

AJAX: Asynchronously Moving Forward

Table of contents

So what is AJAX anyway?	3
How does AJAX work?	4
Why use AJAX?	6
The Benefits of AJAX	6
The Disadvantages of AJAX	6
Is AJAX right for my project?	8
Where not to use AJAX	8
Who's using AJAX?.....	9

The domain where desktop applications beat web applications hands-down is user interaction. In desktop applications, you do not have to wait and stare at a blank screen (most of the time) until an action completes. The user interface is integrated with the actual application and allows instantaneous response. In classic web applications, when some information is submitted or requested you have to wait for an entire new page to load.

The key word that might just change all of that and bring desktop response times to the web is AJAX. This is THE buzz word on the Internet nowadays, appearing everywhere, from [blogs](#) to news and [community sites](#).

Some voices say it's a good thing that will revolutionize the web, while others argue that it is already a dead technology that will never yield anything useful. To truly understand what AJAX is all about, you need to spend some time and browse through a hand-full of articles.

Without covering in detail every little aspect of AJAX, this article will try to shed some light on what AJAX really is, where you should use it and where you shouldn't. No special prerequisite knowledge is required in order to follow this article, but some notions of HTML, CSS and JavaScript will come in handy.

So what is AJAX anyway?

Your first guess is that AJAX is a new technology used in web development. This statement is encouraged by the endless news articles showing sites made with AJAX. In fact AJAX is neither new, nor a technology. It can be seen as a mix of several popular technologies put together to build a friendlier web.

The name itself is an acronym for **A**synchronous **J**JavaScript and **X**ML, which already contains two technologies and a technique for loading information. The name was coined by Jesse James Garrett in [this essay](#) from the Adaptive Path website.

In this mix, XML is used to store and move information between the client and server, HTML and CSS to format the output and JavaScript to bind all of these elements together in a dynamic manner.

The asynchronous part comes from the fact that when a change has to be made to the current page (the user clicks on a link, or submits some information), only the part to change is transferred from the server, and not the entire page.

Besides these, AJAX is about much more:

- The way information is displayed in a browser – using HTML/XHTML and CSS. To present information to the client, you still need to use the language that browsers understand: HTML for content and CSS for layout and formatting.
- In the age of dynamic websites, displaying content is not enough. You have to first retrieve it from a dynamic data source, and process it for final output. This is where XML and XSLT come into play.
Tip: Not familiar with XML and XSLT? Read more [here!](#)
- Getting the information from the server and in the user's browser is the next step. The XMLHttpRequest JavaScript object is used for this part. This JavaScript object will be covered in detail later on this article.

- Once the information is displayed you need to modify it dynamically. For this you will use the Document Object Model (or DOM, for short). To provide dynamics to a static HTML page, DOM manipulation is used to alter the properties of already created page elements. It is done through JavaScript and allows (for example) a div element to become visible, or a button to gray-out when a certain selection has been made.
Tip: To learn more about the Document Object Model check out the [w3c article](#).
- The glue that holds all these elements together is JavaScript. This allows retrieving dynamic information, displaying and modifying the page elements. JavaScript is a lightweight programming language used to perform various tasks on the web client.

The notion AJAX has come to define a development framework which aims to improve web page usability.

As you can see from the list above, none of the technologies used within AJAX are new. What is new is the way these technologies are used together to create new types of web applications that are much more responsive and intuitive than the existing ones. To illustrate this, just have a look at [Gmail](#) and [Google Maps](#). The user interfaces are quick in providing the user with the desired information, while performing the minimum of page reloads possible.

This new approach to user interaction and application responsiveness is part of a wider technology trend, aimed at making the Internet more friendly and faster – Web 2.0.

How does AJAX work?

When an application uses AJAX, a new layer is added to the communication model. In the classic web application, communication between the client (the browser) and the web server were performed directly, using HTTP requests.

Tip: HTTP is the acronym for HyperText Transfer Protocol, the communication protocol used throughout the Internet.

When the visitor requests a page, the server will send the full HTML and CSS code at once. After the visitor fills in a form and submits it, the server processes the information and rebuilds the page. It then sends the full page back to the client. And so on.

