripcion1" style="position:Absolute;left:150; top:10; background-color:#cccccc; visibility:hidden;width:250px;height:58px;font-size: 8pt; font-family: arial,verdana,Helvetica; color: #333399;padding:5px">Los amigos reunidos jam&acute;s ser&acute;n vencidos<br>Con todo el texto que desees</div>
<div id="descripcion2" style="position:Absolute;left:150; top:10; background-color:#cccccc; visibility:hidden;width:250px;height:58px;font-size: 8pt; font-family: arial,verdana,Helvetica; color: #333399;padding:5px">El pueblo
```

m&aacute;s tur&iacute;stico de todo el polo norte, a 20.000 kil&oacute;metros de la gasolinera m&aacute;s cercana</div>

```
<div id="descripcion3" style="position:Absolute;left:150; top:10; background-color:#cccccc;
visibility:hidden;width:250px;height:58px;font-size: 8pt; font-family: arial,verdana,helvetica; color: #333399;padding:5px">Todos los
enlaces que necesitas para entrar en Internet con las m&aacute;ximas garant&iacute;as de &eacute;xito.</div>
```

En este código hay que destacar que cada capa tiene un nombre, asignado gracias al **atributo id** de la etiqueta <DIV>.

Además, todas las capas tienen una buena lista de estilos CSS, el más significativo en este caso es el que hace que en un principio las capas estén ocultas: **visibility:hidden**.

Los demás estilos aplicados a cada capa son nada más que estilos, los puedes cambiar y ponerlos a tu gusto, así como el texto de las capas que será el que convenga en cada momento.

Por último, señalar que si habías incluido más enlaces en la barra de navegación, deberás poner más capas con descripciones de estos.

### 3. Colocamos el Javascript

Hay que recordar que este ejercicio está hecho a continuación del reportaje de [cómo mostrar y ocultar capas con Javascript](#), y que la mayor parte del siguiente código Javascript ya está comentado.

En líneas generales, se debe conservar el mismo Javascript que el de la página de mostrar-ocultar capas. Veamos el código y a continuación lo comentamos:

```
<script language="javascript">
var capa

ns4 = (document.layers)? true:false
ie4 = (document.all)? true:false

function init() {
  if (ns4) {
    capa1 = document.descripcion1
    capa2 = document.descripcion2
    capa3 = document.descripcion3
  }
  if (ie4) {
    capa1 = descripcion1.style
    capa2 = descripcion2.style
    capa3 = descripcion3.style
  }
}

function muestra(obj) {
  if (ns4) obj.visibility = "show"
  else if (ie4) obj.visibility = "visible"
}

function oculta(obj) {
  if (ns4) obj.visibility = "hide"
  else if (ie4) obj.visibility = "hidden"
}
</script>
```

En las primeras líneas detectamos cuál es el navegador que está utilizando el usuario.

Luego tenemos la función **init()**, que es la única que cambia con respecto al ejemplo de mostrar-ocultar capas. En concreto, hemos colocado el código que sirve para inicializar las variables de capa, que se asocian con las capas donde has colocado las descripciones.

Para finalizar con el script, se pueden ver las funciones que muestran y ocultan capas.

### 4. Último detalle

No se nos debe olvidar colocar en la etiqueta <BODY> el evento **onload** para ejecutar la función **init()** una vez que se ha terminado de cargar la página. De esta manera:

```
<body onLoad="init()">
```

## 5. El código completo de la página

```
<html>
<head>
<title>Navegador con capas</title>

<script language="javascript">
var capa

ns4 = (document.layers)? true:false
ie4 = (document.all)? true:false

function init() {
  if (ns4) {
    capa1 = document.descripcion1
    capa2 = document.descripcion2
    capa3 = document.descripcion3
  }
  if (ie4) {
    capa1 = descripcion1.style
    capa2 = descripcion2.style
    capa3 = descripcion3.style
  }
}

function muestra(obj) {
  if (ns4) obj.visibility = "show"
  else if (ie4) obj.visibility = "visible"
}

function oculta(obj) {
  if (ns4) obj.visibility = "hide"
  else if (ie4) obj.visibility = "hidden"
}

</script>
</head>
<body onLoad="init()" link="#333399" vlink="#333399">

<!-- CAPA CON LOS DISTINTOS ENLACES -->
<div align="" id="enlaces" style="font-size: 10pt; font-family: arial,verdana,Helvetica; font-weight: bold; color: #333399; position: absolute; height: 58; width: 100; background-color: #ffffcc; text-align: right; top: 10px; left: 10px">
<a href="#" OnMouseOver="muestra(capa1)" OnMouseOut="oculta(capa1)">Mis Amigos</a>
<br>
<a href="#" OnMouseOver="muestra(capa2)" OnMouseOut="oculta(capa2)">Mi pueblo</a>
<br>
<a href="#" OnMouseOver="muestra(capa3)" OnMouseOut="oculta(capa3)">Enlaces</a>
</div>

<!-- CAPAS CON LAS EXPLICACIONES -->
<div id="descripcion1" style="position:Absolute;left:150; top:10; background-color:#cccccc; visibility:hidden;width:250px;height:58px;font-size: 8pt; font-family: arial,verdana,Helvetica; color: #333399;padding:5px">Los amigos reunidos jam&aacute;s ser&aacute;n vencidos<br>Con todo el texto que desees</div>
<div id="descripcion2" style="position:Absolute;left:150; top:10; background-color:#cccccc; visibility:hidden;width:250px;height:58px;font-size: 8pt; font-family: arial,verdana,Helvetica; color: #333399;padding:5px">El pueblo m&aacute;s tur&iacute;stico de todo el polo norte, a 20.000 kil&oacute;metros de la gasolinera m&aacute;s cercana</div>
<div id="descripcion3" style="position:Absolute;left:150; top:10; background-color:#cccccc; visibility:hidden;width:250px;height:58px;font-size: 8pt; font-family: arial,verdana,Helvetica; color: #333399;padding:5px">Todos los enlaces que necesitas para entrar en Internet con las m&aacute;ximas garant&iacute;as de &eacute;xito.</div>

</body>
</html>
```

Aquí acabó todo. Este ejemplo es muy versátil, seguro que no te cuesta nada añadir nuevas capas y crear unas barras totalmente personalizadas. Si deseas ver el ejemplo en movimiento, [pulsa este enlace](#).

Si lo deseas, puedes descargar los textos y ejemplos de esta ayuda.  [reportajecapas2.zip](#) 4Kb

Artículo por **Miguel Angel Alvarez**



## Reloj en Javascript

Vamos a ver un taller práctico sobre Javascript con el cual podremos incluir un reloj en nuestra página web. Es un sencillo script, que podremos colocar con tan sólo cortar y pegar, aunque procuraremos explicarlo un poco para los que estén en condiciones de entender Javascript.

Para empezar, vamos a ver el relojito que pretendemos crear:



Es un reloj muy sencillito de manera intencionada, para que podamos entender bien el ejercicio. Luego lo complicaremos un poco más para darle algo de vistosidad.

### Construir el formulario

Empezamos por colocar el campo de texto donde se mostrará el reloj. Debemos colocar un pequeño formulario donde tan solo tendremos un campo de texto.

```
<form name="form_reloj">
<input type="text" name="reloj" size="10">
</form>
```

No debería haber ningún problema en estas líneas de HTML. Sólo colocamos las etiquetas de inicio y final del formulario y dentro un campo de texto. Tanto al formulario como al campo de texto les asignamos un nombre para luego accederlos por ese nombre mediante Javascript.

### Función para actualizar el valor del reloj

Tenemos que construir una función que recoja la hora del sistema y la muestre en el campo de texto.

Para tomar la hora vamos a hacer uso del objeto Date de Javascript.

```
momentoActual = new Date()
```

Si creamos una nueva instancia del objeto Date sin pasarle parámetros se inicializa a la fecha y hora actuales.

Luego tenemos que obtener de esa instancia de Date lo que nos interesa, que es la hora, los minutos y segundos. Esto lo hacemos utilizando los correspondientes métodos del objeto Date.

```
hora = momentoActual.getHours()
minuto = momentoActual.getMinutes()
segundo = momentoActual.getSeconds()
```

Ahora combinamos los resultados para construir la hora con el formato que deseamos.

```
horaImprimible = hora + " : " + minuto + " : " + segundo
```

Por último colocamos en el campo de texto del formulario el valor actual de la hora.

```
document.form_reloj.reloj.value = horaImprimible
```

Por ahora la función queda de esta manera:

```
function mueveReloj(){
    momentoActual = new Date()
    hora = momentoActual.getHours()
    minuto = momentoActual.getMinutes()
    segundo = momentoActual.getSeconds()

    horaImprimible = hora + " : " + minuto + " : " + segundo

    document.form_reloj.reloj.value = horaImprimible
}
```

### La función debe llamarse a si misma

Con estas líneas de código obtenemos la hora y la actualizamos en su campo de texto, pero no está hecho todo, necesitamos que esa función se llame constantemente y cada segundo para que actualice el valor de nuestro campo de texto constantemente también.

Para ello, lo mejor es que la función se encargue también de llamarse a si misma al cabo de un segundo, así volverá a hacer todo el proceso de obtención y actualización de la hora y por último se llamará a si misma al cabo de un segundo. Este proceso se repetirá por siempre hasta que nos salgamos de la página.

La línea de código para llamarse a si misma, que colocaremos en la última línea de la función es la siguiente:

```
setTimeout("mueveReloj()",1000)
```

La función `setTimeout` sirve para hacer el retardo antes de ejecutar la sentencia. La sentencia es una simple llamada a la función y el retardo es de 1000 milisegundos (un segundo).

### Poner en marcha el reloj

Finalmente necesitamos poner el reloj en marcha. Esto lo podemos hacer una vez acabada la carga de la página con el manejador de eventos `onload`, que se coloca en el `<body>`.

```
<body onload="mueveReloj()">
```

Esto quiere decir que cuando se termine de cargar la página se llame a la función `mueveReloj()`, que se encargará de mover el reloj y llamarse a si misma para hacer el proceso de manera continuada.

### Código entero

El código de la página queda de esta manera:

```
<html>
<head>
<title>Reloj con Javascript</title>
<script language="JavaScript">
function mueveReloj(){
    momentoActual = new Date()
```

```
hora = momentoActual.getHours()
minuto = momentoActual.getMinutes()
segundo = momentoActual.getSeconds()

horaImprimible = hora + " : " + minuto + " : " + segundo

document.form_reloj.reloj.value = horaImprimible

setTimeout("mueveReloj()",1000)
}
</script>
</head>

<body onload="mueveReloj()">
```

Vemos aquí el reloj funcionando...

```
<form name="form_reloj">
<input type="text" name="reloj" size="10">
</form>

</body>
</html>
```

## Añadidos al reloj

Podemos hacer muchas cosas para mejorar aspectos de este reloj. Por ejemplo darle un poco de estilo al campo de texto o hacer que nadie se pueda posar encima del campo de texto para actualizar manualmente el valor de la hora. Vamos a ver alguna cosilla.

### Estilo al reloj

Con [hojas de estilo](#) podemos cambiar la apariencia del reloj para hacerlo un poco más especial. Este es un ejemplo de lo que se podría poner.

```
<input type="text" name="reloj" size="10" style="background-color : Black; color : White; font-family : Verdana,
Arial, Helvetica; font-size : 8pt; text-align : center;">
```

### Evitar que accedan al campo de texto

Así nadie podrá modificar el texto del reloj manualmente. Lo conseguimos con Javascript. El manejador de eventos onfocus se activa cuando el campo de texto adquiere el foco de la aplicación. En ese momento nosotros sacaremos el foco de la aplicación con el método blur(). El botón quedaría así:

```
<input type="text" name="reloj" size="10" onfocus="window.document.form_reloj.reloj.blur()">
```

### Hacer que siempre tengamos dos dígitos en la hora minutos y segundos.

Para conseguir que la hora tenga siempre un formato hh : mm : ss debemos jugar un poco con los valores de tiempo como si fueran strings. Para ello lo primero que tendremos que hacer es construir un string a partir del valor (hora, minuto o segundo) que queramos formatear. Luego observaremos ese string para saber si tenemos que añadirle un dígito al valor.

Veamos cómo obtenemos el string a partir del número de segundos obtenido. Lo hacemos para los segundos, la hora y los minutos serán realizados de manera similar.

```
str_segundo = new String (segundo)
```

Seguidamente observamos si tenemos un solo carácter en el string, porque si es así le tenemos que concatenar un cero ("0") al principio.

```
if (str_segundo.length == 1)
segundo = "0" + segundo
```

## Todo junto

Veamos otra vez nuestro ejemplo en una sola pieza para ver cómo quedan todas estas mejoras.

```
<html>
<head>
<title>Reloj con Javascript</title>
<script language="JavaScript">
function mueveReloj(){
    momentoActual = new Date()
    hora = momentoActual.getHours()
    minuto = momentoActual.getMinutes()
    segundo = momentoActual.getSeconds()

    str_segundo = new String (segundo)
    if (str_segundo.length == 1)
        segundo = "0" + segundo

    str_minuto = new String (minuto)
    if (str_minuto.length == 1)
        minuto = "0" + minuto

    str_hora = new String (hora)
    if (str_hora.length == 1)
        hora = "0" + hora

    horaImprimible = hora + " : " + minuto + " : " + segundo

    document.form_reloj.reloj.value = horaImprimible

    setTimeout("mueveReloj()",1000)
}
</script>
</head>

<body onload="mueveReloj()">
```

Vemos aquí el reloj funcionando...

```
<form name="form_reloj">
<input type="text" name="reloj" size="10" style="background-color : Black; color : White; font-family : Verdana,
Arial, Helvetica; font-size : 8pt; text-align : center;" onfocus="window.document.form_reloj.reloj.blur()">
</form>

</body>
</html>
```

Este segundo reloj lo podemos ver a continuación.

Si lo deseamos podemos abrir una página web para visualizar:

- [El reloj sencillo](#)
- [El reloj mejorado](#)

Artículo por **Miguel Angel Alvarez**

## **Scripts distintos para cada navegador**

Uno de los mayores problemas con los que se encuentra un programador a la hora de crear páginas con DHTML es los diferentes modelos de objetos que tienen cada uno de los navegadores, y cuyas diferencias no solo se dan con los navegadores de las distintas compañías, sino que además entre la misma compañía hay diferencias en la modelización de los objetos dependiendo de las versiones. Por ejemplo el modelo de objetos de Netscape 6 es distinto del modelo de objetos de Netscape 4.

Todo esto nos crea el problema a la hora de escribir scripts de tener que duplicar el código de cada una de nuestras funciones dependiendo del navegador que se utiliza. Debido a la gran variedad de navegadores y versiones disponibles, esto hace que nuestras funciones acaben siendo mucho más grandes e ilegibles de lo esperado.

Una solución a este problema es la creación de diferentes ficheros que contienen los scripts para cada uno de los navegadores que existen, enlazando en el momento de la carga de la página con el fichero de scripts del navegador y versión que estamos utilizando. De esta forma, los scripts son mucho más simples ya que un script se creará enfocado a una sola versión, y además en el caso de que salga al mercado un nuevo navegador con un modelo de objetos distinto, no tendremos que cambiar todos nuestros scripts, sino que nos bastará con modificar el script que detecta el navegador, y a continuación escribir un nuevo fichero que contiene los mismos scripts que los anteriores pero con el nuevo modelo de objetos.

El siguiente código pretende ser un ejemplo reducido de lo anteriormente expuesto.

Nuestro objetivo va a ser diferenciar entre Netscape e Internet Explorer, de forma que al hacer click sobre un único botón, aparezca un mensaje de alerta distinto dependiendo del navegador que utilicemos.

El primer paso es la creación de los distintos ficheros que van a contener los scripts específicos para cada uno de los navegadores. En nuestro caso van a ser dos ficheros, uno que llamaremos Internet.js que contendrá los scripts para Internet Explorer, y otro llamado Netscape.js que contendrá los scripts para Netscape. El contenido de Internet.js será el siguiente:

```
function sacaAlerta(){  
    alert("Estoy utilizando IE")  
}
```

El contenido de Netscape.js será el siguiente:

```
function sacaAlerta(){  
    alert("Estoy utilizando NS")  
}
```

Por otra parte, en la cabecera de cada una de las páginas, deberemos incluir un script que enlace con cada uno de los navegadores, dependiendo del que utiliza el usuario.

```
<script language="javascript">

ns=(document.layers)? true:false
ie=(document.all)? true:false

ponNs="<script language='javascript1.2' src='netscape.js'> <\script>"
ponIe="<script language='javascript1.2' src='internet.js'> <\script>"

if (ns) {document.write(ponNs)}
if (ie) {document.write(ponIe)}

</script>
```

**\* Fijaros en la contrabarra "\" colocada antes de la barra "/" en </script> en los document.write().**

Por ultimo inclimos en el cuerpo de la página un boton que llama a la funcion que esta definida dentro de los ficheros de scripts:

```
<input type="button" value="Haz Click" onclick="sacaAlerta()">
```

De forma que la página queda con el siguiente aspecto:

```
<html>
<head>
  <title>Untitled</title>
  <script language="javascript">

ns=(document.layers)? true:false
ie=(document.all)? true:false

ponNs="<script language='javascript1.2' src='netscape.js'> <\script>"
ponIe="<script language='javascript1.2' src='internet.js'> <\script>"

if (ns) {document.write(ponNs)}
if (ie) {document.write(ponIe)}

</script>

</head>

<body>

<form>
<input type="button" value="Haz Click" onclick="sacaAlerta()">
</form>

</body>
</html>
```

*Artículo por **Carlos Luis Cuenca***

## Tamaño de los campos relativo al navegador

Si creamos un formulario con unos campos de texto en una página web podremos observar que Internet Explorer no coloca el mismo tamaño a los campos de texto que Netscape. En concreto, los campos que coloca Netscape son de mayor tamaño para un mismo atributo size en el campo.

Generalmente se soluciona poniendo los campos de texto más pequeños para que quepan en el diseño en Netscape, pero en ocasiones queremos que los campos sean del tamaño justo y que en Internet Explorer no sean demasiado pequeños y que en Netscape demasiado grandes. Esto puede ocurrir, por ejemplo, en diseños calculados muy a la medida.

Concretando, el tamaño de los campos se aplica con el atributo size de la etiqueta <INPUT> y queremos hacer que el size indicado dependa del navegador que estamos usando.

### Distinguir los navegadores

Para ello deberemos crear un script de Javascript que distinga entre los dos navegadores y escribir mediante javascript el atributo size con un valor distinto para cada caso.

Distinguir dinámicamente entre los navegadores es una tarea que puede ser bastante complicada y digna de otro reportaje, pero podríamos hacerlo de una manera parecida a esta.

```
if (document.layers)
//hago cosas para Netscape 4
else
// hago cosas para otros navegadores
```

Con este código discrimino tan solo entre la versión 4 de Netscape y todos los demás navegadores. Es por que Netscape 4 es el único navegador cuya jerarquía de objetos incluye un objeto denominado layers. Al introducir ese nombre de objeto en la expresión a evaluar del if, obtendremos casos positivos si el objeto existe (Netscape4) y negativos si no existe (los demás).

### Escribir cosas distintas en cada caso

Ahora vamos a poner el código Javascript que escribiría un campo de texto con un valor de size distinto para cada navegador.

```
if (document.layers)
document.write("<INPUT type=text size=12>")
else
document.write("<INPUT type=text size=16>")
```

### Colocamos el script en un formulario

Es bien simple, sólo nos queda colocar ese código dentro de un formulario y tendremos un campo de texto con sizes distintos pero con el mismo tamaño en la pantalla.

```
<FORM>
<script language="JavaScript">
if (document.layers)
```

```
document.write("<INPUT type=text size=12>")
else
document.write("<INPUT type=text size=16>")
</script>
</FORM>
```

Se puede [ver este ejemplo en una página a parte](#).

*Artículo por Miguel Angel Alvarez*

## Estilos distintos para cada navegador

Gracias a una consulta en nuestra [lista de correo](#) de una persona que quería colocar un fondo de pantalla distinto dependiendo del tipo de navegador se nos han ocurrido un par de scripts que pueden servir para el [taller de Javascript](#).

Se trata de unos scripts que permiten mostrar estilos en la página web dependiendo del navegador con el que se accede a la página, de modo que si un usuario entra con Internet Explorer verá un fondo, tipografías y otros elementos con formas o estilos distintos de los que vería otro usuario que ha accedido con Netscape.

**Nota:** Este artículo hace una distinción de navegadores entre Netscape 4 y otros, para colocar una hoja de estilos distinta en cada caso. El artículo en sí está bien, pero la detección del navegador está poco actualizada, debido a que ahora hay muchos otros navegadores y nuevas versiones que no se tenían en cuenta.

Hemos publicado otro artículo para [enlazar una hoja de estilos para cada navegador detectando navegadores más modernos](#), que será de interesante lectura para los interesados.

Empezamos por el ejemplo exacto que nos pedía el suscriptor de la lista de correo, para poner tan sólo fondos distintos. Es un script bastante específico por lo que muchos de vosotros lo encontraréis inservible, pero por lo menos servirá para aprender alguna cosilla.

### Con la etiqueta <BODY>

La primera idea que se me ha ocurrido es escribir la etiqueta <BODY> entera dentro de un bloque de script donde tenemos un if que nos permite distinguir entre navegadores.

El código queda algo así:

```
<html>
<head>
  <title>Fondos distintos para cada navegador</title>
</head>

<script language="JavaScript">
if (document.layers)
  document.write("<body background=nts.jpg>")
else
  document.write("<body background=ie.jpg>")
</script>

</body>
</html>
```



Vemos que para averiguar si tenemos Netscape o Internet Explorer accedemos al objeto layers del documento. Como sólo Netscape tiene ese objeto, sólo para Netscape esa evaluación será verdadera. Esto es un truco rápido, aunque se debería estudiar a parte porque no funciona bien con todos los navegadores, por ejemplo, Netscape 6 mostraría el fondo de Explorer. Más bien, solo distingue entre Netscape 4 y todos los demás navegadores, ya que ciertamente, ese objeto sólo está disponible en ese navegador.

Pero a lo que íbamos, en el ejemplo, dependiendo si se utiliza Netscape 4 u otro navegador se muestra una etiqueta <BODY> con un atributo, el de la imagen de fondo, distinto.

Este ejemplo se puede [ver en funcionamiento](#), pero recuerda que debes entrar con Netscape 4 y otros navegadores distintos para que se muestren los efectos.

## Con estilos

Hay otra manera de asignar fondos distintos dependiendo del navegador y no sólo los fondos sino también tipos de letra, tamaños y en general todo lo que se puede definir utilizando los estilos CSS.

En este ejemplo suponemos que sabes algo de CSS y la definición de estilos para todo un sitio incluida en un archivo externo con sintaxis CSS. Para conocer las CSS tenemos un excelente manual en DesarrolloWeb.com.

La base del ejemplo es la misma, utilizamos un bloque script en el que averiguamos qué navegador se está utilizando y dependiendo de el navegador se muestra una etiqueta con unos atributos u otros. En concreto la etiqueta que colocamos dinámicamente en función del navegador es <LINK> que sirve para incluir una declaración de estilos externa.

Sería algo así:

```
<html>
<head>
  <title>Enlazo con estilos dinamicamente</title>
  <script LANGUAGE="JavaScript">
if (document.layers) {
  document.write("<LINK REL='stylesheet' HREF='estilo_nt.css' TYPE='text/css'>"); }
else {
  document.write("<LINK REL='stylesheet' HREF='estilo_ie.css' TYPE='text/css'>"); }
  </script>
</head>

<body>

<h1>Bienvenidos a mi página</h1>

</body>
</html>
```

Como vemos, en caso de estar en Netscape 4 se cargará la hoja de estilos llamada "estilo\_nt.css" y si nuestro navegador es otro cualquiera se cargará "estilo\_ie.css".

Este segundo caso se puede utilizar para muchas cosas. Principalmente se puede utilizar para evitar un efecto que hay en el uso de distintos navegadores, que consiste en que a misma definición de fuentes los tamaños de las mismas en Netscape e Internet Explorer difieren. (Las mismas fuentes, en Netscape salen más pequeñas que en Explorer). Es por ello que incluyendo una hoja de estilos distinta se puede poner los tamaños de las fuentes deseados para cada

navegador.

Se puede ver [este script en marcha](#) para comprobar el efecto producido por los estilos distintos dependiendo del navegador.

Eso es todo. Esperemos que os den ideas estos scripts para resolver vuestros problemas.

Sólo comentar de nuevo que hemos publicado una actualización del artículo para tratar de [detectar más tipos de navegadores y en distintas versiones y asignarles una hoja de estilos css distinta](#).

*Artículo por **Miguel Angel Alvarez***

## Tabla de colores con Javascript

En HTML construimos cualquier color mezclando el rojo, verde y azul (RGB) en las proporciones que deseamos. Esto es un hecho que deberíais saber antes de leer este artículo. Se explica con detalle la construcción de colores en el artículo [Los colores y HTML](#). Será necesario que, el que no esté familiarizado con este asunto, se lea el artículo.

Además de cómo construir colores, el artículo [Los colores y HTML](#) muestra también cuáles son los colores puros, que se ven sin problemas en todas las profundidades de color que pueda tener la configuración de los ordenadores de los visitantes. Para la construcción de colores puros mezclamos los colores RGB siempre en estas cantidades 00, 33, 66, 99, CC, FF. Nuevamente, quien no conozca esto debería [leerse el reportaje señalado anteriormente](#).

El ejemplo que pretendemos construir tiene que ver con los colores puros en todas las definiciones. Se trata de construir una tabla en una página web que contenga todos los colores puros, además del código RGB de cada color. Esta tabla puede servir para seleccionar un color que pretendemos utilizar en una página web. Si nos limitamos solamente a utilizar los colores de la tabla tendremos la seguridad que nuestra paleta será respetada en todos los casos.

Para generar todos los colores puros vamos a utilizar tres arrays Javascript con los posibles valores para cada uno de los colores RGB. Por tanto, tendremos tres arrays como los que se puede ver a continuación:

```
var r = new Array("00","33","66","99","CC","FF");  
var g = new Array("00","33","66","99","CC","FF");  
var b = new Array("00","33","66","99","CC","FF");
```

Para escribir la tabla en la página web vamos a hacer un recorrido a estos tres arrays. Para ello vamos a utilizar bucles anidados, que son bucles dentro de otros bucles.

Vamos a tratar de explicar porqué necesitamos los bucles anidados; Si hacemos la cuenta de los colores que debemos ir generando obtendremos una lista como la que sigue:

```
#000000 #000033 #000066 #000099 #0000CC #0000FF #003300 #003333 #003366  
#003399 #0033CC #0033FF #006600 #006633 #006666 #006699 #0066CC #0066FF...
```

Se puede [ver la cuenta completa aquí](#).

Pues vemos que al principio los tres valores de RGB valen 00 y cómo en sucesivas repeticiones se va aumentando el valor de B (El valor asignado al azul) hasta que llegamos a FF. Para continuar se aumenta el valor de G y se vuelve a realizar la cuenta con B. Es como si contásemos y las unidades fueran los valores de RGB.

El caso es que realizamos la cuenta con el valor B, cuando llegamos a FF aumentamos el valor de G y cuando lleguemos a FF en G aumentaríamos en un valor R. Así se puede ver una estructura en pseudocódigo como esta.

```
Contamos desde 00 hasta FF con el valor R{
  Contamos desde 00 hasta FF con el valor G{
    Contamos desde 00 hasta FF con el valor R{
      Imprimimos el valor actual de RGB
    }
  }
}
```

Esta estructura imprime todos los colores puros, y ya es cercana a nuestra solución, aunque todavía no está escrita en Javascript y falta colocar todas las etiquetas HTML que necesitamos para mostrar una tabla en una página web.

Como no importa ir un poco más despacio con tal de que todo el mundo entienda el problema Vamos a escribir en Javascript este bucle para que simplemente liste los colores puros, sin introducirlos todavía en una tabla. Será interesante para ver un poco mejor el funcionamiento de bucles anidados.

```
//creamos los arrays
var r = new Array("00","33","66","99","CC","FF");
var g = new Array("00","33","66","99","CC","FF");
var b = new Array("00","33","66","99","CC","FF");

//hacemos el bucle anidado
for (i=0;i<r.length;i++) {
  for (j=0;j<g.length;j++) {
    for (k=0;k<b.length;k++) {
      //creamos el color
      var nuevoc = "#" + r[i] + g[j] + b[k];
      //imprimimos el color
      document.write(nuevoc + "<br>");
    }
  }
}
```

Los bucles para recorrer un array se crean con un índice que servirá para acceder a la posición actual del array. Los índices en arrays empiezan en 0, es por ello que nuestros bucles for contienen una inicialización a 0 de la variable que va a servir de índice. Además el índice debe crecer de uno en uno hasta que llegue a la última posición del array, que se obtiene accediendo a su propiedad length (que guarda la longitud o número de elementos del array).

Poniendo un bucle dentro de otro podremos realizar esa cuenta. El bucle más exterior será el que menos veces se ejecute, así que con el bucle exterior llevaremos la cuenta de R. El bucle del medio será para llevar la cuenta de G y el más interior (el que más veces se repetirá) para llevar la cuenta de B, que es el valor que va cambiando constantemente.

Podemos [verla en funcionamiento en este enlace](#).

## La tabla de los colores completa

Para acabar vamos a ver el ejemplo completo, con todas las líneas de código que incluyen los textos HTML necesarios para que la tabla salga convenientemente formateada y con los colores de fondo en cada una de las celdas iguales al color actual.

Para ello lo primero que hacemos es escribir una cabecera de la tabla y finalización de la misma, que quedan fuera de la estructura de bucles. Dentro de los bucles realizaremos las sentencias para imprimir cada una de las filas y celdas de la tabla.

Nuestra tabla va a tener en cada fila un conjunto de colores donde los valores RG no cambian y el valor B varía entre todos los posibles. En la siguiente fila se incrementaría en uno el valor G y se haría otra vez la cuenta de valores B... así hasta que terminamos con todos los valores R, G y B posibles.

El código es el siguiente:

```
<table width="80%">
<script language="javascript">
var r = new Array("00","33","66","99","CC","FF");
var g = new Array("00","33","66","99","CC","FF");
var b = new Array("00","33","66","99","CC","FF");

for (i=0;i<r.length;i++){
  for (j=0;j<g.length;j++) {
    document.write("<tr>");
    for (k=0;k<b.length;k++) {
      var nuevoc = "#" + r[i] + g[j] + b[k];
      document.write("<td bgcolor=\"\" + nuevoc + \"\" align=center>");
      document.write(nuevoc);
      document.write("</td>");
    }
    document.write("</tr>");
  }
}
</script>
</table>
```

Vemos que antes de empezar el bucle más interior se crea una nueva celda con `<TR>` y que cuando se acaba se termina también la celda con `</TR>`. Además, dentro del bucle más interior se compone primero el color actual y luego se escribe una celda con el atributo `bgcolor` asignado a ese color actual y dentro de ella el texto del color actual.

La tabla que nos genera este script se puede ver aquí:

#000000	#000033	#000066	#000099	#0000CC	#0000FF
#003300	#003333	#003366	#003399	#0033CC	#0033FF
#006600	#006633	#006666	#006699	#0066CC	#0066FF
#009900	#009933	#009966	#009999	#0099CC	#0099FF
#00CC00	#00CC33	#00CC66	#00CC99	#00CCCC	#00CCFF
#00FF00	#00FF33	#00FF66	#00FF99	#00FFCC	#00FFFF
#330000	#330033	#330066	#330099	#3300CC	#3300FF
#333300	#333333	#333366	#333399	#3333CC	#3333FF
#336600	#336633	#336666	#336699	#3366CC	#3366FF
#339900	#339933	#339966	#339999	#3399CC	#3399FF

#33CC00	#33CC33	#33CC66	#33CC99	#33CCCC	#33CCFF
#33FF00	#33FF33	#33FF66	#33FF99	#33FFCC	#33FFFF
#660000	#660033	#660066	#660099	#6600CC	#6600FF
#663300	#663333	#663366	#663399	#6633CC	#6633FF
#666600	#666633	#666666	#666699	#6666CC	#6666FF
#669900	#669933	#669966	#669999	#6699CC	#6699FF
#66CC00	#66CC33	#66CC66	#66CC99	#66CCCC	#66CCFF
#66FF00	#66FF33	#66FF66	#66FF99	#66FFCC	#66FFFF
#990000	#990033	#990066	#990099	#9900CC	#9900FF
#993300	#993333	#993366	#993399	#9933CC	#9933FF
#996600	#996633	#996666	#996699	#9966CC	#9966FF
#999900	#999933	#999966	#999999	#9999CC	#9999FF
#99CC00	#99CC33	#99CC66	#99CC99	#99CCCC	#99CCFF
#99FF00	#99FF33	#99FF66	#99FF99	#99FFCC	#99FFFF
#CC0000	#CC0033	#CC0066	#CC0099	#CC00CC	#CC00FF
#CC3300	#CC3333	#CC3366	#CC3399	#CC33CC	#CC33FF
#CC6600	#CC6633	#CC6666	#CC6699	#CC66CC	#CC66FF
#CC9900	#CC9933	#CC9966	#CC9999	#CC99CC	#CC99FF
#CCCC00	#CCCC33	#CCCC66	#CCCC99	#CCCCCC	#CCCCFF
#CCFF00	#CCFF33	#CCFF66	#CCFF99	#CCFFCC	#CCFFFF
#FF0000	#FF0033	#FF0066	#FF0099	#FF00CC	#FF00FF
#FF3300	#FF3333	#FF3366	#FF3399	#FF33CC	#FF33FF
#FF6600	#FF6633	#FF6666	#FF6699	#FF66CC	#FF66FF
#FF9900	#FF9933	#FF9966	#FF9999	#FF99CC	#FF99FF
#FFCC00	#FFCC33	#FFCC66	#FFCC99	#FFCCCC	#FFCCFF
#FFFF00	#FFFF33	#FFFF66	#FFFF99	#FFFFCC	#FFFFFF

Podemos ver una [página web donde sólomente está esta tabla](#).

Artículo por **Miguel Angel Alvarez**

## Submenú en otra ventana

Vamos a realizar un pequeño ejemplo sobre cómo trabajar con ventanas secundarias o popups. Las ventanas secundarias son esas ventanitas que se abren a parte de la ventana principal del navegador y por lo general molestan un poco en determinados sitios gratuitos.

Para abrir esas ventanas se utiliza el lenguaje Javascript, más concretamente, el método `open()` del objeto `window`. No lo vamos a explicar aquí, porque ya tenemos un reportaje entero que explica detenidamente la apertura de ventanas secundarias y las características con las que las podemos abrir: [Abrir ventanas secundarias](#)

En este ejemplo vamos a ir un poco más allá, vamos a crear una ventana secundaria y desde ella vamos a acceder a las propiedades de la ventana padre, es decir, la ventana que la abrió. El ejercicio será el siguiente:

Tendremos una página con fondo blanco, un campo de texto vacío y un botón. Cuando se pulse el botón abriremos un popup que contendrá una tabla con los colores puros de HTML. El visitante podrá seleccionar uno de esos colores y entonces el fondo de la página padre cambiará a ese color y el color se escribirá en el campo de texto. Es mucho más fácil [ver el ejemplo en marcha](#) para comprender la explicación.

## Página padre

La página original contendrá un simple formulario con un botón y un campo de texto. Además, contendrá el script Javascript necesario para abrir la ventana secundaria.

```
<html>
<head>
  <title>Submenú en ventana a parte</title>
  <script language="JavaScript">
function lanzarSubmenu(){
  window.open("submenu_ventana2.html","ventana1","width=400,height=400,scrollbars=YES")
}
</script>
</head>
<body bgcolor="#ffffff">
<form>
<input type="text" name="colorin" size="12" maxlength="12">
<br>
<br>
<input type="button" value="Lanzar submenu" onclick="lanzarSubmenu()">
</form>
</body>
</html>
```

La función lanzarSubmenu() es la que contiene el script para abrir el popup. Para ello utiliza el método open() del objeto window, cuyo uso se describió en el artículo de [Abrir ventanas secundarias](#).

El formulario es de lo más normal. Lo único destacable es el atributo onclick del botón, que sirve para definir las acciones a ejecutar cuando se pulsa el botón, en este caso una llamada a la función que abre la ventana secundaria.

## Página secundaria

La página secundaria es un poco más compleja, pero la parte que nos importa a nosotros en este ejercicio es muy sencilla. Lo importante de la página es que tiene que acceder a la ventana padre para modificar su estado y para ello utiliza un objeto Javascript: opener.

El objeto opener está disponible en las páginas que son ventanas secundarias y hace referencia a la ventana que la abrió, es decir, la ventana padre. Dicho de otro modo, el objeto opener en la ventana popup es un sinónimo del objeto window en la ventana padre.

El script que accede a la ventana padre es el siguiente:

```
<script language="JavaScript">
function actualizaPadre(miColor){
```

```
window.opener.document.bgColor = miColor
window.opener.document.forms[0].colorin.value = miColor
}
</script>
```

La función `actualizaPadre()` es la encargada de realizar el trabajo. Recibe el código del color sobre el que se ha pulsado. En la primera línea cambiamos el color de la página web padre y en la segunda línea colocamos el código RGB del color recibido por parámetro en el campo de texto.

Como vemos, el objeto `opener` también depende del objeto `window` de la página, como todos los demás objetos de la jerarquía Javascript.

El resto de la página es parecido a un taller anterior de Javascript en el que [escribíamos con Javascript una tabla de colores puros](#), con unas pequeñas modificaciones para adaptarla a nuestras necesidades. Se puede ver a continuación.

```
<table width="80%" align="center" cellpadding="1" cellspacing="1">
<script language="javascript">
var r = new Array("00","33","66","99","CC","FF");
var g = new Array("00","33","66","99","CC","FF");
var b = new Array("00","33","66","99","CC","FF");
for (i=0;i<r.length;i++)
  for (j=0;j<g.length;j++) {
    document.write("<tr>");
    for (k=0;k<b.length;k++) {
      var nuevoc = "#" + r[i] + g[j] + b[k];
      document.write("<td bgcolor=\"\" + nuevoc + \"\" align=center><font size=1 face=verdana>");
      document.write("<a href='\"javascript:actualizaPadre(\" + nuevoc + \"')\">");
      document.write(nuevoc);
    }
    document.write("</a></font></tr>");
  }
}
</script>
</table>
```

Vista la complicación de este script y dado que no vamos a explicarlo todo otra vez, puede ser altamente recomendable la lectura del artículo [Tabla de colores con Javascript](#).

Lo importante para nosotros ahora es entender que este script crea una tabla con todos los colores puros, colocados en una celda cada uno. Dentro de cada celda se escribe un enlace que llama a la función `actualizaPadre()` pasándole el código del color se ha de utilizar.

Podemos [ver el ejemplo en funcionamiento](#).

*Artículo por **Miguel Angel Alvarez***

## **Degradado de color Javascript**

Vamos a ver un ejemplo típico de efecto especial en una página web: un degradado de colores para el fondo de la página.

El degradado consiste en cambiar el color de fondo de la página poco a poco para ir acercándolo hacia otro color . Podemos hacer muchos degradados distintos para páginas web

con poco esfuerzo, pocos conocimientos técnicos y bastante de imaginación.

Se puede [ver un ejemplo de degradado en una página a parte](#), para hacerse una idea exacta de lo que vamos a explicar en este taller. Además, al final del artículo, veremos un script que permite hacer una amplia gama de degradados, como este [ejemplo de degradado más complejo](#).

## Complicación del degradado

La mayor complicación que encontramos a la hora de crear un script para hacer un degradado es que los colores se expresan en hexadecimal y en Javascript trabajamos con números decimales. De modo que, para convertir nuestros números decimales en hexadecimales y poder así utilizarlos para indicar colores hemos creado una función especial:

```
function convierteHexadecimal(num){
    var hexaDec = Math.floor(num/16)
    var hexaUni = num - (hexaDec * 16)
    return hexadecimal[hexaDec] + hexadecimal[hexaUni]
}
```

La función devuelve el número pasado por parámetro en forma hexadecimal. Por ejemplo, si recibe 140 devuelve 8C. Si recibe 15 devuelve 0F.

Para ello se ayuda de un Array creado anteriormente que guarda conversión de números decimales a hexadecimales de las unidades básicas, es decir, del 0 a la letra F. Esta es la línea que crea el Array.

