OpenAjax alliance

Standardizing Ajax Development

OpenAjax Alliance White Paper

31 October 2006

Contents

1 Introduction

Ajax, which stands for Asynchronous JavaScript and XML, is transforming the Web and Internet-based applications forever. In response to the growing interest in Ajax, technology leaders in the Web community have joined together to promote Ajax interoperability, standards and education in order to accelerate the global adoption of Ajax. This organization is called the OpenAjax Alliance.

The initial user interaction model of the Web is based on page-oriented HyperText Markup Language (HTML). It is designed to display text and graphic information easily in a web browser. Its simple publication and viewing model has helped fuel the explosive adoption of the World Wide Web into the global phenomenon we know today. But this growth has users wanting more; they want web applications to provide desktop-like user experiences, and HTML's page-oriented model falls short.

Ajax substantially improves the end user experience for web-based applications and provides a new user interaction model. In more technical terms, Ajax is a development technique in which Web pages act more like desktop applications because the page does not have to be reloaded on each user input. Instead of repeated page refreshes, small amounts of data are exchanged with the server behind the scenes and the application remains usable by the end-user.

Ajax technology has emerged as a critical technology for launching the Web's next generation, often referred to as Web 2.0. Ajax is increasing end user productivity and enabling new classes of collaboratory applications. In short, Ajax is redefining the Web at a rapid pace, thus requiring every organization with a Web presence or Internet-based business applications to develop a cohesive Ajax strategy.

Many suppliers have entered the market with a wide range of tools, techniques and solutions for developing, deploying and managing Ajax applications. Because of the newness of this technology, questions remain regarding implementation costs, interoperability of different Ajax products, and best adoption strategies. This white paper has been developed by the OpenAjax Alliance as a guide to help IT executives evaluate Ajax, create an Ajax strategy and understand the role played by the OpenAjax Alliance.

2 What is Ajax?

Ajax is a design approach and a set of techniques for delivering a highly interactive, desktop-like user experience for Web applications in popular HTML browsers. Ajax, which stands for Asynchronous JavaScript and XML, improves the user's web application experience while retaining the HTML benefits of server-based application deployment. Ajax represents the continued evolution of DHTML to deliver Web 2.0 experiences and Rich Internet Applications - RIAs.

Ajax builds on open standards that are available widely as native features (i.e., without plugins) in popular browsers. Ajax's incremental update techniques are accomplished through built-in features such as JavaScript and the XMLHttpRequest API. Ajax developments often leverage JavaScript libraries that provide a cross-browser abstraction layer.

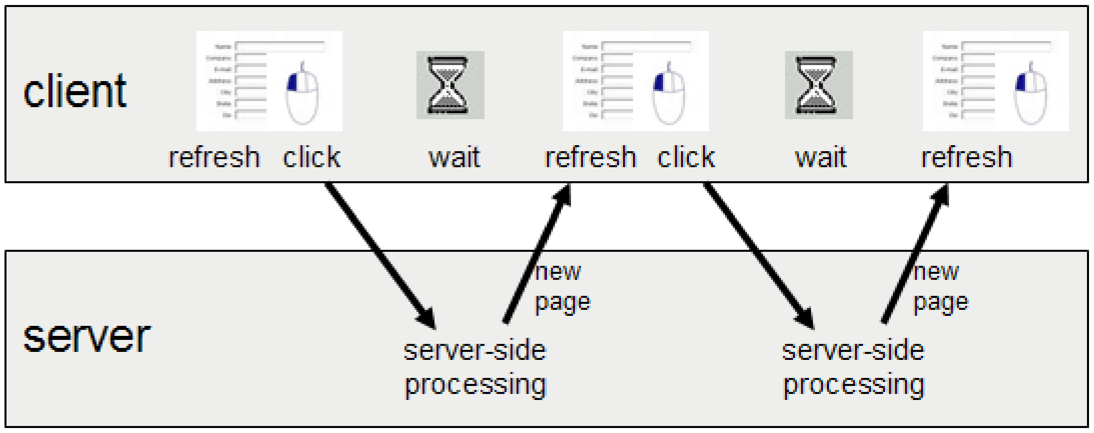
The following sections summarize the key characteristics of Ajax:

2.1 Dynamic and continuous user experiences

The primary difference between HTML applications and Ajax applications is that Ajax allows users to interact with the application while the browser is communicating with the server.

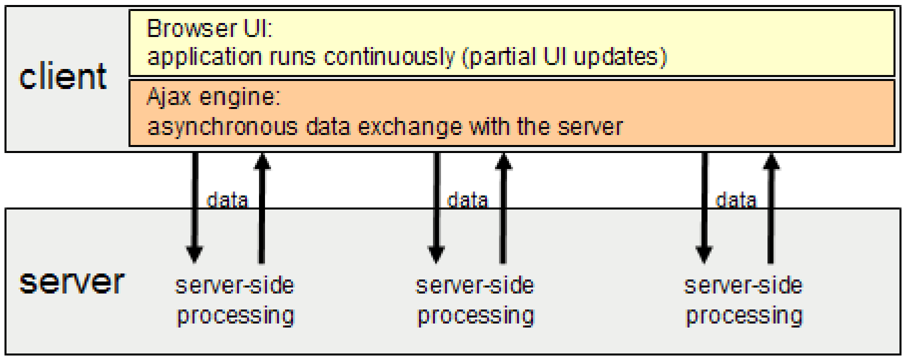
2.1.1 HTML click, wait, refresh

The following picture shows the typical "click, wait, and page refresh" user experience from typical HTML applications:



2.1.2 Ajax-powered user experiences

Ajax minimizes the number of page refreshes because the client issues data requests to the server, not page requests. The Ajax application stays up on the screen continuously while the Ajax engine handles data exchange with the server. Instead of page refreshes, the application updates only those parts of the screen affected by newly arrived data.



