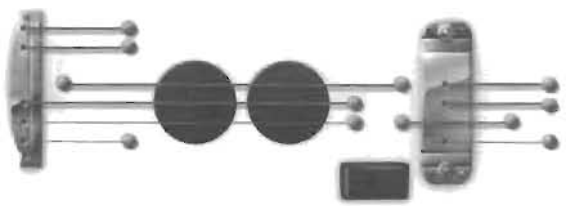
4 Integración de contenido interactivo

OBJETIVOS DEL CAPÍTULO

- ✓ Identificar las necesidades actuales respecto al contenido interactivo.
- ✓ Añadir animaciones a una página web.
- ✓ Añadir interactividad a los diseños web.
- ✓ Desarrollar y agregar animaciones para distintos navegadores.
- ✓ Verificar el funcionamiento de animaciones en diferentes navegadores.

4.1 ELEMENTOS INTERACTIVOS BÁSICOS Y AVANZADOS

La interacción es, junto a la inclusión de elementos multimedia, otra de las claves del desarrollo web. La interacción, según Wikipedia, *es el proceso que establece un usuario con un dispositivo, sistema u objeto determinado. Es una acción recíproca entre el elemento con el que se interacciona y el usuario.* Para entender bien esta idea lo mejor es interactuar con la guitarra que desarrolló Google Logos en honor al nacimiento de Les Paul[65] y que la Figura 4.1 muestra con una captura estática.



Back to Google Logos

Figura 4.1. *Guitarra interactiva Google Logos (les Paul)*

Esta Logo interactivo fue desarrollado con HTML5, CSS3 y JavaScript, aunque tenga la apariencia de un archivo *.swf* hecho con Adoble Flash.

Al igual que ocurría en el capítulo anterior con los elementos multimedia, la interacción en un futuro inmediato pasa por los lenguajes antes citados: HTML5, CSS3 y JavaScript. Los archivos Flash seguirán teniendo su protagonismo, pero cada vez en menor medida[66].

En ese capítulo nos centraremos en la interacción de objetos con jQuery, usando inevitablemente HTML5 y CSS3 (en la sección 3.6.2 ya se han visto algunos ejemplos de animaciones con este lenguaje). jQuery es una biblioteca de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. jQuery es un software libre y de código abierto (licencia MIT y GNU v2) y su misión es ahorrar tiempo y espacio en desarrollo de animaciones comparado con el uso de JavaScript directamente.

jQuery es un lenguaje muy extenso, pero este capítulo se centrará únicamente en lo más relacionado con las interfaces de usuario y la interacción, aunque es inevitable empezar con los conceptos básicos del lenguaje. Para un correcto aprendizaje de lo mostrado en el capítulo, el lector debe estar familiarizado con la tecnología actual de desarrollo web, por ejemplo, XML y DOM.

---

[65] Se puede interactuar con esta guitarra en *http://www.google.com/logos/2011/lespaul.html*

[66] En la sección 3.7 se habla más profundamente sobre Flash vs HTML5 y el futuro de la web.

4.1.1 INTRODUCCIÓN A JQUERY

Para utilizar jQuery, solamente es necesario descargar la librería[67] y enlazarla en la página HTML de la siguiente manera:

```
<script type="text/javascript" src="jquery.js"></script>
```

La librería se puede descargar en dos versiones: comprimida y descomprimida. La primera es más adecuada para que la descargue el cliente ya que ocupa menos espacio. La segunda es más adecuada cuando se está trabajando en el desarrollo.

Una de las primeras instrucciones que todo código jQuery debe tener es:

```
$(document).ready( function() { ... } );
```

Esta instrucción indica que el código jQuery se ejecute cuando el código HTML se ha cargado completamente y el árbol DOM se ha generado[68]. Esto debe ser así ya que jQuery trabaja con los elementos que hay en la página, referenciados a través del DOM. Si los objetos no se han cargado no se debe poder ejecutar ningún código jQuery ya que daría un error.

La función más usada en jQuery es *$()*[69]. Esta función se utiliza para seleccionar un elemento del árbol DOM a través del identificador, a través del nombre de la clase o de la etiqueta (CSS). Por ejemplo:

Seleccionar un elemento con id=#miIdentificador:

```
$("#miIdentificador");
```

Seleccionar todos los elementos <a> (enlaces) de un HTML:

```
$("a");
```

Se pueden seleccionar varios al mismo tiempo, separados por coma:

```
$("#miParrafo, a");
```

Seleccionar mediante el nombre de la clase CSS:

```
$(".miClase");
```

En el ejemplo mostrado en la sección 3.6.2, se selecciona los elementos 'ul.hover_block li' como selectores CSS, mediante el siguiente código:

```
$('ul.hover_block li').hover(.....
```

---

67 Por ejemplo de *http://blog.jquery.com/2011/09/12/jquery-1-6-4-released/*

68 Este comando no espera a que se carguen las imágenes, con los elementos definidos en el HTML es suficiente.

69 Esta función es equivalente al *getElementsByTagName()* y *getElementsByName()* y *getElementById* de JavaScript.

También se pueden seleccionar los elementos que tenga un cierto atributo. Por ejemplo, todos los elementos <a> con atributo *rel*.

```
$("a[rel]");
```

Esta consulta se puede completar pidiendo aquellos con un cierto valor para un atributo. Por ejemplo todos los elementos <input> con atributo *type* = *radio* (es decir, que son radio-buttons)

```
$("input[@type=radio]");
```

O incluso solo aquellos radio-buttons que estén seleccionados:

```
$("input[@type=radio][@checked]");
```

Además de poder seleccionar elementos por el nombre de las etiquetas o los selectores CSS, también se pueden seleccionar usando el lenguaje XPath. Así, por ejemplo. se pueden seleccionar todos los párrafos del documento con un camino XPath:

```
$("/html/body//p");
```

O también elementos que tienen un determinado atributo con un valor. Por ejemplo, los elementos <a> con el atributo *ref*='nofollow'.

```
$("//a[@ref='nofollow']");
```

Estos son solo algunos ejemplos de la flexibilidad que da jQuery para seleccionar los elementos sobre los que luego se aplicarán ciertas acciones. Existen más alternativas, incluso jQuery proporciona sus propias etiquetas para simplificar aún más la selección, sin embargo con los anteriores se abarca un amplio abanico de consultas.

4.1.2 MANEJO DE EVENTOS

Además del evento *ready()* usando en el primer ejemplo, jQuery dispone de varias funciones relacionadas con la gestión de los eventos. De hecho, jQuery tiene un modelo de eventos muy completo que facilita la programación de interacciones entre el usuario y los objetos de la página web.

Para cada evento disponible hay dos posibilidades de manejo:

1. Se le puede pasar una función que determine su comportamiento.
2. No se le pasa función, entonces se ejecuta el evento JavaScript del elemento. jQuery tiene tantas funciones como eventos estándar de JavaScript. El nombre de cada función es el mismo que el del evento, pero sin el habitual prefijo "on" de los eventos:

Un ejemplo del primer caso es el siguiente:

```
$("p").click(function() { alert( $(this).text() );});
```

Este código muestra una ventana de alerta cuando se hace clic (*onclick*) en cualquier párrafo del documento. Esta función ha sido definida expresamente para atender este evento.

Un ejemplo del segundo caso es: el siguiente:

```
$("p").click();
```

En este caso, se ejecuta la función *click()* que es la asociada por defecto al evento *onclick*.

bind() y unbind()

Un método interesante para trabajar con eventos de manera óptima es *bind()*. Este método permite asociar a un elemento un número ilimitado de eventos, todo de una vez. Esto método facilita la asignación de eventos a los elementos, haciendo un código más legible y una ejecución más óptima.

