Integración de contenido interactivo

Tema 4

Módulo: Diseño de Interfaces Gráficas

CGS Desarrollo de Aplicaciones Web

Profesor: Sergio Báez Ferrer

ÍNDICE

ÍNDICE

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 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 *http://www.google.com/logos/2011/lespaul.html*

4.1 ELEMENTOS INTERACTIVOS BÁSICOS Y AVANZADOS

Back to Google Logos

4.1 ELEMENTOS INTERACTIVOS BÁSICOS Y AVANZADOS

- Este 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 tema 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
- En ese capítulo nos centraremos en la interacción de objetos con jQuery, usando inevitablemente HTML5 y CSS3.
- **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.**

4.1.1 INTRODUCCIÓN A JQUERY

- Para utilizar jQuery, solamente es necesario **descargar la librería** (Por ejemplo de http://jquery.com/download/*)* 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:** más adecuada para que la descargue el cliente ya que ocupa menos espacio (jquery.min.js)
  - **Descomprimida:** es más adecuada cuando se está trabajando en el desarrollo (jquery.js)

- 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 (este comando no espera a que se carguen las imágenes, con los elementos definidos en el HTML es suficiente). 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.

4.1.1 INTRODUCCIÓN A JQUERY

La función más usada en jQuery es **$()**. 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).

- **$("#miIdentificador");** //Seleccionar un elemento con id=#mildentificador
- **$("a");**//Seleccionar todos los elementos <a> (enlaces) de un HTML
- **$("#miParrafo, a");**// se pueden seleccionar varios al mismo tiempo
- **$(".miClase");**//Seleccionar mediante el nombre de la clase CSS
- **$("a[rel]");** //Con un determinado atributo
- **$("input[type=radio]");** //Con un valor para un atributo. Por ejemplo todos los elementos <input> con atributo type = radio (es decir, que son radio-buttons)
- **$("input[type=radio]:checked");**//O incluso solo aquellos radio- buttons que estén seleccionados
- **$("/html/body//p");** //Usando XPath

4.1.2 MANEJO DE EVENTOS

- Además del evento ***ready ()*** usado 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:
  - Se le puede pasar una función que determine su comportamiento.

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

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

    $("p").click();

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

4.1.2 MANEJO DE EVENTOS

- **on() y off():** 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

```
$("p").on("click mouseenter mouseleave", function(e){
    if ($(this).css("color")!="rgb(255, 0, 0)")
        $(this).css("color", "rgb(255, 0, 0)");
    else
        $(this).css("color", "rgb(0, 0, 255)");
})
```

- 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 (antes se empleaban los métodos **bind** y **unbind**, ya obsoletos)

4.1.2 MANEJO DE EVENTOS

- Además de ***on(),*** existe el método contrario ***off(),*** que elimina un evento previamente asignado a un elemento o varios de una página. Si se utiliza ***off()*** 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").off();

- 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").off("click");

4.1.2 MANEJO DE EVENTOS

- 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 ***off()*** Para ello habría que colocar la función dentro de una variable, para poder referirse a ella como parámetro de ***off()***. El siguiente ejemplo muestra la definición de la función y cómo se desactiva con ***off()*** la atención al evento *onclic* de *p* al hacer clic sobre el botón con id="desactivar-evento".
- 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.

4.1.2 MANEJO DE EVENTOS

```
$(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").on("click mouseenter mouseleave",funcionqueManeja);

        $("#desactivar-evento").on("click", function(){
                $("p").off("click", funcionqueManeja );
        })
});
```

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 este punto 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.
- Tres ejemplos:
  - 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");
  - 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"
});
```

4.2.1 CAMBIO DE LAS PROPIEDADES CSS

- Otra opción es el empleo de los métodos addClass, removeClass o toggleClass
- Estos elementos permiten añadir dinámicamente una clase a una etiqueta de HTML y como consecuencia aplicar el estilo asociado a la misma.
- **addClass** y **removeClass** como su nombre indica permiten añadir o quitar la clase.
- Mientras que **toggleClass** añade la clase si no existe en la etiqueta y la elimina si ya está presente.
- Ejemplo. Las funciones addClass y removeClass tendrían los mismos parámetros:

```
<style>.highlight{background-color: yellow}</style>

$( "p" ).click(function() {
  $( this ).toggleClass( "highlight" );
});
```

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:

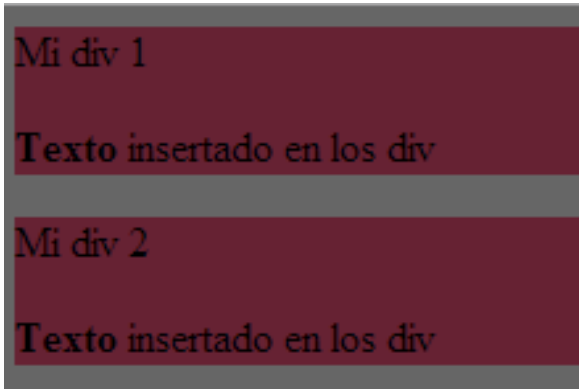
```
<html>
<head>
<title>Ejemplo uso append </title>
<script type="text/javascript" src="jquery.min.js"</script>
<script type="text/javascript">
$(document).ready(function() {
        $(".misdivs").on("click mouseenter mouseleave",
        function(e){
        $(".misdivs").append("<p><b>Texto</b>insertado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

4.2.2 AÑADIR NUEVOS ELEMENTOS

- 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.min.js"></script>
<script type="text/javascript">
        $(document).ready(function() {
        $(".misdivs").on("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>
```



4.2.3 AÑADIR 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:
  - 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("Some 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.
- El método **html**, cuya sintaxis es similar, si que interpretaría las etiquetas del ejemplo anterior.

$ ("p") .html("Some new text.");

4.2.4 OTROS MÉTODOS

- **.innerWidth() e .innerHeight():** para un elemento devuelve sus dimensiones internas (anchura y altura, respectivamente) contando el padding del elemento pero no el borde.
- **.outerWidth() e outerHeight():** devuelve las dimensiones externas (anchura y altura, respectivamente) del elemento contando el padding del elemento y su borde.
- **.offset():** indica la posición del elemento real, teniendo en cuenta los márgenes del elemento. Lo devuelve en un objeto con dos atributos top y left. http://api.jquery.com/offset
- En la API de Jquery se puede encontrar más sobre estos métodos: http://api.jquery.com/category/dimensions/

4.2.4 OTROS MÉTODOS

- **.is("selector")**: Permite comprobar si un elemento se corresponde con el selector que se pasa como parámetro. Ejemplo de uso: http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_traversing_is
- **$("selector").each():** Permite ejecutar una función para los elementos que se corresponden al selector. Ejemplo de uso: http://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_misc_each
- **$("selector").remove():** Borra los elementos que se corresponden al selector y sus hijos. Existe un método análogo denominado "empty" que solo borra los hijos.
- En el manual de W3Schools a la izquierda, puedes ver un menú lateral donde se indican los métodos más utilizados de jQuery.

4.3 EJECUCIÓN DE SECUENCIA 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.
- 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" } ) ;
$ ("fmisdivs").fadeIn (2000) ;
```

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

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

4.3 EJECUCIÓN DE SECUENCIA DE FUNCIONES

- 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 el apartado 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 () {      ->desaparece
      $ ("#micapa") .css ({ 'top' : 300, 'left' : 200 } ) ;
      $ ("#micapa") . fadeIn (1000) ; ->aparece
}) ;
```

4.3 EJECUCIÓN DE SECUENCIA DE FUNCIONES

- 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".

4.3 EJECUCIÓN DE SECUENCIA DE FUNCIONES

```
<!DOCTYPE html >

<html><head>
<script type="text/javascript" src="jquery.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>
        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, vamos a ver 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()** 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.

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:

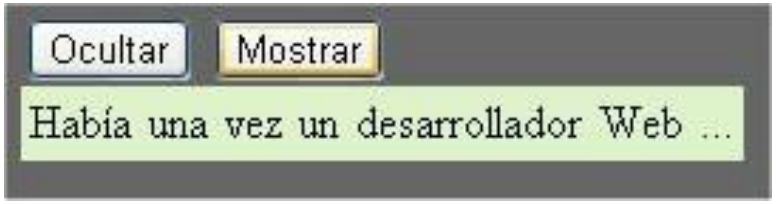
.show ([duración] [, easing] [, callback])

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

4.4.1 EFECTOS BÁSICOS

- **callback:** una función para llamar una vez finalizada la animación (tal y como se mostró antes).
- Todos los parámetros de este método son opcionales (por eso están entre [ ]).
- Un ejemplo para mostrar todos los párrafos lentamente sería:

*$("span").**show(500);***



4.4.1 EFECTOS BÁSICOS

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

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

*$("span:last-child").**hide("fast", ocultar);***

- Por otro lado, también está la función de usar **toggle().** Admite los mismos parámetros, pero alterna los efectos: se muestra si inicialmente está oculto y se oculta si está visible.

