

Desarrollo de aplicaciones web

Julián Villegas Alonso

Investigación

Ajax con JQuery

Marrufo Castillo Javier Efrén

Playa del Carmen Q.Roo 9 de julio de 2018

Universidad Tecnológica de la Riviera Maya

**Métodos Ajax de jQuery**

Como se indicó anteriormente, jQuery posee varios métodos para trabajar con Ajax. Sin embargo, todos están basados en el método $.ajax, por lo tanto, su comprensión es obligatoria. A continuación, se abarcará dicho método y luego se indicará un breve resumen sobre los demás métodos.

Generalmente, es preferible utilizar el método $.ajax en lugar de los otros, ya que ofrece más características y su configuración es muy comprensible.

**$.ajax**

El método $.ajax es configurado a través de un objeto, el cual contiene todas las instrucciones que necesita jQuery para completar la petición. Dicho método es particularmente útil debido a que ofrece la posibilidad de especificar acciones en caso que la petición haya fallado o no. Además, al estar configurado a través de un objeto, es posible definir sus propiedades de forma separada, haciendo que sea más fácil la reutilización del código. Puede visitar [api.jquery.com/jQuery.ajax](http://api.jquery.com/jQuery.ajax/) para consultar la documentación sobre las opciones disponibles en el método.

**Utilizar el método $.ajax**

$.ajax({

// la URL para la petición

url : 'post.php',

// la información a enviar

// (también es posible utilizar una cadena de datos)

data : { id : 123 },

// especifica si será una petición POST o GET

type : 'GET',

// el tipo de información que se espera de respuesta

dataType : 'json',

// código a ejecutar si la petición es satisfactoria;

// la respuesta es pasada como argumento a la función

success : function(json) {

$('<h1/>').text(json.title).appendTo('body');

$('<div class="content"/>')

.html(json.html).appendTo('body');

},

// código a ejecutar si la petición falla;

// son pasados como argumentos a la función

// el objeto de la petición en crudo y código de estatus de la petición

error : function(xhr, status) {

alert('Disculpe, existió un problema');

},

// código a ejecutar sin importar si la petición falló o no

complete : function(xhr, status) {

alert('Petición realizada');

}

});

**NOTA** Una aclaración sobre el parámetro dataType: Si el servidor devuelve información que es diferente al formato especificado, el código fallará, y la razón de porque lo hace no siempre quedará clara debido a que la respuesta HTTP no mostrará ningún tipo de error. Cuando esté trabajando con peticiones Ajax, debe estar seguro que el servidor está enviando el tipo de información que está solicitando y verifique que la cabecera Content-type es exacta al tipo de dato. Por ejemplo, para información en formato JSON, la cabecera Content-type debería ser application/json.

**Opciones del método $.ajax**

El método $.ajax posee muchas opciones de configuración, y es justamente esta característica la que hace que sea un método muy útil. Para una lista completa de las opciones disponibles, puede consultar [api.jquery.com/jQuery.ajax](http://api.jquery.com/jQuery.ajax/); a continuación, se muestran las más comunes:

async Establece si la petición será asíncrona o no. De forma predeterminada el valor es true. Debe tener en cuenta que si la opción se establece en false, la petición bloqueará la ejecución de otros códigos hasta que dicha petición haya finalizado.

cache Establece si la petición será guardada en la cache del navegador. De forma predeterminada es true para todos los dataType excepto para script y jsonp. Cuando posee el valor false, se agrega una cadena de caracteres anti-cache al final de la URL de la petición.

complete Establece una función de devolución de llamada que se ejecuta cuando la petición esta completa, aunque haya fallado o no. La función recibe como argumentos el objeto de la petición en crudo y el código de estatus de la misma petición.

context Establece el alcance en que la/las funciones de devolución de llamada se ejecutaran (por ejemplo, define el significado de this dentro de las funciones). De manera predeterminada this hace referencia al objeto originalmente pasado al método $.ajax.

