jQuery

El framework JavaScript de la Web 2.0 (2a edición)

Este libro se dirige a los **expertos o posibles expertos** en la creación de sitios Web. Conocer, incluso dominar **JavaScript**, las hojas de estilo **CSS**, el**DOM** y **AJAX,** son requisitos previos indispensables para entender y poner en práctica este libro.

En él, el autor ha dado prioridad a un **enfoque estructurado y progresivo**. Cada tema de jQuery se ilustra con un ejemplo, antes de pasar a practicar sobre aplicaciones más especializadas.

Esta **nueva edición del libro** coincide con una **evolución importante del framework jQuery**. La versión jQuery 2.0 se caracteriza por un tamaño reducido del API para acelerar la carga; **desaparecen numerosos métodos y se modifican funciones**. Los lectores que quieran iniciarse en jQuery empezarán con una librería remodelada cuyos principios seguirán siendo los mismos durante mucho tiempo. Para los desarrolladores que utilicen una versión más antigua de jQuery y que deseen migrar a esta nueva versión, el autor avisa a lo largo del libro de los problemas que se pueden encontrar en esta migración.

Después de la presentación del **framework**, el autor dedica un capítulo a los **selectores**, que no sólo muestran la diversidad de jQuery para acceder de manera sencilla a cualquier elemento de la página, sino que también son un concepto fundamental en el aprendizaje de jQuery. En los capítulos siguientes, el lector descubrirá los e**lementos interactivos** de jQuery; en primer lugar, cómo manipular los atributos (añadir, modificar o eliminar sobre la marcha) y después aplicar **hojas de estilo CSS**. Siguen los **eventos**clásicos de **JavaScript,** pero sobre todo de jQuery. También se presentan los **efectos visuales,** tan numerosos como originales y se detalla el estudio del **DOM** y de **AJAX visto por jQuery**. El capítulo final revisa alguno de los numerosos **plug-ins** desarrollados por la **comunidad jQuery,** que permiten añadir, en pocas líneas de código, efectos espectaculares.

Cuando termine de leer el libro, el lector podrá desarrollar aplicaciones web**más interactivas**, **ricas e innovadoras**, todo ello con una facilidad de escritura de JavaScript insospechada.

Los elementos adicionales se pueden descargar en esta página.

Los capítulos del libro:
Prefacio – Primeros pasos con jQuery – Los selectores en jQuery – Trabajar con los atributos y las propiedades – Trabajar con las hojas de estilo CSS – Los eventos – Los efectos – Recorrer el DOM – Trabajar con el DOM – Filtrar el DOM – AJAX en jQuery – Algunas utilidades – Los formularios – Los plug-ins jQuery

---

Luc VAN LANCKER

En los comienzos de Internet, **Luc Van Lancker**, se entusiasmó por la idea de la comunicación universal que vehiculaba este concepto y se dedicó completamente a este entorno. Es un formador apasionado, actualizado constantemente en las nuevas tecnologías relacionadas con el entorno Web y un gran pedagogo. Luc Van Lancker es autor de libros de Ediciones ENI sobre HTML 4.0, HTML5, XHTML, AJAX, jQuery UI, jQuery Mobile y Dojo en diferentes colecciones.

Introducción

JavaScript se consideró hace mucho tiempo un elemento accesorio en la creación de sitios Web. Sin embargo, desde la aparición de AJAX y la Web 2.0, la importancia de JavaScript en las aplicaciones Web ha crecido exponencialmente. La mejor muestra de esta importancia es la integración de los motores JavaScript en los navegadores actuales y la aparición de los JavaScript API en Html5.

Las evoluciones de JavaScript, que han pasado de la versión 1.1 a la 1.8, han hecho de JavaScript un lenguaje de programación cada vez más potente y, al mismo tiempo, más complejo. Han aparecido frameworks de JavaScript para facilitar la integración en las aplicaciones Web y el aumento de la productividad del desarrollo.

Gracias al enfoque innovador de su joven inventor, John Resig, jQuery replantea completamente el valor y la forma de escribir JavaScript. De hecho, era obligatorio tener en cuenta las evoluciones del DOM y de las hojas de estilo CSS, que aparecieron al mismo tiempo que JavaScript. También era importante volver a una sintaxis más intuitiva de JavaScript.

Muy rápidamente, un gran número de informáticos y sitios Web como Google, Nokia y Microsoft han adoptado jQuery. Ahora, esto se cumple incluso en la suite de herramientas de desarrollo de Visual Studio de Microsoft.

Muy respaldado por una comunidad dinámica, el crecimiento de jQuery es constante y rápido. De hecho, las diferentes versiones se suceden rápidamente, lo que hace crecer el impacto de jQuery con relación a sus competidores. Por tanto, podemos afirmar que jQuery se ha convertido en la referencia absoluta en materia de frameworks JavaScript.

En este contexto, parece evidente que es imprescindible un conocimiento profundo de Html (o Xhtml) y de las hojas de estilo CSS. También es importante tener buenas nociones de JavaScript. jQuery se dirige a un público que conoce estas diferentes técnicas.

En este libro vamos a examinar los diferentes temas que trata jQuery. El autor ha aplicado un enfoque estructurado y progresivo. Cada punto de jQuery se ilustra con un ejemplo, antes de pasar a las aplicaciones más importantes. Llamamos la atención del lector sobre el capítulo dedicado a los selectores (capítulo Los selectores en jQuery). No solo ilustra la diversidad de jQuery para trabajar de manera sencilla con cualquier elemento de la página, sino que es un elemento esencial en el aprendizaje de jQuery. Después se tratan los aspectos más interactivos, como los eventos, el uso de las hojas de estilo y del DOM, los efectos visuales, AJAX para jQuery y los plugins.

Esta segunda edición coincide con una importante modificación del framework jQuery. La versión jQuery 2.0 se caracteriza por un tamaño reducido de la API para acelerar la carga. Esta disminución del tamaño es posible gracias a la desaparición de numerosos métodos, así como por modificaciones más sutiles en otras funciones. Aquellos que se quieran iniciar en jQuery empezarán con una librería remodelada cuyos principios seguirán siendo los mismos durante mucho tiempo. No olvidamos a los desarrolladores que utilicen una versión más antigua de jQuery que deseen migrar a esta nueva versión, avisando de los problemas que se podrían encontrar en esta migración.

Cuando termine su aprendizaje, sus aplicaciones Web serán, sin ninguna duda, más interactivas, ricas e innovadoras, y todo ello con una facilidad y concisión en la escritura de JavaScript insospechadas.

El regreso de JavaScript

Las aplicaciones Web actuales se sitúan en un entorno dinámico. Nunca antes los sitios Web habían sido tan ricos en términos de presentación visual y otras funciones, basadas en la gestión de la información.

Desde el comienzo de la Web, JavaScript ha sido un compañero privilegiado en el diseño de las páginas Html, gracias a la interactividad que permitía añadir a estas páginas. Pero su presencia e influencia permanecieron mucho tiempo limitadas, debido sobre todo a las dificultades para hacer scripts verdaderamente compatibles con los diferentes navegadores de la época.

La aparición del DOM (*Document Object Model*), que permite acceder o actualizar el contenido, la estructura y el estilo de los documentos Html, fue la primera renovación de JavaScript. Más allá del DOM, la recomendación del W3C fue adoptada por todos los navegadores, lo que permitió reducir los problemas de interoperabilidad de los scripts.

Después llegó AJAX (*Asynchronous JavaScript and XML*) y las consultas XMLHttpRequest asociadas, que permitieron el nacimiento del JavaScript asíncrono, ofreciendo la posibilidad de modificar una parte de las páginas Web sin tener que volver a cargar la página completa. La puerta quedaba abierta a aplicaciones JavaScript mucho más ricas, que respondían mejor a aspectos relativos a la interactividad de las aplicaciones Web. Aquí también ganó la compatibilidad.

El concepto de la Web 2.0, con sus objetivos de mayor usabilidad y ergonomía, ha fortalecido, a su vez, la interactividad de las páginas y la demanda de aplicaciones ampliadas. Y así es como JavaScript se convierte en un elemento imprescindible en el desarrollo de las aplicaciones Web.

Seguramente, la mejor prueba del posicionamiento de JavaScript sea la aparición de nuevos motores JavaScript en los navegadores más actuales. Ya sea Google Chrome con su motor JavaScript Open Source V8, Opera con el proyecto Carakan, Safari en su versión 4 o Firefox 3.5 con TraceMonkey, todos buscan mejorar (y algunas veces de manera importante) el tratamiento de JavaScript. Internet Explorer, mucho tiempo rezagado con su versión 8, también ha adoptado un nuevo motor JavaScript llamado Chakra.

Presentación de jQuery

jQuery es un framework JavaScript libre y Open Source, del lado cliente, que se centra en la interacción entre el DOM, JavaScript, AJAX y Html. El objetivo de esta librería JavaScript es simplificar los comandos comunes de JavaScript. De hecho, el lema de jQuery es « Escribir menos para hacer más » (*Write less, do more*).

jQuery, al menos en su origen, es obra de una sola persona: John Resig. Este joven prodigio de JavaScript desarrolló la primera versión de jQuery en enero de 2006. En aquel momento tenía solo 20 años. Resig continúa siendo el motor de jQuery, pero ahora le ayuda una comunidad de apasionados.

Las especificaciones de jQuery son numerosas, pero lo principal es asegurar la flexibilidad que aporta para acceder a todos los elementos del documento Html a través de la multitud de selectores que existen (ver el capítulo Los selectores en jQuery). Esta característica se utilizó para dar nombre a este framework: j para JavaScript y Query para buscar o acceder a los elementos.

También podemos observar que esta librería jQuery está en constante evolución. Las actualizaciones y las nuevas versiones se suceden a un ritmo regular:

- Agosto de 2006: versión estable de jQuery 1.0.
- Enero de 2007: jQuery 1.1.
- Enero de 2009: jQuery 1.3.
- Enero de 2010: jQuery 1.4.
- Enero de 2011: jQuery 1.5.
- Mayo de 2011: jQuery 1.6.
- Marzo de 2012: jQuery 1.7.
- Noviembre de 2012: jQuery 1.8.
- Febrero de 2013: jQuery 1.9.
- Junio de 2013: jQuery 1.10 y jQuery 2.0.

Este libro se centra en las versiones jQuery 1.10 y 2.0.

Las grandes firmas de la Web y de la informática reconocen la calidad de jQuery. Por ejemplo, podemos mencionar, entre otros, a Google, Mozilla, Dell, IBM, WordPress, Nokia, Amazon y Twitter. Microsoft lo ha incorporado en su última versión de Visual Studio. Su crecimiento es rápido y supone una seria competencia para otros frameworks como Prototype, Dojo Toolkit y Scriptaculous, por citar algunos.

Los puntos fuertes de jQuery

El framework jQuery es cada vez más aceptado por los desarrolladores, ya que las aportaciones de este entorno son numerosas.

El enfoque de jQuery no consiste solo en una codificación de los scripts más intuitiva y concisa, sino que su filosofía es concentrarse en el conjunto de los elementos que gestiona el DOM. El JavaScript tradicional, en su evolución histórica, se ha tenido que acoplar al Document Object Model. De alguna manera, John Resig, con jQuery, ha reconsiderado el JavaScript manteniéndolo como un verdadero lenguaje de programación, basado en el DOM. Es esto lo que cambia totalmente la manera de aprender y escribir JavaScript.

jQuery permite acceder fácilmente a todos los elementos del DOM. Los métodos `getElementById`,`getElementsByName` y `getElementsByTagName` de JavaScript muestran pronto sus límites, especialmente cuando el diseñador quiere acceder a los atributos y otras propiedades de estilo. jQuery da acceso de manera muy sencilla y sobre todo intuitiva a todos los elementos del documento. El siguiente capítulo, dedicado a los selectores, ilustrará la diversidad que añade jQuery en este aspecto.

El enfoque de jQuery es completo. Los métodos y funciones de jQuery no se limitan a algunas animaciones de tipo estético. Con pocas líneas de código, jQuery puede modificar texto, insertar imágenes, ordenar tablas o reorganizar por completo la estructura del documento Html. Este framework aporta una nueva visión de JavaScript y, después de un corto aprendizaje, simplifica mucho el diseño y la escritura de scripts. Durante todo el libro se remarcará el código tan conciso que se genera.

El código jQuery es compatible con los diferentes navegadores. La desviación de los navegadores, pequeña o más pronunciada, en relación con los estándares del DOM y de las hojas de estilo, es una lamentable realidad en el desarrollo de aplicaciones Web evolucionadas. Gracias a la interfaz de software que añade jQuery, el código de las aplicaciones se vuelve compatible con los principales navegadores del mercado. Hay que rebuscar por las profundidades de los foros especializados para encontrar algunos conflictos, relacionados habitualmente con detalles poco utilizados. Esta compatibilidad se centra, en particular, en los elementos de las hojas de estilo CSS3 que, de momento, los navegadores solo soportan de manera muy fragmentada. Como ejemplo, podemos mencionar la opacidad de los elementos. Los métodos jQuery `fadeIn()`, `fadeOut()` y `fadeTo()` permiten modificar esta opacidad de manera compatible con todos los navegadores.

jQuery se sustenta gracias a una comunidad dinámica de desarrolladores. Esta comunidad, basada en los principios históricos de pasión y compartición de Internet, proporciona una gran variedad de plug-ins, es decir, extensiones de jQuery, dedicadas a tareas concretas. Estos plug-ins, a menudo maravillas de programación, están disponibles libremente en la Web y los diseñadores de los sitios Web los tienen en cuenta. Un carrusel de imágenes o una tabla ordenable implementada en pocos minutos y pocas líneas de código simplifica mucho el trabajo de los desarrolladores.

jQuery 2.0 o jQuery 1.10

Por el momento, jQuery ofrece en paralelo dos versiones, la 1.11 y la 2.1. Es importante conocer las diferencias.

Versión principal

Las dos versiones marcan una gran evolución en las funcionalidades de jQuery. De hecho, la comunidad jQuery ha señalado que la API iba subiendo de tamaño en kb poco a poco pero de manera importante y que era momento de iniciar una "dieta". Este exceso de peso (relativo) ralentiza la carga de la página, especialmente, en la llamada de jQuery por CDN (que verá más adelante en este capítulo).

Esta reducción de peso pasa por la eliminación de una serie de métodos como `live()`, `die()`, `load()`,`unload()`, `size()`, `browser()`, etc. así como por modificaciones más sutiles en otros métodos como`ajaxSend()` por citar alguno.

Si estos cambios no deberían molestar a los nuevos usuarios, existe el riesgo de dar algún problema a los desarrolladores que utilicen versiones anteriores de jQuery que quieran migrar a estas nuevas versiones. jQuery ofrece el plugin Migrate para simplificar la transición desde versiones anteriores. Este plugin en su versión comprimida (*compressed*) restaura las funcionalidades obsoletas de manera que el código antiguo continúa funcionando correctamente. Se genera una versión descomprimida (*uncompressed*) cuando tiene mensajes en la consola para identificar los problemas vinculados a esta migración.

jQuery 2.0

Esta versión es la más compacta con 81,6 kb. Para su información, el peso de la versión 1.7 era de 92,4 kb, un 12% más. Pero esta versión 2.0 de jQuery no tiene en cuenta Internet Explorer 6, 7 y 8. Esto se justifica por la baja utilización de las hojas de estilo CSS modernas en estas versiones.

Esta situación es delicada para los desarrolladores. Si su aplicación o sitio web tiene que ser compatible para todo el mundo, se privará de alrededor del 40% de los usuarios de Internet Explorer en la red. Sin embargo, en los próximos años, la parte del mercado de estas versiones excluidas irá disminuyendo y este inconveniente será menos importante.

jQuery 1.10

Las funcionalidades de la versión 1.10 son idénticas a las de la versión 2.0, excepto aquellas que soportan todavía las versiones 6, 7 y 8 de Internet Explorer.

Esta mayor compatibilidad comporta un aumento de peso bastante sensible con sus 94,1 kb.

La comunidad jQuery afirma que las versiones 1.x tendrán los mismos desarrollos que las versiones 2.x. Pero según el autor, hay que ser realista. Cuando estas versiones de Internet Explorer sean más frágiles, se puede prever que se abandonará la serie de las versiones 1.x.

Aplicación de jQuery

Por CDN

Numerosos CDN, servidores que proporcionan datos de manera ultrarrápida, albergan la librería jQuery. Basta simplemente con llamarla mediante la etiqueta `<script>`. Gracias al tamaño reducido de jQuery, su carga es del orden de 30 milisegundos.

Los CDN son jQuery propiamente (a través de MaxCDN), Google, Microsoft y CDNJS.

jQuery

```
<script src="http://code.jquery.com/jquery-número de versión.min.js">
</script>
```

Google

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/número
de versión/jquery.min.js"></script>
```

Microsoft

```
<script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-número
de versión.min.js"></script>
```

CDNJS

```
<script src=http://cdnjs.cloudflare.com/ajax/libs/jquery/número
de versión/jquery.min.js></script>
```

Ejemplos

- <script src="http://code.jquery.com/jquery-2.0.0.min.js"></script>
- <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.10.0/ jquery.min.js"></script>
- <script src="http://ajax.aspnetcdn.com/ajax/jQuery/ jquery-1.10.0.min.js"></script>
- <script src=http://cdnjs.cloudflare.com/ajax/libs/jquery/2.0.0/ jquery.min.js></script>

Esta llamada mediante CDN se debe colocar entre las etiquetas `<head>...</head>`.

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="http://code.jquery.com/jquery-1.10.0.js"></script>
...
</head>
...
```

Además de la rapidez de los CDN, la librería jQuery quizá ya está presente en la caché del navegador del usuario, lo que acelera todavía más su carga.

En local

En primer lugar, hay que descargar la librería jQuery. La dirección la más indicada para hacerlo es la del

sitio Web jQuery en la dirección: http://jquery.com/download/

En esta página, se le permite elegir la versión 1.x o 2.x. Para cada versión, hay una versión comprimida llamada de producción (*production*) indicada para cuando su aplicación esté en línea y una versión no comprimida de desarrollo más voluminosa (*development*). Esta última contiene numerosos comentarios y permitirá a los más curiosos echar un vistazo al código JavaScript que usa jQuery.

En el momento de escribir este libro, el archivo que se descarga se llama `jquery-1.11.0.min.js`o `jquery-2.1.0.min.js`. En adelante, lo llamaremos `jquery.js`. Después de descargar este archivo, tendrá que ubicarlo en el mismo directorio que la página Html5.

La llamada a la librería se debe colocar entre las etiquetas `<head>...</head>`.

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
...
</head>
...
```

Iniciar un script jQuery

Una vez que el framework se ha configurado y está preparado para utilizarse, es preciso, antes que nada, examinar algunas nociones teóricas.

Todas las instrucciones o, para ser más precisos, todos los scripts jQuery, se articulan alrededor de un modelo único. Éste es:

```
jQuery(function(){
//contenido que se ejecuta cuando se carga el documento
});
```

Para ahorrar pulsaciones del teclado, se utiliza casi siempre el signo dólar ($), que funciona como alias de jQuery.

```
$(function(){
//contenido externo que se ejecuta cuando se cargue el documento
});
```

Haciendo referencia a este modelo, todo script jQuery empieza por:

```
<script>
$(document).ready(function() {
// código jQuery
});
</script>
```

Es decir, crear un objeto jQuery (`$`) a partir del documento (`document`), cuando esté preparado (`ready`).

La particularidad de esta función es que carga los elementos del DOM tan pronto como estén disponibles, es decir, antes de la carga completa de la página.

Esta función de jQuery le diferencia del JavaScript clásico. Por ejemplo, éste usa el clásico`window.onload = function()`, que espera a que la página y todos los elementos que contiene estén completamente cargados. Esto puede ser muy largo, especialmente cuando hay imágenes con un tamaño importante. Es una particularidad esencial de jQuery que se basa, recordemos, de manera nativa en los elementos del DOM.

Este modo de funcionamiento presenta muchas ventajas:

- Todos los elementos de la página están incluidos en un objeto que los selectores, los métodos y las funciones de jQuery pueden manejar.
- El código Html está desprovisto de cualquier mención JavaScript, como por ejemplo las notaciones `<a href="javascript:void(0);">enlace </a>`. Todo el código JavaScript/jQuery se sitúa en una parte separada de la página Html (entre las etiquetas`<head>...</head>`) o en un archivo js externo, lo que respeta completamente el principio de separación del contenido y de la presentación.

Con `$(document).ready()`, los elementos de la página están a disposición del desarrollador, antes de la carga completa y su visualización. Esto es muy práctico para activar los efectos del navegador de aparición o desaparición de la página.

En la Web se encuentran scripts jQuery que comienzan con la escritura abreviada:

```
$(function () {
// código jQuery
});
```

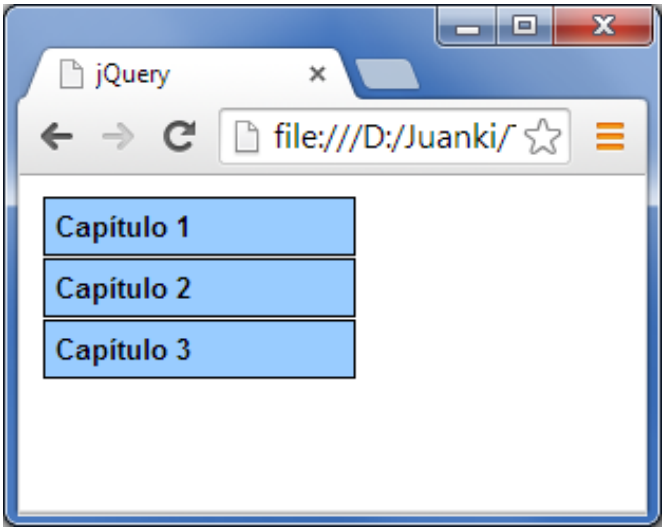
A lo largo del libro, vamos a adoptar la instrucción `$(document).ready()`, más académica y expresiva.

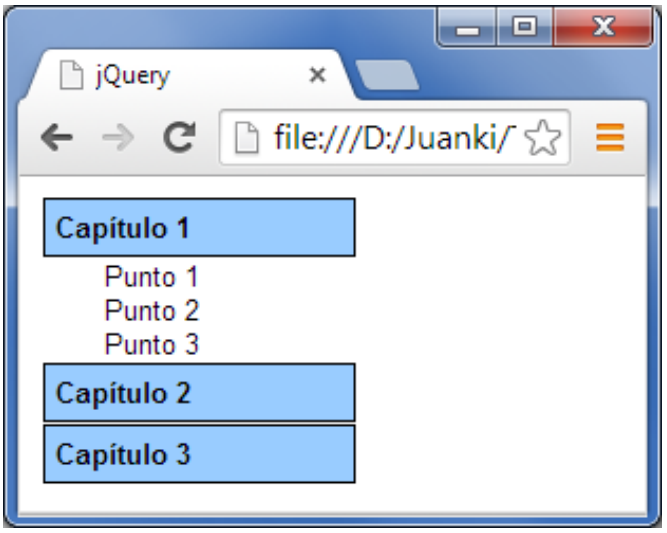
El signo $ también es usado por otros frameworks, como por ejemplo Prototype. El método`jquery.noconflict` (ver el capítulo Algunas utilidades - Evitar los conflictos) permite evitar los conflictos en la llamada del alias $ con otra librería que también use esta nomenclatura para una de sus funciones.

La primera aplicación jQuery

Para entrar en materia, vamos a hacer un menú de navegación vertical desplegable.

Aquí el objetivo no es entender los métodos jQuery, que se estudiarán más adelantes, sino tener una visión general de la implantación y organización general de los scripts jQuery. El objetivo de esta primera aplicación es familiarizarse con el enfoque que sigue el autor, en nuestra exploración de jQuery.

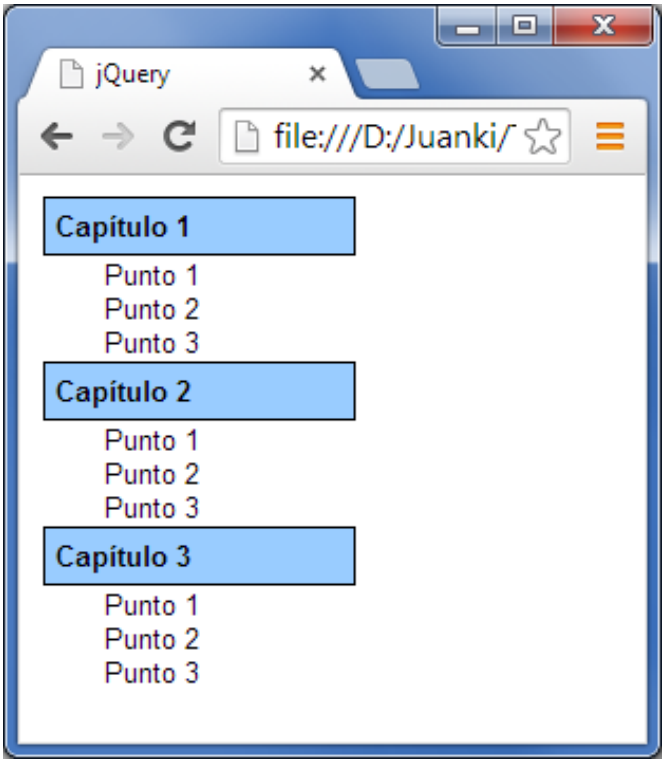




Se presenta de la siguiente manera el archivo Html de inicio:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta ="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("div.menu_punto").hide();
$("p.menu_capitulo").click(function(){
$(this).next("div.menu_punto").slideDown(300)
.siblings("div.menu_punto").slideUp("slow");
});
});
</script>
<style>
```

```
body  { margin: 10px;
        font: 0.8em Arial, Helvetica, sans-serif;}
.menu { width: 150px;}
.menu_capitulo { padding: 5px;
                 cursor: pointer;
                 position: relative;
                 margin:1px;
                 font-weight:bold;
                 background: #9cf;
                 border: 1px solid black;}
.menu_punto a { display:block;
                color:black;
                background-color:white;
                padding-left:30px;
                text-decoration:none;}
.menu_punto a:hover { color: black;
                      text-decoration:underline;}
</style>
</head>
<body>
<div>
<div class="menu">
<p class="menu_capitulo">Capítulo 1</p>
<div class="menu_punto">
<a href="#">Punto 1</a>
<a href="#">Punto 2</a>
<a href="#">Punto 3</a>
</div>
<p class="menu_capitulo">Capítulo 2</p>
<div class="menu_punto">
<a href="#">Punto 1</a>
<a href="#">Punto 2</a>
<a href="#">Punto 3</a>
</div>
<p class="menu_capitulo">Capítulo 3</p>
<div class="menu_punto">
<a href="#">Punto 1</a>
<a href="#">Punto 2</a>
<a href="#">Punto 3</a>
</div>
</div>
</div>
</body>
</html>
```



Ahora vamos a construir el script jQuery paso a paso.

Suponemos que la librería jquery.js se ha descargado e incorporado en el directorio que contiene el archivo Html anterior.

Llamamos a jQuery, poniendo entre las etiquetas `<head>` y `</head>` la siguiente etiqueta de encabezado.

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

El script puede empezar.

```
<script>
$(document).ready(function(){
...
});
</script>
```

Con esta primera instrucción jQuery, se carga el árbol del DOM relativo al documento en un objeto jQuery.

Hay que hacer una primera operación que consiste en no visualizar solo los submenús, es decir, las divisiones cuya clase es `menu_punto`.

```
<script>
$(document).ready(function(){
$("div.menu_punto").hide();
...
});
</script>
```

De esta manera, jQuery (`$`) selecciona estas divisiones `<div>`, cuya clase es `menu_punto(("div.menu_punto"))` y las oculta (`hide()`).

Pasemos ahora a la parte que se encarga de desplegar el submenú cuando se pulsa uno de los capítulos.

```
<script>
$(document).ready(function(){
```

```
$("div.menu_punto").hide();
$("p.menu_capitulo").click(function(){
$(this).next("div.menu_punto").slideDown(300)
.siblings("div.menu_punto").slideUp("slow");
});
});
```

Detallemos esta parte:

```
$("p.menu_capitulo").click(function(){
```

Hacer clic en uno de los párrafos cuya clase es `menu_capitulo` (de manera codificada`$("p.menu_capitulo")`), genera una función que se detallará en las siguientes líneas (`click(function()`).

```
$(this).next("div.menu_punto").slideDown(300)
```

A partir de este elemento (`this`), el script busca el elemento `<div>` que le sigue y que tiene la clase`menu_punto` (es decir, `next("div.menu_punto")`). Después hacemos aparecer esta división con un efecto de deslizamiento hacia abajo (`slideDown(300)`).

```
.siblings("div.menu_punto").slideUp("slow");
```

Hay que cerrar las divisiones abiertas de los otros capítulos. Se pide a jQuery seleccionar los hermanos inmediatos (`siblings`) de cada división `<div>` que tenga una clase `menu_punto`(`siblings("div.menu_punto")`) y cerrar estas divisiones usando un efecto de deslizamiento hacia arriba (`slideUp("slow")`).

Observe que se han codificado muchas acciones en una misma instrucción. Estas diferentes operaciones se unen simplemente por un punto.

```
});
```

Para terminar esta función relativa a un clic en un párrafo.

```
});
```

Fin del ready y fin de script.

Observe cómo de conciso es el código jQuery. Unas pocas líneas son suficientes para desarrollar este script.

El archivo Html completo se presenta de la siguiente manera:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("div.menu_punto").hide();
$("p.menu_capitulo").click(function(){
$(this).next("div.menu_punto").slideDown(300)
.siblings("div.menu_punto").slideUp("slow");
});
});
</script>
<style>
body { margin: 10px;
       font: 0.8em Arial, Helvetica, sans-serif;}
```

```
.menu { width: 150px;}
.menu_capitulo { padding: 5px;
                 cursor: pointer;
                 position: relative;
                 margin:1px;
                 font-weight:bold;
                 background: #9cf;
                 border: 1px solid black;}
.menu_punto a { display:block;
                color:black;
                background-color:white;
                padding-left:30px;
                text-decoration:none;}
.menu_punto a:hover { color: black;
                      text-decoration:underline;}
</style>
</head>
<body>
<div>
<div class="menu">
<p class="menu_capitulo">Capítulo 1</p>
<div class="menu_punto">
<a href="#">Punto 1</a>
<a href="#">Punto 2</a>
<a href="#">Punto 3</a>
</div>
<p class="menu_capitulo">Capítulo 2</p>
<div class="menu_punto">
<a href="#">Punto 1</a>
<a href="#">Punto 2</a>
<a href="#">Punto 3</a>
</div>
<p class="menu_capitulo">Capítulo 3</p>
<div class="menu_punto">
<a href="#">Punto 1</a>
<a href="#">Punto 2</a>
<a href="#">Punto 3</a>
</div>
</div>
</div>
</body>
</html>
```

La documentación de jQuery

La documentación de jQuery es abundante en la Web, aunque la mayor parte está en inglés.

La referencia es la documentación de jQuery en línea, muy completa e ilustrada con ejemplos, disponible en el sitio oficial: http://api.jquery.com/

El sitio w3schools ofrece siempre tutoriales de gran calidad. El relativo a jQuery se puede consultar en la url: http://w3schools.com/jquery/

Encontrará una referencia rápida muy valiosa en la url http://oscarotero.com/jquery. Este sitio reenvía a la documentación oficial para el detalle de los diferentes métodos.

Por último citaremos http://jqapi.com que se ofrece como una alternativa a la documentación oficial.

Para una documentación completa en castellano, puede dirigirse al sitio: http://librojquery.com/

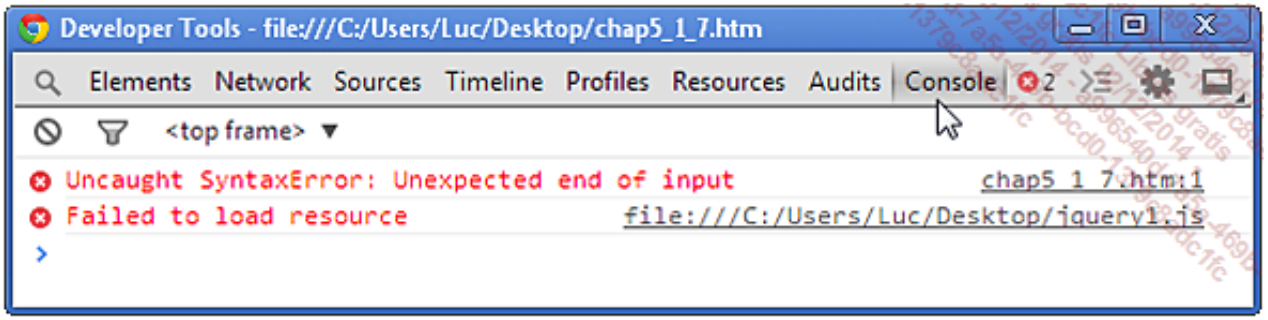
La documentación y los tutoriales relativos a jQuery, no siempre están actualizados a la última versión del framework. En caso de duda, la documentación oficial de jQuery (http://api.jquery.com) es la referencia.

Herramientas de desarrollo y de depuración

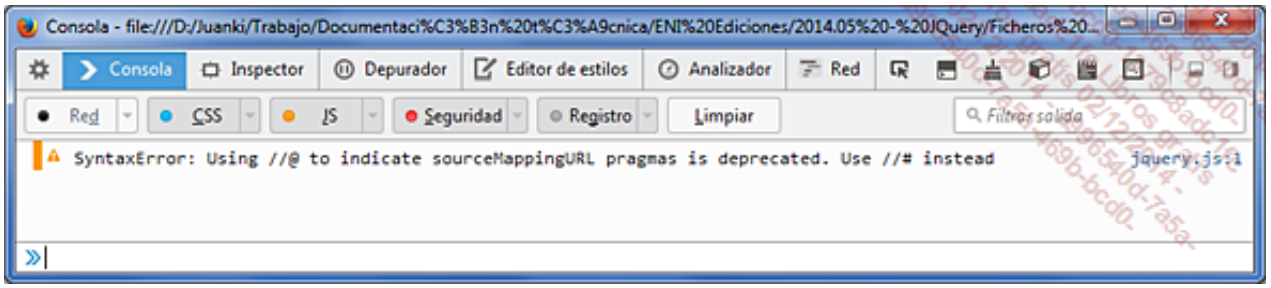
Recordemos que jQuery es un framework del lado cliente. Esto significa que solo necesita un número reducido de herramientas; un navegador y un editor de texto. Sin embargo, para los scripts AJAX (ver el capítulo AJAX en jQuery), hay que prever la instalación de un servidor Web local, pero ya trataremos este tema cuando estudiemos este capítulo.

A efectos de depuración, no existe una solución milagrosa para depurar el código JavaScript. Sin embargo, los navegadores (Google Chrome, Firefox, Internet Explorer) proponen soluciones innovadoras que permiten inspeccionar y depurar las páginas Web y su entorno, es decir, el HTML, las hojas CSS, el DOM, la utilización de la red, los scripts, etc. En el caso del aprendizaje de jQuery, la herramienta **Consola** de estas herramientas de desarrollo será particularmente valiosa para arreglar los errores de código de sus primeras pruebas.

En Google Chrome, se puede acceder a estas herramientas de desarrollo en **Herramientas - Herramientas para desarrolladores** o con el atajo de teclado [Ctrl][Mayús] **I**. También se pueden abrir con un clic derecho en un elemento de la página y seleccionando la opción **Inspeccionar elemento**. Hay un tutorial completo en la url: https://developers.google.com/chrome-developer-tools/



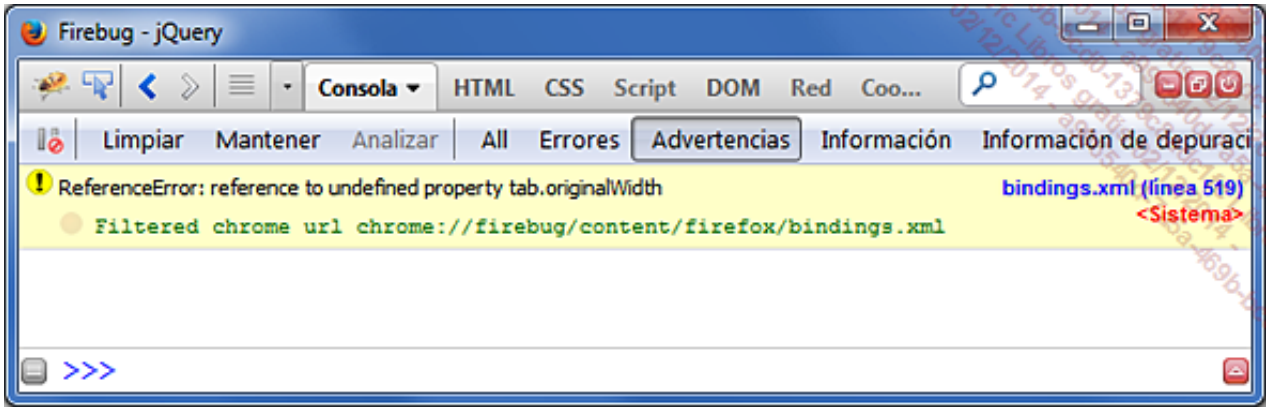
En sus versiones recientes, Firefox ofrece de forma nativa herramientas similares, **Desarrollador - Desarrollador Web**. Se ofrece un tutorial en la dirección: https://developer.mozilla.org/es/docs/Tools



También para Firefox, existe la extensión Firebug, bien conocida por los desarrolladores. No deje de consultar el excelente tutorial en las direcciones:

http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=FireBug

http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=firebugAvanzado



Internet Explorer ha renovado completamente sus herramientas de desarrollo a partir de su versión 11. Se puede acceder a ellas desde el atajo de teclado [F12]. Si su estética es particularmente atractiva, su utilización es menos intuitiva. Se puede consultar un tutorial enhttp://msdn.microsoft.com/es-es/library/ie/bg182636%28v=vs.85%29.aspx. Observe que estas herramientas permiten emular navegadores, tamaños de pantalla y ubicaciones GPS.

Introducción

Los selectores son uno de los aspectos más importantes de la librería jQuery. Utilizan una sintaxis que recuerda a la de las hojas de estilo CSS. Permiten a los diseñadores identificar de forma rápida y sencilla cualquier elemento de la página y aplicar métodos concretos de jQuery.

Entender correctamente qué son los selectores jQuery es un elemento clave para entender cómo usar jQuery.

Hay un sitio Web que ilustra de maravilla el papel de estos selectores:www.codylindley.com/jqueryselectors/

Los selectores básicos

Estos selectores básicos de jQuery no son más que una reformulación de los métodos`getElementById`, `getElementsByClassName` y `getElementsByTagName` de JavaScript tradicional.

La notación de jQuery tiene la ventaja de ser más concisa y mucho más intuitiva.

Estos selectores básicos se usan muy a menudo en los scripts jQuery. Asimilarlos correctamente es imprescindible para la buena comprensión de los múltiples ejemplos de este libro.

1. Selección por identificador

#identificador

Selecciona el elemento (único) cuyo identificador es `id="identificador"`.

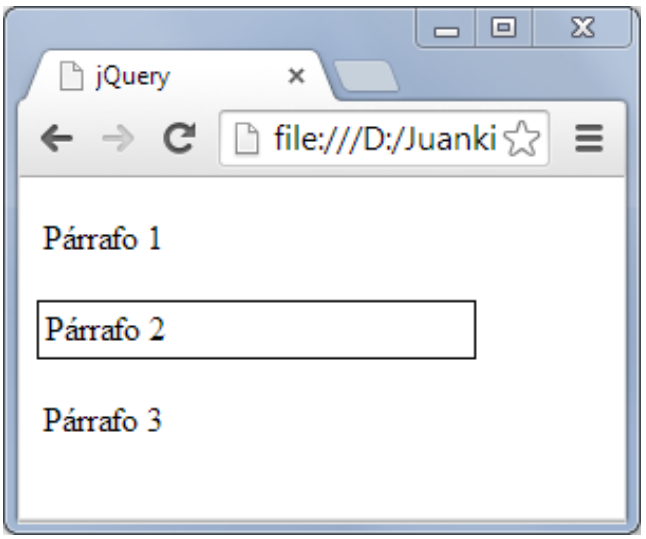
`$("#box")`: selecciona el elemento cuyo id es box.

Es la notación jQuery de `getElementById` en JavaScript clásico.

Recordemos que este identificador debe ser único en la página. Si, por error, no es el caso, jQuery toma el primer elemento con este identificador.

Ejemplo

Rodeamos con un borde el segundo párrafo, cuyo identificador es "dos".



Partiendo del siguiente archivo Html:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js" ></script>
<script>
$(document).ready(function(){
$("#dos").css("border","1px solid black");
});
</script>
<style>
p { width: 200px;
    padding: 3px;}
```

```
</style>
</head>
<body>
<p>Párrafo 1</p>
<p id="dos">Párrafo 2</p>
<p>Párrafo 3</p>
</body>
</html>
```

Vamos a explicar el script jQuery.

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

Inicialización de jQuery. El script está preparado para operar tan pronto como se cargue el DOM.

`$("#dos").css("border","1px solid black");`

Al elemento con identificador "dos" (`$("#dos")`), se aplica el método jQuery que permite modificar las propiedades CSS, en este caso añadir un borde (`css("border","1px solid black")`). Este método jQuery `css()` se estudiará con más detalle en el capítulo Trabajar con las hojas de estilo CSS.

`});`

Fin del script.

2. Selección por nombre de la etiqueta

elemento

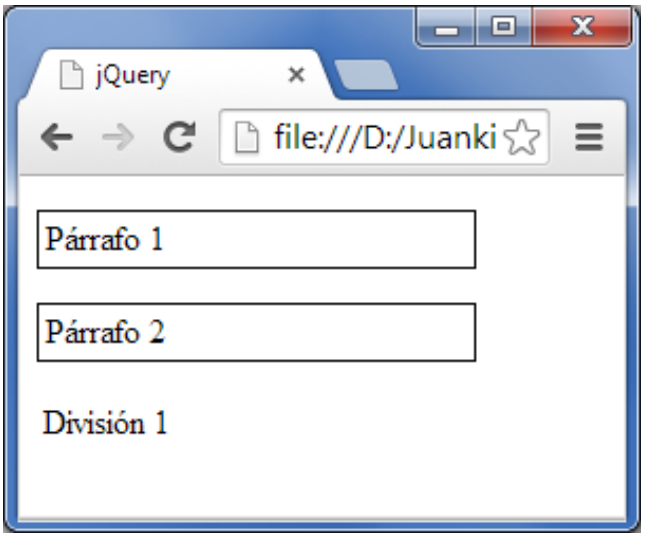
Selecciona todos los elementos (o etiquetas) con el nombre especificado.

`$("div")`: selecciona todas las capas `<div>` de la página.

Es la notación jQuery de `getElementsByTagName` en JavaScript clásico.

Ejemplo

Rodeamos con un borde todos los párrafos del documento Html.




```
<!doctype html>
```

```
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("p").css("border","1px solid black");
});
</script>
<style>
p { width: 200px;
    padding: 3px;}
div { padding: 3px;}
</style>
</head>
<body>
<p>Párrafo 1</p>
<p>Párrafo 2</p>
<div>División 1</div>
</body>
</html>
```

`$("p").css("border","1px solid black");`

jQuery selecciona todas las etiquetas de párrafo `<p>` (`$("p")`) y aplica sobre ellas un borde, usando el método `css()`.

3. Selección por clase

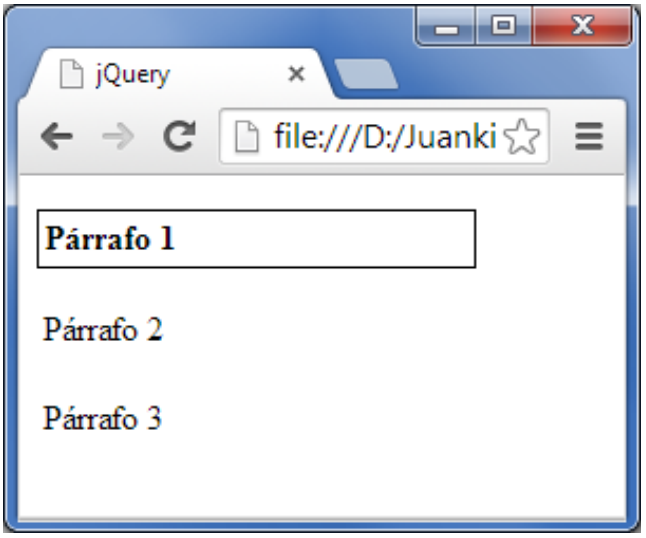
.clase

Selecciona todos los elementos (o etiquetas) con la clase especificada.

`$(".texto")`: selecciona todos los elementos con el atributo `class="texto"`.

Ejemplo

Rodeamos con un borde el párrafo que tiene la clase `negrita`.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
```

```
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$(".negrita").css("border","1px solid black");
});
</script>
<style>
p { width: 200px;
    padding: 3px;}
.negrita { font-weight: bold; }
</style>
</head>
<body>
<p class="negrita">Párrafo 1</p>
<p>Párrafo 2</p>
<p>Párrafo 3</p>
</body>
</html>
```

`$(".negrita").css("border","1px solid black");`

jQuery selecciona la etiqueta con la clase `negrita` con (`$(".negrita")`). Después se le aplica un borde.

Comentarios

Podríamos haber escrito: `$("p.negrita").css("border","1px solid black")`. De esta manera, jQuery selecciona las etiquetas `<p>` con la clase `negrita`.

Según los especialistas, esta notación sería más eficaz, ya que jQuery puede encontrar directamente las etiquetas `<p>` en el DOM y después filtrar las que tengan una clase `negrita`.

La notación que usa varias clases es intuitiva. `$(".clase1.clase2")` selecciona todos los elementos que tienen la clase `clase1` o `clase2`.

El selector estrella * permite seleccionar todos los elementos.

```
$("*").css("border","1px solid black");
```

jQuery permite asociar varios selectores.

```
$("div,span,p.nombre_clase").css("border","1px solid black");
```

Los selectores jerárquicos

La notación DOM con sus padres, descendientes, hijos y hermanos (siblings) está muy relacionada con la escritura de JavaScript. La librería jQuery no podía ignorar esta manera de proceder.

1. Selección de los descendientes

Ascendiente Descendiente

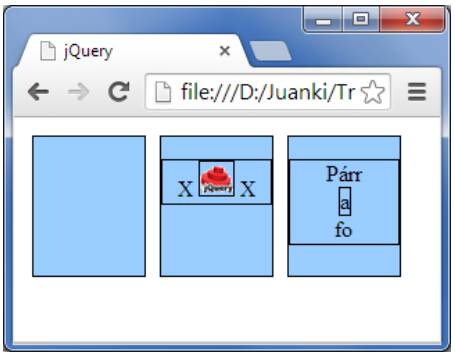
Selecciona todos los descendientes del elemento "Descendiente" con respecto al elemento padre "Ascendiente".

`$("#box p")`: selecciona todos los descendientes de `<p>` contenidos en el elemento padre identificado por `box`.

Los descendientes pueden ser los hijos o los nietos, a cualquier nivel.

Ejemplo

Estudiamos las capas que contienen varios elementos. Buscaremos todos los descendientes del elemento identificado por `id="box"`.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script
$(document).ready(function(){
$("#box *").css("border", "1px double black");
});
</script>
<style>
span#box { display: block;}
div { width: 80px;
      height: 100px;
      margin: 5px;
      float: left;
      background: #9cf; }
p { text-align: center;}
</style>
</head>
<body>
<span id="box">
<div></div>
<div><p>X <img src="jquery.png" alt=""> X</p></div>
<div><p>Párr<br><span>a</span><br>fo</p></div>
</span>
</body>
</html>
```

`$("#box *").css("border", "1px double black");`

El script jQuery selecciona todos los descendientes del elemento con identificador box (`$("#box *")`) y los rodea con un borde.

Observe que se rodean con un borde no solo los hijos, es decir, las capas `<div>`, sino también los nietos, es decir, las etiquetas `<p>` y los biznietos, es decir, las etiquetas `<img>` y `<span>`.

2. Selección de los hijos

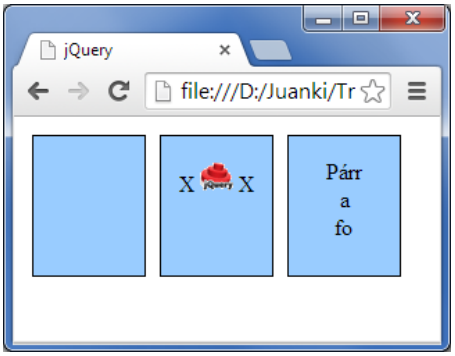
Padre > Hijo

Selecciona todos los elementos llamados "Hijo" que son hijos directos del elemento padre llamado "Padre".

`$("#box > p")`: selecciona todos los hijos inmediatos de `<p>`, contenidos en el elemento padre identificado por `box`.

Ejemplo

Volvemos al ejemplo anterior. Buscamos los hijos (y solo los hijos) del elemento identificado por`id="box"`.



```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#box > *").css("border", "1px double black");
});
</script>
<style>
span#box { display: block;}
div { width: 80px;
      height: 100px;
      margin: 5px;
      float: left;
      background: #9cf; }
p { text-align: center;}
</style>
</head>
<body>
<span id="box">
<div></div>
<div><p>X <img src="jquery.png" alt=""> X</p></div>
<div><p>Párr<br><span>a</span><br>fo</p></div>
</span>
</body>
</html>
```

```
$("#box > *").css("border", "1px double black");
```

El script jQuery selecciona todos los hijos directos del elemento con identificador box (`$("#box > *")`) y los rodea con un borde.

Observe que los nietos `<p>` y los biznietos `<img>` o `<span>` no se rodean con un borde.

Podríamos haber escrito:

```
$("#box > div").css("border", "1px double black");
```

3. Selección de los hermanos siguientes

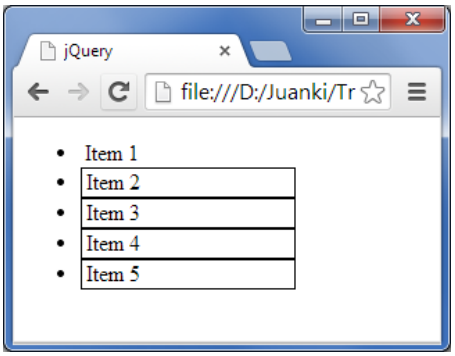
Predecesor ~ Hermano

Obtiene los elementos hermanos situados después del elemento identificado por el selector "Predecesor" y que responden al selector "Hermano".

`$("#prev ~ div")` encuentra todas las capas `<div>` hermanas, después del elemento con identificador `#prev`.

Ejemplo

Partamos de una lista no ordenada. Buscamos los elementos hermanos del primer ítem de la lista.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script
$(document).ready(function(){
$("li.uno ~ li").css("border", "1px double black");
});
</script>
<style>
li { width: 150px;
     padding-left: 3px;}
</style>
</head>
<body>
<ul>
<li class="uno">Item 1</li>
<li class="dos">Item 2</li>
<li class="tres">Item 3</li>
<li class="cuatro">Item 4</li>
<li class="cinco">Item 5</li>
</ul>
</body>
</html>
```

```
$("li.uno ~ li").css("border", "1px double black");
```

El script selecciona todos los hermanos del elemento lista `<li>` con la clase `uno` (`$("li.uno ~ li")`). El resto de los ítems de la lista se dejan.

4. Selección del elemento siguiente

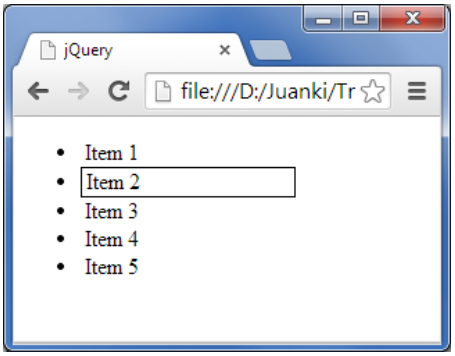
Predecesor + Siguiente

Obtiene el elemento inmediatamente siguiente, situado después del elemento identificado por el selector "Predecesor" y que responde al selector "Siguiente".

`$("#prev + div")` encuentra la capa `<div>` que sigue al elemento con identificador `#prev`.

Ejemplo:

Partamos de una lista no ordenada. Buscamos el elemento siguiente al primer ítem de la lista.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("li.uno + li").css("border", "1px double black");
});
</script>
<style>
li { width: 150px;
     padding-left: 3px;}
</style>
</head>
<body>
<ul>
<li class="uno">Item 1</li>
<li class="dos">Item 2</li>
<li class="tres">Item 3</li>
<li class="cuatro">Item 4</li>
<li class="cinco">Item 5</li>
</ul>
</body>
</html>
```

`$("li.uno + li").css("border", "1px double black");`

Encuentra el hermano inmediatamente siguiente, entre los hermanos del elemento de lista `<li>`, con la clase `uno` (`$("li.uno + li")`).

Los filtros jQuery básicos

jQuery enumera todos los elementos del DOM, por lo que se convierte en una tarea sencilla filtrar determinados elementos como el primero, el último, etc.

1. El primer elemento

:first

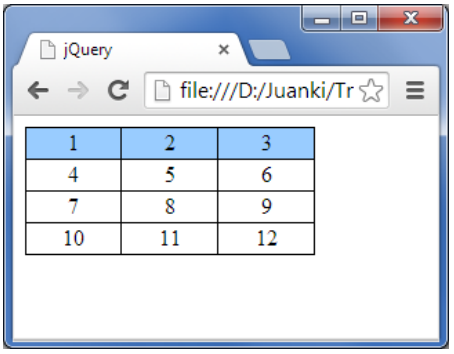
Selecciona la primera instancia de un elemento.

`$("li:first")`: selecciona el primer elemento de lista `<li>` del docu-mento.

El selector `:first` selecciona un único elemento, mientras que `:first-child` (ver Los filtros hijos - El primer hijo en este capítulo), puede seleccionar varios, es decir, uno para cada padre.

Ejemplo

Cambiemos el color de fondo a la primera línea de una tabla.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("tr:first").css("background", "#9cf");
});
</script>
<style>
table { width: 210px;
        border-collapse: collapse;
        border: 1px solid black;}
td { text-align: center;
     border: 1px solid black;}
</style>
</head>
<body>
<table>
<tr><td>1</td><td>2</td><td>3</td></tr>
<tr><td>4</td><td>5</td><td>6</td></tr>
<tr><td>7</td><td>8</td><td>9</td></tr>
<tr><td>10</td><td>11</td><td>12</td></tr>
</table>
</body>
</html>
```

`$("tr:first").css("background", "#9cf");`

`$("tr:first")` selecciona la primera etiqueta `<tr>`.

2. El último elemento

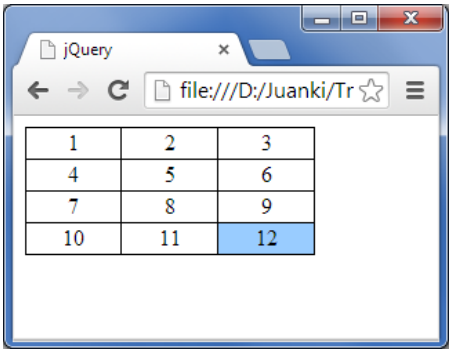
:last

Selecciona la última instancia de un elemento.

`$("li:last")`: selecciona el último elemento de lista `<li>` del documento.

Ejemplo:

Cambiemos el color de fondo a la última celda de una tabla.



```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("td:last").css("background", "#9cf");
});
</script>
<style>
table { width: 210px;
        border-collapse: collapse;
        border: 1px solid black;}
td { text-align: center;
     border: 1px solid black;}
</style>
</head>
<body>
<table>
<tr><td>1</td><td>2</td><td>3</td></tr>
<tr><td>4</td><td>5</td><td>6</td></tr>
<tr><td>7</td><td>8</td><td>9</td></tr>
<tr><td>10</td><td>11</td><td>12</td></tr>
</table>
</body>
</html>
```

```
$("td:last").css("background", "#9cf");
```

`$("td:last")`: el script selecciona la última etiqueta `<td>`.

3. Los elementos pares

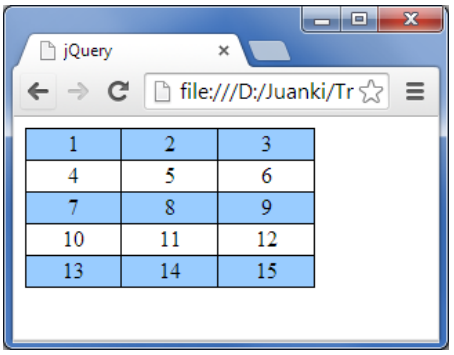
:even

Selecciona los elementos pares según un índice que empieza por 0.

`$("tr:even")`: selecciona las líneas con índice JavaScript 0, 2, 4, es decir, las líneas 1, 3, 5 de la pantalla.

Ejemplo

Aplicamos un efecto de listing a una tabla, poniendo una línea por cada dos, con un color de fondo.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("tr:even").css("background", "#9cf");
});
</script>
<style>
table { width: 210px;
        border-collapse: collapse;
        border: 1px solid black;}
td { text-align: center;
     border: 1px solid black;}
</style>
</head>
<body>
<table>
<tr><td>1</td><td>2</td><td>3</td></tr>
<tr><td>4</td><td>5</td><td>6</td></tr>
<tr><td>7</td><td>8</td><td>9</td></tr>
<tr><td>10</td><td>11</td><td>12</td></tr>
<tr><td>13</td><td>14</td><td>15</td></tr>
</table>
</body>
</html>
```

```
$("tr:even").css("background", "#9cf");
```

jQuery selecciona las etiquetas `<tr>` pares y les pone un color de fondo.

Este efecto, que necesita muchas líneas de código en el JavaScript clásico, se programa en jQuery con una sola línea. Este ejemplo muestra cómo de conciso es el código que se crea en jQuery.

4. Los elementos impares

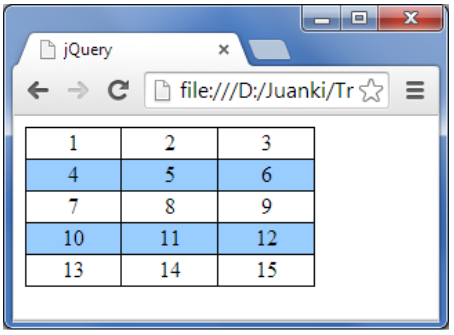
:odd

Selecciona los elementos impares según un índice que empieza por 0.

`$("tr:odd")`: selecciona las celdas con índice JavaScript 1, 3, 5, es decir las celdas 2, 4, 6... de la pantalla.

Ejemplo

Aplicamos el mismo efecto de listing a una tabla, poniendo una línea de cada dos con un color de fondo, pero en este caso las líneas son diferentes a las del ejemplo anterior.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("tr:odd").css("background", "#9cf");
});
</script>
<style>
table { width: 210px;
        border-collapse: collapse;
        border: 1px solid black;}
td { text-align: center;
     border: 1px solid black;}
</style>
</head>
<body>
<table>
<tr><td>1</td><td>2</td><td>3</td></tr>
<tr><td>4</td><td>5</td><td>6</td></tr>
<tr><td>7</td><td>8</td><td>9</td></tr>
<tr><td>10</td><td>11</td><td>12</td></tr>
<tr><td>13</td><td>14</td><td>15</td></tr>
</table>
</body>
</html>
```

```
$("tr:odd").css("background", "#9cf");
```

jQuery selecciona las etiquetas `<tr>` impares y las pone un color de fondo.

5. Un elemento concreto

:eq(índice)

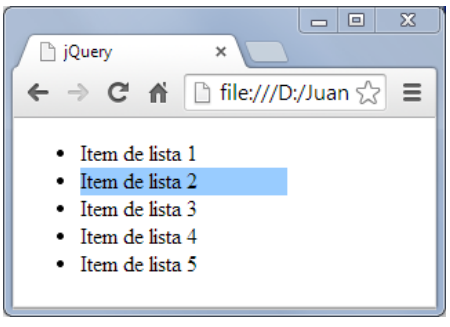
Selecciona el elemento concreto con el valor del índice.

Como los índices JavaScript empiezan por cero, el selector `:eq(0)` selecciona el primer elemento,`:eq(1)` el segundo y así sucesivamente.

`$("td:eq(2)")`: selecciona la tercera celda de una tabla.

Ejemplo

Partamos de una lista enumerada. Buscamos el elemento de lista que aparece en segundo lugar de la pantalla.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("li:eq(1)").css("background", "#9cf");
});
```

```
$("li:eq(1)").css("background", "#9cf");
```

Al segundo elemento se accede con `:eq(1)`. Es suficiente con especificar que es el segundo elemento de lista (`$("li:eq(1)")`).

```
</script>
<style>
li { width: 150px;}
</style>
</head>
<body>
<ul>
<li>Item de lista 1</li>
<li>Item de lista 2</li>
<li>Item de lista 3</li>
<li>Item de lista 4</li>
<li>Item de lista 5</li>
</ul>
</body>
</html>
```

6. Los elementos siguientes

:gt(índice)

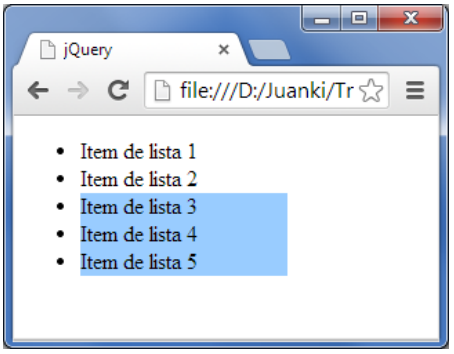
Selecciona los elementos con un valor de índice superior (greater than) al valor proporcionado en el parámetro.

Recordemos que los índices en JavaScript empiezan por 0.

`$("a:gt(1)")`: selecciona todos los enlaces de la página, empezando por el tercero (es decir, después del segundo elemento).

Ejemplo

Partamos de una lista enumerada. Vamos a seleccionar los elementos de lista a partir del ítem 3 hasta el final de la vista.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("li:eq(1)").css("background", "#9cf");
});
</script>
<style>
li { width: 150px;}
</style>
</head>
<body>
<ul>
<li>Item de lista 1</li>
<li>Item de lista 2</li>
<li>Item de lista 3</li>
<li>Item de lista 4</li>
<li>Item de lista 5</li>
</ul>
</body>
</html>
```

`$("li:gt(1)").css("background", "#9cf");`

`("li:gt(1)")`: se seleccionan todos los elementos de la lista que siguen al ítem 2.

7. Los elementos anteriores

:lt(índice)

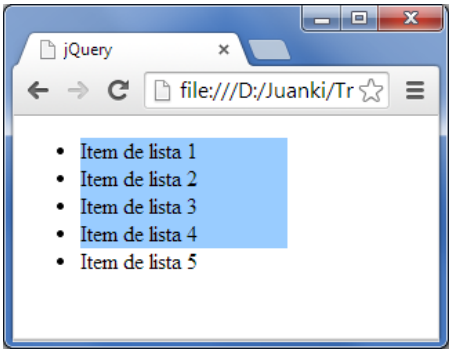
Selecciona los elementos con un valor de índice inferior (*less than*) al valor del parámetro.

Recordemos que los índices en JavaScript empiezan por 0.

`$("p:lt(3)")`: selecciona todos los párrafos situados antes del cuarto, es decir, los tres primeros párrafos.

Ejemplo

Partamos de una lista enumerada. Vamos a seleccionar los cuatro primeros elementos de la lista.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("li:gt(1)").css("background", "#9cf");
});
</script>
<style>
li { width: 150px;}
</style>
</head>
<body>
<ul>
<li>Item de lista 1</li>
<li>Item de lista 2</li>
<li>Item de lista 3</li>
<li>Item de lista 4</li>
<li>Item de lista 5</li>
</ul>
</body>
</html>
```

```
$("li:lt(4)").css("background", "#9cf");
```

`$("li:lt(4)")`: se seleccionan todos los elementos de lista anteriores al ítem 5.

8. Las etiquetas de título

:header

Devuelve todos los elementos que son etiquetas de título, como `<h1>`, `<h2>`, `<h3>`, etc.

`$(":header")`: selecciona todas las etiquetas de título de la página.

Ejemplo

Vamos a seleccionar todas las etiquetas de título de la página.


```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$(":header").css("background", "#9cf");
});
</script>
<style>
body { font-size: 10px; font-family: Arial; }
h1, h2, h3 { margin: 3px 0; }
</style>
</head>
<body>
<h1>Título de nivel 1</h1>
<p>Contenido</p>
<h2>Título de nivel 2</h2>
<p>Contenido</p>
<h3>Título de nivel 3</h3>
<p>Contenido</p>
```

```
$(":header").css("background", "#9cf");
```

`$(":header")`: selecciona todas las etiquetas de título de la página.

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

9. Exclusión de un elemento

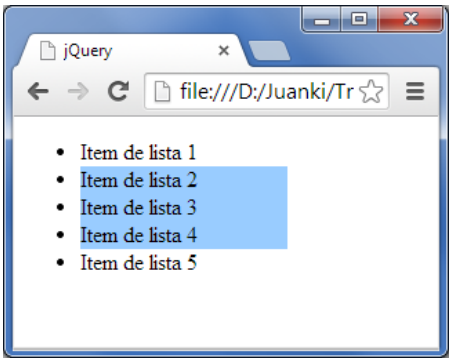
:not(selector)

Excluye de la selección todos los elementos que responden al criterio que especifica el selector.

`$("div:not(#box)")`: selecciona todas las capas, salvo las que tengan identificador box.

Ejemplo

Vamos a seleccionar todos los elementos de una lista, salvo los ítems primero y último.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("li:not(:first,:last)").css("background", "#9cf");
});
</script>
<style>
li { width: 150px;}
</style>
</head>
<body>
<ul>
<li>Item de lista 1</li>
<li>Item de lista 2</li>
<li>Item de lista 3</li>
<li>Item de lista 4</li>
<li>Item de lista 5</li>
</ul>
</body>
</html>
```

`$("li:not(:first,:last)").css("background", "#9cf");`

Selecciona todos los elementos de lista `<li>`, salvo el primero y último.

Hay que prestar atención al orden y la ubicación de los paréntesis abiertos y, sobre todo, de los cerrados.

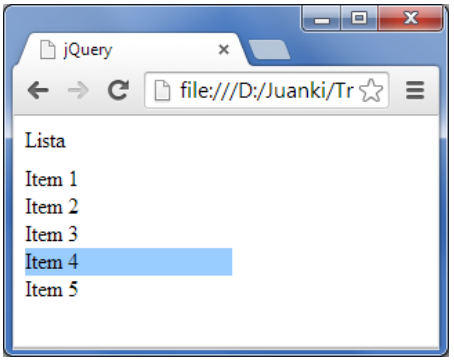
10. Los elementos según su tipo

La especificación jQuery 1.9 introdujo nuevos selectores que permiten encontrar elementos según su tipo.

`:nth-of-type`	Selecciona todos los elementos que son el enésimo hijo de sus padres respecto a los hermanos con el mismo nombre de etiqueta.
`:nth-last-of-type`	Selecciona todos los elementos que son el enésimo hijo de sus padres, contando desde la última etiqueta a la primera.
`:first-of-type`	Selecciona todos los elementos que son el primero de los hermanos con el mismo nombre de etiqueta.
`:last-of-type`	Selecciona todos los elementos que son el último de los hermanos con el mismo nombre de etiqueta.
`:only-of-type`	Selecciona todos los elementos que no tienen hermanos con el mismo nombre de etiqueta.

Ejemplo

Encontremos la quinta etiqueta `<div>`.



```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("div:nth-of-type(5)").css("background", "#9cf");
});
</script>
<style>
div { width: 150px;}
</style>
</head>
<body>
<div style="margin-bottom: 8px;">Lista</div>
<div>Item 1</div>
<div>Item 2</div>
<div>Item 3</div>
<div>Item 4</div>
<div>Item 5</div>
</body>
</html>
```

En este caso la numeración empieza en 1 porque la implementación de este selector proviene de la especificación CSS.

Los filtros hijos

En este capítulo ya hemos tratado los selectores jerárquicos que permiten seleccionar los hijos. Los filtros hijos ordenan los hijos de un elemento.

1. El primer hijo

:first-child

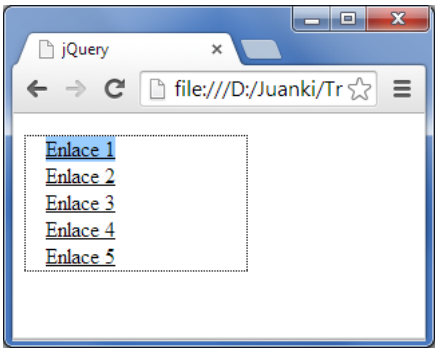
Selecciona todos los elementos que son el primer hijo de su padre.

`$("ul:first-child")`: selecciona el primer hijo (es decir, el primer elemento de lista), de la lista no ordenada `<ul>`.

El selector `:first-child` puede seleccionar varios elementos, uno por cada padre. No hay que confundir esto con `:first`, que solo selecciona un único elemento (ver la sección Los filtros jQuery básicos - El primer elemento, en este capítulo).

Ejemplo

Buscamos el primer enlace dentro de un párrafo.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("p a:first-child").css("background", "#9cf");
});
</script>
<style>
a { color: black;}
p { width: 150px;
    border: 1px dotted black;
    padding-left: 15px;}
</style>
</head>
<body>
<p>
<a href="#">Enlace 1</a><br />
<a href="#">Enlace 2</a><br />
<a href="#">Enlace 3</a><br />
<a href="#">Enlace 4</a><br />
<a href="#">Enlace 5</a>
</p>
</body>
</html>
```

`$("p a:first-child").css("background", "#9cf");`

`$("p a:first-child")`: se selecciona el primer enlace `<a>`, hijo de la etiqueta `<p>`.

2. El último hijo

:last-child

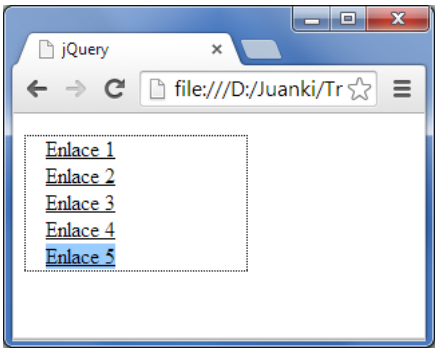
Selecciona todos los elementos que son el último hijo de su padre.

`$("ul:last-child")`: selecciona el último hijo (es decir, el último elemento de lista) de la lista no ordenada `<ul>`.

El selector `:last-child` puede seleccionar varios elementos, uno por cada padre. No hay que confundir esto con `:last`, que solo selecciona un único elemento (ver la sección Los filtros jQuery básicos - El último elemento, en este capítulo).

Ejemplo

Buscamos el último enlace dentro de un párrafo.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("p a:last-child").css("background", "#9cf");
});
</script>
<style>
a { color: black;}
p { width: 150px;
    border: 1px dotted black;
    padding-left: 15px;}
</style>
</head>
<body>
<p>
<a href="#">Enlace 1</a><br />
<a href="#">Enlace 2</a><br />
<a href="#">Enlace 3</a><br />
<a href="#">Enlace 4</a><br />
<a href="#">Enlace 5</a>
</p>
</body>
</html>
```

```
$("p a:last-child").css("background", "#9cf");
```

`$("p a:last-child")`: se selecciona el último enlace `<a>`, hijo de la etiqueta `<p>`.

3. El enésimo hijo

:nth-child(índice)

Selecciona los elementos que son el enésimo hijo de su padre. La posición la da el parámetro índice.

Al contrario de lo que sucede con muchos índices en jQuery, el índice empieza por 1, y no por 0.

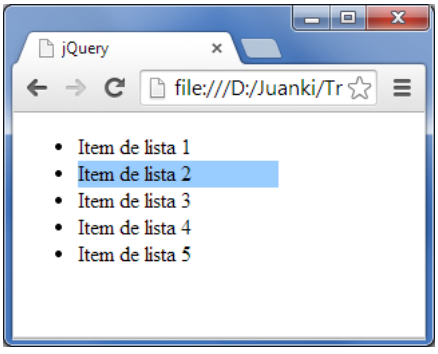
`$("ul li:nth-child(2)")`: selecciona el segundo elemento `<li>` de la lista `<ul>`.

Volvamos a esta famosa excepción respecto al índice. La mayor parte de los índices que usa jQuery llaman a funciones nativas de JavaScript y respetan la convención de empezar los índices por 0. El selector `:nth-child`, selector específico de jQuery, deriva estrictamente de las especificaciones CSS. Por lo tanto, el valor de índice 1 significa que se trata del primer elemento.

La confusión con `:eq(indice)`, que hemos visto en la sección Los filtros jQuery básicos - Un elemento concreto, en este capítulo, cuyo índice empieza por 0, puede ser una fuerte de errores difícilmente detectables en algunos scripts jQuery.

Ejemplo

Buscamos el elemento de lista que aparece en segunda posición de la pantalla. Recuerde que, para este tipo de selectores, la numeración empieza en 1.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("li:nth-child(2)").css("background", "#9cf");
});
</script>
<style>
li { width: 150px;}
</style>
```

```
$("li:nth-child(2)").css("background", "#9cf");
```

El script selecciona todos los elementos de lista `<li>` que están en segunda posición en el orden de los hijos.

Con el filtro `:eq()`, hubiéramos usado la notación:

```
$("li:eq(1)").css("background", "#9cf");
```

```
</head>
<body>
<ul>
<li>Item de lista 1</li>
<li>Item de lista 2</li>
<li>Item de lista 3</li>
<li>Item de lista 4</li>
<li>Item de lista 5</li>
</ul>
</body>
</html>
```

4. Los hijos pares e impares

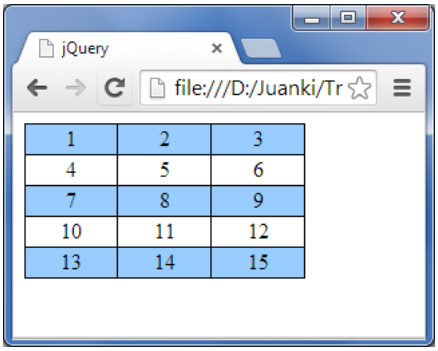
Variante del selector anterior. El selector `nth-child` puede seleccionar los elementos pares e impares.

:nth-child(even u odd)

Selecciona los enésimos elementos que son hijos pares (even) o impares (odd) de su padre.

Como `:nth-child`, que es una variante, el índice empieza en 1.

`$("table tr:nth-child(odd)")`: selecciona las líneas impares `<tr>` de la tabla.



También aquí puede haber confusión con `:even` y `:odd` (ver las secciones Los filtros jQuery básicos - Los elementos pares y Los filtros jQuery básicos -Los elementos impares, en este capítulo), ya que el índice de éstos empieza por 0.

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("table tr:nth-child(odd)").css("background", "#9cf");});
</script>
<style>
table { width: 210px;
        border-collapse: collapse;
        border: 1px solid black;}
td { text-align: center;
     border: 1px solid black;}
</style>
</head>
<body>
<table>
<tr><td>1</td><td>2</td><td>3</td></tr>
<tr><td>4</td><td>5</td><td>6</td></tr>
<tr><td>7</td><td>8</td><td>9</td></tr>
<tr><td>10</td><td>11</td><td>12</td></tr>
<tr><td>13</td><td>14</td><td>15</td></tr>
</table>
</body>
</html>
```

`$("table tr:nth-child(odd)").css("background", "#9cf");})`

El script selecciona todos los elementos de línea `<tr>` que están en el rango impar, en el orden de los hijos de la etiqueta `<table>`.

5. Los hijos únicos

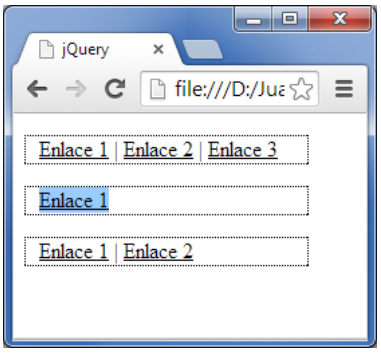
:only-child

Selecciona todos los elementos que son hijos únicos de su padre. Si el padre tiene varios hijos, no se selecciona nada.

`$("div button:only-child")`: encuentra los botones (etiqueta `<button>`) que no tienen hermanos en ninguna de las capas que se detectan.

Ejemplo

Encontremos el enlace que es hijo único de una etiqueta de párrafo `<p>`.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("p a:only-child").css("background", "#9cf");
});
</script>
<style>
a { color: black;}
p { width: 200px;
    border: 1px dotted black;
    padding-left: 10px;
    margin-bottom: 0px;}
</style>
</head>
<body>
<p><a href="#">Enlace 1</a> | <a href="#">Enlace 2</a> | <a
href="#">Enlace 3</a><br /></p>
<p><a href="#">Enlace 1</a></p>
<p><a href="#">Enlace 1</a> | <a href="#">Enlace 2</a></p>
</body>
</html>
```

```
$("p a:only-child").css("background", "#9cf");
```

`$("p a:only-child")`: jQuery revisa los párrafos, examina los enlaces `<a>` que haya y se queda con el párrafo que solo tenga un enlace.

Los filtros de contenido

1. Un texto dado

:contains(texto)

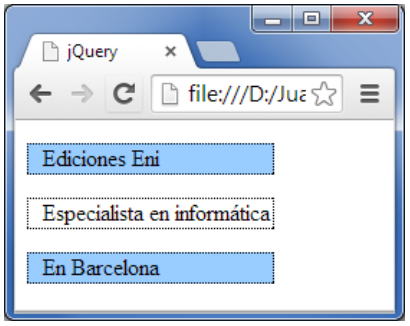
Selecciona los elementos que contengan un texto o fragmento de texto dado como argumento.

Observe que el argumento texto distingue entre mayúsculas y minúsculas (*case sensitive*).

`$("div:contains('Eni')")`: selecciona las capas que contienen el texto "Eni".

Ejemplo:

Resaltamos los párrafos que contienen el fragmento de texto "En".




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("p:contains(En)").css("background", "#9cf");
});
</script>
<style>
a { color: black;}
p { width: 160px;
    border: 1px dotted black;
    padding-left: 10px;
    margin-bottom: 0px;}
</style>
</head>
<body>
<p>Ediciones Eni</p>
<p>Especialista en informática</p>
<p>En Barcelona</p>
</body>
</html>
```

`$("p:contains(En)").css("background", "#9cf");`

`$("p:contains(En)")`: resalta los párrafos que contienen las letras "En".

2. Un contenido vacío

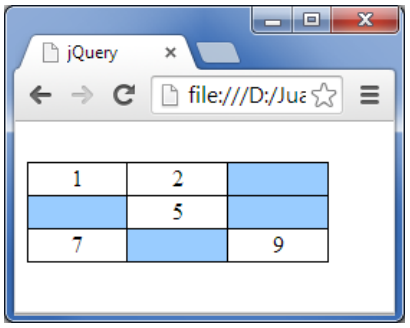
:empty

Selecciona todos los elementos que no tienen hijos o contenido de texto.

`$("div:empty")`: selecciona las capas vacías.

Ejemplo

Buscamos las celdas vacías de una tabla.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
```

`$("td:empty").css("background", "#9cf");});`

`$("td:empty")`: encuentra las celdas de tabla `<td>` vacías.

```
<script
$(document).ready(function(){
$("td:empty").css("background", "#9cf");});
</script>
<style>
table { width: 210px;
        border-collapse: collapse;
        border: 1px solid black;}
td { text-align: center;
     border: 1px solid black;}
</style>
</head>
<body>
<br>
<table>
<tr><td>1</td><td>2</td><td></td></tr>
<tr><td></td><td>5</td><td></td></tr>
<tr><td>7</td><td></td><td>9</td></tr>
</table>
</body>
</html>
```

3. Ser padre

:parent

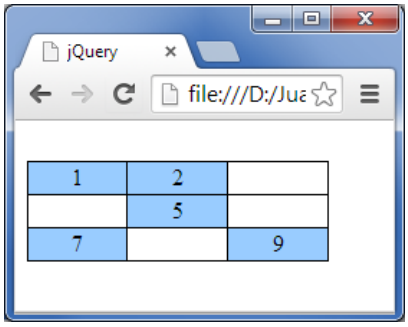
Selecciona los elementos que son padre, es decir, que tienen elementos hijos, incluidos los nodos de texto.

`$("div:parent")`: selecciona las capas que tiene elementos hijos.

Este selector es, de alguna manera, el selector inverso del anterior.

Ejemplo

Volvamos a nuestra tabla y busquemos las celdas no vacías.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("td:parent").css("background", "#9cf");});
</script>
<style>
table { width: 210px;
        border-collapse: collapse;
        border: 1px solid black;}
td { text-align: center;
     border: 1px solid black;}
</style>
</head>
<body>
<br>
<table>
<tr><td>1</td><td>2</td><td></td></tr>
<tr><td></td><td>5</td><td></td></tr>
<tr><td>7</td><td></td><td>9</td></tr>
</table>
</body>
</html>
```

`$("td:parent").css("background", "#9cf");});`

`$("td:parent")`: encuentra las celdas de tabla `<td>` que tienen un contenido y que, por tanto, son padres.

4. Un selector concreto

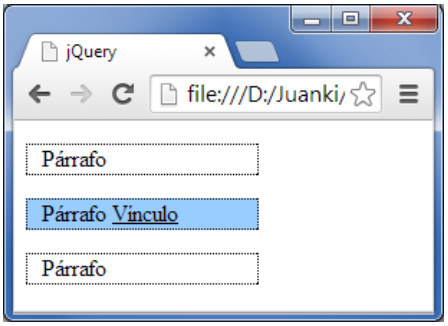
:has(selector)

Selecciona los elementos que contienen el selector dado por argumento.

`$("div:has(p)")`: selecciona las capas que contienen uno o varios párrafos.

Ejemplo:

Buscamos el párrafo que tiene un enlace.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("p:has(a)").css("background", "#9cf");
});
</script>
<style>
a { color: black;}
p { width: 150px;
    border: 1px dotted black;
    padding-left: 10px;
    margin-bottom: 0px;}
</style>
</head>
<body>
<p>Párrafo</p>
<p>Párrafo <a href="#">Enlace</a></p>
<p>Párrafo</p>
</body>
</html>
```

```
$("p:has(a)").css("background", "#9cf");
```

`$("p:has(a)")`: encuentra el párrafo que tiene un enlace.

Los filtros de visibilidad

1. Elemento visible

:visible

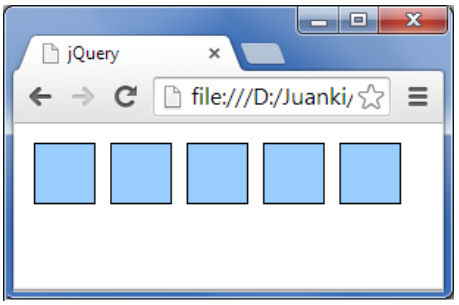
Selecciona los elementos que son visibles.

`$("p:visible")`: selecciona los párrafos visibles.

Para este filtro de jQuery, un elemento se considera como visible si ocupa espacio en el documento. Las propiedades CSS de visibilidad no se toman en cuenta.

Ejemplo

Cambiamos el color de fondo a las capas visibles.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js" ></script>
<script>
$(document).ready(function(){
$("div:visible").css("background", "#9cf");
});
</script>
<style>
.hidden { display:none;}
div { width: 40px;
      height: 40px;
      margin: 5px;
      float: left;
      border: 1px solid black;}
p { text-align: center;}
</style>
</head>
<body>
<div></div>
<div></div>
<div></div>
<div></div>
<div></div>
<div class="hidden"></div>
<div class="hidden"></div>
</body>
</html>
```

```
$("div:visible").css("background", "#9cf");
```

`$("div:visible")`: solo se seleccionan las capas visibles.

2. Elemento oculto

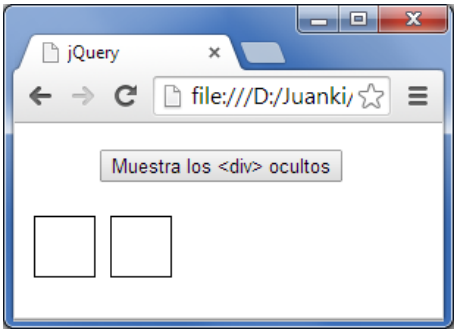
:hidden

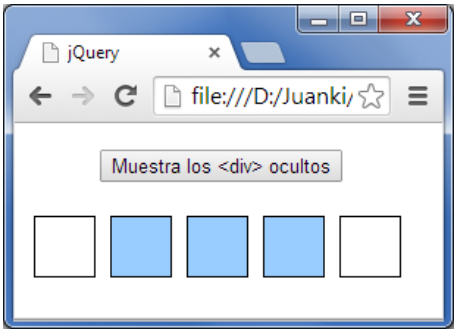
Selecciona los elementos ocultos.

`$("p:hidden")`: selecciona los párrafos ocultos.

Ejemplo

Este script va a mostrar las capas ocultas cuando se haga clic en el botón.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js" ></script>
<script>
$(document).ready(function(){
$("button").click(function () {
$("div:hidden").css("background", "#9cf").show();
});
});
</script>
<style>
button {margin-left: 50px;}
.hidden { display:none; }
div { width: 40px;
      height: 40px;
      margin: 5px;
      float: left;
      border: 1px solid black;}
</style>
</head>
<body>
<p><button>Mostrar los &lt;div&gt; ocultos</button></p>
<div class="box"></div>
<div class="hidden"></div>
<div class="hidden"></div>
<div class="hidden"></div>
<div class="box"></div>
</body>
</html>
```

Vamos a explicar brevemente este script.

```
$("button").click(function () {
$("div:hidden").show("fast");
});
```

Al hacer clic en el botón (`$("button").click(function()`), las capas ocultas (`("div:hidden")`) se muestran (`show("fast")`).

Los filtros de atributo

Los atributos son numerosos en Html. Por ejemplo, `title`, `alt`, `src`, `href`, `style`, etc.

En alguna documentación disponible en la Web puede encontrar la notación `[@attr]` relativa a los filtros de atributo. Esta notación ya no existe desde la versión jQuery 1.3. Es suficiente con eliminar el signo @ de los selectores para que el script sea compatible con las especificaciones más recientes.

1. El atributo

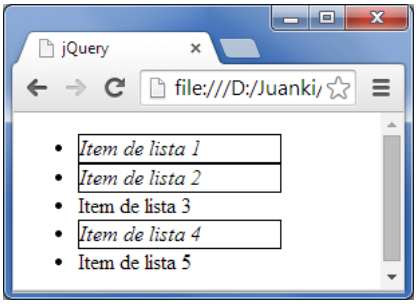
[atributo]

Selecciona los elementos que tienen el atributo especificado.

`$("div[id]")`: selecciona los elementos que tienen un atributo `id`.

Ejemplo

Buscamos los elementos de lista con el atributo `class`.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("li[class]").css("border", "1px solid black");
});
</script>
<style>
li { width: 150px;}
.cursiva { font-style: italic; }
</style>
</head>
<body>
<ul>
<li class="cursiva">Item de lista 1</li>
<li class="cursiva">Item de lista 2</li>
<li>Item de lista 3</li>
<li class="cursiva">Item de lista 4</li>
<li>Item de lista 5</li>
</ul>
</body>
</html>
```

```
$("li[class]").css("border", "1px solid black");
```

`$("li[class]")`: entre los elementos de lista `<li>`, jQuery selecciona aquellos que tienen un atributo de clase.

2. El atributo con un determinado valor

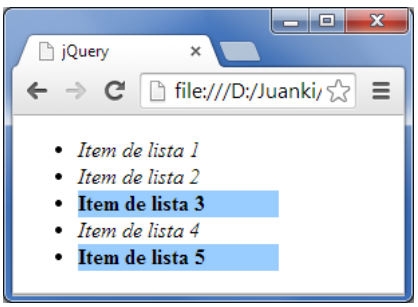
[atributo=valor]

Selecciona los elementos que tienen un atributo con un valor concreto. Es sensible a mayúsculas (*case sensitive*).

`$("input[name='newsletter']")`: selecciona el elemento de formulario `<input>` con un atributo`name="newsletter"`.

Ejemplo

Resaltamos los elementos de lista con el atributo de clase `class="negrita"`.




```
$("li[class='negrita']").css("background", "#9cf");
```

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("li[class='negrita']").css("background", "#9cf");
});
</script>
<style>
li { width: 150px;}
.cursiva { font-style: italic;}
.negrita { font-weight: bolder;}
</style>
</head>
<body>
<ul>
<li class="cursiva">Ítem de lista 1</li>
<li class="cursiva">Ítem de lista 2</li>
<li class="negrita">Ítem de lista 3</li>
<li class="cursiva">Ítem de lista 4</li>
<li class="negrita">Ítem de lista 5</li>
</ul>
</body>
</html>
```

`$("li[class='negrita']")`: entre los elementos de lista `<li>`, jQuery selecciona aquellos que tienen un atributo de clase `class="negrita"`.

3. El atributo que no tiene un valor determinado

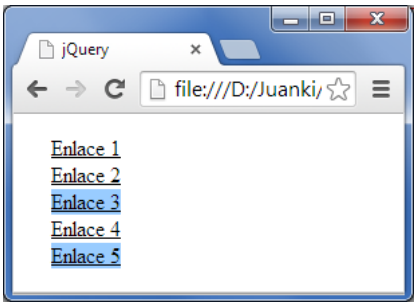
[atributo!=valor]

Selecciona los elementos que no tienen el atributo especificado y aquellos que tienen el atributo especificado, pero no con el valor indicado. El valor distingue entre mayúsculas y minúsculas (*case sensitive*).

`$("input[name!=newsletter]")`: selecciona los elementos de formulario `<input>` que no tienen el atributo `name` y aquellos que lo tienen, pero con un valor diferente a `newsletter`.

Ejemplo

Buscamos los enlaces que no tienen el atributo `title="enlace interno"`.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("a[title!="enlace interno"]").css("background", "#9cf");
});
</script>
<style>
a { color: black;}
p { margin-left: 20px;}
</style>
</head>
<body>
<p>
<a href="#" title="enlace interno">Enlace 1</a><br />
<a href="#" title="enlace interno">Enlace 2</a><br />
<a href="#" title="enlace externo">Enlace 3</a><br />
<a href="#" title="enlace interno">Enlace 4</a><br />
<a href="#" title="enlace externo">Enlace 5</a><br />
</p>
</body>
</html>
```

```
$("a[title!='enlace interno']").css("background", "#9cf
```

`$("a[title!='enlace interno']")`: entre los enlaces `<a>`, nos quedamos con los que no tienen el atributo `title="enlace interno"`.

4. El atributo cuyo valor empieza por

[atributo^=valor]

Selecciona los elementos que tienen el atributo especificado y cuyo valor empieza por los caracteres que se indican. Se distingue entre mayúsculas y minúsculas (*case sensitive*).

`$("input[name^='news']")`: selecciona los elementos de formulario `<input>` con el atributo `name`y cuyo valor empieza por los caracteres "news".

Ejemplo

Entre los enlaces disponibles, nos quedamos con aquellos para los que el atributo `title` empieza por la letra X.


```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("a[title^='X']").css("background", "#9cf");
});
</script>
<style>
a { color: black;}
p { margin-left: 20px;}
</style>
</head>
<body>
<p>
<a href="#" title="Html">Html</a>
<a href="#" title="Xhtml">Xhtml</a><br>
<a href="#" title="Dhtml">Dhtml</a>
<a href="#" title="Xml">Xml</a><br>
<a href="#" title="Xslt">Xslt</a>
<a href="#" title="Xpath">Xpath</a><br>
<a href="#" title="Xforms">Xforms</a>
<a href="#" title="CSS">CSS</a><br>
<a href="#" title="Wml">Wml</a>
<a href="#" title="Rds">Rds</a></p>
</body>
</html>
```

```
$("a[title^='X']").css("background", "#9cf");
```

`$("a[title^='X']")`: entre los enlaces `<a>`, nos quedamos con aquellos para los que el atributo`title` empieza por la letra X.

5. El atributo cuyo valor termina por

[atributo$=valor]

Selecciona los elementos que tienen el atributo especificado y cuyo valor termina por los caracteres que se indican. Se distingue entre mayúsculas y minúsculas (*case sensitive*).

`$("input[name$='letra']")`: selecciona los elementos de formulario `<input>` con el atributo`name`, para los que el valor termina por los caracteres "letra".

Ejemplo

Volviendo al ejemplo anterior, nos quedamos con los enlaces cuyo atributo `title` termina por las letras "ml".


```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("a[title$='ml']").css("background", "#9cf");
});
```

```
$("a[title$='ml']").css("background", "#9cf");
```

```
</script>
<style>
a { color: black;}
p { margin-left: 20px;}
</style>
</head>
<body>
<p>
<a href="#" title="Html">Html</a>
<a href="#" title="Xhtml">Xhtml</a><br>
<a href="#" title="Dhtml">Dhtml</a>
<a href="#" title="Xml">Xml</a><br>
<a href="#" title="Xslt">Xslt</a>
<a href="#" title="Xpath">Xpath</a><br>
<a href="#" title="Xforms">Xforms</a>
<a href="#" title="CSS">CSS</a><br>
<a href="#" title="Wml">Wml</a>
<a href="#" title="Rds">Rds</a></p>
</body>
</html>
```

`$("a[title$='ml']")`: entre los enlaces `<a>`, nos quedamos con aquellos para los que el atributo `title` termina por las letras "ml".

6. El atributo cuyo valor contiene

[atributo*=valor]

Selecciona los elementos que tienen el atributo especificado, cuyo valor contiene los caracteres que se indican. Se distingue entre mayúsculas y minúsculas (*case sensitive*).

`$("input[name*='slet']")`: selecciona los elementos de formulario `<input>` con el atributo `name`, para los que el valor contiene los caracteres "slet".

Se respeta la secuencia (u orden) de las letras.

Ejemplo

Retomando el ejemplo anterior, nos quedamos con los enlaces cuyo atributo `title` contiene las letras "tm".


```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("a[title*='tm']").css("background", "#9cf");
});
</script>
<style>
a { color: black;}
p { margin-left: 20px;}
</style>
</head>
<body>
<p>
<a href="#" title="Html">Html</a>
<a href="#" title="Xhtml">Xhtml</a><br>
<a href="#" title="Dhtml">Dhtml</a>
<a href="#" title="Xml">Xml</a><br>
<a href="#" title="Xslt">Xslt</a>
<a href="#" title="Xpath">Xpath</a><br>
<a href="#" title="Xforms">Xforms</a>
<a href="#" title="CSS">CSS</a><br>
<a href="#" title="Wml">Wml</a>
<a href="#" title="Rds">Rds</a></p>
</body>
</html>
```

`$("a[title*='tm']").css("background", "#9cf");`

`$("a[title*='tm']")`: entre los enlaces `<a>`, nos quedamos con aquellos para los que el atributo `title` contiene las letras "tm".

7. Los filtros múltiples de atributo

[filtro de atributo 1][filtro de atributo 2][filtro de atributo 3]...

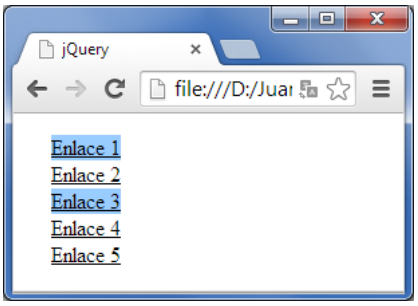
Selecciona los elementos que responden a todos los filtros de atributos especificados.

`$("input[id][name$='man']")`: selecciona las etiquetas `<input>` que tienen un identificador `id`,

para los que el atributo `name` termina por "man".

Ejemplo:

Nos quedamos con los enlaces cuyo atributo `title` empieza por "enlace", termina por "interno" y contiene "capítulo1".




```
<!doctype html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("a[title^='enlace'][title$='interno'][title*='capitulo1']")
.css("background", "#9cf");
});
</script>
<style>
a { color: black;}
p { margin-left: 20px;}
</style>
</head>
<body>
<p>
<a href="#" title="enlace capitulo1 interno">Enlace 1</a><br />
<a href="#" title="enlace capitulo2 interno">Enlace 2</a><br />
<a href="#" title="enlace capitulo1 interno">Enlace 3</a><br />
<a href="#" title="enlace capitulo3 interno">Enlace 4</a><br />
<a href="#" title="enlace externo">Enlace 5</a><br />
```

```
$("a[title^='enlace'][title$='interno'][title*='capitul
```

Los selectores y filtros de formularios

Los formularios en jQuery merecen un lugar aparte en nuestro estudio de jQuery, por lo que los selectores y filtros relativos a los formularios se tratarán en el capítulo Los formularios.

Los selectores y los caracteres especiales

Los símbolos que se usan en la sintaxis de jQuery son un problema cuando se utilizan en la parte literal del código.

Por este motivo, hay que indicar que estos caracteres no son símbolos jQuery. Para ello, tenemos que poner delante de los caracteres especiales dos barras oblicuas inversas \\ (*backslashes*).

Por ejemplo:

```
#foo\\:bar
#foo\\[bar\\]
#foo\\.bar
```

La lista completa de caracteres especiales de la sintaxis de jQuery es: `# ; & , . + * ~ ' : " ! ^ $ [ ] ( ) = > | /`

Lo más razonable es simplemente evitar su uso.

Introducción

Después de ver los numerosos selectores que tiene jQuery, en este capítulo va a empezar a estudiar el aspecto dinámico de JavaScript y de jQuery, que consiste en modificar los elementos.

Añadir o eliminar una clase

addClass(clase)

Añade la clase especificada a todos los elementos seleccionados.

`$("p:last").addClass("selected")`: añade la clase `selected` al último párrafo.

Este método devuelve un objeto jQuery.

Comentarios

Observe que este método no sustituye a una clase, sino que la añade.

Es posible añadir más de una clase al mismo tiempo. Para hacer esto, basta con especificarlas una detrás de otra, separadas por un espacio: `addClass(clase1 clase2 clase3)`.

Este método `addClass()` se asocia normalmente con el método `removeClass()`, lo que permite crear un efecto de conmutación.

removeClass(clase)

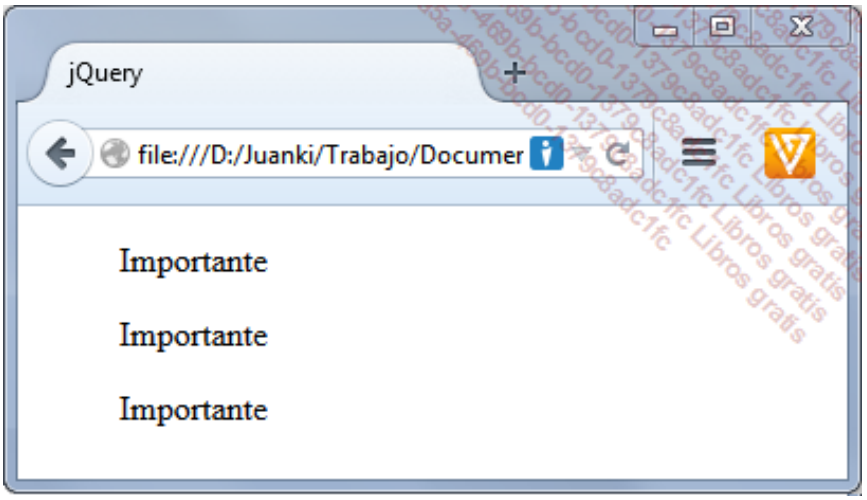
Elimina la clase especificada de todos los elementos seleccionados.

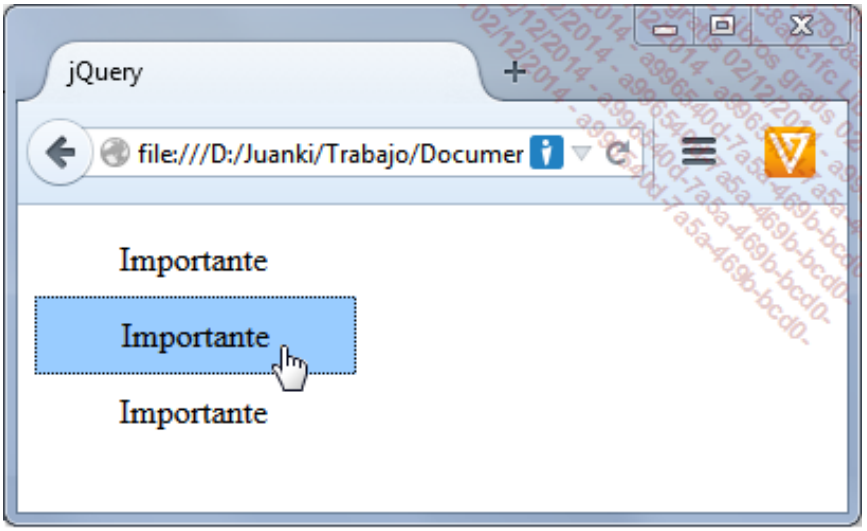
`$("p:last").removeClass("selected")`: elimina la clase `selected` del último párrafo.

Este método devuelve un objeto jQuery.

Ejemplo

Al pasar el ratón por encima de un párrafo, lo resaltamos dándole un color de fondo y un borde. Este efecto se consigue añadiendo una clase al movimiento del ratón.





El documento inicial se presenta a continuación:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
div { width: 150px;
      height: 35px;
      line-height: 35px;
      vertical-align: middle;
      text-align: center;
      cursor: pointer;}
.colorDefondo { background-color: #9cf;
                border: 1px dotted black;}
</style>
</head>
<body>
<div>Importante</div>
<div>Importante</div>
<div>Importante</div>
</body>
</html>
```

Pasemos al script jQuery.

```
<script>
$(document).ready(function(){
$("div").mouseover(function(){
$(this).addClass("colorDefondo");
});
$("div").mouseout(function(){
$(this).removeClass("colorDefondo");
});
});
</script>
```

Detallemos el script.

