Unidad 6. Integración de contenido interactivo

Tabla de contenido

INTRODUCCIÓN A JQUERY

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.

Vincular al archivo de jQuery en un servidor CDN

Tanto Microsoft, jQuery como Google permiten incluir el archivo de jQuery en una de sus páginas web utilizando sus servidores. Por ejemplo, para vincular a la versión 3.1.1 de jQuery usando de Microsoft CDN, deberá añadir esta línea de código en la sección <head> de su página web (justo antes del cierre </ head>), así:

```
<head>

<script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.1.1.min.js"></script>

</head>
```

El listado con todas las versiones disponibles se encuentra en:
https://www.asp.net/ajax/cdn#Using_jQuery_from_the_CDN_21

Utilizando el jQuery CDN, tendrá que utilizar este código:

```
<Script src = "http://code.jquery.com/jquery-1.11.0.min.js"></ script>
```

Accediendo a la página http://code.jquery.com directamente tenemos el enlace a todas las versiones de jQuery disponibles, solo tenemos que insertar el enlace al archivo.

Y el código usando el CDN de Google se parece a esto:

```
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
</head>
```

La página principal de las librerías de google es https://developers.google.com/speed/libraries/

Una gran ventaja de usar el jQuery alojado de Google o Microsoft:

Muchos usuarios ya han descargado jQuery de Google o Microsoft cuando visitan otro sitio. Como resultado, se cargará desde la caché cuando visitan su sitio, lo que lleva a un tiempo de carga más rápido. Además, la mayoría de CDN se asegurará de que una vez que un usuario solicite un archivo de él, se servirá desde el servidor más cercano a ellos, lo que también conduce a un tiempo de carga más rápido.

Usar el archivo jQuery descargado del sitio oficial

La segunda forma consiste en descargar el archivo directamente de http://jquery.com/dowload/ a nuestro servidor e incluirlo con una marca script en la página web de la siguiente manera:

```
<head>
<script src="jquery-3.1.1.min.js"></script>
</head>
```

AÑADIR JQUERY A UNA PÁGINA

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

```
$( 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. 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.

MODIFICAR PÁGINAS WEB

Principalmente vamos a usar JavaScript para manipular una página web mediante la adición de nuevos contenidos, el cambio del código HTML de la página, o aplicar CSS a un elemento. Cada vez que se cambia el contenido, HTML o CSS en una página-si se va a añadir una barra completa con menús emergentes de navegación, la creación de una presentación de diapositivas impulsada JavaScript o simplemente hacer una diapositiva titular a la vista -Tendrás que realizar dos pasos principales:

1. Seleccione un elemento en una página.

Un elemento es cualquier etiqueta existente, y antes de que pueda hacer nada con ese elemento, es necesario seleccionarlo usando JavaScript.

2. Hacer algo con el elemento.

- Cambiar una propiedad del elemento.
- Añadir nuevo contenido.
- Eliminar el elemento.
- Extraer información del elemento.
- Añadir / eliminar un atributo de clase.

Muchas veces, se harán varias de las cosas mencionadas anteriormente, al mismo tiempo. Por ejemplo, digamos que usted quiere asegurarse de que el visitante no se olvide de escribir su dirección de correo electrónico en un campo de formulario. Si intenta enviar el formulario sin su dirección de correo electrónico, se le puede notificar. Esta tarea podría implicar, primero averiguar si no se ha escrito nada en el campo de texto (la información se extrae del elemento), imprimir un mensaje de error (añadiendo nuevos contenidos) si no lo hace, y destacar el campo del formulario (mediante la adición de una clase al campo de texto).

EL MODELO DE OBJETOS

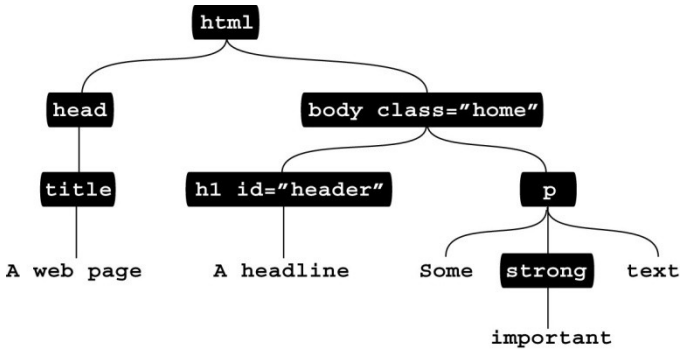
El DOM proporciona la información a JavaScript para que pueda comunicarse con los elementos de la página web. El DOM en sí no es en realidad JavaScript, es un estándar de la World Wide Web Consortium (W3C) que la mayoría de los fabricantes de navegadores han adoptado y añadido a sus navegadores.

Para ver cómo funciona el DOM, eche un vistazo a esta página web muy sencilla:

