Asynchronous JavaScript and XML

Literature study on Ajax as part of the project 'AJAX for Content Management'.
By: Teunis van Wijngaarden (teunis@few.vu.nl)

Abstract

[Voor dit document een aparte abstract schrijven?]

1 Contents

2 Introduction

[Voor dit document een aparte introduction schrijven?]

3 Ajax history

[Mag ik XHTML en CSS als voorkennis beschouwen?]

In the early years of the world wide web, it was just a huge collection of static HTML-pages, interconnected by hyperlinks. There were a few web browsers to access these pages, by pointing to their Unified Resource Locator (URL). In order to read a page, the user enters the URL into the webbrowser. The URL is used to invoke the server and request that specific page or file. The server sends the requested data and the browser displays it.

The (X)HTML format provides a way to format the page [1]. In time, more and more attention was paid to this formatting, using HTML in combination with images. It became common to use some basic elements on a website, e.g. a header and a menu. These elements were the same for most related pages, but needed to be included in every page. This redundancy problem was later resolved by the introduction of frames, see section 3.2.

3.1 Introduction of Javascript

Netscape was the first to introduce scripts at client side: LiveScript [2]. These small programs build into HTML pages made it possible to do simple tasks, like form validation. Before scripting, form validation was done server side. Later on, a similar script called ECMA script became a standard and the currently well known JavaScript was based on that.

[Uitzoeken wat het verschil is met ECMA Script, JScript? Ik kan hier geen academische bron van vinden.]

3.2 Introduction of frames

In the HTML4.0 specification (1998), frames were introduced to split pages in multiple pages [3]. The frameset HTML entity was the new element that could turn a page into a framework for other pages. This is useful to separate the static from the dynamic content: the menu could be on a different page than the content, but in the same framework. A click on a menu link causes the page in the content frame to change. Now, the webdesigner did not need to add the menu to every HTML page – which was very profitable for maintainance – and the amount of data to send over was reduced.

3.3 Ansynchronous requests with ‘old’ techniques

Frames together with JavaScript gave developers the opportunity to contact the server asynchronously [4][2]. A hidden frame – with width or height of zero pixels – was used as a communication channel. Forms on a page in this hidden frame could be filled out and sent with JavaScript. Response could be extracted from the hidden page with JavaScript too. So, the page in the main frame did not reload, only the hidden page did.

Microsoft was the first to introduce the Document Object Model (DOM) in a new form of the old markup language: Dynamic HTML [2]. Although DHTML never made it to a standard, the DOM did in 1997 [5]. The DOM is a way to access parts of a page by their path in the HTML tree or by using DOM-functionality, for instance by using an entity's id attribute. The DOM is now an essential part of Ajax.

In the same year that the DOM became a standard, a new element was introduced to HTML: the iframe. This element could in itself contain a webpage. The frameset became superfluous (for our purpose) and the hidden frame technique could be carried out with a zero by zero iframe. Great advantage was that such an iframe could be created on-the-fly using JavaScript and the DOM. This made asynchronous calls possible without having to plan a hidden frame in the HTML design (which encourages separation of 'code' and presentation).

3.4 An easier way to perform Asynchronous requests

Microsoft was the first to 'legalize' asynchronous requests with the introduction of an ActiveX object XMLHttp [2]. This made it possible to do HTTP requests with JavaScript, which became very popular. Mozilla introduced XMLHttpRequest as a JavaScript object [6], with the same functionality but not using ActiveX. Other browsers followed Mozilla and in the end even Microsoft added Mozilla's implementation to Internet Explorer 7[1].

