



SCHOOL
OF
COMPUTER
SCIENCE

Information
Management
Group

HCW— SASWAT Technical Report 3, September 2008

Techniques for dynamically updating a Web page

Andy Brown, Caroline Jay and Alex Chen

Human Centred Web Lab
School of Computer Science
University of Manchester
UK

World Wide Web (Web) documents, once delivered in a form that remained constant whilst viewed, are now often dynamic, with sections of a page able to change independently (and not requiring a full page reload), either automatically or as a result of user interaction. In order to make these updates, and hence their host pages, accessible, it is necessary to detect when the update occurs and how it has changed the page, before determining how, when and what to present to the user. This can only be achieved with an understanding of the technologies used to achieve dynamic updates and the human factors influencing how people use them.

This report classifies the types of update that may be included on a page and reviews the techniques available to Web developers for achieving them. This is complemented by an analysis of Web pages, including the most popular pages and a random selection, which investigates how frequently some technologies are implemented. While JavaScript is used in nearly all sites, and Flash in about half, AJAX is used in some 20% of the current top 20 pages.

HCW

Human Centred Web

SASWAT

The aim of the SASWAT project is to develop a framework for mapping the competing dynamic micro content produced by Web 2.0 technologies to audio. The SASWAT web pages may be found at <http://hcw.cs.manchester.ac.uk/research/saswat/>.

SASWAT Reports

This report is in the series of HCW SASWAT technical reports. Other reports in this series may be found in our data repository, at <http://hcw-eprints.cs.man.ac.uk/view/subjects/saswat.html>. Reports from other Human Centred Web projects are also available at <http://hcw-eprints.cs.manchester.ac.uk/>.

Acknowledgements

This report forms Deliverable 3, Range and Scope of Dynamic Updates, for the SASWAT project. SASWAT is funded by the EPSRC, reference: EP/E062954/1.

Contents

1	Introduction	1
2	Range of Dynamic Updates	1
2.1	Initiating Event	1
2.1.1	Automatic Updates	2
2.1.2	User-initiated Updates	2
2.2	Page Changes	3
2.2.1	Add	3
2.2.2	Remove	4
2.2.3	Replace	4
2.2.4	Rearrange	4
2.3	Presentation	5
2.4	Semantics	5
2.5	Taxonomy of Updates	6
2.5.1	Timing	6
2.5.2	Page Effects	6
2.6	Design Patterns	6
3	Technologies	8
3.1	DOM	9
3.2	XMLHttpRequest	9
3.2.1	AJAX	10
3.3	DOM Load and Save	12
3.4	Inline Frames	13
3.5	Objects	13
3.6	Java	13
3.7	Flash	14
3.8	Plugin scripting	14
3.9	Image Wrappers	14
3.10	Static page changes	14
4	Usage of Updates	15
4.1	Method	15
4.2	Results	15
5	Summary	16

Human Centred Web Lab
School of Computer Science
University of Manchester
Kilburn Building
Oxford Road
Manchester
M13 9PL
UK

Corresponding author:
Andy Brown
tel: +44 (161) 275 7821
andrew.brown-3@manchester.ac.uk
<http://homepages.cs.manchester.ac.uk/~brown>

1 Introduction

Web pages that update dynamically are an integral part of ‘Web 2.0’, potentially offering benefits to both provider (e.g., updating a small section of a page can use less bandwidth than a full page refresh) and user. Dynamic updates (defined here as changes to a page which occur after its initial loading, but without a full page refresh) can allow the latter group to interact with Web pages in a way that resembles traditional desktop programs, indeed office ‘applications’ are available to use online¹. Accessibility of these updates is, however, thought to be poor, resulting in initiatives such as ARIA (Accessible Rich Internet Applications [3]), where markup allows page authors to indicate the function of controls, and the attributes of dynamic content (such as how important it is). The capabilities of screen readers (and, to a lesser extent, browsers) to deal with this content are discussed in Deliverable 2 of this project [1]. This report examines the nature of updates (Section 2), proposing a taxonomy by which to classify them (Section 2.5), and describes the technologies that may be used to enable them (Section 3). The prevalence of this technology is discussed in Section 4.

2 Range of Dynamic Updates

This section looks at the current range of Web-2.0 sites, exploring the variety of applications for which the technologies are implemented. In particular it considers situations where the content or the structure of a page may change while it is being viewed: dynamic updates. This analysis is intended to inform experiments which are designed to develop an understanding of how users interact with these updates, with emphasis on building a model of how attention is allocated when these updates occur. Accordingly, this subject is approached from the perspective of the user rather than the developer, examining how the update is seen by the user, not how it is implemented.

The analysis focuses on four attributes of an update. The first two describe the behaviour of the update: what causes it to occur (i.e., when does it happen), and what effect the update has on the page structure. The second two describe the update in terms of its content: its presentation and its semantics. These attributes are considered in the following sections, and are summarised in the taxonomy that follows.

2.1 Initiating Event

Updates can be split into two categories based upon the initiating event: they may occur automatically, independent from any user activity, or they may occur as a result of user activity. The distinctions between these are described in more detail below.