2.2 Desktop-like user interfaces

Ajax enables rich user interfaces traditionally only found in installable desktop software. Typical rich user interface features found in Ajax applications include:

- Standard user interface (UI) controls (e.g., buttons, input fields and combo boxes)
- Advanced UI controls (e.g., tabbed palettes, calendar widgets, tree widgets and data grid widgets)
- Flexible, dynamic layout containers that adjust to the size of the embedded content and window size
- Floating palettes and modal dialogs
- Animations and animated effects
- Cut and paste, and drag and drop

2.3 Network programming

Network programming features make it possible to exchange data asynchronously with the web server, enabling next-generation collaborative applications. Ajax runtime libraries often support the ability for servers to push data asynchronously to the client.

2.4 Evolving over time

Ajax will evolve over time as its technologies mature and as new releases of browsers respond to developers' requirements. Today, Ajax offers rich, desktop-like user experiences with powerful network programming capabilities, built on natively-implemented open standards. Going into the future, Ajax technology providers and browser vendors continually make advances that increase the power and performance available to Ajax developers.

3 Ajax Value Proposition

3.1 More productive end-users, lower bandwidth, and strong ROI

In most businesses, decision makers are interested mainly in how information technology can increase revenue, reduce costs, or make better use of information assets. Among the factors that are considered:

1. **Time spent waiting for data to be transmitted**. Over many repetitions, the time employees spend waiting for pages to load can add up to significant costs.
2. **Time spent completing a particular task**. Inefficient user interfaces can translate into long periods of lowered end-user productivity.
3. **Bandwidth consumed for the entire task**. If repetitious tasks consume considerable bandwidth, IT costs can grow dramatically.

Ajax technologies help with all of the above. Because Ajax supports dynamic and continuous user experiences, user productivity increases significantly and measurably (e.g., `seconds-saved-per-transaction x number-of-transactions-per-year`). Because Ajax provides similar advanced user interface features to those in desktop applications (e.g., advanced UI controls, animated effects and adjustable layout controls), thereby providing the visual and interaction tools needed to make the application self-explanatory, users spend less time learning and operating the application. Ajax's partial page update feature offers reduced communications versus HTML's traditional approach of click, wait, and refresh.

3.2 Next-generation applications (Web 2.0 and RIAs)

3.2.1 Replacement for desktop applications

Ajax offers a desktop-like user experience while retaining the benefits of server-based application deployment, centralized infrastructure administration, easier scalability and immediate availability of application updates to all users. As a result, Ajax is accelerating the movement away from installable, two-tier, client-server applications to multi-tier web applications.

3.2.2 The runtime companion for SOA

As the natural evolution of HTML, Ajax's platform-independent runtime technology is well-suited for next-generation service-oriented architecture (SOA) applications. Ajax offers standards compliance and platform independence on the client, while SOA offers similar benefits on the server.

3.2.3 Mashups, dashboards and other composite applications

Ajax enables mixing and matching of multiple component technologies within the same application. This enables Ajax developers to build composite applications that leverage best-of-breed Ajax technologies from multiple suppliers, achieving many different types of composite applications:

- **Mashups** - A mashup is a website or web application that uses content from more than one source to create a completely new service. One concrete example is a custom mapping application that talks to a company's own server to pull down XML address data and then leverages mapping services (e.g., Google or Yahoo) to show a map view in combination in an application-specific custom user interface. Mashups often enable rapid application development by integrating ready-made third-party components.
- **Portals and dashboards** - Portals and dashboard applications consist of multiple panes that often can be configured by the user. Often, each pane is a separate application. Ajax technologies can be used to manage the entire composite application and its integrated applications.
- **Service compositions** - In an SOA environment, composite business applications assemble multiple application services to create new functionality that supports innovative business processes. In some cases, the services from which the new application is composed have no user interface, so the composite application is completely responsible for the user interface. In other cases the services do have user interfaces, so that the composite application knits together not only back-end behavior components, but user interface components as well.

3.2.4 Collaboration

Ajax enables collaborative applications, such as:

- **Next-generation chat and instant messaging (IM)** - Ajax enables web-based messaging applications that run in the browser.
- **Web meetings and whiteboarding** - Ajax's rich user interface, network programming, and vector graphics features provide the infrastructure for browser-based web meetings and shared whiteboards.
- **Collaborative content creation** - Web-based collaborative applications such as wikis can transition from users having to learn arcane wiki markup languages to a WYSIWYG user experiences, comparable to what users experience today with desktop content creation tools.
- **Trip planning** - A social group can use an Ajax-powered application to work together to plan and schedule their activities.
- **Photo sharing** - Ajax enables richer sharing of experiences, such as attaching annotations, tagging or marking up photos.

Collaboration capabilities are based on XMLHttpRequest and other network communications technologies.

3.2.5 Ajax-powered wikis

An important new application area is the Ajax-powered wiki, where wikis go beyond text-based collaborative documents into the following scenarios:

- **Rich, lightweight portals –** Ajax-powered wikis enables fast deployment of Enterprise rich dashboards by allowing a wiki page to contain both text content and Ajax-powered user interface components, such as data grids, forms fillout, mapping, and charting.
- **Personal mashups and dashboards –** Many leading Internet companies provide the option today for customized home pages. When personalization is paired with Ajax-powered components and wiki back-end services, IT gains easy deployment and end-users gain the ability to manage their view on information.