The XMLHttpRequest is at the time writing not a W3C Recommendation, but its draft is in a final state [7]. The formal description of XMLHttpRequest maintained by W3C is: "The XMLHttpRequest object can be used by scripts to programmatically connect to their originating server via HTTP." [8]

3.5 The new approach: Ajax

The introduction of XMLHttpRequest completed the set of technologies that together forms Ajax. In 2005, Jesse James Garrett was the first to combine these technologies and he invented the term Ajax: Asynchronous JavaScript and XML. According to him it incorporates the following [9]:

- XHTML and CSS
- The DOM
- XML and XSLT (a stylesheet for XML)
- XMLHttpRequest
- JavaScript

This definition of Ajax does not include the server side, that plays a central role in handling the requests. Like Garrett's publication, this thesis also emphasizes client side technology. Server side scripts, written in some kind of language (PHP, ASP or else) probably with a connection to a database, will be discussed on the side. These scripts do not use any new

[1] MSDN, XMLHttpRequest object: http://msdn.microsoft.com/en-us/library/ms535874(VS.85).aspx

technology (like XMLHttpRequest in Javascript), only the output format XML is relatively new.

Ajax changes the communicationmodel for web applications[9][2]. Without Ajax the user requests a page (HTTP Request by the browser), waits, and receives a page (HTML and CSS). The real work is done at server side: using a database, form validation etcetera. Now, with Ajax the user requests a page, containing an Ajax application (the Ajax engine). Once the application (HTML, CSS and JavaScript) is received, further HTTP Requests are done by the Ajax engine in the background. Whereas the 'old' web application relies heavily on the server, the new web application moves functionality to the clients side. Figure 0-1 shows this model.

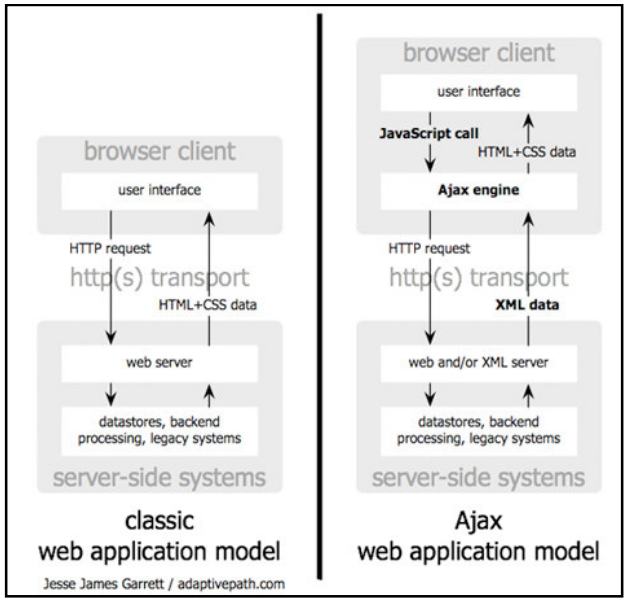

*Figure 0-1. The traditional and Ajax model for Web applications according to Jesse James Garett **[9]**.*

[mag ik dit model zomaar overnemen?]

Why should developers convert their traditional web application to Ajax based applications? A reason could be the new (better?) interaction model. The old applications cause the user to work on a page, request a new page or send a form, and wait until it is loaded. This repetition of working and waiting is called the work-wait model[9][4]. Ajax applications break through this traditional model. They allow the user to work on client side, while the sending and loading happens most of the time behind the scenes by the Ajax engine. In fact Ajax applications behave more like desktop applications.

Another reason to start programming Ajax is the range of new possibilities. Ansynchronous request do not only change the interaction with current web applications, but new applications or websites can be designed. With requests in the background it is possible to reload parts of page, i.e. actualize the news headlines at a pages sidebar while the user reads an article. Preloading pages, for instance the next chapter of an e-book, can be done by Ajax. Other patterns are: pre-submitting parts of a form (i.e. a file) while the user fills out the rest, or buffering submitted information (submission throttling) [10]. More innovative functionality can be found in the examples in the next paragraph or in the chapter on design decisions.

3.6 Ajax's context: Web 2.0

Up until now we discussed Ajax as technology on its own, but it is part of a technological shift. According to Tim O'reilly Ajax is a key component of *web2.0*. Although some companies use it as a buzzword, the term web2.0 does define the 'new internet' with a specific type of website – although the characteristics are not clear [11]. At least, web2.0 websites or applications share some characteristics: rich user experience, user participation,

dynamic content, metadata, web standards and scalability[12]. O'reilly adds to this: lightweight programming and the end of software lifecycles. He mentions Ajax under the header *rich user experience*.