¹For example, see <http://docs.google.com/>

2.1.1 Automatic Updates

It has not been uncommon for web-pages to periodically and automatically reload. This may be achieved with ‘traditional’ techniques by using the following HTML meta tag:

```
<meta http-equiv=refresh content=10>
```

Inserting this into the head section of a page will cause the entire page to reload every 10 seconds. A common use for this has been for sites providing information on rapidly changing events, such as sports fixtures, enabling users to keep up-to-date with the latest score and commentary without needing to manually reload the page.

In contrast, Web-2.0 technologies, such as AJAX, allow individual sections of a page (‘micro-content’) to be refreshed periodically, without the need to reload the whole page. Automatic updates are those where this refresh is not dependent upon any user activity. Examples include:

- Digg Spy²: The page shows a list of recent stories from Digg.com; every 2 seconds a new story is added to the top of the list (the others move down, and the bottom one is removed)
- Yahoo! Finance³: The page displays the current values of various share indices (actually the values are approximately 20 minutes old). These are coloured green or red, depending upon whether the market is up or down for the day, and update every few seconds. Different indices update at different rates.

2.1.2 User-initiated Updates

While the updates discussed above occur automatically, many others are triggered by some action by the user. The triggering action may be one of:

- Mouse click on link, image, button, etc.
- Hovering the mouse over a link, image, button, etc.
- Typing (note that the keyboard may be used to emulate many mouse actions).

Examples include:

- The Yahoo! home page⁴ displays news in tabs (‘In the news’, ‘World’, ‘Local’, ‘Finance’) — clicking the tab title loads relevant content in the box below.
- Google Suggest⁵ is a version of the Google search engine interface that gives users suggestions for completing their query string. Each time a character is typed, a box shows 10 matches, along with the number of hits each of these would produce. This box is interactive, in the sense that the user may navigate (with the mouse or keyboard) to any of the suggestions and activate that suggestion as their search.

²<http://digg.com/spy>

³<http://finance.yahoo.com/intlindices?e=europe>

⁴<http://www.yahoo.com/>

⁵<http://www.google.com/webhp?complete=1&hl=en>

- National Rail Enquiries⁶ has a similar form completion to help with a constrained query. In this case, entries to the form should match a station: as the user types, the system displays all stations that match. Note that with some systems of this type (but not this example) suggestions are not displayed until the user has entered a minimum number of characters (often three).
- The Magic Seaweed⁷ surf report pages have a bar chart giving the predicted size of the swell over a few days. Below this are three map images showing swell size, swell period and wind speed at a given time. Hovering the mouse over a time period on the chart causes the images below to be replaced by maps appropriate for that time. This example is particularly interesting as moving the mouse across the chart effectively creates an animation in the maps below.
- Slide shows. When viewing a photo page on Flickr⁸, there is often a section of the page that shows the next and previous photos from that user. This has controls that allow viewers to scroll through the thumbnails of all photos — these are replaced dynamically, while the rest of the page remains unchanged. (Note that this section of the page may be expanded or contracted dynamically.)

It is probably worth noting that, from the user's perspective, the distinction between user-initiated and automatic is not quite as clear-cut as it might appear. Some user initiated updates might be triggered inadvertently (e.g., those initiated by mouse hover), or unexpectedly, e.g., users might expect a link to lead to a new page, whereas it actually updates their current page. A further complication is where a single action by the user might result in a series of changes, e.g., pressing a play button for a slide show. While the first update is clearly user-initiated, it is not clear how to classify subsequent changes. Experimental evidence will be needed to determine whether these types of update are attended to in the same way as other automatic changes.

2.2 Page Changes

As well as understanding when an update occurs, it is possible to consider it in terms of how it changes the structure of the page: is information added or removed, replaced or rearranged? We propose a second axis of classification, one that is orthogonal to the initiating event, by which updates may be categorised.

2.2.1 Add

Some user-requested updates will simply be a request for additional information, and this will be added to the page as they see it. In these cases, all the content that was visible before the update will still be visible. Note, however, that this often implies a certain degree of reconfiguration of the page. For example, a footer might

⁶<http://www.nationalrail.co.uk/>

⁷<http://magicseaweed.com/Hells-Mouth-Surf-Report/27/>

⁸For example, <http://www.flickr.com/photos/andybrown/2497320979/>

become further from the header (and hence, perhaps, move out of the user's view) as the area containing the new content expands. Examples include the following:

- BBC News⁹ has a section for 'Your local news, weather and sport', which expands to show more content. The content below moves down to accommodate.
- The Yahoo! home page¹⁰ has what appear like buttons on the top right ('weather', 'mail', 'horoscopes', etc.) — when the user hovers the mouse over one of these it expands to show a box below with relevant information.

2.2.2 Remove

This type of update is essentially the opposite of the previous type: information is removed from view. As an example of this and the previous type of update, one might envisage a control that causes a section of a page to expand, revealing additional information about a subject; this could be considered an adding update. When the user has read the information, they may activate the control again to contract the page section, hiding the information: this could be considered a removing update. Note that this type of update does not need to involve communication with the server. An example is the 'Your local news, weather and sport' panel on the BBC page, described above, which may also be hidden.

