|  |
| --- |
| **Introducción** |

Una actividad muy frecuente en JavaScript es hacer aparecer y desaparecer elementos de la página.

Con el fin de utilizar estos efectos, los aplicamos a una selección de jQuery, como por ejemplo, una función. Por ejemplo, para ocultar todas las etiquetas que tengan la clase ***submenu***, escribimos lo siguiente:

*$('.submenu').hide();*

Las funciones que se usan para aplicar efectos pueden utilizar un parámetro de velocidad y una función ***callback***( (de función de devolución). El parámetro de velocidad hace referencia al tiempo necesario para que se complete la tarea, mientras que un ***callback*** es una función que se ejecutará cuando se haya completado el efecto.

Si deseamos asignar una velocidad a un efecto, debemos proporcionar una de las siguientes cadenas: ‘***fast’***, ‘***normal’*** o ‘***slow’***, o un número que represente la cantidad de milisegundos que tardará el efecto en completarse (1000 milisegundos es un segundo). Por ejemplo, la siguiente instrucción hace desaparecer poco a poco un objeto de la pantalla:

*$('.elemento').fadeOut('slow');*

Otra posibilidad es que el elemento tarde 3 segundos en desaparecer:

*$('.elemento').fadeOut(3000);*

Cuando utilizamos un efecto para que desaparezca un elemento de la pantalla, no lo eliminamos realmente de la página cargada en el navegador, ya que continúa existiendo en el modelo DOM y su código permanece en la memoria del navegador. Pero en el caso de que utilicemos el valor ***none***, desaparecerá de la pantalla; dejará de ser visible y dejará libre el espacio que ocupaba en la pantalla, por lo que podríamos ubicar otro elemento.

***Nota***:

Las palabras clave que determinan la velocidad de un efecto son *fast* (200), *normal* (400) y *slow* (600).

|  |
| --- |
| **Efectos** |

**Mostrar y ocultar elementos**

JQuery tiene tres funciones:

* ***Show()***: muestra un elemento oculto. Si no enviamos ningún parámetro de velocidad, el efecto es inmediato. En caso contrario, el elemento aparecerá por la esquina superior izquierda y desaparecerá por la inferior derecha.
* ***Hide()***: oculta un elemento visible. Si no enviamos ningún parámetro de velocidad, el efecto es inmediato. En caso contrario, el elemento se reducirá hasta desaparecer.
* ***Toggle():*** cambia el tipo de visibilidad del elemento. Si estuviese visible, lo ocultará. En caso contrario lo mostrará. Esta función es interesante cuando solo tengamos un control (botón) con el que mostraremos y ocultaremos el elemento.

**Mostrar y difuminar elementos**

JQuery tiene cuatro funciones para difuminar (opacidad) un elemento:

* ***fadeIn():*** muestra poco a poco un elemento oculto. En primer lugar, aparecerá el espacio del elemento (durante este proceso se podrían apartar otros elementos de la página para hacer sitio), y luego va apareciendo de forma gradual. Por defecto, el parámetro de velocidad es normal (400).
* ***fadeOut ():*** difumina poco a poco un elemento *visible* hasta hacerlo desaparecer. Por defecto, el parámetro de velocidad es normal (400).
* ***fadeToogle():*** combina los dos efectos anteriores. Por ejemplo, podemos utilizar esta función para que aparezca/desaparezca una aclaración para el usuario. Es el caso de que queramos que aparezca un apartado de Instrucciones; si tenemos un botón de Instrucciones, al hacer *click* sucesivamente sobre él, aparecería/desaparecería un elemento *div* con instrucciones. Este podría ser el código:

*$('#button').click(function() {*

*$('#instrucciones').fadeToogle(1000)*

*}); // fin del click*

* ***fadeTo():*** difumina una imagen hasta alcanzar un nivel de opacidad. Por ejemplo, podemos hacer que una imagen se vaya difuminando hasta que sea semitransparente. Es obligatorio especificar el parámetro de velocidad. Otro parámetro a aportar es el indicador de opacidad (entre 0 y 1). Ejemplo:

*$('p').fadeTo('normal',.75);*

Esta función modifica la opacidad de un elemento independientemente de si es visible o no. Por ejemplo, si aplicamos esta función a un elemento visible hasta alcanzar la opacidad del 50%, el elemento opaco se difuminará hasta convertirse en semitransparente. Pero si lo aplicamos a un objeto semitransparente, éste volverá a ser opaco.

Si difuminamos un elemento hasta llegar a una opacidad de 0, se ocultará completamente, pero respetará el espacio que ocupaba en la página.

Por otra parte, si difuminamos un elemento al 50% y posteriormente lo ocultamos, la página no utilizará el espacio que ocupaba. Y, si posteriormente mostramos el elemento, el navegador recordará su nivel de opacidad; lo mostrará, pero solo hasta el 50% de opacidad.

**Deslizamiento de elementos**

Podemos deslizar elementos por la pantalla. Si queremos hacerlo visible, lo deslizaremos de fuera a adentro de la pantallas. Si deseamos que desaparezca, haremos lo contrario.