La sintaxis es:

bind("tipo_de_evento1 tipo evento2 ... tipo evento n",manejador (handler))

Un ejemplo de uso es el siguiente:

```
<script type="text/javascript" src="jquery-1.6.4.js"></script>
<script type="text/javascript">
$(document).ready(function() {
    $("p").bind("click mouseenter mouseleave", function(e){
          if ($(this).css("color")!="rgb(255, 0, 0)")
                $(this).css("color", "rgb(255, 0, 0)");
          else
                $(this).css("color", "rgb(0, 0, 255)");
    })
});
</script>
```

Este código averigua para todos los elementos *p* de qué color son cuando el usuario hace clic sobre él (evento clic) o entra o sale con el ratón (*mouseenter* y *mouseleave*). Si no es de color *rojo* entonces le asigna ese color y si lo es entonces lo cambia a *azul*.

Además de *bind()*, existe el método contrario *unbind()*, que elimina un evento previamente asignado a un elemento o varios de una página. Si se utiliza *unbind()* sin ningún parámetro se eliminan todos los manejadores de cualquier tipo de evento. Esto es útil cuando se quiere quitar la interacción a unos elementos de una página por cualquier motivo.

```
$("p").unbind();
```

Sin embargo, lo más habitual es quitar la atención de un tipo de evento en particular. Por ejemplo, para eliminar el manejo de un evento *onclic* sobre un elemento *p*, se escribiría:

```
$("p").unbind("click");
```

Para no eliminar todo el manejo de eventos de un tipo determinado se puede especificar la función que se desea descartar en la lista de parámetros de la llamada a *unbind()*. Para ello habría que colocar la función dentro de una variable[70], para poder referirse a ella como parámetro de *unbind()*. El siguiente ejemplo muestra la definición de la función y cómo se desactiva con *unbind()* la atención al evento *onclic* de *p* al hacer clic sobre el botón con id="desactivar-evento".

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/
xhtml11.dtd">


<html>
<head>


<script type="text/javascript" src="jquery-1.6.4.js"></script>
<script type="text/javascript">


$(document).ready(function() {


    var funcionqueManeja = function() {
          if ($(this).css("color")!="rgb(255, 0, 0)")
                $(this).css("color", "rgb(255, 0, 0)");
                else
                      $(this).css("color", "rgb(0, 0, 255)");
          }


    $("p").bind("click mouseenter mouseleave",funcionqueManeja);
    $("#desactivar-evento").bind("click", function(){
          $("p").unbind("click", funcionqueManeja );})
});


</script>
<style media="screen">body { background: #666; }   </style>
```

[70] En la sección 4.3 se habla más sobre cómo definir una función.

```
</head>
<body>
<p> El texto cambia de color al entrar o salir con el ratón o hacer click</p>
<input id="desactivar-evento" type=button value="Quitar el cambio de color cuando se
hace click" >
</body>


</html>
```

Se puede observar en el código que, al desactivar el manejo de *click()*, el cambio de color sigue funcionando cuando el puntero entra y sale del párrafo ya que *mouseenter* y *mouseleave* no se han desactivado.

toggle()

Toggle() es otro método muy usado en jQuery. A este método se le pasan dos funciones. En función de las veces que se hace clic sobre el elemento que lo define, se ejecutará la primera función o la segunda. La primera vez que se hace clic sobre el elemento (y todas las veces impares), se ejecuta la primera función y la segunda vez que se hace clic el elemento (y todas las veces pares) se ejecuta la segunda función:

El siguiente ejemplo usa el método *toggle()* para cambiar el color de un párrafo *p* cuando se hace clic. Se puede observar en el código que en este caso las funciones que manejan el evento *onclick()* (*poner_Rojo*, *poner_Azul*) han sido definidas fuera en forma de variable.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/
xhtml11.dtd">


<html>
<head>
    <title>Ejemplo uso toggle()</title>
    <script type="text/javascript" src="jquery-1.6.4.js"></script>
    <script type="text/javascript">
          var poner_Rojo= function() {$(this).css("color", "rgb(255, 0, 0)");      }
          var poner_Azul= function() {$(this).css("color", "rgb(0, 0, 255)");      }
          $(document).ready(function() {
                $("p").toggle(poner_Rojo,poner_Azul);
          });
    </script>
    <style media="screen">     body { background: #666; }</style>
</head>
<body>
    <p> El texto cambia de color hacer click</p>
</body>


</html>
```

4.2 CAMBIO DE LAS PROPIEDADES DE UN ELEMENTO

Una vez vistos las funciones básicas de jQuery y cómo se seleccionan elementos sobre los que aplicar esas funciones, en esta sección se explicará cómo se pueden cambiar propiedades de los elementos (propiedades definidas principalmente con CSS) así como añadir nuevos elementos y texto.

4.2.1 CAMBIO DE LAS PROPIEDADES CSS

Si se desea obtener el valor de una propiedad para un elemento determinado se utiliza el método *.css()*. El parámetro de este método es el nombre de la propiedad *css* que se desea obtener.

Por ejemplo, si se desea saber con qué color está definido los elementos <p> se pondría:

```
$("p").css("color");
```

Además, *.css()* permite establecer el valor de una propiedad determinada. Por ejemplo, si se desea cambiar el color (propiedad color) de los elementos <p> se pondría:

```
$("p").css("color","red");
```

El valor del color admite todas las posibilidades que da CSS.

También se pueden establecer varias propiedades a la vez. Por ejemplo, el siguiente código modifica las propiedades CSS (*color*, *background*, *font-weight*) de los elementos <p>.

```
$("p").css({ color: "red", background: "blue", font-weight: "bold" });
```

4.2.2 AÑADIR NUEVOS ELEMENTOS

Además de cambiar propiedades, jQuery permite insertar nuevos elementos en un HTML haciéndolo (dinámico).

.append() permite insertar un contenido como el último hijo (árbol DOM) de cada elemento de la colección seleccionada. Si lo que se desea es insertarlo como primer hijo, entonces se usa *.prepend()* de la misma manera.

Por ejemplo, si se desea añadir una etiqueta <p> dentro de dos llamados "misdivs" se pondría:

```
<html>
<head>
<title>Ejemplo uso append </title>
<script type="text/javascript" src="jquery-1.6.4.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $(".misdivs").bind("click mouseenter mouseleave", function(e){
        $(".misdivs").append("<p><b>Texto</b>insertado en los div</p>");
```

```
        })
    });
</script>
<style media="screen">
    body { background: #666; }
    .misdivs {background: #623; }
</style>
</head>
<body>
    <div class="misdivs">Mi div 1 </div>
    <div class="misdivs">Mi div 2</div>
</body>
</html>
```

De la misma manera que se añade contenido se puede añadir elementos existentes en otra posición del HTML. Por ejemplo, el siguiente código mueve una etiqueta <h1> al interior de los al hacer clic con el ratón.

```
<html>
<head>
<title>Ejemplo uso append </title>
<script type="text/javascript" src="jquery-1.6.4.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $(".misdivs").bind("click", function(e){
                $(".misdivs").append($("h1"));
        })
    });
</script>
<style media="screen">
    body { background: #666; }
    h1{color:#00CC00;}
    .misdivs {background: #623; }
</style>
</head>
<body>
    <h1> Titulo </h1>
    <div class="misdivs">Mi div 1 </div>
    <div class="misdivs">Mi div 2</div>
</body>
</html>
```

4.2.3 AÑADIENDO TEXTO

Para añadir texto a un elemento se utiliza el método *.text()*.

.text(). Este método permite obtener o añadir un texto a un elemento determinado (que permita texto). Puede ser utilizado tanto en HTML como en XML[71]:

En la forma *.text()* obtiene el texto de un elemento (no es válido para elementos de formularios tipo <input> ya que para ello se utiliza *.val()*).

En la forma *.text(cadena_de_texto)* escribe ese texto en el elemento que lo invoca.

Por ejemplo, si se desea escribir dentro de los párrafos <p> de un HTML la cadena "nuevo texto" se podría:

```
$("p").text("<b>Some</b> new text.");
```

Es importante destacar que el texto, aunque sea HTML no se interpreta como tal, y que aparecería en pantalla el texto tal y como está.