```
hexadecimal = new Array("0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F")
```

La función primero calcula los grupos de 16 que se pueden hacer a partir del número que recibimos. El número de estos grupos serían las "hexa-decenas".

Luego calculamos el resto de la división anterior, o dicho de otra forma, cuantas unidades nos restan después de tomar estas "hexa-decenas". Éste segundo número serán las unidades. Al juntar las unidades con los grupos de 16 que hemos obtenido, convirtiendo ambos valores por medio del array de conversión decimal a hexadecimal, conseguimos la conversión buscada.

## Creando la secuencia de colores para el degradado

Una vez podemos convertir del valor decimal (necesario para llevar la cuenta en Javascript) al hexadecimal (necesario para indicar un color) nuestra única tarea será contar en decimal los colores y convertirlos a hexadecimal antes de cambiar la propiedad document.bgColor, que como sabemos es la propiedad que guarda el color de fondo de la página.

Hemos querido hacer un primer ejemplo muy simple para que se pueda entender más fácilmente. El ejemplo lo hemos podido ver anteriormente en funcionamiento. Se trata de una escala de grises que empieza en negro y termina en blanco.

```
color_decimal = 0
```

Primero inicializamos la variable color\_decimal, que es la que guarda el color actual a mostrar, en formato decimal. Sólo guardamos un valor del color pues, al ser una escala de grises, todos los valores RGB son el mismo.



Tendremos una función llamada degradado que será la encargada de realizar la mayor parte del trabajo.

```
function degradado(){
    color_decimal ++
    color_hexadecimal = convierteHexadecimal(color_decimal)
    document.bgColor = color_hexadecimal + color_hexadecimal + color_hexadecimal

    //la llamo con un retardo
    if (color_decimal < 255)
        setTimeout("degradado()",1)
}
```

La función se encarga de incrementar el color\_decimal en una unidad, convertirlo en hexadecimal y colocarlo dentro de propiedad document.bgColor para actualizar el fondo de la página.

Finalmente, y en esto se basan muchos efectos especiales de Javascript, se llama la función a si misma con un retardo. En el ejemplo podemos ver que si el color\_decimal es menor de 255 (que es el máximo que se puede alcanzar en colores) se llama a la función con setTimeout, que es la que nos crea el retardo.

El código de este ejemplo simple se puede ver entero a continuación:

```
<html>
<head>
    <title>Ejemplo de degradado 1</title>
</head>
```

```
<body bgcolor=000000>
```

```
<h1 align="center">Degradando...</h1>
<h2>Ejemplo 1</h2>
```

En esta página podemos ver un ejemplo de degradado de negro a blanco, con una sola pasada. Solo se ve el texto cuando el fondo es suficientemente blanco para que contraste.

```
<script language="javascript">
```

```
hexadecimal = new Array("0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F")
```

```
function convierteHexadecimal(num){
    var hexaDec = Math.floor(num/16)
    var hexaUni = num - (hexaDec * 16)
    return hexadecimal[hexaDec] + hexadecimal[hexaUni]
}
```

```
color_decimal = 0
```

```
function degradado(){
    color_decimal ++
    color_hexadecimal = convierteHexadecimal(color_decimal)
    document.bgColor = color_hexadecimal + color_hexadecimal + color_hexadecimal

    //la llamo con un retardo
    if (color_decimal < 255)
        setTimeout("degradado()",1)
}
```

```
degradado()
</script>
</body>
</html>
```

Si queremos, también se puede [ver el ejemplo en funcionamiento en una página a parte](#).

## Más degradados

Probablemente este no sea el ejemplo más útil para el lector que desea implementar un degradado en su página, pues resulta un poco simple y específico. Para solucionar este asunto hemos creado un sistema donde se puede configurar el tipo de degradado de la página con una serie de variables. Es un ejemplo que se basa en el que acabamos de ver, aunque tiene muchas variables para parametrizar el comportamiento del degradado y que se adapte a las necesidades del desarrollador.

Por falta de tiempo no vamos a explicar todo el script, tan sólo vamos a fijarnos en las variables que permiten configurarlo.

```
color_inicio = new Array(150,150,255)
```

Con `color_inicio` configuramos el color que se muestra al principio en la página. Lo indicamos con un array, donde en cada casilla colocamos el valor decimal de cada uno de los tres colores RGB.

```
color_fin = new Array(255,99,0)
```

Con `color_fin` configuramos el color al que va a tender nuestro degradado, de manera idéntica a como lo hacíamos en el `color_inicio`

```
pasos = 100
```

Es el número de pasos que vamos a utilizar en alcanzar el valor del color final, desde el color de inicio.

```
comportamiento = 1
```

Esta variable sirve para definir el comportamiento del script en cuatro posibles valores:

1. Realiza un bucle infinito desde el color de inicio al color fin y del color fin al color de inicio, para volver a empezar. Se repite por siempre.
2. Realiza una pasada desde el color inicio al color fin. Termina cuando acaba la pasada.
3. Realiza una pasada del color de inicio al de fin y una vuelta desde el fin al inicio. Termina cuando realiza la ida y la vuelta.
4. Un bucle infinito con una parada de 10 segundos entre cuando ha hecho la ida y la vuelta, antes de volver ha empezar otra ida-vuelta.

## Ejemplos de degradado completo

- [Degradado infinito de morado a naranja.](#)
- [Una pasada de azul a rojo.](#)
- [De negro a amarillo y de amarillo a negro. Una pasada..](#)
- [Bucle infinito con parada. De negro a blanco..](#)

Podéis descargar el ejemplo de degradado completo aquí: [degradado.zip](#).

*Artículo por **Miguel Angel Alvarez***

## Validar entero en campo de formulario

Supongamos que tenemos un campo de un formulario donde queremos que figure siempre un valor numérico entero. Un ejemplo podría ser un campo donde queremos guardar el número de un año, donde, lógicamente, no caben decimales ni tampoco letras.

En este ejercicio vamos a realizar un script que procure obtener un número entero del valor que haya escrito el usuario en un campo de texto. Si es un entero o puede convertirlo a un entero, coloca dicho valor entero en el campo. Si no puede obtener un valor numérico entero borra el contenido del campo y lo deja vacío. Lo haremos con Javascript, ya que es el lenguaje del lado del cliente –para operar en el ámbito del navegador– más extendido.

Para aclarar el funcionamiento del ejercicio podemos [ver el ejemplo completo en una página a parte](#).

Este ejercicio sirve también para aprender a manejar las funciones incorporadas de Javascript `parseInt()` e `isNaN()`. La primera sirve para convertir un valor a número entero y la segunda para ver si un dato es un valor numérico. Las dos se pueden conocer en profundidad en el [los primeros capítulos del manual de Javascript II](#).

Otro tema importante que hay que conocer es la jerarquía de objetos del navegador, pero en este caso haremos un esfuerzo en explicarla para aquellas personas que no la conozcan.

### Función `validarEntero()`

Primero vamos a realizar una función que hará la mayor parte del trabajo, puesto que es la encargada de validar si un dato es un número entero. Esta función recibe un valor, que es el dato que deseamos validar y si es un entero lo devolverá tal cual. Si no lo es, lo intentará convertir a entero y si lo consigue devolverá ese valor. Finalmente, si el intento de convertirlo no da resultado, devolverá una cadena vacía.

```
function validarEntero(valor){
    //intento convertir a entero.
    //si era un entero no le afecta, si no lo era lo intenta convertir
    valor = parseInt(valor)

    //Compruebo si es un valor numérico
    if (isNaN(valor)) {
        //entonces (no es numero) devuelvo el valor cadena vacia
        return ""
    }else{
        //En caso contrario (Si era un número) devuelvo el valor
        return valor
    }
}
```

### Formulario

Vemos el formulario que necesitaremos para colocar el campo de texto. Es un formulario como otro cualquiera, el único detalle es que nos hemos preocupado por darle nombre tanto al formulario en si como al campo de texto. Posteriormente utilizaremos esos nombres para referirnos a los elementos mediante Javascript.

También tenemos un campo de formulario de tipo botón, que sirve en este caso para indicar

que cuando se pulse se llamará a la función `validarFormulario()`. Para indicar esto se utiliza el atributo `onclick` del campo botón y entre comillas podemos ver lo que queremos que se ejecute, en este caso la función indicada.

```
<form name=formul>
<input type=text name=texto>
<input type=button value=validar onclick="validarFormulario()">
</form>
```

### **Función `validarFormulario()`**

Esta función extrae el dato del campo de texto y lo pasa a la función `validarEntero()`, que nos devolverá un valor que tendremos que colocar de nuevo en el campo de texto. Para acceder al formulario utilizamos la jerarquía de objetos del navegador, que para el que no lo sepa, es un conjunto de objetos que hacen referencia a todos los elementos de la página.

El acceso a los elementos de la página se realiza empezando en el objeto `window`, que es el primero de la jerarquía. Luego continúa por el objeto `document` –que guarda todo el documento– y en nuestro ejemplo, bajaremos al formulario para poder acceder definitivamente al campo de texto, que era donde queríamos llegar. Con este esquema:

```
window.document.formul.texto
```

Nos fijamos que se utilizan los nombres del formulario y el campo que hemos colocado en el atributo `name` de las etiquetas HTML de estos elementos.

Todos los campos de texto tienen una propiedad `value` que es donde se guarda el texto que lleva escrito dentro. De modo que si queremos acceder a lo que tiene escrito el campo de texto escribiremos esto:

```
window.document.formul.texto.value
```

Ahora que sabemos todo lo anterior deberíamos entender perfectamente el código de la función.

```
function validarFormulario(){
    //extraemos el valor del campo
    textoCampo = window.document.formul.texto.value
    //lo validamos como entero
    textoCampo = validarEntero(textoCampo)
    //colocamos el valor de nuevo
    window.document.formul.texto.value = textoCampo
}
```

### **Conclusión**

Con todo esto tenemos completado el ejercicio. Podemos [ver como funcionaría la página entera](#) para observar los resultados finales.

Por si sólo puede que no haya resultado muy productivo este ejemplo, pero puede ser un comienzo para empezar a validar formularios más complejos. Con un poco de imaginación y esfuerzo podemos hacer funciones que validen otros campos del formulario para ver si lo que contienen son textos, si son lo suficientemente largos o cortos, validar números con decimales, etc. Todo ello se haría de manera similar a como hemos visto, añadiendo código a la función `validarFormulario()` y tal vez construyendo algunas funciones accesorias adicionales como `validarEntero()`.

Esperamos que haya servido de provecho, aunque sea un poco.

Artículo por **Miguel Angel Alvarez**

## Ejemplos de funcionamiento de la clase String

Ahora vamos a ver unos ejemplos sobre cómo se utilizan los métodos y propiedades del objeto String.

### Ejemplo de strings 1

Vamos a escribir el contenido de un string con un carácter separador ("-") entre cada uno de los caracteres del string.

```
var miString = "Hola Amigos"
var result = ""

for (i=0;i<miString.length-1;i++) {
    result += miString.charAt(i)
    result += "-"
}
result += miString.charAt(miString.length - 1)

document.write(result)
```

Primero creamos dos variables, una con el string a recorrer y otra con un string vacío, donde guardaremos el resultado. Luego hacemos un bucle que recorre desde el primer hasta el penúltimo carácter del string -utilizamos la propiedad length para conocer el número de caracteres del string- y en cada iteración colocamos un carácter del string seguido de un carácter separador "-". Como aun nos queda el último carácter por colocar lo ponemos en la siguiente línea después del bucle. Utilizamos la función charAt para acceder a las posiciones del string. Finalmente imprimimos en la página el resultado.

Podemos [ver el ejemplo en funcionamiento en una página a parte](#).

### Ejemplo de strings 2

Vamos a hacer un script que rompa un string en dos mitades y las imprima por pantalla. Las mitades serán iguales, siempre que el string tenga un número de caracteres par. En caso de que el número de caracteres sea impar no se podrá hacer la mitad exacta, pero partiremos el string lo más aproximado a la mitad.

```
var miString = "0123456789"
var mitad1,mitad2

posicion_mitad = miString.length / 2

mitad1 = miString.substring(0,posicion_mitad)
mitad2 = miString.substring(posicion_mitad,miString.length)

document.write(mitad1 + "<br>" + mitad2)
```

Las dos primeras líneas sirven para declarar las variables que vamos a utilizar e inicializar el string a partir. En la siguiente línea hallamos la posición de la mitad del string.

En las dos siguientes líneas es donde realizamos el trabajo de poner en una variable la primera mitad del string y en la otra la segunda. Para ello utilizamos el método substring pasándole como inicio y fin en el primer caso desde 0 hasta la mitad y en el segundo desde la mitad hasta el final. Para finalizar imprimimos las dos mitades con un salto de línea entre ellas.

Podemos [ver el ejemplo en funcionamiento en una página a parte](#).

*Artículo por Miguel Angel Alvarez*

## **Ejemplo de funcionamiento de Date**

En este ejemplo vamos a crear dos fechas, una con el instante actual y otra con fecha del pasado. Luego las imprimiremos las dos y extraeremos su año para imprimirlo también. Luego actualizaremos el año de una de las fechas y la volveremos a escribir con un formato más legible.

```
//en estas líneas creamos las fechas
miFechaActual = new Date()
miFechaPasada = new Date(1998,4,23)

//en estas líneas imprimimos las fechas.
document.write (miFechaActual)
document.write ("<br>")
document.write (miFechaPasada)

//extraemos el año de las dos fechas
anoActual = miFechaActual.getFullYear()
anoPasado = miFechaPasada.getFullYear()

//Escribimos en año en la página
document.write("<br>El año actual es: " + anoActual)
document.write("<br>El año pasado es: " + anoPasado)

//cambiamos el año en la fecha actual
miFechaActual.setFullYear(2005)

//extraemos el día mes y año
dia = miFechaActual.getDate()
mes = parseInt(miFechaActual.getMonth()) + 1
ano = miFechaActual.getFullYear()

//escribimos la fecha en un formato legible
document.write ("<br>")
document.write (dia + "/" + mes + "/" + ano)
```

Si se desea, se puede [ver en funcionamiento este script](#) en una página a parte.

Hay que destacar un detalle antes de acabar y es que el número del mes puede empezar desde 0. Por lo menos en el Netscape con el que realizamos las pruebas empezaba en 0 el mes. Por esta razón sumamos uno al mes que devuelve el método `getMonth`.

Hay más detalles a destacar, pues resulta que en Netscape el método `getFullYear()` devuelve los años transcurridos desde 1900, con lo que al obtener el año de una fecha de, por ejemplo, 2005, indica que es el año 105. Para obtener el año completo tenemos a nuestra disposición el método `getFullYear()` que devolvería 2005 del mismo modo en Netscape e Internet Explorer.

Mucha atención, pues, al trabajo con fechas en distintas plataformas, puesto que podría ser problemático el hecho de que nos ofrezcan distintas salidas los métodos de manejo de fechas, con siempre dependiendo de la marca y versión de nuestro navegador.

**Referencia:** Se puede ver otro ejemplo de trabajo con la clase `Date` en el taller [Calcular la edad en Javascript](#)

Artículo por **Miguel Angel Alvarez**

## Enlace aleatorio Javascript

Vamos a crear un efecto típico en páginas web que consiste en un enlace que nos llevará a un sitio escogido de manera aleatoria. Para hacerse una idea exacta podemos [ver el ejemplo en funcionamiento](#).

Para ello vamos a utilizar Javascript. Aunque por algunas razones no es el mejor lenguaje para hacer este ejercicio, si que va a resultar extremadamente simple y creemos que instructivo para los lectores.

Para empezar, vamos a crear un array con los distintos sitios hacia donde nos podría conducir nuestro enlace. Este array lo tenemos que definir, lógicamente, dentro de un bloque `<script>` en la propia página web. La razón por la que Javascript no es el mejor lenguaje para este ejercicio es justamente esta, que tenemos que escribir en la página todas las posibles direcciones y un usuario avanzado podría leer el código de la página y encontrar todas las opciones escritas en el array.

Esa declaración del array sería algo parecido a esto.

```
var direcciones = new Array("http://www.terra.es","http://www.ozu.es","http://www.hispavista.com")
```

Como se puede ver, en la misma línea en la que se declara el array se introducen los valores de cada una de sus casillas, utilizando el método rápido de declaración y llenado de arrays en Javascript. Cuantos más valores escribamos, más aleatorio será el ejercicio, pudiendo colocar más enlaces sin tener que editar el resto del código del programa. En nuestro ejemplo completo tenemos una lista mucho mayor de enlaces.

Continuamos colocando el enlace que se presentará como "Enlace Aleatorio", que nos llevará a un sitio aleatorio, dentro de las posibilidades.

```
<a href="javascript:enlaceAleatorio()">Enlace Aleatorio</a>
```

Como vemos, el enlace se encarga de llamar a una función que será la que extraiga una URL del array anterior y nos traslade a ese lugar. La función tendrá esta forma:

```
function enlaceAleatorio(){
    aleat = Math.random() * direcciones.length
    aleat = Math.floor(aleat)
    window.location=direcciones[aleat]
}
```

Como se puede ver, lo primero que hace la función es obtener un valor aleatorio entre 0 y "direcciones.length", que es el número de URLs de nuestro array. Si cambiamos el número de URLs del array este script seguirá funcionando perfectamente, porque los límites se extraen directamente de la propiedad length del array que contiene las direcciones.

Para obtener ese número aleatorio se utiliza el método random de la clase Math, que devuelve un número entre 0 y 1. Al multiplicarlo por el número de posiciones del array obtenemos un número entre 0 y el número de posiciones del array. Pero este número está en coma flotante, es decir, es un número decimal, que no nos sirve como índice de un array. Por eso le aplicamos el método floor, también del objeto Math, para obtener la parte entera de ese número.

Por último se actualiza la propiedad location del objeto window con el valor del array en la posición aleatoria, lo que hace que el navegador se dirija a la página aleatoria, dentro de las distintas posibilidades.

## Ejemplo completo

Para ver de manera global este ejercicio transcribimos aquí todo el código utilizado.

```
<html>
<head>
    <title>Enlace Aleatorio</title>
<script>
    var direcciones = new Array("http://www.terra.es", "http://www.ozu.es", "http://www.hispavista.com",
    "http://www.lycos.com", "http://www.yahoo.es", "http://www.altavista.com", "http://www.hotbot.com",
    "http://www.buscopio.com", "http://www.sol.es", "http://www.excite.com", "http://www.elbuscador.com",
    "http://www.ya.com", "http://www.wanadoo.es", "http://www.buscador.com.mx", "http://www.msn.com",
    "http://www.astrolabio.net")
    function enlaceAleatorio(){
        aleat = Math.random() * direcciones.length
        aleat = Math.floor(aleat)
        window.location=direcciones[aleat]
    }
</script>
</head>

<body>
<a href="javascript:enlaceAleatorio()">Enlace Aleatorio</a>
</body>
</html>
```

Si se quiere [ver el ejemplo en marcha se puede pulsar este enlace](#).

*Artículo por **Miguel Angel Alvarez***



## Generación de números aleatorios Javascript

En Javascript disponemos de la clase Math, muy útil cuando queremos hacer cálculos matemáticos de cierta complejidad. Dicha clase está [explicada y documentada en un capítulo del manual de Javascript II](#). Para el que lo necesite, también tenemos [explicaciones de lo que son las clases y los objetos](#).

En este taller de Javascript vamos a construir una simple función para crear un número aleatorio, entre un mínimo y un máximo, que podremos utilizar luego en otros scripts más complejos.

Aquí tenemos el código que hace uso del método random de la clase Math para obtener un número aleatorio con Javascript.

```
var aleatorio = Math.random()
```

Así hemos creado una variable aleatorio a la que asignamos el resultado de ejecutar el método random de la clase Math. El número aleatorio que obtenemos siempre estará comprendido entre 0 y 1.

Si deseamos obtener un número aleatorio en otro rango lo podemos conseguir con un poco de matemáticas y la clase Math. Para ilustrarlo vamos a hacer una función que devuelve un número aleatorio comprendido en un intervalo. El intervalo lo recibe como parámetro con dos variables, una para el límite por la parte inferior y otra para la el límite por la parte superior.

```
function aleatorio(inferior,superior){  
    numPosibilidades = superior - inferior  
    aleat = Math.random() * numPosibilidades  
    aleat = Math.round(aleat)  
    return parseInt(inferior) + aleat  
}
```

La función que hemos hecho es muy sencilla, pero funciona perfectamente para todos los tipos de intervalos que podamos pasarle, tanto con números positivos como negativos. Lo primero que hacemos es obtener el número de posibilidades restando al límite superior el inferior. Luego multiplicamos dicho numero de posibilidades por el número aleatorio obtenido (que está entre 0 y 1), con lo que obtenemos un número aleatorio entre 0 y el número de posibilidades.

**Nota:** Esta función tiene un pequeño problema, pues los números aleatorios que devuelve no tienen las mismas posibilidades de salir. Para solucionarlo, un visitante ha incluido un comentario en el artículo que se aconseja leer.

El número tiene un montón de decimales, y en este ejemplo deseamos obtener un número entero, sin decimales. Por eso luego utilizamos el método round() de la clase Math, que nos da el entero más próximo. Como el número todavía está entre 0 y el número de posibilidades tenemos que sumarle el límite inferior, con lo que estará dentro del rango que deseamos. Este último valor será en que devuelva la función.

Esta función se puede [ver en marcha en una página a parte](#). En el ejemplo hemos construido un pequeño formulario que podemos rellenar con el mínimo y el máximo y cuando apretemos sobre el botón se mostrará el valor aleatorio en el campo de abajo del todo.

Un ejemplo de lo que podemos hacer con un número aleatorio puede ser crear un enlace

aleatorio en una página web. Lo podemos ver en el ejemplo [Enlace aleatorio en Javascript](#). Además, en este ejemplo se crea el número aleatorio de manera ligeramente distinta a como lo hemos visto ahora, lo que puede ser interesante para aprender mejor a usar los métodos de la clase Math.

**Nota:** Insisto, leeros los comentarios que acompañan a este artículo (un poco más abajo de estas líneas) ya que ofrecen scripts mejorados para crear números aleatorios. Además, hay algún comentario con un script para asegurarse que las posibilidades de aparición de cada valor aleatorio sean las mismas.

*Artículo por **Miguel Angel Alvarez***

## Comprobar si las claves son iguales

En un formulario de registro de usuarios es muy habitual que deseemos incluir un campo clave, que podría utilizarlo el usuario para acceder a los servicios de la web donde se está registrando o para actualizar más adelante la información introducida.

Para realizar un formulario donde se escriba la clave, lo normal es que aparezcan dos campos donde introducir la misma clave y que el usuario que la introduzca esté forzado a escribir la clave dos veces, siendo las dos claves introducidas iguales. Esto nos ayuda a que el usuario no se equivoque al escribir la clave por un fallo en la mecanografía, ya que es complicado que se equivoque dos veces en al escribir la clave.

Con Javascript podemos hacer una comprobación -en el cliente- para ver si se ha introducido la misma clave en los dos campos y, si es así, mandar el formulario al servidor o hacer aquellas acciones necesarias para continuar con el registro de ese visitante. En caso de que las dos claves sean distintas se debería informar al usuario de la situación para que vuelva a introducir la clave deseada correctamente.

### Comprobar si dos campos de formulario son iguales

Es una acción bastante sencilla. Simplemente debemos extraer los dos valores guardados en los campos del formulario donde se ha escrito la clave y la repetición de la clave.

Luego, con un enunciado if, podemos comprobar si estos dos datos son el mismo o no lo son y hacer las sentencias necesarias en cada caso.

La función podría tener una forma como esta:

```
function comprobarClave(){
    clave1 = document.f1.clave1.value
    clave2 = document.f1.clave2.value

    if (clave1 == clave2)
        alert("Las dos claves son iguales...\nRealizaríamos las acciones del caso positivo")
    else
        alert("Las dos claves son distintas...\nRealizaríamos las acciones del caso negativo")
}
```

**Nota:** si deseamos mandar el formulario al servidor después de haber visto que las dos claves son iguales, es decir, si después de la comprobación queremos submitir el formulario, podríamos ejecutar el método submit() del objeto form.

```
document.f1.submit()
```

Nótese que el formulario donde estamos accediendo por la jerarquía de objetos del navegador se llama "f1". Eso quiere decir que la etiqueta <FORM> del formulario donde están los dos campos tiene el atributo name="f1". Si nuestro formulario se llama de otra manera habría que cambiar las líneas donde se accede a los dos campos de la clave, sustituyendo "f1" por el nombre de nuestro formulario. Del mismo modo, los dos campos donde se escriben las claves en nuestro ejemplo se supone que se llaman "clave1" y "clave2", si no es así, también habría que cambiar ese nombre por el que toque.

Para verlo más claro, aquí tenemos el código del formulario donde se encuentran los campos y sus nombres.

```
<form action="" name="f1">
Contraseña: <input type="password" name="clave1" size="20">
<br>
Repite contraseña: <input type="password" name="clave2" size="20">
<br>
<input type="button" value="Comprobar si son iguales" onClick="comprobarClave()">
</form>
```

## Conclusión

Dada su sencillez, no debería suponer mucho problema este ejemplo. Nos quedaría ver el código del ejemplo completo y, si lo deseamos, [acceder a una página donde se muestra el ejercicio en marcha](#).

```
<html>
<head>
<title>Validar si la contraseña y la repetición de la contraseña son iguales</title>
<script>
function comprobarClave(){
    clave1 = document.f1.clave1.value
    clave2 = document.f1.clave2.value

    if (clave1 == clave2)
        alert("Las dos claves son iguales...\nRealizaríamos las acciones del caso positivo")
    else
        alert("Las dos claves son distintas...\nRealizaríamos las acciones del caso negativo")
}
</script>
</head>

<body>

<h1>Validar si la contraseña y la repetición de la contraseña son iguales</h1>
<br>
Escribe una supuesta contraseña dos veces, una en cada campo, y pulsa el botón. Javascript te dirá si las dos son iguales.

<br>
<form action="" name="f1">
Contraseña: <input type="password" name="clave1" size="20">
<br>
Repite contraseña: <input type="password" name="clave2" size="20">
<br>
<input type="button" value="Comprobar si son iguales" onClick="comprobarClave()">

</form>
```

```
</body>
</html>
```

Artículo por **Miguel Angel Alvarez**

## Ej. trabajo con formularios. Calculadora sencilla

Para ilustrar un poco el trabajo con formularios, vamos a realizar un ejemplo práctico. Puede que algunas cosas que vamos a tratar queden un poco en el aire porque no se hayan explicado con detenimiento antes, pero seguro que nos sirve para enterarnos de cómo se trabaja con formularios y las posibilidades que tenemos.

### Ejemplo calculadora sencilla

En este ejemplo vamos a construir una calculadora, aunque bastante sencilla, que permita realizar las operaciones básicas. Para hacer la calculadora vamos a realizar un formulario en el que vamos a colocar tres campos de texto, los dos primeros para los operandos y un tercero para el resultado. Además habrán unos botones para hacer las operaciones básicas.

**El formulario de la calculadora se puede ver aquí.**

```
<form name="calc">
<input type="Text" name="operando1" value="0" size="12">
<br>
<input type="Text" name="operando2" value="0" size="12">
<br>
<input type="Button" name="" value=" + " onclick="calcula('+')">
<input type="Button" name="" value=" - " onclick="calcula('-')">
<input type="Button" name="" value=" X " onclick="calcula('*')">
<input type="Button" name="" value=" / " onclick="calcula('/')">
<br>
<input type="Text" name="resultado" value="0" size="12">
</form>
```

Mediante una función vamos a acceder a los campos del formulario para recoger los operandos en dos variables. Los campos de texto tienen una propiedad llamada value que es donde podemos obtener lo que tienen escrito en ese momento. Mas tarde nos ayudaremos de la [función eval\(\)](#) para realizar la operación. Pondremos por último el resultado en el campo de texto creado en tercer lugar, utilizando también la propiedad value del campo de texto.

A la función que realiza el cálculo (que podemos ver a continuación) la llamamos apretando los botones de cada una de las operaciones. Dichos botones se pueden ver en el formulario y contienen un atributo onclick que sirve para especificar las sentencias Javascript que deseamos que se ejecuten cuando el usuario pulse sobre él. En este caso, la sentencia a ejecutar es una llamada a la función calcula() pasando como parámetro el símbolo u operador de la operación que deseamos realizar.

### El script con la función calcula()

```
<script>
```

```
function calcula(operacion){  
    var operando1 = document.calc.operando1.value  
    var operando2 = document.calc.operando2.value  
    var result = eval(operando1 + operacion + operando2)  
    document.calc.resultado.value = result  
}  
</script>
```

La [función eval\(\)](#), recordamos, que recibía un string y lo ejecutaba como una sentencia Javascript. En este caso va a recibir un número que concatenado a una operación y otro número será siempre una expresión aritmética que eval() solucionará perfectamente.

Podemos [ver el ejemplo de la calculadora en funcionamiento](#).

El acceso a otros elementos de los formularios se hace de manera parecida en cuanto respecta a la jerarquía de objetos, aunque como cada elemento tiene sus particularidades las cosas que podremos hacer con ellos diferirán un poco. Lo veremos un poco más adelante.

*Artículo por **Miguel Angel Alvarez***

## **Enviar al navegador a otra página si no tiene Javascript**

Imaginemos una página que, para verse bien, necesite tener habilitada la posibilidad de ejecutar scripts en Javascript y que, si no tiene habilitado Javascript, no funcionase bien y no pudiese mostrar todos los contenidos.

En un caso como este nos sería muy útil disponer de una función que detecte si está habilitado o no Javascript para, en caso de que no sea así, se envíe al navegador a otra dirección.

Pues bien, esa función que detecta si está o no habilitado Javascript no se puede hacer tan ricamente, por lo menos utilizando Javascript. Imagina que no dispones de Javascript, el navegador no podría entonces ejecutar esa función y nunca detectarías que no hay Javascript.

Por suerte tenemos un enunciado `<NOSCRIPT></NOSCRIPT>` que nos sirve para indicar acciones a tomar en caso de que no esté habilitado Javascript.

Utilizando esa etiqueta podemos poner un enlace para que se vea sólo en los navegadores que no tienen Javascript:

```
<NOSCRIPT>  
Tu navegador no soporta Javascript. <a href="no_javas.html">Entra en una página que no lo  
utiliza</a>  
</NOSCRIPT>
```

Podemos ir un paso más allá y utilizar la etiqueta META tipo "Refresh" para que el navegador se refresque automáticamente y se dirija a otra página que no incluya programación en Javascript.

Es una opción mucho más interesante, porque no tenemos que esperar a que el visitante pulse

un enlace y así nos aseguramos que, aunque no encontrase el enlace, el navegador lo redirija correctamente.

```
<NOSCRIPT>
<META HTTP-EQUIV="Refresh" CONTENT="3;URL=no_javas.html">
</NOSCRIPT>
```

Obviamente, esto sólo funcionará si nuestro navegador acepta este tipo de etiquetas de refresco automático, aunque los navegadores más habituales sí las aceptan.

Por cierto, la etiqueta de refresco debe colocarse en la cabecera (Dentro de `<HEAD></HEAD>`). El primer dato del valor de refresco es el tiempo de espera antes de refrescarse en segundos, en este caso 3 segundos. El segundo dato es la dirección a la que queremos enviar el navegador, en este caso `no_javascript.html`.

*Artículo por **Miguel Angel Alvarez***

## Confirmación de envío de formulario

Este artículo viene a ampliar una FAQ de las que mantenemos en la sección de FAQs de DesarrolloWeb.com. La pregunta en concreto que nos realizó el usuario era cómo hacer un formulario que, al enviarlo, nos pregunte si realmente se desea enviar.

### La pregunta en concreto era la siguiente:

*Estoy haciendo un formulario y deseo que, al enviarlo, me muestre una ventana de confirmación de envío del formulario, de esas que tienen un botón de aceptar y otro de cancelar. Entonces, si se acepta el envío, se enviaría el formulario.... si no se acepta, que el formulario no se envíe.*

### Respuesta

Esto tiene mucho que ver con el tema de tratamiento de formularios. La respuesta basa su mayor técnica en el hecho de sustituir el botón de submit por un botón normal. Con el botón normal no se envía el formulario directamente sino que se llama a una función que realiza la confirmación y, en caso positivo, envía el formulario.

El botón que colocaríamos en el formulario en sustitución del botón de submit sería el siguiente:

```
<input type=button onclick="pregunta()" value="Enviar">
```

Nos fijamos en que el botón tiene definida una acción en el momento que se hace clic. La acción en concreto hace que se ejecute la función `pregunta()`, que será la que realice la confirmación y envíe el formulario en caso positivo. Su código se puede ver a continuación.

```
<script language="JavaScript">
function pregunta(){
    if (confirm('¿Estas seguro de enviar este formulario?')){
        document.tuformulario.submit()
    }
}
</script>
```

La caja `confirm` devuelve `true` o `false` dependiendo de si se pulsa el botón de aceptar o

cancelar. Ese valor se utiliza en un enunciado if para decidir si se envía el formulario, con su método submit(), o no se hace nada.

El código completo de una página que realiza esta tarea en un formulario es el siguiente:

```
<html>
<head>
  <title>Confirmación de envío de formulario</title>
  <script language="JavaScript">
function pregunta(){
  if (confirm('¿Estas seguro de enviar este formulario?')){
    document.tuformulario.submit()
  }
}
  </script>
</head>

<body>
<form name=tuformulario action="http://www.desarrolloweb.com">
<input type="text" name="cualquiercampo">
<input type=button onclick="pregunta()" value="Enviar">
</form>

</body>
</html>
```

El ejercicio se puede [ver en marcha aquí](#).

*Artículo por **Miguel Angel Alvarez***

## Javascript para posicionarse en un select

Se trata de un script para posicionarse en un elemento de un select, es decir, para conseguir que, pulsando unas teclas del teclado que podrían corresponder con las primeras letras de un elemento del select, el elemento seleccionado de dicho select sea aquel que corresponda con las letras pulsadas.

Es una descripción un poco larga, pero en realidad el efecto es sencillo. En los select de las páginas web, al pulsar una tecla, el select se mueve al primer elemento que tiene como inicial esa tecla. Sin embargo, si hay muchos elementos en el select, el usuario puede encontrar que esa ayuda se queda un poco corta, ya que tendría que, luego de pulsar la inicial del elemento buscado, repasar todos los elementos que comienzan por esa letra hasta encontrar el que busca. El presente ejemplo mejora esa función de búsqueda en los select, ya que permite realizar la pulsación de varias teclas seguidas y va mostrando aquel elemento que comienza por todas las letras que se han pulsado (una detrás de otra) hasta que se apreta la tecla Enter, momento en el cual se supone que hemos encontrado el elemento adecuado y queremos continuar con el rellenado de otros campos del formulario.

En este ejemplo se ha creado un select con los nombres de distintos países. Si, por ejemplo, queremos buscar el país Estados Unidos, en los selects normales podemos pulsar la E (inicial de Estados Unidos) y buscar entre todos los países hasta que aparece el que queremos. Pero, con la implementación de este script podremos pulsar la E, con lo que se posicionará en el primer país que empiece por E (que no tiene porque ser el que buscamos, en la práctica será Ecuador). Luego podemos pulsar la letra S, con lo que se mostrará España, que no es el que

buscamos. Más tarde se pulsaría la T, apareciendo Estonia y, por último, al pulsar la tecla A, ya aparece el elemento que buscábamos, ESTAdos Unidos.

## El script

El script se encuentra comentado dentro del propio código, para que se pueda comprender fácilmente, o por lo menos sus bases. Básicamente, se utiliza el evento de teclado `onKeyPress` en el elemento `select` de los países, de modo que, cuando se pulse una tecla, si tenemos el foco en el `select`, se llamará a una función que se encargará de hacer el trabajo más duro.

Dicho trabajo consiste en recoger la tecla que se ha pulsado y guardarla en una estructura de datos, además de seleccionar el elemento más próximo al valor actual de la estructura de datos. Por último, si se pulsa la tecla `enter`, se deja el `select` con el último valor seleccionado y se pasa el foco al siguiente elemento del formulario para que el usuario siga rellenándolo.