3.2.6 Cross-device applications (desktop and mobile)

Ajax offers multiple approaches to achieving cross-device applications:

- **Use Ajax desktop technologies on high-end mobile devices** - Today, some mobile devices offer mobile web browsers that support the same feature set (HTML, JavaScript, etc.) as desktop mobile browsers. Examples include the Opera browser and Nokia's mobile browser, which is built from the same KHTML/WebKit code base as the Apple Safari browser. Web developers can reach this subset of mobile devices using the same Ajax source code as they use for desktop Ajax. Over time, as mobile devices become more powerful, increasingly larger percentages of mobile devices will ship with web browsers that offer full desktop Ajax support.
- **Use a mobile subset of Ajax** - The W3C and the Open Mobile Alliance (OMA) are working together to standardize appropriate mobile subsets of XHTML, SVG, CSS and ECMAScript to take into account the constraints of today's mobile devices, and combine them together to form a mobile standard for rich content using Ajax, WICD Mobile.
- **Use a server-side, multi-target Ajax toolkit** - To reach a large number of mobile devices, some of which ship with more limited features sets, web developers can take advantage of Ajax toolkits that provide a cross-platform abstraction layer. These toolkits adapt Ajax source into appropriate client-side instructions, such as mobile subsets of HTML+JavaScript, mobile SVG, or J2ME.

3.3 Open standards

Ajax leverages a combination of open technologies that are native to browsers. Most are official Web standards, while many of the rest have been implemented widely in browsers but have not been formally recognized by a standards body:

- **JavaScript** -- its official standards name is ECMAScript
- **HTML** -- often delivered via well-formed XHTML

- **CSS** -- for marking up and styling information
- **DOM** and **DOM Events** -- for client-side scripting APIs that interact with and control the web page
- **XMLHttpRequest** -- for client-server communications without requiring page refreshes (not yet an official standard, though under consideration by the W3C)
- **XML** -- a format for transferring data between the server and client (other formats can be used with Ajax, including preformatted HTML, plain text and JSON)
- **SVG** -- the standards-compliant vector graphics component for web pages (supported by recent releases of popular browsers; browsers that do not yet support SVG, Ajax libraries use other vector graphics support, such as VML in Internet Explorer)

Ajax builds on open standards that are widely available as native features (i.e., without plugins) in popular browsers. In most cases, Ajax restricts itself to the commonly implemented subset of particular standards (e.g., DOM2 and DOM3, which are not supported completely yet) or sometimes supports commonly implemented extensions (e.g., the innerHTML property within the DOM is implemented by most popular browsers but is not covered by W3C specifications).

3.4 Open source

While open source software is not mandatory for Ajax projects, a large part of Ajax's momentum is due to the open source community's commitment to Ajax. Today, many Ajax open source projects bring the power of community-based open development models and no-cost licensing models to developers. Here are some of today's Ajax open source projects:

ActiveMQ
Ajax Anywhere
Ajax4JSF
ajaxCFC
AjaxTags
Ajax Toolkit Framework
Dojo
DWR
Echo2
Google Web Toolkit
jMaki

JSON-RPC-JAVA
JsOrb
JSP Controls
JWAX
MochiKit
OAT
OpenLaszlo
Plex Toolkit
Prototype
Rialto
Rico
Rich Ajax Platform | RAP

SAJAX
Scriptaculous
Simple Web Framework
Taconite
Tacos
WebWork
Wicket
Xajax
XAP
Yahoo UI Library
Zimbra Collaboration Suite
ZK

3.5 Platform independence (OS, server, browser, IDE)

One of the main attractions of Ajax is that it does not lock developers to a particular hardware platform, operating system, application server, web browser or IDE. Developers are free to choose among many technology providers, commercial and open source, to find the products or open source technologies that best match their unique requirements and preferences, while achieving the key benefit of write-once, run-everywhere, across multiple computing devices, operating systems, and web browsers.

Typically, Ajax toolkits deliver cross-platform and cross-browser functionality by providing a platform-neutral and browser-neutral abstraction layer to the developer. This layer is sometimes delivered as a set of client-side JavaScript libraries, and other times in the form of server-side software (e.g., Java).

3.6 Compatibility with HTML and existing web development technologies

Ajax can be added incrementally to existing HTML applications for a smooth and natural growth path to an Ajax-powered Web 2.0 and RIA user experience. Most of the technology behind Ajax is already familiar to the large pool of web developers who already know HTML and JavaScript. It is easy for them to learn quickly how to leverage Ajax to deliver next-generation solutions.

Ajax is fully compatible with the HTML application development infrastructure that exists today, including application servers (e.g., J2EE and .NET), server scripting languages (e.g., ASP, JSP and PHP), server application frameworks (e.g., JSF and Struts), web services, and service oriented architecture (SOA).

3.7 Option for phased adoption

Organizations have the option of moving from HTML to Ajax in a phased manner.

1. Add snippets of Ajax code within an HTML application
2. Use Ajax for the entire UI for one or more pages within a larger, primarily HTML web application
3. Use Ajax for the entire web application
4. Use Ajax as the basis for all web application development within your organization

3.8 Multiple alternative Ajax programming models

Ajax offers a wide range of architectural options (see OpenAjax Architectures). This diversity allows Ajax developers to choose from many different commercial products and/or open source technologies to find the ones that best match their existing application development infrastructure and technology preferences.

3.9 Developer productivity gains

3.9.1 Developer is fully empowered, but many opportunities exist for higher abstraction levels

With Ajax, the learning curve is shortened and investments minimized since application execution relies on Open standards support in web browsers. As a result, existing development and deployment frameworks and techniques still apply. The developer works in an environment he knows well and keeps full visibility and control, with the ability to code and debug all the way down to the DOM calls that affect what the browser displays. But Ajax also provides productivity advantages. Ajax toolkits typically offer declarative markup and APIs at higher abstraction levels and take care of lower-level details automatically.

The result is that Ajax offers the best of both worlds: automation advantages while still leaving the developer fully empowered.