```
<! DOCTYPE HTML>
<html>
<head>
  <charset meta = "UTF-8">
  <title>A web page</ title>
</ head>
<body class = "home">
  <id h1 = "header"> A headline </ h1>
  <p>Some<strong>important</ strong>text</ p>
</ body>
```

```
</ html>
```

Puede representar la relación entre las etiquetas con una especie de árbol. La etiqueta <html> es la "raíz" del árbol-como el tatara-tatara-tatara abuelo de todas las otras etiquetas en la página, mientras otras etiquetas representan diferentes "ramas" del árbol de la familia; por ejemplo, los <head> y <body>, que contienen cada uno su propio conjunto de etiquetas.



LOS SELECTORES BÁSICOS

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). Por ejemplo:

- Seleccionar un elemento con **id=#mildentificador**:

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

jQuery también permite utilizar selectores CSS más complicados para determinar con precisión las etiquetas que desea seleccionar.

- **Selectores descendientes**: proporcionan una manera de apuntar a una etiqueta dentro de otra etiqueta.

  ```
  $ ('#navBar a')
  ```

- **Selectores afiliados**: se dirigen a una etiqueta que es el hijo de otra etiqueta.

  ```
  $('body > p')
  ```

- **Selectores de elementos adyacentes:** permiten seleccionar una etiqueta que está justo a continuación de otra.

  ```
  $(´h2 + div')
  ```

- **Selectores de atributos:** Permiten seleccionar elementos basándonos en uno de sus atributos. Hay muchos selectores de atributos diferentes:
  - **[atributo]** selecciona elementos que tienen el atributo especificado asignado en el código HTML. Por ejemplo,

    ```
    $ ('a [href]')
    ```

    localiza todas las etiquetas <a> que tienen un atributo href establecido.
  - **[atributo = "valor"]** selecciona los elementos que tienen un atributo particular con un valor específico. Por ejemplo, para encontrar todos los cuadros de texto en un formulario, puede utilizar esto:

    ```
    $ ('input [type = "text"]')
    ```

  - **[atributo ^= "value"]** selecciona los elementos en los que el atributo se inicia con un valor específico. Por ejemplo, si quieres encontrar enlaces que apuntan fuera del sitio web, puede utilizar este código:

    ```
    $ ('a [href ^= "http://"]')
    ```

  - **[atributo $= "valor"]** Selecciona elementos cuyo atributo termina con un valor específico, es ideal para hacer coincidir las extensiones de archivo. Por ejemplo, con este selector, puede encontrar enlaces que apuntan a archivos PDF:

    ```
    $ ('a [href $= ".pdf"]')
    ```

  - **[atributo *= "valor"]** selecciona elementos cuyo atributo contiene un valor específico en cualquier parte del atributo. Por ejemplo, aquí es cómo encontraría un enlace que apunta a http://missingmanuals.com:

    ```
    $ ('a [href *= "missingmanuals.com"]')
    ```

Otros filtros en jQuery

jQuery también proporciona una forma de filtrar sus selecciones en base a ciertas características.

•**: even y :odd** Estos filtros funcionan como un contador. Cada elemento en una selección jQuery tiene un índice. Debemos recordar que los valores del índice para las matrices siempre comienzan en 0. Por eso, como los filtros :even devuelven los valores 0, 2, 4, etc. este filtro, en realidad, devuelve el primer, tercer y quinto artículos (y así sucesivamente) en la selección, en otras palabras, es realmente la selección de todos los elementos impares. El filtro :odd funciona de la misma manera excepto que selecciona cada número de índice impar (1, 3, 5, y así sucesivamente) para seleccionar el segundo, cuarto y sexto elemento.

•**:first y: last** seleccione el primer o el último elemento de un grupo. Por ejemplo, si desea seleccionar el primer párrafo en una página web, usted escribe esto:

```
$('p:first');
```

Y para seleccionar el último párrafo en una página, que le escribe esto:

```
$('p:last);
```

• Puede utilizar **:not()** para encontrar elementos que no coinciden con un tipo de selección en particular. Por ejemplo, digamos que usted desea seleccionar todas las etiquetas <a> excepto los que tienen la clase navButton.

