CHAPTER 8

Learning Ajax/JavaScript widgets and their APIs

Context and Overview of the Chapter

In the context of contemporary Web 2.0 development, Ajax is a big deal, including the ways it allows you to mashup data and services in new and easier ways. Ajax exploits the fact that modern web browsers are programmable and that they are inherently network applications. In addition to sending static HTML to web browsers, programmers can send JavaScript programs to run in the web browser. What can be done with this type of JavaScript-based client-side programming?

* More dynamic interaction without having to reload the entire web page. This capability can be used, for instance, for drop-down menus and other widgets that we are used to having on the desktop.
* In particular, JavaScript can be used to get data from a server without having to reload the entire web page. Developers and designers have increasingly exploited this capacity, collectively known as Ajax techniques.
* Widgets can be created and deployed by other people. These widgets can be used to combine data and services and shown to other people. (Google Maps is the single most mashed up API/service on the public internet.)

Using JavaScript and DHTML are not new phenomena -- but has become extremely popular under the banner of Ajax. Jesse James Garrett says it well:[1]

> *But seeing Ajax as a purely technological phenomenon misses the point. If anything, Ajax is even more of a sea change for designers than it is for developers. Sure, there are a lot of ways in which developers need to change their thinking as they make the transition from building traditional web applications to building Ajax applications. But for those of us who design user experiences, the change brought about by Ajax is even more profound.*

[1] Bruce Perry. *Ajax Hacks.*(O'Reilly & Associates, 2006). http://proquest.safaribooksonline.com/0596101694/ID-d45e101771-Foreword

What One Would Ideally Learn/Know about the Subject

Ajax, along with all its attendant use of JavaScript and the modern web browser, is a rich subject, as can be seen in by the myriad books that have been published recently on the subject. I would like to put Ajax in the larger context of the programmable Web browser. To become a master programmer of the web browser, one would understand:

* both how an "ideal" W3C DOM-standards compliant browser works and how various browsers actually work in various areas: how JavaScript is implemented, object models behavior, CSS, events, etc.
* JavaScript-based APIs, widgets such as Google maps — what are they, how to use any all.
* non-browser environments for JavaScript, such as Google Gadgets, Yahoo Widgets, Adobe Acrobat
* extension mechanisms in browsers (Firefox addons, Safari , IE , Opera)
* JavaScript and browser debugging tools like Firebug
* JavaScript libraries: how they relate and what can be intermixed — and which ones are tied to which web programming frameworks.
* what people have done already on all these fronts using JavaScript and remixing the browser
* how to write JavaScript and JavaScript widgets that can be reused by other people, including cross-platform JavaScript
* ideas of what you can do in terms of mashups

Fortunately, one does need to know all these things to get started. This chapter concentrates on helping you use Ajax to mashup data and services by:

* pointing out the Ajax-based parts of Flickr -- and other -- and contrast the old-style of web development that involved the reloading of an entire page to a new-style development, in which more logic is pushed to the client, opening up more portunities for integration.
* pointing out ways to see the difference between Ajax and non-Ajax apps by asking readers to turn off JavaScript in their browser and see what happens to flickr.com and mapping applications and Gmail.
* Instead of looking in depth at all browers -- old and new, we focus on the lastest versions of Firefox and ask you to learn it well.
* introducing the Yahoo UI Library as a specific example of various JavaScript widget libraries
* using one of the JavaScript widget libraries (the current selection is the Yahoo UI library), shows how to use a widget (e.g., the TreeView widget)
* showing to write a basic Greasemonkey script as a way of mashing up services and data in the browser.

Why Difference Does Ajax Make? Turning JavaScript off in Flickr and Google Maps to See Why

To convince yourself that JavaScript is at work in web applications such as Flickr, Google Maps and Gmail, you can turn off JavaScript in your browser and see what changes in the behavior of the application.