data Establece la información que se enviará al servidor. Esta puede ser tanto un objeto como una cadena de datos (por ejemplo foo=bar&baz=bim)

dataType Establece el tipo de información que se espera recibir como respuesta del servidor. Si no se especifica ningún valor, de forma predeterminada, jQuery revisa el tipo de MIME que posee la respuesta.

error Establece una función de devolución de llamada a ejecutar si resulta algún error en la petición. Dicha función recibe como argumentos el objeto de la petición en crudo y el código de estatus de la misma petición.

jsonp Establece el nombre de la función de devolución de llamada a enviar cuando se realiza una petición JSONP. De forma predeterminada el nombre es *callback*

success Establece una función a ejecutar si la petición a sido satisfactoria. Dicha función recibe como argumentos la información de la petición (convertida a objeto JavaScript en el caso que dataType sea JSON), el estatus de la misma y el objeto de la petición en crudo.

timeout Establece un tiempo en milisegundos para considerar a una petición como fallada. traditional Si su valor es true, se utiliza el estilo de serialización de datos utilizado antes de jQuery 1.4. Para más detalles puede visitar [api.jquery.com/jQuery.param](http://api.jquery.com/jQuery.param/).

type De forma predeterminada su valor es GET. Otros tipos de peticiones también pueden ser utilizadas (como PUT y DELETE), sin embargo pueden no estar soportados por todos los navegadores.

url Establece la URL en donde se realiza la petición. La opción url es obligatoria para el método $.ajax;

**Métodos convenientes**

En caso que no quiera utilizar el método $.ajax, y no necesite los controladores de errores, existen otros métodos más convenientes para realizar peticiones Ajax (aunque, como se indicó antes, estos están basados el método $.ajax con valores pre-establecidos de configuración).

Los métodos que provee la biblioteca son:

* $.get Realiza una petición GET a una URL provista.
* $.post Realiza una petición POST a una URL provista.
* $.getScript Añade un script a la página.
* $.getJSON Realiza una petición GET a una URL provista y espera que un dato JSON sea devuelto.

Los métodos deben tener los siguientes argumentos, en orden:

* url La URL en donde se realizará la petición. Su valor es obligatorio.
* data La información que se enviará al servidor. Su valor es opcional y puede ser tanto un objeto como una cadena de datos (como foo=bar&baz=bim).Nota: esta opción no es valida para el método $.getScript.
* success callback Una función opcional que se ejecuta en caso que petición haya sido satisfactoria. Dicha función recibe como argumentos la información de la petición y el objeto en bruto de dicha petición.
* data type El tipo de dato que se espera recibir desde el servidor. Su valor es opcional. Nota: esta opción es solo aplicable para métodos en que no está especificado el tipo de dato en el nombre del mismo método.

**Utilizar métodos convenientes para peticiones Ajax**

// obtiene texto plano o html

$.get('/users.php', { userId : 1234 }, function(resp) {

console.log(resp);

});

// añade un script a la página y luego ejecuta la función especificada

$.getScript('/static/js/myScript.js', function() {

functionFromMyScript();

});

// obtiene información en formato JSON desde el servidor

$.getJSON('/details.php', function(resp) {

$.each(resp, function(k, v) {

console.log(k + ' : ' + v);

}); });

**$.fn.load**

El método $.fn.load es el único que se puede llamar desde una selección. Dicho método obtiene el código HTML de una URL y rellena a los elementos seleccionados con la información obtenida. En conjunto con la URL, es posible especificar opcionalmente un selector, el cual obtendrá el código especificado en dicha selección.

**Utilizar el método $.fn.load para rellenar un elemento**

$('#newContent').load('/foo.html');

**Utilizar el método $.fn.load para rellenar un elemento basado en un selector**

$('#newContent').load('/foo.html #myDiv h1:first', function(html) {

alert('Contenido actualizado');

});