```
$('a:not(.navButton)');
```

A la función :not() se le pasa el nombre del selector que desea ignorar. En este caso, .navButton es un selector de clase, por lo que este código se traduce como “que no tenga la clase .navButton." Se puede usar :not() con cualquiera de los filtros de jQuery y con la mayoría de los selectores de jQuery; así, por ejemplo, para encontrar todos los enlaces que no empieza con http: //, puede escribir lo siguiente:

```
$('a:not([ href ^= "http://"])')
```

•**: has()** encuentra elementos que contienen otro selector. Por ejemplo, digamos que usted quiere encontrar todas las etiquetas <li>, pero sólo si tienen una etiqueta <a> dentro de ellos escribiríamos:

```
$('li:has(a)')
```

Esta configuración es diferente de un selector descendiente, ya que no selecciona la <a>; selecciona la etiqueta <li>, pero sólo aquellos <li> con un enlace en su interior.

•**: contains()** encuentra elementos que contienen texto específico. Por ejemplo, para encontrar cada enlace que dice "Click Me!", Se puede crear un objeto jQuery así:

```
$('a:contains (“Click Me!”)')
```

•**: hidden** localiza elementos que están ocultos, que incluye elementos que o bien tienen la propiedad *display* CSS establecida en *none* (lo que significa que no los verá en la página), elementos que se esconden utilizando la función hide() de jQuery, elementos con los valores de anchura y altura a 0, y campos de formulario ocultos. (Este selector no se aplica a elementos cuya propiedad CSS *visibility* se establece para que sea invisible.) Por ejemplo, digamos que se han ocultado varias etiquetas ; podemos encontrarlas y luego hacerlas visibles usando jQuery, así:

```
$('div:hidden').show();
```

Esta línea de código no tiene efecto en etiquetas que están actualmente visibles en la página.

• **:visible** es lo contrario de :hidden. Localiza elementos que son visibles en la página.

Algunos ejemplos

```
$('div.foo').has('p');          // el elemento div.foo contiene
elementos <p>
$('h1').not('.bar');            // el elemento h1 no posse la clase
'bar'
$('ul li').filter('.current');  // un item de una lista desordenada
                                // que posse la clase 'current'
$('ul li').first();             // el primer item de una lista
desordenada
$('ul li').eq(5);               // el sexto item de una lista
desordenada
```

Selección de elementos de un formulario

jQuery ofrece varios pseudo-selectores que ayudan a encontrar elementos dentro de los formularios, éstos son especialmente útiles ya que, dependiendo de los estados de cada elemento o su tipo, puede ser difícil distinguirlos utilizando selectores CSS estándar.

- :`button` Selecciona elementos `<button>` y con el atributo `type='button'`
- :`checkbox` Selecciona elementos `<input>` con el atributo `type='checkbox'`
- :`checked` Selecciona elementos `<input>` del tipo checkbox seleccionados
- :`disabled` Selecciona elementos del formulario que están deshabitados
- :`enabled` Selecciona elementos del formulario que están habilitados
- :`file` Selecciona elementos `<input>` con el atributo `type='file'`
- :`image` Selecciona elementos `<input>` con el atributo `type='image'`
- :`input` Selecciona elementos `<input>`, `<textarea>` y `<select>`
- :`password` Selecciona elementos `<input>` con el atributo `type='password'`
- :`radio` Selecciona elementos `<input>` con el atributo `type='radio'`
- :`reset` Selecciona elementos `<input>` con el atributo `type='reset'`
- :`selected` Selecciona elementos `<options>` que están seleccionados
- :`submit` Selecciona elementos `<input>` con el atributo `type='submit'`
- :`text` Selecciona elementos `<input>` con el atributo `type='text'`

Utilizando pseudo-selectores en elementos de formularios

```
// obtiene todos los elementos inputs dentro del formulario #myForm
```

```
$('#myForm :input');
```

AÑADIR CONTENIDO A UNA PÁGINA

Para el estudio de estas funciones, suponga que tiene una página con el siguiente código HTML:

```
<div id = "contenedor">
  <div id = "errores">
    <h2> Errores: </h2>
  </div>
</div>
```

Aquí están las cinco funciones de jQuery más útiles para la manipulación de contenido en una página:

- **.html ()** puede tanto leer el código HTML actual, dentro de un elemento como reemplazar el contenido actual con algún otro HTML.

  `alert ($('#errores').html());` → recuperar el código HTML actualmente dentro de la selección

  `$('#errores').html('<p> Hay cuatro errores en el formulario </ p>.');` → reemplaza el contenido actual dentro de la selección:

  ```
  <div id = "contenedor">
    <div id = "errores">
      <p> Hay cuatro errores en el formulario </ p>
    </ div>
  </ div>
  ```

  NOTA Si utiliza html() o text() para recuperar el HTML o texto de una selección de jQuery que contiene varios elementos, sólo se recupera el HTML o texto del primer elemento de la selección.

  Sin embargo, cuando se utiliza html() o text() para insertar HTML o texto en una selección jQuery, todos los elementos seleccionados se verán afectados por la inserción. Por ejemplo, este código

- **.text ()** funciona como .html (), pero no acepta etiquetas HTML. Es útil cuando se desea reemplazar el texto dentro de una etiqueta.