Macromedia invented a term for web applications that accomplish a rich user experience: Rich Internet Applications (RIA). Technologies to build such applications are not only Ajax, but also Flash/Flex and Java Applets. Their driving forces are: the introduction of broadband, increasing computing power of clients, demand of responsive applications, choice of big companies (like Google) and the emergence of web services and Service Oriented Architectures (and XML) [13].

The use of asynchronous requests is not the only thing that makes a richer experience. The design – that achieves most of the look-and-feel – also adds to this. In fact the techniques used to create an enhanced graphical layout have nothing to do with asynchronous communication, but it does include the some of the Ajax contained techniques: HTML, CSS and XSLT.

Another web2.0 thing are (third party) webservices. Google for instance provides access to their applications (Google Maps, Picase Webalbums and more) by providing an API[2]. Yahoo does the same for their Flickr photoalbum[3]. These API's allow developers to include parts of the services in their website, embedded in their own layout. They also allow some sort of creativity: mashups. An example mashup is photos from a Flickr album to locations on a Google Map. A similar example is traffic information (from a third party or from your own company) on a Google Map. Combining several services creates thousands of new possibilities.

3.7 Real life examples

Form validation, for instance checking an emailfield on the '@' and dot extension, is not real Ajax. No communication takes place. An advanced example of Ajax is Google Suggest. While the user types search keywords, the application contacts Google in the back and requests suggestions. See the example in **Fout! Verwijzingsbron niet gevonden.**: the user typed 'Aj' and suggestions like 'ajax', 'ajc', 'ajc jobs', 'ajilon' pop up.

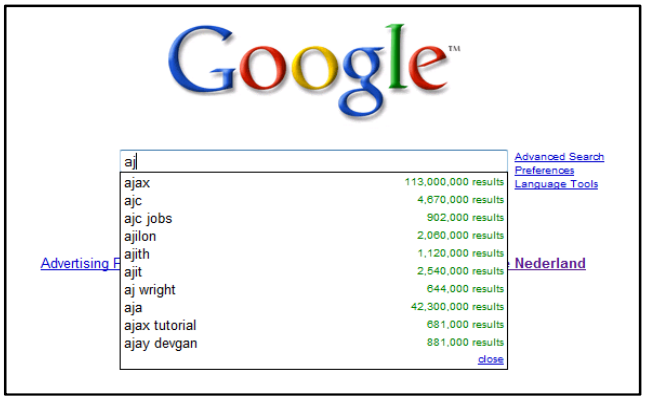

Figure 0-2. Google Suggest uses Ajax to suggest search terms. Google Suggest: http://www.google.com/webhp?complete=1&hl=en

[2] http://code.google.com/
[3] http://www.flickr.com/services/api/

Another Ajax example is Google Maps, launched in 2005[4]. The maps shown by Google Maps are just images, but ajax comes in when moving the map. As the world is round it should be possible to move into infinite, never reaching an end. Loading the world (in full detail, on a high zoom level) would take a lot of time. Google Maps therefore only loads the visible parts of the maps and near regions. While moving around, the ajax engine in the back loads new near regions that 'might be needed in future'.

A much more complex example of Ajax is the online e-mail application Google Gmail (see Figure 0-3). Without reloading the page, new e-mails popup. Attachments to e-mails upload while users type their mail. When they click 'send', the message is out in the web immediately. Google made much more complex ajax applications in Google Docs. Other companies that bring Ajax to a high level are Yahoo [MEER TOEVOEGEN, niet alleen Google apps, ook voorbeeld van Mashup toevoegen].

