

Embedded Ajax

Web 2.0 Optimized for Mobile Devices

NetFront Browser v3.5 takes Web 2.0 beyond the PC

Takuya Harakawa, Product Manager, Browser Technology
Dr. Tomihisa Kamada, Co-founder and CTO
ACCESS Co., Ltd.



1. Introduction

Shortly after Tim O'Reilly (the founder and CEO of O'Reilly Media) published his paper What is Web 2.0 [1] in 2005, the term Web 2.0 came to define the next-generation of the Web and the Web also became widely recognized as a next-generation service framework based on the view of the Web as a platform. Today, Web end-users can experience many new and innovative Web services, powered by Web 2.0 applications, which feature rich user-interfaces and advanced functionality accessed via the Web browser on their desktop PCs.

It can be said that the most compelling Web 2.0 technology is Ajax (Asynchronous JavaScript and XML) [2]. Jesse James Garrett of Adaptive Path first introduced the term Ajax in February 2005. Ajax is an open-standard Web development technique that uses a combination of XHTML, CSS, and JavaScript to enable the creation of highly efficient interactive Web applications. Ajax leverages the XMLHttpRequest object to support the asynchronous exchange of data between a web server and a client. The asynchronous exchange of data shifts a significant amount of application functionality from the Web server to the client, which means that Web pages do not have to be completely reloaded for each end-user interaction. Ajax enables Web 2.0 applications like Google™ Gmail™, Google Suggest, Google Maps, Amazon's A9.com search engine and Yahoo!'s Flickr.com to deliver a rich end-user experience with seamless functionality.

Running in parallel with the emergence of Ajax is the move in the mobile market away from WAP (Wireless Application Protocol) services and the idiosyncrasies of WML (Wireless Markup Language) and WAP gateways, which do not adhere to global Internet standards. WAP 2.0 [3], the next generation of WAP, is also rapidly becoming a out-of-date solution for mobile due to the growing popularity of mobile browsers that provide modern Internet browsing and the rapid evolution of the mobile environment including faster wireless network technologies and ever more powerful mobile devices. The direct result of this movement away from WAP and toward modern Internet mobile browsing is that Web 2.0 applications stand to become very popular throughout the mobile world

ACCESS has been a leading developer of Internet-compatible mobile browser solutions since it first submitted the Compact HTML specification to W3C in 1998. Compact HTML went on to become the basis for NTT DoCoMo's popular i-mode™ service. To date, ACCESS' NetFront™ browser has been deployed in over 300 million devices and the latest version, NetFront v3.5 provides state-of-the-art technologies that bring Web 2.0 to the mobile market. This paper describes the concept and design principles of Embedded Ajax, an Ajax implementation that is optimized for mobile devices. This paper also describes ACCESS' vision for the optimal Web 2.0 applications for the rapidly growing mobile market.

2. Why Embedded Ajax?

Advanced Web browsers with PC-like browsing capabilities are becoming increasingly popular on mobile devices. This growing popularity means that Web 2.0 applications that use Ajax will also soon become popular throughout the mobile world. It is important to note that Ajax is not only an asynchronous data retrieval technique using XMLHttpRequest, it is also a method for utilizing a narrow bandwidth and limited memory to its maximum extent. In other words, Ajax can reduce the amount of data transferred between a Web server and client device, which, in theory, means that it should be possible to use the same Web 2.0 applications for PCs and for mobile devices. This directly brings into question the need for an "Embedded Ajax" solution specifically adapted for mobile devices. Wouldn't using Ajax to build Web applications for mobile devices be enough?