```
$ ('#errores h2').text('No se han encontrado errores.');
```

El <h2> etiqueta se mantiene en su lugar; sólo ha cambiado el texto de dentro. jQuery codifica las etiquetas HTML que se pasan a la función text(), por lo que <p> se traduce a < p &gt ;.

- **.append()** añade HTML como el último elemento hijo del elemento seleccionado. `$ ('#errors').append('<p> Hay cuatro errores en el formulario </ p>.');`

Después de esta función se ejecuta, se termina con HTML así:

```
<div id = "contenedor">
  <div id = "errores">
    <h2> Errores: </ h2>
    <p> Hay cuatro errores en el formulario </ p>
  </ div>
</ div>
```

- **.prepend ()** es igual .append (), pero añade HTML directamente después de la etiqueta de apertura para la selección.

```
$('#errors').prepend('<p> Hay cuatro errores en el formulario </ p>.');
```

Después de esta función prepend (), se termina con el siguiente código HTML:

```
<div id = "contenedor">
<div id = "errores">
    <p> Hay cuatro errores en el formulario </ p>
    <h2> Errores: </ h2>
  </ div>
</ div>
```

- **.before ()** o **.after** () agregan HTML a las afueras de una selección, ya sea antes de la etiqueta de apertura del elemento seleccionado o directamente después de la etiqueta de cierre del elemento.

```
<input type = "text" name = "username" id = "username">
```

Supongamos ahora que cuando el visitante envía el formulario, este campo está vacío. Para agregar un mensaje después de este campo, puede utilizar la función after () así:

```
$('#username').after('<span class = "error"> Nombre de usuario requerido </ span>');
```

Esa línea de código hace que la página Web muestre el mensaje de error, y el componente de HTML se vería así:

```
<input type = "text" name = "nombre de usuario" id =
"nombre de usuario">

<span class = "error"> Nombre de usuario requerido </
span>
```

La función .before() simplemente pone el nuevo contenido antes del elemento seleccionado. Así que esta línea de código:

```
$('#username').before('<span class = "error"> Nombre
de usuario requerido </ span>');
```

generaría este código HTML:

```
<span class = "error"> Nombre de usuario requerido </
span>

<input type = "text" name = "nombre de usuario" id =
"nombre de usuario">
```

Sustitución y eliminación de Selecciones

A veces es posible que desee reemplazar completamente o eliminar un elemento seleccionado. Por ejemplo, digamos que usted ha creado un cuadro emergente de diálogo que en realidad sólo es un flotando en la parte superior de la página. Cuando el usuario hace clic en el botón "Cerrar" en el cuadro de diálogo, naturalmente, desea quitar el cuadro de diálogo de la página. Para ello, puede utilizar la función de jQuery **remove();** puede utilizar el siguiente código para eliminarlo:

```
$('#popup').remove();
```

La función .remove () no se limita a un solo elemento. Digamos que quiere eliminar todos los que tienen una clase error; Puede hacerlo así:

```
$('span.error').remove();
```

También puede reemplazar completamente una selección con nuevos contenidos. Por ejemplo, suponga que tiene una página con fotos de los productos que su empresa vende. Cuando un visitante hace clic en una imagen de un producto, se añade a un carrito de compras. Es posible que desee reemplazar la etiqueta <img> con un poco de texto cuando se hace clic en la imagen ("Añadir al carro", por ejemplo). Así es como se hace eso con jQuery usando la función **replaceWith();**:

```
$('# product101').replaceWith (<p> Añadir al carro </ p>.
');
```

Este código elimina la etiqueta <img> de la página y la sustituye por una etiqueta <p>.

NOTA: jQuery también incluye una función denominada **clone()** que le permite hacer una copia de un elemento seleccionado.

DEFINIR Y LEER LOS ATRIBUTOS DE LAS ETIQUETAS

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).

CAMBIO DE LAS PROPIEDADES CSS

con un parametro obtiene
con 2 parametros cambia el valor a esa propiedad

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.

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

Además, .css() permite establecer el valor de una propiedad determinada.

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

jQuery entiende y devuelve el valor correcto para .css ("background-color") y .css ("backgroundColor").

La recuperación de las propiedades abreviadas CSS (por ejemplo, margin, background, border), no está garantizada. Por ejemplo, si desea recuperar el ancho del borde, utilice:

```
$(elem).css ("borderTopWidth"),
$(elem).css ("borderBottomWidth")
```

y así sucesivamente.

Clases

• **addClass()** añade una clase especificada a un elemento. Agrega addClass() después de una selección jQuery y pase a la función una cadena que representa el nombre de la clase que desea añadir.

