Guía Teórica
jQuery

Indice

jQuery

Que es jQuery

jQuery es una biblioteca de JavaScript, creada inicialmente por John Resig, 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. Fue presentada el 14 de enero de 2006 en el BarCamp NYC. jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privativos.1 jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio. Las empresas Microsoft y Nokia anunciaron que incluirán la biblioteca en sus plataformas.2 Microsoft la añadirá en su IDE Visual Studio3 y la usará junto con los frameworks ASP.NET AJAX y ASP.NET MVC, mientras que Nokia los integrará con su plataforma Web Run-Time.

¿Debo saber JavaScript?

Si, ya que jQuery es javascript solo que implementa métodos, propiedades para agilizar el código. Por ende, se recomienda las nociones básicas de javascript como usar los bucles, declarar variables, funciones, objetos entre otros.

¿Cuál es la diferencia entre jQuery y JavaScript?

Si eres nuevo en el uso de jQuery es importante saber que jQuery no es un lenguaje de programación sino un conjunto de funciones y métodos de JavaScript. En realidad JavaScript es el lenguaje de programación y jQuery se refiere a las librerías que se usan, si se desea, para facilitar las tareas.

Ahora veamos la diferencia entre ambas con un ejemplo sencillo:

JavaScript

```
function cambiarFondo(color){
        document.body.style.background=color;
}
Onload=”cambiarFondo(‘#ccc’);”
```

jQuery

```
$(‘body’).css(‘background’,’#ccc’):
```

Puedes notar la diferencia, es obvia. jQuery logra cambiar el color de fondo de la página web con solo una línea de código mientras que JavaScript lo realiza con cuatro líneas. Además jQuery no tiene ninguna dificultad en ejecutar su código en cualquier navegador sea Firefox, Chrome, Internet Explorer, Opera, etc.

Características

- Selección de elementos DOM.
- Interactividad y modificaciones del árbol DOM, incluyendo soporte para CSS 1-3 y un plugin básico de XPath.
- Eventos.
- Manipulación de la hoja de estilos CSS.
- Efectos y animaciones.
- Animaciones personalizadas.
- AJAX.
- Soporta extensiones.
- Utilidades varias como obtener información del navegador, operar con objetos y vectores, funciones para rutinas comunes, etc.
- Compatible con los navegadores Mozilla Firefox 2.0+, Internet Explorer 6+, Safari 3+, Opera 10.6+ y Google Chrome 8+.

Ventajas

La ventaja principal de jQuery es que es mucho más fácil que sus competidores. Usted puede agregar plugins fácilmente, traduciéndose esto en un ahorro substancial de tiempo y esfuerzo. De hecho, una de las principales razones por la cual Resig y su equipo crearon jQuery fue para ganar tiempo (en el mundo de desarrollo web, tiempo importa mucho).

La licencia open source de jQuery permite que la librería siempre cuente con soporte constante y rápido, publicándose actualizaciones de manera constante. La comunidad jQuery es activa y sumamente trabajadora.

Otra ventaja de jQuery sobre sus competidores como Flash y puro CSS es su excelente integración con AJAX.

En resumen:

- jQuery es flexible y rápido para el desarrollo web
- Viene con licencia MIT y es Open Source
- Tiene una excelente comunidad de soporte
- Tiene Plugins
- Bugs son resueltos rápidamente
- Excelente integración con AJAX

Desventajas

Una de las principales desventajas de jQuery es la gran cantidad de versiones publicadas en el corto tiempo. No importa si usted está corriendo la última versión de jQuery, usted tendrá que hostear la librería usted mismo (y actualizarla constantemente), o descargar la librería desde Google (atractivo, pero puede traer problemas de incompatibilidad con el código).

Además del problema de las versiones, otras desventajas que podemos mencionar:

- jQuery es fácil de instalar y aprender, inicialmente. Pero no es tan fácil si lo comparamos con CSS
- Si jQuery es implementado inapropiadamente como un Framework, el entorno de desarrollo se puede salir de control.