4.4.2 EFECTO *FADDING (OPACIDAD Y TRANSPARENCIAS)*

- ***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:

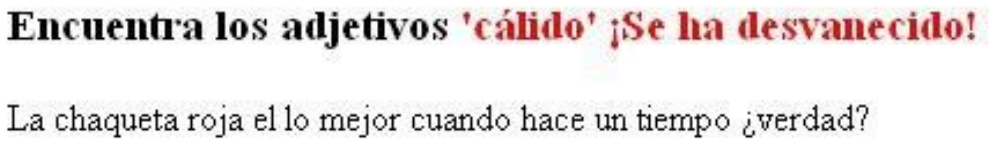
  .fadeIn ([duración] [, easing] [, callback])

  .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.

4.4.2 EFECTO *FADDING (OPACIDAD Y TRANSPARENCIAS)*

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



.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.

4.4.3 EFECTO *SLIDING (DESPLAZAMIENTO)*

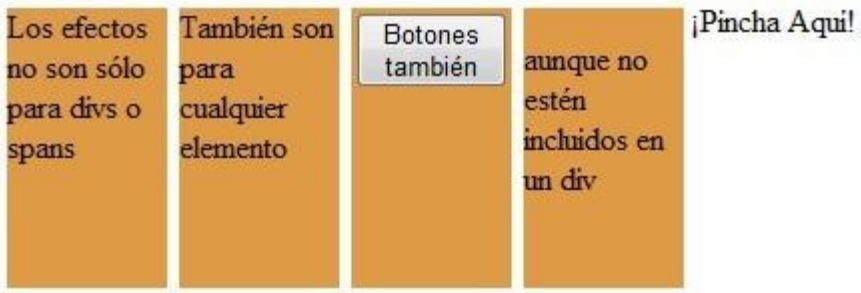
- Sliding son efectos de desplazamiento de elementos. Los *sliding* permiten hacer efectos de plegado y despliegue
  - *.slideUp ([duración] [, easing] [, callback])*
  - *.slideDown ([duración] [, easing] [, callback])*
  - *.slideToggle ([duración] [, easing] [, callback])*

- 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.

4.4.3 EFECTO *SLIDING (DESPLAZAMIENTO)*

- En el siguiente ejemplo 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>
<scripttype="text/javascript" src="jquery.js"></script>
<script type="text/ javascript" >
    $(document).ready (function () {
    $("span").click (function () {
    if ($("div:first").is(":hidden") ) {
        $("div").slideDown ("slow") ;
    } else {
        $("div").slideUp() ;
</script>
```



4.4.4 OTROS EFECTOS: ANIMATE

- El método **animate** crea una secuencia en un elemento según las propiedades CSS que se pongan en el parámetro *propiedades* (solamente sirven las propiedades con un valor numérico). Los parámetros opcionales *[duración], [easing] y [callback]* tienen el mismo significado que en los métodos vistos anteriormente.

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

Como se puede observar, las propiedades se expresan con formato JSON con pares clave-valor {width: "70%", opacity: 0.5…}

```
$("#left").click(function(){//Animación del botón izquierdo.
        $("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 );
        });
```

Pincha AQUÍ también

4.4.4 OTROS EFECTOS: ANIMATE

- También se pueden incluir los valores precedidos de los operadores "-=" o "+=".
- Esto sirve para sumar el valor especificado al actual. Es decir, en el ejemplo de debajo, en lugar de especificar 50px de valor final, se indica que la propiedad left debe restar 50px respecto a su valor inicial debido al operador "-="