4.2.4 OTROS MÉTODOS

Otros métodos relacionados con propiedades de los elementos son:

- **.innerWidth()** e **.innerHeight()**[72]: para un elemento devuelve sus dimensiones internas (anchura y altura, respectivamente) contando el *padding* del elemento pero no el borde.
- **.outerWidth()** y **outerHeight**(): devuelve las dimensiones externas (anchura y altura, respectivamente) del elemento contando el *padding* del elemento y su borde.
- **.offset()**[73]: indica la posición del elemento real, teniendo en cuenta los márgenes del elemento. Lo devuelve en un objeto con dos atributos *top* y *left*.

4.3 EJECUCIÓN DE SECUENCIAS DE FUNCIONES

Antes de entrar de lleno en métodos relacionados directamente con la interacción, en esta sección se trata otro de los aspectos de jQuery más utilizados y que hay que tener en cuenta para una correcta programación: los *callbacks*.

En jQuery cuando se trabaja de cara a la interfaz, es habitual utilizar secuencia de métodos o funciones apilados para lograr efectos más elaborados.

---

[71] Aunque en este capítulo solo se ha trabajado con HTML, jQuery se puede utilizar también para generar y modificar XML.

[72] En el API de jQuery se puede encontrar más sobre estos métodos: *http://api.jquery.com/category/dimensions/*

[73] En el API de jQuery se puede encontrar más sobre este método: *http://api.jquery.com/offset/*

Por ejemplo, si se desea que unos con identificador "#misdivs" se desvanezca tardando 2 segundos, luego se cambie su color y termine apareciendo de nuevo en 2 segundos, se podría usar la siguiente secuencia:

```
$("#misdivs").fadeOut(2000);
$("#misdivs").css({background: "blue"});
$("#misdivs").fadeIn(2000);
```

Esto mismo se hubiese conseguido haciendo una secuencia de este tipo:

```
$("#misdivs").fadeOut(2000).css({background: " blue "}).fadeIn(2000);
```

Para las dos sintaxis el problema es el mismo: para jQuery todo se hace al mismo tiempo. Por eso, como los efectos de aparecer y desvanecerse tardan 2 segundos, y cambiar la propiedad es casi inmediato, entonces lo que se verá será que se cambia la propiedad (color de fondo azul) y luego aparece y desaparece el .

La solución es utilizar efectos *.delay()* (que se verán en la sección 4.4.4) o utilizar la pila de ejecución de funciones. Aquí se explicará esta segunda opción:

Cualquier método o función en jQuery puede recibir como parámetro la función que se ejecutará después de ejecutar el método o función correspondiente. Esta función se llama "*callback*". Al usar *callblack*, en el ejemplo anterior se puede ejecutar el primer desvanecimiento, y en el *callback* de ese método colocar el cambio de la propiedad con *.css()* y que aparezca de nuevo el elemento. El código sería:

```
$("#micapa").fadeOut(1000, function(){
$("#micapa").css({'top': 300, 'left':200});
$("#micapa").fadeIn(1000);
});
```

Sin embargo, también se puede hacer un código más elegante definiendo primero la función y luego invocándola:

```
var ocultar=function () {
                $("#misdivs3").css({background: "green"});
                $("#misdivs3").fadeIn(2000);
                };
```

Para llamarla se haría:

```
$("#misdivs3").fadeOut(2000, ocultar);
```

El siguiente código muestra los problemas comentados para el "#misdivs1" y la solución con *callback* definido dentro del propio método "#misdivs2" y definida fuera "#misdivs3".

```
<html>
<head>
<title>Ejemplo uso append </title>
<script type="text/javascript" src="../jquery-1.6.4.js"></script>
<script type="text/javascript">
    var ocultar=function () {
```

```
                $("#misdivs3").css({background: "green"});
                $("#misdivs3").fadeIn(2000);
                };
    $(document).ready(function() {

            $("#misdivs").fadeOut(2000).css({background: "blue"}).fadeIn(2000);
            $("#misdivs2").fadeOut(2000, function(){
                $("#misdivs2").css({background: "red"});
                $("#misdivs2").fadeIn(2000);
            });
            $("#misdivs3").fadeOut(2000, ocultar);
    });

</script>
<style media="screen">
    body { background: #666; }
    h1{color:#00CC00;}
    .misdivs {background: #623; }
</style>
</head>
<body>
    <h1> Titulo </h1>
    <div id="misdivs">mis divs</div>
    <div id="misdivs2">mis divs</div>
    <div id="misdivs3">mis divs</div>
</body>
</html>
```

4.4 COMPORTAMIENTO DE LOS ELEMENTOS. EFECTOS VISUALES

Una vez vistos los elementos de jQuery más básicos para una correcta programación, esta sección detalla los métodos asociados con efectos visuales de los elementos. De esta manera se pueden conseguir animaciones modificando el comportamiento de los elementos.

A los efectos que se muestran a continuación se les puede añadir el método *toggle()* visto en la sección 4.1.2 para poder intercalar funciones dependiendo del número de clic hechos sobre un elemento, y conseguir con ello efectos visuales muy comunes en los desarrollos web actuales[74].

[74] Los ejemplos mostrados utilizan el evento *click()* para tratar la interacción, sin embargo, en la sección 4.5se mostrarán el resto de eventos que maneja jQuery.

4.4.1 EFECTOS BÁSICOS

jQuery permite que los sitios web incorporen pequeños efectos visuales y animaciones que, bien empleados, mejoran la interacción y la usabilidad del sitio. Los efectos visuales más extendidos son los siguientes:

```
.show ([duración] [, easing] [, callback])
```

Muestra un elemento que estaba oculto (generalmente con una propiedad CSS). Cuando un elemento se muestra (*show*) y se oculta (*hide*) se queda con los valores originales (de las propiedades CSS) que tenía antes de mostrarse.

- **duración**: una cadena (*slow o fast*) o un número (milisegundos) para determinar el tiempo que la animación se ejecutará.
- **easing**: es la función que se usará para el tipo de transición. Por defecto, *show* anima al mostrar y ocultar modificando el alto, ancho y la opacidad del objeto. Para ver más funciones de transición es necesario un *plugin*[75].
- **callback**: una función para llamar una vez finalizada la animación (tal y como se mostró en el apartado 4.3).

Todos los parámetros de este método son opcionales (por eso están entre [ ]).

Un ejemplo para mostrar todos los párrafos lentamente sería: *$("p").show("slow");*

```
.hide ([duración] [, easing] [, callback])
```

En esta función los parámetros tienen el mismo significado que en *show()*, pero con la acción contraría, es decir, la de ocultar un elemento.

El siguiente ejemplo muestra el uso de para mostrar textos que luego puede ser ocultado con *.hide()* a una velocidad de 500ms. Hay dos botones, uno para mostrar los y otro para ocultarlos (uno detrás de otro). Los elementos se ocultan con velocidad 500ms. Este ejemplo está inspirado en el mostrado en el API de jQuery[76]. La Figura 4.2 muestra el resultado del código.

```
<html>

    <head>

        <title>Ejemplo de animación con Show() y Hide()</title>

        <script type="text/javascript" src="jquery-1.6.4.js"></script>
        <script type="text/javascript">
        var ocultar=function () { $(this).prev().hide("fast", ocultar); }
```

[75] *http://jqueryui.com/* Esta librería implementa efectos específicos para Interfaz de Usuario.

[76] *http://api.jquery.com/show/*

```
        $(document).ready(function() {
              $("#hidr").click(function () {
                    $("span:last-child").hide("fast", ocultar);
        });
              $("#showr").click(function () {
                  $("span").show(500);
              });
        });


        </script>
        <style media="screen">
        body { background: #666; }
        span { background:#def3ca; padding:3px; float:left; }
        </style>


    </head>
    <body>
          <button id="hidr">Ocultar</button>
          <button id="showr">Mostrar</button>
          <div>
                <span>Habia</span>
                <span>una </span>
                <span>vez</span>
              <span>un</span>
                <span>desarrollador </span>
                <span>Web</span>
              <span>...</span>
          </div>
    </body>
</html>
```

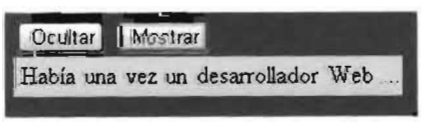


Figura 4.2. *Ejemplo de uso de show() y hide()*

4.4.2 EFECTOS FADING (OPACIDAD-TRANSPARENCIA DE LOS ELEMENTOS)

Fading son efectos de desvanecimiento de elementos. El *fading* juega con la opacidad y la transparencia para lograr un gran atractivo visual a la hora de mostrar y ocultar elementos.