How to turn off JavaScript in your browser:

* In Firefox: Tools -> Options -> Content -> Enable JavaScript (please uncheck)
* In Internet Explorer: See detailed instructions at http://www.umanitoba.ca/acn/support/websites/JavaScript/ie6.html Basically, check Tools-> Internet Options -> Security -> Custom Level -> Scripting -> Active Scripting
* In Opera: Tools->Quick Preferences->Enable JavaScript

Once you have JavaScript turned off, notice the following changes:

* In Flickr, you will see the message " To take full advantage of Flickr, you should use a JavaScript-enabled browser and install the latest version of the Macromedia Flash Player." All the buttons on top of the picture no longer function. Instead of clicking on the title, description, and tags to start editing them, you have to click on a link ("Edit title, description, and tags") before doing so.
* Notice that some apps will gracefully support non-JavaScript enabled browsers -- particularly Google Maps. With JavaScript turned off, you no longer the pan and zoom new-style maps but an old-style map which provides links to move in N/S/E/W or to change the zoom level.

There are interesting and important issues around usability/accessibility. Many computers, including mobile devices, will not be using JavaScript. How should apps gracefully deal with browsers that don't use JavaScript?

Useful Tools: Learning Firebug, DOM Inspector, and JavaScript Shell

Probably useful to have a screenshot.

In learning Ajax and widgets/applications based on Ajax, I recommend using Firefox, the DOM Inspector and the JavaScript Shell (bookmarklet) to manipulate "live" web pages and widgets. The combination allows for "live" interaction with the little apps; you can load a running map, analyze the details of how it is working while running, and issue commands that take immediate effect.

Moreover, the right tools can help you make sense of complicated stuff like Browser Object Model (BOM), Document Object Model (DOM) by letting you interact with the browser, test code out, etc. without having to read a book alone. I will use these tools the rest of the chapter to undergird this experimental/reverse-engineering approach.

DOM Inspector

The DOM Inspector allows you to look at the HTML DOM as a tree and make changes in that DOM. The DOM Inspector comes with Firefox, but not with the default installation options on Windows.[2]

Firebug extension for Firefox

I highly recommend the installation of the Firefox extension called Firebug. Firebug allows you to " edit, debug, and monitor CSS, HTML, and JavaScript live in any web page."[3]

To install this extension in Firefox, click on `http://www.getfirebug.com/releases/firebug1.0-current.xpi` , give permission to Firefox to install the extension, and then install the extension.

Important features of Firebug include:

* ability to view "live source" (that is, what the HTML of the DOM is at the moment, not when the original source)
* "instant HTML editing" -- you can edit the HTML in Firebug and see the changes reflected on the page

What can Firebug be useful for:

* learn CSS -- change style elements and see how things change. (e.g., cascading is shown, overriden properties are struck out!)
* track uses of the XMLHTTPRequest object -- you can use Firebug to see that something is being loaded.
* use "Inspect mode" to mouse over piece of a web page and see the corresponding HTML.

Firefox Shell

The JavaScript Shell is a bookmarklet to use in Firefox that you can find at http://www.squarefree.com/shell/ With it, you can run snippets of JavaScript code that will execute in the context of the page in which you invoked the shell. To install the Shell, just drag it to your links toolbar in Firefox. To invoke the JavaScript shell, put the page you want in the foreground and click on the JavaScript Shell bookmarklet.

---

[2] http://kb.mozillazine.org/DOM_Inspector

[3] http://www.getfirebug.com/

Other Tools

There is "Firebug lite" to use with IE, Opera, and Safari.[4] You might consider using the Venkman JavaScript Debugger.[5]

JavaScript libraries

Instead of programming directly for a specific browser, it is often useful to work at a higher level of abstraction by working with a JavaScript library. There are many cross browser differences and fine technical details that are best left to the JavaScript specialist. JavaScript libraries typically allow you to program the browser as a generic entity rather than having to attend for the differences among browsers.