```
$("#left").click(function(){//Animación del botón izquierdo.
    $(".block").animate({"left": "-=50px"}, "slow");
});
```

- El operador "+=" sería análogo, pero sumando el valor numérico que corresponda en cada caso.

4.4.4 OTROS EFECTOS: ANIMATE

- El método **animate** admite una sintaxis alternativa con un objeto llamado **opciones**

  .animate (propiedades, opciones)

- El parámetro opciones es un objeto con sintaxis similar a las propiedades y cabe destacar las siguientes opciones, de las que básicamente lo que se ha hecho ha sido desglosar la función callback en varias diferentes según el estado:
  - **duration** y **easing** son iguales que la sintaxis anterior, pero ahora se expresan en formato objeto.
  - **step (now, fx):** Función que se llama por cada propiedad del elemento de la animación. Es decir, si se modifican height y width se llama en cada píxel modificado en cada propiedad. El parámetro "now" permite consultar el valor actual y el objeto "fx" permite referenciar una serie de propiedades del objeto de la animación.
  - **progress (animacion, progreso, restante):** Función que se llama en cada milisegundo de la animación. El primer parámetro permite acceder al propio objeto de la animación, progreso es un valor de 0 a 1 según se haya completado y restante se refiere a los milisegundos que faltan.

4.4.4 OTROS EFECTOS: ANIMATE

- Otras propiedades del parámetro options de animate:
  - **complete:** Función que se llama cuando la animación ha finalizado. Parecido al callback, aunque se llama tanto si ha finalizado correctamente como con errores.
  - **done:** Función que solo se invoca cuando la animación se ha ejecutado correctamente
  - **fail:** Función que se invoca en caso de error.
  - **always:** Función que se invoca cuando la animación se completa, ya sea con error o no, o también si se para en medio de la ejecución.
  - **queue:** Permite indicar el nombre de una cola de efectos y emplear una alternativa (veremos esto en el apartado 4.4.6)

- Más información: https://api.jquery.com/animate/

4.4.4 OTROS EFECTOS: ANIMATE

- Ejemplo de sintaxis:

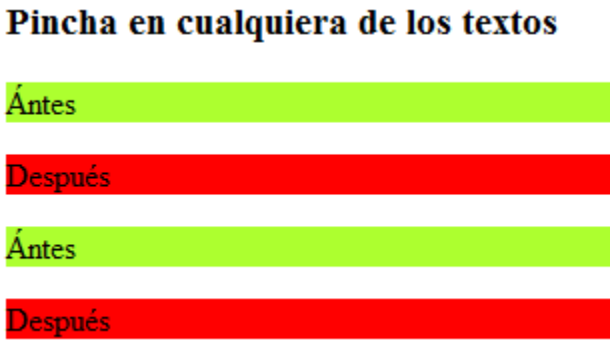
```
$("#idDiv").animate({left: "50px", top: "60px"}, {
   duration:5000,
   easing:"linear",
    queue: "fx",
   step:function(now,b){
       $("#Ptn").val("Current value " + now);
   },
   progress:function(a,prog,ms){
         $("#Ptc").val(ms);
         $("#Pcp").val(prog);
   },
   complete:function(){ alert("Completed"); }
});
```

4.4.4 OTROS EFECTOS: DELAY

.delay(duración [, NombreCola])

- Del método ***.delay()*** ya se comentó algo en el apartado 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 (se explicará esto más adelante)

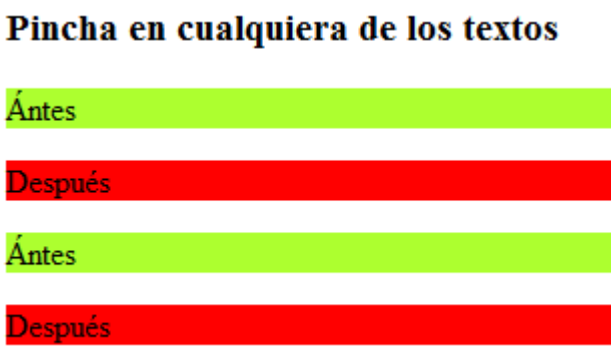
```
$(document).ready(function(){
    $("div").click(function() {
      $("div.antes").slideUp(300);
      $("div.antes").fadeIn(800);
      $("div.despues").slideUp(300);
      $("div.despues").delay(800);
      $("div.despues").fadeIn(800);
      });
  });
```



4.4.4 OTROS EFECTOS: DELAY

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