```
<script language="JavaScript1.2">
var digitos=10 //cantidad de digitos buscados
var puntero=0
var buffer=new Array(digitos) //declaración del array Buffer
var cadena=""

function buscar_op(obj,objfoco){
  var letra = String.fromCharCode(event.keyCode)
  if(puntero >= digitos){
    cadena="";
    puntero=0;
  }
  //si se presiona la tecla ENTER, borro el array de teclas presionadas y salto a otro objeto...
  if (event.keyCode == 13){
    borrar_buffer();
    if(objfoco!=0) objfoco.focus(); //evita foco a otro objeto si objfoco=0
  }
  //sino busco la cadena tipeada dentro del combo...
  else{
    buffer[puntero]=letra;
    //guardo en la posicion puntero la letra tipeada
    cadena=cadena+buffer[puntero]; //armo una cadena con los datos que van ingresando al array
    puntero++;

    //barro todas las opciones que contiene el combo y las comparo la cadena...
    for (var opcombo=0;opcombo < obj.length;opcombo++){
      if(obj[opcombo].text.substr(0,puntero).toLowerCase()==cadena.toLowerCase()){
        obj.selectedIndex=opcombo;
      }
    }
  }
  event.returnValue = false; //invalida la acción de pulsado de tecla para evitar busqueda del primer caracter
}

function borrar_buffer(){
  //inicializa la cadena buscada
  cadena="";
  puntero=0;
}
</script>

<html>
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body bgcolor="#FFFFFF" text="#000000">
```



```
<table width="544" border="0" cellpadding="0" cellspacing="0">
<tr>
<td width="89" height="29"></td>
<td width="114"></td>
<td width="26"></td>
<td width="315"></td>
</tr>
<tr>
<td height="19"></td>
<td valign="top">
<select name="combo1" onkeypress=buscar_op(this,text2) onblur=borrar_buffer() onclick=borrar_buffer(>
  <option>Argentina</option>
  <option>Australia</option>
  <option>Bolivia</option>
  <option>Brasil</option>
  <option>Canada</option>
  <option>Colombia</option>
  <option>Dinamarca</option>
  <option>Estados Unidos</option>
  <option>Estonia</option>
  <option>Austria</option>
  <option>Bulgaria</option>
  <option>Chile</option>
  <option>España</option>
  <option>China</option>
  <option>Costa Rica</option>
  <option>Croacia</option>
  <option>Ecuador</option>
</select>
</td>
<td></td>
<td></td>
</tr>
<tr>
<td height="18"></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td height="28"></td>
<td colspan="2" valign="top">
<input type="text" name="text2">
</td>
<td></td>
</tr>
<tr>
<td height="58"></td>
<td></td>
<td></td>
<td></td>
</tr>
</table>
</body>
</html>
```

Esperamos que podáis entender el script y utilizarlo en vuestras páginas web. Si os interesa, podéis [descargar el script en un archivo comprimido](#).

*Artículo por **Ignacio Rodriguez***

## Inhibir un campo texto de formulario con Javascript

Esta vez toca un taller muy rápido y sencillo con Javascript para hacer que un campo de formulario de tipo texto se encuentre inhibido, es decir, que no podamos colocarnos encima de él para editar su contenido.

**Referencia:** Si lo que queremos es inhibir un campo de formulario de tipo radio (radio button) será necesaria otra técnica relatada en un taller distinto: [Inhibir radio button con Javascript](#)

### Focus y Blur

La manera de hacerlo requiere el conocimiento de dos conceptos habituales de Javascript relacionados con el foco de la aplicación.

**El concepto focus**, está relacionado con ganar foco de la aplicación. El método **focus()**, que tienen los campos de texto y otros elementos de formulario, sirve otorgar el foco de la aplicación a ese elemento. El manejador de evento **onfocus** salta cuando un elemento gana el foco de la aplicación.

**El concepto blur**, está asociado a perder el foco de la aplicación. El método **blur()** sirve para que los elementos de formulario pierdan el foco y el manejador de eventos **onblur** se activa cuando el elemento al que lo apliquemos pierda el foco de la aplicación.

### El ejercicio

Para inhibir un campo de formulario podemos hacer que el usuario nunca se pueda posar en ese elemento o bien, que si se llega a posar, se expulse inmediatamente. Para esto lo único que tenemos que hacer es retirar el foco de un elemento cuando lo haya ganado

Nosotros utilizaremos el evento onfocus para detectar el instante en el que el elemento gana el foco y en ese momento haremos uso del método blur() para retirar el foco.

El código es extremadamente simple para tanta explicación:

```
<form>
<input type="text" value="122" onfocus="this.blur()">
</form>
```

El único detalle que merece la pena señalar es el uso de la palabra this, que hace referencia al elemento donde se está utilizando, en ese caso el campo de texto. this.blur() sería una simple llamada al método blur() en el elemento de formulario donde está colocada.

Puede verse en funcionamiento aquí:



Artículo por **Miguel Angel Alvarez**

## Capas con Internet Explorer 5, 6, Netscape 6, 7 y Opera

Todos los programadores han tenido alguna vez que enfrentarse a los problemas de compatibilidad que presentan los navegadores desarrollados por Netscape y Microsoft.

Debido a que tenían dos **DOM** (Modelo de objetos del documento) distintos, la forma de acceder de uno y otro eran diferentes, provocando la necesidad de utilizar distintos trucos para conseguir que los scripts fueran compatibles con los dos navegadores.

Estos problemas se han solucionado en cierta medida desde la adopción por parte de las compañías del DOM definido por el W3C, al poder escribir el mismo código y que sea compatible con los dos navegadores. El único problema que presenta es que el script no será compatible con las versiones anteriores de dichos navegadores, provocando en caso de necesitar compatibilidad hacia atrás, la necesidad de añadir el código específico para las versiones anteriores que se deseen implementar.

Una de las funciones clave para trabajar con los dos navegadores al mismo tiempo, es la función **getElementById(elemento)** que recibe por parámetro el nombre de un elemento de la página, y devuelve el objeto correspondiente. A través de esta función, se podrán acceder a todas las propiedades del objeto. Esta función está definida en el estándar del W3C.

Una vez tenemos el objeto, bastará con acceder a la propiedad `style`, para a través de ella, acceder a todos los estilos definidos en el elemento.

La ventaja de utilizar este método, es que no necesitaremos distinguir que navegador se está utilizando, ya que funciona para los dos.

*Artículo por **Carlos Luis Cuenca***

## Mostrar y ocultar capas con IE 5,6 NS 6,7

Lo primero, es definir las funciones de mostrar y ocultar. Estas dos funciones, recibirán por parámetro, el nombre de la capa que se quiere mostrar u ocultar, a continuación, mediante la función `getElementById`, accederemos a las propiedades de la capa, y a través de `style` a los estilos que nos definen si la capa está visible o no (propiedad `visibility`), activándola o desactivándola según la función.

**Referencia:** Hablamos sobre la función `getElementById()` en el artículo [Capas con Internet Explorer 5, 6, Netscape 6, 7 y Opera](http://www.desarrolloweb.com/manuales/22/)

```
<script language="Javascript">
function mostrar(nombreCapa){
document.getElementById(nombreCapa).style.visibility="visible";
}
function ocultar(nombreCapa){
document.getElementById(nombreCapa).style.visibility="hidden";
}
</script>
```

A continuación es necesario definir las capas, y llamar en algún momento a las funciones definidas. Se utilizarán los eventos `onMouseOver` y `onMouseOut` de la capa 1 para mostrar y

ocultar la capa 2.

```
<div id="capa1" style="position:absolute;width:100;height:100;top:100;left:100;background-color:blue"
onmouseout="ocultar('capa2')" onmouseover="mostrar('capa2')">Capa 1</div>
<div id="capa2" style="position:absolute;width:100;height:100;top:100;left:200;background-
color:red;visibility:hidden">Capa 2</div>
```

Al pasar por encima de la capa 1 se mostrará la 2, y al salir de la capa 1, se ocultará la dos.

### Ejemplo completo:

```
<html>
<head>
<title>Untitled</title>
<script language="Javascript">
function mostrar(nombreCapa){
document.getElementById(nombreCapa).style.visibility="visible";
}
function ocultar(nombreCapa){
document.getElementById(nombreCapa).style.visibility="hidden";
}
</script>
</head>
<body>
<div id="capa1" style="position:absolute;width:100;height:100;top:100;left:100;background-color:blue"
onmouseout="ocultar('capa2')" onmouseover="mostrar('capa2')">Capa 1</div>
<div id="capa2" style="position:absolute;width:100;height:100;top:100;left:200;background-
color:red;visibility:hidden">Capa 2</div>
</body>
</html>
```

Podemos [ver el ejemplo en marcha](#).

*Artículo por **Carlos Luis Cuenca***

## Movimiento de Capas con IE 5,6 NS 6,7

Para desplazar una capa por la pantalla, es necesario modificar los atributos Top y Left.

**Referencia:** Hablamos sobre la función `getElementById()`, necesaria para entender la manera de trabajar en este artículo, en el documento [Capas con Internet Explorer 5, 6, Netscape 6, 7 y Opera](#)

**Top** define la posición vertical de la capa desde la parte superior de la pantalla

**Left** define la posición horizontal de la capa desde la parte izquierda de la pantalla.

Al modificar los valores de Top y Left lograremos el movimiento de la capa.

Por último, hay que reseñar que los valores, almacenan una cadena de texto con el valor y las unidades, por lo que es necesario convertir los valores que se almacenan dentro de las propiedades antes de poder utilizarlos.

Por ejemplo, un valor almacenado en top o left podrá ser 140px.

Por este motivo es necesario utilizar la función `parseInt` para poder convertir los valores a números.

A continuación se muestra un ejemplo, que desplaza una capa 5 pixels cada vez que se pulsa el botón:

La función recibe por parámetro el nombre de la capa que se desea mover, accede mediante la función `getElementById` a la propiedad de `top`, mediante `parseInt` la convierte a un entero, le suma 5 unidades, y a continuación escribe el nuevo valor en el estilo de la capa.

```
<script language="Javascript">
function mover(nombreCapa){
valor=document.getElementById(nombreCapa).style.top;
numero=parseInt(valor);
numero+=5;
document.getElementById(nombreCapa).style.top=numero;
}
</script>
```

Ejemplo completo:

```
<html>
<head>
<title>Untitled</title>
<script language="Javascript">
function mover(nombreCapa){
valor=document.getElementById(nombreCapa).style.top;
numero=parseInt(valor);
numero+=5;
document.getElementById(nombreCapa).style.top=numero;
}
</script>
</head>
<body>
<div id="capa1" style="position:absolute;width:100;height:100;top:100;left:100;background-color:blue">Capa
1</div>
<form name="miform" action="#">
<input type="button" onclick="mover('capa1')" value="Mover Capa">
</form>
</body>
</html>
```

[Ver el ejemplo](#)

*Artículo por **Carlos Luis Cuenca***

## Escritura en las Capas con IE 5, 6, NS 6, 7

El texto que se encuentra almacenado dentro de una capa, se encuentra dentro de la propiedad `innerHTML`. Esta propiedad, esta implementada en Internet Explorer desde la versión 4, y Netscape lo ha incorporado en las versiones 6 y 7 a pesar de que no se encuentra definido en el estandar del W3C.

**Referencia:** Este artículo ofrece ayudas para las que es necesario conocer un poco Javascript y el método `getElementById()`. Recomendamos consultar estos enlaces:

### [Programación en JavaScript](#) [Capas con Internet Explorer 5, 6, Netscape 6, 7 y Opera](#)

La propiedad almacena el código HTML que se corresponde con lo que se visualiza dentro de la capa, es decir, si la capa contiene " hola " la propiedad contendrá " hola " sin embargo, si la propiedad contiene un hipervínculo <http://www.desarrolloweb.com> entonces la cadena de texto contendrá "<a href='http://www.desarrolloweb.com'>http://www.desarrolloweb.com</a>", por lo que se podrá escribir cualquier tipo de contenido dentro de la capa.

El siguiente ejemplo, contiene un formulario con una caja de texto, el usuario puede escribir cualquier texto ó código HTML, y al pulsar el botón, este se mostrará dentro de la capa.

La función recibe por parámetro el nombre de una capa, y el formulario que contiene la caja de texto, guarda el valor del cuadro de texto dentro de una variable y a continuación se la asigna a la propiedad innerHTML de la capa:

```
function escribeCapa(capa,formulario){
texto=formulario.caja.value;
document.getElementById(capa).innerHTML=texto;
}
```

#### **Ejemplo completo:**

```
<html>
<head>
<title>Untitled</title>
<script language="JavaScript">
function escribeCapa(capa,formulario){
texto=formulario.caja.value;
document.getElementById(capa).innerHTML=texto;
}
</script>
</head>
<body>
<div id="micapa" style="position:absolute;width:100;height:100;top:100;left:100;background-color:yellow"> </div>

<form name="miformulario" action="#">
Texto: <input type="text" name="caja" size="50"> <input type="button" onclick="escribeCapa('micapa',this.form)"
value="escribir">
</form>
</body>
</html>
```

[Ver ejemplo en una página a parte.](#)

*Artículo por **Carlos Luis Cuenca***

## **Como iluminar tablas, celdas o filas**

**Paso 1:** poner este script en el head <head> xxx </head>

```
<script>
function uno(src,color_entrada) {
src.bgColor=color_entrada;src.style.cursor="hand";
}
```

```
function dos(src,color_default) {  
    src.bgColor=color_default;src.style.cursor="default";  
}  
</script>
```

**Paso 2:** Si lo que quieres es que se ilumine una celda observa el siguiente código:

```
<table border="0" cellspacing="0" cellpadding="0" bgcolor="#000000" align="center">  
<tr>  
<td>  
<table border="0" cellspacing="1" cellpadding="0" align="center" width="278">  
<tr bgcolor="#FFFFFF">  
<td onmouseover="uno(this,'cccccc');" onmouseout="dos(this,'FFFFFF');" align="center" width="100"  
valign="middle"><font face="Arial, Helvetica, sans-serif" size="1">  
PASA POR ENCIMA</font></td>  
<td width="95"> </td>  
<td width="83"> </td>  
</tr>  
<tr bgcolor="#FFFFFF">  
<td width="100"> </td>  
<td width="95"> </td>  
<td width="83"> </td>  
</tr>  
<tr bgcolor="#FFFFFF">  
<td width="100"> </td>  
<td width="95"> </td>  
<td width="83"> </td>  
</tr>  
</table>  
</td>  
</tr>  
</table>
```

PASA POR ENCIMA

---

**Paso 3:** Si lo que quieres es que se ilumine una fila observa el siguiente código:

```
<table border="0" cellspacing="0" cellpadding="0" bgcolor="#000000" align="center" width="317">  
<tr>  
<td>  
<table border="0" cellspacing="1" cellpadding="0" align="center" width="325">  
<tr onmouseover="uno(this,'cccccc');" onmouseout="dos(this,'FFFFFF');" bgcolor="#FFFFFF">  
<td align="center" width="108" valign="middle" height="17">  
<font face="Arial, Helvetica, sans-serif" size="1">PASA POR ENCIMA</font></td>  
<td width="114" height="17">  
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASA POR ENCIMA</font></div>  
</td>  
<td width="99" height="17">  
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASA POR ENCIMA</font></div>  
</td>  
</tr>  
<tr bgcolor="#FFFFFF">  
<td width="108"> </td>  
<td width="114"> </td>  
<td width="99"> </td>  
</tr>  
<tr bgcolor="#FFFFFF">  
<td width="108"> </td>  
<td width="114"> </td>  
<td width="99"> </td>  
</tr>
```

```
</table>
</td>
</tr>
</table>
```

PASA POR ENCIMA PASA POR ENCIMA PASA POR ENCIMA

**Paso 3:** Si lo que quieres es que se ilumine una tabla completa observa el siguiente código:

```
<table border="0" cellspacing="1" cellpadding="0" bgcolor="#000000" align="center" width="317">
<tr>
<td>
<table border="0" cellspacing="1" cellpadding="0" align="center" width="325" height="62"
onMouseOver="uno(this,'cccc');" onMouseOut="dos(this,'FFFFF');" bgcolor="#FFFFFF">
<tr>
<td align="center" width="108" valign="middle" height="17">
<font face="Arial, Helvetica, sans-serif" size="1">PASA POR ENCIMA</font></td>
<td width="114" height="17">
<div align="center"><font face="Arial, Helvetica, sans-serif" size="1">PASA POR ENCIMA</font></div>
</td>
<td width="99" height="17">
<div align="center"><font face="Arial, Helvetica, sans-serif" size="1">PASA POR ENCIMA</font></div>
</td>
</tr>
<tr>
<td width="108">
<div align="center"><font face="Arial, Helvetica, sans-serif" size="1">PASA POR ENCIMA</font></div>
</td>
<td width="114">
<div align="center"><font face="Arial, Helvetica, sans-serif" size="1">PASA POR ENCIMA</font></div>
</td>
<td width="99">
<div align="center"><font face="Arial, Helvetica, sans-serif" size="1">PASA POR ENCIMA</font></div>
</td>
</tr>
<tr>
<td width="108">
<div align="center"><font face="Arial, Helvetica, sans-serif" size="1">PASA POR ENCIMA</font></div>
</td>
<td width="114">
<div align="center"><font face="Arial, Helvetica, sans-serif" size="1">PASA POR ENCIMA</font></div>
</td>
<td width="99">
<div align="center"><font face="Arial, Helvetica, sans-serif" size="1">PASA POR ENCIMA</font></div>
</td>
</tr>
</table>
</td>
</tr>
</table>
```

PASA POR ENCIMA	PASA POR ENCIMA	PASA POR ENCIMA
PASA POR ENCIMA	PASA POR ENCIMA	PASA POR ENCIMA
PASA POR ENCIMA	PASA POR ENCIMA	PASA POR ENCIMA

**Paso 3:** Si lo que quieres es que se iluminen los bordes de una tabla observa el siguiente código:

```
<table border="0" cellspacing="0" cellpadding="0" bgcolor="#CCCCCC" align="center" width="317">
<tr>
<td>
```



```
<table onMouseOver="uno(this,'000000');" onMouseOut="dos(this,'CCCCC');" border="0" cellspacing="1"
cellpadding="0" align="center" width="325" height="60">
<tr bgcolor="#FFFFFF">
<td align="center" width="108" valign="middle" height="17">
<font face="Arial, Helvetica,sans-serif" size="1">PASA POR ENCIMA</font></td>
<td width="114" height="17">
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASA POR ENCIMA</font></div>
</td>
<td width="99" height="17">
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASA POR ENCIMA</font></div>
</td>
</tr>
<tr bgcolor="#FFFFFF">
<td width="108">
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASA POR ENCIMA</font></div>
</td>
<td width="114">
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASA POR ENCIMA</font></div>
</td>
<td width="99">
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASA POR ENCIMA</font></div>
</td>
</tr>
<tr bgcolor="#FFFFFF">
<td width="108">
<div align="center"><font face="Arial, Helvetica, sans-serif" size="1">PASA POR ENCIMA</font></div>
</td>
<td width="114">
<div align="center"><font face="Arial, Helvetica, sans-serif" size="1">PASA POR ENCIMA</font></div>
</td>
<td width="99">
<div align="center"><font face="Arial,Helvetica, sans-serif" size="1">PASA POR ENCIMA</font></div>
</td>
</tr>
</table>
</td>
</tr>
</table>
```

PASA POR ENCIMA	PASA POR ENCIMA	PASA POR ENCIMA
PASA POR ENCIMA	PASA POR ENCIMA	PASA POR ENCIMA
PASA POR ENCIMA	PASA POR ENCIMA	PASA POR ENCIMA

Artículo por **Fabio Núñez**

## Inhibir radio button con Javascript

En este taller vamos a ver cómo se puede deshabilitar un elemento de formulario de tipo radio button. Dicho de otra manera, vamos a ver la manera de hacer que, al pulsar un campo tipo radio de un formulario, no cambie la opción escogida por defecto en el código HTML de la página.

Se puede ver el ejemplo en marcha bajo estas líneas. Observad que, al pulsar los radio button, no cambia la opción escogida inicialmente, que es la primera.

- ☐ Hola ponte encima!
- ☐ Ponte aquí tambien
- ☐ Ponte aquí tambien

## Detalles previos

Los campos de formulario tipo radio se manejan como un grupo. En la jerarquía de objetos del navegador quedan debajo del objeto form y dentro de un array que toma el nombre asignado al grupo de botones. Se puede ver esta formación en el artículo [Control de botones de radio en Javascript](#).

## Solución

La manera de solucionar este asunto que hemos implementado nosotros es definir una variable con el índice del array del botón de radio que tiene que estar seleccionado. Además tendremos una función que se llamará al hacer clic en cualquier botón de radio que se encargará de seleccionar el botón de radio por defecto, de este modo, aunque seleccionemos otro elemento del conjunto, se seleccionará automáticamente el elemento marcado por defecto. Además, la función recibirá el índice del botón de radio pulsado y le retirará el foco de la aplicación de dicho elemento.

Podemos ver el código a continuación.

```
<html>
<head>
  <title>Ejemplo para deshabilitar radio butons</title>

  <script>
    indice_marcado = 0
    function deshabilitar(formulario,idradio){
      formulario.miradio[indice_marcado].checked = true
      formulario.miradio[idradio].blur()
    }
  </script>
</head>

<body>
<h1>Ejemplo para deshabilitar radio butons</h1>

<form name="f1">
<input type="radio" name="miradio" value="Lo que sea" onclick="deshabilitar(this.form,0)" checked> Hola ponte
encima!
<br>
<input type="radio" name="miradio" value="otra cosa" onclick="deshabilitar(this.form,1)"> Ponte aquí tambien
<br>
<input type="radio" name="miradio" value="más cosas" onclick="deshabilitar(this.form,2)"> Ponte aquí tambien
</form>

</body>
</html>
```

En los elementos de formulario de tipo radio button tenemos el manejador de eventos onclick que se llama cuando se hace clic en un ese botón de radio. Dicho manejador de eventos llama a una función pasándole el formulario donde estamos trabajando y el índice del botón de radio actual, que empieza en cero.

La función deshabilitar(), definida en el bloque de script de la cabecera, contiene dos sentencias. La primera vuelve a poner la selección en el botón de radio adecuado, utilizando la propiedad checked del radiobutton, y la segunda retira el foco del elemento pulsado, con el método blur().

Podemos [ver el ejemplo en marcha en una página aparte](#).

Artículo por **Miguel Angel Alvarez**

## Actualizar dos frames con un solo enlace

Con lo que sabemos ya sobre el control de frames podemos realizar un ejemplo para afianzar los conocimientos. Se trata de un ejercicio muy sencillo para conseguir que, al pulsar un solo enlace, se actualice la página contenida en dos frames distintos.

Como un enlace sólo sirve para actualizar el contenido de un frame, necesitaremos ejecutar unas sentencias javascript que sí nos permitan actualizar dos frames a la vez.

Si no se entiende el objetivo perseguido en este ejemplo, podemos [verlo en marcha en una página a parte](#).

**Referencia:** Para entender este ejercicio deberíamos conocer [cómo se hace referencia con Javascript desde un frame a otro](#).

Empecemos viendo la declaración de frames, que no tiene ninguna complicación pues es simplemente un código HTML que pudimos aprender a programar en los [artículos dedicados a frames del manual de HTML](#).

```
<html>
<head>
  <title>Ejemplo de frames numero 1</title>
</head>
<frameset rows="50%,*">
  <frame name="frame1" src="pagina1.html" marginwidth="10" marginheight="10" scrolling="auto"
frameborder="0">
  <frame name="frame2" src="pagina2.html" marginwidth="10" marginheight="10" scrolling="auto"
frameborder="0">
</frameset>
</html>
```

Ahora veamos el código del primero de los frames, que es el que tiene la función Javascript para controlar los frames.

```
<html>
<head>
  <title>Pagina 1</title>
  <script language="JavaScript">
function actualiza_2_frames(){
  window.parent.frames[1].location="http://www.google.com"
  window.location="http://www.yahoo.es"
}
</script>
</head>
<body bgcolor="#ff9999">
<br>
<br>
<a href="javascript:actualiza_2_frames()">Actualiza dos frames con un solo enlace</a>
</body>
</html>
```

Al pulsar el enlace se llama a una función, colocada en la cabecera de la página, por comodidad y para evitar tener que escribir varias sentencias en el atributo href del enlace.

La función, donde verdaderamente está el cogollo de este ejercicio, es extremadamente simple. La primera sentencia accede al frame colocado en segundo lugar (que tiene el índice 1) y actualiza su propiedad location, que es la URL de la página que se está visualizando. En este caso coloca la web de Google en dicho marco, aunque es indiferente lo que deseemos colocar y podríamos haber situado una dirección con un camino relativo al documento actual.

En la segunda sentencia accedemos directamente a la propiedad location del objeto window, porque deseamos actualizar el mismo frame donde está colocado el script. Podríamos haber utilizado una sentencia como la siguiente:

```
window.parent.frames[0].location=" http://www.yahoo.es "
```

Pero en este caso no es necesario acceder a la declaración de frames y luego al frame 0 porque, como decía, estamos ya en él.

Por último veamos el código del segundo frame, que no tiene nada de especial.

```
<html>
<head>
  <title>Pagina 1</title>
</head>
<body bgcolor="#9999ff">
<br>
<br>
Este es el cuerpo del frame 2, que tiene índice 1 en el vector de frames
</body>
</html>
```

Se puede [ver el ejemplo en marcha en una página a parte](#).

*Artículo por **Miguel Angel Alvarez***

## Calcular la edad en Javascript

En este artículo vamos a explicar una función que calcula la edad de una persona. Para ello recibe un string con la fecha de nacimiento de la persona y devuelve el número de años que tiene. Estamos ante un ejercicio que ilustra muy bien el trabajo con fechas en Javascript.

**Referencia:** Para aprender algo que nos sirva de base en el cálculo de fechas sería interesante leer el artículo [Clase Date en Javascript](#).

### El método de trabajo

Nosotros estamos pensando en recibir una fecha en formato español: algo como "12/10/1975", de tipo string. Lo primero será separar los distintos valores de año, mes, día. Para ello utilizamos el método split(), que pertenece a la clase String (tipo de la fecha que vamos a recibir), que devuelve un array con el valor de cada una de las partes de la cadena, utilizando como separador el carácter "/". Después de la separación, en el array devuelto, deberíamos tener tres casillas, donde la primera (la de índice 0) guardará el día, la segunda el mes y la tercera el año.

**Referencia:** Los métodos de la clase String se pueden ver en el artículo [Clase String en Javascript](#).

Vamos a realizar seguidamente algunas comprobaciones para asegurarnos que la fecha es correcta, es decir, que tenemos un valor numérico como día, otro como mes y otro como año. Si no es así devolveremos false, que debería interpretarse como que la función es incapaz de calcular la edad, porque la fecha de nacimiento pasada no es correcta.

A continuación restaremos el número de años de la fecha actual, que podrían ser 2003, con el número de año de la fecha de nacimiento, que será algo como 1975. En este caso nos daría 28, pero nosotros vamos a considerar 27, pues no sabemos si la supuesta persona ha cumplido años en el año en curso, o no. Es decir, hoy que es junio, si cumplió los años en marzo, esa persona ya tendría 28 años, pero si cumple los años en agosto, tendría ahora 27 años.

Así que nuestro próximo paso será saber en qué mes cumplió años la persona y de ello podríamos tener tres casos.

1. Si el mes actual es menor que el mes de nacimiento. Entonces es que no ha cumplido años todavía en este periodo anual. (Los años, en el ejemplo anterior, serían 27)
2. Si el mes actual es mayor que el mes de nacimiento, querría decir que esa persona sí ha celebrado su cumpleaños este año. (Los años, en el ejemplo anterior, serían 28)
3. Si los dos meses son iguales, deberíamos fijarnos en el día, de una manera similar a como se ha realizado para los meses:
  1. Si el día actual es menor que el día de nacimiento, es que le faltan unos días todavía para su cumpleaños (Dado el ejemplo anterior, los años serían 27).
  2. Si el día actual es mayor o igual que el día de nacimiento es que sí ha cumplido años (Dado el ejemplo anterior, los años serían 28).

## El script para calcular la edad

Bueno, con estas explicaciones esperamos que cualquiera con un nivel medio de Javascript pudiera realizar el código de esta función, pero el objetivo es mostraros nuestra propuesta de código, que está comentada para que se pueda entender fácilmente.

```
//calcular la edad de una persona
//recibe la fecha como un string en formato español
//devuelve un entero con la edad. Devuelve false en caso de que la fecha sea incorrecta o mayor que el día actual
function calcular_edad(fecha){

    //calculo la fecha de hoy
    hoy=new Date()
    //alert(hoy)

    //calculo la fecha que recibo
    //La descompongo en un array
    var array_fecha = fecha.split("/")
    //si el array no tiene tres partes, la fecha es incorrecta
    if (array_fecha.length!=3)
        return false

    //compruebo que los año, mes, día son correctos
    var año
    año = parseInt(array_fecha[2]);
    if (isNaN(año))
        return false

    var mes
    mes = parseInt(array_fecha[1]);
```

```
if (isNaN(mes))
    return false

var dia
dia = parseInt(array_fecha[0]);
if (isNaN(dia))
    return false

//si el año de la fecha que recibo solo tiene 2 cifras hay que cambiarlo a 4
if (ano<=99)
    ano +=1900

//resto los años de las dos fechas
edad=hoy.getYear()- ano - 1; //-1 porque no se si ha cumplido años ya este año

//si resto los meses y me da menor que 0 entonces no ha cumplido años. Si da mayor si ha cumplido
if (hoy.getMonth() + 1 - mes < 0) //+ 1 porque los meses empiezan en 0
    return edad
if (hoy.getMonth() + 1 - mes > 0)
    return edad+1

//entonces es que eran iguales. miro los dias
//si resto los días y me da menor que 0 entonces no ha cumplido años. Si da mayor o igual si ha cumplido
if (hoy.getUTCDate() - dia >= 0)
    return edad + 1

return edad
}
```

**Nota:** Para entender esta función será necesario saber que, cuando se ejecuta return dentro de una función, se devuelve el valor indicado y se sale de la función, sin que se puedan ejecutar otras sentencias que existan debajo del return.

*Artículo por **Miguel Angel Alvarez***

## ***Iluminar formularios con CSS y Javascript***

El leer el artículo sobre [iluminación de tablas, celdas, filas](#) me parece muy interesante y necesario en algunas labores de programación de páginas webs. Interesado en dicho artículo y con ayuda de [Programación en Javascript II](#), que también se publica en este site, se muestra este pequeño trabajo.

### **Paso 1:**

Los colores de fondo de las cajas de texto y algunas otras propiedades se manipulan más a menudo utilizando hojas de estilo. En este primer paso lo hacemos bastante simple. Poner este script en el head <head> xxx </head>

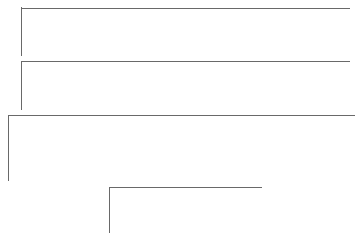
```
<script>
function form_uno(num_form,num_elem_form,color_entrada) {
document.forms[num_form].elements[num_elem_form].style.backgroundColor=color_entrada;
document.forms[num_form].elements[num_elem_form].focus();
}
function form_dos(num_form,num_elem_form,color_default) {
document.forms[num_form].elements[num_elem_form].style.backgroundColor=color_default;
}
}</script>
```

## Paso 2:

Agregamos el código de la siguiente manera:

```
<form method="post" action="" name="miformulario">
<p>
<input type="text" name="campo1" onMouseOver="form_uno(0,0,'Lavender');" onMouseOut="form_dos(0,0,'ffffff');"
class="cajon" >
<br>
<input type="text" name="campo2" onMouseOver="form_uno(0,1,'red');" onMouseOut="form_dos(0,1,'ffffff');"
class="cajon">
<br>
<textarea name="campo3" onMouseOver="form_uno(0,2,'blue');" onMouseOut="form_dos(0,2,'ffffff');"
class="cajon"></textarea>
<br>
<select name="campo4" onMouseOver="form_uno(0,3,'cccccc');" onMouseOut="form_dos(0,3,'ffffff');"
class="cajon">
<option value="1">uno</option>
<option value="2">dos</option>
<option value="3">tres</option>
<option value="4">cuatro</option>
<option value="5">cinco</option>
</select>
</form>
```

El resultado es el siguiente:



Hasta este punto vemos que nuestro formulario se ve bastante convencional, para mejorar un poco nuestra presentación se debe incluir código de estilo, como el que sigue:

## Paso 3:

Poner este script en el head <head> xxx </head>:

```
<style type="text/css">
<!--
.cajon {
PADDING-RIGHT: 0.1em; PADDING-LEFT: 0.1em; PADDING-BOTTOM: 0.1em; FONT: 9pt "Verdana, Tahoma, Arial";
MARGIN-LEFT: 0.1em; PADDING-TOP: 0.1em; BACKGROUND-COLOR: #FFFFFF0; TEXT-ALIGN: left
}
.boton {
BORDER-RIGHT: #000000 1px solid; BORDER-TOP: #000000 1px solid; FONT-SIZE: 11px; BACKGROUND: #FFFFFF0;
BORDER-LEFT: #000000 1px solid; COLOR: #000000; BORDER-BOTTOM: #000000 1px solid; FONT-STYLE: normal;
FONT-FAMILY: verdana, arial, "trebuchet MS", helvetica, sans-serif
}
-->
</style>
```

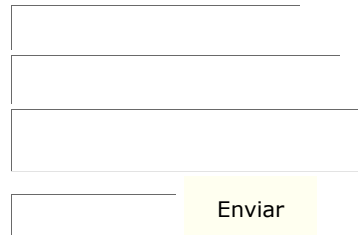
## Paso 4:

Lo que queda solamente es hacer el enlace de los objetos del formulario con el respectivo estilo que deseamos. Entonces el código quedaría así como sigue y el resultado se ve en el siguiente cuadro.

```
<form method="post" action="" name="miformulario2">
<input type="text" name="campo12" onMouseOver="form_uno(1,0,'Lavender');" >
```

```
onMouseOut="form_dos(1,0,'FFFFFF0');" class="cajon" >
<br>
<input type="text" name="campo22" onMouseOver="form_uno(1,1,'red');"
onMouseOut="form_dos(1,1,'FFFFFF0');">
<br>
<textarea name="textarea" onMouseOver="form_uno(1,2,'blue');"
onMouseOut="form_dos(1,2,'FFFFFF0');" class="cajon"></textarea>
<br>
<select name="select" onMouseOver="form_uno(1,3,'cccccc');"
onMouseOut="form_dos(1,3,'FFFFFF0');" class="cajon">
<option value="1">uno</option>
<option value="2">dos</option>
<option value="3">tres</option>
<option value="4">cuatro</option>
<option value="5">cinco</option>
</select>
<input type="submit" name="Submit" value="Enviar" class="boton">
</form>
```

El resultado 2 es el siguiente:



Las funciones form\_uno y form2 se explican a continuación:

### **function form\_uno(num\_form,num\_elem\_form,color\_entrada)**

Los 2 primeros parámetros corresponden a números que representan a un formulario (num\_form) puedes fijarte que el primer formulario se llama miformulario y el segundo miformulario2, en vez de emplear estos nombres se emplean valores numéricos, el siguiente parámetro representa un objeto de dicho formulario (num\_elem\_form) y el tercer parámetro representa un valor de un color o su respectivo nombre.

El cuerpo de cada función corresponden a la manipulación de las propiedades de los objetos de los formularios.

**Referencia:** Para entender mejor parte del código java script se recomienda leer el manual [Programación en Javascript II](#).  
 Definitivamente este código se puede mejorar y un pequeño ejemplo de esto lo pueden observar en: <http://www.ayacuchoonlinea.com/copechi/inscripcion.html>

*Artículo por **Richar Mendoza***

## **Autozoom de texto con Javascript**

Este script hace que el tamaño de la letra de un banner de texto vaya creciendo hasta alcanzar un valor máximo predefinido.



El funcionamiento es bien sencillo, como cualquier animación en Javascript, necesitas un temporizador que cada cierto tiempo se dispara llamando a una función responsable del efecto de animación, en este caso modificar el tamaño de un texto. Esta función se llama `creceLetra()`.

Al cargar la página se llama a la función `resetear()` que es la que inicia todos los valores: objeto al que se aplica el efecto (un elemento DIV), tamaño mínimo y máximo del texto y velocidad con la que crecerá. Esta función activa el temporizador que llama a `creceLetra()`, encargada de aumentar en 1 px el tamaño del texto actuando sobre la propiedad `style.fontSize` del objeto al que se aplica el efecto, y de reactivar el temporizador.

Así hasta que el texto alcance el tamaño máximo en cuyo caso vuelve a llamar a `resetear()` (esta vez sin argumentos) que repone el tamaño inicial y se repite el ciclo. Esta llamada se hace mediante un temporizador para mantener en pantalla el texto a tamaño grande durante un cierto tiempo (500 miliseg en el ejemplo).

## El código Javascript

```
<script language="JavaScript">
var banZoom = null

function creceLetra()
{
var obj = banZoom
var tma
tma = parseInt(obj.style.fontSize)
window.status = obj.style.fontSize
if (tma<obj.maxTam)
{
obj.style.fontSize = tma + 1
setTimeout("creceLetra("+obj.maxTam+")",20)
}
else
setTimeout("resetear()",500)
}

function resetear(mn, mx, rapidez, idBan)
{
if (banZoom == null)
{
banZoom = document.getElementById(idBan)
banZoom.maxTam = mx
banZoom.minTam = mn
banZoom.rapidez = rapidez
}
banZoom.style.fontSize = banZoom.minTam
setTimeout("creceLetra()",rapidez)
}
</script>
```

## El código HTML

El evento `onload` se vincula a `resetear()` con los argumentos iniciales que son, por este orden, tamaño mínimo de las letras, tamaño máximo que deben alcanzar, rapidez con que crecerán y el identificador del elemento al que se aplica.

En este caso este elemento es un bloque DIV cuyo identificador ID es 'letras'. Este bloque DIV

puedes diseñarlo como quieras en cuanto a colores, tipo de letra a usar, bordes, imagen de fondo...

```
<body onload="resetear(10, 48, 10, 'letras')">
<DIV id="letras" style="position:absolute; font-size:10px; height:56px; width: 761px; background-color:
#CCFFCC;border: 1px none #000000;"> Toda la frase va creciendo</DIV>
<p> </p>
<p> </p>
<p>Y qué pasa con el resto </p>
</body>
```

El ejercicio se puede [ver en marcha en una página aparte](#).

*Artículo por **Juan Carlos Gámez***

## **JavaScript para evitar que la página se muestre en un frame**

Existe una utilidad muy sencilla sobre el control de frames en Javascript que también resulta muy útil para cualquier sitio web. Se trata de evitar que nuestra página se muestre dentro de cualquier división de frames y puede ser muy interesante para evitar que un enlace de cualquier sitio web introduzca nuestra página dentro de su diseño o estructura de menús.

Puede que aparecer dentro de un frame en muchos casos no nos importe demasiado, pero reducen el espacio para mostrar nuestra propia página y la encorsetan en un diseño que no tiene porque hacerle ningún bien.

### **El script**

Tan solo una línea de código es suficiente para crear este efecto. Esta línea se puede poner en cualquier parte del documento HTML, aunque sería recomendable que quedase por la parte superior o dentro de la cabecera, para que tenga que cargarse la página entera para expandirse a todo el espacio de la ventana.

```
<script language="JavaScript">
<!--// evito que se cargue en otro frame
if (top.location != self.location)top.location = self.location;
//-->
</script>
```

En este script se comprueba si las propiedades top.location, que hace referencia a la URL de la declaración de frames, en caso de que hubiera y self.location, que hace referencia a la URL del documento donde está el script.

Si las dos URLs son iguales significaría que la página no está cargada dentro de un frame y si son distintas querría decir que sí esta mostrándose en el espacio de un marco.

En caso de que sean distintas, simplemente se indica que en la ventana del navegador al completo (top.location) se muestre la URL de la página donde está el script (self.location).

No tiene muchas complicaciones. Se puede [ver en una página aparte](#).

Artículo por **Miguel Angel Alvarez**

## Elementos de formulario select asociados

Vamos a conocer uno de los trucos más solicitados de Javascript, que tiene mucha relación con el tema de formularios y donde se utiliza el evento onchange de Javascript. Es un ejemplo sobre cómo realizar una página con un par de selects donde, según el valor escogido en uno de ellos, cambien las opciones posibles del otro select.

Lo mejor para ver lo que vamos a hacer es ver una [página web donde se muestra en funcionamiento el script](#). Para ver su funcionamiento debemos cambiar la selección del primer select y comprobaremos como las opciones del segundo select cambian automáticamente.

El ejemplo que hemos ilustrado utiliza provincias y países. Al escoger en el primer select un país, en el segundo debe mostrarnos las provincias de ese país para que escojamos una provincia, pero sólo una que tenga que esté en el país seleccionado en primer término.

### Conocer el objeto select y los option

Es importante conocer los objetos de formulario select y los option. Los select corresponden con las cajas de selección desplegables y los option con cada una de las opciones de la caja desplegable. Podemos [ver un artículo que habla de ello](#).

En concreto nos interesa hacer varias cosas que tienen que ver con extraer el valor de un select en cualquier momento, fijar su número de opciones y, para cada opción, colocar su valor y su texto asociado. Todo esto aprenderemos a hacerlo en este ejemplo.

**Referencia:** Para conocer el trabajo con formularios y la jerarquía de objetos javascript (Todo eso es la base del trabajo con los elementos de las páginas en Javascript) debemos haber leer el [manual de Javascript II](#).

### Modo de llevar a cabo el problema

Para empezar, vamos a utilizar un formulario con dos selects, uno para el país y otro para la provincia.