I think that ideally there would be one obvious choice for an excellent JavaScript library, and anyone would use it. The current situation is that there are many JavaScript libaries -- and it is not at all obvious how they compare. For instance, Simon Willison thinks that the big four are:[6]

* Dojo
* Mochikit
* Prototype/Scriptaculous
* Yahoo UI Library

Others have pointed out OpenLaszlo[7]

In this chapter and those that follow, I will concentrate on using YUI.

YUI Widgets

There are a lot of technical issues wrapped up in AJAX. The way I recommend using AJAX is to use a library. There are many libraries to choose from. The one I will use here is the Yahoo UI library as the place to start: `http://developer.yahoo.com/yui/`

Read the "Yahoo! UI Library -- Getting Started"[8] The best way is to learn about the library is to look around . `http://developer.yahoo.com/yui/` and try the various examples. Also use the Javascript shell and Firebug extension to learn how things work.

Try out the pieces on the Yahoo site (e.g., the Treeview conroller (`http://developer.yahoo.com/yui/examples/treeview/index.html`) Try some commands on the Javascript shell. I will walk you through the use of two YUI widget: the calendar and the TreeView widget.

---

[4] http://www.getfirebug.com/lite.html

[5] http://www.mozilla.org/projects/venkman/

[6] http://simonwillison.net/2006/Jun/26/libraries/

[7] http://www.openlaszlo.org/

[8] http://developer.yahoo.com/yui/#start

YUI Calendar

The YUI Calendar component is " a UI control that enables users to choose one or more dates from a graphical calendar presented in a one-month ("one-up") or two-month ("two-up") interface."[9] To learn how to use it, you can try out the calendar examples[10] and refer to the API documentation[11] to get a list of methods for the widget.

You can use the Firebug extension of JavaScript shell to get a feel for how to program the component:

1. Go to `http://developer.yahoo.com/yui/examples/calendar/quickstart/solution.html`
2. Click on a date.
3. In the console of Firebug extension or JavaScript shell, type

```
YAHOO.example.calendar.cal1.getSelectedDates()
```

to get back the date you selected

4. Try out other methods to see how the calendar works:

```
YAHOO.example.calendar.cal1.hide()
YAHOO.example.calendar.cal1.show()
YAHOO.example.calendar.cal1.setMonth(1)
YAHOO.example.calendar.cal1.render()
```

YUI TreeView

The TreeView component is "a UI control that can be used to create a variety of hierarchical dynamic tree structures."[12] In addition to reading the API documentation for the TreeView[13], you can use Firebug and the JavaScript Shell to experiment with the component. To do so:

1. Go to `http://developer.yahoo.com/yui/examples/treeview/default.html?mode=dist`
2. In the console of Firebug extension or JavaScript shell, type

```
tree.expandAll()
tree.collapseAll()
```

to expand and collapse the tree respectively.

---

[9] http://developer.yahoo.com/yui/calendar/

[10] http://developer.yahoo.com/yui/examples/calendar/index.html

[11] http://developer.yahoo.com/yui/docs/YAHOO.widget.Calendar.html

[12] http://developer.yahoo.com/yui/treeview/

[13] http://developer.yahoo.com/yui/docs/YAHOO.widget.TreeView.html

Installing YUI on your own host account

To use YUI in your own applications, you should set up YUI to use on your own web-hosting. I will use as a concrete example `http://examples.mashupguide.net`, which is mapped to the UNIX directory `~/examples.mashupguide.net`. Substitute your own values. My goal is to set up YUI so that it is accessible at `http://examples.mashupguide.net/lib/yui/`.

1. Download the library to your machine or your web hosting. Go to http://developer.yahoo.com/yui/download/ Alternatively, if you want to use curl and download the version used here (2.2a) -- the latest as of writing -- , you can invoke
   `curl -O http://internap.dl.sourceforge.net/sourceforge/yui/yui_2.2.0a.zip`

2. Unzip and copy the files to the right location In my case, I unnzipped and copied the unzipped directory (which is named yui) to `/home/rdhyee/examples.mashupguide.net/lib/yui` -- which maps to the yui directory `http://examples.mashupguide.net/lib/yui/` The important part of the library for runtime purposes is the `yui/build` directory.

