APRENDERAPROGRAMAR.COM

QUÉ ES Y PARA QUÉ SIRVE AJAX. DIFERENCIA CON JAVASCRIPT . VENTAJAS E INCONVENIENTES. (CU01204F)

Sección: Cursos
Categoría: Tutorial básico del programador web: Ajax desde cero

Fecha revisión: 2031

Resumen: Entrega nº4 del Tutorial básico “Ajax desde cero”.

Autor: Alex Rodríguez

QUÉ ES Y PARA QUÉ SIRVE AJAX

Ajax puede verse como una extensión de JavaScript que facilita técnicas que nos permiten conectar con un servidor web dinámicamente. Una posible definición para Ajax es que es "el método o conjunto de técnicas que permiten intercambiar información con un servidor y actualizar parte de los contenidos de una web sin necesidad de recargar la página web completamente".

Se pueden encontrar distintas definiciones para Ajax. En algunos casos se dice que es una tecnología o herramienta, en otros casos se dice que es un lenguaje de programación, en otros casos que no es un lenguaje sino una extensión de JavaScript, y en otros casos que no es nada de lo anterior sino un concepto. La mejor forma de entender qué es Ajax es trabajar con esta tecnología, y de eso nos vamos a ocupar a lo largo del curso, viendo los conceptos y ejemplos necesarios.

Antes de empezar a trabajar con código vamos a repasar algunas cuestiones básicas que nos serán útiles para comprender Ajax.

VENTAJAS Y LÍMITES DE JAVASCRIPT

La primera cuestión a tratar brevemente es que si Ajax extiende JavaScript, ¿qué es JavaScript? JavaScript es un lenguaje utilizado para dotar de efectos y procesos dinámicos e "inteligentes" a documentos HTML. Un documento HTML viene siendo coloquialmente "una página web". Así, podemos decir que el lenguaje JavaScript sirve para ejecutar acciones rápidas y efectos animados en las páginas web. Las acciones controladas por JavaScript pueden ser el despliegue de un menú, hacer aparecer, desaparecer o cambiar texto e imágenes, realizar cálculos y mostrar resultados, mostrar mensajes de aviso (por ejemplo si faltan datos en un formulario) y "efectos animados" en general.

JavaScript es principalmente utilizado por parte de programadores web para dar respuestas rápidas a las acciones del usuario sin necesidad de enviar la información al servidor y esperar respuesta de éste (lo que haría más lento los procesos). El código JavaScript se carga al mismo tiempo que el código HTML en el navegador, y reside en el cliente (computador en el que nos encontramos), por lo que JavaScript sigue funcionando incluso aunque se produzca un corte en la conexión a internet (en este caso no podremos seguir navegando hacia otras direcciones web, pero sí podremos ejecutar procesos "locales" en nuestro computador para la página web en que nos encontráramos).

La navegación por internet suele basarse en el proceso básico de envío de una petición (que puede llevar incorporada información como los datos de un formulario) a un servidor, esperar respuesta por parte del servidor y recibir la respuesta en nuestro computador. Cada proceso de este tipo consume tiempo, el tiempo total podríamos verlo desde el lado de nuestro computador como Tiempo Total Proceso = tiempo envío petición + tiempo procesamiento petición + tiempo recepción respuesta.

Aún con velocidades rápidas de navegación cuantos más procesos de este tipo realicemos más lenta será la navegación web. JavaScript podemos decir que supone que las respuestas del servidor sean más completas y permite que se realicen más procesos en nuestro computador (aquellos procesos que realmente pueden ser resueltos en nuestro propio computador sin necesidad de estar enviando peticiones al servidor), de modo que se reduce el número de peticiones y respuestas necesarias entre cliente y servidor. Todo esto lleva a que la navegación e interacción con las páginas web sea más cómoda y rápida.

El código JavaScript es interpretado directamente por el navegador web, sin necesidad de otros programas o procesos intermedios.

No vamos a extendernos con JavaScript porque suponemos que si estás siguiendo este curso ya conoces este lenguaje y lo que supone. Una ventaja importante de JavaScript es hacer más ágil y dinámica la navegación por páginas web, evitando los tiempos de espera.