```
<form name="f1">
<select name=pais onchange="cambia_provincia()">
<option value="0" selected>Seleccione...
<option value="1">España
<option value="2">Argentina
<option value="3">Colombia
<option value="4">Francia
</select>

<select name=provincia>
<option value="-">-
</select>
</form>
```

Nos fijamos en el select asociado al país de este formulario que, cuando se cambia la opción de país, se debe llamar a la función cambia\_provincia(). Veremos más adelante esta función,

ahora es importante fijarse que está asociada al evento onchange que se activa cuando cambia la opción en el select.

Todo lo demás será código Javascript. Empezamos definiendo un montón de arrays con las provincias de cada país. En este caso tenemos sólo 4 países, entonces necesitare 4 arrays. En cada array tengo la lista de provincias de cada país, colocada en cada uno de los elementos del array. Además, dejaré una primera casilla con un valor "-" que indica que no se ha seleccionado ninguna provincia.

```
var provincias_1=new Array("-", "Andalucía", "Asturias", "Balears", "Canarias", "Castilla y León", "Castilla-La Mancha", "...")
var provincias_2=new Array("-", "Salta", "San Juan", "San Luis", "La Rioja", "La Pampa", "...")
var provincias_3=new Array("-", "Cali", "Santamarta", "Medellin", "Cartagena", "...")
var provincias_4=new Array("-", "Aisne", "Creuse", "Dordogne", "Essonne", "Gironde", "...")
```

Hay que fijarse que los índices del array de cada país se corresponden con los del select del país. Por ejemplo, la opción España, tiene el valor asociado 1 y el array con las provincias de España se llama provincias\_1.

El script se completa con una función que realiza la carga de las provincias en el segundo select. El mecanismo realiza básicamente estas acciones:

- Detecto el país que se ha seleccionado
- Si el valor del país no es 0 (el valor 0 es cuando no se ha seleccionado país)
  - Tomo el array de provincias adecuado, utilizando el índice del país.
  - Marco el número de opciones que debe tener el select de provincias
  - Para cada opción del select, coloco su valor y texto asociado, que se hace corresponder con lo indicado en el array de provincias.
- SI NO (El valor de país es 0, no se ha seleccionado país)
  - Coloco en el select de provincia un único option con el valor "-", que significaba que no había provincia.
- Coloco la opción primera del select de provincia como la seleccionada.

La función tiene el siguiente código. Está comentado para que se pueda entender mejor.

```
function cambia_provincia(){
    //tomo el valor del select del pais elegido
    var pais
    pais = document.f1.pais[document.f1.pais.selectedIndex].value
    //miro a ver si el pais está definido
    if (pais != 0) {
        //si estaba definido, entonces coloco las opciones de la provincia correspondiente.
        //selecciono el array de provincia adecuado
        mis_provincias=eval("provincias_" + pais)
        //calculo el numero de provincias
        num_provincias = mis_provincias.length
        //marco el número de provincias en el select
        document.f1.provincia.length = num_provincias
        //para cada provincia del array, la introduzco en el select
        for(i=0;i<num_provincias;i++){
            document.f1.provincia.options[i].value=mis_provincias[i]
            document.f1.provincia.options[i].text=mis_provincias[i]
        }
    }else{
        //si no había provincia seleccionada, elimino las provincias del select
        document.f1.provincia.length = 1
        //coloco un guión en la única opción que he dejado
        document.f1.provincia.options[0].value = "-"
        document.f1.provincia.options[0].text = "-"
    }
}
```

```
}  
//marco como seleccionada la opción primera de provincia  
document.f1.provincia.options[0].selected = true  
}
```

Podemos [ver una página con el ejemplo en funcionamiento](#).

*Artículo por **Miguel Angel Alvarez***

## Cuenta los caracteres escritos en un textarea

El diseño de este script fue motivado por la necesidad de hacer la típica caja para enviar mensajes SMS desde Internet. El tamaño máximo de un mensaje SMS de móvil es de 160 caracteres, con lo que es muy útil que la propia página te informe sobre el número de caracteres que se llevan escritos en el mensaje, para que el interesado no se pase del máximo permitido.

El funcionamiento es muy sencillo de entender, pero en cualquier caso podemos [ver el script en marcha](#) para saber exactamente qué nos proponemos.

### El formulario

La página presentará un formulario con dos campos. El primero con el textarea donde el usuario escribirá el mensaje y el segundo, un campo de texto donde mostraremos en todo momento los caracteres escritos.

El único detalle a tener en cuenta relacionado con Javascript es el par de eventos que tenemos definidos dentro del campo textarea, que sirven para llamar a la función que realiza la cuenta de los caracteres en el momento que el usuario presiona o suelta las teclas. Concretamente se utiliza el evento `onKeyDown` para definir las acciones a realizar cuando se aprete la tecla y `onKeyUp`, para definir acciones a ejecutar cuando se suelta la tecla apretada.

```
<form action="#" method="post">  
<table>  
<tr>  
  <td>Texto:</td>  
  <td><textarea cols="40" rows="5" name="texto" onKeyDown="cuenta()" onKeyUp="cuenta()"></textarea></td>  
</tr>  
<tr>  
  <td>Caracteres:</td>  
  <td><input type="text" name="caracteres" size="4"></td>  
</tr>  
</table>  
</form>
```

### El script que cuenta caracteres

Con el formulario y el par de eventos introducidos tenemos todo lo necesario para que se cuenten -y recuenten- los caracteres cada vez que el visitante, situado sobre el textarea, pulsa sobre las teclas, es decir, cada vez que se escribe texto en el textarea. Ahora simplemente nos queda definir la función que se encarga de realizar la cuenta propiamente dicha y situarla en el otro campo de texto del formulario.

```
<script>
function cuenta(){
    document.forms[0].caracteres.value=document.forms[0].texto.value.length
}
</script>
```

Puede que a muchos haya sorprendido la sencillez del script, pero es que no hace falta más.

La propiedad value del textarea contiene el texto escrito y a su vez, la propiedad length guarda el número de caracteres de dicho texto. Así, document.forms[0].texto.value.length equivale al número de caracteres introducidos dentro del textarea. Este valor se asigna al contenido del campo de texto del formulario donde guardamos el número de caracteres, mediante la propiedad value del campo: document.forms[0].caracteres.value.

Con todo ello, se mostrará en el campo de texto el número de caracteres del textarea. Se puede [ver el ejemplo en una página aparte](#).

*Artículo por **Miguel Angel Alvarez***

## Otro navegador con capas

Desde hace tiempo he estado buscando la forma de crear menús dinámicos como los que ustedes utilizan en su encabezado, y he encontrado herramientas, incluso la que ustedes recomiendan y que creo usan "[sothink](#)", que son buenas pero no gratis. Gracias a [este artículo](#) se me ocurrió que con esto podría al fin crear mis propios menús a mi estilo, así que lo intenté y lo logré. Aquí les presento las modificaciones al código original:

**Referencia:** Este artículo amplía las funcionalidades de otro sistema de barra de navegación por capas que estaba publicado en DesarrolloWeb.com en esta dirección:  
<http://www.desarrolloweb.com/articulos/12.php>

Tomen en cuenta que los submenús pueden tener tablas, imágenes, links, etc...

```
<html>
<head>
<title>Navegador con capas</title>

<script language="javascript">
var capa1, capa2, capa3
ie4 = (document.all)? true:false

function init() {
    if (ie4) {
        capa1 = descripcion1.style
        capa2 = descripcion2.style
        capa3 = descripcion3.style
    }
}
function oculta() {
    if (descripcion1.style.visibility == "visible")
        descripcion1.style.visibility="hidden"
    else if (descripcion2.style.visibility == "visible")
        descripcion2.style.visibility = "hidden"
    else if (descripcion3.style.visibility == "visible")
```

```
descripcion3.style.visibility = "hidden"
}
function muestra(obj, sup){
oculta()
if (ie4)
{obj.visibility = "visible"
obj.top=sup}
}
</script>
<style type="text/css">
.submenu {
position:Absolute;left:110; top:10; background-color:#add8e6; visibility:hidden;width:250px;height:58px;font-size:
8pt; font-family: arial,verdana,helvetica; color: #333399;padding:5px
}
.menu {
font-size: 10pt; font-family: arial,verdana,helvetica; font-weight: bold; color: #333399; position: absolute; height:
58; width: 100; background-color: #ffffcc; text-align: right;top: 10px;left: 10px
}
</style>
</head>
<body onLoad="init()" link="#333399" vlink="#333399">

<div align="" id="enlaces" class="menu">
<a href="#" OnMouseOver="muestra(capa1, 10)" onmouseout="oculta()">Mis Amigos</a>
<br>
<a href="#" OnMouseOver="muestra(capa2, 30)" onmouseout="oculta()">Mi pueblo</a>
<br>
<a href="#" OnMouseOver="muestra(capa3, 45)" onmouseout="oculta()">Enlaces</a>
</div>

<div id="descripcion1" OnMouseOver="muestra(capa1, 10)" onmouseout="oculta()" class="submenu">
<a href="#" >Lo que deseas1</a><br>
<a href="#" >Lo que deseas2</a><br>
<a href="#" >Lo que deseas3</a><br>
</div>
<div id="descripcion2" OnMouseOver="muestra(capa2, 30)" onmouseout="oculta()" class="submenu">
<a href="#" >El pueblo mas turistico1</a><br>
<a href="#" >El pueblo mas turistico2</a><br>
<a href="#" >El pueblo mas turistico3</a><br>
</div>
<div id="descripcion3" OnMouseOver="muestra(capa3, 45)" onmouseout="oculta()" class="submenu">
<a href="#" >Todos los Enlaces1</a><br>
<a href="#" >Todos los Enlaces2</a><br>
<a href="#" >Todos los Enlaces3</a><br>
</div>

</body>
</html>
```

Como no tengo Netscape sólo hice el código para IE (pero supongo que debe funcionar de igual forma). Por otro lado, al código le hace falta crear una matriz de objetos para ocultar los menús dentro de un bucle, y no uno por uno en una estructura IF. Así que si alguien tiene ideas háganlas llegar comentando este artículo o mandando un email.

*Artículo por **David***

## **Cargador universal de imágenes**

El propósito del script que puede verse/descargarse más abajo tiene como objeto utilizar un

único javascript que nos sirva SIEMPRE y en cualquier página web para cargar las imágenes que se vayan a utilizar. El [ya utilizado](#) en el punto 13 de este manual es válido igualmente, pero está enfocado a las 3 imágenes concretas del ejemplo, guardadas en unas rutas/directorios concretos y que utiliza 3 variables flags que dan por sentado que sólo precargaremos esas 3 imágenes. De esto, deducimos que para adaptar ese script a nuestra Web particular habría que modificar el código sustancialmente, lo que sería bastante engorroso.

Lo que proponemos con el siguiente código es:

1. Transformar el script ya conocido para adaptarlo a un uso global.
2. Con un mínimo esfuerzo indicaremos rutas, nombres de imágenes, Variables y Aspecto de la Precarga de TODAS las imágenes.
3. El ejemplo concreto está orientado a una página web home.html, por ejemplo, que es donde se precarga todo y que al terminar nos redirige al verdadero inicio de nuestra página inicio.html, que es donde se emplearán todas las imágenes cargadas.

## INICIO DE CODIGO

```
<HTML>
<HEAD><TITLE>Cargando Imagenes</TITLE>
<STYLE TYPE="text/css">
DIV { background-color: black; color: white; border: 2px solid red; width: 30% }
</STYLE>
<SCRIPT TYPE="text/javascript">
//Script por José.A Torres
//Majadahonda 2003

miArray = new Array ("imagenes/ElMenu_over00.jpg", "imagenes/foto1.bmp", "imagenes/foto2.bmp",
"imagenes/foto3.bmp", "imagenes/pedazo1.jpg", "imagenes/pedazo2.png",
"imagenes/pedazo3.jpg", "imagenes/pedazo5.jpg", "imagenes/pedazo6.png",
"imagenes/pedazo7.png", "imagenes/pedazo8.jpg", "imagenes/pedazo9.jpg");
document.write("IMAGENES BAJANDO A LA CACHE DE SU NAVEGADOR<BR>");
patron = new RegExp("imagenes/");

for (cont=0;cont<miArray.length;cont++) {
document.write("<DIV NAME='capa'>");
document.write("Imagen: " + miArray[cont].replace(patron,""));
document.write("<br>");
document.write("...en cola de Precarga OK<br></DIV>");

eval ("imagen" + cont + " = new Image();");
eval ("imagen" + cont + ".src = " + "\"" + miArray[cont] + "\"");

}
document.write("TODAS LAS IMAGENES YA ESTAN EN PRECARGA");

cont=0;
function pruebaCarga() {

if (eval("imagen"+cont).complete == true) {
window.defaultStatus= "Imagen"+cont+ "/" + miArray.length + " CARGADA";
cont++;
}

if (cont<miArray.length) setTimeout('pruebaCarga()',500);
else {
window.defaultStatus="Carga de imágenes TERMINADA";
window.open("inicio.html", "MenuPrincipal", "fullscreen=yes");
}
```



```
}  
}  
</SCRIPT>  
</HEAD>  
<BODY BGCOLOR="#808AC2" onLoad="pruebaCarga();return true;">  
<!-- cuerpo de la pagina -->  
</BODY>  
</HTML>
```

## FIN DE CODIGO

### Explicación:

En un *Array* metemos las rutas relativas de todas las imágenes a precargar. En este caso, se encuentran todas en la carpeta *Imágenes*, pero no hay problema en que estén en varias distintas mientras se indique en el array la ruta correcta.

El script crea automáticamente las referencias a cada objeto imagen para poder tener acceso a cada objeto y a sus propiedades. La denominación automática es *imagen0, imagen1, imagen2.. [...].imagen+miArray.length-1*

La línea de código

```
patron = RegExp("imagenes/");
```

y más adelante:

```
+ miArray[cont].replace(patron, "")
```

son pura estética. Con ambas indicamos a `document.write` que de los argumentos del Array omita escribir en pantalla la subcadena: "imagenes/" para no mostrar la ruta completa, pues en ciertos casos ésta podría ser bastante larga si tenemos una ruta demasiado larga. Si las rutas fuesen por ejemplo "paginainicio/articulos/electrónica/imágenes/camaras..."

"paginainicio/articulos/electrónica/imágenes/moviles..."

podríamos asignar:

```
patron = RegExp("paginainicio/articulos/electrónica/imágenes/");
```

para omitir en cada imagen cargada esa cadena.

Lo mismo ocurre con las etiquetas `<DIV> ...</DIV>` se han utilizado para que el usuario mediante hojas de Estilo incrustadas o por el atributo `STYLE` de `DIV` defina cómo se van a mostrar las líneas que nos informan del progreso de carga en pantalla: colores, tamaños, fondo, bordes, etc. Otro aspecto meramente de Forma.

Finalmente, cuando el navegador lee la última etiqueta HTML se llama repetidamente a una función `function pruebaCarga()` que comprueba el estado de CADA imagen. Sobre los tiempo de llamada (medio segundo en el ejemplo, o 500 milisegundos) conviene no abusar poniendo valores como 1 milisegundo o en general valores más bajos de 250 o 300 pues podríamos forzar demasiado los recursos del navegador y, según en qué casos, obtener un error de "out of memory".

Cada imagen que se detecta como cargada se reflejará en la barra de estado del navegador (esquina inferior izquierda) y el total de las que hay. Finalmente, cuando todo está cargado la página nos autoconduce con `window.open` a *inicio.html* donde ya podremos hacer libre uso de todas las imágenes.

Podemos comprobar que todo ha funcionado de una forma simple: borramos antes de nada la Caché de nuestro navegador y lo configuramos en "Actualizar la página cada vez que se visita". A continuación ejecutamos la página cargadora de imágenes. Al terminar veremos en

*Herramientas/Opciones de Internet/Configuración/Ver Archivos* (en caso de Internet Explorer) que efectivamente las imágenes han "bajado" correctamente, pese a no advertir el clásico aviso del navegador de "Cargando imagen...", pues todo se hace en segundo plano.

Si volvemos a cargar la página una segunda vez, veremos que en la barra de Status las imágenes se cargan a una gran velocidad, pues se detecta que ya están en la caché y no hay necesidad de volver a cargarlas.

## AVISO IMPORTANTE

Para probar el código **es necesario** "colgar" realmente las páginas en internet, o al menos que las rutas indicadas en el *Array* apunten a cualquier carpeta contenedora de imágenes en internet. Si probamos el ejemplo en "Local", en nuestro propio Pc, con las imágenes también en nuestro ordenador, el script parece no funcionar, pero sólo parece. Esto es así porque el Navegador no almacena las imágenes en su Caché, pues detecta que YA se encuentran en nuestro disco duro y no hay necesidad de "bajarlas". Para que el **Status** "Imagen4/x Cargada" se visualice **en tiempo real**, la ventana debe de tener el foco activo. Si al ejecutar la página la minimizamos, el script seguirá ejecutándose correctamente y precargando, pero sólo al seleccionarla otra vez veremos el nuevo *Status* actualizado.

*Artículo por José Alberto Torres Arroyo*

## Paso de parametros en HTML con client-side Javascript

El script siguiente se colocará "a pelo" en la página que debe de recibir los parametros, o podrá copiarse en un fichero .js vinculado. En ambos casos el código no debe estar en ninguna función, para que se ejecute siempre que se carga la página. Una vez se haya ejecutado el script tendremos los valores recibidos en las correspondientes variables. Aquellas que no hayamos recibido tendrán el valor por defecto que hemos fijado.

El Código:

```
<script language="javascript">
//Autor: Bruno Suárez Laffargue
//Version: 1.1

//Definimos las variables necesarias
variable=unValorPorDefecto;//Habrá que establecerlo

//Capturamos la URL
var callingURL = document.URL;

//Separamos los parametros
var cgiString = callingURL.substring(callingURL.indexOf('?')+1,callingURL.length);

//Fijamos el separador entre parametros
var DELIMITER = '&';

//Eliminamos la almohadilla, si es que existe... cortamos por lo sano!
if (cgiString.indexOf('#')!=-1){
    cgiString=cgiString.slice(0,cgiString.indexOf('#'));
}

//Troceamos el cgiString ya limpiado, separando cada par variable=valor
//en una de las posiciones del array
```

```
var arrayParams=cgiString.split(DELIMITER);

//Recorremos el array de parametros evaluando cada uno de los pares variable=valor
for (var i=0;i<arrayParams.length;i++){
    eval(arrayParams[i].substring(0,arrayParams[i].indexOf('=')+1)+"\""+
    arrayParams[i].substring(arrayParams[i].indexOf('=')+1,arrayParams
    [i].length)+"\"");
}
</script>
```

Una de las restricciones funcionales para que esto funcione es tener tantas variables javascript definidas como parametros se van a recibir, inicializadas a un valor por defecto. Además estas variables han de llamarse exactamente igual que los parametros, ya que si no, no funciona. En ningún caso.

*Artículo por **Bruno Suárez Laffargue***

## **Recuadro dinámico en Javascript con texto que cambia**

Realizamos un sencillo script en Javascript para realizar un cuadro con información dinámica, que cambia con cada impresión de la página. El ejercicio consiste en una tabla HTML que muestra una información que cambia, utilizando Javascript para que nos proporcione un texto aleatorio, que luego imprimiremos dentro de la tabla HTML.

Para empezar vamos a ver cómo obtener un texto aleatorio. La idea a desarrollar es bien simple. Creamos un array con los distintos textos, entre los que se escogerá uno aleatoriamente. Se obtiene un valor numérico aleatorio entre 0 y máximo índice del array y se imprime el texto que hay en el array en esa posición aleatoria.

```
function texto_aleatorio(){
    var textos = new Array()
    textos[0] = "Tenemos los mejores productos del mercado, con controles de calidad intensivos."
    textos[1] = "Distribuimos en todo el mundo con los mejores tiempos de entrega y fiabilidad de los envíos."
    textos[2] = "No tenemos competidores que nos hagan sombra. Contrate con nosotros y compuébelo. Así de fácil."
    textos[3] = "Disponga del mejor servicio de atención al cliente y una respuesta rápida a sus problemas."
    textos[4] = "Los mejores servicios, productos y, como no, los menores precios. Todo son ventajas."

    aleat = Math.random() * (textos.length)
    aleat = Math.floor(aleat)

    document.write(textos[aleat])
}
```

En la primera línea se crea el array y en las siguientes se inicializan sus distintas casillas con textos variados. Tal como hemos hecho el ejercicio, el número de casillas que tenga el array es indiferente, por lo que podríamos aumentar sus casillas, introduciendo nuevas frases, y así las posibilidades de textos serán más variadas.

Más adelante en la función se obtiene un número aleatorio. Para obtenerlo utilizamos la clase Math, concretamente el método random(). Random devuelve un valor decimal aleatorio entre 0 y 1. Algo como 0.453. Si multiplicamos ese valor por el número de casillas del array obtendremos un número entre 0 y el número de casillas, pero todavía tiene valores decimales y nosotros deseamos que sea entero para poder utilizarlo como índice en el array. Para convertir ese valor a entero, lo redondeamos hacia abajo con floor(), que devuelve el número

más próximo, redondeando por abajo.

En la última línea de la función se imprime el valor aleatorio.

El código HTML del recuadro

```
<table width="180" border="0" cellspacing="1" cellpadding="2" bgcolor="000000">
  <tr>
    <td class="barraa" bgcolor="993333"><font color="#FFFFFF"><b>Nuestras ventajas</b></font></td>
  </tr>
  <tr>
    <td class="fuente8" bgcolor="#FFFFFF"><script language=javascript>texto_aleatorio()</script></td>
  </tr>
</table>
```

Es una tabla HTML muy sencilla. Simplemente muestra una celda con el encabezado o titular de la caja y una segunda celda en la que simplemente hay una llamada a la función que escribe el texto aleatorio.

El resultado puede verse en una [página](#) aparte. Tener en cuenta que hay que actualizar la página para ver como van mostrándose distintos mensajes, entre todos los que hemos configurado.

Este ejemplo puede complicarse todo lo que se desee para crear páginas dinámicas que cambien los contenidos entre distintos accesos del visitante. Podríamos hacer que incluyesen diferentes elementos aleatorios, como imágenes, animaciones, enlaces, etc.

*Artículo por **Miguel Angel Alvarez***

## Creación de gráficas de barras con Javascript

Vamos a presentar un sistema para realizar gráficas de barras en páginas web, con la ayuda de Javascript. Sin duda, cualquier persona podrá encontrar varias soluciones para este problema, aunque nosotros vamos a recomendar una que de verdad nos puede simplificar mucho la vida y que no requiere de ningún requisito especial. Simplemente que el navegador del visitante soporte Javascript.

Las gráficas de barras sirven para hacer muchas cosas y es una de las tareas típicas que se puede necesitar en un proyecto un poco avanzado. Generalmente se utilizan para mostrar resultados de informes, estadísticas o temas similares de una manera muy visual.

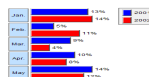


Imagen de una de las gráficas que se pueden generar con este mecanismo.

Con el siguiente ejemplo vamos a aprender a hacer gráficas dinámicamente, es decir, que se construyen con unos datos que pueden ser variables. Si la gráfica fuera siempre a construirse con los mismos datos, podríamos crearla con algún programa como el Excel y luego convertirla en imagen y presentarla dentro de la web. Sin embargo, es muy habitual que los datos se extraigan de una fuente como una base de datos o algo similar, donde pueden variar con el tiempo. En este caso, cada vez que se imprima la página, se debería acceder a la base de

datos, extraer la información y generar la gráfica dinámicamente.

## BAR\_GRAPH

El sistema que presentamos aquí se llama BAR\_GRAPH y sirve para generar gráficas de barras de distintos tipos con la ayuda de Javascript. Las gráficas tienen un aspecto visual muy adecuado y son totalmente personalizables, pudiendo configurar el sistema para que muestre gráficas horizontales, verticales y de progreso. También incluye la posibilidad de crear etiquetas, leyendas, agrupar ciertos valores, utilizar colores distintos, etc.

El producto es gratuito para uso personal, aunque si se utiliza de manera comercial, el autor solicita que se haga una donación de 6 euros, a través de Paypal.

Para la creación de las gráficas no se necesita de ningún material, excepto unas librerías Javascript que son las que ofrecen todas las funcionalidades de creación de gráficas.

Para incluir la librería en nuestras páginas web, basta con descargarla del sitio web del autor del producto: <http://www.gerd-tentler.de/tools/graphs/> y llamarla desde nuestros scripts.

Para incluir una librería Javascript, lo habitual es colocar en la cabecera de la página una llamada al archivo donde está el código. Algo como esto:

```
<head>
<title>Ejemplo de gráficas</title>
<script src="graphs.js" type="text/javascript"></script>
</head>
```

Nos fijamos que se incluye un archivo llamado graphs.js que se supone estará en el mismo directorio que la página web. Dentro de ese archivo encontraremos la declaración de un objeto que será el encargado de recibir los datos de origen con los que se desea construir la gráfica y también de mostrarla cuando se hayan introducido todos los datos iniciales.

Para crear y mostrar una gráfica, lo único que tenemos que hacer es instanciar el objeto gráfica, insertarle los valores iniciales y llamar al método que hace que se muestre la gráfica.

Se puede ver un ejemplo a continuación de las instrucciones iniciales de creación de la gráfica:

```
//instanciamos la gráfica
graph = new BAR_GRAPH("hBar");
//insertamos valores
graph.values = "380,150,260,310,430";
//mostramos la gráfica en la página
document.write(graph.create());
```

Se puede complicar un poco más este sistema para incorporar, por ejemplo, dos tandas de valores, agrupadas en dos grupos.

```
//instanciamos la gráfica
graph = new BAR_GRAPH("hBar");
//definimos las etiquetas que acompañarán a cada pareja de valores
graph.labels = "Jan.,Feb.,Mar.,Apr.,May";
//inicializamos los datos. 1 elemento tiene un grupo de dos valores.
graph.values = "380;420, 150;340, 260;120, 310;250, 430;370";
//definimos una leyenda
```

```
graph.legend = "2001,2002";  
//mostramos la gráfica en la página  
document.write(graph.create());
```

Estos dos ejemplos de gráficas se pueden ver en una [página](#) aparte. Si deseamos ver muchos más ejemplos de gráficas posibles sería muy interesante que accediéramos a la página del producto, donde veremos los tipos de gráficas más habituales que se pueden realizar.

<http://www.gerd-tentler.de/tools/graphs/>

*Artículo por **Miguel Angel Alvarez***

## Juegos en Javascript

Javascript se encaja dentro de los lenguajes para la web del lado del cliente. Es compatible con los navegadores más habituales, aunque no con todos. Otra de las características de este lenguaje es que puede variar de un navegador a otro, incluso puede ser distinto en versiones distintas del mismo navegador.

Por todo ello y porque Javascript resulta muy sencillo de utilizar para pequeños programas y muy complicado para aplicaciones más elaboradas, no es un lenguaje muy apropiado para hacer juegos para la web. Podría ser que un juego no se viera correctamente en todos los navegadores o que no funcionase en ciertas plataformas. En cualquier caso, seguro que resulta muy difícil de mantener y de mejorar una vez realizado.

No obstante, en el mundo hay gente para todo y muchos locos del Javascript nos han dejado auténticas obras de arte y de ingenio, que nos demuestran que no es tan difícil hacer aplicaciones avanzadas para la web con este lenguaje. En este artículo vamos a mostrar varios juegos de Javascript, realizados con mucha dedicación y que en casi todos los casos, podemos incluir libremente en nuestras páginas web.

### JsTetris

El conocido juego de piezas que caen y tenemos que colocar de la manera más ordenada. Funciona tanto en Internet Explorer como en Netscape. Lo único que no acaba de funcionar todo lo rápido que debería son los eventos del teclado, que a veces no responden todo lo rápido que se desearía.

<http://gosu.pl/dhtml/JsTetris.html>

### Buscaminas

El típico juego del Windows, muy útil por cierto no sólo para ejercitar la lógica, sino también para adquirir una mayor soltura con el manejo del ratón, para aquellos a los que les resulta difícil moverlo.

<http://www.ecirce.com/dhtml/minesweeper/>

### Memory Game

Un juego en el que tenemos varias cartas que se deben agrupar por parejas. En este caso, se proporciona el juego en varias versiones para Javascript, Flash, Java y PHP...

<http://mypage.bluewin.ch/katzenseite/docs/en/games/games.html>

## La serpiente

Un clásico donde los haya. Manejas una serpiente que se come bolitas y cada vez se hace más grande.

<http://www.dynamicdrive.com/dynamicindex12/snake/index.htm>

## Puzzle de burbujas

Se trata de lanzar burbujas de colores. Cuando se consiguen 3 burbujas juntas del mismo color, se descartan de la pantalla. Se pierde cuando no caben más burbujas en la pantalla. Es algo parecido al Tetris.

<http://www.javafile.com/games/bubbles/bubbles.php>

Existen muchos más juegos en Javascript. Aquí sólo hemos señalado algunos que nos han parecido divertidos o muy avanzados técnicamente. Para encontrar otros juegos en Javascript recomendamos acceder a algunas páginas con colecciones de scripts como Hotscripts

<http://www.hotscripts.com> o javafile <http://www.javafile.com>.

*Artículo por Miguel Angel Alvarez*

## **Cross-Browser. DHTML compatible con todos los navegadores.**

Es de sobra conocido, para todos los que hayan experimentado con capas y Javascript, el problema de la compatibilidad de navegadores y la necesidad de realizar distintos códigos DHTML dependiendo del explorador que tenga el cliente.

Si no has encontrado todavía problemas de compatibilidad de tus scripts, posiblemente nunca te hayas enfrentado con la programación del lado del cliente en Javascript, más concretamente con el trabajo con las capas y por tanto, no te interesará mucho este artículo.

Ahora bien, si tienes problemas con las capas al programar una web y no consigues que se vea bien en los distintos navegadores, puede que este artículo te haga más sencilla la vida. Esto gracias a [cross-browser.com](http://cross-browser.com), un sitio web que ofrece una interesante librería de funciones, que vamos a presentar a continuación, para el trabajo con HTML Dinámico compatible con todos los navegadores más habituales.

### **X Library VS CBE API**

En [cross-browser.com](http://cross-browser.com) nos proporcionan dos librerías distintas para el trabajo con DHTML. En realidad se trata de un mismo conjunto de funciones, siendo una librería la evolución de otra. CBE API es la librería original y X Library es su evolución, que todavía (en el momento de publicar este artículo) se encuentra en fase experimental.

CBE API es la librería más antigua y todavía tiene mayores funcionalidades, pero en un breve espacio de tiempo X Library dispondrá de las mismas funcionalidades que la otra, aunque con diversas mejoras como una considerable reducción del peso de los archivos Javascript.

En principio, si entramos a [cross-browser.com](http://cross-browser.com), nos presentan X Library, aunque un poco más escondido encontraremos su antecesor CBE API. El consejo es que se utilice X Library para proyectos nuevos. De todos modos, CBE API sigue perfectamente vigente (por ahora). Por ejemplo, en DesarrolloWeb.com utilizamos la librería CBE API para gestionar el tema de las capas, en concreto en el menú de la cabecera de la página.

## Cómo utilizar X Library

Tenemos que empezar por descargar la versión actual de X Library. Para ello entramos en [cross-browser.com](http://cross-browser.com) y accedemos a la sección de Toys (para el creador de este sitio web, sus juguetitos son librerías DHTML). Aquí podemos hacer el download de la última versión, donde encontraremos las librerías y ejemplos de uso.

Una vez descomprimido el paquete, podemos encontrar una carpeta llamada "x", donde están las librerías. Nos fijamos en que hay una buena cantidad de archivos .JS, cada uno con diversas funciones. Dependiendo de la acción que se desee realizar tendremos que llamar a una función u otra, y tendremos que saber en qué archivo .JS se encuentra. Las funciones más habituales se encuentran en el archivo `x_core.js`. En nuestras páginas no tenemos por qué incluir todos los archivos del directorio, sólo los que vayamos a utilizar. Podemos ver en qué archivo está cada función en la documentación del propio programa, más concretamente en la referencia de funciones (el directorio donde se aloja la documentación está dentro de la carpeta "docs" que está dentro del directorio "x").

Dentro del directorio "x" también hay una carpeta de ejemplos, llamada "examples", a los que podemos echar un vistazo para empezar a familiarizarnos con las posibilidades de la librería. Nosotros hemos creado un par de ejemplos para comprobar las bondades de este software. Para ello hemos seguido el esquema del código de la propia página [cross-browser.com](http://cross-browser.com) y el de los distintos ejemplos que se ofrecen. Algunas funcionalidades de los ejemplos, como la función `xInclude()` están todavía en fase de experimentación, por lo que nosotros no las hemos utilizado y en su lugar incluimos los archivos que necesitamos tal como se hace habitualmente en Javascript.

## Ejemplo mostrar y ocultar capa

X Library dispone de una sencilla función para mostrar y ocultar una capa. Simplemente recibe el identificador de la capa que se desea mostrar u ocultar. En principio, podemos enviarle a la función el propio objeto capa o bien un string con su identificador (que se indica con el atributo `id` en la etiqueta `<div>`).

**Referencia:** En el manual de CSS tenemos explicaciones de [lo que son las capas](#) y la forma de crearlas y definir su posicionamiento y apariencia.

```
<html>
<head>
  <title>Ejemplo de utilización de Cross-Browser</title>
  <script type='text/javascript' src='x_core.js'></script>
  <script type='text/javascript'>
function muestra(){
  xShow('c1');
}
function oculta(){
  xHide('c1');
}
</script>
</head>
```



```
<body>
<div id="c1" style="position:absolute; left: 200px; top: 100px; background-color:#9999aa; width:100px;
height:80px;">
Hola!
</div>

<form>
<input type=button onclick="muestra()" value="Muestra Capa">
<input type=button onclick="oculta()" value="Oculta Capa">
</form>

</body>
</html>
```

En este ejemplo hemos creado una capa (con identificador "c1") y un par de botones. Cuando se pulsa uno se muestra la capa y cuando se pulsa el otro se oculta. Para ello hemos creado también un par de funciones, que se llaman cuando se hace clic en los botones, cuyo cometido es mostrar y ocultar la capa utilizando la librería X Library.

Concretamente vamos a hacer uso de las funciones xShow() y xHide(), que reciben el identificador de la capa que hay que mostrar u ocultar respectivamente. Estas dos funciones se encuentran en el archivo "x\_core.js", que hemos incluido en la página como bloque de script externo.

Se puede [visualizar este sencillo ejemplo en una página web aparte](#).

## Ejemplo para hacer un movimiento de capa

El segundo ejemplo que hemos creado es una capa que se mueve por la página de izquierda a derecha. Esta página tiene también un par de botones, para detener el movimiento o ponerlo en marcha.

```
<html>
<head>
  <title>hacemos un scroll</title>
  <script type='text/javascript' src='x_core.js'></script>
  <script type='text/javascript'>
function inicia(){
  velocidad=2
}
function detiene(){
  velocidad=0
}
function mueve(){
  posicion+=velocidad
  posicion %= 500
  xMoveTo('c1',posicion,100)
  setTimeout("mueve()",100)
}

window.onload = function()
{
  velocidad = 0
  posicion = 200
  mueve()
}

</script>
</head>

<body>
```

```
<div id="c1" style="position:absolute; left: 200px; top: 100px; background-color:red; width:100px; height:20px;">
Hola!
</div>

<form>
<input type=button onclick="inicia()" value="Mueve la Capa">
<input type=button onclick="detiene()" value="Para la Capa">
</form>

</body>
</html>
```

En este caso, por lo que respecta a la librería, hemos utilizado la función `xMoveTo()`, que recibe el identificador de la capa a mover y las nuevas coordenadas donde colocarla. Esta función también se encuentra dentro del archivo "x\_core.js", que hemos incluido con el primer bloque de script.

Para entender el movimiento de la capa tenemos que haber visto alguna vez la función `setTimeout()`, que recibe una instrucción Javascript a ejecutar y una cantidad de milisegundos que deben de pasar antes de la ejecución. En el ejemplo tenemos una función que se llama `mueve()`, que se encarga de variar la posición actual de la capa. Esta función se llama a si misma por medio de `setTimeout()`, con un retardo de 100 milisegundos, por tanto, la función `mueve()` nunca para de ejecutarse, en concreto 10 veces por segundo.

**Nota:** [setTimeout es un método del objeto window](#). Tenemos varios artículos que utilizan esta función, por ejemplo: [Reloj en Javascript](#) o [Texto en movimiento en la barra de estado](#).

Luego hemos definido una variable `velocidad`, que es el número de pixels que se desplaza la capa en cada llamada a `mueve()`. Los dos botones lo único que hacen es modificar el valor de esa variable `velocidad`. El botón que para el movimiento simplemente asigna el valor cero a la velocidad.

Otra cosa que tenemos que ver es que se ha definido una serie de instrucciones a ejecutar cuando se carga la página, en el bloque `window.onload = function()`. Entre esas acciones a ejecutar se encuentra la configuración de la posición inicial de la capa y la velocidad del movimiento. Además, se hace una llamada inicial a `mueve()`, que desencadena el flujo del movimiento, pues la función `mueve` se encarga de llamarse a si misma hasta que el usuario abandona la página.

Si se desea, se puede [ver el ejemplo en funcionamiento](#).

## Conclusión

No era nuestra intención explicar todas las funcionalidades de esta herramienta, sino más bien darla a conocer. Tampoco es nuestra intención explicar cómo realizar un movimiento de una capa ayudados por `setTimeout()`, aun así pedimos disculpas para aquel lector que no haya podido entender por qué realizamos un script de movimiento de esa forma.

Nuestro objetivo era presentar las librerías y mostrar como, con muy poco código y sin tener que conocer los entresijos de cada navegador, se puede realizar un ejemplo ya bastante avanzado del manejo de capas.

Recomendamos que no reinventéis la rueda. Ya que se dispone de herramientas de trabajo con capas tan versátiles como X Library, es mucho más interesante basar nuestros scripts en ella que rompernos la cabeza para inventar mecanismos compatibles con cada navegador.

Hemos creado un manual donde vamos a ir comentando varios ejemplos de efectos interesantes que se pueden realizar utilizando estas librerías. El manual en concreto se llama [Taller de Cross-Browser DHTML](#).

*Artículo por Miguel Angel Alvarez*

## Calcular la letra del DNI en Javascript

Los números de identidad personales, por lo menos en España, tienen una parte numérica y otra parte de texto. El número es variable para cada español y la letra se calcula con una fórmula matemática a partir del número.

En este taller de Javascript vamos a ver una función para calcular la letra de un DNI. La función recibe el número del DNI desde un campo de texto de un formulario, hace el cálculo de la letra correspondiente y escribe en el campo otra vez el DNI con la letra calculada.

Vamos a ver el ejemplo:

Para empezar, dentro de la cabecera de la página -en el head- colocaríamos la función que calcula el DNI:

```
<script>
function averigua ()
{
  cadena="TRWAGMYFPDXBNJZSQVHLCKET"
  posicion = formulario.dni.value % 23
  letra = cadena.substring(posicion,posicion+1)
  document.formulario.dni.value=formulario.dni.value+" - "+letra
}
</script>
```

Ahora vamos a ver el formulario que colocaríamos en el cuerpo de la página. Contiene un campo de texto y un botón. En el campo de texto colocaríamos el número del DNI y al pulsar el botón, se realizaría el cálculo de la letra, colocándola dentro del campo junto con el número del DNI.