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

Cuando se carga el DOM.

```
$("div").mouseover(function(){
$(this).addClass("colorDefondo");
});
```

Al pasar el ratón (`mouseover`) por encima de una capa `<div>`, el script añade (`addClass()`) a esta capa (`this`) la clase `colorDefondo`.

```
$("div").mouseout(function(){
$(this).removeClass("colorDefondo");
});
```

Cuando el cursor sale de la capa (`mouseout`), jQuery elimina (`removeClass()`) de esta capa (`this`) la clase `colorDefondo`.

```
});
```

Fin de script.

El archivo Html completo es:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("div").mouseover(function(){
$(this).addClass("colorDefondo");
});
$("div").mouseout(function(){
$(this).removeClass("colorDefondo");
});
});
</script>
<style>
div { width: 150px;
      height: 35px;
      line-height: 35px;
      vertical-align: middle;
      text-align: center;
      cursor: pointer;}
.colorDefondo { background-color: #9cf;
                border: 1px dotted black;}
</style>
</head>
<body>
<div>Importante</div>
<div>Importante</div>
<div>Importante</div>
</body>
</html>
```

En lugar de pasar una cadena de caracteres (el nombre de la clase), desde la versión 1.4 de jQuery es posible añadir una o varias clases (separadas por un espacio) a los elementos seleccionados por una función.

addClass(función(índice, clase actual))

Donde:

- "función" especifica una función que devuelve uno o varios nombres de clase que se deben

añadir.

- "índice" (opcional) es la posición del índice del elemento seleccionado.
- clase actual (opcional) es el nombre de la clase actualmente presente.

Ejemplo:

```
$("p").addClass(function(1){
addedClass = "nuevaClase";
return addedClass;
}
```

Añade la clase `nuevaClase` al segundo párrafo.

También es posible pasar por una función para eliminar una clase.

removeClass(función(índice, clase actual))

Comprobar la presencia de una clase

hasClass(clase)

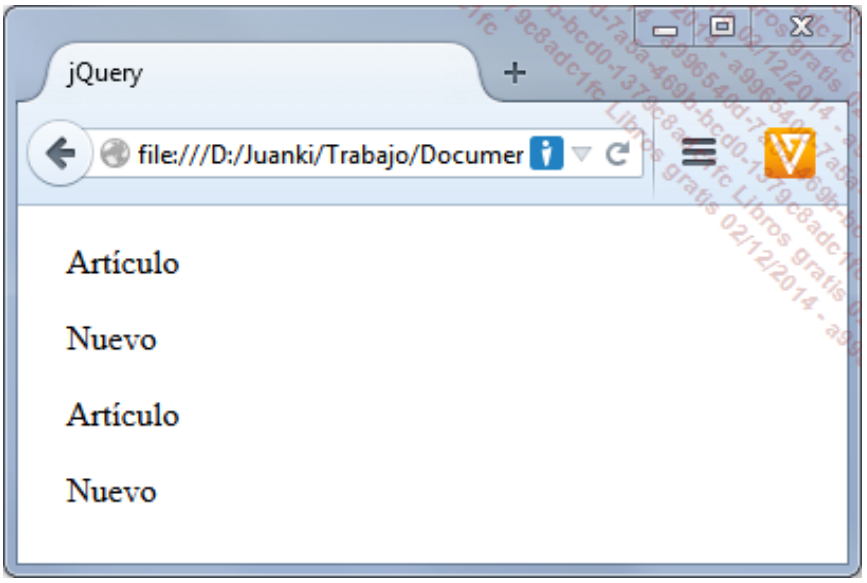
Comprobar si la clase que se especifica en el argumento está presente para los elementos de destino. Devuelve *true* si la clase especificada está presente para, al menos, uno de los elementos de destino; *false* en caso contrario.

`$("#p1").hasClass("box")`: comprobar si el elemento identificado por `p1` tiene la clase `box`.

El método envía un booleano (true o false).

Ejemplo

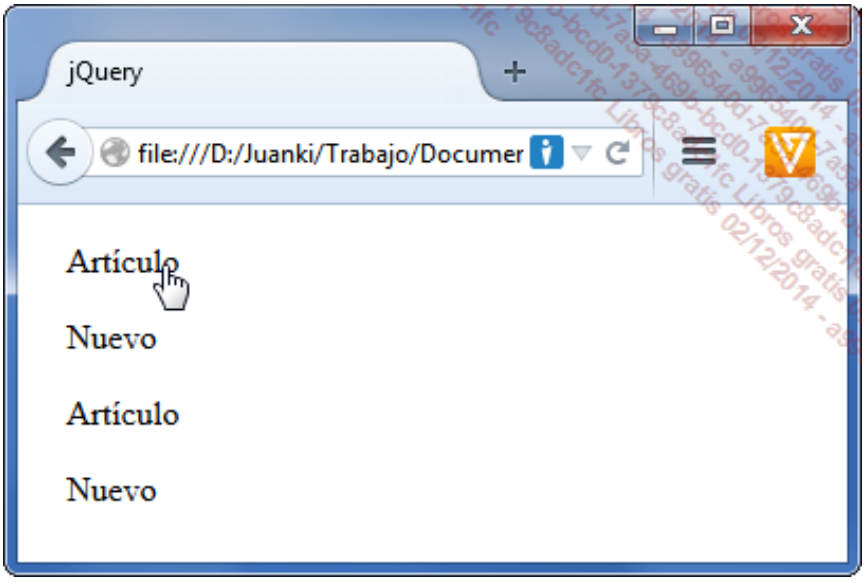
Consideremos una serie de párrafos:

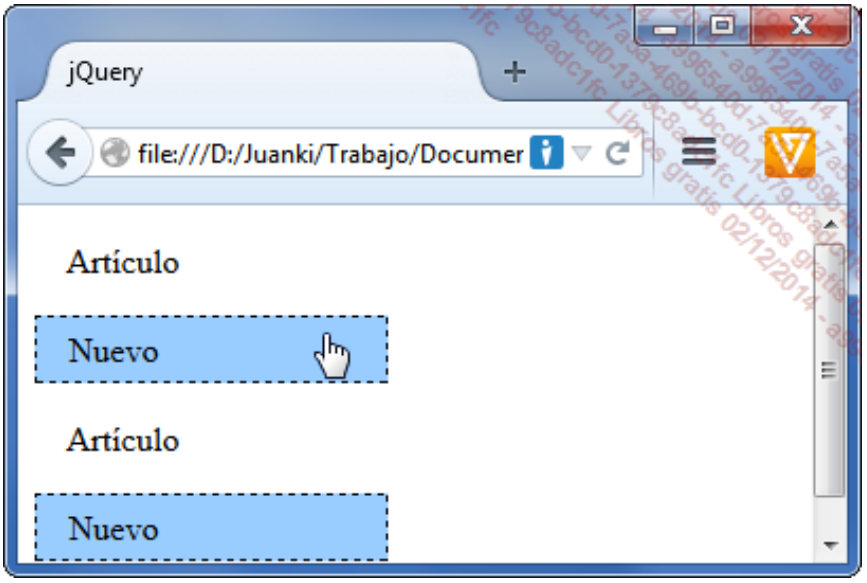


Inicialmente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
p { width: 150px;
    padding-left: 15px;
    cursor: pointer;}
.colorDefondo { height: 30px;
                line-height: 30px;
                vertical-align: middle;
                background-color: #9cf;
                border: 1px dashed black;}
</style>
</head>
<body>
<p>Artículo</p>
<p class="new">Nuevo</p>
<p>Artículo</p>
<p class="new">Nuevo</p>
</body>
</html>
```

Al pasar el ratón por encima del párrafo, el script jQuery va a dar un color de fondo y un borde solo a los párrafos con la clase `new`.





El script jQuery:

```
<script>
$(document).ready(function(){
$("p").mouseover(function(){
if ($(this).hasClass("new") ) {
$(this).addClass("colorDefondo")
};
});
});
</script>
```

La explicación del script es:

```
$(document).ready(function(){
$("p").mouseover(function(){
```

Al cargar el DOM y cuando se pasa el ratón por encima de los párrafos.

```
if ($(this).hasClass("new") ) {
$(this).addClass("colorDefondo")
};
```

El script comprueba si (`if`) el elemento (`this`) sobre el que está el ratón tiene la

clase `new(hasClass("new"))`. En caso afirmativo, se le añade la clase `colorDefondo` (`addClass()`). Observe que nada impide mezclar JavaScript clásico con JavaScript jQuery.

```
});
});
```

Fin de script

El resultado final se muestra en el siguiente archivo Xhtml:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("p").mouseover(function(){
if ($(this).hasClass("new") ) {
$(this).addClass("colorDefondo")
};
});
});
</script>
<style>
p { width: 150px;
    padding-left: 15px;
    cursor: pointer;}
.colorDefondo { height: 30px;
                line-height: 30px;
                vertical-align: middle;
                background-color: #9cf;
                border: 1px dashed black;}
</style>
</head>
<body>
<p>Artículo</p>
<p class="new">Nuevo</p>
<p>Artículo</p>
<p class="new">Nuevo</p>
</body>
</html>
```

Cambiar entre dos clases

La librería jQuery tiene varios métodos que permiten desencadenar algunas veces una acción y otras, una acción diferente. Este efecto de permutación se llama *toggle*. Lo encontraremos varias veces en nuestra exploración de jQuery.

Al margen de los efectos espectaculares, estos métodos ahorran muchas líneas de código.

toggleClass(clase)

Añade la clase especificada si no existe, o la elimina si no está presente.

`$(p).toggleClass("clase1")`: aplica la clase `clase1` a los párrafos, si no existe. Si existe, la elimina.

Este método devuelve un objeto jQuery.

Desde la versión 1.4 de jQuery, es posible hacer este efecto de permutación pasando por una función.

toggleClass(función(índice, clase actual), [conmutador])

Donde:

- "función" especifica una función que devuelve el nombre de la clase que se tiene que permutar.
- "índice" (opcional) es la posición del índice del elemento seleccionado.
- "clase actual" (opcional) es el nombre de la clase actual.
- "conmutador" es un valor booleano (`true` o `false`), que determina si la clase se tiene que añadir o eliminar.

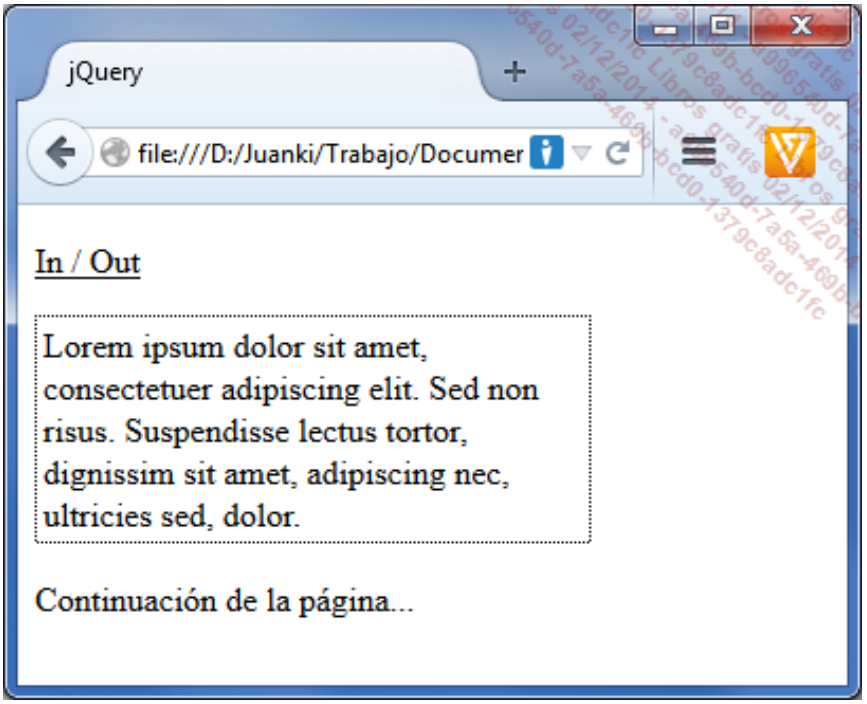
Ejemplo

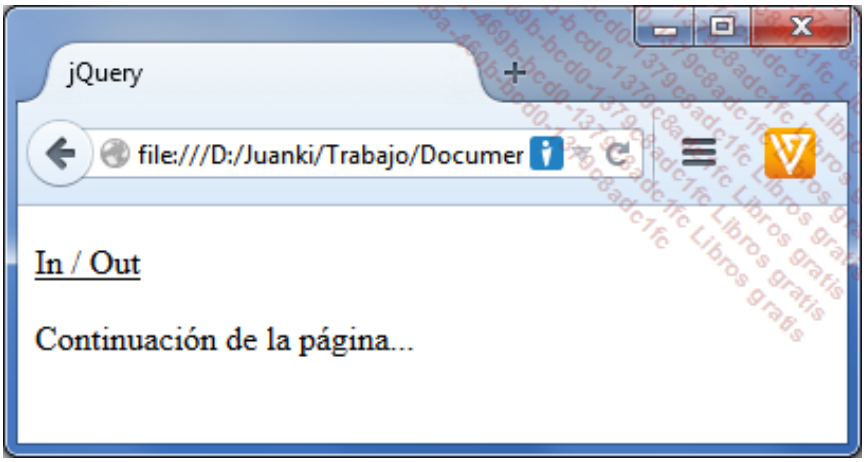
```
$('div.pequeña').toggleClass(function() {
if ($(this).parent().is('.forma')) {
return 'clase1';
}
else {
return 'clase2';
}
});
```

Este ejemplo va a permutar la clase `clase1` para los elementos `<div class="pequeña">` si su elemento padre tiene una clase `forma`. En caso contrario va a cambiar a la clase `clase2`.

Ejemplo

Haciendo clic en el enlace, haremos que aparezca o desaparezca una capa de la página.





Inicialmente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
a{color: black}
div { width: 255px;
      padding: 3px;
      border: 1px dotted black;
      cursor: pointer;}
.ocultar { display: none;}
</style>
</head>
<body>
<p><a href="#">In / Out</a></p>
<div>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed
non risus. Suspendisse lectus tortor, dignissim sit amet,
adipiscing nec, ultricies sed, dolor.</div>
<p>Resto de la página...</p>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$("a").click(function(){
$("div").toggleClass("ocultar");
});
});
</script>
```

Exploremos el script paso a paso.

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

Cuando se carga la página o, para ser preciso, el DOM.

```
$("a").click(function(){
```

Al hacer clic con el ratón en el enlace `<a>`.

```
$("div").toggleClass("ocultar");
```

Permuta (`toggleClass()`) la clase `ocultar` en la capa `<div>`.

```
});
});
```

Fin de script.

Al final:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("a").click(function(){
$("div").toggleClass("ocultar");
});
});
</script>
<style>
a{color: black}
div { width: 255px;
      padding: 3px;
      border: 1px dotted black;
      cursor: pointer;}
.ocultar { display: none;}
</style>
</head>
<body>
<p><a href="#">In / Out</a></p>
<div>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed
non risus. Suspendisse lectus tortor, dignissim sit amet,
adipiscing nec, ultricies sed, dolor.</div>
<p>Resto de la página...</p>
</body>
</html>
```

Hay otras formas de conseguir este efecto en jQuery; por ejemplo, con los métodos `show()` y `hide()` (ver el capítulo Los efectos).

Conocer el valor de un atributo

Este método jQuery corresponde a `getAttribute()` del JavaScript clásico.

attr(nombre del atributo)

Accede al valor del atributo mencionado.

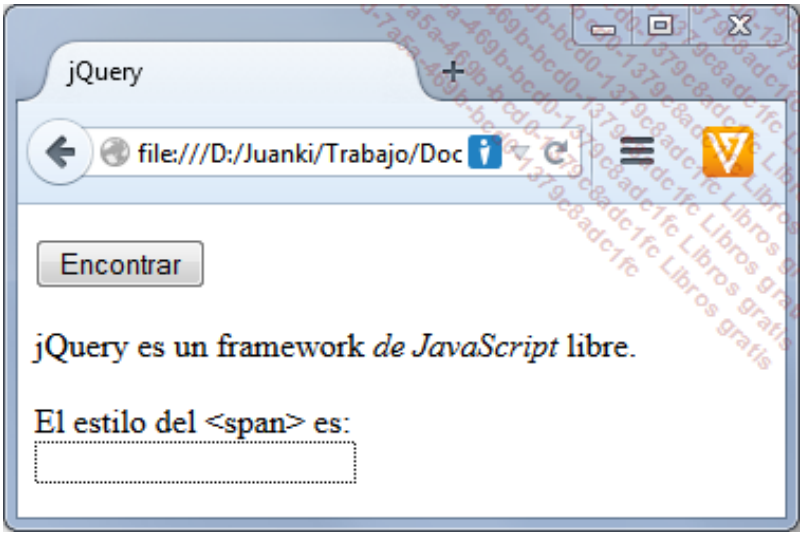
Este método es muy útil para encontrar el valor de un atributo del elemento seleccionado, o del primer elemento seleccionado si hay varios. Si el elemento no tiene el atributo con este nombre, se devuelve el valor `undefined`.

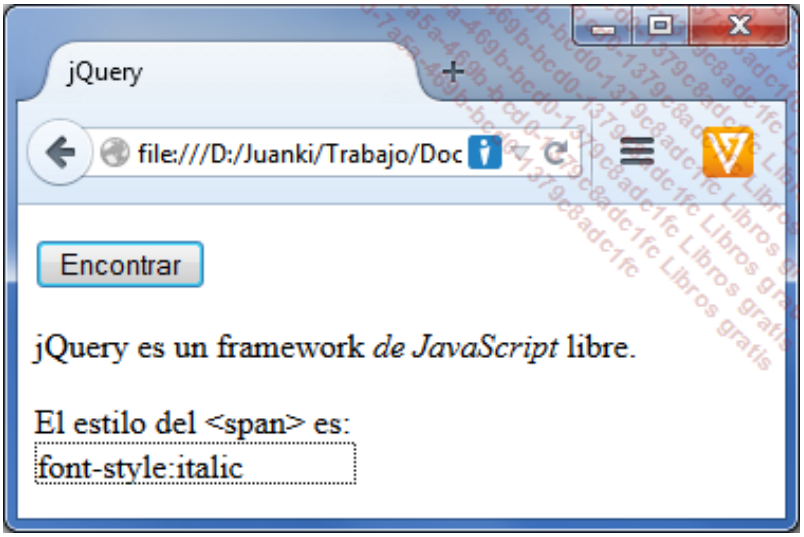
`$("a").attr("title")`: recupera el valor del atributo `title` del primer enlace que se encuentra.

Este método devuelve un objeto jQuery.

Ejemplo

Al hacer clic en el botón, buscamos el estilo de la etiqueta `<span>JavaScript</span>`. El resultado se mostrará en una capa prevista para este efecto.





El archivo inicial es:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
```

```
<title>jQuery</title>
<style>
div { width: 150px;
      height: 18px;
      border: 1px dotted black;}
</style>
</head>
<body>
<p><button>Encontrar</button></p>
<p>
jQuery es un framework <span style="font-
style:italic">JavaScript</span> libre.
</p>
El estilo del &lt;span&gt; es:<div></div>
</body>
</html>
```

El script jQuery se presenta de la siguiente manera:

```
<script>
$(document).ready(function(){
$("button").click(function(){
var css = $("span").attr("style");
$("div").text(css);
});
});
</script>
```

Detallamos a continuación.

```
$(document).ready(function(){
$("button").click(function(){
```

Cuando se carga el DOM y se hace clic en el botón.

```
var css = $("span").attr("style");
```

El script carga en la variable `css` el valor del atributo `style` de la etiqueta `<span>`.

```
$("div").text(css);
```

Esta variable se mostrará como texto (`text(css)`) en la capa `<div>` prevista para este efecto.

```
});
});
```

Fin del script.

El documento final sería:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("button").click(function(){
var css = $("span").attr("style");
$("div").text(css);
});
```

```
});
</script>
<style>
div { width: 150px;
      height: 18px;
      border: 1px dotted black;}
</style>
</head>
<body>
<p><button>Encontrar</button></p>
<p>
jQuery es un framework <span style="font-
style:italic">JavaScript</span> libre.
</p>
El estilo del &lt;span&gt; es:<div></div>
</body>
</html>
```

Añadir un atributo y su valor

attr(atributo, valor)

Asigna un par atributo/valor a todos los elementos implicados.

`$("#foto").attr("alt","parque eólico")`: asigna al elemento identificado por `#foto` el atributo`alt="parque eólico"`.

Este método devuelve un objeto jQuery.

También es posible añadir un atributo con la función

attr(nombre del atributo, función(índice, valor actual)

Donde:

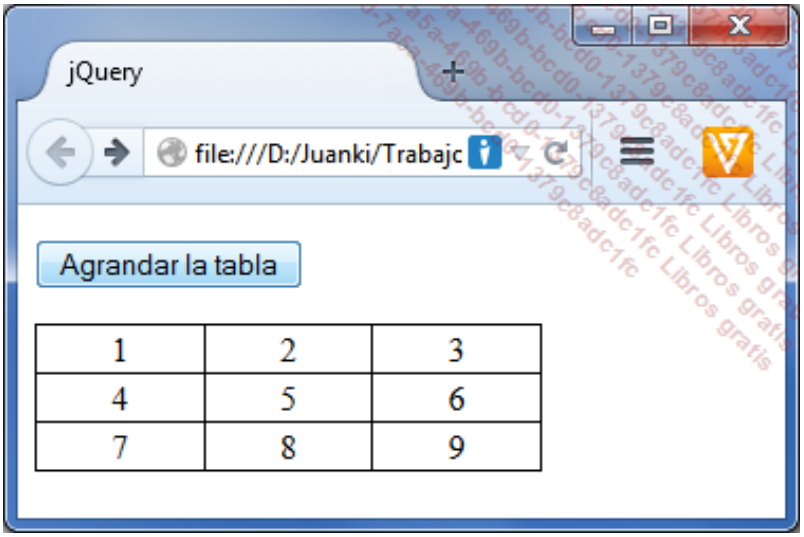
- "función" especifica una función que devuelve el nuevo valor del atributo.
- "índice" (opcional) es la posición del índice del elemento seleccionado.
- "valor actual" (opcional) es el valor actual o anterior del valor del atributo.

Ejemplo

```
$('#foto').attr('alt', function(i, val) {
return val + 'Vacaciones en Francia'
});
```

Ejemplo

Al hacer clic en el botón, se mostrará una tabla de datos con una longitud más grande, para que sea más sencillo leerla.



El archivo de inicio:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
table { border-collapse: collapse;
        border: 1px solid black;}
td { text-align: center;
     border: 1px solid black;}
</style>
</head>
<body>
<p><button>Agrandar la tabla</button></p>
<table>
<tr>
<td>1</td><td>2</td><td>3</td>
</tr>
<tr>
<td>4</td><td>5</td><td>6</td>
</tr>
<tr>
<td>7</td><td>8</td><td>9</td>
</tr>
</table>
</body>
</html>
```

El script jQuery es:

```
<script>
$(document).ready(function(){
$("button").click(function(){
$("table").attr("width","240px");
});
});
</script>
```

Explicaciones:

```
$(document).ready(function(){
$("button").click(function(){
```

Cuando se carga el DOM y al hacer clic en el botón.

```
$("table").attr("width","240px");
```

El atributo `width`, con un valor de `240 px`, se añade a la etiqueta de tabla `<table>`.

```
});
});
```

Fin del script.

Un efecto espectacular con poco código.

El archivo completo:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("button").click(function(){
$("table").attr("width","240px");
});
});
</script>
<style>
table { border-collapse: collapse;
        border: 1px solid black;}
td { text-align: center;
     border: 1px solid black;}
</style>
</head>
<body>
<p><button>Agrandar la tabla</button></p>
<table>
<tr>
<td>1</td><td>2</td><td>3</td>
</tr>
<tr>
<td>4</td><td>5</td><td>6</td>
</tr>
<tr>
<td>7</td><td>8</td><td>9</td>
</tr>
</table>
</body>
</html>
```

Añadir varios atributos y sus valores

attr({propiedades})

Permite asignar un conjunto de pares atributo/valor a los elementos seleccionados.

Las diferentes propiedades se separan por una coma.

`$("img").attr({ src: "hat.gif", alt: "Logo jQuery!" })`: asigna los atributos `src` y `alt` a las imágenes.

Este método devuelve un objeto jQuery.

Ejemplo

Pasamos de una imagen a otra, simplemente haciendo clic en un enlace.

Las imágenes del ejemplo están disponibles para descarga en la página Información.

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
a { color: black;}
</style>
```

```
</head>
<body>
<p><a href="#">Imagen siguiente</a></p>
<div>
<img src="panelsolar1.png" alt="Panel solar 1"/>
</div>
</body>
</html>
```

El script jQuery se presenta de la siguiente manera.

```
<script>
$(document).ready(function(){
$("a").click(function(){
$("img").attr({ src: "panelsolar2.png",
                alt: "Panel solar 2",
                title: "Ecología"
});
});
});
</script>
```

Explicaciones paso a paso.

```
$(document).ready(function(){
$("a").click(function(){
```

Cuando se carga jQuery (el DOM) y al hacer clic en el enlace.

```
$("img").attr({ src: "panelsolar2.png",
                alt: "Panel solar 2",
                title: "Ecología"
});
```

En primer lugar, el script añade a la etiqueta `<img>` el atributo `src`. Esto permite cargar la nueva imagen. Adicionalmente, también se añaden los atributos `alt` y `title`.

```
});
});
```

Fin de script.

El archivo completo queda como sigue:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("a").click(function(){
$("img").attr({ src: "panelsolar2.png",
                alt: "Panel solar 2",
                title: "Ecología"
});
});
});
</script>
<style>
a { color: black;}
</style>
```

```
</head>
<body>
<p><a href="#">Imagen siguiente</a></p>
<div>
<img src="panelsolar1.png" alt="Panel solar 1"/>
</div>
</body>
</html>
```

Eliminar un atributo

removeAttr(nombre del atributo)

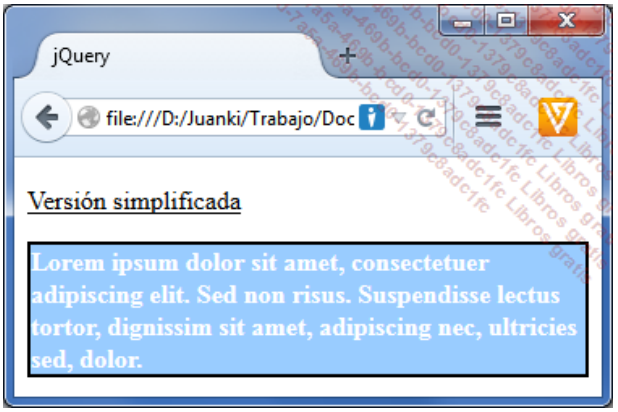
Elimina un atributo de los elementos implicados.

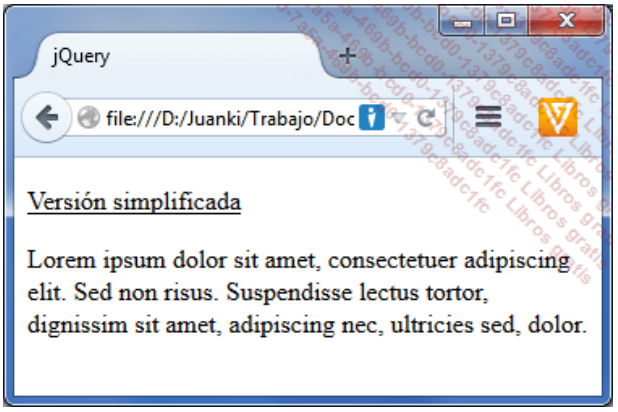
`$("#div1").removeAttr("disabled")`: elimina el atributo `disabled` del elemento identificado por`div1`.

Este método devuelve un objeto jQuery.

Ejemplo

Eliminamos el atributo de estilo de la capa para que la lectura sea más sencilla.





El documento Html inicial:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
a { color: black}
</style>
</head>
<body>
<p><a href="#">Versión simplificada</a></p>
<div style="background-color: #9cf; border: 2px solid black;
color: white; font-weight: bold;">Lorem ipsum dolor sit amet,
consectetuer adipiscing elit. Sed non risus. Suspendisse lectus
tortor, dignissim sit amet, adipiscing nec, ultricies sed,
dolor.</div>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$("a").click(function(){
$("div").removeAttr("style");
});
});
</script>
```

```
$(document).ready(function(){
$("a").click(function(){
```

Cuando se carga el DOM y se hace clic en el enlace.

```
$("div").removeAttr("style");
```

El script elimina el atributo de estilo actual (ver el código Xhtml) de la división.

```
});
});
```

Fin de script.

El archivo Html final queda como sigue:

```
<!doctype html>
<html lang="es">
```

```
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("a").click(function(){
$("div").removeAttr("style");
});
});
</script>
<style>
a { color: black}
</style>
</head>
<body>
<p><a href="#">Versión simplificada</a></p>
<div style="background-color: #9cf; border: 2px solid black;
color: white; font-weight: bold;">Lorem ipsum dolor sit amet,
consectetuer adipiscing elit. Sed non risus. Suspendisse lectus
tortor, dignissim sit amet, adipiscing nec, ultricies sed,
dolor.</div>
</body>
</html>
```

Conocer el atributo value

Este punto trata de los formularios y su atributo `value`.

val()

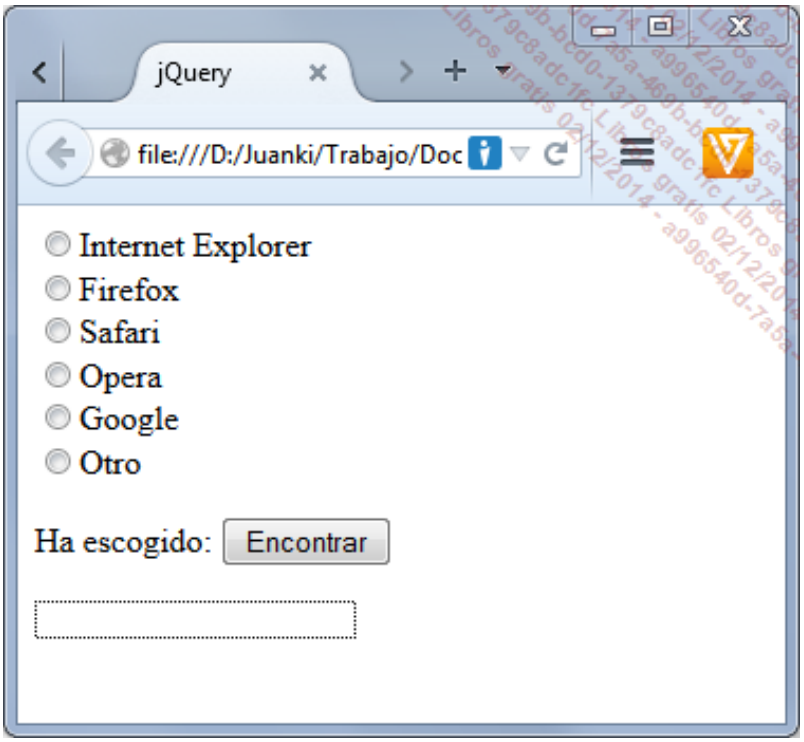
Recupera, en forma de cadena de caracteres, el contenido del atributo `value` del primer elemento de la selección.

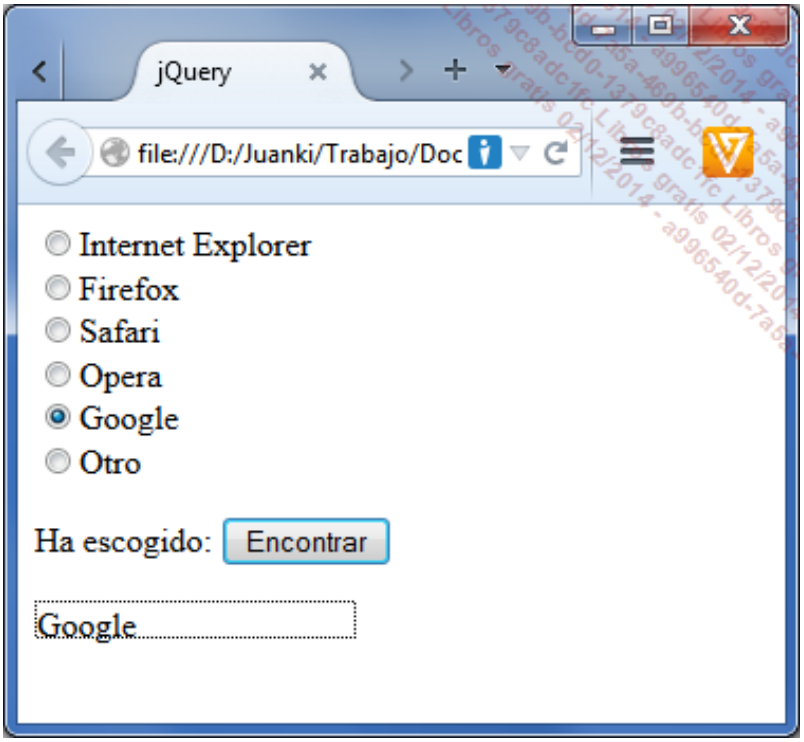
`$("input").val()`: recupera el contenido del atributo `value` del primer campo del formulario de tipo`<input>`.

Este método devuelve una cadena de caracteres.

Ejemplo

Recuperamos el valor del botón de radio que se ha marcado.





Inicialmente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
div { width: 150px;
      height: 16px;
      border: 1px dotted black;}
</style>
</head>
<body>
<form action="">
<input type="radio" name="1" value="Internet Explorer" />Internet
Explorer<br>
<input type="radio" name="1" value="Firefox">Firefox<br>
<input type="radio" name="1" value="Safari">Safari<br>
<input type="radio" name="1" value="Opera">Opera<br>
<input type="radio" name="1" value="Google">Google<br>
<input type="radio" name="1" value="Otro">Otro
</form>
<p>Su opción es:
<button>Encontrar</button></p>
<div></div>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$("button").click(function(){
var eleccion = $('input:radio:checked').val();
$("div").text(eleccion);
});
});
```

```
</script>
```

Explicaciones:

```
$(document).ready(function(){
$("button").click(function(){
```

Cuando se carga el DOM y al hacer clic en el botón.

```
var eleccion = $('input:radio:checked').val();
```

El contenido del botón de radio seleccionado (`'input:radio:checked'`), se almacena en la variable`eleccion`.

```
$("div").text(eleccion);
```

El contenido de la variable `eleccion` se muestra (`text(eleccion)`) en la capa.

```
});
});
```

Fin de script.

Al final:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("button").click(function(){
var eleccion = $('input:radio:checked').val();
$("div").text(eleccion);
});
});
</script>
<style>
div { width: 150px;
      height: 16px;
      border: 1px dotted black;}
</style>
</head>
<body>
<form action="">
<input type="radio" name="1" value="Internet Explorer" />Internet
Explorer<br />
<input type="radio" name="1" value="Firefox">Firefox<br>
<input type="radio" name="1" value="Safari">Safari<br>
<input type="radio" name="1" value="Opera">Opera<br>
<input type="radio" name="1" value="Google">Google<br>
<input type="radio" name="1" value="Otro">Otro
</form>
<p>Su opción es:
<button>Encontrar</button></p>
<div></div>
</body>
</html>
```

Modificar el atributo value

Es una variante de `val()`, que hemos visto en la sección anterior. En este caso, jQuery permite modificar el atributo `value`.

val(valor)

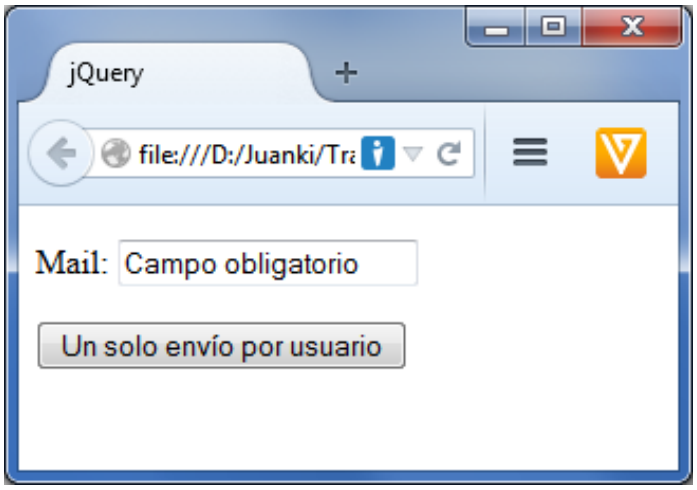
Asigna un nuevo valor al atributo `value` del elemento seleccionado.

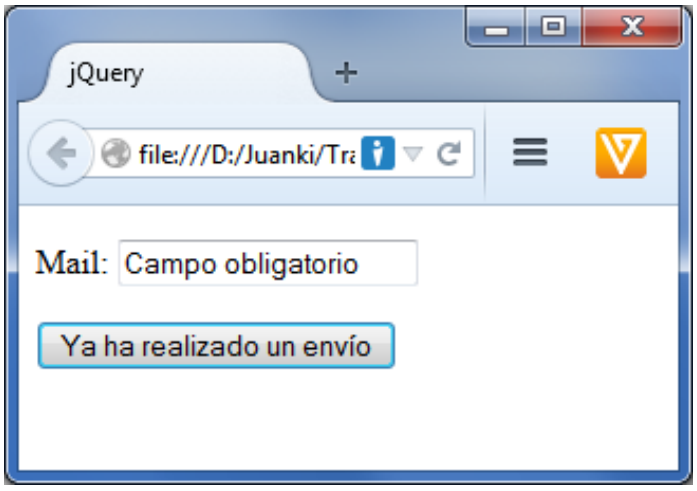
`$("#input").val("Test")`: añade el valor `Test` a una línea de texto.

Este método devuelve un objeto jQuery.

Ejemplo

Después de un primer envío del formulario, se modifica el texto del botón de envío.





El archivo inicial:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
a { color: black;}
</style>
</head>
<body>
<form action="">
```

```
<p>Mail: <input type="text" id="input1" value="Valor
obligatorio"></p>
<p><input type="submit" id="enviar" value="Un solo envío
por usuario"></p>
</form>
</body>
</html>
```

Pasamos al script jQuery.

```
<script>
$(document).ready(function(){
$("#enviar").click(function(){
$("#enviar").val("Ya ha realizado un envío");
return false;
});
});
</script>
```

Lo detallamos a continuación.

```
$(document).ready(function(){
$("#enviar").click(function(){
```

Cuando se carga el DOM y al hacer clic en el botón de envío.

```
$("#enviar").val("Ya ha realizado un envío");
return false;
```

Un nuevo valor (`"Ya ha realizado un envío"`) se transmite al botón de envío. Observe que la mención anterior también se sustituye.

```
});
});
```

Fin del script.

Al final:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#enviar").click(function(){
$("#enviar").val("Ya ha realizado un envío");
return false
});
});
</script>
<style>
a { color: black;}
</style>
</head>
<body>
<form action="">
<p>Mail: <input type="text" id="input1" value="Valor
obligatorio"></p>
<p><input type="submit" id="enviar" value="Un solo envío
por usuario"></p>
```

```
</form>
</body>
</html>
```

Conocer la propiedad de un elemento

prop(propiedad)

Accede al valor de la propiedad del elemento en el DOM.

Este método recupera el valor de la propiedad del elemento seleccionado o del primer elemento si hay varios. Si el elemento no tiene ninguna propiedad que responda a la que se quiere recuperar en el código, se retorna el valor `undefined`.

`$("input[type='checkbox']").prop("checked")` recupera el valor `true` o `false` de la propiedad `checked` de la casilla de verificación.

Este método devuelve un objeto jQuery.

Al cargar la página, el navegador crea el DOM (*Document Object Model*) de la página a partir de las etiquetas y los atributos del Html.

Por ejemplo, el atributo `checked` de una casilla de verificación en el código fuente solo se utiliza para determinar el valor inicial de la casilla. El atributo `checked` no cambia con el estado de esta casilla de verificación. Por el contrario, en el DOM, la propiedad `checked` cambia (`true` o `false`) según el usuario marque o desmarque esta casilla.

El método `prop()` permite acceder a los elementos del DOM como `nodeName`, `nodeType`,`selectedIndex` y `childNodes` que no tienen equivalente en los atributos de Html y que, por esta razón, no son accesibles a través del método `atrr()`.

Como propiedades que se puede modificar, podemos citar:

- `checked` de las etiquetas `<input>` de tipo radio o checkbox.
- `selected` de la etiqueta `<option>`
- `disabled` de las etiquetas `<input>`, `<textarea>`, `<button>`, `<select>`, `<option>` y`<optgroup>`.
- `readonly` de las etiquetas `<input>` de texto o de contraseña y `<textarea>`.
- `multiple` de la etiqueta `<select>`.

La diferencia entre `prop()` y `attr()` es que el método `attr()` coge el atributo que se ha determinado en el código Html, mientras que el método `prop()` coge una propiedad del DOM.

Ejemplo

Ilustremos en el siguiente ejemplo la diferencia entre atributo `(attr())` y propiedad `(prop())`.

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
p { margin: 15px 0 0;}
</style>
<script src="jquery.js"></script>
</head>
<body>
<input type="checkbox" checked="checked">
<p></p>
<script>
$(document).ready(function(){
```

```
$("input").change(function() {
var $input = $( this );
$("p").html(
"<b>Atributo</b> .attr(\"checked\"): <b>" +
$input.attr("checked") + "</b><br>" +
"<b>Propiedad</b> .prop(\"checked\"): <b>" +
$input.prop("checked")) + "</b>";
})
.change();
})
</script>
</body>
</html>
```

Observe el atributo `checked` del campo de formulario `<input>`.

Al cargar la página:

El método `attr("checked")` retorna el valor del atributo `checked`, en este caso `checked`, como se indica en el código. El método `prop("checked")` devuelve el booleano que representa el estado de la casilla de verificación, en este caso `true`.

Si se desmarca la casilla de verificación:

Como el código Html de la página es idéntico, el método `attr("checked")` devuelve siempre el valor del atributo `checked`, en este caso `checked`. Por el contrario, el valor del método`prop("checked")` devuelve el valor `false` que corresponde al nuevo estado de la casilla de verificación.

Modificar la propiedad de un elemento

prop(propiedad, valor)

Asigna un valor a una propiedad de los elementos seleccionados.

`$("input[type='checkbox']").prop("checked", true)`: asigna el valor `true` a la propiedad `checked` de la casilla de verificación.

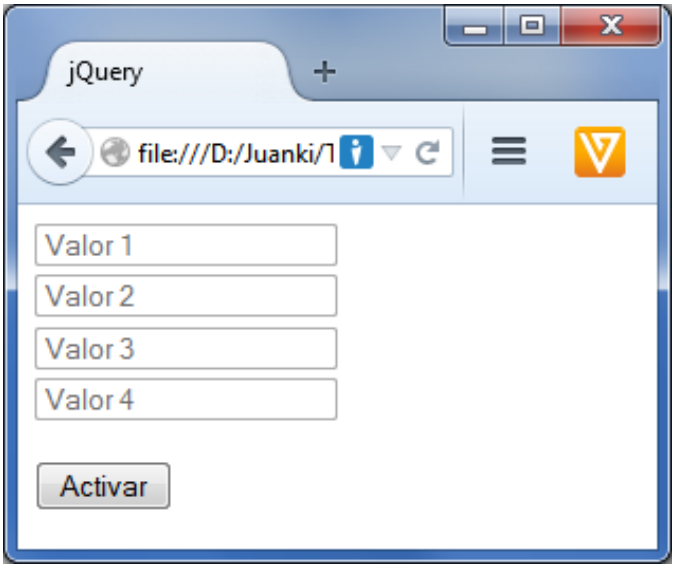
Este método devuelve un objeto jQuery.

Igualmente es posible agregar una propiedad por una función.

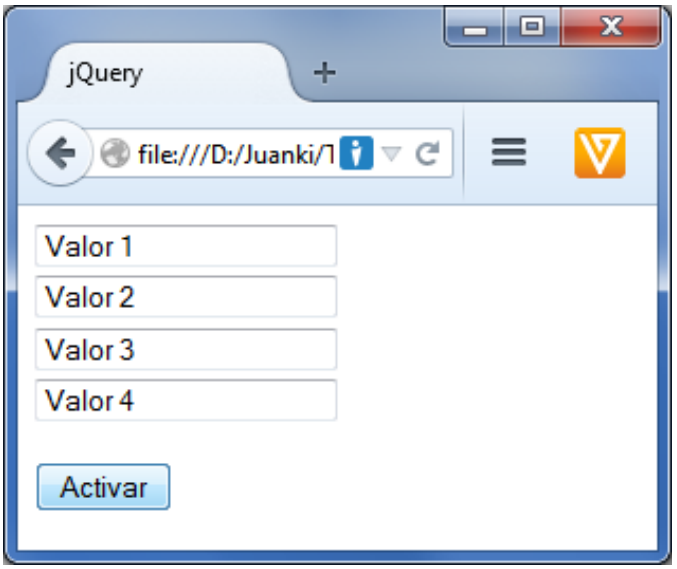
Ejemplo

Permitamos que con un clic de un botón se activen las líneas de texto que se han desactivado en el código Html.

Al cargar la página, las líneas de texto se muestran en gris.



Al hacer clic en el botón:




```
<!doctype html>
<html lang="es">
<head>
```

```
<meta charset="utf-8">
<title>jQuery</title>
<style>
button { margin-top: 15px;}
input { padding-left: 4px;
        margin-bottom: 4px;}
</style>
<script src="jquery.js"></script>
</head>
<body>
<input type="text" disabled="disabled" value="Valor 1"><br>
<input type="text" disabled="disabled" value="Valor 2"><br>
<input type="text" disabled="disabled" value="Valor 3"><br>
<input type="text" disabled="disabled" value="Valor 4"><br>
<button>Activar</button>
<p></p>
<script>
$(document).ready(function(){
$("button").click(function(){
$("input").prop("disabled",false);
});
});
</script>
</body>
</html>
```

Eliminar la propiedad de un elemento

removeProp(propiedad)

Elimina una propiedad de los elementos.

`$("input[type='checkbox']").removeProp("checked");` elimina la propiedad `checked`de la casilla de verificación.

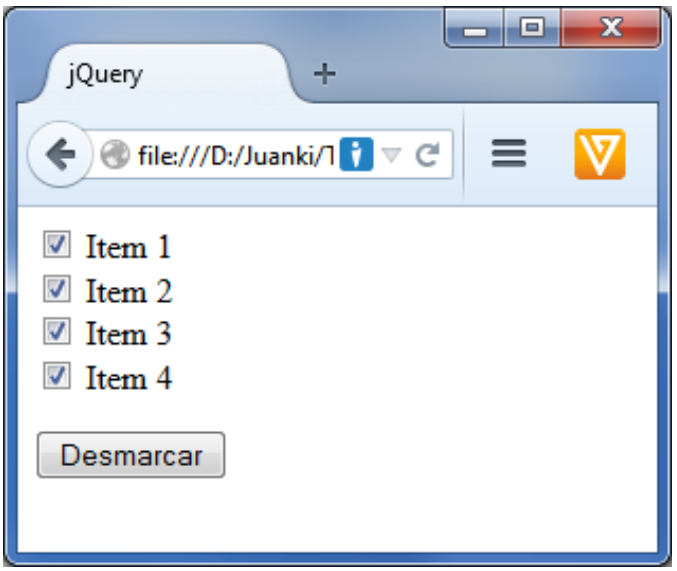
La documentación de jQuery especifica que para propiedades como `checked`, `disabled` o`selected`, una vez se eliminado la propiedad con `removeProp()`, no será posible volver a agregarla. Es más correcto el uso del método `prop()` con un valor `false`.

Este método devuelve un objeto jQuery.

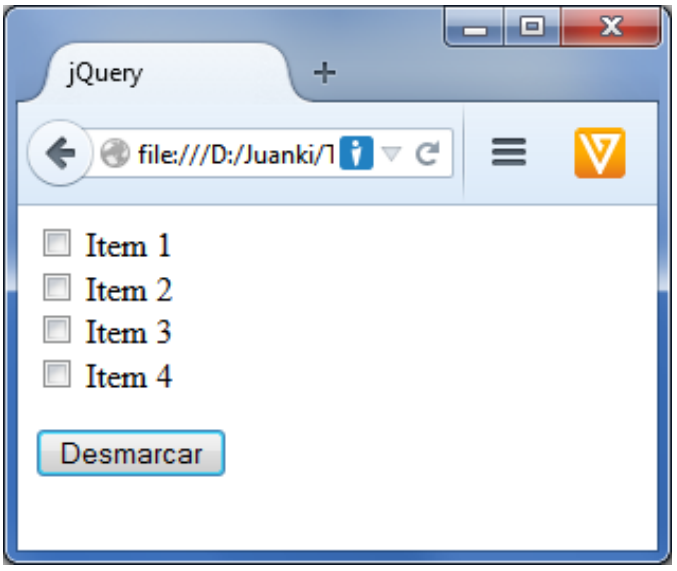
Ejemplo

Tomemos como ejemplo casillas de verificación checkbox. Con código jQuery, en un primer momento estas casillas estarán marcadas. Al hacer clic con un botón, las desmarcaremos.

Al cargar la página:



Al hacer clic con el botón:




```
<!doctype html>
```

```
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
button{ margin-top: 15px;}
</style>
<script src="jquery.js"></script>
</head>
<body>
<input type="checkbox"> Item 1<br>
<input type="checkbox"> Item 2<br>
<input type="checkbox"> Item 3<br>
<input type="checkbox"> Item 4<br>
<button>Desmarcar</button>
<p></p>
<script>
$(document).ready(function(){
$("input").prop("checked", true);
$("button").click(function(){
$("input").removeProp("checked");
});
});
</script>
</body>
</html>
```

Comentario

```
$("input").prop("checked", true);
```

Al cargar la página, las casillas de verificación (`$("input")`) se activan (`prop("checked", true)`).

```
$("button").click(function(){
$("input").removeProp("checked");
});
```

Al hacer clic en el botón, se desmarcan las casillas (`removeProp("checked")`).

Introducción

¿Es necesario resaltar la importancia de las hojas de estilo en la escritura del código de las páginas Web? Con jQuery, la modificación dinámica de las propiedades de estilo CSS se hace más fácil de implementar.

El método `css()` que estudiamos en los tres primeros puntos de este capítulo recuerda al método`attr()` del capítulo anterior.

Acceder a una propiedad de estilo

El método `css()` se usa de tres maneras diferentes. La primera solo permite acceder a la propiedad de estilo CSS de un elemento dado.

css(nombre)

Permite acceder a una propiedad de estilo del primer elemento que se encuentra.

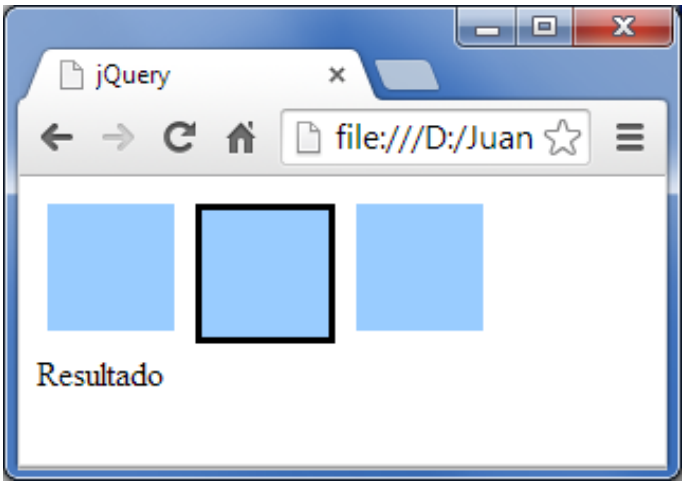
El nombre es una cadena de caracteres con la propiedad de estilo a la que se va a acceder.

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

Este método envía una cadena de caracteres (*string*).

Ejemplo

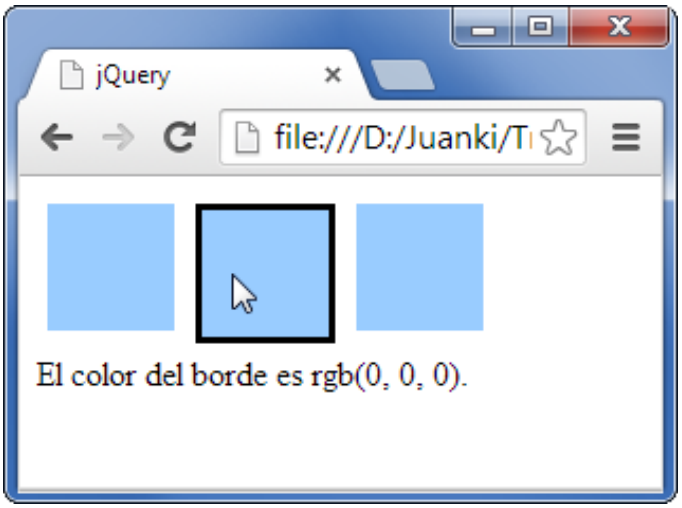
Supongamos que tenemos una página con tres elementos de tipo bloque.



El documento Html:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
div { width: 60px;
      height: 60px;
      margin: 5px;
      float: left;
      background-color:#9cf;"}
</style>
</head>
<body>
<div id="div1"></div>
<div id="div2" style="border: 3px solid black;"></div>
<div id="div3" style="visibility: visible;"></div>
<p style="clear: left;" id="resultado">Resultado</p>
</body>
</html>
```

Al hacer clic en un elemento de tipo bloque, el script mostrará respectivamente, de izquierda a derecha, el color de fondo, el color del borde y la visibilidad.



Tenemos el script jQuery:

```
<script>
$(document).ready(function(){
$("#div1").click(function () {
var color = $(this).css("background-color");
$("#resultado").html("El color es <span>" + color +
"</span>.");
});
$("#div2").click(function () {
var borde = $(this).css("border-color");
```

```
$("#resultado").html("El color del borde es <span>" +
borde + "</span>.");
});
$("#div3").click(function () {
var visibility = $(this).css("visibility");
$("#resultado").html("La visibilidad es <span>" + visibility +
"</span>.");
});
});
</script>
```

Lo detallamos a continuación.

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

Tan pronto como se carga el DOM.

```
$("#div1").click(function () {
var color = $(this).css("background-color");
```

El clic del ratón en el primer cuadrado (`id="div1"`) carga en la variable `color` la propiedad del color de fondo de este elemento, usando `css("background-color")`.

```
$("#resultado").html("El color es <span>" + color + "</span>.");
});
```

El valor de la propiedad de estilo se muestra como Html en la etiqueta `<p>`, identificada por`resultado`.

```
$("#div2").click(function () {
var border = $(this).css("border-color");
```

El clic del ratón en el segundo cuadrado (`id="div2"`) carga en la variable `borde` el color del borde de este elemento, usando `css("border-color")`.

```
$("#resultado").html("El color del borde es <span>" + border +
"</span>.");
});
```

El valor se mostrará en la página Html.

```
$("#div3").click(function () {
var visibility = $(this).css("visibility");
```

El clic del ratón en el tercer cuadrado (`id="div3"`) carga en la variable `visibility` el estado de visibilidad de este elemento, usando `css("visibility")`.

```
$("#resultado").html("La visibilidad es <span>" + visibility +
"</span>.");
});
```

El valor se mostrará en la página.

```
});
```

Fin del script.

El archivo final es:

```
<!doctype html>
<html lang="es">
<head>
```

```
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js" ></script>
<script>
$(document).ready(function(){
$("#div1").click(function () {
var color = $(this).css("background-color");
$("#resultado").html("El color es <span>" + color +
"</span>.");
});
$("#div2").click(function () {
var borde = $(this).css("border-color");
$("#resultado").html("El color del borde es <span>" +
borde + "</span>.");
});
$("#div3").click(function () {
var visibility = $(this).css("visibility");
$("#resultado").html("La visibilidad es <span>" + visibility +
"</span>.");
});
});
</script>
<style>
div { width:60px;
      height:60px;
      margin:5px;
      float:left;
      background-color:#9cf;"}
</style>
</head>
<body>
<div id="div1"></div>
<div id="div2" style="border: 3px solid black;"></div>
<div id="div3" style="visibility: visible;"></div>
<p style="clear: left;" id="resultado">Resultado </p>
</body>
</html>
```

Modificar las propiedades de estilo

La función `css()`, que admite parámetros, también permite modificar las propiedades de estilo de los elementos de la página.

css({propiedad de estilo})

Modifica las propiedades de estilo de un elemento dado usando la notación CSS clave/valor para las propiedades de estilo que se desea transformar.

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

Observe la presencia de las etiquetas habituales para las declaraciones de estilo.

Si la clave contiene un guion de unión, como por ejemplo `background-color`, ésta se debe ubicar entre comillas (`"background-color"`).

También se puede adoptar la notación JavaScript (*CamelCase*), es decir `backgroundColor`, en lugar de `background-color`.

Este método envía un objeto jQuery.

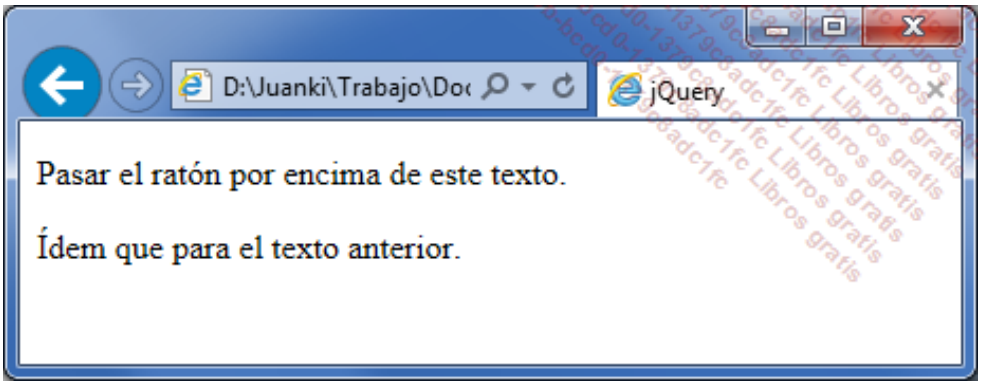
Desde la versión 1.6 de jQuery, podemos usar valores relativos para modificar el valor de una propiedad de estilo CSS. Esto se puede codificar con "+=" o "-=", con respecto al valor actual.

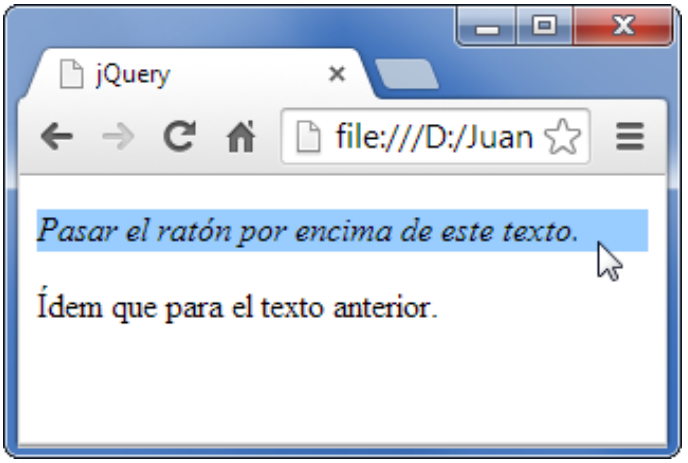
```
$ ("#item").css("left", "+=10px")
```

Aumentar 10 píxeles el desplazamiento a la izquierda del elemento.

Ejemplo

Vamos a ilustrar este método de jQuery con un ejemplo. Al pasar el ratón por encima de un párrafo, éste toma un color de fondo y el tipo de letra se muestra en itálica.





```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
p { font: Arial sans-serif;}
</style>
</head>
<body>
<p>Pasar el ratón por encima de este texto.</p>
<p>Ídem que para el texto anterior.</p>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$("p").mouseover(function () {
$(this).css({'background-color': '#9cf', 'font-style':
'italic'});
});
$("p").mouseout(function () {
$(this).css({'background-color': '', 'font-style': ''});
});
});
</script>
```

Explicaciones:

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

Carga del DOM.

```
$("p").mouseover(function () {
$(this).css({'background-color': '#9cf', 'font-style': 'italic'});
});
```

Al pasar el ratón por encima (`mouseover()`) del párrafo `<p>`, la función `css()` modifica su color de fondo y pone los caracteres en itálica (`css({'background-color': '#9cf', 'font-style': 'italic'})`).

```
$("p").mouseout(function () {
$(this).css({'background-color': '', 'font-style': ''});
});
```

Cuando el cursor sale del párrafo (`mouseout()`), las modificaciones se anulan.

```
});
```

Fin del script.

Las propiedades que se usan según la notación CSS también se pueden usar en notación JavaScript. El script sería:

```
<script>
$(document).ready(function(){
$("p").mouseover(function () {
$(this).css({backgroundColor: '#9cf', fontStyle: 'italic'});
});
$("p").mouseout(function () {
```

```
$(this).css({backgroundColor: '', fontStyle: ''});
});
});
</script>
```

El documento completo es:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js" ></script>
<script>
$(document).ready(function(){
$("p").mouseover(function () {
$(this).css({'background-color': '#9cf', 'font-style':
'italic'});
});
$("p").mouseout(function () {
$(this).css({'background-color': '', 'font-style': ''});
});
});
</script>
<style>
p { font: Arial sans-serif;}
</style>
</head>
<body>
<p>Pasar el ratón por encima de este texto.</p>
<p>Ídem que para el texto anterior.</p>
</body>
</html>
```

Desde la versión 1.4 de jQuery, es posible modificar una propiedad de estilo de los elementos de la selección usando una función.

En ese caso, la sintaxis es:

css(nombre de la propiedad, función(índice, valor actual))

Donde:

- "función" especifica una función que devuelve el nuevo valor.
- "índice" (opcional) es la posición del índice del elemento seleccionado.
- "valor actual" (opcional) es el valor actual o el anterior.

Ejemplo:

```
$(this).css({top: function(índice, valor) {
return parseFloat(valor) * 1.2;
}
});
```

Asignar las propiedades de estilo

La función `css()` de jQuery propone una última manera de codificar las transformaciones de las propiedades de estilo.

css(clave,valor)

Modifica las propiedades de estilo de un elemento dado usando la notación clave/valor para las propiedades de estilo que se van a transformar.

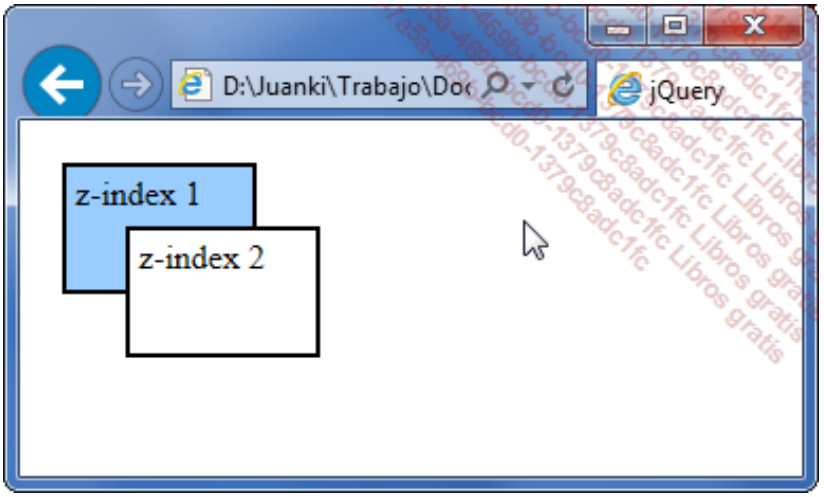
- "clave" (cadena de caracteres) corresponde al nombre de la propiedad de estilo que se va a modificar.
- "valor" (cadena de caracteres o número) es el nuevo valor de la propiedad. Si se especifica un número, jQuery lo convierte automáticamente a píxeles.

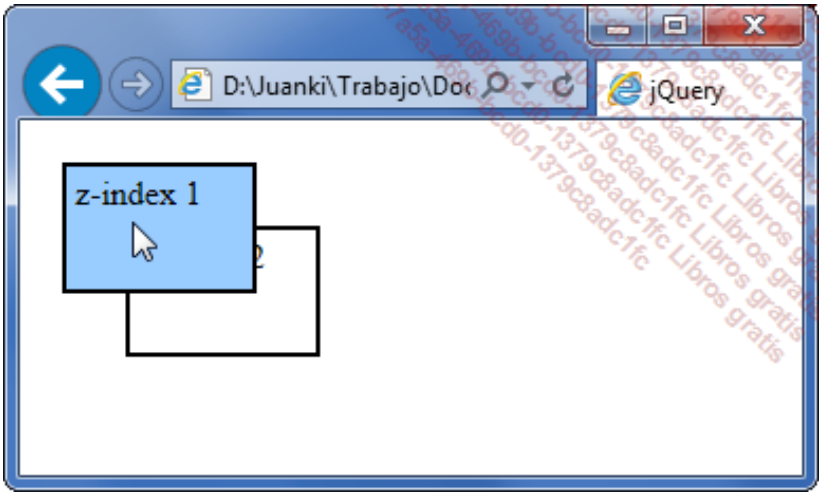
```
$("p").css("color","red");
```

Este método envía un objeto jQuery.

Ejemplo

Tomemos dos capas superpuestas. Con un script jQuery, vamos a modificar el orden de estas capas al pasar por encima el ratón.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
```

```
<title>jQuery</title>
<style>
#caja1 { position: absolute;
         left: 20px;
         top: 20px;
         width: 80px;
         height: 50px;
         padding: 4px;
         border: 2px solid black;
         background-color: #9cf;}
#caja2 { position: absolute;
         left: 50px;
         top: 50px;
         width: 80px;
         height: 50px;
         padding: 4px;
         border: 2px solid black;
         background-color: white;}
</style>
</head>
<body>
<div id="caja1">
z-index 1
</div>
<div id="caja2">
z-index 2
</div>
</body>
</html>
```

Observemos que ninguna propiedad de estilo relativa a la superposición de las capas (`z-index`) está presente en el código del archivo de inicio.

El script jQuery:

```
<script>
$(document).ready(function(){
$("#caja1").mouseover(function () {
$(this).css('z-index' , '10');
});
$("#caja1").mouseout(function () {
$(this).css('z-index' , '');
});
});
</script>
```

Explicaciones:

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

Cuando se carga el DOM.

```
$("#caja1").mouseover(function () {
$(this).css('z-index' , '10');
});
```

Al pasar el ratón por encima (`mouseover()`), a la capa cuyo identificador es `caja1` se le asigna un valor para `z-index` de `10`, lo que la pone al frente con respecto a la capa identificada por `caja2`.

```
$("#caja1").mouseout(function () {
$(this).css('z-index' , '');
});
```

Cuando el cursor sale de la capa `caja1`, se restaura el valor por defecto de `z-index`.

```
});
```

Fin del script.

El documento completo es:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js" ></script>
<script>
$(document).ready(function(){
$("#caja1").mouseover(function () {
$(this).css('z-Index' , '10');
});
$("#caja1").mouseout(function () {
$(this).css('z-Index' , '');
});
});
</script>
<style>
#caja1 { position: absolute;
         left: 20px;
         top: 20px;
         width: 80px;
         height: 50px;
         padding: 4px;
         border: 2px solid black;
         background-color: #9cf;}
#caja2 { position: absolute;
         left: 50px;
         top: 50px;
         width: 80px;
         height: 50px;
         padding: 4px;
         border: 2px solid black;
         background-color: white;}
</style>
</head>
<body>
<div id="caja1">
z-index 1
</div>
<div id="caja2">
z-index 2
</div>
</body>
</html>
```

El dimensionamiento

jQuery propone una serie de métodos relativos a la dimensión de los elementos.

height()

Devuelve la altura, expresada en píxeles, de un elemento.

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

Este método devuelve una cadena de caracteres (String).

height(valor)

Asigna una altura a los elementos especificados. Si no se especifica la unidad (como `em` o `%`), la unidad por defecto será `px`.

- "valor" (cadena de caracteres o entero): valor de la altura que se asigna.

```
$("p").height(120);
```

Este método devuelve un objeto jQuery.

Después de la altura, la anchura…

width()

Devuelve la anchura, expresada en píxeles, de un elemento.

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

Este método devuelve una cadena de caracteres (String).

width(valor)

Asigna una anchura a los elementos especificados. Si no se especifica ninguna unidad (como `em` o `%`), la unidad por defecto será `px`.

- "valor" (cadena de caracteres o entero): valor de la anchura asignada.

```
$("p").width(120);
```

Este método devuelve un objeto jQuery.

Desde la versión 1.4 de jQuery, es posible usar una función para modificar la altura y la anchura de los elementos de una selección.

En este caso, la sintaxis sería:

height(función(índice, valor actual))

width(función(índice, valor actual))

Donde:

- "función" especifica una función que devuelve el nuevo valor de la altura o de la anchura.
- "índice" (opcional) es la posición del índice del elemento seleccionado.
- "valor actual" (opcional) es el valor actual o el anterior.

Ejemplo:

```
$("div").height(30)
```

Fija la altura de las capas a 30 píxeles.

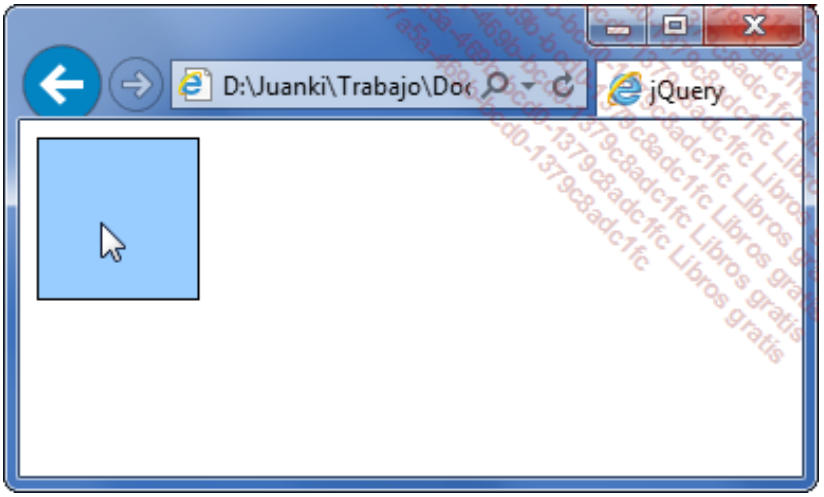
Vamos a mencionar algún método más:

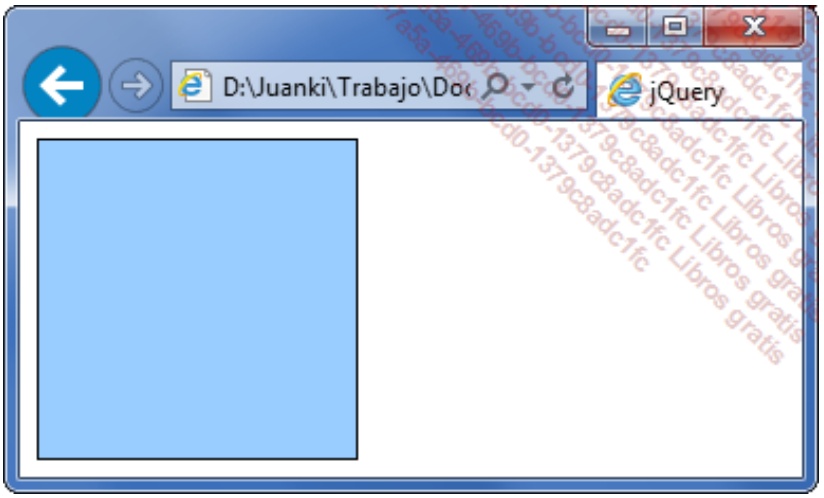
- **innerHeight()**: devuelve la altura interior (el borde no está incluido, pero sí el padding) del primer elemento de la selección.
- **innerWidth()**: recupera la anchura interior (el borde no está incluido, pero sí el padding) del primer elemento de la selección.
- **outerHeight(options)**: devuelve la altura exterior (por defecto se incluyen el borde y el padding) del primer elemento de la selección.
- **outerWidth(options)**: devuelve la anchura exterior (por defecto se incluyen el borde y el padding) del primer elemento de la selección.

Estos métodos funcionan tanto para los elementos visibles como para los no visibles.

Ejemplo

Al hacer clic en un elemento caja, un script jQuery detecta la anchura y la altura, y se multiplican por dos estas dimensiones.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
div { position: relative;
      width: 75px;
      height: 75px;
      border: 1px solid black;
```

```
      background-color: #9cf;}
</style>
</head>
<body>
<div></div>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$("div").click(function () {
altura=$(this).height()
anchura=$(this).width()
$(this).height(altura*2).width(anchura*2);
});
});
</script>
```

Explicaciones:

```
$(document).ready(function(){
$("div").click(function () {
```

Después de la carga y al hacer clic en la caja.

```
altura=$(this).height()
anchura=$(this).width()
```

La altura y la anchura del elemento seleccionado se almacenan en una variable.

```
$(this).height(altura*2).width(anchura*2);
```

La altura y la anchura del elemento se multiplican por 2.

```
});
});
```

Fin del script.

El documento final se presenta a continuación:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js" ></script>
<script>
$(document).ready(function(){
$("div").click(function () {
altura=$(this).height();
anchura=$(this).width();
$(this).height(altura*2).width(anchura*2);
});
});
</script>
<style>
div { position: relative;
      width: 75px;
      height: 75px;
```

```
      border: 1px solid black;
      background-color: #9cf;}
</style>
</head>
<body>
<div></div>
</body>
</html>
```

El posicionamiento

De manera paralela a los métodos de jQuery relacionados con los elementos, jQuery tiene métodos para determinar la posición de éstos.

position()

Devuelve el valor `top` y `left` de la posición de un elemento relativo a su elemento padre.

```
$("p:first").position();
```

Este método devuelve un objeto de tipo `object{top,left}`.

offset()

Devuelve el valor `top` y `left` de la posición de un elemento relativo al documento.

```
$("p:first").offset();
```

Este método devuelve un objeto de tipo `object{top,left}`.

Los métodos `scrollTop(valor)` y `scrollLeft(valor)` son más particulares, ya que permiten modificar, para un elemento, su separación con respecto al borde superior o al borde izquierdo. En cierto modo, estos métodos permiten controlar la amplitud de la barra de desplazamiento vertical y horizontal.

scrollTop(valor)

Modifica la separación (en píxeles) entre el borde superior del documento (`top`) y el elemento seleccionado, tomando el valor que se pasa en el argumento.

valor: número positivo que representa la nueva distancia que se quiere aplicar (en píxeles).

```
$("div").scrollTop(300);
```

Este método devuelve un objeto jQuery.

scrollLeft(valor)

Modifica la separación (en píxeles) entre el borde izquierdo del documento (`left`) y el elemento seleccionado, tomando el valor que se pasa en el argumento.

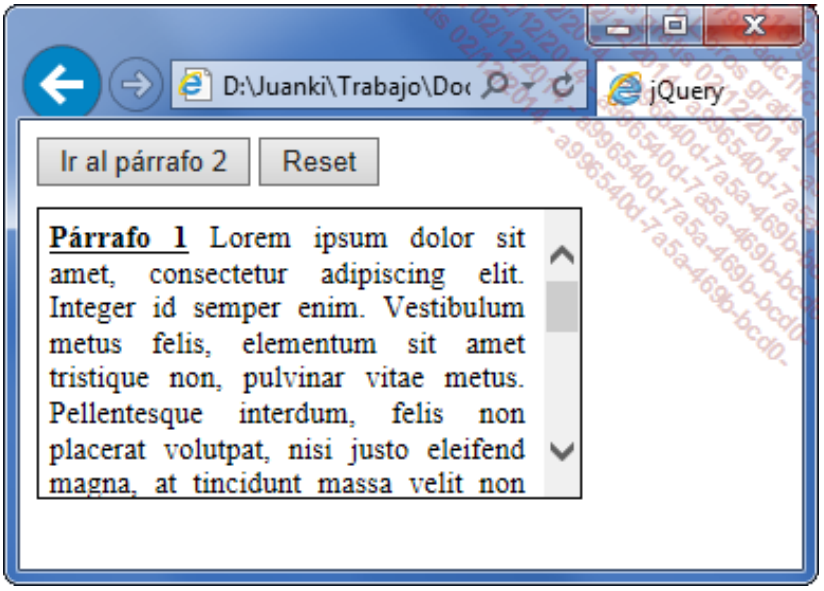
valor: número positivo que representa la nueva distancia que se quiere aplicar (en píxeles).

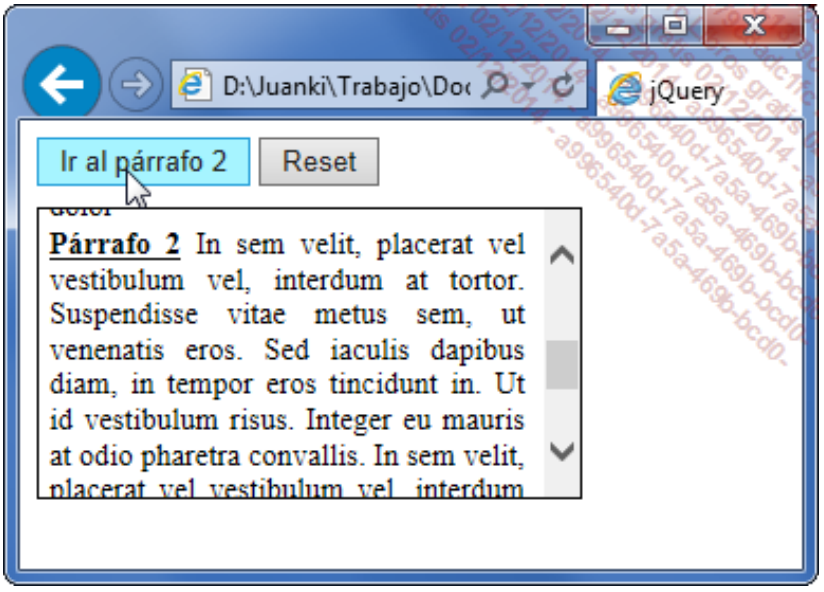
```
$("div").scrollLeft(300);
```

Este método devuelve un objeto jQuery.

Ejemplo

Vamos a ilustrar esto con un ejemplo. Al hacer clic en un botón, vamos a permitir al usuario ir directamente al segundo párrafo de un texto.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
div { position: relative;
      width: 250px;
      height: 130px;
      border: 1px solid black;
      margin-top: 10px;
      padding: 3px;
      font-size: 0.9em;
      overflow: auto;}
p { margin:2px;
    width:225px;
    text-align: justify;}
span { font-weight: bold;
       text-decoration: underline}
</style>
</head>
<body>
<button id="go">Ir al párrafo 2</button>
<button id="reset">Reset</button>
```

```
<div class="demo">
<p><span>Párrafo 1</span> Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Integer id semper enim. Vestibulum
metus felis, elementum sit amet tristique non, pulvinar vitae
metus. Pellentesque interdum, felis non placerat volutpat, nisi
justo eleifend magna, at tincidunt massa velit non dolor</p>
<p><span>Párrafo 2</span> In sem velit, placerat vel vestibulum
vel, interdum at tortor. Suspendisse vitae metus sem, ut venenatis
eros. Sed iaculis dapibus diam, in tempor eros tincidunt in. Ut id
vestibulum risus. Integer eu mauris at odio pharetra convallis.
In sem velit, placerat vel vestibulum vel, interdum at tortor.
Suspendisse vitae metus sem, ut venenatis eros. Sed iaculis
dapibus diam, in tempor eros tincidunt in. Ut id vestibulum risus.
Integer eu mauris at odio pharetra convallis.</p>
</div>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$("#go").click(function () {
$("div").scrollTop(148);
});
$("#reset").click(function () {
$("div").scrollTop(0);
});
});
</script>
```

Explicación del script:

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

Cuando se carga el DOM.

```
$("#go").click(function () {
$("div").scrollTop(148);
});
```

Al hacer clic en el botón, se modifica la posición del texto con respecto al borde superior de la capa, para ir al segundo párrafo.

```
$("#reset").click(function () {
$("div").scrollTop(0);
});
```

El botón Reset restaura la separación hacia arriba a su posición inicial, para que aparezca de nuevo el primer párrafo.

```
});
```

Fin del script.

El documento completo se presenta a continuación.

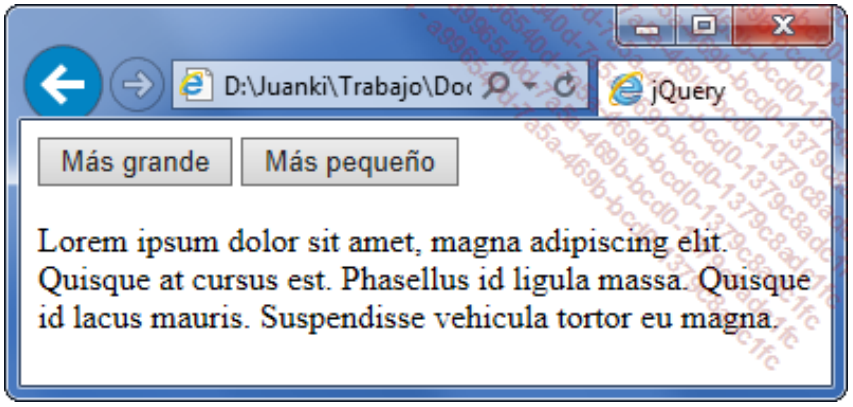
```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
```

```
<script src="jquery.js" ></script>
<script>
$(document).ready(function(){
$("#go").click(function () {
$("div").scrollTop(148);
});
$("#reset").click(function () {
$("div").scrollTop(0);
});
});
</script>
<style>
div { position: relative;
      width: 250px;
      height: 130px;
      border: 1px solid black;
      margin-top: 10px;
      padding: 3px;
      font-size: 0.9em;
      overflow: auto;}
p { margin:2px;
    width:225px;
    text-align: justify;}
span { font-weight: bold;
       text-decoration: underline}
</style>
</head>
<body>
<button id="go">Ir al párrafo 2</button>
<button id="reset">Reset</button>
<div class="demo">
<p><span>Párrafo 1</span> Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Integer id semper enim. Vestibulum
metus felis, elementum sit amet tristique non, pulvinar vitae
metus. Pellentesque interdum, felis non placerat volutpat, nisi
justo eleifend magna, at tincidunt massa velit non dolor</p>
<p><span>Párrafo 2</span> In sem velit, placerat vel vestibulum
vel, interdum at tortor. Suspendisse vitae metus sem, ut venenatis
eros. Sed iaculis dapibus diam, in tempor eros tincidunt in. Ut id
vestibulum risus. Integer eu mauris at odio pharetra convallis.
In sem velit, placerat vel vestibulum vel, interdum at tortor.
Suspendisse vitae metus sem, ut venenatis eros. Sed iaculis
dapibus diam, in tempor eros tincidunt in. Ut id vestibulum risus.
Integer eu mauris at odio pharetra convallis.</p>
</div>
</body>
</html>
```

Aplicaciones

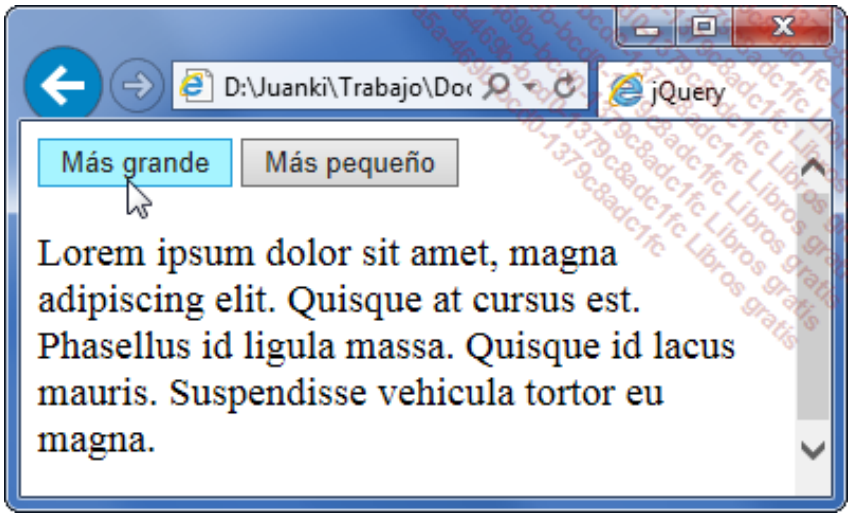
1. Redimensionar el tamaño de los caracteres

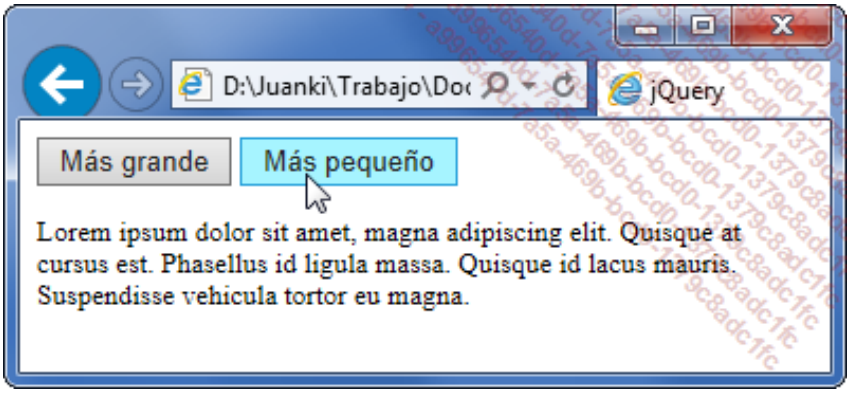
Vamos a permitir al usuario modificar el tamaño de los caracteres a su voluntad, para mejorar la comodidad de la lectura.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
p { font-size: 1em;}
</style>
</head>
<body>
<input type="button" value="Más grande" id="mas">
<input type="button" value="Más pequeño" id="menos">
<p>Lorem ipsum dolor sit amet, magna adipiscing elit. Quisque at
cursus est. Phasellus id ligula massa. Quisque id lacus mauris.
Suspendisse vehicula tortor eu magna.</p>
</body>
</html>
```

Al hacer clic en un botón, el script jQuery va a permitir aumentar el tamaño de los caracteres. Existe un segundo botón para disminuirlo.






```
<script>
$(document).ready(function(){
$('input').click(function(){
var texto = $('p');
var tamaño = texto.css('fontSize');
var numero = parseFloat(tamaño, 10);
var unidad = tamaño.slice(-2);
if(this.id == 'mas') {
numero *= 1.2;
}
else if (this.id == 'menos'){
numero /=1.2;
}
texto.css('fontSize', numero + unidad);
});
});
</script>
```

Explicaciones:

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

Cuando se carga el DOM.

```
$('input').click(function(){
```

Al hacer clic en uno de los botones.

```
var texto = $('p');
var tamaño = texto.css('fontSize');
```

El tamaño de caracteres (`fontSize`) del texto se almacena en una variable (`tamaño`).

```
var numero = parseFloat(tamaño, 10);
```

La función JavaScript `parseFloat()` convierte esta cadena de caracteres en un número (base 10).

```
var unidad = tamaño.slice(-2);
```

Con la función `slice()` de jQuery, obtenemos la unidad que se usa para el tamaño de caracteres (`px` o `em`). El valor negativo permite empezar desde el final de la selección.

```
if(this.id == 'mas')
{numero *= 1.2;
}
```

Si se pulsa el botón "mas", el tamaño de los caracteres se aumenta un 20%.

```
else if (this.id == 'menos')
{numero /=1.2;
}
```

Si se activa el botón "menos", el tamaño de caracteres disminuye en la misma proporción.

```
texto.css('fontSize', numero + unidad);
```

Se asigna el nuevo valor del tamaño de caracteres.

```
});
});
```

Fin del script.

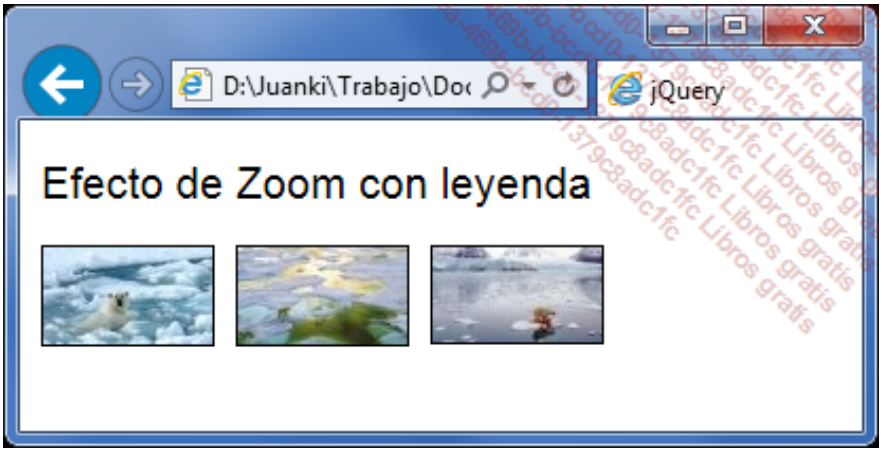
Nuestro archivo es:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$('input').click(function(){
var texto = $('p');
var tamaño = texto.css('fontSize');
var numero = parseFloat(tamaño, 10);
var unidad = tamaño.slice(-2);
if(this.id == 'mas') {
numero *= 1.2;
}
else if (this.id == 'menos'){
numero /=1.2;
}
texto.css('fontSize', numero + unidad);
});
});
</script>
<style>
p { font-size: 1em;}
</style>
</head>
<body>
<input type="button" value="Más grande" id="mas">
<input type="button" value="Más pequeño" id="menos">
<p>Lorem ipsum dolor sit amet, magna adipiscing elit. Quisque at
cursus est. Phasellus id ligula massa. Quisque id lacus mauris.
Suspendisse vehicula tortor eu magna.</p>
</body>
</html>
```

2. Zoom sobre una imagen con una leyenda

Imaginemos una página que presenta las imágenes en miniatura. Al pasar el ratón por encima de una imagen, ésta se mostrará en su tamaño real. El script ofrece la posibilidad de añadir una leyenda a la imagen.

Las imágenes de esta aplicación están disponibles para descarga en la página Información.



El archivo Html inicial es:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
body { margin: 0px;
       padding: 10px;
       font:80% Arial, sans-serif;}
h2 { clear:both;
     font-size:160%;
     font-weight:normal;
     margin: 0px;
     padding-top: 10px;
     padding-bottom: 15px;}
a { text-decoration:none;
    color:black;}
img { border: 1px solid black;}
ul,li { margin: 0px;
        padding: 0px;}
li { list-style: none;
```

```
     float: left;
     display: inline;
     margin-right: 10px;}
#zoom { position: absolute;
        border: 1px solid #333;
        background: #333;
        padding: 3px;
        display: none;
        color: #fff;}
</style>
</head>
<body>
<h2>Efecto de Zoom con leyenda</h2>
<ul>
<li><a href="oso1.jpg" class="zoom" title="Oso polar en
peligro"><img src="pt_oso1.jpg" alt=""></a></li>
<li><a href="oso2.jpg" class="zoom" title="Oso polar en
peligro"><img src="pt_oso2.jpg" alt=""></a></li>
<li><a href="oso3.jpg" class="zoom" title="Oso polar en
peligro"><img src="pt_oso3.jpg" alt=""></a></li>
</ul>
</body>
</html>
```

Comentarios

La leyenda que aparece en la imagen agrandada se toma del atributo title del enlace.

Gracias al enlace a la imagen en su dimensión inicial, la página permanece accesible a los usuarios que hubieran desactivado el JavaScript.

El script jQuery:

```
<script>
this.zoom_imagen = function(){
xOffset = 10;
yOffset = 30;
$("a.zoom").hover(function(e){
this.texto = this.title;
this.title = "";
var leyenda = (this.texto != "") ? "<br/>" + this.texto: "";
$("body").append("<p id='zoom'><img src='"+ this.href +"'
alt='Visualización imagen' />"+ leyenda +"</p>");
$("#zoom")
.css("top",(e.pageY - xOffset) + "px")
.css("left",(e.pageX + yOffset) + "px")
.fadeIn("slow");
},
function(){
this.title = this.texto;
$("#zoom").remove();
});
$("a.zoom").mousemove(function(e){
$("#zoom")
.css("top",(e.pageY - xOffset) + "px")
.css("left",(e.pageX + yOffset) + "px");
});
};
$(document).ready(function(){
zoom_imagen();
});
</script>
```

Este script necesita algunas explicaciones.

```
this.zoom_imagen = function(){
xOffset = 10;
yOffset = 30;
```

Antes de ejecutar el jQuery y el efecto que aporta, será necesario escribir la función `zoom_imagen`. Vamos a empezar creando dos variables (`xOffset` e `yOffset`), que definen el desplazamiento horizontal y vertical del pop-up de la imagen con respecto al cursor. El diseñador de la página puede ajustar estos valores.

```
$("a.zoom").hover(function(e){
```

El script aplica el método `hover()` a los enlaces con la clase `zoom`. El evento que se asocia al cursor se transmite como parámetro a la función (`function(e)`).

```
this.texto = this.title;
this.title = "";
```

Al pasar el ratón por encima de la miniatura, el texto del atributo `title` se toma de la variable texto.

```
var leyenda = (this.texto != "") ? "<br>" + this.texto: "";
```

Si el atributo `title` no está vacío, la leyenda de la imagen (variable `leyenda`) se construye a partir del texto de éste.

```
$("body").append("<p id='zoom'><img src='"+ this.href +"'
alt='Visualización de la imagen'>"+ leyenda +"</p>");
```

El script inserta (`append()`) en el body un párrafo con el identificador `zoom`, que contiene la imagen cuya dirección se proporciona en el enlace (`img src='"+ this.href`), así como la leyenda.

```
$("#zoom")
.css("top",(e.pageY - xOffset) + "px")
.css("left",(e.pageX + yOffset) + "px")
.fadeIn("slow");
```

Este párrafo (`id='zoom'`) se muestra según las propiedades de estilo `top` y `left`, añadiendo el desplazamiento que se ha definido más arriba a la posición del cursor. Se añade un efecto de`fadeIn()`.

```
},
```

Fin de la primera función del método `hover()`.

```
function(){
this.title = this.texto;
$("#zoom").remove();
```

Cuando el cursor sale de la miniatura (`$("#zoom")`), el pop-up de la imagen desaparece.

```
});
```

Fin de la segunda función del método `hover()`.

```
$("a.zoom").mousemove(function(e){
```

Hay que prever que el usuario, siempre sobre la miniatura (`$("a.zoom")`), pueda mover el cursor del ratón (`mousemove()`). El evento asociado al cursor se transmite como parámetro a la función (`function(e)`).

```
$("#zoom")
```

```
.css("top",(e.pageY - xOffset) + "px")
.css("left",(e.pageX + yOffset) + "px");
});
```

El pop-up contiene la imagen que sigue a los movimientos del cursor.