2.2.3 Replace

The third type of user-requested update covers all situations where some information on the page is replaced by some other information. Examples include slide shows, where an image and its title and caption might all be replaced, while the surrounding context (e.g., a summary of the show, thumbnails of all images) remains, or tabbed panels (such as those on the Yahoo home page described above).

This is an important category, within which updates may be placed on a scale reflecting the similarity of the original information with that which replaced it (see presentation and semantics, below). At one end of this scale are updates where the structure does not change, only a small piece of information. A typical use might be to present sports scores, as per automatic updates but giving the user control over when to request the update: here only a single number might change. At the other end of the scale, some tabbed panes may contain radically different information in each tab (note that the Yahoo tabs have identical formats to each other). A special type of replacement is where information is replaced by a rearrangement of the original. The situations in which this may be considered a replacement update versus a rearrange update are discussed in the next section.

2.2.4 Rearrange

Rearrangements are where the update does not actually change the information seen by the user, just the arrangement of it on the page. We may further refine this

⁹<http://news.bbc.co.uk/>

¹⁰<http://www.yahoo.com/>

category to distinguish between those situations where the user has control of the rearrangement (e.g., a classic ‘drag and drop’, such as iGoogle¹¹) and those where there is no explicit control. The latter case, exemplified by the re-ordering of the rows of a table, may be considered a special case of replacement. This is particularly pertinent in the case of tables, where the table might actually span multiple pages, meaning its rearrangement does change the content. It would not be desirable to classify single page table rearrangements differently from rearrangements of long tables spanning multiple pages.

2.3 Presentation

Attention is likely to be attracted differently according to how the information is presented [5]: movement [6], and certain colours [2], are known to be visually salient, and it is possible that different forms of information, such as images, text, or videos, are attended to differently. In addition to the intrinsic nature of the new information, how it differs from what was being viewed previously may also change its saliency. Some updates change only the meaning of the information (e.g., a stock value might change from 3.14 to 3.18, but presentation remains identical), while others might result in a different visual appearance.

2.4 Semantics

This attribute describes how relevant the changes are likely to be to the user, and thus depends upon the *content*, or semantics, of the update. This is expected to have a major impact on how important the user considers the information, and thus how much attention they allocate to it. Inevitably, this attribute is less concrete, and more difficult to divide into distinct classes, but should it prove important, it may be possible to allocate an update a level of ‘importance’, perhaps inferred from factors such as its position on the page.

As examples of the range of types of content, consider the following:

Yahoo Finance. Updates refresh the current values of the displayed stock indices.

It is reasonable to assume that this is the ‘primary’ information on the page, and that these updates have high relevance.

News tickers. News sites often have tickers giving the latest headlines. The relatively small size and discreet nature of these suggest that the page authors consider them of ‘secondary’ importance, so it can be assumed that these updates have relatively low relevance. (This is supported by eye-tracking studies [4].)

Google Suggest. Auto-suggest lists, such as Google Suggest, or National Rail Enquiries, update to give the user suggestions when completing a form. Sometimes they list all possible options (e.g., if entry is constrained), while in other cases, they simply provide suggestions. In either situation, the information is not essential for task completion, but is supplementary, aiding the user. The

¹¹<http://www.google.com/ig>

nature of these means that they are inevitably directly relevant to the users task.

These three examples give updates characteristic of three types that might be labelled primary, secondary, and supplementary relevance. These are vague categories, and assigning updates to them is unlikely to be an objective process, so the validity of using this concept in the taxonomy is not clear. It is therefore suggested at this stage, but not included below. It is possible, however, that experiments on how users allocate attention to updates might suggest a way of classifying updates on this axis.

2.5 Taxonomy of Updates

From the attributes described above, it is possible to tentatively propose the following classification system for updates. This is based purely on the behavioural characteristics of the updates, and not upon presentation or semantics. The taxonomy is based on a largely theoretical consideration about how updates may behave, and whilst backed up with examples, a comprehensive study of their applications on the Web might highlight areas for revision. Nevertheless, it should serve to inform investigations into how users allocate attention to updates, and thus into making these updates accessible.

2.5.1 Timing

What causes the update to occur?

Automatic Updates which occur independently from any user activity.

User-Initiated Updates which occur as a direct result of user activity.

2.5.2 Page Effects

What is the effect on the page of the update?

Add Information that the user has not seen before is added to the page.

Remove Information is removed from the page and not replaced.

Replace Existing content is changed — information that was visible before the user's action is no longer shown, but different information takes its place.

Rearrange The information contained in the page does not change, but its structure is modified in a way over which the user has direct control.

2.6 Design Patterns

The Yahoo! pattern library¹² is a set of design patterns, i.e., reusable solutions to common problems, for Web design. Typically, these patterns are described in terms

¹²<http://developer.yahoo.com/ypatterns/index.php>

of some information that needs to be conveyed to the user, and a method for doing this. For example, the *Auto Complete* pattern¹³ is described as follows (this is a summary — the library gives much more detail, including advice on when to use, why it works, and accessibility issues):

Problem Summary: The user needs to enter an item into a text box which could be ambiguous or hard to remember and therefore has the potential to be mis-typed.