¿Significa esto que podemos hacer todo mediante JavaScript? No, JavaScript presenta limitaciones que hacen que no sea adecuado como para "hacerlo todo con JavaScript".

Hay varios motivos por lo que en los desarrollos web profesionales se combinan procesos del lado del cliente con procesos del lado del servidor. Vamos a citar algunos y para ello nos valdremos del ejemplo de una tienda de comercio electrónico.

a) Los datos en la web cambian con frecuencia. Para que los datos se mantengan actualizados es necesario refrescar la información haciendo nuevas peticiones al servidor. Para que el usuario vaya navegando por la tienda quizás podamos enviar los datos de 10 ó 12 productos pero para cargar nuevos productos será lógico hacer una nueva petición al servidor.

b) Los datos pueden sobrecargar el computador del usuario. Si tenemos una tienda con 7.000 productos y enviáramos todos los datos al computador del usuario para que fueran gestionados mediante JavaScript tendríamos problemas. En primer lugar, el envío de volúmenes muy grandes de información consume mucho tiempo (y posiblemente el usuario se vaya a otra tienda si lo hacemos esperar demasiado). En segundo lugar, el computador del usuario puede tener problemas para gestionar volúmenes demasiado grandes de información (sobrecarga). Los volúmenes grandes de información normalmente residen en bases de datos gestionadas por el servidor y los datos son servidos en pequeños paquetes de datos a medida que resulta necesario.

c) Hay procesos que tienen que ser realizados del lado del servidor porque necesitan de verificaciones de seguridad que no pueden residir en el computador de un usuario. Por ejemplo, para el pago con una tarjeta de crédito es necesario que el usuario envíe el número de su tarjeta de crédito al servidor y que éste mediante un proceso seguro verifique la tarjeta y el pago. Sería disparatado pensar en enviar los números de tarjetas de crédito válidas al computador del usuario y que el proceso tuviera lugar en el computador cliente para luego informar al servidor de que el pago es correcto.

Conforme vayamos adquiriendo experiencia como programadores nos daremos cuenta de que hay procesos que claramente es más adecuado realizarlos del lado del servidor, otros que claramente es más adecuado realizarlos del lado del cliente, y otros que podrían realizarse tanto del lado del servidor como del lado del cliente. También con la experiencia iremos aprendiendo a tomar decisiones relacionadas con esto.

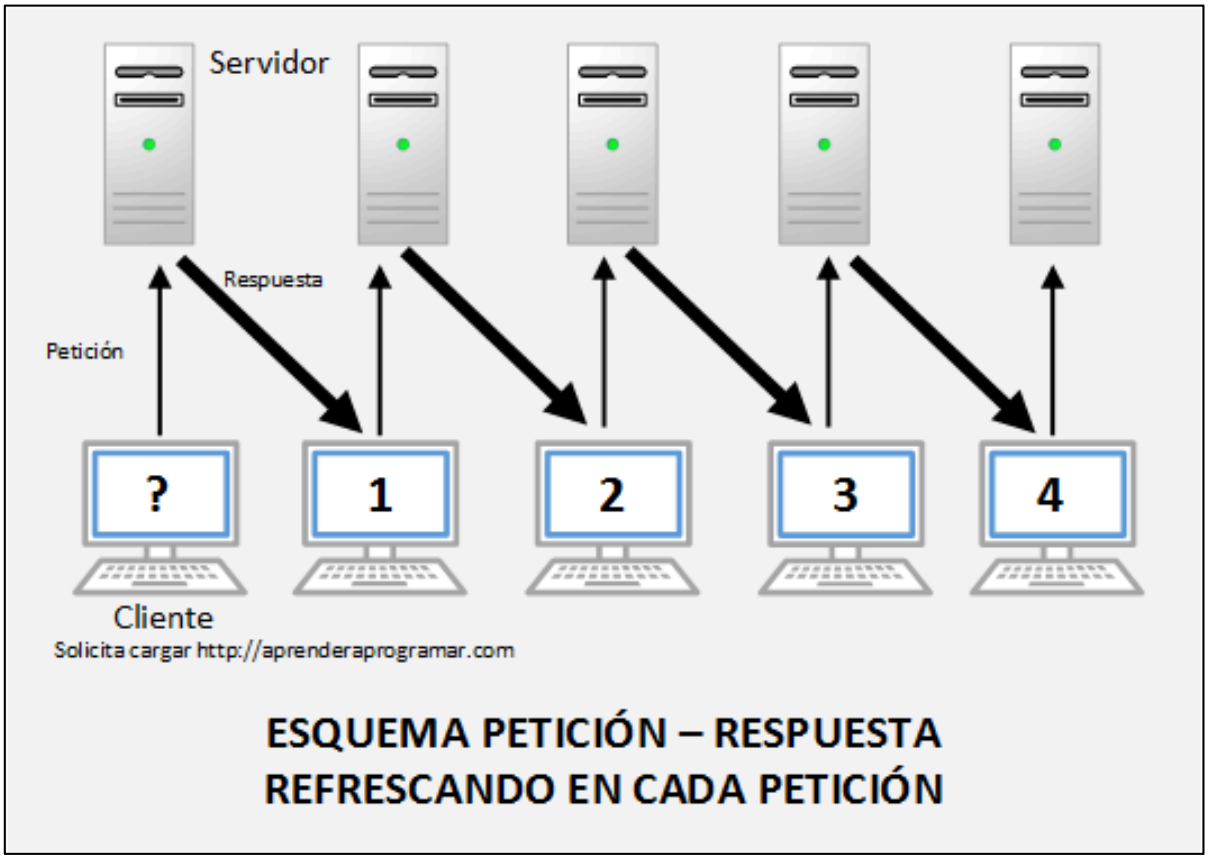
En este esquema vemos cómo se combina un lenguaje del lado del servidor con JavaScript y HTML (no citamos CSS pero obviamente CSS debe incluirse también en la respuesta del servidor).