```
};
```

Fin de hover().

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

Ahora que se han presentado todas las funciones, jQuery puede aplicar la función `zoom_imagen()` al cargar el DOM.

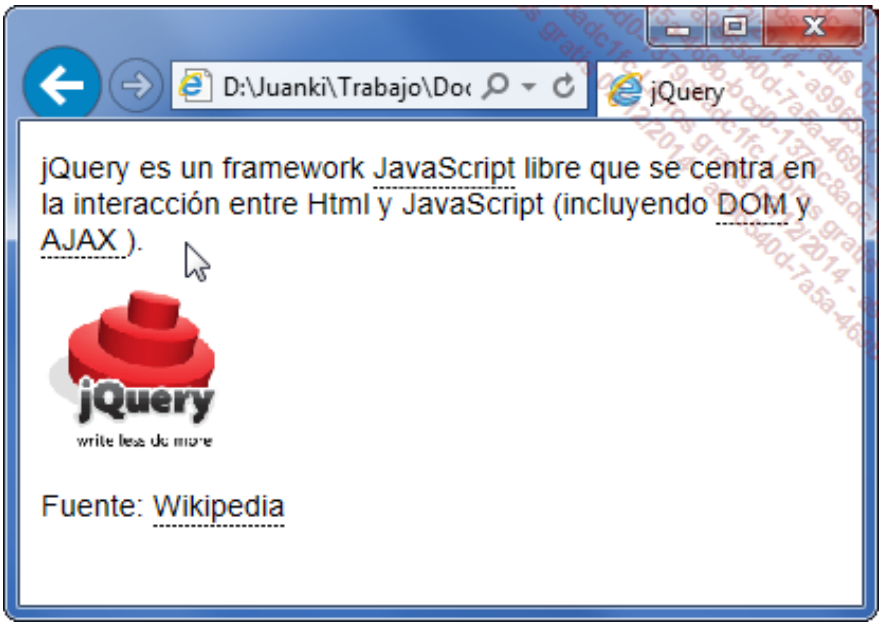
El archivo Html final es:

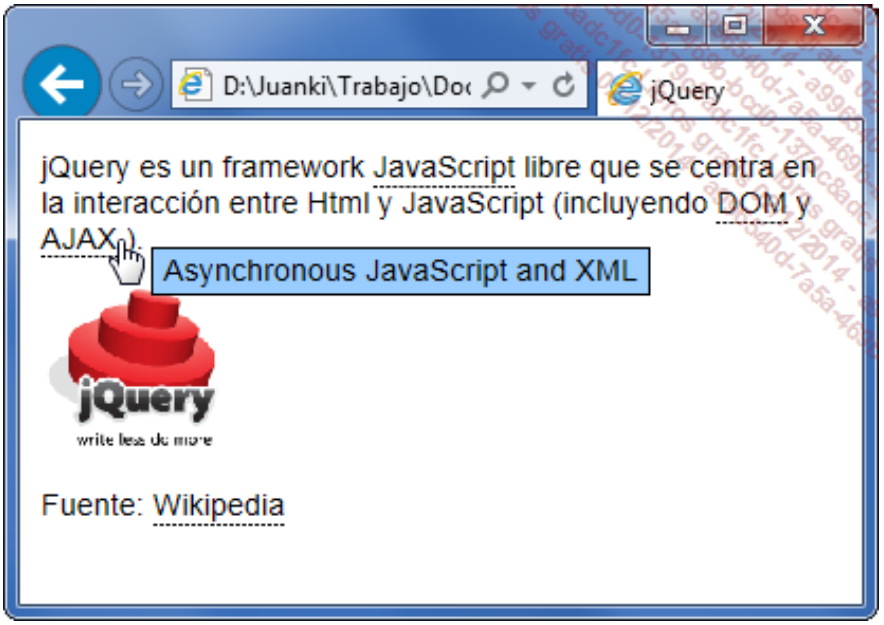
```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
this.zoom_imagen = function(){
xOffset = 10;
yOffset = 30;
$("a.zoom").hover(function(e){
this.texto = this.title;
this.title = "";
var leyenda = (this.texto != "") ? "<br>" + this.texto: "";
$("body").append("<p id='zoom'><img src='"+ this.href +"'
alt='Visualización de la imagen'>"+ leyenda +"</p>");
$("#zoom")
.css("top",(e.pageY - xOffset) + "px")
.css("left",(e.pageX + yOffset) + "px")
.fadeIn("slow");
},
function(){
this.title = this.texto;
$("#zoom").remove();
});
$("a.zoom").mousemove(function(e){
$("#zoom")
.css("top",(e.pageY - xOffset) + "px")
.css("left",(e.pageX + yOffset) + "px");
});
};
$(document).ready(function(){
zoom_imagen();
});
</script>
<style>
body { margin: 0px;
       padding: 10px;
       font:80% Arial, sans-serif;}
h2 { clear:both;
     font-size:160%;
     font-weight:normal;
     margin: 0px;
     padding-top: 10px;
     padding-bottom: 15px;}
```

```
a { text-decoration:none;
    color:black;}
img { border: 1px solid black;}
ul,li { margin: 0px;
        padding: 0px;}
li { list-style: none;
     float: left;
     display: inline;
     margin-right: 10px;}
#zoom { position: absolute;
        border: 1px solid #333;
        background: #333;
        padding: 3px;
        display: none;
        color: #fff;}
</style>
</head>
<body>
<h2>Efecto de Zoom con leyenda</h2>
<ul>
<li><a href="oso1.jpg" class="zoom" title="Oso polar en
peligro"><img src="pt_oso1.jpg" alt=""></a></li>
<li><a href="oso2.jpg" class="zoom" title="Oso polar en
peligro"><img src="pt_oso2.jpg" alt=""></a></li>
<li><a href="oso3.jpg" class="zoom" title="Oso polar en
peligro"><img src="pt_oso3.jpg" alt=""></a></li>
</ul>
</body>
</html>
```

3. Un tooltip con jQuery

Al pasar el ratón sobre algunas palabras, nostramos un tooltip.






```
doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
body { margin: 0;
       padding-left: 10px;
       padding-right: 10px;
       font: 90% Arial, sans-serif;}
a { text-decoration: none;
    border-bottom: 1px dashed black;
    color: black;}
#tooltip { position: absolute;
           border: 1px solid black;
           background: #9cf;
           padding:2px 5px;
           display: none;}
</style>
</head>
<body>
<p>jQuery es un framework <a href="#" class="tooltip"
title="Lenguaje de scripts que se usa en las páginas web
interactivas">JavaScript</a> libre que se centra en la interacción
entre Html y JavaScript (incluyendo <a href="#"
class="tooltip" title="Document Object Model">DOM</a> y <a
href="#" class="tooltip" title="Asynchronous JavaScript and
XML">AJAX </a>).</p>
<p><img border="0" src="jquery-logo.gif" alt=""></p>
<p>Fuente: <a href="#" class="tooltip" title="Enciclopedia
universal disponible en la Web">Wikipedia</a></p>
</body>
</html>
```

Observemos que el atributo `title` del enlace contiene el texto del tooltip.

El script jQuery:

```
<script>
this.popup = function(){
xOffset = 10;
yOffset = 20;
```

```
$("a.tooltip").hover(function(e){
this.texto = this.title;
this.title = "";
$("body").append("<p id='tooltip'>"+ this.texto +"</p>");
$("#tooltip")
.css("top",(e.pageY - xOffset) + "px")
.css("left",(e.pageX + yOffset) + "px")
.fadeIn("fast");
},
function(){
this.title = this.texto;
$("#tooltip").remove();
});
$("a.tooltip").mousemove(function(e){
$("#tooltip")
.css("top",(e.pageY - xOffset) + "px")
.css("left",(e.pageX + yOffset) + "px");
});
};
$(document).ready(function(){
popup();
});
</script>
```

Vamos a explicar este script, que es muy parecido a la aplicación anterior.

```
this.popup = function(){
xOffset = 10;
yOffset = 20;
```

Aplicación de la función `popup`, que define el desplazamiento horizontal y vertical.

```
$("a.tooltip").hover(function(e){
```

Al pasar el ratón por encima de un enlace con una clase tooltip (`$("a.tooltip")`), se aplica un método jQuery `hover()`.

```
this.texto = this.title;
this.title = "";
$("body").append("<p id='tooltip'>"+ this.texto +"</p>");
$("#tooltip")
.css("top",(e.pageY - xOffset) + "px")
.css("left",(e.pageX + yOffset) + "px")
.fadeIn("fast");
},
```

Cuando el cursor se sitúa en el enlace, el contenido del atributo `title` del enlace se toma de la variable `texto`. El script añade (`append()`) al cuerpo (`body`) del documento un párrafo cuyo identificador es `tooltip`, que contiene la variable `texto` y se muestra en la página con las coordenadas `top` y `left`. Esta visualización se hace con un efecto (`fadeIn("fast")`).

```
function(){
this.title = this.texto;
$("#tooltip").remove();
});
```

Cuando el cursor sale del enlace, el tooltip se elimina (`remove()`).

```
$("a.tooltip").mousemove(function(e){
$("#tooltip")
.css("top",(e.pageY - xOffset) + "px")
.css("left",(e.pageX + yOffset) + "px");
});
```

```
};
```

Vamos a prever el caso de que el usuario mueva ligeramente el cursor (`mousemove()`) cuando está situado en el enlace. Esto permite al script acompañar estos movimientos.

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

Ahora que están definidas todas las funciones, el script jQuery puede activar la función `popup()`.

El documento final es:

```
doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>

this.popup = function(){
xOffset = 10;
yOffset = 20;
$("a.tooltip").hover(function(e){
this.texto = this.title;
this.title = "";
$("body").append("<p id='tooltip'>"+ this.texto +"</p>");
$("#tooltip")
.css("top",(e.pageY - xOffset) + "px")
.css("left",(e.pageX + yOffset) + "px")
.fadeIn("fast");
},
function(){
this.title = this.texto;
$("#tooltip").remove();
});
$("a.tooltip").mousemove(function(e){
$("#tooltip")
.css("top",(e.pageY - xOffset) + "px")
.css("left",(e.pageX + yOffset) + "px");
});
};
$(document).ready(function(){
popup();
});

</script>
<style>
body { margin: 0;
       padding-left: 10px;
       padding-right: 10px;
       font: 90% Arial, sans-serif;}
a { text-decoration: none;
    border-bottom: 1px dashed black;
    color: black;}
#tooltip { position: absolute;
           border: 1px solid black;
           background: #9cf;
           padding:2px 5px;
           display: none;}
</style>
</head>
<body>
```

```
<p>jQuery es un framework <a href="#" class="tooltip"
title="Lenguaje de scripts que se usa en las páginas web
interactivas">JavaScript</a> libre que se centra en la interacción
entre Html y JavaScript (incluyendo <a href="#"
class="tooltip" title="Document Object Model">DOM</a> y <a
href="#" class="tooltip" title="Asynchronous JavaScript and
XML">AJAX </a>).</p>
<p><img border="0" src="jquery-logo.gif" alt=""></p>
<p>Fuente: <a href="#" class="tooltip" title="Enciclopedia
universal disponible en la Web">Wikipedia</a></p>
</body>
</html>
```

Introducción

Los eventos son los desencadenantes de la interactividad que aporta JavaScript. Se pueden asociar a las acciones del usuario, a las funciones y a los métodos. Con la librería jQuery, el principio sigue siendo el mismo, pero algunos administradores de eventos se han adaptado y se han introducido nuevos métodos.

Los administradores de eventos

La librería de jQuery pone a disposición administradores de eventos bastante similares a los del JavaScript tradicional. De esta manera, a `onmouseover` de JavaScript le corresponde `mouseover` en jQuery. El prefijo `"on"` simplemente ha desaparecido.

1. Al hacer clic con el ratón

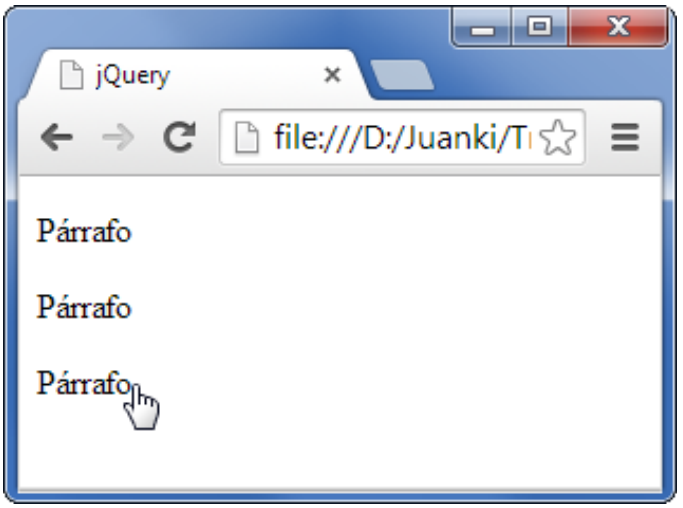
click()

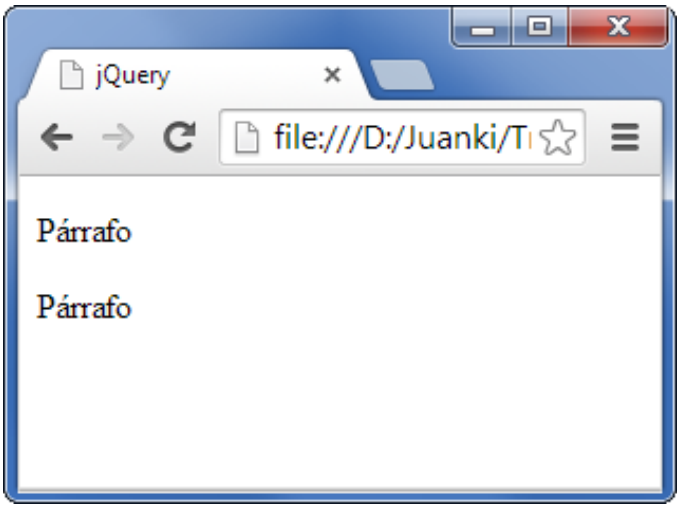
Asocia una función al evento clic de los elementos de la selección.

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

Ejemplo

Al hacer clic en un párrafo, éste desaparece.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
```

```
$("p").click(function () {
$(this).slideUp();
});
});
</script>
<style>
p { cursor: pointer;}
</style>
</head>
<body>
<p>Párrafo</p>
<p>Párrafo</p>
<p>Párrafo</p>
</body>
</html>
```

Veamos el script jQuery:

```
$(document).ready(function(){
$("p").click(function () {
```

Al hacer clic en los párrafos, se ejecuta la siguiente función.

```
$(this).slideUp();
```

Desaparece el elemento pulsado (`this`), deslizándose hacia arriba.

```
});
});
```

Fin del script.

2. Al hacer doble clic

dblclick()

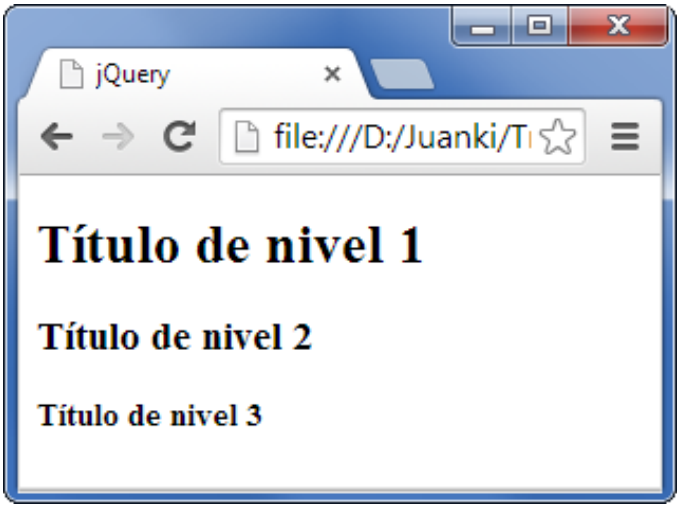
Asocia una función al evento doble clic de los elementos de la selección.

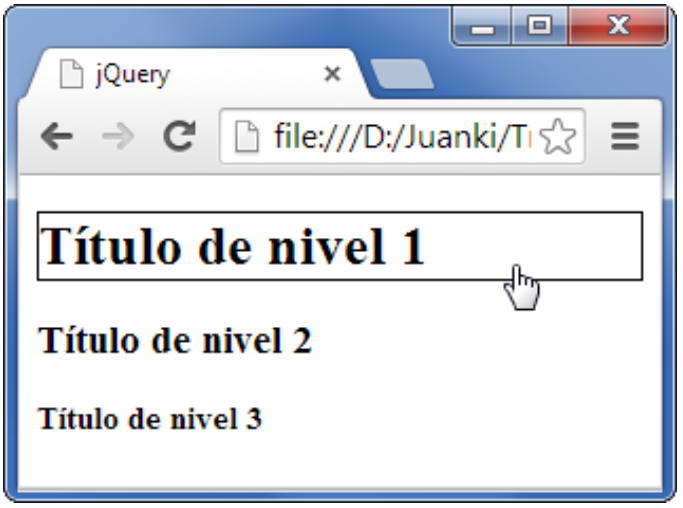
```
$("div").dblclick();
```

La Web es el reino del clic único; el doble clic está poco extendido.

Ejemplo

Al hacer doble clic en un título, éste aparece rodeado por un borde.






```
doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("h1, h2, h3").dblclick(function () {
$(this).toggleClass('borde');
});
});
</script>
<style>
body { font: arial, sans-serif;
       font-size: 0.8em; }
.borde { border: 1px solid black;}
h1, h2, h3 { cursor: pointer;}
</style>
</head>
<body>
<h1>Título de nivel 1</h1>
<h2>Título de nivel 2</h2>
<h3>Título de nivel 3</h3>
</body>
</html>
```

Echemos un vistazo al script:

```
$(document).ready(function(){
$("h1, h2").dblclick(function () {
```

Al hacer doble clic en un título de nivel 1 y 2.

```
$(this).toggleClass('borde');
```

Se activa o se desactiva (`toggleClass()`) la clase `borde`.

```
});
});
```

Fin del script.

3. El foco

focus()

Asocia una función al evento focus de los elementos especificados.

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

La versión jQuery 1.4 ha añadido los eventos `focusin()` y `focusout()`.

focusin()

Asocia una función cuando un elemento, o cualquiera de sus elementos hijo, obtiene el foco.

```
$("p").focusin(function() {
$(this).find("span").css('display','inline').fadeOut(1000);
});
```

focusout()

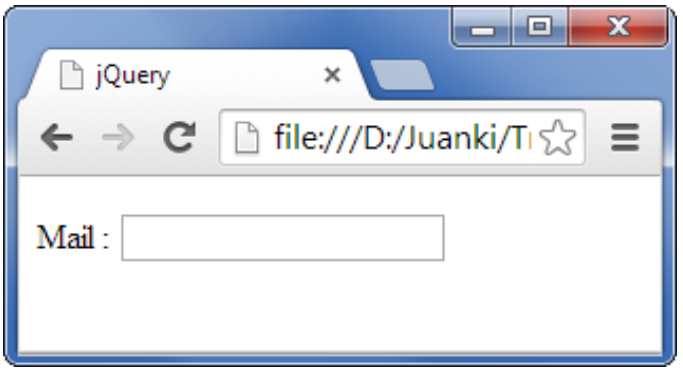
Asocia una función cuando un elemento, o cualquiera de sus elementos hijo, pierde el foco.

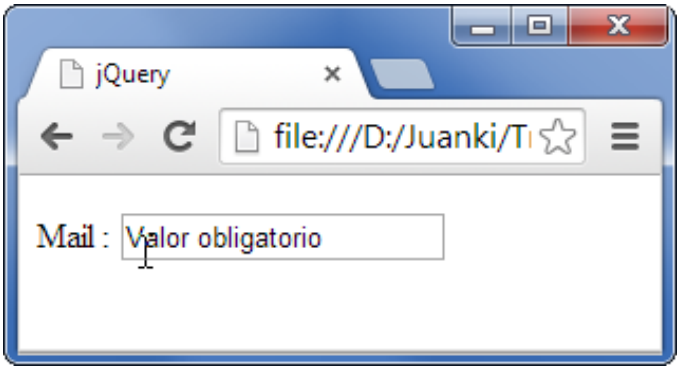
Seguramente algunos lectores se preguntarán la diferencia entre los eventos `focus` y `focusin` o`blur` (vea la siguiente sección) y `focusout`.

Los eventos `focusin` y `focusout` propagan el evento a los elementos hijo, mientras que `focus` y`blur` no conocen esta propagación de eventos. Para más detalles sobre la propagación de eventos, consulte la sección Entrada y salida del cursor de este capítulo.

Ejemplo:

Al obtener el foco de una línea de texto, añadimos Valor obligatorio.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
```

```
$(document).ready(function(){
$("#mail").focus(function () {
$(this).val("Valor obligatorio");
});
});
</script>
</head>
<body>
<p>Mail: <input id="mail" type="text" value=""></p>
</body>
</html>
```

Algunas explicaciones respecto al script:

```
$(document).ready(function(){
$("#mail").focus(function () {
```

Al obtener el foco la línea de texto `mail`.

```
$(this).val("Valor obligatorio");
```

La añadimos el valor (`val("Valor obligatorio")`).

```
});
});
```

Fin del script.

4. Perder el foco

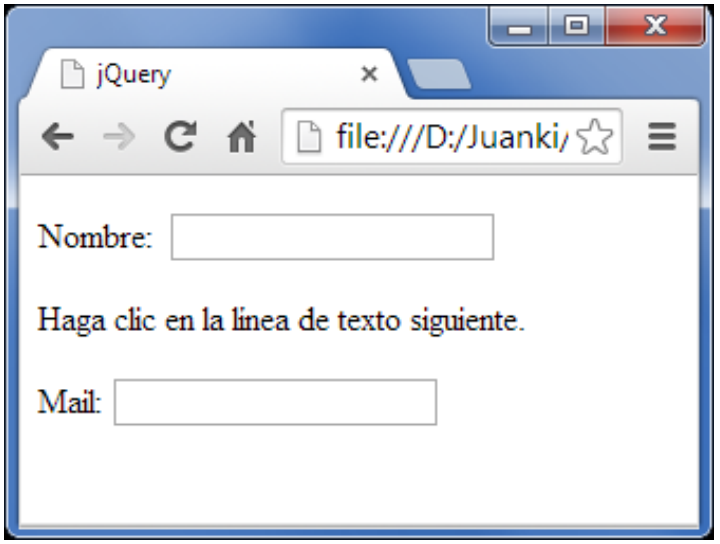
blur()

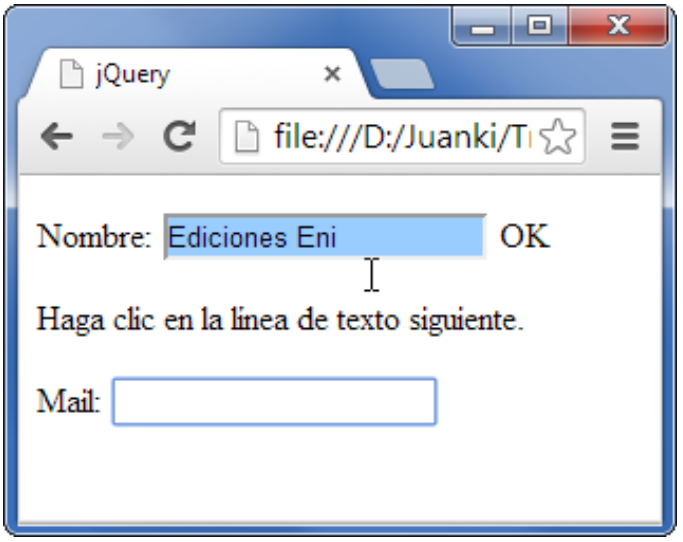
Desencadena el evento que se produce cuando el elemento pierde el foco. El efecto es desencadenar todas las funciones asociadas a este evento para los elementos seleccionados.

```
$("input").blur();
```

Ejemplo

Después de rellenar el campo relativo al nombre y salir de él, el fondo se colorea y se añade "OK".






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#nombre").focus();
$("#nombre").blur(function () {
$(this).css({"background-color":"#9cf"});
$("p:first").append(" OK");
});
});
</script>
</head>
<body>
<p>Nombre: <input id="nombre" type="text" value=""></p>
<p>Haga clic en la línea de texto siguiente.</p>
<p>Mail: <input id="mail" type="text" value=""></p>
</body>
</html>
```

Algunos detalles relativos al script:

```
$(document).ready(function(){
$("#nombre").focus();
```

Inicialmente, el script da el foco a la línea de texto relativa al nombre.

```
$("#nombre").blur(function () {
$(this).css({"background-color":"#9cf"});
$("p:first").append(" OK");
```

Al salir del campo, el fondo de la línea de texto se colorea y se añade "OK".

```
});
});
```

Fin del script.

5. La barra de desplazamiento

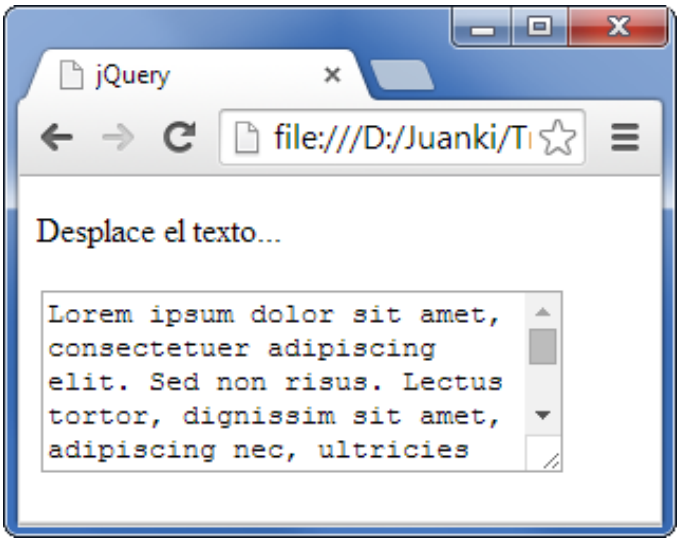
scroll()

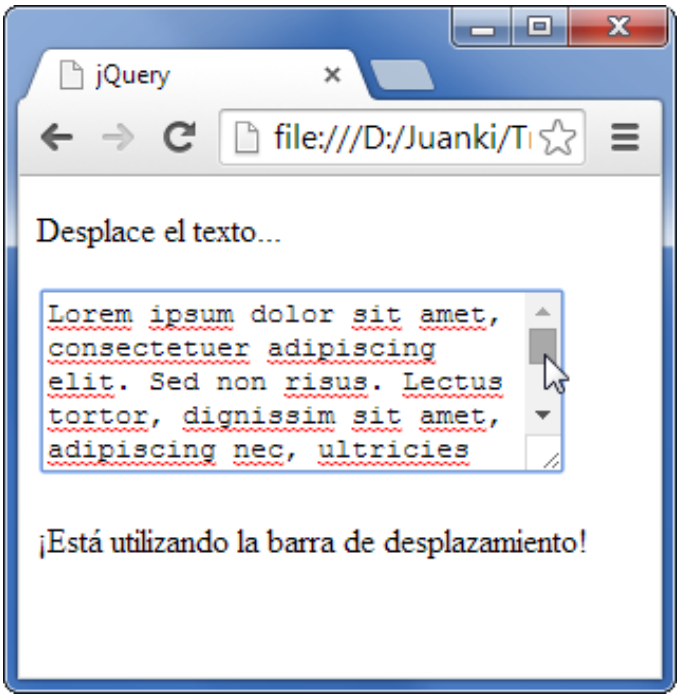
Asocia una función al uso de la barra de desplazamiento de un elemento.

```
$(window).scroll();
```

Ejemplo

Aparece un texto cuando el usuario usa la barra de desplazamiento.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("textarea").scroll(function () {
$("span").css({"display": "inline"}).fadeOut("slow");
});
});
```

```
</script>
<style>
span { display:none;}
</style>
</head>
<body>
<p>Desplace el texto...</p>
<textarea cols="28" rows="5">Lorem ipsum dolor sit amet,
consectetuer adipiscing elit. Sed non risus. Lectus tortor,
dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras
elementum ultrices diam. Maecenas ligula massa, varius a, semper
congue, euismod non, mi. Proin porttitor, orci nec nonummy
molestie, enim es eleifend mi, non fermentum diam nisl sit amet
erat. Duis semper.</textarea>
<p><span>¡Está utilizando la barra de desplazamiento!</span></p>
</body>
</html>
```

Explicaciones:

```
$(document).ready(function(){
$("textarea").scroll(function () {
```

Cuando se usa la barra de desplazamiento de la zona de texto (`<textarea>`).

```
$("span").css({"display": "inline"}).fadeOut("slow");
```

El texto de la etiqueta `<span>` se muestra con un efecto.

```
});
});
```

Fin del script.

6. El botón del ratón

mousedown()

Asocia una función a los elementos seleccionados cuando el usuario pulsa un botón del ratón.

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

mouseup()

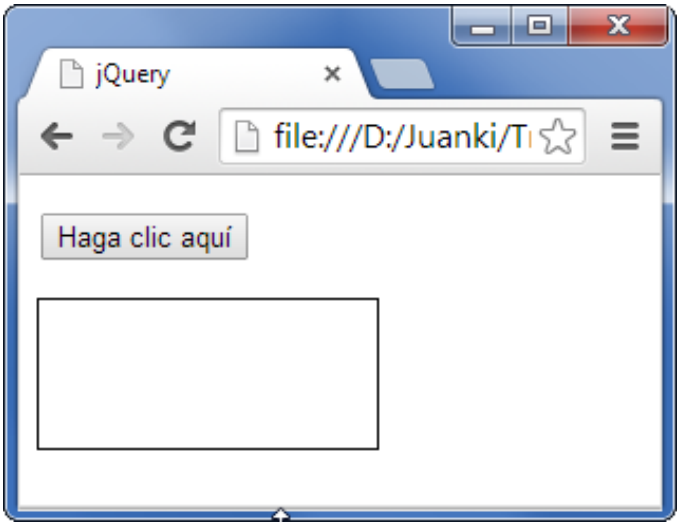
Asocia una función a los elementos de la selección cuando el usuario suelta el botón del ratón.

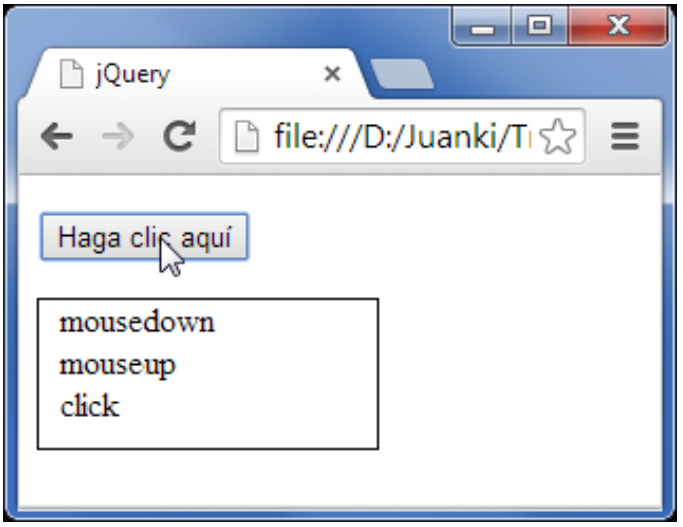
```
$("p").mouseup();
```

El evento clic se activa cuando se detecta un `mousedown` y un `mouseup`.

Ejemplo

Al hacer clic en una capa, detallamos los eventos `mousedown`, `mouseup` y `click`.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>

$(document).ready(function() {
$("input").mousedown(function(){
$("#salida").append("mousedown<br>");
});
$("input").mouseup(function(){
$("#salida").append("mouseup<br>");
});
$("input").click(function(){
$("#salida").append("click<br>");
});
});
//]]>
</script>
<style type="text/css">
div { width: 150px;
      height: 70px;
      overflow: auto;
      border: solid 1px black;
      padding-left: 10px;}
</style>
```

```
</head>
<body>
<p><input type="button" value="Haga clic aquí" /></p>
<div id="salida"></div>
</body>
</html>
```

Exploremos el script jQuery:

```
$(document).ready(function() {
$("input").mousedown(function(){
$("#salida").append("mousedown<br>");
});
```

Al presionar el botón del ratón (`mousedown()`), la palabra "mousedown" se inserta en la capa identificada por `salida`.

```
$("input").mouseup(function(){
$("#salida").append("mouseup<br>");
});
```

Al soltar el botón del ratón (`mouseup()`), la palabra "mouseup" se inserta en la capa.

```
$("input").click(function(){
$("#salida").append("click<br>");
});
```

Cuando se detecta el evento `click`, se añade la palabra "clic".

```
});
```

Fin del script.

7. El desplazamiento del cursor

Detectar el menor movimiento del cursor es un evento interesante, pero es necesario ser consciente del elevado consumo de recursos de sistema que implica. De hecho, las acciones asociadas al movimiento del ratón se desencadenan cuando el cursor se mueve un píxel. Este proceso puede disminuir la usabilidad de un sitio Web, en particular para ordenadores antiguos.

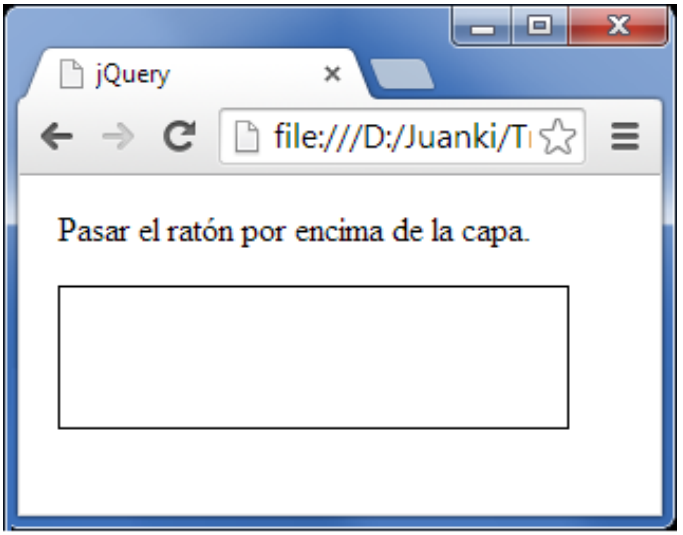
mousemove()

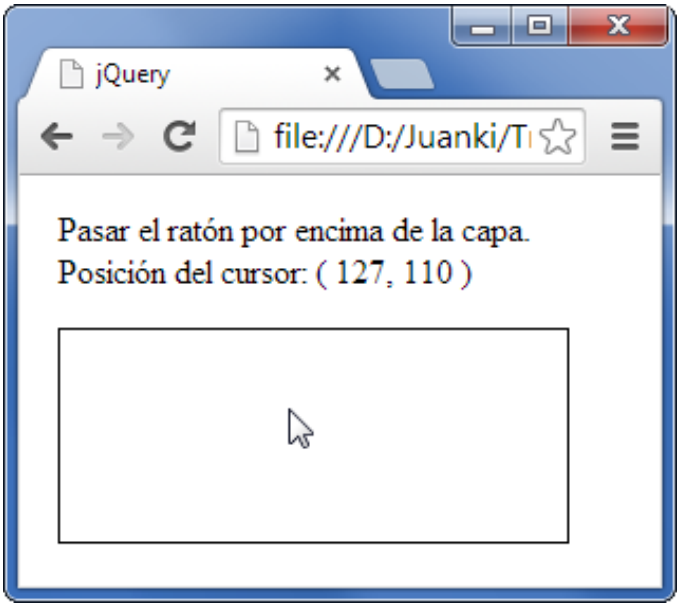
Asocia una función cuando el usuario mueve el cursor del ratón.

```
$("div").mousemove();
```

Ejemplo

Buscamos las coordenadas del cursor del ratón en cada uno de sus movimientos.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("div").mousemove(function(e){
var paginaCoords = "( " + e.pageX + ", " + e.pageY + " )";
$("span:first").text("Posición del cursor:  " + paginaCoords);
});
$("div").mouseout(function(e){
$("span:first").text("");
});
});
</script>
<style>
div { width:240px;
      height:100px;
      margin-left:10px;
      border:1px solid black; }
p { margin-left:10px; }
span { display:block; }
</style>
```

```
</head>
<body>
<p>Pasar el ratón por encima de la capa.
<span> </span>
</p>
<div></div>
</body>
</html>
```

Detallemos el script:

```
$(document).ready(function(){
$("div").mousemove(function(e){
```

Al mover el cursor en la capa.

```
var paginaCoords = "( " + e.pageX + ", " + e.pageY + " )";
```

La posición horizontal y vertical del cursor se almacena en la variable `paginaCoords`.

```
$("span:first").text("Posición del cursor: " + paginaCoords);
});
```

La posición se mostrará como texto en la primera etiqueta `<span>`.

```
$("div").mouseout(function(e){
$("span:first").text("");
});
```

Cuando el cursor sale de la capa (`mouseout()`), no se muestra nada (`text("")`).

```
});
```

Fin del script.

8. Entrada y salida del cursor

Los habituados a JavaScript, se reencuentran con `onmouseover` y `onmouseout` que con jQuery se convierten en `mouseover` y `mouseout`.

mouseover()

Asocia una función cuando el usuario coloca el cursor del ratón encima de un elemento.

```
$("div").mouseover();
```

mouseout()

Asocia una acción al evento cuando el cursor del ratón sale de un elemento.

```
$("div").mouseout();
```

De forma (a priori) muy similar, jQuery añade los eventos `mouseenter` y `mouseleave`.

mouseenter()

Ejecuta una función cuando el cursor entra en un elemento.

```
$("div").mouseenter();
```

mouseleave()

Desencadena una función cuando el cursor sale de un elemento.

```
$("div").mouseleave();
```

Sin embargo, el funcionamiento del administrador de eventos es diferente. De hecho, `mouseover`, heredada de JavaScript `onmouseover`, propaga el evento a la jerarquía de objetos de la página.

Este fenómeno se conoce como propagación de eventos o desbordamiento de eventos. Por el contrario, `mouseenter` evita esta propagación del evento.

Sucede lo mismo para `mouseout`, que se desencadena cada vez que el cursor se mueve hacia un elemento hijo o desde él. A la inversa, `mouseleave` solo se desencadena una vez: cuando el cursor sale del elemento actual.

En algunas situaciones de programación, esta propagación puede ser molesta y jQuery aporta una solución.

Vamos a ver esta diferencia, más bien tenue, entre `mouseover` y `mouseenter` usando los siguientes ejemplos.

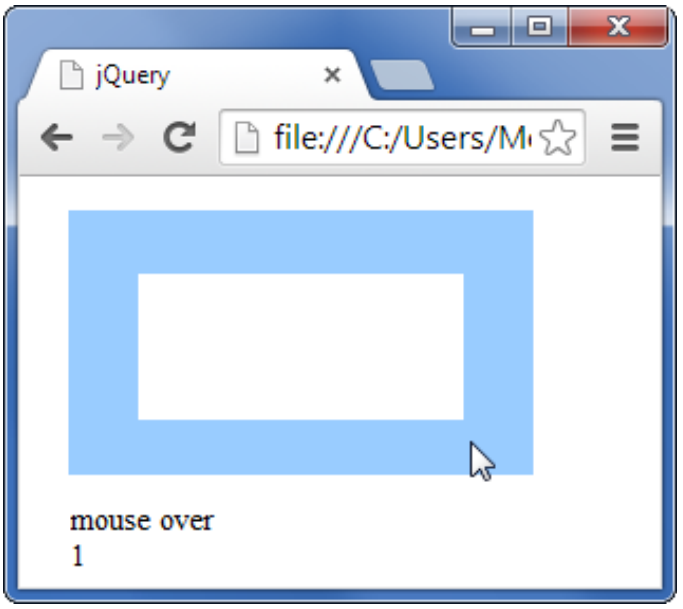
Ejemplo

Inicialmente, nos centramos en `mouseover`.

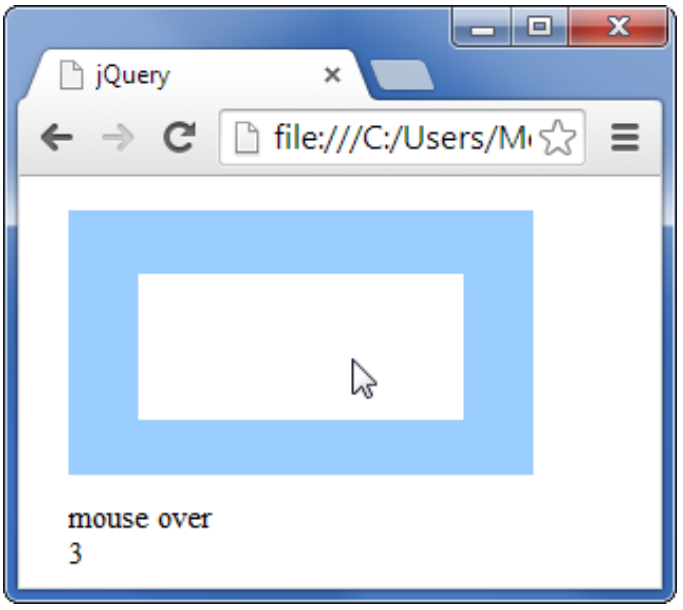
```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
var i = 0;
$("div.overout").mouseover(function(){
$("#mostrar").text("mouse over");
$("#contador").text(++i);
})
.mouseout(function(){
$("#mostrar").text("mouse out");
});
});
</script>
<style>
div.out { width: 220px;
          height: 125px;
          margin-left: 15px;
          background-color: #9cf;}
div.in { width: 70%;
         height: 55%;
         background-color: white;
         margin: 15px auto;}
p { line-height:0.9em;
    padding:0;}
#mostrar { margin-top: 10px;
           margin-left: 15px;
           font-size: 1.2em;}
#contador { margin-left: 15px;
            font-size: 1.2em;}
</style>
</head>
<body>
<div class="out overout"><p> </p>
<div class="in overout"></div>
</div>
```

```
<div id="mostrar"></div>
<div id="contador"></div>
</body>
</html>
```

Cuando el cursor del ratón pasa por encima (`mouseover`) de la capa coloreada, el contador implementado en el script indica la cifra 1.



Por el contrario, cuando el cursor del ratón pasa por encima de la capa blanca (elemento hijo de la capa coloreada), el contador marca, no 2, sino la cifra 3. El evento de pasar encima del elemento hijo se propaga al evento de pasar sobre el elemento padre. Por tanto, tenemos 3 eventos.



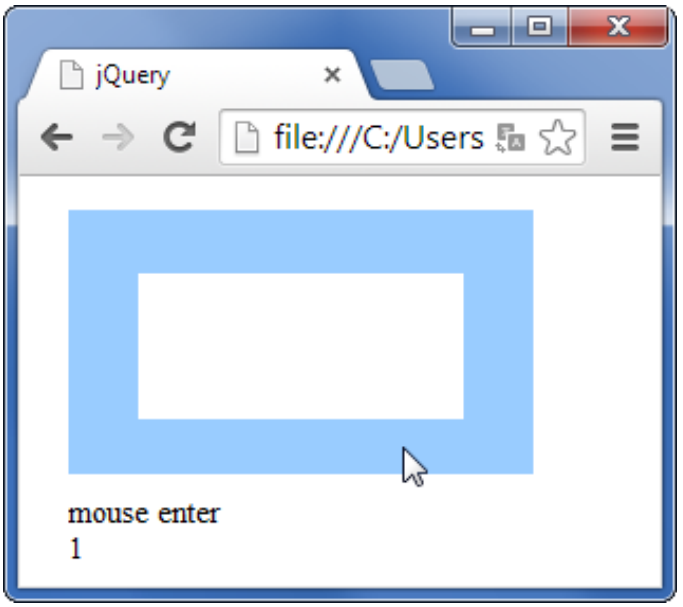
Ejemplo

Volvemos al mismo código, pero esta vez con `mouseenter` y `mouseleave`.

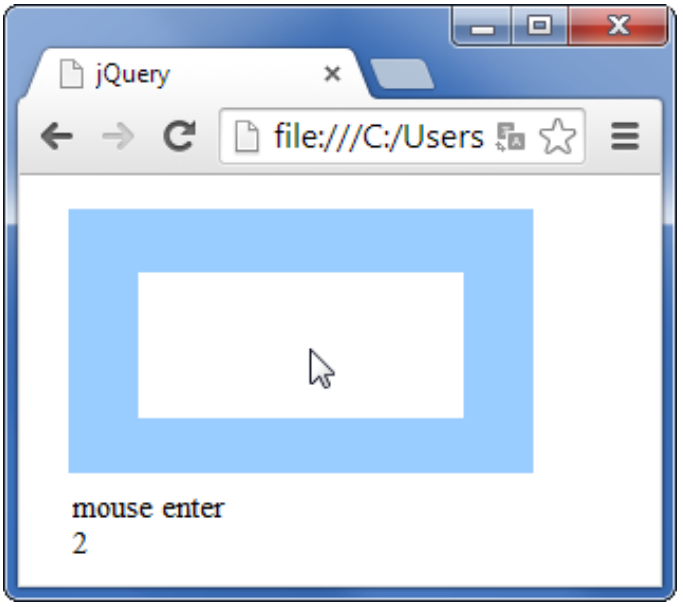
```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
```

```
$(document).ready(function(){
var i = 0;
$("div.overout").mouseenter(function(){
$("#mostrar").text("mouse enter");
$("#contador").text(++i);
})
.mouseleave(function(){
$("#mostrar").text("mouse leave");
});
});
</script>
<style>
div.out { width: 220px;
          height: 125px;
          margin-left: 15px;
          background-color: #9cf;}
div.in { width: 70%;
         height: 55%;
         background-color: white;
         margin: 15px auto;}
p { line-height:0.9em;
    padding:0;}
#mostrar { margin-top: 10px;
           margin-left: 15px;
           font-size: 1.2em;}
#contador { margin-left: 15px;
            font-size: 1.2em;}
</style>
</head>
<body>
<div class="out overout"><p> </p>
<div class="in overout"></div>
</div>
<div id="mostrar"></div>
<div id="contador"></div>
</body>
</html>
```

Cuando el cursor del ratón entra (`mouseenter`) en la capa coloreada, el contador que se implementa en el script indica la cifra 1.



Y cuando el cursor del ratón entra en la capa blanca (elemento hijo de la capa coloreada), el contador marca la cifra 2. El evento `mouseenter` no lo propaga.



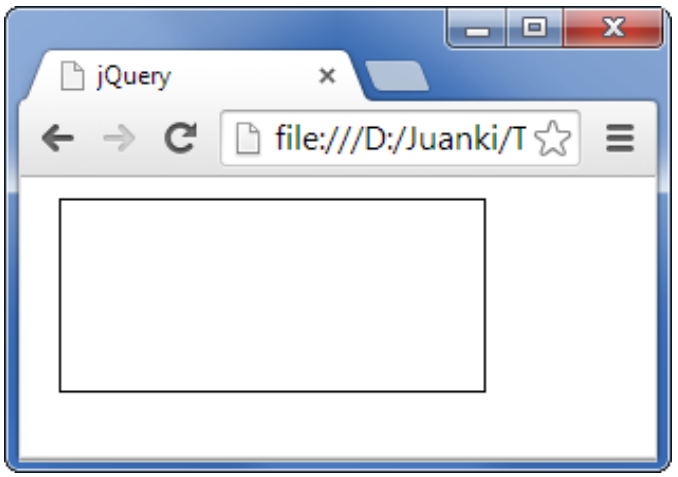
Este desbordamiento del evento incluye efectos indeseables cuando hay elementos padres con elementos hijos y se aplica un `mouseover` o `mouseout` al elemento padre. El movimiento del ratón en los elementos hijo puede desencadenar un evento `mouseout` en sus elementos padres. El evento empieza por el elemento hijo y pasa de padre a padre. La propagación es ascendente.

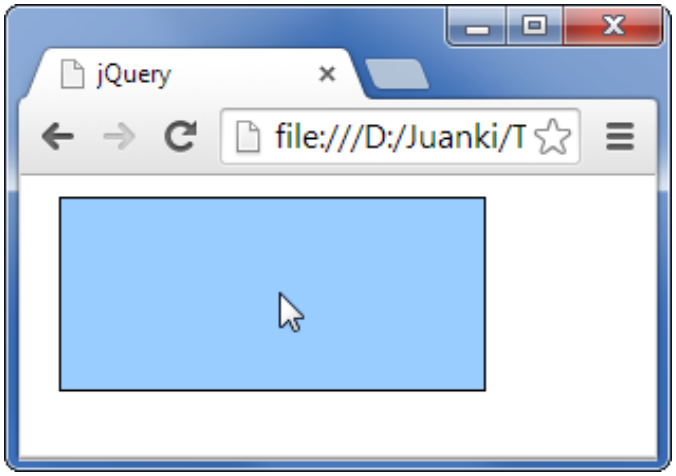
Para saber más, haga una búsqueda en Google con las palabras clave "propagación de evento", "desbordamiento de evento" o "event bubling".

Volvemos a nuestros eventos jQuery, por ejemplo `mouseenter` y `mouseleave`.

Al pasar el ratón por encima de la división, se aplica un color de fondo. Cuando el cursor sale, se restaura la situación inicial.

Ejemplo






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("div").mouseenter(function(){
$(this).css({"background-color":"#9cf"});
})
.mouseleave(function(){
$(this).css({"background-color":"white"});
});
});
</script>
<style type="text/css">
div { width: 200px;
      height: 90px;
      margin: 10px;
      border: 1px solid black;}
</style>
</head>
<body>
<div></div>
</body>
</html>
```

Explicaciones del script.

```
$(document).ready(function(){
$("div").mouseenter(function(){
$(this).css({"background-color":"#9cf"});
})
```

Al pasar el ratón (`mouseenter()`) por encima de la capa `<div>`, se modifica su propiedad CSS de color de fondo (`css({"background-color":"#9cf"})`).

```
.mouseleave(function(){
$(this).css({"background-color":"white"});
});
```

Cuando el cursor sale de la capa (`mouseleave()`), el color de fondo se cambia a blanco (`css({"background-color":"white"})`).

```
});
```

Fin del script.

Comentario

Como en este ejemplo la capa no tiene elemento hijo, el uso de `mouseover` y de `mouseout` hubiera tenido el mismo resultado.

9. Enviar una consulta

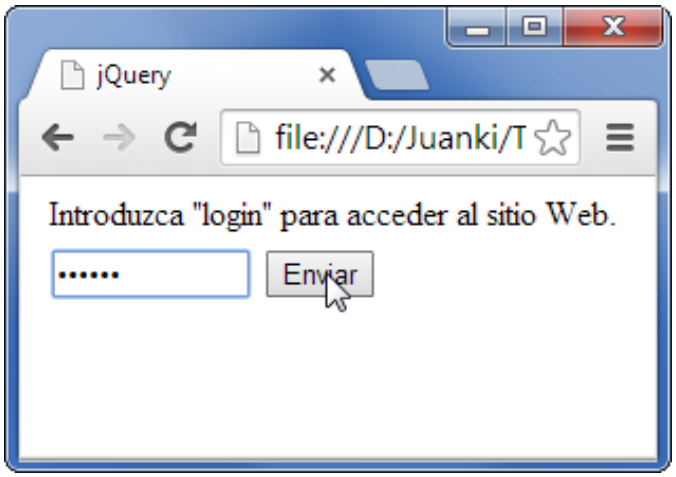
submit()

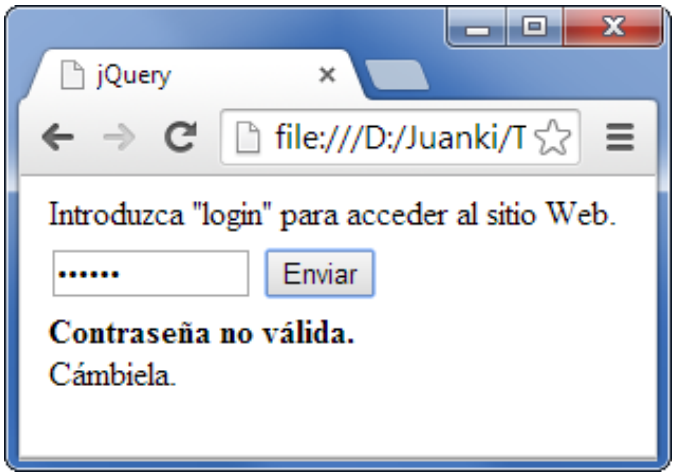
Desencadena el evento de envío del formulario.

```
$("form").submit();
```

Ejemplo:

Un formulario solicita una contraseña (login en nuestro ejemplo) para acceder al sitio Web. Cuando se envía, si la contraseña es correcta, se redirige al usuario a la dirección deseada. En caso contrario, se muestra un mensaje de error.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
```

```
$(document).ready(function(){
$("form").submit(function() {
if ($("input:first").val() == "login") {
return true;
}
$("span").html("<b>Contraseña no válida.</b><br>Cámbiela.
").show();
return false;
});
});

</script>
<style>
div,p,span { margin: 5px 0 0 5px;}
span { display: block;}
</style>
</head>
<body>
<p>Introduzca "login" para acceder al sitio Web.</p>
<form action="javascript:window.location.href='http://www.google.es';">
<div>
<input type="password" size="10">
<input type="submit">
</div>
</form>
<span></span>
</body>
</html>
```

Explicaciones.

```
$(document).ready(function(){
$("form").submit(function() {
```

Al enviar el formulario.

```
if ($("input:first").val() == "login") {
return true;
}
```

Si la contraseña introducida es correcta, se ejecuta la acción que se define en el código.

```
$("span").html("<b>Contraseña no válida.</b><br>Cámbiela.").show();
return false;
```

Si la contraseña no es correcta, se inserta un mensaje Html de error en la etiqueta `<span>` y se muestra.

```
});
});
```

Fin del script.

10. Otros eventos

Existen otros eventos, pero se usan menos. Vamos a repasarlos rápidamente.

- **change()**: desencadena un evento cuando se modifica un control de formulario, por ejemplo cuando se activa una casilla de selección de formulario.
- **keydown()**, **keyup()** y **keypress()**: desencadenanun evento cuando se pulsa una tecla del teclado (hacia abajo), cuando se suelta una tecla del teclado (hacia arriba) y cuando se

escribe un carácter.

- **resize()**: asocia un evento cuando se modifica el tamaño de un elemento, normalmente la ventana del navegador.
- **select()**: se produce cuando el usuario selecciona un texto (o una parte de él). Algunas veces se aplica a los campos de formulario de tipo línea de texto o zona de texto (textarea).

No olvidemos el evento básico de todo script jQuery, es decir, **ready()**, que asocia una función cuando el DOM está preparado para ser manejado.

Métodos o administradores de eventos avanzados

1. Unir un evento a un objeto (on)

on(evento, [selector], [datos], función)

Asigna el evento a un elemento determinado:

- evento (cadena de caracteres): designa el evento asociado. Si se especifican varios eventos, se deben separar simplemente por un espacio.
- selector (opcional): un filtro para seleccionar los hijos del elemento en el que se aplica el evento.
- datos (opcional): datos que se le pueden proporcionar a la función. Su uso es poco frecuente.
- función: el código que se ejecuta al desencadenarse el evento.

```
$("button").on("click", function() {
alert($(this).text());
});
```

o

```
function saludo(event) {
alert("Buenos días " + event.data.nombre);
}
$("boton").on("click", {nombre: "Carlos"}, saludo);
```

Este método devuelve un objeto jQuery.

El método `off()` elimina las acciones asociadas a un evento mediante el método `on()`.

Los métodos `on()` y `bind()` (ver Unir un evento a un objeto de este capítulo) son más potentes que los eventos específicos, como `click()` o `mouseover()`, que hemos visto anteriormente.

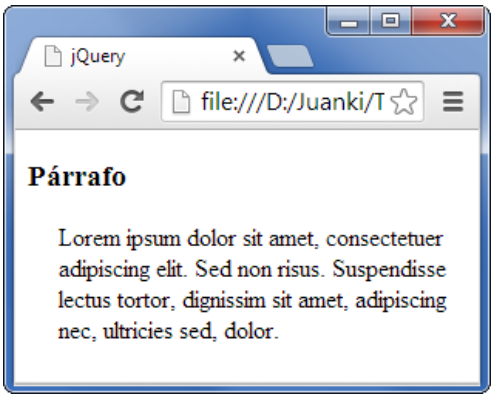
El método permite no solamente asignar uno o varios eventos a un objeto jQuery, en el que se ejecutará la función que se pasa como argumento, sino también transmitir datos a esta función. De esta manera, un clic en un enlace o pasar el ratón por encima de una imagen puede asignar información diferente al administrador de eventos. Por tanto, la función relacionada con el evento se podrá ejecutar de manera diferente según el contexto que proporcionan los datos.

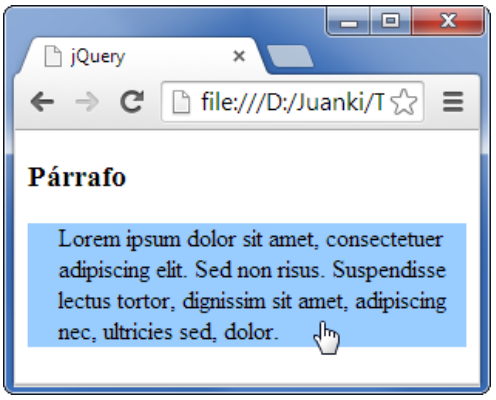
El método `on()` se introdujo en la versión 1.7 de jQuery después de una reescritura completa de la API que implementa la gestión de eventos para asegurar una mayor coherencia y un mejor rendimiento. El empleo de este método `on()` está muy recomendado en la documentación de jQuery ya que está destinado a reemplazar los métodos más antiguos de gestión de eventos como`bind()`, `delegate()` y `live()`. Este último ya quedó obsoleto a partir de la versión 1.7 y se eliminó en las siguientes versiones.

La gestión de los eventos por el método `on` puede utilizarse no solo sobre los elementos existentes sino también sobre los elementos futuros que se pueden crear desde un script. Esta última característica está vinculada al filtro selector que permite ampliar el administrador de eventos a elementos hijos todavía no creados.

Ejemplo

Vinculemos los eventos `mouseover` y `mouseout` a una etiqueta de párrafo. Esta nueva asociación tiene como acción poner o no un fondo de color.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("p").on("mouseover mouseout", function(e){
$(this).toggleClass("over");
});
});
</script>
<style>
p { cursor: pointer;
    padding-left: 20px;}
p.over { background: #9cf;}
</style>
</head>
<body>
<h3>Párrafo</h3>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed
non risus. Suspendisse lectus tortor, dignissim sit amet,
adipiscing nec, ultricies sed, dolor.</p>
</body>
</html>
```

Al cargar el DOM y en la inicialización de jQuery.

```
$("p").on("mouseover mouseout", func
```

Los eventos `mouseover` y `mouseout` se vinculan (`on()`) al párrafo `<p>`.

```
$(this).toggleClass("over");
});
```

La función asociada es alternar (`toggleClass()`) a la clase CSS over.

```
});
```

Fin de script.

2. Unir un evento a un objeto (bind)

bind(evento, [datos], función)

Asigna el evento a un elemento dado.

- "evento" (cadena de caracteres): designa el evento asociado. Si se especifican varios eventos, simplemente se tienen que separar por un espacio.
- "datos" (opcional): datos que normalmente se proporcionan a la función.
- "función": el código que se tiene que ejecutar cuando se desencadene el evento.

```
$("button").bind("click", function() {
alert($(this).text() );
});
```

o

```
var mensaje = "Bienvenido";
$("#ejemplo").bind( "click", {msg: mensaje}, function(event) {
alert( event.data.msg );
});
```

Este método devuelve un objeto jQuery.

El método `unbind()` elimina las acciones que el método `bind()` ha asociado a un evento.

Este método `bind()` proviene del origen de jQuery (versión 1.0). No obstante, presenta ciertos límites:

- No puede agregar un administrador de eventos más que para elementos existentes.
- Puede consumir muchos recursos cuando se aplica a un selector que pueda señalar a numerosas ocurrencias. De hecho, este método `bind()` une el administrador de eventos a todos los elementos identificados por la selección. Si un evento `click` está asociado a una etiqueta de vínculo y su documento tiene 100 etiquetas `<a>`, se vincularán 100 administradores de eventos a su fichero. Esto disminuirá la velocidad de ejecución del script.

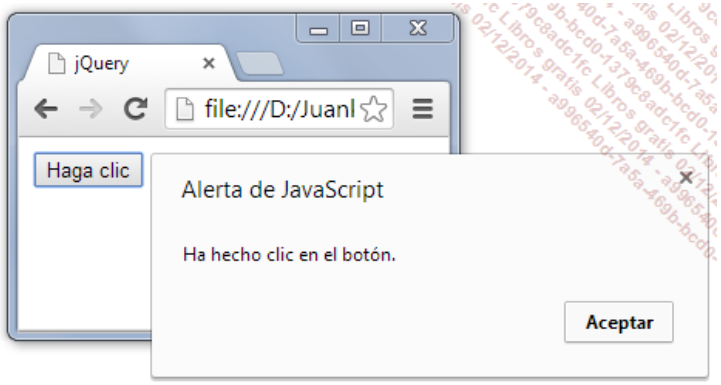
No es de extrañar que jQuery aconseje no utilizar este método `bind()` y dar preferencia al método `on()` más completo y más competitivo.

El método `live()` que igualmente vinculaba un evento a un elemento se ha eliminado. Para los que empiezan con jQuery no será un problema ya que este método ya no existe. Por el contrario, para los desarrolladores que quieran actualizar una aplicación jQuery más antigua

utilizando este método `live()`, esta eliminación puede entrañar sorpresas desagradables. Lo mismo ocurre con el método `die()` que eliminaba las acciones asociadas a un evento mediante el método `live()`.

Ejemplo

Mostremos un mensaje de aviso al hacer clic en un botón.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("button").bind("click", function(){
alert(”Ha hecho clic en el botón.”);
});
});
</script>
</head>
<body>
<button>Haga clic</button>
</body>
</html>
```

```
$("button").bind("click",function(){
```

El evento `click` está vinculado (`bind()`) al botón (`button`).

```
alert("Ha hecho clic en el botón.");
});
```

La función asociada consiste en mostrar un cuadro de aviso (`alert`) en JavaScript clásico.

```
});
```

Fin del ready.

3. Delegar un evento

delegate(selector, evento, función)

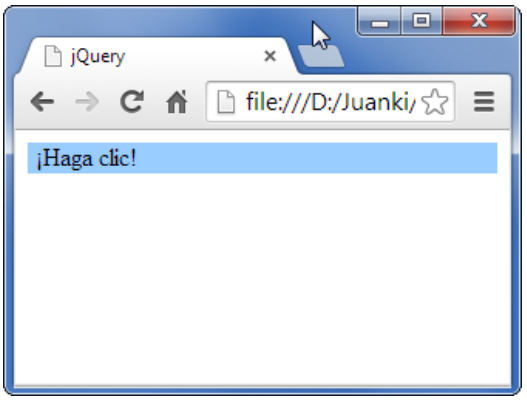
Vincula una función a un evento para todos los elementos que correspondan al selector ahora y en el futuro.

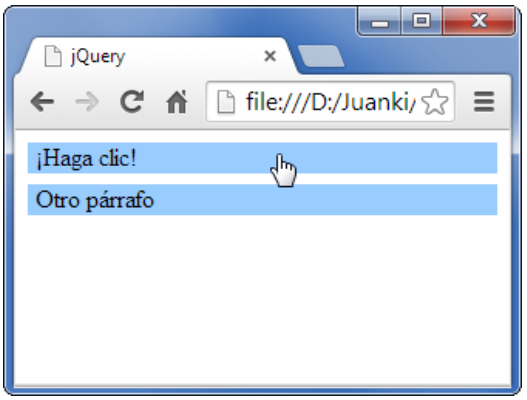
Observe la introducción del selector al que lleva el evento directamente en la sintaxis del método.

Introducido en la versión 1.4 de jQuery, el método `delegate()` era el más utilizado para vincular un evento a un elemento. Sin embargo, la documentación ya no aconseja su utilización y recomienda el método `on()`.

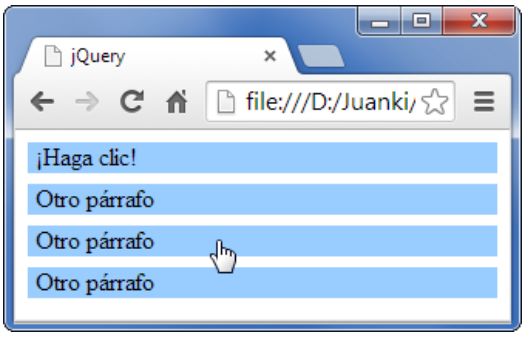
El método `undelegate()` elimina la asociación creada por `delegate()`.

Ejemplo





Haciendo clic en el párrafo inicial o en el nuevo, la función añade un nuevo párrafo (que no existía en el fichero inicial). Y así sucesivamente…




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("body").delegate("p", "click", function(){
$(this).after("<p>Otro párrafo</p>");
});
});
</script>
<style>
p { background: #9cf;
    cursor: pointer;
    padding-left: 5px;
    margin : 7px 0 7px 0;}
</style>
</head>
<body>
<p>¡Haga clic!</p>
</body>
</html>
```

4. Ejecutar una función solo una vez

one(evento,[datos],función)

Asocia una función a un evento dado. La diferencia con la función `bind()` es que la función asociada al evento solo se ejecutará, como máximo, una vez para cada elemento de la selección.

- "evento" (cadena de caracteres): tipo de evento implicado.
- "datos" [opcional]: datos adicionales que se pasan al administrador de eventos. Su uso es poco frecuente.
- "función": función que se asocia al evento.

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

o

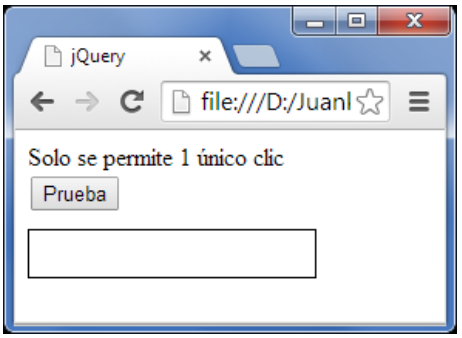
```
$("body").one("click", "#ejemplo", function() {
alert( "Me muestro si #ejemplo es el primer elemento
en el que se hace clic en el body.");
});
```

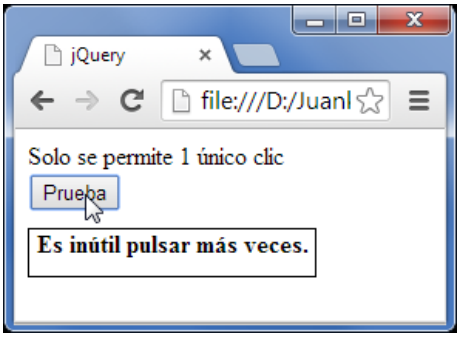
Este método devuelve un objeto jQuery.

Esta función puede ser muy útil en algunos scripts.

Ejemplo

Imaginemos un botón que solo podemos pulsar una vez. Cuando se hace el primer clic, se muestra un mensaje.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("button").one("click", function(){
$("div").html("<b>Es inútil pulsar más veces.</b>");
});
});
</script>
<style>
div { width: 180px;
      height: 30px;
      margin-top:10px;
      padding-left: 5px;
      border: 1px solid black;}
</style>
</head>
<body>
Solo se permite 1 único clic<br>
<button>Prueba</button>
<div></div>
</body>
</html>
```

```
$(document).ready(function(){
$("button").one("click", function(){
```

Al hacer clic en el botón, se ejecuta la siguiente función.

```
$("div").html("<b>Es inútil pulsar m
});
```

La frase "Es inútil pulsar más veces." se mostrará como Html en la capa. Como se ha usado el método `one()`, la frase solo se muestra una vez y cualquier otro clic en el botón no realizará ninguna acción.

```
});
```

Fin del script.

5. Desencadenar un evento particular

trigger(evento)

Desencadena un evento particular para los elementos de la selección. Esto también va a desencadenar la acción por defecto del navegador para este tipo de evento (si existe). Por ejemplo, usar el tipo de evento '`submit`' en la función también va a desencadenar el envío del formulario por parte del navegador. Esta acción por defecto se puede evitar devolviendo "`false`" en una de las funciones asociadas al evento, para uno de los elementos de la selección.

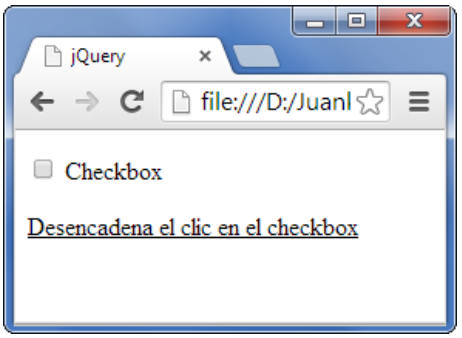
También puede usar la función `bind()`.

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

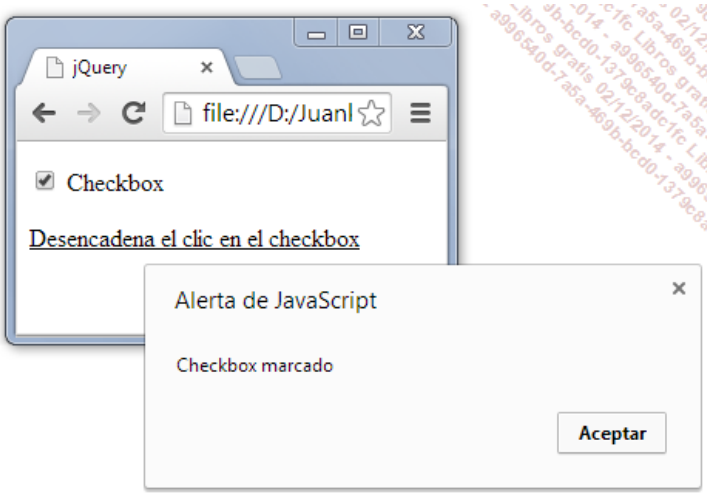
Este método devuelve un objeto jQuery.

Ejemplo

Al hacer clic en el enlace, se seleccionará la casilla de selección ***Checkbox****.*



Al hacer clic en el enlace, el script marca la casilla ***Checkbox****, lo que desencadena un mensaje de alerta.*




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function () {
$('input').on('click', function() {
alert('Checkbox marcado');
});
$('a').on('click', function() {
$('input').trigger('click');
return false;
});
});
</script>
<style>
a { color: black;}
</style>
</head>
<body>
<p><input type="checkbox"> Checkbox</p>
<p><a href="#">Desencadena el clic en el checkbox</a></p>
</body>
</html>
```

6. Al pasar el ratón

Esta función, propia de jQuery, agrupa los eventos `onmouseover` y `onmouseout` de JavaScript o`mouseover` y `mouseout` de jQuery, es decir, cuando el cursor pasa por encima un elemento y sale de él.

hover(función 1, función 2)

El método `hover()` de jQuery une dos eventos que se usan muy frecuentemente en el elemento seleccionado; cuando el cursor pasa por encima y cuando sale de él.

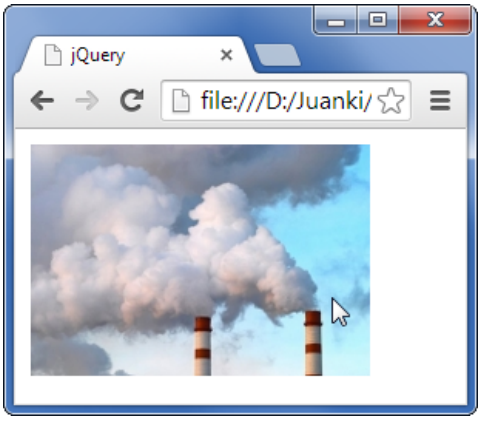
Por tanto, este método se ejecuta en dos partes. Cuando el cursor se sitúa sobre un elemento concreto, se ejecuta la primera función que se ha pasado como parámetro. Cuando el cursor sale del ámbito del elemento, se ejecuta la segunda función.

```
$("p").hover(function(){
$(this).addClass("hover");
},
function(){
$(this).removeClass("hover");
});
```

Al pasar el ratón por encima de los párrafos, se añade la clase `hover`. Cuando el cursor sale de la zona, se elimina la clase. Este método devuelve un objeto jQuery.

Ejemplo

El clásico, una imagen que cambia cuando el ratón pasa por encima de ella.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
.image { position: absolute;
         top: 10px;
         left: 10px;}
</style>
</head>
<body>
<div class="image">
<img src="polucion1.jpg" alt="">
</div>
</body>
</html>
```

```
<script>
$(document).ready(function () {
$('img').hover(function () {
this.src = 'polucion2.jpg';
},
function () {
this.src = 'polucion1.jpg';
});
});
</script>
```

El código final es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script">
$(document).ready(function () {
```

El script jQuery se escribe de la siguiente manera:

Lo detallamos a continuación.

```
$(document).ready(function () {
$('img').hover(function () {
```

Se aplica un método `hover()` a las imágenes.

```
this.src = 'polucion2.jpg';
},
```

En primer lugar, al pasar el cursor por encima, se carga la imagen "polucion2.jpg", modificando el atributo `src` de la etiqueta `<img>`.

```
function () {
this.src = 'polucion1.jpg';
});
```

Después, cuando el cursor sale de la imagen, el script vuelve a la situación inicial.

```
});
```

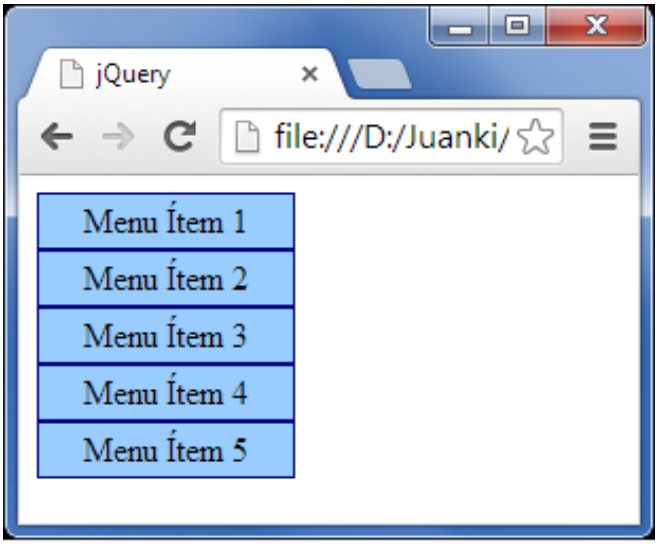
Fin del script.

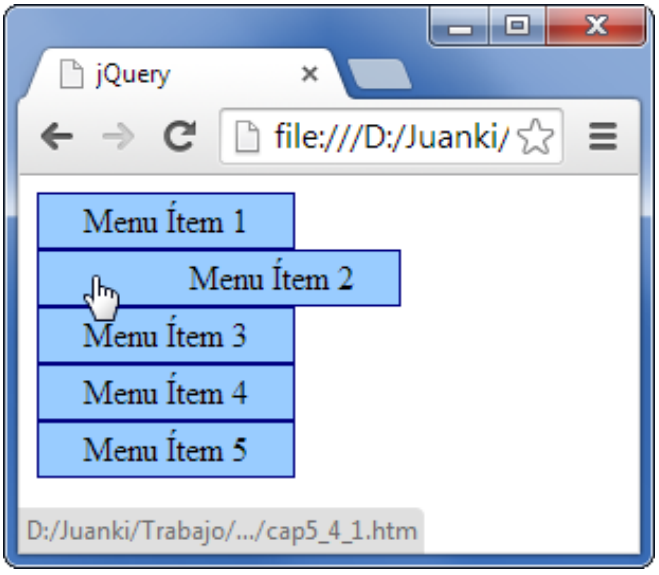
```
$('img').hover(function () {
this.src = 'polucion2.jpg';
},
function () {
this.src = 'polucion1.jpg';
});
});
</script>
<style>
.image { position: absolute;
         top: 10px;
         left: 10px;}
</style>
</head>
<body>
<div class="image">
<img src="polucion1.jpg" alt="">
</div>
</body>
</html>
```

Aplicaciones

1. Un menú desplazado

Al pasar el ratón por encima de un ítem del menú de navegación, éste se va a desplazar a la derecha.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
#nav ul { list-style: none;
          padding: 0px;
          margin: 0px;}
#nav li a { display: block;
            height: 25px;
            line-height: 25px;
            width: 120px;
            background: #99ccff;
            border: 1px solid navy;
            color: black;
            text-decoration: none;
```

```
            text-align: center;}
</style>
</head>
<body>
<div id="nav">
<ul id="menu">
<li><a href="#">Menú Ítem 1</a></li>
<li><a href="#">Menú Ítem 2</a></li>
<li><a href="#">Menú Ítem 3</a></li>
<li><a href="#">Menú Ítem 4</a></li>
<li><a href="#">Menú Ítem 5</a></li>
</ul>
</div>
</body>
</html>
```

El script jQuery se presenta de la siguiente manera:

```
<script>
$(document).ready(function() {
$('ul#menu li a').hover(function() {
$(this).stop().animate( { paddingLeft:"50px" }, 400 );
},
function() {
$(this).stop().animate( { paddingLeft:"0" }, 200 )
})
});
</script>
```

Explicaciones.

```
$(document).ready(function() {
$('ul#menu li a').hover(function() {
```

El método `hover()` se ha asociado a los elementos de la lista que forman el menú de navegación.

```
$(this).stop().animate( { paddingLeft:"50px" }, 400 );
},
```

Al pasar el ratón por encima de cada ítem, el script detiene cualquier animación que se esté desarrollando (`stop()`). Después, aplica una anima-ción (`animate()`), que consiste en aumentar la distancia con respecto al borde izquierdo (`paddingLeft:"50px"`).

```
$(this).stop().animate( { paddingLeft:"0" }, 200 )
})
```

Cuando el cursor sale del ítem, la segunda parte del método `hover()` hace que el ítem regrese a su posición inicial.

```
});
```

Fin del ready y del script.

El documento final es el siguiente:

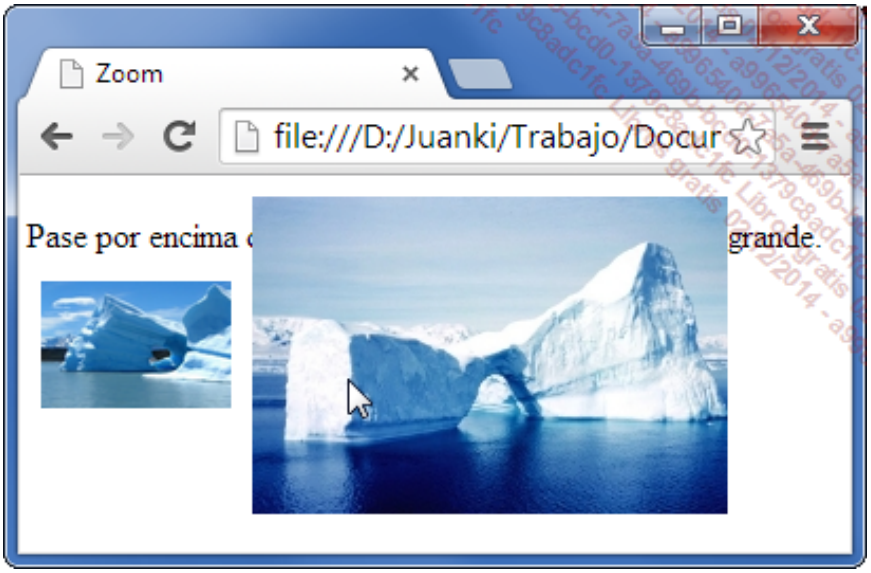
```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
```

```
$(document).ready(function() {
$('ul#menu li a').hover(function() {
$(this).stop().animate( { paddingLeft:"50px" }, 400 );
},
function() {
$(this).stop().animate( { paddingLeft:"0" }, 200 )
})
});
</script>
<style>
#nav ul { list-style: none;
          padding: 0px;
          margin: 0px;}
#nav li a { display: block;
            height: 25px;
            line-height: 25px;
            width: 120px;
            background: #99ccff;
            border: 1px solid navy;
            color: black;
            text-decoration: none;
            text-align: center;}
</style>
</head>
<body>
<div id="nav">
<ul id="menu">
<li><a href="#">Menú Ítem 1</a></li>
<li><a href="#">Menú Ítem 2</a></li>
<li><a href="#">Menú Ítem 3</a></li>
<li><a href="#">Menú Ítem 4</a></li>
<li><a href="#">Menú Ítem 5</a></li>
</ul>
</div>
</body>
</html>
```

2. Zoom en una viñeta

Vamos a diseñar un script que permita mostrar una imagen en tamaño real al pasar el ratón por encima de una viñeta de ésta.

La transcripción en papel no permite percibir la parte dinámica del script. Vaya al espacio de descarga para disfrutarlo completamente.



El archivo Html inicial:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Zoom</title>
<style>
body { padding: 3px;
       margin: 0px;}
.container { position: absolute;
             top: 50px;
             left: 10px;}
ul.thumb { padding: 0px;
           list-style-type: none;
           margin: 0px;
           width: 260px;
           float: left;}
ul.thumb li { position: relative;
              padding: 0px;
              margin: 0px;
              width: 100px;
              height: 100px;
              float: left;}
ul.thumb li img { position: absolute;
                  padding: 0px;
                  width: 90px;
                  height: 60px;
                  left: 0px;
                  border: none;}
</style>
</head>
<body>
<p>Pase por encima de la viñeta para hacer la imagen más grande.</p>
<div class="container">
<ul class="thumb">
<li><img alt="" src="iceberg1.jpg"></li>
<li><img alt="" src="iceberg2.jpg"></li>
</ul>
</div>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$("ul.thumb li").hover(function() {
$(this).css({'z-index': '10'});
$(this).find('img').addClass("hover").stop()
.animate({ marginTop: '-90px',
           marginLeft: '-50px',
           top: '50%',
           left: '50%',
           width: '225px',
           height: '150px',
           }, 200);
} ,
function() {
$(this).css({'z-index': '0'});
$(this).find('img').removeClass("hover").stop()
.animate({ marginTop: '0',
           marginLeft: '0',
           top: '0',
           left: '0',
           width: '90px',
           height: '60px',
           }, 400);
});
});
</script>
```

Este script necesita algunas aclaraciones:

```
$(document).ready(function(){
$("ul.thumb li").hover(function() {
```

Aplicación de un `hover()` en la viñeta.

```
$(this).css({'z-index': '10'});
.animate({ marginTop: '-90px',
           marginLeft: '-50px',
           top: '50%',
           left: '50%',
           width: '225px',
           height: '150px',
           }, 200);
} ,
```

Al pasar el ratón por encima de la viñeta, se modifica la propiedad de estilo `z-index` (`css({'z-index': '10'})`) para que la imagen pase al frente. Después una animación (`animate()`) modifica una serie de parámetros.

```
function() {
$(this).css({'z-index': '0'});
$(this).find('img').removeClass("hover").stop()
.animate({ marginTop: '0',
           marginLeft: '0',
           top: '0',
           left: '0',
           width: '90px',
           height: '60px',
           }, 400);
});
```

Cuando el cursor sale de la imagen, la propiedad de estilo `z-index` se restablece a su estado inicial, lo que provoca, a su vez, una animación (`animate()`) en una serie de parámetros.

```
});
```

Fin del script.

El archivo final es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>Zoom</title>
<style>
body { padding: 3px;
       margin: 0px;}
.container { position: absolute;
             top: 50px;
             left: 10px;}
ul.thumb { padding: 0px;
           list-style-type: none;
           margin: 0px;
           width: 260px;
           float: left;}
ul.thumb li { position: relative;
              padding: 0px;
              margin: 0px;
              width: 100px;
              height: 100px;
              float: left;}
ul.thumb li img { position: absolute;
                  padding: 0px;
                  width: 90px;
                  height: 60px;
                  left: 0px;
                  border: none;}
</style>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("ul.thumb li").hover(function() {
$(this).css({'z-index': '10'});
$(this).find('img').addClass("hover").stop()
.animate({ marginTop: '-90px',
           marginLeft: '-50px',
           top: '50%',
           left: '50%',
           width: '225px',
           height: '150px',
           }, 200);
} ,
function() {
$(this).css({'z-index': '0'});
$(this).find('img').removeClass("hover").stop()
.animate({ marginTop: '0',
           marginLeft: '0',
           top: '0',
           left: '0',
           width: '90px',
           height: '60px',
           }, 400);
});
});
</script>
</head>
<body>
```

```
<p>Pase por encima de la viñeta para hacer la imagen más grande.</p>
<div class="container">
<ul class="thumb">
<li><img alt="" src="iceberg1.jpg"></li>
<li><img alt="" src="iceberg2.jpg"></li>
</ul>
</div>
</body>
</html>
```

Introducción

Las animaciones visuales son una parte esencial de JavaScript. Cuando se usan adecuadamente y con moderación, pueden potenciar la funcionalidad de un elemento del contenido. La Web 2.0 las ha adoptado extensamente.

Hay que admitir que hacer una animación gráfica avanzada usando únicamente JavaScript se convierte con rapidez en una pesadilla de programación, sobre todo si hay que tener en cuenta cada una de las particularidades de los diferentes navegadores, incluso en cada versión de ellos. El framework jQuery propone una serie de efectos visuales fáciles de codificar y totalmente compatibles. En otras palabras, jQuery le ofrece también la posibilidad de crear sus propias animaciones.

Este capítulo es muy visual, con efectos y otras animaciones. Se aconseja consultar los ejemplos que se proporcionan en el espacio de descarga para entender el funcionamiento real.

Mostrar y ocultar

Los métodos `show()` y `hide()` de jQuery permiten hacer aparecer y desaparecer los elementos.

show(velocidad, función a la que se llama)

Muestra un elemento seleccionado (siempre que esté oculto).

La animación modifica dinámicamente la altura, la anchura y la opacidad del elemento. Desde la especificación jQuery 1.3, los márgenes externos e internos también se pueden modificar para obtener un efecto más fluido.

- "velocidad" (opcional): cadena de caracteres que representa una de las tres velocidades predefinidas (`'slow'`, `'normal'` o `'fast'`) o el número en milisegundos que corresponden a la duración del efecto.

```
$('p').show('slow');
```

- "función a la que se llama" (callback) (opcional): función que se tiene que ejecutar cuando termina el efecto.

```
$('p').show('slow', function(){alert("Fin");});
```

Este método devuelve un objeto jQuery.

hide(velocidad, función a la que se llama)

Oculta un elemento seleccionado (siempre que esté visible).

La animación modifica dinámicamente la altura, la anchura y la opacidad del elemento. Desde la especificación jQuery 1.3, los márgenes externos e internos también se pueden modificar para obtener un efecto más fluido.

- "velocidad" (opcional): cadena de caracteres que representa una de las tres velocidades predefinidas (`'slow'`, `'normal'` o `'fast'`) o el número en milisegundos que corresponden a la duración del efecto.

```
$('p').hide('fast');
```

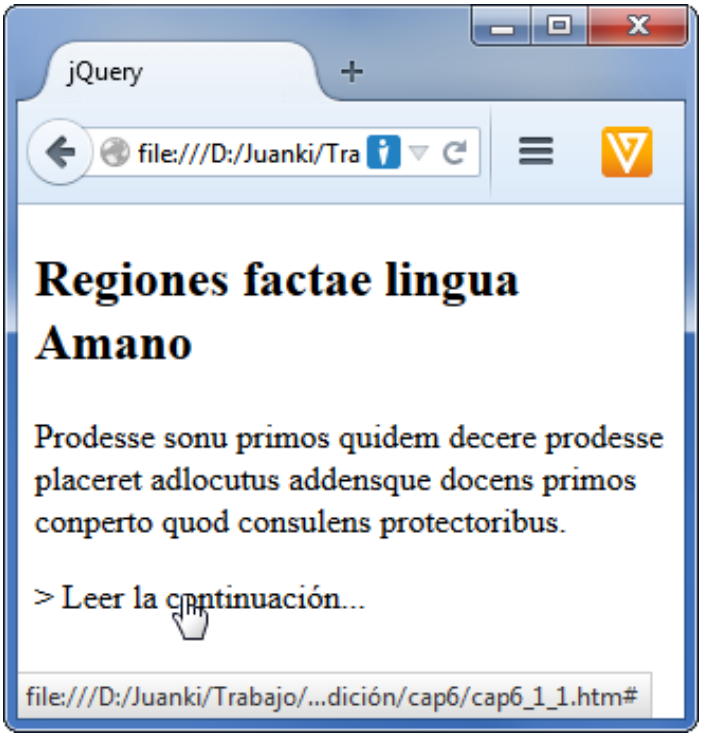
- "función a la que se llama" (callback) (opcional): función que se tiene que ejecutar cuando termina el efecto.

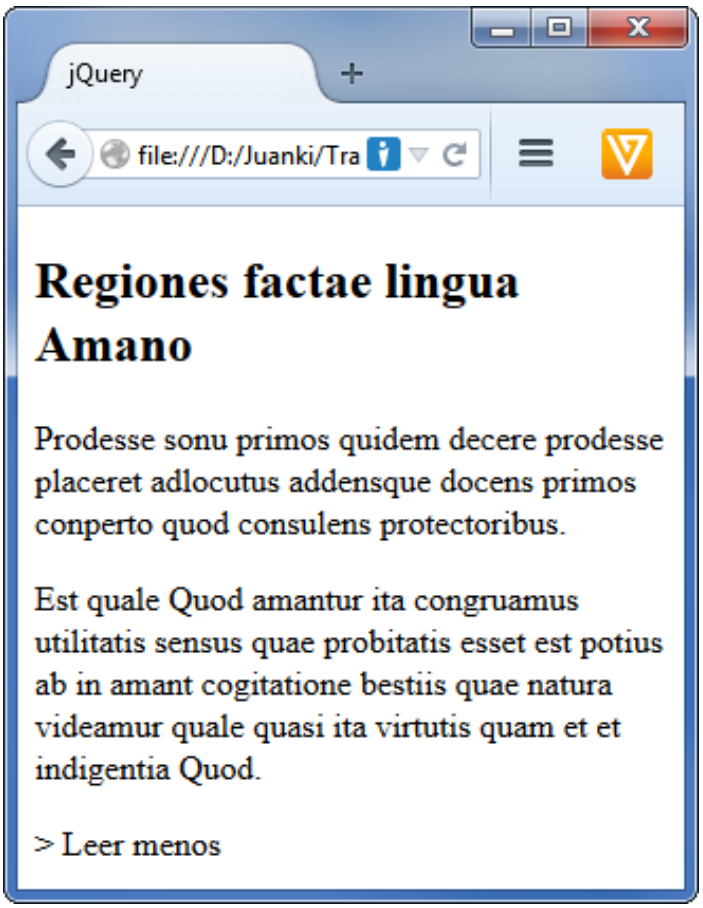
```
$('p').hide('normal', function(){alert("Fin");});
```

Este método devuelve un objeto jQuery.

1. Mostrar y ocultar texto

El ejemplo siguiente propone al lector mostrar la continuación de un artículo, para lo que tiene que hacer clic en el enlace.





El archivo Html inicial es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
a { color: black;
```

```
    text-decoration: none;}
</style>
</head>
<body>
<h2>Regiones factae lingua Amano</h2>
<p>
Prodesse sonu primos quidem decere prodesse placeret adlocutus
addensque docens primos conperto quod consulens protectoribus.
</p>
<p>
Est quale Quod amantur ita congruamus utilitatis sensus quae
probitatis esset es potius ab in amant cogitatione bestiis quae
natura videamur quale quasi ita virtutis quam y et indigentia
Quod.
</p>
<div>
<a href="#" class="continuar">> Leer la continuación...</a>
</div>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function() {
var $parrafo2 = $('p:eq(1)');
$parrafo2.hide();
$('a.continuar').on("click", function() {
if ($parrafo2.is(':hidden')) {
$parrafo2.show('slow');
$(this).text('> Leer menos');
} else {
$parrafo2.hide('slow');
$(this).text('> Leer la continuación...');
return false;
}
});
});
</script>
```

Lo detallamos a continuación:

```
$(document).ready(function() {
var $parrafo2 = $('p:eq(1)');
```

Cuando se carga el DOM, el segundo párrafo se carga en la variable `$parrafo2`. Recordemos que el método `eq()` empieza en 0, como sucede normalmente en JavaScript.

Los programadores usan habitualmente la convención de asignar identificadores a las variables en un script jQuery de modo que empiecen por el signo $.

```
$parrafo2.hide();
```

El segundo párrafo se oculta cuando se carga la página.

```
$('a.continuar').on("click", function() {
if ($parrafo2.is(':hidden')) {
$parrafo2.show('slow');
$(this).text('> Leer menos');
```

Al hacer clic en el enlace **> Leer la continuación...**, si el segundo párrafo se ha ocultado correctamente (`hidden`), entonces se muestra a una velocidad lenta predefinida y el texto del enlace se cambia por **> Leer menos**.

```
} else {
$parrafo2.hide('slow');
$(this).text('> Leer la continuación...');
return false;
}
```

En caso contrario, se oculta el segundo párrafo y el texto del enlace se cambia por "> Leer la continuación...".

```
});
});
```

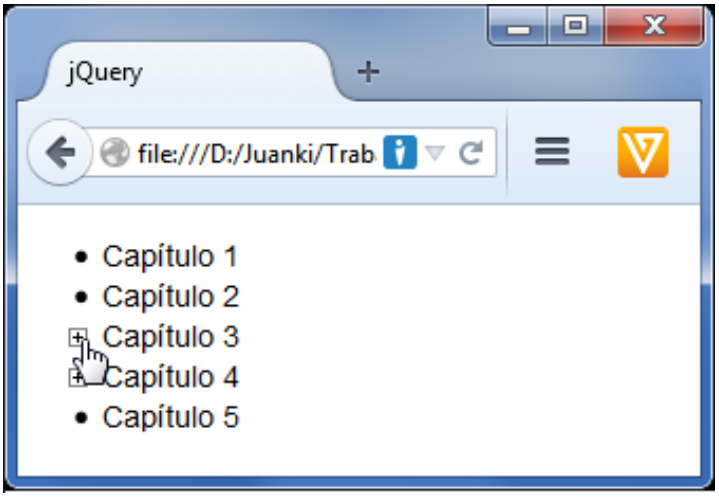
Fin del script.

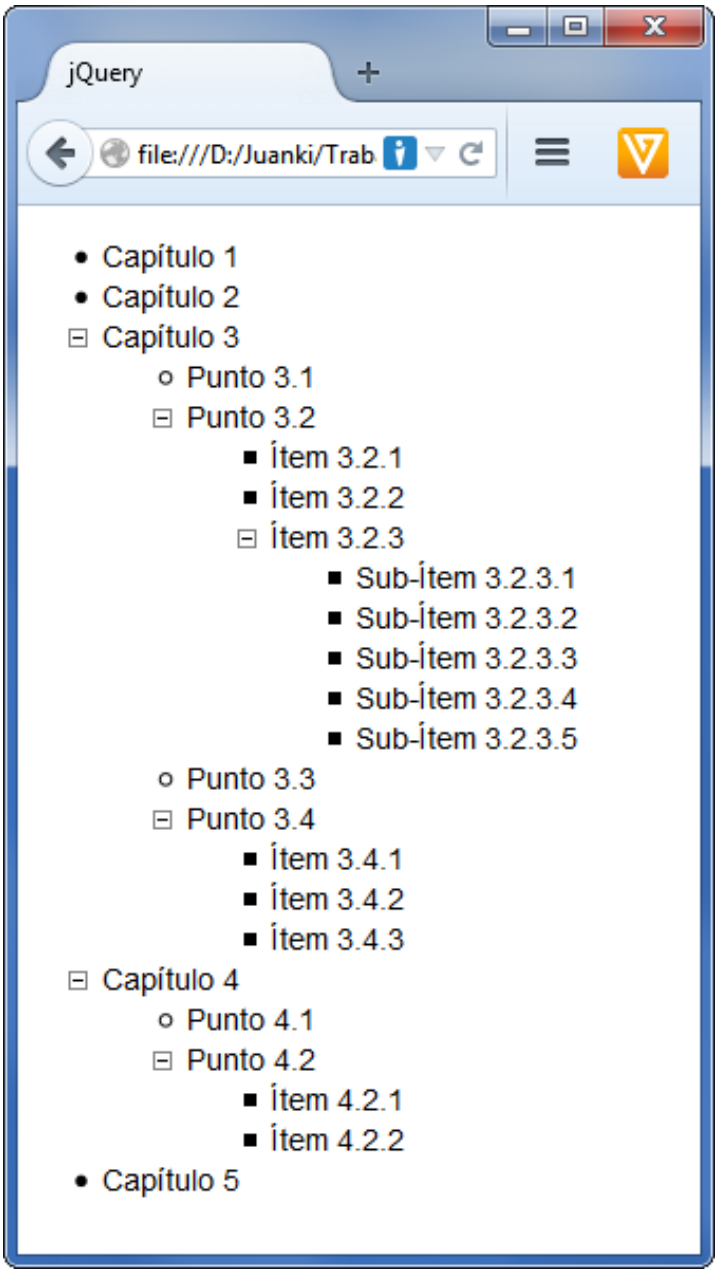
El archivo completo se muestra a continuación:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function() {
var $parrafo2 = $('p:eq(1)');
$parrafo2.hide();
$('a.continuar').on("click", function() {
if ($parrafo2.is(':hidden')) {
$parrafo2.show('slow');
$(this).text('> Leer menos');
} else {
$parrafo2.hide('slow');
$(this).text('> Leer la continuación...');
return false;
}
});
});
</script>
<style>
a { color: black;
    text-decoration: none;}
</style>
</head>
<body>
<h2>Regiones factae lingua Amano</h2>
<p>
Prodesse sonu primos quidem decere prodesse placeret adlocutus
addensque docens primos conperto quod consulens protectoribus.
</p>
<p>
Est quale Quod amantur ita congruamus utilitatis sensus quae
probitatis esset es potius ab in amant cogitatione bestiis quae
natura videamur quale quasi ita virtutis quam y et indigentia
Quod.
</p>
<div>
<a href="#" class="continuar">> Leer la continuación...</a>
</div>
</body>
</html>
```

2. Desplegar listas anidadas

Vamos a permitir el despliegue de las listas anidadas haciendo clic en una imagen.





Las imágenes mas.gif ⊞ y menos.gif ⊟ están disponibles para descarga en la página Información.

El archivo Html inicial es el siguiente:

```
<!doctype html>
```

```
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
body{ font-family: Arial,sans-serif;
      font-size: 0.9em;
      margin-left: 0px;}
</style>
</head>
<body>
<ul>
<li>Capítulo 1</li>
<li>Capítulo 2</li>
<li>Capítulo 3
<ul>
<li>Punto 3.1</li>
<li>Punto 3.2
<ul>
<li>Ítem 3.2.1</li>
<li>Ítem 3.2.2</li>
<li>Ítem 3.2.3
<ul>
<li>Sub-ítem 3.2.3.1</li>
<li>Sub-ítem 3.2.3.2</li>
<li>Sub-ítem 3.2.3.3</li>
<li>Sub-ítem 3.2.3.4</li>
<li>Sub-ítem 3.2.3.5</li>
</ul>
</li>
</ul>
</li>
<li>Punto 3.3</li>
<li>Punto 3.4
<ul>
<li>Ítem 3.4.1</li>
<li>Ítem 3.4.2</li>
<li>Ítem 3.4.3</li>
</ul>
</li>
</ul>
</li>
<li>Capítulo 4
<ul>
<li>Punto 4.1</li>
<li>Punto 4.2
<ul>
<li>Ítem 4.2.1</li>
<li>Ítem 4.2.2</li>
</ul>
</li>
</ul>
</li>
<li>Capítulo 5</li>
</ul>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function() {
$('li:has(ul)')
.click(function(){
if ($(this).children().is(':hidden')) {
```

```
$(this).css('list-style-image','url(menos.gif)')
       .children().show();
}
else {
$(this).css('list-style-image','url(mas.gif)')
       .children().hide();
}
return false;
})
.css('cursor','pointer')
.click();
$('li:not(:has(ul))').css({cursor: 'default',
                           'list-style-image':'none'
});
});
</script>
```

Lo detallamos a continuación.

```
$(document).ready(function() {
$('li:has(ul)')
```

Cuando se carga el DOM, el script selecciona todos los elementos de lista (`<li>`) que tienen una lista anidada (`<ul>`).