Solution: As the user types, display a list of suggested items that most closely match what the user has typed. Continue to narrow or broaden the list of suggested items based on the user's input.

The library contains many examples of how, and why, dynamic updates can be used to make users Web tasks more efficient. The existence of libraries such as this is only likely to increase the usage of these updates. Here, these patterns are used to exemplify application of the taxonomy described above. Note that many of the patterns do not relate to dynamic updates (e.g., methods for presenting a user's status in a community) — these are not discussed here.

The largest class of patterns are the *transition* patterns, which are used to highlight change on a page, to ensure the user is aware of it. Different transition patterns may be used for different types of page change, and may be placed into the page-effects categories from the classification above. These may be used for either automatic or user-initiated updates. Table 1 shows how all relevant patterns fit into this taxonomy. Two transitions not included, *Brighten transition* and *Dim transition*, are used to highlight whether or not a region is active.

A second distinct group of pattern contains the four *invitation* patterns, which indicate to the user that a particular item on a page may be interacted with. These include *Cursor invitation*, where the mouse cursor changes to indicate interactivity, and *Tool-tip invitation*, where a tool-tip appears when an area is hovered over, and explains how it may be interacted with — these patterns may be applied to controls for any user-initiated update. The *Drop invitation* indicates suitable targets for drag and drop operations, and thus is applicable to user-initiated rearrange updates. The *Hover invitation* is similar to the tool-tip, but typically gives more information, and that information is added directly onto the web page. Thus it is implemented via an update that is user-initiated (hover) and may add-to or replace content.

The other patterns that use dynamic updates are as follows. *Auto complete* is user-initiated addition (for the first key press), or replacement (for subsequent key-presses); *Drag and drop modules* is user-initiated rearrangement; *Carousel* (a set of pictures from which the user must choose, not unlike a slide show) and *Module tabs* (e.g., Yahoo! home page) are both user-initiated replacement. In addition to these, which necessarily involve dynamic updates, the *Calendar picker*, *Rating an object*, *Writing a review*, and *Sign-in continuity* patterns may be implemented using dynamic updates.

The Yahoo! pattern library gives another view on dynamic updates — that of designers (rather than users or developers) who wish to communicate some informa-

¹³<http://developer.yahoo.com/ypatterns/pattern.php?pattern=autocomplete>

Pattern	Page Effect				Initiation	
	Add	Remove	Replace	Rearrange	Automatic	User
<i>Transitions</i>						
Animate				✓	✓	✓
Collapse		✓			✓	✓
Cross-fade			✓		✓	✓
Expand	✓				✓	✓
Fade-in	✓				✓	✓
Fade-out		✓			✓	✓
Self-healing		✓			✓	✓
Slide	✓				✓	✓
Spotlight			✓		✓	✓
<i>Invitations</i>						
Cursor	✓	✓	✓	✓		✓
Tool-tip	✓	✓	✓	✓		✓
Drop				✓		✓
Hover	✓		✓			✓
<i>Other</i>						
Auto complete	✓		✓		✓	
Drag and drop				✓		✓
Carousel			✓			✓
Module tabs			✓			✓

Table 1: Classification of Yahoo! patterns. Note that this classification is done according to the update, which the pattern may be loosely associated with (e.g., drop invitation, or which the pattern demands (e.g., auto complete).

tion to the user. In many cases this is done by dynamically updating the page. This brief introduction to the patterns has shown how the classification system might work, although precise categorisation of many patterns will depend upon the actual implementation.

3 Technologies

This section describes the techniques that developers can use to update their pages dynamically. The options include:

- XMLHttpRequest
- The Document Object Model (DOM) Level 3 Load and Save Specification
- Inline frames
- HTML 4 Objects
- Java Applets
- Adobe Flash Player

- Plugin Scripting
- Image wrappers

These are discussed below, following a brief introduction to the Document Object Model (DOM).

3.1 DOM

The Document Object Model (DOM)¹⁴ is a way of representing documents as a hierarchy, and defines a standard way to manipulate and access HTML documents. It provides a platform and language neutral interface that allows programs or scripts to access and update the content and style of an HTML or XML document dynamically. DOM is divided into three parts: Core, HTML and XML. The Core interface can represent any structured document by providing a low-level set of objects, while both HTML and XML are higher level interfaces, used with the core interface for simpler access to specific types of documents.

The DOM specifications are divided into several levels, of which the first three are of most interest here:

- **Level 1:** This level concentrates on the document models (core, HTML, and XML). It is used to navigate and manipulate a document.
- **Level 2:** This level concentrates on the style sheet model. It is used to manipulate the style information.
- **Level 3:** This level concentrates on loading and saving the document as well as key events and event groups.

3.2 XMLHttpRequest

XMLHttpRequest¹⁵ is an application programming interface (API) that can be used by scripting languages (e.g., JavaScript¹⁶) to transfer XML (extensible markup language) to and from a web server over Hypertext Transfer Protocol (HTTP). The connection is independent from the connection by which the page was originally loaded, thus allowing data transfer to be asynchronous. The World Wide Web Consortium (W3C) also notes that the name is misleading on all three counts: it supports any text-based format of data, not just XML; it may also use HTTPS (HTTP over Secure Socket Layer); and it may be used for all activity involved with HTTP requests or responses for the defined HTTP methods (GET, POST, HEAD, PUT, DELETE, OPTIONS).