```
<form name="formulario">
DNI: <br>
<input type="text" name="dni" maxlength="11" size="11">
<input type="button" value="OK" language="JavaScript" onclick="averigua()">
</form>
```

Para terminar, podemos [ver el ejemplo en funcionamiento en una página aparte](#).

*Artículo por Manuel Estévez Simonet*

## HTML Area. Editor WYSIWYG

Vamos a presentar una herramienta gratuita que sirve para crear elementos de formulario,

parecidos a los <textarea>, pero con la particularidad de que permiten introducir texto con estilos, como negritas, subrayados, distintos tipos de fuentes e incluso, tablas o imágenes. En definitiva, provee de las funciones típicas de los editores HTML WYSIWYG (what you see is what you get), pero dentro de un campo de formulario de una página web.

htmlArea se incluye en la página con unas pocas líneas Javascript fáciles de escribir. Con ello obtenemos un editor que permite funcionalidades como:

- Formatear texto con negritas, cursivas y subrayados
- Cambiar la tipografía y el color
- Alinear los distintos párrafos
- Incluir listas, líneas horizontales, links, images...

Como el programa está realizado en Javascript y trabaja únicamente del lado del cliente, lo podremos utilizar en cualquier tipo de servidor (no requiere programación ASP o PHP). La desventaja es que funciona únicamente en versiones de Internet Explorer 5.5 o superiores. Por lo menos, en otros navegadores, no dará errores, sino que simplemente veremos un campo <textarea> normal.

### Cómo se inserta el htmlArea

Vamos a ver cómo se incluye este tipo de editor en una página web. Como primer paso, descargamos el software que podremos encontrar en la página de inicio de [htmlArea](#). El archivo es muy pequeño. Viene en .zip, por lo que debemos descomprimirlo en nuestro ordenador.

Una vez descomprimido, encontramos ejemplos e instrucciones para su funcionamiento. Ejecutando el archivo llamado example.html podemos hacer una pequeña prueba, para ver si nuestro navegador soporta htmlArea. Si todo es correcto, podemos hacer nuestra primera prueba.

Vamos a crear un archivo HTML, para hacer nuestro primer ejemplo, en el mismo directorio donde hemos descomprimido el software. Lo editamos y colocamos en la cabecera el siguiente código:

```
<script language="Javascript1.2">
<!--
// Carga de htmlarea
_editor_url = "" // URL del archivo htmlarea var win_ie_ver = parseFloat(navigator.appVersion.split("MSIE")[1]);
if (navigator.userAgent.indexOf('Mac') >= 0) { win_ie_ver = 0; }
if (navigator.userAgent.indexOf('Windows CE') >= 0) { win_ie_ver = 0; }
if (navigator.userAgent.indexOf('Opera') >= 0) { win_ie_ver = 0; }
if (win_ie_ver >= 5.5) {
document.write('<scr' + 'ipt src="' + _editor_url + 'editor.js"');
document.write(' language="Javascript1.2"></scr' + 'ipt>');
} else { document.write('<scr' + 'ipt>function editor_generate() { return false; }</scr' + 'ipt>'); }
// -->
</script>
```

Lo único que hay que editar en este código es la variable "\_editor\_url", a la que tenemos que asignar la ruta donde se encuentran los archivos de htmlArea. Como en este caso el archivo de ejemplo está en el mismo directorio que htmlArea, asignamos un string vacío a la variable:

```
_editor_url = "" // URL del archivo htmlarea
```

Continuamos colocando un formulario con un campo textarea dentro.

```
<form>
<textarea name="campo1" style="width:500px; height:200px;">
</textarea>
</form>
```

Nos fijamos que el campo textarea tiene un nombre, que luego utilizaremos. También hemos definido con atributos de estilos el ancho y alto del campo. Esto último es opcional pues, como cualquier campo textarea, también podríamos haber definido sus dimensiones con los atributos cols y rows.

Por último, debemos indicar que ese campo debe mostrarse como un contenedor HTML y no como un <textarea> normal. Para ello debemos incluir, a continuación del <textarea>, este código Javascript.

```
<script language="JavaScript1.2" defer>
editor_generate('campo1');
</script>
```

Es una llamada a la función que se encarga de generar el editor HTML a partir del campo <textarea>. Nos fijamos que la función recibe un string con el nombre del campo <textarea> que se desea convertir a htmlArea.

Con esto ya obtenemos el campo WYSIWYG. [Podemos verlo en una página aparte.](#)

Para conseguir que el campo htmlArea tenga un texto por defecto, simplemente debemos insertar ese texto entre <textarea> y </textarea>. Podemos insertar código HTML y se mostrará dentro de la propia página.

Podemos [ver un segundo ejemplo](#) que contiene un texto definido por defecto.

Podemos ver el código de ambos ejemplos con la opción "Ver código fuente" del navegador.

## Conclusión

Nos hemos dejado por ver las múltiples opciones de configuración del programa, para limitar o mejorar las opciones de edición disponibles o la interfaz. En la propia documentación del programa podemos estudiarlas.

Existen varios productos parecidos a htmlArea. Hemos empezado comentando éste porque es muy sencillo, tanto en su instalación como en la utilización. Es muy interesante también porque se puede utilizar simplemente desde Javascript. No hay necesidad concreta de disponer de un servidor con posibilidades de programación en ASP o PHP, aunque lo lógico es que los utilicemos para combinarlos con el editor WYSIWYG y permitir que los usuarios actualicen información de la base de datos, incluyendo sus estilos, imágenes, etc.

Tiene muchas más ventajas, como su gratuidad. Podemos utilizarlo para cualquier propósito y modificar el código para adaptarlo a nuestras necesidades.

**Podemos encontrar más información en:**

<http://www.htmlarea.com>

Artículo por **Miguel Angel Alvarez**

## Ocultar un email de un enlace para evitar el spam

Uno de los mecanismos que utilizan las personas que envían spam (spammers) para obtener direcciones de correo para su lista de distribución es rastrear la web en busca de direcciones email. Todas las direcciones que aparecen en las páginas web, a la vista o escritas en el código, son susceptibles de ser capturadas y utilizadas para el envío de spam. Por eso, no es mala idea proteger nuestros correos para ponerles la tarea difícil a los spammers, y evitar que en poco tiempo comencemos a recibir mensajes no deseados.

Un enlace a un correo electrónico es así:

```
<a href="mailto:correo@midominio.com">correo@midominio.com</a>
```

Tanto en href como en el texto del enlace aparece nuestro correo electrónico. En este artículo veremos un par de ideas para evitar que aparezcan nuestros datos, de modo que no puedan captar las direcciones.

### Utilizar una imagen en el texto del enlace

Una buena solución consiste en utilizar una simple imagen donde aparece el correo. Esta imagen tendrá el texto del correo electrónico, para que el visitante pueda visualizar la dirección en la página, pero escrita sobre una imagen. Eso es indetectable por un robot que escanee la página y nuestros clientes podrán ver claramente cuál es el correo donde deben escribirnos.

Si no ponemos el enlace y colocamos sólo la imagen, acabarían nuestros problemas. El visitante no podría pulsar la dirección de correo en la propia página para enviarnos un mail, pero muy probablemente sea suficientemente avisado para copiarla por el mismo en el programa de correo que utilice.

### Utilizar Javascript para ocultar la dirección

Podemos por mediación de Javascript hacer un pequeño programa para que nuestra dirección no aparezca en el código, por lo menos no tan clara. Podemos, por ejemplo, partirla en diferentes trozos y luego concatenarla, de manera que no pueda verse por completo en ningún sitio del código de la página.

Veamos este script:

```
<script language="JavaScript">
usuario="pepe"
dominio="alpepone.com"
conector="@"
```

```
function dame_correo(){
    return usuario + conector + dominio
}
```

```
function escribe_enlace_correo(){
    document.write("<a href='mailto:' + dame_correo() + '>' + dame_correo() + "</a>")
}
</script>
```

Primero se definen tres variables que forman el correo electrónico que deseamos ocultar. Luego tenemos dos funciones útiles:

La función `dame_correo()` devuelve el correo electrónico que se desea ocultar. Simplemente concatena las partes del correo electrónico, que se habían definido en variables más arriba.

Por su parte, `escribe_enlace_correo()`, escribe en la página web un enlace a correo electrónico completo. Un enlace a un correo electrónico es así:

```
<a href="mailto:correo@midominio.com">correo@midominio.com</a>
```

Esta función se apoya en la `dame_correo()` para obtener el correo que se deseaba ocultar.

Para que aparezca en la página el enlace al correo electrónico debemos hacer una llamada a la función `escribe_enlace_correo()`, en el lugar del cuerpo que deseamos que se muestre.

**Nota:** si el ordenador del usuario no tiene Javascript o lo tiene deshabilitado, no podrá ver esas direcciones de correo escritas en la página desde Javascript. Por ello, puede ser buena idea combinar este truco con el de mostrar una imagen con el correo, para que por lo menos se vea la imagen. Aunque todavía hay navegadores en sólo texto, con lo cual ni siquiera verían la imagen. En fin, todo un mundo de posibilidades.

Este sería el código para mostrar en cualquier parte de la página.

```
<body>
<!-- en cualquier parte del cuerpo de la página -->
<script>escribe_enlace_correo()</script>
</body>
```

Si el robot del spammer es muy listo, probablemente pueda poner en ejecución el Javascript para interpretarlo y saber dónde está escondida la dirección de correo. Eso parece por el momento poco probable. Hay tantas direcciones en las páginas web, que es posible que no se vayan a entretener en realizar cábalas para obtener direcciones ocultas en el código de la página.

No obstante, seguro que hay maneras de ocultar un poquito mejor con Javascript esa dirección. Puede que lo de las variables definidas arriba del todo puede resultar un poco obvio. Dejo para vosotros investigar esa tarea si lo deseáis. Puede que hablemos de ella en un artículo posterior. Enviar vuestros comentarios si tenéis alguna ayuda para mejorar el script.

El ejemplo de ocultar el correo utilizando Javascript se puede [ver en funcionamiento en una página aparte](#).

**Nota:** Tenemos otro artículo relacionado: [Esconder con CSS el email a los spambots](#), así como un manual dedicado exclusivamente a las [técnicas y trucos para evitar el spam](#).

*Artículo por **Miguel Angel Alvarez***

## Validación de un formulario con Javascript

Vamos realizar un ejemplo de un formulario completo para validar. Las validaciones se hacen en el propio navegador antes de enviarlo. Si hubo algún campo no relleno o con información errónea, el formulario muestra el campo que está incorrecto y solicita al usuario que lo cambie. Si todos los datos del formulario son correctos se envía el formulario.

Hemos querido hacer un formulario sencillo, para que el ejercicio no se haga demasiado complicado. No obstante, se realizan validaciones en campos con distintos valores, para hacerlo más variado. Se comprueba un campo donde debe figurar un texto, otro donde debe introducirse un número mayor que 18 y un último con un campo select donde deben haber seleccionado un valor.

**Referencia:** para comprender este ejercicio hace falta conocer el [trabajo con formularios con Javascript](http://www.desarrolloweb.com/trabajo-con-formularios-con-javascript/). Podemos aprender también Javascript desde cero <http://www.desarrolloweb.com/javascript/>, si es que fuera necesario.

Se puede [ver el ejemplo en funcionamiento](#) para hacerse una idea más exacta del objetivo buscado.

### El código del formulario

El formulario con el que vamos a trabajar es el siguiente:

```
<form name="fvalida">
<table>
<tr>
  <td>Nombre: </td>
  <td><input type="text" name="nombre" size="30" maxlength="100"></td>
</tr>
<tr>
  <td>Edad: </td>
  <td><input type="text" name="edad" size="3" maxlength="2"></td>
</tr>
<tr>
  <td>Interés:</td>
  <td>
    <select name=interes>
    <option value="Elegir">Elegir
    <option value="Comercial">Contacto comercial
    <option value="Clientes">Atención al cliente
    <option value="Proveedores">Contacto de proveedores
    </select>
  </td>
</tr>
<tr>
  <td colspan="2" align="center"><input type="button" value="Enviar" onclick="valida_envia()"></td>
</tr>
</table>
</form>
```

Es un formulario cualquiera. Los únicos puntos donde debemos prestar atención son:

- El nombre del formulario, "fvalida", que utilizaremos para referirnos al él mediante Javascript.
- El botón de enviar, que en lugar de ser un submit corriente, es un botón que llama a una función, que se encarga de validar el formulario y enviarlo si todo fue correcto.

### Función Javascript para validar el formulario



Ahora veremos la función que hemos creado para validar el formulario. Se llama `valida_envia()`. Simplemente, para cada campo del formulario, comprueba que el valor introducido es correcto. Si no es correcto, muestra un mensaje de alerta, pone el foco de la aplicación en el campo que ha dado el error y abandona la función retornando el valor 0.

Si todos los campos eran correctos, la función continúa hasta el final, sin salirse, por no estar ningún campo incorrecto. Entonces ejecuta la sentencia última, que es el envío del formulario.

Veamos la función entera, aunque luego la expliquemos por partes.

```
function valida_envia(){
    //valido el nombre
    if (document.fvalida.nombre.value.length==0){
        alert("Tiene que escribir su nombre")
        document.fvalida.nombre.focus()
        return 0;
    }

    //valido la edad. tiene que ser entero mayor que 18
    edad = document.fvalida.edad.value
    edad = validarEntero(edad)
    document.fvalida.edad.value=edad
    if (edad==""){
        alert("Tiene que introducir un número entero en su edad.")
        document.fvalida.edad.focus()
        return 0;
    }else{
        if (edad<18){
            alert("Debe ser mayor de 18 años.")
            document.fvalida.edad.focus()
            return 0;
        }
    }

    //valido el interés
    if (document.fvalida.interes.selectedIndex==0){
        alert("Debe seleccionar un motivo de su contacto.")
        document.fvalida.interes.focus()
        return 0;
    }

    //el formulario se envia
    alert("Muchas gracias por enviar el formulario");
    document.fvalida.submit();
}
```

En el primer bloque se valida el campo nombre. La validación que se hace es simplemente si se ha escrito algo en el campo. Para ello comprueba si el número de caracteres escritos en el campo nombre es cero. En ese caso, muestra el mensaje de alerta, sitúa el foco en el campo de texto y se sale de la función devolviendo el valor 0.

**Nota:** el foco de la aplicación es el lugar donde está situado el cursor. El foco puede estar en cualquier sitio. Por ejemplo en un campo de texto, en un select, en un enlace o en la propia página. Si presionamos una tecla del teclado afecta al lugar donde está situado el foco. Si, por ejemplo, el foco está en un campo de texto, al operar con el teclado estaremos escribiendo en ese campo de texto.

La validación de la edad mayor que 18 años tiene dos partes. Primero debemos comprobar que en el campo de texto hay escrito un valor entero. Luego, si teníamos un entero, habría que comprobar que es mayor que 18. Para hacer esta validación nos vamos a apoyar en una

función que ya hemos visto en otro artículo de DesarrolloWeb.com, que habla sobre cómo [validar un entero en un campo de formulario](#). Esa función devuelve un string vacío en caso de que no sea un entero y el propio entero, si es que lo era.

Antes de realizar la validación de la edad propiamente dicha, se obtiene el valor introducido en el campo de formulario edad y se guarda en una variable llamada edad. Luego se ejecuta la función pasando esta edad. Su resultado lo utilizamos para volcarlo otra vez al campo de texto. Entonces, se comprueba si el valor devuelto por la función es un string vacío. En ese caso, es que el valor escrito en el formulario no era un entero, por lo que se muestra el mensaje de error, se sitúa el foco y se sale de la función.

En caso de que el campo edad contuviese un entero, se debe comprobar a continuación si es mayor que 18. En caso de que sea menor, se muestra el error y se sale. En caso contrario -entonces el valor sería mayor o igual que 18-, se continúa con las comprobaciones.

Por último se valida el campo select, donde aparece el interés del supuesto visitante, que le motiva para enviarnos el formulario. En ese campo se debe haber seleccionado cualquier opción menos la primera. Para asegurarnos, simplemente se comprueba si el atributo selectedIndex del campo select tiene el valor 0. Ese atributo almacena el índice seleccionado en el menú desplegable. El primer campo tiene el índice 0, el segundo el índice 1...

Si se comprueba que selectedIndex vale 0, se muestra un mensaje de alerta, se pone el foco en el campo del formulario y se sale de la función.

Si hemos llegado hasta este punto sin salirnos de la función es que todos los campos del formulario estaban rellenos correctamente. En ese caso se debe enviar el formulario. Antes de enviar el formulario se muestra un mensaje de alerta, agradeciendo que se haya rellenado correctamente. Este mensaje se puede suprimir si se desea.

Para enviar el formulario se hace una llamada al método submit() de dicho formulario.

## Conclusión

Este ejercicio es de lo más básico y útil que se puede hacer en Javascript. Requiere ciertos conocimientos, ya ligeramente avanzados, pero en el fondo no resulta complicado. Incluso ampliarlo es bastante sencillo, siempre que sigamos un esquema similar para cada uno de los campos.

Podemos [ver el resultado obtenido en una página aparte](#).

Artículo por **Miguel Angel Alvarez**

## Función en Javascript para la inserción de fechas

La siguiente es una función que realice para hacer más fácil la escritura de campos fecha con formatos de dd/mm/aaaa y una pequeña validación, también les mando una pequeña función para saber si un valor es numérico:

1 debemos tener un formulario html con un campo con la siguiente forma:

```
<input name="fecha" type="text" size="10" maxlength="10" onKeyUp = "this.value=formateafecha(this.value);">
```

Aquí se puede observar que hacemos un llamado a la función: `formateafecha`

2- en el mismo html colocamos los javascript con las funciones o lo ponemos aparte en un archivo.js hacien donde deberíamos hacerle un llamado.

```
function IsNumeric(valor)
{
var log=valor.length; var sw="S";
for (x=0; x<log; x++)
{ v1=valor.substr(x,1);
v2 = parseInt(v1);
//Compruebo si es un valor numérico
if (isNaN(v2)) { sw= "N";}
}
if (sw=="S") {return true;} else {return false; }
}

var primerslap=false;
var segundoslap=false;
function formateafecha(fecha)
{
var long = fecha.length;
var dia;
var mes;
var ano;

if ((long>=2) && (primerslap==false)) { dia=fecha.substr(0,2);
if ((IsNumeric(dia)==true) && (dia<=31) && (dia!="00")) { fecha=fecha.substr(0,2)+"/"+fecha.substr(3,7);
primerslap=true; }
else { fecha=""; primerslap=false;}
}
else
{ dia=fecha.substr(0,1);
if (IsNumeric(dia)==false)
{fecha="";}
if ((long<=2) && (primerslap=true)) {fecha=fecha.substr(0,1); primerslap=false; }
}
if ((long>=5) && (segundoslap==false))
{ mes=fecha.substr(3,2);
if ((IsNumeric(mes)==true) && (mes<=12) && (mes!="00")) { fecha=fecha.substr(0,5)+"/"+fecha.substr(6,4);
segundoslap=true; }
else { fecha=fecha.substr(0,3);; segundoslap=false;}
}
else { if ((long<=5) && (segundoslap=true)) { fecha=fecha.substr(0,4); segundoslap=false; } }
if (long>=7)
{ ano=fecha.substr(6,4);
if (IsNumeric(ano)==false) { fecha=fecha.substr(0,6); }
else { if (long==10){ if ((ano==0) || (ano<1900) || (ano>2100)) { fecha=fecha.substr(0,6); } } }
}

if (long>=10)
{
fecha=fecha.substr(0,10);
dia=fecha.substr(0,2);
mes=fecha.substr(3,2);
ano=fecha.substr(6,4);
// Año no viciesto y es febrero y el día es mayor a 28
if ( ( ano%4 != 0) && (mes ==02) && (dia > 28) ) { fecha=fecha.substr(0,2)+"/"; }
}
return (fecha);
}
```

Artículo por **Juan Carlos Aleman Paez**

## DHTML Calendar

Cuando realizamos una interfaz de usuario, es típico tener campos donde el visitante deba introducir una fecha. Éstas tienen formatos bastante estrictos y son complicadas de escribir, por lo que es muy cómodo para el usuario contar con la posibilidad de utilizar un calendario para seleccionar la fecha.

En DesarrolloWeb.com hemos publicado un manual donde se explica cómo construir un [calendario con PHP](#), un conocido lenguaje de programación de webs del lado del servidor. Ahora bien, no siempre el visitante va a tener la capacidad de entender la programación en PHP, o acceso a un servidor que permita la publicación de contenidos programados con PHP.

Por ello, será muy interesante conocer otras maneras de implementar un calendario en una página web. En este caso vamos a presentar DHTML Calendar, un calendario realizado en Javascript, compatible para todos los navegadores. Este script para incorporar un calendario es gratuito, por lo que podemos utilizarlo sin ningún tipo de límite.

### Cómo es DHTML Calendar

Es un sistema muy potente y fácilmente configurable, con una interesante interfaz, totalmente dinámica. Se puede incluir de diversas maneras dentro de una página, como un popup, o directamente en el cuerpo de la página, lo que lo hace útil en diversas situaciones.

El script para configurar el calendario variará de un modo de presentación a otro. En la descarga del calendario se ofrecen algunos ejemplos rápidos para mostrar el calendario. Ejemplos para los impacientes, que pueden venir muy bien para empezar rápidamente. Una de las maneras más típicas de presentar el calendario puede ser utilizando un campo de texto y un botón. Al pulsar el botón se muestra el calendario y, una vez seleccionada una fecha, se escribe en el campo de texto.

El código del ejemplo sería el siguiente, muy parecido a uno de los ejemplos para los impacientes proporcionados en el paquete de descarga.

```
<html>
<head>

  <title>Calendario de pruebas</title>

  <!--Hoja de estilos del calendario -->
  <link rel="stylesheet" type="text/css" media="all" href="calendar-green.css" title="win2k-cold-1" />

  <!-- librería principal del calendario -->
  <script type="text/javascript" src="calendar.js"></script>

  <!-- librería para cargar el lenguaje deseado -->
  <script type="text/javascript" src="lang/calendar-es.js"></script>

  <!-- librería que declara la función Calendar.setup, que ayuda a generar un calendario en unas pocas líneas de código -->
  <script type="text/javascript" src="calendar-setup.js"></script>
```

```
</head>

<body>

<!-- formulario con el campo de texto y el botón para lanzar el calendario-->
<form action="#" method="get">
<input type="text" name="date" id="campo_fecha" />
<input type="button" id="lanzador" value="..." />
</form>

<!-- script que define y configura el calendario-->
<script type="text/javascript">
  Calendar.setup({
    inputField    : "campo_fecha",    // id del campo de texto
    ifFormat     : "%d/%m/%Y",       // formato de la fecha que se escriba en el campo de texto
    button       : "lanzador"        // el id del botón que lanzará el calendario
  });
</script>

</body>
</html>
```

El código anterior está comentado para que se entienda más fácilmente. Tiene varias partes.

- En la cabecera se incluyen varios ficheros con las funciones y estilos del calendario. Estos ficheros se encuentran en los archivos de descarga del calendario. Existen varios estilos que se pueden utilizar para ajustar el aspecto del calendario al diseño de la página. También se debe incluir el lenguaje en el que presentar el calendario, en este caso español, que está también incluido en el paquete de descarga.
- Ya en el cuerpo de la página, se muestra el formulario. Es muy simple, pues sólo tiene un campo de texto y un botón para mostrar el calendario.
- En el script de javascript, también dentro del cuerpo de la página, se utiliza la función `Calendar.setup`, que sirve para cargar el calendario y configurarlo con los valores que deseemos. Todas las opciones de configuración tienen valores por defecto, aunque siempre vamos a tener que definir, como mínimo, los datos que ponemos en este ejemplo. El dato `"inputField"` sirve para indicar el identificador (atributo `id`) del campo `input` donde se tiene que escribir la fecha. El valor `"ifFormat"` sirve para ajustar el formato de la fecha que se desea escribir en el campo de texto. Por último, el valor `"button"` debe contener el identificador del botón que lanzará el calendario al pulsarlo.

Ha sido un ejemplo lo más simplificado posible. Podemos ver la marcha del [calendario que genera este código](#) en una página aparte.

En la documentación podremos encontrar más ejemplos del calendario y una explicación detallada de su utilización.

Más información en la página del producto: <http://www.dynarch.com/projects/calendar/>

DHTML Calendar es un proyecto alojado en SourceForge.net, con la siguiente página de proyecto: <http://sourceforge.net/projects/jscalendar>

*Artículo por **Miguel Angel Alvarez***

## Generar un color aleatorio con Javascript

Este ejercicio es muy sencillo, pero puede resultar útil para algunas personas. Se trata de una pequeña función que sirve para generar un color aleatorio, en formato hexadecimal, que es el utilizado en la creación de webs.

Lo hemos extraído de otro ejemplo realizado en Javascript, en el que necesitábamos generar un color de manera aleatoria. Creemos que puede ser interesante para comentarlo en un artículo aparte, por si alguien en sus páginas necesita crear un color totalmente aleatorio.

Para crear un color aleatorio necesitamos simplemente 6 números en hexadecimal (números del 0 a la F). Si obtenemos aleatoriamente estos 6 números hexadecimales, habremos creado el código de un color aleatorio.

### Función para generar aleatorios

Por tanto, para generar un número aleatorio, vamos a apoyarnos en una función relatada en otro artículo de DesarrolloWeb.com: [Generación de números aleatorios Javascript](#). En este artículo hay que fijarse también en el comentario de un visitante que mejora la función de creación de aleatorios.

La función para generar aleatorios que vamos a utilizar entonces es la siguiente:

```
function aleatorio(inferior,superior){  
    numPosibilidades = superior - inferior  
    aleat = Math.random() * numPosibilidades  
    aleat = Math.floor(aleat)  
    return parseInt(inferior) + aleat  
}
```

Como nosotros necesitamos un aleatorio hexadecimal, para apoyarnos en esta función, vamos a generar un número aleatorio en decimal, que luego convertiremos a hexadecimal. Para hacer esa conversión utilizaremos un array de valores hexadecimales como este:

```
hexadecimal = new Array("0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F")
```

Ahora, para obtener ese valor aleatorio hexadecimal, lo único que tenemos que hacer es obtener un índice entre 0 y el número de casillas de este array. Entonces, el valor aleatorio hexadecimal será lo que haya en el array en la casilla cuyo índice se ha obtenido aleatoriamente. Esto se hace de esta manera:

```
posarray = aleatorio(0,hexadecimal.length)  
valor_hexadecimal_aleatorio = hexadecimal[posarray]
```

En la primera línea se obtiene el índice del array aleatorio, que está entre 0 y el número de posiciones del array. En la segunda valor\_hexadecimal\_aleatorio se obtendrá accediendo al array, en la posición generada aleatoriamente.

### Obtener el color aleatorio

Ahora que ya hemos visto una manera de obtener un valor hexadecimal aleatorio, vamos a ver cómo obtener un color aleatorio, que no es más que obtener 6 valores hexadecimales aleatorios y concatenarlos.

```
hexadecimal = new Array("0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F")
color_aleatorio = "#";
for (i=0;i<6;i++){
    posarray = aleatorio(0,hexadecimal.length)
    color_aleatorio += hexadecimal[posarray]
}
```

Después de la ejecución de este código, la variable color\_aleatorio contendrá el color generado aleatoriamente.

### Ponerlo en una función

Para acabar esta pequeña práctica, vamos a ver cómo se puede poner todo esto en una función, que podremos utilizar en cualquier contexto.

```
function dame_color_aleatorio(){
    hexadecimal = new Array("0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F")
    color_aleatorio = "#";
    for (i=0;i<6;i++){
        posarray = aleatorio(0,hexadecimal.length)
        color_aleatorio += hexadecimal[posarray]
    }
    return color_aleatorio
}
```

Para [ver este ejemplo en marcha](#) hemos creado una página que muestra una serie de 50 colores generados aleatoriamente.

*Artículo por **Miguel Angel Alvarez***

## El aprendizaje en Internet

Aunque el ordenador no es la panacea de la futura enseñanza, nadie pone en duda que es un medio didáctico muy potente que puede cambiar la forma de enseñar de los profesores y el modo de aprender de los alumnos.

Cuando el chico se enfrenta con el ordenador y con un software educativo o con Internet, el protagonista ya no es el profesor sino el propio alumno, quien lee, comprende y aprende por sí mismo.

El profesor Jesús A. Beltrán Lera distingue la pedagogía de la reproducción o pasiva de la pedagogía de la imaginación o activa e innovadora.

También las páginas web se pueden dividir en dos clases: las que presentan textos y dibujos que se pueden copiar e imprimir (pero son pasivas y estáticas) y las páginas interactivas (con un diálogo entre el ordenador y el niño). En el mundo de la enseñanza es muy aconsejable que los ejercicios sean del segundo tipo.

Un modelo interactivo puede ser éste: La pantalla le presenta una "Ayuda" con el contenido que el chico necesita para hacer bien los ejercicios; después el ordenador le plantea una pregunta con varias alternativas de respuesta y el alumno contesta a una de ellas; si la

respuesta es acertada el ordenador le premia cambiando el color de la respuesta de negro a rojo y si es falsa le informa de su error con una alerta dándole la información correcta. La interactividad permite la corrección instantánea de la contestación del alumno, lo que supone un factor motivador.

Al terminar todas las preguntas, el alumno accede a un botón de "Puntuación" que al pulsarlo nos da tres informaciones: el número de aciertos, el número de errores y una calificación con uno de estos tres comentarios de evaluación: "Muy bien. Magnífico", si no ha cometido errores; "Bien, pero puedes mejorar", si la puntuación (de 0 a 10) es de 7 o superior; y "Debes repetir el ejercicio", si la puntuación es inferior a 7 puntos. Esta interactividad produce una realimentación positiva que incrementa la motivación del chico y estimula la actividad escolar.

Con html y javascript se pueden preparar ejercicios como los comentados anteriormente y sobre distintos temas como ortografía, lectura comprensiva, poesías, adivinanzas y cálculo.

Veamos dos ejemplos de ortografía, uno con la ayuda en "alert" y el otro utilizando el "popup".

En el primer ejemplo pondremos dentro de la cabecera de la página (entre el y ) un script con las funciones que controlan los aciertos, errores y puntuaciones:

```
<script LANGUAGE="JavaScript">

var sumafa=0
var sumaaci=0
var res=0
var nota=0

function resbien(sumafa2)
{
    sumaaci=sumaaci+1;
    return true
}

function resmal(sumafa2)
{
    sumafa=sumafa+1;
    return true
}

function averiguarNota(nota2){
    res=sumaaci+sumafa
    res=res/10
    nota=sumaaci/res
    if (res <=0)
    {
        alert("Puedes empezar este ejercicio. ¡Suerte!")
    }
    else
    if (nota >=10)
    {
        alert("Has tenido "+sumaaci+" aciertos y "+sumafa+" errores. Muy Bien. Magnífico")
    }
    else
    if (nota >7)
    {
        alert("Has tenido "+sumaaci+" aciertos y "+sumafa+" errores. Bien, pero puedes mejorar.")
    }
}
```



```

}
else
{
    alert("Has tenido "+sumaaci+" aciertos y "+sumafa+" errores. Debes repetir el ejercicio.")
}
sumaaci=0
sumafa=0 }

</script>

```

Después pondremos el formulario en el cuerpo de la página. Contiene el ejercicio propiamente dicho y los botones de "Ayuda", "Puntuación" y "Borrar contestaciones":

```

<h1 align="center"><input Type="Button" Value="Ayuda" onclick="alert('abrir, advertir, afirmativo, amable, cueva, \n \nsubir, obstruir, atrever, avión, bandido.')"></h1>
<form NAME="bv">
<div align="center"><center><table border="8" width="346" cellpadding="1" cellspacing="1">
<tr>
<td width="182" bgcolor="#FFFFFF"><div align="center"><center><p><small><font face="Verdana"
color="#FF0000"><b>a_rir</b></font></small></td>
<td width="152" align="center" bgcolor="#FFFFFF"><b><div align="center"><center><p>
<input Type="Button" Value=" b " onclick="resbien(); this.style.color='red'">
<input Type="Button" Value=" v " onclick="resmal(); alert('abrir, advertir, afirmativo, amable, cueva, \n \nsubir,
obstruir, atrever, avión, bandido.')">
<input Type="Button" Value=" w " onclick="resmal(); alert('abrir, advertir, afirmativo, amable, cueva, \n \nsubir,
obstruir, atrever, avión, bandido.')">
</b></td>
</tr>
<tr align="center">
<td width="182" bgcolor="#FFFFFF"><div align="center"><center><p><small><font
face="Verdana" color="#FF0000"><b>ad_ertir</b></font></small></td>
<b><td width="152" align="center" bgcolor="#FFFFFF"><b><input Type="Button" Value=" b "
onclick="resmal(); alert('abrir, advertir, afirmativo, amable, cueva, \n \nsubir, obstruir, atrever, avión, bandido. ')">
<input Type="Button" Value=" v " onClick="resbien(); this.style.color='red'"> <input
Type="Button" Value=" w "
onClick="resmal(); alert('abrir, advertir, afirmativo, amable, cueva, \n \nsubir, obstruir, atrever, avión, bandido.')">
</b></b></td>
</tr>
<tr align="center">
<td width="182" bgcolor="#FFFFFF"><div align="center"><center><p><small><font
face="Verdana" color="#FF0000"><b>afirmati_o</b></font></small></td>
<b><td width="152" align="center" bgcolor="#FFFFFF"><b><input Type="Button" Value=" b "
onclick="resmal(); alert('abrir, advertir, afirmativo, amable, cueva, \n \nsubir, obstruir, atrever, avión, bandido. ')">
<input Type="Button" Value=" v " onClick="resbien(); this.style.color='red'"> <input
Type="Button" Value=" w "
onClick="resmal(); alert('abrir, advertir, afirmativo, amable, cueva, \n \nsubir, obstruir, atrever, avión, bandido.')">
</b></b></td>
</tr>
</table>
</center></div><div align="center"><center><p><strong>
<input Type="Button" value="Puntuación" onclick="averiguarNota()">
<input TYPE="SUBMIT" VALUE="Borrar contestaciones" onblur="this.style.color='black'"> </p>
</center></div></form></strong>

```

Podemos ver el [ejemplo primero en funcionamiento](#) en una hoja aparte.

El segundo ejemplo que se propone tiene el mismo script de puntuaciones, un script con la función del popup y el código de ortografía de G y J. El código del script del popup es:

```

<script LANGUAGE="JavaScript">
cnt = 0;
var wnd;
function popup(pagina)

```

```
{
  cnt++;
  if( wnd != null )
  {
    wnd.close();
    wnd = null;
  }
  wnd = open(pagina, "gl"+cnt, "width=650, left=70, top=128, height=270");
}

</script>
```

Y el código del cuerpo de la página:

```
<p align="center"><input TYPE="BUTTON" NAME="Ayuda" VALUE="AYUDA" onClick="popup('frayu05.htm');"> <p> </p>
<form NAME="bv" onsubmit>
<div align="center"><center>
<table border="8" width="100%" cellpadding="1" cellspacing="1">
<tr>
<td width="75%" bgcolor="#FFFFFF"><div align="center"><center><p><small><b><font
face="Verdana">Avísame cuando <font color="#FF0000">despe_e </font>el avión</font></b></small></td>
<td width="25%" align="center" bgcolor="#FFFFFF"><input Type="Button" Value=" g " onClick="resmal(); alert('El
sonido G suave con A, O, U, se escribe GA, GO, GU y con E, I, se escribe GUE, GUI. \n \nEjemplos: goma, galleta,
guapa, Miguel, guitarra, gorro, guerra.')">
<input Type="Button" Value=" gu " onClick="resbien(); this.style.color='red'"> </td></tr>
<tr align="center">
<td width="75%" bgcolor="#FFFFFF"><div align="center"><center><p><small><b><font
face="Verdana">Tu mamá es muy <font color="#FF0000">ele_ante</font></font></b></small></td>
<td width="25%" align="center" bgcolor="#FFFFFF"><input Type="Button" Value=" g "
onClick="resmal(); this.style.color='red'"> <input Type="Button" Value=" gu "
onClick="resmal(); alert('El sonido G suave con A, O, U, se escribe GA, GO, GU y con E, I, se escribe GUE, GUI. \n
\nEjemplos: goma, galleta, guapa, Miguel, guitarra, gorro, guerra.')">
</td></tr>
<tr align="center">
<td width="75%" bgcolor="#FFFFFF"><div align="center"><center><p><small><b><font
face="Verdana">Los Reyes Magos me <font color="#FF0000">tra_eron </font>una
bicicleta</font></b></small></td>
<td width="25%" align="center" bgcolor="#FFFFFF"><input Type="Button" Value=" g "
onClick="resmal(); alert('Llevan J las formas de los verbos que no tienen G ni J en el infinitivo. \n \nEjemplos: de
decir, dijeron; de traer, trajimos, trajeron.')">
<input Type="Button" Value=" j " onClick="resbien(); this.style.color='red'"> </td></tr>
</table></center></div>
<div align="center"><center><p><strong>
<input Type="Button" value="Puntuación" onClick="averiguarNota()">
<input TYPE="SUBMIT" VALUE="Borrar contestaciones" onBlur="this.style.color='black'"> </p>
</center></div></form></strong>
```

El segundo [ejemplo](#) lo podemos ver en una hoja aparte.

*Artículo por **Arturo Ramo García***

## **Marcar o desmarcar todos los checkbox de un formulario con Javascript**

El ejercicio que vamos a relatar en este artículo es bastante típico de trabajo con formularios en Javascript. Se trata de hacer un enlace para que se puedan marcar todos los campos

checkbox que haya en un formulario de una sola vez, es decir, sin tener que pulsarlos uno a uno para marcarlos todos. También haremos la función que permita deseleccionar todos los campos checkbox del formulario de una sola vez.

El ejercicio es simple de entender, [pero podemos verlo en funcionamiento en una página aparte](#) para hacernos una idea exacta de nuestras intenciones.

## El formulario HTML

Tenemos un formulario creado con HTML que es donde estarán los checkboxes que hay que marcar y desmarcar automáticamente. El formulario es muy sencillo. Lo vemos a continuación.

```
<form name="f1">
Nombre: <input type="text" name="nombre">
<br>
<input type="checkbox" name="ch1"> Opcion 1
<br>
<input type="checkbox" name="ch2"> Opcion 2
<br>
<input type="checkbox" name="ch3"> Opcion 3
<br>
<input type="checkbox" name="ch4"> Opcion 4
<br>
//Otro campo de formulario:
<select name=otro>
<option value="1">Seleccion 1
<option value="2">Seleccion 2
</select>
<br>
<input type="submit">
<br>
<br>
<a href="javascript:seleccionar_todo()">Marcar todos</a> |
<a href="javascript:deseleccionar_todo()">Marcar ninguno</a>
</form>
```

Lo único que debemos fijarnos es que hemos colocado diversos tipos de elementos en el formulario. En realidad sólo vamos a trabajar con el estado de los checkbox, pero hemos incluido otros elementos porque lo habitual en un formulario es que hayan elementos de varios tipos.

Al final del formulario tenemos un par de enlaces para marcar o desmarcar todos los checkboxes de una sola vez. Estos enlaces llaman a un par de funciones Javascript que veremos ahora.

## Funciones de Javascript

```
function seleccionar_todo(){
  for (i=0;i<document.f1.elements.length;i++){
    if(document.f1.elements[i].type == "checkbox")
      document.f1.elements[i].checked=1
  }
}
```

La función `seleccionar_todo()` realiza un recorrido por todos los elementos del formulario. Para hacer un recorrido por todos los campos se utiliza el array "elements", que guarda una referencia con cada elemento que haya dentro del formulario.

En el recorrido comprueba si el elemento actual es de tipo "checkbox" (recordar que el array `elements` contiene todos los elementos, pero sólo deseamos operar con los que sean checkbox) y en ese caso, simplemente se actualiza el atributo "checked" al valor 1, con lo que el checkbox se marcará.