```
.click(function(){
```

Al hacer clic en estos elementos.

```
if ($(this).children().is(':hidden')) {
$(this).css('list-style-image','url(menos.gif)')
       .children().show();
}
```

Si las listas anidadas de este elemento (es decir, las descendientes) están ocultas, la imagen se cambia a menos.gif y éstas se despliegan.

```
else {
$(this).css('list-style-image','url(mas.gif)')
       .children().hide();
}
return false;
})
```

En caso contrario, se mostrará la imagen mas.gif y las listas anidadas se ocultan.

```
.css('cursor','pointer')
.click();
```

Solo falta ocuparnos de la forma del cursor del ratón. Para los elementos seleccionados por las líneas de código anteriores (`li:has(ul)`), el cursor se transforma en una mano (`pointer`).

```
$('li:not(:has(ul))').css({cursor: 'default',
                           'list-style-image':'none'
});
```

Para los elementos que no tienen lista anidada (`'li:not(:has(ul))'`), se conserva la forma del cursor por defecto y no se muestra ningún icono.

```
});
```

Fin del script.

El archivo completo se muestra a continuación:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
body{ font-family: Arial,sans-serif;
      font-size: 0.9em;
      margin-left: 0px;}
</style>
<script src="jquery.js"></script>
<script>
$(document).ready(function() {
$('li:has(ul)')
.click(function(){
if ($(this).children().is(':hidden')) {
$(this).css('list-style-image','url(menos.gif)')
       .children().show();
}
else {
$(this).css('list-style-image','url(mas.gif)')
       .children().hide();
}
return false;
})
.css('cursor','pointer')
.click();
$('li:not(:has(ul))').css({'cursor': 'default',
                           'list-style-image':'none'
});
});
</script>
</head>
<body>
<ul>
<li>Capítulo 1</li>
<li>Capítulo 2</li>
<li>Capítulo 3
<ul>
<li>Punto 3.1</li>
<li>Punto 3.2
<ul>
<li>Ítem 3.2.1</li>
<li>Ítem 3.2.2</li>
<li>Ítem 3.2.3
<ul>
<li>Sub-ítem 3.2.3.1</li>
<li>Sub-ítem 3.2.3.2</li>
<li>Sub-ítem 3.2.3.3</li>
<li>Sub-ítem 3.2.3.4</li>
<li>Sub-ítem 3.2.3.5</li>
</ul>
</li>
</ul>
</li>
<li>Punto 3.3</li>
<li>Punto 3.4
<ul>
<li>Ítem 3.4.1</li>
<li>Ítem 3.4.2</li>
<li>Ítem 3.4.3</li>
</ul>
</li>
```

```
</ul>
</li>
<li>Capítulo 4
<ul>
<li>Punto 4.1</li>
<li>Punto 4.2
<ul>
<li>Ítem 4.2.1</li>
<li>Ítem 4.2.2</li>
</ul>
</li>
</ul>
</li>
<li>Capítulo 5</li>
</ul>
</body>
</html>
```

Desplazar verticalmente

Las funciones `slideDown()` y `slideUp()` permiten jugar dinámicamente con la altura de un elemento, normalmente una capa `<div> ... </div>`.

slideDown(velocidad, función a la que se llama)

Desplazar hacia abajo (down) un elemento seleccionado.

La animación modifica solo la altura. Desde la especificación jQuery 1.3, los márgenes verticales, externos e internos también se pueden modificar para obtener un efecto más fluido.

- "velocidad" (opcional): cadena de caracteres que representa una de las tres velocidades predefinidas (`'slow'`, `'normal'` o `'fast'`) o el número en milisegundos que corresponden a la duración del efecto.

  ```
  $('div').slideDown('fast');
  ```

- "función a la que se llama" (callback) (opcional): función que se tiene que ejecutar cuando termina el efecto.

  ```
  $('div').slideDown('fast', function(){alert("Fin");});
  ```

Este método devuelve un objeto jQuery.

slideUp(velocidad, función a la que se llama)

Desplazar hacia arriba (up) un elemento seleccionado.

La animación modifica solo la altura. Desde la especificación jQuery 1.3, los márgenes verticales, externos e internos también se pueden modificar para obtener un efecto más fluido.

- "velocidad" (opcional): cadena de caracteres que representa una de las tres velocidades predefinidas (`'slow'`, `'normal'` o `'fast'`) o el número en milisegundos que corresponden a la duración del efecto.

  ```
  $('div').slideUp('fast');
  ```

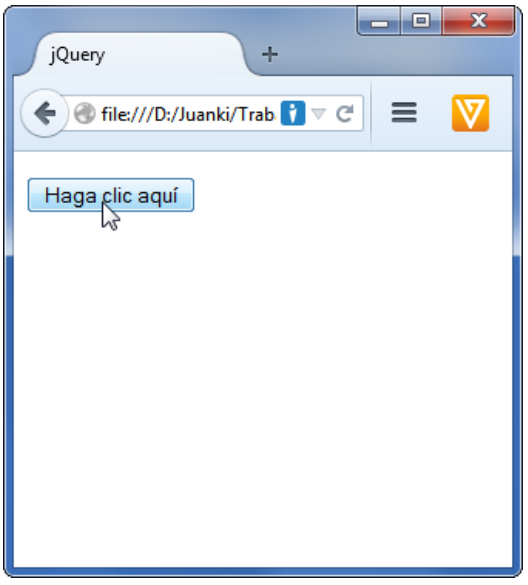
- "función a la que se llama" (callback) (opcional): función que se tiene que ejecutar cuando termina el efecto.

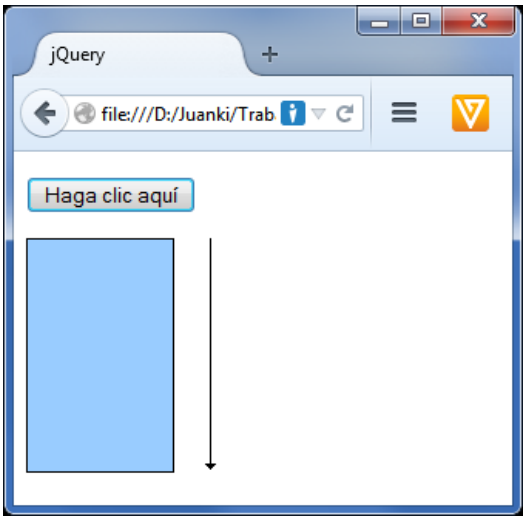
  ```
  $('div').slideUp('fast', function(){alert("Fin");});
  ```

Este método devuelve un objeto jQuery.

1. Desplazar una capa

Al hacer clic en el botón, vamos a desplegar una capa.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
div { width:94px;
      height:150px;
      background: #9cf;
      border: 1px solid black;
      display:none;}
</style>
</head>
<body>
<p>
<button>Haga clic aquí</button>
</p>
<div></div>
</body>
</html>
```

El script jQuery:

Explicaciones:

```
$(document).ready(function(){
$("button").on("click", function ()
```

Cuando se ha cargado el DOM, al hacer clic con el ratón en la etiqueta `<button>`.

```
if ($("div").is(":hidden")) {
$("div").slideDown("slow");
}
```

Si la capa (`<div>`) está oculta, se desplaza hacia abajo.

```
else
{$("div").slideUp("slow");
}
```

En caso contrario, la capa se desplaza hacia arriba.

```
});
});
```

Fin del script.

La página completa:

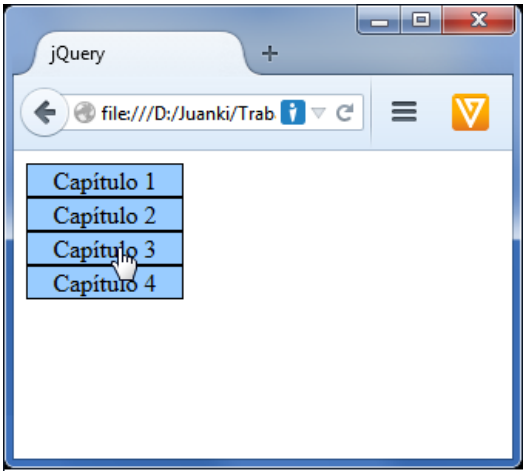
```
<script>
$(document).ready(function(){
$("button").on("click", function() {
if ($("div").is(":hidden")) {
$("div").slideDown("slow");
}
else
{$("div").slideUp("slow");
}
});
});
</script>
```

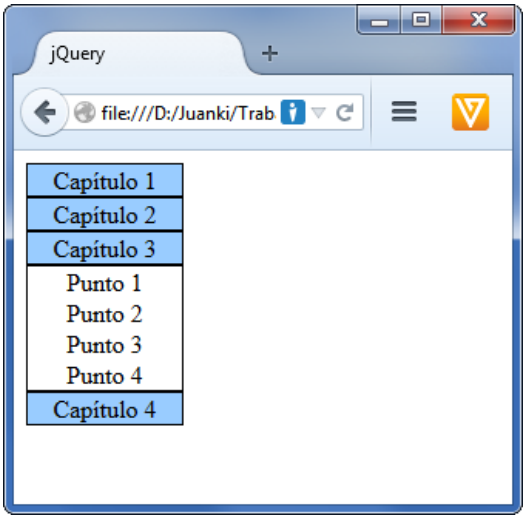
```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("button").on("click", function () {
if ($("div").is(":hidden")) {
$("div").slideDown("slow");
}
else
{$("div").slideUp("slow");
}
});
});
</script>
<style>
div { width:94px;
      height:150px;
      background: #9cf;
      border: 1px solid black;
      display:none;}
</style>
</head>
<body>
<p>
<button>Haga clic aquí</button>
```

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

2. Un menú desplegable vertical

Vamos a hacer un menú desplegable vertical muy sencillo.





El archivo Html inicial es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
a { text-decoration: none;
    color: black;}
.capitulo { width: 100px;
            background: #9cf;
            border-bottom: 1px solid black;
            cursor: pointer;
            text-align: center;
            border: 1px solid black;}
.items { width: 100px;
         text-align: center;
         border: 1px solid black;
         display: none;}
</style>
</head>
<body>
<div class="capitulo">Capítulo 1</div>
<div class="items">
<a href="">Punto 1</a><br />
<a href="">Punto 2</a><br />
<a href="">Punto 3</a>
</div>
<div class="capitulo">Capítulo 2</div>
<div class="items">
<a href="">Punto 1</a><br />
<a href="">Punto 2</a>
</div>
```

El script jQuery:

```
$(document).ready(function() {
$('div.capitulo').click(function() {
```

Cuando se carga el DOM, al hacer clic en la capa cuya clase es `capitulo`.

```
$('div.items').slideUp('normal');
```

Las capas `items` se cierran (desplazan hacia arriba).

```
$(this).next().slideDown('normal');
});
```

El script consigue llegar a los elementos siguientes (`next`) del elemento seleccionado (`this`), gracias al clic en la capa `capitulo`.

```
$("div.items").hide();
```

Se ocultan las capas `items`, para que el menú se presente colapsado cuando se abra la página.

```
});
```

Fin del script.

```
<div class="capitulo">Capítulo 3</div>
<div class="items">
<a href="">Punto 1</a><br />
<a href="">Punto 2</a><br />
<a href="">Punto 3</a><br />
<a href="">Punto 4</a>
</div>
<div class="capitulo">Capítulo 4</div>
<div class="items">
<a href="">Punto 1</a><br />
<a href="">Punto 2</a><br />
<a href="">Punto 3</a>
</div>
</body>
</html>
```

El archivo completo es el siguiente:

```
<script>
$(document).ready(function() {
$('div.capitulo').click(function() {
$('div.items').slideUp('normal');
$(this).next().slideDown('normal');
});
$("div.items").hide();
});
</script>
```

```
doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function() {
$('div.capitulo').click(function() {
$('div.items').slideUp('normal');
$(this).next().slideDown('normal');
});
$("div.items").hide();
});
</script>
<style>
a { text-decoration: none;
    color: black;}
.capitulo { width: 100px;
            background: #9cf;
            border-bottom: 1px solid black;
            cursor: pointer;
            text-align: center;
            border: 1px solid black;}
.items { width: 100px;
         text-align: center;
         border: 1px solid black;
         display: none;}
</style>
</head>
<body>
<div class="capitulo">Capítulo 1</div>
<div class="items">
<a href="">Punto 1</a><br>
<a href="">Punto 2</a><br>
<a href="">Punto 3</a>
</div>
<div class="capitulo">Capítulo 2</div>
<div class="items">
<a href="">Punto 1</a><br>
<a href="">Punto 2</a>
</div>
<div class="capitulo">Capítulo 3</div>
<div class="items">
<a href="">Punto 1</a><br>
<a href="">Punto 2</a><br>
<a href="">Punto 3</a><br>
<a href="">Punto 4</a>
</div>
<div class="capitulo">Capítulo 4</div>
<div class="items">
<a href="">Punto 1</a><br>
<a href="">Punto 2</a><br>
<a href="">Punto 3</a>
</div>
</body>
</html>
```

Hacer un efecto de fundido

Este efecto de aparición o desaparición de un elemento, modificando progresivamente su opacidad, posiblemente resulte ser el efecto más bonito de jQuery de todos los que se presentan en este capítulo, dedicado a los efectos.

fadeIn(velocidad, función a la que se llama)

Hace aparecer el elemento seleccionado, siguiendo un efecto de fundido.

Esta animación se obtiene ajustando solo la opacidad. El elemento seleccionado debe tener una anchura y una altura concreta.

- "velocidad" (opcional): cadena de caracteres que representa una de las tres velocidades predefinidas (’slow’, ’normal’ o ’fast’) o el número en milisegundos que corresponden a la duración del efecto.

```
$(’p:first’).fadeIn(4000);
```

- "función a la que se llama" (callback) (opcional): función que se tiene que ejecutar cuando termina el efecto.

```
$(’p:first’).fadeIn(4000, function(){alert("Fin");});
```

Este método devuelve un objeto jQuery.

fadeOut(velocidad, función a la que se llama)

Hace desaparecer el elemento seleccionado, siguiendo un efecto de fundido.

Esta animación se obtiene ajustando solo la opacidad. El elemento seleccionado debe tener una anchura y una altura concretas.

- "velocidad" (opcional): cadena de caracteres que representa una de las tres velocidades predefinidas (’slow’, ’normal’ o ’fast’) o el número en milisegundos que corresponden a la duración del efecto.

```
$(’p:first’).fadeOut(4000);
```

- "función a la que se llama" (callback) (opcional): función que se tiene que ejecutar cuando termina el efecto.

```
$(’p:first’).fadeOut(4000, function(){alert("Fin");});
```

Este método devuelve un objeto jQuery.

fadeTo(velocidad, opacidad, función a la que se llama)

Modifica la opacidad del elemento seleccionado hasta el valor que se proporciona en el script.

- "velocidad" (opcional): cadena de caracteres que representa una de las tres velocidades predefinidas (’slow’, ’normal’ o ’fast’) o el número en milisegundos que corresponden a la duración del efecto.

```
$( ’p:first’).fadeTo(’slow’, 0.33);
```

- "opacidad" (número): determina el valor de la opacidad que se tiene que alcanzar (número comprendido entre 0 y 1).
- "función a la que se llama" (callback) (opcional): función que se tiene que ejecutar cuando termina el efecto.

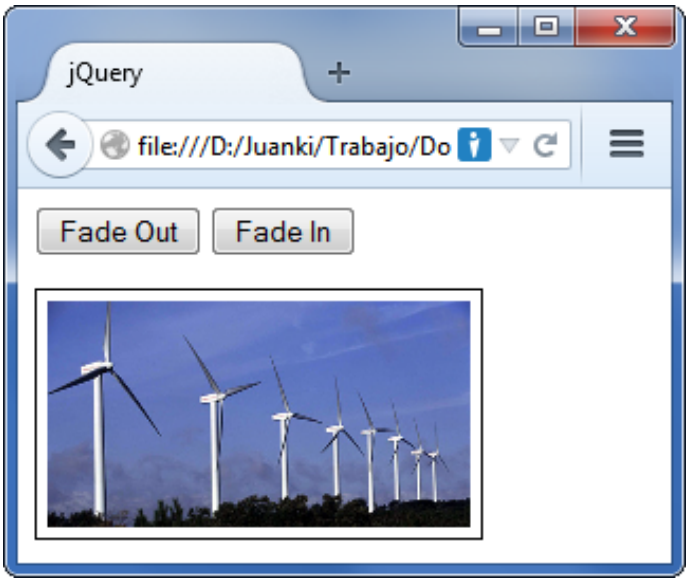
```
$(’p:first’).fadeTo(’slow’, 0.33, function(){alert("Fin");});
```

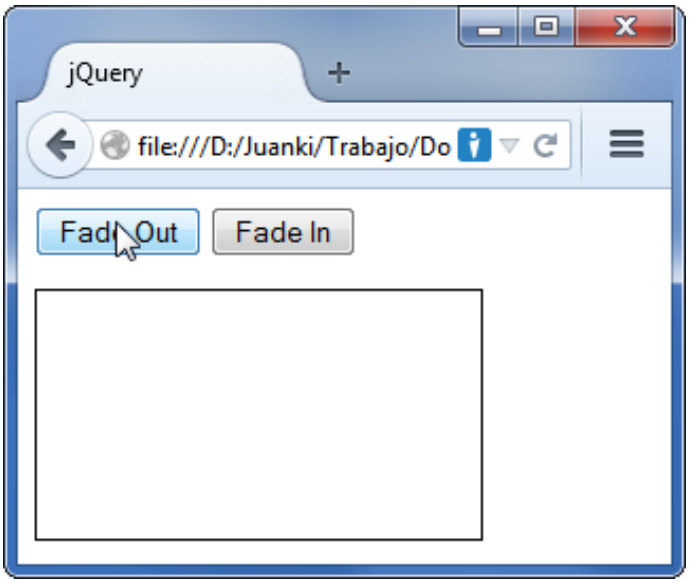
Este método devuelve un objeto jQuery.

1. Aparición y desaparición progresiva

Pongamos en acción estos métodos de jQuery en una imagen.

Las imágenes de este ejemplo están disponibles para descarga en la página Información.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
#bloc { width: 210px;
        height: 117px;
        border: 1px solid black;
        margin-top: 15px;}
#image { margin: 5px;}
</style>
</head>
<body>
<div>
<input type="button" id="botonFadeOut" value="Fade Out">
<input type="button" id="botonFadeIn" value="Fade In">
</div>
<div id="bloc">
<img src="molino3.jpg" alt="" width="200" height="107" id="image" />
</div>
</body>
</html>
```

La parte del script:

```
<script>
$(document).ready(function(){
$("#botonFadeOut").click(function () {
$("#image").fadeOut();
});
$("#botonFadeIn").click(function () {
$("#image").fadeIn();
});
});</script>
```

Lo detallamos a continuación.

```
$(document).ready(function(){
$("#botonFadeOut").click(function () {
$("#image").fadeOut();
});
```

Después de cargar el DOM, hacer clic en el botón **Fade Out** hace desaparecer al imagen con un efecto de fundido.

```
$("#botonFadeIn").click(function () {
$("#image").fadeIn();
});
```

El clic en el botón **Fade In** hace aparecer la imagen con un efecto de fundido.

```
});
```

Fin del script.

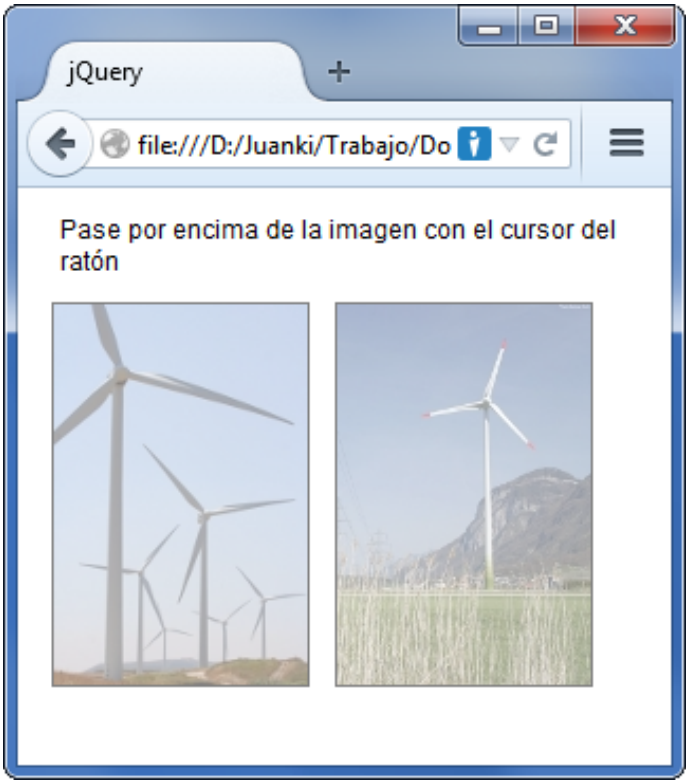
El archivo final:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title> jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#botonFadeOut").click(function () {
$("#image").fadeOut();
});
$("#botonFadeIn").click(function () {
$("#image").fadeIn();
});
});
</script>
<style>
#bloc { width: 210px;
        height: 117px;
        border: 1px solid black;
        margin-top: 15px;}
#image { margin: 5px;}
</style>
</head>
<body>
<div>
<input type="button" id="botonFadeOut" value="Fade Out">
<input type="button" id="botonFadeIn" value="Fade In">
</div>
<div id="bloc">
<img src="molino3.jpg" alt="" width="200" height="107" id="image">
</div>
</body>
</html>
```

2. Jugar con la opacidad

Vamos a hacer que aparezca clara una imagen que, cuando se carga la página, está sombreada.

Las imágenes de este ejemplo están disponibles para descarga en la página Información.



El archivo Html inicial es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
p { font-family:Arial, Helvetica, sans-serif;
    margin-left: 12px;
    font-size: 12px;}
.image { border: 1px solid black;
```

```
        margin-left: 8px;}
</style>
</head>
<body>
<p>Pase por encima de la imagen con el cursor del ratón</p>
<div>
<img src="molino1.jpg" alt="" width="120" height="180" class="imagen">
<img src="molino2.jpg" alt="" width="120" height="180" class="imagen">
</div>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$(".image").fadeTo("slow", 0.5);
$(".image").hover(function(){
$(this).fadeTo("slow", 1.0);
},function(){
$(this).fadeTo("fast", 0.5);
});
});
</script>
```

Explicaciones:

```
$(document).ready(function(){
$(".image").fadeTo("slow", 0.5);
```

Cuando se carga el DOM, se muestran las imágenes sombreadas, con una opacidad de 0,5.

```
$(".imagen").hover(function(){
$(this).fadeTo("slow", 1.0);
```

Al pasar el ratón por encima de la imagen, está se mostrará con su claridad normal (opacidad 1). El efecto de fundido es lento.

```
},function(){
$(this).fadeTo("fast", 0.5);
```

Cuando el cursor sale de la imagen, ésta vuelve a su estado inicial (opacidad 0,5).

```
});
});
```

Fin del script.

El archivo completo es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$(".image").fadeTo("slow", 0.5);
$(".image").hover(function(){
$(this).fadeTo("slow", 1.0);
},function(){
```

```
$(this).fadeTo("fast", 0.5);
});
});
</script>
<style>
p { font-family:Arial, Helvetica, sans-serif;
    margin-left: 12px;
    font-size: 12px;}
.image { border: 1px solid black;
         margin-left: 8px;}
</style>
</head>
<body>
<p>Pase por encima de la imagen con el cursor del ratón</p>
<div>
<img src="molino1.jpg" alt="" width="120" height="180" class="imagen">
<img src="molino2.jpg" alt="" width="120" height="180" class="imagen">
</div>
</body>
</html>
```

Cambiar de un efecto a otro

Este cambio de un estado a otro o de una función a otra es un clásico de jQuery y ya se ha tratado en el capítulo Los eventos.

toggle()

Permite cambiar el estado de visualización del elemento seleccionado. Si el elemento se muestra, la función lo hace desaparecer (con la función `hidden()`) y al revés (con la función `show()`).

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

Este método devuelve un objeto jQuery.

toggle(función 1,función2)

Permite cambiar (*toggle*) entre dos funciones con cada clic en el elemento seleccionado. Con el clic inicial, se ejecuta la primera función. Con el siguiente clic, se ejecuta la segunda. Si se hace clic de nuevo, se vuelve a ejecutar la primera función y así sucesivamente.

```
$("p").toggle(function(){
$(this).addClass("selected");
},function(){
$(this).removeClass("selected");
});
```

Este método devuelve un objeto jQuery.

Este efecto de cambio también se aplica a la función de deslizamiento (ver la sección Desplazar verticalmente, en este capítulo).

slideToggle(velocidad, función a la que se llama)

Esta función desplaza hacia abajo un elemento que está en estado "Up" y desplaza hacia arriba un elemento que está en estado "Down".

La animación solo modifica la altura. Desde la especificación jQuery 1.3, los márgenes verticales, externos e internos también se pueden modificar para obtener un efecto más fluido.

- "velocidad" (opcional): cadena de caracteres que representa una de las tres velocidades predefinidas (`'slow'`, `'normal'` o `'fast'`) o el número en milisegundos que corresponden a la duración del efecto.

  ```
  $('div').slideToggle('fast');
  ```

- "función a la que se llama" (callback) (opcional): función que se tiene que ejecutar cuando termina el efecto.

  ```
  $('div').slideToggle('fast', function(){alert("Fin");});
  ```

Este método devuelve un objeto jQuery.

fadeToggle(velocidad, función a la que se llama)

Muestra u oculta los elementos jugando con su opacidad.

- "velocidad" (opcional): cadena de caracteres que representa una de las tres velocidades definidas (`'slow'`, `'normal'`, `'fast'`) o el número en milisegundos que corresponden a la duración del efecto.

  ```
  $("p:first").fadeToggle("slow");
  ```

- "función a la que se llama" (opcional): función que se tiene que ejecutar cuando termina el efecto.

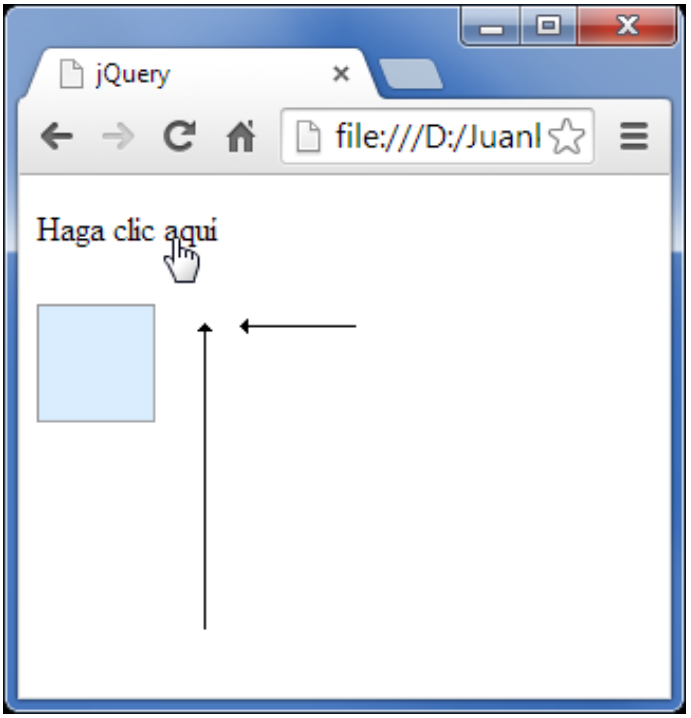
```
$("p:last").fadeToggle("fast", function () {
$("#log").append("<div>Terminado</div>");
});
```

Este efecto se ha añadido durante la versión 1.4.4.

Este método devuelve un objeto jQuery.

1. Ejemplo del efecto de cambio

Presentamos un script sencillo para describir esta función `toggle()`.




```
<!doctype html>
<html lang="es">
<head>
```

```
<meta charset="utf-8">
<title>jQuery</title>
<style>
#div_toggle { background-color: #9cf;
              width: 150px;
              height: 150px;
              border: 1px solid black;}
p { cursor: pointer;
    margin-bottom: 25px;}
</style>
</head>
<body>
<p id="start">
Haga clic aquí
</p>
<div id="div_toggle"></div>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function() {
$("#start").click(function () {
$("#div_toggle").toggle("slow");
});
});
</script>
```

Explicaciones:

```
$(document).ready(function() {
$("#start").click(function () {
```

Al hacer clic en el párrafo identificado por `id="start"`.

```
$("#div_toggle").toggle("slow");
```

Se aplica la función de cambio (`toggle`) a la capa.

```
});
});
```

Fin del script.

El documento final:

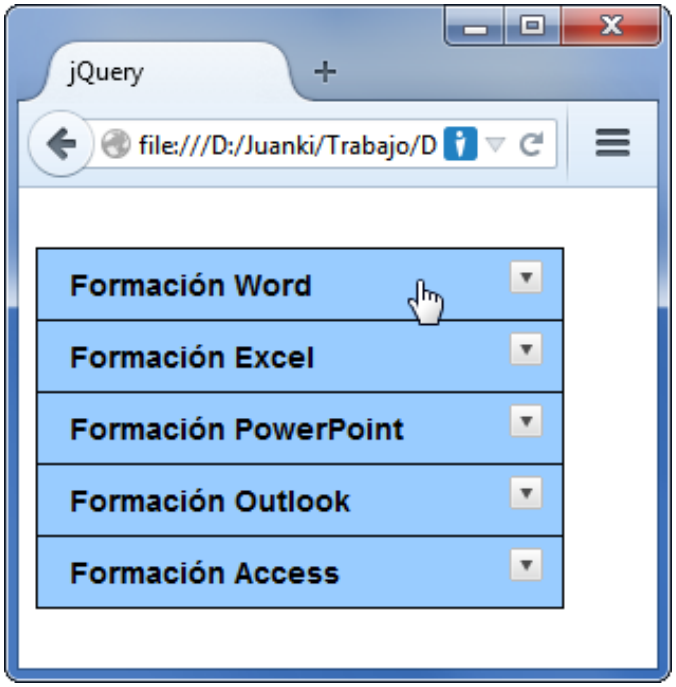
```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function() {
$("#start").click(function () {
$("#div_toggle").toggle("slow");
});
});
</script>
<style>
#div_toggle { background-color:#9cf;
              width:150px;
```

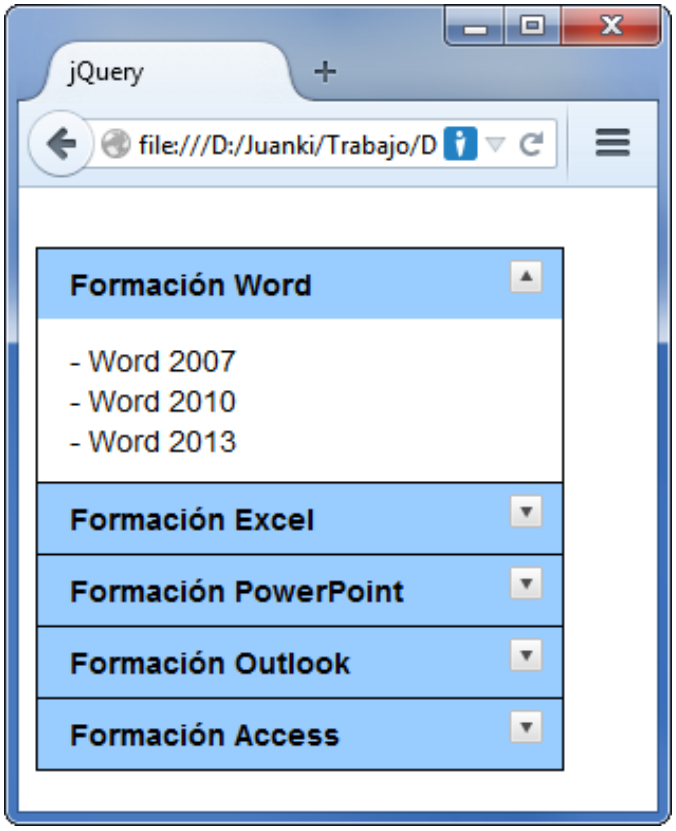
```
            height:150px;
            border: 1px solid black;}
p { cursor: pointer;
    margin-bottom: 25px;}
</style>
</head>
<body>
<p id="start">
Haga clic aquí
</p>
<div id="div_toggle"></div>
</body>
</html>
```

2. Un menú acordeón

Presentamos un menú vertical que se despliega y se pliega cuando hacemos clic con el ratón.

La imagen flecha.gif de este menú está disponible para descarga en la página Información.





El archivo de inicio es:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
.acordeon { width: 250px;
            border-bottom: solid 1px black;}
.acordeon h3 { margin: 0;
               background: #9cf url(flecha.gif) no-repeat right
               -51px;
               padding: 7px 15px;
               font: bold 0.9em Arial, sans-serif;
               border: solid 1px black;
               border-bottom: none;
               cursor: pointer;}
.acordeon h3:hover { background-color: #ccecff;}
.acordeon h3.activa { background-position: right 5px;}
.acordeon p { margin: 0;
              font: 0.9em Arial, sans-serif;
              padding: 10px 15px 10px;
              border-left: solid 1px black;
              border-right: solid 1px black;}
</style>
</head>
<body>
<br />
<div class="acordeon">
<h3>Formación Word</h3>
<p>- Word 2007<br>- Word 2010<br>- Word 2013</p>
<h3>Formación Excel</h3>
<p>- Excel 2007<br>- Excel 2010<br>- Excel 2013</p>
<h3>Formación PowerPoint</h3>
<p>- PowerPoint 2007<br>- PowerPoint  2010<br>- PowerPoint 2013</p>
<h3>Formación Outlook</h3>
```

```
<p>- Outlook 2007<br>- Outlook 2010<br>- Outlook 2013</p>
<h3>Formación Access</h3>
<p>- Access 2007<br>- Access 2010<br>- Access 2013</p>
</div>
</body>
</html>
```

Observe el uso de una única imagen (arrow.gif) para mostrar tanto la flecha hacia arriba como la flecha hacia abajo. La flecha hacia abajo se obtiene con una posición en el fondo de -51 píxeles, mientras que la flecha hacia arriba (la clase activa) tiene una posición de 5 píxeles.

El script jQuery:

```
<script>
$(document).ready(function(){
$(".acordeon p").hide();
$(".acordeon h3").click(function(){
$(this).next("p").slideToggle("slow")
.siblings("p:visible").slideUp("slow");
$(this).toggleClass("activa");
$(this).siblings("h3").removeClass("activa");
});
});
</script>
```

Lo detallamos a continuación.

```
$(document).ready(function(){
$(".acordeon p").hide();
```

Cuando se carga la página (el DOM, para ser precisos), el método `hide()` carga todos los submenús.

```
$(".acordeon h3").click(function(){
$(this).next("p").slideToggle("slow")
.siblings("p:visible").slideUp("slow");
```

Al hacer clic con el ratón en un elemento del menú (`.acordeon h3`), el script llama al elemento siguiente (es decir, una etiqueta `<p>`) y despliega el párrafo que contiene los submenús. Los elementos hermanos de la etiqueta `<p>` que son visibles, es decir, los otros submenús, se desplazan hacia arriba para desaparecer.

```
$(this).toggleClass("activa");
$(this).siblings("h3").removeClass("activa");
```

Estas dos líneas de código aseguran la visualización de la pequeña flecha hacia arriba o hacia abajo de los elementos del menú. Después del clic que se ha hecho antes, la flecha apunta hacia arriba, como define la clase activa. Para los otros elementos del menú, la flecha apuntará hacia abajo porque se ha eliminado la clase `activa`.

```
});
});
```

Fin del script.

El código final es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$(".acordeon p").hide();
$(".acordeon h3").on("click", function(){
$(this).next("p").slideToggle("slow")
.siblings("p:visible").slideUp("slow");
$(this).toggleClass("activa");
$(this).siblings("h3").removeClass("activa");
});
});
</script>
<style>
.acordeon { width: 250px;
            border-bottom: solid 1px black;}
.acordeon h3 { margin: 0;
               background: #9cf url(flecha.gif) no-repeat right
               -51px;
               padding: 7px 15px;
               font: bold 0.9em Arial, sans-serif;
               border: solid 1px black;
               border-bottom: none;
               cursor: pointer;}
.acordeon h3:hover { background-color: #ccecff;}
.acordeon h3.activa { background-position: right 5px;}
.acordeon p { margin: 0;
              font: 0.9em Arial, sans-serif;
              padding: 10px 15px 10px;
              border-left: solid 1px black;
              border-right: solid 1px black;}
</style>
</head>
<body>
<br />
<div class="acordeon">
<h3>Formación en Word</h3>
<p>- Word 2007<br>- Word 2010<br>- Word 2013</p>
<h3>Formación en Excel</h3>
<p>- Excel 2007<br>- Excel 2010<br>- Excel 2013</p>
<h3>Formación en PowerPoint</h3>
<p>- PowerPoint  2007<br>- PowerPoint 2010<br>- PowerPoint 2013</p>
<h3>Formación en Outlook</h3>
<p>- Outlook 2007<br>- Outlook 2010<br>- Outlook 2013</p>
<h3>Formación en Access</h3>
<p>- Access 2007<br>- Access 2010<br>- Access 2013</p>
</div>
</body>
</html>
```

Retrasar un efecto

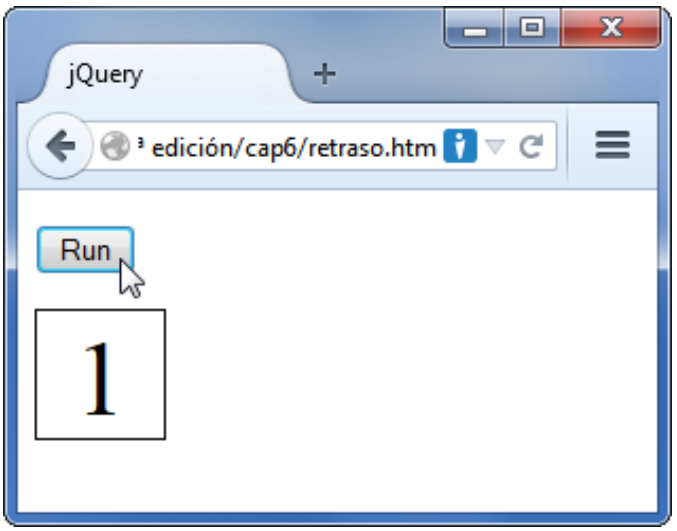
delay(duración)

Permite retrasar la ejecución de un efecto.

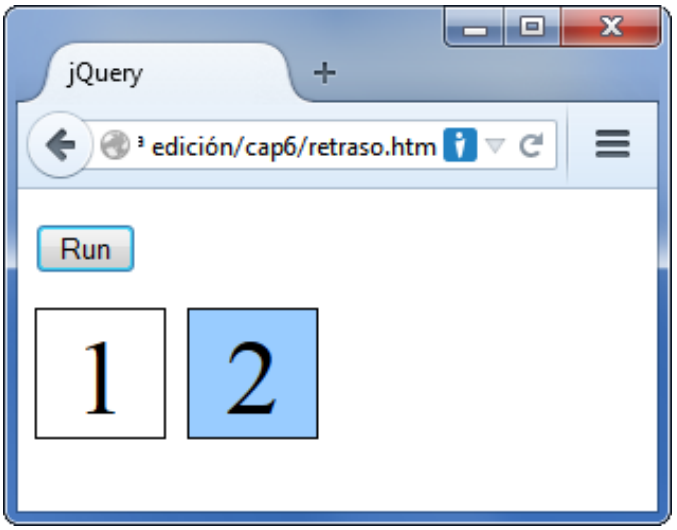
- "duración" es un número entero que expresa, en milisegundos, el retardo que se debe considerar a la hora de ejecutar el siguiente efecto.

Ejemplo:

Vamos a hacer que aparezcan las capas en dos tiempos, con un retraso de 2 segundos para la segunda.



Al hacer clic en el botón, solo se muestra la capa 1.



La capa 2 aparece con un retraso de 2 segundos.

El código es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
div { width: 60px; height: 60px;
      float: left;
      border: 1px solid black;
```

```
      font-size: 52px;
      text-align: center;}
.primero { background-color: white }
.segundo { background-color: #9cf;
           margin-left: 10px;}
</style>
<script src="jquery.js"></script>
<script>
$(document).ready(function() {
$("button").click(function() {
$("div.primero").slideUp(300).fadeIn(600);
$("div.segundo").slideUp(300).delay(2000).fadeIn(600);
});
});
</script>
</head>
<body>
<p><button type="button">Run</button></p>
<div class="primero">1</div>
<div class="segundo">2</div>
</body>
</html>
```

Explicaciones.

```
$(document).ready(function() {
$("button").click(function() {
```

Cuando se ha cargado el DOM, al hacer clic en el botón.

```
$("div.primero").slideUp(300).fadeIn(600);
```

La capa con la clase `primero` (`$("div.primero")`) se desplaza hacia arriba (`slideUp(300)`) y aparece progresivamente (`fadeIn(600)`).

```
$("div.segundo").slideUp(300).delay(2000).fadeIn(600);
```

La capa con la clase segundo (`$("div.segundo")`) se desplaza hacia arriba (`slideUp(300)`). Se produce una pausa de 2000 milisegundos (`delay(2000)`). Cuando termina esta pausa, aparece progresivamente (`fadeIn(600)`).

```
});
});
```

Fin del script.

Crear una animación

La función `animate()` le permite crear y configurar sus propias animaciones, siguiendo su propia creatividad.

animate(parámetros, velocidad, easing, función a la que se llama)

El aspecto clave de esta función es el objeto que componen las propiedades de estilo en las que se basará la animación. Cada parámetro del objeto representa una propiedad en la que se apoyará la animación (por ejemplo: `height`, `top` u `opacity`). El valor asociado a la clave indica cómo se animará la propiedad. Si el valor es un número, el estilo de la propiedad pasará de su valor actual al valor especificado. Si se especifican los valores `hide`, `show` o `toggle`, se construye una animación por defecto para esta propiedad.

Observe que estas propiedades se tienen que especificar siguiendo la notación JavaScript (CamelCase). Por ejemplo `marginLeft` en lugar de la notación CSS `margin-left`.

- "parámetros": contenedores de atributos de estilo que desea animar y con qué valor.
- "velocidad" (opcional): cadena de caracteres que representa una de las tres velocidades predefinidas (`'slow'`, `'normal'` o `'fast'`) o el número en milisegundos que corresponden a la duración del efecto.

```
animate( {fontSize:"24px", left:300, width: "200px", opacity: 0.5} , 1000 )
```

o

```
$("p").animate({ height: 'toggle', opacity: 'toggle' }, "slow");
```

- "easing" (opcional): nombre del efecto personalizado que desea usar (se necesita el plugin).
- "función a la que se llama" (opcional): función que se tiene que ejecutar cuando termina la animación.

Ejemplo

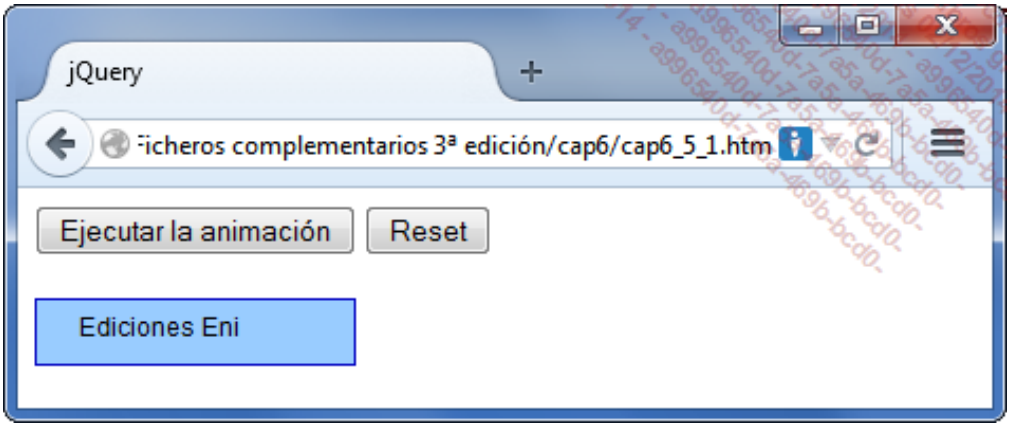
```
$("#boton").click(function() {
$("#contenido").animate({"height": "300px", "width": "250px"},
"slow", "linear", function(){
$(this).html("Animación terminada");
});
});
```

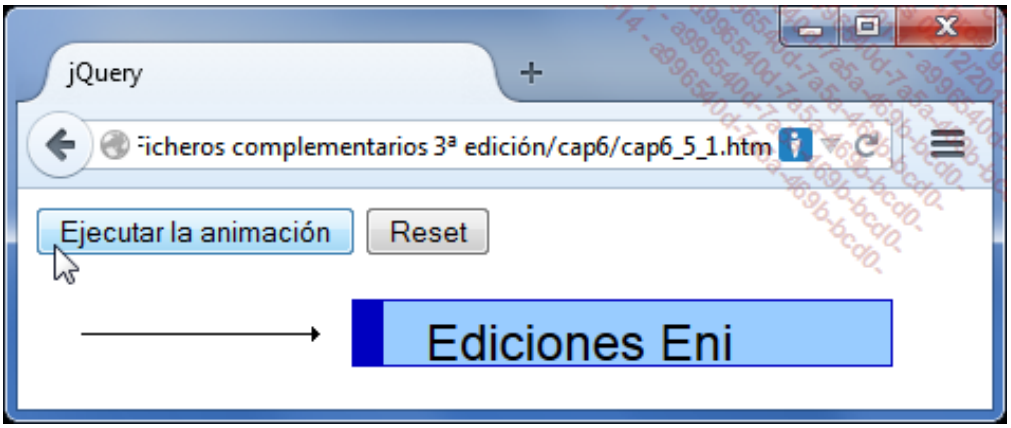
Este método devuelve un objeto jQuery.

Comentario

Todos los efectos jQuery, los del método animate() incluidos, se pueden desactivar usando la instrucción `jQuery.fx.off = true`, hecho que valorarán los usuarios que tengan problemas de visión.

1. Una animación en una capa






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
#box { position: relative;
       width: 110px;
       height: 25px;
       background-color: #9cf;
       margin: 20px 0px;
       padding-left: 20px;
       padding-right: 20px;
       padding-top: 5px;
       font: 0.75em Arial, sans-serif;
       border: 1px solid #0000c0;}
</style>
</head>
<body>
<input type="button" id="start" value="Ejecutar la animación">
<input type="button" id="reset" value="Reset">
<div id="box">Ediciones Eni</div>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$('#start').click(function(){
$('#box').animate({left:150,
          width: "200px",
          opacity: 0.5
```

```
          }, 1500)
.animate( { fontSize:"24px" } , 1000 )
.animate( { borderLeftWidth:"15px" }, 1000)
.animate( { opacity: 1 }, 1000);
$("#reset").click(function(){
$("#box").css({width:"",left:"",fontSize:"",opacity:"",borderWidth:""});
});
});
});
</script>
```

Detallamos esta animación.

```
$(document).ready(function(){
$('#start').click(function(){
```

Al hacer clic en el botón **Ejecutar la animación**.

```
$('#box').animate({left: 150,
          width: "200px",
          opacity: 0.5
          }, 1500)
```

La caja `box` se desplaza hacia la derecha 150 píxeles (`left: 150`), su anchura pasa a ser de 200 píxeles y su opacidad baja a la mitad. La animación dura 1.500 milisegundos.

```
.animate( { fontSize:"24px" } , 1000 )
```

Después, el tamaño de la letra se agranda.

```
.animate( { borderLeftWidth:"15px" }, 1000)
```

El grosor del borde izquierdo pasa a ser de 15 píxeles.

```
.animate( { opacity: 1 }, 1000);
```

Y, para terminar, la opacidad se establece en el valor 1.

```
$("#reset").click(function(){
$("#box").css({width:"",left:"",fontSize:"",opacity:"",borderWidth:""});
```

El clic en el botón **Reset** vuelve a poner la caja en su posición inicial, anulando todas las modificaciones que se han hecho en la anchura, la posición y la opacidad, así como en la anchura del borde izquierdo.

```
});
});
});
```

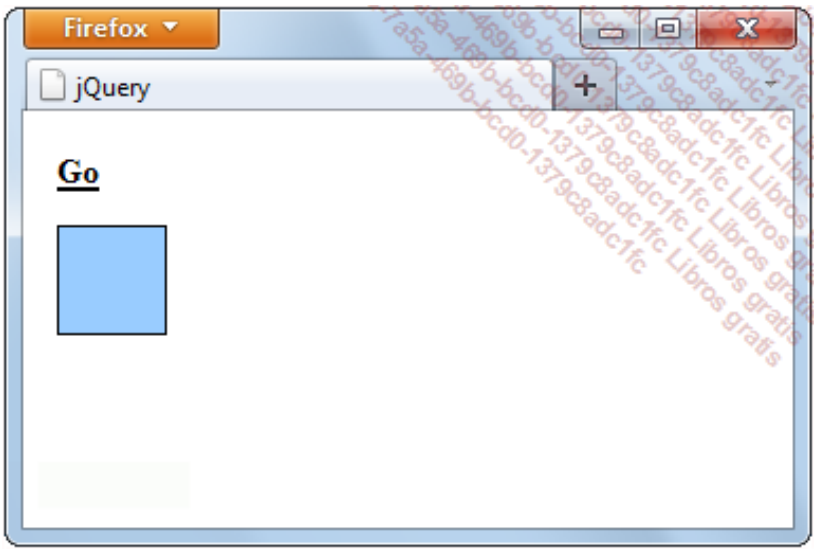
Fin del script.

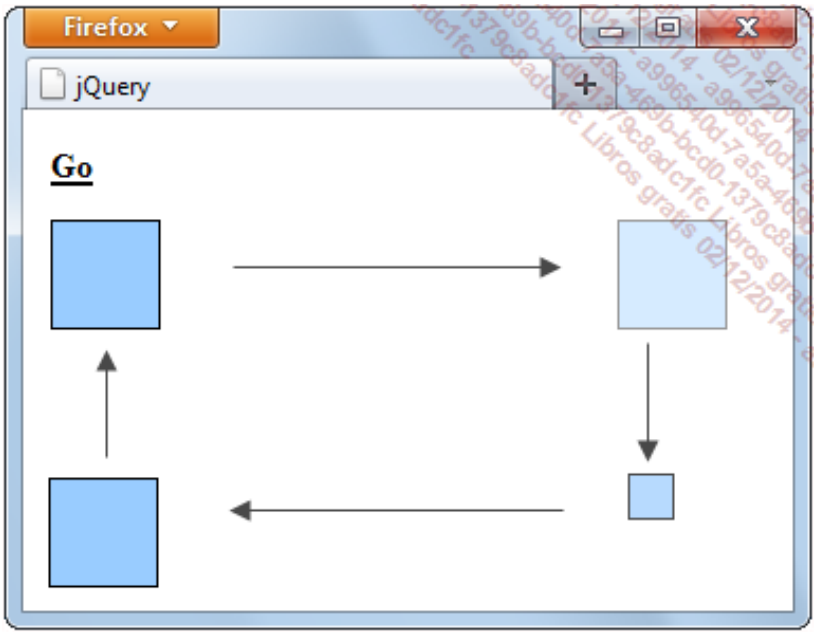
Por tanto, la página final es:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$('#start').click(function(){
```

```
$('#box').animate({left:150,
          width: "200px",
          opacity: 0.5
          }, 1500)
.animate( { fontSize:"24px" } , 1000 )
.animate( { borderLeftWidth:"15px" }, 1000)
.animate( { opacity: 1 }, 1000);
$("#reset").click(function(){
$("#box").css(
{width:"",left:"",fontSize:"",opacity:"",borderWidth:""});
});
});
});
</script>
<style>
#box { position: relative;
       width: 110px;
       height: 25px;
       background-color: #9cf;
       margin: 20px 0px;
       padding-left: 20px;
       padding-right: 20px;
       padding-top: 5px;
       font: 0.75em Arial, sans-serif;
       border: 1px solid #0000c0;}
</style>
</head>
<body>
<input type="button" id="start" value="Ejecutar la animación">
<input type="button" id="reset" value="Reset">
<div id="box">Ediciones Eni</div>
</body>
</html>
```

2. Una animación avanzada





El archivo Html inicial es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
a { color: black;
    font-weight: bold;
    font: Arial 10px;}
#box { position: relative;
       width: 50px;
       height: 50px;
       background: #9cf;
       border: 1px solid black;}
</style>
</head>
<body>
<p>
<a class="go" href="#">Go</a></p>
<div id="box"></div>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$(".go").click(function(){
$("#box").animate({left: "+=270", opacity: "0.4"}, 1200)
.animate({top: "+=120", opacity: "0.7", height: "20", width:
"20"}, "slow")
.animate({left: "0", opacity: "1", height: "50", width: "50"},
"slow")
.animate({top: "0"}, "fast")
.slideUp()
.slideDown()
return false;
});
});
</script>
```

Explicación del script:

```
$(document).ready(function(){
$(".go").click(function(){
```

Al hacer clic en el enlace.

```
$("#box").animate({left: "+=270", opacity: "0.4"}, 1200)
```

La caja (`box`) se desplaza hacia la derecha 270 píxeles (`left: "+=270`) y su opacidad disminuye a 0,4. Esta animación dura 1200 milisegundos.

```
.animate({top: "+=120", opacity: "0.7", height: "20", width: "20"},
"slow")
```

Después desciende 120 píxeles, su opacidad pasa a 0,7, su anchura se reduce a 20 píxeles, igual que su altura.

```
.animate({left: "0", opacity: "1", height: "50", width: "50"}, "slow")
```

Vuelve a su posición horizontal y aspecto iniciales (opacidad 1, altura y anchura de 50 píxeles).

```
.animate({top: "0"}, "fast")
```

La caja vuelve a su posición vertical inicial.

```
.slideUp()
.slideDown()
return false;
```

Una animación final desplaza la caja hacia arriba para descender después.

```
});
});
```

Fin del script.

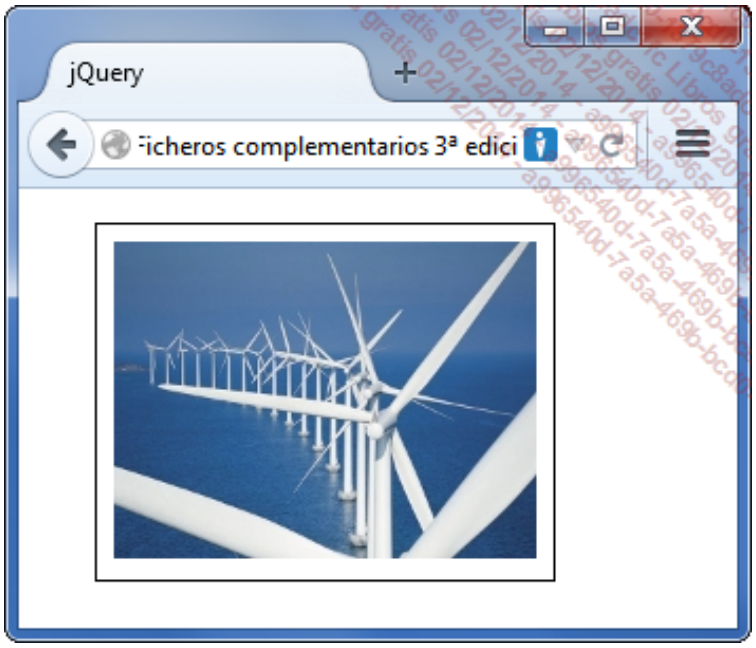
La página Html final es:

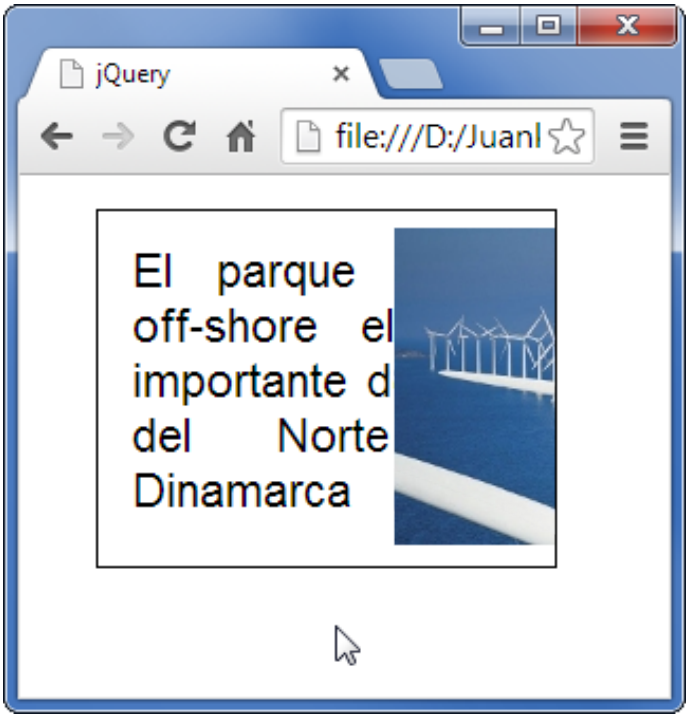
```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$(".go").click(function(){
$("#box").animate({opacity: "0.4", left: "+=270"}, 1200)
.animate({opacity: "0.7", top: "+=120", height: "20", width:
"20"}, "slow")
.animate({opacity: "1", left: "0", height: "50", width: "50"},
"slow")
.animate({top: "0"}, "fast")
.slideUp()
.slideDown()
return false;
});
});
</script>
<style>
```

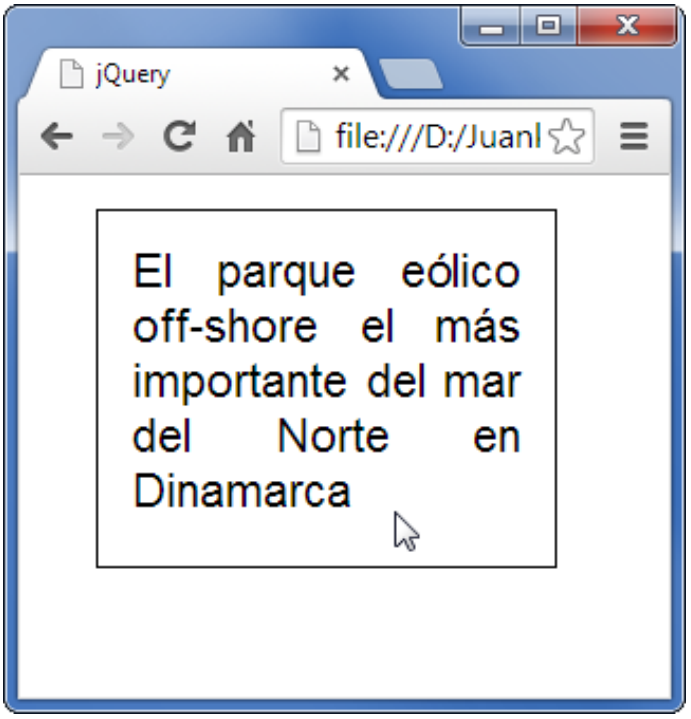
```
a { color: black;
    font-weight: bold;
    font: Arial 10px;}
#box { position: relative;
       width: 50px;
       height: 50px;
       background: #9cf;
       border: 1px solid black;}
</style>
</head>
<body>
<p>
<a class="go" href="#">Go</a></p>
<div id="box"></div>
</body>
</html>
```

3. Un efecto original al pasar el ratón por encima

Al pasar el ratón por encima de la imagen, ésta se desplaza hacia la derecha para que aparezca la leyenda.







Inicialmente:

```
doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
ul{margin-left:-12px;}
ul.hover_bloc li { position: relative;
                   width: 184px;
                   height: 136px;
                   padding: 16px;
                   list-style-type: none;
                   font: 1.4em Arial, sans-serif;
                   text-align: justify;
```

```
                    color: black;
                    overflow: hidden;
                    border: 1px solid black;}
ul.hover_bloc li img { position: absolute;
                        top: 8px;
                        left: 8px;
                        border: 0px;}
</style>
</head>
<body>
<ul class="hover_bloc">
<li>
<img alt="" src="parque_eolico.jpg">
Parque eólico off-shore más importante del mar del Norte de
Dinamarca
</li>
</ul>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$('ul.hover_bloc li').hover(function(){
$(this).find('img').animate({left:'300px'},{duration:500});
}, function(){
$(this).find('img').animate({left:'8px'},{duration:500});
});
});
</script>
```

Lo detallamos a continuación.

```
$(document).ready(function(){
$('ul.hover_bloc li').hover(function(){
```

Al pasar el ratón por encima de la zona.

```
$(this).find('img').animate({left:'300px'},{duration:500});
```

El script encuentra la imagen (`find('img')`) y la desplaza hacia la derecha 300 píxeles (`left:'300px'`). La animación dura 500 milisegundos.

```
}, function(){
$(this).find('img').animate({left:'8px'},{duration:500});
```

Cuando el cursor sale de la zona, la imagen vuelve a su posición inicial.

```
});
});
```

Fin del script.

Es admirable la concisión del código generado por jQuery.

Por tanto, la página final es:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
```

```
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$('ul.hover_bloc li').hover(function(){
$(this).find('img').animate({left:'300px'},{duration:500});
}, function(){
$(this).find('img').animate({left:'8px'},{duration:500});
});
});
</script>
<style>
ul { margin-left: -12px;}
ul.hover_bloc li { position: relative;
                   width: 184px;
                   height: 136px;
                   padding: 16px;
                   list-style-type: none;
                   font: 1.4em Arial, sans-serif;
                   text-align: justify;
                   color: black;
                   overflow: hidden;
                   border: 1px solid black;}
ul.hover_bloc li img { position: absolute;
                       top: 8px;
                       left: 8px;
                       border: 0px;}
</style>
</head>
<body>
<ul class="hover_bloc">
<li>
<img alt="" src="parque_eolico.jpg">
Parque eólico off-shore más importante del mar del Norte de
Dinamarca
</li>
</ul>
</body>
</html>
```

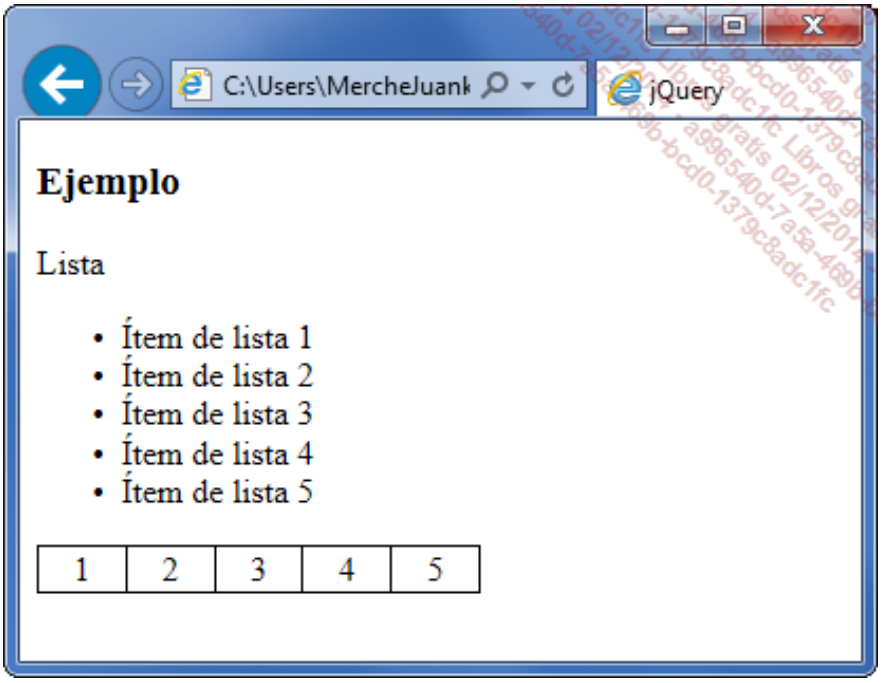
Introducción

El DOM, que permite al diseñador acceder a cada uno de los elementos de la página, ha relanzado el JavaScript. Pero hay que admitir que el acceso de padres a hijos y a otros hermanos no siempre es sencillo. En otras palabras, una simple modificación implica normalmente una reescritura completa del código. La librería de jQuery remedia en gran medida estos inconvenientes, gracias a sus numerosos selectores (véase el capítulo Los selectores en jQuery) y métodos específicos para recorrer y manejar los elementos del DOM.

Para este capítulo vamos a usar una página tipo. Ésta tiene una lista no ordenada con 5 ítems y una tabla con una fila y 5 columnas.

Observe también las capas `ejemplo` y `contenido`, ya que intervendrán más adelante en nuestro estudio.

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
table { border: 1px solid black;
        border-collapse:collapse;
        width: 210px;}
td { border: 1px solid black;
     text-align: center;}
.contenido { width: 210px;}
.borde { border: 1px solid black;}
.colorDefondo { background-color: #9cf;}
</style>
</head>
<body>
<div id="ejemplo">
<h3>Ejemplo</h3>
<div class="contenido">
<p>Lista</p>
<ul>
<li>Ítem de lista 1</li>
<li>Ítem de lista 2</li>
<li>Ítem de lista 3</li>
<li>Ítem de lista 4</li>
<li>Ítem de lista 5</li>
</ul>
<table>
<tr>
<td>1</td><td>2</td><td>3</td><td>4</td><td>5</td>
</tr>
</table>
</div>
</div>
</body>
</html>
```


C:\Users\MercheJuank
jQuery
Ejemplo
Lista
Ítem de lista 1
Ítem de lista 2
Ítem de lista 3
Ítem de lista 4
Ítem de lista 5
1
2
3
4
5

Encontrar los hijos

children()

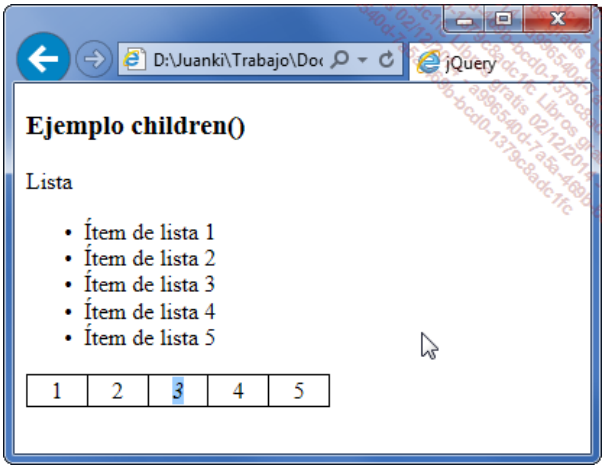
Devuelve un grupo de elementos que contienen los hijos inmediatos de cada uno de los elementos implicados en la selección.

```
$("div").children()
```

Este método devuelve un objeto jQuery.

Ejemplo:

Vamos a añadir un borde a los hijos de la capa `<div class="contenido">` y un fondo a los hijos de la celda 3 de la tabla.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$(".contenido").children().addClass("borde");
$("#select_table").children().addClass("colorDefondo");
});
</script>
<style>
table { border: 1px solid black;
        border-collapse:collapse;
        width: 210px;}
td { border: 1px solid black;
     text-align: center;}
.contenido { width: 210px;}
.borde { border: 1px solid black;}
.colorDefondo { background-color: #9cf;}
</style>
</head>
<body>
<div id="ejemplo">
<h3>Ejemplo children()</h3>
<div class="contenido">
<p>Lista</p>
<ul>
<li>Ítem de lista 1</li>
<li>Ítem de lista 2</li>
<li>Ítem de lista 3</li>
<li>Ítem de lista 4</li>
<li>Ítem de lista 5</li>
</ul>
<table>
<tr>
<td>1</td><td>2</td><td
id="select_table"><i>3</i></td><td>4</td><td>5</td>
</tr>
</table>
</div>
</div>
</body>
</html>
```

```
$(".contenido").children().addClass("borde");
```

El método `children()` se aplica a la capa con la clase `contenido` y dibuja un borde a sus hijos inmediatos, es decir, el párrafo `<p>Lista</p>` y la lista no ordenada `<ul> ... </ul>`.

```
$("#select_table").children().addClass("color
```

El método `children()` se aplica a la celda cuyo identificador es `select_table` y añade un fondo de color a sus hijos inmediatos, es decir, su contenido (la cifra 3).

Comentario

Es posible filtrar los elementos que devuelve jQuery usando una expresión opcional. En este caso, la forma del método `children()` es `children(expresión)`, lo que permite recuperar solo los elementos hijos que responden a la expresión que se ha introducido.

Ejemplo

Vemos el extracto de código siguiente:

```
<ul>
<li class="selected">Ítem de lista 1</li>
<li>Ítem de lista 2</li>
<li class="selected">Ítem de lista 3</li>
```

Es posible quedarse, entre los hijos del elemento `<ul>`, solo con los elementos con la clase`selected`.

El código jQuery es:

```
<li>Ítem de lista 4</li>
<li class="selected">Ítem de lista 5</li>
</ul>
```

```
$("ul").children(.selected);
```

Encontrar los padres directos

parent()

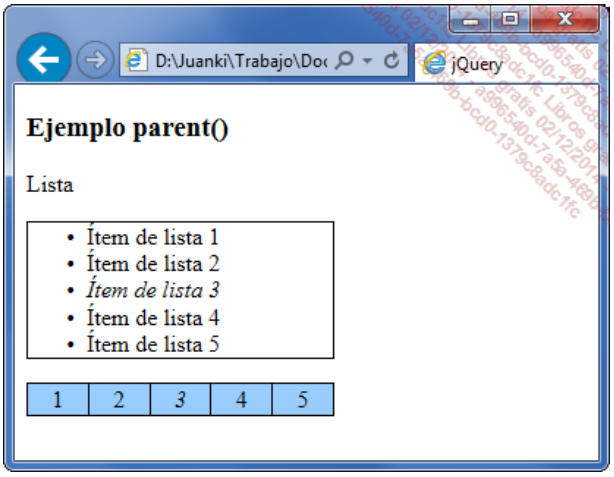
Devuelve un grupo de elementos que contienen los padres inmediatos de cada uno de los elementos implicados por la selección.

```
$("span").parent()
```

Este método devuelve un objeto jQuery.

Ejemplo

Rodeamos con un borde los padres inmediatos del tercer elemento de la lista y añadimos un color de fondo a los padres inmediatos de la tercera celda de la tabla.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#select_li").parent().addClass("borde");
$("#select_table").parent().addClass("colorDefondo");
});
</script>
<style>
table { border: 1px solid black;
        border-collapse:collapse;
        width: 210px;}
td { border: 1px solid black;
     text-align: center;}
.contenido { width: 210px;}
.borde { border: 1px solid black;}
.colorDefondo { background-color: #9cf;}
</style>
</head>
<body>
<div id="ejemplo">
<h3>Ejemplo parent()</h3>
<div class="contenido">
<p>Lista</p>
<ul>
<li>Ítem de lista 1</li>
<li>Ítem de lista 2</li>
<li id="select_li"><i>Ítem de lista 3</i></li>
<li>Ítem de lista 4</li>
<li>Ítem de lista 5</li>
</ul>
<table>
<tr>
<td>1</td><td>2</td><td
id="select_table"><i>3</i></td><td>4</td><td>5</td>
</tr>
</table>
</div>
</div>
</body>
</html>
```

```
$("#select_li").parent().addClass("borde");
```

El método `parent()` se aplica al tercer ítem de la lista (`id="select_li"`) y rodea con un borde a sus padres, es decir, solo la etiqueta `<ul>`.

```
$("#select_table").parent().addClass("color
```

El método `parent()` se aplica a la tercera celda de la tabla (`select_table`) y añade un fondo a sus padres directos, es decir, solo la etiqueta de tabla `<table>`.

Comentario

Con `parent(expresión)`, se pueden filtrar los elementos que se devuelven, para quedarse solo con aquellos que cumplan una expresión determinada.

Encontrar todos los padres

parents()

Devuelve un grupo de elementos que contiene todos los padres de cada uno de los elementos implicados por la selección.

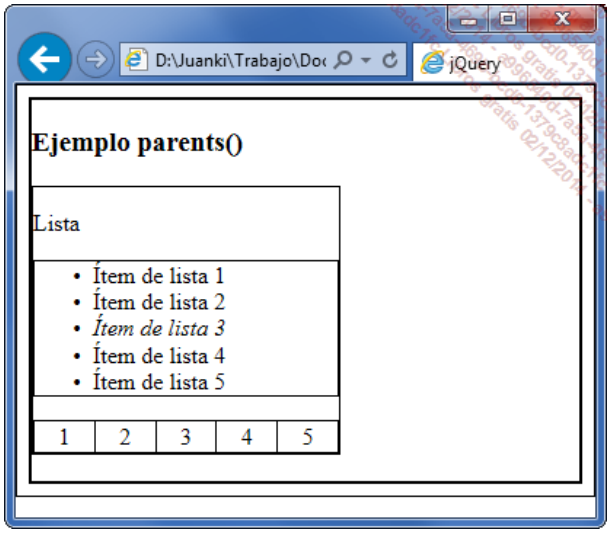
```
$("li").parents()
```

Este método devuelve un objeto jQuery.

El método `parents()` devuelve todos los ascendientes, mientras que `children()` solo tiene en cuenta los elementos hijos inmediatos.

Ejemplo

Rodeamos con un borde los padres del tercer elemento de la lista.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#select_li").parents().addClass("borde");
});
</script>
<style>
table { border: 1px solid black;
        border-collapse:collapse;
        width: 210px;}
td { border: 1px solid black;
     text-align: center;}
.contenido { width: 210px;}
.borde { border: 1px solid black;}
.colorDefondo { background-color: #9cf;}
</style>
</head>
<body>
<div id="ejemplo">
<h3>Ejemplo parents()</h3>
<div class="contenido">
<p>Lista</p>
<ul>
<li>Ítem de lista 1</li>
<li>Ítem de lista 2</li>
<li id="select_li"><i>Ítem de lista 3</i></li>
<li>Ítem de lista 4</li>
<li>Ítem de lista 5</li>
</ul>
<table>
<tr>
<td>1</td><td>2</td><td>3</td><td>4</td><td>5</td>
</tr>
</table>
</div>
<br />
</div>
</body>
</html>
```

```
$("#select_li").parents().addClass("borde");
```

El método `parents()` se aplica al tercer ítem de la lista (`id="select_li"`) y rodea con un borde todos sus padres, es decir, la etiqueta `<ul>`, la capa `contenido` y la etiqueta `<body>`.

Comentario

Con `parents(expresión)`, se pueden filtrar los elementos que se devuelven para quedarse solo con aquellos que cumplan una expresión determinada.

Encontrar los padres hasta

parentsUntil(selector o elemento)

Devuelve los padres de cada elemento hasta (no incluido) el elemento que se especifica en el selector.

```
$('div.box').parentsUntil('body');
```

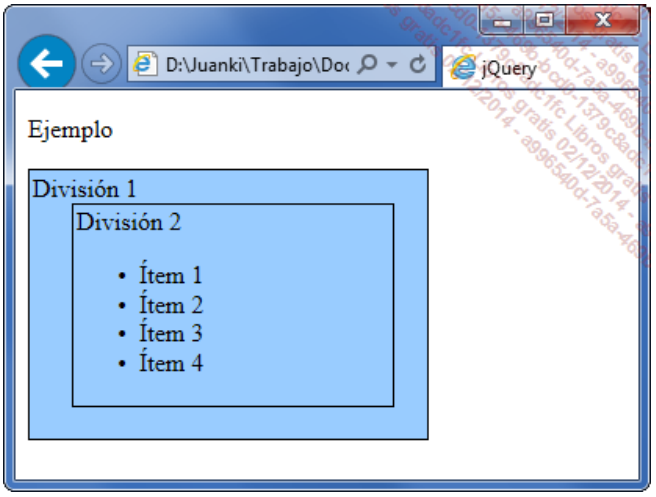
Todos los elementos padres de la capa con la clase box hasta la etiqueta `<body>`.

Este método devuelve un objeto jQuery.

Este método es una novedad de la versión 1.4 de jQuery.

Ejemplo

Supongamos que tenemos una capa con una lista de cuatro ítems (División 2). Esta capa está dentro de una capa (División 1). Añadimos un color de fondo a los padres del ítem 3 de la lista hasta la etiqueta `<body>`.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
#division1 { width: 250px;
             border: 1px solid black;
             padding: 2px;}
#division2 { width: 200px;
             border: 1px solid black;
             margin-left: 25px;
             padding: 2px;}
</style>
<script src="jquery.js"></script>
<script>
$(document).ready(function() {
$('li.start').parentsUntil('body')
.css('background-color', '#9cf');
});
</script>
</head>
<body>
<p id="párrafo">Ejemplo</p>
<div id="division1">
División 1
<div id="division2">
División 2
<ul>
<li>Ítem 1</li>
<li>Ítem 2</li>
<li class="start">Ítem 3</li>
<li>Ítem 4</li>
</ul>
</div>
<br>
</div>
</body>
</html>
```

`$('li.start').parentsUntil('body')`

Al inicio del ítem de lista con la clase start (`$('li.start')`), recuperar todos los padres de éste hasta (`parentsUntil`) la etiqueta `<body>` no incluida (`parentsUntil('body')`).

El párrafo `<p>` no se recupera, ya que no forma parte de los padres del elemento de inicio.

Encontrar los hermanos

siblings()

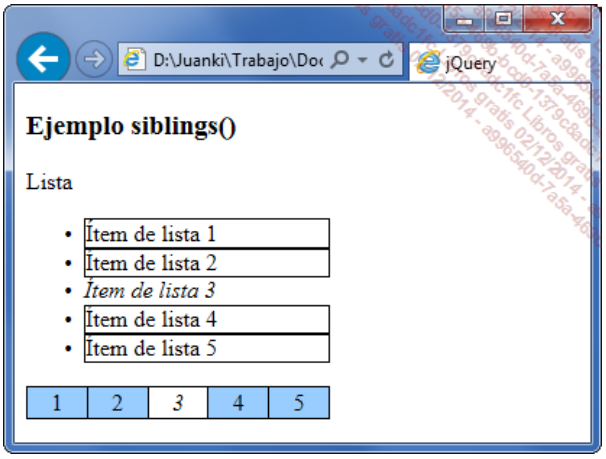
Devuelve la lista de los hermanos inmediatos de cada elemento de la selección.

```
$("div").siblings()
```

Este método devuelve un objeto jQuery.

Ejemplo

Rodeamos con un borde los hermanos del tercer elemento de la lista y añadimos un color de fondo a los hermanos de la tercera celda de la tabla.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#select_li").siblings().addClass("borde");
$("#select_table").siblings().addClass("colorDefondo");
});
</script>
<style>
table { border: 1px solid black;
        border-collapse:collapse;
        width: 210px;}
td { border: 1px solid black;
     text-align: center;}
.contenido { width: 210px;}
.borde { border: 1px solid black;}
.colorDefondo { background-color: #9cf;}
</style>
</head>
<body>
<div id="ejemplo">
<h3>Ejemplo siblings()</h3>
<div class="contenido">
<p>Lista</p>
<ul>
<li>Ítem de lista 1</li>
<li>Ítem de lista 2</li>
<li id="select_li"><i>Ítem de lista 3</i></li>
<li>Ítem de lista 4</li>
<li>Ítem de lista 5</li>
</ul>
<table>
<tr>
<td>1</td><td>2</td><td
id="select_table"><i>3</i></td><td>4</td><td>5</td>
</tr>
</table>
</div>
</div>
</body>
</html>
```

```
$("#select_li").siblings().addClass("borde");
```

El método `siblings()` se aplica al tercer ítem de la lista (`id="select_li"`) y rodea con un borde sus hermanos inmediatos, es decir, las otras etiquetas `<li>`.

```
$("#select_table").siblings().addClass("color
```

El método `siblings()` se aplica a la tercera celda de la tabla (`select_table`) y añade un fondo a sus hermanos inmediatos, es decir, las otras etiquetas `<td>`.

Comentario

Con `siblings(expresión)`, se pueden filtrar los elementos que se devuelven para quedarse solo con aquellos que cumplan una expresión determinada.

Encontrar el hermano anterior

prev()

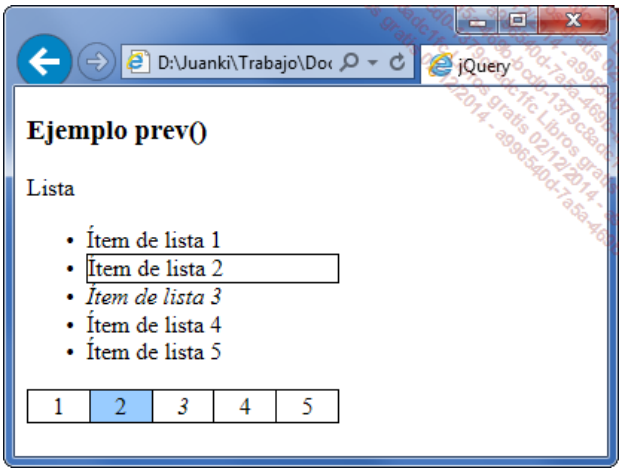
Devuelve el hermano inmediato anterior de cada elemento de la selección.

```
$("td").prev()
```

Este método devuelve un objeto jQuery.

Ejemplo

Rodeamos con un borde el hermano anterior del tercer elemento de la lista y añadimos un color de fondo al hermano anterior de la tercera celda de la tabla.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#select_li").prev().addClass("borde");
$("#select_table").prev().addClass("colorDefondo");
});
</script>
<style>
table { border: 1px solid black;
        border-collapse:collapse;
        width: 210px;}
td { border: 1px solid black;
     text-align: center;}
.contenido { width: 210px;}
.borde { border: 1px solid black;}
.colorDefondo { background-color: #9cf;}
</style>
</head>
<body>
<div id="ejemplo">
<h3>Ejemplo prev()</h3>
<div class="contenido">
<p>Lista</p>
<ul>
<li>Ítem de lista 1</li>
<li>Ítem de lista 2</li>
<li id="select_li"><i>Ítem de lista 3</i></li>
<li>Ítem de lista 4</li>
<li>Ítem de lista 5</li>
</ul>
<table>
<tr>
<td>1</td><td>2</td><td
id="select_table"><i>3</i></td><td>4</td><td>5</td>
</tr>
</table>
</div>
</div>
</body>
</html>
```

`$("#select_li").prev().addClass("borde");`

El método `prev()` se aplica al tercer ítem de la lista (`id="select_li"`) y rodea con un borde su hermano inmediatamente anterior, es decir, la etiqueta `<li>Ítem de lista 2</li>`.

`$("#select_table").prev().addClass("color`

El método `prev()` se aplica a la tercera celda de la tabla (`select_table`) y añade un fondo a su hermano inmediatamente anterior, es decir, la etiqueta `<td>2</td>`.

Comentario

Con `prev(expresión)`, se pueden filtrar los elementos que se devuelven para quedarse solo con aquellos que cumplan una expresión determinada.

Encontrar los hermanos anteriores

prevAll()

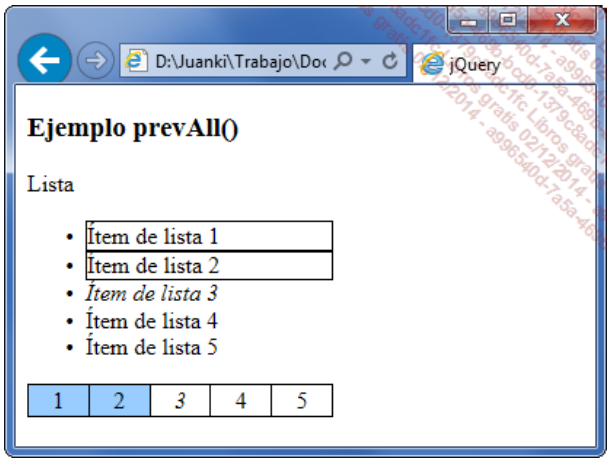
Devuelve los hermanos inmediatos anteriores de cada elemento de la selección.

```
$("td").prevAll()
```

Este método devuelve un objeto jQuery.

Ejemplo

Rodeamos con un borde el hermano anterior del tercer elemento de la lista y añadimos un color de fondo al hermano anterior de la tercera celda de la tabla.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#select_li").prevAll().addClass("borde");
$("#select_table").prevAll().addClass("colorDefondo");
});
</script>
<style>
table { border: 1px solid black;
        border-collapse:collapse;
        width: 210px;}
td { border: 1px solid black;
     text-align: center;}
.contenido { width: 210px;}
.borde { border: 1px solid black;}
.colorDefondo { background-color: #9cf;}
</style>
</head>
<body>
<div id="ejemplo">
<h3>Ejemplo prevAll()</h3>
<div class="contenido">
<p>Lista</p>
<ul>
<li>Ítem de lista 1</li>
<li>Ítem de lista 2</li>
<li id="select_li"><i>Ítem de lista 3</i></li>
<li>Ítem de lista 4</li>
<li>Ítem de lista 5</li>
</ul>
<table>
<tr>
<td>1</td><td>2</td><td
id="select_table"><i>3</i></td><td>4</td><td>5</td>
</tr>
</table>
</div>
</div>
</body>
</html>
```

```
$("#select_li").prevAll().addClass("borde");
```

El método `prevAll()` se aplica al tercer ítem de la lista (`id="select_li"`) y rodea con un borde sus hermanos anteriores, es decir, las etiquetas `<li>Ítem de lista 1</li>` y `<li>Ítem de lista 2</li>`.

```
$("#select_table").prevAll().addClass("color
```

El método `prevAll()` se aplica a la tercera celda de la tabla (`select_table`) y añade un fondo a sus hermanos anteriores, es decir, las etiquetas `<td>1</td>` y `<td>2</td>`.

Comentario

Con `prevAll(expresión)`, se pueden filtrar los elementos que se devuelven para quedarse solo con aquellos que cumplan una expresión determinada.

Encontrar los hermanos anteriores hasta

prevUntil(selector o elemento)

Devuelve los hermanos anteriores de cada elemento hasta (no incluido) el elemento que especifica el selector.

```
$('p').prevUntil('h1').
```

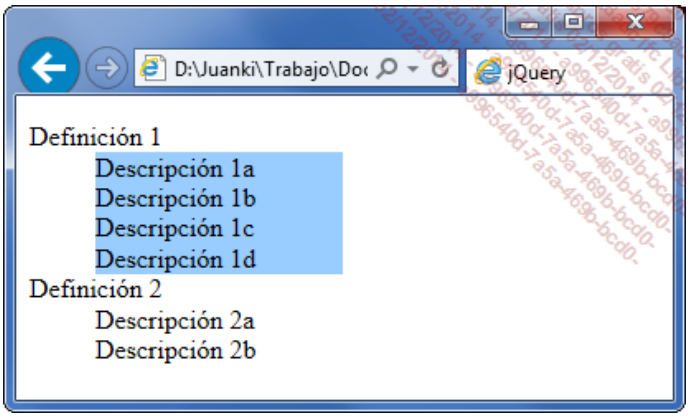
Devuelve los párrafos anteriores hasta la etiqueta <h1>.

Este método devuelve un objeto jQuery.

Se trata de una novedad de la versión 1.4 de jQuery.

Ejemplo

Supongamos una lista de definición (etiqueta `<dl>`), que tiene dos definiciones (etiqueta `<dt>`). Añadimos un color de fondo a los elementos anteriores a la segunda definición hasta la primera definición no incluida.




```
$("#def2").prevUntil("dt")
```

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
dd { width: 150px;}
</style>
<script src="jquery.js"></script>
<script>
$(document).ready(function() {
$("#def2").prevUntil("dt")
.css("background-color", "#9cf")
});
</script>
</head>
<body>
<dl>
<dt>Definición 1</dt>
<dd>Descripción 1a</dd>
<dd>Descripción 1b</dd>
<dd>Descripción 1c</dd>
<dd>Descripción 1d</dd>
<dt id="def2">Definición 2</dt>
<dd>Descripción 2a</dd>
<dd>Descripción 2b</dd>
</dl>
</body>
</html>
```

Al inicio del elemento identificado por id="def2" (`$("#def2")`), recuperar todos los elementos hermanos hasta (`prevUntil`) la etiqueta `<dt>` (`prevUntil("dt")`) no incluida.

Encontrar el hermano siguiente

next()

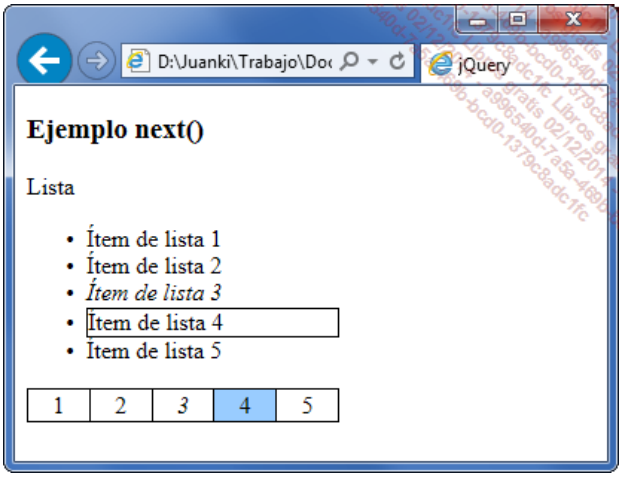
Devuelve el hermano inmediato siguiente de cada elemento de la selección.

```
$("td").prev()
```

Este método devuelve un objeto jQuery.

Ejemplo

Rodeamos con un borde el hermano siguiente del tercer elemento de la lista y añadimos un color de fondo al hermano siguiente de la tercera celda de la tabla.




```
doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#select_li").next().addClass("borde");
$("#select_table").next().addClass("colorDefondo");
});
</script>
<style>
table { border: 1px solid black;
        border-collapse:collapse;
        width: 210px;}
td { border: 1px solid black;
     text-align: center;}
.contenido { width: 210px;}
.borde { border: 1px solid black;}
.colorDefondo { background-color: #9cf;}
</style>
</head>
<body>
<div id="ejemplo">
<h3>Ejemplo next()</h3>
<div class="contenido">
<p>Lista</p>
<ul>
<li>Ítem de lista 1</li>
<li>Ítem de lista 2</li>
<li id="select_li"><i>Ítem de lista 3</i></li>
<li>Ítem de lista 4</li>
<li>Ítem de lista 5</li>
</ul>
<table>
<tr>
<td>1</td><td>2</td><td
id="select_table"><i>3</i></td><td>4</td><td>5</td>
</tr>
</table>
</div>
</div>
</body>
</html>
```

`$("#select_li").next().addClass("borde");`

El método `next()` se aplica al tercer ítem de la lista (`id="select_li"`) y rodea con un borde su hermano inmediatamente siguiente, es decir, la etiqueta `<li>Ítem de lista 4</li>`.

`$("#select_table").next().addClass("color`

El método `next()` se aplica a la tercera celda de la tabla (`select_table`) y añade un fondo a su hermano inmediatamente siguiente, es decir, la etiqueta `<td>4</td>`.

Comentario

Con `next(expresión)`, se pueden filtrar los elementos que se devuelven para quedarse solo con aquellos que cumplan una expresión determinada.

Encontrar los hermanos siguientes

nextAll()

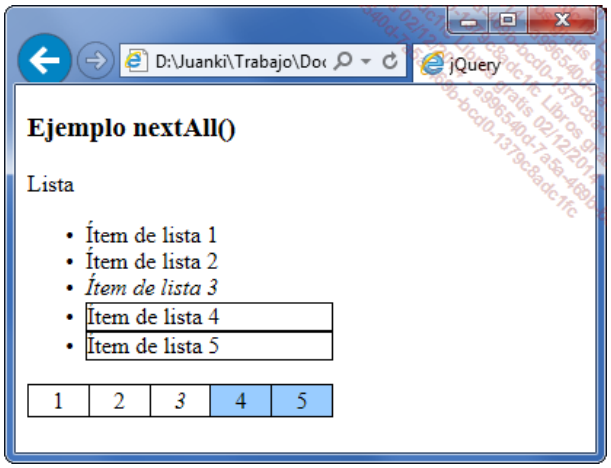
Devuelve los hermanos inmediatos siguientes de cada elemento de la selección.

```
$("td").nextAll()
```

Este método devuelve un objeto jQuery.

Ejemplo

Rodeamos con un borde los hermanos inmediatamente siguientes del tercer elemento de la lista y añadimos un color de fondo a los hermanos inmediatamente siguientes de la tercera celda de la tabla.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#select_li").nextAll().addClass("borde");
$("#select_table").nextAll().addClass("colorDefondo");
});
</script>
<style>
table { border: 1px solid black;
        border-collapse:collapse;
        width: 210px;}
td { border: 1px solid black;
     text-align: center;}
.contenido { width: 210px;}
.borde { border: 1px solid black;}
.colorDefondo { background-color: #9cf;}
</style>
</head>
<body>
<div id="ejemplo">
<h3>Ejemplo nextAll()</h3>
<div class="contenido">
<p>Lista</p>
<ul>
<li>Ítem de lista 1</li>
<li>Ítem de lista 2</li>
<li id="select_li"><i>Ítem de lista 3</i></li>
<li>Ítem de lista 4</li>
<li>Ítem de lista 5</li>
</ul>
<table>
<tr>
<td>1</td><td>2</td><td
id="select_table"><i>3</i></td><td>4</td><td>5</td>
</tr>
</table>
</div>
</div>
</body>
</html>
```

```
$("#select_li").nextAll().addClass("borde");
```

El método `nextAll()` se aplica al tercer ítem de la lista (`id="select_li"`) y rodea con un borde sus hermanos siguientes, es decir, las etiquetas `<li>Ítem de lista 4</li>` y `<li>Ítem de lista 5</li>`.

```
$("#select_table").nextAll().addClass("color
```

El método `nextAll()` se aplica a la tercera celda de la tabla (`select_table`) y añade un fondo a sus hermanos siguientes, es decir, las etiquetas `<td>4</td>` y `<td>5</td>`.

Comentario

Con `nextAll(expresión)`, se pueden filtrar los elementos que se devuelven para quedarse solo con aquellos que cumplan una expresión determinada.

Encontrar los hermanos siguientes hasta

nextUntil(selector o elemento)

Devuelve los hermanos siguientes de cada elemento hasta (no incluido) el elemento que especifica el selector.

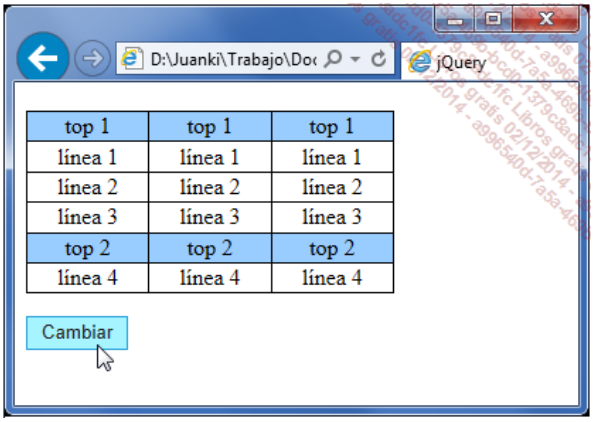
```
$("ul >:first").nextUntil( ":last" );
```

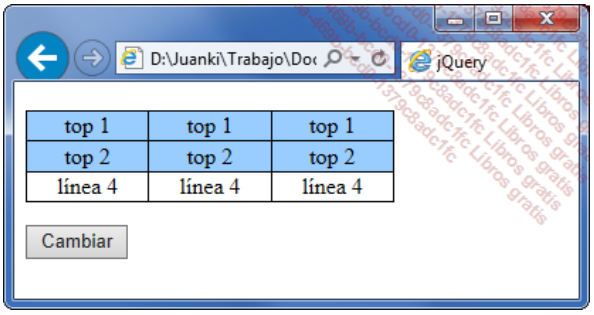
Devuelve todos los ítems de lista comprendidos entre el primero y el último.

Se trata de una novedad de la versión 1.4 de jQuery.

Ejemplo

Hacemos desaparecer y aparecer filas de la tabla según un determinado criterio de selección.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
table { margin-top: 20px;
        width: 260px;
        border: 1px solid black;
        border-collapse:collapse;}
td { border: 1px solid black;
     text-align: center;}
.top { background-color: #9cf;}
</style>
<script src="jquery.js"></script>
<script>
$(document).ready(function() {
$("button").click(function() {
$("tr:first").nextUntil('tr:not(.linea)').toggle();
});
});
</script>
</head>
<body>
<table>
<tr class="top">
<td>top 1</td>
<td>top 1</td>
<td>top 1</td>
</tr>
<tr class="linea">
<td>línea 1</td>
<td>línea 1</td>
<td>línea 1</td>
</tr>
<tr class="linea">
<td>línea 2</td>
<td>línea 2</td>
<td>línea 2</td>
</tr>
<tr class="linea">
<td>línea 3</td>
<td>línea 3</td>
```

```
$("tr:first").nextUntil("tr:not(.bt)").toggle();
```

Empezando por la primera fila de la tabla (`$("tr:first")`), seleccionar las filas hasta (`nextUntil`) la fila, no incluida, con una clase diferente a `linea` (`nextUntil("tr:not(.linea)")`) y hacerlas desaparecer o aparecer (`toggle()`).

Observe que la fila 4, que no pertenece al criterio de selección, permanece visible.

```
<td>linea 3</td>
</tr>
<tr class="top">
<td>top 2</td>
<td>top 2</td>
<td>top 2</td>
</tr>
<tr class="linea">
<td>linea 4</td>
<td>linea 4</td>
<td>linea 4</td>
</tr>
</table>
<p><button type="button">Cambiar</button></p>
</body>
</html>
```

Encontrar el contenido

contents()

Encuentra todos los nodos hijos situados en los elementos de la selección (incluyendo los nodos de texto).

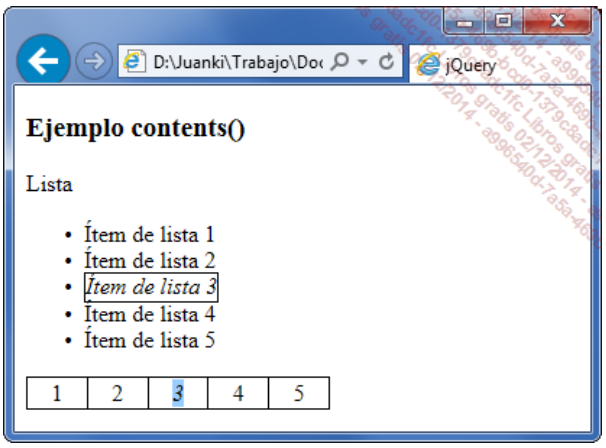
Si el elemento especificado es una etiqueta `<iframe>`, `contents()` encuentra el contenido del documento.

```
$("p").contents()
```

Este método devuelve un objeto jQuery.

