AJAX

Medellín - 2014

AJAX.

Descrito de una manera muy resumida, AJAX es una tecnología que nos permite realizar acciones en una página web que necesiten respuesta del servidor sin recargar. Con ello conseguimos que nuestra web sea dinámica y por tanto obtener un diseño más atractivo. Algunos ejemplos de lo que podemos hacer:

- Un enlace que compruebe si un valor existe ya en una base de datos.
- Completar un campo de texto de búsqueda con valores sugeridos por nuestra web.
- Enviar datos para sincronización y luego devolver el estado de los datos.
- Un chat web

Hablando ya de una manera más acertada, Ajax es el acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

En concepto parte del hecho que desde JavaScript podemos realizar solicitudes (HttpRequests) y que por medio que JavaScript no espera el resultado de dicha solicitud para continuar su flujo.

AJAX y jQuery.

jQuery trae incorporado un "módulo" de AJAX, que hace muy fácil la utilización de este. Accedemos a la función de la siguiente manera:

Personalmente siento que el concepto es algo confuso con jQuery, ya

que propiamente no se debería llamar AJAX si no HttpRequest o simplemente Request.

En adelante cuando se mencione AJAX en la jerga de jQuery nos referimos a una función de jQuery que realiza un HttpRequest, espero que quede más claro con algunos ejemplos.

jQuery admite un gran número de parámetros para ajax. Estos son:

Parámetro	Explicacion	Valores
async	Determina que la cargada del objeto ajax sea síncrona o asíncrona. Por lo general asíncrona es más útil, ya que la forma síncrona puede trabar el navegador hasta que la carga este completa	Boolean: true por defecto
beforeSend	Permite llamar una función antes de mandar el objeto ajax.	Función: El único parámetro el objeto.
complete	Es una función que se ejecuta cuando el llamado al ajax está completo. Permite saber si fue exitoso	Función: Los valores que retorna son el objeto XMLHttpRequest y un string que indica el resultado.
contentType	Se usa cuando se mandan datos a los servidores a modo de encabezado.	String: "application/x-www-form-urlencoded" funciona perfectamente
data	Se usa para especificar datos a mandar. Estos tienen la siguiente forma: foo=bar&foo2=bar2;. Si los datos a enviar son un vector (array) jQuery los convierte a varios valores con un mismo nombre (si foo["alt1","alt2"], foo="alt1"&foo="alt2";)	Array / String con la forma antes mencionada.

dataType	Indica el tipo de datos que se van a llamar. Si no se especifica jQuery automáticamente encontrará el tipo basado en el header del archivo llamado (pero toma más tiempo en cargar, así que especifícalo (u_u)	"xml": Devuelve un documento XML. "html": Devuelve HTML con texto plano, y respeta las etiquetas. "script": Evalúa el JavaScript y devuelve texto plano. "json": Evalúa la respuesta JSON y devuelve un objeto Javascript
error	Se ejecuta si ocurre un error al llamar el archivo. Devuelve 3 parámetros: El objeto, un string con el error, y un objeto adicional de error, si este ocurre.	
global	Permite hacer que el objeto ajax obedezca o desobedezca las reglas para objetos ajax que el usuario pone.	Boolean: Por defecto true
ifModified	Permite que el objeto ajax se active solo si la página a cargar se ha modificado.	Boolean: Por defecto false
processData	Por defecto, cada objeto que no sea un string y sea pasado de otro documento, será transformado en cadena. Para evitar que esto pase, este parámetro se puede poner en false	Boolean: Por defecto true.
success	Permite ejecutar código al ser exitoso un llamado.	Función: Recibe los datos que fueron llamados
timeout	Permite definir un tiempo de espera antes de ejecutar un objeto ajax.	Número: Un número de milisegundos (ms)
type	Indica el método que se va a usar	"GET" o "POST"

url	Indica la url de la que va a cargar datos el objeto ajax.	String: La url local del documento.