3.9.2 Large ecosystem, off-the-shelf components

With so many Ajax commercial products vendors and open source initiatives, developers are likely to find the off-the-shelf components, toolkits, frameworks, educational materials, and other resources they need to deliver and maintain next-generation Web 2.0 applications built with Ajax.

3.9.3 Declarative UI

Many Ajax technologies provide declarative options (HTML/XML) for defining large parts of an Ajax application. These declarative options often automate large parts of the application development process and enable better leverage of IDEs.

3.9.4 Data management and data binding

Ajax libraries often provide the following features to enable client-side management of data:

- Data providers (e.g., web services)
- Validation
- Bi-directional data binding between client-side data blocks and associated UI controls

3.9.5 IDE integration

Some Ajax libraries deliver various JavaScript-oriented application development convenience features, such as JavaScript packaging and debugging aids. Some Ajax libraries go even further and deliver full application development platforms, sometimes in conjunction with associated software products such as IDEs. IDEs sometimes provide both server-side and client-side debugging.

4 OpenAjax Architectures

4.1 Multiple architecture options

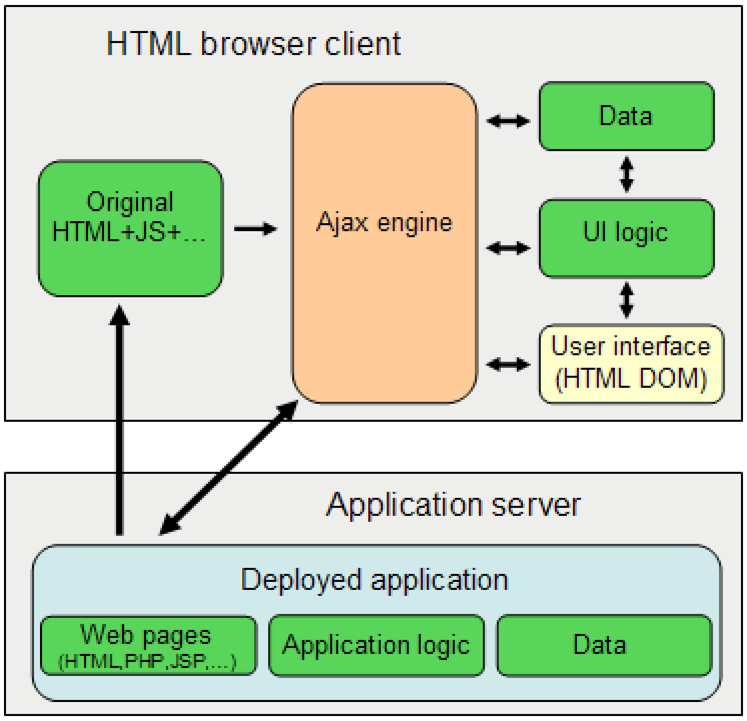
Ajax developers have the ability to choose among multiple technical approaches to find the ones whose programming model best fits their needs. The following are some of the ways to categorize Ajax toolkits and frameworks.

4.2 Client-side vs. server-side

Most Ajax technologies transform a platform-independent definition of the application (usually consisting of a combination of markup and logic) into the appropriate HTML and JavaScript content that is then processed by the browser to deliver a rich user experience. Some Ajax designs perform most of their transformations on the client. Others perform transformations on the server.

4.2.1 Client-side Ajax transformations

The diagram below shows a common approach to client-side Ajax.



With client-side Ajax, the Ajax engine runs on the client. The server delivers Web content (HTML, CSS, JavaScript, etc.) which is processed by the client-side Ajax engine into revised Web content. The browser renders the revised HTML/etc. content that comes out of the Ajax engine.

With this architecture, the application development team typically provides the following server-side components:

- Web pages (e.g., *.html, *.php, *.jsp, *.asp)
- application logic (e.g., Java)
- data management (e.g., via a SQL database and/or Web Services)

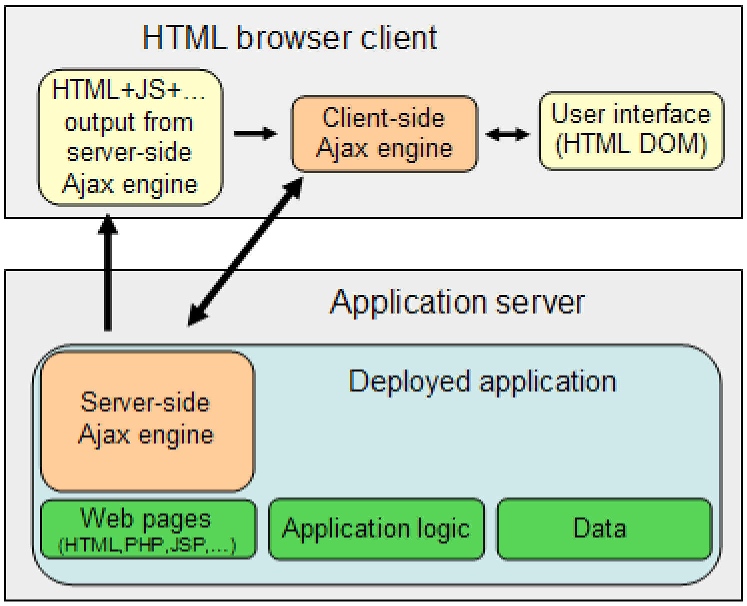
and the following client-side components:

- client-side user interface logic, such as event handlers for UI events
- client-side data management (e.g.., manage client-server data communications and update user interface elements to reflect new data)

One advantage of this option is the independence from the server side technology. The server code creates and serves the page and responds to the client's asynchronous requests. This way either side can be swapped with another implementation approach.