When using AJAX, the page is loaded entirely only once, the first time it is requested. Besides the HTML and CSS code that make up the page, some JavaScript files are also downloaded: the AJAX engine. All requests for data to the sever will then be sent as JavaScript calls to this engine. The AJAX engine then requests information from the web server asynchronously. Thus, only small page bits are requested and sent to the browser, as they are needed by the user. The engine then displays the information without reloading the entire page. This leads to a much more responsive interface, because only the necessary information is passed between the client and server, not the whole page. This produces the feeling that information is displayed immediately, which brings web applications closer to their desktop relatives.

To better illustrate the communication between the client (browser) and the remote server, as well as the differences between the classic and the AJAX-powered applications, take a look at the diagram below:

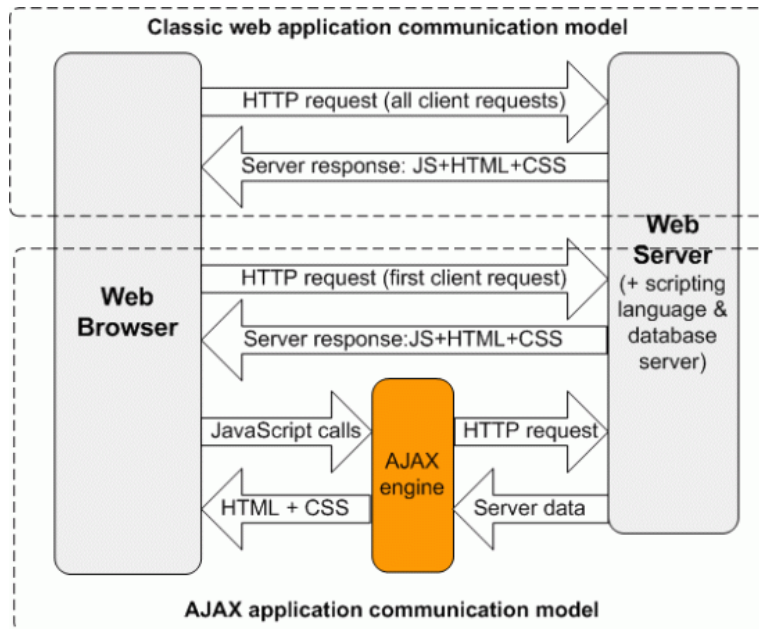


Illustration 1: Classic and AJAX communication models

At the heart of the AJAX method of communicating with the server lies the AJAX engine. This is nothing more than some JavaScript code that instantiates and uses the XMLHttpRequest object. This is a JavaScript object that allows sending, receiving and processing HTTP requests to and from the server without refreshing the entire page. In AJAX-powered applications, HTTP requests for data can be made completely in the background, without the user experiencing any interruptions. This means the user can continue working and using the application, while the necessary page sections are received from the server. The XMLHttpRequest object was implemented as an ActiveX object in Internet Explorer, and has later become a native JavaScript object in most modern browsers (FireFox, Safari).

Although adding an extra layer to any kind of model should add to the response time, this is an exception. Through the use of this new layer – the AJAX engine – response time shortens and the user interface seems much more connected to the application logic. Moreover, the user no longer has to wait around for the page to load.

If you want to get a grip on AJAX development and learn to manipulate the XMLHttpRequest object, check out this tutorial from the [Mouse Whisperer website](#).

Why use AJAX?

Building a web application that employs AJAX is more difficult than building it the classic way. You need to go to extra lengths to make sure that the user experience is and remains enjoyable, and not plain annoying. With all the hype going around AJAX, a lot of applications that do not have a real need for it have been built. These are toys meant to prove that AJAX is fun, but with little or no use in real-life situations. At the same time, some applications that do not use AJAX are labeled as such. You can find a list of AJAX applications

by using a search engine, or directly [here](#).

The decision whether an application would benefit from using AJAX or not is entirely up to the developer. Here are a few hints to help you take the right decision.