According to the W3C, for a user-agent to conform to XMLHttpRequest it must support some version of DOM Events and DOM Core, and must also support some version of the Window Object¹⁷. There are some differences between browsers in

¹⁴<http://www.w3.org/DOM/>

¹⁵<http://www.w3.org/TR/XMLHttpRequest/>

¹⁶<http://www.ecmascript.org/>

¹⁷The Window object “provides a global namespace for web scripting languages, access to other documents in a compound document by reference, navigation to other locations, and timers.”

the implementation of XMLHttpRequest, but these can be worked around by using a JavaScript wrapper.

3.2.1 AJAX

AJAX is a Web development technique based on XMLHttpRequest and JavaScript, allowing asynchronous updates. It uses a client-side scripting language (typically JavaScript) to get new data from the server (via XMLHttpRequest or IFrames) and to interact with the Document Object Model (DOM) of the page.

An example of AJAX can be used to demonstrate the technical aspects of its functioning. Introduced above, Google Suggest is a version of the Google search page that uses AJAX to display search suggestions when a user types in the search box. The suggestions are retrieved dynamically, and update with each keystroke. The essence of the code is that typing into the box invokes JavaScript to make an XMLHttpRequest, then takes the data (plain text) and generates an HTML table which is inserted into the Document Object Model of the page. To the user, this table appears just below the search box. Some of the more relevant JavaScript for this page is below.

From an AJAX perspective, the key function is the one that creates an XMLHttpRequest object, which may be used to communicate with the server. Note that this is complicated by the need to create these objects differently according to the browser.

```
function wa(){
  var a=null;
  try{
    a=new ActiveXObject("Msxml2.XMLHTTP");
  }catch(b){
    try{
      a=new ActiveXObject("Microsoft.XMLHTTP");
    }catch(c){
      a=null;
    }
  }
  if(!a && typeof XMLHttpRequest!="undefined"){
    a=new XMLHttpRequest;
  }
  return a;
}
```

The second function uses the XMLHttpRequest object to make a call to the server and has JavaScript to handle different potential responses.

```
function Ea(a){ if(l&&l.readyState!=0 && l.readyState!=4){ // if the
  request state is neither 'uninitialized' // or 'loaded', then
  abort. l.abort(); } l=wa(); // call above function to create
  new XMLHttpRequest if(l){ // if created properly
  l.open("GET",P+"&js=true&q="+a,true); // open communication
  l.onreadystatechange=function(){
    if(l.readyState==4&&l.responseText){ // script to handle
      incoming data // (depends upon http status code) } } };
  l.send(null); } }
```

In the case of Google Suggest, the response is plain text. For example, if the variable 'a' above was 'hello', the response would be:

```
window.google.ac.Suggest_apply(frameElement, "hello", new Array(2,
"hello kitty", "6,770,000 results",
"hello magazine", "21,100,000 results",
"hellogoodbye", "2,560,000 results",
"hellogoodbye lyrics", "1,180,000 results",
"hello world", "132,000,000 results",
"halloween", "3,360,000 results",
"hello holidays", "4,300,000 results",
"hello lyrics", "5,140,000 results"),
new Array(""));
```

Another part of the script modifies the page DOM so that a table, which is empty before the user starts typing, has its contents modified. The resulting HTML is shown below (simplified for ease of reading).

```
<tbody>
  <tr>
    <td>hello kitty</td>
    <td>6,770,000 results</td>
  </tr>
  <tr>
    <td>hello magazine</td>
    <td>21,100,000 results</td>
  </tr>
  <tr>
    <td>hellogoodbye</td>
    <td>2,560,000 results</td>
  </tr>
  <tr>
    <td>hellogoodbye lyrics</td>
    <td>1,180,000 results</td>
  </tr>
  <tr>
    <td>hello world</td>
    <td>132,000,000 results</td>
  </tr>
  <tr>
    <td>halloween</td>
    <td>3,360,000 results</td>
  </tr>
  <tr>
    <td>hello holidays</td>
    <td>4,300,000 results</td>
  </tr>
  <tr>
    <td>hello lyrics</td>
    <td>5,140,000 results</td>
  </tr>
  <tr>
    <td colspan="2" colspan="2" >
      <span>close</span>
    </td>
  </tr>
</tbody>
```

3.3 DOM Load and Save

DOM Load and Save is the W3C recommendation¹⁸ for updating a page without a reload. This is a platform and language neutral interface, and may be implemented in any language. A JavaScript example is below¹⁹. This example creates an LSParser object to parse the XML from the file at URI “data.xml”: the content of this is placed in the (initially empty) div with id=“loadtarget”. For real applications, this code should also contain a check that the browser supports DOM 3 load and save.

```
function test_dom3ls_parse() {
    var impl = document.implementation;

    // create a parser object
    var lsp = impl.createLSParser(DOMImplementationLS.MODE_SYNCHRONOUS, null);

    // parse the data at the given URI
    var resultdoc = lsp.parseURI("data.xml");

    // import parsed XML into current document
    var resultelem = resultdoc.documentElement;
    var elem = document.importNode(resultelem, true);
    document.getElementById('loadtarget').appendChild(elem);
}
```