```
$ ('a[href ^= "http://"]').addClass('externalLink');
```

Este código tomaría un código HTML como este:

```
<a href="http://www.oreilly.com/">
```

Y lo cambiaría a lo siguiente:

```
<a href="http://www.oreilly.com/" class="externalLink">
```

Para que esta función sea de alguna utilidad, tendrá que crear un estilo de clase CSS de antemano y agregarlo a la hoja de estilo de la página.

NOTA addClass ('externalLink') es correcto, pero addClass ('. ExternalLink') es incorrecta.

• **removeClass()** es lo contrario de addClass(). Se elimina la clase especificada de los elementos seleccionados.

```
$('#Alertbox').removeClass('destacado');
```

• **toggleClass()** añade la clase si no existe ya, o elimina la clase si existiese. La inversión es una manera popular de mostrar un elemento, ya sea en un estado activado o desactivado. Digamos que tiene un botón en una página web que, al hacer clic, cambia la clase del elemento <body>. De este modo, se puede añadir un cambio de estilo completo a una página web mediante la elaboración de un segundo conjunto de estilos utilizando selectores de descendientes.

```
$('#changeStyle').click(function() {
  $("body").toggleClass('altStyle');
});
```

La línea en negrita del código muestra la función toggleClass (); añade o elimina la clase altStyle con cada clic del botón.

Leer y definir atributos HTML

La función **attr()** le permite leer un atributo HTML específico de una etiqueta. Por ejemplo, para determinar el archivo gráfico actual un determinado <img>, se pasa la cadena 'src' (por la propiedad src del <img> de la etiqueta) a la función:

```
var imageFile = $('#banner img').attr('src');
```

La función attr() devuelve el valor del atributo, tal y como se ha especificado en HTML.

Si pasa un segundo argumento a la función attr(), se puede establecer el atributo de la etiqueta.

```
$('#banner img').attr('src', 'images/newImage.png');
```

Si desea eliminar por completo un atributo de una etiqueta, utilice la función **removeAttr()**. Por ejemplo, el código elimina la propiedad bgColor de la etiqueta <body>:

```
$("body").removeAttr ('bgColor');
```

DIMENSIONES

El código mostrado en el ejemplo "Métodos básicos sobre Dimensiones" es solo un breve resumen de las funcionalidades relaciones a dimensiones en jQuery; para un completo detalle puede consultar api.jquery.com/category/dimensions.

Métodos básicos sobre Dimensiones

```
$('h1').width('50px');  // establece el ancho de todos los elementos
H1
$('h1').width();        // obtiene el ancho del primer elemento H1

$('h1').height('50px'); // establece el alto de todos los elementos H1
$('h1').height();       // obtiene el alto del primer elemento H1

// devuelve un objeto conteniendo información sobre la posición
// del primer elemento relativo al "offset" (posición) de su elemento
padre
$('h1').position();
```

ACTUAR SOBRE CADA ELEMENTO EN UNA SELECCIÓN

Una de las cualidades únicas de jQuery es que la mayoría de sus funciones usan un bucle de forma automática a través de cada elemento en una selección de jQuery. Por

ejemplo, para hacer que todos los <img> en una página se desvanecen, sólo necesita una línea de código JavaScript:

```
$('img').fadeOut ();
```

Pero hay muchas ocasiones en las que nos vendría bien realizar un bucle a través de una selección de elementos y realizar una serie de acciones en cada elemento. jQuery proporciona la función **each()** para este propósito.

Por ejemplo, supongamos que desea añadir la lista de todos los enlaces externos de una web en una caja de bibliografía al final de la página, tal vez titulado "Otros sitios mencionados en este artículo.", así puede crear esa caja:

1. Recupere todos los enlaces que apuntan fuera de su sitio.
2. Consiga el atributo HREF de cada enlace (URL).
3. Añada la URL a la otra lista de enlaces en el cuadro de bibliografía.

jQuery no tiene una función integrada que realice estos pasos exactos, pero se puede utilizar la función each() para hacerlo usted mismo:

```
$('selector').each();
```

FUNCIONES ANÓNIMAS

Para utilizar la función each(), se pasa de un tipo especial de argumento, una función anónima. La **función anónima** es simplemente una función que contiene lo que se desea realizar en cada elemento seleccionado. Se llama anónima debido a que no es necesario darle un nombre. Aquí está la estructura básica de una función anónima:

```
función() {

  // su código va aquí

}
```

Como no hay nombre, no se tiene una manera de llamar a la función. En su lugar, se utiliza la función anónima como un argumento que se pasa a otra función (extraño y confuso, pero cierto!). He aquí cómo se incorpora una función anónima como parte de la función each ():