The Benefits of AJAX

To help you decide whether AJAX is for you or not, here are some of the advantages it has over classic web development techniques:

1. The interface is much more responsive, as explained before, because only a small part of the page is transferred at a time. The user has the feeling that changes are instantaneous.
2. In a classic web application, when the web server sends a web page to the browser, it can use multiple connection threads to speed up delivery. However, this happens for content only – what is between the `<body>` tags. All script and CSS files linked in the page's `<head>` section are transferred using only one connection thread, which diminishes performance. With AJAX, you only have to load the basic scripts and CSS files, and request the rest as content, through multiple connections.
3. Waiting time is reduced – when the visitor submits a form, they no longer have to wait for the entire page to be rebuilt and re-transmitted by the server. Instead, only the relevant content changes, and in non-critical cases, the visitor can still work while the data is being submitted.
4. If a page section encounters an error, other sections are not affected (if not logically linked) and the data already entered by the user is not lost. This way, the application fails gracefully, without causing head-aches for the users.
5. Traffic to and from the server is reduced considerably – because you do not have to send the entire page content (CSS, images, static sections), the bandwidth usage is most likely to decrease.

The Disadvantages of AJAX

As with most new web development techniques, AJAX has its critics. There are a couple of disadvantages that need to be considered before considering an AJAX-based solution:

1. Building an AJAX-powered application can increase development time and costs.

It is usually considered more difficult than building a classic web application, because of the mixture of technologies and the special concern about everything going smoothly. However, because it is dealing with relatively known technologies, AJAX is not [rocket science](#).

2. Although AJAX relies on existing and mature technologies like Javascript, HTML and CSS, the available frameworks and components still need to be completely mature. Tools like the [Dojo toolkit](#) or the [Rico framework](#) are just a milestone on this long road. More frameworks that target a specific area are likely to come your way. At InterAKT Online, we're working on an AJAX-based e-commerce platform to be announced soon. We also released [a free JavaScript editor](#) for the Eclipse platform, that you can [download](#) from our site.
3. Not all concerns regarding security and user privacy have been answered. This fueled a wave of criticism about how safe is the AJAX way of building applications. Earle Castledine points out some of his concerns regarding user privacy in [this article](#) from DevX.com.
4. Using AJAX to asynchronously load bits of content into an existing page conflicts with the way we are used to navigate and create bookmarks in modern browsers. Because viewing some part of the page information no longer corresponds to a newly loaded page, the browser history and bookmarks will not be able to restore the exact page state. Also, clicking the Back button in a browser might not have any effect, since the URL was unchanged (even if parts of the page were changed). To overcome these problems you must implement a way to save the current page state so that it can be restored later on, when called from a bookmark or from the browser history.
5. The biggest concern with AJAX is accessibility. This is because not all browsers (especially older ones) have complete support for JavaScript or the XMLHttpRequest object. Some of the visitors do have browsers with the required features, but they choose or have to turn off JavaScript support. When you design the application you must make sure that a fail-safe solution exists for those users, so it can be accessed by anyone. Further more, the way to access and use the XMLHttpRequest object in Internet Explorer and other browsers is different, which leads to a fork in the code and the need to treat each case separately.
6. The last disadvantage lies in the actual XMLHttpRequest object itself. Due to security constraints, you can only use it to access information from the host that served the initial page. If you need to display information from another server, it is not possible within the AJAX paradigm.

Given these advantages and disadvantages of AJAX, you have to decide whether you need to use it or not before you start the design and implementation of your next project.

Is AJAX right for my project?

AJAX can be used to improve most parts of a web application. Before investing resources in AJAX development, take a moment and consider if this is the best solution for your clients – the visitors.

You can easily tell that you need AJAX if:

- The application you are building involves heavy server requests, with multiple web forms that submit data to the server. Corporate intranets with planning and meeting programming pages that use HTML forms in multiple steps benefit from this feature, as well as sites that allow users enter information and undergo heavy traffic.
- You want to display large amounts of data in a sequential manner without refreshing the entire page. In this category enter professional photo galleries, portal websites that aggregate data from a multitude of sources, in a variety of formats (news, blog feeds, directories, image feeds from a variety of sources).
- Users interact frequently with the application through forms to decide what to display, rather than submitting data.
- Application response time is a concern.
- Loading times have to be as short as possible.