Ejemplo:

Rodeamos con un borde el contenido del tercer elemento de la lista y añadimos un color de fondo al contenido de la tercera celda de la tabla.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#select_li").contents().addClass("borde");
$("#select_table").contents().addClass("colorDefondo");
});
</script>
<style>
table { border: 1px solid black;
        border-collapse:collapse;
        width: 210px;}
td { border: 1px solid black;
     text-align: center;}
.contenido { width: 210px;}
.borde { border: 1px solid black;}
.colorDefondo { background-color: #9cf;}
</style>
</head>
<body>
<div id="ejemplo">
<h3>Ejemplo contents()</h3>
<div class="contenido">
<p>Lista</p>
<ul>
<li>Ítem de lista 1</li>
<li>Ítem de lista 2</li>
<li id="select_li"><i>Ítem de lista 3</i></li>
<li>Ítem de lista 4</li>
<li>Ítem de lista 5</li>
</ul>
<table>
<tr>
<td>1</td><td>2</td><td
id="select_table"><i>3</i></td><td>4</td><td>5</td>
</tr>
</table>
</div>
</div>
</body>
</html>
```

```
$("#select_li").contents().addClass("borde");
```

El método `contents()` se aplica al tercer ítem de la lista (`id="select_li"`) y rodea con un borde su nodo de texto hijo, es decir, las palabras "`Ítem de lista 3`".

```
$("#select_table").contents().addClass("color
```

El método `contents()` se aplica a la tercera celda de la tabla (`select_table`) y añade un fondo a su nodo de texto hijo, es decir, la cifra 3.

Comentario

Con `contents(expresión)`, se pueden filtrar los elementos que se devuelven para quedarse solo con aquellos que cumplan una expresión determinada.

Encontrar determinados padres

closest(selector)

Devuelve el conjunto de elementos que contiene el padre más cercano del elemento seleccionado que responde al selector, incluido el elemento de inicio.

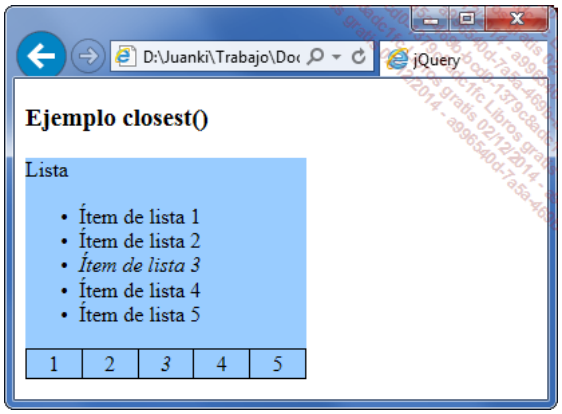
En primer lugar, el método `closest()` comprueba si el elemento actual responde a la expresión especificada. En caso afirmativo, devuelve simplemente el elemento especificado. En caso contrario, continúa recorriendo el documento hacia arriba, padre a padre, hasta que encuentre un elemento que responda a la condición de la expresión. Si no se encuentra ningún elemento, el método no devuelve nada.

```
$("div").closest("p")
```

Este método devuelve un objeto jQuery.

Ejemplo

Buscamos los padres del tercer elemento de la lista hasta la capa con la clase contenido y añadimos un color de fondo.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#select_li").closest(".contenido").addClass("colorDefondo");
});
</script>
<style>
table { border: 1px solid black;
        border-collapse:collapse;
        width: 210px;}
td { border: 1px solid black;
     text-align: center;}
.contenido { width: 210px;}
.borde { border: 1px solid black;}
.colorDefondo { background-color: #9cf;}
</style>
</head>
<body>
<div id="ejemplo">
<h3>Ejemplo closest()</h3>
<div class="contenido">
<p>Lista</p>
<ul>
<li>Ítem de lista 1</li>
<li>Ítem de lista 2</li>
<li id="select_li"><i>Ítem de lista 3</i></li>
<li>Ítem de lista 4</li>
<li>Ítem de lista 5</li>
</ul>
<table>
<tr>
<td>1</td><td>2</td><td
id="select_table"><i>3</i></td><td>4</td><td>5</td>
</tr>
</table>
</div>
</div>
</body>
</html>
```

```
$("#select_li").closest(".contenido").addClass("colorDefo
```

Con el método `closest()`, vamos a recorrer el DOM hasta la capa con la clase `contenido`.

Es posible hacer otras operaciones:

```
$("#select_li").closest("#ejemplo").addClass("colorDefonc
```

o

```
$("#select_li").closest("ul").addClass("colorDefondo");
```

Comentario:

Con `closest(expresión)`, se pueden filtrar los elementos que se devuelven para quedarse solo con aquellos que cumplan una expresión determinada.

Encontrar determinados descendientes

find(expresión o selector)

Busca los elementos descendientes que responden a las condiciones del selector especificado.

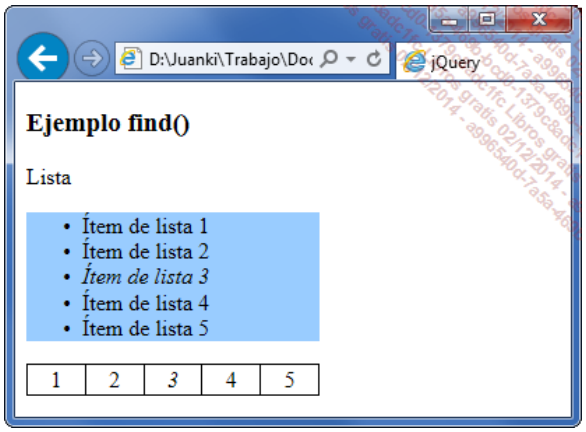
```
$("div").find("p")
```

Este método devuelve un objeto jQuery.

Este método `find()` no busca el elemento especificado, sino sus descendientes.

Ejemplo:

Encuentre la etiqueta no ordenada `<ul>` que es un descendiente de la capa identificada por `ejemplo`.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#ejemplo").find("ul").addClass("colorDefondo");
});
</script>
<style>
table { border: 1px solid black;
        border-collapse:collapse;
        width: 210px;}
td { border: 1px solid black;
     text-align: center;}
.contenido { width: 210px;}
.borde { border: 1px solid black;}
.colorDefondo { background-color: #9cf;}
</style>
</head>
<body>
<div id="ejemplo">
<h3>Ejemplo find()</h3>
<div class="contenido">
<p>Lista</p>
<ul>
<li>Ítem de lista 1</li>
<li>Ítem de lista 2</li>
<li id="select_li"><i>Ítem de lista 3</i></li>
<li>Ítem de lista 4</li>
<li>Ítem de lista 5</li>
</ul>
<table>
<tr>
<td>1</td><td>2</td><td
id="select_table"><i>3</i></td><td>4</td><td>5</td>
</tr>
</table>
</div>
</div>
</body>
</html>
```

```
$("#ejemplo").find("ul").addClass("colorDefondo");
```

Empezando por la capa `id="ejemplo"`, encontramos entre los descendientes una lista no ordenada (`<ul>`) y se le asigna un color de fondo.

Es posible hacer otras operaciones:

```
$("#ejemplo").find("p").addClass("colorDefondo");
```

o

```
$("ul").find("#select_li").addClass("colorDefondo"
```

Añadir elementos a la selección

add(selector o elemento(s) o Html)

Añade los elementos que se especifica en el argumento al conjunto de los elementos seleccionados en la búsqueda.

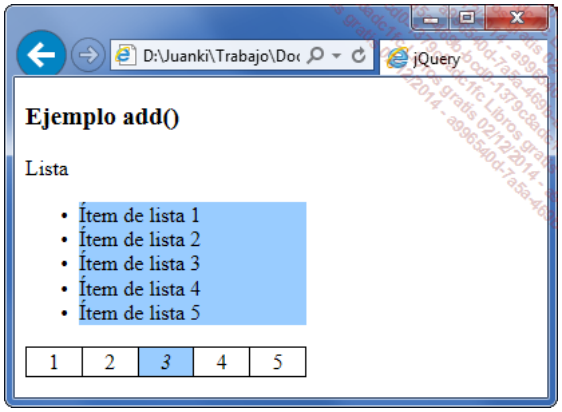
Los parámetros pueden ser:

- un selector jQuery: `$("p").add("span");`
- uno o varios elementos: `$("p").add(document.getElementById("a"));`
- código Html: `$("p").add("<span>Again</span>").`

Este método devuelve un objeto jQuery.

Ejemplo

Después de conseguir los elementos de la lista, añadimos a la selección la celda de la tabla identificada por el identificador `select_table`.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("ul").children().add("#select_table").addClass("colorDefondo");
});
</script>
<style>
table { border: 1px solid black;
        border-collapse:collapse;}
td { border: 1px solid black;
     text-align: center;
     width: 210px;}
.contenido { width: 210px;}
.borde { border: 1px solid black;}
.colorDefondo { background-color: #9cf;}
</style>
</head>
<body>
<div id="ejemplo">
<h3>Ejemplo add()</h3>
<div class="contenido">
<p>Lista</p>
<ul>
<li>Ítem de lista 1</li>
<li>Ítem de lista 2</li>
<li>Ítem de lista 3</li>
<li>Ítem de lista 4</li>
<li>Ítem de lista 5</li>
</ul>
<table>
<tr>
<td>1</td><td>2</td><td
id="select_table"><i>3</i></td><td>4</td><td>5</td>
</tr>
</table>
</div>
</div>
</body>
</html>
```

```
$(document).ready(function(){
$("ul").children().add("#select_table").addClass("colorDe
});
```

Usando `$("ul").children()`, se seleccionan los elementos de la lista. El código añade la celda de la tabla con `add("#select_table")`. A todos estos elementos se les cambia el color de fondo (`addClass("color-Defondo")`).

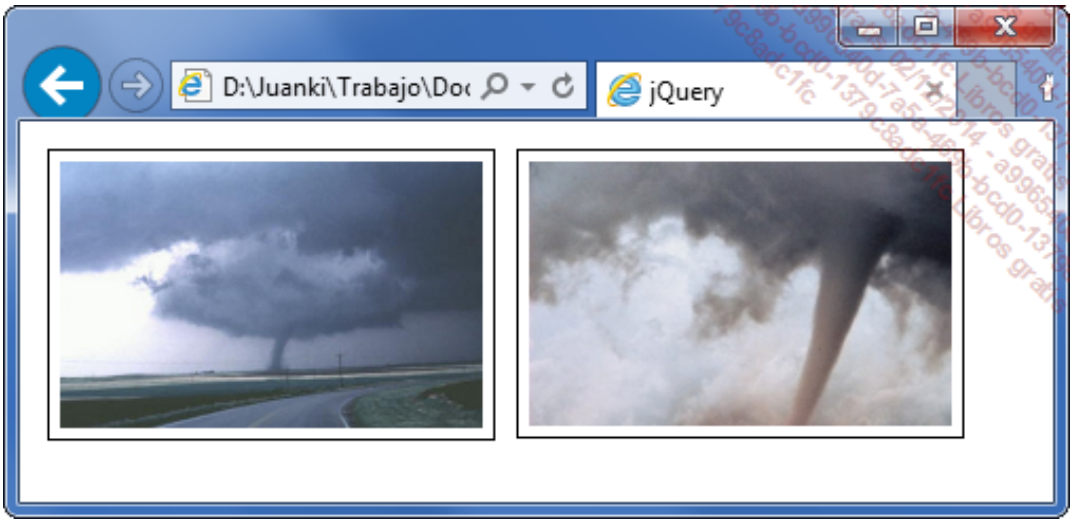
Una lupa para agrandar las viñetas

Al pasar el ratón por encima de una viñeta, vamos a hacer que aparezca una lupa para que el usuario pueda ampliar esa viñeta.

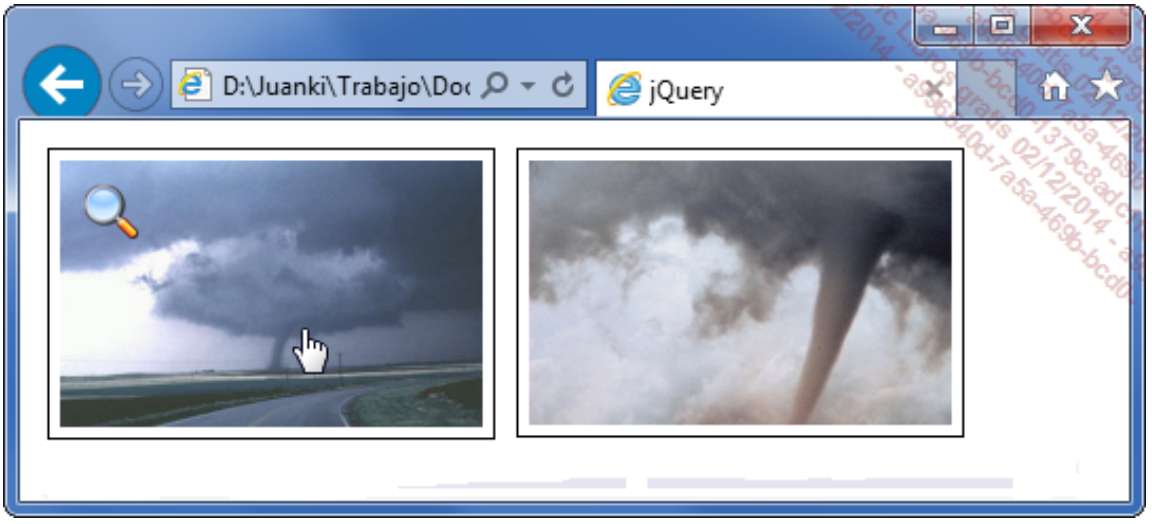
Las diferentes imágenes están disponibles para descarga en la página Información.

El archivo inicial es:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
#galeria { width: 100%;
           overflow: hidden;}
#galeria a { position:relative;
             float:left;
             margin:5px;}
#galeria a span { background-image:url(lupa.png);
                  background-repeat:no-repeat;
                  width:30px;
                  height:30px;
                  display:none;
                  position:absolute;
                  left:15px;
                  top:15px;}
#galeria img { border: solid 1px black;
               padding:5px;}
</style>
</head>
<body>
<div id="galeria">
<a href="tornado.jpg">
<img src="tornado_min.jpg" alt="">
</a>
<a href="tornado1.jpg">
<img src="tornado1_min.jpg" alt="">
</a>
</div>
</body>
</html>
```



El script jQuery hace aparecer una lupa, con un efecto de visualización progresivo, al pasar por encima de la viñeta.



El truco consiste en añadir usando jQuery una etiqueta `<span>` que contiene la lupa y hacerla aparecer al pasar por encima el cursor del ratón.

```
<script>
$(document).ready(function(){
$("#galeria a").append("<span></span>");
$("#galeria a").hover(function(){
$(this).children("span").fadeIn(600);
},
function(){
$(this).children("span").fadeOut(200);
});
});
</script>
```

Es decir:

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

Inicio de jQuery.

```
$("#galeria a").append("<span></span>");
```

El método `append()` añade una etiqueta `<span>` a la etiqueta `<a>` de la capa `galeria`.

```
$("#galeria a").hover(function(){
$(this).children("span").fadeIn(600);
},
```

Al pasar el ratón por encima (`hover`) del enlace `<a>`, el script selecciona, entre los hijos de éste, aquellos que tienen una etiqueta `<span>` (`children("span")`) y se le aplica un efecto de aparición progresiva (`fadeIn(600)`).

```
function(){
$(this).children("span").fadeOut(200);
});
```

Cuando el cursor sale del enlace, esta etiqueta `<span>` desaparece.

```
});
```

Fin del script jQuery.

El código final es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#galeria a").append("<span></span>");
$("#galeria a").hover(function(){
$(this).children("span").fadeIn(600);
}, function(){
$(this).children("span").fadeOut(200);
});
});
</script>
<style>
#galeria { width: 100%;
           overflow: hidden;}
#galeria a { position:relative;
             float:left;
             margin:5px;}
#galeria a span { background-image:url(lupa.png);
                  background-repeat:no-repeat;
                  width:30px;
                  height:30px;
                  display:none;
                  position:absolute;
                  left:15px;
                  top:15px;}
#galeria img { border: solid 1px black;
               padding:5px;}
</style>
</head>
<body>
<div id="galeria">
<a href="tornado.jpg">
<img src="tornado_min.jpg" alt="">
</a>
<a href="tornado1.jpg">
<img src="tornado1_min.jpg" alt="">
</a>
</div>
</body>
</html>
```

Introducción

La integración del DOM ha modificado profundamente la escritura de JavaScript mediante el acceso de los elementos. Pero su verdadera revolución es, sin ninguna duda, la posibilidad de modificar y añadir elementos en la página Html sobre la marcha.

Modificar el contenido

text()

Devuelve el contenido del texto del elemento implicado.

Este método funciona para los documentos Html, Xhtml y XML.

`$("div").text()`: devuelve, en formato texto, el contenido de la etiqueta `<div>`.

Este método devuelve una cadena de caracteres (String).

text(valor)

Asigna un nuevo contenido de texto (ver el argumento valor) a los elementos implicados.

`$("div").text("los nuevos elementos de texto")`: inserta, en formato texto, el contenido`"los nuevos elementos de texto"` en la etiqueta `<div>`.

Este método devuelve un objeto jQuery.

text(función)

Desde la versión 1.4, el método `text()` permite definir el contenido del texto, pasando por una función.

```
$('ul li').text(function(index) {
return 'item number ' + (index + 1);
});
```

html()

Devuelve, en formato Html, el contenido del elemento implicado.

Este método no funciona para los documentos XML (excepto los documentos Xhtml).

`$("div").html()`: devuelve, en formato Html, el contenido de la etiqueta `<div>`.

Este método devuelve una cadena de caracteres (String).

html(valor)

Asigna un nuevo contenido Html (ver el argumento valor) a los elementos implicados.

Esta propiedad no está disponible para los documentos XML, aunque sí para los documentos Html.

`$("div").html("<b>nuevo contenido</b>")`: inserta como Html los elementos que se indican en el argumento, en la etiqueta `<div>`.

Este método devuelve un objeto jQuery.

html(función)

Desde la versión 1.4, el método `html()` permite definir el contenido del texto que se pasa por una función.

```
$('div.demo').html(function() {
return "<p>Nuevo contenido</p>";
});
```

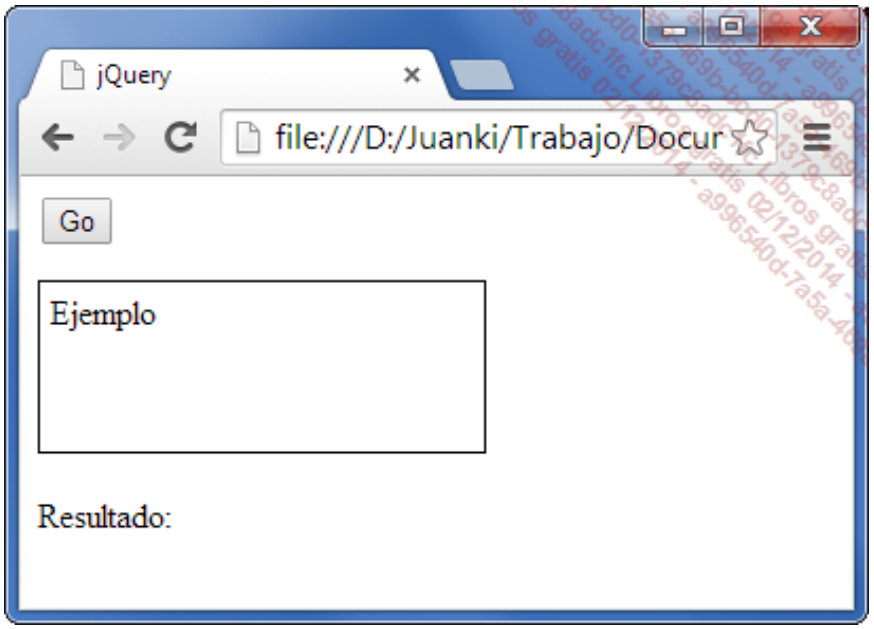
Con los métodos `text(valor)` y `html(valor)`, el nuevo contenido elimina el contenido anterior.

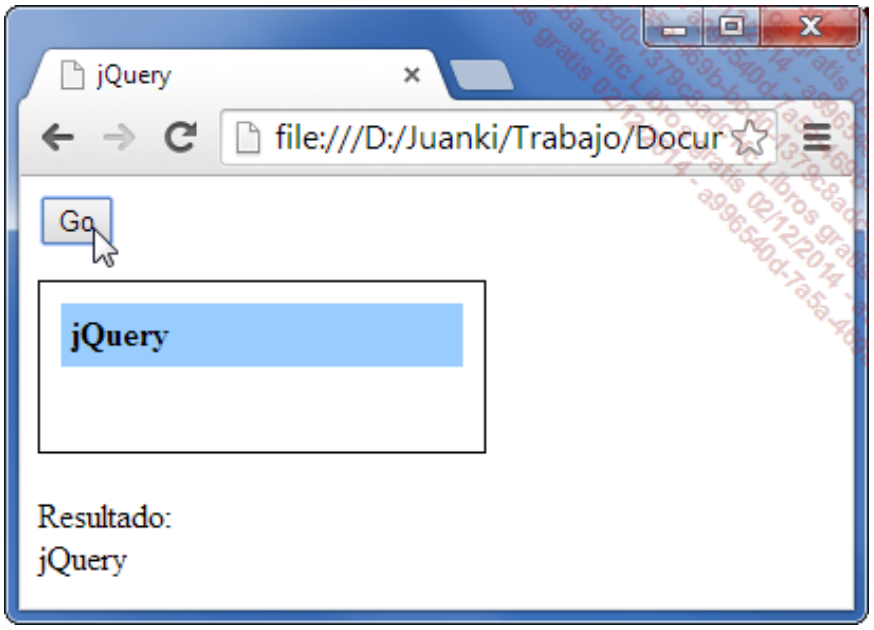
Los métodos `html()` de jQuery usan la propiedad de JavaScript innerHTML. Como recordatorio, esta propiedad, originalmente de Internet Explorer, ha sido adoptada por los otros

navegadores, pero su interpretación plantea algunas veces problemas de compatibilidad.

Ejemplo:

Pensemos en un elemento caja. Al hacer clic en el botón, se inserta un nuevo contenido dentro de la caja. Cuando se completa la operación, el método `text()` *muestra el nuevo contenido.*





Observe que el contenido anterior se elimina.

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
#box { border: 1px solid black;
       width: 200px;
       height: 70px;
       margin-top: 15px;
       padding: 5px;}
```

```
p { background-color: #9cf;
    margin: 5px;
    padding: 5px;}
</style>
</head>
<body>
<input id="boton" type="button" value="Go">
<div id="box">Ejemplo</div><br>
Resultado:
<div id="resultado"></div>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$("#boton").click(function(){
$("#box").html("<p><b>jQuery</b></p>");
var contenido = $("#box").text();
$("#resultado").text(contenido);
});
});
</script>
```

Las explicaciones:

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

Cuando se carga el DOM.

```
$("#boton").click(function(){
$("#box").html("<p><b>jQuery</b></p>")
```

Al hacer clic en el botón, `<p><b>jQuery</b></p>` se inserta en la caja `box` como Html.

```
var contenido = $("#box").text();
$("#resultado").text(contenido);
```

El método `text()` recupera el nuevo contenido de la caja `box` en la variable `contenido` y se muestra en la caja `resultado`.

```
});
});
```

Fin del script.

El archivo final es:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#boton").click(function(){
$("#box").html("<p><b>jQuery</b></p>");
var contenido = $("#box").text();
$("#resultado").text(contenido);
});
```

```
});
</script>
<style>
#box { border: 1px solid black;
       width: 200px;
       height: 70px;
       margin-top: 15px;
       padding: 5px;}
p { background-color: #9cf;
    margin: 5px;
    padding: 5px;}
</style>
</head>
<body>
<input id="boton" type="button" value="Go">
<div id="box">Ejemplo</div><br>
Resultado:
<div id="resultado"></div>
</body>
</html>
```

Insertar en el interior

1. Primer método

append(contenido)

Añade el contenido al final, pero en el interior del elemento especificado.

El contenido puede ser una cadena de caracteres, Html o un objeto jQuery.

append(función)

Desde la versión 1.4 de jQuery, el elemento que se añade también puede estar incluido en una función.

`$("p").append("<b>Hello</b>")`: inserta al final del párrafo los elementos que se proporcionan en los argumentos.

Este método devuelve un objeto jQuery.

Los expertos en JavaScript y en escritura del DOM habrán reconocido el método `appendChild()`.

prepend(contenido)

Añade el contenido al comienzo, pero en el interior del elemento especificado.

El contenido puede ser una cadena de caracteres, Html o un objeto jQuery.

prepend(función)

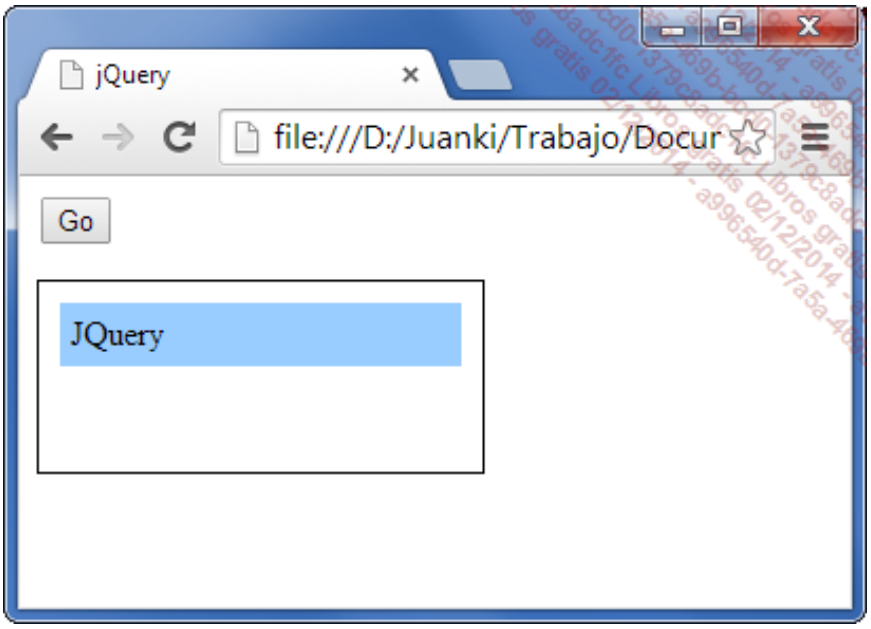
Desde la versión 1.4 de jQuery, el elemento que se añade también puede estar incluido en una función.

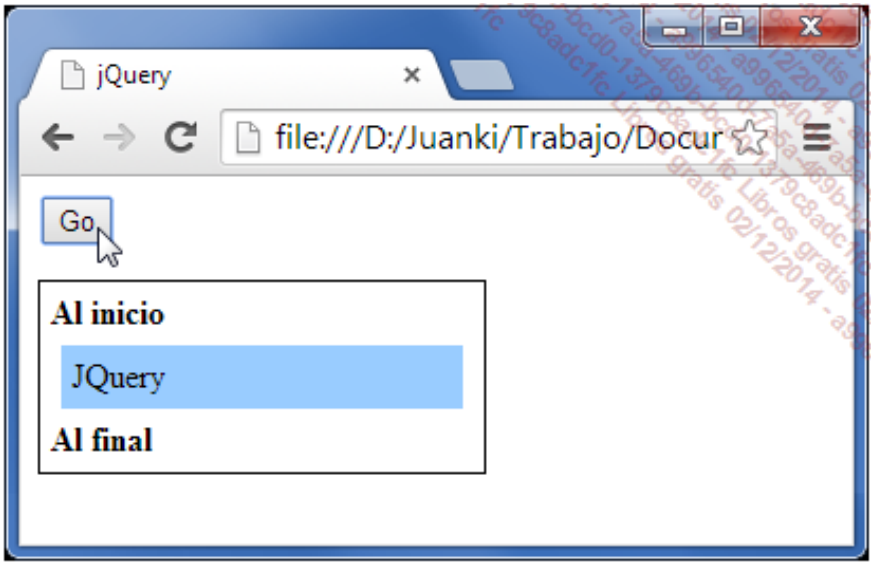
`$("p").prepend("<b>Hello</b>")`: inserta al inicio del párrafo los elementos que se proporcionan en los argumentos.

Este método devuelve un objeto jQuery.

Ejemplo:

Añadimos el contenido al inicio y al final del elemento caja del ejemplo anterior.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
#box { border: 1px solid black;
       width: 200px;
       height: 80px;
       margin-top: 15px;
       padding: 5px;}
 p { background-color: #9cf;
     margin: 5px;
     padding: 5px;}
</style>
</head>
<body>
<input id="boton" type="button" value="Go">
<div id="box">
<p>JQuery</p>
</div>
</body>
</html>
```

Pasamos al script jQuery:

```
<script>
$(document).ready(function(){
$("#boton").click(function(){
$("#box").prepend("<div><b>Al inicio</b></div>");
$("#box").append("<div><b>Al final</b></div>");
});
});
</script>
```

Explicaciones:

```
$(document).ready(function(){
$("#boton").click(function(){
```

Cuando se ha cargado el DOM y al hacer clic en el botón.

```
$("#box").prepend("<div><b>Al inicio</b></div>");
```

El script añade, con el método `prepend()`, el contenido del argumento al inicio del elemento caja`box`.

```
$("#box").append("<div><b>Al final</b></div>");
```

El script añade, con el método `append()`, el contenido del argumento al final del elemento caja `box`.

```
});
});
```

Fin del script.

En la captura de pantalla anterior se puede observar que los elementos se han insertado en el interior del elemento caja.

El archivo completo es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#boton").click(function(){
$("#box").prepend("<div><b>Al inicio</b></div>");
$("#box").append("<div><b>Al final</b></div>");
});
});
</script>
<style>
#box { border: 1px solid black;
       width: 200px;
       height: 80px;
       margin-top: 15px;
       padding: 5px;}
p { background-color: #9cf;
    margin: 5px;
    padding: 5px;}
</style>
</head>
<body>
<input id="boton" type="button" value="Go">
<div id="box">
<p>JQuery</p>
</div>
</body>
</html>
```

2. Segundo método

Los métodos `appendTo()` y `prependTo()` realizan las mismas tareas que `append()` y `prepend()`. La única diferencia es la ubicación en el código, del contenido y del destino. Con `append()` o`prepend()`, el selector anterior al método es el contenedor en el que se inserta el contenido. Con`appendTo()` o `prependTo()`, el contenido precede al método y por lo tanto se inserta en el contenedor de destino. El siguiente ejemplo mostrará claramente todo esto.

appendTo()

Añade los elementos especificados por el selector `A` al final de los especificados por `B`, según la expresión `$(A).appendTo(B)`.

`$("p").appendTo("#box")`: añade el contenido de los elementos `<p>` a la capa con

identificador"box" y al final de ésta.

Este método devuelve un objeto jQuery.

prependTo()

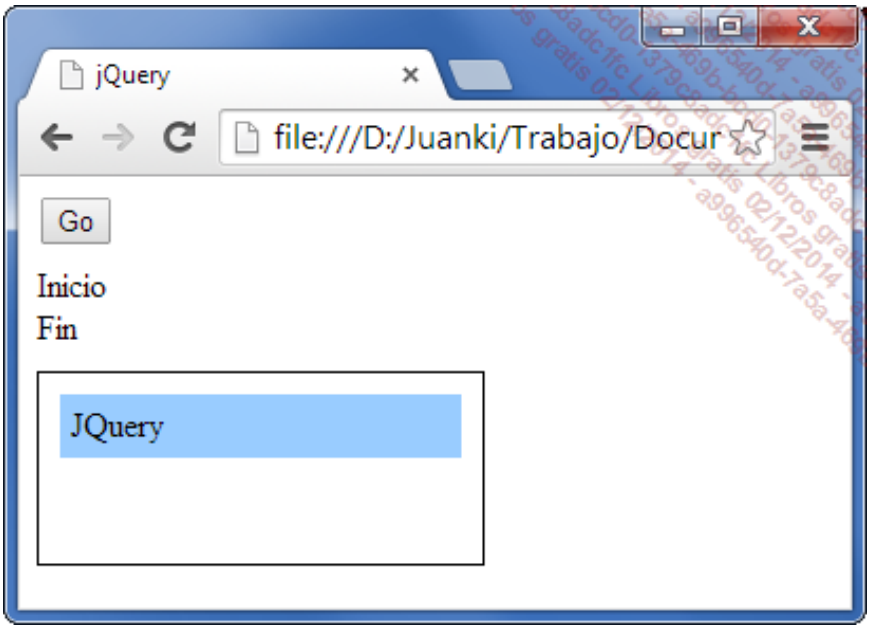
Añade los elementos especificados por el selector A al inicio de los especificados por B, según la expresión $(A).appendTo(B).

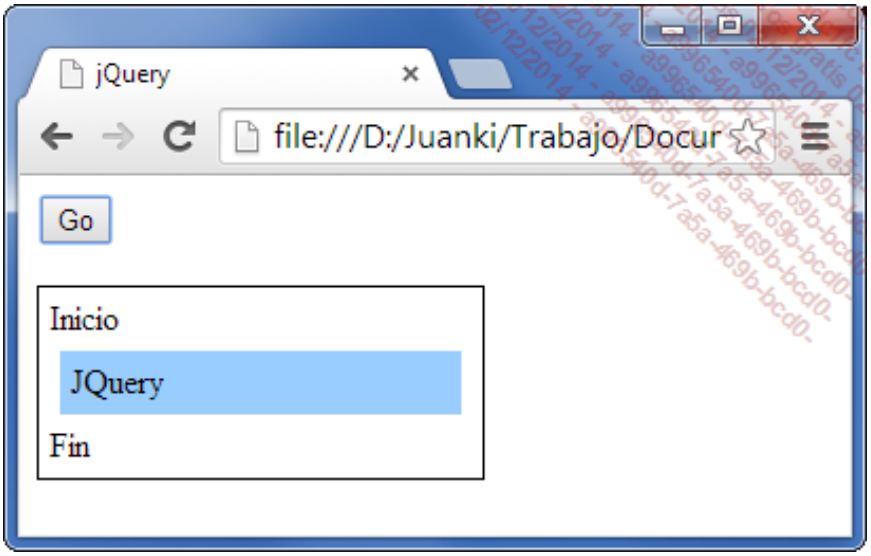
$("p").prependTo("#box"): añade el contenido de los elementos <p> a la capa con identificador"box" y al inicio de ésta.

Este método devuelve un objeto jQuery.

Ejemplo:

Añadamos de nuevo contenido al principio y al final del elemento caja.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
```

```
input { margin-bottom: 10px;}
#box { border: 1px solid black;
       width: 200px;
       height: 80px;
       margin-top: 10px;
       padding: 5px;}
p { background-color: #9cf;
    margin: 5px;
    padding: 5px;}
</style>
</head>
<body>
<input id="boton" type="button" value="Go">
<div id="inicio">Inicio</div>
<div id="fin">Fin</div>
<div id="box">
<p>JQuery</p>
</div>
</body>
</html>
```

El script es:

```
<script>
$(document).ready(function(){
$("#boton").click(function(){
$("#inicio").prependTo("#box");
$("#fin").appendTo("#box");
});
});
</script>
```

Lo detallamos a continuación.

```
$(document).ready(function(){
$("#boton").click(function(){
```

Cuando se carga el DOM y al hacer clic en el botón.

```
$("#inicio").prependTo("#box");
```

Añade los elementos especificados por el selector `#inicio` al inicio de los elementos especificados por el selector `#box`. Dicho de otra manera, añade la capa `<div id="inicio">Inicio</div>` al inicio de los elementos del elemento caja `box`.

```
$("#fin").appendTo("#box");
```

Añade los elementos especificados por el selector `#fin` al final de los elementos especificados por`#box`. Es decir, añade la capa `<div id="inicio">Fin</div>` al final de los elementos del elemento caja `box`.

```
});
});
```

Fin del script.

El Html completo:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
```

```
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#boton").click(function(){
$("#inicio").prependTo("#box");
$("#fin").appendTo("#box");
});
});
</script>
<style>
input { margin-bottom: 10px;}
#box { border: 1px solid black;
       width: 200px;
       height: 80px;
       margin-top: 10px;
       padding: 5px;}
p { background-color: #9cf;
    margin: 5px;
    padding: 5px;}
</style>
</head>
<body>
<input id="boton" type="button" value="Go">
<div id="inicio">Inicio</div>
<div id="fin">Fin</div>
<div id="box">
<p>JQuery</p>
</div>
</body>
</html>
```

Insertar en el exterior

after(contenido)

Añade el contenido especificado en el argumento, después del elemento de selección.

El contenido puede ser una cadena de caracteres, Html o un objeto jQuery.

`$("p").after("<b>Hello</b>")`: añade, después de la etiqueta de párrafo, el contenido que se proporciona en el argumento.

Este método devuelve un objeto jQuery.

after(función)

La versión 1.4 de jQuery también permite pasar por una función, que devuelve los elementos que se van a insertar.

```
$('p').after(function() {
return "<div>Insertion</div>";
});
```

Este ejemplo inserta una etiqueta `<div>` después de cada párrafo.

before(contenido)

Añade el contenido que se especifica en el argumento, antes de cada elemento de la selección.

El contenido puede ser una cadena de caracteres, Html o un objeto jQuery.

`$("p").before("<b>Hello</b>")`: añade, antes de la etiqueta de párrafo, el contenido que se proporciona en el argumento.

before(función)

La versión 1.4 de jQuery permite también pasar por una función, que devuelve los elementos que se van a insertar.

```
$('p').before(function() {
return "<div>Insertion</div>";
});
```

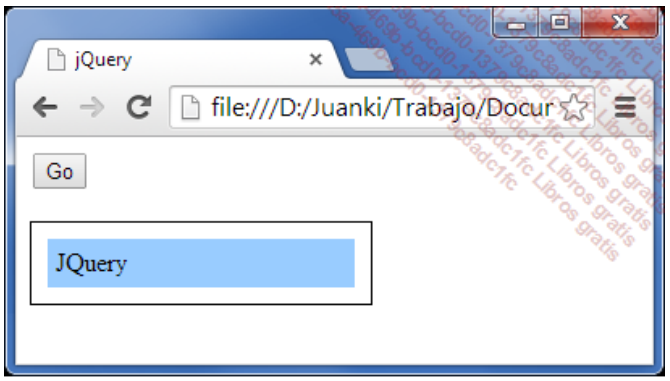
Este ejemplo inserta una etiqueta antes de cada párrafo.

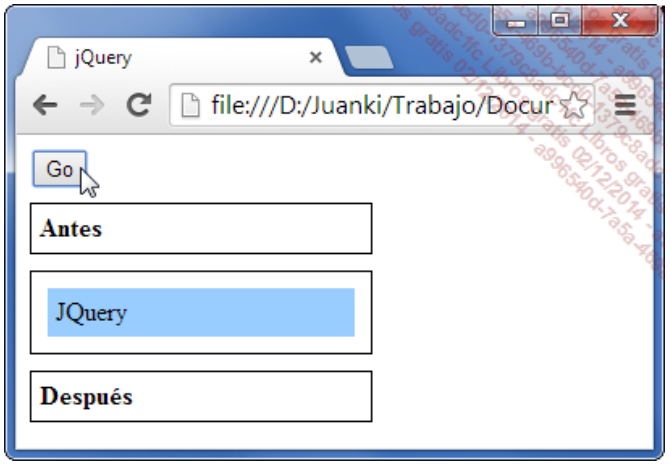
Este método devuelve un objeto jQuery.

Los expertos en JavaScript clásico y en la notación del DOM habrán reconocido el método`insertBefore()`.

Ejemplo

Añadimos el contenido antes y después del elemento caja en el ejemplo anterior.





```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
input{margin-bottom: 10px;}
div {border: 1px solid black;
     width: 200px;
     padding: 5px;}
#box { border: 1px solid black;
       width: 200px;
       height: 40px;
       margin-top: 10px;
       margin-bottom: 10px;
       padding: 5px;}
p { background-color: #9cf;
    margin: 5px;
    padding: 5px;}
</style>
</head>
<body>
<input id="boton" type="button" value="Go">
<div id="box">
<p>JQuery</p>
</div>
</body>
</html>
```

```
<script>
$(document).ready(function(){
$("#boton").click(function(){
$("#box").before("<div><b>Antes</b></div>");
$("#box").after("<div><b>Después</b></div>");
});
});
</script>
```

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#boton").click(function(){
$("#box").before("<div><b>Antes</b></div>");
$("#box").after("<div><b>Después</b></div>");
});
});
</script>
<style>
input{margin-bottom: 10px;}
div {border: 1px solid black;
     width: 200px;
     padding: 5px;}
#box { border: 1px solid black;
       width: 200px;
       height: 40px;
       margin-top: 10px;
       margin-bottom: 10px;
       padding: 5px;}
p { background-color: #9cf;
    margin: 5px;
    padding: 5px;}
</style>
</head>
<body>
<input id="boton" type="button" value="Go">
<div id="box">
<p>JQuery</p>
</div>
</body>
</html>
```

El script jQuery:

```
$(document).ready(function(){
$("#boton").click(function(){
```

Cuando se ha cargado el DOM y al hacer clic en el botón.

```
$("#box").before("<div><b>Ante
```

El script añade con el método `before()` una nueva capa antes del elemento caja `box`.

```
$("#box").after("<div><b>Despu
```

El script añade con el método `after()` una nueva capa después del elemento caja `box`.

```
});
});
```

Fin del script.

En la captura de pantalla se puede observar que los elementos se han insertado en el exterior del elemento caja inicial (`box`).

El archivo completo es el siguiente:

Rodear un elemento

wrap(html o elemento)

Rodea cada elemento seleccionado con el elemento que se proporciona en el argumento. Este procedimiento es muy útil para inyectar una estructura de código adicional en un documento sin modificar la semántica original del documento.

```
$("p").wrap("<div class='wrap'></div>");
```

Este método devuelve un objeto jQuery.

Desde la versión 1.4, esta acción también se puede hacer por una función.

wrapAll(html o elemento)

Rodea todos los elementos de la selección con un único elemento. Es diferente de la función `wrap()`, que rodea cada elemento de la selección con un nuevo elemento (ver los siguientes ejemplos).

```
$("p").wrapAll("<div></div>");
```

Este método devuelve un objeto jQuery.

wrapInner(html o elemento)

Rodea los hijos de un elemento (los nodos de texto incluidos) con otro elemento.

```
$("p").wrapInner("<b></b>");
```

Este método devuelve un objeto jQuery.

Desde la versión 1.4, esto también se puede hacer con una función.

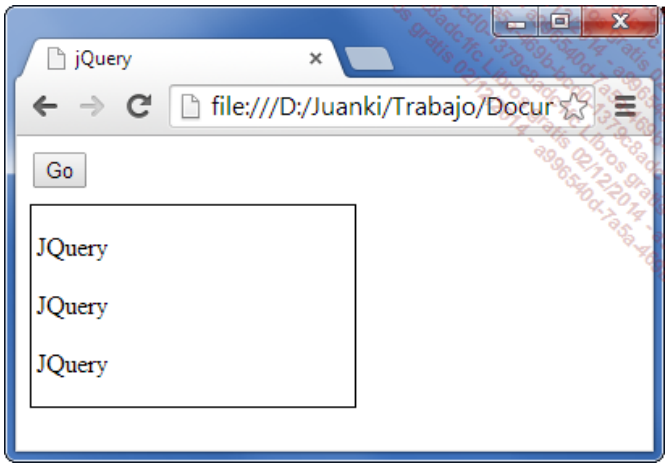
unwrap()

Desde la versión 1.4, el método `unwrap()` permite anular la acción realizada con `wrap()`.

```
if ( p ){
p.appendTo("body");
}
else {
p = $("p").detach();
}
```

Ejemplo

Aplicamos estos métodos a un ejemplo similar a los anteriores.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
input{margin-bottom: 10px;}
.wrap { border: 2px dashed black;
        margin: 3px;
        background-color: #9cf; }
#box { width: 200px;
       background-color: white;
       border: 1px solid black;}
p { padding-left: 3px;}
</style>
</head>
```

El script jQuery con `wrap()`:

```
$(document).ready(function(){
$("#boton").click(function(){
```

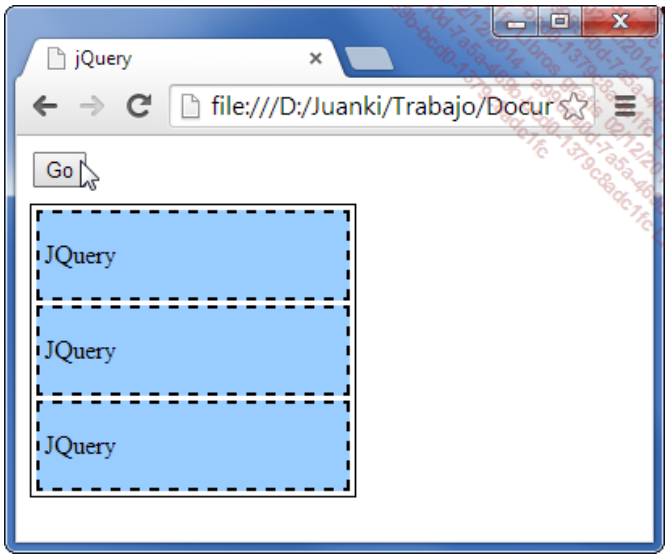
Cuando se carga el DOM y al hacer clic en el botón.

```
$("p").wrap("<div class='wrap'
```

El método `wrap()` rodea cada aparición de párrafo `<p>` con la capa que tiene la clase `wrap`, con un color de fondo y un borde punteado.

```
<body>
<input id="boton" type="button" value="Go">
<div id="box">
<p>JQuery</p>
<p>JQuery</p>
<p>JQuery</p>
</div>
</body>
</html>
```

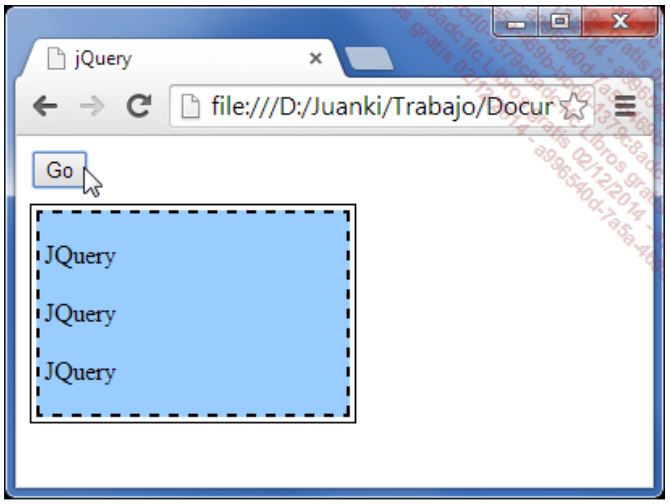
```
<script>
$(document).ready(function(){
$("#boton").click(function(){
$("p").wrap("<div class='wrap'></div>");
});
});
</script>
```



Con el método `wrapAll()`, el script se transforma en:

```
<script>
$(document).ready(function(){
$("#boton").click(function(){
$("p").wrapAll("<div class='wrap'></div>");
});
});
</script>
```

El método `wrapAll()` rodea todas las apariciones de párrafo `<p>` con la capa que tiene la clase `wrap`, con un color de fondo y un borde punteado.



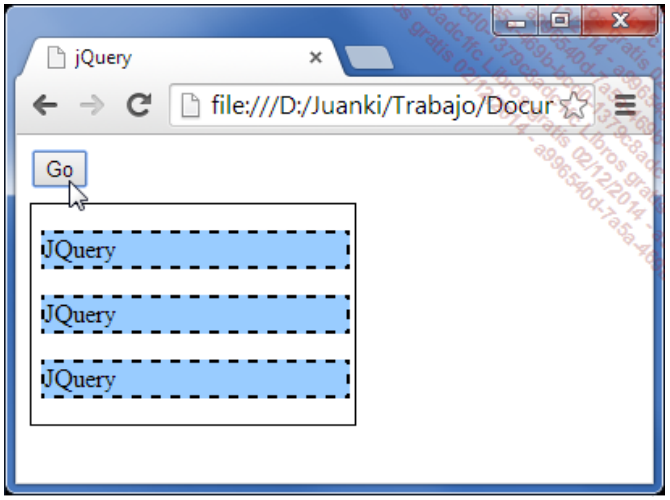
Para terminar, con el método `wrapInner()` el script sería:

```
<script>
$(document).ready(function(){
$("#boton").click(function(){
$("p").wrapInner("<div class='wrap'></div>");
});
});
```

El método `wrapInner()` rodea todos los hijos de los párrafos `<p>` (en este caso el nodo de texto) con la capa que tiene la clase `wrap`, con un color de fondo y un borde

```
</script>
```

punteado.



El archivo completo con `wrap()` es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#boton").click(function(){
$("p").wrap("<div class='wrap'></div>");
});
});
</script>
<style>
input{margin-bottom: 10px;}
.wrap { border: 2px dashed black;
        margin: 3px;
        background-color: #9cf; }
#box { width: 200px;
       background-color: white;
       border: 1px solid black;}
p { padding-left: 3px;}
</style>
</head>
<body>
<input id="boton" type="button" value="Go">
<div id="box">
<p>JQuery</p>
<p>JQuery</p>
<p>JQuery</p>
</div>
</body>
</html>
```

Sustituir un elemento

replaceWith()

Sustituye el elemento actual por el contenido especificado, en formato de código Html o de objeto DOM. La función devuelve el elemento jQuery que se acaba de sustituir y lo elimina del DOM.

```
$("#div1").replaceWith("<div>Nueva capa</div>");
```

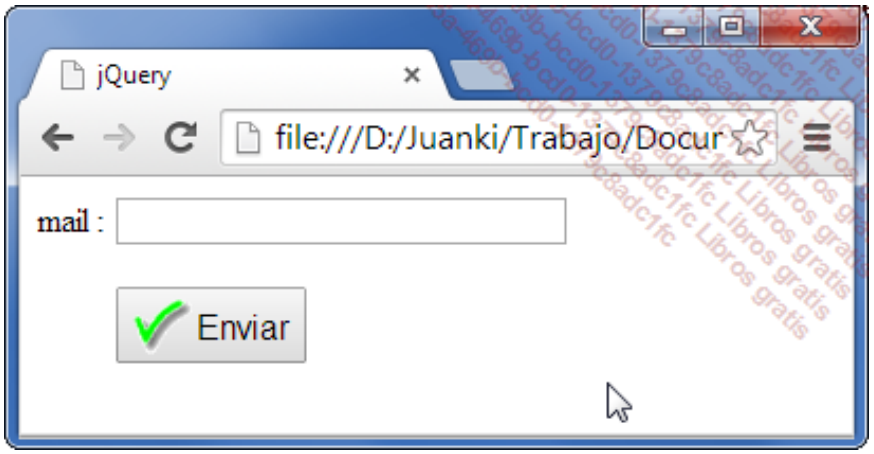
Este método devuelve un objeto jQuery.

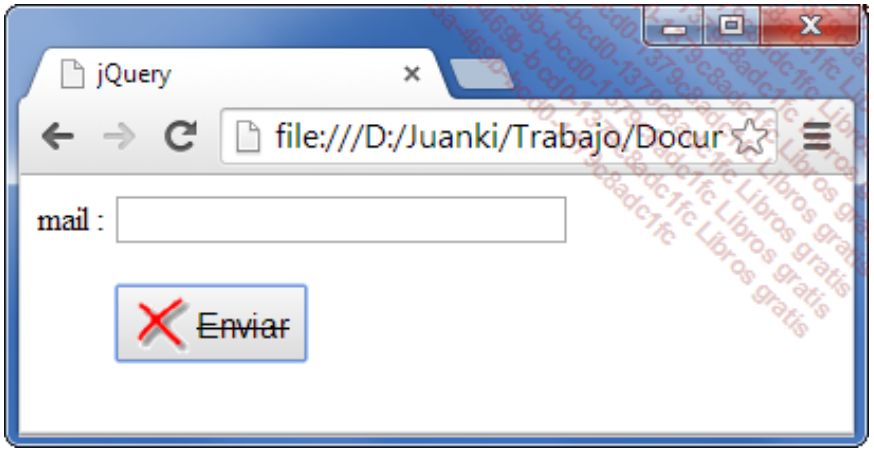
Comentarios

El método `html()` sustituye el contenido del elemento, mientras que `replaceWith()` sustituye el elemento en sí mismo.

Ejemplo

Modificamos el botón de envío de formulario después de hacer clic en él. Las imágenes del ejemplo están disponibles para descarga en la página Información.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
button { cursor: pointer;}
img { float: left;
      width: 30px;
      height: 30px;}
span { float: left;
```

```
       margin-top: 7px;
       font: 1.2em arial,sans-serif}
p { margin-left: 35px;}
</style>
</head>
<body>
mail: <input type="text" size="30"><br>
<p><button id="yes"><img src="yes.gif" alt="">
<span>Enviar</span> </button></p>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$("button").click(function(){
$("img").replaceWith("<img src='no.gif' alt='' />");
$("span").css("text-decoration","line-through");
});
});
</script>
```

Explicaciones:

```
$(document).ready(function(){
$("button").click(function(){
```

Cuando se carga el DOM y al hacer clic en el botón.

```
$("img").replaceWith("<img src='no.gif' alt='' />");
```

Sustituye la imagen inicial por la que se proporciona en el argumento del método `replaceWith()`.

```
$("span").css("text-decoration","line-through");
```

Modifica el estilo de la etiqueta `<span>` tachando el texto.

```
});
});
```

Fin del script.

El documento completo se presenta de la siguiente manera:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("button").click(function(){
$("img").replaceWith("<img src='no.gif' alt='' />");
$("span").css("text-decoration","line-through");
});
});
</script>
<style>
button { cursor: pointer;}
img { float: left;
```

```
      width: 30px;
      height: 30px;}
span { float: left;
       margin-top: 7px;
       font: 1.2em arial,sans-serif}
p { margin-left: 35px;}
</style>
</head>
<body>
mail: <input type="text" size="30"><br>
<p><button id="yes"><img src="yes.gif" alt="">
<span>Enviar</span> </button></p>
</body>
</html>
```

Eliminar un elemento

1. Eliminar un elemento

remove() o remove(selector)

Elimina del árbol del DOM todos los elementos que responden a los criterios de selección.

`$("p").remove()`: elimina el párrafo.

`$('div').remove('.box')`: elimina las divisiones con la clase box.

Este método devuelve un objeto jQuery.

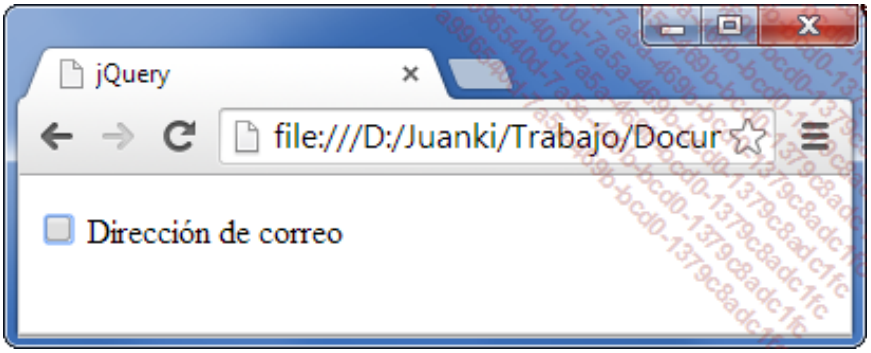
detach() o detach(selector)

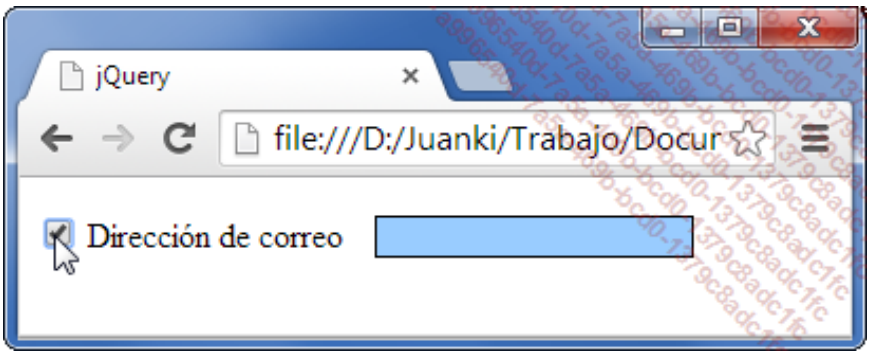
La versión 1.4 de jQuery introdujo el método `detach()`. Es idéntico a `remove()`, pero conserva todos los datos asociados a los elementos que se eliminan. Puede ser de utilidad cuando los datos que se eliminan se insertarán más adelante en el script.

`$("p").detach(":contains('Bienvenido')")`: elimina el párrafo que contiene el dato "Bienvenido".

Ejemplo

Pensemos en un botón de tipo casilla de selección (checkbox). Cuando se selecciona, se muestra un formulario de línea de texto. Cuando se deselecciona, la línea de texto desaparece.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
#mas { border: 1px solid black;
       background-color: #9cf;
       margin-left: 15px;}
```

```
</style>
</head>
<body>
<p id="mail"><input id="clic" type="checkbox"> Dirección de mail</p>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$("#clic").click(function(){
if ($(this).is(":checked")){
$("#mail").append("<input id='mas' type='text' />");
}
else {
$("#mas").remove();
}
});
});
</script>
```

Vamos a detallar los elementos del script.

```
$(document).ready(function(){
$("#clic").click(function(){
```

Cuando se carga el DOM y se marca la casilla de selección.

```
if ($(this).is(":checked")){
```

Si la casilla de selección está marcada.

```
$("#mail").append("<input id='mas' type='text' />");
}
```

La línea de texto se añade en el párrafo.

```
else {
```

Si la casilla de selección no está marcada.

```
$("#mas").remove();
```

La línea de texto se elimina.

```
});
});
```

Fin del script.

Comentario

El mismo efecto se puede obtener con los métodos `show()` y `hide()` (ver el capítulo Los efectos - Mostrar y ocultar):

```
<script>
$(document).ready(function(){
$("#mas").css("display","none");
$("#clic").click(function(){
if ($(this).is(":checked")){
$("#mas").show("fast");
```

```
}
else {
$("#mas").hide("fast");
}
});
});
</script>
```

El archivo completo es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#clic").click(function(){
if ($(this).is(":checked")){
$("#mail").append("<input id='mas' type='text'>");
}
else {
$("#mas").remove();
}
});
});
</script>
<style>
#mas { border: 1px solid black;
       background-color: #9cf;
       margin-left: 15px;}
</style>
</head>
<body>
<p id="mail"><input id="clic" type="checkbox"> Dirección de mail</p>
</body>
</html>
```

2. Vaciar un elemento

empty()

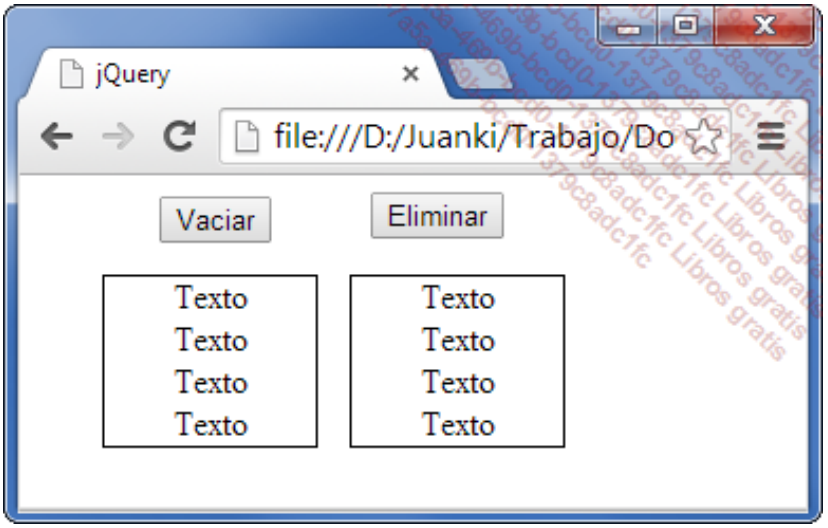
Vacía el elemento seleccionado de todos sus nodos hijos.

`$("p").empty()`: elimina el contenido del párrafo.

Este método devuelve un objeto jQuery.

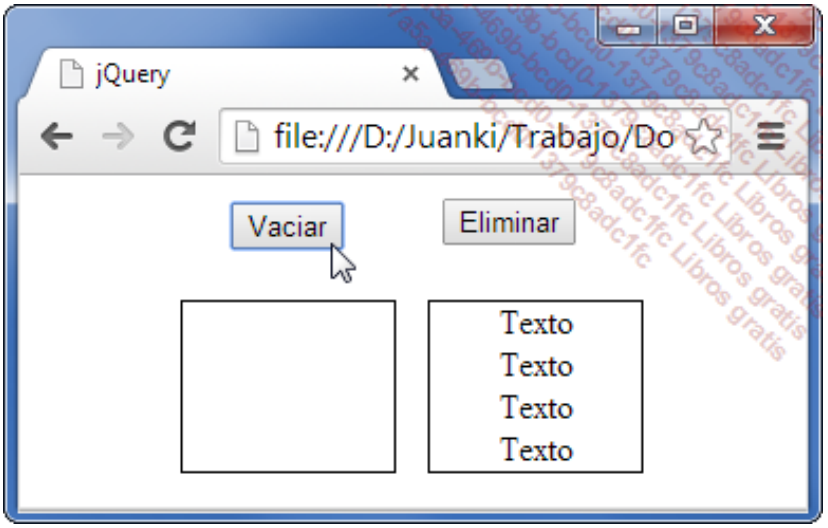
Ejemplo

Usamos un ejemplo para ver la diferencia entre los métodos `empty()` y `remove()`.

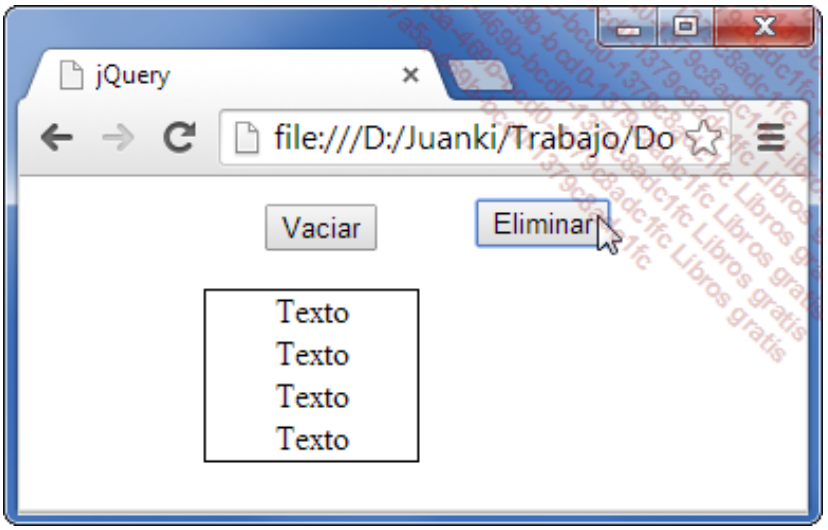



```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
#boton1 { margin-left: 25px;
          display: block;
          float: left;}
#boton2 { margin-left: 125px;
          display: block;}
#div1 { border: 1px solid black;
        margin-top: 15px;
        margin-left: 35px;
        width: 100px;
        height: 80px;
        text-align: center;
        float: left;}
#div2 { border: 1px solid black;
        width: 100px;
        height: 80px;
        margin-top: 15px;
        margin-left: 15px;
        text-align: center;
        float: left;}
</style>
</head>
<body>
<div><button id="boton1">Vaciar</button> <button
id="boton2">Eliminar</button></div>
<div id="div1">
Texto<br>Texto<br>Texto<br>Texto
</div>
<div id="div2">
Texto<br>Texto<br>Texto<br>Texto
</div>
</body>
</html>
```

Al hacer clic en el botón **Vaciar**, el método `empty()` elimina el contenido (es decir, los nodos hijo), pero la capa, que se muestra con un borde, continúa ahí.



Al hacer clic en el botón **Eliminar**, el método `remove()` hace que toda la capa desaparezca.



El script jQuery es:

```
<script>
$(document).ready(function(){
$("#boton1").click(function () {
$("#div1").empty();
});
$("#boton2").click(function () {
$("#div2").remove();
});
});
</script>
```

Lo detallamos a continuación.

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

Al cargar el DOM.

```
$("#boton1").click(function () {
$("#div1").empty();
});
```

Al hacer clic en el botón 1, el contenido de la capa 1 se vacía.

```
$("#boton2").click(function () {
$("#div2").remove();
});
```

Al hacer clic en el botón 2, la capa 2 se elimina.

```
});
```

Fin del ready.

El archivo Html completo es:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#boton1").click(function () {
$("#div1").empty();
});
$("#boton2").click(function () {
$("#div2").remove();
});
});
</script>
<style>
#boton1 { margin-left: 25px;
          display: block;
          float: left;}
#boton2 { margin-left: 125px;
          display: block;}
#div1 { border: 1px solid black;
        margin-top: 15px;
        margin-left: -70px;
        width: 100px;
        height: 80px;
        text-align: center;
        float: left;}
#div2 { border: 1px solid black;
        width: 100px;
        height: 80px;
        margin-top: 15px;
        margin-left: 15px;
        text-align: center;
        float: left;}
</style>
</head>
<body>
<div><button id="boton1">Vaciar</button> <button
id="boton2">Eliminar</button></div>
<div id="div1">
Texto<br>Texto<br>Texto<br>Texto
</div>
<div id="div2">
Texto<br>Texto<br>Texto<br>Texto
</div>
</body>
</html>
```

Copiar un elemento

clone()

Copia (clona) los elementos del DOM encontrados y los selecciona. Esta función es útil para crear copias de elementos y desplazarlas hacia otro lugar especificado del DOM.

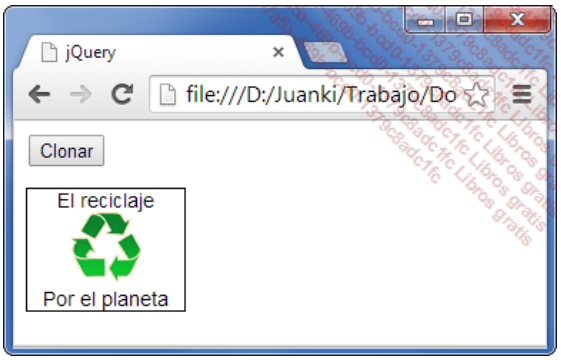
`$("p").clone()`: copia el párrafo.

Parámetros (opcional): indica `true` si desea clonar los administradores de eventos asociados a la selección.

Este método devuelve un objeto jQuery.

Ejemplo

Clonamos una capa y su contenido (`<div id="div1">`), para desplazarla a otro lugar de la página (`<div id="div2">`).



El resultado:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
button {margin-bottom: 15px;}
#div1 { border: 1px solid black;
        width: 110px;
        font: 0.9em arial, sans-serif;
        float: left;
        text-align: center;}
#div2 { margin-left: 20px;
        float: left;}
</style>
</head>
<body>
<button id="boton">Clonar</button><br>
<div id="div1">
El reciclaje<br>
<img src="reciclaje.gif" alt=""><br>
Por el planeta
</div>
<div id="div2"></div>
</body>
</html>
```

El código jQuery:

```
<script>
$(document).ready(function(){
$("#div2").hide();
$("button").click(function () {
$("#div1").clone().prependTo("#div2");
$("#div2").show();
});
});
</script>
```

`$(document).ready(function(){$("#div2").hide();`

Cuando se carga el script, la capa 2 se oculta.

`$("button").click(function () {`

Al hacer clic en el botón.

`$("#div1").clone().prependTo("#div2");`

La capa 1 se copia (`clone()`) y se inserta (`prependTo()`) en la segunda capa.

```
$("#div2").show();
```

Se muestra la capa 2.

```
});
});
```

Fin del script.

El código final es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#div2").hide();
$("button").click(function () {
$("#div1").clone().prependTo("#div2");
$("#div2").show();
});
});
</script>
<style>
button { margin-bottom: 15px;}
#div1 { border: 1px solid black;
        width: 110px;
        font: 0.9em arial, sans-serif;
        float: left;
        text-align: center;}
#div2 { margin-left: 20px;
        float: left;}
</style>
</head>
<body>
<button id="boton">Clonar</button><br>
<div id="div1">
El reciclaje<br>
<img src="reciclaje.gif" alt=""><br>
Por el planeta
</div>
<div id="div2"></div>
</body>
</html>
```

Algunas aplicaciones

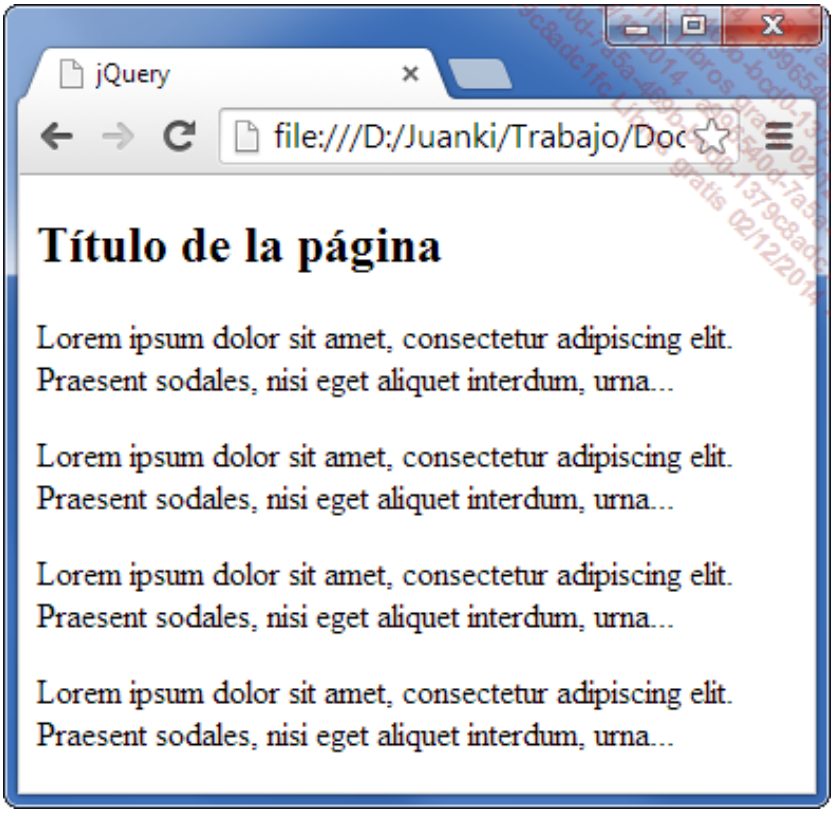
1. Añadir un pie de página y los enlaces de regreso

Vamos a añadir, con algunas líneas de jQuery, un pie de página y los enlaces de regreso hacia la parte superior de la página.

El archivo Html:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
a { color: black;}
#foot { border-top: 1px solid black;
        margin-top: 15px;}
</style>
</head>
<body>
<h2>Título de la página</h2>
<div id="contenido">
<p> Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Praesent sodales, nisi eget aliquet interdum, urna risus sodales
urna, at auctor nisl tellus rutrum nibh. Pellentesque lorem diam,
tincidunt eget tincidunt non, rutrum non felis. In hac habitasse
platea dictumst. Ut nibh leo, placerat a tristica consequat,
vestibulum at tortor. In nec tristica turpis.</p>
<p> Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Praesent sodales, nisi eget aliquet interdum, urna risus sodales
urna, at auctor nisl tellus rutrum nibh. Pellentesque lorem diam,
tincidunt eget tincidunt non, rutrum non felis. In hac habitasse
platea dictumst. Ut nibh leo, placerat a tristica consequat,
vestibulum at tortor. In nec tristica turpis.</p>
<p> Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Praesent sodales, nisi eget aliquet interdum, urna risus sodales
urna, at auctor nisl tellus rutrum nibh. Pellentesque lorem diam,
tincidunt eget tincidunt non, rutrum non felis. In hac habitasse
platea dictumst. Ut nibh leo, placerat a tristica consequat,
vestibulum at tortor. In nec tristica turpis.</p>
<p> Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Praesent sodales, nisi eget aliquet interdum, urna risus sodales
urna, at auctor nisl tellus rutrum nibh. Pellentesque lorem diam,
tincidunt eget tincidunt non, rutrum non felis. In hac habitasse
platea dictumst. Ut nibh leo, placerat a tristica consequat,
vestibulum at tortor. In nec tristica turpis.</p>
</div>
</body>
</html>
```

En este estado, los pies de página y los enlaces de regreso no aparecen en el archivo Html.



Añadimos el código jQuery.

```
<script>
$(document).ready(function(){
$('<a id="top" name="top"></a>').prependTo('body');
$("p").after("<a href='#top'>Parte alta de la página</a>");
$("#contenido").after("<div id='foot'><i>Notas a pie de
página</i></div>");
});
</script>
```

Explicaciones.

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

Inicialización del DOM en jQuery.

```
$('<a id="top" name="top"></a>').prependTo('body');
```

En primer lugar, ponemos el anchor (`<a id="top" name="top"></a>`) al inicio de la página. El método `prepend()` se aplica a la etiqueta `<body>` y permite situar el anchor justo después.

```
$("p").after("<a href='#top'>Inicio de página</a>");
```

Esta línea de código sitúa el enlace de regreso hacia la parte superior de la página (`<a href='#top'>Inicio de página</a>`), después de cada aparición de la etiqueta `<p> ... </p>`.

```
$("#contenido").after("<div id='foot'><i>Notas de pie de página
</i></div>");
```

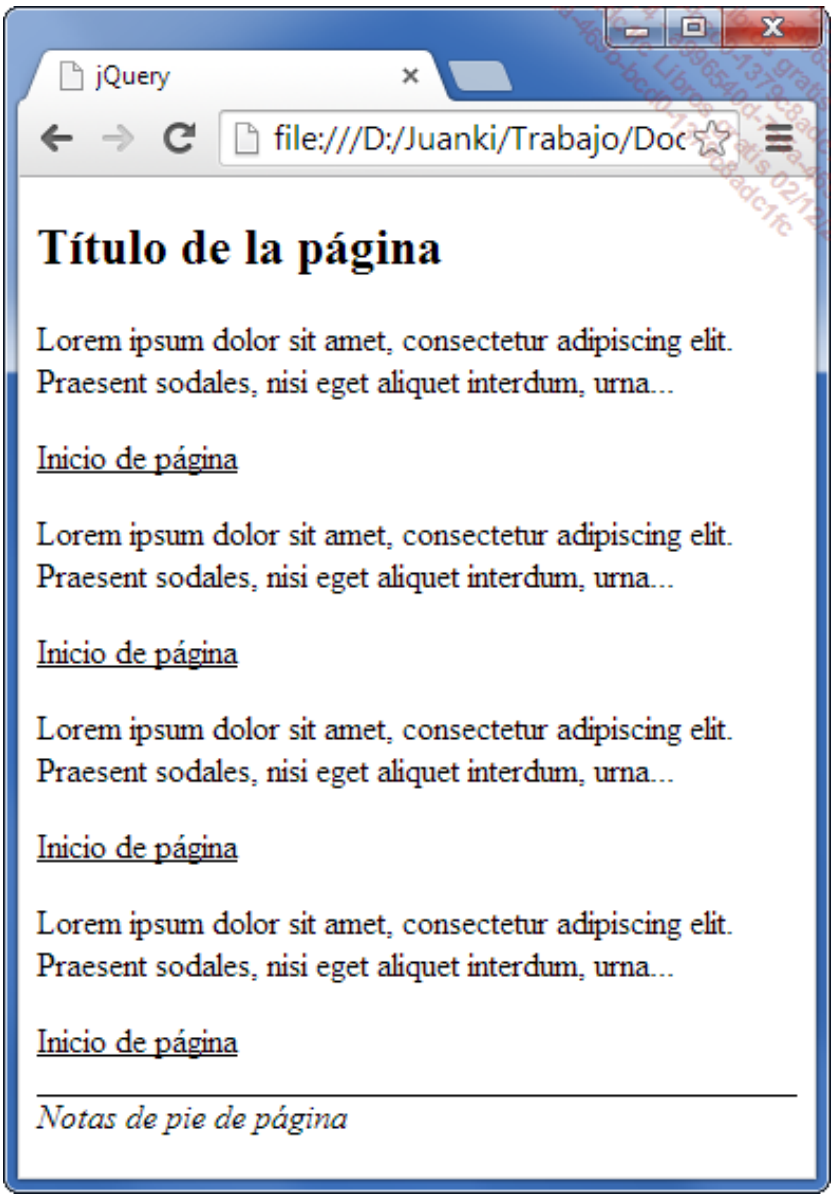
En cuanto al pie de página (`<div id='foot'><i>Notas de pie de página</i></div>`), se insertará después de la capa que contiene los diferentes párrafos.

```
});
```

Fin del script.

Este ejemplo ilustra bien, en nuestra opinión, cómo de conciso puede llegar a ser el código jQuery, incluso para llevar a cabo operaciones complicadas.

El resultado:



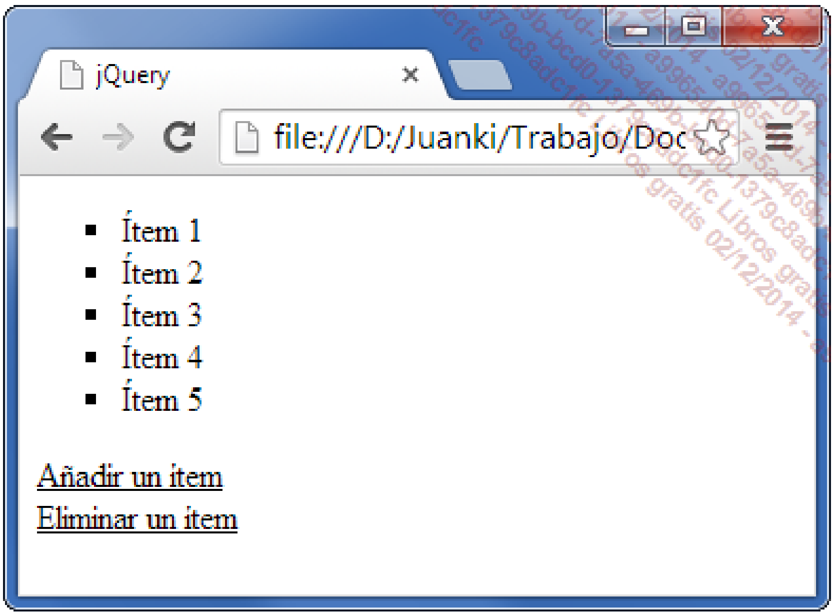
El archivo completo es el siguiente, con el script jQuery:

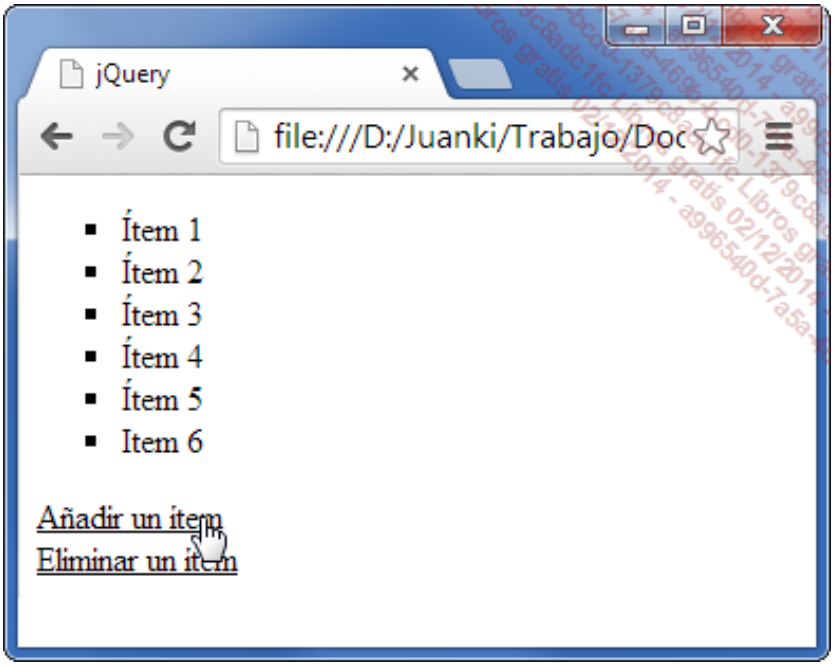
```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$('<a id="top" name="top"></a>').prependTo('body');
$("p").after("<a href='#top'>Inicio de página</a>");
$("#contenido").after("<div id='foot'><i>Notas de pie de
página</i></div>");
});
</script>
<style>
a { color: black;}
```

```
#foot { border-top: 1px solid black;
        margin-top: 15px;}
</style>
</head>
<body>
<h2>Título de la página</h2>
<div id="contenido">
<p> Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Praesent sodales, nisi eget aliquet interdum, urna risus sodales
urna, at auctor nisl tellus rutrum nibh. Pellentesque lorem diam,
tincidunt eget tincidunt non, rutrum non felis. In hac habitasse
platea dictumst. Ut nibh leo, placerat a tristica consequat,
vestibulum at tortor. In nec tristica turpis.</p>
<p> Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Praesent sodales, nisi eget aliquet interdum, urna risus sodales
urna, at auctor nisl tellus rutrum nibh. Pellentesque lorem diam,
tincidunt eget tincidunt non, rutrum non felis. In hac habitasse
platea dictumst. Ut nibh leo, placerat a tristica consequat,
vestibulum at tortor. In nec tristica turpis.</p>
<p> Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Praesent sodales, nisi eget aliquet interdum, urna risus sodales
urna, at auctor nisl tellus rutrum nibh. Pellentesque lorem diam,
tincidunt eget tincidunt non, rutrum non felis. In hac habitasse
platea dictumst. Ut nibh leo, placerat a tristica consequat,
vestibulum at tortor. In nec tristica turpis.</p>
<p> Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Praesent sodales, nisi eget aliquet interdum, urna risus sodales
urna, at auctor nisl tellus rutrum nibh. Pellentesque lorem diam,
tincidunt eget tincidunt non, rutrum non felis. In hac habitasse
platea dictumst. Ut nibh leo, placerat a tristica consequat,
vestibulum at tortor. In nec tristica turpis.</p>
</div>
</body>
</html>
```

2. Añadir y eliminar elementos de una lista

Supongamos una lista no ordenada. Al hacer clic en un enlace, añadimos un elemento de la lista. Al hacer clic en otro enlace, eliminamos un elemento de la lista.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
ul { list-style-type: square;}
a { color: black;}
</style>
</head>
<body>
<ul>
<li>Ítem 1</li>
<li>Ítem 2</li>
<li>Ítem 3</li>
<li>Ítem 4</li>
<li>Ítem 5</li>
</ul>
<a href="#" id="add">Añadir un ítem</a><br>
<a href="#" id="remove">Eliminar un ítem</a>
</body>
</html>
```

El código jQuery:

```
<script>
$(document).ready(function(){
var i = $('li').size() + 1;
$('a#add').click(function() {
$('<li>Item ' + i + '</li>').appendTo('ul');
i++;
});
$('a#remove').click(function() {
$('li:last').remove();
i--;
});
});
</script>
```

Explicaciones:

El script se articula alrededor de las dos operaciones siguientes.

```
$('a#add').click(function() {
$('<li>Item ' + i + '</li>').appendTo('ul );
```

Al hacer clic en el enlace **Añadir un ítem**, se añade, como último elemento, un elemento `<li> ... </li>` a la lista no ordenada.

```
$('a#remove').click(function() {
$('li:last').remove();
```

Al hacer clic en el enlace **Eliminar un ítem**, se elimina el último elemento `<li> ... </li>`(`li:last`).

Sin embargo, hay que añadir un contador para incrementar o disminuir el número asociado al ítem.

```
var i = $('li').size() + 1;
$('a#add').click(function() {
$('<li>Item ' + i + '</li>').appendTo('ul );
i++;
});
```

Se añade un contador (`var i`). Se inicia con el número de elementos de la lista (`size()`). Tendremos que añadir una unidad ya que, como sucede normalmente en JavaScript, empieza sus contadores por 0. Cuando se ha añadido el elemento de lista, se incrementa la variable `i` en una unidad (`i++`).

```
$('a#remove').click(function() {
$('li:last').remove();
i--;
});
```

De manera parecida, cada vez que se elimina un elemento de la lista, el contador `i` se decrementa en una unidad.

```
});
```

Fin del script.

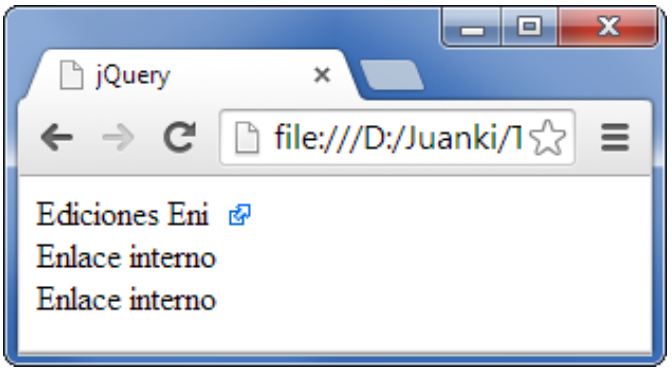
El archivo final:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
var i = $('li').size() + 1;
$('a#add').click(function() {
$('<li>Item ' + i + '</li>').appendTo('ul );
i++;
});
$('a#remove').click(function() {
$('li:last').remove();
i--;
});
});
</script>
<style>
ul {list-style-type: square;}
a {color: black;}
</style>
</head>
```

```
<body>
<ul>
<li>Ítem 1</li>
<li>Ítem 2</li>
<li>Ítem 3</li>
<li>Ítem 4</li>
<li>Ítem 5</li>
</ul>
<a href="#" id="add">Añadir un ítem</a><br>
<a href="#" id="remove">Eliminar un ítem</a>
</body>
</html>
```

3. Añadir un icono a los enlaces externos

El pequeño icono que sigue a los enlaces externos está muy de moda en los sitios Web 2.0. jQuery permite añadir fácilmente estas pequeñas imágenes.



El icono está disponible para descarga en la página Información.

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function() {
$("a").filter(function() {
return this.hostname !== location.hostname;
})
.after('<img src="externo.png" class="externa">');
});
</script>
<style>
a { color: black;
    text-decoration: none;}
.externa { border: none;
           margin-left: 10px;}
</style>
</head>
<body>
<a href="http://www.edicioneseni.com/" title="Enlace
externo">Ediciones Eni</a><br>
<a href="#">Enlace interno</a><br>
<a href="#">Enlace interno</a><br>
</body>
</html>
```

Para identificar los enlaces externos, probamos el nombre de dominio de la página

(`location.hostname`) con respecto al nombre de dominio (`this.hostname`) de los enlaces. Nos aseguramos de que los dos son diferentes (`this.hostname && location.hostname !== this.hostname`).

```
 .after('<img src="externo.png" class="externa">');
```

Cuando se aplica este filtro, el script inserta después del enlace (`after`) el icono de enlace externo (`<img src="externo.png" class="externa">`).

Introducción

Recorrer el DOM y trabajar con él son nociones que ya existían en el JavaScript tradicional. La librería jQuery permite que estas acciones sean más fáciles y, sobre todo, menos tediosas. Cuando se introduce la noción de filtrado de los elementos, jQuery, en su afán de recuperar más rápidamente los elementos, aporta un concepto innovador que permite reducir los resultados de la búsqueda según los criterios especificados por el desarrollador.

El filtrado de los elementos del DOM

El filtrado se puede hacer según dos criterios:

- Mediante una expresión
- Mediante una función

1. Por una expresión

filter(expresión)

Elimina todos los elementos seleccionados que no responden a la búsqueda iniciada, según la expresión que se proporciona como parámetro.

- "expresión" (cadena de caracteres): expresión que se desea buscar.

```
$("div").filter(".selected")
```

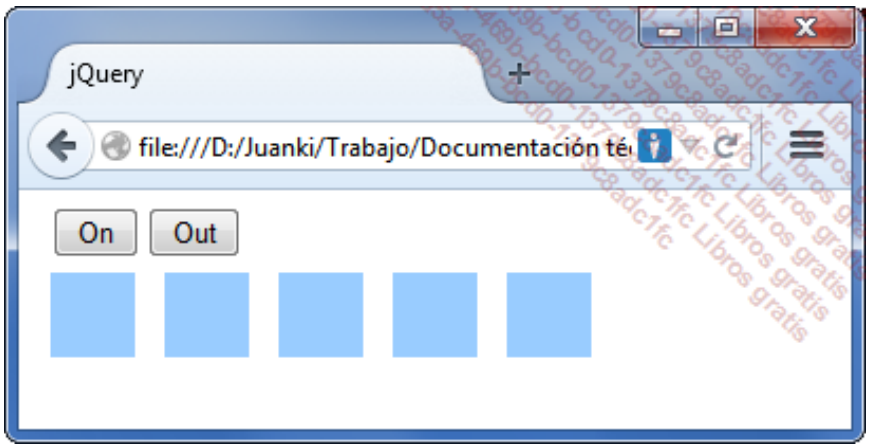
Es posible prever varias expresiones, separadas por comas, para aplicar múltiples filtros de búsqueda.

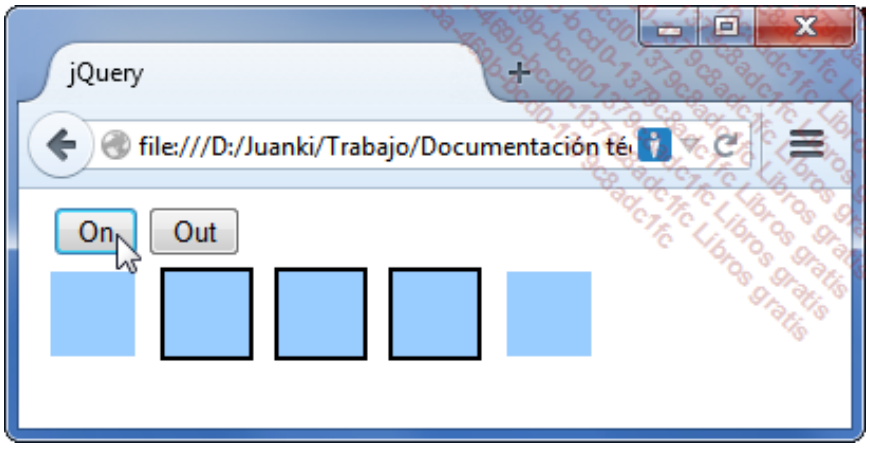
```
$("div").filter(".selected,:first")
```

El método devuelve un objeto jQuery.

Ejemplo

Pensemos en una página que tiene múltiples capas. Al hacer clic en un botón, pedimos a jQuery que recupere solo los elementos con la clase `middle`*. Otro botón restituye la página a su estado inicial.*





El archivo de inicio es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
#on { margin-left: 8px;}
div { width:40px;
      height:40px;
      margin:5px;
      float:left;
      border:2px solid white;
      background-color: #9cf;}
</style>
</head>
<body>
<button id="on">On</button> <button id="out">Out</button>
<br />
<div></div>
<div class="middle"></div>
<div class="middle"></div>
<div class="middle"></div>
<div></div>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$("#on").click(function(){
$("div").filter(".middle")
        .css("border-color", "black");
});
$("#out").click(function(){
$("div").filter(".middle")
        .css("border-color", "white");
});
});
</script>
```

Explicaciones:

```
$(document).ready(function(){
$("#on").click(function(){
```

Cuando se carga el DOM y al hacer clic en el botón **On**.

```
$("div").filter(".middle")
        .css("border-color", "black");
});
```

Aplicamos a la selección de todas las capas (`$("div")`), un filtro sobre la clase `middle`(`filter(".middle"`). Se aplica un borde negro a los elementos devueltos.

```
$("#out").click(function(){
$("div").filter(".middle")
        .css("border-color", "white");
});
```

Al hacer clic en el botón **Out**, se restablece el estado inicial aplicando un borde blanco.

```
});
```

Fin del script.

La página final:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js" ></script>
<script>
$(document).ready(function(){
$("#on").click(function(){
$("div").filter(".middle")
.css("border-color", "black");
});
$("#out").click(function(){
$("div").filter(".middle")
        .css("border-color", "white");
});
});
</script>
<style>
#on { margin-left: 8px;}
div { width:40px;
      height:40px;
      margin:5px;
      float:left;
      border:2px solid white;
      background-color: #9cf;}
</style>
</head>
<body>
<button id="on">On</button> <button id="out">Out</button>
<br />
<div></div>
<div class="middle"></div>
<div class="middle"></div>
<div class="middle"></div>
<div></div>
</body>
</html>
```

Es interesante mostrar la flexibilidad de este método usando la aplicación de otros filtros.

```
$("div").filter(":first").css("border-color", "black");
```

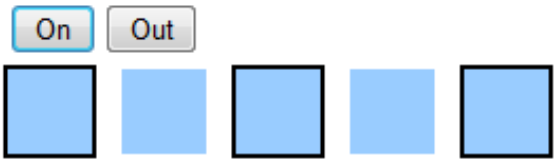
Se selecciona la primera capa.


```
$("div").filter(":last").css("border-color", "black");
```

Se seleccionan la última capa.

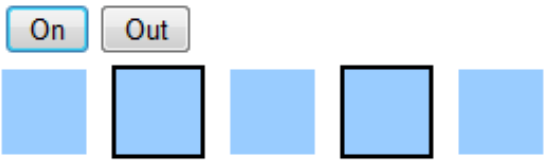

```
$("div").filter(":even").css("border-color", "black");
```

Se seleccionan las capas impares.



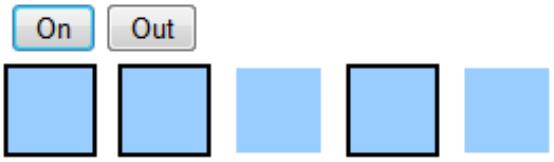

```
$("div").filter(":odd").css("border-color", "black");
```

Se seleccionan las capas pares.




```
$("div").filter(": first,:odd").css("border-color", "black");
```

Se seleccionan la primera capa y las pares.



2. Por una función

Este filtrado también se puede asociar a una función dada como argumento. El valor devuelto por la función (`true` o `false`) para cada elemento de la selección determina si se eliminará o no. Si la función devuelve `false`, el elemento se elimina. Si la función devuelve `true`, el elemento se conserva.

filter(función)

Elimina de la selección todos los elementos seleccionados que no respondan a los criterios de la función dada como argumento.

```
$("p").filter(function(index) {
return $("ol", this).length == 0;
})
```

Selecciona todos los párrafos y después elimina los que tienen como elemento hijo una lista ordenada (`<ol>`).

El método devuelve un objeto jQuery.

La versión 1.4 de jQuery facilita el trabajo del desarrollador permitiendo especificar simplemente un elemento del DOM o un objeto jQuery.

```
$("div").filter(document.getElementById("unico"))
$("div").filter( $("#unico"))
```

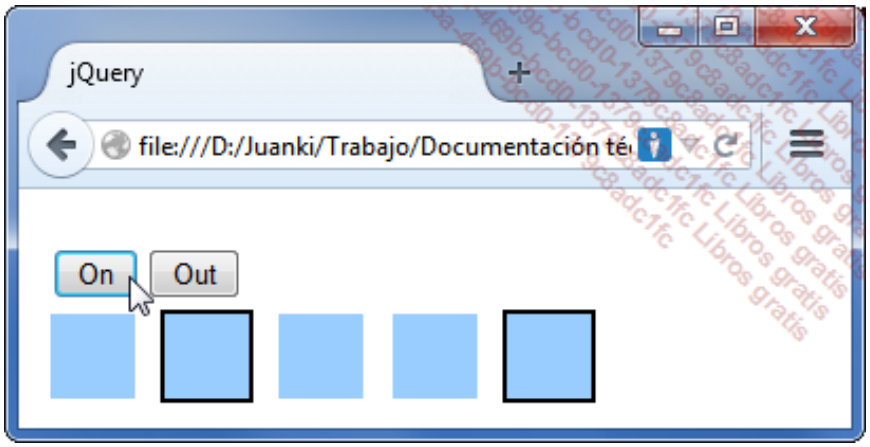
Devuelve todas las capas y filtra la selección conservando solo la que tiene el identificador "unico".

Ejemplo

Volvamos a un ejemplo con múltiples capas.

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
#on { margin-left: 8px;}
div { width:40px;
      height:40px;
      margin:5px;
      float:left;
      border:2px solid white;
      background-color: #9cf;}
</style>
</head>
<body>
<br />
<button id="on">On</button> <button id="out">Out</button>
<br />
<div id="uno"></div>
<div id="dos"></div>
<div id="tres"></div>
<div id="cuatro"></div>
<div id="cinco"></div>
</body>
</html>
```

El script jQuery recupera las capas segunda y quinta y las rodea con un borde negro.




```
<script>
$(document).ready(function(){
$("#on").click(function(){
$("div").filter(function (index) {
return index == 1 || $(this).attr("id") == "cinco";
```

```
})
.css("border-color", "black");
});
$("#out").click(function(){
$("div").filter(function (index) {
return index == 1 || $(this).attr("id") == "cinco";
})
.css("border-color", "white");
});
});
</script>
```

Vamos a detallar este script.

```
$(document).ready(function(){
$("#on").click(function(){
```

Cuando se carga el DOM y al hacer clic en el botón **On**.

```
$("div").filter(function (index) {
return index == 1 || $(this).attr("id") == "cinco";
})
```

Las divisiones se filtran por una función. Esta función indica que los elementos que se tienen que recuperar son la segunda capa (`index == 1`) y la que tiene el atributo `id` con valor `cinco`(`attr("id") == "cinco"`). Recordemos que en JavaScript la numeración empieza por 0.

```
.css("border-color", "black");
});
```

Se añade un borde negro a los elementos que devuelve el código anterior.

```
$("#out").click(function(){
```

Al hacer clic en el botón **Out**.

```
$("div").filter(function (index) {
return index == 1 || $(this).attr("id") == "cinco";
})
```

Se aplica el mismo filtro a las capas.

```
.css("border-color", "white");
```

El borde se vuelve blanco para dejar las capas en su estado inicial.

```
});
});
```

Fin del script.

El código final es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js" ></script>
<script>
$(document).ready(function(){
$("#on").click(function(){
```

```
$("div").filter(function (index) {
return index == 1 || $(this).attr("id") == "cinco";
})
.css("border-color", "black");
});
$("#out").click(function(){
$("div").filter(function (index) {
return index == 1 || $(this).attr("id") == "cinco";
})
.css("border-color", "white");
});
});
</script>
<style>
#on { margin-left: 8px;}
div { width:40px;
      height:40px;
      margin:5px;
      float:left;
      border:2px solid white;
      background-color: #9cf;}
</style>
</head>
<body>
<br />
<button id="on">On</button> <button id="out">Out</button>
<br />
<div id="uno"></div>
<div id="dos"></div>
<div id="tres"></div>
<div id="cuatro"></div>
<div id="cinco"></div>
</body>
</html>
```

Encontrar un elemento concreto

El método `eq()` (eq significa "equal"), permite dirigir la búsqueda directamente a un elemento específico.

eq(index)

Reduce el resultado de la búsqueda a un elemento cuya posición se proporciona como argumento (índice).

- "índice" (entero): determina la posición del elemento. El intervalo de las posiciones empieza en 0 y termina en el tamaño del índice - 1.