Aquí hemos indicado como lenguaje del lado del servidor PHP, pero podría ser cualquier otro como ASP, JSP, etc.

El problema en este esquema es que para cualquier cambio de información en la web que debamos traer del servidor tendríamos que hacer una petición y recarga completa de la página, y esto consume tiempo (y desespera al usuario).

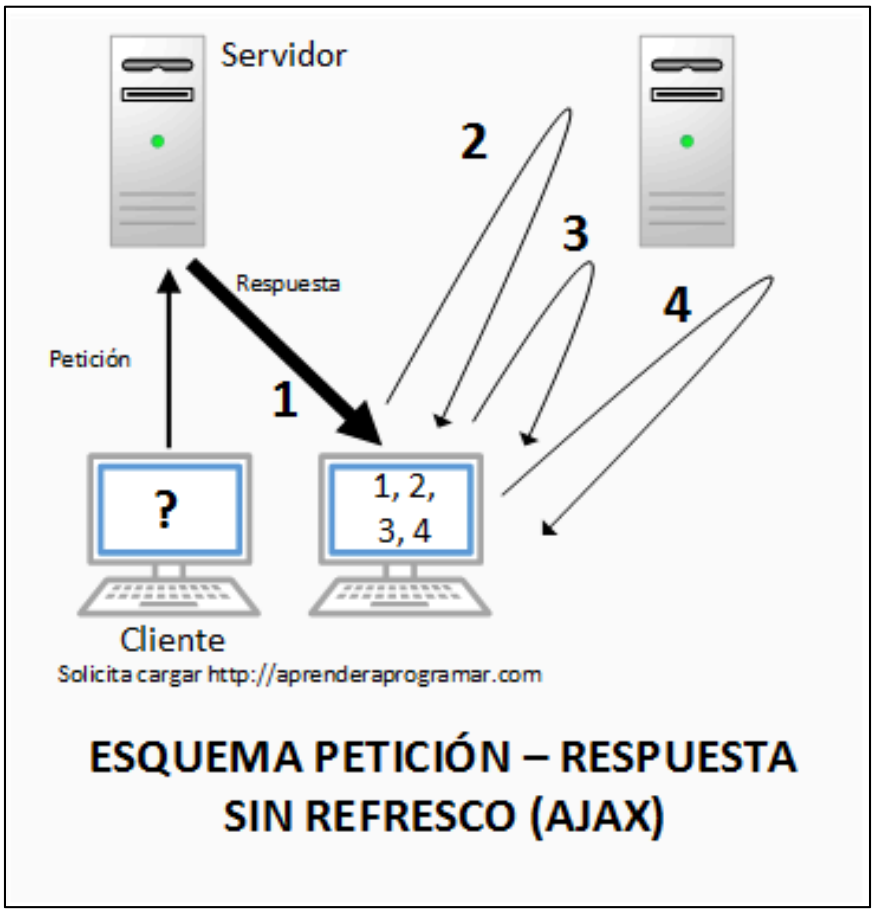
LAS VENTAJAS DE AJAX

Ajax introduce la posibilidad de intercambiar información con el servidor sin necesidad de recargar completamente todo el contenido de la página web. Los siguientes esquemas reflejan las ventajas que brinda Ajax. Empezaremos viendo cómo podría ser un proceso sin usar Ajax:



En este esquema suponemos que el navegador inicialmente contiene una página cualquiera (?) y se le hace la petición para que muestre una página como http://aprenderaprogramar.com. Una vez realizada la solicitud la página se carga en el navegador (1). Suponemos que el usuario realiza una acción, por ejemplo elige entre varios tipos de producto la opción “automóviles”. Como respuesta a ese evento, se envía la información (opción automóviles) al servidor y se realiza una recarga completa de la página donde se le muestran los tipos de automóviles disponibles, por ejemplo autobuses, furgonetas y coches (2). Ahora el usuario elige un tipo de automóvil, por ejemplo coches, y como respuesta a ese evento, se envía la información al servidor, se recarga completamente la página y se le muestran los tipos de coches disponibles, por ejemplo “de gasolina”, “diesel”, “eléctricos” e “híbridos” (3). Ahora el usuario elige una opción, por ejemplo “eléctricos”, y como respuesta a ese evento, se envía la información al servidor y se recarga completamente la página para mostrarle los tipos de coches eléctricos disponibles (4). En total hemos realizado el envío de 4 paquetes de datos y 4 cargas completas de la página. Supongamos que cargar la página supone la transferencia de 250 Kb de media y que el envío de datos es despreciable frente a lo que supone la carga de la página. En total habremos tenido que hacer 4 recargas y transferir 1000 Kb, aproximadamente 1 Mb.

Veamos ahora el esquema usando Ajax:



En este esquema suponemos que el navegador inicialmente contiene una página cualquiera (?) y se le hace la petición para que muestre una página como http://aprenderaprogramar.com. Una vez realizada la solicitud la página se carga en el navegador (1). Suponemos que el usuario realiza una acción, por ejemplo elige entre varios tipos de producto la opción “automóviles”. Como respuesta a ese evento, se envía la información (opción automóviles) al servidor en segundo plano y se reciben datos de respuesta del servidor también en segundo plano, sin necesidad de recargar de nuevo toda la página. Con la información recibida del servidor y usando JavaScript se modifica la página únicamente allí donde es necesario, de modo que ahora se le muestran al usuario los tipos de automóviles disponibles, por ejemplo autobuses, furgonetas y coches (2). Ahora el usuario elige un tipo de automóvil, por ejemplo coches, y como respuesta a ese evento, se envía la información al servidor en segundo plano y se reciben datos de respuesta del servidor también en segundo plano, sin necesidad de recargar de nuevo toda la página. Con la información recibida del servidor y usando JavaScript se modifica la página únicamente allí donde es necesario, de modo que ahora se le muestran al usuario los tipos de coches disponibles, por ejemplo “de gasolina”, “diesel”, “eléctricos” e “híbridos” (3). Ahora el usuario elige una opción, por ejemplo “eléctricos”, y como respuesta a ese evento, se envía la información al servidor en segundo plano y se reciben datos de respuesta del servidor también en segundo plano, sin necesidad de recargar de nuevo toda la página. Con la información recibida del servidor y usando JavaScript se modifica la página únicamente allí donde es necesario, de modo que ahora se le muestran al usuario para mostrarle los tipos de coches eléctricos disponibles (4). En total hemos realizado 1 cargas completa de la página, y enviado 4 paquetes de datos al servidor y recibido 4 paquetes de datos del

servidor. Supongamos que cargar la página supone la transferencia de 250 Kb de media y que cada paquete enviado tiene 1 Kb de media y cada paquete recibido 5 Kb de media. En total habremos tenido que hacer 1 recarga y transferir 250 + 4*1 + 4*5 Kb, aproximadamente 250 + 4 +20 = 274 Kb

La diferencia que ha introducido el uso de Ajax está en que:

- No hemos tenido que cargar la página completa varias veces.
- Hemos sido más rápidos en la operación y al mostrar respuestas al usuario.
- Hemos transferido un menor volumen de datos.
- La transferencia de datos con el servidor es en segundo plano. Esto permite al usuario seguir interaccionando con la página web. En cambio en el caso de recargas completas tendría que esperar a que terminara la recarga para continuar interaccionando.