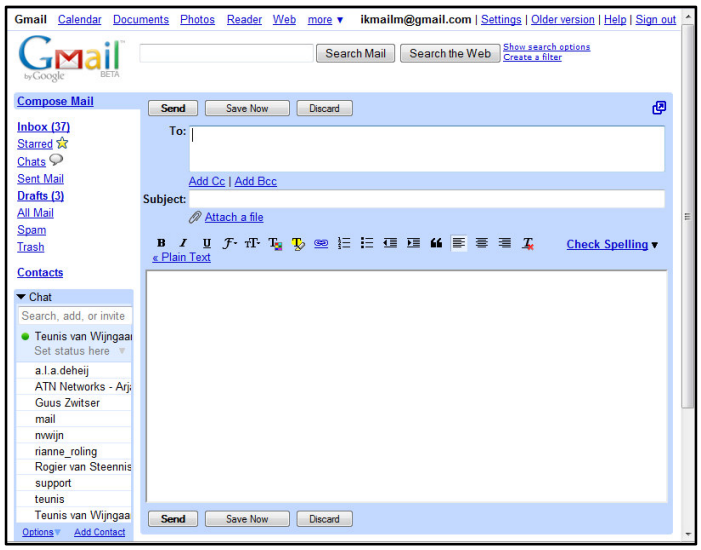

Figure 0-3. The Google Gmail application behaves like a desktop application. Google Gmail: http://www.gmail.com

4 Ajax into practice

This chapter handles how to program Ajax: the basics of an HTTP Request, basic DOM manipulation and some practical information on dataformats. The basics of XHTML and CSS are considered to be known. If not, W3 Schools might be helpful[5].

4.1 The Document Object Model

The DOM, short for Document Object Model, provides a way to handle structured documents, like (X)HTML documents. According to W3C, the DOM is "a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page." [5] The DOM's history at W3C starts in 1998 with DOM level 1. Development stopped in 2004 with DOM Level 3. We will use the DOM to access and manipulate XHTML documents.

[4] http://www.google.com/corporate/history.html
[5] W3 Schools, http://www.w3schools.com

The DOM views an XHTML document as a tree, with the XHTML elements as nodes[6]. A way access a node is by navigating through the tree. A node's attributes to do this are: `parentNode`, `childNodes` (which returns a list of nodes), `firstChild`, `lastChild`, `previousSibling` and `nextSibling`. Another way of navigating is by retrieving a list of nodes with a certain tag by using `getElementsByTagName()`. In addition, an easy way to access a node with a known id attribute is `getElementById()`.

JavaScript programmers are probably not satisfied with read-only access and want to manipulate the document or nodes. Therefore, the DOM provides ways to add and remove nodes, for instance `insertBefore()`, `removeChild()` or `appendChild()`. In addition there is a way to manipulate the nodes itself by changing their inner HTML or attributes.

Our focus is on using the DOM in JavaScript to access (XHTML) pages. W3Schools[7] offers an overview of functionality to do so. The best way to get to know the DOM is learning by doing. Figure 0-4 shows some simple JavaScript functions that manipulates the XHTML webpage of Figure 0-5.

```
1  function helloWorld() {
2      // get reference to the element:
3      var element = document.getElementById("text");
4      // change the inner HTML:
5      element.innerHTML = "Hello World!";
6  }
7  function addBorder(size) {
8      var element = document.getElementById("text");
9      // change the style using the size variable:
10     element.style.border = size+"px #000000 solid";
11 }
12 function addFrame() {
13     // create a new iframe element:
14     var iFrame = document.createElement("iframe");
15      // add the source attribute:
16     iFrame.setAttribute("src","http://www.vu.nl");
17
18     // some style settings:
19     iFrame.style.position = "absolute";
20     iFrame.style.left = "10%";
21     iFrame.style.width="80%";
22     iFrame.style.height="500px";
23
24     // append the new element to the body
25     document.body.appendChild(iFrame);
26 }
```

Figure 0-4. Some simple JavaScript functions that change the XHTML source of a page.