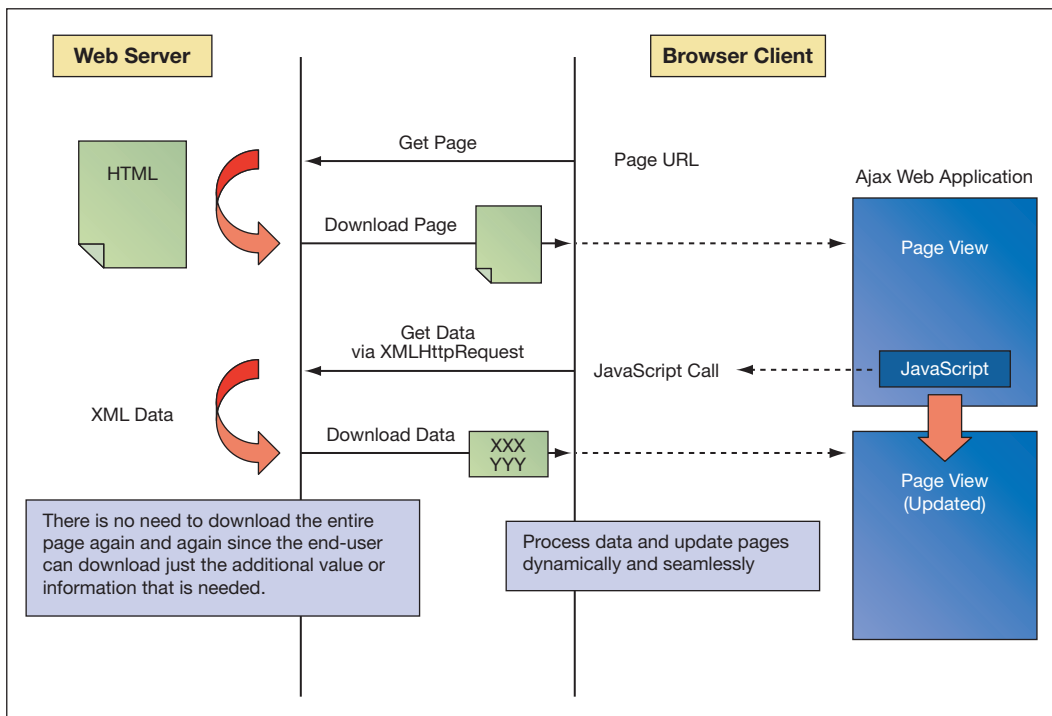


Figure 1: Ajax data transfer between server and client

Since there are still constraints and limitations in mobile device hardware and wireless networks, it will be difficult to deliver a PC-like Web 2.0 experience without some considered adjustments. In order to fully leverage the potential of Web 2.0 on mobile devices, it will be necessary to take the PC service model and end-user experience and adapt it for the mobile world while also overcoming the following issues inherent in mobile devices:

- Limited memory capacity
- Low-power CPUs (to reduce power consumption)
- Narrow bandwidth and low-speed networks (compared to PC networks)
- Limited key input

2.1 Performance

Although Ajax offers a method for the asynchronous retrieval of data required to dynamically run a Web application, the most important functionality is actually the JavaScript bindings with XHTML, CSS and DOM, which perform dynamical display changes and interactions. That is, Ajax requires optimized performance for running complicated JavaScript code on mobile devices. In order to realize this higher level of performance, a mobile device will need not only a highly efficient JavaScript engine, but also a smart framework to run the JavaScript code relevant to Ajax applications such as prototype.js [5] at the highest possible speed.

2.2 Memory Consumption

Memory consumption is another important issue that needs to be carefully considered. Memory management quality has a significant influence on the response and stability of a Web application. For example, on NetFront for mobile devices, the standard version of Google Maps acquires over 100Kbytes of content, and in order to display a map, it requires memory consumption at over 10Mbytes. As you can easily imagine, these requirements are not realistic for the majority of mobile devices currently available. Certainly, downsizing the image itself can reduce the memory consumption required to display an image. However, the potential vulnerabilities of memory leaking by operating DOM via JavaScript must also be considered. In Google's API [5], it's recommended to call the `GUUnload()` function on the `unload` event handler in order to clear unexpected memory leaks. With Embedded Ajax, a smart garbage collecting functionality for the authors of Web applications can be easily created.