Como transportar datos entre PHP y AJAX.

Ajax, jQuery y PHP es una combinación de las que uno siempre ha soñado. Nos permite, con relativamente poco esfuerzo, crear aplicaciones interactivas, dinámicas y atractivas.

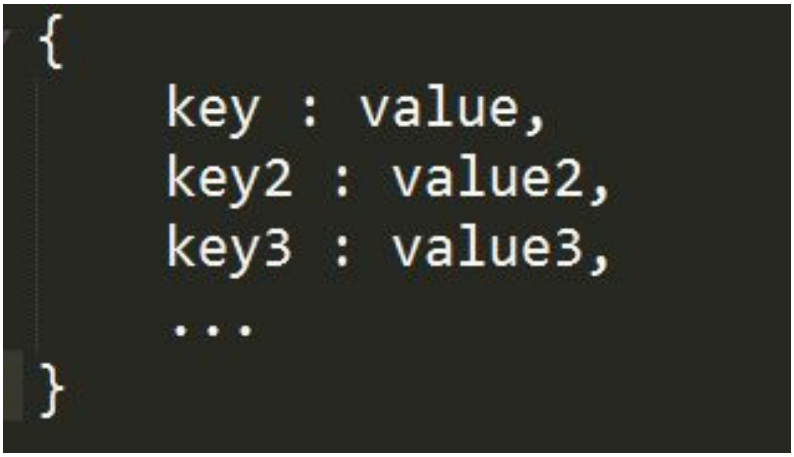
Para el transporte de datos añadimos JSON como formato de intercambio de datos y de esta manera la cosa aún se pone mejor. En el video tutorial que les grabare vamos a ver un pequeño ejemplo, explicado paso a paso, de todo esto junto y funcionando: jQuery, Ajax, PHP y JSON.

Qué es JSON

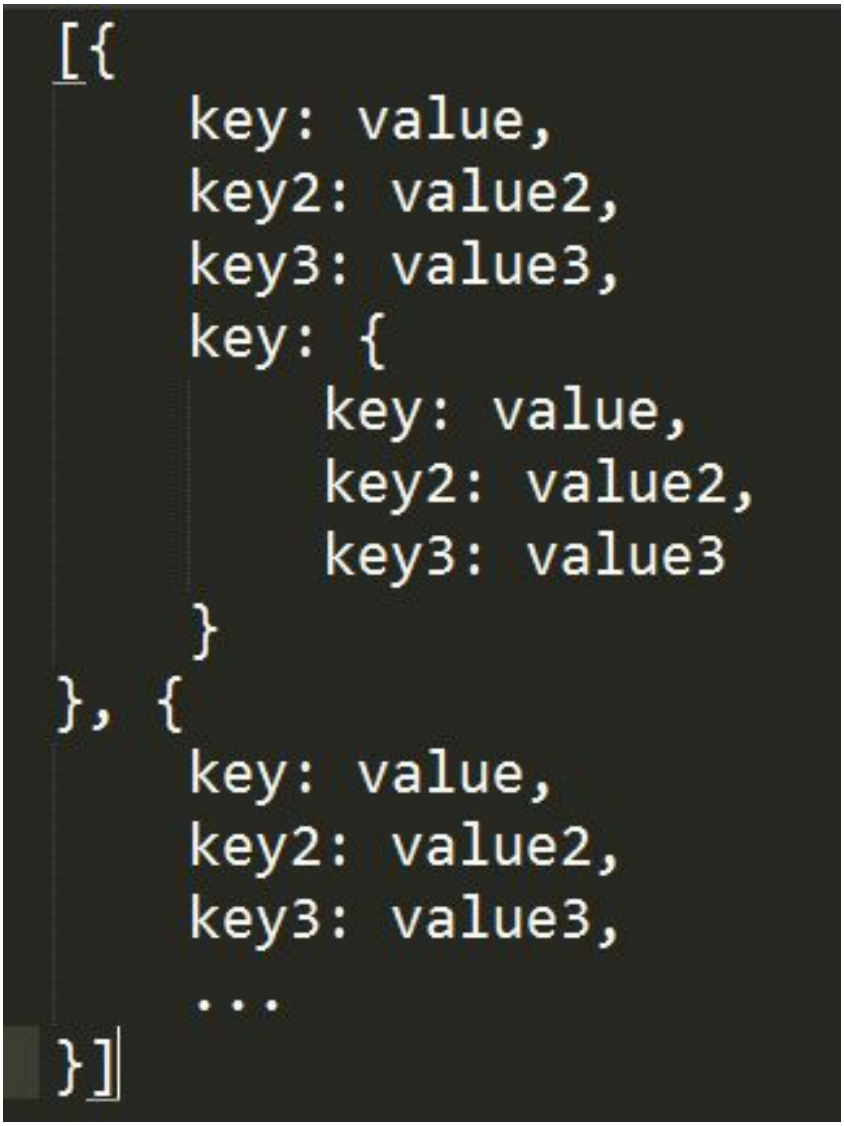
Aunque con Ajax se puede solicitar cualquier tipo de recurso web, el intercambio de datos entre la página web y el servidor ha sido realizado tradicionalmente, como el propio nombre indica, **en formato XML (eXtensible Markup Language)**, un lenguaje de marcas que permite definir una gramática específica y, por tanto, permite el intercambio de información estructurada y legible.

