ISSN 1407-7493 COMPUTER SCIENCE DATORZINĀTNE 2009-7493

APPLIED COMPUTER SYSTEMS
LIETIŠĶĀS DATORSISTĒMAS

ANALYSIS OF ASP.NET AJAX ARCHITECTURE

ASP.NET AJAX ARHITEKTŪRAS ANALĪZE

Andrey Lesovsky, Mg. sc. ing., assistant, doctoral student, Riga Technical University, Meza 1/3, Riga, LV 1048, Latvia, phone: +371 28841619, andreyl@inbox.lv

ASP.NET AJAX, framework, architecture, client, server

1. Introduction

Ajax (Asynchronous JavaScript and XML), or AJAX, is a group of interrelated web development techniques used for creating interactive web applications or rich Internet applications. With AJAX, web applications can retrieve data from the server asynchronously in the background without interfering with the display and behavior of the existing page [1].
Web 2.0 is the move toward a more social, collaborative, interactive and responsive web. It is a change in the philosophy of web companies and web developers, but more than that, Web 2.0 is a change in the philosophy of a society as a whole. Web 2.0 marks a change in us as a society as well as the Internet as a technology. In the early days of the web, we used it as a tool. Today, we aren't just using the Internet as a tool - we are becoming a part of it.
Web 2.0 is about creating a social web, it is also about creating a more interactive and responsive web. It is in this way that methodologies such as AJAX become central to the idea of Web 2.0. AJAX technology has become one of the key technologies for such Web 2.0 concepts realization as rich user experience, user participation, dynamic content, metadata and scalability. Providing a developer with opportunities to create rich and user friendly interface, AJAX makes it possible to improve Internet users experience without requiring them to install any additional software.
An AJAX website is a website built using the AJAX methodology that allows it to pass information back and forth seamlessly. This means the website can be more dynamic. Because it can load up just pieces of a page instead of the entire page, it can transform a large and confusing website into a simple application.
To realize AJAX true potential and to use AJAX in the development effectively, it is important to understand how it is designed. The aim of this paper is to provide a detailed analysis of AJAX and ASP.NET AJAX architectures and to analyze development models available to ASP.NET AJAX developers.

2. ASP.NET AJAX architecture

Many consider AJAX to be a new technology, but it is wrong as AJAX is an effective combination of the known technologies. AJAX combines 4 technologies which are the following:

- JavaScript – is a scripting language most often used for client-side web development and is supported by all modern browsers.
- Document Object Model (DOM) – is a platform- and language-independent standard object model for representing HTML or XML and related formats.
- Cascading Style Sheets (*CSS*) – is a stylesheet language used to describe the presentation of a document written in a markup language. It's most common application is to style web pages written in HTML and XHTML.

- XMLHttpRequest – is an API that can be used by JavaScript and other web browser scripting languages to transfer XML and other text data between a web server and a browser.

Figure 1 illustrates how these technologies interact with one another from the browser.

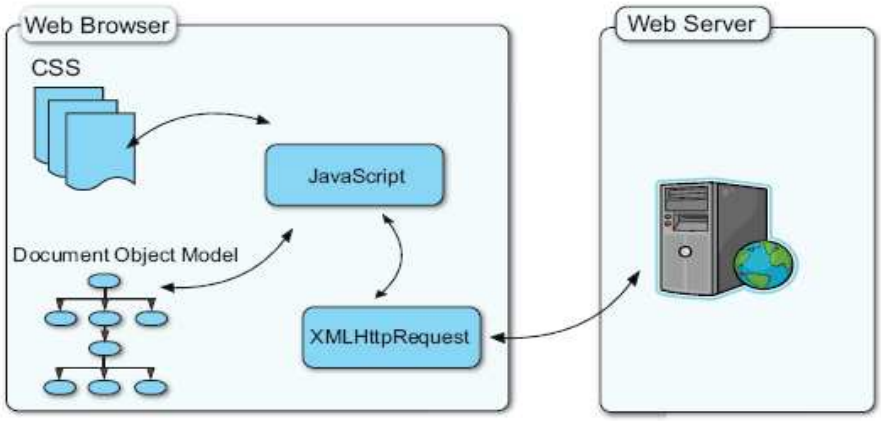

Fig.1. AJAX components interaction

ASP.NET AJAX, formerly code-named Atlas, is a set of extensions to ASP.NET developed by Microsoft for implementing Ajax functionality. Including both client-side and server-side components, ASP.NET AJAX allows the developer to create web applications in ASP.NET which can update data on a web page without a complete reload of the page. The key technology which enables this functionality is the XMLHttpRequest object, along with JavaScript and DHTML.
ASP.NET AJAX was released as a standalone extension to ASP.NET in January 2007 after a lengthy period of beta-testing. It was subsequently included with version 3.5 of the .NET Framework, which was released alongside Visual Studio 2008 in November 2007. This led to the improvement of its position as web application development technology.

