Ajax con jQuery (y III)

En este artículo vamos a introducir el método más potente y flexible que jQuery nos ofrece (al menos hasta hoy) para trabajar con Ajax. Se llama, como no podía ser de otro modo, $.ajax() y veremos que posee un gran número de posibilidades de configuración que lo hacen apto para cubrir cualquier necesidad que podamos tener en este sentido.

EL MÉTODO $.ajax()

El método $.ajax() es, sin duda, el más potente de todos los que hemos visto, por lo que es imprescindible conocerlo y saber manejarlo. Tiene muchas opciones de configuración, por lo que aquí vamos a empezar por la forma más simple de usarlo, y luego iremos ampliando conceptos.

Una cosa que nos llama la atención la primera vez que vemos este método es el hecho de que todos los parámetros que recibe se pasan como miembros de un objeto de JavaScript. En los métodos anteriores reservábamos este sistema sólo para los datos que le pasábamos al script al que llamábamos por Ajax, pero el resto de los parámetros no iban en notación de objeto.

Vamos a dejarnos de rollos teóricos, y a ver un ejemplo real. Mira el código index_ajax_1.php:

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8">
  <title>Pruebas Ajax</title>
 </head>
 <body>
  Nombre:
  <input type="text" value="" id="campo_nombre">
  <br>
  Email:
  <input type="email" value="" id="campo_email">
  <br>
  <button id="envio">Enviar</button>
  <div id="capaDeEstado"></div>
  <div id="capaDeDatos"></div>
  <div id="capaDeStatusDeResultado"></div>
  <script language="javascript" src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
  <script language="javascript">
   $(function(){
    $('#envio').on('click', function(){
     $('#capaDeEstado').html('Cargando datos...');
     $.ajax({
      url:"leer_datos.php",
      data:{
       'dato_nombre':$('#campo_nombre').prop('value'),
```

```
      'dato_email':$('#campo_email').prop('value')
     },
     type:'GET',
     dataType:'html',
     success:function(resultado, estado) {
      $('#capaDeEstado').html('DATOS CARGADOS');
      $('#capaDeDatos').html(resultado);
      $('#capaDeStatusDeResultado').html(estado);
     }
    });
   });
  });
 </script>
</body>
</html>
```

El método $.ajax() está definido en las líneas resaltadas del listado. Como puedes ver, todos los datos que vamos a pasar se incluyen como miembros de un objeto, tal cómo hemos comentado anteriormente. Vamos a irlos analizando uno a uno.

EL MIEMBRO url

Este es clarísimo. Es el nombre (y, en su caso, la ruta) del script al que vamos a llamar por Ajax.

EL MIEMBRO data

Este es, a su vez, un objeto cuyos miembros son los datos que vamos a pasar al script que hemos mencionado en el miembro anterior.

EL MIEMBRO type

Establece si los datos se pasan al script llamado por GET o por POST.

EL MIEMBRO dataType

Este es nuevo. Es muy importante establecerlo correctamente. Nos indica en que formato va a llegar la respuesta del script al que llamamos por Ajax. Si la respuesta no llegase en el formato esperado, se produciría un error en el funcionamiento de $.ajax(). Se trata de un error difícil de localizar, ya que no siempre se obtiene ninguna información acerca del mismo durante el proceso. Simplemente, "algo no funciona" pero no sabemos qué. Por lo tanto, es muy importante que tengamos claro en que formato nos va a responder el script que establecimos en el miembro url. Los posibles valores para dataType son los siguientes:

html. Esperamos texto plano en formato HTML, lo que implica que puede contener etiquetas de HTML 5.
text. Esperamos texto plano, sin más.
json. Esperamos un objeto JSON, que será procesado como tal.
xml. Esperamos una colección de datos en formato XML.
script. Esperamos un fragmento de código JavaScript.

En la práctica, en el día a día real, la mayoría de las necesidades se resuelven con los cuatro primeros tipos mencionados.

EL MIEMBRO success

Este miembro contiene una función que se ejecuta si todo ha ido bien. Esto significa que durante la llamada por Ajax al script especificado en url, así como durante la respuesta de este, no se ha producido ningún error. Si ha habido algún problema, esta función no llegará a ejecutarse nunca. Uno de los problemas más habituales es cuando, por ejemplo, el miembro dataType está mal especificado.

Esta función recibe dos parámetros. El primero es el resultado devuelto por el script al que hemos llamado por Ajax. El segundo es un código relativo al resultado de esta función que, por supuesto, siempre será **success** ya que, cómo hemos dicho, esta función solo llega a ejecutarse si todo el proceso (llamada y respuesta) ha sido correcto. En un ejemplo posterior de este artículo veremos para que nos sirve este segundo parámetro que, por ahora, parece que no nos aporta nada.

El script al que llamamos por Ajax en este ejemplo, de nombre leer_datos.php, obedece al siguiente listado:

```
<?php
 header('Content-type: text/html');
```

```
$campo_nombre = $_GET["dato_nombre"];
$campo_email = $_GET["dato_email"];
$texto = "El nombre es: ".$campo_nombre." y el email es: ".$campo_email.".";
sleep(3);
echo $texto;
?>
```

Cómo ves, poco de particular tiene, salvo la línea resaltada, donde indicamos cual va a ser el tipo de respuesta, para que coincida con la especificada en el miembro dataType del objeto $.ajax().

OTROS MIEMBROS DE $.ajax()

El método $.ajax() puede recibir otros muchos miembros, de los que aquí vamos a ver los más habituales. Los que vamos a recopilar aquí son los más importantes, según mi experiencia profesional y la de otros compañeros.

EL MIEMBRO async

Puede recibir el valor true (que es el valor por defecto, si no especificamos lo contrario). El método $.ajax() trabajará de forma asíncrona, es decir, se ejecutará en un hilo paralelo, mientras se sigue ejecutando el resto del JavaScript o jQuery de la página. Si especificamos el valor false, $.ajax() se comportará de modo síncrono, es decir, bloqueará el funcionamiento del resto del JavaScript hasta haber concluido.

EL MIEMBRO error

Este miembro es complementario de success. Contiene una función que se ejecuta cuando se ha producido un error en la llamada o en la respuesta y, por lo tanto, no se han podido obtener los resultados que se esperaban. Recibe tres parámetros. El primero contiene un objeto XMLHttpRequest que, en este contexto, no nos aporta gran cosa. El segundo parámetro es más relevante. Contiene una etiqueta referente al error que se ha producido. Los más habituales son:

timeout. Cuando el script al que hemos llamado por Ajax ha excedido el tiempo de ejecución que le permite el servidor.
error. Es un error genérico. Es decir, "algo ha fallado, pero no te digo el qué".
abort. El proceso se ha interrumpido de forma inesperada. Puede darse si, por ejemplo, el script llamado por ajax lanza un error fatal.
parsererror. En ocasiones esta indicación se refiere a que el script llamado por Ajax ha intentado devolver un resultado en un formato que no coincide con lo especificado en el miembro dataType, aunque este indicador puede tener otros orígenes, que deberemos explorar en cada caso. Por ejemplo. puede que dataType indique que esperamos un XML, y el resultado sea, efectivamente XML, pero tenga algún error interno o esté mal formado.

El tercer parámetro de la función del miembro error suele ser una pequeña explicación más detallada del error que apareció en el segundo parámetro, y, en ocasiones, ayuda a localizar el fallo.
Obvio es indicar que si el proceso Ajax se ha ejecutado correctamente, la función definida en este miembro no llegará a ejecutarse nunca.

EL MIEMBRO complete

Este miembro contiene una función que se ejecutará siempre que el método $.ajax() termine su trabajo, tanto si todo ha ido bien (y, por tanto, se ejecutó la función del mmiembro success), como si ha habido algún fallo (con lo que se jecutó la función del miembro error). La función que incluye este miembro recibe dos argumentos. El que nos puede interesar realmente es el segundo, que contiene una etiqueta relativa al resultado, que puede ser: success, notmodified, nocontent, error, timeout, abort o parsererror.

PARA FINALIZAR

Este método es más potente de lo que parece. De hecho, una de sus utilidades es el envío por Ajax, no sólo de datos planos, sino, también, de ficheros. Te sugiero que leas el post al respecto que hay en este enlace.