Pero, hay gente que nunca se queda con lo mismo y de esta manera llegó JSON **(JavaScript Object Notation),** más ligero y con una notación más simple. Este se convirtió en el formato más utilizado para el intercambio de datos cuándo se trabaja con Ajax. Además, con una ligera variación (JSONP) puede utilizarse en solicitudes Ajax entre diferentes dominios (cross-domain).

El formato JSON tiene la siguiente notación:



Y también puede ser serializado y multidimensional, por ejemplo:



Todo lo relaciona con los formatos o con las especificaciones del formato JSON lo podemos encontrar en su página oficial http://json.org.

JSON en PHP

En cualquier instalación estándar de PHP, desde la versión 5.2.0, se **incorpora de forma predeterminada la extensión JSON** y es muy fácil pasar los datos de un array ha notación JSON. **Aunque si no se cuenta con esta extensión se puede construir la notación JSON de forma manual**, es recomendable tenerla si vas a trabajar de forma frecuente con datos en formato JSON.

Entre las funciones de la extensión JSON para PHP que nos interesan**, la más imprescindible es la función json_encode().** Con esta función podemos pasar nuestros datos a formato JSON rápida y fácilmente. Sólo es necesario tener los datos en forma de array u objeto.

Por ejemplo:

```php
<?php

$jsondata = array();

if( isset($_POST['user']) ) {

    if( $_POST['user'] == 'admin' ) {

        $jsondata['success'] = true;
        $jsondata['message'] = 'Hola! El usuario recibido es correcto.';

    } else {

        $jsondata['success'] = false;
        $jsondata['message'] = 'Hola! El usuario recibido no es correcto.';

    }

    //Aunque el content-type no sea un problema en la mayoría de casos, es re
    header('Content-type: application/json; charset=utf-8');
    echo json_encode($jsondata);
    exit();
}

?>
```

Ajax con JSON y jQuery

Uno de los argumentos del método jQuery.ajax() es dataType y aquí podemos especificar que vamos a utilizar datos JSON. Si no especificamos nada en dataType, jQuery intentará detectar de forma automática el formato de los datos recibidos. Si en la respuesta desde el servidor se especifica, como hicimos en el ejemplo anterior, el tipo de contenido, será más fácil para jQuery tratar los datos correctamente. Utilizando jQuery.ajax() tendría un aspecto similar a:

```
function process1() {
    $.ajax({
        type: "POST",
        url: "process.php",
        data: {
            "user": "admin"
        },
        dataType: "json",
        success: function(data) {
            console.log(
                "La solicitud se ha completado correctamente."
            );
        },
        error: function(jqXHR, textStatus, errorThrown) {
            console.log(jqXHR);
            console.log(textStatus);
            console.log(errorThrown);
            console.log("La solicitud a fallado: " + textStatus
                .statusText);
        }
    });
}
```