2.1. Asynchronous programming

The A in Ajax stands for asynchronous and this is a key behavior in the Ajax programming pattern. Asynchronous means not synchronized or not occurring at the same time.
Normally, a user action such as clicking a button on a form invokes an HTTP request back to the web server. The server then processes the request, possibly doing some calculations or performing a few database operations; and then returns back to the client a whole new page to render [2]. The server goes through the entire page lifecycle again and returns to the browser the HTML, CSS and any other resources it needs to refresh the page. But this technique doesn't present the user with a desirable interactivity and they're exposed to a stop-start-stop pattern where they temporarily and unwillingly lose interaction with the page and are left waiting for it to be updated. Figure 2 illustrates the flow of a traditional web application in a synchronous manner.

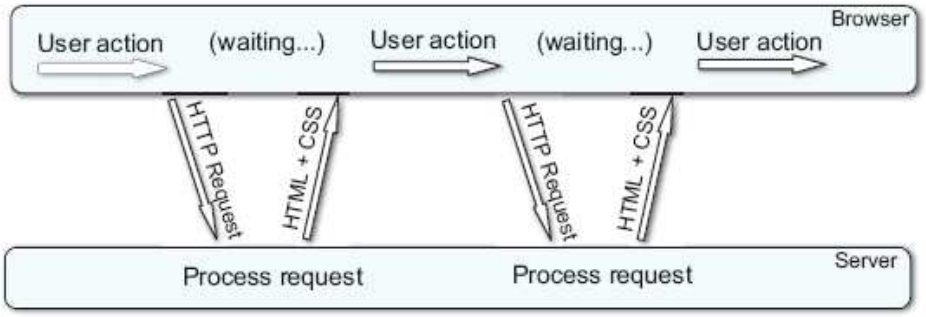

Fig. 2. Traditional web applications behave in a synchronous manner

An Ajax-enabled application works differently with the introduction of an Ajax agent placed between a client and a server. This agent communicates with the server asynchronously, on behalf of the client, to make the HTTP request to the server and return the data needed to update the contents of the page [2]. Figure 3 demonstrates this asynchronous model.

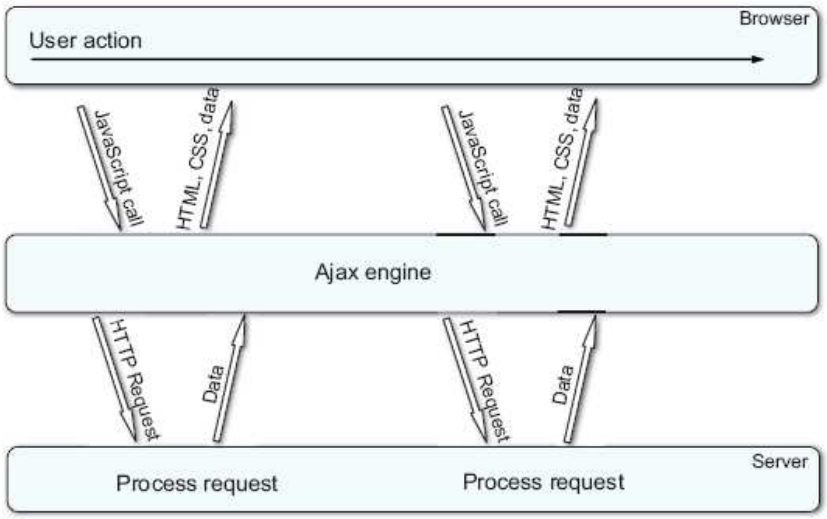

Fig. 3. The asynchronous web application model leverages an Ajax engine to make an HTTP request to the server