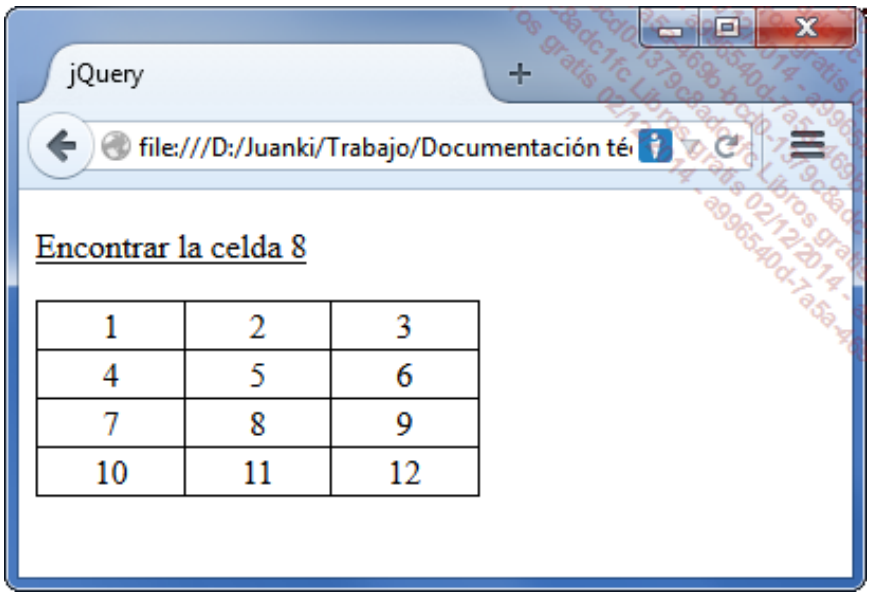
  Desde la versión 1.4 de jQuery, es posible usar un entero negativo. La cuenta se efectúa a partir del último elemento hasta el primero.

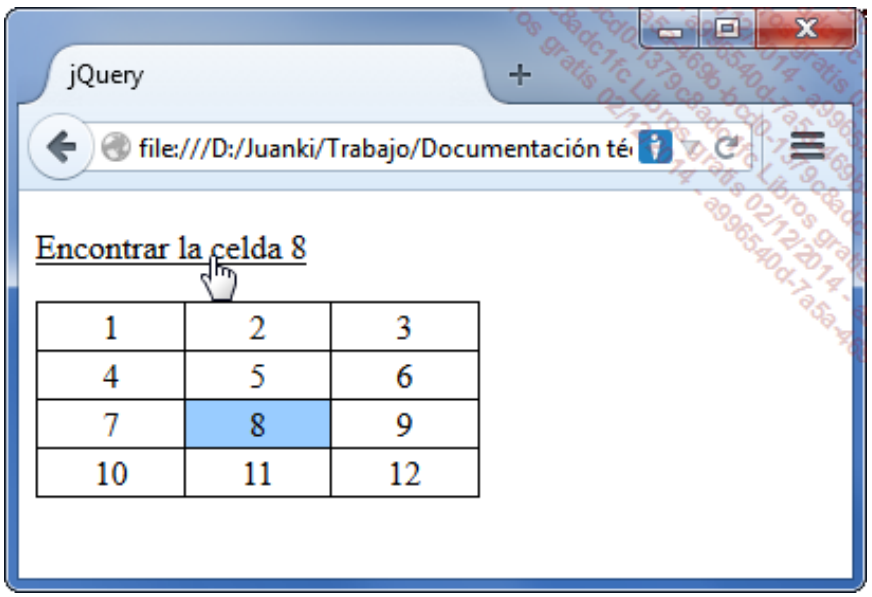
`$("p").eq(1)`: selecciona el segundo párrafo.

El método devuelve un objeto jQuery.

Ejemplo

Supongamos una tabla de cuatro filas y tres columnas. Al hacer clic en el enlace, la celda 8 adquiere un color de fondo.





```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
a { color: black;}
.azul { background-color: #9cf;}
table { width: 210px;
        border-collapse: collapse;
        border: 1px solid black;}
td { text-align: center;
     border: 1px solid black;}
</style>
</head>
<body>
<p><a href="#">Encontrar la celda 8</a></p>
<table>
<tr><td>1</td><td>2</td><td>3</td></tr>
<tr><td>4</td><td>5</td><td>6</td></tr>
<tr><td>7</td><td>8</td><td>9</td></tr>
<tr><td>10</td><td>11</td><td>12</td></tr>
</table>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$("a").click(function(){
$("td").eq(7).addClass("azul");
});
});
</script>
```

Explicaciones.

```
$(document).ready(function(){
$("a").click(function(){
```

Cuando se carga el DOM y al hacer clic en el enlace.

```
$("td").eq(7).addClass("azul");
```

Las celdas de la tabla se cargan en un objeto jQuery (`$("td")`). Entre ellas, se devuelve la celda cuya posición de índice es 7 (`eq(7)`). Entonces se le aplica la clase `azul` (`addClass("azul")`).

Como las posiciones de índice empiezan por 0 (al igual que en JavaScript clásico), la posición 7 corresponde a la celda 8.

```
});
});
```

Fin del script.

El archivo final es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
```

```
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("a").click(function(){
$("td").eq(7).addClass("azul");
});
});
</script>
<style>
a { color: black;}
.azul { background-color: #9cf;}
table { width: 210px;
        border-collapse: collapse;
        border: 1px solid black;}
td { text-align: center;
     border: 1px solid black;}
</style>
</head>
<body>
<p><a href="#">Encontrar la celda 8</a></p>
<table>
<tr><td>1</td><td>2</td><td>3</td></tr>
<tr><td>4</td><td>5</td><td>6</td></tr>
<tr><td>7</td><td>8</td><td>9</td></tr>
<tr><td>10</td><td>11</td><td>12</td></tr>
</table>
</body>
</html>
```

Encontrar una secuencia de elementos

slice(posición de inicio [,posición de fin])

Extrae una secuencia entre los elementos de la búsqueda.

- "posición de inicio" (entero): indica la posición del primer elemento de la secuencia. Este entero puede ser negativo. En este caso, la selección empieza desde el final de la selección inicial.
- "posición de fin" (entero) (opcional): indica la posición (no incluida, estrictamente inferior) del último elemento de la secuencia.

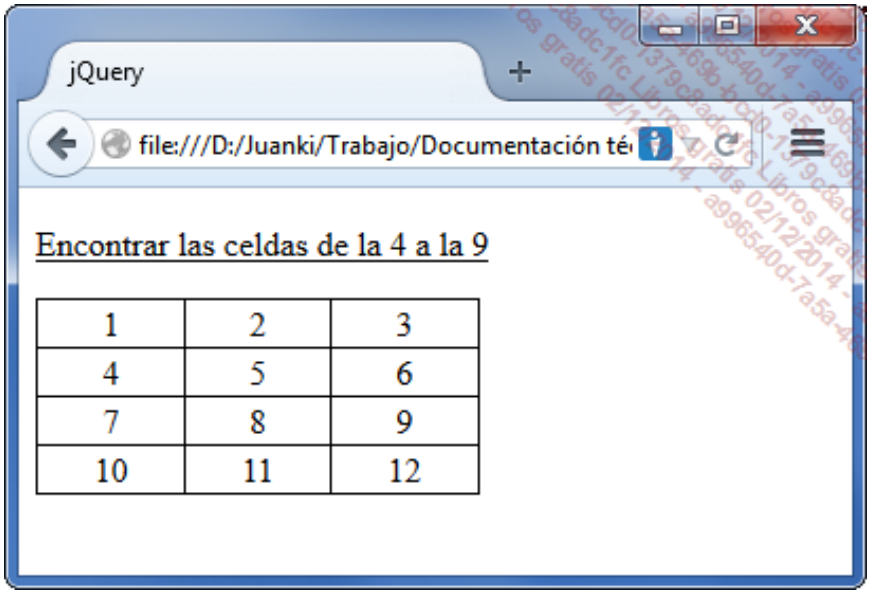
El índice de las posiciones empieza en 0.

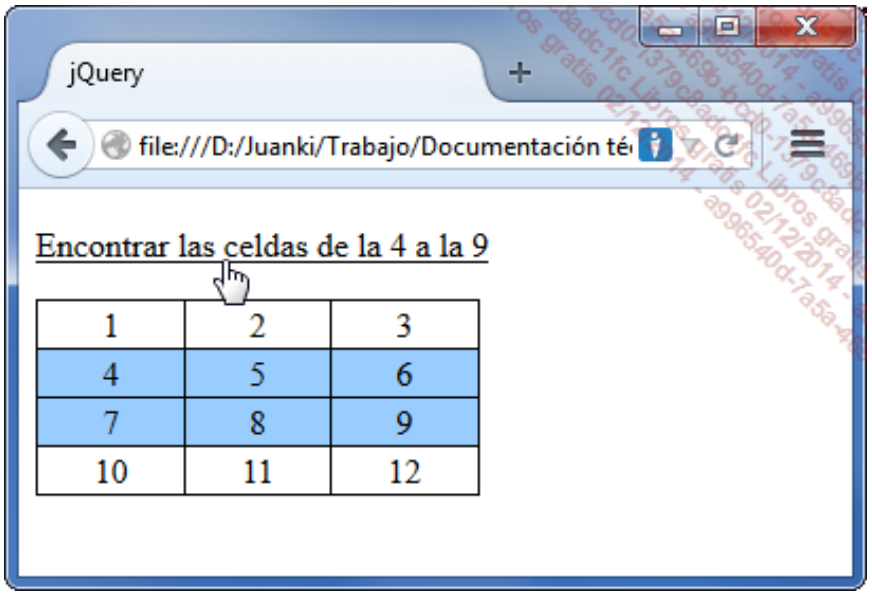
```
$("div").slice(4, 6).css("background", "yellow");
```

El método devuelve un objeto jQuery.

Ejemplo

Volvamos a la tabla de cuatro filas y tres columnas. Vamos a rellenar con un color de fondo las celdas de la 4 a la 9 (es decir, la segunda y la tercera fila).





```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
a { color: black;}
.azul { background-color: #9cf;}
table { width: 210px;
        border-collapse: collapse;
        border: 1px solid black;}
td { text-align: center;
     border: 1px solid black;}
</style>
</head>
<body>
<p><a href="#">Encontrar las celdas de la 4 a la 9</a></p>
<table>
<tr><td>1</td><td>2</td><td>3</td></tr>
<tr><td>4</td><td>5</td><td>6</td></tr>
<tr><td>7</td><td>8</td><td>9</td></tr>
<tr><td>10</td><td>11</td><td>12</td></tr>
</table>
</body>
</html>
```

El script jQuery:

```
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("a").click(function(){
$("td").slice(3,9).addClass("azul");
});
});
</script>
```

Explicaciones paso a paso:

```
$(document).ready(function(){
$("a").click(function(){
```

Después de la carga y al hacer clic en el enlace.

```
$("td").slice(3,9).addClass("azul");
```

El script selecciona las diferentes celdas (`$("td")`). Extrae la secuencia desde la posición 3 (es decir, la celda 4) hasta la posición 9 no incluida (es decir, hasta la celda 10 no incluida por lo tanto hasta la celda 9).

```
});
});
```

Fin del script.

El archivo final:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
```

```
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("a").click(function(){
$("td").slice(3,9).addClass("azul");
});
});
</script>
<style>
a { color: black;}
.azul { background-color: #9cf;}
table { width: 210px;
        border-collapse: collapse;
        border: 1px solid black;}
td { text-align: center;
     border: 1px solid black;}
</style>
</head>
<body>
<p><a href="#">Encontrar las celdas de la 4 a la 9</a></p>
<table>
<tr><td>1</td><td>2</td><td>3</td></tr>
<tr><td>4</td><td>5</td><td>6</td></tr>
<tr><td>7</td><td>8</td><td>9</td></tr>
<tr><td>10</td><td>11</td><td>12</td></tr>
</table>
</body>
</html>
```

Comentarios

```
$("td").slice(3).addClass("azul");
```

Si no se indica la posición final, la secuencia empieza en la posición de inicio hasta el final de la selección.

1	2	3
4	5	6
7	8	9
10	11	12

```
$("td").slice(3,4).addClass("azul");
```

Esta formulación solo devuelve un elemento de la secuencia.

1	2	3
4	5	6
7	8	9
10	11	12

```
$("td").slice(-2).addClass("azul");
```

Un valor negativo indica que la selección empieza al final de la secuencia. El valor -2 significa que devuelve dos elementos comenzando desde el final.

1	2	3
4	5	6
7	8	9
10	11	12

```
$("td").slice(2,-1).addClass("azul");
```

El código `slice(2,-1)` devuelve una secuencia desde el tercer elemento hasta el penúltimo.

1	2	3
4	5	6
7	8	9
10	11	12

```
$("td").slice(0).addClass("azul");
```

De esta manera, se devuelven todos los elementos en la secuencia.

1	2	3
4	5	6
7	8	9
10	11	12

Encontrar un elemento según un criterio

is(expresión)

Indica si la selección responde al criterio que se especifica en el argumento expresión. Devuelve `true` o `false`.

- "expresión" (cadena de caracteres): expresión que corresponde al criterio que se va a comprobar.

```
$(":checkbox").parent().is("form")
```

El método devuelve un booleano.

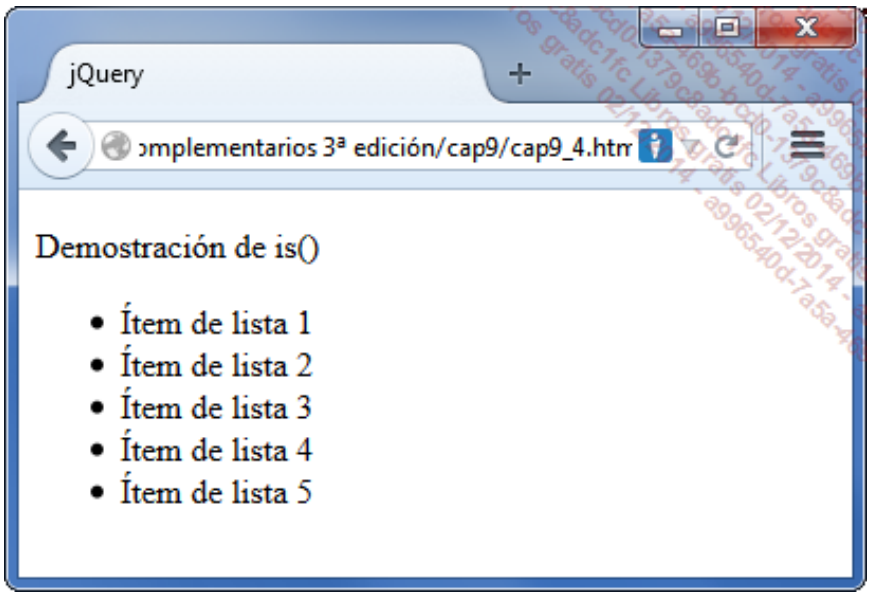
A partir de la versión 1.6 de jQuery, un elemento del DOM o un objeto jQuery también puede servir de selector.

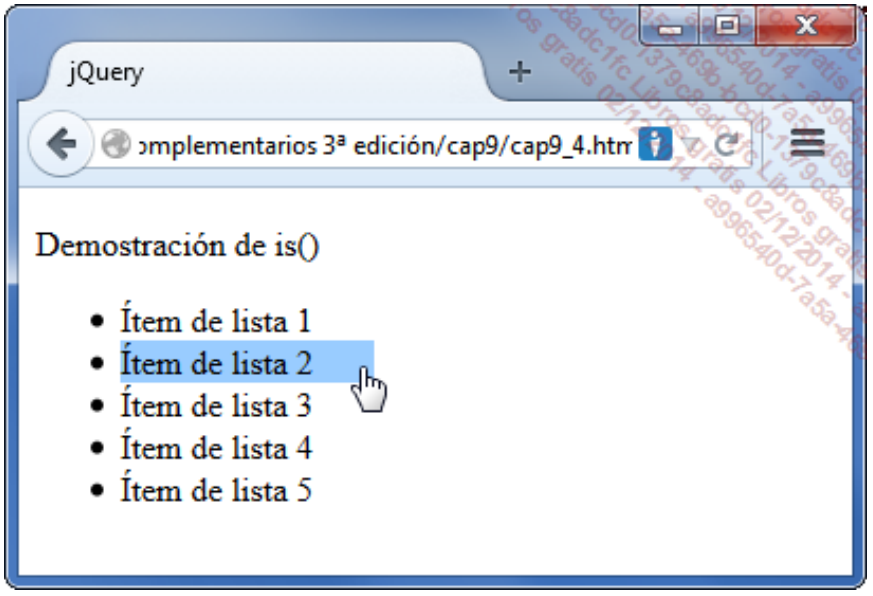
```
$("p").is(document.getElementById("unico"))
$("p").is( $("#único"))
```

Compruebe si la selección es el elemento identificado por el id="unico".

Ejemplo

Supongamos una lista de cinco elementos. Al hacer clic en un elemento de la lista, se le añade un color de fondo.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
.highlight { background-color: #9cf;}
a { color: black;}
li { width: 120px;
     cursor: pointer;}
</style>
</head>
<body>
<p>Demostración de is()</p>
<ul id="ejemplo">
<li>Ítem de lista 1</li>
<li>Ítem de lista 2</li>
<li>Ítem de lista 3</li>
<li>Ítem de lista 4</li>
<li>Ítem de lista 5</li>
</ul>
</body>
</html>
```

El script jQuery:

```
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("ul").click(function(event){
if ($(event.target).is("li")) {
$(event.target).addClass('highlight');
}
});
});
</script>
```

Lo detallamos a continuación.

```
$(document).ready(function(){
$("ul").click(function(event){
```

Cuando se carga el DOM y al hacer clic en un elemento de la lista no ordenada.

```
if ($(event.target).is("li")) {
$(event.target).addClass('highlight');
}
```

Si el clic (`event.target`) se hace sobre un elemento `<li>` de la lista, éste se resalta usando un color de fondo (`addClass('highlight')`).

```
});
});
```

Fin del script.

El archivo completo es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("ul").click(function(event){
if ($(event.target).is("li")) {
$(event.target).addClass('highlight');
}
});
});
</script>
<style>
.highlight { background-color: #9cf;}
a { color: black;}
li { width: 120px;
     cursor: pointer;}
</style>
</head>
<body>
<p>Demostración de is()</p>
<ul id="ejemplo">
<li>Ítem de lista 1</li>
<li>Ítem de lista 2</li>
<li>Ítem de lista 3</li>
<li>Ítem de lista 4</li>
<li>Ítem de lista 5</li>
</ul>
</body>
</html>
```

Reducir al primer elemento

first()

Reduce un conjunto de elementos seleccionados al primer elemento.

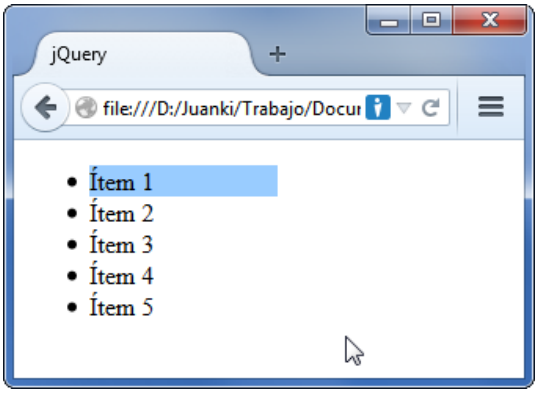
```
$("p span").first()
```

Este método devuelve un objeto jQuery

El método `first()` está disponible desde la versión 1.4 de jQuery.

Ejemplo

Reducimos el conjunto de ítems de una lista al primer elemento.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
.style{ background-color: #9cf;}
li { width: 120px;}
</style>
<script src="jquery.js"></script>
<script>
$(document).ready(function() {
$("li").first().addClass("style");
});
</script>
</head>
<body>
<ul>
<li>Ítem 1</li>
<li>Ítem 2</li>
<li>Ítem 3</li>
<li>Ítem 4</li>
<li>Ítem 5</li>
</ul>
</body>
</html>
```

```
$("li").first().addClass("style");
```

Del conjunto de los elementos de la lista <li> (`$("li")`), recuperar el primer elemento (`first()`) y añadirle la clase estilo (`addClass("style")`).

Reducir al último elemento

last()

Reduce un conjunto de elementos seleccionados al último elemento.

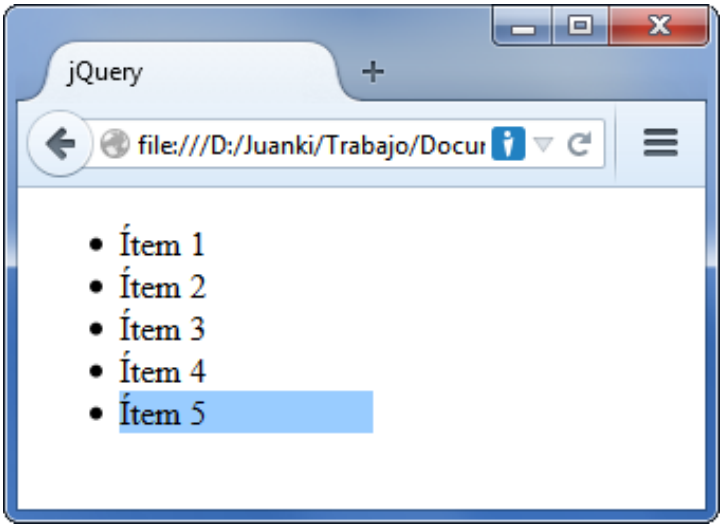
```
$("p span").last()
```

Este método devuelve un objeto jQuery.

El método `last()` está disponible desde la versión 1.4 de jQuery.

Ejemplo

Reducimos el conjunto de elementos de una lista al último ítem.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
.style{ background-color: #9cf;}
li { width: 120px;}
</style>
<script src="jquery.js"></script>
<script>
$(document).ready(function() {
$('li').last().addClass('style');
});
</script>
</head>
<body>
<ul>
<li>Ítem 1</li>
<li>Ítem 2</li>
<li>Ítem 3</li>
<li>Ítem 4</li>
<li>Ítem 5</li>
</ul>
</body>
</html>
```

Explicaciones:

```
$("li").last().addClass("style");
```

Del conjunto de los elementos de una lista <li> (`$("li")`), recuperar el último elemento (`last()`) y añadirle la clase estilo (`addClass("style")`).

Reducir a un elemento concreto

has(selector)

Reduce el conjunto de elementos seleccionados a los descendientes que corresponden al selector.

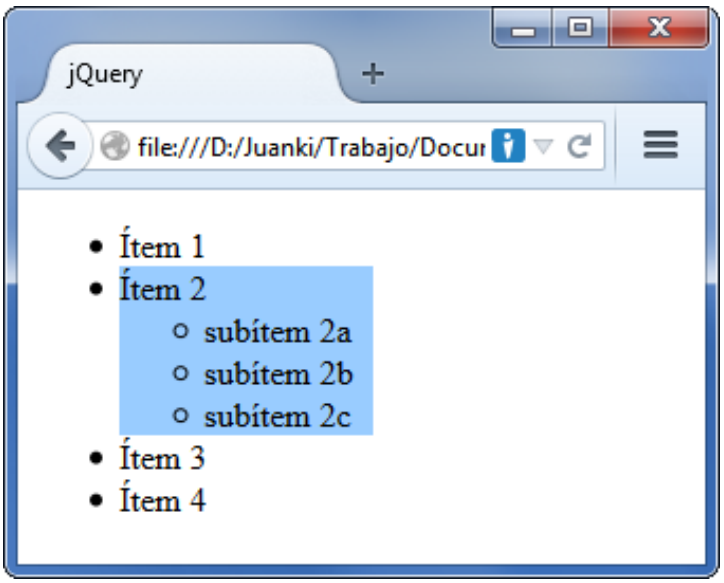
```
$("li").has("ul")
```

Este método devuelve un objeto jQuery.

El método `has()está` disponible desde la versión 1.4 de jQuery.

Ejemplo

Supongamos dos listas anidadas. Añadimos un color de fondo al ítem que contiene una anidación.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
.style{ background-color: #9cf;}
li { width: 120px;}
</style>
<script src="jquery.js"></script>
<script>
$(document).ready(function() {
$('li').has('ul').addClass('style');
});
</script>
</head>
<body>
<ul>
<li>Ítem 1</li>
<li>Ítem 2
<ul>
<li>subítem 2a</li>
<li>subítem 2b</li>
<li>subítem 2c</li>
</ul>
</li>
<li>Ítem 3</li>
<li>Ítem 4</li>
```

```
</ul>
</body>
</html>
```

Explicaciones:

```
$("li").has("ul").addClass("style");
```

Del conjunto de los elementos de la lista <li> (`$("li")`), recuperar el elemento que tiene una etiqueta <ul> entre sus descendientes (`has("ul")`) y añadirle la clase `style` (`addClass("style")`). En nuestro ejemplo, solo el ítem 2 cumple esta condición.

Eliminar un elemento

La función `not()` se usa para eliminar un elemento de un conjunto de elementos seleccionados.

not(selector o expresión)

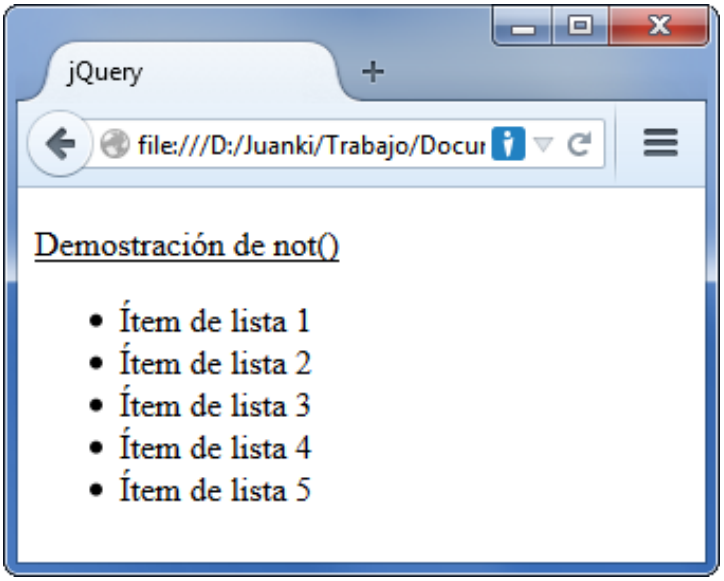
Elimina de la selección el elemento que responde a la expresión especificada.

```
$("p").not("#selected")
```

Este método devuelve un objeto jQuery.

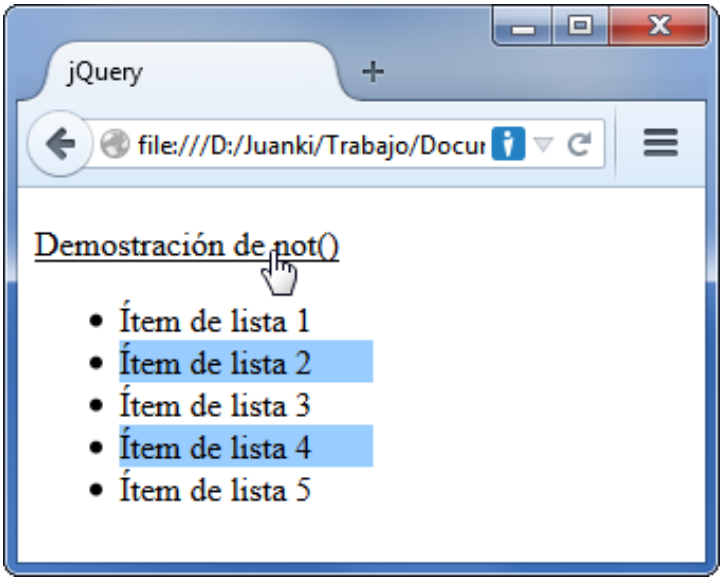
Ejemplo

Supongamos una lista no ordenada de cinco elementos.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
.highlight { background-color: #9cf;}
a { color: black;}
li { width: 120px;}
</style>
</head>
<body>
<p><a href="#">Demostración de not()</a></p>
<ul id="ejemplo">
<li>Ítem de lista 1</li>
<li>Ítem de lista 2</li>
<li>Ítem de lista 3</li>
<li>Ítem de lista 4</li>
<li>Ítem de lista 5</li>
</ul>
</body>
</html>
```

El script jQuery va a seleccionar las líneas pares y a eliminar de la selección las impares.




```
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("a").click(function(){
$('#ejemplo li').not(':even').addClass('highlight');
});
});
</script>
```

Explicaciones.

```
$(document).ready(function(){
$("a").click(function(){
```

Después de la carga y al hacer clic en el enlace.

```
$('#ejemplo li').not(':even').addClass('highlight');
```

El script elimina de la selección los elementos cuyo índice es un número par (`not(':even')`) de la lista no ordenada (`$('#ejemplo li')`). A los elementos que quedan, se les aplica un color de fondo, es decir, los elementos cuyo índice es un número impar. Recuerde que el índice en JavaScript empieza en 0.

```
});
});
```

Fin del script.

El archivo final es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("a").click(function(){
$('#ejemplo li').not(':even').addClass('highlight');
});
});
```

```
</script>
<style>
.highlight { background-color: #9cf;}
a { color: black;}
li { width: 120px;}
</style>
</head>
<body>
<p><a href="#">Demostración de not()</a></p>
<ul id="ejemplo">
<li>Ítem de lista 1</li>
<li>Ítem de lista 2</li>
<li>Ítem de lista 3</li>
<li>Ítem de lista 4</li>
<li>Ítem de lista 5</li>
</ul>
</body>
</html>
```

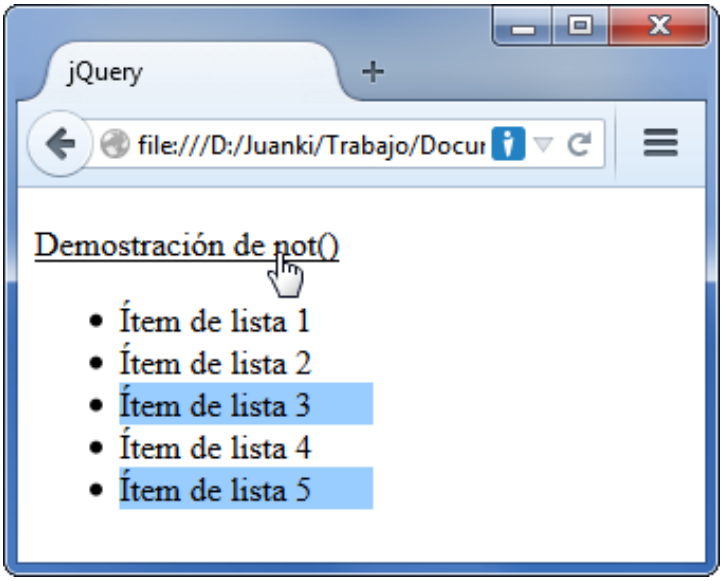
Comentario:

Es posible combinar las expresiones. Por ejemplo:

```
$('#ejemplo li').not(':odd,:first').addClass('highlight');
```

De esta manera, con el código anterior, el script elimina de la selección el primer elemento y aquéllos cuyo índice es impar.

El resultado es:



Formar una tabla (Array) de elementos

map(Función de callback)

Devuelve una tabla de elementos (`Array`) como resultado de una acción en un conjunto de elementos. Cada línea de la tabla es el resultado de la función que se aplica a un elemento.

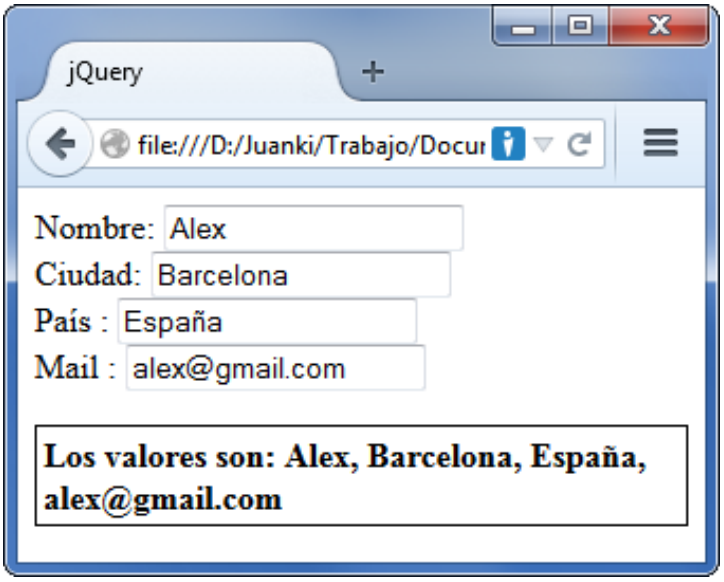
Función de callback: función que se aplica a los elementos de destino.

```
map(function(){ return $(this).val();})
```

El método devuelve un objeto jQuery.

Ejemplo

Creamos una tabla de tipo `Array`, con los valores de las diferentes líneas de texto de un formulario.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
p { font-weight: bold;
    border: 1px solid black;
    padding: 3px; }
</style>
</head>
<body>
<form action="">
Nombre: <input type="text" name="nombre" value="Alex"><br>
Ciudad: <input type="text" name="ciudad" value="Barcelona"><br>
País: <input type="text" name="país" value="España"><br>
Mail: <input type="text" name="mail" value="alex@gmail.com">
</form>
<p><b>Los valores son: </b></p>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
```

```
$("p").append($("input").map(function(){
return $(this).val();
})
.get().join(", ") );
});
</script>
```

Lo detallamos a continuación:

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

Cuando se carga el DOM.

```
$("p").append($("input").map(function(){
return $(this).val();
})
.get().join(", ") );
```

El script inserta en el párrafo `<p>` (`$("p").append`) los valores que devuelve el método `map()`, es decir, el valor (`val()`) de las diferentes etiquetas `<input>` del formulario.

El método jQuery `get()` permite acceder a todos los elementos del formulario. Para terminar, el método JavaScript clásico `join()` convierte la tabla `Array` en una cadena de caracteres con todos los elementos, separados por comas.

```
});
```

Fin del script.

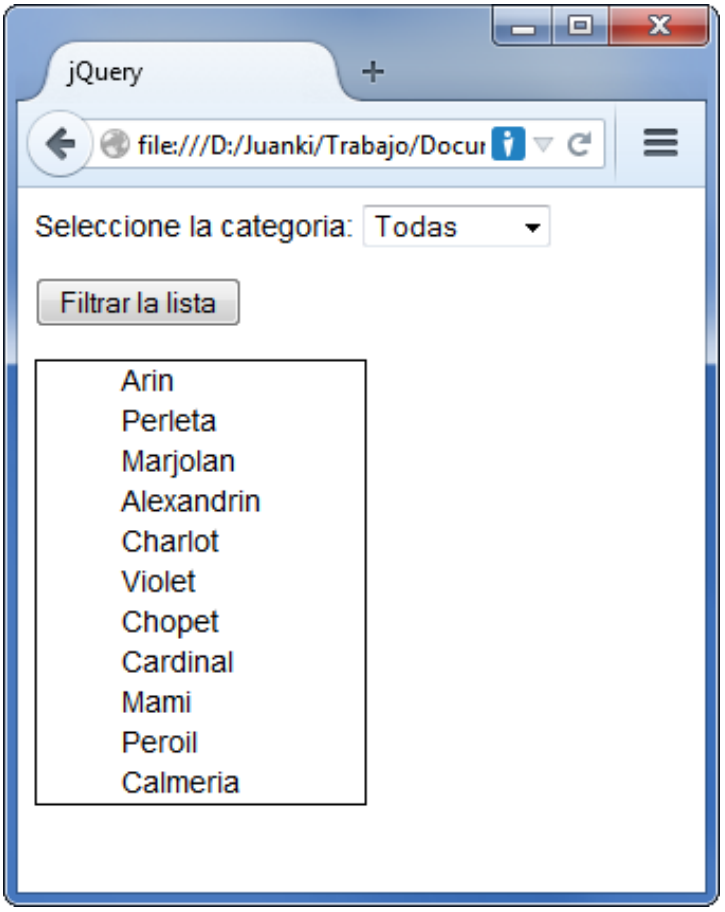
El código final es el siguiente:

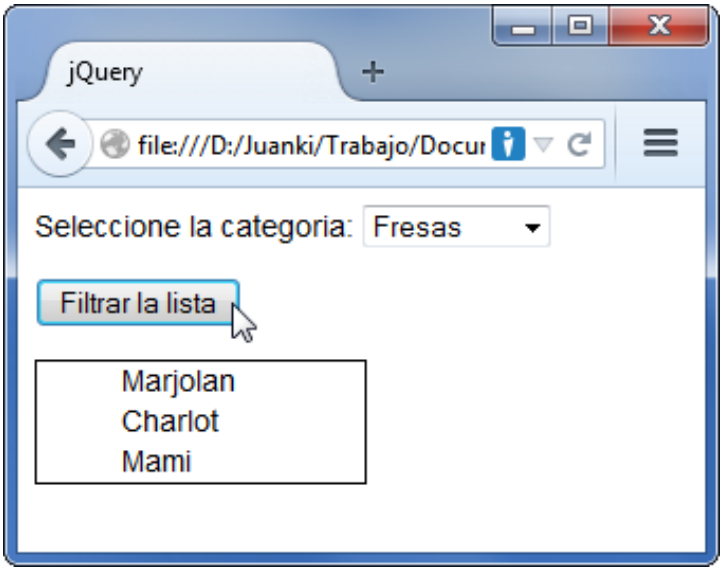
```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("p").append( $("input").map(function(){
return $(this).val();
})
.get().join(", ") );
});
</script>
<style>
p { font-weight: bold;
    border: 1px solid black;
    padding: 3px; }
</style>
</head>
<body>
<form action="">
Nombre: <input type="text" name="nombre" value="Alex"><br>
Ciudad: <input type="text" name="ciudad" value="Barcelona"><br>
País: <input type="text" name="país" value="España"><br>
Mail: <input type="text" name="mail" value="alex@gmail.com"/>
</form>
<p><b>Los valores son: </b></p>
</body>
</html>
```

Aplicaciones

1. Filtrar una lista

Supongamos una lista de frutas de diferentes tipos. Cuando se selecciona un elemento de formulario de tipo `<select>`, el script solo mantendrá un tipo de fruta concreta (manzana, pera, uva o fresa).





El archivo de inicio es.

```
<!doctype html>
<html lang="es">
```

```
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
body { font-family: Arial;
       font-size: 0.9em;
ul { list-style-type: none;
     width: 8em;
     border: 1px solid black;}
li { padding: 5px;}
</style>
</head>
<body>
Seleccione la categoría: <select id="categoria">
<option selected="selected" value="todos">Todos</option>
<option value="manzana">Manzanas</option>
<option value="pera">Peras</option>
<option value="uva">Uvas</option>
<option value="fresa">Fresas</option>
</select>
<p><input id="boton" type="button" value="Filtrar la lista"></p>
<ul>
<li class="item manzana">Arin</li>
<li class="item uva">Perleta</li>
<li class="item fresa">Marjolan</li>
<li class="item pera">Alexandrín</li>
<li class="item fresa">Charlot</li>
<li class="item manzana">Violet</li>
<li class="item manzana">Chopet</li>
<li class="item uva">Cardinal</li>
<li class="item fresa">Mami</li>
<li class="item pera">Peroil</li>
<li class="item uva">Calmeria</li>
</ul>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$("#boton").click(function(){
var seleccion = $("#categoría").val();
if (seleccion == "all"){
$("li").filter(".item").show();
}
else {
$("li").filter(".item").hide();
$("li").filter("."+seleccion).show();
}
});
});
</script>
```

Explicaciones.

```
$(document).ready(function(){
$("#boton").click(function(){
```

Después de la carga y al hacer clic en el botón **Filtrar la lista**.

```
var seleccion = $("#categoria").val();
```

El script carga en la variable `selection` el valor que ha seleccionado el usuario de la lista de

selección.

```
if (seleccion == "all"){
$("li").filter(".item").show();
}
```

Si se selecciona `Todos`, se muestran todos los elementos de la lista no ordenada según un filtro que tiene la clase `item`.

```
else {
$("li").filter(".item").hide();
$("li").filter("."+seleccion).show();
}
```

Si se selecciona otro valor de la lista de selección, en primer lugar se oculta la lista completa de los ítems. Después, los elementos de la lista no ordenada se filtran según el criterio almacenado en la variable `seleccion` y se muestran.

```
});
});
```

Fin del script.

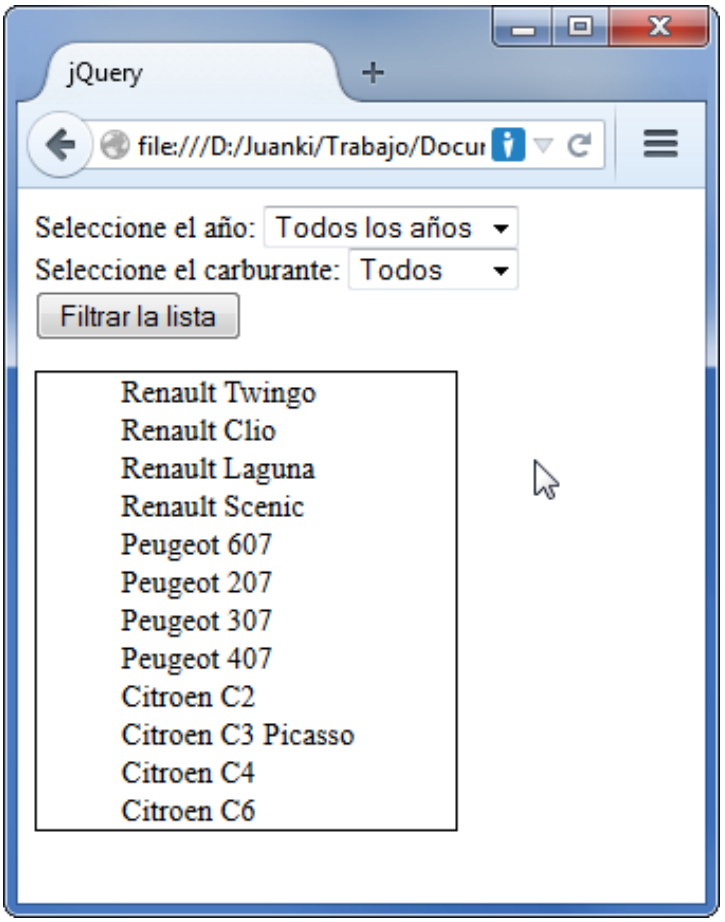
El código final es el siguiente:

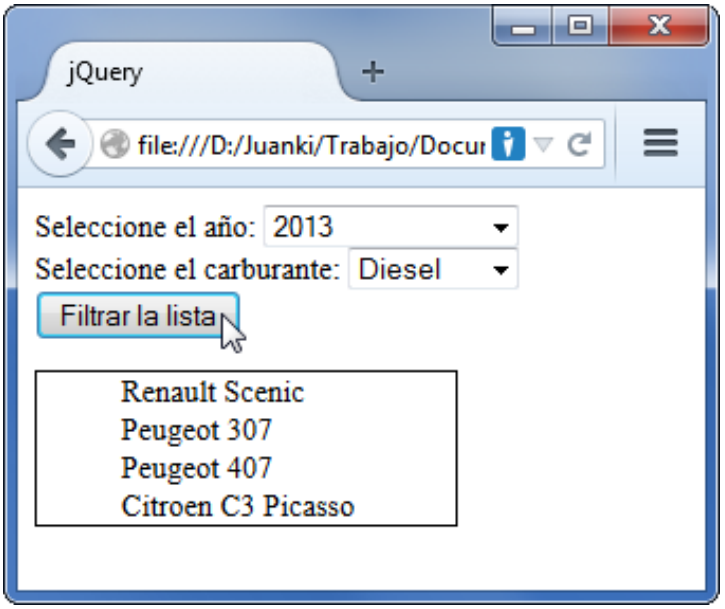
```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
body { font-family: Arial;
       font-size: 0.9em;}
ul { list-style-type: none;
     width: 8em;
     border: 1px solid black;}
li { padding: 5px;}
</style>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#boton").click(function(){
var seleccion = $("#categoria").val();
//show all items
if (seleccion == "all"){
$("li").filter(".item").show();
}
else {
$("li").filter(".item").hide();
$("li").filter("."+seleccion).show();
}
});
});
</script>
</head>
<body>
Seleccione la categoría: <select id="categoria">
<option selected="selected" value="todas">Todos</option>
<option value="manzana">Manzanas</option>
<option value="pera">Peras</option>
<option value="uva">Uvas</option>
<option value="fresa">Fresas</option>
</select>
```

```
<p><input id="boton" type="button" value="Filtrar la lista"></p>
<ul>
<li class="item manzana">Arin</li>
<li class="item uva">Perleta</li>
<li class="item fresa">Marjolan</li>
<li class="item pera">Alexandrín</li>
<li class="item fresa">Charlot</li>
<li class="item manzana">Violet</li>
<li class="item manzana">Chopet</li>
<li class="item uva">Cardinal</li>
<li class="item fresa">Mami</li>
<li class="item pera">Peroil</li>
<li class="item uva">Calmeria</li>
</ul>
</body>
</html>
```

2. Filtrar una lista según dos criterios

Esta aplicación se parece a la anterior, pero esta vez se utilizan dos criterios de selección. Supongamos una lista de coches. Una selección incluirá el año (2012, 2013, 2014) y otra el tipo de carburante (gasolina o diesel).






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
body { font-family: Arial sans-serif;
       font-size: 0.9em;}
ul { list-style-type: none;
     width: 11em;
     border: 1px solid black;}
li { padding: 5px;}
</style>
</head>
<body>
Seleccione el año: <select id="año">
<option selected="selected" value="todos">Todos
los años</option>
<option value="2014">2014</option>
<option value="2013">2013</option>
<option value="2012">2012</option>
</select>
<br>
Seleccione el carburante: <select id="carburante">
<option selected="selected" value="todos" >Todos
los carburantes </option>
<option value="gasolina">Gasolina</option>
<option value="diesel">Diesel</option>
</select>
<br>
<input id="boton" type="button" value="Filtrar la lista">
<ul>
<li class="item 2014 gasolina">Renault Twingo</li>
<li class="item 2013 gasolina">Renault Clio</li>
<li class="item 2012 diesel">Renault Laguna</li>
<li class="item 2013 diesel">Renault Scenic </li>
<li class="item 2012 diesel">Peugeot 607</li>
<li class="item 2014 gasolina">Peugeot 207</li>
<li class="item 2013 diesel">Peugeot 307</li>
<li class="item 2013 diesel">Peugeot 407</li>
<li class="item 2012 gasolina">Citroen C2</li>
<li class="item 2013 diesel">Citroen C3 Picasso</li>
<li class="item 2012 diesel">Citroen C4</li>
<li class="item 2014 diesel">Citroen C6 </li>
```

```
</ul>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$("#boton").click(function(){
var año = $("#año").val();
var carburante = $("#carburante").val();
var selector_año = '';
var selector_carburante = '';
if (año == "todas" && carburante == "todos"){
$(".item").show();
}
else {
if (carburante != "todos"){
selector_carburante = '.' + carburante
}
if (año != "todos")
{
selector_año = '.' + año
}
$(".item").hide();
$("li"). filter(selector_carburante + selector_año).show();
}
});
});
</script>
```

Explicaciones.

```
$(document).ready(function(){
$("#boton").click(function(){
```

Después de la carga y al hacer clic en el botón.

```
var año = $("#año").val();
var carburante = $("#carburante").val();
var selector_año = '';
var selector_carburante = '';
```

Creamos una serie de variables. La variable `año` toma el valor que se selecciona de la primera lista de selección. La variable `carburante`, el de la segunda. Las variables `selector_año` y`selector_carburante` contendrán (como su propio nombre indica) los selectores de clase del filtrado.

```
if (año == "todas" && carburante == "todos"){
$(".item").show();
}
```

Si el valor de selección de la lista de años es `Todos` y también el de la lista de carburantes, se muestran todos los elementos de la lista no ordenada (`.item`).

```
else {
if (carburante != "todos"){
selector_carburante = '.' + carburante
}
```

En caso de que no sea así, el script comprueba si la lista de selección relativa al carburante no está en `Todos`. En este caso, la variable `selector_carburante` tendrá un punto (para la clase) y el tipo

de carburante devuelto.

```
if (año != "todos"){
selector_año = '.' + año
}
```

De manera análoga, si la lista de selección de los años no está colocada en `Todos`, la variable`selector_año` contendrá la clase con el año devuelto.

```
$(".item").hide();
$(selector_carburante + selector_año).show();
```

Se oculta la lista completa de los ítems de la lista no ordenada. Y se muestra la lista filtrada según el carburante y el año devueltos.

```
}
```

Fin del else.

```
});
});
```

Fin del script jQuery.

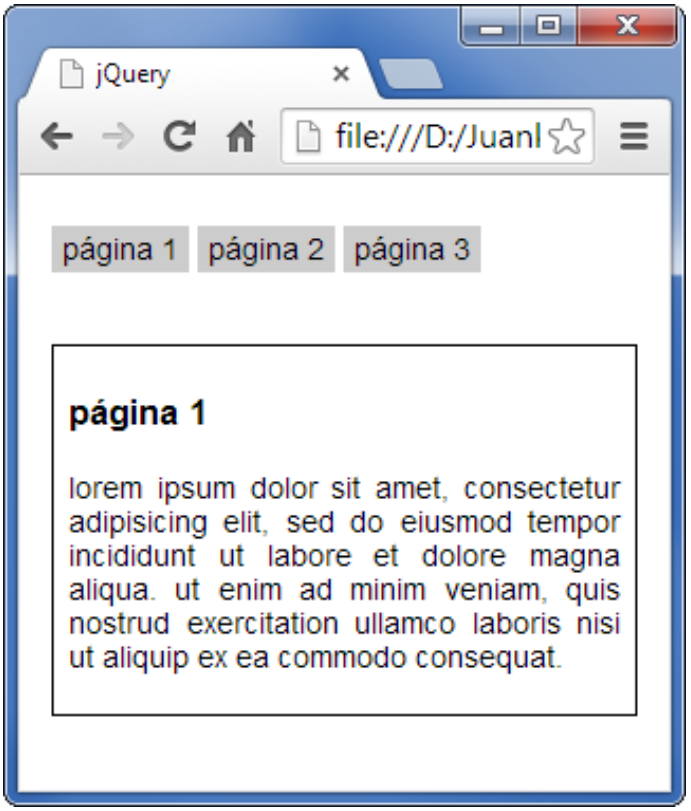
El archivo final:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
body { font-family: Arial sans-serif;
       font-size: 0.9em;}
ul { list-style-type: none;
     width: 11em;
     border: 1px solid black;}
li { padding: 5px;}
</style>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#boton").click(function(){
var año = $("#año").val();
var carburante = $("#carburante").val();
var selector_año = '';
var selector_carburante = '';
if (año == "todos" && carburante == "todos"){
$(".item").show();
}
else {
if (carburante != "todos"){
selector_carburante = '.' + carburante
}
if (año != "todos")
{
selector_año = '.' + año
}
$(".item").hide();
$("li"). filter(selector_carburante + selector_año).show();
}
});
});
```

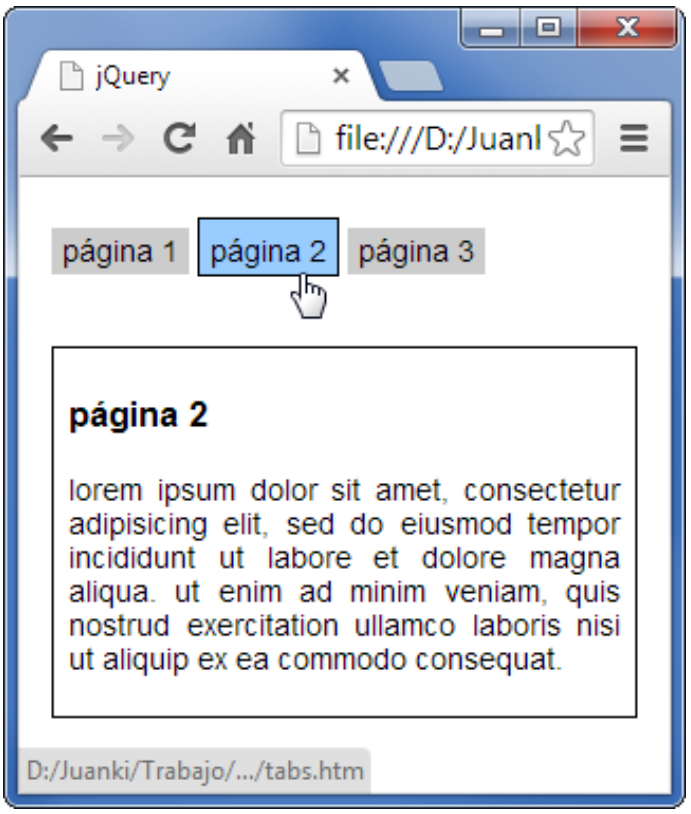
```
//]]>
</script>
</head>
<body>
Seleccione el año: <select id="año">
<option selected="selected" value="todos">Todos los
años</option>
<option value="2014">2014</option>
<option value="2013">2013</option>
<option value="2012">2012</option></select>
<br />
Seleccione el carburante: <select id="carburante">
<option selected="selected" value="todos" >Todos los carburantes
</option>
<option value="gasolina">Gasolina</option>
<option value="diesel">Diesel</option>
</select>
<br />
<input id="boton" type="button" value="Filtrar la lista">
<ul>
<li class="item 2014 gasolina">Renault Twingo</li>
<li class="item 2013 gasolina">Renault Clio</li>
<li class="item 2012 diesel">Renault Laguna</li>
<li class="item 2013 diesel">Renault Scenic </li>
<li class="item 2012 diesel">Peugeot 607</li>
<li class="item 2014 gasolina">Peugeot 207</li>
<li class="item 2013 diesel">Peugeot 307</li>
<li class="item 2013 diesel">Peugeot 407</li>
<li class="item 2012 gasolina">Citroen C2</li>
<li class="item 2013 diesel">Citroen C3 Picasso</li>
<li class="item 2012 diesel">Citroen C4</li>
<li class="item 2014 diesel">Citroen C6 </li>
</ul>
</body>
</html>
```

3. Navegación por pestañas

Esta aplicación es muy frecuente en la Web 2.0. Dependiendo de la pestaña que se pulse, se muestra un contenido diferente en la página o, más precisamente, en una capa de ésta.



Al hacer clic en una pestaña, se muestra la página correspondiente.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
body { padding: 0px;
       margin: 15px;
```

```
       font: 90% arial, sans-serif;}
ul.navigation { padding: 0px;
                list-style-type: none;
                line-height: 40px;
                overflow: hidden;}
ul.navigation li { display: inline;}
ul.navigation li a { padding-top: 3px;
                     padding-bottom: 3px;
                     background-color: #ccc;
                     padding-left: 5px;
                     padding-right: 5px;
                     color: #000;
                     text-decoration: none;
                     line-height: 27px;
                     overflow: hidden;}
ul.navigation li a.selected { background-color: #9cf;
                              border: 2px solid black;
                              color: #000;
                              padding-top: 7px;}
ul.navigation li a:hover { background-color: #9cf;
                           color: #000;
                           padding-top: 7px;
                           border: 1px solid black;}
div.menu > div { padding: 5px;
                 margin-top: 3px;}
#un { margin-top: 25px;
      text-align: justify;
      border: 1px solid black;
      padding-left: 7px;
      padding-right: 7px;}
#dos { margin-top: 25px;
       text-align: justify;
       border: 1px solid black;
       padding-left: 7px;
       padding-right: 7px;}
#tres { margin-top: 25px;
        text-align: justify;
        border: 1px solid black;
        padding-left: 7px;
        padding-right: 7px;}
</style>
</head>
<body>
<div class="menu">
<ul class="navegacion">
<li><a href="tabs.htm#un">página 1</a></li>
<li><a href="tabs.htm#dos">página 2</a></li>
<li><a href="tabs.htm#tres">página 3</a></li></ul>
<div id="uno">
<h3>página 1</h3>
<p>lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
do eiusmod
tempor incididunt ut labore y dolore magna aliqua. ut enim ad
minim veniam, quis nostrud exercitación ullamco laboris nisi ut
aliquip ex ea commodo consequat.</p>
</div>
<div id="dos">
<h3>página 2</h3>
<p>lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
do eiusmod
tempor incididunt ut labore y dolore magna aliqua. ut enim ad
minim veniam, quis nostrud exercitación ullamco laboris nisi ut
aliquip ex ea commodo
consequat.</p>
</div>
<div id="tres">
```

```
<h3>página 3</h3>
<p>lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
do eiusmod
tempor incididunt ut labore y dolore magna aliqua. ut enim ad
minim veniam, quis nostrud exercitación ullamco laboris nisi ut
aliquip ex ea commodo consequat.</p>
</div>
</div>
</body>
</html>
```

El script jQuery:

Al hacer clic en una pestaña, se muestra la página que corresponda.

```
<script>
$(document).ready(function(){
var cajas_pagina = $('div.menu > div');
cajas_pagina.hide()
.filter(':first').show();
$('div.menu ul.navigation a').click(function () {
cajas_pagina.hide();
cajas_pagina.filter(this.hash).show();
$('div.menu ul.navigation a').removeClass('selected');
$(this).addClass('selected');
return false;
});
});
</script>
```

Explicaciones:

```
$(document).ready(function(){
var cajas_pagina = $('div.menu > div');
```

Cuando se carga la página, se cargan los elementos `<div>` hijos de `<div id="menu">` en la variable `cajas_pagina`.

```
cajas_pagina.hide()
.filter(':first').show();
```

En primer lugar, las diferentes páginas se ocultan para reiniciar el proceso. Sin embargo, se mostrará por defecto la primera página (`filter(':first')`).

```
$('div.menu ul.navigation a').click(function () {
cajas_pagina.hide();
cajas_pagina.filter(this.hash).show();
```

Al hacer clic en una pestaña, se oculta el contenido anterior (`hide()`) y el método de filtrado (`filter(this.hash)`) se aplica para devolver la página implicada y se muestra (`show()`). Se usa la propiedad JavaScript (clásica) `window.location.hash`, que permite recuperar el enlace de una URL.

```
$('div.menu ul.navigation a').removeClass('selected');
$(this).addClass('selected');
return false;
```

Se aplica la clase `selected` a la pestaña que ha seleccionado el usuario.

```
});
});
```

Fin del script.

El código final es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
body { padding: 0px;
       margin: 15px;
       font: 90% arial, sans-serif;}
ul.navigation { padding: 0px;
                list-style-type: none;
                line-height: 40px;
                overflow: hidden;}
ul.navigation li { display: inline;}
ul.navigation li a { padding-top: 3px;
                     padding-bottom: 3px;
                     background-color: #ccc;
                     padding-left: 5px;
                     padding-right: 5px;
                     color: #000;
                     text-decoration: none;
                     line-height: 27px;
                     overflow: hidden;}
ul.navigation li a.selected { background-color: #9cf;
                              border: 2px solid black;
                              color: #000;
                              padding-top: 7px;}
ul.navigation li a:hover { background-color: #9cf;
                           color: #000;
                           padding-top: 7px;
                           border: 1px solid black;}
div.menu > div { padding: 5px;
                 margin-top: 3px;}
#un { margin-top: 25px;
      text-align: justify;
      border: 1px solid black;
      padding-left: 7px;
      padding-right: 7px;}
#dos { margin-top: 25px;
       text-align: justify;
       border: 1px solid black;
       padding-left: 7px;
       padding-right: 7px;}
#tres { margin-top: 25px;
        text-align: justify;
        border: 1px solid black;
        padding-left: 7px;
        padding-right: 7px;}
</style>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
var cajas_pagina = $('div.menu > div');
cajas_pagina.hide()
.filter(':first').show();
$('div.menu ul.navigation a').click(function () {
cajas_pagina.hide();
cajas_pagina.filter(this.hash).show();
$('div.menu ul.navigation a').removeClass('selected');
$(this).addClass('selected');
return false;
});
});
```

```
</script>
</head>
<body>
<div class="menu">
<ul class="navegacion">
<li><a href="tabs.htm#un">página 1</a></li>
<li><a href="tabs.htm#dos">página 2</a></li>
<li><a href="tabs.htm#tres">página 3</a></li></ul>
<div id="uno">
<h3>página 1</h3>
<p>lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
do eiusmod tempor incididunt ut labore y dolore magna aliqua.
ut enim ad minim veniam, quis nostrud exercitación ullamco laboris
nisi ut aliquip ex ea commodo consequat.</p>
</div>
<div id="dos">
<h3>página 2</h3>
<p>lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
do eiusmod tempor incididunt ut labore y dolore magna aliqua.
ut enim ad minim veniam, quis nostrud exercitación ullamco laboris
nisi ut aliquip ex ea commodo consequat.</p>
</div>
<div id="tres">
<h3>página 3</h3>
<p>lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
do eiusmod tempor incididunt ut labore y dolore magna aliqua.
ut enim ad minim veniam, quis nostrud exercitación ullamco laboris
nisi ut aliquip ex ea commodo consequat.</p>
</div>
</div>
</body>
</html>
```

Introducción

En algunos años, AJAX (*Asynchronous JavaScript and XML*) se ha hecho un lugar indiscutible en el diseño de las páginas Web. Su aparición responde al concepto de Web 2.0 que, de alguna manera, ha sido el responsable por su enfoque innovador y por las técnicas nuevas que aporta.

La librería de jQuery tenía que contemplar el AJAX. Vamos a ver en este capítulo la extrema concisión del código y la facilidad que aporta jQuery en esta materia.

Todo nuestro estudio se ha desarrollado, hasta ahora, en el lado cliente, lo que resultaba muy práctico, ya que son suficientes un editor de texto y un navegador. Es este capítulo dedicado a AJAX es útil, si no imprescindible, la instalación de un servidor Web local, también llamado servidor Web personal, para probar el código sin pasar por el procedimiento obligatorio de descarga por FTP. Como servidor Web local, Microsoft IIS o EasyPHP son las soluciones más fiables y fáciles de aplicar.

Las consultas AJAX abreviadas

1. Cargar un archivo

El método `load()` permite cargar de un archivo de una manera muy sencilla, siguiendo el procedimiento que aplica AJAX.

Más adelante detallaremos el método `ajax()` (ver la sección La consulta AJAX completa, en este capítulo), que permite efectuar la misma operación de manera más sofisticada y detallada. Para el desarrollador principiante o para las aplicaciones sencillas, el método `load()` es suficiente y, gracias a su sencillez, hace las delicias de los diseñadores.

load(url[, datos,][, función])

Carga el código Html (o Xhtml) a partir de un archivo dado y lo sitúa en el elemento seleccionado.

- "url": una cadena de caracteres que contiene la URL del archivo Html que se quiere cargar.
- "datos" (opcional): lista de pares en forma de clave/valor que se enviarán como datos al servidor.
- "función" (opcional): la función que se debe ejecutar si la consulta tiene éxito. Por defecto, los datos se cargan en el elemento seleccionado.

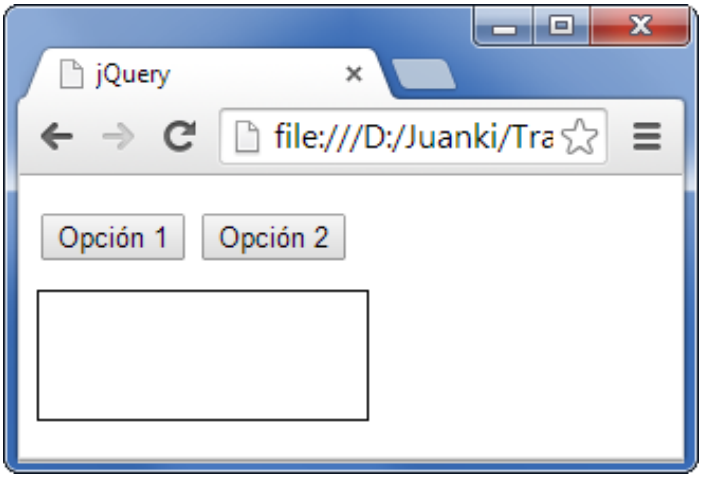
`$("div").load("test.htm")`: carga los datos del archivo test.htm y los ubica en la capa `<div>`.

Este método devuelve un objeto jQuery.

Se devuelve el método GET por defecto, salvo si los datos se proporcionan como argumento.

Ejemplo

Cargar en una misma capa contenidos diferentes usando dos botones.



El archivo Html de inicio se presenta de la siguiente manera:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
button { margin-top: 10px;}
div { width: 145px;
```

```
      height: 50px;
      border: 1px solid black;
      margin-top: 12px;
      padding: 5px;
      text-align: center;}
</style>
</head>
<body>
<button id="boton1">Opción 1</button>
<button id="boton2">Opción 2</button>
<div>
</div>
</body>
</html>
```

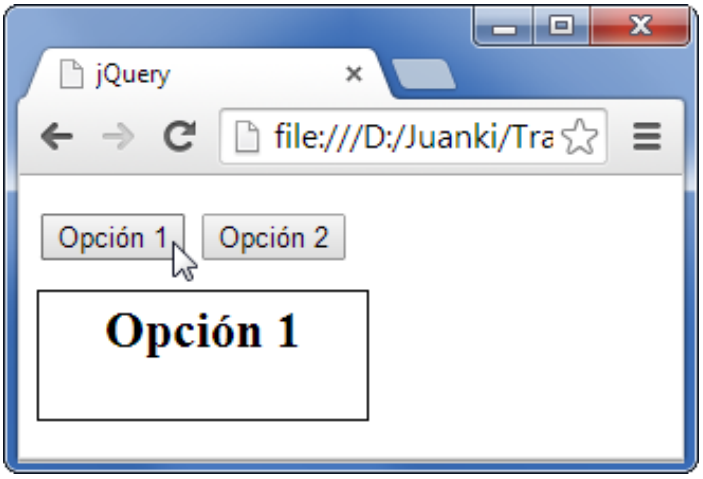
El archivo opcion1.htm, situado en el servidor, tiene simplemente el código Html siguiente:

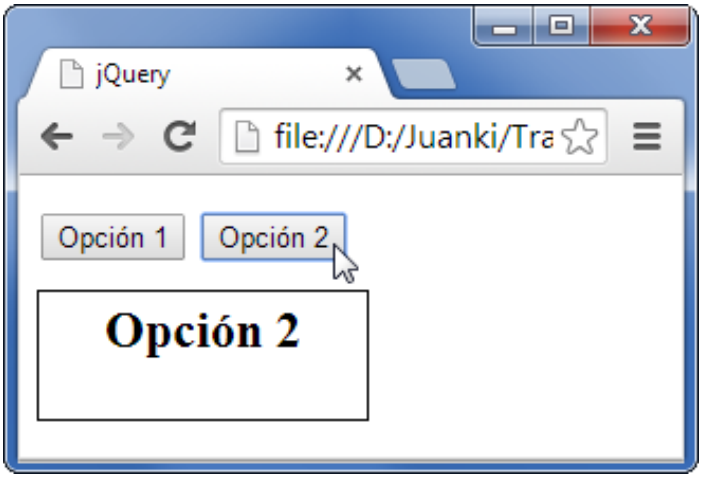
```
<b><font size="5">Opción 1</font></b>
```

y el archivo opcion2.htm:

```
<b><font size="5">Opción 2</font></b>
```

Al hacer clic en el botón 1, el script jQuery debe cargar en la capa `<div>` el archivo opcion1.htm y, al hacer clic en el botón 2, debe cargar el archivo opcion2.htm.





El script jQuery:

```
<script>
```

```
$(document).ready(function(){
$("#boton1").click(function() {
$("div").load("opcion1.htm");
});
$("#boton2").click(function() {
$("div").load("opcion2.htm");
});
});
</script>
```

Lo detallamos a continuación.

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

Inicialización de jQuery cuando se carga el DOM.

```
$("#boton1").click(function() {
$("div").load("opcion1.htm");
});
```

Al hacer clic en el botón 1 (`$("#boton1").click()`), se carga el archivo opcion1.htm (`load("opcion1.htm")`) en la capa `<div>`.

```
$("#boton2").click(function() {
$("div").load("opcion2.htm");
});
```

Al hacer clic en el botón 2 (`$("#boton2").click()`), se carga el archivo opcion2.htm (`load("opcion2.htm")`) en la capa `<div>`.

```
});
```

Fin del script.

El archivo final es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#boton1").click(function() {
$("div").load("opcion1.htm");
});
$("#boton2").click(function() {
$("div").load("opcion2.htm");
});
});
</script>
<style>
button { margin-top: 10px;}
div { width: 145px;
      height: 50px;
      border: 1px solid black;
      margin-top: 12px;
      padding: 5px;
      text-align: center;}
</style>
</head>
<body>
```

```
<button id="boton1">Opción 1</button>
<button id="boton2">Opción 2</button>
<div>
</div>
</body>
</html>
```

2. Solo cargar en caso de modificación

El método `loadIfModified()`, equivalente al método `load()` que hemos visto en el punto anterior, solo carga el archivo si se ha modificado desde la última consulta.

loadIfModified(url[, datos,][, función])

Carga el código Html a partir de un archivo dado y lo ubica en el elemento seleccionado si el fichero se ha modificado desde la última consulta.

- "url": una cadena de caracteres que contiene la URL del archivo Html que se va a cargar.
- "datos" (opcional): lista de pares en forma de clave/valor que se enviarán como datos al servidor.
- "función" (opcional): la función que se debe ejecutar si la consulta tiene éxito. Por defecto, los datos se cargan en el elemento seleccionado.

`$("div").loadIfModified("test.htm")`: carga los datos del archivo test.htm y solo los ubica en la capa `<div>` si el archivo ha cambiado desde la última consulta.

Este método devuelve un objeto jQuery.

3. Cargar siguiendo el método GET o POST

Otras formas abreviadas que ofrece jQuery para hacer consultas AJAX son los métodos `$.get()` y`$.post()`.

$.get(url[, datos][, función][, tipo])

Carga un archivo del servidor, según una consulta HTTP GET.

- "url": una cadena de caracteres que contiene la URL del archivo que se quiere cargar.
- "datos" (opcional): lista de pares en forma de clave/valor que se enviarán como datos al servidor.
- "función" (opcional): la función que se debe ejecutar si la consulta tiene éxito.
- "tipo" (opcional): cadena de caracteres que especifica el tipo de datos que se transmiten a la función: "xml", "html", "script", "json", "jsonp" o "text".

```
$.get("test.html", function(data){
$("#resultado").html(data);
});
```

o

```
$.get("test.cgi", {nombre: "Román", cuidad: "Tarragona"},
function(data){alert("Datos cargados: " + data);}
);
```

Este método devuelve un objeto XMLHttpRequest.

$.post(url[, datos][, función][, tipo])

Carga un archivo del servidor según una consulta HTTP POST.

- "url": una cadena de caracteres que contiene la URL del archivo que se quiere cargar.
- "datos" (opcional): lista de pares en forma de clave/valor que se enviarán como datos al servidor.
- "función" (opcional): la función que se debe ejecutar si la consulta tiene éxito.
- "tipo" (opcional): cadena de caracteres que especifica el tipo de datos que se transmiten a la función: "xml", "html", "script", "json", "jsonp" o "text".

```
$.post("test.html", function(data){
$("#resultado").html(data);
});
```

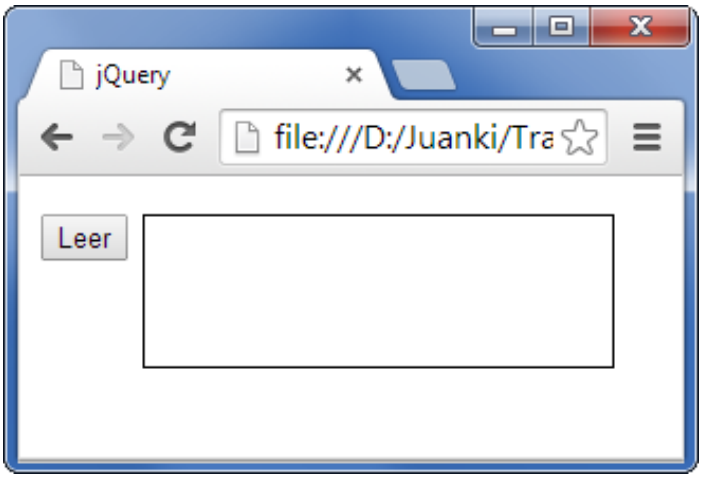
o

```
$.post("resultado.php", {nombre: "Carla"},
function(data){$("#box").html(data);}
);
```

Este método devuelve un objeto XMLHttpRequest.

Ejemplo

Al hacer clic en un botón, mostrar en una capa el contenido de un artículo. Se usa el método `$.get()`.



El archivo de inicio es.

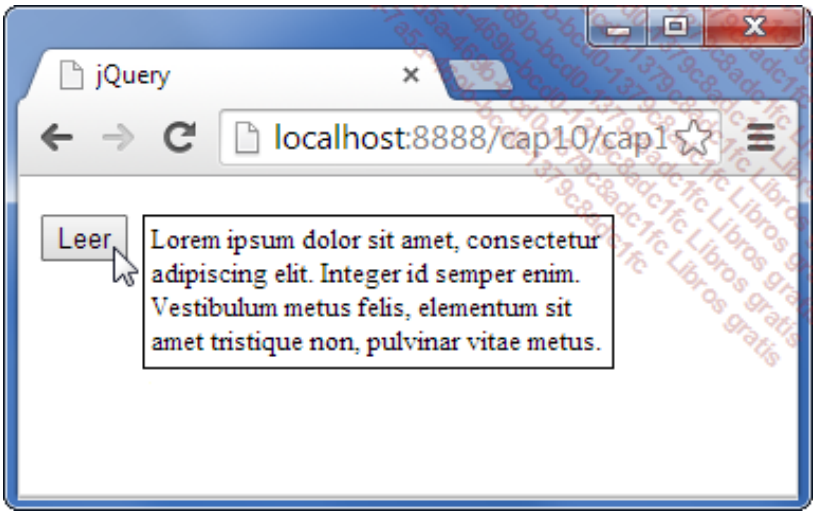
```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
button { margin-top: 10px;}
#izquierdo { width: 50px;
             float: left;}
#derecho { width: 215px;
           height: 65px;
           border: 1px solid black;
           margin-top: 10px;
           padding: 3px;
           font-size: 0.8em;
           float: left;}
</style>
</head>
<body>
<div id="izquierdo">
```

```
<button id="boton">Leer</button>
</div>
<div id="derecho">
</div>
</body>
</html>
```

El archivo que se carga en el servidor (lorem.htm), tiene el contenido siguiente:

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer
id semper enim. Vestibulum metus felis, elementum sit amet
tristica non, pulvinar vitae metus.
```

El script jQuery usa `$.get()` para cargar el archivo desde el servidor y lo muestra en la capa derecha, rodeada con un borde.




```
<script>
$(document).ready(function(){
$("#boton").on("click", function() {
$.get("lorem.htm", function(contenido){
$("#derecho").append(contenido);
});
});
});
</script>
```

Explicación del script:

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

Inicialización del DOM y de jQuery.

```
$("#boton").on("click", function() {
```

Al hacer clic en el botón.

```
$.get("lorem.htm", function(contenido){
$("#derecho").append(contenido);
});
```

La consulta se hace por `$.get()`, que pide el archivo lorem.htm. Después, una función añade (`append()`) el contenido del archivo en la capa identificada por `derecho`.

```
});
```

```
});
```

Fin del script.

Al final, el código de la página es:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#boton").on("click", function() {
$.get("lorem.htm", function(contenido){
$("#derecho").append(contenido);
});
});
});
</script>
<style>
button { margin-top: 10px;}
#izquierdo { width: 50px;
             float: left;}
#derecho { width: 215px;
           height: 65px;
           border: 1px solid black;
           margin-top: 10px;
           padding: 3px;
           font-size: 0.8em;
           float: left;}
</style>
</head>
<body>
<div id="izquierdo">
<button id="boton">Leer</button>
</div>
<div id="derecho">
</div>
</body>
</html>
```

Comentario

En paralelo a `loadIfModified( url[, datos,][, función])`, también existe el método`$.getIfModified( url[, datos][, función][, tipo])`. Su funcionamiento es idéntico.

Por el contrario, no existe la opción `IfModified` con `$.post()`, ya que con POST las páginas nunca van a la caché.

4. Cargar un script

$.getScript(url[, función])

Carga un script JavaScript del servidor usando el método HTTP GET y ejecutándolo.

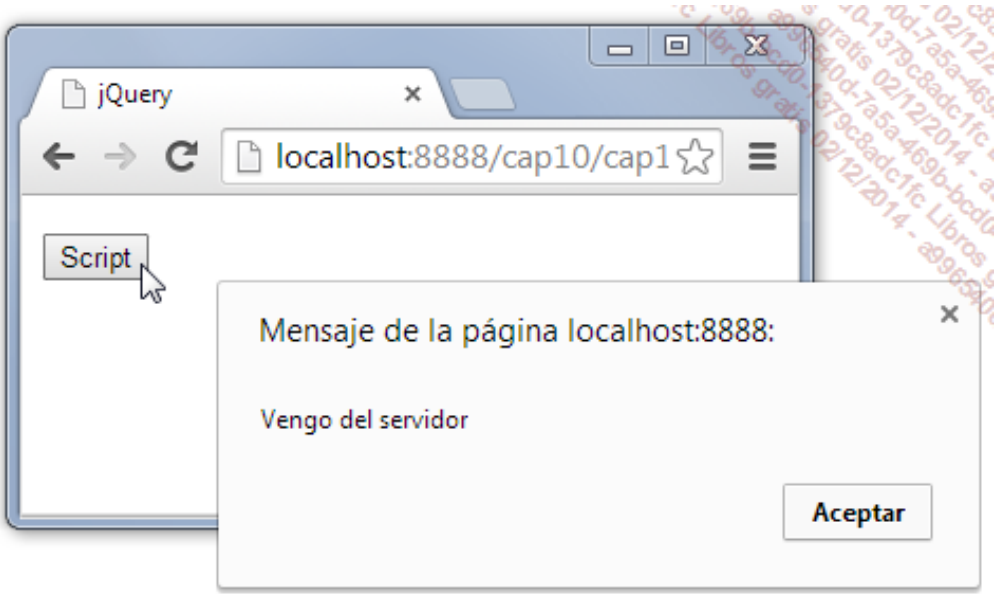
- "url": una cadena de caracteres que indica la dirección en la que se encuentra el script que desea cargar.
- "función" (opcional): una función que se debe ejecutar si la consulta tiene éxito. Normalmente, esta función no es necesaria, ya que el script se ejecuta automáticamente.

```
$.getScript("test.js");
```

Este método devuelve un objeto XMLHttpRequest.

Ejemplo

Mostrar un mensaje de alerta para indicar que el JavaScript se ha cargado desde el servidor.



El archivo de inicio es muy sencillo.

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
boton { margin-top: 10px;}
</style>
</head>
<body>
<button id="boton">Script</button>
</body>
</html>
```

El mensaje de alerta se halla en el archivo alerta.js.

```
alert("Vengo del servidor");
```

Al hacer clic en el botón, el script jQuery va a cargar el archivo alerta.js y a ejecutarlo.

```
<script>
$(document).ready(function(){
$("#boton").click(function() {
$.getScript("alerta.js");
});
});
</script>
```

El código final es el siguiente:

```
<!doctype html>
```

```
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#boton").click(function() {
$.getScript("alerta.js");
});
});
</script>
<style>
button { margin-top: 10px;}
</style>
</head>
<body>
<button id="boton">Script</button>
</body>
</html>
```

La consulta AJAX completa

Este método permite hacer una consulta AJAX controlando los diferentes argumentos y etapas, gracias a las numerosas opciones disponibles.

ajax(options)

Ejecuta una consulta HTTP asíncrona (AJAX).

```
$.ajax({
url: "test.htm",
success: function(data ) {
$("#resultado").html(data);
$().log("Terminada");
},
});
```

El método devuelve un objeto XMLHttpRequest.

Vamos a revisar las numerosas opciones disponibles.

- "`url`" (obligatoria): una cadena de caracteres que contiene la dirección de la consulta.
- "`type`" (opcional): una cadena de caracteres que define el método HTTP que se va a usar para la consulta (GET o POST). El valor por defecto es GET. Se pueden usar otros métodos de envío HTTP, como PUT o DELETE, pero no están soportados por todos los navegadores.
- "`dataType`" (opcional): una cadena de caracteres que especifica el formato de los datos que se enviarán por el servidor (xml, html, json o script). Si no se especifica nada, jQuery usa el tipo MIME para determinar el formato adecuado: `responseXML` o `ResponseText`. Los tipos disponibles son:
  - "`xml`": devuelve un documento XML que podrá tratarse con jQuery.
  - "`html`": devuelve código Html en formato texto.
  - "`script`": evalúa la respuesta en JavaScript y la devuelve en formato texto.
  - "`json`": evalúa la respuesta en JSON y devuelve un objeto JavaScript.
- "`ifModified`" (opcional): un valor booleano que indica si el servidor debe comprobar que los datos devueltos sean diferentes a la última consulta antes de reenviar el archivo con éxito. Por defecto, esta opción vale `false`.
- "`timeout`" (opcional): número de milisegundos tras el cual la consulta se considera como fallida.
- "`global`" (opcional): un valor booleano que permite lanzar la ejecución del visor de eventos global de AJAX. Por defecto, el valor es `true`. Con un valor `false`, se ignoran los desencadenadores de eventos de tipo `ajaxStart()` o `ajaxStop()`.
- "`beforeSend`" (opcional): una función que se debe ejecutar antes del envío de la consulta. Esto permite modificar el objeto `XMLHttpRequest` antes de que se envíe para especificar, por ejemplo, encabezados HTTP personalizados.
- "`error`" (opcional): una función que se debe ejecutar si falla la consulta. La función tiene tres argumentos: el objeto `XMLHttpRequest`, una cadena de caracteres que describe el tipo de error que se ha producido y un objeto exception, si se ha creado.
- "`success`" (opcional): función que se invoca si la consulta se ejecuta con éxito. Solo se pasa un argumento, esto es, los datos devueltos por el servidor.
- "`complete`" (opcional): función que se tiene que ejecutar cuando la consulta termina. La función tiene dos argumentos: el objeto `XMLHttpRequest` y una cadena de caracteres que describe el tipo de éxito de la consulta.

- "data" (opcional): datos que se envían al servidor. El objeto debe estar en forma de pares de clave/valor. Los datos se convierten en cadena de caracteres (si no lo están ya). Ver la opción processData más adelante para evitar este proceso automático.
- "processData" (opcional): valor booleano que indica si los datos de la opción data se deben convertir a cadena de caracteres. El valor por defecto es true. Para impedir la conversión, pase este valor a false.
- "contentType" (opcional): cadena de caracteres que contiene el MIME de los datos cuando se envían al servidor. Por defecto, se conserva el MIME aplication/x-www-form-urlencoded.
- "async" (opcional): un valor booleano que indica si la consulta se debe efectuar síncrona o asíncronamente. El valor por defecto para una consulta AJAX es true.

Hay disponible otras opciones, aunque se usan con menos frecuencia. Algunas son:

- "cache" (opcional): un valor booleano que, si su valor es false, impide colocar en la caché del navegador la página cargada.
- "password" (opcional): si el aceso HTTP de la consulta necesita una contraseña.
- "username" (opcional): si el aceso HTTP de la consulta necesita un nombre de usuario (username).
- "scriptCharset" (opcional): fuerza a las consultas de tipo script a interpretarse con un charset particular.
- "xhr" (opcional) permite crear el ActiveXObject (Internet Explorer) o el XMLHttpRequest(para el resto).
- "statusCode" (opcional) permite llamar a una función cuando se produce un error con un código específico. Es una novedad de la versión 1.5. de jQuery.

El método ajax() se ha reescrito completamente en la versión 1.5 de jQuery para aumentar su rendimiento e integrar las funciones diferidas (consulte la sección Las funciones diferidas, en este capítulo). Los desarrolladores de jQuery lo usan para añadir algunos parámetros que sobrepasan, con mucho, el marco operativo de este libro (contents, header, isLocal, mimeType, xhrFields). Los lectores que estén interesados pueden encontrar la documentación oficial de jQuery en:http://api.jquery.com/jQuery.ajax/

Comentarios

La forma más sencilla de esta función $.ajax() debe especificar, al menos, la URL de los datos que se deben cargar.

```
$.ajax({
url: "test.htm",
});
```

En la sección Definir una consulta por defecto de este capítulo vamos a ver que, a su vez, este único parámetro puede convertirse en opcional con el método ajaxSetup().

Observe que el método ajax() carga los contenidos de la URL especificada, pero (al contrario queload(), por ejemplo) no hace nada con dicho contenido. Para que este contenido aparezca en la página, hay que indicar las operaciones que hay que ejecutar usando las funciones asociadas a las opciones success o complete.

```
$.ajax({
url: "test.htm",
success: function(data ) {
$("div").html(data);
},
});
```

De esta manera, en caso de éxito de la consulta, los datos que carga en la dirección test.htm se

muestran como Html en la capa `<div>`.

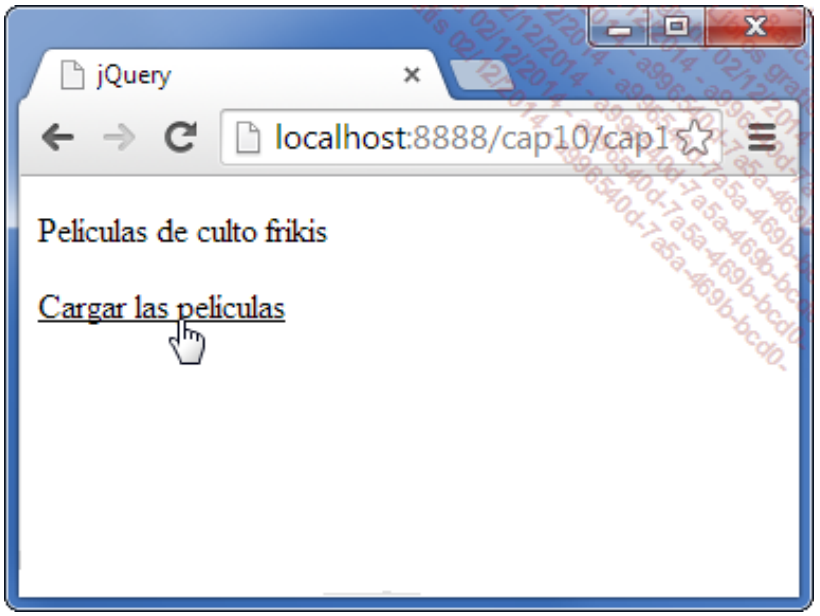
Para operaciones tan sencillas como las del ejemplo, es mejor usar `load()` o `$.get()`.

Ejemplo

Al hacer clic en un enlace, aparece un contenido que viene del servidor.

El archivo inicial es el siguiente:

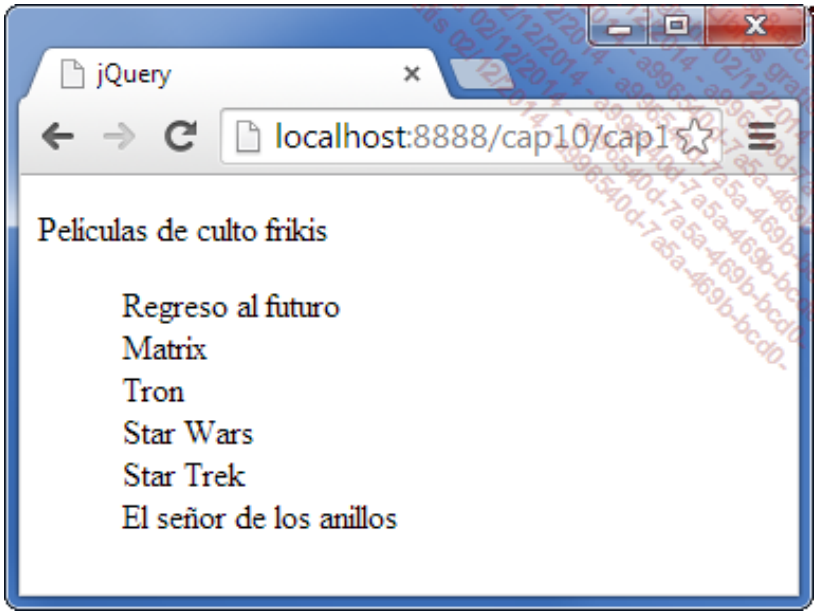
```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
a { color: black;}
li { list-style-type: none;}
</style>
</head>
<body>
<p>Películas de culto frikis</p>
<a href="#">Cargar las películas</a>
<div id="contenido">
</div>
</body>
</html>
```



El archivo que se va a cargar (peliculas.htm) incluye la siguiente lista:

```
<ul>
<li>Regreso al futuro</li>
<li>Matrix</li>
<li>Tron</li>
<li>Star Wars</li>
<li>Star Trek</li>
<li>El señor de los anillos</li>
</ul>
```

El script jQuery debe ejecutar una consulta AJAX sobre el archivo peliculas.htm y mostrar el contenido en la capa `contenido` de la página.




```
<script>
$(document).ready(function(){
$('a').click(function() {
$('a').hide();
$("#contenido").empty();
$.ajax({
url: 'peliculas.htm',
async: true,
type: 'GET',
global: false,
cache: false,
success: function(html){
$("#contenido").append(html)
}
});
});
});
</script>
```

Explicaciones.

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

Carga del DOM.

```
$('a').click(function() {
```

Al hacer clic en el enlace `<a>`.

```
$('a').hide();
```

El enlace y su contenido se ocultan.

```
$("#contenido").empty();
```

Se vacía el contenido de la capa `contenido`.

```
$.ajax({
url: 'peliculas.htm',
async: true,
type: 'GET',
global: false,
```

```
cache: false,
success: function(html){
$("#contenido").append(html)
}
});
```

La consulta AJAX de jQuery carga el archivo peliculas.htm. El proceso se lleva a cabo de modo asíncrono. Se usa el método HTTP GET. El administrador de eventos global de AJAX se desactiva. El archivo que viene del servidor no va a la caché. Para terminar, si la consulta tiene éxito, los datos del archivo que se ha cargado se añaden a la capa `contenido`.

```
});
});
```

Fin del script.

El archivo final es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$('a').click(function() {
$('a').hide();
$("#contenido").empty();
$.ajax({
url: 'peliculas.htm',
async: true,
type: 'GET',
global: false,
cache: false,
success: function(html){
$("#contenido").append(html)
}
});
});
});
</script>
<style>
a { color: black;}
li { list-style-type: none;}
</style>
</head>
<body>
<p>Películas de culto frikis</p>
<a href="#">Cargar las películas</a>
<div id="contenido">
</div>
</body>
</html>
```

Definir una consulta por defecto

Como se sugiere en el punto anterior, es posible definir la URL por defecto de las consultas AJAX de la página (siempre y cuando sea idéntica para todas las consultas AJAX de la página).

ajaxSetup(parámetros)

Define los parámetros globales para todas las consultas AJAX de la página.

```
$.ajaxSetup( {
url: test.htm",
global: false,
type: "POST"
});
```

Todas las consultas AJAX de la página se configuran con la URL indicada, el administrador de eventos AJAX se desactiva y los envíos se realizan usando el método POST, en lugar de GET.

De esta manera, la URL por defecto se puede definir por el código:

```
$.ajaxSetup({
url: "test.htm",
});
```

Cada vez que se hace una consulta, la URL se usa automáticamente. De esta manera, es suficiente con escribir:

```
$.ajax({});
```

Los eventos asociados a la consulta

1. ajaxSend()

ajaxSend(función)

Asigna una función que se ejecutará antes del envío de la consulta AJAX.

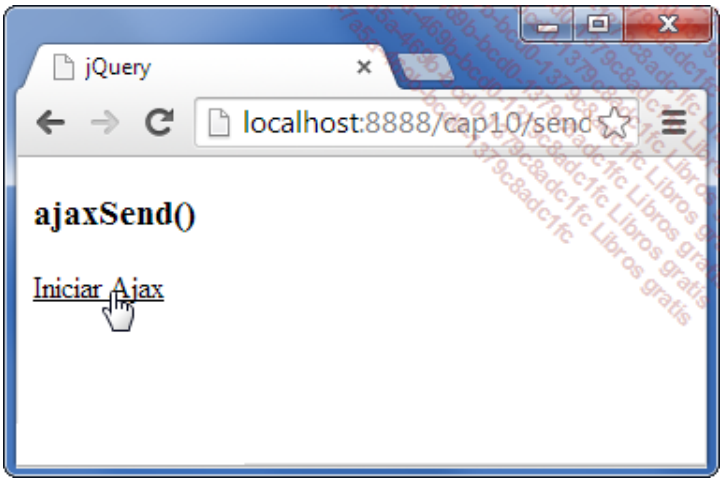
```
$(document).ajaxSend(function(){
$(this).show();
});
```

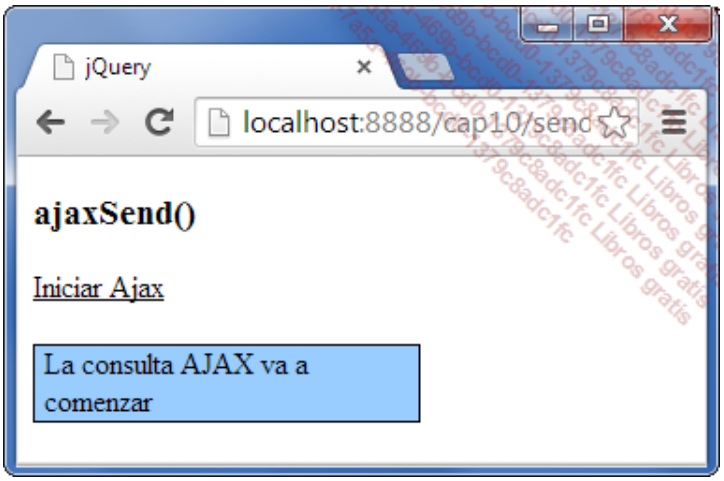
Este método devuelve un objeto jQuery.

Desde la versión 1.8 de jQuery, los eventos asociados a la consulta como `ajaxSend()`,`ajaxStart()`,`ajaxStop()`,`ajaxSuccess()`,`ajaxComplete()` y `ajaxError()` solo se pueden vincular con el documento.

Ejemplo

Mostrar un mensaje antes de que empiece la consulta.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<style>
a { color: black;}
```

Vamos a explicar las diferentes líneas del código jQuery.

```
$(document).r
```

```
#mensaje { width: 200px;
           border: 1px solid black;
           margin-top: 15px;
           background-color: #9cf;
           padding-left: 5px;}
</style>
<script>
$(document).ready(function(){
$("#mensaje").hide();
$("#resultado").hide();
$(document).ajaxSend(function() {
$("#mensaje").append("La consulta AJAX va a comenzar<br>").show();
});
$("a").click(function() {
$("#resultado").load("a.html");
});
});
</script>
</head>
<body>
<h3>ajaxSend()</h3>
<a href="#">Iniciar AJAX</a>
<div id="mensaje"></div>
<div id="resultado"></div>
</body>
</html>
```

Carga de jQuery.

`$("#mensaje")`

Cuando se carga la página, se oculta la

capa `mensaje`, que contendrá el mensaje asociado al evento `ajaxSend()`.

```
$("#resultado").hide();
```

Se oculta la capa `resultado`, que contendrá el contenido del archivo cargado, ya que este contenido no tiene importancia en este ejemplo.

```
$(document).ajaxSend(function() {
$(this).append('La consulta AJAX va a comenzar<br>').show();
});
```

Cuando se produce el evento `ajaxSend()`, se añade la frase `"La consulta AJAX va a empezar"` a la capa `mensaje` y se hace visible.

```
$('a').click(function() {
$('#resultado').load('a.html');
});
```

Al hacer clic en el enlace, se carga el archivo a.html y se inserta en la capa `resultado`. Recuerde, esta capa se ha ocultado al inicio del script.

```
});
```

Fin de jQuery.

2. ajaxStart()

ajaxStart(función)

Asigna una función que se tiene que ejecutar cuando comienza una consulta AJAX.