La sintaxis de los efectos Fading de los que dispone jQuery es la siguiente:

.fadeIn ([duración] [, easing] [, callback])
.fadeOut ([duración] [, easing] [, callback])
.fadeToggle ([duración] [, easing] [, callback])

Todos los parámetros son opcionales y tienen el mismo significado que los vistos en *show()* y *hide()*.

- *fadeIn()*: muestra un elemento con opacidad gradual (hasta llegar a 1).
- *fadeOut()*: oculta un elemento con opacidad gradual(hasta llegar a 0).
- *fadeToggle()*: al igual que Toggle, alterna el proceso de aparecer y desvanecer según se haga clic sobre el elemento un número par de veces o un número impar.

.fadeTo (duración, opacidad [, easing] [, callback])

Ese método coloca el elemento a una opacidad determinada (*opacidad*) en un tiempo establecido (*duración* - número que indica milisegundos, *fast* o *slow*). La opacidad toma como valor un valor entre 0 y 1. Tanto *duración* como *opacidad* son parámetros obligatorios en este método.

El siguiente ejemplo[77] muestra el uso de *fadeOut()*. En un texto se seleccionan la palabras que son *adjetivos*, y al hacer clic sobre ellas, éstas desaparecen y se muestra un mensaje en un (inicialmente vacío).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/
xhtml11.dtd">
<html>
<head>
<title>Ejemplo de animación con FadeOut()</title>
<script type="text/javascript" src="jquery-1.6.4.js"></script>
<script type="text/javascript">
$(document).ready(function() {
$("span").hover(function () { $(this).addClass("hilite"); }, function () {    $(this).
removeClass("hilite"); });


    $("span").click(function () {
        $(this).fadeOut(1000, function () {
          $("div").text("'" + $(this).text() + "' ¡Se ha desvanecido!");
          $(this).remove();
          }); //function del Fade
}); //Function del Click
```

[77] Código inspirado en los ejemplos del API jQuery: *http://api.jquery.com/fadeOut/*

```
    });
</script>
<style media="screen">
    span { cursor:pointer; }
    span.hilite { background:yellow; }
    div { display:inline; color:red; } /*Se pone inline para que al quitarlo se ocupe su hueco */
</style>


</head>
<body>
<h3>Encuentra los adjetivos <div></div></h3>
<p> La chaqueta <span>roja</span> el lo mejor cuando hace un tiempo <span>cálido</span>
¿verdad?</p>
</body>
</html>
```

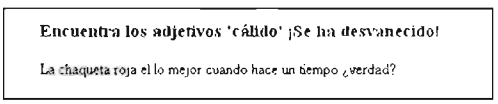


Figura 4.3. *Ejemplo de uso de fadeOut()*

La Figura 4.3 muestra el resultado. En el ejemplo la función *$("span").hover* activa una nueva clase asociada a <span> que lo pone con fondo amarillo al entrar el ratón y lo deja normal (removeClass) al sacar el ratón del <span> (en la sección 4.5.1 se muestra el evento *hover* más ampliamente). Por su parte, *$(this).fadeOut* desvanece el y después rellena el con el nombre del desvanecido y elimina (*remove*) el del árbol DOM. Al estar el definido en el estilo como *inline*, al eliminarlo, todo el texto se desplaza ocupando su hueco.

ACTIVIDADES 4.1

- Modifique el código que muestra como resultado la Figura 4.3., para que cuando el usuario haga clic sobre el texto que aparece en el todo los ocultos aparezcan de nuevo en su posición gradualmente (*FadeIn()*).

SOLUCIÓN

La solución de la actividad se consigue con las siguientes modificaciones del código.

```
$("span").click(function () {
 $(this).fadeOut(1000, function () {
 $("div").text("'" + $(this).text() + "' ¡Se ha desvanecido!");
//$(this).remove(); SE ELIMINA PORQUE SI DESPARECE DEL DOM YA NO LO PUEDO MOSTRAR
CON FADEIN()
  }); //function del Fade
 }); //Function del Click
 $("div").click(function () { $("span").fadeIn(); });
```

4.4.3 EFECTOS SLIDING (DESPLAZAMIENTO)

Sliding son efectos de desplazamiento de elementos. Los *sliding* permiten hacer efectos de plegado y despliegue.

La sintaxis de los efectos Sliding de los que dispone jQuery es la siguiente:

.slideUp ([duración] [, easing] [, callback])

.slideDown ([duración] [, easing] [, callback])

.slideToggle ([duración] [, easing] [, callback])

Todos los parámetros son opcionales y tienen el mismo significado que los vistos en *show()* y *hide()*.

- *slideUp()*: recoge los elementos en altura de manera gradual.
- *slideDown()*: despliega los elementos en altura de manera gradual.
- *slideToggle()*: al igual que Toggle, alterna el proceso de plegado y desplegado en altura según se haga clic sobre el elemento un número par de veces o un número impar.

En el siguiente ejemplo[78] se muestran cuatro elementos que se despliegan hacia abajo. Tres de ellos tienen en su interior texto y uno tiene un botón. Al hacer clic en el texto *¡Pulsa aquí!* se despliegan los . A hacer otra vez, de vuelven a contraer (con *hide()*). El código es el siguiente:

```
<html>
    <head>
    <title>Ejemplo de animación con Show() y Hide()</title>
    <script type="text/javascript" src="jquery-1.6.4.js"></script>
    <script type="text/javascript">
    $(document).ready(function() {
          $(document.body).click(function () {
                if ($("div:first").is(":hidden")) {
                $("div").slideDown("slow");
                } else {
                $("div").hide();
                }
          });
    });
</script>
<style>
    div { background:#de9a44; margin:3px; width:80px; height:140px; display:none;
float:left; }
    </style>
```

[78] Código inspirado en los ejemplos del API jQuery: *http://api.jquery.com/slideDown/*

```
    </head>
    <body>
    ¡Pulsa Aquí!
    <div>Los efectos no son solo para divs o spans</div>
    <div>También son para cualquier elemento </div>
    <div><button id="hidr">Botones también</button></div>
    <div><p> aunque no estén incluidos en un div </p></div>
    </body>
</html>
```

La Figura 4.4 muestra el resultado al hacer clic en **¡Pulsa Aquí!**. Si se hace clic de nuevo se vuelve a la situación original.

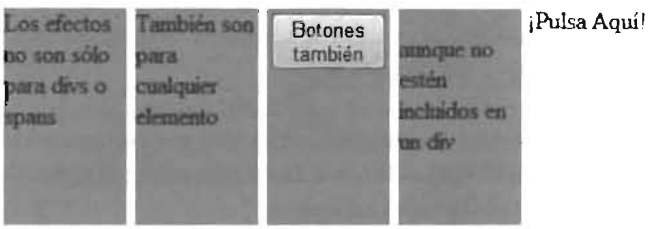


Figura 4.4. *Ejemplo de uso de slideDown()*

ACTIVIDADES 4.2

Modifique el código que muestra como resultado la Figura 4.4 para que cuando el usuario haga clic sobre el texto **¡Pulsa Aquí!** por segunda vez se recojan los hacia arriba (*slideUp()*).

SOLUCIÓN

La solución pasa por sustituir *$("div").hide();* por *$("div").slideUp("slow");* .En ambos casos los quedan ocultos (hidden=true).

4.4.4 OTROS EFECTOS

Además de los efectos vistos en las secciones anteriores existen otros que aumentan las prestaciones a la hora de hacer animaciones.

.animate (propiedades, [duración] [, easing] [, callback])

Este método anima un elemento según las propiedades CSS que se pongan en el parámetro *propiedades*. Los parámetros opcionales [*duración*], [*easing*] y [*callback*] tienen el mismo significado que en los métodos vistos en las secciones anteriores.

animate() es similar a usar el método *.css()*[79] salvo que el rango de posibildiades en *animate()* es muy menor: la propiedades CSS que se utilicen deben tener valores numéricos, aquellas que no son numéricas no puedes animarse. Por ejemplo. *width* o *height* reciben valores numéricos y pueden animarse, sin embargo, *background* no puede animarse si se le da un valor no numérico. Los valores de medida son tratados en píxeles (px) a menos que se indique lo contrario (em o %)[80].

Es interesante destacar que el valor de las propiedades puede ser relativo al que tenga actualmente. Para ello se utiliza "+=" o "-=".

El siguiente código[81] utiliza un para cambiarlo de tamaño (al hacer clic sobre él) y para moverlo cuando se usen los botones (*left* y *right*)