2.2. XMLHttpRequest object

The *XMLHttpRequest* object enables JavaScript to make requests to the server and process the responses. It was delivered in the form of an ActiveX object when released in Internet Explorer 5, and it's supported by most current browsers. Other browsers (such as Firefox, Opera and Safari) deliver the same functionality in the form of a native JavaScript object. Internet Explorer 7 now implements the object in native JavaScript as well, although differences between browsers remain. The fact that there are different implementations of the object based on browsers and their versions require you to write browser-sensitive code when instantiating it from script. Listing 1 demonstrates a technique called object detection to determine which XMLHttpRequest object is available:

```
function getXMLHTTP( )
      {
      var XMLHTTP = null;

      try
            {
            XMLHTTP = new ActiveXObject("Msxml2.XMLHTTP");
            }
      catch (e)
            {
            try
                  {
                  XMLHTTP = new ActiveXObject("Microsoft.XMLHTTP");
                  }
            catch (e)
                  {
                  if (typeof XMLHttpRequest != "undefined")
                        {
                        XMLHTTP = new XMLHttpRequest( );
                        }
                  }
```

```
        }

return XMLHTTP;
}                                                                    (1)
```

The XMLHttpRequest object, no matter which browser created it, always has a set of properties and methods that are used for sending HTTP requests and receiving the server's response. In most cases you must follow these four steps to create an HTTP request and evaluate the return values:

1. Create an XMLHttpRequest object as shown in the preceding examples.

2. Call the object's open( ) method to prepare the request.

The open( ) method can accept up to five parameters, but usually you only need the first two: the method type of the request (usually "GET" or "POST"), and the target URL (relative or absolute).

The third parameter of open( ) defaults to true, meaning that the request is an asynchronous one. If you set it to false, the request is synchronous, meaning that the script halts until the response has been completed. Generally, you want the asynchronous behavior, so you either omit the parameter or set it to true. If the HTTP request requires authentication, you can use the fourth and fifth parameter to provide a username and a password.

3. Provide a reference to a callback function in the onreadystatechange property.

This function will be called when the server returns an HTTP response to the HTTP request.

4. Send the HTTP request using the send( ) method.

This starts the HTTP request. If you are using asynchronous communication, the script continues executing and the user can continue to interact with the page [3].

Setting the onreadystatechange property of the XMLHttpRequest object provides the callback mechanism for the HTTP response. The property name suggests its function, which is to specify the action to be taken when a change occurs in the value of another XMLHttpRequest property, readyState, as listed in Table 1. The readyState property indicates the state of the XMLHttpRequest object, which can be set to five possible values.

Table 1

Possible values for readyState

readyState value	Description
0	Object is uninitialized
1	Request is loading
2	Request is fully loaded
3	Request is waiting for user interaction
4	Request is complete

When the value of readyState changes, the function provided in the onreadystatechange property is called. In this function, you first check the value of readyState. Usually, developers are determining whether the value is 4, which indicates that the request has been returned.

When a function is called in response to a change in readyState, some other properties of the XMLHttpRequest object come into play. The status property contains the HTTP status returned by the request; if everything worked, the status is 200. The statusText property holds the associated textual description of the HTTP status. For HTTP status 200, the value of statusText is "OK". Checking the status property, however, is a more reliable method, because different web servers might return different text for the status codes.

Two properties provide access to the return value from the server:

1. responseText – returns the response data as a string;
2. responseXML – returns the response data as an XML document.

2.3. ASP.NET AJAX architecture

The ASP.NET AJAX framework enables developers to create rich, interactive and highly personalized web applications that are cross-browser compliant. It is important to point out what even though the framework is primarily an Ajax library it also offers many other features that can increase the productivity and quality of your web applications.
As figure 4 demonstrates the architecture of the ASP.NET AJAX framework spans both the client and server. In addition to a set of clientside libraries and components, there are also a lot of ASP.NET server controls and services.