Where not to use AJAX

Although you could use AJAX in almost any type of project, there are some situations where AJAX is neither necessary, nor useful:

- Plain HTML, static sites.
- The application must be accessed by older browsers or a very wide range of browser versions, and you cannot build an alternative system. Applications that are destined to be viewed by a large number of visitors, belonging to different categories in terms of browsers and platforms, sites that will be accessed by people with disabilities that use screen readers or other plain-text browsers. Users do not interact with the application on a frequent basis. Most presentation sites for small companies, public institutions, content repositories fall under this section. Loading times and bandwidth usage is not a primary concern. In this category fall sites that rely mostly on plain text content or do not have many graphics. The application does not vary content very much, and dynamic loading of the information would not provide a significant improvement in the navigation experience.

When the cost and time factors are paramount for the delivery of your project and your development team cannot afford to build a complex AJAX framework. Of course, the final decision is up to you. In the next section, you will learn about some of the uses of AJAX in real-life applications.

Who's using AJAX?

As [this survey](#) shows, AJAX is already widely used in web development. However, its main use is in the development phase, not in the production phase of projects.

One of the most important players on the AJAX scene is Google. With Gmail, Google Groups, Google Suggest or Google Maps, AJAX is being put to good use by millions of visitors every day. The user interfaces respond quickly and are pretty intuitive, leaving no time for users to admire a blank screen.

AJAX was used in [Meebo](#) to allow users to connect to multiple instant messaging networks from a web-based application, and in [Flickr](#), to allow photo sharing and tagging. Following this path, InterAKT Online is also committed to developing intelligent, browser-based applications that run like in a desktop environment. [KTML](#) is an online visual HTML editor that emulates the familiar Microsoft Word. You can use it for web content authoring just as you would use any other desktop-based text editor.

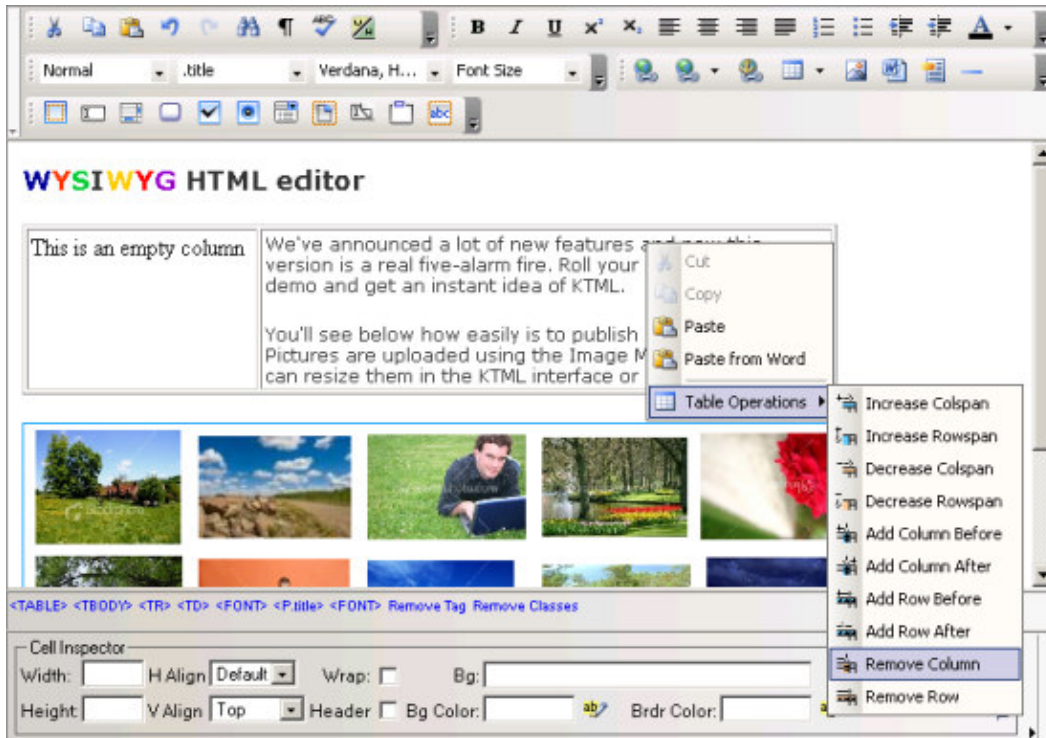


Illustration 2: KHTML - the online HTML editor

[MX Query Builder](#) is a visual tool for creating and managing SQL queries. It runs in a browser window, but can be launched from Macromedia Dreamweaver. Its many panels allow asynchronous interaction with your database:

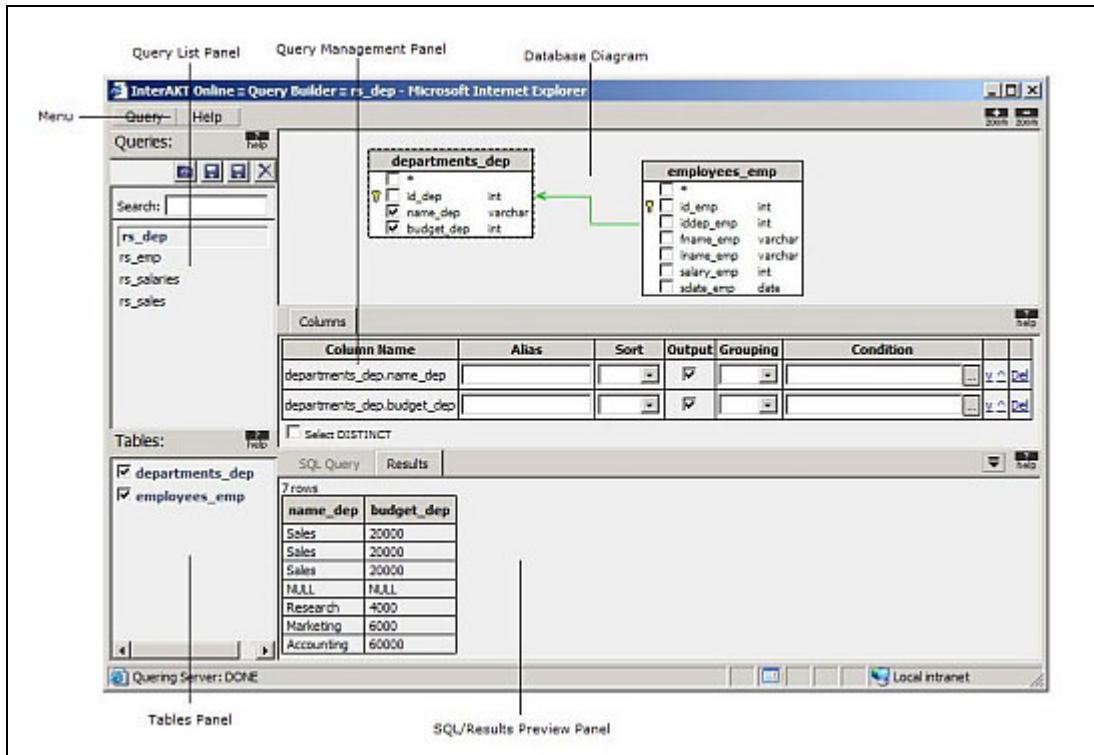


Illustration 3: MX Query Builder - manage queries efficiently

When you're done, you can save your query and use it in Dreamweaver as a regular recordset.

You can find a lot more AJAX-enabled web applications on the [AJAX Review website](#). The AJAX approach to building web applications sheds a new light on the web and helps close the gap between web applications and desktop applications in terms of usability and responsiveness. As the technique approaches maturity, many benefits and pitfalls will be uncovered, but the overall principles promise to revolutionize the web as we know it. To see what else lies ahead, check out this article on Web 2.0 trends.[\[link:line article\]](#) If you wish to get a deeper understanding of AJAX, you can also have a look at some of these articles:

- [“Ajax: A New Approach to Web Applications”](#) from the Adaptive Path website. A [step-by-step tutorial](#) on building your first AJAX application.

You can [speak your mind](#) about AJAX and your vision on it on our forums.