```
function deseleccionar_todo(){
  for (i=0;i<document.f1.elements.length;i++)
    if(document.f1.elements[i].type == "checkbox")
      document.f1.elements[i].checked=0
}
```

La función `deseleccionar_todo()` es casi igual que la anterior. Realiza un recorrido a todos los elementos y en el caso que sean checkbox, se fija a cero el atributo "checked" para que la caja de selección se quede desmarcada.

El ejemplo no tiene más misterio. [Se puede ver en marcha en una página aparte.](#)

*Artículo por Miguel Angel Alvarez*

## Inhabilitar el menú contextual del navegador con Javascript

En este artículo vamos a mostrar un método para inhabilitar el menú contextual del navegador, que aparece pulsando con el botón derecho en cualquier área de la página. Así podemos evitar que el usuario tenga acceso a algunas de las opciones del navegador, como ver el código fuente.

Lo primero que hay que destacar es que este ejemplo no protege para nada el código de las páginas web que estamos publicando. Simplemente pone algunas trabas para ver cómo hemos hecho la página, pero cualquier usuario avisado podrá acceder al código de la página si realmente se lo propone.

Para empezar, desde la barra de menús del navegador, en "ver - código fuente", se puede acceder también al código fuente de las páginas. Así que si deseamos que no vean nuestro código, tendremos que mostrar la página en una nueva ventana del navegador, que debemos abrir mediante Javascript para que no incluya las barras de menús.

**Referencia:** en un manual de DesarrolloWeb.com tenemos [todo sobre control de ventanas secundarias](#). Allí podemos aprender a abrir ventanas como queramos.

Si conseguimos evitar mostrar la barra de menús y el menú contextual, todavía un usuario podría deshabilitar Javascript para intentar ver el menú contextual sin que se lo impida la página.

Pero debemos saber que, cuando se envía una página a un visitante, el archivo HTML se guarda en el disco duro local de ese usuario, por lo que en último caso, la persona interesada simplemente tiene que acceder a sus archivos temporales de Internet para localizar la página que tiene el código que desea visualizar. Como el archivo está físicamente en su ordenador, podrá hacer lo que desee con él: abrirlo, modificarlo, guardarlo con otro nombre, etc. Así que nuestros códigos nunca estarán totalmente seguros.

**Nota:** La mejor solución para proteger un código es escribirlo en el lado del servidor, con lenguajes como ASP o PHP. Al estar en el lado del servidor, los scripts se ejecutarán en el servidor y el visitante sólo recibirá el código generado de esa ejecución, no el propio código ASP o PHP.

Por tanto, no se puede hacer nada definitivo para ocultar un código que se ejecuta en el cliente, así que esta solución propuesta es sólo un detalle que puede entorpecer la captación de un código, pero no sirve para asegurarlo definitivamente.

Pues dicho esto, el código que vamos a proponer es mucho más sencillo de lo que se podría suponer. Simplemente utilizaremos un evento de Javascript que se llama "oncontextmenu" y depende de "document". Asignaremos una función a este evento, que se ejecutará en el momento que el usuario haga clic en el botón derecho para visualizar el menú contextual.

La función que vamos a asignar a este evento es la siguiente:

```
function inhabilitar(){
    alert ("Esta función está inhabilitada.\n\nPerdonen las molestias.")
    return false
}
```

La función muestra un mensaje de advertencia, pero fijémonos en el return false: es necesario para que no se llegue a mostrar el menú contextual, porque si no la ponemos, se mostraría el mensaje de alerta, pero a continuación se mostraría también el menú contextual, con lo que no serviría de nada el script.

Para asignar esta función al evento oncontextmenu, realizamos este código:

```
document.oncontextmenu=inhabilitar
```

Tan simple como eso. El script completo, que colocaríamos entre <head> y </head> quedaría así:

```
<script language=JavaScript>
<!--

function inhabilitar(){
    alert ("Esta función está inhabilitada.\n\nPerdonen las molestias.")
    return false
}

document.oncontextmenu=inhabilitar

// -->
</script>
```

Podemos [ver el ejemplo en marcha en una página aparte](#).

*Artículo por **Miguel Angel Alvarez***

## ***Página que cambia aleatoriamente el color de fondo***

Vamos a crear una página que tiene un color de fondo aleatorio, de modo que, cada vez que se visite, se muestre con un fondo distinto. Ahora bien, como el color de la página va a ser distinto cada vez, para asegurarnos que el texto se pueda leer correctamente, haremos que el texto de la página sea o blanco o negro, dependiendo de la gama del color de fondo: si es oscuro, el texto de la página será blanco y si el fondo es claro, el texto se verá en negro.

Hay que darse cuenta que, si el color es aleatorio, a veces saldrá más oscuro y a veces más claro. Para que se lea bien el texto, su color tiene que contrastar lo suficiente con el color de fondo, por eso calcularemos la oscuridad o claridad del fondo para fijar el color del texto.

Se puede [ver en marcha el ejemplo](#) que se va a desarrollar en esta página.

En un [artículo anterior del taller de Javascript](#) ya explicamos una manera de [conseguir un color aleatorio en Javascript](#).

Aunque en el mencionado artículo ya estaba la función Javascript para obtener un color aleatorio, la transcribimos aquí, pues hemos hecho un par de cambios minúsculos al código:

```
function dame_numero_aleatorio(superior, inferior){
    var numPosibilidades = (superior + 1) - inferior;
    var aleat = Math.random() * numPosibilidades;
    aleat = Math.floor(aleat);
    aleat = (inferior + aleat);
    return aleat
}

function dame_color_aleatorio(){
    color_aleat=""
    hexadecimal = new Array("0","1","2","3","4","5","6","7","8","9","A","B","C","D","E","F")
    var inferior = 0;
    var superior = hexadecimal.length-1;
    for (i=0;i<6;i++){
        color_aleat += hexadecimal[dame_numero_aleatorio(superior, inferior)];
    }
    return color_aleat
}
```

Ahora veremos la manera de conocer la oscuridad o claridad de un color con aleatorio generado por Javascript. Para calcular la oscuridad (o claridad) de un color en formato RGB hexadecimal, vamos a realizar varios pasos:

Tenemos que saber que, a mayores valores de RGB, el color resultante será mas claro. Si los valores de RGB son más bajos, el color será más oscuro. Los valores de R, G y B, por separado pueden ir, en decimal, desde 0 a 255. Diremos que es claro cuando sea mayor que  $255 / 2$  y que es oscuro cuando sea menor de  $255 / 2$ . Vamos a suponer un umbral a partir del cual el color lo consideramos más oscuro o más claro. Digamos que si sumamos por separado los valores rojo, verde y azul y nos dan más de la mitad de  $((255 + 255 + 255) / 2)$ , es que el color es claro. Si está por debajo de ese umbral, el color es oscuro.

1. Separaremos los valores hexadecimales de los tres componentes del color (rojo, verde y azul)
2. Convertiremos esos valores a enteros en base 10
3. Sumamos los valores de cada color, obteniendo un número entero, del que vamos a deducir la claridad u oscuridad.
4. Si el número resultado de la suma es menor que  $((255 + 255 + 255) / 2)$ , entonces es que el color de fondo es oscuro, luego el color del texto debe ser claro. Y al revés, si deducimos que el color de fondo es claro, entonces el color de texto tendrá que ser oscuro.

Esto se hace de la siguiente manera, en código Javascript, teniendo un color en un string con el formato #RRGGBB:

```
//obtengo un aleatorio
colorin = dame_color_aleatorio()

//voy a extraer las tres partes del color
rojo = colorin.substring(1,3)
verde = colorin.substring(3,5)
```

```
azul = colorin.substring(5,7)

//voy a convertir a enteros los string, que tengo en hexadecimal
introjo = parseInt(rojo,16)
intverde = parseInt(verde,16)
intazul = parseInt(azul,16)

//ahora sumo los valores
oscuridad = introjo + intverde + intazul

//si el valor oscuridad es menor que ((255 + 255 + 255) / 2) es que es un color más oscuro
//si es oscuro, el color del texto será blanco
if (oscuridad < (255+255+255)/2)
    colortexto = "#ffffff"
else
    colortexto = "#000000"
```

Para actualizar el color de fondo y de texto de una página web se podría hacer con estas líneas de código:

```
document.fgColor=colortexto
document.bgColor=colorin
```

Pero esto da un problema en algunos navegadores, al cambiar el color del texto, que no se puede hacer si previamente se ha escrito algo en la página.

Entonces, vamos a marcar el color del fondo y del texto utilizando los conocidos atributos bgcolor y text de la etiqueta <body>.

Escribiremos el <body> mediante javascript, colocando los valores de color aleatorio y color del texto que extraemos de las variables que los contienen.

```
document.write('<body bgcolor="' + colorin + '" text="' + colortexto + '">')
```

Eso es todo. Ya tenemos la página con el color de fondo aleatorio y el color del texto con suficiente contraste.

La página donde hemos implementado este ejercicio se puede [ver en marcha aquí](#). Podemos ver su código fuente para obtener script del ejemplo completo.

*Artículo por **Miguel Angel Alvarez***

## **Script de recarga de la página con javascript**

En algunas ocasiones necesitamos que una página recargue sus contenidos cada cierto tiempo, para mostrar información actualizada a las personas que la visitan. Esto es a veces típico de los servicios que muestran información en tiempo real, según se va produciendo.

Este artículo surge como respuesta a la duda de un visitante que, además, necesitaba que el tiempo que pasase entre cada recarga de la página fuera siempre distinto. Para ello simplemente hacemos que se recargue la página después de un número de segundos aleatorio.

Un tema que además es necesario para que todo funcione correctamente es que la página no se muestre desde la caché del navegador. Sabemos que cuando una página ya se ha solicitado previamente, se queda muchas veces en la caché de nuestros navegadores de modo que, si se vuelve a solicitar, se muestre la copia que tenemos almacenada localmente, en lugar de solicitarse y descargarse desde el servidor de Internet. En estos sistemas de recarga hay que asegurarse que los contenidos no se obtengan desde la caché, para que las actualizaciones se puedan ver correctamente con cada recarga de la página.

## La recarga con Javascript

Veamos cómo hemos resuelto todas estas necesidades, en este caso mediante Javascript.

Para empezar, tenemos que obtener un número aleatorio de segundos, para que la recarga se realice a intervalos irregulares, tal como nos solicitaban.

Para ello vamos a utilizar la siguiente función de cálculo de números aleatorios, que hemos comentado y probado en otros ejemplos relatados en DesarrolloWeb.com.

```
function aleatorio(inferior,superior){
    numPosibilidades = superior - inferior
    aleat = Math.random() * numPosibilidades
    aleat = Math.floor(aleat)
    return parseInt(inferior) + aleat
}
```

Llamaremos a la función para obtener un número aleatorio, en este caso entre 5 y 10:

```
num_aleatorio = aleatorio(5, 10)
```

Para solucionar el tema de que la página no se muestre desde la caché del navegador vamos a enviarle un parámetro por la URL, así la URL que solicitemos será siempre distinta y nuestro navegador se verá obligado a solicitar la página al servidor cada vez que se recargue. Podríamos haber utilizado otras técnicas como poner en la cabecera del http la orden para que no se guarde en la caché, pero por mi experiencia, esta es la única manera que nos asegura que todos los navegadores van a recargar la página solicitándola siempre al servidor.

Voy a generar un string para enviarlo por parámetro a esta misma página. Como decíamos, el parámetro lo pasaremos por URL. No haremos nada con ese dato, pero como cada vez será distinto, nos asegura que el navegador siempre solicitará al servidor la página, en vez de mostrar otra vez la que tiene en caché. Utilizaremos la fecha y tiempo para generar el dato que cambie siempre.

```
miFecha = new Date()
dato_url = miFecha.getYear().toString() + miFecha.getMonth().toString() + miFecha.getDate().toString() +
miFecha.getHours().toString() + miFecha.getMinutes().toString() + miFecha.getSeconds().toString()
```

En la variable dato\_url guardamos el año, seguido del mes, día, horas, minutos y segundos. Como todos los datos de fecha, se tienen que extraer desde un objeto date, que creamos con la sentencia new Date(). Luego a este objeto le invocamos diversos métodos para obtener los datos que necesitamos. Los datos se nos devuelven en tipo entero y para concatenarlos como si fueran string, necesitamos aplicarles el método toString(), que tienen todos los objetos de Javascript para convertirlos en cadenas.

Ya sólo queda realizar la recarga propiamente dicha. Para ello tenemos que aplicar un retardo,



que conseguiremos con la función `setTimeout()`, que recibe como primer parámetro la instrucción que se quiere ejecutar y como segundo parámetro, el tiempo en milisegundos que se quiere esperar.

```
setTimeout("window.location='pagina.html?parametro=" + dato_url + "'", num_aleatorio * 1000)
```

Si nos fijamos, hemos utilizado `window.location` para asignar una nueva URL al navegador. Luego hemos utilizado la variable `dato_url` para pasarla como parámetro. Además, para marcar el retardo entre recargas hemos utilizado la variable `num_aleatorio`, multiplicada por 1000 para pasar a milisegundos.

Esto es todo. Se puede [ver el ejemplo en marcha en este enlace](#).

Artículo por **Miguel Angel Alvarez**

## ***Cambiar el color a las celdas de una tabla con Javascript***

Veremos en este artículo un par de soluciones a una pregunta típica de trabajo con Javascript: cambiar el color a la celda de una tabla dinámicamente y como respuesta a acciones del usuario. Este ejemplo se puede realizar muy fácilmente y ofreceremos un par de soluciones, que podemos aplicar dependiendo del caso concreto en que nos encontremos.

La primera solución que vamos a ver es cómo cambiar el color de una celda al pasar el ratón por encima. Cuando entremos con el ratón en la celda debe cambiarse por un color y cuando se salga lo cambiaremos por otro.

El segundo caso que vamos a realizar es cambiar el color de la celda como respuesta a eventos de elementos que están fuera de la propia celda. En concreto, cambiaremos el color de una celda cuando se accione un formulario que está fuera de las celdas a cambiar.

### **Cambiar el color de una celda al pasar el ratón por encima**

Es un caso muy típico que se puede ver en muchas webs. Este ejemplo resulta bastante sencillo de hacer con CSS, sin necesidad de escribir ni una línea de código Javascript, pero a pesar de ello lo mostraremos aquí. No obstante, podemos consultar la FAQ [¿Cómo puedo cambiar el color de las celdas cuando paso el puntero por encima de ellas?](#) para obtener más información sobre CSS para hacer este efecto.

En el ejemplo tendremos una tabla con varias celdas y para cada una tendremos definido el evento `onmouseover` y `onmouseout`, para llamar a una función que se encargue de cambiar el color cuando se entre en la celda y cuando se salga, respectivamente.

La tabla tendrá esta forma:

```
<table width=100>
<tr>
  <td bgcolor="#dddddd" id="celda1" onmouseover="cambiar_color_over(this)"
onmouseout="cambiar_color_out(this)">Casilla numero 1</td>
</tr>
<tr>
  <td bgcolor="#dddddd" id="celda2" onmouseover="cambiar_color_over(this)"
```

```
onmouseout="cambiar_color_out(this)">Casilla numero 2</td>
</tr>
...
<tr>
  <td bgcolor="#dddddd" id="celda10" onmouseover="cambiar_color_over(this)"
onmouseout="cambiar_color_out(this)">Casilla numero 10</td>
</tr>
</table>
```

Las funciones que se encargan de alterar el color reciben por parámetro la palabra `this`, que es una referencia a la celda donde está el evento. Si estamos en el evento de la celda 1, `this` hace referencia a esa misma celda 1. Las funciones son las siguientes:

```
function cambiar_color_over(celda){
  celda.style.backgroundColor="#66ff33"
}
function cambiar_color_out(celda){
  celda.style.backgroundColor="#dddddd"
}
```

Como se puede ver, reciben la celda cuyo color se desea cambiar (que se envió con la palabra `this` en el manejador del evento). Luego ejecutan la sentencia necesaria para cambiar el color.

### Cambiar el color a una celda sin pasar la referencia `this`

El segundo caso que habíamos adelantado que íbamos a hacer, era cambiar el color a una celda desde otra parte de la página. Hemos visto que, si estamos en la propia celda, podemos enviar una referencia de la propia casilla con la palabra `this`, pero si no estamos codificando un evento de la celda, sino que estamos en otro lugar del código de la página, no tendremos posibilidad de enviar esa sencilla referencia.

Entonces se nos hace necesario obtener la referencia de la celda por otro mecanismo. Entra en juego la función de Javascript (que en realidad es un método del objeto `document`) llamada `getElementById()`. Esta función recibe el nombre de un identificador y devuelve una referencia al elemento que tiene ese identificador.

Se asignan identificadores a los elementos de HTML con el atributo `id`. De esta manera:

```
<td id="celda1">
```

Pudimos ver en el código de la tabla, escrito líneas arriba, que cada celda tenía un identificador definido. Utilizaremos ese identificador para obtener la referencia a la celda que deseamos alterar su color.

Por ejemplo, si queremos obtener una referencia a la celda con identificador "celda1", utilizaríamos la llamada a esa función así:

```
celda = document.getElementById("celda1")
```

Luego, con la referencia de la celda, podemos cambiar el color como vimos antes:

```
celda.style.backgroundColor="#dddddd"
```

En nuestro ejemplo para ilustrar este método hemos creado un formulario con dos campos de texto y un botón. En el primer campo de texto escribiremos el número de la celda cuyo color queremos cambiar. En el segundo, escribiremos el nombre del color que queremos poner a la

casilla, o su código RGB. Cuando apretemos al botón llamaremos a una función que se encargará de cambiar el color de la celda.

El formulario tendrá un código como este:

```
<form name=fcolor>
Numero de celda: <input type=text name=celda size=3>
<br>
Color: <input type=text name=micolor size=8>
<br>
<input type=button value="Cambiar color" onclick="cambia_color()">
</form>
```

Y la función javascript que se encargará de cambiar el color tendrá este código:

```
function cambia_color(){
    celda = document.getElementById("celda" + document.fcolor.celda.value)
    celda.style.backgroundColor=document.fcolor.micolor.value
}
```

Como se puede ver, para obtener la referencia utilizamos la función `document.getElementById()` y le pasamos el id de la celda que queremos cambiar su color. El identificador de la celda se compone por la palabra "celda" y el número de la celda, que sacamos del formulario.

Luego se pone en la celda el color que se saca del otro campo del formulario.

### Cambio de color en marcha

Hemos realizado una página con todo el código Javascript comentado en este ejercicio para ilustrar el modo de cambiar el color a las celdas de una tabla. Se puede ver en [http://www.desarrolloweb.com/articulos/ejemplos/tallerjs/cambiar\\_color\\_celda.html](http://www.desarrolloweb.com/articulos/ejemplos/tallerjs/cambiar_color_celda.html)

*Artículo por Miguel Angel Alvarez*

## Popups DHTML – OpenPopups

Todos sabemos ya que la mayoría de los navegadores disponen de sistemas para bloquear los molestos popups, y cuando estos no los bloquean, existen barras de navegación, como la de Google, que también bloquea la presentación de popups. La mayoría de las veces, estos popups son muy molestos y tenemos que celebrar que ahora la mayoría se puedan detectar y no permitir su apertura, pero muchos de nuestros sitios utilizan este sistema para mostrar información legítima que nuestros visitantes deberían conocer.

En cualquier caso, existen métodos para mostrar popups que pueden resultar más complicados de bloquear, como los popups DHTML, que son una emulación de las ventanas secundarias, pero que funciona por capas y HTML dinámico para mostrar u ocultar su contenido. Este tipo de popups no se tienen en cuenta como ventanas secundarias, por lo que no se bloquean.

Tan sólo los navegadores que tengan deshabilitado Javascript dejarán de mostrar estos popups. Recordemos que Internet Explorer, depende de cómo esté configurado, a veces te muestra un mensaje de alerta cuando se intenta ejecutar un script en Javascript. El usuario es

el responsable de permitir, o no, ejecutar scripts en la página. Por eso no es tan raro que incluso los popups DHTML se puedan bloquear, pero por lo menos significan un avance con respecto a las ventanas secundarias habituales.

## OpenPopups

Todo lo anterior sirva para presentar un script Javascript Open Source (gratuito y de código libre) para crear Popups DHTML. Merece la pena conocer este script, porque seguro puede resultar muy interesante para nuestras páginas web.

La web donde se puede descargar el sistema de popups DHTML es:

<http://www.openwebware.com/products/openpopups/>

Desde dicha web se pueden descargar los archivos necesarios para la instalación del sistema de popups y algún código de ejemplo. De todos modos, lo explicaremos aquí en español, para que se pueda entender por todos.

**Referencia:** En DesarrolloWeb.com hemos publicado algunos otros artículos sobre cómo hacer un popup DHTML, pero utilizando la librería Cross-Browser. Pueden ser de interesante lectura para quien quiera profundizar en el tema o encontrar otras posibilidades para realizar popups DHTML. Está en nuestro manual de [Taller de Cross-Browser DHTML](#).

Se tienen que descomprimir los archivos que se descargan desde la web, manteniendo la misma estructura de directorios.

Una vez están descomprimidos, en un directorio dentro de nuestro sitio web, que llamaremos por ejemplo "d\_openpopups", ya podemos accederlos desde cualquier página para mostrar popups DHTML. Para ello, lo primero es incluir el Javascript con la librería.

```
<script language="JavaScript" type="text/javascript"
src="/d_openpopups/openpopups/openpopups.js"></script>
```

Donde "d\_openpopups" debe ser el directorio donde hemos descomprimido los archivos. Tal como está escrita la ruta hacia el script, se supone que hemos puesto este directorio en la raíz del directorio de publicación de la web.

Luego, tenemos que añadir un evento onload en la etiqueta <body>, para ocultar los popups al cargar la página.

```
<body onload="hideDiv()">
```

La función hideDiv() recibe el número de popups que vamos a utilizar en la página. Si tenemos un solo popup DHTML llamaremos pasando un 1 como parámetro: hideDiv(1). Si tenemos 5 popups DHTML, le pasaremos un 5 como parámetro: hideDiv(5).

A continuación, tenemos que crear las capas con el código fuente de los popups a mostrar. Algo como:

```
<div id="Div1">
  Código del Popup
</div>
```

Hay que fijarse que la capa tiene como identificador (atributo id) "Div1". Eso es para el popup 1. Si tuviéramos otros popups, deberíamos darles nombres con números consecutivos: Div2, Div3...

Para acabar, tenemos que hacer la llamada a la función Javascript que debe mostrar el popup. Esa función se llama createWindow() y recibe varios parámetros:

1. Título de la ventana
2. Ancho de la ventana (el alto será el necesario para que quepa todo el contenido)
3. Color de fondo de la ventana
4. El identificador de la capa (sólo el numero, 1, 2, 3...)

5. Si queremos que se muestre el icono para minimizar (1 para mostrarlo y 0 si no queremos que se muestre)
6. La posición "left" de la ventana (el número de píxeles a la izquierda de la ventana)
7. La posición "top" de la ventana (el número de píxeles que debe haber arriba de la ventana).

Por ejemplo, una llamada posible a esta función sería:

```
createWindow('Título', 300, '#ffff88', 1, 0, 100, 25);
```

Un detalle que a nosotros nos ha hecho falta cambiar para que todo funcionase correctamente, aunque no he visto explicado nada de esto en la documentación del producto, son los directorios de las imágenes y las declaraciones de estilos que utilizan los popups DHTML. Esos directorios vienen especificados en el archivo de scripts javascript llamado openpopups.js.

En las siguientes líneas del código se especifican los directorios de las imágenes y los CSS:

```
// CSS Directory
cssDir = "/d_styles/";
```

```
// Images Directory
imageDir = "images/";
```

En principio, según entiendo, no habría por qué tocar esas líneas, porque no he modificado la estructura de directorios del archivo de descarga, pero si no las toco los ejemplos no funcionan correctamente. Para que las rutas se encuentren, he tenido que colocar la estructura de directorios desde la raíz del dominio hasta las carpetas donde están los archivos CSS y las imágenes. Sería algo como esto:

```
// CSS Directory
cssDir = "/d_openpopups/openpopups/styles/";

// Images Directory
imageDir = "/d_openpopups/openpopups/images/";
```

## Código completo

Vamos a mostrar el código de una página que tiene dos popups DHTML y con un par de métodos de carga de los popups, uno por medio de un botón y otro por medio de un enlace.

```
<html>
<head>
  <title>Ejemplo OpenPopups</title>
  <script language="JavaScript" type="text/javascript" src="/d_openpopups/openpopups/openpopups.js">
  </script>
</head>
<body onLoad="hideDiv(2);">
Esta página muestra un par de popups DHTML.
<br>
<br>
Esperamos que sean interesantes.
<form>
  <input type="button" value="Abrir Popup DHTML 1" onClick="createWindow('Ejemplo 1', 150, '#ffff88', 1, 1, 20,
40);">
</form>
<p>
Ahora veamos el ejemplo 2, apertura con un enlace:
<a href="#" onClick="createWindow('Anuncio MercadoProfesional.com', 468, '#EEEEEE', 2, 0, 240, 165);">Abre el
segundo popup</a>

<div id="Div1">
  <div style="border: 1px solid #ff8800; background-color: #FFFF88; padding: 5px;">
    <b>Aquí pondríamos colocar tanto texto como queramos! Y todo tipo de contenido HTML!
  </div>
```

```
<ul>
<li>Con listas</li>
<li>Enlaces</li>
<li>Tablas</li>
<li>...</li>
</ul>
</div>

<div id="Div2">
  <div align="center"><a href="http://www.mercadoprofesional.com" target="_blank"></a></div>
</div>

</body>
</html>
```

Se puede [ver el ejemplo en marcha en una página aparte](#).

*Artículo por **Miguel Angel Alvarez***

## **Validar la extensión de un archivo a subir con Javascript**

En este artículo vamos a mostrar como validar la extensión de un archivo mediante Javascript. Tenemos un formulario con un campo file y cuando se va a enviar el archivo, se realiza una comprobación para ver si la extensión está entre las permitidas. Si lo estuviera, se realiza el envío del formulario para hacer el upload del fichero.

En este script sólo se realiza la comprobación de la extensión, en ningún caso la recepción del fichero y su almacenamiento en el servidor, pues con Javascript no podemos realizar esas acciones, ya que es un lenguaje que se ejecuta en el cliente y no tiene acceso al servidor para hacer un upload.

En este ejemplo vamos a definir un formulario con un campo file y un botón de enviar que llama a una función, que será encargada de comprobar si la extensión está permitida y submitir el formulario si todo es correcto. El formulario sería el siguiente:

```
<form method=post action="#" enctype="multipart/form-data">
<input type=file name="archivoupload">
<input type=button name="Submit" value="Enviar" onclick="comprueba_extension(this.form,
this.form.archivoupload.value)">
</form>
```

Ahora veremos la función `comprueba_extension()` que recibe una referencia al formulario y la ruta del archivo que deseamos subir desde dentro de nuestro ordenador. La función realizará una serie de comprobaciones que veremos a continuación. El código será el siguiente:

```
function comprueba_extension(formulario, archivo) {
  extensiones_permitidas = new Array(".gif", ".jpg", ".doc", ".pdf");
  mierror = "";
  if (!archivo) {
    //Si no tengo archivo, es que no se ha seleccionado un archivo en el formulario
    mierror = "No has seleccionado ningún archivo";
  }else{
    //recupero la extensión de este nombre de archivo
    extension = (archivo.substring(archivo.lastIndexOf("."))).toLowerCase();
    //alert (extension);
  }
```

```
//compruebo si la extensión está entre las permitidas
permitida = false;
for (var i = 0; i < extensiones_permitidas.length; i++) {
    if (extensiones_permitidas[i] == extension) {
        permitida = true;
        break;
    }
}
if (!permitida) {
    mierror = "Comprueba la extensión de los archivos a subir. \nSólo se pueden subir archivos con extensiones: "
+ extensiones_permitidas.join();
} else {
    //submito!
    alert ("Todo correcto. Voy a submitir el formulario.");
    formulario.submit();
    return 1;
}
}
//si estoy aqui es que no se ha podido submitir
alert (mierror);
return 0;
}
```

Lo primero que hacemos es definir un array con las extensiones permitidas para hacer el upload. También definimos una variable llamada mierror, donde vamos a guardar el texto explicativo del error, si es que se produce.

Luego comprobamos si hemos recibido una ruta del archivo. Si no hay tal ruta, se define el error correspondiente "No has seleccionado ningún archivo". En caso contrario es que tenemos una ruta, con lo que vamos a buscar el nombre del archivo.

La ruta que podemos recibir puede tener una forma como esta:  
C:\directorio\otro directorio\archivo.doc

De la ruta nos interesa obtener sólo la extensión del archivo. Por lo que vamos a obtener la parte que hay después del último punto. Esto se hace utilizando varios métodos de los objetos string de javascript:

```
extension = (archivo.substring(archivo.lastIndexOf(".")).toLowerCase());
```

Simplemente estamos seleccionando la parte del string que hay después del último punto. Y estamos pasando la extensión a minúsculas, por si acaso estuviera escrita con mayúsculas.

A continuación, para comprobar si esta extensión está entre las permitidas hacemos un bucle for, que recorre todo el array de extensiones permitidas y las va comparando a la extensión que hemos recortado del nombre del archivo. En el momento que encuentra una coincidencia se sale del bucle y pone la variable booleana permitida a true. Si no encontrase coincidencias esa variable booleana quedaría como false.

Luego se comprueba la variable booleana permitida. Si está en false es que no se permite la extensión, entonces defino el correspondiente error. Si estaba a true es que la extensión figuraba entre las permitidas, entonces se envía el formulario y se sale de la función.

Al final de la función se muestra el posible error que se haya detectado. Sólo se mostrará el error si no se llegó a mandar el formulario, porque si se hubiera enviado, se habría salido anteriormente de la función.

Esperamos que esta validación haya sido de utilidad. Se puede [ver en funcionamiento en una página aparte](#).

Artículo por **Miguel Angel Alvarez**

## Copiar en el portapapeles con Javascript

El script que vamos a comentar sirve para seleccionar un texto que hay en un textarea y copiarlo en el portapapeles. Esta es una tarea que se puede realizar fácilmente en Internet Explorer, pero que da algún mayor quebradero de cabeza en Firefox.

Internet Explorer dispone de un par de mecanismos para copiar texto en el portapapeles. En [MSDN](#) he obtenido una referencia para hacer esto de una manera sencilla, a partir de la cual he confeccionado el siguiente script:

```
<form name="f1">
<textarea cols="50" rows="5" name="campo1">Hola. Este texto es el que vamos a seleccionar y copiar.</textarea>
<br>
<input type="button" value="Copiar" onclick="copia_portapapeles()">
</form>

<script language="javascript">

function copia_portapapeles(){
    document.f1.campo1.select()
    window.clipboardData.setData("Text", document.f1.campo1.value);
}
</script>
```

Tenemos un formulario con un textarea y un botón. En el textarea hay un texto y al pulsar el botón se seleccionará y copiará el texto. Utilizamos el método setData() del objeto clipboardData. Este método recibe dos parámetros. El primero indica el tipo de formato de la información a copiar y el segundo es la cadena de texto que se va a introducir en el portapapeles.

Lo malo es que este script no funciona con Firefox. Esto es porque este navegador no tiene el objeto clipboardData. Parece ser que este objeto es propio de Internet Explorer, aunque Firefox también debe considerar que copiar en el portapapeles conlleva algunos posibles agujeros de seguridad, por lo que no permite en principio trabajar con el portapapeles.

Para que esta función no lance un error javascript al ejecutarla, tenemos que prevenir la llamada al método setData() del objeto clipboardData, para que no se realice si no existe tal objeto. La función quedaría de la siguiente manera:

```
function copia_portapapeles(){
    document.f1.campo1.select()
    if (window.clipboardData){
        window.clipboardData.setData("Text", document.f1.campo1.value);
    }
}
```

Simplemente se comprueba si existe window.clipboardData, evaluándolo en un if, antes de ejecutar su método. En Firefox, que no tiene window.clipboardData, simplemente se



seleccionará el texto, pero no se llegará a copiar en el portapapeles.

Para hacer un script que copie un texto en el portapapeles y funcione en Firefox se debe utilizar otro mecanismo más complejo, que no he investigado a fondo. El tema es que, aunque funciona, debería configurarse Firefox para permitir ese tipo de acciones, o realizar un script firmado (<http://www.mozilla.org/projects/security/components/signed-scripts.html>).

Os paso la URL donde se puede obtener más información sobre este script para copiar en el portapapeles compatible con Firefox es:

[http://www.krikk.it/howto\\_javascript\\_copy\\_clipboard.html](http://www.krikk.it/howto_javascript_copy_clipboard.html)

El script que utilizan es el siguiente:

```
<script language="javascript" type="text/javascript">
<!--
function copy_clip(meintext)
{
  if (window.clipboardData)
  {

    // the IE-manier
    window.clipboardData.setData("Text", meintext);

    // waarschijnlijk niet de beste manier om Moz/NS te detecteren;
    // het is mij echter onbekend vanaf welke versie dit precies werkt:
  }
  else if (window.netscape)
  {

    // dit is belangrijk maar staat nergens duidelijk vermeld:
    // you have to sign the code to enable this, or see notes below
    netscape.security.PrivilegeManager.enablePrivilege("UniversalXPConnect");

    // maak een interface naar het clipboard
    var clip = Components.classes["@mozilla.org/widget/clipboard;1"]
      .createInstance(Components.interfaces.nsIClipboard);
    if (!clip) return;

    // maak een transferable
    var trans = Components.classes["@mozilla.org/widget/transferable;1"]
      .createInstance(Components.interfaces.nsITransferable);
    if (!trans) return;

    // specificeer wat voor soort data we op willen halen; text in dit geval
    trans.addDataFlavor('text/unicode');

    // om de data uit de transferable te halen hebben we 2 nieuwe objecten
    // nodig om het in op te slaan
    var str = new Object();
    var len = new Object();

    var str = Components.classes["@mozilla.org/supports-string;1"]
      .createInstance(Components.interfaces.nsISupportsString);

    var copytext=meintext;

    str.data=copytext;

    trans.setTransferData("text/unicode",str,copytext.length*2);

    var clipid=Components.interfaces.nsIClipboard;
```

```
if (!clip) return false;

clip.setData(trans,null,clipid.kGlobalClipboard);

}
alert("Following info was copied to your clipboard:\n\n" + meintext);
return false;
}
//-->
</script>
```

A su vez, esta página ha sacado la pista de una web en alemán, por eso aparecen comentarios en dicho idioma. Las referencias son:

<http://www.xulplanet.com/tutorials/xultu/clipboard.html>

<http://www.codebase.nl/index.php/command/viewcode/id/174>

*Artículo por **Miguel Angel Alvarez***

## **Esquinas redondeadas con CSS y Javascript, sin imágenes**

Hacer cajas con las esquinas redondeadas es una de las típicas preguntas que se pueden hacer sobre CSS. Existen varias soluciones al problema, que hemos visto en algunos [artículos de DesarrolloWeb.com](http://www.desarrolloweb.com), pero en estas cajas con CSS utilizábamos imágenes, que es lo más directo.

Ahora vamos a ver un método para crear cajas con las esquinas redondeadas que no utiliza imágenes, sino simplemente CSS y un poco de Javascript. El desarrollo de esta técnica no corre por nuestra cuenta. En este caso simplemente vamos a comentar un desarrollo de Alessandro Fulciniti que se llama Nifty Corners Cube que ya está en su tercera versión y se puede ver en la URL: <http://www.html.it/articoli/niftycube>

Como hemos dicho, el sistema crea cajas o bloques de contenido con esquinas redondeadas o suavizadas, pero sin usar imágenes. En lugar de imágenes utiliza CSS y Javascript compatible con la mayoría de los navegadores. Aunque según se informa en la página de producto, los navegadores Internet Explorer 5, así como los que tengan deshabilitado Javascript, verán cajas cuadradas en lugar de ver las esquinas redondeadas. En la primera versión ya se presentaba como una interesante opción, pero en la segunda y sobretodo en la tercera versión se ha visto mejorado el script y simplificado su manejo, dentro de lo posible.

Lo mejor de todo es que se presenta con licencia GPL, luego cualquiera puede usarlo en sus desarrollos libremente.

Dispone de tres elementos para ponerlo en marcha:

1. Un archivo Javascript
2. Un archivo CSS
3. Llamadas Javascript dentro de las páginas que quieran mostrar cajas con esquinas redondeadas.

En realidad el archivo CSS para hacer esquinas redondeadas lo incluye internamente Javascript, por lo que nosotros simplemente tendremos que incluir un archivo externo .js, que es el código Javascript con las funciones que sirven para hacer las esquinas redondeadas.

```
<script type="text/javascript" src="niftycube.js"></script>
```

Luego será necesario hacer unas llamadas a Javascript para redondear las esquinas de las capas que deseemos. De esta manera:

```
<script type="text/javascript">
window.onload=function(){
Nifty("div#box","big");
}
</script>
```

Como se puede ver, se ha definido una función que se ejecutará con el evento onload (cuando termine de cargarse la página). Esa función invoca a otra función llamada Nifty() que está definida en el Javascript que habíamos incluido como archivo aparte. La función Nifty() recibe dos parámetros. El primero es el selector CSS de la capa que se desea redondear y el segundo sirve para indicar opciones específicas del redondeo.

El primer parámetro, que decíamos es el selector CSS, tiene mucha versatilidad. Permite especificar el redondeo de elementos de la página variados, como todas las apariciones de una etiqueta concreta, una clase de CSS, una etiqueta con un identificador determinado, etc. En la documentación del producto se pueden ver todos los tipos de selectores CSS, pero algunos son estos:

Selector de etiqueta: "p" o bien "h2"

Se mostrarán con los bordes redondeados todas las apariciones de una etiqueta, como los párrafos o los encabezamientos h2.

Selector por identificador "div#capax" o bien "p#parrafoy"

Esto sirve como selector de las siguientes etiquetas con sus identificadores:

```
<span class=codigo>
<div id="capax">Div con su identificador</div>
<p id="parrafoy">P con su identificador</p>
```

Selector de clase "div.nave" o bien "span.fuentepequena".

Cada vez que apliquemos esa clase, se pondrá con las esquinas redondeadas.

Selector descendente "div#cabecera h1"

Hace referencia a la etiqueta h1 dentro de la capa con id="cabecera".

Hay otros selectores que se pueden ver en la documentación. Además en el primer parámetro se pueden especificar varios selectores a la vez, separados por una coma:

```
Nifty("div#box,div#prueba,p","big");
```

Esto afectará a las capas box, prueba y a todas las etiquetas de párrafos.

El segundo parámetro son las opciones de redondeo que se aplicarán a los selectores en cada función. Las distintas opciones se deben escribir separadas por comas. Existe una lista de opciones bastante grande, pero comentamos algunas que parecen más útiles:

tr: redondear sólo la esquina superior derecha.

tl: redondear sólo la esquina superior izquierda.

br: redondear sólo la esquina inferior derecha.

bl: redondear sólo la esquina inferior izquierda.  
top: esquinas de arriba  
bottom: esquinas de abajo  
left: esquinas de la izquierda  
right: esquinas de la derecha  
all: todas las esquinas (es la opción por defecto)  
none: ninguna esquina se redondea

Con estas opciones juntas se pueden definir redondeos de varias esquinas, pero no todas, para que quede alguna sin redondear:

```
Nifty("p","tl,bl,br");
```

small: se utilizan esquinas pequeñas, de 2px  
normal: se utilizan esquinas normales 5px (opción por defecto)  
big: se utilizan esquinas grandes de 10px

Con estas opciones definimos tamaños de las esquinas.

Luego hay otras opciones que son un poco menos claras, que dejamos para que cada uno se las estudie si las llegase a necesitar: transparent, fixed-height, same-height. Parece útil la opción same-height, que hace que todas las capas tengan la misma altura, por si queremos hacer un diseño en distintas columnas donde cada columna tiene la misma altura.

### Ejemplo muy básico de esquinas redondeadas sin imágenes

Hemos hecho unas pruebas del sistema y además hemos tratado de conseguir el código más básico con el que probar este script. El resultado es el siguiente ejemplo, en el que se redondean las esquinas de todos los párrafos de la página. Hemos colocado dos párrafos con dos colores distintos de fondo.