```
$(document).ajaxStart(function(){
$(this).show();
});
```

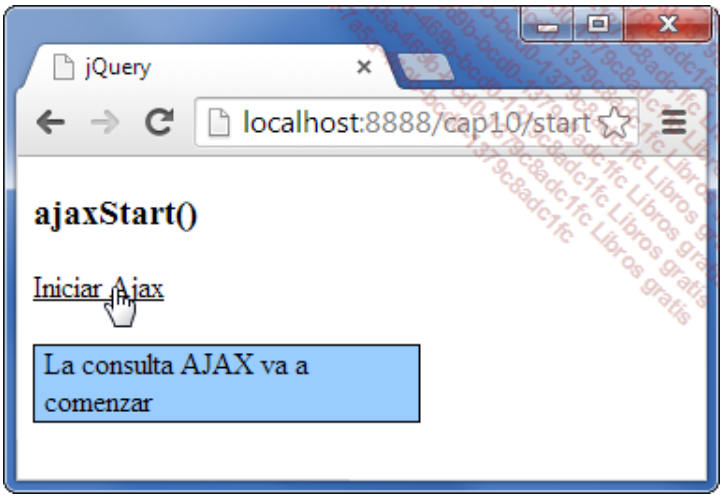
Este método devuelve un objeto jQuery.

Ejemplo:

Aplicado al ejemplo anterior, el script con `ajaxStart()` se convierte en:

```
<script>
```

```
$(document).ready(function(){
$("#mensaje").hide();
$("#resultado").hide();
$(document).ajaxStart(function() {
$("#mensaje").append("La consulta AJAX va a comenzar<br>").show();
});
$("a").click(function() {
$("#resultado").load("a.html");
});
});
</script>
```



3. ajaxStop()

ajaxStop(función)

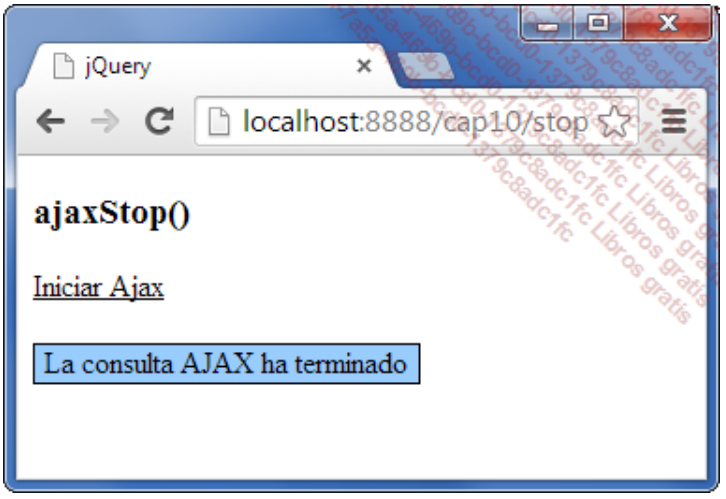
Asigna una función que se ejecutará cada vez que termina una consulta.

Este método devuelve un objeto jQuery.

Ejemplo

Aplicado al ejemplo anterior, el script con `ajaxStop()` se convierte en:

```
<script>
$(document).ready(function(){
$("#mensaje").hide();
$("#resultado").hide();
$(document).ajaxStop(function() {
$("#mensaje").append("La consulta AJAX ha terminado<br>").show();
});
$("a").click(function() {
$("#resultado").load("a.html");
});
});
</script>
```



4. ajaxSuccess()

ajaxSuccess(función)

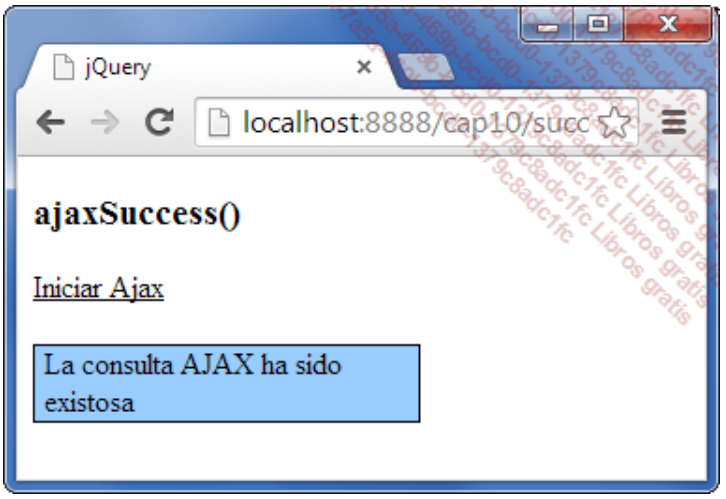
Asigna una función que se ejecutará cada vez que termine con éxito una consulta AJAX.

Este método devuelve un objeto jQuery.

Ejemplo

Aplicado al ejemplo anterior, el script con `ajaxSuccess()` *se convierte en:*

```
<script>
$(document).ready(function(){
$("#mensaje").hide();
$("#resultado").hide();
$(document).ajaxSuccess(function() {
$("#mensaje").append("La consulta AJAX ha sido existosa<br>").show();
});
$("a").click(function() {
$("#resultado").load("a.html");
});
});
</script>
```



5. ajaxComplete()

ajaxComplete(función)

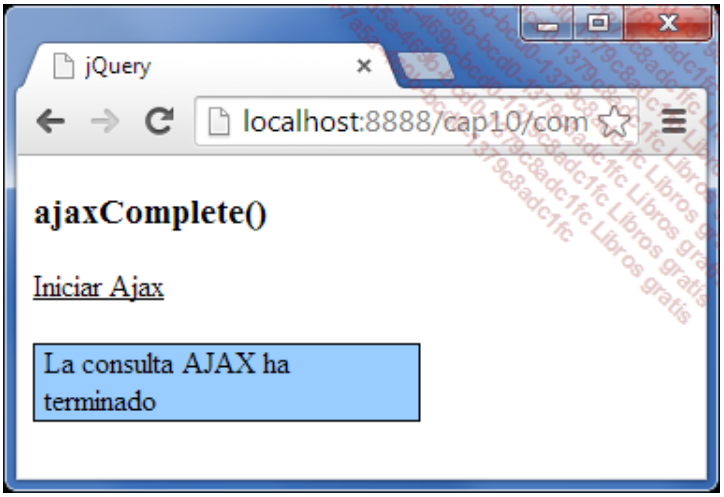
Asigna una función que se ejecutará cuando termine el proceso completo de la consulta AJAX.

Este método devuelve un objeto jQuery.

Ejemplo:

Aplicado al ejemplo anterior, el script con `ajaxComplete()` se convierte en:

```
<script>
$(document).ready(function(){
$("#mensaje").hide();
$("#resultado").hide();
$(document).ajaxComplete(function() {
$("#mensaje").append("La consulta AJAX ha terminado.<br>").show();
});
$("a").click(function() {
$("#resultado").load("a.html");
});
});
</script>
```



6. ajaxError()

ajaxError(función)

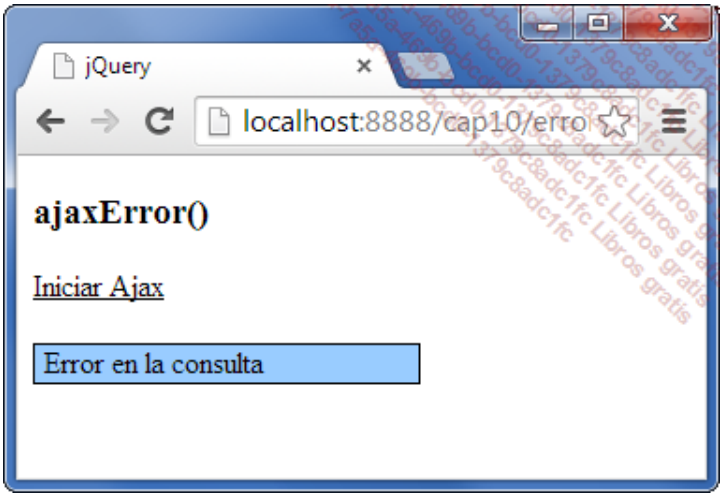
Asigna una función que se ejecutará si la consulta AJAX falla.

Este método devuelve un objeto jQuery.

Ejemplo

Retomamos nuestro archivo de ejemplo, pero (voluntariamente) con una URL que no existe en el servidor, por ejemplo x.html.

Esto provoca el fallo de la consulta AJAX.



Aplicado al ejemplo anterior, el script con `ajaxError()` se convierte en:

```
<script>
$(document).ready(function(){
$("#mensaje").hide();
$("#resultado").hide();
$(document).ajaxError(function() {
$("#mensaje").append("Error en la consulta<br>").show();
});
$("a").click(function() {
$("#resultado").load("x.html");
});
});
</script>
```

Las funciones diferidas

La versión 1.5 de jQuery introdujo el concepto de objetos diferidos, que permiten administrar, en el futuro, los elementos que no existen todavía en el instante presente. Esto es habitual en las consultas AJAX asíncronas.

En las versiónes anteriores de jQuery, solo se gestionaban las situaciones de éxito (`success`) o fracaso (`error`) de las consultas AJAX. Ahora, los objetos diferidos permiten externalizar las funciones de llamada fuera del contexto de la consulta, sin preocuparse del momento en que termine la consulta AJAX.

Esta característica, importante en la gestión de las consultas AJAX, ya estaba presente en los frameworks JavaScript Dojo y MochiKit. Los desarrolladores en jQuery ahora pueden disponer de este avance notorio a nivel de la programación de los procesos asíncronos.

Con jQuery, estos objetos diferidos no se limitan a las consultas AJAX y también se pueden usar en otros contextos.

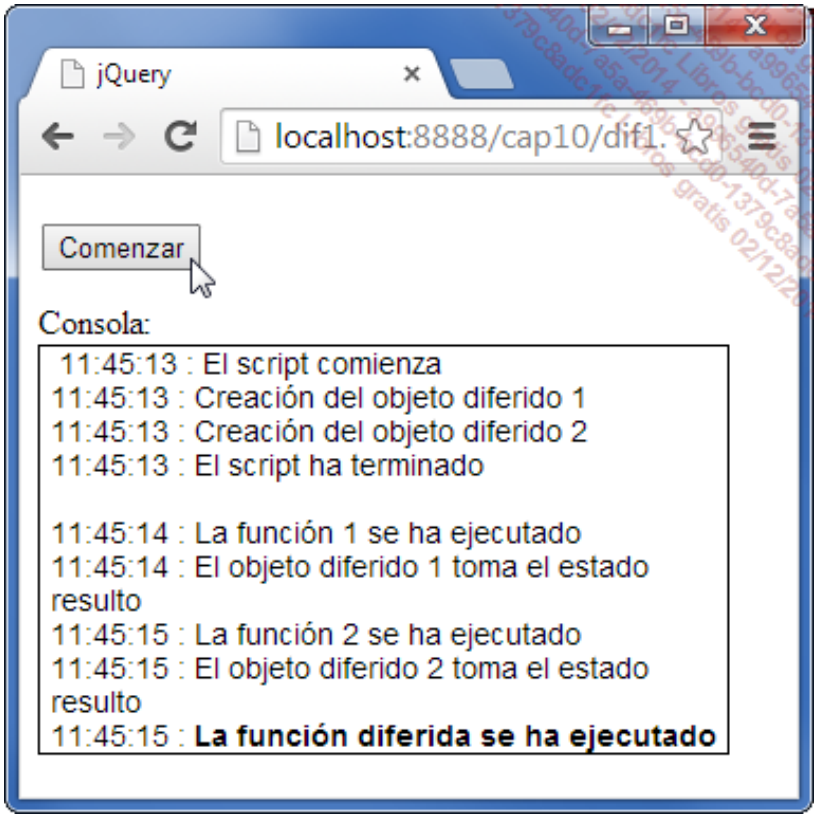
La siguiente tabla proporciona las principales funcionalidades de estos objetos diferidos.

jQuery.Deferred()	Crea un nuevo objeto diferido. Este objeto diferido puede tener tres estados: no resuelto (`unresolved`), resuelto (`resolved`) o rechazado (`rejected`). Este método Deferred() puede tener como parámetro una función que se ejecutará cuando se haya creado el objeto diferido.
jQuery.when()	Permite definir las funciones que se tienen que ejecutar cuando se hacen una o varias funciones.
deferred.then()	Determina las funciones que se tienen que llamar cuando el objeto diferido se resuelve o rechaza. Este método tiene dos argumentos: la función que se llama cuando el objeto diferido tiene el estado de resuelto y la función que se llama cuando el objeto diferido tiene el estado rechazado.
deferred.done()	Llamada de funciones cuando el objeto diferido se resuelve.
deferred.fail()	Llamada de funciones cuando el objeto diferido se rechaza.
deferred.always()	Llamada de funciones que siempre se ejecutarán, independientemente de si el objeto diferido se resuelve o se rechaza.
deferred.isResolved	Permite saber si el objeto diferido tiene el estado de resuelto.
deferred.isRejected	Permite saber si el objeto diferido tiene el estado de rechazado.
deferred.resolve()	Da al objeto diferido el estado de resuelto y llama a las funciones definidas por deferred.done().
deferred.reject()	Da al objeto diferido el estado de rechazado y llama a las funciones definidas por deferred.fail().

Vamos a ver un primer ejemplo para ilustrar esta noción, muy abstracta, de los objetos diferidos.

Ejemplo 1

Ejecutamos una función diferida después de la ejecución de otras dos funciones (función 1 y función 2). Vamos a añadir un temporizador setTimeout para separar la ejecución de estas dos funciones. Estas funciones no se hallan en un contexto de consultas asíncronas.



A las 11:45:13, empieza la ejecución del script. Se crean los objetos diferidos 1 y 2. En este estado, el código de la función diferida ya está presente, pero su ejecución está diferida.

A las 11:45:14, gracias al temporizador, se ejecuta la función 1 y el código da al objeto diferido 1 el estado de resuelto.

A las 11:45:15, gracias al temporizador, se ejecuta la función 2 y el objeto diferido 2 toma a su vez el estado de resuelto.

Ahora que se han ejecutado las dos funciones, se ejecuta la función diferida.

El código es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style type="text/css">
#consola { font-family: Arial;
           font-size: 14px;
           border: 1px solid black;
           padding-left: 5px;
           width: 320px;}
#boton { margin-top:15px;
         margin-bottom: 15px;}
</style>
<script src="jquery.js"></script>
<script>
function log(mensaje) {
var texto= $("#consola").html();
var time = new Date().toTimeString();
time = time.substring(0, time.indexOf(' '));
texto += time + ": " + mensaje + "<br>";
$("#consola").html(texto);
}
function funcion1() {
```

```
var deferred = $.Deferred(log("Creación del objeto diferido 1"));
setTimeout(function() {log("La función 1 se ha realizado");
deferred.resolve(log("El objeto diferido 1 tiene el estado resuelto"));},
1000);
return deferred;
}
function funcion2() {
var deferred = $.Deferred(log("Creación del objeto diferido 2"));
setTimeout(function() {log("La función 2 se ha realizado");
deferred.resolve(log("El objeto diferido 2 tiene el estado
resuelto<div> <div>"));},
2000);
return deferred;
}
function run() {
log("El script empieza");
$.when(funcion1(), funcion2())
.then(
function() {log("<b>La función diferida se ha realizado<b>");},
function() {log("<i>La función diferida no se ha realizado</i>");}
);
log("El script termina<div> <div>");
}
$(function() {
$("#boton").click(function(){
run();
});
});
</script>
</head>
<body>
<input type="button" value="Comenzar" id="boton" /><br>
Consola:
<div id="consola"> </div>
</body>
</html>
```

Explicaciones.

```
function log(mensaje) {
var texto= $("#consola").html();
var time = new Date().toTimeString();
time = time.substring(0, time.indexOf(' '));
texto += time + ": " + mensaje + "<br>";
$("#consola").html(texto);
}
```

Esta función `log()` sirve para mostrar en la capa consola los diferentes mensajes que tiene este script.

Pasemos a la parte principal del script (la función `run()`).

```
function run() {
log("El script empieza");
```

Un primer mensaje indica que el script empieza.

```
$.when(funcion1(), funcion2()) .then(
function() {log("<b>La función diferida se ha realizado<b>");},
function() {log("<i>La función diferida no se ha realizado</i>");}
);
```

Esta parte aborda el código de la función diferida. Cuando (`when`) las funciones 1 y 2 se completan, entonces (`then`) se muestra el texto "La función diferida se ha realizado" en la consola. Si éste no es el caso, se muestra el texto "La función diferida no se ha realizado".

```
log("El script termina<div> <div>");
}
```

Se muestra un pequeño mensaje para anunciar el fin del script principal.

Veamos el funcionamiento de la función 1 (`funcion1`).

```
function funcion1() {
var deferred = $.Deferred(log("Creación del objeto diferido 1"));
```

Empieza creando un objeto diferido (`var deferred = $.Deferred`), lo que se confirma con el mensaje.

```
setTimeout(function() {log("La función 1 se ha realizado");
deferred.resolve(log("El objeto diferido 1 tiene el estado resuelto"));},
1000);
return deferred;
}
```

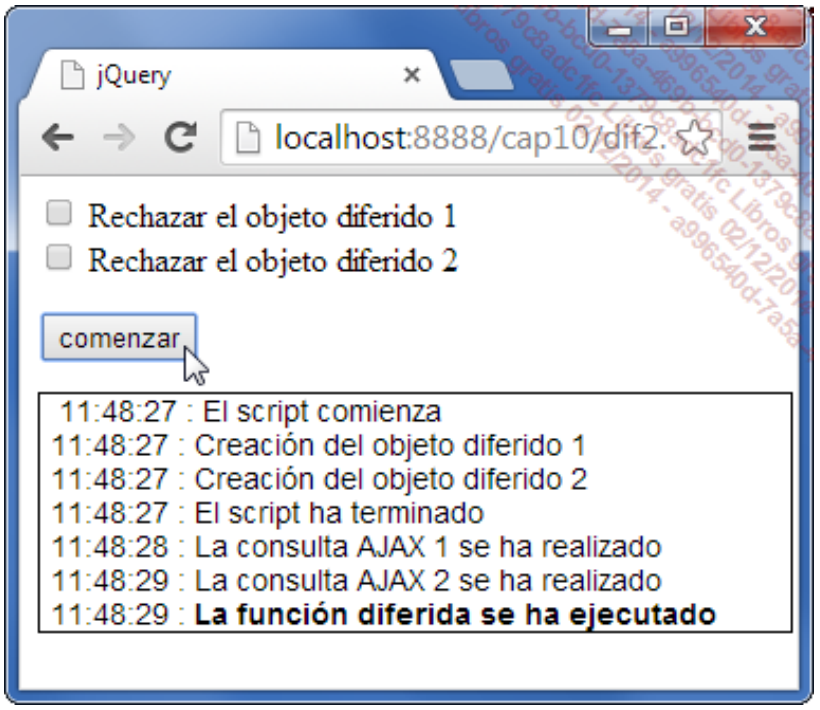
El temporizador `setTimeout` ejecuta la función después de 1 segundo (1000). Este último consiste simplemente en mostrar (`log`) el texto "La función 1 se ha realizado" y en dar al objeto diferido el estado de resuelto (`deferred.resolve()`), lo que se confirma por otro mensaje.

La función 2 (`funcion2()`) se inspira en la función 1, fijando el temporizador en 2 segundos.

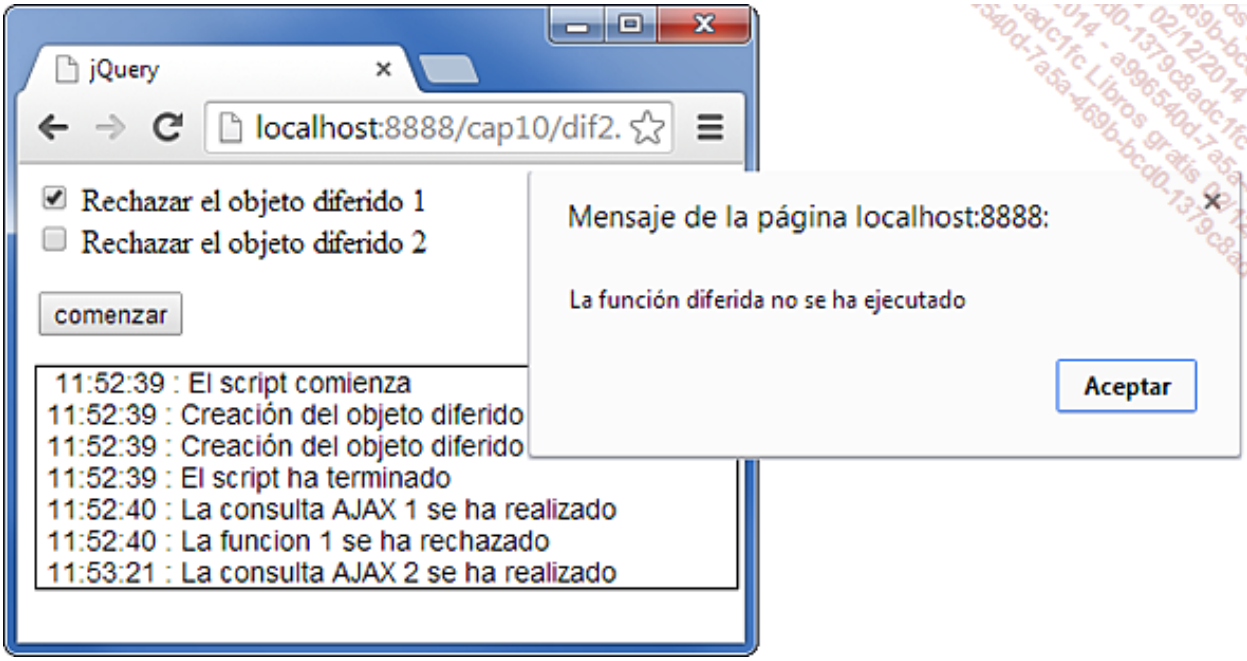
Ejemplo 2

Continuando con el ejemplo anterior, añadimos una casilla de selección que permite dar el estado de rechazado al objeto diferido. Esta vez llamamos a consultas asíncronas.

Si no se rechaza ningún objeto diferido, se ejecuta la función diferida después de la ejecución de las dos consultas AJAX.



Si un objeto diferido se rechaza, se muestra un mensaje de alerta después de la ejecución de las dos consultas AJAX.



El código:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
#consola { font-family: Arial;
           font-size: 14px;
           border: 1px solid black;
           padding-left: 5px;
           width: 350px;}
#boton { margin-top:15px;
         margin-bottom: 15px;
}
</style>
<script src="jquery.js"></script>
<script>
function log(mensaje) {
var texto= $("#consola").html();
var time = new Date().toTimeString();
time = time.substring(0, time.indexOf(' '));
texto += time + ": " + mensaje + "<br>";
$("#consola").html(texto);
}
function funcion1() {
var deferred = $.Deferred(log("Creación del objeto diferido 1"));
setTimeout(function() {
$.ajax({ url: "opcion1.htm",
         success: log("La consulta AJAX 1 se ha realizado") });
interna();},
1000);
return deferred;
function interna() {
if ($("#rechazar1:checked").val() == "on")
deferred.reject(log("La función 1 se ha rechazado"));
else
deferred.resolve();
}
}
function funcion2() {
```

```
var deferred = $.Deferred(log("Creación del objeto diferido 2"));
setTimeout(function() {
$.ajax({ url: "opcion2.htm",
         success: log("La consulta AJAX 2 se ha realizado") });
interna();},
2000);
return deferred;
function interna() {
if ($("#rechazar2:checked").val() == "on")
deferred.reject(log("La función 2 se ha rechazado"));
else
deferred.resolve();
}
}
function run() {
log("El script empieza");
$.when(funcion1(), funcion2()).then(
function() {log("<b>La función diferida se ha realizado</b>");},
function() {alert("La función diferida no se ha ejecutado");}
);
log("El script termina");
}
$(function() {
$("#boton").click(function(){
run();
});
});
</script>
</head>
<body>
<input type="checkbox" id="rechazar1"> Rechazar el objeto diferido
1<br>
<input type="checkbox" id="rechazar2"> Rechazar el objeto diferido
2<br>
<input type="button" value="Comenzar" id="boton">
<div id="consola"> </div>
</body>
</html>
```

Explicaciones.

```
function log(mensaje) {
var texto= $("#consola").html();
...
$("#consola").html(texto);
}
```

Como en el ejemplo 1, la función `log()` sirve para mostrar los mensajes.

```
function run() {
log("El script empieza");
```

Un primer mensaje indica que el script empieza.

```
$.when(funcion1(), funcion2()).then(
function() {log("<b>La función diferida se ha realizado</b>");},
function() {alert("La función diferida no se ha ejecutado");}
);
```

Esta parte aborda el código de la función diferida. Cuando (`when`) las funciones 1 y 2 se completan, entonces (`then`) se muestra el texto "La función diferida se ha realizado" en la consola. En caso contrario, se muestra un mensaje de alerta.

```
log("El script termina");
```

```
}
```

Se muestra un pequeño mensaje para anunciar el fin del script principal.

Veamos la función 1.

```
function funcion1() {
var deferred = $.Deferred(log("Creación del objeto diferido 1"));
```

Creación del objeto diferido y visualización de un mensaje de confirmación.

```
setTimeout(function() {
$.ajax({ url: "opcion1.htm",
         success: log("La consulta AJAX 1 se ha realizado") });
interna();},
1000);
return deferred;
```

Se difiere (`setTimeout`) la ejecución de la consulta AJAX (`$.ajax`) un segundo. Ésta carga el archivo opcion1.htm (`url: "opcion1.htm"`) y muestra un mensaje cuando termina con éxito (`success: log(...)`). También llamamos a la función `interna()` que gestiona el estado del objeto diferido.

```
function interna() {
if ($("#rechazar1:checked").val() == "on")
deferred.reject(log("La función 2 se ha rechazado"));
```

Si la primera casilla de selección está activada, el objeto diferido se rechaza (`deferred.reject`) y se muestra un mensaje en la consola.

```
else
deferred.resolve();
}
}
```

Si la primera casilla de selección no está activada, el objeto diferido se marca como resuelto (`deferred.resolve()`).

La función 2 usa el mismo esquema, pero con una consulta AJAX en el archivo opcion2.htm y un retraso de 2 segundos.

Serializar los datos

Este método transforma los datos de los campos del formulario en una cadena de caracteres que los contiene.

Este proceso es muy útil si se desea enviar estos datos al servidor para una consulta AJAX en un formato compatible con la mayor parte de los lenguajes de programación del lado servidor.

Para el correcto funcionamiento del método `serialize()`, los campos del formulario deben tener un atributo name.

serialize()

Transforma los datos de los campos del formulario en una cadena de caracteres.

```
$("form").serialize();
```

Este método devuelve una cadena de caracteres (String).

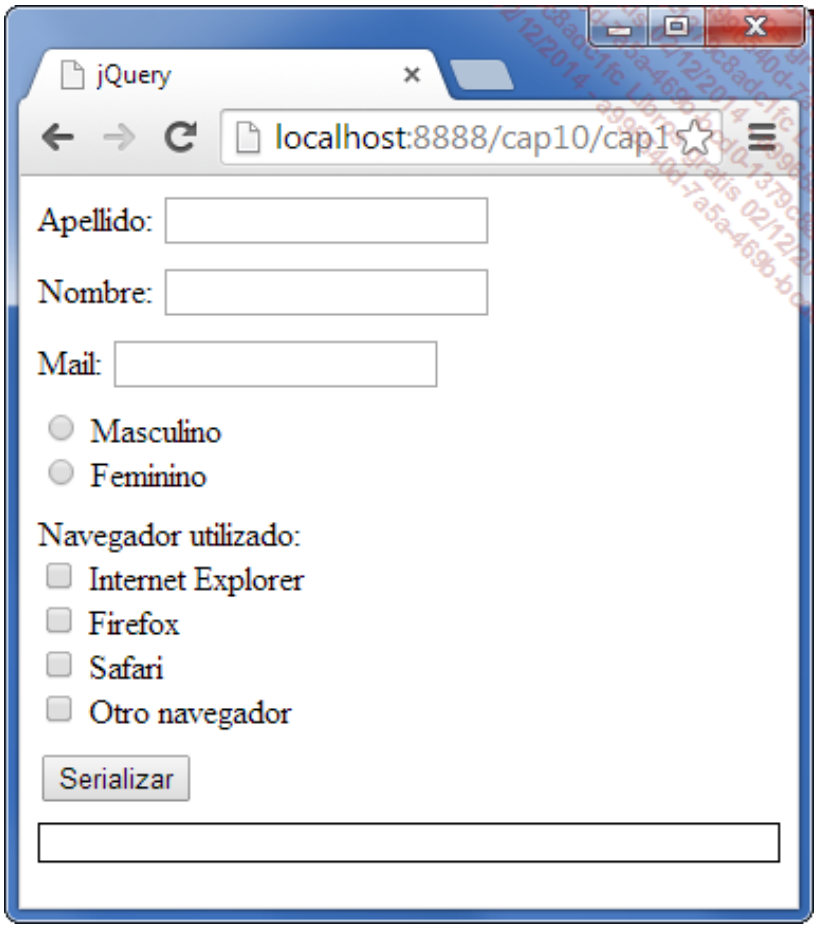
serializeArray()

Transforma los datos de los campos de formulario en una variable Array.

Este método devuelve un objeto Array.

Ejemplo

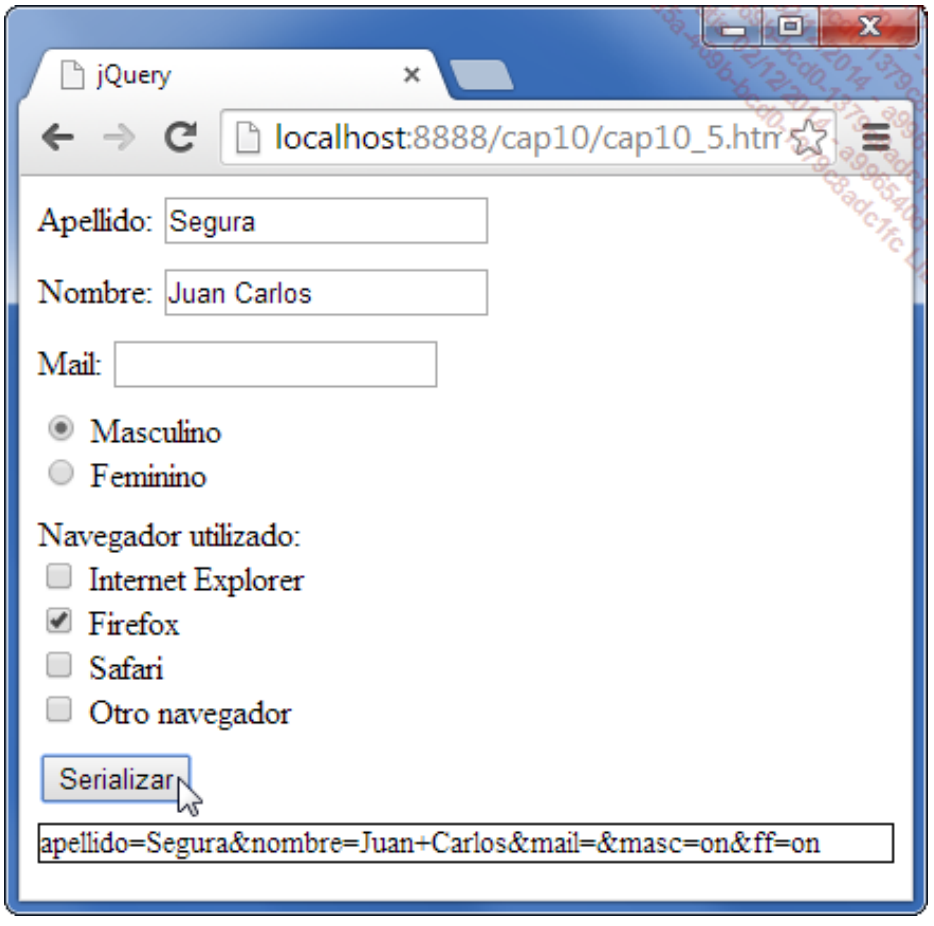
Supongamos un formulario de inicio.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
```

```
<style>
div { margin-bottom: 8px;}
p { margin-top: 8px;
    font-size:14px;
    border: 1px solid black;}
</style>
</head>
<body>
<form action="">
<div>Apellido: <input type="text" name="apellidos" id="apellidos"></div>
<div>Nombre: <input type="text" name="nombre" id="nombre"></div>
<div>Mail: <input type="text" name="mail" id="mail"></div>
<div>
<input type="radio" name="masc"> Masculino<br>
<input type="radio" name="fem"> Feminino<br>
</div>
<div>
Navegador utilizado:<br />
<input type="checkbox" name="ie"> Internet Explorer<br>
<input type="checkbox" name="ff"> Firefox<br>
<input type="checkbox" name="saf"> Safari<br>
<input type="checkbox" name="otro_nav"> Otro navegador
</div>
</form>
<button>Serializar</button>
<p id="resultado"> </p>
</body>
</html>
```

Al hacer clic en el botón, los datos introducidos por el usuario se serializan.



Se usa el siguiente script:

```
<script>
```

```
$(document).ready(function(){
$('button').on("click", function() {
var str = $("form").serialize();
$("#resultado").text(str);
});
});
</script>
```

Explicaciones:

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

Inicialización de jQuery.

```
$('button').on("click", function() {
```

Al hacer clic en el botón.

```
var str = $("form").serialize();
```

El formulario (`$("form")`) se serializa (`serialize()`) y almacena en la variable `str`.

```
$("#resultado").text(str);
```

El contenido de la variable `str` se añade como texto (`text(str)`) en el párrafo identificado por`resultado`.

```
});
});
```

Fin del script.

El archivo final es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$('button').on("click", function() {
var str = $("form").serialize();
$("#resultado").text(str);
});
});
</script>
<style>
div { margin-bottom: 8px;}
p { margin-top: 8px;
    font-size:14px;
    border: 1px solid black;}
</style>
</head>
<body>
<form action="">
<div>Apellido: <input type="text" name="apellidos"
<div>Nombre: <input type="text" name="nombre" id="nombre"></div>
id="apellidos"></div>
<div>Mail: <input type="text" name="mail" id="mail"></div>
<div>
<input type="radio" name="masc"> Masculino<br>
```

```
<input type="radio" name="fem"> Feminino<br>
</div>
<div>
Navegador utilizado:<br />
<input type="checkbox" name="ie"> Internet Explorer<br>
<input type="checkbox" name="ff"> Firefox<br>
<input type="checkbox" name="saf"> Safari<br>
<input type="checkbox" name="otro_nav"> Otro navegador
</div>
</form>
<button>Serializar</button>
<p id="resultado"> </p>
</body>
</html>
```

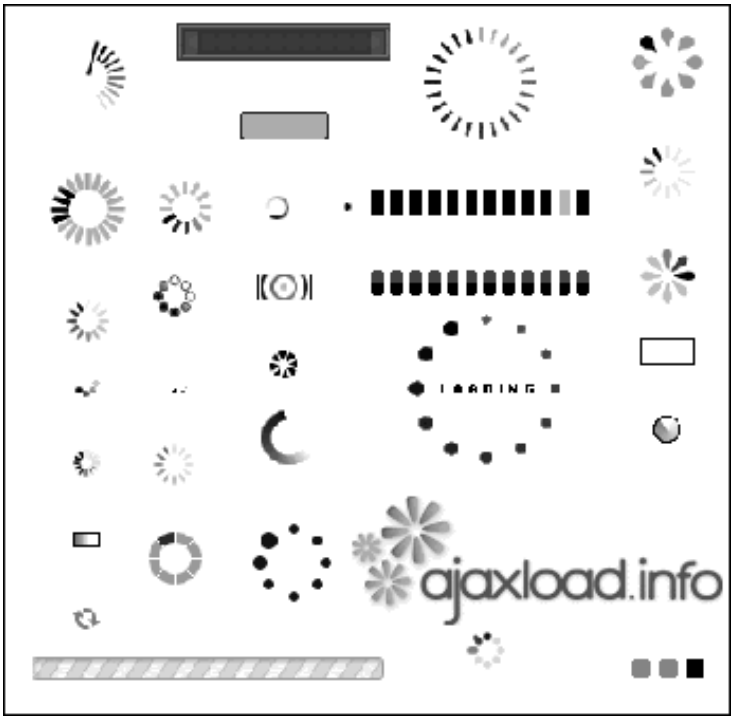
Aplicaciones

1. Un icono de carga

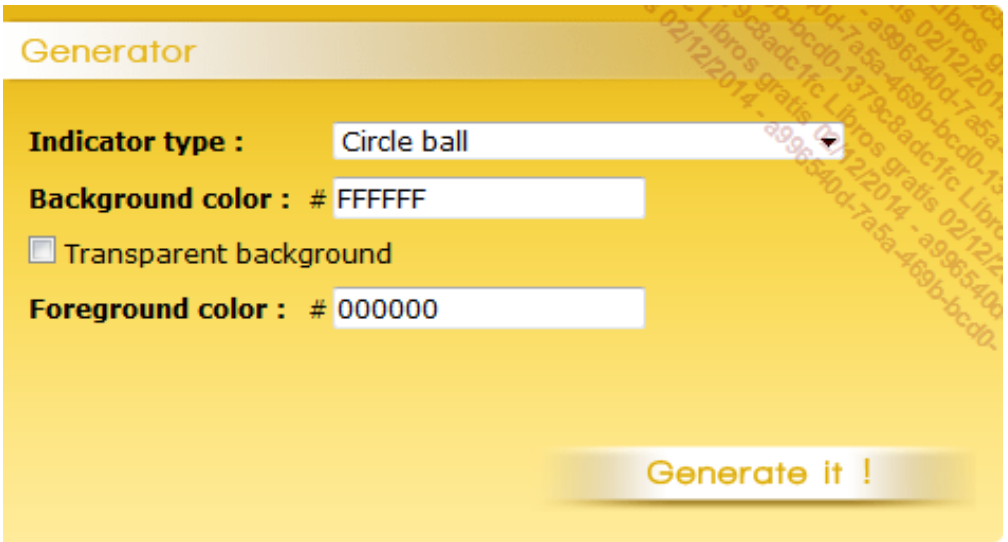
Algunas veces, las consultas AJAX pueden ser lentas a la hora de cargar el archivo externo, en función del tamaño de éste, de la carga del servidor o de la calidad de la conexión.

Para evitar que el usuario se canse durante esos momentos de espera, el diseñador puede prever un pequeño icono que indique que la carga está en curso. Este icono es una imagen gif animada que aparece al inicio de la consulta y desaparece cuando la consulta termina con éxito.

En el sitio ajaxload (http://www.ajaxload.info/), se ofrece una serie impresionante de este tipo de iconos animados para descargarlos.

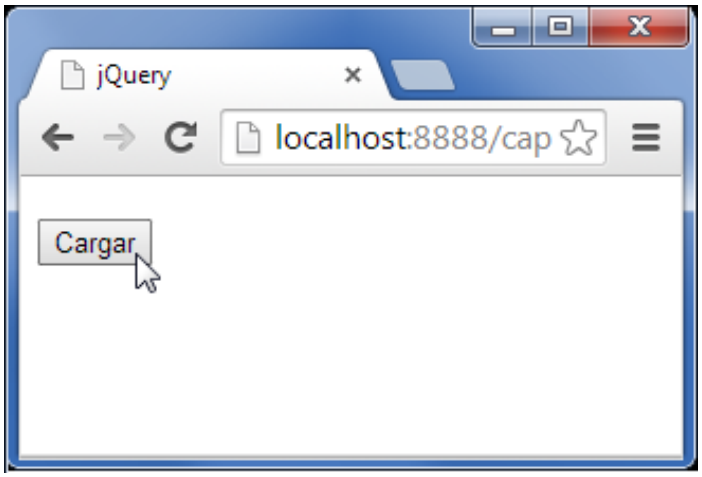


Su interfaz permite elegir el tipo de animación (Indicator type), el color de fondo (Background color) y el color de la animación (Foreground color). El botón **Generate it !** crea la imagen. El botón**Download it !** le permite descargar el icono animado. No hay nada más sencillo: perfecto para no perder el tiempo creando estas imágenes de espera. Por supuesto, es gratuito.



Ejemplo

El archivo de inicio simplemente incluye un botón y una capa destinada a recibir el archivo cuando se carga.

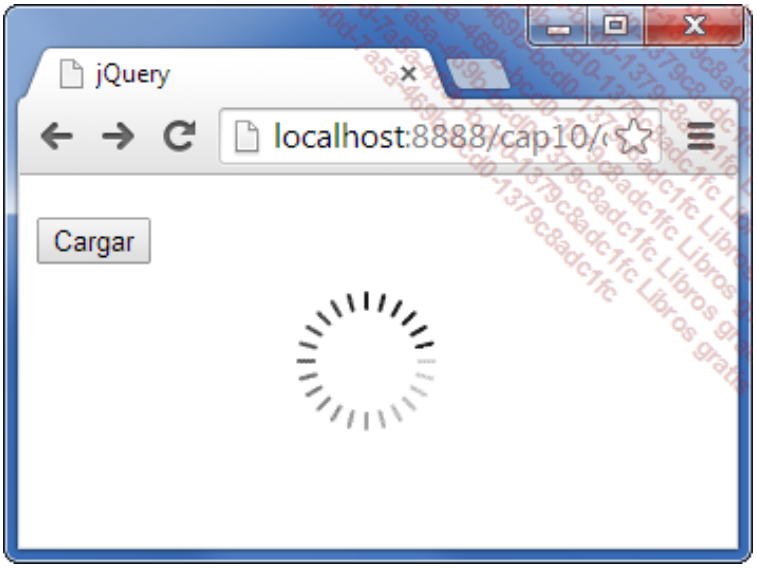


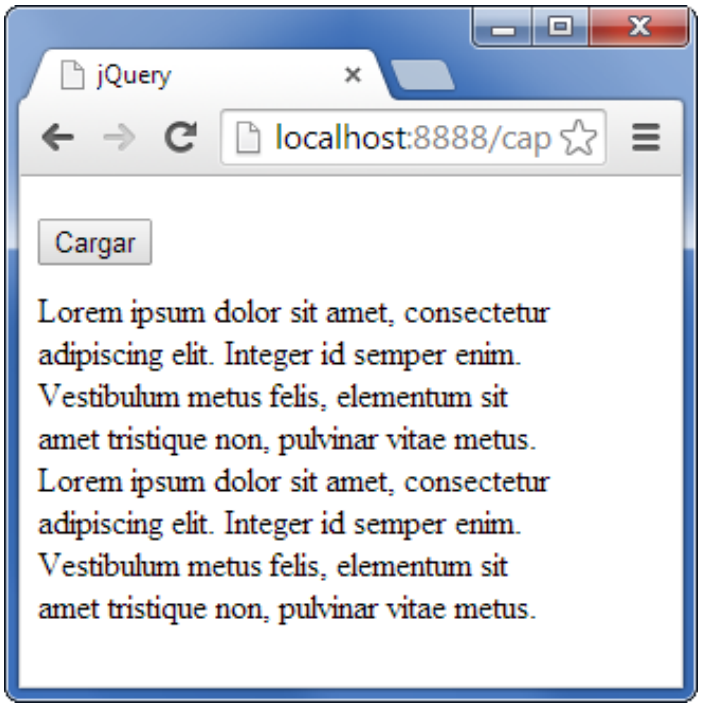

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
button { margin: 12px 0 12px 0;}
#contenido { width: 250px;}
             display: none;}
</style>
</head>
<body>
<button id="boton">Cargar</button>
<div id="contenido"></div>
</body>
</html>
```

El archivo que se va a cargar desde el servidor se llama lorembis.htm. Durante la fase de desarrollo y de pruebas internas, la aparición del icono de descarga puede no verse. Para visualizarlo, lo mejor es usar un archivo grande.

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer
id semper enim. Vestibulum metus felis, elementum sit amet
tristica non, pulvinar vitae metus.Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Integer id semper enim. Vestibulum
metus felis, elementum sit amet tristica non, pulvinar vitae
metus.
...
```

El script jQuery debe tener en cuenta no solo la consulta AJAX, sino también la aparición y desaparición de la imagen de carga.





El icono está disponible para descarga en la página Información.

```
<script>
$(document).ready(function() {
$('#boton').click(function() {
$('#contenido').hide().load('lorembis.htm', function() {
$(this).fadeIn(4000);
});
return false;
});
$('<div id="loading"><img src="ajax-loader.gif"></div>')
.insertBefore('#contenido')
.ajaxStart(function() {
$(this).show();
})
.ajaxStop(function() {
$(this).hide();
});
});
</script>
```

Explicaciones.

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

Inicialización de jQuery.

```
$('#boton').click(function() {
```

Al hacer clic en el botón.

```
$('#contenido').hide().load('lorembis.htm', function() {
$(this).fadeIn(4000);
});
```

Se realiza una consulta AJAX sobre el archivo lorembis.htm y su contenido se inserta en la capa`contenido`. Se añade un efecto de aparición progresiva `fadeIn(4000)`.

```
return false;
});
```

Fin de la parte de la carga.

```
$('<div id="loading"><img src="ajax-loader.gif" /></div>')
.insertBefore('#contenido')
```

La imagen ajax-loader.gif de la capa `loading` se inserta (`insertBefore`) antes de la capa`contenido`.

```
.ajaxStart(function() {
$(this).show();
})
```

Cuando se inicia la consulta AJAX (`ajaxStart`), esta imagen (`this`) se vuelve visible (`show()`).

```
.ajaxStop(function() {
$(this).hide();
});
```

Cuando la consulta AJAX termina (`ajaxStop`), esta imagen (`this`) se oculta (`hide()`).

```
});
```

Fin del script jQuery.

La página definitiva es la siguiente.

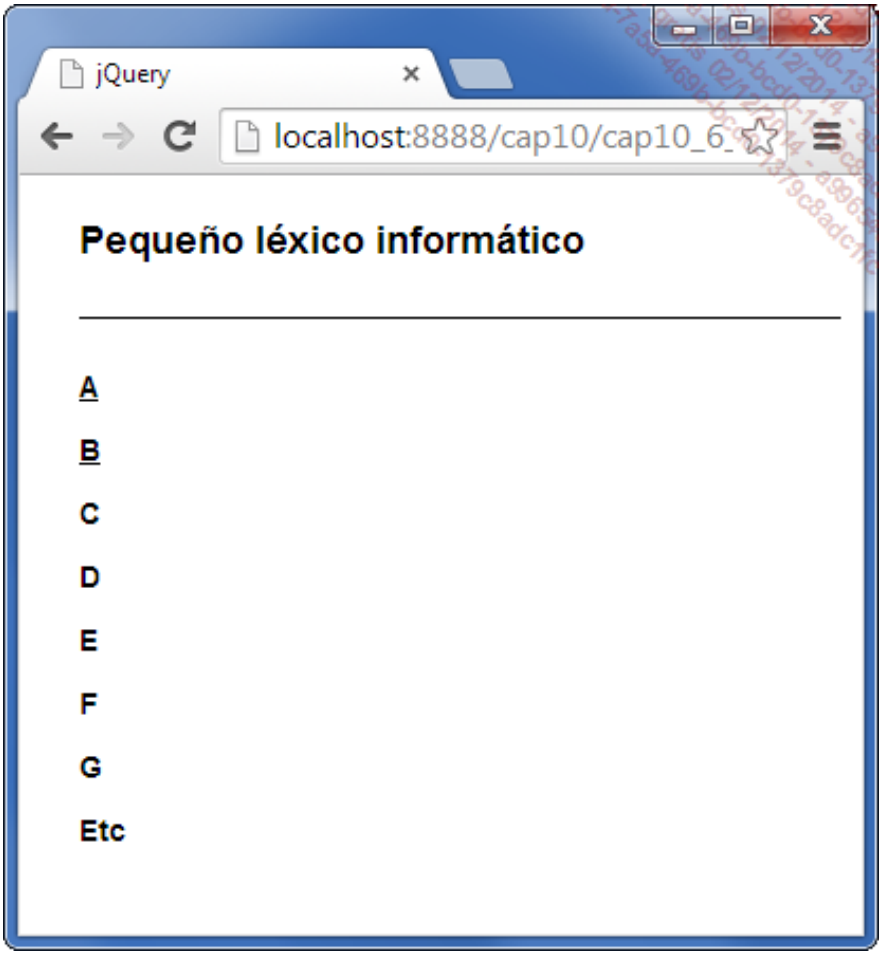
```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function() {
$('#boton').click(function() {
$('#contenido').hide().load('lorembis.htm', function() {
$(this).fadeIn(4000);
});
return false;
});
$('<div id="loading"><img src="ajax-loader.gif"></div>')
.insertBefore('#contenido')
```

```
.ajaxStart(function() {
$(this).show();
})
.ajaxStop(function() {
$(this).hide();
});
});
</script>
<style>
button { margin: 12px 0 12px 0;}
#contenido { width: 250px;}
#loading { margin: 30px 0 0 30px;
           position: absolute;
           display: none;}
</style>
</head>
<body>
<button id="boton">Cargar</button>
<div id="contenido"></div>
</body>
</html>
```

2. Un léxico en AJAX

Vamos a elaborar un pequeño léxico informático, cuyas definiciones se cargan por consultas AJAX en la página principal.

Esta página principal es la siguiente.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
```

```
<title>jQuery</title>
<style>
body { font: 62.5% Arial, Verdana, sans-serif;}
a { color: black}
#container { font-size: 1.2em;
             margin: 10px 20px;
             width: 360px;}
#header { margin-top: 20px;
          margin-bottom: 10px;
          padding-bottom: 10px;
          border-bottom: 1px solid black;}
#alfabeto { float: left;
            width: 30px;
            padding-right: 10px;
            margin-right: 10px; }
#lexico { float: left;
          width: 300px;}
</style>
</head>
<body>
<div id="container">
<div id="header">
<h2>Pequeño léxico informático</h2>
</div>
<div id="alfabeto">
<div id="letra_a">
<h3><a href="#">A</a></h3>
</div>
<div id="letra_b">
<h3><a href="#">B</a></h3>
</div>
<div><h3>C</h3></div>
<div><h3>D</h3></div>
<div><h3>E</h3></div>
<div><h3>F</h3></div>
<div><h3>G</h3></div>
<div><h3>Etc</h3></div>
</div>
<div id="lexico">
</div>
</div>
</body>
</html>
```

El archivo de las definiciones de la letra A (letra_a.htm) es el siguiente:

```
<h3>ActiveX</h3>
<div>
Tecnología Microsoft. La tecnología ActiveX permite la creación
de componentes que se pueden incluir en las páginas Html.
</div>
<h3>AJAX</h3>
<div>
El acrónimo AJAX se aplica a un conjunto de técnicas de
programación de sitios de Internet y procede del inglés
Asynchronous JavaScript and XML.
</div>
<h3>AVI</h3>
<div>
Audio Video Interleave. Formato de archivo de Microsoft para los
datos de audio y vídeo.
</div>
```

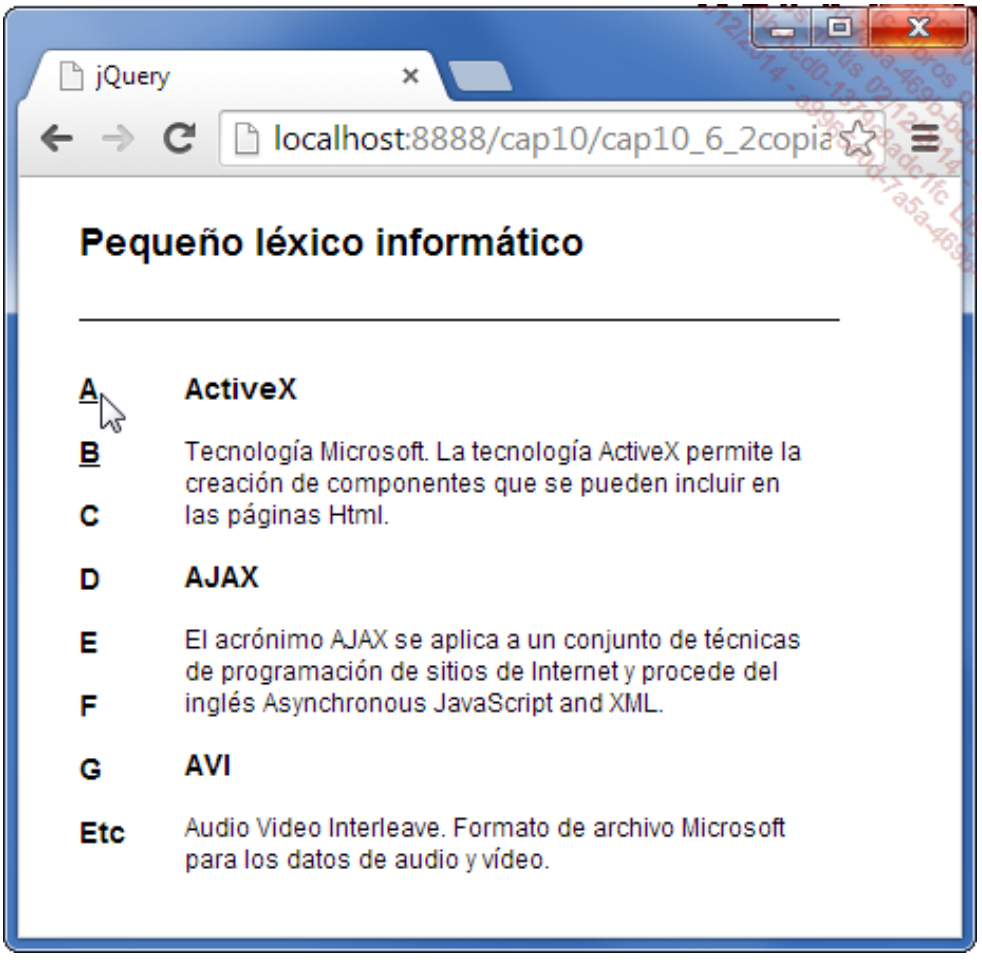
El archivo de las definiciones de la letra B (letra_b.htm) es el siguiente:

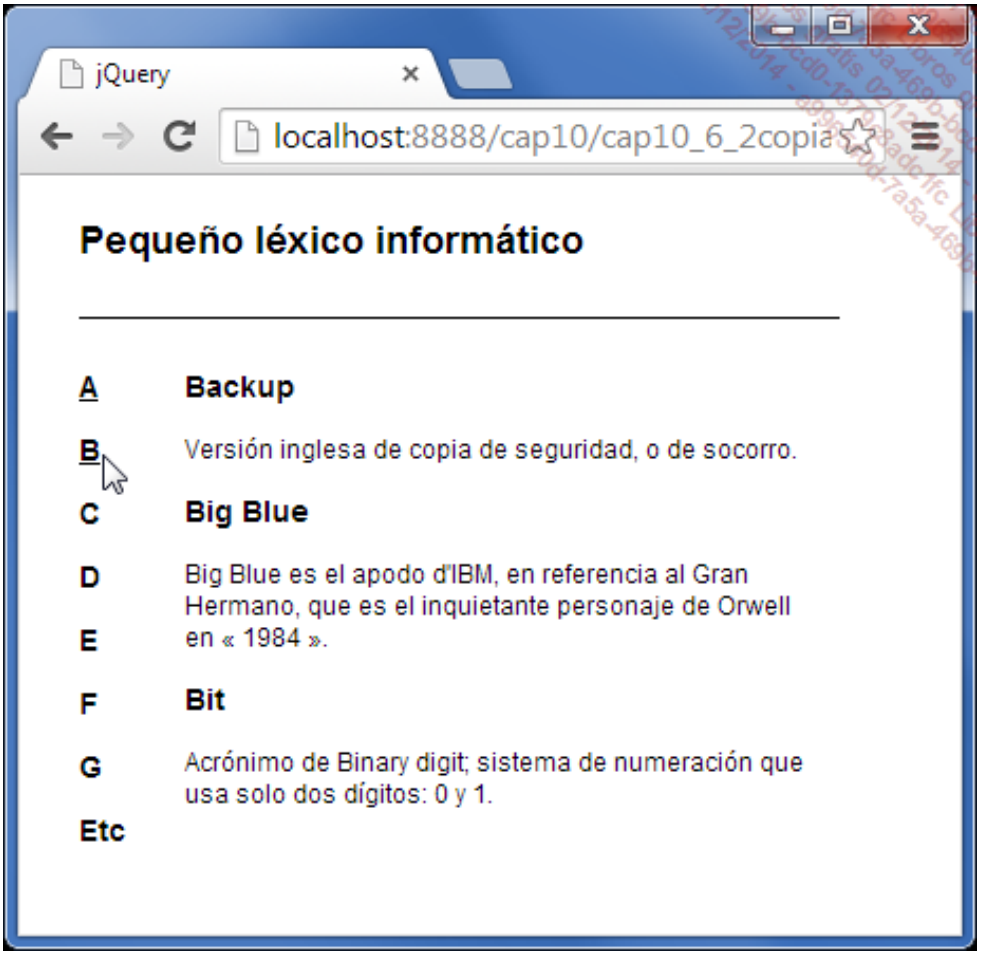
```
<h3>Backup</h3>
<div>
Versión inglesa de copia de seguridad o de socorro.
</div>
<h3>Big Blue</h3>
<div>
Big Blue es el apodo d'IBM, en referencia al Gran Hermano, que
es el inquietante personaje de Orwell en « 1984 ».
</div>
<h3>Bit</h3>
<div>
Acrónimo de Binary digit; sistema de numeración que usa solo
dos dígitos:
0 y 1.
</div>
```

Estos archivos están disponibles para descarga en la página Información.

Cuando los archivos tienen acentos, se recomienda guardarlos en formato UTF-8.

Al hacer clic en la letra A o B, una consulta AJAX carga las definiciones correspondientes (letra_a.htm o letra_b.htm), en la capa `lexico`.





El script jQuery tiene la forma:

```
<script>
$(document).ready(function() {
$('#letra_a').on("click", function() {
$('#lexico').hide().load('letra_a.htm', function() {
$(this).slideDown(1000);
});
return false;
});
$('#letra_b').on("click", function() {
$('#lexico').hide().load('letra_b.htm',on("click", function()
{ function() {
$(this).slideDown(1000);
});
return false;
});
});
</script>
```

A estas alturas del libro, las explicaciones pueden ser más escuetas.

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

Inicialización de jQuery.

```
$('#letra_a').on("click", function() {
```

Al hacer clic en la letra A.

```
$('#lexico').hide().load('letra_a.htm', function() {
$(this).slideDown(1000);
```

```
});
```

Se ejecuta una consulta AJAX en jQuery (`load()`) sobre el archivo letra_a.htm y su contenido se inserta en la capa `lexico`. Se añade un efecto visual al script. En un primer momento, el contenido de `lexico` se oculta (`hide()`) y el nuevo contenido aparece con un efecto de deslizamiento hacia abajo (`slideDown(1000)`).

La página final se convierte en:

```
<!doctype html>
html lang="es"
<head>
<meta charset"=utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function() {
$('#letra_a').on("click", function() {
$('#lexico').hide().load('letra_a.htm', function() {
$(this).slideDown(1000);
});
return false;
});
$('#letra_b').on("click", function() {
$('#lexico').hide().load('letra_b.htm',on("click", function()
{ function() {
$(this).slideDown(1000);
});
return false;
});
});
</script>
<style>
body { font: 62.5% Arial, Verdana, sans-serif;}
a { color: black}
#container { font-size: 1.2em;
             margin: 10px 20px;
             width: 360px;}
#header { margin-top: 20px;
          margin-bottom: 10px;
          padding-bottom: 10px;
          border-bottom: 1px solid black;}
#alfabeto { float: left;
            width: 30px;
            padding-right: 10px;
            margin-right: 10px; }
#lexico { float: left;
          width: 300px;}
</style>
</head>
<body>
<div id="container">
<div id="header">
<h2>Pequeño léxico informático</h2>
</div>
<div id="alfabeto">
<div id="letra_a">
<h3><a href="#">A</a></h3>
</div>
<div id="letra_b">
<h3><a href="#">B</a></h3>
</div>
<div><h3>C</h3></div>
<div><h3>D</h3></div>
```

```
<div><h3>E</h3></div>
<div><h3>F</h3></div>
<div><h3>G</h3></div>
<div><h3>Etc</h3></div>
</div>
<div id="lexico">
</div>
</div>
</body>
</html>
```

Introducción

Desde su origen, jQuery es un framework JavaScript completo. Incorpora asimismo una serie de funciones llamadas utilidades. No es posible revisarlas todas en este libro; se pueden consultar en la documentación de jQuery, en la dirección: http://api.jquery.com/category/utilities

Aquí vamos a mostrar alguna de ellas.

Evitar los conflictos

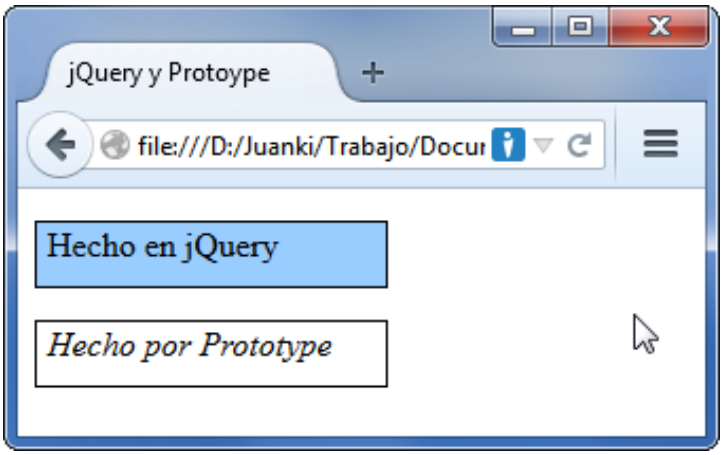
Los frameworks JavaScript, como jQuery, Mootools o Prototype, se usan muy frecuentemente para el desarrollo de las aplicaciones recientes. Cuando conviven, normalmente hay problemas ya que el signo $ lo usan todos ellos. Recuerde que jQuery usa el $ como alias de "jQuery".

El método `jQuery.noConflict()` permite evitar los posibles conflictos con otros frameworks. De esta manera, la llamada a $ en el código del script no se considerará como código jQuery y se reservará a las otras librerías. La llamada inicial jQuery se tomará como código jQuery.

Para más detalles, puede leer: http://learn.jquery.com/using-jquery-core/avoid-conflicts-other-libraries

Ejemplo

Supongamos dos capas. El contenido de una lo gestiona jQuery y el de la otra, Prototype.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery y Protoype</title>
<script src="prototype.js"></script>
<script src="jquery.js"></script>
<style>
div { width: 160px;
      height: 30px;
      border: 1px solid black;
      margin-bottom: 15px;
      padding-left: 5px;}
.jquery { background-color: #9cf;
          margin-top: 15px;}
.prototype { background-color: white;
             font-style: italic; }
</style>
</head>
<body>
<div id="jquery">Hecho en jQuery</div>
<div id="prototype">Hecho por Prototype</div>
<script>
// Parte jQuery
jQuery.noConflict();
jQuery('#jquery').addClass('jquery');
// Parte Prototype
$('prototype').addClassName('prototype');
</script>
</body>
</html>
```

Explicaciones.

```
// Parte jQuery
jQuery.noConflict();
```

El método `jQuery.noConflict()` indica a jQuery que no tenga en cuenta las variables $.

```
jQuery('#jquery').addClass('jquery');
```

jQuery sustituye el signo $.

```
// Parte Prototype
$('prototype').addClassName('prototype');
```

Instrucción gestionada por Prototype. El signo $ ya no se considera como jQuery, gracias a la instrucción `jQuery.noConflict()`.

Iteraciones en jQuery

El bucle for en JavaScript clásico requiere siempre especificar múltiples datos (por ejemplo, `for (i=1; i<6; i++)`). Hay que indicar el nombre de la variable del contador, así como su valor inicial, la condición que fija el límite del bucle y, para terminar, una instrucción que incrementa (o decrementa) el contador.

Para hacer esto, jQuery propone el método `each()`.

each(función)

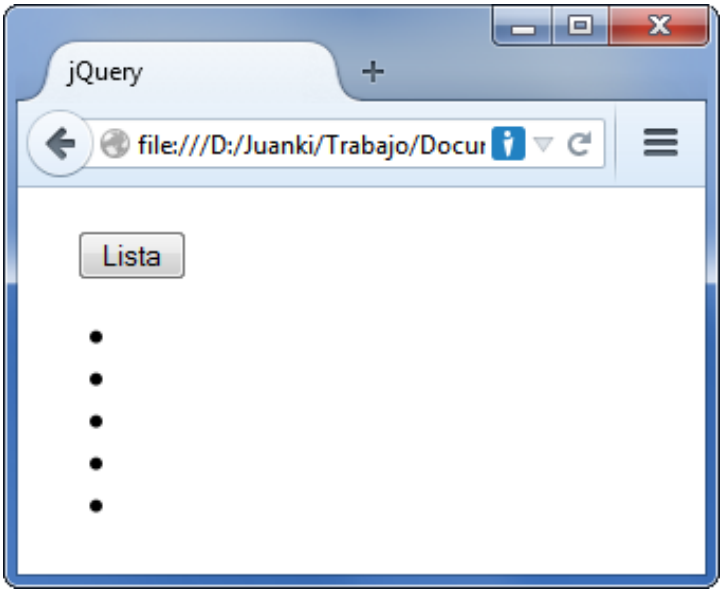
Ejecuta la función que se pasa como argumento por cada aparición del objeto seleccionado.

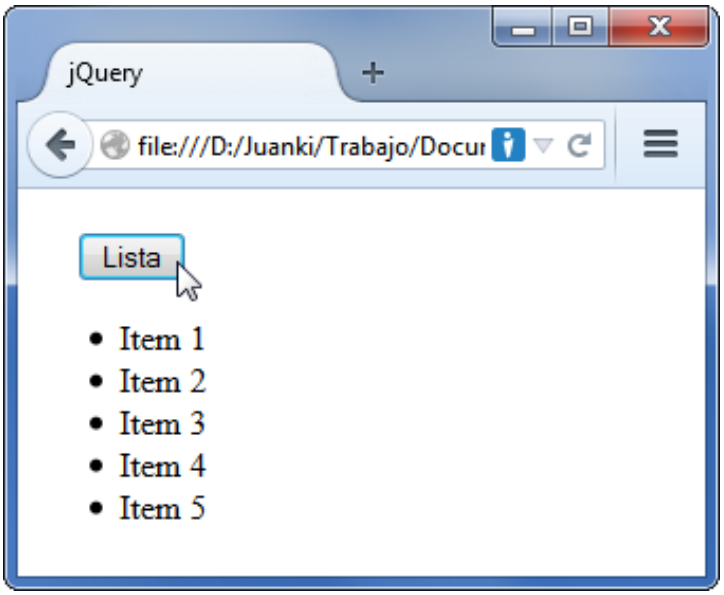
La función debe tener un argumento (un entero), que representará la posición del elemento que se está tratando actualmente.

```
$("img").each(function(i){
this.src = "test" i ".jpg";
});
```

Ejemplo

Al hacer clic en un botón, el script rellena los elementos de una lista numerada.





```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(  function() {
$("button").click(function() {
$("#lista").find("li").each( function(i) {
$(this).html( $(this).html() + " Item " + (i+1) );
});
});
});
</script>
<style>
.boton { margin-top: 12px;
         margin-left: 20px;}
</style>
</head>
<body>
<button class="boton">Lista</button>
<ul id="lista">
<li></li>
<li></li>
<li></li>
<li></li>
<li></li>
</ul>
</body>
</html>
```

Explicaciones.

```
$(document).ready( function() {
$("button").click(function() {
```

Cuando se carga el DOM y al hacer clic en el botón.

```
$("#lista").find("li").each( function(i) {
$(this).html( $(this).html() + " Item " + (i+1) );
});
```

El script hace un bucle por cada elemento (each) de la lista (`$("#lista").find("li")`). Observe el argumento de la función asociada a `each() (function(i))`. Por cada elemento `<li>` encontrado, la función añade la mención "Ítem" y el valor de `i` aumentado en 1.

```
});
});
```

Fin del script.

Almacenar y recuperar datos

El método `data()` permite almacenar y recuperar cualquier dato.

Almacenar datos

Para asociar un valor a un elemento de la página, por ejemplo una división `<div>`.

```
data(clave, valor)
```

- `clave`. Cadena de caracteres que especifica el nombre dado a los datos almacenados.
- `valor`. Los datos almacenados. En forma de cadena de caracteres, array u objeto.

```
$("div").data("numero",2015);
```

Recuperar datos almacenados

Para recuperar en el script un valor almacenado, basta con hacer una llamada con su nombre.

```
data(clave)
```

- `clave`. Cadena de caracteres que recupera el nombre dado a los datos almacenados.

```
$("div").data("numero")
```

Eliminar datos almacenados

Para no saturar la memoria del navegador, puede ser útil eliminar los datos almacenados que ya no sirven.

```
removeData(clave)
```

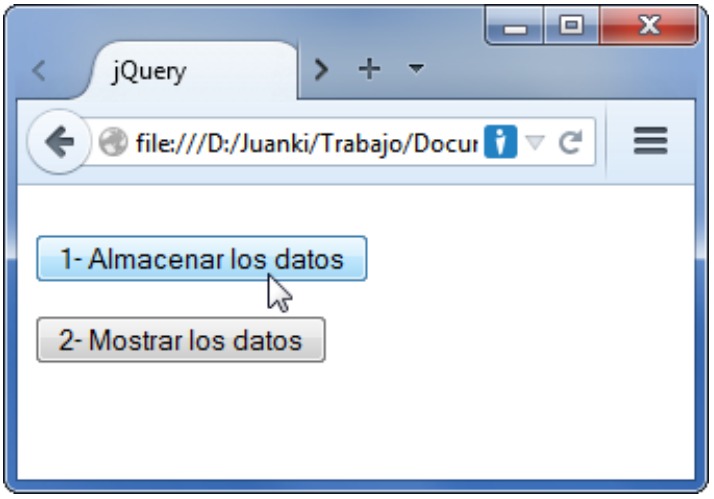
- `clave`. Cadena de caracteres que recupera el nombre dado a los datos almacenados

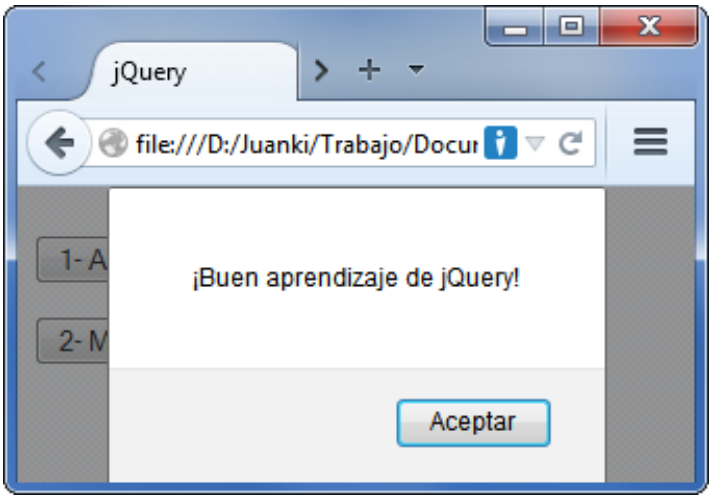
```
$("div").removeData("numero");
```

Para los expertos en JavaScript, el método `data()` puede ser un truco para transformar una variable local en global.

Ejemplo

Creemos una página donde con un clic de un botón, se almacenarán datos. Otro botón mostrará estos datos en un cuadro de diálogo.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<style>
button { margin-top: 15px;}
</style>
<script>
$(document).ready(function(){
$("#boton1").click(function(){
$("div").data("quiere", "¡Buen aprendizaje de jQuery!");
});
$("#boton2").click(function(){
alert($("div").data("quiere"));
});
});
</script>
</head>
<body>
<button id="boton1">1- Almacenar los datos</button><br>
<button id="boton2">2- Mostrar los datos</button>
<div></div>
</body>
</html>
```

Comentarios

```
$("#boton1").click(function(){
```

```
$("div").data("quiere", "¡Buen aprendizaje de jQuery!");
});
```

Al hacer clic (`click`) en el botón 1 (`#boton1`), el método `data()` almacena en la división `<div>`, en la clave `quiere`, la cadena de caracteres `¡Buen aprendizaje de jQuery!` Esta acción no tiene ningún efecto visual.

Encontrar un elemento del DOM

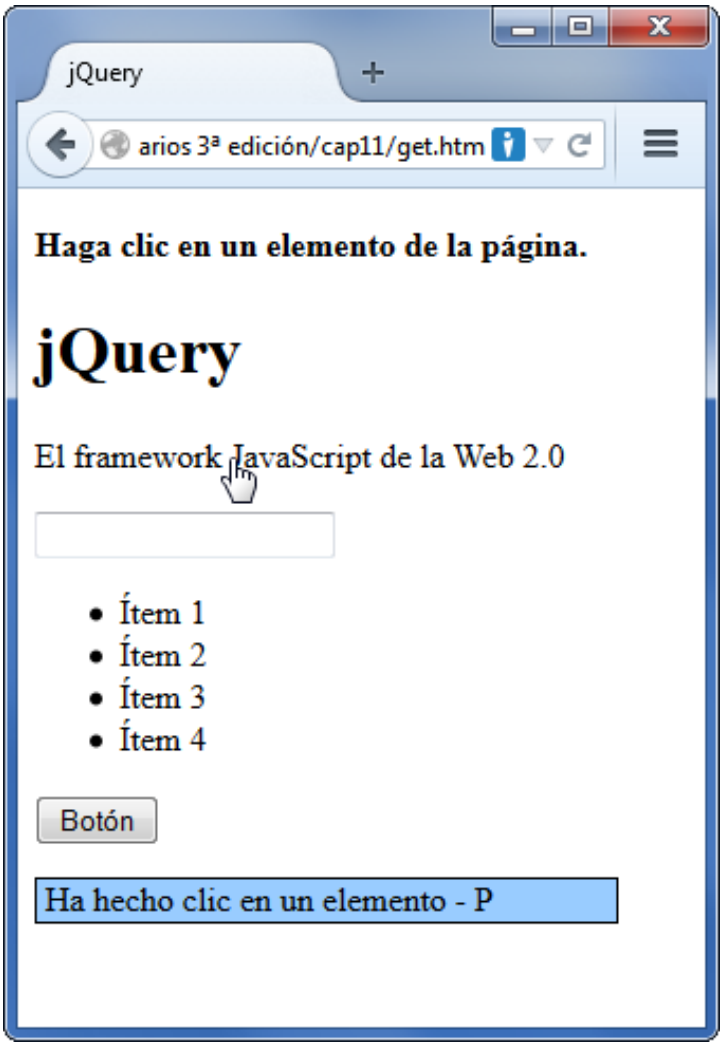
El método `get()` recupera los elementos del DOM especificados por un selector.

```
$(selector).get(index)
```

o `index` (opcional) determina el enésimo elemento (numeración JavaScript).

Ejemplo

Encontremos la etiqueta del elemento sobre el que se hace un clic en una página.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<style>
div { width: 270px;
      border: 1px solid black;
      background-color: #9cf;
      margin-top: 15px;
      padding-left: 4px;}
body { cursor: pointer;}
</style>
</head>
<body>
<p><b>Haga clic en un elemento de la página.</b></p>
```

```
<h1>jQuery</h1>
<p>El framework JavaScript de la Web 2.0</p>
<input type="text">
<ul>
<li>Ítem 1</li>
<li>Ítem 2</li>
<li>Ítem 3</li>
<li>Ítem 4</li>
</ul>
<button>Boton</button>
<div> </div>
<script>
$("*", document.body).click(function(event) {
event.stopPropagation();
var element_dom = $(this).get(0);
$("div").text( "Ha hecho clic en un elemento - " +
element_dom.nodeName );
});
</script>
</body>
</html>
```

Comentario

```
$("*", document.body).click(function(event) {
```

Al hacer clic en un elemento de la página (`$("*", document.body)`)

```
event.stopPropagation();
```

Impide la propagación del evento a los elementos padres.

```
var element_dom = $(this).get(0);
$("div").text( "Ha hecho clic en un elemento - " +
element_dom.nodeName );
});
```

La variable `element_dom` contiene la etiqueta (`get(0)`) del elemento en el que se ha hecho clic (`$(this)`). Ésta (`element_dom.nodeName`) se muestra entonces en una división.

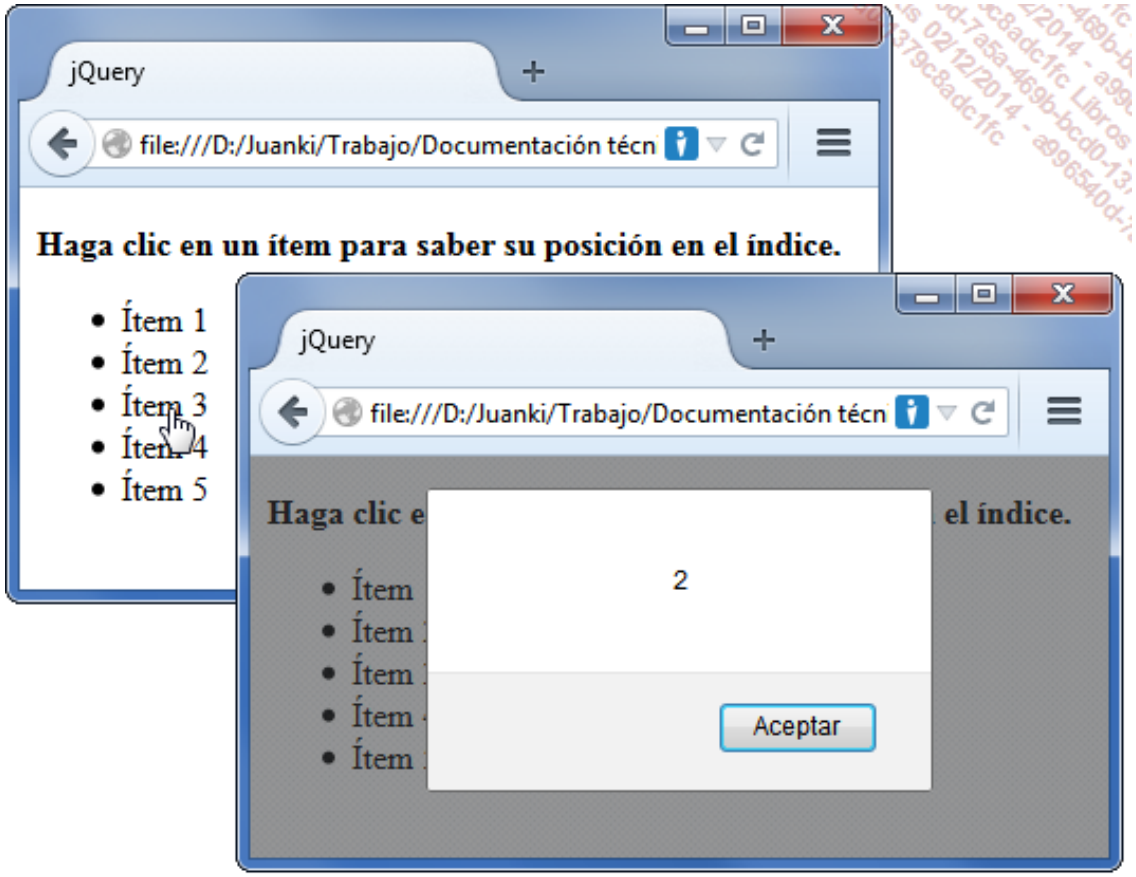
Buscar un elemento en concreto

El método `index()` devuelve la posición de índice de elementos especificados por un selector jQuery o un elemento del DOM. Si no se encuentra ningún elemento, el método devuelve el valor `-1`.

```
$(selector).index()
```

Ejemplo

Consideremos una lista de cinco elementos. Al hacer clic en un ítem, su posición se muestra en un cuadro de diálogo.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("li").on("click", function(){
alert($(this).index());
});
});
</script>
</head>
<body>
<p><b>Haga clic en un ítem para saber su posición en el índice.</b></p>
<ul>
<li>Ítem 1</li>
<li>Ítem 2</li>
<li>Ítem 3</li>
<li>Ítem 4</li>
<li>Ítem 5</li>
```

```
</ul>
</body>
</html>
```

Comentario

```
$("li").on("click", function(){
alert($(this).index());
});
});
```

Al hacer clic en un elemento de lista (`$("li")`), un cuadro de diálogo (`alert`) muestra la posición de índice (`index()`) del elemento que se ha hecho clic (`$(this)`).

Conocer el número de elementos

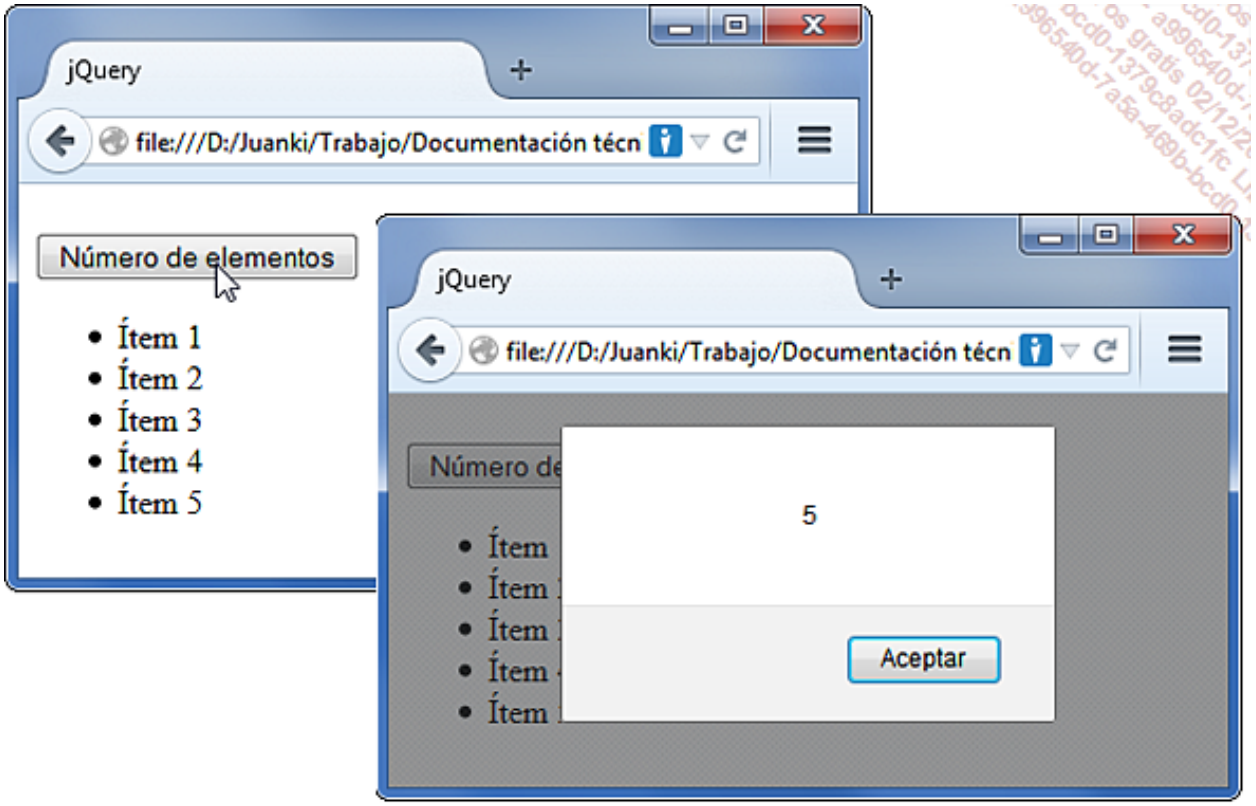
El método `length()` devuelve el número de elementos que corresponden a un selector.

El método `size()` hace la misma función pero está obsoleto (*deprecated*) desde la versión 1.8 de jQuery.

```
$(selector).length()
```

Ejemplo

Al hacer clic en un botón, mostramos en un cuadro de diálogo, el número de elementos de una lista.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<style>
button { margin-top: 15px;}
</style>
<script>
$(document).ready(function(){
$("button").on("click", function(){
alert($("li").length);
});
});
</script>
</head>
<body>
<button>Número de elementos</button>
<ul>
<li>Ítem 1</li>
<li>Ítem 2</li>
<li>Ítem 3</li>
```

```
<li>Ítem 4</li>
<li>Ítem 5</li>
</ul>
</body>
</html>
```

Comentario

```
$("button").on("click", function(){
alert($("li").length);
});
});
```

Al hacer clic en el botón (`$("button")`), se muestra un cuadro de diálogo (`alert`) con el número de elementos (`length`) de la lista (`$("li")`).

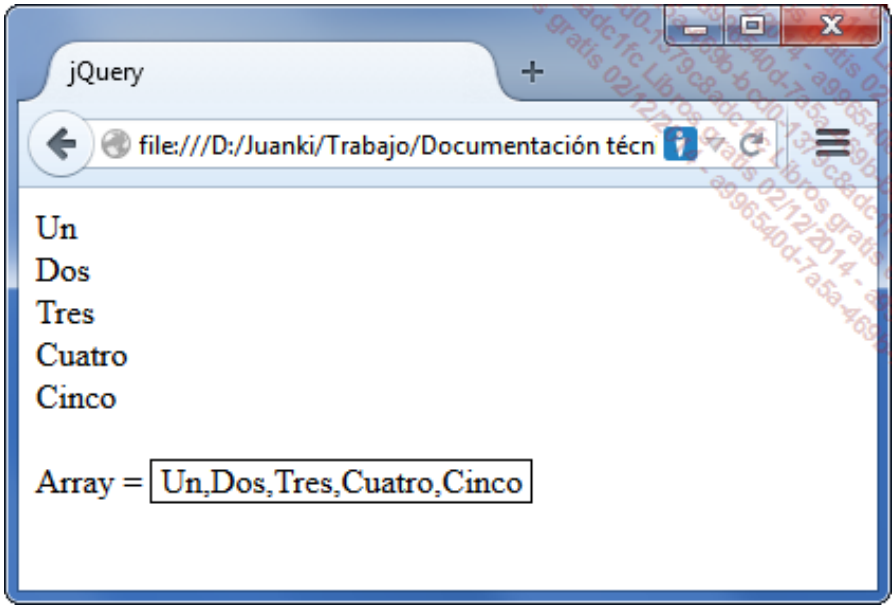
Convertir en Array los elementos encontrados

El método `toArray()` devuelve los elementos del DOM en forma de Array.

```
$(Selector).toArray()
```

Ejemplo

Agrupemos en un Array los elementos de una lista.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
span { border: 1px solid black;
       padding-left: 4px;
       padding-right: 4px;}
p { margin-top: 20px;}
</style>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
var y = [];
x=$("div").toArray()
for (i=0;i<x.length;i++){
y.push(x[i].innerHTML);
}
$("span").text(y);
});
</script>
</head>
<body>
<div>Uno</div>
<div>Dos</div>
<div>Tres</div>
<div>Cuatro</div>
<div>Cinco</div>
<p>Array = <span></span></p>
</body>
</html>
```

Comentarios

```
var y = [];
```

Definición de y como una variable Array.

```
x=$("div").toArray()
```

La variable `x` contendrá las etiquetas `<div>` (`$("div")`) en forma de un Array (`toArray()`).

```
for (i=0;i<x.length;i++){
y.push(x[i].innerHTML);
}
```

El bucle `for` pasa por los diferentes elementos de la variable Array `x`. Para cada elemento (`x[i]`), la función `push()` añade el contenido de éste (`innerHTML`) al final del Array `y`.

```
$("span").text(y);
});
```

Al final del bucle, la variable `y` se muestra (`text()`) en la etiqueta `<span>` (`$("span")`).

Introducción

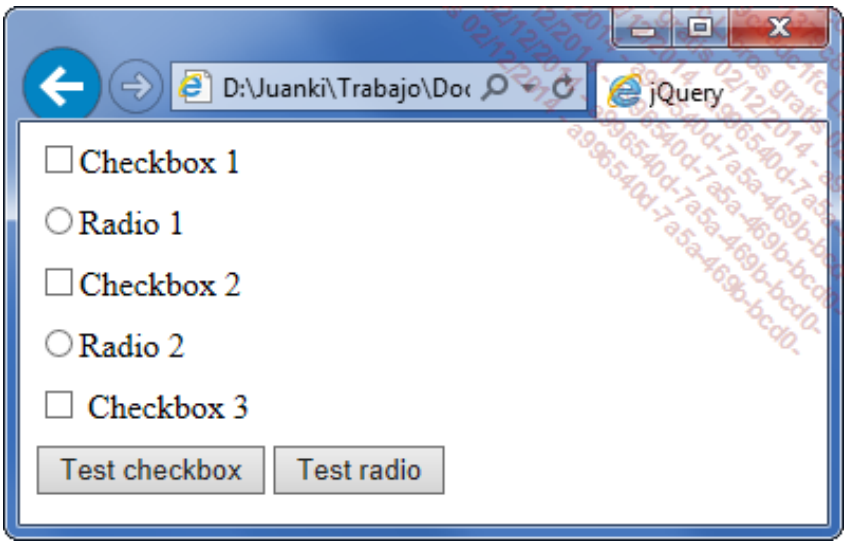
Aunque jQuery no aporta nada nuevo en esta materia, dedicamos este capítulo a los formularios, ya que siempre representan un desarrollo particular en la elaboración de las aplicaciones Web.

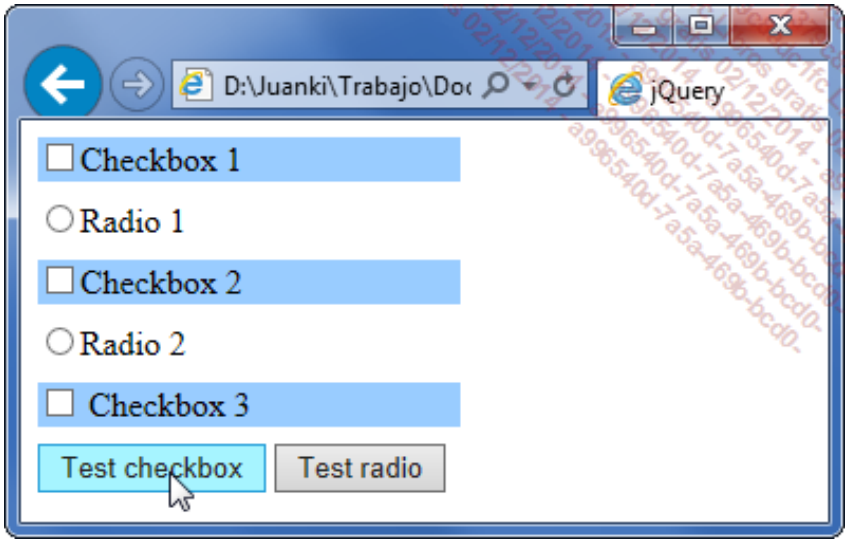
Los selectores de formularios

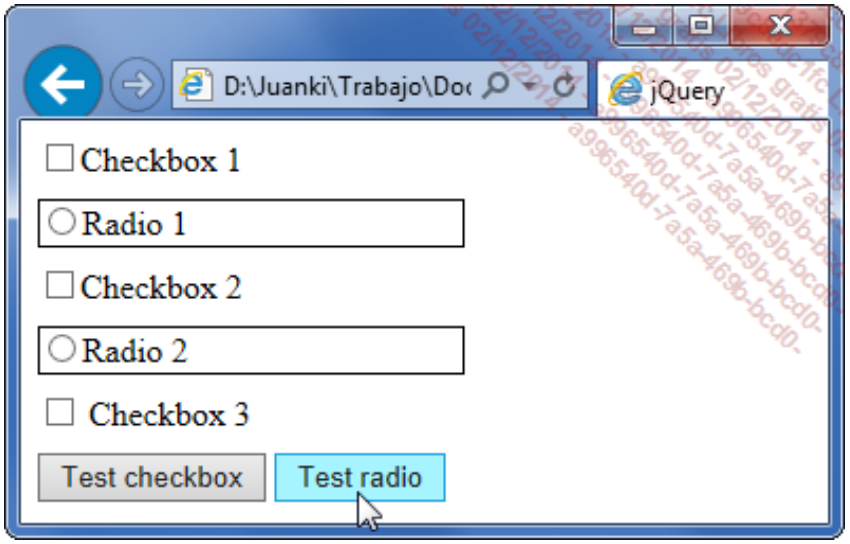
:input	Selecciona todos los elementos del formulario (input, textarea, select y botones).
:text	Selecciona todos los elementos del formulario de tipo línea de texto.
:password	Selecciona todos los elementos del formulario de tipo contraseña.
:radio	Selecciona todos los elementos del formulario de tipo botones de radio.
:checkbox	Selecciona todos los elementos del formulario de tipo casillas de selección.
:submit	Selecciona todos los elementos del formulario de tipo submit.
:image	Selecciona todos los elementos del formulario de tipo imagen.
:reset	Selecciona todos los elementos del formulario de tipo reset.
:button	Selecciona todas las etiquetas `<button>` y todos los elementos de tipo button.
:file	Selecciona todos los elementos del formulario de tipo archivo.
:hidden	Selecciona todos los elementos del formulario de tipo hidden.
:focus	Selecciona el elemento del formulario que tiene el foco. Se introdujo en la versión 1.6 de jQuery.
:enabled	Selecciona todos los elementos activos.
:disabled	Selecciona todos los elementos no activos.
:checked	Conserva todos los elementos marcados o seleccionados.
:selected	Conserva todos los elementos seleccionados.

Ejemplo

Supongamos las siguientes casillas de selección y botones de radio. Al hacer clic en un botón, obtenemos las primeras, y al hacer clic en el otro, los segundos.







El archivo de inicio:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<style>
div { width: 200px; height: 21px;
      margin-bottom: 8px;}
</style>
</head>
<body>
<div><input type="checkbox">Checkbox 1</div>
<div><input type="radio">Radio 1<br></div>
<div><input type="checkbox">Checkbox 2</div>
<div><input type="radio">Radio 2</p>
<div><input type="checkbox">Checkbox 3</div>
<button id="test1">Prueba checkbox</button>
<button id="test2">Prueba radio</button>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$("#test1").on("click",function(){
$("input:checkbox").parent().css({"background": "#9cf"});
});
$("#test2").on("click",function(){
$("input:radio").parent().css({"border": "1px solid black"});
});
});
</script>
```

Explicaciones.

```
$("test1").on("click",function(){
$("input:checkbox").parent().css({"background": "#9cf"});
});
```

Cuando se hace clic en el botón (`$("test1")`), el script encuentra las casillas de selección mediante el selector (`$("input:checkbox")`), sube al elemento padre (`parent()`), es decir, las divisiones`<div>` y modifica la propiedad de estilo `(css)` cambiando el color de fondo.

```
$("#test2").on("click",function(){
$("input:radio").parent().css({"border": "1px solid black"});
});
```

Al hacer clic en el botón (`$("#test2")`), el script busca los botones radio mediante el selector (`$("input:radio")`), sube al elemento padre (`parent()`), es decir, las divisiones `<div>` y modifica la propiedad de estilo (`css`) del borde.

El archivo final:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<style>
div { width: 200px; height: 21px;
      margin-bottom: 8px;}
</style>
</head>
<body>
<div><input type="checkbox">Checkbox 1</div>
<div><input type="radio">Radio 1</div>
<div><input type="checkbox">Checkbox 2</div>
<div><input type="radio">Radio 2</div>
<div><input type="checkbox"> Checkbox 3</div>
<button id="test1">Test checkbox</button>
<button id="test2">Test radio</button>
<script>
$(document).ready(function(){
$("#test1").on("click",function(){
$("input:checkbox").parent().css({"background": "#9cf"});
});
$("#test2").on("click",function(){
$("input:radio").parent().css({"border": "1px solid black"});
});
});
</script>
</body>
</html>
```

Los filtros de selección

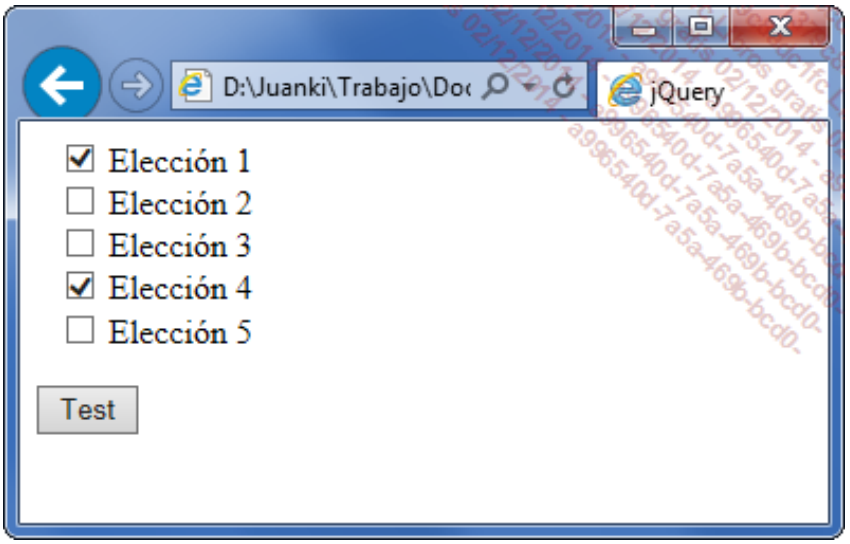
1. Los elementos marcados

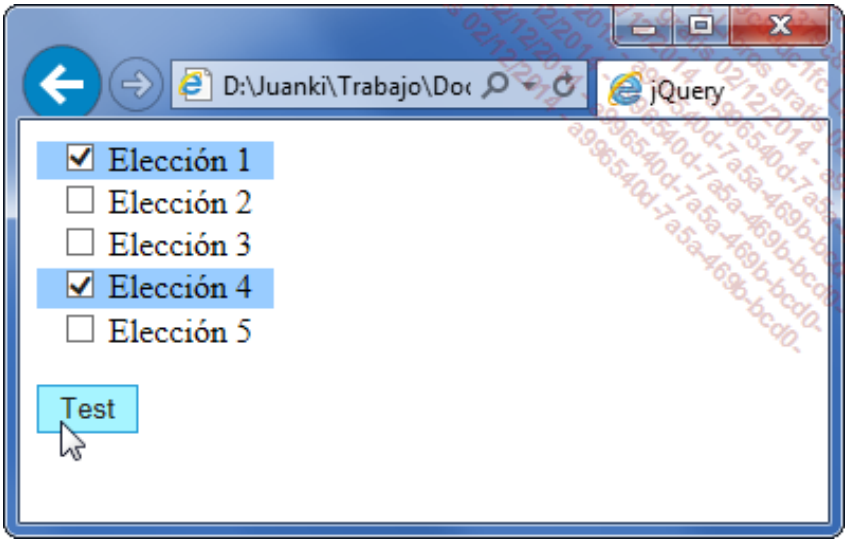
:checked

Devuelve todos los elementos de tipo botón de radio o casillas de selección seleccionados.

Ejemplo

Al hacer clic en el botón, buscamos los elementos seleccionados de las casillas de selección.