Las funciones son muy similares a las de visualización de los elementos.

* ***slideDown():*** muestra un elemento oculto. Primero aparecerá la parte superior del objeto y posteriormente el resto. Si el elemento ya está visible, no tiene ningún efecto. Por defecto la velocidad es *normal*.
* ***slideUp():***oculta un elemento visible. Primero desaparecerá la parte superior del objeto y posteriormente el resto. Si el elemento ya está visible, no tiene ningún efecto. Por defecto la velocidad es *normal*.
* ***slideToogle():*** aplicará la función *slideDown()* si el elemento estuviese oculto y la función *slideUp()* en caso contrario.

|  |
| --- |
| Uso de CSS |
| Al ocultar un elemento de la página, normalmente el resto de los elementos ocupa el espacio que se ha dejado libre.  Del mismo modo, si hacemos aparecer un elemento, habrá que desplazar a los que ya están en la página con el fin de dejar espacio libre. En este caso, podemos usar CSS para establecer la posición absoluta fuera del flujo normal del contenido de la página. Es decir, podemos hacer que aparezca un elemento div, una imagen o un párrafo en la parte superior de la página, como si estuviese en su propia capa. A este fin, podemos usar la propiedad de posición CSS.  Con el fin de conseguir que un elemento aparezca al principio de la página, definimos su posición como un valor absoluto. Para especificar su ubicación, usamos las propiedades *left*, *right*, *top* y *bottom*. Por ejemplo, supongamos que tenemos una etiqueta <*div>* que contiene un formulario de registro. Por regla general, este formulario está oculto pero, cuando un usuario haga *click* sobre el enlace, el formulario aparecerá deslizándose por la parte superior de la página. Este sería el código que aplicaríamos al elemento *<div>:*  *#login*  *{*  *position: absolute;*  *left: 500px;*  *top: 0;*  *width: 400px;*  *}*  Este código ubica al elemento div al inicio de la ventana del navegador y a 500px del borde izquierdo. También podríamos usar la propiedad *right* para tomar como referencia el borde derecho, etc.  También podemos definir la posición de un elemento tomando como referencia otros objetos de la ventana. Es el caso de una ayuda emergente que aparece al colocar el cursor sobre una herramienta. Una palabra podría tener un signo de interrogación al lado (***?***); al hacer click sobre él podríamos abrir un cuadro de diálogo con información. En este caso no situaríamos el cuadro de diálogo tomando como referencia los bordes de la ventana, sino junto a la palabra. A este fin, tendremos que proporcionar una posición relativa al elemento que rodea a un objeto cuya posición será absoluta; por ejemplo:  *<span class="palabra">Beamont&eacute;s*  *<span class="definicion"></span>*  *Faccion de Navarra que era enemiga de los agramonteses*  *</span>*  Para que el cuadro de ayuda aparezca debajo de la palabra, tendremos que saber la posición relativa de la palabra, y luego utilizar un posicionamiento absoluto.  *.palabra {position: relative;}*  *.definicion {*  *position:absolute;*  *bottom:-20px;*  *left:0;*  *width:200px;*  *}* |

**Ejemplo: (1. form\_login.html)**

Queremos crear un panel deslizante que aparecerá y desaparecerá al hacer click con el ratón.

En primer lugar, debemos tener un párrafo que contenga el mensaje de Login. Para ello, debemos adjuntar un controlador de eventos *click* a dicho párrafo. JavaScript necesita eventos para ser interactivo. El usuario deberá interactuar con la selección del ***párrafo login*** para que suceda algo.

Podremos también cambiar la visibilidad del objeto.

**Procesos a desarrollar**

Creamos un archivo denominado **1.** ***form\_login.html***.

Creamos el siguiente script:

*<script>*

*$(document).ready(function() {*

*$('#open').click(function(****evt****) {*

*evt.preventDefault();*

*$('#login form').slideToggle(300);*

*$(this).toggleClass('close');*

*}); // final del click*

*});*

*</script>*

El texto ***Login*** se encuentra dentro de una etiqueta de párrafo que aparece rodeada por un vínculo:

*<a href="form.html"><p id="open">Login</p></a>*

El vínculo conducirá a los usuarios que hayan desactivado el JavaScript a otra página. Dado que ellos no verán el panel deslizante, tendrán una forma alternativa de acceder al formulario de registro.

El texto ***$('#open')*** se encarga de seleccionar el párrafo.

*$('#open').click(function(evt)*

Este código añade un controlador de eventos; esto permitirá que cuando un usuario haga *click* en el párrafo, suceda algo. En este ejemplo, al hacer un *click* sobre el párrafo aparecerá el formulario y cuando haga *click* una segunda vez, desaparecerá. Esto sucederá de forma continua. Para este cometido *JQuery* tiene tres funciones:

*toggle()*

*fadeToogle()*

*slideToggle()*

La diferencia entre ellas está en el aspecto que tendrá el efecto.