[6] http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/core.html
[7] http://www.w3schools.com/js/js_obj_htmldom.asp

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>JavaScript Example</title>
    <script type="text/javascript" src="scripts.js"></script>
  </head>
  <body>
    <p>
      <!-- links that call the functions -->
      <a href="javascript: helloWorld();">Hello?</a> |
      <a href="javascript: addBorder(2);">Add thin border</a> |
      <a href="javascript: addBorder(6);">Add thick border</a> |
      <a href="javascript: addBorder(0);">Remove border</a> |
      <a href="javascript: addFrame();">Open Frame</a>
    </p>

    <!-- an empty div element that
         will be filled by the script -->
    <div id="text"></div>
  </body>
</html>
```

Figure 0-5. An XHTML page with a reference to the script of the previous figure. The hyperlinks invoke the functions of the script and thereby change or add elements and attributes.

4.2 XMLHttpRequest

Most popular browsers have implemented XMLHttpRequest in their newest versions[8], but for older versions Microsoft's implementation differs. In Internet Explorer 6 and before, the object was not a native part of the browser, see section 3.4. Instead ActiveX was used. This is why the else clause is added to the example in Figure 0-6.

```
if (window.XMLHttpRequest) {
    /* XMLHttpRequest is implemented in the browser,
       use it. */
    xhr = new XMLHttpRequest();
} else if (window.ActiveXObject) {
    /* if XMLHttpRequest is not implemented, we
       probably deal with Internet Explorer 6 or
       older. Try to make a connection via
       ActiveX */
    xhr = new ActiveXObject("Microsoft.XMLHTTP");
} else {
    /* the browser does not have one of the objects
       to use Ajax */
    alert("Ajax is not supported.");
}
```

Figure 0-6. Using XMLHttpRequest or the IE6 equivalent.

[8] Firefox, http://developer.mozilla.org/en/XMLHttpRequest
Internet Explorer, http://msdn.microsoft.com/en-us/library/ms535874(VS.85).aspx
Opera, http://www.opera.com/docs/specs/opera9/xhr/
Safari, http://developer.apple.com/internet/safari/faq.html#anchor19

Once the object is created, requests can be send. Figure 0-7 shows an example of a `GET` and a `POST` request. Both use the similar syntax. The `open()` function of the XMLHttpRequest object is called. Any HTTP command could be used, but `GET` and `POST` are most useful for Ajax purposes. There are other types of `open` functions that require more parameters, but those are not discussed here to keep the example simple.

```
// create a GET request
xhr.open("GET", "localfile.txt");
xhr.setRequestHeader("Cache-Control", "no-cache");
xhr.setRequestHeader("Pragma", "no-cache");

// set the function that should
// handle the response
xhr.onreadystatechange = handleResponse;

// send the request
xhr.send(null);
```

```
// create a POST request
xhr.open("POST", "localscript.php");
xhr.setRequestHeader("Cache-Control", "no-cache");
xhr.setRequestHeader("Pragma", "no-cache");

// set the function that should
// handle the response
xhr.onreadystatechange = handleResponse;

// send the request
xhr.send("min-age=30&max-age=40");
```

Figure 0-7. Example of a GET and a POST request. The POST request send its variables in the send function. It is also possible to route formfields directly to this function.

There are some more functions and variables in the XMLHttpRequest object, having to do with headers, status and MIME types. Please consult the W3C website for information about this[9].

The variables of the example `POST` request are probably send to a server side script (at the same domain as where the script came from). That script might consult a database and return an answer. All kinds of databases and PHP, ASP or other scripts can be used to generate the respond, but this is not covered in this thesis. Just assume that the response is there. Figure 0-8 shows how the function `handleResponse()` handles the respond.

