|  |  |
| --- | --- |
| Geo Parking | AJAX  2014 |

El documento describe información de AJAX que servirá como capacitación en el desarrollo del producto Geo Parking.

# Control de la documentación

### Control de la Configuración.

|  |  |
| --- | --- |
| Título: | AJAX |
| Referencia: | GeoP\_Proyecto\_Ajax.docx |
| Autores: | Lucas Toneatoo |
| Fecha: | 23/08/2014 |

### Histórico de Versiones.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Versión | Fecha | Estado | Responsable | Cambios |
| 1.0 | 2/05/2014 | Aprobado | Lucas Toneatto | - |
|  |  |  |  |  |
|  |  |  |  |  |

Contenido

[Control de la documentación 2](#_Toc396560074)

[Control de la Configuración. 2](#_Toc396560075)

[Histórico de Versiones. 2](#_Toc396560076)

[Introducción a AJAX 3](#_Toc396560077)

[Funciones para AJAX en JQuery 7](#_Toc396560078)

[Referencias 8](#_Toc396560079)

[Ejemplos 8](#_Toc396560080)

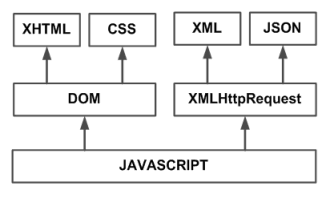
# Introducción a AJAX

AJAX es un acrónimo de *Asynchronous JavaScript + XML*, que se puede traducir como "JavaScript asíncrono + XML".

*“Ajax no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes.”*

Las tecnologías que forman AJAX son:

* XHTML y CSS, para crear una presentación basada en estándares.
* DOM, para la interacción y manipulación dinámica de la presentación.
* XML, XSLT y JSON, para el intercambio y la manipulación de información.
* XMLHttpRequest, para el intercambio asíncrono de información.
* JavaScript, para unir todas las demás tecnologías.

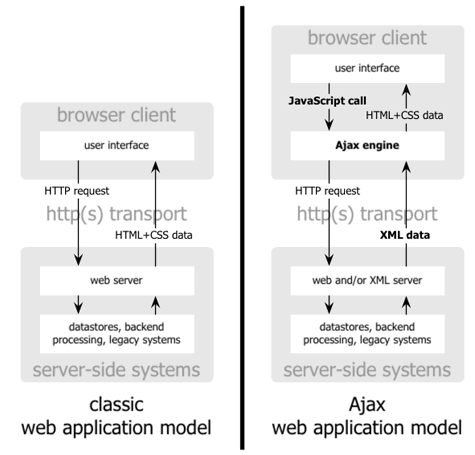


**Figura** Tecnologías agrupadas bajo el concepto de AJAX

Desarrollar aplicaciones AJAX requiere un conocimiento avanzado de todas y cada una de las tecnologías anteriores.

En las aplicaciones web tradicionales, las acciones del usuario en la página (pinchar en un botón, seleccionar un valor de una lista, etc.) desencadenan llamadas al servidor. Una vez procesada la petición del usuario, el servidor devuelve una nueva página HTML al navegador del usuario.

En el siguiente esquema, la imagen de la izquierda muestra el modelo tradicional de las aplicaciones web. La imagen de la derecha muestra el nuevo modelo propuesto por AJAX:



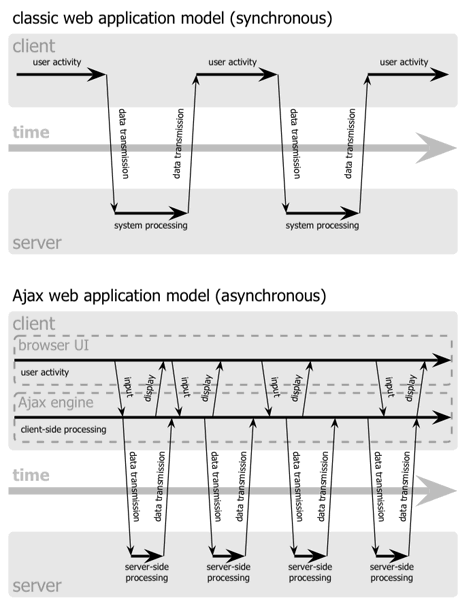
**Figura 1.2** Comparación gráfica del modelo tradicional de aplicación web y del nuevo modelo propuesto por AJAX.

Esta técnica tradicional para crear aplicaciones web funciona correctamente, pero no crea una buena sensación al usuario. Al realizar peticiones continuas al servidor, el usuario debe esperar a que se recargue la página con los cambios solicitados. Si la aplicación debe realizar peticiones continuas, su uso se convierte en algo molesto

AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano.

Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor.

El siguiente esquema muestra la diferencia más importante entre una aplicación web tradicional y una aplicación web creada con AJAX. La imagen superior muestra la interación síncrona propia de las aplicaciones web tradicionales. La imagen inferior muestra la comunicación asíncrona de las aplicaciones creadas con AJAX.