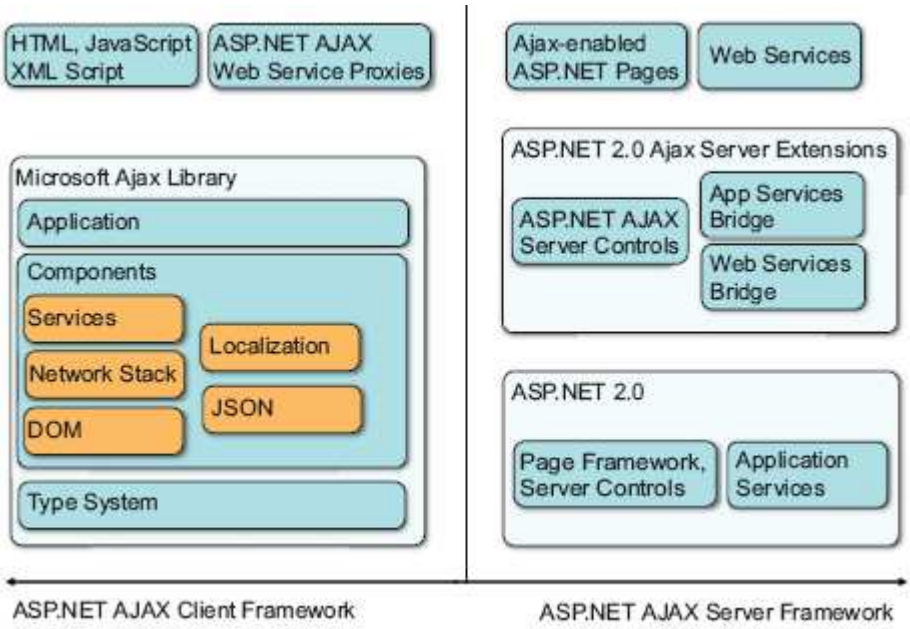

Fig. 4. ASP.NET AJAX architecture

2.4. Client framework

Client framework's core library isn't reliant on the server components. The core library can be used to develop applications built in Cold Fusion, PHP, and other languages and platforms. With this flexibility, the architecture can be divided logically into two pieces: the client framework and the server framework. Understanding how things work in the client framework is essential even for server-side developers, because this portion brings web pages to life. At the core is the *Microsoft Ajax Library*.
Microsoft Ajax Library is also known as the core library. The library consists of a set of JavaScript files that can be used independently from the server features. The library has the following layers:

1. Type system layer

Type system introduces well known object-oriented programming concepts to JavaScript – such as classes, inheritance, interfaces, and event-handling. This layer also extends existing JavaScript types. The type system lays the groundwork for the rest of the Ajax core library.

2. Components layer

This layer provides support for JSON serialization, network communication, localization, DOM interaction and ASP.NET application services like authentication and profiles. It also introduces the notion of building reusable modules that can be categorized as controls and behaviors on a page.

3. Application layer

Application layer or as it is also called application model is similar to the page lifecycle in ASP.NET. This layer provides an event-driven programming model that developers can use to work with DOM elements, components, and the lifecycle of an application in the browser [4].
ASP.NET AJAX-enabled web pages are written in HTML, JavaScript and a new XMLbased, declarative syntax called XML Script. This provides developers with more than one option for

authoring a client-side code – they can code declaratively with XML Script and imperatively with JavaScript. Elements declared in XML Script are contained in a new script tag:

<script type="text/xml-script">.

The browser can detect the script tag but don't have a mechanism for processing the xml-script type. Instead, the JavaScript files from the ASP.NET AJAX framework can parse the script and create an instance of components and controls on the page.

The client framework offers the ability to call web services from JavaScript via a set of *client-side proxies* that are generated from the server. These proxies can be leveraged much like a web reference in managed .NET code.

2.5. Server framework

Server framework is a valuable set of controls and services that extend the existing framework with Ajax support [4]. This tier of the server framework is called the ASP.NET AJAX server extensions. The server extensions are broken into three areas: server controls, the web services bridge, and the application services bridge. Each of these components interacts closely with the application model on the client to improve the interactivity of existing ASP.NET pages.

Server controls set is predominantly driven by two controls. The first of these controls is the *ScriptManager*, which is considered the brains of an Ajax-enabled page. One of the many responsibilities of the ScriptManager is to manage the regions on the page that are dynamically updated during asynchronous postbacks. The second control, named the *UpdatePanel*, is used to define the regions on the page that are designated for partial updates. These two controls work together to enhance the user experience by replacing traditional postbacks with asynchronous postbacks. This results in regions of the page being updated incrementally rather than all at once with a full page refresh.

Usually web applications are limited to resources on their local servers. Aside from a few external resources, like images and CSS files, applications are not granted access to resources that are not in the scope of the client application. To overcome these hurdles the server extensions in the ASP.NET AJAX framework include a *web services bridge* that creates a gateway for you to call external web services from client-side script. This type of technology can be used to aggregate or consume data from third-party services.

Because ASP.NET AJAX is so tightly integrated with ASP.NET, access to some of the application services like authentication and profile can be added to an existing application almost effortlessly. This feature enables tasks like verifying a user's credentials and accessing their profile information to originate from the client script.