With the files in your own directory, you can, for instance, look at the calendar example on your own server.
`http://examples.mashupguide.net/lib/yui/examples/calendar/quickstart/solution.html`

Learning Google Maps as a well-know Ajax component

Just as there are different UI elements packaged up in one of the major JavaScript libraries, vendors have already started to create reusable JavaScript components. The most famous of these Ajax widgets is Google Maps. In this section, we look at how to embed a Google map using the Google Maps API. The online documentation on how to get started with the maps at the Google website are good. [14] The approach given there, one that I can certainly recommend, is to give you source code for increasingly more complex examples, that you either copy and paste to your own site. I can recommend that approach.

What I offer here is setting up a simple map and then using the JavaScript shell to let you work with a "live map" so that you can invoke a command and see an immediate response. The intended effect is that you see the widgets as dynamic programs that respond to commands, whether that command comes in a program or from you entering that command one by one.

Getting started with Google Maps and JavaScript Shell

We will use the Google Maps API to make a simple map.

[14] http://www.google.com/apis/maps/documentation/#Introduction

1. Make sure you have a public web directory to host your map and know the URL of that directory. Any Google Map that uses the free, public API, needs to be publicly visible

2. Go to the signup page for a key to access the Google Maps.[15] You will need a key for any given domain in which you host Google maps. (It is through these keys that Google regulates the use of the Google maps API.)

3. Read the terms of service[16] and if you agree to it, enter the URL directory on the host that you want to place your test file. For example, in my case, the URL is `http://raymondyee.dreamhosters.com`. When I type in this URL, I get the following key: `ABQIAAAAdjiS7YH6Pzk2NrliO2b5xxTO2tB7zlPmV4JaKyl2exQNZXIJ_xTOHgLDumjgyGWFarOlCOJGtI6scw` Since your domain will be different, your key will be different. Take down that key. (You can, in fact, enter `http://raymondyee.dreamhosters.com` and you should get the same key as I list here to confirm that you are doing the right thing.)

4. Copy and paste the HTML code into your own page on your web hosting directory. You should get something like my own example:[17]

```
<!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">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>Google Maps JavaScript API Example</title>
    <script
src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=ABQIAAAAdjiS7YH6Pzk2NrliO2
b5xxTO2tB7zlPmV4JaKyl2exQNZXIJ_xTOHgLDumjgyGWFarOlCOJGtI6scw"
      type="text/JavaScript"></script>
    <script type="text/JavaScript">

    //<![CDATA[

    function load() {
      if (GBrowserIsCompatible()) {
        var map = new GMap2(document.getElementById("map"));
        map.setCenter(new GLatLng(37.4419, -122.1419), 13);
      }
    }

    //]]>
    </script>
  </head>
  <body onload="load()" onunload="GUnload()">
```

[15] http://www.google.com/apis/maps/signup.html

[16] http://www.google.com/apis/maps/terms.html

[17] http://www.google.com/maps/api_signup?url=http%3A%2F%2Fraymondyee.dreamhosters.com


```
    <div id="map" style="width: 500px; height: 300px"></div>
  </body>
</html>
```

5. Now make one modification to the example by adding the line

```
window.map = map
```

after

```
var map = new GMap2(document.getElementById("map"));
```

to expose the map object to the JavaScript shell utility[18]:

```
<!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">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>Google Maps JavaScript API Example</title>
    <script
src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=ABQIAAAAdjiS7YH6Pzk2Nrli02
b5xxTO2tB7zlPmV4JaKyl2exQNZXIJ_xTOHgLDumjgyGWFarOlCOJGtI6scw"
      type="text/JavaScript"></script>
    <script type="text/JavaScript">

    //<![CDATA[

    function load() {
      if (GBrowserIsCompatible()) {
        var map = new GMap2(document.getElementById("map"));
        window.map = map
        map.setCenter(new GLatLng(37.4419, -122.1419), 13);
      }
    }

    //]]>
    </script>
  </head>

  <body onload="load()" onunload="GUnload()">
    <div id="map" style="width: 500px; height: 300px"></div>
  </body>
</html>
```