An extract of relevant HTML code to use this is below - when the user clicks on the link, data from data.xml is inserted into the div.

```
<p>
  <a href="javascript:;" onclick="test_dom3ls_parse()">
    Click to test the DOM3 Load and Save parser!</a>
</p>

<div id="loadtarget"></div>
```

For example, if data.xml contained:

```
<?xml version="1.0"?>
<div xmlns="http://www.w3.org/1999/xhtml">
  <span style="color:green;">Test successful!</span>
</div>
```

Then after clicking the link, the document would be:

```
<p>
  <a href="javascript:;" onclick="test_dom3ls_parse()">
    Click to test the DOM3 Load and Save parser</a>
</p>

<div id="loadtarget">
  <div xmlns="http://www.w3.org/1999/xhtml">
    <span style="color:green;">Test successful!</span>
  </div>
</div>
```

¹⁸<http://www.w3.org/TR/DOM-Level-3-LS/load-save.html>

¹⁹A version of this example is at http://hcw.cs.manchester.ac.uk/research/saswat/experiments/techTests/dom3ls_parse.xhtml.

It is not clear how well supported this technique is by browsers (the above example works in Opera 9.23, but not Internet Explorer 7 or Firefox 3.0.1), nor how well-known this is by developers.

3.4 Inline Frames

Inline Frames (IFrames) allow a web page to contain another page. The contents of the `<iframe>` are defined by its `src` attribute, and may be changed by a script, or by the user activating a link. In the latter case, the link must specify its target as the id of the iframe; clicking the link will then replace the iframe content with whatever the link pointed to.

It is also possible to use a hidden frame to communicate with a server²⁰. In this case, the iframe is made invisible (by giving it zero height and width) and made to load a page from the server containing a script (e.g., when a link is clicked). This script passes data to another script on the host page; this then displays the data somehow (JavaScript alert, modification of host DOM, etc). Extra functionality may be gained by passing data to the server page, either from a form, or encoded in its URL.

3.5 Objects

Objects were introduced in HTML 4, and allow cascading. For example, if the object is a movie, there may be another object embedded within that is a picture, and within that may be some descriptive text. Then, if the browser can display movies, it does so, otherwise it displays the picture; if that is not possible only the text is shown. An object need not be multimedia, but might just be HTML, in which case it can behave in a similar (but not identical²¹) fashion to iframes. As with iframes, the contents of an object may be changed by user action, or automatically.

The following HTML example, shows an object that displays the contents of `anotherPage.html`. When the links to some data or the ticker are followed, the contents of the object change accordingly. If the browser cannot display the contents, it just displays the text "An object".

```
<p>Select contents of object below by clicking a link:  
<a target="ob" href="data.xml">some XMLdata</a> or  
<a target="ob" href="ticker.html">news ticker</a>.</p>  
  
<object id="ob" data="anotherPage.html">An object</object>
```

3.6 Java

Java Applets may communicate with a server. This can happen directly to a Common Gateway Interface (CGI) program on the server through the HTTP protocol, or indeed any protocol. Alternatively, an applet may connect to a Java HTTP Servlet

²⁰See <http://developer.apple.com/internet/webcontent/iframe.html>

²¹See <http://www.w3.org/TR/html401/struct/objects.html##embedded-documents>

running on the server. Finally, Applets may communicate using the Java techniques of Remote Method Invocation (RMI), Java Database Connectivity (JDBC), or Common Object Request Broker Architecture (CORBA).

3.7 Flash

Adobe's Flash allows relatively sophisticated user interaction, and can also exchange data with the server. This data may be text (including XML), images or many other forms, and can be loaded dynamically using the `LoadVars()` class.

3.8 Plugin scripting

`LiveConnect`²² is an API to allow Java and JavaScript to call each other's methods. `LiveConnect` appears to have been superseded in Mozilla by an extension to the Netscape Plugin API (plugins include PDF readers, Java applets, Flash players) known as `NPRuntime`²³. This allows conforming plugins to access browser objects such as the window object or the DOM element that loaded the plugin, and thereby influence the DOM of the surrounding page. Similarly, plugin methods may be called from a script on the page. If a plugin is able independently to communicate with a server, this technology allows it not just to change itself, but also to dynamically manipulate the rest of the page.

3.9 Image Wrappers

This old method is much more complicated than the methods above, and appears to be rarely used. It is based on the ability of JavaScript to modify the source of an image. The image used is often a blank single pixel image, so as to remain hidden.

3.10 Static page changes

The techniques above concentrate on technologies that allow communication between the browser and the server so that new information can be downloaded and presented to the user without needing to refresh the whole page. While this is a common phenomenon (see below), it is actually much more common that the content of a page appears to change, despite no browser-server communication taking place. In these cases, all content is downloaded with the original page, but JavaScript (typically) is used to modify the appearance.