3. Development models

3.1. Client-centric development model

The flexible design of the ASP.NET AJAX architecture provides two development scenarios. The first scenario is primarily implemented on the client side and is known as the client-centric development model. The second is developed mainly on the server side and is identified as the server-centric development model. In the client-centric model, the presentation tier is driven from the client-script using DHTML and JavaScript. This means that more interactive application is delivered from the server to the browser when the page is first loaded. Afterward, interaction between the browser application and the server is limited to retrieving the relevant data necessary to update the page. This model encourages a lot more interactivity between a user and a browser application, resulting in a richer and more intuitive experience. Figure 5 illustrates the client-centric development model.

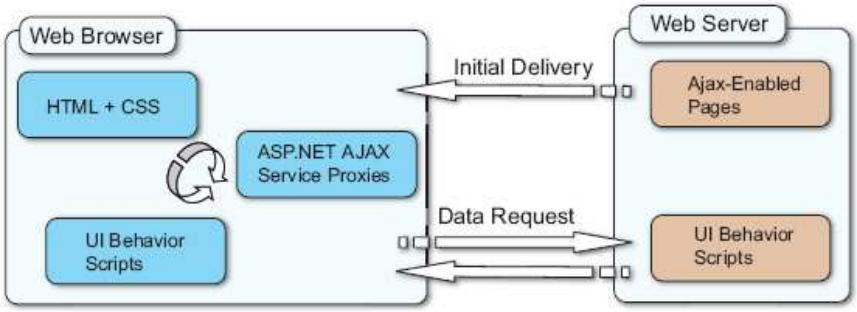

Fig. 5. The client-centric development model

The client-centric model is also ideal for mashups and applications that wish to fully exploit all the features DHTML has to offer.
Mashups examples are the following:

- **Flash Earth** (http://www.flashearth.com/) – is the site that lets you enter a location and compare the imagery from Google Maps, Yahoo Maps, Windows Live Local, Ask.com Maps and more;
- **Retrievr** (http://labs.systemone.at/retrievr/) – retrievr is an experimental service which lets you search and explore in a selection of Flickr (http://flickr.com) images by drawing a rough sketch [5].

3.2. Server-centric development model

In the server-centric mode, the application logic and most of the UI rationale remain on the server. Incremental changes for the UI are passed down to the browser application instead of the changes being made from the client-side script. This approach resembles the traditional ASP.NET page model where the server renders the UI on each postback and sends back down to the browser a new page to render. The difference between this model and the traditional model in ASP.NET is that only the portions of the UI that need to be rendered are passed down to the browser application, rather than the whole page. As a result interactivity and latency are both improved significantly. Figure 6 illustrates the nature of the server-centric development model.

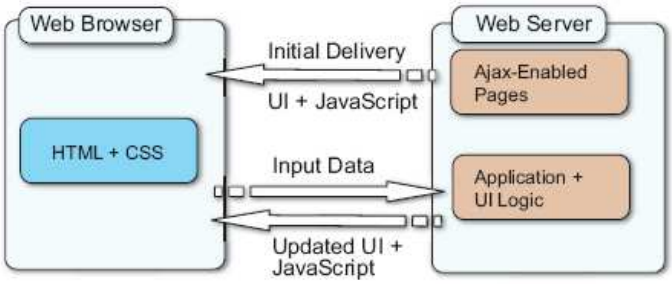

Fig. 6. Server-centric development model

This approach appeals to many ASP.NET developers because it grants them the ability to keep the core UI and application logic on the server. It's also has an ability to behave as a normal application if the user disables JavaScript in the browser. If a developer uses controls like the GridView and Repeater in ASP.NET, the server-centric model offers the simplest and most reliable solution.

4. Conclusions

After examining the architecture and features of the framework we can see that the main objective is to simplify the efforts of adding Ajax functionality to web applications. This is accomplished by

providing a rich client library and a comprehensive set of server controls that are easy to use and integrate into the existing applications.
Support for Internet Explorer, Firefox, Safari and Opera extracts away the hassle of dealing with browser differences and discrepancies. New XMLHttpRequest creation is an exception, although this problem can be easily solved with the provided solution.
Server controls provide ASP.NET developers a familiar paradigm for developing web applications. Server controls emit the JavaScript needed to Ajax-enable a page with little effort or knowledge of JavaScript and the XMLHttpRequest object.
Components and tools built on top of the framework not only extend the framework but also provide the development community with a rich collection of tools to leverage and build on. This includes tools for debugging, tracing, and profiling.
The effective use of all provided technologies makes ASP.NET AJAX a perfect technology for a development of any type of web applications, whether it is a small personal blog or enterprise portal. Besides a support from powerful companies such as Google and Microsoft allows a developer to be assured that this technology will stay on the market for a while.