```
<html>
<head>
    <title>Ejemplo de animación con animate()</title>


    <script type="text/javascript" src="jquery-1.6.4.js"></script>
    <script type="text/javascript">
    $(document).ready(function() {
        $("#right").click(function(){ //Animación del botón derecho.
                $(".block").animate({"left": "+=50px"}, "slow");
        });
        $("#left").click(function(){//Animación del botón izquierdo.
          $(".block").animate({"left": "-=50px"}, "slow");
        });
        $("div").click(function(){ //al hacer click en div, cambia de tamaño
          $("div").animate({
            width: "70%",
            opacity: 0.5,
            marginLeft: "0.6in",
            fontSize: "3em",
            borderWidth: "10px"
          }, 1500 );
        });
});


    </script>
    <style>
    div {
```

[79] El método *css()* ha sido explicado en la sección 4.2.1

[80] En el Capítulo 2 se tratan las propiedades CSS más ampliamente

[81] Inspirado en el ejemplo disponible en el API jQuery: *http://api.jquery.com/animate/*

```
            position:absolute;
            background-color:#abc;
            left:50px;
            width:90px;
            height:90px;
            margin:5px;
        }
        </style>
    </head>
    <body>
        <button id="left">&laquo;</button> <button id="right">&raquo;</button>
        <div class="block">Pulsa AQUÍ también</div>
    </body>
    </html>
```

La Figura 4.5 muestra el resultado después de una ejecución:

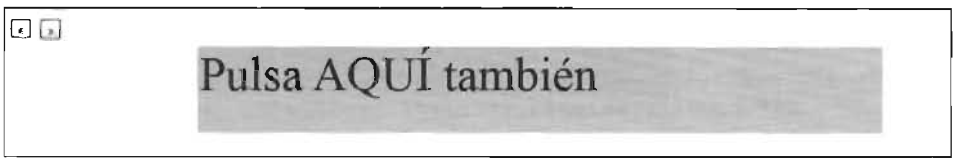


Figura 4.5. *Ejemplo de uso de animate()*

.delay(duración [, NombreCola])

Del método *.delay()* ya se comentó algo en la sección 4.3. Este método retrasa un tiempo especificado (duración) la animación de los elementos que se encuentran en una estructura cola (NombreCola), a la espera de ser animados. El significado de sus parámetros es el siguiente.

- *duración*: un número en milisegundos que indica cuánto tiempo se retrasará el comienzo de una animación con respecto al siguiente elemento de una cola.
- *NombreCola*: una cadena que contiene el nombre de la cola que contiene los elementos que se retrasarán .Por defecto existe la cola estándar *fx* que es la que se usa si no se pone valor a este parámetro[82].

El siguiente código, al hacer clic sobre cualquiera de los párrafos, se recogen todos (*slideUp(300);*) ellos, aparece (*fadeIn(800);*)los que tienen identificador "antes" y más tarde (*delay(800);*) los que tienen identificador "después".

```
<html>
<head>
    <title>Ejemplo de animación con delay()</title>
    <script type="text/javascript" src="jquery-1.6.4.js"></script>
```

[82] Para saber más sobre colas (*queue*) y su métodos (*clearQueue()*, *doQueue()*, etc.) ver: *http://api.jquery.com/category/effects/*

```
    <script type="text/javascript">
    $(document).ready(function(){
          $("div").click(function() {
             $("div.antes").slideUp(300);
             $("div.antes").fadeIn(800);
             $("div.despues").slideUp(300);
             $("div.despues").delay(800);
             $("div.despues").fadeIn(800);
             });
    });
    </script>
<style>
    div.antes {  background-color:#aac; }
    div.despues {background-color:#abc;     }
</style>
</head>
<body>
    <h3> Pulsa en cualquiera de los textos </h3>
    <div class="antes"><p> Antes </p> </div>
    <div class="despues"><p> Después  </p> </div>
    <div class="antes"><p> Antes </p> </div>
    <div class="despues"><p>Después  </p> </div>
</body>
</html>
```

La Figura 4.6. muestra los párrafos *Antes* y mientras comienza a mostrarse los párrafos *Después* (*fadeIn(800);*).

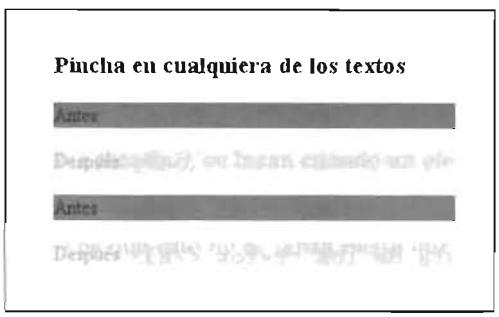


Figura 4.6. *Ejemplo de uso de delay()*

Junto con *delay()* otro método interesante es *stop()* que para la animación del elemento seleccionado si está actualmente en proceso.

ACTIVIDADES 4.3

Modifique el código que muestra como resultado la Figura 4.6 para que cada uno de los párrafos aparezcan secuencialmente al cargarse la página. Cada uno con un retraso de 1 segundo (1.000 ms) respecto al anterior.

SOLUCIÓN

Una posible solución sería:

```
<html>
<head>
    <title>Ejemplo de animación con delay()</title>
    <script type="text/javascript" src="jquery-1.6.4.js"></script>
    <script type="text/javascript">
    $(document).ready(function(){