```
$(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);
    });
});
```



4.4.5 STOP Y FINISH

- Estas funciones permiten parar las animaciones que se ejecutan o están en cola de espera, modificando su estado en función de los parámetros que reciben

  La sintaxis es *$("selector").stop(bool_opcional1, bool_opcional2)*

  $("selector").finish()

  El primer parámetro de la función stop permite parar todas las animaciones que se están ejecutando o esperando

  El segundo parámetro de la función stop permite que cuando se para la animación el estilo sea automáticamente como quedaría al finalizar la misma. Por ejemplo, si una animación desplaza 50px en 10 segundos. Si la paramos a los 5 segundos, se desplazan automáticamente los 50px en lugar de dejar el elemento en un estado intermedio.

4.4.5 STOP Y FINISH

En resumen…

- **stop(false, false)** o sin parámetros, detiene solo la animación actual y la deja en el estado que se encuentra cuando se ha parado.
- **stop(true, false)** detiene todas las animaciones que se están ejecutando o están esperando y las deja en el estado actual.
- **stop(true, true)** detiene todas las animaciones que se están ejecutando o en cola de espera y deja las que se ejecutan en el estado final.
- **stop(false, true)** detiene la animación actual y la deja en el estado final.
- **finish()** detiene todas las animaciones que se están ejecutando o en cola de espera y deja todas en el estado final.

4.4.6 COLAS DE EFECTOS

Los efectos de jQuery se almacenan en una estructura similar a una cola de procesos.

Estas colas de procesos se pueden manipular manualmente mediante la función **queue,** cuyas variantes se indican a continuación.

- $(selector).queue()
- $(selector).queue(NombreDeLaCola)
- $(selector).queue(funcion)

Las dos primeras funciones nos devuelven la cola de efectos con todas las funciones por ejecutarse en formato de array.

La última permite añadir manualmente funciones a la cola de efectos.

4.4.6 COLAS DE EFECTOS

Imaginemos el caso en el que añadimos funciones manualmente como el ejemplo de debajo:

```
$("div")
        .animate({ left:"+=500px" }, 1000 )
        .animate({ top:"0px" }, 1000 )
        .queue(function() {
                $(this).toggleClass("red").dequeue();
        })
        .animate({ left:"50px", top:"150px" }, 1000 );
```

Con la función queue se ejecuta una función (que no es necesariamente un efecto) después del segundo método animate. Si no ejecutamos la función **dequeue** el código no continuará hasta el último animate. Esto ocurre, ya que si manipulamos las colas manualmente, la ejecución del resto de funciones también será manual

4.4.6 COLAS DE EFECTOS

Por tanto, la función **dequeue** permite ejecutar la siguiente función encolada en una cola de efectos, posteriormente dicha función se elimina de la cola. En resumen, empleamos dequeue para ejecutar la función posterior a las que se han invocado manualmente con queue.

Otra opción es no ejecutar más funciones de las que están esperando en la cola. Esto se consigue con **clearQueue.** Por ejemplo, $("selector").clearQueue();

Las funciones dequeue y clearQueue se pueden ejecutar sin parámetros (en este caso sobre la cola por defecto "fx") o con un parámetro que se refiere al nombre de la cola de efectos (si creamos alguna manualmente)

4.5 COMPORTAMIENTO INTERACTIVO

Una vez vistos los efectos más comunes usando jQuery, vamos a ver los diferentes eventos (ratón y teclado) que atiende jQuery para la interacción del usuario con objetos de la web y 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.

4.5.1 EVENTOS DE RATÓN

- **.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 *clickO* 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.

4.5.1 EVENTOS DE RATÓN

- **.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().*

4.5.1 EVENTOS DE RATÓN

- ***.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".

```
$("li#localizacion").hover(
        function () {
                $("span#opcion_loc").show();
        }, function () {
                $("span#opcion_loc").hide();
        });
```

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.

4.5.2 EVENTOS DE TECLADO

- **.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 ha 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 carácter introducido.

4.5.2 EVENTOS DE TECLADO

- 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.*

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

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.

4.5.3 EVENTOS DE VENTANA

```
<!DOCTYPE html>
 <html>
 <head>
 <style>
 div {overflow: scroll; width: 300px; height: 100px; font-family: Arial, Helvética, sans-serif; color: #888888; padding:
       7px  3px 0 4px; font-size: 12px; }
span {display:inline;}
</style> <script type="text/javascript" src="jquery.min.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","red"); }  else{
                  $('div') .css("background", "green") ; } });
        });
 </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.
       </div>
<span id="textoevento"> Se hace Scroll </span>
 <span id="repite"> PULSA Y lanzo el evento pero no hago el scroll (desplazamiento)> </span>
 </body> </html>
```

4.6 CONCLUSIONES

- En la web hay gran cantidad de código realizado por la comunidad jQuery que ofrece desarrollos de interfaz bastante completos: botones, menús, animaciones, etc. Se podrá interpretar esos ejemplos y modificarlos para poder incluirlos en sus páginas (si la licencia lo permite).
- 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.

4.6 CONCLUSIONES

- En la web hay gran cantidad de tutoriales y bibliografía relacionada con esta tecnología. Aquí se muestra solo algunos ejemplos en inglés:
  - **jQuery** API: toda la documentación on line sobre el API de jQuery con ejemplos. http://api.jquery.com
  - **W3Schools:** tutorial on line sobre las funciones jQuery. http://www.w3schools.com/jquery/default.asp