|  |
| --- |
| **¿Qué es Ajax y que es JSON?** |

Ajax son las siglas de *Asynchronous JavaScript And XML* y es una tecnología que permite a una página web actualizarse de forma dinámica sin que tenga que recargarse completamente. JavaScript es el encargado de comunicarse con el servidor enviando y recibiendo datos desde la página web, en el servidor la solicitud es procesada y se envía una respuesta que es interpretada de nuevo por JavaScript en la página web.

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.

JSON (*JavaScript Object Notation*), más ligero y con una notación más simple, 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:

{key : value, key2 : value2, key3 : value3,...}

Y también puede ser serializado y multidimensional, por ejemplo:

[{key : value, key2 : value2, key3 : value3, key : { key : value, key2 : value2, key3 : value3} },{key : value, key2 : value2, key3 : value3,...}]

Puedes ver todas las especificaciones del formato JSON en [json.org](http://www.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](http://www.php.net/manual/es/book.json.php) siendo muy fácil pasar los datos de un array a notación en 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 formado JSON rápida y fácilmente. Sólo es necesario tener los datos en forma de array u objeto. (<https://www.tutorialspoint.com/json/json_php_example.htm>)

Por ejemplo:

*<?php*

*$jsondata = array();*

*if( isset($\_GET['param']) ) {*

*if( $\_GET['param'] == 'valor' ) {*

*$jsondata['exito'] = true;*

*$jsondata['mensaje'] = 'Hola! El valor recibido es correcto.';*

*}*

*else {*

*$jsondata['exito'] = false;*

*$jsondata['mensaje'] = 'Hola! El valor recibido no es correcto.';*

*}*

*//Aunque el content-type no sea un problema en la mayoría de casos, es*

*recomendable especificarlo*

*header('Content-type: application/json; charset=utf-8');*

*echo json\_encode($jsondata);*

*exit();*

*}*

## Ajax con JSON y jQuery

Uno de los parámetros del método jQuery.ajax() es dataType. Aquí podemos especificar que vamos a **utilizar datos JSON en la respuesta**. Si no especificamos nada en dataType, jQuery intetará 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:

<?php

$.ajax({

// En data puedes utilizar un objeto JSON, un array o un query string

data: {"parametro1" : "valor1", "parametro2" : "valor2"},

//Cambiar a type: POST si necesario

type: "GET",

// Formato de datos que se espera en la respuesta

dataType: "json",

// URL a la que se enviará la solicitud Ajax

url: "guion.php",

})

.done(function( data, textStatus, jqXHR ) {

if ( console && console.log ) {

console.log( "La solicitud se ha completado correctamente." );

}

})

.fail(function( jqXHR, textStatus, errorThrown ) {

if ( console && console.log ) {

console.log( "La solicitud a fallado: " + textStatus);

}

});

?>

Además del método anterior, podemos utilizar el atajo [**getJSON()**](http://api.jquery.com/jquery.getjson/), que quedaría de esta forma para hacer exactamente lo mismo:

$.getJSON( "script.php", { "parametro1" : "valor1", "parametro2" : "valor2" } )

.done(function( data, textStatus, jqXHR ) {

if ( console && console.log ) {

console.log( "La solicitud se ha completado correctamente." );

}

})

.fail(function( jqXHR, textStatus, errorThrown ) {

if ( console && console.log ) {

console.log( "Algo ha fallado: " + textStatus" );

}

});

Debemos tener en cuenta que .getJSON(), como su nombre indica, **siempre hace una solicitud con el método GET.** Si necesitamos enviar la solicitud mediante método POST tendremos que utilizar otras funciones de jQuery como [.ajax()](http://api.jquery.com/jquery.ajax/) con el parámetro type puesto a POST o su alias [.post()](http://api.jquery.com/jquery.post/). Por ejemplo:

<?php

$.ajax({

data: {"parametro1" : "valor1", "parametro2" : "valor2"},

type: "POST",

dataType: "json",

url: "script.php",

})

.done(function( data, textStatus, jqXHR ) {

if ( console && console.log ) {

console.log( "La solicitud se ha completado correctamente." );

}

})

.fail(function( jqXHR, textStatus, errorThrown ) {

if ( console && console.log ) {

console.log( "La solicitud a fallado: " + textStatus);

}

});

//Equivalente a lo anterior

$.post( "script.php", { "parametro1" : "valor1", "parametro2" : "valor2" }, null, "json" )

.done(function( data, textStatus, jqXHR ) {

if ( console && console.log ) {

console.log( "La solicitud se ha completado correctamente." );

}

})

.fail(function( jqXHR, textStatus, errorThrown ) {

if ( console && console.log ) {

console.log( "La solicitud a fallado: " + textStatus);

}

});

?>

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

{"exito" : true, "mensaje" : "Hola! El valor recibido es correcto."};

Este objeto es recibido en el método .done() a través del argumento data y podemos acceder al valor de cada miembro del objeto JSON del siguiente modo:

.done( function(data) {

data.exito;

data.mensaje;

});

|  |
| --- |
| Ejemplo paso a paso Vamos a ver un ejemplo en el que vamos a realizar una solicitud **ajax** a un script PHP que interaccionará con una base de datos para buscar los registros de los usuarios. El ejemplo cubre varias posibilidades, entre ellas que se solicite sólo un registro de la base de datos o varios. |