2.3 Limited Key Input

Many mobile devices have limited key inputs (4-way navigation, enter, cancel, several soft keys, no mouse pointer) relative to desktop PCs. These hardware differences/restrictions create significant difficulties when developing rich Web applications for mobile devices. For example, when Web browsers are implemented on mobile devices, the Right/Left key is assigned to the history operation, so these keys cannot be used to interact with a Web application. This inability to use the Right/Left keys creates a barrier to sharing the same user-interface for a Web application on a PC relative to a mobile device. A different event system needs to be developed to overcome the interface restrictions created by the limited key input on mobile devices, and this can be accomplished with Embedded Ajax.

3. Design Goals

To overcome the performance, memory, and input restrictions on mobile devices ACCESS has developed Embedded Ajax, a new technology for mobile devices based on Ajax that provides enhanced performance and usability for Web 2.0 applications that use Ajax. Embedded Ajax is included in NetFront™ Browser v3.5, ACCESS' latest Web browser solution for mobile devices. NetFront Browser v3.5 is scheduled for release in autumn of 2007.

3.1 NetFront Browser v3.5, ACCESS's Browser for Mobile

NetFront Browser is an optimized browser solution that provides a PC-like Internet browsing experience for embedded devices, especially mobile devices. Unlike downsized desktop browsers, NetFront was originally designed to overcome restrictions like limited CPU speed or limited memory, while offering fast performance and extensive features all within a compact code size that is ideally suited for mobile devices. By including Embedded Ajax, NetFront Browser v3.5 offers the best value for the Web 2.0 mobile ecosystem, including network operators, handset vendors, content providers, and end-users.

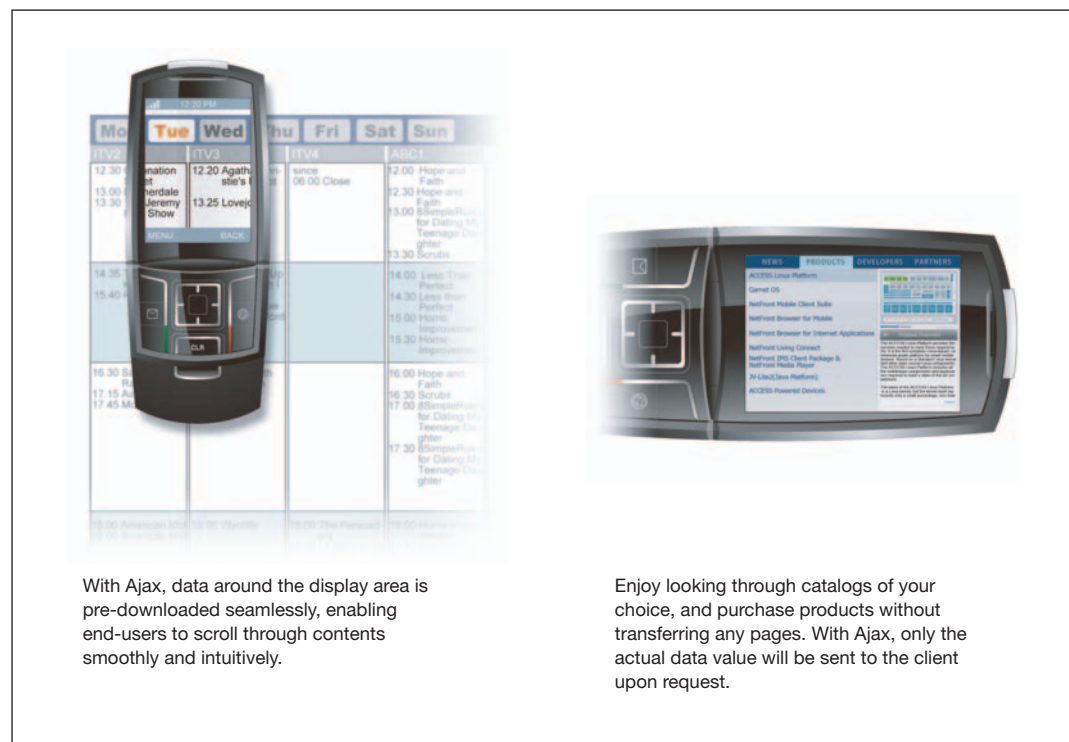


Figure 2: Example of Ajax applications on mobile phone

3.2 Architecture Overview

ACCESS developed Embedded Ajax to introduce more effective and attractive Web applications with a new, dynamical user-interface framework based on standard Ajax technology.