Otros proyectos de la fundación jQuery

jQuery User Interface. Conocido como jQuery UI es una biblioteca de componentes para el framework jQuery que le añaden un conjunto de plug-in, así como efectos visuales con el objetivo de crear aplicaciones web. La página oficial es: https://jqueryui.com/

QUnit JS Unit Testing. Su página oficial es: https://qunitjs.com/, QUnit JavaScript es un marco potente y fácil de usar pruebas unitarias.Es usado por el jQuery, jQuery UI y proyectos jQuery Mobile y es capaz de probar cualquier código JavaScript genérico, incluyéndose a sí mismo !

jQuery Mobile. Este framework te permite el desarrollo de aplicaciones para teléfonos móviles (dispositivos táctiles). Con él podemos acelerar la creación de aplicaciones. Su página oficial es: http://jquerymobile.com/

Sizzle CSS Selector Engine. Su página oficial es: https://sizzlejs.com/. Es un motor selector de JavaScript pura y CSS. Funciona completamente independiente. Diseñado para un rendimiento óptimo con la delegación de eventos.

Descargando y agregando jQuery a nuestro proyecto.

1. Ingresar a la web oficial y descargar la última versión del framework http://jquery.com/

2. Agregarlo a nuestras etiquetas head, usando la etiqueta

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

3. Y abrimos las etiquetas de apertura, declaramos un evento al document, diciéndole que cuando este haya sido cargado, todo lo que escribamos dentro pueda ejecutarse recién, esto es una buena práctica.

También podemos consultar la API de JQuery en la siguiente dirección: http://api.jquery.com

Conceptos Básicos de jQuery

$(document).ready()

No es posible interactuar de forma segura con el contenido de una página hasta que el documento no se encuentre preparado para su manipulación. jQuery permite detectar dicho estado a través de la declaración$(document).ready() de forma tal que el bloque se ejecutará sólo una vez que la página esté disponible.

- **El bloque $(document).ready()**

```
$(document).ready(function() {
    console.log('el documento está preparado');
});
```

Existe una forma abreviada para $(document).ready() la cual podrá encontrar algunas veces; sin embargo, es recomendable no utilizarla en caso que este escribiendo código para gente que no conoce jQuery.

- **Forma abreviada para $(document).ready()**

```
$(function() {
    console.log('el documento está preparado');
});
```

Además es posible pasarle a $(document).ready() una función nombrada en lugar de una anónima:

- **Pasar una función nombrada en lugar de una función anónima**

```
function readyFn() {
    // código a ejecutar cuando el documento esté listo
}


$(document).ready(readyFn);
```

Selección de Elementos

El concepto más básico de jQuery es el de "seleccionar algunos elementos y realizar acciones con ellos". La biblioteca soporta gran parte de los selectores CSS3 y varios más no estandarizados. En http://api.jquery.com/category/selectors/ se puede encontrar una completa referencia sobre los selectores de la biblioteca.

A continuación se muestran algunas técnicas comunes para la selección de elementos:

- **Selección de elementos en base a su ID**

```
$('#myId'); // notar que los IDs deben ser únicos por página
```

- **Selección de elementos en base al nombre de clase**

```
$('div.myClass'); // si se especifica el tipo de elemento,
                  // se mejora el rendimiento de la selección
```

- **Selección de elementos por su atributo**

```
$('input[name=first_name]'); // tenga cuidado, que puede ser muy lento
```

- **Selección de elementos en forma de selector CSS**

```
$('#contents ul.people li');
```

- **Pseudo-selectores**

```
$('a.external:first');  // selecciona el primer elemento <a>
```

```
                    // con la clase 'external'
  $('tr:odd');          // selecciona todos los elementos <tr>
                    // impares de una tabla
  $('#myForm :input');  // selecciona todos los elementos del tipo
input
                    // dentro del formulario #myForm
  $('div:visible');     // selecciona todos los divs visibles
  $('div:gt(2)');       // selecciona todos los divs excepto los tres
primeros
  $('div:animated');    // selecciona todos los divs actualmente
animados
```