The BBC Homepage²⁴ has several examples of this: there are plus and minus buttons for several of the sections that allow the user to display more or fewer stories. A fixed number of these are part of the page source; the buttons merely control how many are visible. The functionality is provided by JavaScript (if disabled, a default of three stories are shown, and the buttons have no effect).

²²See <http://developer.mozilla.org/en/docs/LiveConnect>

²³http://developer.mozilla.org/en/docs/Gecko_Plugin_API_Reference:Scripting_plugins

²⁴<http://bbc.co.uk/>

4 Usage of Updates

This section explores how updates are actually used on Web sites: which technologies are deployed, how many sites use AJAX, and how this is changing. It reports on an analysis of historical and current Web pages that examined the technologies used.

4.1 Method

Four categories of sites were examined, two of which gave information over the last 10 years, and two of which gave a snapshot of the current situation. The selections were designed to compare the most popular sites with a random selection from the Web as a whole. Popularity was determined using the Alexa rankings²⁵; the top 20 pages from June 2008 were analysed to gain historical data, while for the snapshot of the current situation the most popular 500 pages were used. A random selection of sites (taken from Google directories) were used to determine the state of the Web as a whole, with 500 pages being analysed each year for historical data, and 5000 for the current situation. For the historical data, the same sites were used over the entire period (i.e., the data does not necessarily represent the most popular sites for a given year). Since these pages were not always archived in a given year, there are some missing data; values are therefore presented as a percentage of the available pages.

For the snapshot data, searches were performed on the page source ((X)HTML) and all associated JavaScript files. For historical data, it was not always the case that JavaScript files were archived (the Internet archive selects URLs by popularity), so it is possible that searches may return fewer results than was actually the case. A further limitation is if a single JavaScript file is required by several pages, it might contain functions that are not used by the particular page being examined but will nevertheless be searched, and might return a false-positive result.

4.2 Results

JavaScript is, and has been for at least 10 years, very commonly used by Web designers. The analysis reveals that it has been used in at least 90% of the top 20 sites and more than 80% of the random 500 pages for all of the last 10 years. The data are shown in Figure 1.

Analysis of the data reveals that Flash is becoming increasingly common. Currently 8 (40%) of the Alexa top 20 sites and 18% of the random 5000 sites use Flash; usage is showing a steady increase (Figure 2). Note that Flash is often used for animations and videos, and it is likely that only a small proportion give similar functionality to the dynamic updates under consideration here.

AJAX techniques are also increasing in frequency: XMLHttpRequest is found in around 20% of most popular pages (20% of the top 20; 22% of the top 500), and in around 3% of the random selection of 5000 pages (Figure 3). In contrast, a search for “createLSParser” found no matches over any of the page selections; this suggests

²⁵<http://www.alexa.com/>

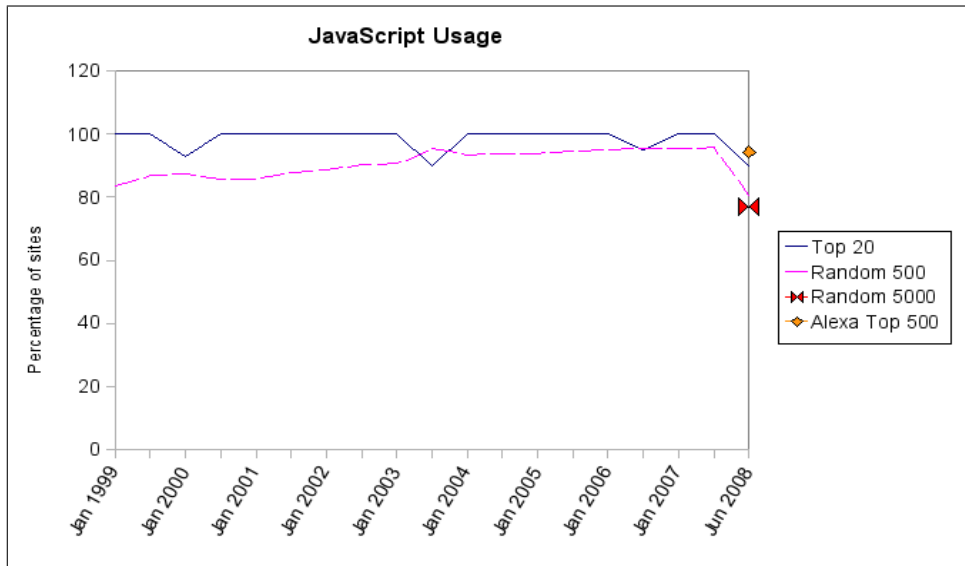


Figure 1: JavaScript usage 1998 to 2008.