Embedded Ajax incorporates:

- Optimized extra-DOM operations for Ajax applications
- Garbage collecting functionalities for JavaScript
- Alternative virtual key event handling for mobile devices

Embedded Ajax embeds and provides the following functionalities introduced by many Ajax utilities including Prototype.js:

- Extended functionalities for accessing a JavaScript object
- Functions that support smart coding

Embedded Ajax also provides three particular functionalities, in order to achieve high performance and usability for Web 2.0 applications that use Ajax technology on mobile devices.

- (1) Highly sophisticated garbage collecting system.
- (2) A function that can invoke garbage collection when a content creator requires.
- (3) Virtual event handling system for input keys to facilitate the development of Web application for mobile devices.

4. Concluding Remarks

In this paper, we described the concept and design of Embedded Ajax, and also discussed the issues surrounding the introduction of Web 2.0 applications that use Ajax on mobile devices. Then, we presented how Embedded Ajax and NetFront Browser v3.5 can resolve these issues.

Embedded Ajax can deliver a fast, flexible, and seamless end-user experience for Web 2.0 applications accessed via mobile devices. ACCESS believes that Embedded Ajax will become the standard, optimized Ajax technology for all mobile devices.

5. References

- [1] Tim O'Reilly, *What Is Web 2.0*, September, 2005
<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>
- [2] Jesse James Garrett, *Ajax: A New approach to Web Applications*, February, 2005
<http://www.adaptivepath.com/publications/essays/archives/000385.php>
- [3] Open Mobile Alliance, *Wireless Application Protocol*
<http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html>
- [4] Tomihisa Kamada, *Compact HTML for Small Information Appliances*, W3C, February, 1998
<http://www.w3.org/TR/1998/NOTE-compactHTML-19980209/>
- [5] Sam Stephenson, *Prototype*, 2005
<http://www.prototypejs.org/>
- [6] Google, *Google Maps API*, 2006
<http://www.google.com/apis/maps/documentation/>

Appendix A: extra-DOM Operations

Table 1 shows a part of the extra-DOM operations provided by Embedded Ajax. Some of the operations have already been implemented on Ajax utilities such as prototype.js.

NOTE: The functionality and the name of the methods described below are tentative.

| Method | Description |
|--------------------------------------|---|
| <code>getElementByClassName()</code> | Specify the element by its class name |
| <code>Insertion.After()</code> | Insert HTML after specified element |
| <code>Insertion.Before()</code> | Insert HTML before specified element |
| <code>Insertion.Bottom()</code> | Insert HTML as a last child of specified element |
| <code>Insertion.Top()</code> | Insert HTML as a first child of specified element |
| <code>addClassName()</code> | Add the given name to specified element as a class name |
| <code>classNames()</code> | Get the class name of specified element |
| <code>hasClassName()</code> | Check if specified element has class name |
| <code>removeClassName()</code> | Remove the class name of specified element |
| <code>visible()</code> | Check if specified element is visible |
| <code>escapeHTML()</code> | Make escaped string as suitable for HTML characters |
| <code>evalScript()</code> | Evaluates given string as script |
| <code>extractScripts()</code> | Extract script string from given string |
| <code>stripScripts()</code> | Remove script string from given string |
| <code>stripTags()</code> | Remove tag string from given string |

Table 1: A partial list of extra-DOM operations in Embedded Ajax

Appendix B: Explicit Garbage Collection

In order to optimize memory usage, the JavaScript engine normally invokes garbage collection to free unused objects. The timing of garbage collection strongly depends on the implementation of the JavaScript engine, however, with Ajax, it is useful for an author of Web content to be able to control this timing on demand, as required by the content. NetFront Browser v3.5 provides the following method for garbage collection.

```
void window.eajax.gc();
```