- **Elección de Selectores**

La elección de buenos selectores es un punto importante cuando se desea mejorar el rendimiento del código. Una pequeña especificidad — por ejemplo, incluir el tipo de elemento (como div) cuando se realiza una selección por el nombre de clase — puede ayudar bastante. Por eso, es recomendable darle algunas "pistas" a jQuery sobre en que lugar del documento puede encontrar lo que desea seleccionar. Por otro lado, demasiada especificidad puede ser perjudicial. Un selector como #miTabla thead tr th.especial es un exceso, lo mejor sería utilizar#miTabla th.especial.

jQuery ofrece muchos selectores basados en atributos, que permiten realizar selecciones basadas en el contenido de los atributos utilizando simplificaciones de expresiones regulares

```
// encontrar todos los <a> cuyo atributo rel terminan en "thinger"
```

```
$("a[rel$='thinger']");
```

Estos tipos de selectores pueden resultar útiles pero también ser muy lentos. Cuando sea posible, es recomendable realizar la selección utilizando IDs, nombres de clases y nombres de etiquetas.

- **Comprobar Selecciones**

Una vez realizada la selección de los elementos, querrá conocer si dicha selección entregó algún resultado. Para ello, pueda que escriba algo así:

```
if ($('div.foo')) { ... }
```

Sin embargo esta forma no funcionará. Cuando se realiza una selección utilizando $(), siempre es devuelto un objeto, y si se lo evalúa, éste siempre devolverá true. Incluso si la selección no contiene ningún elemento, el código dentro del bloque if se ejecutará.

En lugar de utilizar el código mostrado, lo que se debe hacer es preguntar por la cantidad de elementos que posee la selección que se ejecutó. Esto es posible realizarlo utilizando la propiedad JavaScript length. Si la respuesta es 0, la condición evaluará falso, caso contrario (más de 0 elementos), la condición será verdadera.

- **Evaluar si una selección posee elementos**

```
if ($('div.foo').length) { ... }
```

➢ Guardar Selecciones

Cada vez que se hace una selección, una gran cantidad de código es ejecutado. jQuery no guarda el resultado por sí solo, por lo tanto, si va a realizar una selección que luego se hará de nuevo, deberá salvar la selección en una variable.

• Guardar selecciones en una variable

```
var $divs = $('div');
```

Nota

En el ejemplo "Guardar selecciones en una variable", la variable comienza con el signo de dólar. Contrariamente a otros lenguajes de programación, en JavaScript este signo no posee ningún significado especial — es solamente otro carácter. Sin embargo aquí se utilizará para indicar que dicha variable posee un objeto jQuery. Esta práctica — una especie de Notación Húngara — es solo una convención y no es obligatoria.

Una vez que la selección es guardada en la variable, se la puede utilizar en conjunto con los métodos de jQuery y el resultado será igual que utilizando la selección original.

➢ Refinamiento y Filtrado de Selecciones

A veces, puede obtener una selección que contiene más de lo que necesita; en este caso, es necesario refinar dicha selección. jQuery ofrece varios métodos para poder obtener exactamente lo que desea.

• Refinamiento de selecciones

```
$('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

```
$('#myForm :input'); // obtiene todos los elementos inputs
                    // dentro del formulario #myForm
```

Trabajar con Selecciones

Una vez realizada la selección de los elementos, es posible utilizarlos en conjunto con diferentes métodos, éstos, generalmente, son de dos tipos: obtenedores (en inglés *getters*) y establecedores (en inglés *setters*). Los métodos obtenedores devuelven una propiedad del elemento seleccionado; mientras que los métodos establecedores fijan una propiedad a todos los elementos seleccionados.

➢ 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 luzca más agradable para leer.

➢ Obtenedores (Getters) & Establecedores (Setters)

jQuery "sobrecarga" sus métodos, en otras palabras, el método para establecer un valor posee el mismo nombre que el método para obtener un valor. Cuando un método es utilizado para establecer un valor, es llamado método establecedor (en inglés *setter*). En cambio, cuando un método es utilizado para obtener (o leer) un valor, es llamado obtenedor (en inglés *getter*).

- **El método $.fn.html utilizado como establecedor**

```
$('h1').html('hello world');
```

- **El método html utilizado como obtenedor**