Añadimos un argumento “***evt***” al método *click* de la función anónima. A todos los elementos se les entrega automáticamente un objeto del evento que contiene métodos y propiedades. Necesitamos este objeto para decirle a *JQuery* que evite que el navegador siga el vínculo de la página.

***Nota:***

Una etiqueta *<p>* dentro de una etiqueta *<a>* no es válida en HTML4 pero sí en HTML5. Podemos utilizar este comando para envolver los elementos que se encuentren al nivel de bloque, como es el caso de las etiquetas *<p>,* *<h1>* y *<div>,* con vínculos.

***Evt.preventDefault();***

Este método evita que el navegador siga al vínculo que rodea al párrafo. La única finalidad del vínculo es ayudar a los usuarios que tengan desactivado JavaScript en su navegador, para que puedan acceder al formulario de registro. Pero, para los navegadores que tengan activado JavaScript, tendremos que decirle al navegador que permanezca en la página y que ejecute algo de código JavaScript.

*$('#loginform').slideToggle(300);*

Este código selecciona el formulario de registro, lo desliza por la ventana en el caso de que estuviese oculto, y luego lo desliza fuera de la vista.

*$(this).toggleClass('close');*

Esta función añade la clase especificada si no existiese, o la elimina en caso contrario. En este caso, hay un estilo de clase denominado ***.close*** en la hoja de estilos de la página. En este estilo añadimos una imagen de fondo para indicar que el usuario puede cerrar el panel deslizante.

***Nota:***

Cuando nos encontramos dentro de un controlador de eventos, podemos utilizar ***$(this)*** para referirnos al elemento que responde al evento. En este caso, ***$(this)*** hace referencia al párrafo sobre el que ha hecho click el usuario (***$(‘#open’)*** de la segunda línea del script).

Podemos probar a sustituir *slideToogle()* por *toogle()* o *fadeToogle()*.

***¿Qué podemos hacer si queremos aplicar dos efectos diferentes?***

Vamos a aplicar un efecto para hacer que el panel que aparezca en la pantalla se deslice desde la parte inferior de la página y otro para hacerlo desaparecer.

La función click() no permite utilizar dos acciones diferentes. Habrá que realizar un pequeño truco: cuando el usuario haga click en el vínculo ***Login***, tendremos que comprobar si el formulario está oculto. Si lo está, lo mostramos; en caso contrario lo ocultamos.

*Código:*

*$(document).ready(function() {*

*$('#open').click(function(evt) {*

*evt.preventDefault();*

*if ($('#login form').is(':hidden')) {*

*$('#login form').fadeIn(1200);*

*$(this).addClass('close');*

*} else {*

*$('#login form').slideUp(600);*

*$(this).removeClass('close');*

*}*

*}); // final del click*

*}); //*

El código anterior forma parte del ejemplo **1.** ***form\_login2.html***

|  |
| --- |
| **Animaciones** |

***Nota:*** Ejemplos: **2.** ***easing\_ejemplo1.html y 2. easing\_ejemplo2.html***

No estamos limitados a trabajar con las funciones jQuery. Con la función ***animate()*** podemos animar cualquier propiedad CSS que acepte valores numéricos tales como píxeles o porcentajes. Podemos animar, por ejemplo, el tamaño del texto, la posición de un elemento en la página, la opacidad de un objeto o el ancho de un borde.

***Nota:***

jQuery no puede animar el color. Sin embargo, ***jQuery UI*** tiene varios efectos animados adicionales, entre los cuáles se incluye la capacidad de animar el color.

Para utilizar esta función, entregaremos un objeto que contendrá una lista con las propiedades CSS que queramos modificar, y los valores que aplicaremos a la animación. Imaginemos, por ejemplo, que queremos animar un elemento moviéndolo 700 píxeles desde el margen izquierdo de la página, cambiar su opacidad al 50% y aumentar el tamaño de la fuente a 24 píxeles. Este código crea un objeto con estas propiedades y valores:

*{*

*left: '700px',*

*opacity: .5,*

*fontSize: '24px'*

*}*

Si quisiéramos utilizar los nombres reales de las propiedades CSS deberemos escribirlos entre comillas:

*{*

*'font-size': '26px';*

*'border-left-width': '6%'*

*}*

Supongamos que queremos utilizar estos parámetros para animar un elemento cuyo ID sea *message*. Podríamos utilizar una función *animate()* como ésta:

*$('message').animate(*

*{*

*left: '700px',*

*opacity: .5,*

*fontSize: '26px'*

*},*

*1500*

*}*

La función ***animate()*** puede utilizar varios argumentos. El primero es un literal de un objeto que contendrás las propiedades CSS que queramos animar. El segundo es la duración de la animación, medida en milisegundos. En el código anterior la animación duraba 1500 milisegundos o, lo que es lo mismo, 1.5 segundos.