4.2.2 Server-side transformations

Server-side Ajax sometimes follows the model shown below:



For server-side Ajax, an Ajax server component performs most or all of the Ajax transformations. The server component generates the necessary Web content (HTML, CSS, JavaScript, etc.) to deliver the desired user experience. Often, the server-side Ajax toolkit downloads its own client-side Ajax library which communicates directly with the toolkit's server-side Ajax component.

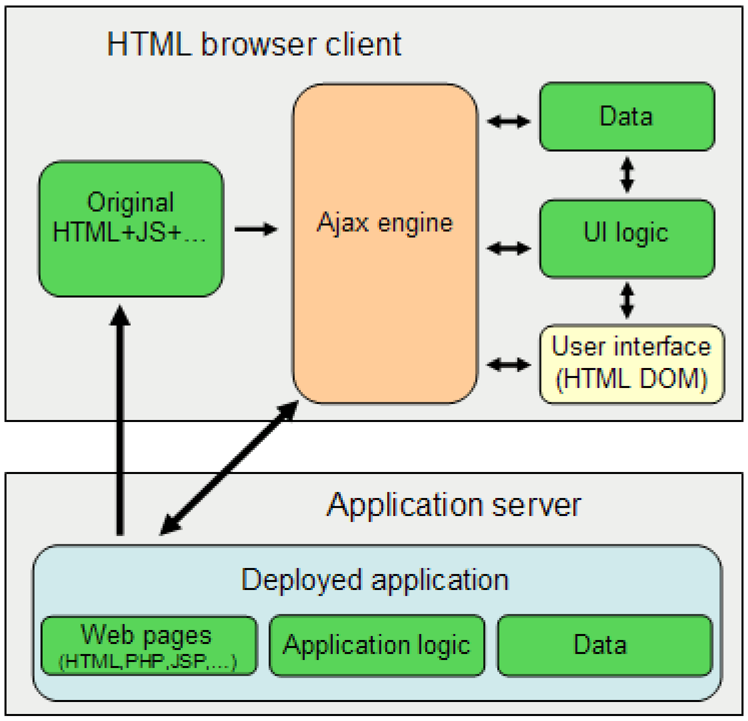
With this architecture, the application development team typically only provides server-side components (Web pages, application logic, and data management) and entrusts client-side logic to the Ajax toolkit.

The main benefit of this approach is that it allows the use of server-side languages for debugging, editing, and refactoring tools with which developers are already familiar, but at the price of dependence on a particular server-side technology. As a general rule, server-side Ajax frameworks expect application code to be written in the server-side language (e.g. Java or RoR). These frameworks typically hide all the JavaScript that runs in the browser inside widgets, including their events. If pre-built capabilities don't suffice, however, new component development expose the programmer to JavaScript. The implementation strategies of server-based Ajax frameworks vary. At one side of the spectrum, the server handles all the application events. On the other side of the spectrum, many events are handled on the client. For some frameworks, the development and debugging phase handles all events on the server, but in production mode many events are handled on the client.

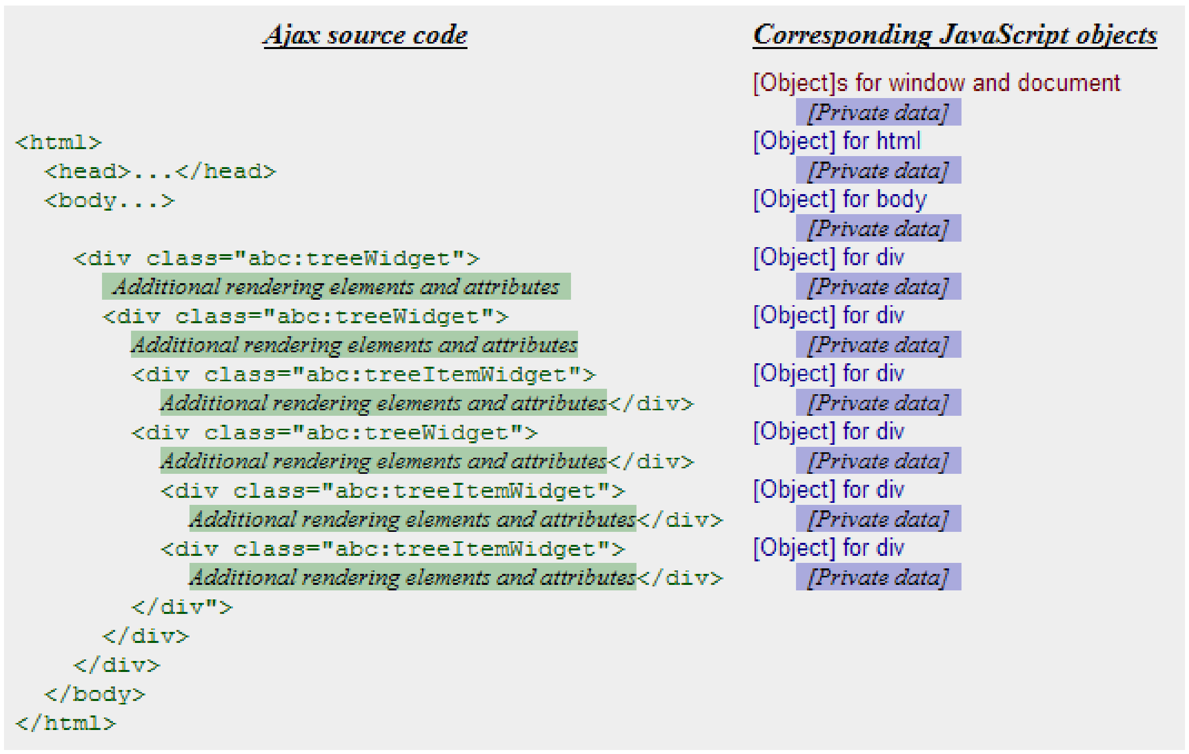
4.3 Single-DOM vs. Dual-DOM (client-side and server-side)