[18] http://raymondyee.dreamhosters.com/book/map/google.map.1.html

6. Invoke the JavaScript shell for your map, by hitting the JavaScript shell bookmarklet in the context of your map. Type in the following code fragments and see what happens. (Note that another approach is to modify your code directly with these code fragments and reload your page.) One can correlate the actions to the documentation for v2 of the Google Maps API.[19]

To return the current zoom level of the map (which goes from 0 to 17, 17 for the most detailed) (the reponse from the JavaScript shell is shown right :

```
map.getZoom()
13
```

To obtain the (latitude, longitude) of the center of the map:

```
map.getCenter()
(37.4419, -122.1419)
```

To center the map around the Campanile for UC Berkeley.

```
map.setCenter(new GLatLng(37.872035,-122.257844), 13);
```

You can pan to that location instead:

```
map.panTo(new GLatLng(37.872035,-122.257844));
```

To add a small map control (to control the zoom level) and the

```
map.addControl(new GSmallMapControl());
map.addControl(new GMapTypeControl());
```

To GMap keyboard navigation on, use:

```
window.kh = new GKeyboardHandler(map);
[object Object]
```

To fully zoomed out the map

```
map.setZoom(0)
```

To zoom in all the way: (which may go from 15-17)

```
map.setZoom(17)
```

To set the variable `maptypes` to an array holding 3 objects:

```
maptypes = map.getMapTypes()
[object Object],[object Object],[object Object]
```

To get the name of the first entry in maptypes:

```
map.getMapTypes()[0].getName()[20]
Map
To get the current map type, you can get the object and the name of that type object
map.getCurrentMapType()
[object Object]
map.getCurrentMapType().getName()
Map
```

To set the maptype to "satellite":

```
map.setMapType(maptypes[1]);
```

You can zoom one level in and out if you are not already at the max or min zoom levels:

```
map.zoomIn();
map.zoomOut();
```

To make an overlay, try this

```
point = new GLatLng (37.87309185260284, -122.25508689880371);
(37.87309185260284, -122.25508689880371)
```

---

[19] http://www.google.com/apis/maps/documentation/reference.html#GMap2

[20] 1 corresponds to "Satellite", while 2 corresponds to "Hybrid" map type

```
marker = new GMarker(point);
[object Object]
map.addOverlay(marker);
```

To make something happen when you click on the marker:

```
GEvent.addListener(marker, 'click', function() { marker.openInfoWindowHtml('hello');
});
[object Object]
```

There are many more things to explore, such as polylines and overlays and draggable points. To learn more, I certainly recommend the Google Maps API: Introduction[21] Note the assumed indicates that "this documentation is designed for people familiar with JavaScript programming and object-oriented programming concepts. You should also be familiar with Google Maps from a user's point of view."

Accessing Flickr via JavaScript

In Chapter 10, we will be building a mashup that integrates Flickr data and Google Maps within the browser context. That is, we will need to call the Flickr API from JavaScript within the browser (in true Ajax style.) Flickr provides JSON output to its API.[22] Let's jump into how it works:

1. Go to the Flickr Explorer to the flickr.photos.search page (http://www.flickr.com/services/api/explore/?method=flickr.photos.search) and do a search for 'flower' and set the per_page parameter to 3 (to make for a more manageable number of photos for now). Do not sign the call. Hit the "Call Method" button and grab the REST URL. Substitute the parameter with your own Flickr API key. You will get something like:

```
http://api.flickr.com/services/rest/?method=flickr.photos.search&api_key=e81ef8102a5
160154ef4662adcc9046b &tags=flower&per_page=3
```

   from which you get the Flickr rsp XML output with which you are familiar from previous chapters.