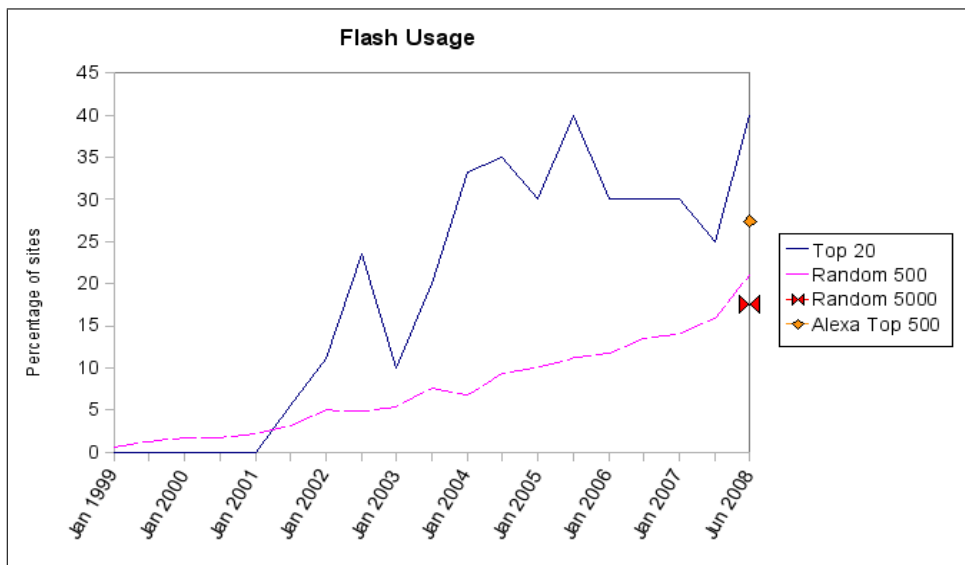


Figure 2: Flash usage 1998 to 2008.

that the W3C recommended DOM 3 Load and Save method is not in common use; unsurprising given the lack of browser support.

5 Summary

Web 2.0 technology allows Web pages to change dynamically, with updates of micro-content rather than an entire page. Changes may occur automatically or as a result of the user's interaction with the page. They may affect the page in a variety of ways, by adding, removing, replacing or rearranging content. These behavioural charac-

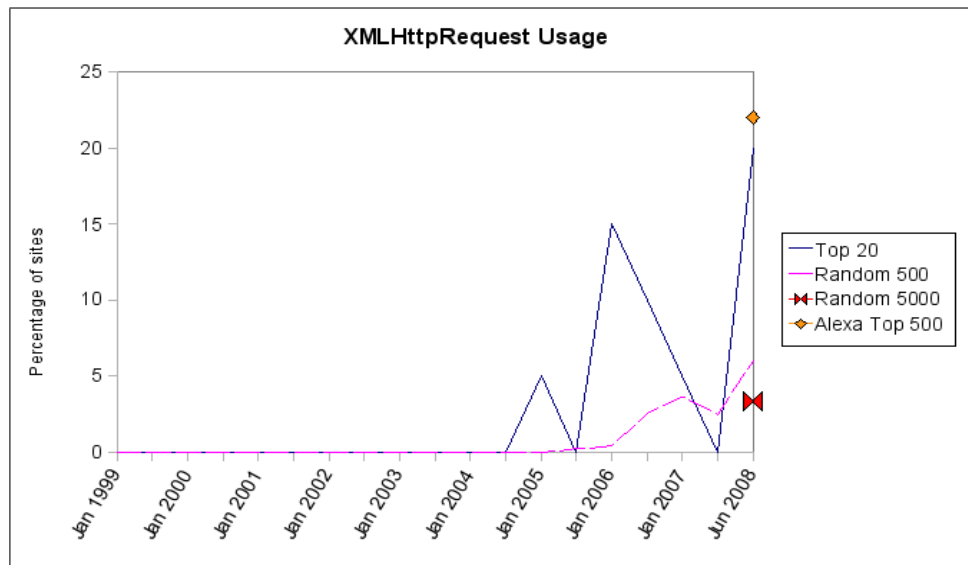


Figure 3: XMLHttpRequest usage 1998 to 2008.

teristics have been classified in a taxonomy, which is designed to inform experiments investigating how sighted users interact with updates. The intrinsic (presentational and semantic) attributes have also been discussed and, while not sufficiently objective to be included in the taxonomy, these should also be considered when exploring how users allocate attention to updates.

Many techniques are available to developers wishing to implement Web pages that have dynamically updating micro-content. An increasingly common technology used to achieve dynamic updates is XMLHttpRequest, (AJAX) but other techniques may be used to achieve the same result.

References

- [1] A.J. Brown and C. Jay. A review of assistive technologies: Can users access dynamically updating information? Technical report, University of Manchester, 2008.
- [2] R. Carmi and L. Itti. Visual causes versus correlates of attention selection in dynamic scenes. *Vision Research*, 46:4333–4345, 2006.
- [3] Becky Gibson. Enabling an accessible web 2.0. In *W4A '07: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A)*, pages 1–6, New York, NY, USA, 2007. ACM.
- [4] C. Jay and A.J. Brown. User review document: Results of initial sighted and visually disabled user investigations. Technical report, University of Manchester, 2008.

- [5] Derrick Parkhursta, Klinton Law, and Ernst Niebur. Modeling the role of salience in the allocation of overt visual attention. *Vision Research*, 42(1):107–123, January 2002.
- [6] H. Petersen and J. Nielsen. The eye of the user: the influence of movement on users' visual attention. *Digital Creativity*, 13(2):109–121, 2002.