```
$('h1').html();
```

Los métodos establecedores devuelven un objeto jQuery, permitiendo continuar con la llamada de más métodos en la misma selección, mientras que los métodos obtenedores devuelven el valor por el cual se consultó, pero no permiten seguir llamando a más métodos en dicho valor.

CSS, Estilos, & Dimensiones

jQuery incluye una manera útil de obtener y establecer propiedades CSS a los elementos.

Nota

Las propiedades CSS que incluyen como separador un guión del medio, en JavaScript deben ser transformadas a su estilo *CamelCase*. Por ejemplo, cuando se la utiliza como propiedad de un método, el estilo CSS font-size deberá ser expresado como fontSize. Sin embargo, esta regla no es aplicada cuando se pasa el nombre de la propiedad CSS al método $.fn.css — en este caso, los dos formatos (en*CamelCase* o con el guión del medio) funcionarán.

- **Obtener propiedades CSS**

```
$('h1').css('fontSize'); // devuelve una cadena de caracteres como "19px"
  $('h1').css('font-size'); // también funciona
```

- **Establecer propiedades CSS**

```
$('h1').css('fontSize', '100px'); // establece una propiedad individual CSS
  $('h1').css({
    'fontSize' : '100px',
```

```
    'color' : 'red'
  }); // establece múltiples propiedades CSS
```

Notar que el estilo del argumento utilizado en la segunda línea del ejemplo — es un objeto que contiene múltiples propiedades. Esta es una forma común de pasar múltiples argumentos a una función, y muchos métodos establecedores de la biblioteca aceptan objetos para fijar varias propiedades de una sola vez.

A partir de la versión 1.6 de la biblioteca, utilizando $.fn.css también es posible establecer valores relativos en las propiedades CSS de un elemento determinado:

- **Establecer valores CSS relativos**

```
$('h1').css({
    'fontSize' : '+=15px',          // suma 15px al tamaño original del elemento
    'paddingTop' : '+=20px'     // suma 20px al padding superior original del elemento
  });
```

- **Utilizar Clases para Aplicar Estilos CSS**

Para obtener valores de los estilos aplicados a un elemento, el método $.fn.css es muy útil, sin embargo, su utilización como método establecedor se debe evitar (ya que, para aplicar estilos a un elemento, se puede hacer directamente desde CSS). En su lugar, lo ideal, es escribir reglas CSS que se apliquen a clases que describan los diferentes

estados visuales de los elementos y luego cambiar la clase del elemento para aplicar el estilo que se desea mostrar.

- **Trabajar con clases**

```
var $h1 = $('h1');

$h1.addClass('big');
$h1.removeClass('big');
$h1.toggleClass('big');

if ($h1.hasClass('big')) { ... }
```

Las clases también pueden ser útiles para guardar información del estado de un elemento, por ejemplo, para indicar que un elemento fue seleccionado.

- **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

    $('h1').position();       // devuelve un objeto
conteniendo
```

```
                                    // información sobre la
posición
                                    // del primer elemento
relativo al
                                    // "offset" (posición)
de su elemento padre
```

Atributos

Los atributos de los elementos HTML que conforman una aplicación pueden contener información útil, por eso es importante poder establecer y obtener esa información.

El método $.fn.attr actúa tanto como método establecedor como obtenedor. Además, al igual que el método$.fn.css, cuando se lo utiliza como método establecedor, puede aceptar un conjunto de palabra clave-valor o un objeto conteniendo más conjuntos.

- **Establecer atributos**

```
$('a').attr('href', 'allMyHrefsAreTheSameNow.html');
  $('a').attr({
    'title' : 'all titles are the same too',
    'href' : 'somethingNew.html'
  });
```

En el ejemplo, el objeto pasado como argumento está escrito en varias líneas. Como se explicó anteriormente, los espacios en blanco no importan en JavaScript, por lo cual, es libre de utilizarlos para hacer el código más legible. En entornos de producción, se pueden utilizar herramientas de minificación, los cuales quitan los espacios en blanco (entre otras cosas) y comprimen el archivo final.