4.3.1 Single-DOM

Some Ajax runtime toolkits use a Single-DOM approach where the toolkit uses the browser's DOM for any original source markup and any HTML+JavaScript that results from the toolkit's Ajax-to-HTML transformation logic. The following example illustrates the Single-DOM Approach (same image as shown for Client-side Ajax transformations):



The figure below shows a detailed view of how an Ajax runtime library might manipulate the HTML DOM:



Ajax source code

```
<html>
  <head>...</head>
  <body...>

    <div class="abc:treeWidget">
      Additional rendering elements and attributes
      <div class="abc:treeWidget">
        Additional rendering elements and attributes
        <div class="abc:treeItemWidget">
          Additional rendering elements and attributes</div>
        <div class="abc:treeWidget">
          Additional rendering elements and attributes</div>
          <div class="abc:treeItemWidget">
            Additional rendering elements and attributes</div>
          <div class="abc:treeItemWidget">
            Additional rendering elements and attributes</div>
        </div">
      </div>
    </div>
  </body>
</html>
```

Corresponding JavaScript objects

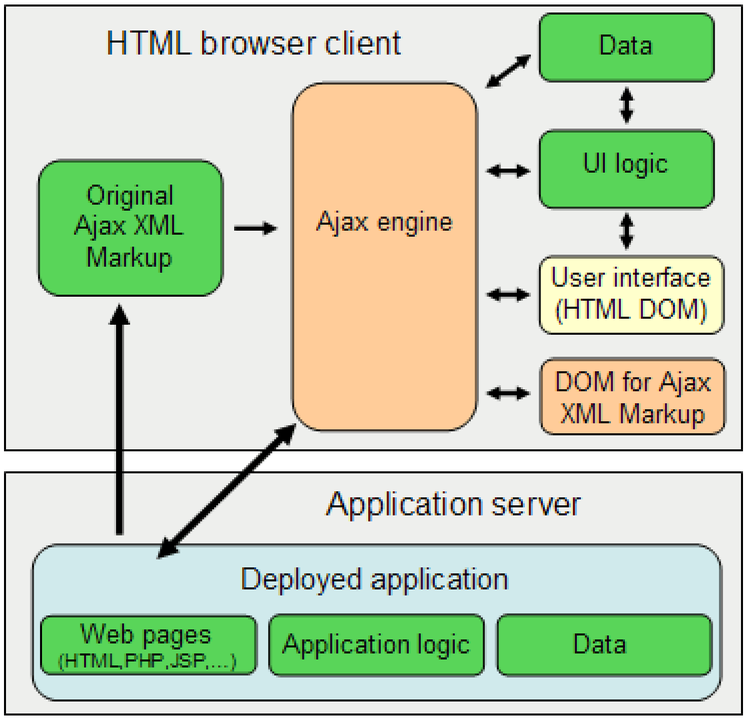
```
[Object]s for window and document
    [Private data]
[Object] for html
    [Private data]
[Object] for body
    [Private data]
[Object] for div
    [Private data]
[Object] for div
    [Private data]
[Object] for div
    [Private data]
[Object] for div
    [Private data]
[Object] for div
    [Private data]
[Object] for div
    [Private data]
```

In this example, the developer is using tree widgets from an Ajax runtime library. On the left, you see both the original HTML markup (unshaded) and the additional HTML markup inserted by the Ajax toolkit (shaded). On the right, you see the DOM objects that correspond to the elements in the HTML markup (e.g., the DOM object that represents a particular `<div>` element), where JavaScript/DOM objects from unshaded objects correspond to original HTML markup and shaded objects are ones that have been inserted by the Ajax toolkit. Typically, the Ajax toolkit inserts various rendering constructs such as `<img>` , `<div>`, and `<p>` elements, inline within the original HTML markup (e.g., adding child elements to an existing `<div>` element), thereby providing the various graphics and text necessary to produce the desired visual representation for the tree widgets. The shaded sections on the right reflect the private data that Ajax libraries typically add to various DOM and JavaScript objects in order to store private data, such as run-time state information.

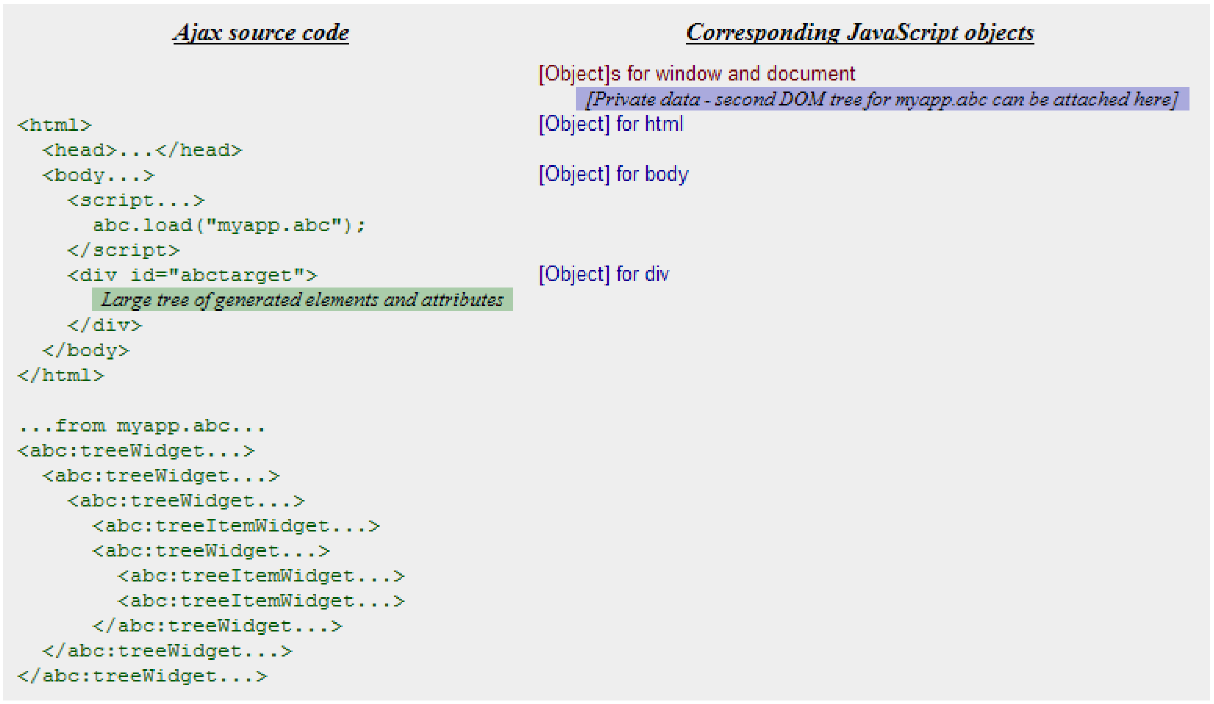
The Single-DOM approach is particularly well-suited for situations where the developer is adding islands of Ajax capability within an otherwise non-Ajax DHTML application, as the programming model matches the traditional approach used in DHTML applications.