***Nota:***

Para animar la posición de un elemento utilizando las propiedades CSS *left*, *right*, *top* y *bottom*, tendremos que definir la propiedad CSS de la posición de un elemento, utilizando una referencia absoluta o relativa. Éstas son las dos únicas propiedades de posición que permiten que les asignemos valores. Para definir una posición relativa al valor actual, podemos utilizar ***+=*** o ***-=*** como parte de las opciones de la animación. Por ejemplo, supongamos que queremos animar un elemento moviéndolo 50 píxeles a la derecha cada vez que hagamos *click* sobre él. Este puede ser el código:

*$('#moveIt').click(function() {*

*$(this).animate(*

*{*

*left:'+=50px'*

*},*

*1000; // fin de animate*

*}); // fin de click*

**Easing**

Las funciones para efectos de jQuery (*slideUp()*, *fadeIn()*, etc) y la función *animate()* aceptan otro argumento que controlará la velocidad a la que se desarrollará la animación: ***easing***. Hace referencia a la velocidad a la que tienen lugar los diferentes puntos de animación. Por ejemplo, al mover el elemento por la página, podríamos hacer que comenzase lentamente, que luego se acelerase, y que finalizase lentamente.

jQuery incluye solo dos métodos para hacer easing: ***swing*** y ***linear***. El método *linear* proporciona una animación continua donde todos los pasos serán iguales (por ejemplo, si estamos desplazando un elemento por la pantalla, en todos los pasos se avanzará la misma distancia). El método *swing* es más dinámico ya que la animación empieza rápido y la velocidad disminuye poco a poco. Es el método predeterminado.

El método *easing* es el segundo argumento de cualquier efecto jQuery. Por eso, para hacer que un elemento se deslice hacia arriba de forma lineal, utilizaremos este código:

*$('#element').slideUp(1000,'linear');*

Con la función *animate()*, el método *easing* es el tercer argumento que aparecerá después del objeto que contiene las propiedades CSS que queremos animar y de la velocidad global de la animación. Por ejemplo, para utilizar el método *easing* *linear* con la animación que vimos anteriormente, escribiríamos lo siguiente:

*$('#message').animate(*

*{*

*left: '700px',*

*opacity: .5,*

*fontSize: '26px'*

*},*

*1500,*

*'linear'*

*);*

No estamos limitados a usar únicamente los dos métodos *easing* de jQuery. La librería ***jQuery* *UI*** incluye otros métodos *easing* adicionales. Las veremos mejor en otro Sitio que desarrollaremos. Para usar jQueryUI (un archivo externo de JavaScript), tendremos que adjuntar el archivo a nuestra página después del código que lo vincula con la librería jQuery. Luego, podremos utilizar cualquiera de los métodos *easing* disponibles (la lista completa se encuentra en <http://api.jqueryui.com/easings/>). Por ejemplo, supongamos que queremos incrementar el tamaño de una etiqueta *div* cuando el usuario haga click sobre ella, y que aplicaremos el método *easingInBounce* para que la animación sea más interesante. Suponiendo que el identificador del elemento div será *animate*, nuestro código tendría este aspecto:

1. *<script src="\_js/jquery.min.js"></script>*
2. *<script src="\_js/jquery-ui.min.js"></script>*
3. *<script>*
4. *$(document).ready(function() {*
5. *$('animate').click(function() {*
6. *$(this).animate(*
7. *{*
8. *width: '500px',*
9. *height: '500px'*
10. *},*
11. *1000,*
12. *'easeInBounce'); // fin de animate*
13. *} // fin de click*
14. *} // fin de ready*

*(15)</script>*

Las líneas 1 y 2 cargan *jQuery* y *jQuery* *UI*. La línea 5 se encarga de añadir un controlador del evento click a *div*. El núcleo de la acción se encuentra entre las líneas 6 y 12.cuando nos encontramos dentro de un evento, *$(this)* hace referencia al elemento que responde a dicho evento. En este caso es la etiqueta *<div>*. Al hacer click en el elemento *<div>*, lo animaremos modificando su ancho y su alto (líneas 8 y 9). La línea 11 se encarga de que la animación dure un segundo y la línea 12 indica que se utilizará el método *easyInBounce* (podríamos utilizar cualquier otro método *easing*, como *easeInOutSine*, *easeInCubic*, etc) 🡪 *Consultar en Web*.

|  |
| --- |
| **Acciones después de completar un efecto** |

***Nota***: Ejemplos **3.** ***callback.html*** y **4.** ***multiples\_callback-html***.

Podemos desear hacer algo después de completar un efecto. Imaginemos que queremos que aparezca un título después de que se termine de difuminar una foto. Normalmente, los efectos no se llevan a cabo uno tras otro, sino cuando se les llama. Por este motivo, si nuestro script tiene una línea de código que se encarga de difuminar la imagen, y otra que se encarga de que aparezca el título, éste aparecerá antes de que termine de desaparecer la foto.

Para lograrlo, tenemos que utilizar las funciones ***callback()***. Se trata de funciones que solo se ejecutarán una vez completado el efecto. Podemos pasar estas funciones como segundo parámetro a buena parte de los efectos (será el tercer argumento para la función *fadeTo()*).