```
$('selector').each(function () {

// el código va aquí

});
```

THIS Y $ (THIS)

Cuando utilice la función each (), querrá tener acceso o establecer atributos para cada elemento, para, por ejemplo, la URL para cada enlace externo. Para acceder al elemento actual a través de cada bucle, se utiliza una palabra clave especial que se llama **this**. La primera vez que se ejecuta el bucle, this se refiere al primer elemento en la selección jQuery, mientras que la segunda iteración del bucle, this se refiere al segundo elemento, y así sucesivamente.

La selección especial jQuery le permite aprovechar todas las maravillosas funciones de jQuery. Así que para convertir this en su equivalente jQuery, escribiremos**$(this)**.

Para ayudar a aclarar cómo utilizar $(this), echaremos otro vistazo a la tarea descrita al comienzo de esta sección: una lista de enlaces externos en una sección de bibliografía en la parte inferior de una página.

Supongamos que de la página HTML ya tiene una etiqueta lista para los enlaces externos. Por ejemplo:

```
<div id = "bibliography">
<h3> páginas Web que se hace referencia en este artículo </
h3>
<ul id = "bibList">
</ul>
</div>
```

El primer paso es obtener una lista de todos los enlaces que apuntan fuera de su sitio. Puede hacerlo a través de un selector de atributos:

```
$('a[href ^= "http://"]')
```

Ahora, para recorrer cada enlace, agregar la función each ():

```
$('a[href ^= "http: //"]').each()
```

A continuación, agregue una función anónima:

```
$('a[href ^= "http: //"]').each(function () {

});
```

El primer paso en la función anónima es recuperar la URL del enlace. Debido a que cada enlace tiene una dirección URL diferente, debe tener acceso al elemento actual cada vez ejecute el bucle. $(this) le permite hacer precisamente eso:

```
$('a[href ^= http: //]') each (function () {
var extLink = $(this).attr('href');
});
```

El código en el medio, la línea en negrita hace varias cosas: En primer lugar, se crea una nueva variable (extLink) y almacena el valor de la propiedad href del elemento actual. Cada vez que se ejecute el bucle, $(this) se refiere a un enlace diferente en la página, así que cada vez que se ejecuta el bucle, la variable *extLink* cambia.

Después de eso, es sólo cuestión de añadir un nuevo elemento a la lista <ul>, así:

```
$('a[href ^= http://]').each(function(){

varextLink = $(this).attr('href');

  $('#bibLista').append('<li>' + extLink + '</ li>.');

});
```

NOTA El script de ejemplo que se utiliza en esta sección es una buena manera de ilustrar el uso de la palabra reservada $(this), pero probablemente no es la mejor manera de llevar a cabo la tarea de escribir una lista de enlaces externos a una página. En primer lugar, si no hay enlaces, aparecerá la etiqueta (que fue codificado en de la página HTML), pero va a ser vacía. Además, si alguien visita la página sin JavaScript activado, no va a ver los enlaces, pero verá la caja vacía. Un mejor enfoque es utilizar JavaScript para crear etiqueta también.

ENCADENAMIENTO

Si en una selección se realiza una llamada a un método, y éste devuelve un objeto jQuery, es posible seguir un "encadenado" de métodos en el objeto.

Encadenamiento

```
$('#content').find('h3').eq(2).html('nuevo texto para el tercer
elemento h3');
```

Por otro lado, si se está escribiendo un encadenamiento de métodos que incluyen muchos pasos, es posible escribirlos línea por línea, haciendo que el código sea más agradable para leer.

Formateo de código encadenado

```
$('#content')
    .find('h3')
    .eq(2)
    .html('nuevo texto para el tercer elemento h3');
```

Si desea volver a la selección original en el medio del encadenado, jQuery ofrece el método $.fn.end para poder hacerlo.

Restablecer la selección original utilizando el método $.fn.end

```
$('#content')
    .find('h3')
    .eq(2)
        .html('nuevo texto para el tercer elemento h3')
    .end() // reestablece la selección a todos los elementos h3 en
#content
    .eq(0)
        .html('nuevo texto para el primer elemento h3');
```

Nota El encadenamiento es muy poderoso y es una característica que muchas bibliotecas JavaScript han adoptado desde que jQuery se hizo popular. Sin embargo, debe ser utilizado con cuidado. Un encadenamiento de métodos excesivo puede hacer un código extremadamente difícil de modificar y depurar. No existe una regla que indique como de largo o corto debe ser el encadenado — pero es recomendable que tenga en cuenta este consejo.

RECORRER EL DOM

Una vez obtenida la selección, es posible encontrar otros elementos utilizando a la misma selección. En api.jquery.com/category/traversing puede encontrar una completa documentación sobre los métodos de recorrido de DOM (en inglés *traversing*) que posee jQuery.

Nota Debe ser cuidadoso en recorrer largas distancias en un documento — recorridos complejos obligan que la estructura del documento sea siempre la misma, algo que es difícil de garantizar. Uno o dos pasos para el recorrido están bien, pero generalmente hay que evitar atravesar desde un contenedor a otro.

Moverse a través del DOM utilizando métodos de recorrido