4.3.2 Dual-DOM (client-side)

Other Ajax runtime toolkits adopt a Dual-DOM approach that separates the data for the Ajax-related markup into an "Ajax DOM" tree that is kept separate from the original "Browser DOM" tree. The image below illustrates the Dual-DOM (client-side) approach:



The figure below shows a detailed view of how the two DOMs might work (i.e., the HTML DOM and the XML DOM corresponding to the Ajax-specific XML markup for the user interface elements):



Ajax source code
Corresponding JavaScript objects
[Object]s for window and document
[Private data - second DOM tree for myapp.abc can be attached here]
[Object] for html
[Object] for body
[Object] for div
Large tree of generated elements and attributes

```
<html>
  <head>...</head>
  <body...>
    <script...>
      abc.load("myapp.abc");
    </script>
    <div id="abctarget">
      Large tree of generated elements and attributes
    </div>
  </body>
</html>

...from myapp.abc...
<abc:treeWidget...>
  <abc:treeWidget...>
    <abc:treeWidget...>
      <abc:treeItemWidget...>
      <abc:treeWidget...>
        <abc:treeItemWidget...>
        <abc:treeItemWidget...>
      </abc:treeWidget...>
  </abc:treeWidget...>
</abc:treeWidget...>
```

The above example shows a separate file, "myapp.abc", which contains the user interface definition for the tree widgets, which in this case are to be placed into the HTML tree inside the <div> element with id="abctarget". Even though the example shows the use of a separate file, some Dual-DOM Ajax runtime libraries support inline XML. In either case, a Dual-DOM Ajax runtime library builds a separate DOM tree, typically using its own XML parser rather than relying on the browser's HTML parser. Sometimes (as shown in the example) the separate DOM tree is attached to the 'window' or 'document' objects.

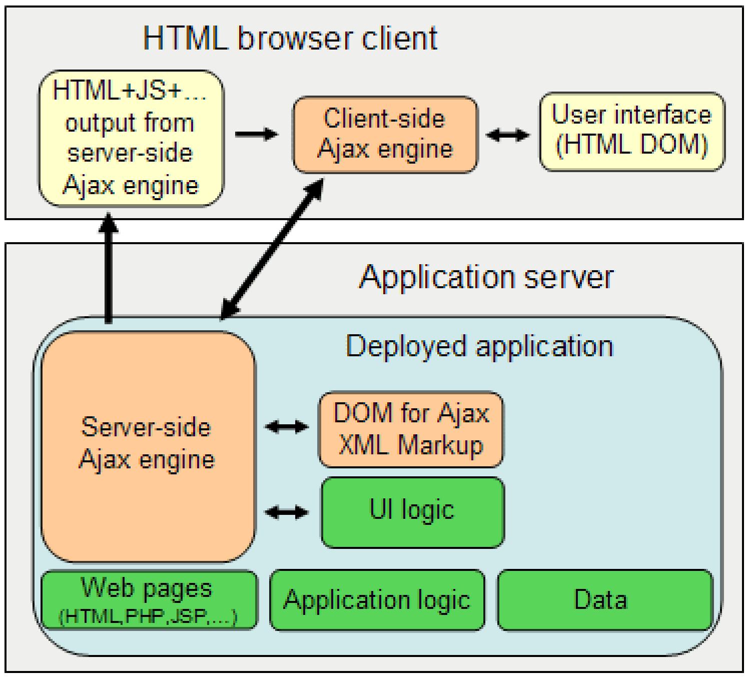
With this approach, in model view controller (MVC) terms, the Ajax DOM can be thought of as the model, the Browser DOM as the generated view, and the Ajax runtime toolkit as the controller.

It is usually necessary to establish bidirectional event listeners between the Ajax DOM and the Browser DOM in order to maintain synchronization. Sometimes having a separate Ajax DOM enables a more complete set of XML and DOM support, such as full support for XML Namespaces, than is possible in the Browser DOM.

The Dual-DOM approach has two flavors: Client-side Dual-DOM and Server-side Dual-DOM. With Client-side Dual-DOM, the second DOM tree typically consists of a separate tree of JavaScript objects. With Server-side Dual-DOM, the second DOM tree is stored on the server. The following section describes Server-side Dual-DOM.