```
function handleResponse() {
    if (xhr.readyState == 4) {
        if (xhr.status == 200) {
            if (xhr.responseText != null) {
                // put the result in a div with id 'text'
                document.getElementById("text").innerHTML = xhr.responseText;
            }
        }
    }
}
```

Figure 0-8. This function handles the Ajax call response. The text contained in the returned package is put into an HTML element's inner HTML.

The `onreadystatechange`-function will not only be invoked when the request is finished and the result is there, but also in intermediate states: when open has not been called (0), when open has been executed (1), when send has been executed (2), when the server has returned a chunk of data (3) and when the request is complete and the server is finished sending data (4) [4]. Furthermore the status property of the XMLHttpRequest object contains the HTTP status, i.e. 200 for success or 404 in case the file is not found.

[9] http://www.w3.org/TR/2008/WD-XMLHttpRequest2-20080930/#references

Important to remark is the cross-domain issue. Browser do not allow a script that is acquired from one server to communicate with a another server. So, the developer is bound to his own server with his own services. There are some ways to overcome this, for instance when a (third party) service 'knows you' and allows your requests. In example, the services of Google use a identification key for every domain. Another solution is using a (simple) proxy, for instance a server side PHP script that downloads and displays the requested page.

4.3 Data formats: XML and JSON

The term Ajax, in which the 'x' stand for XML, suggests that XML is the only data format used for communication. This is not true, every format is possible. Though, XML is a good choice, because it is a standard data format[14] and programming languages on both client and server side already contain XML parsers, or those are made available by third parties. Another data format, covered by some programming languages, is JavaScript Object Notation (JSON), which has the advantage of being lightweight.

There is a lot of discussion on the web about which technique, JSON or XML, is best in what situation. This thesis will not cover that topic, except when it comes to the choice of techniques for the final content management system.

4.3.1 eXtensible Markup Language (XML)

The eXtensible Markup Language (XML) is a simple text format that Ajax applications can use to communicate with the server. Two of W3C's design goals for XML are that it should "support a wide variety of applications" and it should be "be easy to write programs which process XML documents" [14]. These goals are met when it comes to Ajax applications, which are very various and which are based on many sorts of server side languages that support XML. The scripting language PHP for instance provides an extension for processing XML[10].

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE news SYSTEM "news.dtd">

<?xml-stylesheet type="text/xsl" href="news.xsl"?>
<news>
    <news-item>
      <title>This is the title of a news-item</title>
      <article>
        This is the content of the article. Between these
        tags is a lot of text just like in the newspaper!
      </article>
      <author>
        <author_name>van Wijngaarden</author_name>
        <author_initials>T. D.</author_initials>
      </author>
      <publication_date>28 februari 2006</publication_date>

      <keyword><word>news</word></keyword>
      <keyword><word>paper</word></keyword>
      <keyword><word>test</word></keyword>

    </news-item>
</news>
```

Figure 0-9. An example XML file.

XML, like HTML, is derived from SGML, which is a system for defining markup languages. Each SGML application contains a reference to a document type definition (DTD) that defines the syntax of markup constructs, like the XHTML example in section 4.1 [15]. See Figure 0-9 for an

[10] PHP: XML Parser, http://nl2.php.net/manual/en/book.xml.php

example `DOCTYPE` declaration in an XML file. The DTD syntax is out of scope for this thesis, please consult the W3C website for more information[11]. The example XML document in the figure shows that XML is very similar to XHTML, which is in fact XML and which is also derived from SGML.

The root element in the example in Figure 0-9 is `news`. This elements contains one or more `news-item`s. Those items have some properties, saved in child nodes. The definition of what elements `news` and `new-item` can contain is in the DTD. This agreement on the format is useful when communication takes place between different applications (i.e. written in different languages, running on different platforms).

The browsers Internet Explorer and Firefox display XML documents as a tree, with collapsible nodes, unless the creator refers to a stylesheet (as in the example). The stylesheets in XSLT format define the style of elements and thereby determine the layout. Stylesheets are part of Ajax, according to the definition of Jesse James Garrett [9], but this technology will not be used in this thesis.

4.3.2 JavaScript Object Notation

JavaScript Object Notation (JSON) is a lightweight data-interchange format[15], according to David Crockford, who proposed this new data format [2]. In fact, JSON exists as long as JavaScript does, but it was not used before Ajax. [Over JSON op zich is nauwelijks een wetenschappelijke bron te vinden, ook W3C niet echt.]

JSON is based on a subset of Javascript, containing object and array. Objects contain pairs of property and value. Arrays contain values. A value could be a string, number, object array, true, false or null[15]. On average, JSON requires less characters, and so less bytes, than the same data in XML. Because it uses JavaScript syntax, it requires less parsing than XML when used in Ajax Applications [2].

```
[
    {
        "title" : "This is the title of a news-item",
        "article" : "This is the content of the article. Between...",
        "author" : { "author_name" : "van Wijngaarden",
                     "author_initials" : "T.D." },
        "publication_date" : "28 februari 2006",
        "keyword" : ["news", "paper", "test"]
    }
]
```

Figure 4-10. An example JSON file. Objects are enclosed by curly brackets ({ }) and arrays by squared brackets ([ ]). Name-value pairs are separated by a colon (:) and separated by comma's (except the last one). A sequence of news-items can be created by placing a comma behind the last curly bracket and repeating the news-item syntax (between the highest level curly brackets).