              $("div.primero").fadeIn(300);
              $("div.segundo").delay(1000);
              $("div.segundo").fadeIn(800);
              $("div.tercero").delay(2000);
              $("div.tercero").fadeIn(800);
              $("div.cuarto").delay(3000);
              $("div.cuarto").fadeIn(800);
    });
    </script>
<style>
    div.primero {background-color:#aac;display:none; }
    div.segundo {background-color:#abc;    display:none;}
    div.tercero {background-color:#abc;display:none;       }
    div.cuarto {background-color:#abc;     display:none;}
</style>
</head>
<body>
    <h3> Pulsa en cualquiera de los textos </h3>
    <div class="primero"><p>   Primero </p> </div>
    <div class="segundo"><p>   Segundo   </p> </div>
    <div class="tercero"><p>   Tercero  </p> </div>
    <div class="cuarto"><p>   Cuarto  </p> </div>
</body>
</html>
```

4.5 COMPORTAMIENTOS INTERACTIVOS

Una vez vistos los efectos más comunes usando jQuery, en esta sección se muestras los diferentes eventos (ratón y teclado) que atiende jQuery para la interacción del usuario con objetos de la web. Además, en la última parte se describen los eventos asociados a las ventanas.

4.5.1 EVENTOS DE RATÓN

Los eventos de ratón ejecutan una función cuando el usuario utiliza el puntero del ratón en un elemento determinado.

La estructura básica de todos los eventos es:

Elemento.Evento (función (handler));

Sobre un *elemento*, cuando el usuario lanza un *evento* lo atiende (se ejecuta) la *función*. A continuación se describen los eventos de ratón disponibles. En la descripción se supone una configuración de ratón para diestros, donde el botón izquierdo de ratón es el que se usa para seleccionar.

- **.click()**. Se lanza un evento clic cuando el usuario, colocado sobre un elemento (objeto) presiona el botón izquierdo del ratón y lo levanta, todo dentro del mismo objeto. Cualquier elemento HTML puede recibir este evento.
- **.dblclick()**. Es el llamado doble clic. Este evento se lanza cuando se repite dos veces la secuencia de un clic, es decir, sobre un elemento (objeto) el usuario presiona el botón izquierdo del ratón, lo levanta, lo vuelve a presionar y lo levanta, toda la secuencia dentro del mismo objeto. Cualquier elemento HTML puede recibir este evento.
- **.focusin()**. Se lanza este evento cuando un elemento (objeto) gana el foco. Por ejemplo, cuando se selecciona una caja de texto para poder escribir sobre ella. No todos los elementos HTML pueden recibir el foco.
- **.focusout()**. Es el evento contrario a *focusin()*, se lazan cuando un elemento (objeto) pierde el foco.
- **.mousedown()**. Este evento se lanza cuando el usuario presiona el botón del ratón sobre un elemento (objeto). La diferencia con respecto a *click()* es que éste no se lanza hasta que se suelta el botón dentro de ese mismo objeto. Cualquier elemento HTML puede recibir este evento.
- **.mouseup()**. Este evento se lanza cuando el usuario levanta el botón de ratón (previamente presionado). Cualquier elemento HTML puede recibir este evento.
- **.mouseenter()**. Se lanza este evento cuando el puntero del ratón entra en un elemento (objeto). No hace falta presionar ningún botón para que este evento se lance cuando el puntero entra en la región definida por el elemento. Cualquier elemento HTML puede recibir este evento.
- **.mouseleave()**.Se lanza este evento cuando el puntero del ratón sale de la región delimitada por un elemento (objeto). Es el evento opuesto a *mouseenter()*.Cualquier elemento HTML puede recibir este evento.

- **.mousemove()**. Este evento se lanza cuando el puntero de ratón se mueve dentro de un elemento. Cualquier elemento HTML puede recibir este evento.
- **.mouseover()**. Es parecido a *mouseenter()*, pero se diferencia cuando hay elementos anidados (por ejemplo, un <p> dentro de un *hijo* anidado dentro de un *padre*): se lanza el *mouseover()* del padre al pasar de <p> a hijo o de hijo a <p> o de hijo al padre.
- **.mouseout()**. Es el opuesto de *mouseover()*. Es parecido al *mouseleave()* pero con la diferencia con objeto anidados mostrada con *mouseover()*.

El siguiente ejemplo, cuando se ejecuta, muestra la diferencia entre *mouseover()* y *mouseenter()*.

```
<!DOCTYPE html>
<html>
<head>
<style>
div.first { position:relative;background-color: #3f3; left: 0px; height:400px;
width:400px;}
div.second { position:relative;background-color: #a3f; top:100px;left:100px;height:20
0px; width:200px;}
p {background:#33CC99;}
</style>
    <script type="text/javascript" src="jquery-1.6.4.js"></script>
    <script type="text/javascript">
          $(document).ready(function() {
                var cuenta_enter=0;
                var cuenta_over=0;
          $("div.first").mouseenter(function() {
                      $("p:first").text("con_mouseEnter "+ cuenta_enter++);
                });
                $("div.first").mouseover(function() {
                      $("p:last").text("con_mouseOver "+ cuenta_over++);


            });
      });
</script>


</head>
<body>
<div class="first">
    <div class="second">
          <p></p>
          <p></p>
```

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

En el ejemplo, al pasar el ratón de los párrafos <p> al hijo aumenta la cuenta de *cuenta_over*. Lo mismo al pasar del hijo al padre. Sin embargo, la *cuenta_enter* solo se incremente al pasar de fuera de los al padre.

.hover(). Este evento se lanza cuando el puntero está sobre un elemento o sale de él. Su principal ventaja es que permite definir una función para cuando el puntero entra en el elemento y otra para cuando salga. Es un evento muy usado con menús, cuando se desea hacer, por ejemplo, seleccionar opciones de menú, destacando alguna de sus propiedades o menús que se despliegan. Su sintaxis es:

Elemento.hover (función_entrada (handler), función_salida (handler));

El siguiente ejemplo muestra un submenú que aparece al colocarse con el ratón en la opción "localización".

```
<!DOCTYPE html>
<html>
<head>


 <script type="text/javascript" src="jquery-1.6.4.js"></script>
<script type="text/javascript">
    $(document).ready(function() {


                $("li#localizacion").hover(function () {

                        $("span#opcion_loc").show();


                        }, function () {

                        $("span#opcion_loc").hide();
                });

          });


</script>
 <style>
  ul { margin-left:20px; color:blue; }
  li { cursor:default; }
  span { display:none }
```

```
  </style>
</head>
<body>
  <ul>
    <li id="localizacion">Localización</li>
    <span id="opcion_loc">
    <ul>
          <li>Mapa</li>
        <li>Dirección</li>
        <li>Teléfonos</li>
    </ul>
    </span>
    <li id="recetas">Recetas</li>
  </ul>



</body>
</html>
```

ACTIVIDADES 4.4

- Modifique el código anterior, apoyándose en los eventos vistos en esta sección, para añadirle la siguiente funcionalidad:

  a. Que al hacer clic sobre una opción se despliegue el submenú y al hacer clic de nuevo se vuelva a ocultar.

  b. Que al moverse el puntero del ratón por encima de cualquier opción (menú o submenú) ésta cambie de color (a rojo), y al quitarse vuelva a su color original (a azul).

  La Figura 4.7 muestra cómo podría quedar el resultado con los dos menús desplegados:

Figura 4.7. *Ejemplo de menú desplegable*

SOLUCIÓN

La solución para mostrar el código de la Figura 4.7 es el siguiente:

```
<!DOCTYPE html>
<html>
<head>

 <script type="text/javascript" src="jquery-1.6.4.js"></script>
<script type="text/javascript">
    $(document).ready(function() {

            $("li").hover(function () {

                    $(this).css("color", "red");

                    }, function () {

                    $(this).css("color", "blue");
                });


            $("li#localizacion").toggle(function () {

                    $("span#opcion_loc").show();

                    }, function () {

                    $("span#opcion_loc").hide();
                });


            $("li#recetas").toggle(function () {

                    $("span#opcion_rec").show();

                    }, function () {

                    $("span#opcion_rec").hide();
                });

        });

</script>
```

```
 <style>
  body { background:#99CCFF;}
  ul { margin-left:20px; color:blue; }
  li { cursor:default; }
  span { display:none; background:#CCCCFF ;}
  </style>
</head>
<body>
  <ul>
    <li id="localizacion">Localización</li>
    <span id="opcion_loc">
    <ul>
          <li>Mapa</li>
        <li>Dirección</li>
        <li>Teléfonos</li>
    </ul>
    </span>
    <li id="recetas">Recetas</li>
    <span id="opcion_rec">
    <ul>
          <li>de Carne</li>
        <li>de Pasta</li>
        <li>de Verduras</li>
    </ul>
    </span>
  </ul>


</body>
</html>
```

4.5.2 EVENTOS DE TECLADO

Los eventos de teclado ejecutan una función cuando el usuario utiliza las teclas del teclado en un determinado elemento.

La estructura básica de todos los eventos es:

Elemento.Evento (función (handler));

Sobre un *elemento*, cuando el usuario lanza un *evento* lo atiende (se ejecuta) la *función*. A continuación se describen los eventos de teclado disponibles. La mayoría de ellos van asociados a formularios ya que suelen ser usados cuando el usuario entra mediante teclado.

- **.focusin()**. Se lanza este evento cuando un elemento (objeto) gana el foco. Por ejemplo, cuando se selecciona una caja de texto para poder escribir sobre ella. No todos los elementos HTML pueden recibir el foco.
- **.focusout()**. Es el evento contrario a *focusin()*, se lazan cuando un elemento (objeto) pierde el foco.
- **.keydown()**. Se lanza el evento cuando el usuario presiona una tecla. El evento se lanza cuando el elemento tiene el foco. Para determinar que tecla has sido presionada se examina el objeto *event* que se le pasa a la función que atiende el evento. jQuery ofrece la propiedad *.which* de *event* para recuperar el código de la tecla que se presiona- Si lo que se desea es obtener el texto que se ha introducido es mejor opción usar *.keypress()*.
- **.keyup()**. Se lanza el evento cuando el usuario libera una tecla. El evento se lanza cuando el elemento tiene el foco.
- **.keypress()**. Se lanza el evento cuando el usuario presiona una tecla pero con la idea de registrar qué carácter se ha pulsado. También se puede conseguir con *.keydown()*, sin embargo si se pulsa una tecla y se mantiene, *keydown()* solo registra un evento para esa tecla, mientras que *keypress()* registra uno por cada caracter introducido.

El siguiente ejemplo muestra un sencillo detector de qué tecla se ha presionado en cada momento. El parámetro *e* es de tipo *event* y es el que contiene la tecla pulsada, obtenida con *e.which*. Con *e.preventDefault();* se consigue no se escriba nada en el *textarea*, es decir se inhabilita el comportamiento habitual del evento, que es escribir las teclas en el *textarea*.