- **Obtener atributos**

```
$('a').attr('href');          // devuelve el atributo href perteneciente
                              // al primer elemento <a> del documento
```

Recorrer el DOM

Una vez obtenida la selección, es posible encontrar otros elementos utilizando a la misma selección.

En http://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 esta bien, pero generalmente hay que evitar atravesar desde un contenedor a otro.

- **Moverse a través del DOM utilizando métodos de recorrido**

```
$('h1').next('p');                  // seleccionar el
inmediato y próximo
                                        // elemento <p>
con respecto a H1
    $('div:visible').parent();      // seleccionar
el elemento contenedor
                                        // a un div
visible
```

```
    $('input[name=first_name]').closest('form');  // seleccionar el elemento
                                                  // <form> más cercano a un input
    $('#myList').children();          // seleccionar todos los elementos
                                      // hijos de #myList
    $('li.selected').siblings();      // seleccionar todos los items
                                      // hermanos del elemento <li>
```

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**

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

Obtener y Establecer Información en Elementos

Existen muchas formas por las cuales de 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.

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.

$.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 Remover 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.

- **Remover elementos**

Existen dos formas de remover elementos de una página: Utilizando $.fn.remove o $.fn.detach. Cuando desee remover de forma permanente al elemento, utilize el método $.fn.remove. Por otro lado, el método $.fn.detach también remueve 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 esta 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 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 vector 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 extensos. 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'
    });
```

Eventos

Introducción

Recordemos que los eventos en javascript son aquellas instrucciones que permiten variar el estado de la página mediante una acción. Esta acción puede venir por parte del usuario o llevarse a cabo en un cierto momento por estar así indicada. En javascript los eventos son instrucciones que podemos poner tanto en el lenguaje HTML (como atributos) como en el código javascript.

Los eventos, en javascript, se escriben todos con el prefijo "on", (onclick, onblur, onmouseout, onkeypress, etc). Cada evento indica algo que ocurre en algun elemento de la página (o en todo el documento). El evento registra un cambio en la página, y pone en marcha un mecanismo para introducir un código.

Cómo funcionan los eventos en jQuery

```
$(".mienlace").click(function(mievento){
  mievento.preventDefault();
  alert("Has hecho clic. Como he hecho preventDefault, no te llevaré al href");
});
```

- El evento se define sobre todos los objetos seleccionados mediante el selector jQuery

  En este caso todos los elementos con *class="mienlace"*

- El tipo de evento lo definimos con la función click o similares.
- El evento define como parámetro una función que será la manejadora del evento.

- La función manejadora del evento tiene a su vez un parámetro *mievento* que nos permite utilizar las propiedades o métodos del evento en cuestión.

  En este caso utilizaremos el método *preventDefault()*

Métodos jQuery para eventos

JQuery tiene también sus métodos para detectar los eventos. Estos se incluyen siempre en el lenguaje javascript. La mayoría de estos métodos se escriben igual que en javascript, pero sin el prefijo "on", por ejemplo onclick se convierte en .click()

Por supuesto podemos seguir utilizando los atributos javascript en el html, y mandar las acciones a ejecutar a una función:

```
<p id="p1" onclick="cambiar()">Pulsa para cambiar</p>
```

El método es el mismo que el utilizado sin jQuery. La diferencia es que ahora, dentro de la función del evento (function cambiar()..) podemos utilizar código jQuery.

Sin embargo podemos crear el evento desde el mismo código javascript-jQuery, utilizando las funciones de evento, por ejemplo, si tenemos un elemento html como el siguiente:

```
<p id="p1">Pulsa para cambiar</p>
```

En el código con jQuery (dentro del "document ready"), escribimos lo siguiente:

```
$("#p1").click(cambiar);
```

Esto crea un evento "onclick" asociado al elemento html, el código asociado al evento (la función "cambiar", debemos escribirlo dentro del paréntesis (Escribimos el nombre de la función sin paréntesis, "cambiar", ya que si escribiéramos "cambiar()" -con paréntesis- estaríamos llamando al valor de retorno de la función). Esto nos obliga también a crear una función "cambiar()" aparte, en la que escribir el código de lo que sucede al desencadenarse el evento.

Esta solución, aunque válida, no es la más adecuada, ya que obliga a crear otra función, con lo que se complica el código. La solución más adecuada (y la más utilizada), es crear una función anónima dentro del paréntesis, en la cual indicamos el código asociado al evento:

```
$("#p1").click(function(){
//Aquí va el código asociado al evento
});
```

Hemos visto ejemplos de esto en páginas anteriores con el método .click(), por lo que creo que no tiene mayor dificultad usarlo con otros eventos.

Métodos de eventos más utilizados

Aunque los eventos que se usan con jQuery son en su mayoría los mismos que los usados en javascript, puede haber algunas diferencias. Aquí vamos a poner los eventos más usados. Vamos a poner un ejemplo de cada uno de ellos. Podemos ver su funcionamiento en el recuadro de la derecha.

- **Eventos del ratón**

.click() : pulsar una vez el ratón sobre un elemento:

```
$("#click").click(function()    {
alert("click simple");
});
```

.dblclick() : pulsar el ratón dos veces seguidas sobre un elemento:

```
$("#dblclick").dblclick(function()    {
alert("doble click");
});
```

.mouseenter() : El ratón se sitúa encima de un elemento:

```
$("#mouse1").mouseenter(function() {
$(this).css("color","red");
});
```

.mouseleave() : El ratón, que estaba situado encima de un elemento, sale de él:

```
$("#mouse1").mouseleave(function() {
$(this).css("color","blue");
});
```

.hover() : Este evento es una combinación de los dos anteriores, detecta la posición del raton, y se produce tanto cuando éste se coloca encima del elemento, como cuando sale de él. Admite dos funciones, que las escribiremos separadas por una coma, La primera de ellas se produce cuando el ratón entra, y la segunda cuando sale:

```
$("#hover").hover(function() {
$(this).css("color","red");
},
function() {
$(this).css("color","blue");
});
```

.mousedown() : Momento en que presiona el botón, independientemente de si se suelta o no, vale tanto para el botón izquierdo como para el derecho.

```
$("#pulsaRaton").mousedown(function() {
$(this).html("<b>mousedown</b>, mouseup: pulsado")
});
```

.mouseup() : soltar un botón del ratón después de hacer click. El evento se produce sólo en el momento de soltar el botón.

```
$("#pulsaRaton").mouseup(function() {
$(this).html("mousedown, <b>mouseup</b>: sin pulsar")
});
```

.mousemove() : El evento se produce al mover el ratón sobre un elemento.

```
n1=0;
$("#capa2").mousemove(function(){
n1++;
```

```
$("#mousemove").html("mousemove: "+n1+" movimientos");
});
```

.mouseover() : El evento se produce al situarse el ratón sobre un elemento. El evento puede producirse varias veces mientras está encima del elemento.

```
n2=0;
$("#capa2").mouseover(function(){
n2++;
$("#mouseover").html("mouseover: "+n2+" veces");
});
```

.toggle() : El evento se produce al hacer click con el ratón. Tiene la particularidad de que admite varias funciones, que se van alternando en su ejecución cuando el usuario realiza clicks.

```
$("#toggle").toggle(function() {
$(this).css("color","red");
},
function() {
$(this).css("color","blue");
},
function() {
$(this).css("color","green");
});
```

➢ Eventos del teclado

.keydown() : El evento se produce en el momento que se presiona una tecla, independientemente de si se libera la presión o se mantiene. Se produce una única vez en el momento exacto de la presión.

.keypress() : Se produce al tener pulsada una tecla, Si se mantiene pulsada, el evento se produce varias veces, de la misma manera que se escriben varios caracteres al mantenerla pulsada. Se corresponde con el hecho de tener la tecla pulsada.

.keyup() : El evento se produce en el momento de dejar de presionar una tecla que teníamos pulsada.

Los eventos de teclado se suelen aplicar al documento (document), y no a un elemento concreto. veamos el código del ejemplo del cuadro de la derecha:

```
$(document).keydown(function(){
$("#teclado").html("tecla pulsada");
});
$(document).keyup(function(){
$("#teclado").html("tecla sin pulsar");
});
```

➢ Eventos combinados teclado-ratón

Son los eventos que controlan el "foco" o elemento seleccionado. Podemos llevar el foco a un elemento mediante el ratón (haciendole click), o mediante el teclado (tabulador o inter). En una página los elementos que admiten tener el foco suelen ser los elementos de formulario.

.focusin() : El elemento adquiere el foco de la aplicación.

.focusout() : El elemento pierde el foco de la aplicación.

.focus() : El elemento tiene el foco de la aplicación. Igual que el evento "onfocus" de javascript.

Asociación de DOM y Eventos

Métodos bind(), live() y delegate()

- En la versión de jQuery 1.7 han intentado unificar las APIs de manejo de eventos en los métodos on() y off().
- Estos métodos sustituyen a los antiguos bind(), delegate() y live().
- Información del resto de métodos

Método on() - Introducción

- $(elements).on(events [, selector] [, data], handler);
- Las funciones para eventos vistas hasta ahora, como click, son atajos a la función on():

```
// Asignar un manejador de eventos usando click
$("#header a").click(function() {
   // manejar el evento
});

// Asignar un manejador de eventos usando on
$("#header a").on("click", function() {
   // manejar el evento
});
```

Estas dos construcciones son equivalentes, pero on() permite hacer muchas más cosas...

Método on() - Ventajas I

- Permite usar un mismo manejador de eventos para múltiples elementos html suscribiendo el manejador a un elemento padre
- Dado el siguiente código html:

```
<div id="content">
  <a href="#">enlace1</a>
  <a href="#">enlace2</a>
  <a href="#">enlace2</a>
</div>
```

Si queremos asignar el mismo manejador de eventos a todos los elementos <a>, podemos hacerlo de la siguiente forma:

```
$("#content").on("click", "a", function() {
  // manejar el evento
  // $(this) apunta al <a> que ha generado el evento
});
```

- Esto tiene varias ventajas:

- Sólo se crea una única función, independientemente del número de elementos <a> que tengamos, reduciendo el consumo de recursos.
- Es válido para elementos que no existen todavía. Si apareciese un nuevo elemento <a> dentro del

  , automáticamente estaríamos manejando su evento click. Esto es especialmente útil cuando generamos html dinámicamente.

Método on() - Ventajas II

- Permite definir un mismo manejador de eventos para distintos tipos de eventos

```
$("p").on("click mouseenter mouseleave", function(e){
 if ($(this).css("color")!="rgb(250, 100, 0)")
  $(this).css("color", "rgb(250, 100, 0)");
 else
  $(this).css("color", "rgb(150, 0, 255)");
 })
```

Método off()

- Para esta acción, contábamos con varios métodos como unbind(), die() o undelegate().
- De nuevo, el objetivo principal de la nueva instrucción es reemplazarlos a todos de un modo consistente.
- La sintaxis de off() resulta similar a la de on():

$(elements).off([events] [, selector] [, handler]); Con off(), todos los parámetros son opcionales. Cuando se utiliza en su forma más simple, $(elements).off(), se eliminan todos los eventos asociados al conjunto seleccionado.

Ejemplo

- El evento click se asocia o se elimina del elemento #theone.
- Uso de find para optimizar el código
- Uso de click() para mayor legibilidad

```
<!DOCTYPE html>
<html>
 <head>
  <style>
     button { margin:5px; }
     button#theone { color:red; background:yellow; }
  </style>
  <script src="http://code.jquery.com/jquery-
latest.js"></script>
 </head>
 <body>
   <button id="theone">Does nothing...</button>
   <button id="bind">Add Click</button>
   <button id="unbind">Remove Click</button>
   <div style="display:none;">Click!</div>
   <script>
      function aClick() {
         $("div").show().fadeOut("slow");
      }
      $("#bind").click(function () {
         $("body").on("click", "#theone", aClick)
         .find("#theone").text("Can Click!");
      });
      $("#unbind").click(function () {
         $("body").off("click", "#theone", aClick)
         .find("#theone").text("Does nothing...");
      });
   </script>
 </body>
</html>
```