Appendix C: Alternative Virtual Key Event Handling

Embedded Ajax offers an alternative virtual key event called `NavEvent` for the input keys of mobile devices as follows.

```
interface NavEvent : UIEvent {
    const unsigned short NAVKEY_OK = 1;
    const unsigned short NAVKEY_CANCEL = 2;
    const unsigned short NAVKEY_BACKWARD = 3;
    const unsigned short NAVKEY_FORWARD = 4;
    const unsigned short NAVKEY_UP = 5;
    const unsigned short NAVKEY_RIGHT = 6;
    const unsigned short NAVKEY_DOWN = 7;
    const unsigned short NAVKEY_LEFT = 8;
    const unsigned short NAVKEY_UP_RIGHT = 9;
    const unsigned short NAVKEY_UP_LEFT = 10;
    const unsigned short NAVKEY_DOWN_RIGHT = 11;
    const unsigned short NAVKEY_DOWN_LEFT = 12;

    readonly attribute unsigned short navType;
};
```

In order to obtain the object that has the focus, Web application authors can use the following attribute and document object method.

```
readonly attribute Element activeElement;
boolean hasFocus();
```



ACCESS and NetFront are trademarks or registered trademarks of ACCESS Co., Ltd. in Japan and other countries. PalmSource, Palm OS, Palm Powered and certain other trade names, trademarks and logos are trademarks which may be registered in the United States, France, Germany, Japan, the United Kingdom and other countries and are either owned by ACCESS Systems Americas, Inc., or its affiliates, or are licensed by ACCESS Systems Americas, Inc., from Palm Trademark Holding Company, LLC. These marks may not be used in connection with any product or service that does not belong to ACCESS Systems Americas, Inc. (except as expressly permitted by a license with ACCESS Systems Americas, Inc.), in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits ACCESS Systems Americas, Inc., its licensor, its subsidiaries or affiliates. ACCESS Linux Platform and GHost are codenames subject to change upon release of the final product without prior notice, in the sole discretion of ACCESS Co., Ltd., ACCESS Systems Americas, Inc., or the applicable third party. Linux is the registered trademark of Linus Torvalds in the U.S. and other countries. Copyright © 2007 ACCESS Co., Ltd. and ACCESS Systems Americas, Inc. (formerly PalmSource, Inc.)

ACCESS CO., LTD.

Hirata Bldg, 2-8-16 Sarugaku-cho, Chiyoda-ku,
Tokyo 101-0064 Japan
PHONE +81-3-5259-3511 FAX +81-3-5259-3544
E-mail: adinfo@access.co.jp

ACCESS CO., LTD. Taipei Office

Suite105, 17F/B, No.167, Tun Hwa N.Rd., Taipei, 10549, Taiwan
PHONE +886-2-2717-1999 EXT. 1151-3
FAX +886-2-2717-2199

ACCESS Systems Europe GmbH

Essener Strasse 5 TZU-IV D-46047 Oberhausen, Germany
PHONE +49-208-6290-6464
FAX +49-208-6290-6465

ACCESS SEOUL CO., LTD.

8F/9F, PAXTOWER, 231-13, Nonhyun-Dong, Gangnam-Gu,
Seoul, 135-010, Korea
PHONE +82-2-513-2000
FAX +82-2-513-2010

ACCESS (Beijing) CO., LTD.

(爱可信(北京)技术有限公司)

Suite 2012, Floor 20, China Merchants Tower,
No.118 Jian Guo Road, Chao Yang District, Beijing, 100022, China
PHONE +86-10-6566-9636
FAX +86-10-6566-9637

ACCESS Systems Americas, Inc.

1188 East Arques Avenue, Sunnyvale CA 94085 U.S.A.
PHONE +1-408-400-3000
FAX +1-408-400-1500

Developer

<http://www.access-company.com>