References

1. What is Ajax? [Electronic resource] / RIAspot.com, 2008. – http://www.riaspot.com/articles/entry/What-is-Ajax-. – 03.10.2008.
2. Gallo A., Barkoll D., Vavilala R. K. ASP.NET AJAX in action. – Greenwich: Manning, 2008. – P. 7-12.
3. Wenz C. Programming ASP.NET AJAX. O'Reilly, 2007. – 475 p.
4. J. Kanjilal, S. Putrevu. Teach Yourself ASP.NET Ajax in 24 Hours. Indiana: Sams, 2008. – P. 28-34.
5. About retrievr [Electronic resource] / Christian Langreiter, 2008. – http://labs.systemone.at/retrievr/about. – 03.10.2008.

Lesovskis A. ASP.NET AJAX arhitektūras analīze
Ajax (Asynchronous JavaScript and XML), jeb AJAX, ir savstarpēji saistīto tīmekļa izstrādes tehnoloģiju apvienojums, kuru izmanto interaktīvo tīmekļa lietojumprogrammu un bagāto Internet lietotņu izveidošanai. AJAX piedāvā izstrādātājiem iespējas izveidot interaktīvu un draudzīgu lietotāja interfeisu un palīdz atvieglot lietotājiem tīmekļa resursu izmantošanu bez nepieciešamības instalēt papildus programmatūru. ASP.NET AJAX, agrāk zināms ar nosaukumu Atlas, ir Microsoft izstrādātā ASP.NET papildinājumu kopa, kas ļauj izmantot ASP.NET iespējas AJAX programmās. ASP.NET AJAX piedāvā izstrādātājiem iespēju izmantot klientcentrisku vai servercentrisku izstrādes modeli, kas ļauj noteikt, kur atradīsies un izpildīsies lielāka programmatūras koda daļa. Tas ļauj izvēlēties izstrādes pieeju, kas ir piemērotāka konkrētam uzdevumam un var efektīvāk sadalīt noslodzi uz serveri un klientu un optimizēt tīkla izmantošanu. Šī raksta mērķis ir izpētīt AJAX un ASP.NET AJAX arhitektūras. Tiek analizētas tādas tēmas kā AJAX un ASP.NET AJAX arhitektūras, klienta un servera ietvari, klientcentrisks un servercentrisks izstrādes modeli.

Lesovsky A. Analysis of ASP.NET AJAX architecture
Ajax (Asynchronous JavaScript and XML), or AJAX, is a group of interrelated web development techniques used for creating interactive web applications or rich Internet applications. Providing developer with opportunities to create rich and user friendly interface, AJAX makes it possible to improve Internet users experience without requiring them to install any additional software. ASP.NET AJAX, formerly code-named Atlas, is a set of extensions to ASP.NET developed by Microsoft for implementing Ajax functionality. ASP.NET AJAX provides developers with two development scenarios: client-centric development model and server-centric development model. It provides a developer with the opportunity to choose a model which is suitable for specific task and which will distribute load between client and server effectively and optimize network usage. This article discusses the architecture of AJAX and ASP.NET AJAX technologies. The covered themes are AJAX and ASP.NET AJAX architectures, client and server frameworks, client-centric and server-centric development models.

Лесовский А. Анализ архитектуры ASP.NET AJAX.

Ajax (Asynchronous JavaScript and XML), или AJAX, это группа взаимосвязанных технологий разработки интерактивных сетевых программ и полнофункциональных Internet-приложений. Предоставляя разработчикам возможности создавать богатые по своим возможностям и лёгкие в использовании интерфейсы, AJAX позволяет облегчить использование Internet ресурсов пользователям без необходимости устанавливать дополнительную программатуру на своих системах. ASP.NET AJAX, ранее известное как Atlus, это разработанное Microsoft дополнение к ASP.NET, которое позволяет использовать возможности ASP.NET в AJAX программах. ASP.NET AJAX предоставляет разработчикам две модели разработки приложений: клиенто-центричная модель и серверо-центричная модель. Это позволяет выбрать наиболее подходящую для конкретного задания модель и таким образом эффективнее распределить нагрузку между клиентом и сервером и оптимизировать использование сетевого трафика. В данной статье рассматриваются архитектуры технологий AJAX и ASP.NET AJAX. Рассмотрены такие темы как архитектуры AJAX и ASP.NET AJAX, клиентская и серверная подсистемы, клиент-центричная и серверо-центричная модели разработки.