```
// seleccionar el inmediato y próximo elemento <p> con respecto a H1
$('h1').next('p');

// seleccionar el elemento contenedor a un div visible
$('div:visible').parent();

// seleccionar el elemento <form> más cercano a un input
$('input[name=first_name]').closest('form');

// seleccionar todos los elementos hijos de #myList
$('#myList').children();

// seleccionar todos los items hermanos del elemento <li>
```

```
$('li.selected').siblings();
```

También es posible interactuar con la selección utilizando el método $.fn.each. Dicho método interactúa con todos los elementos obtenidos en la selección y ejecuta una función por cada uno. La función recibe como argumento el índice del elemento actual y al mismo elemento. De forma predeterminada, dentro de la función, se puede hacer referencia al elemento DOM a través de la declaración this.

Interactuar en una selección

indice, valor

```
$('#myList li').each(function(idx, el) {
    console.log(
        'El elemento ' + idx +
        'contiene el siguiente HTML: ' +
        $(el).html()
); });
```

MANIPULACIÓN DE ELEMENTOS

Una vez realizada la selección de los elementos que desea utilizar, *la diversión comienza*. Es posible cambiar, mover, eliminar y duplicar elementos. También crear nuevos a través de una sintaxis simple.

La documentación completa sobre los métodos de manipulación puede encontrarla en la sección Manipulation.

Obtener y establecer información en elementos

Existen muchas formas por las cuales se puede modificar un elemento. Entre las tareas más comunes están las de cambiar el HTML interno o algún atributo del mismo. Para este tipo de tareas, jQuery ofrece métodos simples, funcionales en todos los navegadores modernos. Incluso es posible obtener información sobre los elementos utilizando los mismos métodos, pero en su forma de método obtenedor.

Nota Realizar cambios en los elementos, es un trabajo trivial, pero hay debe recordar que el cambio afectará a todos los elementos en la selección, por lo que, si desea modificar un sólo elemento, tiene que estar seguro de especificarlo en la selección antes de llamar al método establecedor, por ejemplo, con el método $.fn.eq.

Nota Cuando los métodos actúan como obtenedores, por lo general, solamente trabajan con el primer elemento de la selección. Además, no devuelven un objeto jQuery, por lo cual no es posible encadenar más métodos en el mismo. Una excepción es el método $.fn.text, el cual permite obtener el texto de los elementos de la selección.

metodo obtenedor: .html()
con un parametro cambia el contenido

- `$.fn.html` Obtiene o establece el contenido HTML de un elemento.
- `$.fn.text` Obtiene o establece el contenido en texto del elemento; en caso se pasarle como argumento código HTML, este es despojado.
- `$.fn.attr` Obtiene o establece el valor de un determinado atributo.
- `$.fn.width` Obtiene o establece el ancho en pixeles del primer elemento de la selección como un entero.
- `$.fn.height` Obtiene o establece el alto en pixeles del primer elemento de la selección como un entero.
- `$.fn.position` Obtiene un objeto con información sobre la posición del primer elemento de la selección, relativo al primer elemento padre posicionado. Este método es solo obtenedor.
- `$.fn.val` Obtiene o establece el valor (*value*) en elementos de formularios.

Cambiar el HTML de un elemento

```
$('#myDiv p:first')
    .html('Nuevo <strong>primer</strong> párrafo');
```

Mover, copiar y eliminar elementos

Existen varias maneras para mover elementos a través del DOM; las cuales se pueden separar en dos enfoques:

- Querer colocar el/los elementos seleccionados de forma relativa a otro elemento
- Querer colocar un elemento relativo a el/los elementos seleccionados.

Por ejemplo, jQuery provee los métodos $.fn.insertAfter y $.fn.after. El método $.fn.insertAfter coloca a el/los elementos seleccionados después del elemento que se haya pasado como argumento; mientras que el método $.fn.after coloca al elemento pasado como argumento después del elemento seleccionado. Otros métodos también siguen este patrón: $.fn.insertBefore y $.fn.before; $.fn.appendTo y $.fn.append; y $.fn.prependTo y $.fn.prepend.

La utilización de uno u otro método dependerá de los elementos que tenga seleccionados y el tipo de referencia que se quiera guardar con respecto al elemento que se está moviendo.

Mover elementos utilizando diferentes enfoques

```
// hacer que el primer item de la lista sea el último
var $li = $('#myList li:first').appendTo('#myList');