```
<!DOCTYPE html>
<html>
<head>
<style>
p {background:#33CC99;}
</style>
<script type="text/javascript" src="jquery-1.6.4.js"></script>
<script type="text/javascript">
          $(document).ready(function() {

                  $('#objetivo').keypress(function(e) {
                    e.preventDefault();
                  $("#salidapress").html(e.which + ": " + String.fromCharCode(e.which))

                  });
          });
</script>
</head>
<body>
<form>
  <fieldset>
    <label for="objetivo">Teclea cualquier cosa</label>
```

```
    <input id="objetivo" type="text" />
    <div id="salidapress"></div>
  </fieldset>
</form>
</body>
</html>
```

4.5.3 EVENTOS DE VENTANA

Los eventos principales de ventana son: *scroll* y *resize*. La estructura básica de los dos es la misma que en los eventos anteriores de ratón y teclado.

- **.scroll()**. Este evento se atiende cuando sobre un elemento que tiene barras de desplazamiento se mueve una de ellas. El evento se suele aplicar a elemento "ventana" pero también sobre *frames* o elementos con la propiedad CSS *overflow* puesta a *scroll*.
- **.resize()**. El evento se lanza cuando a un elemento tipo ventana se le cambia el tamaño.

El siguiente ejemplo muestra el comportamiento de ambos eventos. *Scroll()* se activa cuando se mueven las barras de desplazamiento o se pulsa en el texto "*PULSA Y lazo el evento pero no hago el scroll (desplazamiento)*". Al activarse aparece un texto "se hace *Scroll*", pero este se desvanece (*fadeout*). Por otro lado, cuando se modifica el tamaño de la ventana del navegador, si el ancho de la ventana es mayor de 500 px se pone un color de fondo para la caja de texto y si es menor otro.

```
<!DOCTYPE html>
<html>
<head>
<stylе>
div {overflow: scroll;
    width: 300px;
    height: 100px;
    font-family: Arial, Helvetica, sans-serif;
    color: #888888;
    padding: 7px 3px 0 4px;
    font-size: 12px;


}
span {display:inline;}
</style>
<script type="text/javascript" src="jquery-1.6.4.js"></script>
<script type="text/javascript">
```

```
        $(document).ready(function() {
                $('#prueba').scroll(function() {
                $("span#textoevento").css("display", "inline").fadeOut("slow"); });
                $('#repite').click(function() {  $('#prueba').scroll();
                });
                $(window).resize(function() {

                        if ($(window).width()>500)
                        {
                                $('div').css("background","#99CC99");
                        }
                        else
                        {
                                $('div').css("background","#9999CC");
                        }