Esta era la forma más antigua de trabajar con AJAX en jQuery, pero desde versiones recientes se empezó a implementar lo siguiente:

```
function process2() {
    $.ajax({
        statusCode: {
            404: function() {
                alert("page not found");
            },
            500: function() {
                alert("500");
            },
        },
        data: {
            "user": "admin"
        },
        //Cambiar a type: GET si necesario
        type: "POST",
        dataType: "json",
        url: "process.php",
    }).done(function(data, textStatus, jqXHR) {
        console.log("La solicitud se ha completado correctamente.");
    }).fail(function(jqXHR, textStatus, errorThrown) {
        console.log("La solicitud a fallado: " + textStatus);
    });
}
```

Como podemos observar, esta nueva forma de trabajar con AJAX no trae funciones de success o error, pues ya trae las funciones done y fail que son mucho más rápidas y seguras.

Hay también otra manera de trabajar, la cual es llamar el tipo de petición que se realizará. Este ejemplo es el siguiente:

```
function process3() {
    //Equivalente a lo anterior
    $.post("process.php", {
        "user": "admin"
    }, null, "json").done(function(data, textStatus, jqXHR) {
        console.log("La solicitud se ha completado correctamente.");
    }).fail(function(jqXHR, textStatus, errorThrown) {
        console.log("La solicitud a fallado: " + textStatus);
    });
}
```

Como podemos observar, esta función por defecto sabe que se enviara por post y recibe 4 parámetros:

1. Archivo que procesara el Ajax
2. Datos
3. Función de callback para que cuando devuelva los datos podamos obtenerlos
4. El tipo de datos que recibirá el Ajax.

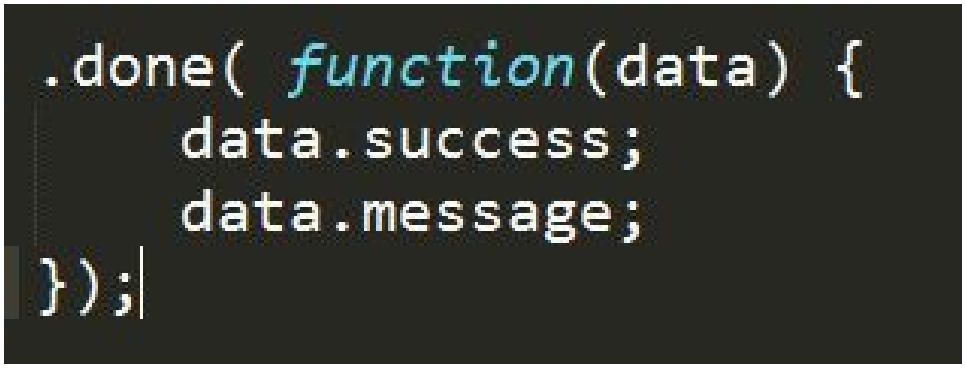
De esta manera, también podemos acceder a las funciones **done y fail.**

En el ejemplo PHP anterior, en el objeto JSON había dos miembros: success y message. Un posible objeto JSON recibido podría ser el siguiente:

```
{"success" : true, "message" : "Hola! El valor recibido es correcto."}
```

Este objeto es recibido en el método **.done() o en el success** a través del argumento data y podemos acceder al valor de cada miembro del

objeto JSON del siguiente modo:



Espero que podamos seguir navegando en este mundo del Ajax y conocer un poco más de sus funcionalidades.

Un saludo.