// otro enfoque para el mismo problema
$('#myList').append($('#myList li:first'));

// debe tener en cuenta que no hay forma de acceder a la
// lista de items que se ha movido, ya que devuelve la lista en sí
```

Clonar elementos

Cuando se utiliza un método como $.fn.appendTo, lo que se está haciendo es mover al elemento; pero a veces en lugar de eso, se necesita mover un duplicado del mismo elemento. En este caso, es posible utilizar el método $.fn.clone.

Obtener una copia del elemento

```
// copiar el primer elemento de la lista y moverlo al final de la misma
$('#myList li:first').clone().appendTo('#myList');
```

Nota Si se necesita copiar información y eventos relacionados al elemento, se debe pasar true como argumento de $.fn.clone.

Eliminar elementos

Existen dos formas de eliminar elementos de una página: Utilizando $.fn.remove o $.fn.detach. Cuando desee eliminar de forma permanente al elemento, utilize el método $.fn.remove. Por otro lado, el método $.fn.detach también elimina el elemento, pero mantiene la información y eventos asociados al mismo, siendo útil en el caso que necesite reinsertar el elemento en el documento.

Nota El método $.fn.detach es muy útil cuando se está manipulando de forma severa un elemento, ya que es posible eliminar al elemento, trabajarlo en el código y luego restaurarlo en la página nuevamente. Esta forma tiene como beneficio no tocar el DOM mientras se está modificando la información y eventos del elemento.

Por otro lado, si se desea mantener al elemento pero se necesita eliminar su contenido, es posible utiliza el método $.fn.empty, el cual "vaciará" el contenido HTML del elemento.

Crear nuevos elementos

jQuery provee una forma fácil y elegante para crear nuevos elementos a través del mismo método $() que se utiliza para realizar selecciones.

Crear nuevos elementos

```
$('<p>Un nuevo párrafo</p>');
$('<li class="new">nuevo item de la lista</li>');
```

Crear un nuevo elemento con atributos utilizando un objeto

```
$('<a/>', {
    html : 'Un <strong>nuevo</strong> enlace',
    'class' : 'new',
    href : 'foo.html'
});
```

Note que en el objeto que se pasa como argumento, la propiedad class está entre comillas, mientras que la propiedad href y html no lo están. Por lo general, los

nombres de propiedades no deben estar entre comillas, excepto en el caso que se utilice como nombre una palabra reservada (como es el caso de class).

Cuando se crea un elemento, éste no es añadido inmediatamente a la página, sino que se debe hacerlo en conjunto con un método.

Crear un nuevo elemento en la página

```
var $myNewElement = $('<p>Nuevo elemento</p>');
$myNewElement.appendTo('#content');

// eliminará al elemento <p> existente en #content
$myNewElement.insertAfter('ul:last');

// clonar al elemento <p> para tener las dos versiones
$('ul').last().after($myNewElement.clone());
```

Estrictamente hablando, no es necesario guardar al elemento creado en una variable — es posible llamar al método para añadir el elemento directamente después de $(). Sin embargo, la mayoría de las veces se deseará hacer referencia al elemento añadido, por lo cual, si se guarda en una variable no es necesario seleccionarlo después.

Crear y añadir al mismo tiempo un elemento a la página

```
$('ul').append('<li>item de la lista</li>');
```

Nota La sintaxis para añadir nuevos elementos a la página es muy fácil de utilizar, pero es tentador olvidar que hay un costo enorme de rendimiento al agregar elementos al DOM de forma repetida. Si está añadiendo muchos elementos al mismo contenedor, en lugar de añadir cada elemento uno por vez, lo mejor es concatenar todo el HTML en una única cadena de caracteres para luego anexarla al contenedor. Una posible solución **es utilizar un array que posea todos los elementos, luego reunirlos utilizando join y finalmente anexarla.**

```
var myItems = [], $myList = $('#myList');

for (var i=0; i<100; i++) {
    myItems.push('<li>item ' + i + '</li>');
}

$myList.append(myItems.join(''));
```

Manipulación de atributos

Las capacidades para la manipulación de atributos que ofrece la biblioteca son extensas. La realización de cambios básicos son simples, sin embargo el método $.fn.attr permite manipulaciones más complejas.

Manipular un simple atributo

```
$('#myDiv a:first').attr('href', 'newDestination.html');
```

Manipular múltiples atributos

```
$('#myDiv a:first').attr({
```

```
    href : 'newDestination.html',
    rel : 'super-special'
});
```

Utilizar una función para determinar el valor del nuevo atributo

```
$('#myDiv a:first').attr({
    rel : 'super-special',
    href : function(idx, href) {
        return '/new/' + href;
    }
});

$('#myDiv a:first').attr('href', function(idx, href) {
    return '/new/' + href;
});
```