Supongamos que tenemos una animación de una página cuyo identificador es foto, y un párrafo justamente debajo con el identificador *caption*. Vamos a difuminar la imagen hasta que desaparezca y luego utilizaremos una función *callback()* para mostrar el párrafo. Lo haríamos así:

*$('#foto').fadeIn(1000, function() {*

*$('#caption').fadeIn(1000);*

*});*

Si queremos ejecutar la función al cargar la página, tendremos que comenzar por ocultar la foto y el párrafo. A continuación, aplicaremos el efecto *fadeIn*:

*$('#foto, #caption').hide();*

*$('#foto').fadeIn(1000, function() {*

*$('#caption').fadeIn(1000);*

*});*

Si usamos la función *animate()*, la función *callback()* debería aparecer después de cualquier parámetro: el objeto que contiene las propiedades CSS que vamos a animar, la duración de la animación y el método *easing*. El último parámetro es opcional por lo que podemos limitarnos a pasar la función *animate()*, la lista de propiedades, la duración y la función *callback()*.

Por ejemplo, vamos a suponer que no solo queremos difuminar la foto hasta hacerla desaparecer sino también aplicar un efecto zoom, es decir, incrementar su ancho y su alto desde 0 hasta que ocupen toda la pantalla (**3.** ***callback.html***):

*<script>*

*$(document).ready(function() {*

*$('#photo').width(0).height(0).css('opacity',0);*

*$('#caption').hide();*

*$('#photo').animate(*

*{*

*width: '400px',*

*height: '300px',*

*opacity: 1*

*},*

*1000,*

***function() {***

***$('#caption').fadeIn(1000);***

***}***

***); // find de animate***

*});*

*</script>*

La primera línea define el alto, ancho y opacidad de la foto; al ser sus valores 0, se oculta la foto y queda lista para iniciar la animación. La función *animate()* define la animación. De hecho, la función *callback()* sucede dentro de ella (área resaltada en negrita).

Las funciones *callback*() pueden complicarse al intentar animar diferentes elementos en una fila. Por ejemplo, para conseguir que una imagen se mueva hasta el centro de la pantalla, que a continuación aparezca progresivamente un párrafo y que, por último, tanto la imagen como el párrafo se difuminen hasta desaparecer, necesitaremos entregar una función *callback()* a otra función *callback()* (**4.** ***multiples\_callback.html***):

*<script>*

*$(document).ready(function() {*

*$('#photo').animate(*

*{*

*left: '+=400px',*

*},*

*1000,*

*function() { // primera llamada a la funcion callback*

*$('#caption').fadeIn(1000,*

*function() { // segunda llamada a la funcion callback*

*$('#photo, #caption').fadeOut(1000);*

*} // fin del segundo callback*

*); // fin de fadeIn*

*} // fin de la primera llamada a la funcion callback*

*); // fin de animate*

*});*

*</script>*

Pero en el caso de que quisiéramos añadir animaciones adicionales al mismo elemento de la página, no haría falta recurrir a las funciones *callback()*. Vamos a suponer que queremos mover una foto por la pantalla y luego difuminarla hasta que desaparezca. Bastaría solo con usar la función *animate()* para conseguir los dos efectos:

*<script>*

*$(document).ready(function() {*

*$('#photo').animate(*

*{*

*left: '+=400px',*

*},*

*1000,*

*); // fin de animate*

*});*

*$('#photo').fadeOut(4000);*

*</script>*

En este caso, aunque el navegador va a ejecutar el código inmediatamente, jQuery colocará los efectos en cola, de forma que ejecutará la animación en primer lugar y luego la función *fadeOut()*; también podríamos escribir el código anterior de este modo:

*<script>*

*$(document).ready(function() {*

*$('#photo').animate(*

*{*

*left: '+=400px',*

*},*

*1000).fadeOut(4000)*

*); // fin de animate*

*});*

*</script>*

Y en el caso de que queramos que la foto aparezca progresivamente, se difumine hasta desaparecer y que volviese a aparecer de forma progresiva, podemos usar el encadenamiento:

*$('#photo').fadeIn(2000).fadeOut(4000).fadeIn(500);*

Existe una función denominada ***delay()*** que nos ayuda a encadenar efectos. Esta función espera un número determinado de milisegundos antes de iniciar el siguiente efecto de la cola. Por ejemplo, vamos a suponer que queremos mostrar una imagen progresivamente, esperar 8 segundos y luego difuminarla hasta que desaparezca:

*$('#photo').fadeIn(2000).delay(8000).fadeOut(400);*

|  |
| --- |
| **Panel animado** |

***Nota***: Ejemplo **5.** ***animate.html***.

Vamos a mover un elemento *div* que se encuentre oculto en la parte izquierda de la pantalla, hasta visualizarla dentro de la zona de trabajo. Para ubicar el elemento *div* usaremos el posicionamiento absoluto. Cuando el usuario coloque el ratón sobre la parte visible del borde del elemento *div*, éste se desplazará hacia la derecha. El proceso es:

1. Seleccionamos la etiqueta *<div>*
2. Adjuntamos un evento *hover*. Este evento es una función de JQuery. Permite llevar a cabo un conjunto de acciones cuando el usuario coloque el cursor del ratón sobre un elemento, y luego otro grupo de acciones cuando el usuario lo retire.
3. Añadimos la función *animate* para el evento *mouseEnter*. Cuando el usuario coloque el cursor sobre el elemento *div*, lo animaremos para que aparezca por la parte izquierda de la ventana del navegador. Además animaremos el color de fondo del *div*.
4. Añadimos otra función *animate* para el evento *mouseLeave*. Cuando el cursor del ratón deje de estar sobre el elemento *div*, lo devolveremos a su posición original y utilizaremos el primer color de fondo.