The example in Figure 4-10 shows almost the same information as the XML example in Figure 0-9. JSON is probably less readable, because there is no description of what the objects stand for, like the `news-item` element in XML tells you something about the content.

[11] http://www.w3.org/TR/REC-html40/sgml/dtd.html

JSON is also covered by server side technology like PHP.

5 Discussion

[Moet er bij dit aparte document ook een soort afsluiten? Bijvoorbeeld een discussion?]

6 References

[1] **W3C [World Wide Web Consortium].** XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition). [Online] 08 01, 2002. [Cited: 09 24, 2008.] http://www.w3.org/TR/xhtml1.

[2] **Zakas, Nicholas C, Jeremy, McPeak and Joe, Fawcett.** *Professional Ajax.* 2nd edition. Indianapolis : Wiley Publishing, Inc, 2007.

[3] **W3C [World Wide Web Consortium].** HTML 4.0 changes. [Online] 04 24, 1998. [Cited: 09 24, 2008.] http://www.w3.org/TR/1998/REC-html40-19980424/appendix/changes.html.

[4] **Crane, David, Bibeault, Bear and Sonneveld, Jord.** *Ajax in Practice.* Greenwich : Manning Publications Co., 2007.

[5] **W3C [World Wide Web Consortium].** Document Object Model (DOM). [Online] 01 19, 2005. [Cited: 09 26, 2008.] http://www.w3.org/DOM/.

[6] **Mozilla.** XMLHttpRequest. *Mozilla developer center.* [Online] 09 18, 2008. [Cited: 09 26, 2008.] http://developer.mozilla.org/en/XMLHttpRequest.

[7] **W3C [World Wide Web Consortium].** W3C Technical Reports and Publications. [Online] [Citaat van: 2008 09 26.] http://www.w3.org/TR/#last-call.

[8] —. The XMLHttpRequest Object. [Online] 10 26, 2007. [Cited: 09 26, 2008.] http://www.w3.org/TR/2007/WD-XMLHttpRequest-20071026/#xmlhttprequest.

[9] **Garrett, Jesse James.** Ajax: A New Approach to Web Applications. [Online] 02 18, 2005. [Cited: 10 01, 2008.] http://www.adaptivepath.com/ideas/essays/archives/000385.php.

[10] **Mahemoff, Michael.** *Ajax Design Patterns.* sl : O'Reilly Media, Inc., 2006.

[11] **O'Reilly, Tim.** What Is Web 2.0. [Online] 09 30, 2005. [Cited: 10 01, 2008.] http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html.

[12] **Best, David.** *Web 2.0: Next Big Thing or Next Big Internet Bubble?* Eindhoven : Technische Universiteit Eindhoven, 11 01 2006.

[13] **Noda, Tom en Helwig, Shawn.** *Rich Internet Applications: Technical Comparison and Case Studies of AJAX, Flash and Java based RIA.* Wisconsin : UW Business Consortium, UW E-Business Consortium, 2005.

[14] **W3C [World Wide Web Consortium].** Extensible Markup Language (XML) 1.0 (Second Edition). [Online] 10 06, 2000. [Cited: 10 08, 2008.] http://www.w3.org/TR/2000/REC-xml-20001006#sec-origin-goals.

[15] —. Introduction to SGML. *On SGML and HTML.* [Online] 1999. [Cited: 10 15, 2008.] http://www.w3.org/TR/REC-html40/intro/sgmltut.html#h-3.1.

[16] **JSON.org.** JSON The x in Ajax. *Introducing JSON.* [Online] 2006. [Cited: 10 08, 2008.] http://www.json.org/json.pdf.