Ajax con jQuery (II)

En el artículo anterior vimos el uso del método load() de jQuery para comunicaciones por ajax. Este método es muy flexible, ya que, a pesar de lo simple de su uso, nos permite elegir, entre otras cosas, si el envío de parámetros va a hacerse por POST o por GET.
En este artículo vamos a conocer dos herramientas de jQuery más especializadas, siendo cada una de ellas para cada uno de los métodos de envío: $.post() y $.get(). Estos métodos funcionan de un modo inherentemente síncrono, con respecto a los datos que gestionan, pero asíncrono con respecto al resto del script. No te asustes: en seguida entenderás este concepto.

CONEXION AJAX POR POST
Vamos a ver un ejemplo del uso de $.post(), para comprobar su funcionamiento. El script principal se llama envio_post.php y su listado es el siguiente:

```
<!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>
  <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...');
     $.post(
      "leer_datos.php",
      {
       'dato_nombre':$('#campo_nombre').prop('value'),
       'dato_email':$('#campo_email').prop('value')
      },
      function (resultado){
```

```
      $('#capaDeEstado').html('DATOS CARGADOS');
      $('#capaDeDatos').html(resultado);
     }
    );
   });
  });
 </script>
</body>
</html>
```

El método $.post() recibe tres argumentos:

La URL del script que vamos a procesar por Ajax, indicando, en su caso, la ruta.
Un objeto, en la notación típica de JavaScript, que contiene los pares nombre-valor de los datos que queremos pasar al script que llamamos en el primer parámetro.
Una función callback, es decir, que se ejecutará cuando haya concluido su trabajo el script al que hemos llamado en el primer parámetro.

Donde debemos fijar nuestra atención es en la función callback del tercer argumento. Esta recibe un parámetro, que corresponde a lo que envía el script al que se llama por Ajax. Este resultado vive, únicamente, dentro del cuerpo de la función, por lo que todo lo que tengamos que hacer con ese resultado deberemos hacerlo dentro del cuerpo de la función.
Además, esa función no se ejecuta hasta que el script al que hemos llamado concluya su trabajo. Por lo tanto, con respecto a los resultados del script llamado, el método $.post() se ejecuta de modo síncrono. Sin embargo, mientras que la función está esperando un resultado, el resto del código que hay fuera del cuerpo del método $.post() continúa ejecutándose, por lo que decimos que, con respecto al código global de la página, el método $.post() se ejecuta de modo asíncrono.
En realidad, lo que ocurre es que, cuando se invoca a este método, hay dos hilos de ejecución corriendo en paralelo: uno es el del propio método y el otro es el global que afecta a todo el JavaScript de la página.

ATENCIÓN. Este método, así cómo el que vamos a comentar a continuación, se limita, en la función callback, a recibir lo que envíe el script al que llamamos en el primer parámetro. No efectúa ningún control sobre ese resultado y, ni siquiera, comprueba que se haya recibido algún resultado o un error. Es responsabilidad del script llamado enviar de vuelta los resultados correctos, o de la función callback cotejar si se ha recibido un resultado del script y si este es lo que se esperaba.

CONEXION AJAX POR GET

Si queremos enviar los datos por GET en lugar de hacerlo por POST, empleamos el métod $.get(). Su funcionamiento es exactamente el mismo que el anterior. Lo único que cambia es el método por el que se envían los datos y, por lo tanto, la forma en que se reciben en el script al que llamamos por Ajax.
Hay que tener en cuenta una cosa que nos llamará la atención si estamos familiarizados con el método load() que vimos en el artículo anterior. Aquí, cuando se envían los datos por GET lo que cambia es el método empleado. Los datos que pasamos al script llamado en el segundo parámetro de $.get() se incluyen, del mismo modo que en $.post(), en notación de objeto, no se cambia a notación de cadena, cómo hacíamos con load().

CONCLUYENDO

En este artículo hemos conocido otra forma de emplear Ajax con jQuery. Se trata de dos métodos que, usados en conjunto en nuestro proyecto, pueden resolver la máyoría de las necesidades de Ajax. Es una forma de trabajar más especializada que la que veíamos con el método load().
En el próximo artículo veremos una forma de trabajar con Ajax más potente y flexible que, definitivamente, nos permitirá cubrir el 100% de nuestras necesidades Ajax, cuando los métodos que hemos visto hasta el momento se nos queden cortos.