```
<html>
<head>
<title>Ejemplo esquinas redondeadas basico</title>
<script type="text/javascript" src="niftycube.js"></script>
<script type="text/javascript">
window.onload=function(){
Nifty("p");
}
</script>
</head>
<body>
<p style="background:#ccccff;padding:5px;">
Esto es una prueba
<br>mola cantidad!!!
</p>
<br>
<p style="background:#cccccc;padding:5px;">
De nuevo, Hola mi amigo!!!
</p>
</body>
</html>
```

Este ejemplo se puede [ver en una página aparte](#).

Haciendo pruebas hemos comprobado que si ponemos el color de fondo de un párrafo con el

nombre de un color en lugar de su código RGB, no funciona.

```
<p style="background: red;">
```

Si cambiamos el color de fondo de la página, las esquinas redondeadas también se ven perfectamente.

```
<body style="background: #66ff00;">
```

Pero si ponemos el color de fondo de la página con un nombre en lugar de RGB, tampoco funciona.

Podemos ver multitud de ejemplos y descargar los códigos necesarios para hacer esquinas redondeadas y los ejemplos, en la web del propio sistema:

<http://www.html.it/articoli/niftycube/index.html>

*Artículo por **Miguel Angel Alvarez***

## Rotación de banners con Javascript

Vamos a hacer un sistema de rotación de banners bien simple, programado en el lado del cliente con Javascript. En un espacio de la página se mostrará un banner y al cabo de unos segundos ese banner se cambiará por otro y luego por otro, hasta acabar con los banners disponibles, empezando luego otra vez por el primer banner. Todo ello en la misma página, sin necesidad de refrescarla.

El sistema es sencillo. Simplemente tenemos que hacer una función que se encargue de cambiar la imagen del banner actual por la imagen del banner siguiente. Además tendrá que cambiarse el enlace, pues suponemos que cada banner lleva a una página distinta.

Podemos [ver el ejemplo en marcha](#), a fin de hacerse una idea concreta del objetivo de este ejercicio.

Lo primero que vamos a ver es el código HTML que tendremos para visualizar el banner. Es el siguiente:

```
<a href="http://www.salvarpatrimonio.org/"></a>
```

Como se puede ver, es un simple HTML para incluir un enlace y una imagen. Tanto el atributo href del banner como el src de la imagen están definidos con el primer banner a mostrar, para que mientras se cargue la página se muestre un anuncio, en lugar de aparecer la imagen rota. Nos debemos fijar también que a la imagen le hemos puesto un atributo name="banner", que es un nombre con el que haremos referencia a la imagen más tarde en el código Javascript.

Para definir los distintos banner y las URLs a las que van dirigidos cada banner, se crearán unos arrays con las imágenes de los banner y las URLs de destino.

```
//creo array de imágenes  
array_imagen = new Array(4)  
array_imagen[0] = new Image(120,41)  
array_imagen[0].src = "salvarpatrimonio.gif"  
array_imagen[1] = new Image(120,41)  
array_imagen[1].src = "guiarte.gif"
```

```
array_imagen[2] = new Image(120,41)
array_imagen[2].src = "estiloymoda.gif"
array_imagen[3] = new Image(120,41)
array_imagen[3].src = "websitealbum.gif"
```

```
//creo el array de URLs
array_url = new Array(4)
array_url[0] = "http://www.salvarpatrimonio.org/"
array_url[1] = "http://www.guiarte.com/"
array_url[2] = "http://www.estiloymoda.com/"
array_url[3] = "http://www.websitealbum.com/"
```

Como hemos visto, en el array de imágenes hemos cargado una serie de objetos image, con el mismo tamaño y con sus atributos src (origen de las imágenes) distintos. Tenemos un [manual que explica el tratamiento de imágenes con Javascript](#), que será interesante si no conocemos el objeto image y sus propiedades.

En el array de URLs hemos cargado las distintas direcciones donde se dirigiría al navegador al pinchar cada banner.

Ahora veamos la función que se encargará de:

- 1) Actualizar la imagen para mostrar el banner siguiente
- 2) Actualizar el link del banner siguiente
- 3) Incrementar una variable que lleva la cuenta del banner que se tiene que mostrar
- 4) Llamarse a si misma con un retardo, para seguir la rotación de los banner.

Por cierto, la variable que se encargará de llevar la cuenta del banner que hay que mostrar se debe definir fuera de la función, para que sea global y su valor permanezca entre las distintas llamadas a la función.

```
//variable para llevar la cuenta de la imagen siguiente
contador = 0

//función para rotar el banner
function alternar_banner(){
    window.document["banner"].src = array_imagen[contador].src
    window.document.links[0].href = array_url[contador]
    contador ++
    contador = contador % array_imagen.length
    setTimeout("alternar_banner()",1000)
}
```

El código de la función es sencillo, pero contiene la mayor complejidad de este taller.

Con la primera sentencia se indica que el atributo src la imagen llamada "banner" debe actualizarse al src del array de imágenes que toca en este momento.

Con la segunda línea, indicamos que la URL de destino del enlace debe actualizarse a la que toque en el array\_url. El enlace lo tenemos que referenciar con el índice cero "0" dentro del array de links de la jerarquía de objetos del navegador, pues es el primer enlace que hay en la página. Si hubiera otros enlaces por delante en el código HTML de la página, tendríamos que cambiar el índice "0" por el número de enlace correspondiente al banner.

Para entender estas dos primeras líneas de código sería bueno conocer la [jerarquía de objetos del navegador](#) y aprender a [trabajar con esta jerarquía de objetos de Javascript](#).

Luego incrementamos en uno la variable contador, para pasar al siguiente índice de arrays de imágenes y banners, para que en la sucesiva llamada a la función se muestre el banner siguiente. Además, a fin de que no se nos desborde la variable contador, hacemos una operación "resto de la división" entre la longitud del array de banners.

Por último, hacemos una nueva llamada a la función, pero con un retardo de 1000

milisegundos, es decir, un segundo.

Con esto estaría todo el script comentado. Sólo nos quedaría hacer la llamada a la función una vez cargada la página, para que comiencen a rotar los banner una vez se haya terminado de mostrar la página. Esto lo hacemos con el atributo onload de la etiqueta <body>

```
<body onload="alternar_banner()">
```

Ahora podemos ver el código completo de la página web:

```
<html>
<head>
  <title>Rotación de banners con Javascript</title>
<script>
//creo array de imágenes
array_imagen = new Array(4)
array_imagen[0] = new Image(120,41)
array_imagen[0].src = "salvarpatrimonio.gif"
array_imagen[1] = new Image(120,41)
array_imagen[1].src = "guiarte.gif"
array_imagen[2] = new Image(120,41)
array_imagen[2].src = "estiloymoda.gif"
array_imagen[3] = new Image(120,41)
array_imagen[3].src = "websitealbum.gif"

//creo el array de URLs
array_url = new Array(4)
array_url[0] = "http://www.salvarpatrimonio.org/"
array_url[1] = "http://www.guiarte.com/"
array_url[2] = "http://www.estiloymoda.com/"
array_url[3] = "http://www.websitealbum.com/"

//variable para llevar la cuenta de la imagen siguiente
contador = 0

//función para rotar el banner
function alternar_banner(){
  window.document["banner"].src = array_imagen[contador].src
  window.document.links[0].href = array_url[contador]
  contador ++
  contador = contador % array_imagen.length
  setTimeout("alternar_banner()",1000)
} </script>
</head>

<body onload="alternar_banner()">

<a href="#"></a>

</body>
</html>
```

El [ejemplo se puede ver en marcha en este enlace](#).

**Referencia:** Tenemos otro [taller de DHTML y Javascript para rotar banners](#) que permite utilizar todo tipo de banners, con cualquier código, como banners Flash, banners HTML, imágenes, etc.

*Artículo por **Miguel Angel Alvarez***

## **Detectar la resolución de la pantalla del usuario con Javascript**

Con Javascript se puede calcular la resolución de la pantalla del usuario que nos visita. Atención, no nos referimos al tamaño de la ventana del navegador, sino al tamaño total en pixels que tengamos configurado en nuestro sistema.

Las resoluciones de pantalla pueden ser valores como 800x600, 1024x760, 1280x800... y se configuran por el usuario en el panel de control, en propiedades de pantalla.

Nosotros con Javascript podemos acceder a esos valores a través del objeto screen:

Con screen.width obtenemos el ancho en pixels de la definición de pantalla.  
Con screen.height obtenemos el alto en pixels.

Entonces, si quisiéramos escribir en la página los valores ancho y alto de la resolución de pantalla podríamos utilizar un javascript como este:

La resolución actual de tu pantalla es:

```
<script language="JavaScript">
document.writeln(screen.width + " x " + screen.height)
</script>
```

Si lo deseamos, podemos hacer distintas cosas dependiendo de la definición de pantalla del usuario, por ejemplo, con una estructura if:

Tu pantalla la consideramos:

```
<script language="JavaScript">
if (screen.width<1024)
  document.write ("Pequeña")
else
  if (screen.width<1280)
    document.write ("Mediana")
  else
    document.write ("Grande")
</script>
```

Este código mostrará el tamaño de la ventana como pequeña (menos de 1024 píxeles de ancho), mediana (Mayor o igual a 1024 píxeles de ancho y menor de 1280) o grande (1280 píxeles de ancho o más). Pero podríamos haber hecho otras cosas distintas dependiendo de la resolución detectada.

Los ejemplos de este artículo se pueden [ver en una página aparte](#).

*Artículo por **Miguel Angel Alvarez***

## **Esconder la URL de un enlace en la barra de estado**

Cuando ponemos el ratón encima de un enlace se muestra en la barra de estado del navegador



la URL a la que va dirigido. Esto es algo que resulta muy útil cuando navegamos, porque podemos ver, antes de pulsar el enlace, la dirección a la que va a dirigirnos el navegador si hacemos clic. Pero por muy útil que resulte a los navegantes, a veces los webmaster por unas razones u otras preferimos que no se vea la URL a la que va a enviarnos un enlace al poner el ratón encima.

Con Javascript podemos alterar el texto que aparece en la barra de estado del navegador en cualquier momento, así que será ideal para esconder el texto que aparece en la barra de estado.

**Nota:** La decisión de eliminar la URL que aparece en la barra de estado al ponerse encima de un enlace es una mala idea desde el punto de vista de la usabilidad. Todos utilizamos esa información para tener una referencia y eliminarla puede ser molesto para el visitante.

Tenemos dos maneras de esconder el texto de la barra de estado. La primera sería modificando la etiqueta del enlace, pero tendríamos que hacerlo enlace a enlace para todos los que se desee. También mostraremos un modo de hacer esto para todos los enlaces de la página a la vez.

### Esconder el texto de la barra de estado de enlace a enlace

Simplemente vamos a asignar un comportamiento cuando se pose el ratón encima de un enlace y otro comportamiento para cuando se salga del enlace.

```
<a href="http://www.guiarte.com" onmouseover="window.status='Guiarte, sitio de turismo y arte';return true" onmouseout="window.status='';return true"> Guiarte.com
```

Si vemos esta etiqueta del enlace se comprobará que tiene dos eventos definidos:

- onmouseover, para definir acciones cuando se pose el ratón sobre el enlace.  
En este evento indica con window.status un nuevo texto para la barra de estado. Luego hacemos el return true para que no se realice ninguna acción adicional por este evento.
- onmouseout, para definir acciones cuando se sale el ratón del enlace.  
En este evento borramos el texto de la barra de estado que aparecía al posarse sobre el enlace.

La ventaja de este modo es que podemos poner un texto distinto en la barra de estado para cada enlace de la página. Como decíamos, la desventaja es que tenemos que hacerlo en cada enlace que queramos evitar que se vea la URL.

Se puede [ver un ejemplo en una página aparte](#).

**Nota:** En la configuración predeterminada de Firefox no se permite alterar el texto de la barra de estado, por lo que este script no cambiará ese texto. Pero como tenemos el "return true" en el manejador del evento, al menos evitará que se vea la URL del enlace.

### Ocultar el texto de la barra de estado para todos los enlaces

Ahora veamos otro método de hacer esto, de una vez para todos los link que haya en la página. Simplemente vamos a hacer un código para borrar el texto de la barra de estado, que se va a ejecutar indefinidamente cada intervalo de tiempo. Así, aunque aparecerá la URL del enlace en la barra de estado durante unos instantes, nuestro código se ejecutará cada poco para borrarlo.

Veamos la siguiente sentencia Javascript:

```
setInterval ("window.status = '',10);
```

Esto es una llamada al método de window setInterval(), que sirve para ejecutar un código javascript indefinidamente en intervalos definidos. El primer parámetro es la instrucción que va a ejecutar window.status = "", que sirve para borrar el texto de la barra de estado. El siguiente parámetro son los milisegundos que tienen que transcurrir entre ejecuciones de la sentencia, en este caso 10 milisegundos.

Si ponemos esa instrucción en un script en cualquier parte de la página, preferiblemente en la cabecera, haremos que desaparezca lo escrito en la barra de estado en cuestión de instantes.

```
<script language="JavaScript">
setInterval ("window.status = '',10);
</script>
```

Se puede ver en funcionamiento el script [aquí](#).

**Nota:** En la configuración por defecto de Firefox no se permite cambiar el texto de la barra de estado, por lo que este script no parecerá tener ningún efecto.

Para encontrar más información sobre cómo cambiar esta configuración de Firefox consultando la FAQ: [¿Por qué no se cambia el texto de la barra de estado en Firefox con Javascript?](#)

## Conclusión

Aunque el texto de la barra de estado es útil, tal vez prefiramos que no aparezca, o que se muestre un mensaje personalizado. Espero que estas dos soluciones sean útiles para esos casos.

*Artículo por **Miguel Angel Alvarez***

## ¿Como integrar contenido RSS en mí pagina?

**Atención:** este artículo se apoya en un script que ya no se puede utilizar, porque los creadores lo han debido borrar del servidor. Pero hemos creado otro artículo para hacer lo mismo, colocar titulares extraídos de feeds RSS de otras webs en nuestra página, que hemos puesto el script para descarga en DesarrolloWeb.com, para asegurarnos que no vuelva a pasar lo mismo. Entra en: [Lector RSS con Javascript](#).

Uno de los beneficios que nos puede ofrecer el uso de RSS es el poder integrar el contenido de Feeds dentro de nuestras páginas lo cual nos aporta información actualizada para mantener nuestras paginas al día con lo que siempre mantendremos a nuestros visitantes con información renovada y sin necesidad de dedicarnos nosotros a mantener el sitio pues para la elaboración de este proceso hay varias opciones en el mercado las cuales las podemos desarrollar nosotros, comprarlas o incluso código reutilizable que se consigue por la red y GRATIS., en esta oportunidad les voy a comentar sobre este ultimo y es que conseguí en una página un lector de RSS feeds gratuito y es tan sencillo como copiar 2 líneas de un javascript en integrarlo en tu código y luego colocas la página de donde va a tomar la información y listo

ya tienes contenido actualizado en tu página, bastante bien, no?

Pues aquí le anexo las líneas que deben incluir en su sitio, el lector esta desarrollado en php.

### Coloquen estas líneas de código:

```
<script language="JavaScript" type="text/JavaScript" src="http://arupbhanja.com/rssfeed.php?file=http://construcanarias.bitacoras.com/rss1.xml&max=10"></script>
```

### Lo explico

Indica que viene código javascript...

```
<script language="JavaScript" type="text/JavaScript"
```

Es la dirección de donde saca el lector de feeds...

```
src="http://arupbhanja.com/rssfeed.php?
```

La dirección del feeds...

```
file=http://construcanarias.bitacoras.com/rss1.xml
```

Cantidad máxima de tópicos a mostrar, en este caso son 10 pero se puede modificar...

```
&max=10"></script>
```

Les recomiendo también visiten [Feeddigest \(en ingles\)](#).

*Artículo por César Pietri*

## ***Hacer que un iframe se ajuste a la altura de una ventana con Javascript***

Tengo una página que tiene un iframe y quiero que ocupe el espacio máximo disponible, pero no dispongo de toda la página, porque hay otros contenidos en la página. Además, como a veces la ventana del navegador es más grande o más pequeña, el espacio que puedo asignar al iframe es distinto.

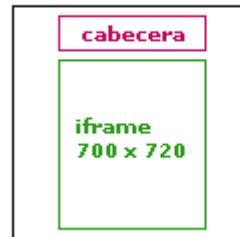
En este taller de Javascript vamos a realizar un cálculo del espacio disponible en la página para que un iframe que tenemos dentro ocupe la mayor área posible. Todo teniendo en cuenta que cada usuario puede entrar con una definición de pantalla distinta y con un navegador distinto.

**Nota:** Recordamos que un iframe es un frame que se puede insertar en el cuerpo de una página, asignando una altura y una anchura. Podemos ver una explicación detallada en <http://www.desarrolloweb.com/articulos/667.php>

Antes que nada me gustaría que se entendiese bien el problema con el que me encuentro, al no saber qué área hay disponible en la página para cada usuario que nos visita.

Veamos esta imagen, que nos puede aclarar rápidamente el caso en el que nos encontramos.

**Caso 1) Definición 800 x 600**

**Caso 2) Definición 1280 x 768**


Imaginemos una definición de 800 x 600. Entonces el espacio para el iframe será el tamaño útil donde se visualiza la página, menos el espacio reservado para la cabecera. Ahora, por ejemplo en una definición de 1280 x768, como el espacio útil para la página es mayor, el espacio en el que quiero que se vea mi iframe también será mayor. Sigue siendo el tamaño útil donde se visualiza la página, menos el espacio reservado para la cabecera, pero como ahora el espacio útil es mayor, el iframe también tiene que presentarse con mayor tamaño.

La solución pasa por utilizar un Javascript para calcular el espacio útil de la página y restarle el espacio de la cabecera. Entonces tendremos la dimensión altura que tiene que tener el iframe.

Para calcular este dato tenemos que tener en cuenta que Internet Explorer y Firefox tienen modos distintos. Es decir, la propiedad espacio útil de la página es distinta en estos dos browser, por lo que el script se puede complicar un poco.

En Internet Explorer: el espacio útil se calcula con la propiedad `document.body.clientHeight`.

En Mozilla Firefox: el espacio útil nos lo devuelve la propiedad `window.innerHeight`

Con este script podemos calcular el tamaño que debemos reservar al iframe:

```
if (window.innerHeight){
    //navegadores basados en mozilla
    espacio_iframe = window.innerHeight - 110
}else{
    if (document.body.clientHeight){
        //Navegadores basados en IExplorer, es que no tengo innerheight
        espacio_iframe = document.body.clientHeight - 110
    }else{
        //otros navegadores
        espacio_iframe = 478
    }
}
```

El primer if sirve para los navegadores Firefox, Netscape y similares, que entienden la propiedad `window.innerHeight`

El segundo if es para IExplorer que conoce `document.body.clientHeight`.

En los dos casos tenemos que restarle 110, que es el espacio que ocupa la cabecera. El último if es por si acaso no entiende ninguna de las dos propiedades el javascript, para darle un valor por defecto.

Luego, escribiríamos mediante javascript la etiqueta iframe con los datos obtenidos previamente:

```
document.write ('<iframe frameborder="0" src="mipagina.html" width="770" height="' + espacio_iframe + '">')
document.write ('</iframe>')
```

¿Y qué pasaría si los navegadores no entienden Javascript, o está deshabilitado?

En ese caso nos conviene utilizar la etiqueta noscript, para mostrar un iframe con los valores por defecto (noscript sólo se tiene en cuenta si no hay soporte para javascript):

```
<noscript>
<iframe frameborder="0" src="mipagina.html" width="770" height=478>
</iframe>
</noscript>
```

El código completo sería el siguiente:

```
<script>
if (window.innerHeight){
    //navegadores basados en mozilla
    espacio_iframe = window.innerHeight - 110
}else{
    if (document.body.clientHeight){
        //Navegadores basados en IEExplorer, es que no tengo innerheight
        espacio_iframe = document.body.clientHeight - 110
    }else{
        //otros navegadores
        espacio_iframe = 478
    }
}
document.write ('<iframe frameborder="0" src="mipagina.html" width="770" height="' + espacio_iframe + '">')
document.write ('</iframe>')
</script>
<noscript>
<iframe frameborder="0" src="mipagina.html" width="770" height=478>
</iframe>
</noscript>
```

*Artículo por **Miguel Angel Alvarez***

## ***¿Es Ventajoso el uso de ParseInt para validar números?***

La utilización del parseInt para validar números en muchos casos no resulta ser la solución más efectiva, debido a que permite la presencia de letras y/o espacios, y el resultado podría no ser el esperado.

### **¿Por qué parseInt puede causar problemas?**

Esta pregunta se responde a sí misma viendo varios ejemplos sobre el funcionamiento de parseInt:

- "123456": este String retorna como resultado el número 123456 el cual es el resultado esperado.
- "123456asd": este String retorna como resultado el número 123456 a pesar de que el String contenía letras (¿ventaja o desventaja?).
- "asd": este String retorna como resultado NaN el cual es el resultado esperado.

- "": este String vacío retorna como resultado NaN el cual es el resultado esperado.
- " 123456asd": este String (que contiene varios espacios al principio del número y letras al final) retorna como resultado el número 123456 (¿ventaja o desventaja?).
- " 123 123 asd" este String (que contiene espacios y letras) retorna como resultado el número 123 (¿ventaja o desventaja?).

Como se puede observar, parseInt presenta el siguiente comportamiento:

1. Retornará un número válido si: El String empieza por un número.
2. El String empieza por espacio(s) seguido de un número.
  - Ejemplos de números válidos: "123456"
  - " 123456"
  - "12345asdasd"
  - " 12345 asdd"
3. Todo String que cumpla con las 2 reglas anteriores (ser un número válido), será truncado: cuando se encuentre una letra, espacio o caracteres especiales (comas, acentos,...) dentro del String. Como resultado, retornará los dígitos que estén más a la izquierda de la primera letra (espacio o caracter) encontrada.
  - Ejemplos de números válidos truncados: "123456" retorna como resultado 123456
  - " 123456" retorna como resultado 123456
  - "12345asdasd" retorna como resultado 12345
  - " 123.. asdd" retorna como resultado 123

Una alternativa al parseInt, que valida que los String contengan solo números la tenemos a continuación:

```
function validarNumero(c_numero)
{
    //chequeo la longitud de c_numero:
    // Si (c_numero.length es igual a Cero) quiere decir que c_numero es una cadena Vacía.
    // Si (c_numero.length es distinto(mayor) de Cero) podemos asegurar que c_numero contiene por lo menos una
    letra
    //a la cual se le puede hacer la validación
    if (c_numero.length == 0)
    {
        return "NaN";
    }
    else
    {
        //Se recorre c_numero por todos sus caracteres chequeando que todos sean dígitos
        //la condición >="0" y <="9" es basada en el valor ascii que tienen los números en la tabla ascii.
        //Si alguno de los caracteres no es un número la función retornará un NaN
        //Si no retornará el Número
        for (i = 0; i < c_numero.length; i++)
        {
            if (!((c_numero.charAt(i) >= "0") && (c_numero.charAt(i) <= "9")))
                return "NaN";
        }
        return c_numero;
    }
}
```

## Ejemplos de validación de números

utilizando la función `parseInt`:

Resultado de aplicar la función:

utilizando la función `validarNumero` (llamando a la función `validar`):

Resultado de aplicar la función:

utilizando la función `validarNumero` (llamando a la función `validarComplejo`):

Resultado de aplicar la función:

*Artículo por **José Antonio Jiménez Garelli***

## **Efecto para inhabilitar/habilitar el fondo de la Página**

Bueno, no es una ventana emergente como tal, es más bien un simulacro pero que hace las veces de ventana emergente y sin peligro de que el navegador te bloquee dicha ventana. Es necesario tener conocimientos (por lo menos básicos) de:

- HTML - pueden ver la sección de HTML de DesarrolloWeb haciendo [clic Aquí](#).
- CSS - pueden ver la sección de CSS de DesarrolloWeb haciendo [clic Aquí](#).
- JavaScript - pueden ver la sección de JavaScript de DesarrolloWeb haciendo [clic Aquí](#).
- Para el ejemplo de este artículo se va a utilizar imágenes transparentes con distintos niveles de opacidad, por lo cual es necesario que sepas utilizar un editor de imágenes (photoshop, firework, ...) para crear las imágenes transparentes a tu gusto. En caso de que no tengas un editor de imágenes, puedes utilizar las del ejemplo sin ningún problema.

Puedes ver el ejemplo en funcionamiento [Aquí](#), así tendrás una idea más clara de lo que vamos a hacer. Todo el artículo se basa en la explicación del ejemplo. El ejemplo fue probado en Internet Explorer versión 6 y 7, y en el Mozilla FireFox 2, y todos con resultados positivos.

### **Explicación del Ejemplo:**

¡¡Por fin!! La parte buena de este artículo.

A muy grosso modo, el cuerpo principal (body) del archivo Html del ejemplo, está compuesto por 3 párrafos los cuales tienen la finalidad de llenar la página, logrando así que se vea cargada de información. Al final de cada párrafo hay un enlace el cual es el encargado de mostrar la ventana emergente a través de código hecho en JavaScript. Esta es la única diferencia significativa entre los 3 párrafos (el llamado a la función JavaScript). El código del primer párrafo es el siguiente:

<p>

Este es el contenido del primer párrafo, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla.<br>

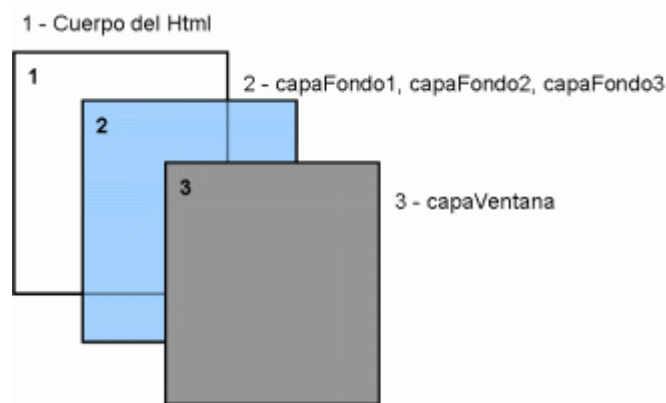
Este es el contenido del primer párrafo, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla.<br>Este es el contenido del primer párrafo, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla.<br>Este es el contenido del primer párrafo, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla, bla.<br><a href="javascript:abrirVentana('1');">Simulacro de ventana Emergente 1</a><br><p>

Una vez que hayas visto los 3 párrafos en el código del ejemplo, lo que verás a continuación son 4 capas (div) llamadas capaVentana, capaFondo1, capaFondo2 y capaFondo3. A estas capas le he aplicado código CSS y en principio todas están oculta. La capa capaVentana es la más compleja de todas y es porque en ella está el código de lo que he llamado "ventana emergente". El código es más sencillo de lo que parece:

Usé una tabla principal para maquetar una "ventanita de información" exacta a las del sistema operativo Windows XP, en donde a cada celda le asigné una imagen, y en la celda central, hice otra tabla donde coloqué el mensaje de la ventanita y el botón de Aceptar.

Las otras capas no tienen absolutamente nada. Más adelante veremos por qué.

Con esto finalizo lo que sería la explicación del cuerpo del Html. Antes de explicar los códigos que hice en JavaScript, es importante conocer la arquitectura de la página, es decir, la forma en que diagramé la página para lograr el efecto deseado:



Usando CSS se puede dar un nivel de profundidad a las capas. **iii**Esto es lo que he hecho!!! He aplicado el atributo z-index para colocar capas encima de otras. La capa que tenga un mayor valor numérico para el atributo z-index, es la que se verá más por encima de todas y las demás quedarán por debajo de acuerdo al atributo z-index. Para el ejemplo de este artículo, el cuerpo principal de la página tiene un z-index = 1, mientras que las capas llamadas capaFondo1, capaFondo2 y capaFondo3 les he asignado un valor de 2. A la capa capaVentana le he asignado 3 porque es la que quiero que quede por encima de todas las demás. Como dije anteriormente, todas las capas inicialmente están oculta (se utiliza el atributo visibility con el valor hidden) y la idea es aplicar JavaScript para mostrar dichas capas.

En las 3 capas centrales (capaFondo1, ...ondo3), es donde radica el truco de que el cuerpo principal de la ventana quede desactivado cuando se muestre la ventana emergente. Utilizando CSS hago que estas capas tengan el máximo ancho (width) que puedan tener y un largo (heigth) que he puesto a mi conveniencia, de modo que cubren todo el cuerpo principal. Las capas centrales se diferencian por las imágenes que en ellas se muestran. Todas tienen imágenes transparente con opacidades de 40%, 50% y 60% y diferentes filtros de transparencia. Al usar imágenes con tendencia (opacidad) al transparente sobre el fondo de la capa, la capa permite que se vea lo que hay por debajo de ella, en este caso, el cuerpo



principal de la página. Es importante destacar, que si colocan una imagen 100% transparente será equivalente a no colocar imagen, y el fondo se verá normal. La idea es colocar una imagen que no sea totalmente transparente, de modo que se vea el color de la imagen y lograr así que el fondo parezca inactivo.

Hay 2 funciones JavaScript, las cuales muestran y ocultan las capas. El código de la función que muestra las capas es el siguiente:

```
function abrirVentana(ventana)
{
    if (ventana=="1")
    {
        document.getElementById("capaFondo1").style.visibility="visible";
        document.getElementById("capaFondo2").style.visibility="hidden";
        document.getElementById("capaFondo3").style.visibility="hidden";
    }
    else if (ventana=="2")
    {
        document.getElementById("capaFondo1").style.visibility="hidden";
        document.getElementById("capaFondo2").style.visibility="visible";
        document.getElementById("capaFondo3").style.visibility="hidden";
    }
    else
    {
        document.getElementById("capaFondo1").style.visibility="hidden";
        document.getElementById("capaFondo2").style.visibility="hidden";
        document.getElementById("capaFondo3").style.visibility="visible";
    }
    document.getElementById("capaVentana").style.visibility="visible";
    document.formulario.bAceptar.focus();
}
```

Esta es la función que se ejecuta cada vez que se hace clic en cualquiera de los 3 enlaces. Al hacer clic en el enlace ubicado en el 1er párrafo, éste hace un llamado a la función y le pasa como parámetro el número uno (1), el cual indica el número del párrafo. Al hacer clic en los otros dos enlaces, se pasará como parámetro 2 y 3 de acuerdo al párrafo. Dentro de la función se obtiene los estilos de cada capa y se utiliza la propiedad visibility para mostrar u ocultar según sea el caso. Dentro de esta función se le da el foco al botón Aceptar.

Una vez que se muestra la ventanita emergente, esta se puede quitar (ocultar) presionando sobre el botón Aceptar o sobre la X. Esto hará un llamado a la función JavaScript respectiva:

```
function cerrarVentana()
{
    document.getElementById("capaFondo1").style.visibility="hidden";
    document.getElementById("capaFondo2").style.visibility="hidden";
    document.getElementById("capaFondo3").style.visibility="hidden";
    document.getElementById("capaVentana").style.visibility="hidden";
    document.formulario.bAceptar.blur();
}
```

Esta función se explica por sí misma. Oculta todas las capas y le quita el foco al botón Aceptar de la ventanita. Con esto finalizo la explicación del artículo. Espero le sirva y puedan aplicarlo en sus creaciones.

Puedes [ver el resultado final](#) de este script en una página aparte.

*Artículo por **José Antonio Jiménez Garelli***

## Validar número de checkbox marcados con Javascript

Aquí os dejo unas líneas de mi cosecha, en la creación de un script Javascript que he tenido que hacer para comprobar el estado de elementos checkbox o casillas de verificación de formularios.

Se trata de utilizar las típicas casillas de verificación pero con un limitador de grupo. Se puede utilizar en quinielas de varios resultados, en los futuros test de las autoescuelas con la posibilidad de marcar varias respuestas, etc.

Tenemos una serie de grupos de checkbox y lo que queremos hacer es asegurarnos que en cada grupo, de manera independiente, no se hayan marcado más de un número definido de casillas. Por ejemplo, tenemos x grupos de 3 casillas de verificación cada uno. Si el usuario marca una casilla de casillas de uno de los grupos no pasa nada. Si marca 2 casillas tampoco pasa nada, pero si intenta marcar los tres checkbox del grupo Javascript no lo permite y muestra un mensaje de error.

Podemos [ver el ejemplo en marcha](#) para hacernos una idea más concreta.

### Formulario HTML

Vamos a tener un formulario con, en este caso, dos grupos de casillas de verificación.

```
<form action="" method="post" enctype="multipart/form-data" name="formulario" id="formulario">
<table width="76">
<td width="20" valign="top"><input type="checkbox" onclick='validar(formulario.checkbox1,0)' name='checkbox1'
value='checkbox1'></td>
<td width="20" valign="top"><input type="checkbox" onclick='validar(formulario.checkbox2,0)' name='checkbox2'
value='checkbox2'></td>
<td width="20" valign="top"><input type="checkbox" onclick='validar(formulario.checkbox3,0)' name='checkbox3'
value='checkbox3'></td>
<tr>
<td width="20" valign="top"><input type="checkbox" onclick='validar(formulario.checkbox4,1)' name='checkbox4'
value='checkbox4'></td>
<td width="20" valign="top"><input type="checkbox" onclick='validar(formulario.checkbox5,1)' name='checkbox5'
value='checkbox5'></td>
<td width="20" valign="top"><input type="checkbox" onclick='validar(formulario.checkbox6,1)' name='checkbox6'
value='checkbox6'></td>
</tr>
</table>
</form>
```

Como podemos ver, el nombre de cada casilla es distinto. Y además tenemos una función que se ejecuta cuando se pulsa sobre el checkbox (evento onclick), que será la encargada de realizar la verificación.

### Función para verificar checkbox por grupos

Veamos el código javascript que utilizamos para realizar la comprobación de que varios checkbox no puedan estar pulsados a la vez en el mismo grupo.

Primero definimos un par de variables globales, que utilizaremos para definir las casillas máximas que pueden estar marcadas al mismo tiempo, y otra para llevar la cuenta de las

casillas que hay marcadas en cada grupo.

```
//Número máximo de casillas marcadas por cada fila  
var maxi=2;
```

```
//El contador es un array de forma que cada posición del array es una línea del formulario  
var contador=new Array(0,0);
```

Ahora la función que realizará la cuenta de casillas e informará de un posible fallo en la comprobación, si se pulsan más que las que se debe.

```
function validar(check,grupo) {  
    //Compruebo si la casilla está marcada  
    if (check.checked==true){  
        //está marcada, entonces aumento en uno el contador del grupo  
        contador[grupo]++;  
        //compruebo si el contador ha llegado al máximo permitido  
        if (contador[grupo]>maxi) {  
            //si ha llegado al máximo, muestro mensaje de error  
            alert('No se pueden elegir más de '+maxi+' casillas a la vez.');            //desmarco la casilla, porque no se puede permitir marcar  
            check.checked=false;  
            //resto una unidad al contador de grupo, porque he desmarcado una casilla  
            contador[grupo]--;  
        }  
    }else {  
        //si la casilla no estaba marcada, resto uno al contador de grupo  
        contador[grupo]--;  
    }  
}
```

La función recibe dos parámetros. Primero el campo de formulario checkbox que se ha pulsado. Luego el número de grupo al que pertenece ese checkbox.

El checkbox lo necesita para conocer su estado y para cambiarlo si fuera necesario. El grupo lo utiliza para saber a qué contador debe referirse, para saber el número de casillas que hay pulsadas en ese grupo.

La función está comentada para facilitar su lectura y comprensión.

El ejemplo en marcha se puede [ejecutar en una venta aparte](#).

*Artículo por **Javier Bernal Lérda***

## **Evitar que un textarea sobrepase un número de caracteres permitidos**

Este script de Javascript es bastante utilizado en muchos sitios web. Se trata de controlar el tamaño del texto que se escribe en un textarea para evitar que los caracteres escritos sobrepasen de los permitidos. El control de los caracteres escritos se hace con Javascript, dinámicamente en el lado del cliente, de modo que cuando el usuario llega a la longitud permitida, no se permite escribir más contenido en el campo textarea.

[Veamos el ejemplo en marcha](#) para hacernos una idea exacta del objetivo de este artículo.

El ejemplo es sencillo. Simplemente vamos a definir un número de caracteres permitidos. Con cada letra que escriba el usuario vamos a comprobar si la cantidad de caracteres que hay en el textarea es permitida.

- Si es permitida, no hacemos nada.
- Si no es permitida, porque estemos sobrepasando el número de caracteres que puede contener el textarea, no se deja escribir más texto en el campo del formulario. Eso lo conseguiremos colocando el texto que había antes de que se escribiese ese carácter no permitido.

Adicionalmente, vamos a llevar la cuenta de los caracteres escritos en un campo de texto, para que el usuario pueda visualizar los caracteres que lleva escritos. Además, cuando se llegue al límite de caracteres permitidos se pondrá en rojo el campo de texto que cuenta los caracteres del textarea.

Este ejercicio está realizado a partir de otro ejercicio que hemos publicado anteriormente en DesarrolloWeb.com, que sería bueno leer: [Contar caracteres escritos en un textarea](#)

El ejercicio tiene dos partes, el script Javascript y el formulario HTML. Empecemos viendo el formulario:

```
<form action="#" method="post">
<table>
<tr>
  <td>Texto: </td>
  <td><textarea cols="40" rows="5" name="texto" onKeyDown="valida_longitud()"
onKeyUp="valida_longitud()"></textarea></td>
</tr>
<tr>
  <td>Caracteres: </td>
  <td><input type="text" name="caracteres" size="4"></td>
</tr>
</table>
</form>
```

No tiene ninguna complicación. Pero hay que prestar atención a los eventos del textarea, que son onKeyDown y onKeyUp, que se desatan cuando el usuario aprieta o suelta teclas del teclado. En ambos eventos se llama a la función javascript valida\_longitud(), que se encargará de hacer todo el trabajo.

Veamos ahora el Javascript:

```
<script>
contenido_textarea = ""
num_caracteres_permitidos = 10

function valida_longitud(){
  num_caracteres = document.forms[0].texto.value.length

  if (num_caracteres > num_caracteres_permitidos){
    document.forms[0].texto.value = contenido_textarea
  }else{
    contenido_textarea = document.forms[0].texto.value
  }

  if (num_caracteres >= num_caracteres_permitidos){
    document.forms[0].caracteres.style.color="#ff0000";
  }
}
```

```
}else{
    document.forms[0].caracteres.style.color="#000000";
}

cuenta()
}
function cuenta(){
    document.forms[0].caracteres.value=document.forms[0].texto.value.length
}
</script>
```

Primero se definen dos variables:

```
contenido_textarea = ""
num_caracteres_permitidos = 10
```

La variable `contenido_textarea` almacena el contenido del campo `textarea`. Al principio está inicializada a la cadena vacía, porque el `textarea` suponemos que está vacío.

Tenemos también una variable `num_caracteres_permitidos`, que almacena el número de caracteres que se permite escribir en el `textarea`. En este caso lo hemos definido como 10.

Ahora nos metemos con la función `valida_longitud()`. Lo primero que hacemos es averiguar la cantidad de caracteres escritos, y lo almacenamos en la variable `num_caracteres`.