2. Let's now study the JSON output by tacking on the parameter format=json to the URL[23]. Now you get (in prettified rendition):

```
jsonFlickrApi( {
   "photos" : {
      "page" : 1, "pages" : 283266, "perpage" : 3, "total" : "849797", "photo" : [ {
         "id" : "397677823", "owner" : "28374750@N00", "secret" : "cab3f3db01",
"server" : "124", "farm" : 1, "title" : "DSC_0026", "ispublic" : 1, "isfriend" : 0,
"isfamily" : 0}
```

[21] http://www.google.com/apis/maps/documentation/#Introduction

[22] http://www.flickr.com/services/api/response.json.html

[23] http://api.flickr.com/services/rest/?method=flickr.photos.search&api_key=e81ef8102a5160154ef4662adcc9046b &tags=flower&per_page=3&format=json

```
      , {
         "id" : "397677820", "owner" : "28374750@N00", "secret" : "f70cf0bb19",
"server" : "174", "farm" : 1, "title" : "red flowers", "ispublic" : 1, "isfriend" :
0, "isfamily" : 0}
      , {
         "id" : "397677553", "owner" : "37015070@N00", "secret" : "7329c71748",
"server" : "158", "farm" : 1, "title" : "Rose In Vase", "ispublic" : 1, "isfriend" :
0, "isfamily" : 0}
      ]}
   , "stat" : "ok"}
)
```

Note that what is being returned: a piece of JavaScript containing the function named jsonFlickrApi with a single parameter, a JavaScript object that represents the photos.

3. Let's now write a bit of HTML and JavaScript to test out how to write JavaScript to access the various pieces of the Flickr response:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Flickr JSON</title>
  </head>
<script>
function jsonFlickrApi(rsp) {
  window.rsp = rsp;
}
</script>
<script
src="http://api.flickr.com/services/rest/?method=flickr.photos.search&api_key=e81ef8
102a5160154ef4662adcc9046b &tags=flower&per_page=3&format=json"></script>
</html>
```

You can load this page in your browser and invoke the JavaScript shell to learn a few key lines to access parts of the response:

```
props(rsp.photos)
Fields: page, pages, perpage, photo, total
rsp.photos.perpage
3
rsp.photos.photo[0].id
397694840
```

4. Let's now have jsonFlickrApi produce a display of the photos:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Flickr JSON</title>
```

```
  </head>
<body>
  <script>
  function jsonFlickrApi(rsp) {
    window.rsp = rsp;
    var s = "";
    // http://farm{farm-id}.static.flickr.com/{server-id}/{id}_{secret}_[mstb].jpg
    // http://www.flickr.com/photos/{user-id}/{photo-id}
    s = "total number is: " + rsp.photos.photo.length + "<br/>";

    for (var i=0; i < rsp.photos.photo.length; i++) {
      photo = rsp.photos.photo[i];
      t_url = "http://farm" + photo.farm + ".static.flickr.com/" + photo.server +
"/" + photo.id + "_" + photo.secret + "_" + "t.jpg";
      p_url = "http://www.flickr.com/photos/" + photo.owner + "/" + photo.id;
      s +=  '<a href="' + p_url + '">' + '<img alt="'+ photo.title + '"src="' +
t_url + '"/>' + '</a>';
    }
    document.writeln(s);
  }
  </script>
  <script
src="http://api.flickr.com/services/rest/?method=flickr.photos.search&api_key=e81ef8
102a5160154ef4662adcc9046b &tags=flower&per_page=50&format=json"></script>
</body>

</html>
```