Explicación del código:

1. En la función *$(document).ready(function() {*

*$('#tablero').hover();*

*$('#tablero')* selecciona la etiqueta *<div>* cuyo identificador es tablero. La función *hover()* trabaja con dos argumentos (dos funciones) que describen qué ocurrirá cuando el usuario coloque el cursor del ratón sobre el elemento *div*, y cuando lo retire.

Dentro de la primera función anónima generamos el siguiente código:

*function() {*

*$(this).animate(*

***{***

***left: '0',***

***backgroundColor: 'rgb(25,55,95)'***

***}****,*

*500,*

*'easeInSine'*

*); // fin de animate*

*}*

El primer argumento de la función *animate()* es un objeto que contiene las propiedades CSS. En este caso, el elemento div se encuentra a -92px del borde izquierdo de la ventana, por lo que al situar este valor a 0, colocaremos el objeto dentro de la ventana, pegado a su borde izquierdo. Hacemos lo mismo con el color de fondo.

El segundo argumento (500), indica el número de milisegundos que dura la animación.

El tercer argumento *'easeInSine'*, indica que el movimiento debe de hacerse poco a poco, acelerándose con el paso del tiempo.

1. Dentro de la segunda función anónima generamos el siguiente código:

*function() {*

*$(this).animate(*

*{*

*left: '-92px',*

*backgroundColor: 'rgb(255,40,40)'*

*},*

*1500,*

*'easeOutBounce'*

*); // fin de animate*

Este código invierte la primera animación. En este caso utilizamos un tiempo diferente y un método *easing* diferente.

Si hiciésemos la prueba dejando el código así, el elemento *div* aparecerá en la pantalla y luego desaparecerá. Sin embargo, si repetimos este movimiento muchas veces y de forma muy rápida, observaremos un comportamiento extraño: el elemento *div* seguirá moviéndose incluso después de que hayamos detenido el ratón. El motivo está en la forma en que jQuery crea una cola con las animaciones que debe aplicar a un elemento. Lo que ocurre en nuestro caso es que, cada vez que coloquemos o retiremos el ratón del elemento *div*, se añadirá una animación a la cola ya existente. Por eso, al repetir esta acción muchas veces con un movimiento muy rápido, estaremos añadiendo muchas animaciones a dicha cola. El resultado es que jQuery seguirá aplicando animaciones incluso después de haber detenido el ratón. La solución a este problema es detener todas las animaciones incluso después de haber detenido el ratón. En otras palabras, al colocar el cursor sobre el elemento *div*, jQuery iniciará la animación que mostrará dicho objeto en la pantalla. Pero si retiramos el cursor, jQuery debería detener la animación en curso y proceder con la nueva animación. Para conseguirlo, usaremos la función *stop()*.

*$(document).ready(function() {*

*$('#tablero').hover(*

*function() {*

***$(this).stop().animate****(*

*{*

*left: '0',*

*backgroundColor: 'rgb(25,55,95)'*

*},*

*500,*

*'easeInSine'*

*); // fin de animate*

*},*

*function() {*

***$(this).stop().animate(***

*{*

*left: '-92px',*

*backgroundColor: 'rgb(255,40,40)'*

*},*

*1500,*

*'easeOutBounce'*

*); // fin de animate*

*}*

*); // fin de hover*

*}); // fin de ready*

La función ***stop()*** se encarga de detener la animación en curso antes de que *div* inicie una nueva, evitando así que se acumulen animaciones en la cola de jQuery.

|  |
| --- |
| **Animaciones y transiciones de CSS3** |

***Ejemplo***: **6.** ***jquery-trigger-css-transition.html***.

Existen animaciones que se pueden llevar a cabo a través de las propiedades CSS, pero no todos los navegadores lo permiten. Por ejemplo, no podemos animar la propiedad Font-family para crear una transición de una fuente de texto a otra. Pero podemos utilizar muchas propiedades CSS para crear transiciones y animaciones; en esta dirección se encuentra la lista completa: <https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_animated_properties>.