                });
        });
</script>
</head>
<body>
<div id="prueba" >
  Lorem ipsum dolor sit amet, consectetur adipisicing elit,
  sed do eiusmod tempor incididunt ut labore et dolore magna
  aliqua. Ut enim ad minim veniam, quis nostrud exercitation
  ullamco laboris nisi ut aliquip ex ea commodo consequat.
  Duis aute irure dolor in reprehenderit in voluptate velit
  esse cillum dolore eu fugiat nulla pariatur. Excepteur
  sint occaecat cupidatat non proident, sunt in culpa qui
  officia deserunt mollit anim id est laborum.
</div>
<span id="textoevento">
  Se hace Scroll
</span>
<span id="repite"> PULSA Y lazo el evento pero no hago el scroll (desplazamiento)
</span>
</body>
</html>
```

4.6 REPRODUCCIÓN DE SONIDO, VÍDEO Y ANIMACIÓN

Como se comentó en el Capítulo 3, actualmente la reproducción de vídeo y audio está recayendo sobre las etiquetas <video> y <audio> de HTML5. Sin embargo, en algunos casos sustituyendo y en otros complementando, existen *plugins* específicos para jQuery para insertar vídeo y audio en la web.

Estos *plugins*, en algunos casos, gracias a la flexibilidad que ofrecen, son una alternativa personalizada a <video> y <audio> que permite diseños más potentes y visuales. Sin embargo, no siempre es así, y en otros muchos casos usar estos *plugins* perjudica (por rendimiento o por tiempo de desarrollo) más que ayuda, y es preferible usar <video> <audio>.

A continuación se muestran algunos de los *plugins* mejor valorados para manejar vídeo y audio con jQuery.

- **Acorn Media Player**[83]: es un *plugin* de jQuery que permite personalizar el reproductor de HTML5 <video> haciendo hincapié en la accesibilidad del recurso. Está destinada a la reproducción de vídeo.
- **Video JS**[84]: es un reproductor muy conocido para JavaScript y jQuery. Está destinada a la reproducción de vídeo.
- **jPlayer**[85]: otro reproductor muy versátil adecuado para un gran número de navegadores y formatos tanto de vídeo como de audio.
- **Flare Video**[86]: otro reproductor que ofrece solución para todos los navegadores. Está destinada a la reproducción de vídeo.
- **Media Element**[87]: un reproductor jQuery fácil de usar. Esa adecuado tanto para vídeo como para audio.
- **Simple Player**[88]: un reproductor solo de audio. Soporta CSS para definir la apariencia del reproductor. Es solo adecuado para los navegadores que soprotan HTML con formatos *mp3* y *ogg*.

En los sitios web de cada reproductor se puede obtener información y ejemplos sobre cómo usar cada *plugin* e insertarlo en las páginas.

---

83 *http://ghinda.net/acornmediaplayer/*

84 *http://videojs.com/*

85 *http://www.jplayer.org/*

86 *http://flarevideo.com/*

87 *http://mediaelementjs.com/*

88 *http://jay-notes.blogspot.com/2010/06/simple-player-very-simple-html5-audio.html*

ACTIVIDADES 4.5

Modifique el código de la Actividad 2.7 para incluir los siguientes efectos:

a. Al cargar la página solo debe aparecer el pie de página. El resto de elementos deben permanecer ocultos.
b. Dentro del pie de página insertar 4 , cada uno de un color.
c. Cuando el usuario hace clic (un número impar de veces) en cada uno de los tres primeros aparecerá el correspondiente *encabezado*, *menú* y *cuerpo de la página*.
d. Cuando el usuario hace clic (un número par de veces) en cualquier de los tres desaparecerá el correspondiente *encabezado*, *menú* y *cuerpo de la página*.
e. Para el cuarto cuando el usuario hace clic, desaparecen los 4 de forma gradual quedando en el pie de página un texto "aquí va el nuevo pie de página".

Si el usuario hace clic en el último antes de mostrar todos las partes de la página, ya no se podrán mostrar ya que todos los estarán invidibles.

La Figura 4.8 muestra la página al inicio, la Figura 4.9 la página al hacer clic en los tres primeros y luego en el último.

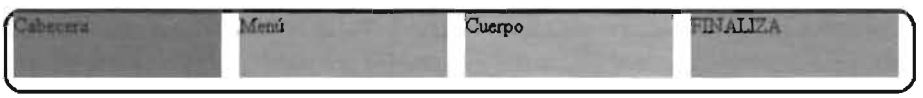


Figura 4.8. *Antes de la animación*

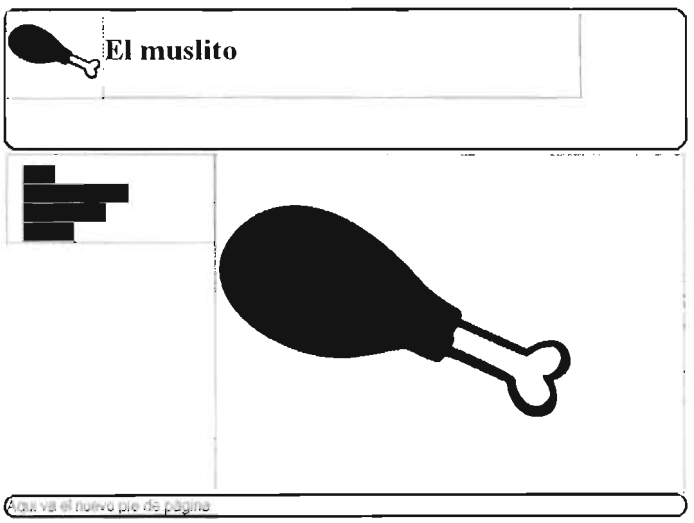


Figura 4.9. *Al final de la animación*

SOLUCIÓN

El código de solución está disponible en el material adicional: *Actividad final_solucion.html*

4.7 CONCLUSIONES

En este capítulo se ha hecho una introducción a jQuery desde el punto de vista del diseño y creación de interfaces de usuario. Eventos de ratón, teclado, efectos y animaciones han sido los puntos principales que se han tratado. La mayor parte de la funcionalidad ha sido ilustrada con códigos de ejemplo, lo cual es una ayuda no solo para entender los aspectos de interfaz, sino también de cómo se incluye el código en una página web.

En la web hay gran cantidad de código realizado por la comunidad jQuery que ofrece desarrollos de interfaz bastante completos: botones, menús, animaciones, etc. El lector familiarizado con jQuery podrá interpretar esos ejemplos y modificarlos para poder incluirlos en sus páginas (si la licencia lo permite). Un ejemplo es mostrado en Onextrapixel[89] para el desarrollo de un botón con JQuery y CSS.

Sin embargo, con todo lo visto, jQuery es un lenguaje mucho más amplio, ya que como lenguaje de programación ofrece alternativas relacionadas con las condiciones, bucles, acceso a datos, etc. Junto a esto, JavaScript en general y jQuery en particular es el lenguaje con el que se implementa la tecnología AJAX (Asynchronous JavaScript And XML) lo que hace que este lenguaje sea todavía más extenso y completo.

Por lo tanto, se recomienda que lector interesando en el desarrollo de aplicaciones web profundice más en el tema, abarcando el conocimiento de otras funcionalidades proporcionadas por jQuery, *plugins* sobre jQuery y su trabajo con AJAX. En la web hay gran cantidad de tutoriales y bibliografía relacionada con esta tecnología. Aquí se muestra solo algunos ejemplos en inglés:

- **jQuery API**[90]: toda la documentación on line sobre el API de jQuery con ejemplos.
- **w3schools**[91]: tutorial *on line* sobre las funciones jQuery.
- **Canal**[92] **de vídeo-tutoriales**: vídeos sobre el API de jQuery en *YouTube.com* (inglés).

[89] *http://www.onextrapixel.com/2010/12/06/creating-3d-buttons-with-css-jquery/*

[90] *http://api.jquery.com/*

[91] *http://www.w3schools.com/jquery/default.asp*

[92] *http://www.youtube.com/watch?v=GNb8T5NBdQg&list=PL0EFA1232C66601D7&index=1&feature=plpp_video*

RESUMEN DEL CAPÍTULO

En este capítulo se ha mostrado mediante ejemplos el uso de jQuery para desarrollar animaciones y efectos en la web. Aunque hasta hace poco tiempo, las animaciones en la Web y la interacción era exclusividad de aplicaciones Flash, actualmente la Web tiende a utilizar jQuery, HTML5 y CSS para desarrollar animaciones fácilmente soportadas por todos los navegadores.

El capítulo no pretende mostrar todo lo relacionado con jQuery, ni siquiera todo los relacionado con el desarrollo de interfaces de usuario con jQuery (eso también daría para un libro de cientos de páginas). Sin embargo, sí pretende dar al lector una muestra de qué permite hacer y cómo, acercándolo de manera práctica al uso de esta tecnología.

EJERCICIOS PROPUESTOS

1. Busque en Internet dos herramientas libres diferentes a las descritas en el capítulo que permitan crear código jQuery automáticamente para hacer animaciones. Describa su funcionalidad.

2. En parejas, crear dos animaciones con alguna de las herramientas de generación de código mostradas en el capítulo. Incluir cada miembro del equipo esa animación en el código de la web creada en el ejercicio 5 del Capítulo 2. Las animaciones creadas deben estar en consonancia con el diseño de la plantilla.

3. En grupos de dos, para la web creada en el ejercicio anterior, crear con jQuery directamente una pantalla de bienvenida que se muestre al principio de cargar la página y que sirva de introducción. Esa página debe tener animación. Cuando la animación termina desaparece y aparece la web lista para usarse. Enumeren qué dificultades técnicas se han encontrado al hacer esta actividad.

4. En grupos de dos, buscar en Internet dos diseños de botones creados con JQuery+CSS y HTML5. Los botones seleccionados deben estar en consonancia con el diseño de la plantilla seleccionada para la Web del ejercicio anterior.

5. Continuando con el ejercicio anterior, cada miembro debe insertar el botón descargado en la Web y asociarle una función. ¿Qué dificultadas se han encontrado al hacer la integración?

TEST DE CONOCIMIENTOS

1 No se puede decir sobre jQuery:

a) Es una librería basada en JavaScript muy adecuada para hacer animaciones en un contexto de HTML5.

b) Es un lenguaje alternativo al uso de Flash como herramienta para crear animaciones.

c) Es un lenguaje sencillo que no necesita de herramientas que generen código jQuery automáticamente a partir de modelos visuales.

2 En jQuery es falso que:

a) Se pueden cambiar propiedades HTML pero no se pueden modificar propiedades CSS.

b) Se pueden modificar propiedades CSS usando el método *css()*.

c) Se puede añadir texto a un HTML usando el método *text()*.

3 Respecto a los efectos de animación en jQuery:

a) *delay()* permite retrasar un tiempo especificado la ejecución para un elemento de un efecto.

b) Si se desea transformar una elemento a una opacidad determinada se utiliza *Fade()*.

c) El método *SlideToggle()* solo funciona asociado con el método *show()*.

4 Respecto a los eventos de ratón *mouseover()* y *mouseenter()*:

a) Ambos se comportan de manera idéntica.

b) Ambos son idénticos salvo cuando hay elementos anidados. *mouseover()* se ejecuta cuando se pasa de un hijo a un padre.

c) Ambos son idénticos salvo cuando hay elementos anidados. *mouseenter()* se ejecuta cuando se pasa de un hijo a un padre.

5 Un evento *click()* en *jQuery()*:

a) Si se captura un *click()* no se puede capturar un *mousedown()*.

b) Se ejecuta cuando sobre un objeto se presiona el botón izquierdo del ratón (diestros) y se suelta sobre el mismo elemento.

c) Un *click()* es incompatible con capturar un *dblclick()*.

6 Sobre algunos métodos de jQuery es falso que:

a) *hover()* permite llamar a una función cuando el puntero entra en un elemento y a otra cuando sale.

b) *toggle()* permite asociarle dos funciones. En los *click()* pares se ejecutará una y en los impares la otra.

c) *bind()* permite asignar varios eventos a un elemento, pero de uno en uno, con tantas llamadas a *bind()* como eventos se quieran asignar.