5. The above example is simple but limited by the fact that loading JSON data immediately calls the function jsonFlickrApi. We want to change the name of the variable created and also not have a callback. Adding the `nojsoncallback=1` parameter[24] gets you:

```
{
   "photos" : {
      "page" : 1, "pages" : 283353, "perpage" : 3, "total" : "850057", "photo" : [ {
         "id" : "397750427", "owner" : "98559475@N00", "secret" : "f59b1ae9e1",
"server" : "180", "farm" : 1, "title" : "sakura", "ispublic" : 1, "isfriend" : 0,
"isfamily" : 0}
      , {
         "id" : "397750433", "owner" : "81222973@N00", "secret" : "0023e79dff",
"server" : "133", "farm" : 1, "title" : "just before spring", "ispublic" : 1,
"isfriend" : 0, "isfamily" : 0}
      ]}
   , "stat" : "ok"}
```

---

24 http://api.flickr.com/services/rest/?method=flickr.photos.search&api_key=e81ef8102a5160154ef4662adcc9046b%20&tags=flower&per_page=3&format=json&nojsoncallback=1

which is just the object. Using the `jsoncallback` parameter to change the name of the callback function invoked.

What we just went through is the the easiest way to access Flickr via JavaScript. We will come back in Chapter 10 to handle more general cases-- including dealing with cross-domain restrictions in directly accessing the REST interface from the browser.

Greasemonkey Example

Greasemonkey is an extension for Firefox. It allows you to to change the behavior of web pages that come into your browser. That includes creating mashups -- combining different sources of information. GMiF -- which we'll study in depth next chapter -- is an example. There are some good references that you can use to get the best of Greasemonkey:

* http://diveintogreasemonkey.org/
* *Greasemonkey Hacks* by Mark Pilgrim.
* http://en.wikipedia.org/wiki/Greasemonkey
* http://www.greasespot.net/ -- the offical blog of the Greasemonkey project.

NYT Permlinker

Links that show up on the NY Times online site typically expire after a week. That is, instead of going to the article, you are given an excerpt and a chance to purchase a copy of the article. However, in 2003, Dave Winer struck a deal with the NY Times to provide a mechanism to get "weblog safe"/ Permlinks to Times.25 Aaron Swartz wrote a New York Times Link Generator that is compiling those permlinks and makes them available for lookup via a web form or a JavaScript bookmarklet.26 That is, you give it a URL to a NY Times article and it will return you a more permanent link.

Let's look at an example. If you
`http://www.nytimes.com/2007/04/04/education/04colleges.html`
goes to
`http://www.nytimes.com/2007/04/04/education/04colleges.html?ex=1333339200&en=3b7aac16a1ce4512&ei=5090&partner=rssuserland&emc=rss`

You can see this for yourself by going to
`http://nytimes.blogspace.com/genlink?q=http://www.nytimes.com/2007/04/04/education/04colleges.html`
When there is no permlink for an article, you will see a different type of output from the NY Times Link Generator. For example, consider
`http://www.nytimes.com/aponline/us/AP-Imus-Protests.html`

---

[25] http://www.scripting.com/davenet/2003/06/06/newYorkTimesArchiveAndWebl.html

[26] http://nytimes.blogspace.com/genlink

which doesn't have a permlink as you can see from http://nytimes.blogspace.com/genlink?q=http://www.nytimes.com/aponline/us/AP-Imus-Protests.html

Where's a good place to stick a UI element for the permanent link on the New York Times page. Lots of choices but a good one is toolbar with such elements as e-mail/print/single-page/save.

So, the basic logic of the Greasemonkey script we want to write consists of the following:

1. If we are on at NY Times article, send the link to the New York Times Link Generator.

2. If there is a permlink, (which we will know is true if the href attribute of the first a tag will start with 'http:' and not 'genlink'), insert a new li element at the end of the <ul id="toolsList">.