Sin embargo, aunque queramos trabajar con las animaciones y transiciones CSS, JQuery sigue siendo un gran punto de apoyo. A diferencia de JavaScript, CSS no es capaz de trabajar con eventos. Existe una **pseudoclase**, *:hover*, que permite aplicar un estilo a *mouseEnter*, y recuperar el estilo original con *mouseLeave*. La pesudoclase *:active* puede simular un click de ratón. Pero no existen clases equivalentes en CSS para eventos como doble click, el desplazamiento o las pulsaciones de teclado. Es decir, para que se inicie una animación cuando el usuario escribe un texto en un campo o haga doble click sobre un botón, no podremos utilizar solo CSS. Además, trabajando únicamente con CSS, no hay forma de iniciar una animación sobre un elemento cuando el usuario interactúe con otro objeto de la página; por ejemplo, cuando haga click sobre un botón que permite mostrar un elemento *<div>* en otra parte de la pantalla.

**jQuery y las transiciones**

Una transición CSS anima un cambio de una propiedad CSS. Una forma simple de hacerlo es aplicar un nuevo estilo al elemento y animar el cambio. Por ejemplo, podemos crear un estilo de clase para un botón (por ejemplo *.button*) que tuviese un color verde. Con una pseudoclase *:hover(.button:hover)* podríamos cambiar el fondo a azul. Al añadir una propiedad de transición al estilo *.button*, estaremos pidiéndole al navegador que anime el cambio de color del fondo, del verde al azul cuando el visitante coloque el cursor sobre dicho botón y que invierta la animación cuando el cursor deje de estar encima del botón.

Podríamos añadir una transición para animar el cambio de una propiedad CSS a otra. Por ejemplo, supongamos que queremos utilizar transiciones CSS para difuminar todas las imágenes de la pantalla hasta que desaparezcan. Comenzamos asignando un estilo a las imágenes:

*img {*

*opacity: 1;*

*}*

Esta propiedad controla la transparencia de un elemento. Un valor 1 implica que el objeto es completamente opaco, mientras que el valor 0 indica que es completamente transparente. A continuación, podemos crear una clase CSS que asigne el valor 0 a la opacidad:

*Img.faded {*

*opacity: 0;*

*}*

Para añadir una animación entre los estilos utilizamos la propiedad *transition* de CSS. Debemos añadir esta propiedad al estilo original (el que se aplica en primer lugar al elemento de la página). En este caso, se trata del estilo *img*. Además, para asegurarnos de que funcione en todos los navegadores compatibles con las transiciones de CSS, utilizaremos los prefijos de cada fabricante:

*img {*

*opacity: 1;*

*-webkit-transition: opacity 1s;*

*-moz-transition: opacity 1s;*

*-o-transition: opacity 1s;*

*transition: opacity 1s;*

En este ejemplo especificamos que deseamos animar cualquier cambio que se lleve a cabo sobre *opacity*. También queremos que la última animación dure un segundo.

Los prefijos permiten que el navegador compruebe que puede ejecutar correctamente dicha propiedad antes de poner en marcha todos los cambios asociados a ella. Cuando el navegador ve que la propiedad CSS se puede utilizar de forma segura, publican nuevas versiones de sus navegadores donde ya no hace falta utilizar los prefijos y se puede emplear el nombre estándar de la propiedad CSS, como por ejemplo *–transition* en lugar de *–webkit-transition*.

Después de la transición, añadimos la clase ***faded*** a la imagen. Se encargará de pasar de 100x100 de opacidad a una opacidad del 0 por ciento en el transcurso de un segundo. Esto equivale a utilizar la función *fadeout()* de JQuery durante un segundo. La clave es aplicar esta nueva clase (***faded***) a la imagen. Si quisiéramos aplicar el estilo cuando el visitante haga click sobre una imagen, utilizaremos la función de eventos click():

*$('img').click(function() {*

*$(this).AddClass('faded');*

De este modo, cuando el usuario haga click sobre una imagen, JQuery le añadirá la clase. El navegador se encargará de realizar el trabajo duro animando el cambio de opacidad. Si quisiéramos que la imagen volviese a su estado original de forma paulatina cuando el usuario hiciese click de nuevo sobre ella, podríamos utilizar la función *toogleClass()*:

*$('img').click(function(evt) {*

*$(this).toggleClass('faded');*

La función *toogleClass* añadirá una clase siempre que no se estuviese aplicando ya el elemento, y la eliminará en caso contrario. Ya que CSS no dispone de un selector para utilizarlo cuando el usuario haga click sobre un elemento de la página, utilizamos JQuery porque ofrece una forma sencilla de iniciar las transiciones CSS.

***Observación***: al modificar la opacidad de un elemento, no se llega a eliminar el elemento de la página. Sigue ocupando el espacio pero no es posible. Al hacer click sobre el hueco, la imagen vuelve a aparecer. Sin embargo, si se ocultase la imagen realmente con *display:hidden*, se eliminarían de la página y no se podría volver a hacer click una segunda vez sobre ellos.

**jQuery y las animaciones**

***Ejemplos: 7. jquery-trigger-css-animation1.html y 8. jquery-trigger-css-animation2.html***

Las animaciones CSS ofrecen un mayor control que las transiciones. Con una animación CSS definimos una serie de pasos intermedios que determinarán las propiedades CSS que se utilizarán en diferentes puntos de la animación. Por ejemplo, podemos hacer que un botón cambie de color verde a rojo, luego al naranja y por último al verde; o por ejemplo, mover un elemento *<div>* desde la parte superior de la pantalla hasta la parte inferior, etc. En otras palabras, mientras que las transiciones CSS nos permiten especificar dos estilos (inicio y final) del elemento, las animaciones CSS permiten especificar estilos intermedios por los que pasará la animación del objeto hasta llegar al estado final.