**Figura 1.3** Comparación entre las comunicaciones síncronas de las aplicaciones web tradicionales y las comunicaciones asíncronas de las aplicaciones.

Las peticiones HTTP al servidor se sustituyen por peticiones JavaScript que se realizan al elemento encargado de AJAX. Las peticiones más simples no requieren intervención del servidor, por lo que la respuesta es inmediata. Si la interacción requiere una respuesta del servidor, la petición se realiza de forma asíncrona mediante AJAX. En este caso, la interacción del usuario tampoco se ve interrumpida por recargas de página o largas esperas por la respuesta del servidor.

Desde su aparición, se han creado cientos de aplicaciones web basadas en AJAX. En la mayoría de casos, AJAX puede sustituir completamente a otras técnicas como Flash. Además, en el caso de las aplicaciones web más avanzadas, pueden llegar a sustituir a las aplicaciones de escritorio.

# Funciones para AJAX en JQuery

Las funciones y utilidades relacionadas con AJAX son parte fundamental de jQuery. El método principal para realizar peticiones AJAX es $.ajax(). A partir de esta función básica, se han definido otras funciones relacionadas, de más alto nivel y especializadas en tareas concretas: $.get(), $.post(), $.load(), etc.

La sintaxis de $.ajax() es muy sencilla:

$.ajax(opciones);

La URL que se solicita también se incluye dentro del array asociativo de opciones. A continuación se muestra un ejemplo básico que se utiliza con $.ajax():

$.ajax({

url: '/ruta/hasta/pagina.php',

type: 'POST',

async: true,

data: 'parametro1=valor1&parametro2=valor2',

success: procesaRespuesta,

error: muestraError

});

La siguiente tabla muestra todas las opciones que se pueden definir para el método $.ajax():

|  |  |
| --- | --- |
| **Opción** | **Descripción** |
| **async** | Indica si la petición es asíncrona. Su valor por defecto es true, el habitual para las peticiones AJAX |
| **beforeSend** | Permite indicar una función que modifique el objeto XMLHttpRequest antes de realizar la petición. El propio objeto XMLHttpRequest se pasa como único argumento de la función |
| **complete** | Permite establecer la función que se ejecuta cuando una petición se ha completado (y después de ejecutar, si se han establecido, las funciones de success o error). La función recibe el objeto XMLHttpRequest como primer parámetro y el resultado de la petición como segundo argumento |
| **contentType** | Indica el valor de la cabecera Content-Type utilizada para realizar la petición. Su valor por defecto es application/x-www-form-urlencoded |
| **data** | Información que se incluye en la petición. Se utiliza para enviar parámetros al servidor. Si es una cadena de texto, se envía tal cual, por lo que su formato debería ser parametro1=valor1&parametro2=valor2. También se puede indicar un array asociativo de pares clave/valor que se convierten automáticamente en una cadena tipo *query string* |
| **dataType** | El tipo de dato que se espera como respuesta. Si no se indica ningún valor, jQuery lo deduce a partir de las cabeceras de la respuesta. Los posibles valores son: xml (se devuelve un documento XML correspondiente al valor responseXML), html (devuelve directamente la respuesta del servidor mediante el valor responseText), script (se evalúa la respuesta como si fuera JavaScript y se devuelve el resultado) y json (se evalúa la respuesta como si fuera JSON y se devuelve el objeto JavaScript generado) |
| **error** | Indica la función que se ejecuta cuando se produce un error durante la petición. Esta función recibe el objeto XMLHttpRequest como primer parámetro, una cadena de texto indicando el error como segundo parámetro y un objeto con la excepción producida como tercer parámetro |
| **ifModified** | Permite considerar como correcta la petición solamente si la respuesta recibida es diferente de la anterior respuesta. Por defecto su valor es false |
| **processData** | Indica si se transforman los datos de la opción data para convertirlos en una cadena de texto. Si se indica un valor de false, no se realiza esta transformación automática |
| **success** | Permite establecer la función que se ejecuta cuando una petición se ha completado de forma correcta. La función recibe como primer parámetro los datos recibidos del servidor, previamente formateados según se especifique en la opción dataType |
| **timeout** | Indica el tiempo máximo, en milisegundos, que la petición espera la respuesta del servidor antes de anular la petición |
| **type** | El tipo de petición que se realiza. Su valor por defecto es GET, aunque también se puede utilizar el método POST |
| **url** | La URL del servidor a la que se realiza la petición |

# Referencias

* <http://librosweb.es/ajax/>
* <http://www.uco.es/~lr1maalm/manualdeajax.pdf>
* <http://www.w3schools.com/ajax/default.ASP>
* <http://api.jquery.com/jquery.ajax/>
* <http://www.w3schools.com/jquery/jquery_ajax_intro.asp>

# Ejemplos

* <http://www.esasp.net/2009/11/ajax-jquery-en-aspnet.html>
* <http://aspsnippets.com/Articles/Call-ASPNet-Page-Method-using-jQuery-AJAX-Example.aspx>
* <http://codehero.co/jquery-desde-cero-ajax/>