El uso de Ajax introduce una ventaja clara, de ahí que la mayor parte de las webs hoy día usen Ajax.

INCONVENIENTES DE AJAX

Dado que Ajax permite el intercambio de datos, podríamos pensar en usar una sola url para mostrar numerosos artículos según la elección del usuario. Por ejemplo en la url aprenderaprogramar.com podríamos mostrar artículos sobre Java si el usuario elige Java, artículos sobre C# si el usuario elige C#, artículos sobre Visual Basic si el usuario elige Visual Basic, etc.

Pero esto no es buena idea por varios motivos:

- Introducir urls y refresco de la página web permite la trazabilidad y depuración de la programación web de forma más adecuada que si todo se mantiene en una sola url. Imagina que tienes una consulta médica donde tienes distintos aparatos para realizar procesos (por ejemplo radiografías, análisis de sangre, resonancia magnética, etc.). ¿Qué será más adecuado, tener todos los aparatos y procesos en una gran sala o tener cada proceso en su sala independiente? Posiblemente tener cada proceso en su sala independiente. Si tienes todos los aparatos y procesos en una sola sala será más fácil equivocarse, por ejemplo que el paciente se realice una prueba incorrecta o que una muestra quede mal etiquetada. Sin embargo no será cómodo para un paciente que lo lleves a una sala para quitarse la ropa y tener que atravesar el vestíbulo sin ropa para hacerse la radiografía en otra sala. En resumen, hay que agrupar procesos siempre que sea lógico y razonable, sin pretender aglutinarlo todo en una sola ubicación.

- Introducir urls y refresco de la página web permite un mejor indexado de los contenidos por los buscadores y un mejor posicionamiento SEO. Imagina que tienes varios artículos en una misma url y que se muestra uno u otro según la elección del usuario. Esto resulta difícil de rastrear para un buscador de Google, Bing o similar.

- El usuario puede perder la capacidad para hacer cosas que hacía con webs tradicionales puesto que no hay cambio de página web. Por ejemplo usar los botones de avance y retroceso del navegador o añadir una página a favoritos puede dejar de ser posible. Esto en algunos casos no es deseable.

- Existen problemas y restricciones de seguridad relacionados con el uso de Ajax. Hay que tener en cuenta que por motivos de seguridad no todos los procesos se pueden realizar del lado del cliente (que por su propia naturaleza es “manipulable”). También existen restricciones de seguridad para impedir la carga de contenidos mediante Ajax desde sitios de terceras partes.

En resumen, Ajax tiene grandes ventajas pero debe usarse teniendo en cuenta criterios de diseño, adecuada estructuración de archivos y contenidos, facilidad de depuración y ampliación del código, posicionamiento en buscadores, etc. Usado sin un criterio adecuado Ajax puede resultar negativo para una página web.

EJERCICIO

Considera un proceso de compra en una página web que consta de 7 pasos, siendo el tamaño de la web completa de aproximadamente 350 Kb:

a) Sabiendo que 1 Mb son 1024 Kb, calcula cuántos Mb habremos transferido al final del proceso suponiendo que la transferencia de datos es despreciable frente a la carga completa de la página y que cada paso del proceso requiere la recarga completa de la página.

b) Calcula cuántos Mb habremos transferido si usamos Ajax de modo que sólo se carga la página una única vez y cada envío de datos supone transferir 1 Kb y cada recepción de datos supone transferir 6 Kb.

c) Calcula el tiempo requerido para transferir datos en cada caso, suponiendo que transferir 256 Kb requiere 1 segundo y despreciando otros posibles factores intervinientes.

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU01205F

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=83&Itemid=212