El archivo de inicio:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<style>
span { padding:  0 10px 0 10px;}
</style>
</head>
```

```
<body>
<form action="">
<span><input type="checkbox" name="box" checked=> Elección 1
</span><br>
<span><input type="checkbox" name="box"> Elección 2</span><br>
<span><input type="checkbox" name="box"> Elección 3</span><br>
<span><input type="checkbox" name="box" checked> Elección 4
</span><br>
<span><input type="checkbox" name="box"> Elección 5</span><br>
</form>
<p><button>Test</button></p>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$(":button").on("click", function () {
$("input:checked").parent().css({"background": "#9cf"});
});
});
</script>
```

El script sigue el enfoque anterior. Por lo tanto, no son necesarias explicaciones adicionales. Simplemente observe el selector `input:checked`, que permite recuperar solo los campos del formulario que están marcados.

El código final es el siguiente:

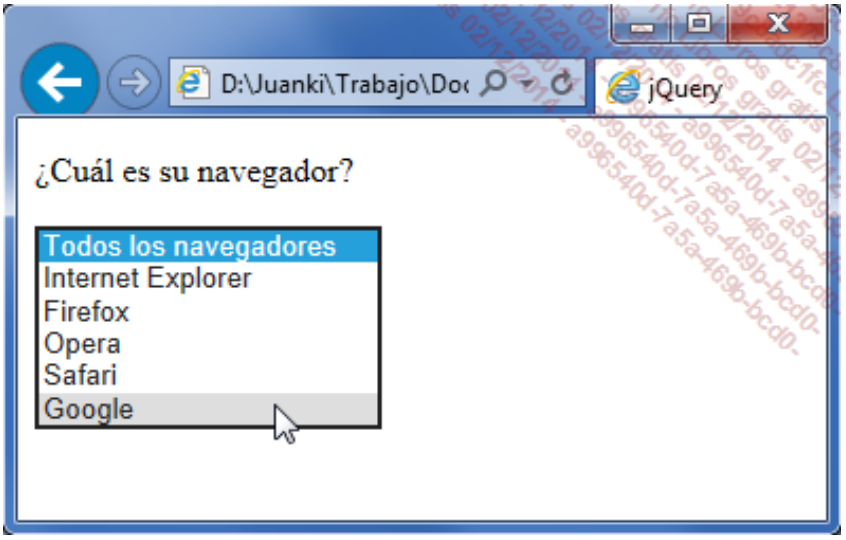
```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$(":button").on("click",function () {
$("input:checked").parent().css({"background": "#9cf"});
});
});
</script>
<style>
span { padding: 0 10px 0 10px;}
</style>
</head>
<body>
<form action="">
<span><input type="checkbox" name="box" checked>
Elección 1</span><br>
<span><input type="checkbox" name="box"> Elección 2</span><br>
<span><input type="checkbox" name="box"> Elección 3</span><br>
<span><input type="checkbox" name="box" checked> Elección 4
</span><br>
<span><input type="checkbox" name="box"> Elección 5</span><br>
</form>
<p><button>Test</button></p>
</body>
</html>
```

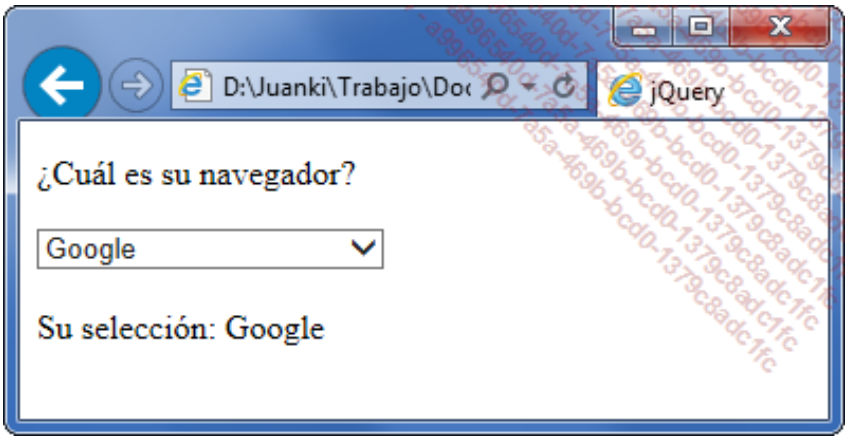
2. Los elementos seleccionados

:selected

Devuelve todos los elementos de una lista de elección que están seleccionados.

Ejemplo





El archivo Html inicial es el siguiente:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
</head>
<body>
<p>¿Cuál es su navegador?</p>
<form action="">
<select id="navegadores">
<option selected="selected" value="0">Todos los
navegadores</option>
<option value="1">Internet Explorer</option>
<option value="2">Firefox</option>
<option value="3">Opera</option>
<option value="4">Safari</option>
<option value="5">Google</option>
</select>
</form>
<br>
<div id="resultado"></div>
</body>
</html>
```

El script se presenta de la siguiente manera:

```
<script>
$(document).ready(function(){
$("#navegadores").change(onSelectChange);
});
function onSelectChange(){
var selected = $("#navegadores option:selected");
var output = "";
if(selected.val() != 0){
output = "Su selección: " + selected.text();
}
$("#resultado").html(output);
}
</script>
```

Explicaciones.

```
$(document).ready(function(){
$("#navegadores").change(onSelectChange);
});
```

Cuando se hace un cambio en la lista de selección (`change`), se llama a la función `onSelectChange`.

```
function onSelectChange(){
var selected = $("#navegadores option:selected");
var output = "";
```

En primer lugar, la función `onSelectChange` define la variable `selected` para conservar el valor de la lista seleccionado por el usuario (`$("#navegadores option:selected")`). Se inicializa la variable `output`, que contendrá el valor de la selección del usuario.

```
if(selected.val() != 0){
output = "Su selección: " + selected.text();
}
```

Si el atributo valor de la selección es diferente de 0 (`(selected.val() != 0)`), se construye la variable `output`, teniendo en cuenta la selección hecha.

```
$("#resultado").html(output);
}
```

El valor de la variable `output` se mostrará en la capa identificada por resul-tado.

```
}
```

Fin del script.

El archivo completo es el siguiente:

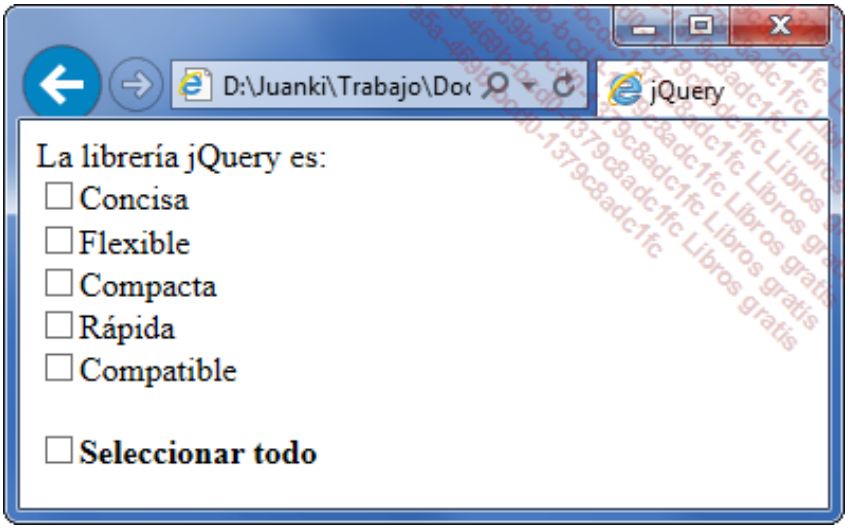
```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#navegadores").change(onSelectChange);
});
```

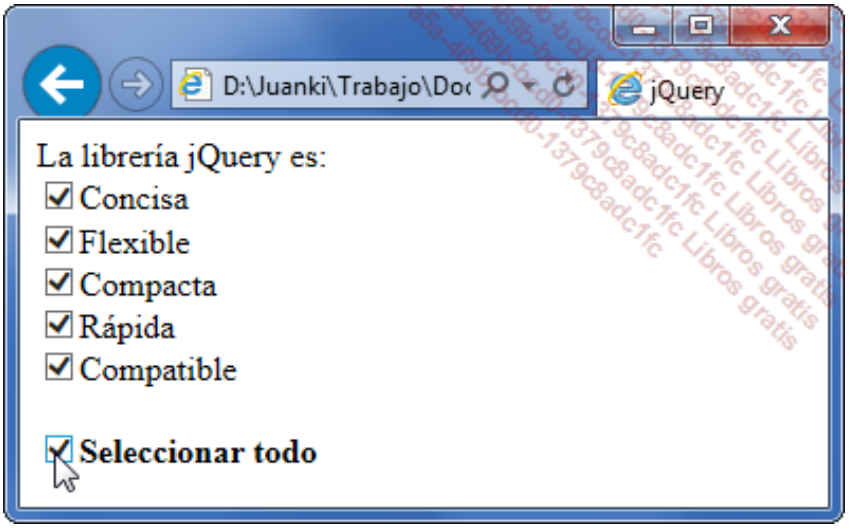
```
function onSelectChange(){
var selected = $("#navegadores option:selected");
var output = "";
if(selected.val() != 0){
output = "Su selección: " + selected.text();
}
$("#resultado").html(output);
}
</script>
</head>
<body>
<p>¿Cuál es su navegador?</p>
<form action="">
<select id="navegadores">
<option selected="selected" value="0">Todos los navegadores
</option>
<option value="1">Internet Explorer</option>
<option value="2">Firefox</option>
<option value="3">Opera</option>
<option value="4">Safari</option>
<option value="5">Google</option>
</select>
</form>
<br>
<div id="resultado"></div>
</body>
</html>
```

Aplicaciones

1. Seleccionar todas las casillas de selección

Muy popular en los sitios actuales, este script ofrece la posibilidad al usuario de activar todas las casillas de selección con una sola acción.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#todos").click(function(){
var checked_status = this.checked;
$("input[name=casilla]").each(function(){
this.checked = checked_status;
});
});
});
</script>
</head>
<body>
```

```
La librería jQuery es:<br />
<input type="checkbox" name="casilla">Concisa<br>
<input type="checkbox" name="casilla">Flexible<br>
<input type="checkbox" name="casilla">Compacta<br>
<input type="checkbox" name="casilla">Rápida<br>
<input type="checkbox" name="casilla">Compatible<br>
<br />
<input type="checkbox" id="todos"><b>Seleccionar todo</b>
</body>
</html>
```

El script jQuery:

```
<script>
$(document).ready(function(){
$("#todos").click(function(){
var checked_status = this.checked;
$("input[name=casilla]").each(function(){
this.checked = checked_status;
});
});
});
</script>
```

Explicaciones:

```
$(document).ready(function(){
$("#todos").click(function(){
```

Inicialización de jQuery y al hacer clic en la casilla de selección que permite seleccionar todos los elementos.

```
var checked_status = this.checked;
```

La variable `checked_status` comprueba que se marca la casilla.

```
$("input[name=casilla]").each(function(){
this.checked = checked_status;
});
```

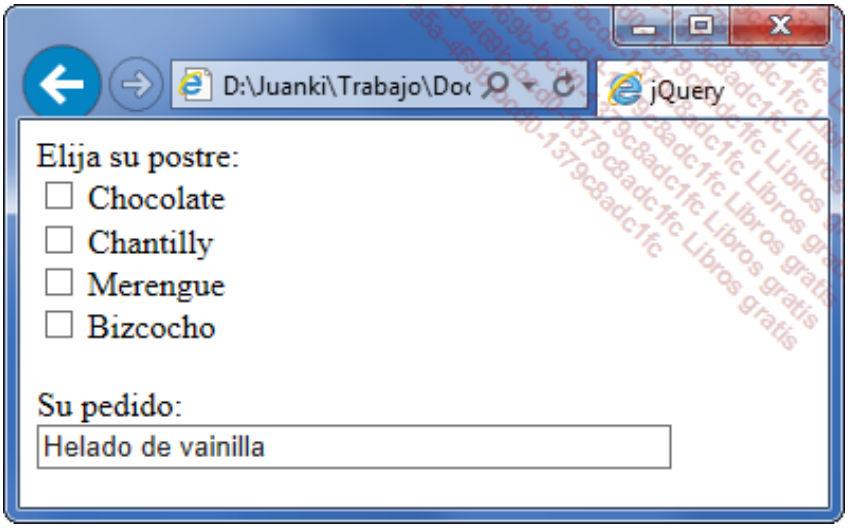
El método `each()` revisa todas las casillas de selección (`input[name=casilla]`) y las marca.

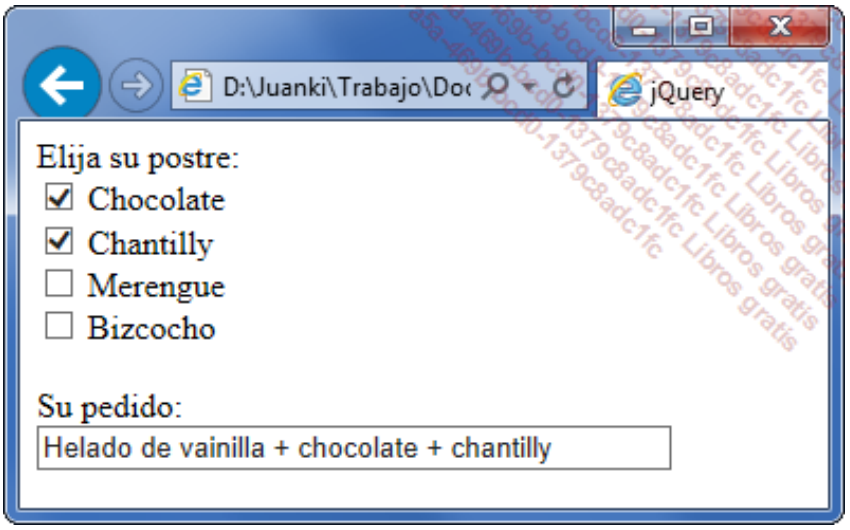
```
});
});
```

Fin del script.

2. Confirmar un comando

Reunimos en una línea de texto los elementos marcados de un formulario.






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script>
$(document).ready(function() {
$('input:checkbox[name=postre]').click(function(){
var check = this.checked;
var val = $(this).val();
var keys = $("input[name='comando']").val().split(', ');
var newKeys = new Array();
var inArray = false;
$.each(keys, function(i){
var key=this;
if (key == val) {
if (check){
newKeys.push(key);
}
inArray = true;
}
else {
newKeys.push(key);
}
});
if (!inArray && check) newKeys.push(val);
$("input[name='comando']").val(newKeys.join(' + '));
```

```
});
});
</script>
</head>
<body>
Elija su postre:
<form action="">
<input value="chocolate" type="checkbox" name="postre">
Chocolate<br>
<input value="chantilly" type="checkbox" name="postre">
Chantilly<br>
<input value="merengue" type="checkbox" name="postre">
Merengue<br>
<input value="bizcocho" type="checkbox" name="postre">
Bizcocho<br>
<br>
Su pedido: <br>
<input type="text" name="comando" size="45" value="Helado
de vainilla">
</form>
</body>
</html>
```

Explicación del script:

```
$(document).ready(function() {
$('input:checkbox[name=postre]').click(function(){
```

Inicialización de jQuery y al hacer clic en una casilla de selección.

```
var check = this.checked;
var val = $(this).val();
```

La variable `check` comprueba que la casilla se ha marcado. La variable `val` devuelve el valor que corresponde a la casilla.

```
var keys = $("input[name='comando']").val().split(', ');
```

Preparación del contenido de la línea de texto que confirma el pedido, obteniendo el valor inicial del atributo value.

```
var newKeys = new Array();
var inArray = false;
```

Creación de una tabla (Array) llamada `newKeys`, destinada a contener los títulos de las casillas marcadas.

```
$.each(keys, function(i){
var key=this;
if (key == val) {
if (check){
newKeys.push(key);
}
inArray = true;
}
else {
newKeys.push(key);
}
});
```

El método JavaScript clásico `Array.push()` añade los títulos de las casillas marcadas al final de la tabla `newkeys`.

```
if (!inArray && check) newKeys.push(val);
$("input[name='comando']").val(newKeys.join(' + '));
```

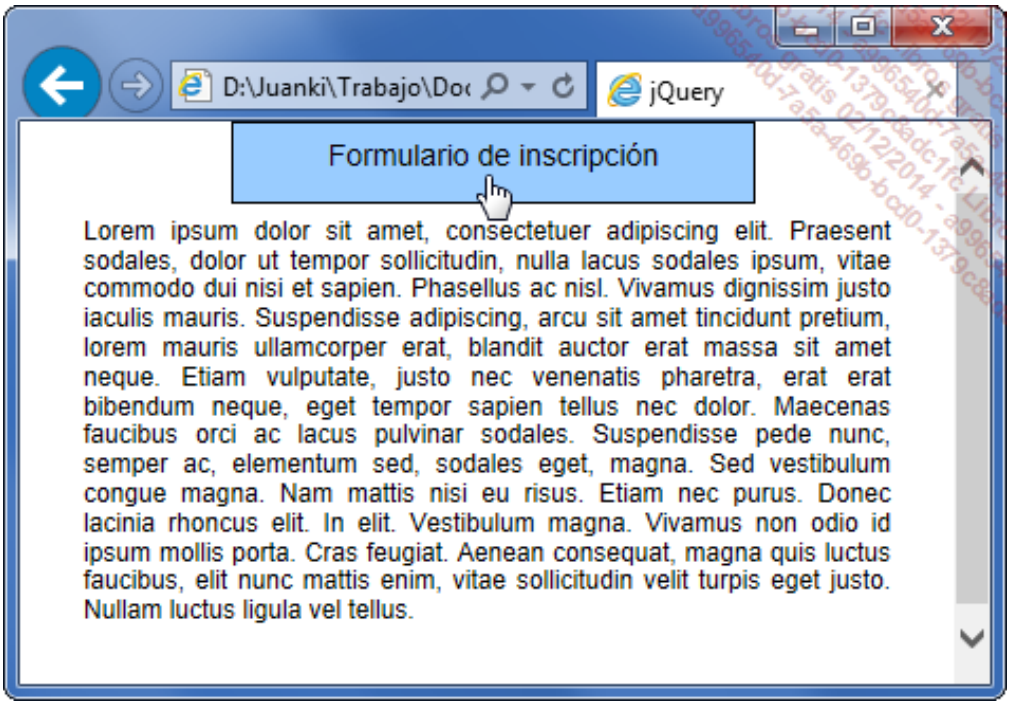
La confirmación del pedido se muestra en la línea de texto, modificando su atributo value.

```
});
});
```

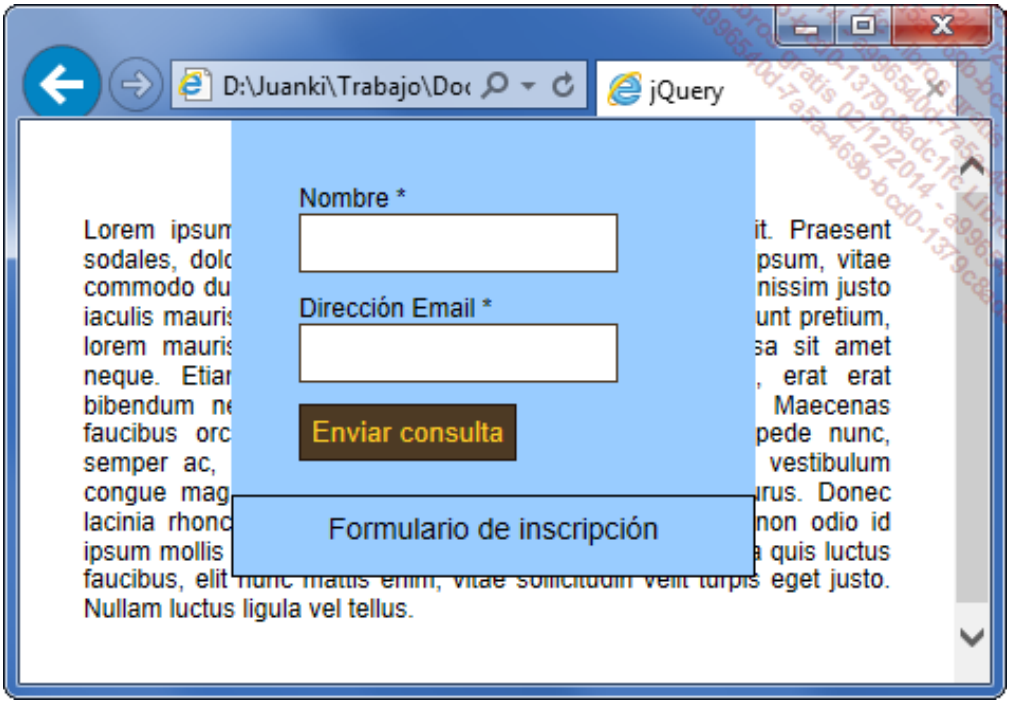
Fin del script.

3. Un formulario de inscripción original

Los formularios de contacto o inscripción son una parte imprescindible de cualquier sitio Web. Por lo general se implementan en una página separada y raramente son creativos. Este ejemplo ilustra cómo crear con jQuery un formulario disponible desde la página de inicio.



Al hacer clic en la capa **Formulario de inscripción**, aparece el formulario de contacto, deslizándose verticalmente hacia abajo.




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<style>
body{ margin: 0px;
      font-family: Arial, Sans-Serif;
      font-size: 0.75em;}
.box { margin: 0px auto;
       background-color: #ffffff;
       text-align: justify;
       position: relative;}
.content { padding: 20px 30px;}
#container { position: absolute;
             left: 100px;
             float: right;}
#contactForm { height: 177px;
               width: 248px;
               background-color: #9cf;
               display: none;}
#contactForm fieldset { padding: 30px;
                        border: none; }
#contactForm label { display: block;
                     color: black;}
#contactForm input[type=text] { display: block;
                                border: solid 1px #4d3a24;
                                margin-bottom: 10px;
                                height: 24px;}
#contactForm input[type=submit] { background-color: #4d3a24;
                                  border: solid 1px #23150c;
                                  color: #fecd28;
                                  padding: 5px;}
#contact { height: 30px;
           width: 246px;
           background-color: #9cf;
           border: 1px solid black;
           display: block;
           cursor: pointer;
           font-size: 14px;
           padding-top: 7px;
```

```
            text-align: center;}
p { margin-top: 25px;}
</style>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
$("#contact").click(function(){
if ($("#contactForm").is(":hidden")){
$("#contactForm").slideDown("slow");
}
else{
$("#contactForm").slideUp("slow");
}
});
});
function closeForm(){
$("#contactForm").slideUp("slow");
}
</script>
</head>
<body>
<div class="box">
<div id="container">
<div id="contactForm">
<fieldset>
<label for="Name">Nombre *</label>
<input id="name" type="text">
<label for="Email">Dirección Email *</label>
<input id="Email" type="text">
<input id="envio" type="submit" name="submit"
onclick="closeForm()">
</fieldset>
</div>
<div id="contact">Formulario de inscripción</div>
</div>
<div class="content">
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Praesent sodales, dolor ut tempor sollicitudin, nulla lacus
sodales ipsum, vitae commodo dui nisi y sapien. Phasellus ac
nisl. Vivamus dignissim justo iaculis mauris. Suspendisse
adipiscing, arcu sit amet tincidunt pretium, lorem mauris
ullamcorper erat, blandit auctor erat massa sit amet neque.
Etiam vulputate, justo nec venenatis pharetra,... </p>
</div>
</div>
</body>
</html>
```

El script jQuery es (en este punto de su aprendizaje) relativamente sencillo.

```
$(document).ready(function(){
$("#contact").click(function(){
```

Cuando se carga el DOM y al hacer clic en la capa `contact`.

```
if ($("#contactForm").is(":hidden")){
$("#contactForm").slideDown("slow");}
```

Si el formulario de contacto (`contactForm`) está oculto (`is(":hidden")`), éste aparece deslizándose lentamente hacia abajo (`slideDown("slow")`).

```
else{
$("#contactForm").slideUp("slow");
}
```

En caso contrario, sube (`slideUp("slow")`) para volver a su posición inicial.

```
});
});
```

Fin del ready.

```
function closeForm(){
$("#contactForm").slideUp("slow");
}
```

Falta por definir la función (`closeForm()`), asociada al botón de envío. Ésta hace que el formulario (`slideUp("slow")`) vuelva a su posición inicial.

Introducción

Los plug-ins jQuery son scripts dedicados a tareas específicas, como ordenar una tabla, la implementación de un carrusel de imágenes o la validación de formularios.

Estos plug-ins, que se empezaron a construir por la comunidad de jQuery, son numerosos, en general de excelente calidad y normalmente disponibles de forma libre en la Web. Puede hacerse una idea de su variedad en la dirección: http://plugins.jquery.com

Estos plug-ins ahorran tiempo, evitando reescribir un código que ya se ha escrito antes. Algunas veces también permiten ejecutar operaciones con un nivel de programación superior al de un diseñador medio en jQuery.

Diseñar un plug-in jQuery

1. Aspectos teóricos

La escritura de un plug-in jQuery pasa por etapas concretas y hay que respetar determinadas reglas precisas.

El plug-in jQuery toma la forma de un archivo JavaScript externo (extensión .js) que se ubica inmediatamente después de la etiqueta de llamada de jQuery (ver la sección Usar un plug-in jQuery, en este capítulo):

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

Asignar un nombre al plug-in

Un plug-in siempre se debe nombrar con el formato jquery.nombre_del_plug-in.js. De esta manera, se identifica inmediatamente.

Aislar el código

En este nuevo archivo js, hay que englobar el código del plug-in en una función anónima. De esta manera, todas las variables del plug-in no entrarán en conflicto con las de otros scripts de la página.

```
(function () {
// código jQuery
}) ()
```

Pasamos la variable jQuery usando su alias `$` a esta función, lo que permitirá usar la variable `$`dentro de ella.

```
(function ($) {
// código jQuery
}) ()
```

Añadir el nuevo método a jQuery

Casi estamos preparados para escribir nuestro plug-in. Hay que añadir este nuevo método a los objetos jQuery usando la instrucción `$.fn.nombre_del_plug-in`. En esta función, la palabra clave`this` representará al objeto jQuery seleccionado por el usuario del plug-in.

```
(function($) {
$.fn.nombre_del_plugin = function () {
// código jQuery
};
}) (jQuery)
```

La documentación de jQuery respecto al diseño de plug-ins insiste en que cada método o función debe terminar con un punto y coma. En caso contrario, se corre el riesgo de que el código no funcione después de comprimir el plug-in.

Éstas son las reglas fundamentales de la elaboración de un plug-in. Sin embargo, no hay que perder de vista los siguientes puntos:

- Cuando varios elementos son o pueden ser seleccionados, se recomienda el método `each()` para efectuar bucles en los elementos seleccionados.
- Es útil pasar argumentos al método del plug-in para facilitar la personalización de éste por el usuario.

- Salvo excepciones, el plug-in debe devolver el objeto que se trata (return this).

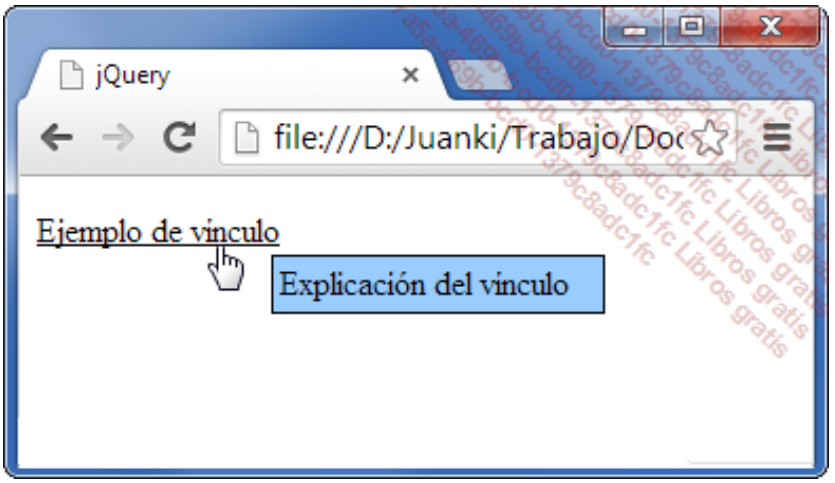
Para terminar, aunque normalmente no es imprescindible, hay que comprimir el código para reducir el tiempo de descarga del plug-in. Existe una pequeña aplicación en línea (http://dean.edwards.name/packer/) que hace esto a la perfección. Copie su código y péguelo en el campo del formulario, marque **Base62 encode** y/o **Shrink variables**, haga clic en el botón **Pack** y se muestra el resultado.

El código obtenido empieza como sigue:

```
eval(function(p,a,c,k,e,r){e=function(c){return(c
```

2. Una aplicación práctica

El plug-in de nuestro ejemplo añade un tooltip explicativo a los enlaces.



La primera etapa consiste en darle un nombre al plug-in. Siguiendo las convenciones de jQuery, lo vamos a llamar `jquery.mi_tooltip.js`.

Abra un editor de texto y comience con la codificación.

Empezamos por determinar el entorno jQuery.

```
(function($){
jQuery.fn.mi_tooltip = function(options) {
// código del plug-in
};
})(jQuery)
```

Ahora incluimos el código del plug-in propiamente dicho.

```
(function($){
jQuery.fn.mi_tooltip = function(options) {
var elemento = document.createElement("div");
$(element).addClass(options.tooltipcss).hide();
document.body.appendChild(element);
return this.each(function() {
$(this).hover(function() {
$(element).show();
$(element).html($(this).attr("rel"));
$(this).mousemove(function(e) {
$(element).css({ "position": "absolute",
                 "top": e.pageY + options.offsetY,
                 "left": e.pageX + options.offsetX
});
```

```
});
}, function() {
$(element).hide()
});
});
};
})(jQuery)
```

El script crea una capa `<div>` (la capa del tooltip), a la que añade la clase de estilo `tooltipcss`. Esta capa se añade al cuerpo (`body`) del documento. Al pasar el ratón por encima del enlace, se muestra esta capa. El contenido de ésta toma los elementos del atributo `rel` del enlace. Hay que prever una leve separación horizontal y vertical del tooltip con respecto al enlace (left y top).

Se van a configurar tres elementos:

- La clase que se encarga del aspecto del tooltip (`options.tooltipcss`).
- La separación vertical del tooltip con respecto al enlace (`options.offsetY`).
- La separación horizontal del tooltip con respecto al enlace (`options.offsetX`).

Con esto, nuestro plug-in está preparado para utilizarse.

Usar un plug-in jQuery

El uso de un plug-in es muy sencillo. Basta con llamarlo (por su nombre) en el código de la página.

Ejemplo

Vamos a usar nuestro plug-in en una página con varios enlaces.

El documento Html inicial es:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
</head>
<style>
.tooltip { width: 150px;
           padding: 3px;
           background-color: #9CF;
           border: black 1px solid;
           color: black;}
a { color: black}
</style>
</head>
<body>
<p><a href="#" rel="Explicación del enlace 1" class="info">Vínculo 1
</a></p>
<p><a href="#" rel="Explicación del enlace 2" class="info">Vínculo 2
</a></p>
<p><a href="#" rel="Explicación del enlace 3" class="info">Vínculo 3
</a></p>
</body>
</html>
```

Observaciones:

El atributo `rel` de las etiquetas del vínculo `<a>` ya contiene el texto del tooltip.

El aspecto del tooltip ya está previsto por la clase `.tooltip`.

Veamos el script:

```
<script>
$(document).ready(function(){
var options = {
offsetX: 30,
offsetY: 5,
tooltipcss: "tooltip"
};
$("a.info").mi_tooltip(options);
});
</script>
```

Algunas explicaciones:

```
var options = {
offsetX: 30,
offsetY: 5,
tooltipcss: "tooltip"
};
```

Determina los parámetros que se pasan al plug-in, es decir, la separación vertical, horizontal y el

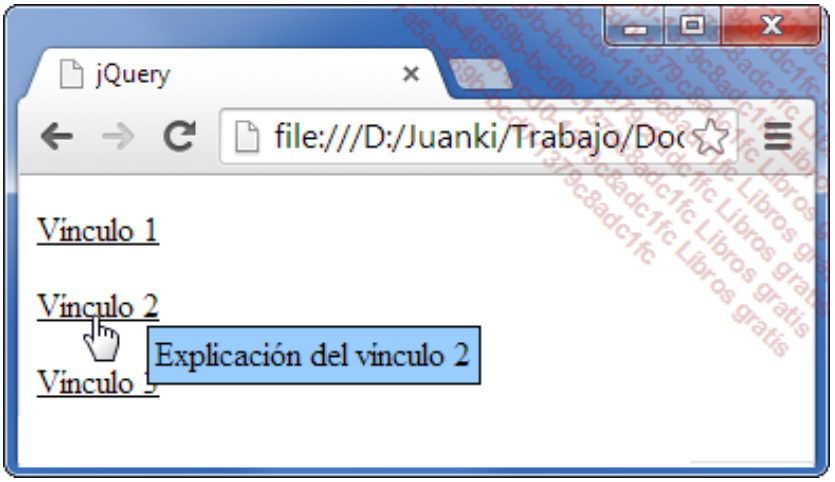
nombre de la clase de estilo de la ventana tooltip.

```
$("a.pop").mi_tooltip(options);
```

Para todos los enlaces que tengan la clase `pop`, se va a llamar al plug-in `mi_tooltip`.

El archivo final completo se convierte en:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="jquery.js"></script>
<script src="jquery.mi_tooltip.js"></script>
<script>
$(document).ready(function(){
var options = {
offsetX: 30,
offsetY: 5,
tooltipcss: "tooltip"
};
$("a.info").mi_tooltip(options);
});
</script>
<style>
.tooltip { width: 150px;
           padding: 3px;
           background-color: #9CF;
           border: black 1px solid;
           color: black;}
a { color: black;}
</style>
</head>
<body>
<p><a href="#" rel="Explicación del enlace 1" class="info">Vínculo
1</a></p>
<p><a href="#" rel="Explicación del enlace 2" class="info">Vínculo
2</a></p>
<p><a href="#" rel="Explicación del enlace 3" class="info">Vínculo
3</a></p>
</body>
</html>
```



Algunos plug-ins

Hemos visto algunos ejemplos de plug-ins. Para apreciar en su totalidad su lado más espectacular, le invitamos a consultar el espacio de descarga dedicado a este libro.

1. jQuery UI

a. Requisitos

jQuery UI (UI por *User Interface*) ofrece numerosos widgets para enriquecer la interfaz visual de sus páginas o aplicaciones HTML como los cuadros de diálogo, los tooltips, los menús acordeón, los menús con pestañas, los botones, arrastrar y soltar (*drag/drop*), el redimensionamiento y la reorganización de los elementos.

Además de que estos elementos se han convertido en clásicos, jQuery UI integra igualmente las nuevas etiquetas de Html5 con los cursores, los calendarios, las barras de progreso, los contadores numéricos y los formularios con sugerencias. Estos widgets permiten incorporar en el diseño de las páginas Html estas etiquetas Html5 innovadores que finalmente son operativas en todos los navegadores del mercado. ¡Incluso en los más antiguos!

Ante el número y la riqueza de estos módulos externos, jQuery UI se posiciona como un complemento indispensable de jQuery. Tanto si desea aplicaciones Web altamente interactivas, como si desea simplemente añadir un widget como un calendario, jQuery se convierte en una excelente solución.

Estos widgets son altamente configurables mediante el sistema de opciones jQuery UI y son perfectamente adaptables a la identidad gráfica de su sitio mediante la herramienta ThemeRoller.

b. Instalación

Paso 1

jQuery UI (http://jqueryui.com/) es una extensión del framework jQuery dedicada a la interfaz de sus aplicaciones. De este modo, jQuery UI necesita a jQuery para funcionar. Esta capa de jQuery soporta toda la gestión de funcionalidades JavaScript implementadas por los múltiples widgets introducidos por jQuery UI. Encontramos así una llamada (en este caso por CDN) a la librería jQuery.

Ejemplo

```
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
```

Paso 2

Diferentes CDN ofrecen acceso al API jQuery UI. Ésta es la de jQuery:

```
<script src="http://code.jquery.com/ui/número de versión/jquery-ui.js">
</script>
```

Ejemplo

```
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js">
</script>
```

Paso 3

Solo queda ocuparse de la presentación gráfica de los widgets. jQuery UI ofrece una veintena de temas que recogen todas las propiedades de estilo CSS que necesitan los widgets así como una combinación de color de fondo, de fuentes, de bordes redondeados, sombras, etc. Recordemos que estos temas se pueden personalizar mediante la herramienta ThemeRoller.

```
http://code.jquery.com/ui/número de versión de jQuery UI/themes/nombre
del tema/jquery-ui.css
```

Para nuestros ejemplos tomaremos el tema por defecto smoothness.

```
<link rel="stylesheet"
href=http://code.jquery.com/ui/1.10.4/themes/smoothness/jquery-ui.css">
```

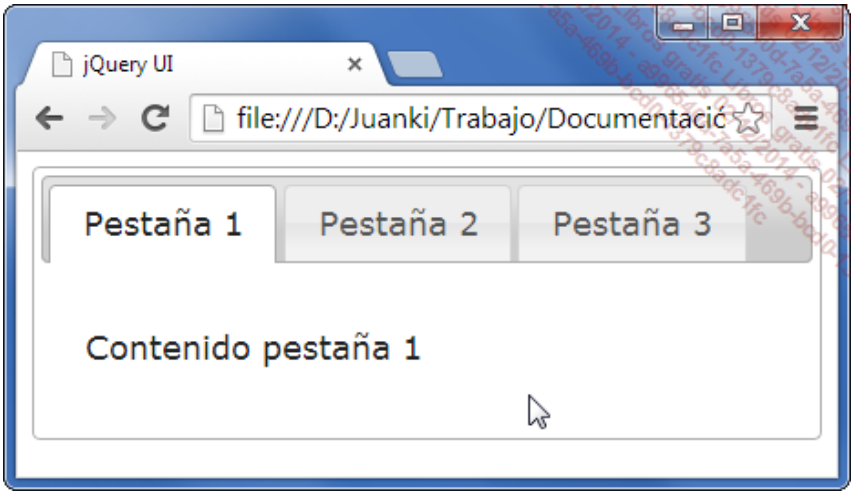
El código completo de instalación de jQuery UI (mediante CDN) es:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery UI</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.4/
themes/smoothness/jquery-ui.css">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js">
</script>
</head>
...
```

c. El menú con pestañas

El menú con pestañas es una opción que permite agrupar la información de un sitio por asunto o por tema. Es lo que se consigue a través del método jQuery UI `tabs()`.

Ejemplo



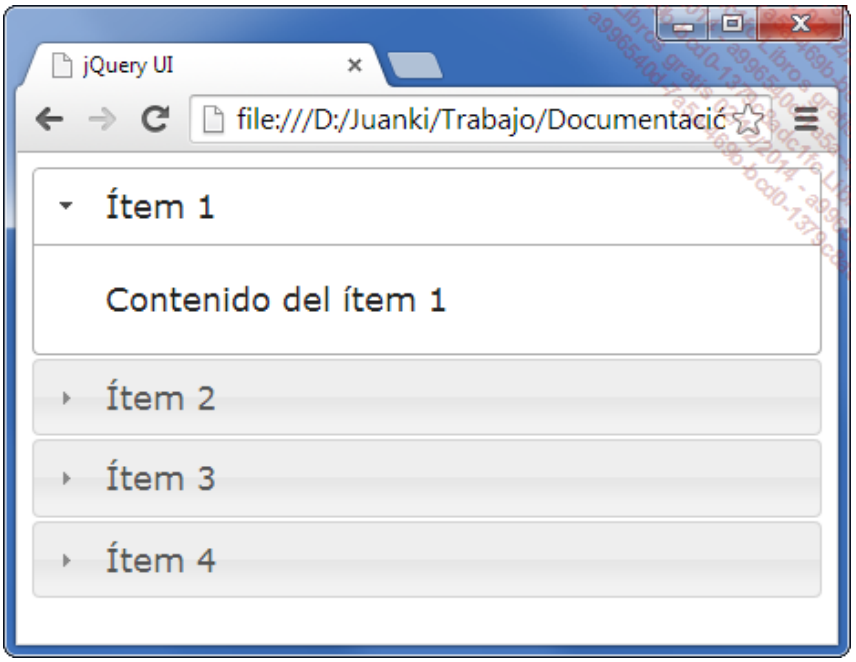

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery UI</title>
<link rel="stylesheet"
href="http://code.jquery.com/ui/1.10.4/themes/smoothness/jquery-
ui.css">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js">
</script>
<script>
$(function() {
$("#pestanas").tabs();
});
</script>
</head>
<body>
<div id="pestanas">
<ul>
<li><a href="#pestana-1">Pestaña 1</a></li>
<li><a href="#pestana-2">Pestaña 2</a></li>
<li><a href="#pestana-3">Pestaña 3</a></li>
</ul>
<div id="pestana-1">
<p>Contenido pestaña 1</p>
</div>
</div id="pestana-2">
<p>Contenido pestaña 2</p>
</div>
</div id="pestana-3">
<p>Contenido pestaña 3</p>
</div>
</div>
</body>
```

```
</html>
```

d. El menú en acordeón

La presentación llamada en acordeón es una alternativa interesante en la organización de la información en el diseño de páginas Web. Como recordatorio, este tipo de organización agrupa el contenido en paneles separados que el usuario puede abrir o cerrar. Lo que permite una ganancia de espacio apreciable maximizando la cantidad de información disponible. Los acordeones se implementan con la función jQuery UI `accordion()`.

Ejemplo

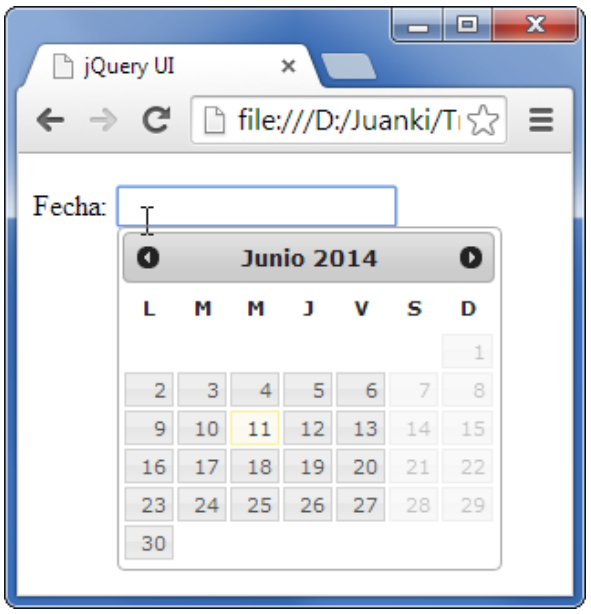



```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery UI</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.4/
themes/smoothness/jquery-ui.css">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js">
</script>
<script>
$(function() {
$("#acordeon").accordion();
});
</script>
</head>
<body>
<div id="acordeon">
<h3><a href="#">Ítem 1</a></h3>
<div>Contenido del ítem 1</div>
<h3><a href="#"> Ítem 2</a></h3>
<div> Contenido del ítem 2</div>
<h3><a href="#"> Ítem 3</a></h3>
<div> Contenido del ítem 3</div>
<h3><a href="#"> Ítem 4</a></h3>
<div> Contenido del ítem 4</div>
</div>
</body>
</html>
```

e. Los calendarios

Es útil mostrar los calendarios que tienen un importante papel en las aplicaciones de reserva de habitaciones de hotel, de coches de alquiler, de billetes de tren o de avión, etc. Utilizaremos el método jQuery UI `datepicker()`.

Ejemplo



Comentario

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery UI</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.4/
themes/smoothness/jquery-ui.css">
<style>
div.ui-datepicker { font-size: 12px;}
</style>
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js">
</script>
<script src="jquery.ui.datepicker-es.js"></script>
<script>
$(function() {
$("#calendario").datepicker({
firstDay: 1,
beforeShowDay: $.datepicker.noWeekends
});
});
</script>
</head>
<body>
<p><label for="calendario">Fecha:</label>
<input type="text" id="calendario"></p>
</body>
</html>
```

```
<script src="jqu
```

Por defecto, el calendario muestra los meses y los días en inglés. Para los hispanohablantes es necesario llamar a un script suplementario para la visualización en español.

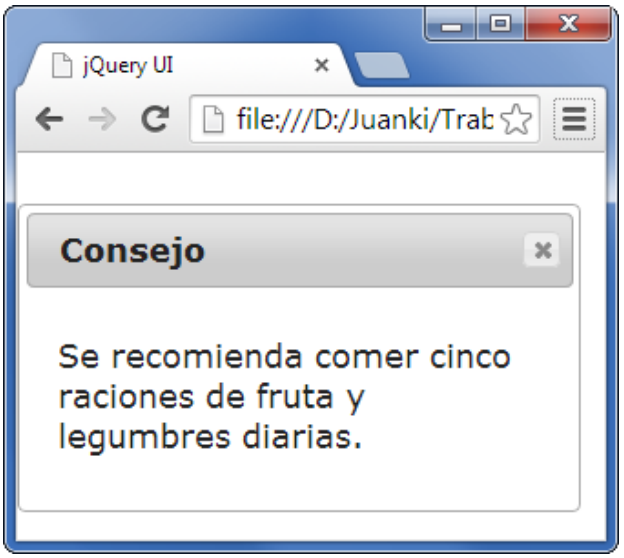
```
$("#calendario").datepicker({
firstDay: 1,
beforeShowDay: $.datepicker.noWeekends
});
});
```

Los widgets de jQuery UI tienen numerosas opciones de configuración. En este script `firstday: 1` permite que la semana comience en lunes y `beforeShowDay: $.datepicker.noWeekends` excluir de la selección los sábados y los domingos.

f. Los cuadros de diálogo

Los cuadros de diálogo del tipo mensaje de alerta forman parte de las páginas Web desde 2011. Es una manera práctica de advertir al usuario o de pedirle que confirme su elección. Aunque su aspecto se ha modificado últimamente, hay que admitir que tienen un estilo un tanto anticuado y por lo general son discordantes con el aspecto general de la aplicación. jQuery UI renueva completamente la presentación de estos cuadros de diálogo con bordes redondeados y permite personalizarlos tanto a nivel de sus funciones como a nivel gráfico. El método jQuery `UI dialog()` implementa estos cuadros de diálogo.

Ejemplo

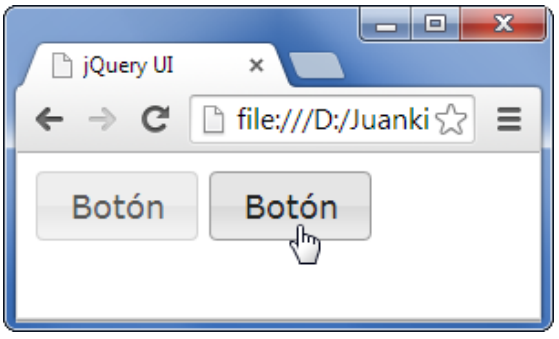



```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery UI</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.4/
themes/smoothness/jquery-ui.css">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js">
</script>
<script>
$(function() {
$("#dialogo").dialog();
});
</script>
</head>
<body>
<div id="dialogo" title="Consejo">
<p>Se recomienda comer cinco raciones de fruta y legumbres
diarias.</p>
</div>
</body>
</html>
```

g. Los botones

Los botones son quizás los elementos más fácilmente modificables por los CSS3 permitiendo decorarlos con bordes redondos, degradados y sombras. Pero no todos los navegadores que podemos encontrar en la web reconocen todavía los CSS3. Con jQuery UI, se dota a los botones de las mismas propiedades de estilo pero esta vez de manera totalmente compatible. Éstos se implementan mediante el método jQuery `UI button()`.

Ejemplo



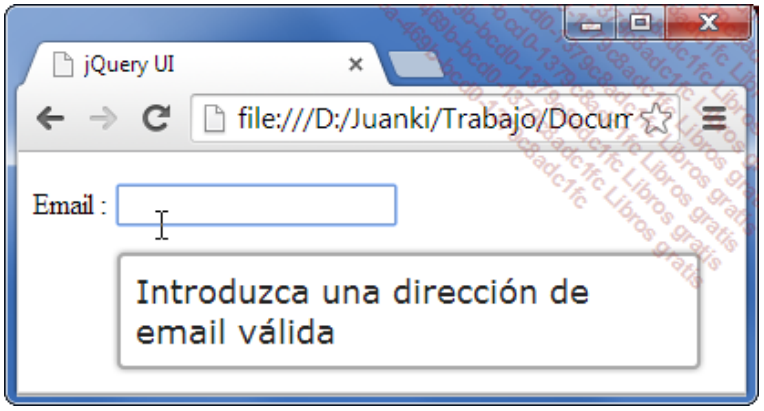

```
<!doctype html>
<html lang="es">
<head>
```

```
<meta charset="utf-8">
<title>jQuery UI</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.4/
themes/smoothness/jquery-ui.css">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js">
</script>
<script>
$(function() {
$("#boton1, #boton2").button();
});
</script>
</head>
<body>
<button id="boton1">Botón</button>
<div id="boton2">Botón</div>
</body>
</html>
```

h. Los tooltips

Un tooltip utilizado con prudencia puede aportar un complemento de información apreciable. Estos se implementan mediante el método jQuery UI `tooltip()`.

Ejemplo

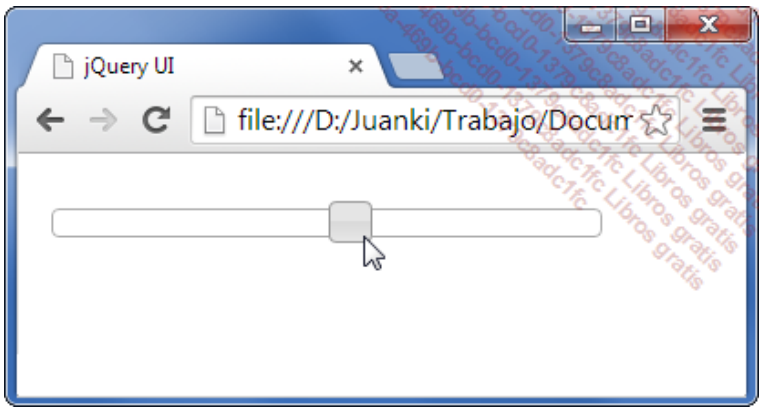



```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery UI</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.4/
themes/smoothness/jquery-ui.css">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js">
</script>
<script>
$(function() {
$(document).tooltip();
});
</script>
</head>
<body>
<p><label for="email">Email:</label>
<input id="email" title="Introduzca una dirección
de email válida "></p>
</body>
</html>
```

i. Los cursores

Los cursores, introducidos por Html5 (`<input type="range">`) permiten al usuario determinar un valor numérico modificando la posición de un tirador. Estos cursores innovadores e intuitivos todavía no están muy desplegados en nuestras páginas Web por miedo a no ser reconocidos por navegadores antiguos. Con jQuery UI, se pueden incorporar, sea cual sea el navegador del usuario. ¿Por qué no utilizarlos? Se implementan en jQuery mediante el método `slider()`.

Ejemplo

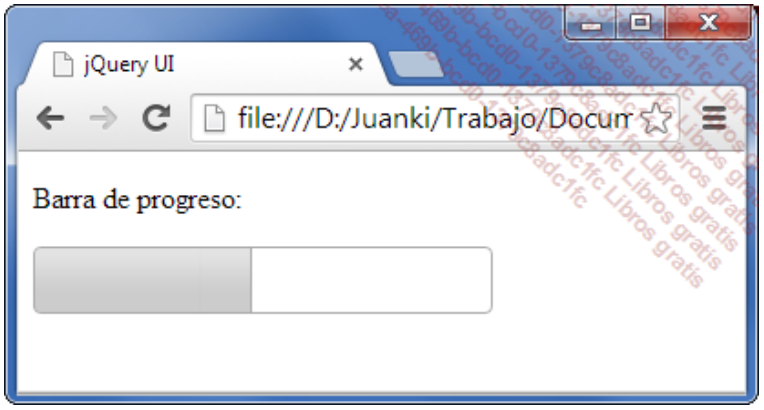



```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery UI</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.4/
themes/smoothness/jquery-ui.css">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js">
</script>
<script>
$(function() {
$("#cursor").slider();
});
</script>
<style>
#cursor { margin-top: 30px;
          margin-left: 10px;
          width: 300px;}
</style>
</head>
<body>
<div id="cursor"></div>
</body>
</html>
```

j. La barra de progreso

La barra de progreso es un elemento muy conocido que permite visualizar el avance de una tarea, como la transferencia de un fichero o la inicialización de un programa. Ofrece al usuario un excelente feedback visual. La barra de progreso se introdujo en Html5 mediante la etiqueta`<progress>` pero solo la reconocen las versiones más recientes de los navegadores. Para utilizarla hay que añadir el método jQuery UI `progressbar()`.

Ejemplo



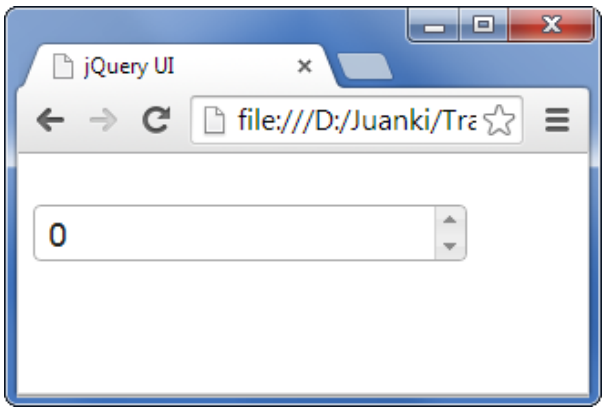

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery UI</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.4/
```

```
themes/smoothness/jquery-ui.css">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js">
<script>
$(function() {
$("#progreso").progressbar();
var value = 0;
var timer = setInterval (function (){
$("#progreso").progressbar("value", value);
value++;
if (value > 100) clearInterval(timer);
}, 100);
});
</script>
</script>
<style>
#progreso { width: 250px;}
</style>
</head>
<body>
<p>Barra de progreso: </p><div id="progreso"></div>
</body>
</html>
```

k. Los contadores numéricos

Los contadores numéricos resultan familiares en el mundo informático ya que los podemos encontrar en las aplicaciones ofimáticas y otras suites. También se han integrado en Html5 mediante la etiqueta `<input type="number">`. En jQuery UI se implementan con el método `spinner()`.

Ejemplo



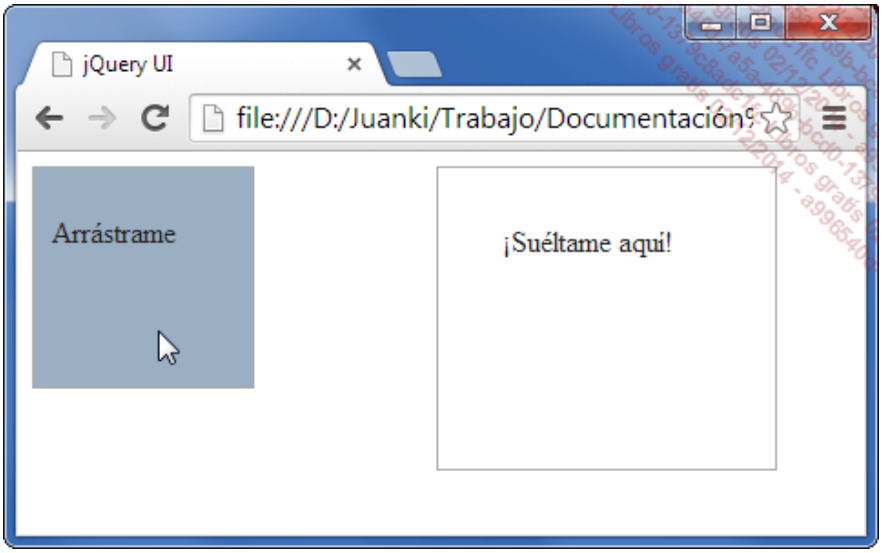

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery UI</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.4/
themes/smoothness/jquery-ui.css">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js">
</script>
<script>
$(function() {
$("#contador").spinner();
});
</script>
</head>
<body>
<br>
<input id="contador" value="0">
</body>
</html>
```

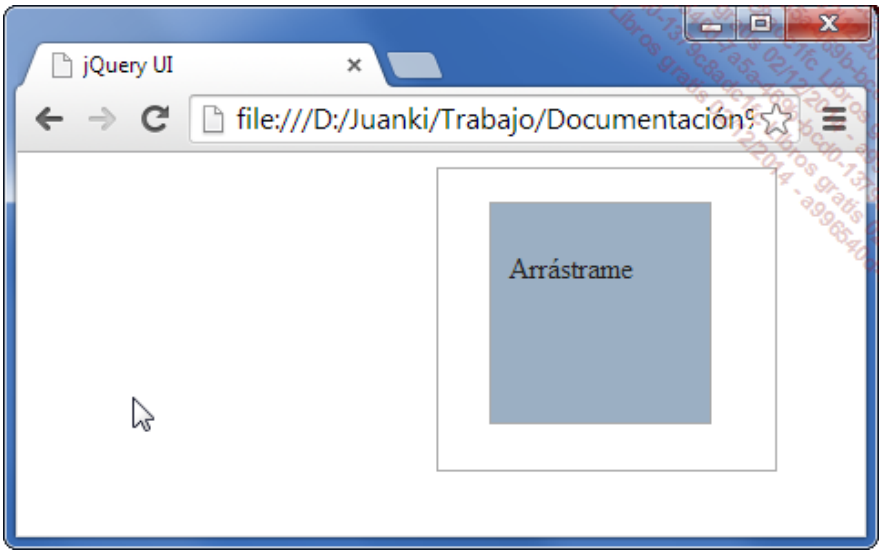
l. Arrastrar/soltar (drag/drop)

El Arrastrar/soltar (*drag/drop*) es una operación que se ha vuelto muy común en las páginas Web. Podemos citar como ejemplo las tiendas on-line donde se puede mover con el ratón la imagen del artículo elegido para transferirla a la

cesta de la compra. En jQuery el arrastrar/soltar se implementa mediante los métodos `draggable()` y `droppable()`.

Ejemplo






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery UI</title>
<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.4/
themes/smoothness/jquery-ui.css">
<script src="http://code.jquery.com/jquery-1.10.2.js"></script>
<script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js">
</script>
<style>
#box { width: 100px; height: 100px;
       padding: 10px;
       background: #9bafc3;
       float: left;}
#drop { width: 150px; height: 150px;
        margin-left: 100px;
        padding-top: 15px;
        padding-left: 35px;
        float: left;}
</style>
<script>
$(function() {
$("#box").draggable();
$("#drop").droppable();
});
```

Y más...

jQuery ofrece incluso elementos

```
</script>
</head>
<body>
<div id="box" class="ui-widget-content">
<p>Arrástrame</p>
</div>
<div id="drop" class="ui-widget-content">
<p>¡Suéltame aquí!</p>
</div>
</body>
</html>
```

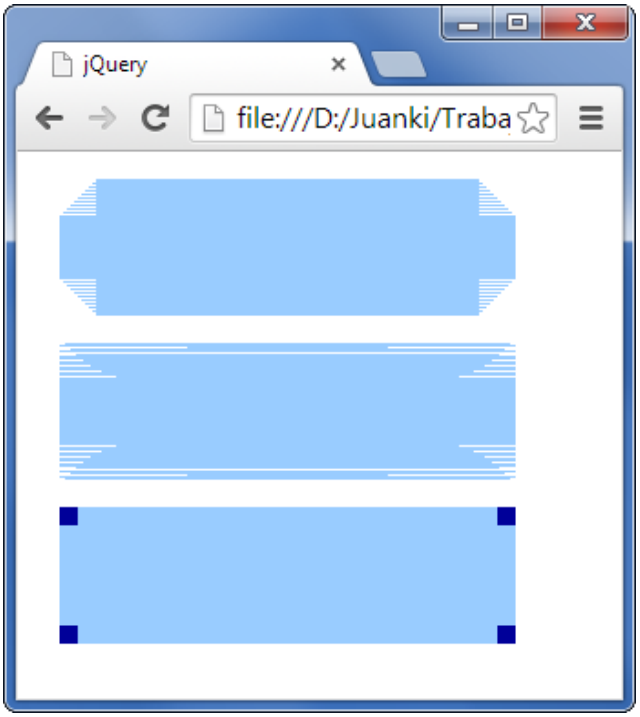
redimensionables, la reorganización de los elementos y el formulario con sugerencias.

2. Bordes variados

Los bordes redondeados son una aportación muy utilizada en los CSS3 y se anuncian nuevos bordes más variados para los futuros CSS4. Por qué no anticiparlos con el plug-in `jquery.corner`que ofrece bordes muy originales.

Se puede acceder al plug-in y la documentación en la página jQuery Corner Demo en la dirección:http://jquery.malsup.com/corner/

Ejemplo




```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8">
<title>jQuery</title>
<script src="http://code.jquery.com/jquery-2.1.0.min.js">
</script>
<script src="jquery.corner.js"></script>
<script>
$(document).ready(function(){
$("#div1").corner("dog 20px");
$("#div2").corner("wicked 20px");
$("#div3").corner("cc:#009 notch");
})
</script>
<style>
div.ejemplo { margin: 15px;
              width: 250px; height: 75px;
              background: #9cf;}
</style>
</head>
```

```
<body>
<div id="div1" class="ejemplo"></div>
<div id="div2" class="ejemplo"></div>
<div id="div3" class="ejemplo"></div>
</body>
</html>
```

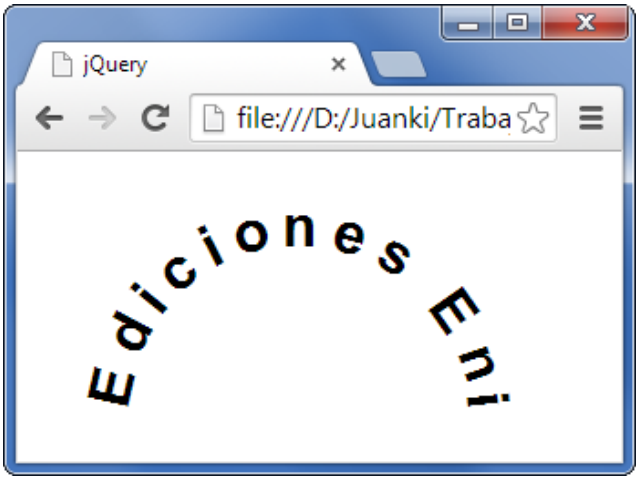
Hemos llamado mediante CDN a la versión 2.1 de la librería jQuery. Como recordará, esta versión no tiene en cuenta las versiones 6, 7 y 8 de Internet Explorer. Para un script compatible con estos últimos, utilice la versión 1.10 de jQuery.

3. Textos redondeados

Después del éxito de los bordes redondos, ¿por qué no utilizar también textos redondeados? Esto es lo que ofrece el plug-in `arctext`.

Una demostración, explicaciones y una descarga le esperan en la dirección:http://tympanus.net/codrops/2012/01/24/arctext-js-curving-text-with-css3-and-jquery/

Ejemplo

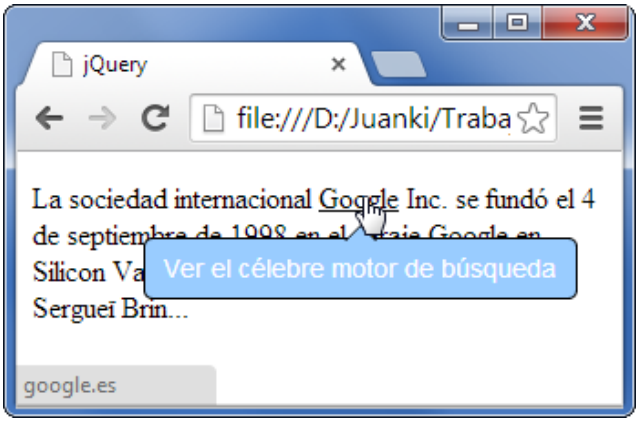



```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8">
<title>jQuery</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js">
</script>
<script src="jquery.arctext.js"></script>
<style>
h1 { margin-left: 40px;
     font-family: sans-serif;}
</style>
</head>
<body>
<h1 id="ejemplo">Ediciones Eni</h1>
<script>
$(document).ready(function(){
$('#ejemplo').arctext({radius: 100});
});
</script>
</body>
</html>
```

4. Tooltips

Un tooltip puede aportar un elemento de información apreciable para el usuario. Existen numerosos plug-ins disponibles en la red. Hemos elegido tooltipster en la url: http://calebjacob.com/tooltipster/

Ejemplo




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<link rel="stylesheet" type="text/css"
href="css/tooltipster.css">
<style>
.my-custom-theme { border-radius: 5px;
                   border: 1px solid #000;
                   background: #9cf;
                   color: #fff;}
.my-custom-theme .tooltipster-content { font-family: sans-serif;
                                        font-size: 14px;
                                        line-height: 16px;
                                        padding: 8px 10px;}
a { color: black;}
</style>
<script src="http://code.jquery.com/jquery-1.10.1.min.js">
</script>
<script type="text/javascript"
src="js/jquery.tooltipster.js"></script>
<script type="text/javascript">
$(document).ready(function() {
$(".tooltip").tooltipster({
theme: ".my-custom-theme"
});
});
</script>
</head>
<body>
<p>La sociedad internacional <a href="http://google.fr"
class="tooltip" title="Ver el célebre motor de
búsqueda">Google</a> Inc. se fundó el 4 de septiembre de 1998 en
el garaje Google en Silicon Valley, en California, por Larry Page
y Sergueï Brin...</p>
</body>
</html>
```

5. Desvelar las contraseñas

Cuando oculta una contraseña, los caracteres ocultos aparecen en forma de asteriscos. El hecho de ocultar así las contraseñas no aumenta la seguridad pero aumenta considerablemente las posibilidades de equivocación debido a errores al teclear. Muchos sitios ofrecen la posibilidad de desvelar su contraseña.

El plug-in `Show password` (http://unwrongest.com/projects/show-password) deja al usuario la elección de ocultar o no las contraseñas que teclee.

Ejemplo


```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<link href="screen.css" type="text/css" rel="stylesheet">
</head>
<body>
<div class="form">
<p>
<label>Login</label>
<span class="w">
<input id="password" type="password" data-typetoggle='#show-
password' class="input">
<label><input id="show-password" type="checkbox"> Revelar la
contraseña</label>
</span>
</p>
</div>
<script src="http://code.jquery.com/jquery-1.10.1.min.js">
</script>
<script src="jquery.showpassword.js"></script>
<script>
$(document).ready(function(){
$('#password').showPassword();
});
</script>
</body>
</html>
```

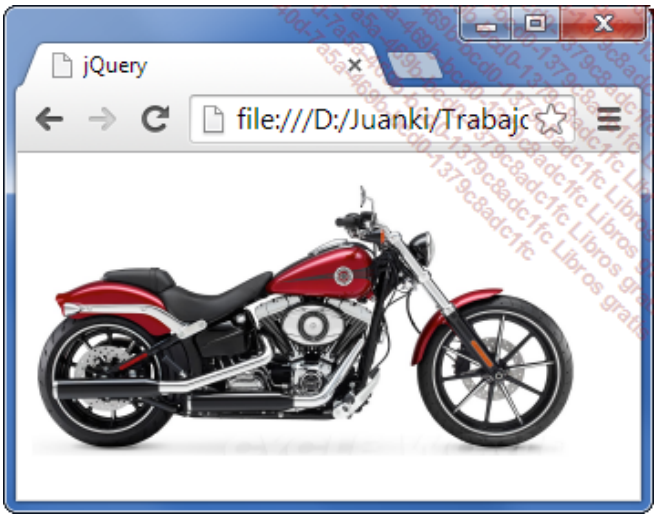
6. Hacer zoom sobre una imagen

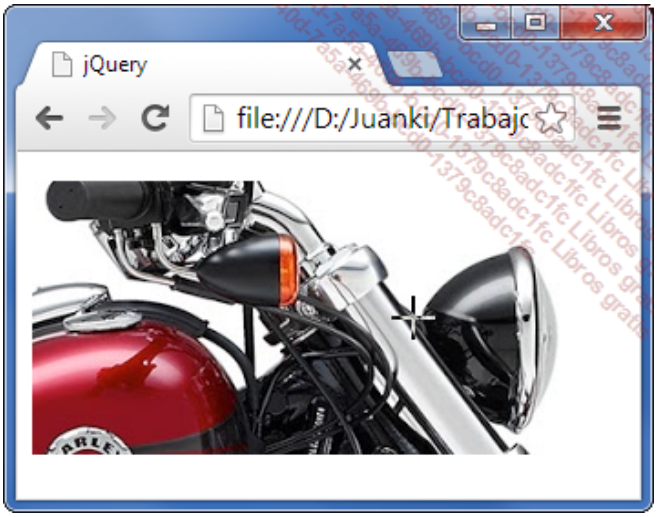
El plug-in jQuery `Image Zoom` permite hacer grande una parte de la imagen. Tiene numerosas opciones de configuración como el redimensionamiento de la lupa, el posicionamiento interno o externo de la imagen sobre la que hacemos el zoom, la utilización de la rueda del ratón para variar el zoom, etc.

La presentación y los primeros elementos de instalación los puede encontrar en la direcciónhttp://www.elevateweb.co.uk/image-zoom. También dispone de diferentes ejemplos en la páginahttp://www.elevateweb.co.uk/image-zoom/examples y la descarga del plug-in en la urlhttp://www.elevateweb.co.uk/image-zoom/download.

El truco consiste en utilizar dos imágenes. Una en baja resolución para la imagen visible y la otra en alta resolución para la imagen grande.

Ejemplo





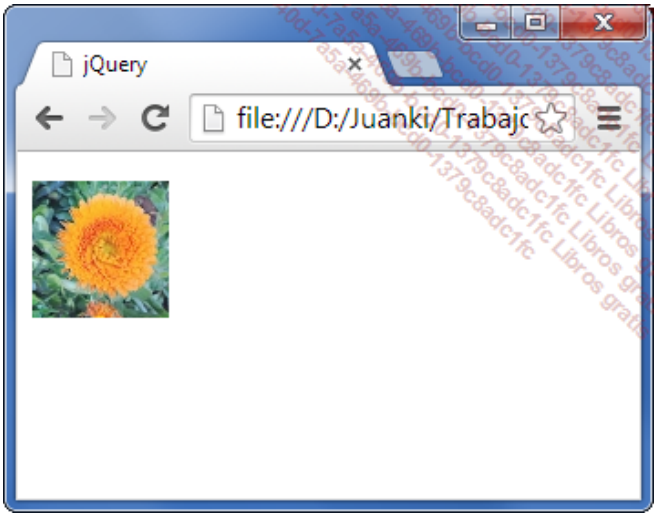

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<script src="http://code.jquery.com/jquery-1.10.1.min.js">
</script>
<script src='jquery.elevatezoom.js'></script>
</head>
<body>
<p>
<img id="zoom" src='images/small/hds.png' data-zoom-
image="images/large/hdl.jpg">
</p>
<script>
$(document).ready(function(){
$("#zoom").elevateZoom({
zoomType: "inner",
cursor: "crosshair"
});
});
</script>
</body>
</html>
```

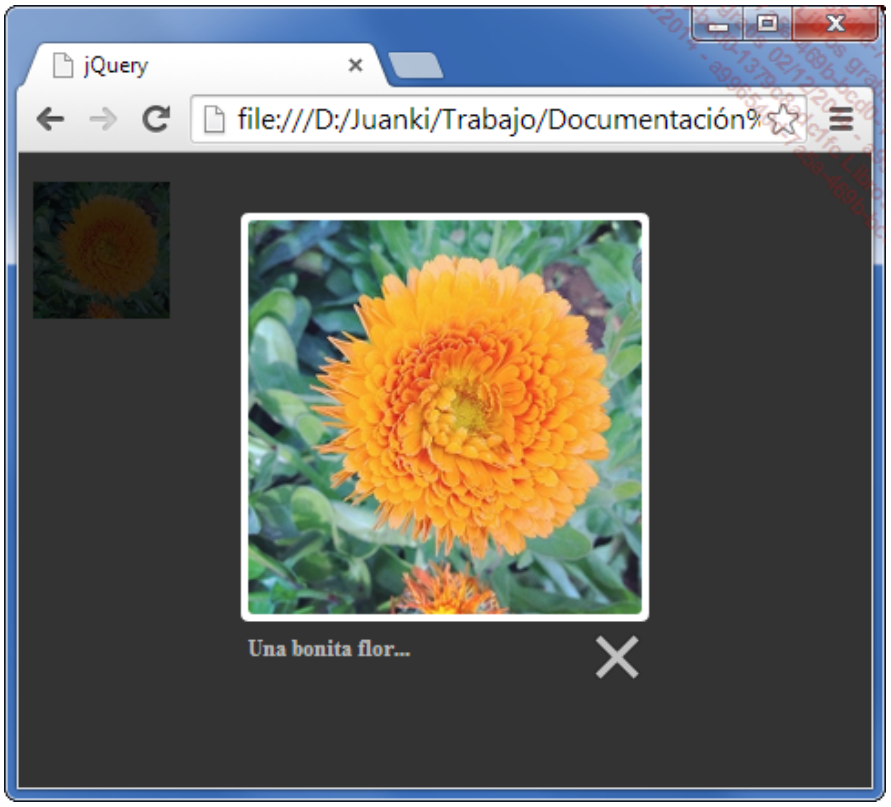
7. Ampliar una miniatura

Inicialmente introducido por Google Image, el efecto lightbox que permite ampliar una imagen con un fondo de color se ha convertido en un clásico de las aplicaciones Web.

Entre los numerosos plug-ins disponibles, hemos elegido `Lightbox2`. En la página http://lokeshdhakar.com/projects/lightbox2/ puede encontrar ejemplos, documentación detallada y el enlace de descarga.

Ejemplo






```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<link rel="stylesheet" href="css/lightbox.css">
<script src="http://code.jquery.com/jquery-1.10.1.min.js">
</script>
<script src="js/lightbox-2.6.min.js"></script>
</head>
<body>
<p><a href="img/images/flor.jpg" data-lightbox="flor"
```

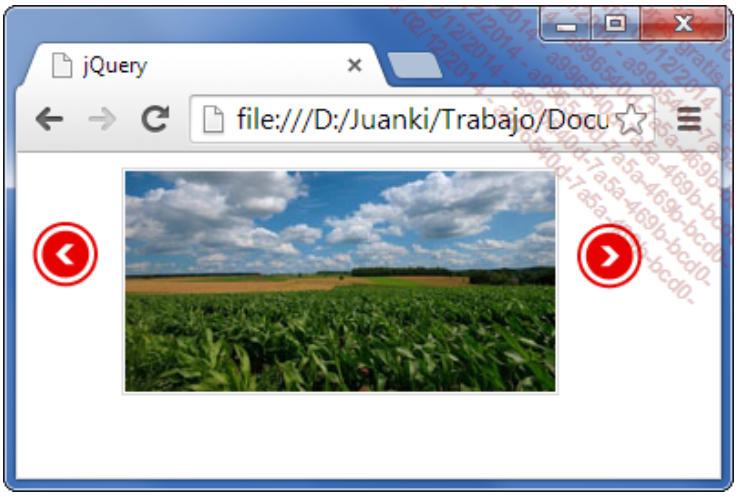
```
title="Una bonita flor...">
<img class="example-image" src="img/images/thumb-flor.jpg"
alt=""></a>
</p>
</body>
</html>
```

8. Un carrusel de imágenes

En la web hay numerosos scripts o plug-ins de carrusel de imágenes. Hemos escogido `Tiny Carousel` por su simplicidad en la implementación.

En la página http://baijs.nl/tinycarousel/ puede encontrar ejemplos, opciones de configuración y el enlace de descarga.

Ejemplo



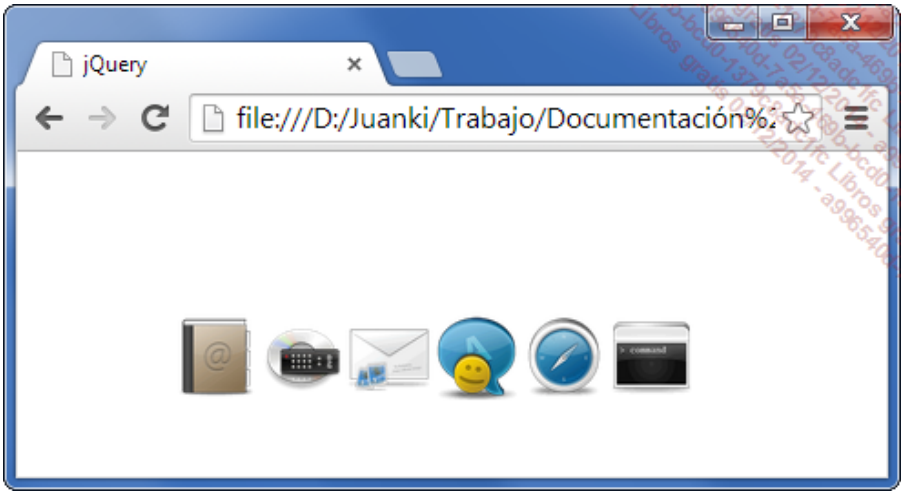

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<link rel="stylesheet" href="css/website.css">
<script src="http://code.jquery.com/jquery-2.1.0.min.js">
</script>
<script src="js/jquery.tinycarousel.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
$("#slider1").tinycarousel();
});
</script>
</head>
<body>
<div id="slider1">
<a class="buttons prev" href="#">left</a>
<div class="viewport">
<ul class="overview">
<li><img src="images/picture6.jpg"></li>
<li><img src="images/picture5.jpg"></li>
<li><img src="images/picture4.jpg"></li>
<li><img src="images/picture3.jpg"></li>
<li><img src="images/picture2.jpg"></li>
<li><img src="images/picture1.jpg"></li>
</ul>
</div>
<a class="buttons next" href="#">right</a>
</div>
</body>
</html>
```

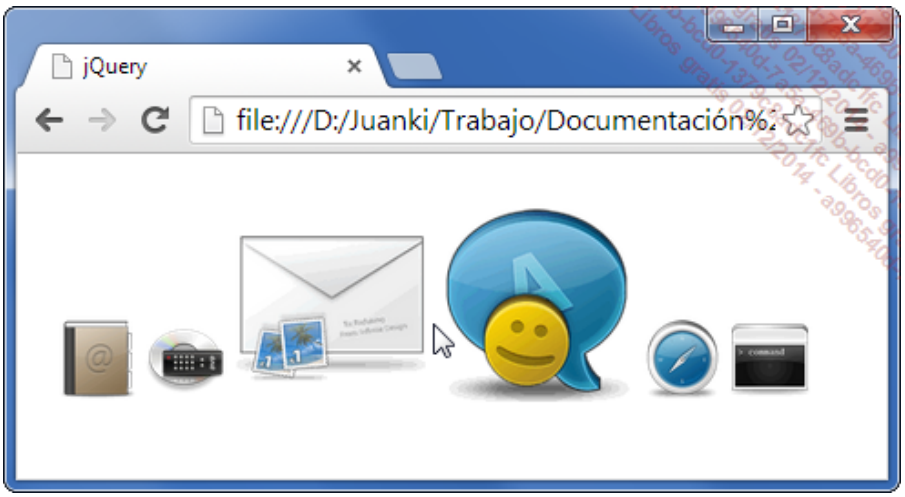
Hemos hecho la llamada por CDN a la versión 2.1 de la librería jQuery. Recuerde que esta versión no soporta las versiones 6, 7 y 8 de Internet Explorer. Para un script compatible con estas últimas, utilice la versión 1.10 de jQuery.

9. Un menú de tipo Mac

El plug-in `jqdock` (http://www.wizzud.com/jqDock/) transforma una serie de imágenes en un menú que recuerda a la funcionalidad Dock del entorno gráfico del Mac OS X con sus pequeños iconos que aumentan de tamaño al pasar por encima el ratón.

Ejemplo






```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<link href="style.css" rel="stylesheet">
<script src="http://code.jquery.com/jquery-1.10.1.min.js">
</script>
<script src="jquery.jqdock.min.js"></script>
<script>
jQuery(document).ready(function($){
var jqdockOptions = {align: "bottom"};
$('#menu').jqDock(jqdockOptions);
});
</script>
</head>
<body>
<div id='page'>
<div id='menu'>
<img src='images/contacts.png' alt="">
<img src='images/player.png' alt="">
<img src='images/email.png' alt="">
<img src='images/messaging.png' alt="">
<img src='images/safari.png' alt="">
<img src='images/cmd.png' alt="">
</div>
</div>
```

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

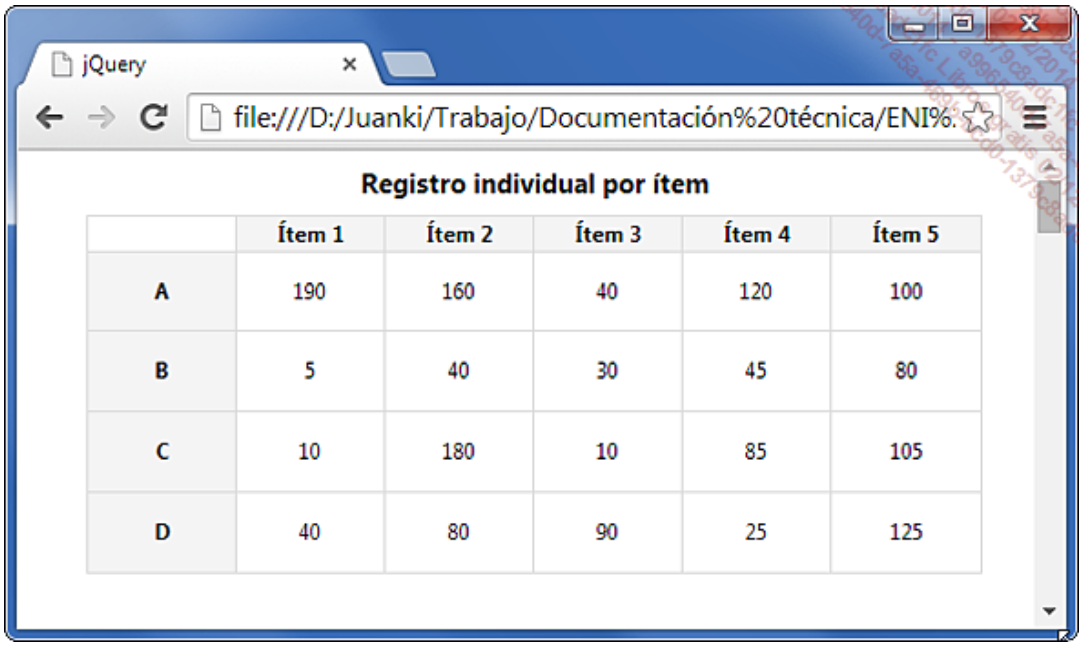
10. Gráficos a partir de una tabla

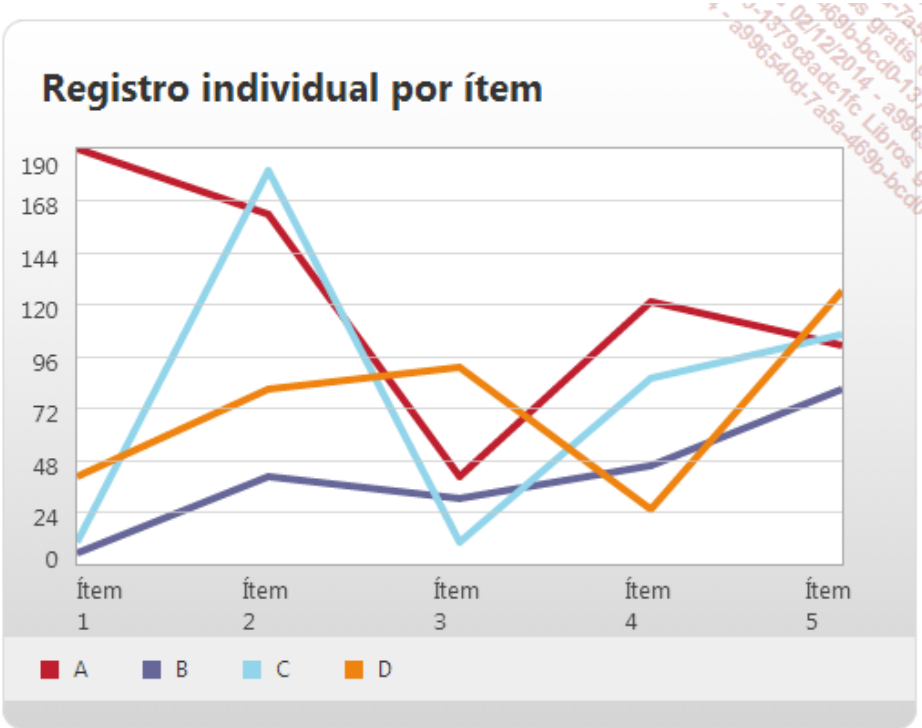
Sin duda, un gráfico es más expresivo que una tabla de datos. Para mostrar un gráfico en las páginas Html, solo existía hasta ahora la alternativa de hacer una captura de pantalla de la tabla en una hoja de cálculo de tipo Excel.

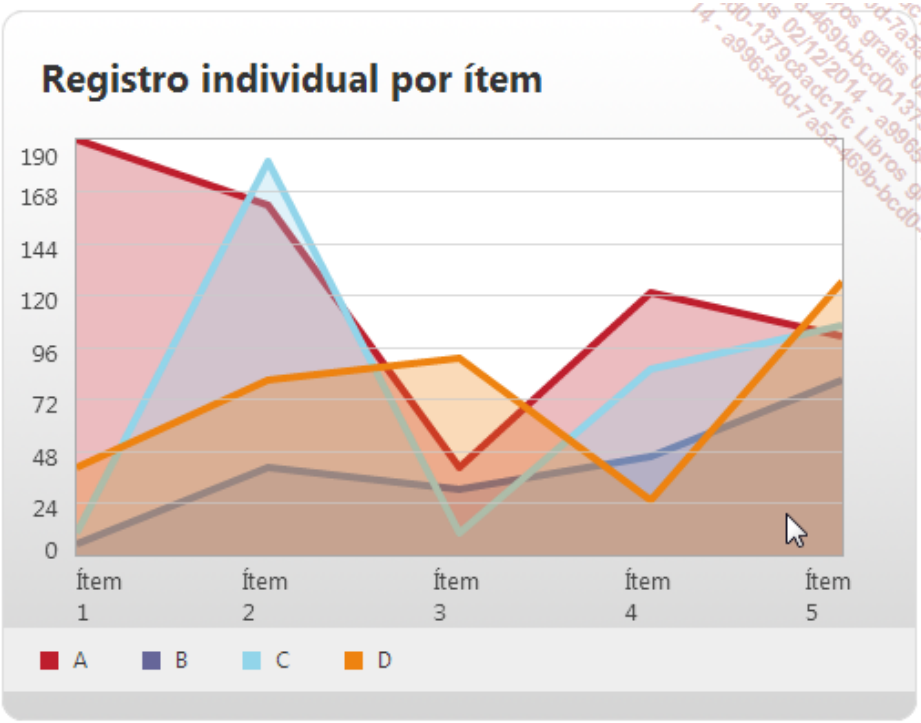
El plug-in `visualize` permite crear un gráfico sobre la marcha a partir de los datos de una tabla y mostrarlo directamente en la página. Este plug-in es espectacular y particularmente útil cuando la tabla de datos se actualiza con mucha frecuencia.

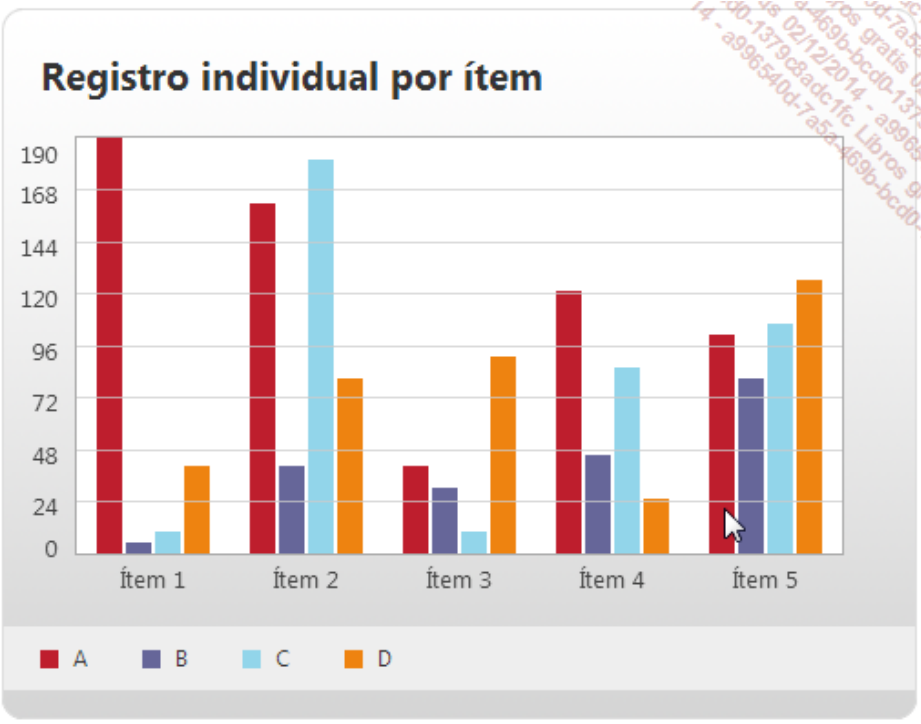
Puede encontrar una documentación muy detallada y el plug-in en la url:http://www.filamentgroup.com/lab/update_to_jquery_visualize_accessible_charts_with_html5_from_designing_with/

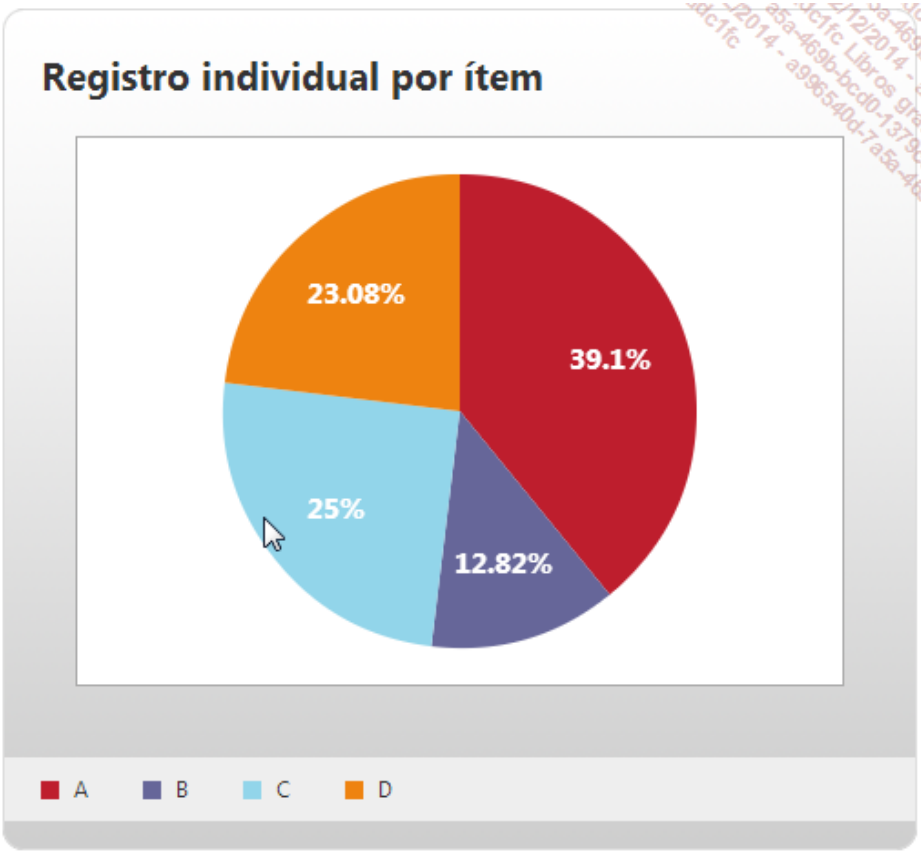
Ejemplo






Registro individual por ítem
190
168
144
120
96
72
48
24
0
Ítem 1
Ítem 2
Ítem 3
Ítem 4
Ítem 5
A
B
C
D


Registro individual por ítem
190
168
144
120
96
72
48
24
0
Ítem 1
Ítem 2
Ítem 3
Ítem 4
Ítem 5
A
B
C
D




```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<title>jQuery</title>
<link href="css/basic.css" rel="stylesheet">
<script
src="http://filamentgroup.github.com/EnhanceJS/enhance.js">
</script>
<script>
enhance({
loadScripts: [
{src: 'js/excanvas.js', iecondition: 'all'},
'js/jquery.js',
'js/visualize.jQuery.js',
'js/example.js'
],
loadStyles: [
'css/visualize.css',
'css/visualize-light.css'
]
});
</script>
</head>
<body>
<table >
<caption>Registro individual por ítem</caption>
<thead>
<tr>
<td></td>
<th>Ítem 1</th>
<th>Ítem 2</th>
<th>Ítem 3</th>
<th>Ítem 4</th>
<th>Ítem 5</th>
</tr>
</thead>
<tbody>
<tr>
<th>A</th>
<td>190</td>
<td>160</td>
```

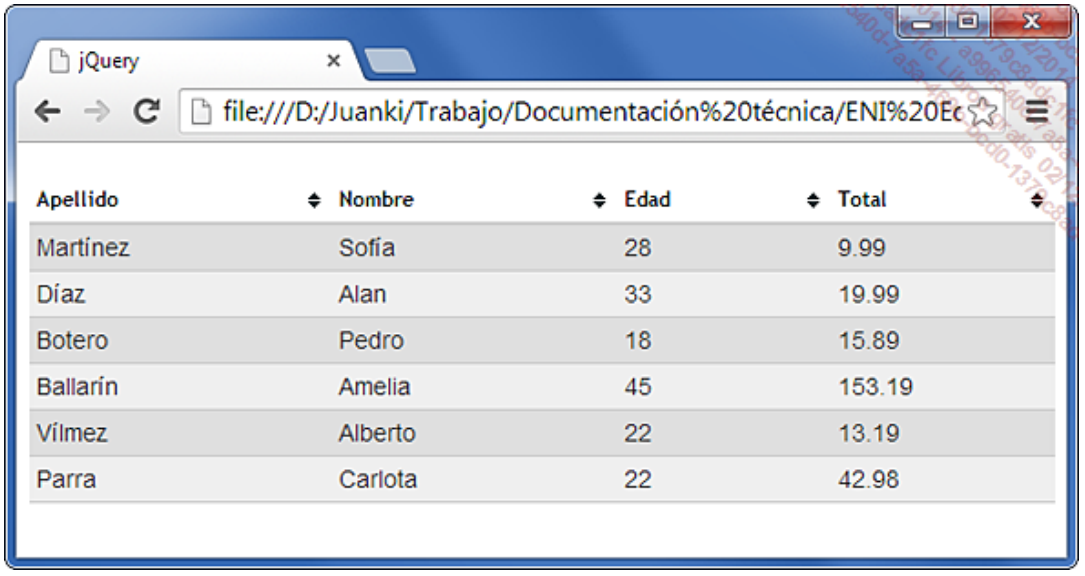
```
<td>40</td>
<td>120</td>
<td>100</td>
</tr>
<tr>
<th>B</th>
<td>5</td>
<td>40</td>
<td>30</td>
<td>45</td>
<td>80</td>
</tr>
<tr>
<th>C</th>
<td>10</td>
<td>180</td>
<td>10</td>
<td>85</td>
<td>105</td>
</tr>
<tr>
<th>D</th>
<td>40</td>
<td>80</td>
<td>90</td>
<td>25</td>
<td>125</td>
</tr>
</tbody>
</table>
</body>
</html>
```

11. Ordenar una tabla de datos

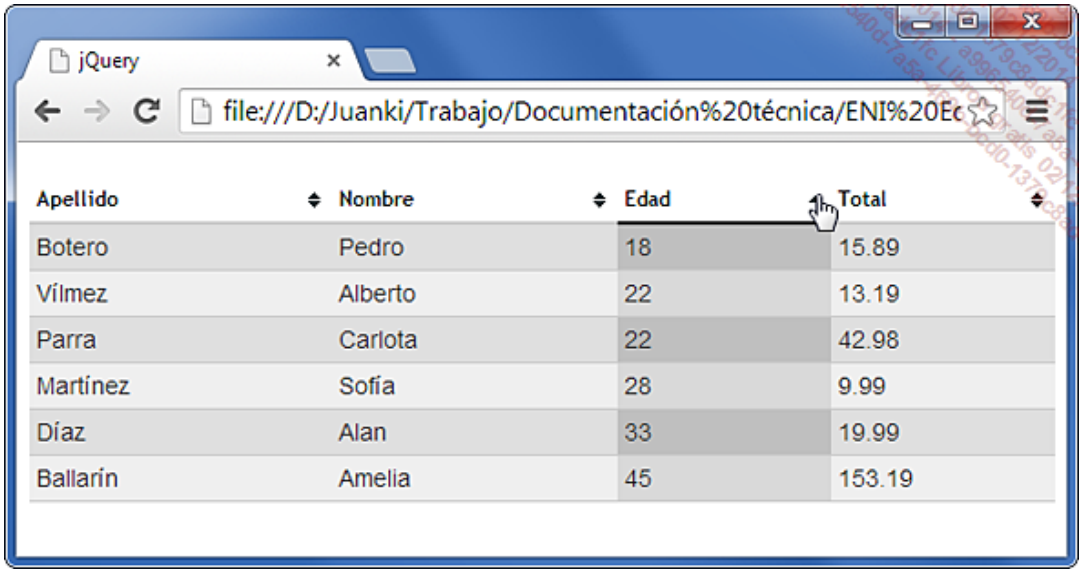
Otra maravilla de la programación, el plug-in tablesorter permite ordenar los datos de cualquier columna de una tabla, en orden ascendente o descendente. El script detecta automáticamente el tipo de información que contiene la columna. Reconoce, entre otros, números, fechas, direcciones IP, etc.

La documentación (muy completa) y el plug-in están disponibles en la dirección:http://tablesorter.com/docs/

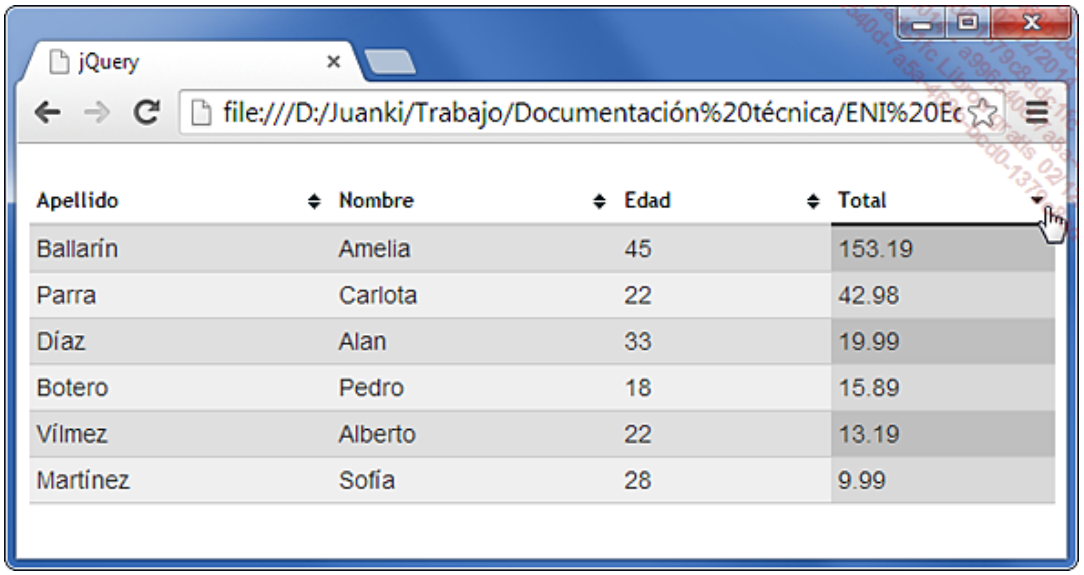
Ejemplo



Es posible ordenar según la edad:



u ordenar descendientemente según el total.




```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="UTF-8">
<title>jQuery</title>
<link href="css/jq.css" rel="stylesheet">
<link href="css/theme.default.css" rel="stylesheet">
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script src="js/jquery.tablesorter.min.js"></script>
<script src="js/jquery.tablesorter.widgets.min.js"></script>
<script>
$(function(){
$('table').tablesorter({
widgets : ['zebra', 'columns'],
usNumberFormat : false,
sortReset      : true,
sortRestart    : true
});
});
</script>
</head>
<body>
<div class="demo">
<p>
<table class="tablesorter">
<thead>
<tr>
<th>Apellido</th>
```

```
<th>Nombre</th>
<th>Edad</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Martínez</td>
<td>Sofía</td>
<td>28</td>
<td>9.99</td>
</tr>
<tr>
<td>Díaz</td>
<td>Alan</td>
<td>33</td>
<td>19.99</td>
</tr>
<tr>
<td>Botero</td>
<td>Pedro</td>
<td>18</td>
<td>15.89</td>
</tr>
<tr>
<td>Ballarín</td>
<td>Amelia</td>
<td>45</td>
<td>153.19</td>
</tr>
<tr>
<td>Vílmez</td>
<td>Alberto</td>
<td>22</td>
<td>13.19</td>
</tr>
<tr>
<td>Parra</td>
<td>Carlota</td>
<td>22</td>
<td>42.98</td>
</tr>
</tbody>
</table>
</p>
</div>
</body>
</html>
```