Veamos un ejemplo: supongamos que queremos que un elemento *<div>* cambie de color y se haga más ancho cuando el usuario haga click sobre un botón. Podríamos utilizar este efecto también para mostrar un texto oculto que se encontrase dentro de un cuadro.

El primer paso es crear la animación:

*@keyframes barraProgresoCrece{*

*0% {*

*width: 0%;*

*background-color: red;*

*}*

*50% {*

*background-color: blue;*

*}*

*100% {*

*width:88%;*

*background-color: green;*

*}*

*}*

Este código define una animación a la que denomina *barraProgresoCrece* y una serie de pasos intermedios. Cada paso puede identificar una o más propiedades CSS que se pueden modificar durante la animación. En el código anterior indicamos que el elemento debe tener un ancho de 0% y que debe utilizar el color rojo como fondo. El elemento cambia de color pasando del rojo al azul y posteriormente al verde. El valor porcentual de cada paso indica cuando se debe aplicar la propiedad CSS. Si esta animación durase 10 segundos, cuando han transcurrido 0 segundos, el color de fondo será rojo, y el elemento estará al 0% de ancho. Cuando transcurra el 50% de los 10 segundos (5 segundos), el color será azul. Cuando la animación se encuentre al 100% del tiempo (10 segundos), el color del elemento será verde y tendrá un ancho del 90%.

En este caso indicamos el ancho al inicio y al final, por lo que la anchura cambiará del 0% al 90% durante la animación.

Una vez creados los pasos intermedios, podremos añadir esta animación a todos los elementos que queramos. Por ejemplo, supongamos que tenemos el siguiente código:

*<div class="progressBar">*

Podríamos añadir esta animación al elemento *div*:

*.progressBar{*

*animation-name: barraProgresoCrece;*

*animation-duration: 10s;*

*animation-fill-mode: forwards;*

*}*

Este código aplicará la animación al elemento y la ejecutará durante 10 segundos. La última línea del CSS permite que cuando finalice la animación, el elemento tanga las propiedades definidas en el último paso. Sin estos parámetros, el elemento recuperaría los estilos que tenía antes de ejecutarse la animación.

En el código anterior, la animación se ejecutará de inmediato, en el momento en que se cargue la página web. Como queremos iniciar la animación cuando el usuario haga click en el botón, deberemos detener temporalmente la animación (detenerla hasta que la ponga en marcha el usuario) utilizando la propiedad de las animaciones: *animation-play-state*. Con el fin de evitar que una animación se inicie de forma inmediata, añadiremos la propiedad al estilo utilizando el valor *paused*:

*.progressBar{*

*animation-name: barraProgresoCrece;*

*animation-duration: 10s;*

*animation-fill-mode: forwards;*

*animation-play-state: paused;*

*}*

Nos falta solo añadir jQuery con el fin de iniciar la animación. Solo tenemos que modificar la propiedad *animation-play-state* y asignarle el valor *running* para que la inicie. Lo haremos con la función *css()*. Por ejemplo, podemos crear un botón con el identificador *start*. Cuando el usuario haga click en el botón, se iniciará la animación; si tenemos un botón de pausa, detendríamos la animación. Este es el código:

*$(document).ready(function() {*

*$('#play').click(function(evt) {*

*$('#pause').removeClass('active');*

*$(this).addClass('active');*

***$('.progressBar').css('animation-play-state','running');***

*}); // fin de click*

*$('#pause').click(function(evt) {*

*$('#play').removeClass('active');*

*$(this).addClass('active');*

***$('.progressBar').css('animation-play-state','paused');***

*}); // fin de click*

*});*

Otra forma de conseguir lo mismo, es crear los pasos intermedios y crear un estilo para la clase independiente donde recibamos todas las propiedades de la animación (***jquery-trigger-css-animation2.html***):

*.animate {*

*-webkit-animation-name: barraProgresoCrece;*

*-webkit-animation-duration: 3s;*

*-webkit-animation-fill-mode: forwards;*

*animation-name: barraProgresoCrece;*

*animation-duration: 3s;*

*animation-fill-mode: forwards;*

*}*

Al terminar la carga de la página, el elemento que queremos animar no tendrá aplicado aún el estilo de clase *.animate*. Por dicha razón, no se iniciará la animación. Este es el efecto que queremos. Posteriormente, podemos saltarnos la animación *animation-play-state*.

*$('#start').click(function(){*

*$('.progressBar').addClass('animate');*

*});*

**Nota**: las animaciones tienen el inconveniente de las versiones de navegadores que las soportan. Por ejemplo, *internet Explorer 9* y las versiones anteriores no las soportan.

Por otra parte, el control de las animaciones con CSS, no es tan avanzado y sencillo como lo es con jQuery.