```
num_caracteres = document.forms[0].texto.value.length
```

Luego hacemos la parte más importante de este script: Vemos si los caracteres escritos son menores o iguales que los permitidos, para actuar en consecuencia.

```
if (num_caracteres <= num_caracteres_permitidos){
    contenido_textarea = document.forms[0].texto.value
}else{
    document.forms[0].texto.value = contenido_textarea
}
```

Si los caracteres escritos son menores o iguales que los caracteres permitidos, entonces todo va bien. Lo que hacemos es actualizar la variable que mantiene el contenido del `textarea`, `contenido_textarea`, introduciendo lo que hay en el `textarea` actualmente, que es de un tamaño permitido.

Si lo escrito en el `textarea` es mayor que lo permitido, se trata de una situación que no se puede aprobar. Entonces simplemente escribimos en el `textarea` lo que hay en la variable `contenido_textarea`, que era lo que había antes y que estaba validado en longitud correctamente.

Eso es todo, es sencillo! Pero ahora vamos a hacer una pequeña mejora para que cuando el `textarea` llegue a la longitud máxima permitida el campo de texto que lleva la cuenta de los caracteres se ponga de color rojo.

```
if (num_caracteres >= num_caracteres_permitidos){
    document.forms[0].caracteres.style.color="#ff0000";
}else{
    document.forms[0].caracteres.style.color="#000000";
}
```

```
}
```

Como se puede ver, simplemente se comprueba de nuevo si el número de caracteres es mayor o igual que los permitidos. Entonces, si es así, se actualiza la propiedad `style.color` del campo de texto "caracteres", que muestra el número de caracteres escritos. Con `style.color` se puede modificar la propiedad de estilo CSS que define el color del texto del campo. Si se había llegado a los caracteres permitidos, se pone color rojo, en caso contrario, se pone color negro.

Por último hacemos una llamada a la función `cuenta()`, que ya habíamos creado en el artículo anterior:

```
function cuenta(){  
    document.forms[0].caracteres.value=document.forms[0].texto.value.length  
}
```

Esta función simplemente actualiza el campo de texto, colocando el número de caracteres escritos en el textarea.

Podemos [ver de nuevo el ejemplo en marcha](#).

*Artículo por **Miguel Angel Alvarez***

## **Definir un archivo de estilos CSS para cada navegador con Javascript**

Queremos dar una solución actual para las personas que desean tener varias hojas de estilos para su página y que, dependiendo del navegador del usuario, puedan enlazar con una hoja u otra.

Esta utilidad es muy recurrida y preguntada, debido a que los navegadores a menudo interpretan de manera distinta algunos atributos de hojas de estilo. Con ello ocurre que una página con estilos CSS no se ve exactamente igual en un navegador que otra. Por ello una solución podría ser incluir una declaración de estilos específica para cada explorador de cada usuario que nos visite.

Ahora bien, no siempre es una buena idea proporcionar una hoja de estilos para cada navegador: Es laborioso crear las distintas hojas de estilo y luego es difícil de mantener el código. Si salen nuevos navegadores o no identificamos bien el que nos visita podemos mostrar una página con estilos optimizados para otras plataformas. Lo ideal es tener una hoja de estilos que funcione bien en todos los navegadores, o tener un par de hojas de estilo, una para navegadores habituales y otra para dispositivos móviles, por ejemplo. Siempre podemos construir la página con una declaración de estilos compatible y a menudo existen algunos trucos para que todo se vea más o menos igual en los navegadores más habituales.

Este tema ya lo tratamos en un artículo antiguo del taller de Javascript: [Estilos distintos para cada navegador](#), lo que ocurre es que en ese artículo hicimos una detección del navegador que hoy no sirve, después de la aparición de nuevos Browser y nuevas versiones.

Ahora la detección del navegador la vamos a dejar a cargo de un script más complejo, que nos informa bien sobre los distintos programas y versiones. El script de detección del navegador no es nuestro, lo hemos obtenido de una página web donde tratan el tema con detalle:

<http://www.quirksmode.org/js/detect.html>. Este script para detección del navegador crea un objeto con tres propiedades que podemos utilizar de la siguiente manera para obtener información sobre el cliente:

```
//muestro el navegador
document.write("Tu navegador: ");
document.write(BrowserDetect.browser);
document.write(" / ");
document.write(BrowserDetect.version);
document.write(" / ");
document.write(BrowserDetect.OS);
```

Nosotros hemos utilizado ese objeto resultado de la detección del navegador para crear una estructura de if, con la que poder aplicar unas cuantas hojas de estilo dependiendo de varios posibles navegadores.

```
if (BrowserDetect.browser == "Firefox") {
    document.write("<LINK REL='stylesheet' HREF='estilo_firefox.css' TYPE='text/css'>");
}
else {
    if (BrowserDetect.browser == "Explorer"){
        if (BrowserDetect.version >= 7){
            document.write("<LINK REL='stylesheet' HREF='estilo_ie7.css' TYPE='text/css'>");
        }else{
            document.write("<LINK REL='stylesheet' HREF='estilo_ie6.css' TYPE='text/css'>");
        }
    }else{
        if (BrowserDetect.browser == "Opera"){
            if (BrowserDetect.version < 9){
                document.write("<LINK REL='stylesheet' HREF='estilo_opera.css' TYPE='text/css'>");
            }else{
                document.write("<LINK REL='stylesheet' HREF='estilo_opera9.css' TYPE='text/css'>");
            }
        }else{
            document.write("<LINK REL='stylesheet' HREF='estilo_otros.css' TYPE='text/css'>");
        }
    }
}
```

Como hemos podido ver, hemos incluido una estructura que detecta un navegador Firefox, un Internet Explorer versión 7, Internet Explorer versión inferior a la 7, un Opera (también con distinción de versiones) y por último un caso para todos los demás navegadores.

El código completo de la página será el siguiente:

```
<html>
<head>
    <title>Link con estilos dinámico</title>
<script language="Javascript">
```

```
//detección del navegador por http://www.quirksmode.org/js/detect.html
```

```
var BrowserDetect = {
    init: function () {
        this.browser = this.searchString(this.dataBrowser) || "An unknown browser";
        this.version = this.searchVersion(navigator.userAgent)
            || this.searchVersion(navigator.appVersion)
            || "an unknown version";
        this.OS = this.searchString(this.dataOS) || "an unknown OS";
    },
    searchString: function (data) {
        for (var i=0;i<data.length;i++) {
            var dataString = data[i].string;
            var dataProp = data[i].prop;
            this.versionSearchString = data[i].versionSearch || data[i].identity;
```

```
        if (dataString) {
            if (dataString.indexOf(data[i].subString) != -1)
                return data[i].identity;
        }
        else if (dataProp)
            return data[i].identity;
    }
},
searchVersion: function (dataString) {
    var index = dataString.indexOf(this.versionSearchString);
    if (index == -1) return;
    return parseFloat(dataString.substring(index+this.versionSearchString.length+1));
},
dataBrowser: [
    { string: navigator.userAgent,
      subString: "OmniWeb",
      versionSearch: "OmniWeb/",
      identity: "OmniWeb"
    },
    {
      string: navigator.vendor,
      subString: "Apple",
      identity: "Safari"
    },
    {
      prop: window.opera,
      identity: "Opera"
    },
    {
      string: navigator.vendor,
      subString: "iCab",
      identity: "iCab"
    },
    {
      string: navigator.vendor,
      subString: "KDE",
      identity: "Konqueror"
    },
    {
      string: navigator.userAgent,
      subString: "Firefox",
      identity: "Firefox"
    },
    {
      string: navigator.vendor,
      subString: "Camino",
      identity: "Camino"
    },
    { // for newer Netscapes (6+)
      string: navigator.userAgent,
      subString: "Netscape",
      identity: "Netscape"
    },
    {
      string: navigator.userAgent,
      subString: "MSIE",
      identity: "Explorer",
      versionSearch: "MSIE"
    },
    {
      string: navigator.userAgent,
      subString: "Gecko",
      identity: "Mozilla",
      versionSearch: "rv"
    },
]
```



```
{ // for older Netscapes (4-)
  string: navigator.userAgent,
  subString: "Mozilla",
  identity: "Netscape",
  versionSearch: "Mozilla"
},
],
dataOS : [
{
  string: navigator.platform,
  subString: "Win",
  identity: "Windows"
},
{
  string: navigator.platform,
  subString: "Mac",
  identity: "Mac"
},
{
  string: navigator.platform,
  subString: "Linux",
  identity: "Linux"
}
]
};
BrowserDetect.init();

//script para poner estilos distintos para cada navegador
if (BrowserDetect.browser == "Firefox") {
  document.write("<LINK REL='stylesheet' HREF='estilo_firefox.css' TYPE='text/css'>");
}
else {
  if (BrowserDetect.browser == "Explorer"){
    if (BrowserDetect.version>=7){
      document.write("<LINK REL='stylesheet' HREF='estilo_ie7.css' TYPE='text/css'>");
    }else{
      document.write("<LINK REL='stylesheet' HREF='estilo_ie6.css' TYPE='text/css'>");
    }
  }else{
    if (BrowserDetect.browser == "Opera"){
      if (BrowserDetect.version<9){
        document.write("<LINK REL='stylesheet' HREF='estilo_opera.css' TYPE='text/css'>");
      }else{
        document.write("<LINK REL='stylesheet' HREF='estilo_opera9.css' TYPE='text/css'>");
      }
    }else{
      document.write("<LINK REL='stylesheet' HREF='estilo_otros.css' TYPE='text/css'>");
    }
  }
}
</script>
</head>

<body>

<h1>Detección navegador</h1>
<script>
//muestro el navegador
document.write("Tu navegador: ");
document.write(BrowserDetect.browser);
document.write(" / ");
document.write(BrowserDetect.version);
document.write(" / ");
document.write(BrowserDetect.OS);
```

```
</script>
</body>
</html>
```

Se puede [ver en funcionamiento en este enlace](#).

Artículo por **Miguel Angel Alvarez**

## Javascript no intrusivo

Una de las premisas mas importantes planteadas al diseñar mediante el uso de estándares es la separación de capas lógicas, es decir, por un lado tenemos el maquetado, que se representa mediante lenguaje (x)html, por otra parte esta el diseño visual, que normalmente se adjunta mediante hojas de estilo (css) hasta aquí todo esta muy claro.

Pero que ocurre con el comportamiento que se le quiera asignar a algunos objetos del documento, aquí es donde entra en juego el lenguaje JavaScript.

Imaginemos por ejemplo que tenemos un enlace al que le queremos dar una funcionalidad un poco diferente al resto, abrirlo en una ventana nueva por ejemplo, a la mayoría lo primero que nos viene en mente es hacer algo mas o menos similar a lo siguiente:

```
<a href="popup.html" onclick="window.open('popup.html', 'width=400,height=450,resizable=yes')">Abrir popup</a>
```

Lamentablemente esta linea acaba con toda nuestra teoría de separación de capas, por fortuna, existen maneras alternativas para asignar eventos usando JavaScript, en el caso concreto de este ejemplo una manera mas limpia de realizar lo mismo necesitaba algo mas de código para poder llevarse a cabo.

Primero tendremos que asignar una identidad única al enlace, y luego mediante el DOMDocument Object Model asignaremos el evento a dicha id, algo mas o menos tal que así.

```
<a href="popup.html" id="mypopup">Abrir popup</a>
```

```
<script type="text/javascript">
```

```
var x = getElementById('mypopup');
```

```
x.onclick = function() {
```

```
    window.open('popup.html', 'width=400,height=450,resizable=yes')
```

```
}
```

```
</script>
```

Sencillo, ¿no? bueno, quizás no resulte tan sencillo, pero gracias a mentes inquietas como la de [Ben Nolan](#), disponemos de algunas herramientas que si que hacen que resulte una tarea sencilla.

Me refiero a [behaviour](#), una librería JavaScript basada en la función document.getElementsBySelector escrita por [Simon Willison](#). Este fantástico "trozo de código"

nos permite olvidarnos de programar complejas funciones que asignen eventos según clase, id o selector.

Continuando con el ejemplo anterior, si incluimos esta librería podemos conseguir el mismo efecto asignando la función directamente al id seleccionado,

```
<script type="text/javascript">

var myrules = {

  '#mypopup' : function(element){

    element.onclick = function(){

      window.open('popup.html', 'width=400,height=450,resizable=yes')

    }

  }

};

Behaviour.register(myrules);

</script>
```

Personalmente creo que resulta una librería de inmensa utilidad, ahora solo falta ver como poder sacarle partido de manera óptima.

*Artículo por **Alex Sancho***

## **Control de introducción de caracteres de un campo de texto con Javascript**

Esto puede ser útil para campos que sólo admitir números o letras.

Por ejemplo, vamos a hacer que en un campo de texto de un formulario sólo se permitan meter números decimales del tipo 9999.99:

Necesitamos una función en JavaScript (por ejemplo):

```
function fieldNumber (objeto)
{
  var valorCampo;
  var evento_key = window.event.keyCode;
  var numPosPunto = 0;
  var strParteEntera = "";
  var strParteDecimal = "";
  var NUM_DECIMALES = 2;

  switch (evento_key)
  {
    case 48:
    case 49:
    case 50:
    case 51:
```

```
case 52:
case 53:
case 54:
case 55:
case 56:
case 57:
case 46:
break;
default:
window.event.keyCode = 0;
return false;
}

valorCampo = objeto.value;
if (evento_key == 46)
if (valorCampo.indexOf(".") != -1)
{
window.event.keyCode = 0;
return false;
}
/* Sólo puede teclear el número de decimales indicado en NUM_DECIMALES */
if ((numPosPunto = valorCampo.indexOf(".")) != -1)
{
strParteEntera = valorCampo.substr(0,(numPosPunto - 1));
strParteDecimal = valorCampo.substr((numPosPunto + 1), valorCampo.length)
if (strParteDecimal.length > (NUM_DECIMALES - 1))
{
window.event.keyCode = 0;
return false;
}
}
return true;
}
```

Tendremos una página con el formulario y la caja de texto. Tendremos que llamar a la función "fieldNumber" en el evento onkeypress:

```
<input type="text" name="txtImporte" onkeypress="fieldNumber(this)">
```

Si tenéis algún problema no dudéis en consultármelo mandándome un mail a [iszori@hotmail.com](mailto:iszori@hotmail.com)

*Artículo por **Ismael Zori***

## Listado de distintos Framework Javascript

Estoy haciendo una investigación sobre Frameworks Javascript y Ajax para elegir uno de ellos y utilizarlo en uno de nuestros proyectos. En principio he visto que en la web hay infinidad de opciones, algunas con muy buena pinta.

Parece que el mundo de los framework para Javascript se está popularizando mucho, a juzgar por las numerosas opciones. Nosotros hasta ahora para hacer Javascript Cross-browser (compatible con todos los navegadores) venimos utilizando unas librerías que explicamos en el [manual Cross Browser Javascript DHTML](#). Por otra parte, para trabajar con Ajax y PHP venimos utilizando las librerías Xajax, que también hemos relatado en el [manual Trabajo con Ajax en PHP utilizando Xajax](#). Pero claro, con un Framework de Javascript igual matamos dos pájaros

de un tiro y nos facilita mucho la creación de interfaces de usuario avanzadas en Javascript, necesarias para hacer proyectos de la web 2.0.

Para empezar estoy haciendo un listado de las distintas opciones que he encontrado. Luego investigaré a fondo los framework que he visto que están teniendo más aceptación por la comunidad de desarrolladores y los probaré. Entonces escribiré artículos más técnicos y didácticos.

Entonces, sin más tardar, aquí va el listado de Frameworks Javascript:

### **Mootools: "El framework javascript compacto"**

Este producto tiene buena pinta. Según parece es sencillo y bien planificado. Entre las virtudes que he visto más destacadas es que es ligero, pudiendo incluso definir qué partes del framework incluir y cuales no, para que se carguen los scripts más rápido en el cliente. A mi algunas personas me han hablado muy positivamente de este framework, así que quizás sea por el que empiece la investigación en detalle.

<http://mootools.net/>

### **jQuery: "Librería Javascript para escribir menos y hacer más"**

Parece ser que este es uno de los frameworks con más aceptación, por estar estupendamente documentado y por ser muy simple y permitir desarrollar con un código limpio y elegante. El peso de las librerías es razonable y además tiene muchos fans incondicionales, por lo que no me cabe duda que será un buen proyecto.

<http://jquery.com/>

### **Prototype: "El framework javascript cuyo propósito es facilitar el desarrollo de aplicaciones dinámicas"**

Este framework también resulta muy interesante, pues hay muchos usuarios que lo utilizan habitualmente y con éxito. Parece una opción altamente profesional y además tiene la garantía que lo utilizan para la creación de sus webs empresas muy conocidas a nivel mundial. A mi me ofrece muchas garantías, pero hay ciertos detractores que acusan a este framework de ser muy pesado y ralentizar los sitios web donde se utiliza.

<http://www.prototypejs.org/>

### **YUI: "The Yahoo! User Interface Library"**

Es un framework que utilizan los desarrolladores de Yahoo! para hacer su portal, que hace tiempo se ha distribuido para uso libre. Que provenga de Yahoo! para mi ya resulta una importante garantía y parece que tiene desarrollados una importante gama de controles y componentes. Tendría que probarlo personalmente para dar una opinión, pero parece que hay muchas personas que también lo acusan de ser un poco pesado.

<http://developer.yahoo.com/yui/>

### **Dojo: "Experiencias grandes... para cualquiera"**

Parece un producto también bastante atractivo y una opción seria. No obstante, he leído opiniones discordantes acerca de él. Algunos no dudan en calificarlo entre los mejores frameworks Javascript y otros acusan que es pesado y poco depurado, que arroja errores bastante fácilmente.

<http://www.dojotoolkit.org/>

### **Qooxdoo: "La nueva era del desarrollo web"**

Es un framework Javascript ajax multipropósito, opensource con dos tipos de licencia. He leído pocas opiniones sobre este software, pero parece digno de considerar.

<http://qooxdoo.org/>

### **GWT Google Web Toolkit: "construye aplicaciones Ajax en lenguaje Java"**

Es un conjunto framework opensource desarrollado en Java, con el que se han creado aplicaciones populares de Google, como Google Maps o Gmail. Sin duda, al tratarse de un producto de Google, no cabe duda que es una opción a considerar seriamente. Tiene un compilador que convierte las clases Java en código Javascript y HTML compatible con todos los navegadores.

<http://code.google.com/webtoolkit/>

### **Rico: "Javascript para aplicaciones de Internet de contenido enriquecido"**

Otra de las opciones más conocidas para desarrollar aplicaciones para la web 2.0. Es open source y ya se encuentra en la versión 2.0, con lo que se supone que el tiempo de vida le haya ayudado a ser más depurado. He leído por ahí que está poco documentado.

<http://openrico.org/rico/home.page>

### **Ext JS: "Documentación, diseño y código limpio"**

Este framework Javascript parece ser otra de las opciones serias. Se distribuye bajo licencia Open Source (gratis) y licencia comercial (de pago, pero con soporte y alguna funcionalidad adicional). Lo utilizan empresas bastante importantes, como Adobe. Me ha llamado la atención que tiene soporte para Adobe Air.

<http://extjs.com/>

Todavía quedan más opciones, pero voy a dejarlas listadas sin muchos comentarios, porque tampoco he investigado mucho y no las he visto en ningún sitio comentadas como opciones de primera línea.

- The Foo Framework (un framework basado en Prototype): <http://foo.riiv.net/>
- script.aculo.us (también basado en Prototype): <http://script.aculo.us/>
- AJS (Framework Javascript ultraligero): <http://orangoo.com/labs/AJS/>
- ZK (Ajax web framework): <http://www.zkoss.org/>

Esto es todo por el momento. Ahora queda empezar a trabajar para aprender a manejar los Frameworks y sacar conclusiones más serias. Espero que pronto podremos publicar más sobre el tema.

*Artículo por **Miguel Angel Alvarez***

## **Script para detección de soporte a Ajax, Cookies y ActiveX**

El sitio de Xajax Project ha publicado unos scripts interesantes para poder detectar si un navegador es compatible con la tecnología Ajax, para estar seguros que la web que estamos desarrollando se va a poder mostrar correctamente en cualquier cliente web que tenga el usuario. Además estos scripts sirven para mostrar mensajes de error si el navegador no tiene soporte a Ajax, de modo que el usuario sea consciente que no va a poder ver esa web convenientemente.

Estos scripts detectan las capacidades del navegador y se pueden ejecutar para mostrar mensajes de alerta si no están disponibles ciertas funcionalidades, ya sea porque el navegador del usuario no las soporta o porque estén deshabilitadas.

El script contiene tres funciones:

`browserSupportsCookies()`

Detecta si el navegador soporta cookies y devuelve true en caso que estén soportadas y false si no es así.

`browserSupportsAjax()`

Comprueba si el navegador tiene compatibilidad con la tecnología Ajax, devuelve true si es así y false si no soporta Ajax por cualquier cuestión.

`ActiveXEnabledOrUnnecessary()`

Esta función detecta si el navegador soporta ActiveX o bien si ActiveX es innecesario para la ejecución de Ajax. En el navegador Internet Explorer 6 Ajax se ejecuta a través de ActiveX, así que necesita disponer ActiveX para que todo funcione. Así que esta función devolverá false sólo si el navegador es Internet Explorer 6 y tiene inhabilitado ActiveX.

Las funciones no las voy a escribir en el texto de este artículo, simplemente voy a poner un link al lugar donde se muestran las funciones en la página de Xajax Project:

[http://xajaxproject.org/wiki/Xajax\\_%28any%29:\\_Tips\\_and\\_Tricks:\\_Detecting\\_Support](http://xajaxproject.org/wiki/Xajax_%28any%29:_Tips_and_Tricks:_Detecting_Support)

Pero también voy a dejar un enlace a una página en DesarrolloWeb.com donde hemos implementado estos scripts, para que los podáis ver en funcionamiento en vuestros navegadores. Así mismo, podéis ver el código fuente de la página para ver la implementación de los scripts que hemos hecho en DesarrolloWeb.com y obtener el código de las funciones en caso que cambien la URL en la página de Xajax.

<http://www.desarrolloweb.com/articulos/ejemplos/comprobar-compatibilidad-ajax.html>

*Artículo por **Miguel Angel Alvarez***

## **Lector RSS con Javascript**

He estado investigando en diferentes sitios la manera de crear sistema en Javascript que lea RSS de otras webs, para publicar los titulares en una página. Finalmente encontré un script lector de RSS que voy a comentar en este artículo.

El sistema permite leer una hoja XML que contiene un feed RSS y escribe las entradas del RSS en la página. En el contenido de la página no figura el RSS, sino que está en un archivo externo y con el script se escribe el texto de las distintas entradas, con sus enlaces y otras informaciones.

El script lo tienen publicado en la página <http://www.cstruter.com/downloads.php>

Yo lo he descargado y he colocado en el servidor de DesarrolloWeb.com, por si acaso lo quitan de la web donde lo he obtenido (como ya nos ha pasado con otros scripts que hemos comentado en este sitio). Se puede [descargar con este enlace](#). No obstante, recomiendo entrar en la página donde lo he obtenido, por si acaso publican versiones nuevas.

## Condiciones para el uso del lector RSS con Javascript

Antes de continuar explicando el funcionamiento hay que decir que existe una restricción de uso de este script, que resulta importante porque en Firefox no funcionará. Se trata de que Firefox, como medida de seguridad, no permite leer el contenido de otras webs. Como el RSS con los titulares lo sacamos de otras webs, pues en Firefox vamos a tener problemas, porque no va a permitir su lectura y la presentación de los titulares en la página. Este problema no lo tiene Internet Explorer, pero aun así tendremos que buscar otras soluciones.

La solución más sencilla sería la de publicar el RSS en nuestro servidor. Es decir, descargarlo de la web deseada y subirlo por FTP a nuestro sitio. Claro que esto nos obligaría a realizar una operación manual cada vez que queremos que los titulares se actualicen y ello puede significar que perdamos una de las ventajas de presentar titulares RSS de otras webs, que es disponer siempre de contenido actualizado. Además se de resultar un poco pesada la tarea de descargar el feed RSS y subir todos los días el archivo XML a nuestra web.

La solución más óptima sería crear un script en programación del lado del servidor, con por ejemplo PHP, ASP o .NET que realice la tarea de descargar el RSS con los titulares y lo copie en nuestro servidor. Este script se podría ejecutar cada cierto tiempo o cada vez que un usuario acceda a la página donde se lee el RSS remoto para presentar los titulares. La desventaja de esta opción es que necesitamos que nuestro servidor soporte programación de scripts en PHP, ASP o similares. Aparte que si hacemos programación del lado del servidor para extraer el feed RSS podríamos directamente tratarlo para presentar los datos en la página, sin necesidad de este script Javascript. No obstante, cabe señalar que en DesarrolloWeb.com, en las secciones monotemáticas de PHP o ASP tenemos materiales para aprender a leer un archivo remoto, que esté en otro servidor.

## Uso del lector RSS Javascript

El script es extremadamente sencillo de utilizar, ya que toda la parte complicada la hace por sí mismo. Simplemente tenemos que especificar en una línea de código el archivo RSS del que tiene que extraer los titulares.

Del archivo de descarga, simplemente tenemos que cambiar la siguiente línea:

```
ReadRSS('cnn_tech_rss.xml','rssBodyTemplate','rssTitleTemplate');
```

La función ReadRSS(), que es el lector RSS, en el primer parámetro tiene el nombre del archivo RSS que debe leer. Nosotros lo podemos cambiar por el nombre del archivo que pretendemos mostrar sus titulares.

Este archivo lo podríamos cambiar por la URL completa del feed RSS del servidor donde se lo tienen publicado. Por ejemplo, para leer el RSS con las novedades de FAQ que publicamos en DesarrolloWeb.com se llamaría la función así:

```
ReadRSS('http://www.desarrolloweb.com/rss/faq_rss.php','rssBodyTemplate','rssTitleTemplate');
```

En Internet Explorer no hay ningún problema con este uso de la función, simplemente veremos que los titulares tardan un poco más en generarse, debido a que tiene que conectar con la página remota para descargar el RSS. Pero podremos comprobar que la función, cuando se ejecuta en Firefox, muestra un mensaje advirtiendo del problema y sugiriendo que se copie el archivo remoto en nuestro propio servidor para poder funcionar.



[Descargaros el script](#) al que hacemos referencia y realizar vuestras propias pruebas.

Artículo por **Miguel Angel Alvarez**

## Funciones para validación alfanumérica de strings en Javascript

He creado una serie de funciones de string para realizar unas comprobaciones sobre cadenas de caracteres, que voy a utilizar más adelante en un script más complejo. Comienzo explicando estas funciones sueltas, que quizás sirvan de utilidad a los lectores. También con la intención de presentar poco a poco la complejidad de mi objetivo final. Las funciones de validación de strings sirven para saber si las cadenas tienen o no números, letras, letras mayúsculas y minúsculas. Son una serie de funciones bien sencillas y parecidas entre si, que hacen uso de los [métodos de la clase string de Javascript](#).

Estas funciones simplemente hacen el recorrido por el string en búsqueda de caracteres de un determinado tipo. Hacen el recorrido completo por todo el string hasta que encuentran un carácter del tipo buscado, de modo que se pueda saber seguro si hay o no caracteres de ese tipo. Veremos a continuación las distintas funciones:

### Saber si el string contiene caracteres numéricos

Esta función recibe un string y devuelve 1 si se encuentran caracteres numéricos y 0 si no se encuentran.

```
var numeros="0123456789";

function tiene_numeros(texto){
  for(i=0; i<texto.length; i++){
    if (numeros.indexOf(texto.charAt(i),0)!=-1){
      return 1;
    }
  }
  return 0;
}
```

Hemos creado primero una variable global con los números posibles, del 0 al 9, que queremos buscar. Esa variable la utilizaremos dentro de la función.

La función hace un recorrido a todos los caracteres del string. Para cada uno se comprueba si está o no dentro de la variable numeros, creada con anterioridad.

Si estaba, se devuelve 1 (como se ejecuta el return, se sale de la función devolviendo el valor 1).

Si no estaba, entonces se hará el recorrido del string, carácter a carácter, hasta el final. Entonces se termina el bucle y se devuelve 0.

Esta función se puede probar con estas sentencias:

```
alert(tiene_numeros("ASAS1"));
alert(tiene_numeros("2asasasas"));
alert(tiene_numeros("asas2sG"));
alert(tiene_numeros("a..."));
alert(tiene_numeros("A22323G2.12"));
```

Podemos [ver el ejemplo en marcha](#) en una página aparte.

## Saber si un string contiene letras

Veamos una función muy similar, para saber si un string contiene un carácter que sea letra, valiendo tanto las mayúsculas como minúsculas.

```
var letras="abcdefghijklmnopqrstuvwxyz";

function tiene_letras(texto){
    texto = texto.toLowerCase();
    for(i=0; i<texto.length; i++){
        if (letras.indexOf(texto.charAt(i),0)!=-1){
            return 1;
        }
    }
    return 0;
}
```

El algoritmo es prácticamente el mismo que la función anterior. Primero hemos creado un string con todas las letras del alfabeto. Luego hacemos un recorrido buscando en cada uno de los caracteres del string recibido por parámetro una de las letras del alfabeto.

La única particularidad es la llamada al método `toLowerCase()`, para convertir el texto que se recibe por parámetro a minúsculas y así que la búsqueda de letras no tenga en cuenta si son minúsculas o mayúsculas (ya que sólo buscamos si el string contiene letras).

Podemos probar el string con estas líneas de código:

```
alert(tiene_letras("1"));
alert(tiene_letras("2232323s"));
alert(tiene_letras("2232323sf"));
alert(tiene_letras("a2232323"));
alert(tiene_letras("A22323G2.12"));
```

Podemos [ver el ejemplo en marcha aquí](#).

## Saber si un string tiene letras minúsculas

Aquí vemos la función para comprobar si un string tiene caracteres en minúsculas. Es muy parecida a las que hemos visto anteriormente.

```
var letras="abcdefghijklmnopqrstuvwxyz";

function tiene_minusculas(texto){
    for(i=0; i<texto.length; i++){
        if (letras.indexOf(texto.charAt(i),0)!=-1){
            return 1;
        }
    }
    return 0;
}
```

Con su batería de pruebas:

```
alert(tiene_minusculas("1"));
alert(tiene_minusculas("22323232s"));
alert(tiene_minusculas("22323232sf"));
alert(tiene_minusculas("a2232323"));
alert(tiene_minusculas("A22323G2.12"));
```

Lo podemos [ejecutar aquí](#).

### Comprobar si un string tiene letras mayúsculas

Ahora veremos la función para ver si un string tiene letras mayúsculas. Es igual que la anterior, simplemente que tenemos una cadena con todas las letras en mayúscula para buscar en el texto recibido por parámetro.

```
var letras_mayusculas="ABCDEFGHJKLMNÑOPQRSTUVWXYZ";

function tiene_mayusculas(texto){
  for(i=0; i<texto.length; i++){
    if (letras_mayusculas.indexOf(texto.charAt(i),0)!=-1){
      return 1;
    }
  }
  return 0;
}
```

Estas podrían ser las pruebas para comprobar el funcionamiento.

```
alert(tiene_mayusculas("1"));
alert(tiene_mayusculas("22323232s"));
alert(tiene_mayusculas("22323232sG"));
alert(tiene_mayusculas("a2232323"));
alert(tiene_mayusculas("A22323G2.12"));
```

Puedes ver en funcionamiento el script en el [siguiente enlace](#).

Estas funciones las utilizaremos en el siguiente artículo, en el que voy a explicar la creación de un sistema para mostrar la seguridad de una clave, para decir qué tan segura es en función del número de caracteres, si tiene números y letras, si tiene mayúsculas y minúsculas, etc.

*Artículo por **Miguel Angel Alvarez***

## Script para informar de la seguridad de una clave, con Javascript

Vamos a ver un sencillo script en Javascript para comprobar el grado de seguridad de una clave escrita por el usuario. Como es un script javascript del lado del cliente, permitirá mostrar el nivel de seguridad de la clave al mismo tiempo que el usuario la escribe en un campo de formulario.

Este script lo podremos utilizar libremente en nuestras páginas, de modo que ofrezcamos a los

visitantes una información sobre lo segura o insegura que es la clave que están eligiendo, lo que les motivará a escribir claves más seguras que las que habitualmente se escriben.

Podemos [ver un ejemplo](#) del objetivo buscado antes de continuar.

En un artículo anterior del taller de Javascript estuvimos mostrando la manera de [hacer varias funciones para comprobar un string](#) y saber si tiene letras, números, mayúsculas y minúsculas. Estas funciones las utilizaremos ahora para este script de información de seguridad de la contraseña.

Para valorar el grado de seguridad de una contraseña vamos a tener en cuenta estas puntuaciones sobre distintos conceptos:

Tiene letras y números: +30%  
Tiene mayúsculas y minúsculas: +30%  
Tiene entre 4 y 5 caracteres: +10%  
Tiene entre 6 y 8 caracteres: +30%  
Tiene más de 8 caracteres: +40%

Podríamos haber escogido cualquier otra puntuación para la seguridad de la contraseña, pero esta valdrá. También podríamos haber creado otros criterios para decidir el grado de seguridad. En cualquier caso, para que quede claro este baremo, pongo un par de ejemplos:

A) Una clave con números y letras, con 7 caracteres tendría: 30% por letras y números + 30% por tener entre 6 y 8 caracteres = 60% de seguridad.  
B) Otra clave con letras mayúsculas y minúsculas, sin números, y con 8 caracteres: 30% por mayúsculas y minúsculas + 40% por más de 8 caracteres = 70% de seguridad.

Para controlar la seguridad, apoyándonos en las [funciones de validación alfanumérica de strings](#) vistas anteriormente, haremos una función como esta:

```
function seguridad_clave(clave){  
    var seguridad = 0;  
    if (clave.length!=0){  
        if (tiene_numeros(clave) && tiene_letras(clave)){  
            seguridad += 30;  
        }  
        if (tiene_minusculas(clave) && tiene_mayusculas(clave)){  
            seguridad += 30;  
        }  
        if (clave.length >= 4 && clave.length <= 5){  
            seguridad += 10;  
        }else{  
            if (clave.length >= 6 && clave.length <= 8){  
                seguridad += 30;  
            }else{  
                if (clave.length > 8){  
                    seguridad += 40;  
                }  
            }  
        }  
    }  
    return seguridad  
}
```

Vamos comprobando si el string tiene diversas cosas, como letras, números, mayúsculas, minúsculas, así como su longitud, para ir asignando un mayor valor a la seguridad.

## Ejemplo de uso en un formulario que pide una clave

Ahora, veamos un sencillo ejemplo de uso de la función en un formulario, que muestra la seguridad de una clave escrita por el usuario:

El formulario podría ser como este:

```
<form>
Clave: <input type="password" size=15 name="clave" onkeyup="muestra_seguridad_clave(this.value, this.form)">
<i>seguridad:</i> <input name="seguridad" type="text" style="border: 0px; background-color:ffffff; text-
decoration:italic;" onfocus="blur()">
</form>
```

Como vemos, tenemos 1 campo INPUT de tipo PASSWORD donde escribiremos la clave. A este campo se le ha introducido un evento ONKEYUP que se ejecuta cuando el usuario pulsa una tecla, pero en el momento de soltarla. Esa función será la encargada de hacer que se visualice la seguridad de la clave.

Pero además, hemos colocado otro campo de texto, para colocar el valor de seguridad de la contraseña. Este campo lo hemos forzado a que no se pueda escribir en él con el evento onfocus="blur()">. Sólo se podrá modificar mediante Javascript.

Veamos la función muestra\_seguridad\_clave(), que es la que se encarga de recibir tanto la clave escrita como el formulario donde se encuentra, para actualizar el valor de seguridad.

```
function muestra_seguridad_clave(clave,formulario){
    seguridad=seguridad_clave(clave);
    formulario.seguridad.value=seguridad + "%";
}
```

Como vemos, se hace uso de la función que devuelve la seguridad de un string que se va a utilizar como contraseña. Luego se mete ese valor en el campo de texto adicional que hay en el formulario.

Podemos [ver el ejemplo](#) en una página aparte.

*Artículo por **Miguel Angel Alvarez***

## Editor de texto WYSIWYG Javascript: TinyMCE

TinyMCE es un editor HTML que es capaz de convertir los textareas de un formulario en campos WYSIWYG para poder incluir etiquetas HTML dentro de los campos de texto.

### Características

- Es fácil de integrar en las páginas web, ya que solo tiene dos líneas de código.
- Se puede personalizar a través de temas y plugins.
- También se pueden instalar paquetes de idiomas.
- Es compatible con la mayoría de los navegadores como Firefox, Internet Explorer, Opera y Safari, aunque este último está en fase experimental.

- Con el compresor GZip para PHP/.NET/JSP/Coldfusion, hace que TinyMCE sea un 75% más pequeño y mucho más rápido de cargar.
- Se puede utilizar AJAX para guardar y cargar el contenido.

## Integración de TinyMCE

Para poder utilizar TinyMCE en las páginas web, el navegador tiene que ser compatible y tener Javascript habilitado.

Primero hay que descargar TinyMCE desde la siguiente página de descargas: <http://tinymce.moxiecode.com/download.php>. Después hay que descomprimirlo y guardarlo en el servidor de la página web para poder utilizarlo en los textareas de los formularios.

En la página que se vaya a utilizar, primero hay que incluir la librería tiny\_mce.js incluyendo el archivo externo de código Javascript.

```
<script language="javascript" type="text/javascript" src="/tinymce/jscripts/tiny_mce/tiny_mce.js"></script>
```

A continuación hay que inicializar TinyMCE para convertir los textareas en campos de texto WYSIWYG editables.

```
<script language="javascript" type="text/javascript">
tinyMCE.init({
    mode : "textareas",
    theme : "simple"
});
</script>
```

## Ejemplo de integración de TinyMCE

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
<head>
    <title>Ejemplo TinyMCE</title>
    <script language="javascript" type="text/javascript" src="/tinymce/jscripts/tiny_mce/tiny_mce.js"> </script>
    <script language="javascript" type="text/javascript">
        tinyMCE.init({
            mode : "textareas",
            theme : "advanced"
        });
    </script>
</head>

<body>
    <form method="post" name="tinymce">
        <textarea name="texto" cols="50" rows="15"></textarea>
    </form>
</body>
</html>
```

En este ejemplo primero hemos incluido la librería tiny\_mce.js dentro de las etiquetas <head>. También dentro de estas etiquetas también hemos inicializado TinyMCE para que en el textarea del formulario se convierta en un campo de texto WYSIWYG.

Puedes ver el ejemplo en funcionamiento en el [siguiente enlace](#).

*Artículo por **Gema Maria Molina Prados***