Here, I walk you through the steps to get this functionality working in your own FireFox browser installation:

1. Install the Greasemonkey extension if you don't already have it installed.[27] (The following example is based on the latest version of GM as of the writing of this text: 0.6.8.20070314.0) [28]

2. Create a new script in the Greasemonkey in one of two ways: 1) You can go to http://examples.mashupguide.net/ch08/newyorktimespermalinker.user.js and click "Install" or 2) Tools->Greasemonkey->New User Script and fill in Name/Namespace/Description/Includes and then copy and paste the following code. (Note the use GM_xmlhttpRequest to find out what a more "permanent link" is.[29])

```
// ==UserScript==
// @name           New York Times Permlinker
// @namespace      http://mashupguide.net
// @description    Adds a link to a "permlink" or  "weblog-safe" URL for the NY
Times article, if such a link exists
// @include        http://*.nytimes.com/*
// ==/UserScript==

function rd(){

  // the following code is based on the bookmarklet written by Aaron Swartz at
http://nytimes.blogspace.com/genlink

  var x,t,i,j;
  // change %3A -> : and %2F -> '/'
  t=location.href.replace(/[%]3A/ig,':').replace(/[%]2f/ig,'/');
```

---

[27] https://addons.mozilla.org/en-US/firefox/addon/748

[28] http://addons.mozilla.org/en-US/firefox/downloads/file/13931/greasemonkey-0.6.8.20070314.0-firefox.xpi

[29] http://wiki.greasespot.net/GM_xmlhttpRequest and http://diveintogreasemonkey.org/api/gm_xmlhttprequest.html

```
  // get last occurence of "http://"
  i=t.lastIndexOf('http://');

  // lop off stuff after '&'
  if(i>0){
    t=t.substring(i);
    j=t.indexOf('&');
    if(j>0)t=t.substring(0,j)
  }

  var url = 'http://nytimes.blogspace.com/genlink?q='+t;

  // send the NY Times link to the nytimes.blogspace.com service. If there is a
permlink, then the href attribute of the first a tag will start with 'http:' and not
'genlink'.
  // if there is a permlink, then insert a new li element at the end of the <ul
id="toolsList">.

  GM_xmlhttpRequest({
  method:"GET",
  url:url,
  headers:{
    "User-Agent":"monkeyagent",
    "Accept":"text/html",
  },
  onload:function(details) {
    var s = details.responseText;
    var p = /a href="(.*)"/;
    var plink = s.match(p)[1];
    if ( plink.match(/^http:/) && (tl = document.getElementById('toolsList')) )  {
      plink = plink + "&pagewanted=all";
      plinkItem = document.createElement('li');
      plinkItem.innerHTML = '<a href="' + plink + '">PermLink</a>';
      tl.appendChild(plinkItem);
    }
  }
});

}

rd();
```

3. What you will see now with this Greasemonkey script if you go to `http://www.nytimes.com/2007/04/04/education/04colleges.html` you will see a new entry "PermLink" underneath the Share button.

Note a few things about this Greasemonkey script. It is certainly very sensitive to changes in both the way the way NY Times articles are laid out: older articles have a different page structure and therefore need some other logic to put the permlink in the right place.

Learning More about JavaScript and Ajax

It's good to have handy some references on JavaScript programming while working on Ajax-related projects:

* *JavaScript, the Definitive Guide* (5th edition) by David Flanagan[30]
* *Pro JavaScript Techniques* by John Resig[31]
* *Learning JavaScript* by Shelley Powers[32]

I'm particularly fond of Peter-Paul Koch's website, which has tons of useful resources:

* The general resource page[33] -- see His introduction ot JavaScript.[34]
* He has all the example scripts from his book that you can study: http://www.quirksmode.org/book/examplescripts.html
* He is selling a book: http://www.quirksmode.org/book/

Simon Willison gave a great talk at ETech:

* http://simonwillison.net/2006/Mar/7/etech/
* http://www.windley.com/archives/2006/03/introduction_to.shtml is a detailed recap. Very useful.

Conclusion

In this chapter, you have learned about how Ajax has changed the style of programming in contemporary web applications. You have doens so by turning off JavaScript in your browser and see what happens to Flickr and Goo-urn5sie stoff Javalibontemrby.6(atihy)-6(e u have).xambasichas