4.3.3 Dual-DOM (server-side)

Some Ajax technologies combine server-side Ajax transformations with a Dual-DOM approach:



The key difference between Server-side Dual-DOM and Client-side Dual-DOM is that, with Server-side Dual-DOM, the Ajax DOM and most user interface logic is managed on the server. In this scenario, the primary job of the client Ajax engine is to reflect back to the server any interaction state changes, deferring data handling, UI state management and UI update logic to the server.

Server-side Dual-DOM enables tight application integration with server-side development technologies such as Java Server Faces (JSF).

4.4 Procedural vs. declarative client-side frameworks

Most Ajax runtime toolkits combine procedural and declarative approaches.

With the procedural approach, the application developer directs the toolkit almost exclusively using JavaScript APIs to handle all the events and manipulate widgets.

With the declarative approach, the application developer directs the toolkit almost exclusively via HTML or XML markup to define the components on the page and the relationships between them. Custom XML tags are often defined in the framework's namespace. JavaScript code may be required in some circumstances, but one of the framework's goals is to minimize the need for logic.

5 OpenAjax Alliance - Fulfilling Ajax's Promise

5.1 Mission and objectives

The OpenAjax Alliance is an organization of vendors, open source initiatives, and web developers dedicated to the successful adoption of open and interoperable Ajax-based web technologies. The prime objective is to accelerate customer success with Ajax by promoting a customer's ability to mix and match solutions from Ajax technology providers and by helping to drive the future of the Ajax ecosystem.

The OpenAjax Alliance will provide value to the software community through both the marketing and technical fronts. The alliance will help educate the community how to achieve success with Ajax using open technologies. Through its committee activities, the alliance will address key Ajax interoperability issues so that developers can successfully use multiple Ajax technologies within the same web application.

The set of technologies known as "OpenAjax" will provide the following benefits to web developers:

- Lower development costs and faster delivery of Web 2.0 innovations
- Vendor choice and interoperability
- Richer web experience and greater collaboration that can be added incrementally to existing HTML websites or used for creating new applications

5.2 Process

The OpenAjax Alliance has a lightweight formal governance model defined by its Members Agreement. To become a Member of OpenAjax Alliance, an organization must sign and submit the Members Agreement. Instructions for becoming a Member can be found at http://www.openajax.org/join.html.

5.3 Activities

5.3.1 Committees

Most of the technical work is accomplished in committees. The following committees were established at the initial kick-off meeting:

- **Marketing / Architecture Committee** - Ajax definitions, white papers, block diagrams
- **Interoperability Committee** - Focuses on the ability to mix Ajax components from different Ajax toolkits within the same web application. Among the first

topics: JavaScript name collision prevention, toolkit loading, event management in the presence of multiple Ajax runtimes used in the same web application, and mixing markup (HTML and/or XML markup) from different Ajax toolkits within the same web application.

For more recent information about OpenAjax Alliance activities, visit the Web site (http://www.openajax.org).

5.3.2 Standards-related activities

The OpenAjax Alliance is not a formal standards body, but it will engage in standards-related activities as a means to achieve its objectives of greater interoperability, vendor choice, and promoting innovation. Here are some of its standards-related activities:

- **Collaboration with standards bodies** - Representatives of the OpenAjax Alliance may participate in standards activities within formal standards bodies to help accelerate the advancement of OpenAjax technologies and products.
- **Collaboration with open source efforts** - Sometimes it helps to implement a specification while it is being defined. OpenAjax Alliance will look to partner with open source organizations to further technical approaches.
- **Original technical work** - When other parts of the industry are not pursuing necessary technical specifications, the OpenAjax Alliance will develop specifications and/or open source to fill critical industry gaps. In these cases, the expected typical path will be to turn over such work at an appropriate point to other organizations, such as a formal standards organization or to an open source project. Often, specifications and corresponding implementation work will occur in parallel.

5.3.3 OpenAjax Hub

One of the first technical products of the OpenAjax Alliance will be a small, simple and lightweight JavaScript library called the OpenAjax Hub that will fill critical Ajax runtime interoperability requirements. For different Ajax toolkits to work together, there needs to be a central facility to prevent JavaScript and HTML/XML markup collisions and to provide appropriate bridging. The four main interoperability issues addressed with the first version of the OpenAjax Hub are JavaScript collision checking, toolkit loading, markup mixing and event management. The OpenAjax Hub will consist of both detailed specifications and open source JavaScript code.

5.3.4 Education and communication

The OpenAjax Alliance will engage in various educational and communication activities. Its web site will provide a standard vocabulary for industry terms such as "Ajax" and "OpenAjax" and include white papers and block diagrams on Ajax technologies and associated best practices, with a focus on cross-vendor interoperability. Representatives will speak about OpenAjax at conferences and other industry events.

The OpenAjax Alliance web site will provide a central point of information about the OpenAjax vision, explaining how to adopt Ajax successfully so that IT developers will feel safe about their technology and vendor choices.

5.3.5 Best Practices

In addition to committee work on technical specifications that promote interoperability, the OpenAjax Alliance will define OpenAjax Best Practices - recommendations for technical approaches that lead to optimal customer results, such as better interoperability, portability, accessibility, and internationalization.

5.3.6 Ecosystem leadership and Ajax's future

As needed to advance the future success of OpenAjax, the OpenAjax Alliance intends to play a leadership role in the Ajax ecosystem to ensure the success of Ajax using open technologies. The alliance's leadership activities will include marketing communications, vision white papers, speaking engagements, evangelism with technology providers, and industry events.

As part of its leadership role, the OpenAjax Alliance works with key web infrastructure players to raise the level of the least common denominator for the Ajax platform. This will involve coordination and evangelism with leading browser vendors, standards organizations, open source organizations, and software vendors.