JQuery, JSON, AJAX

AJAX: Async JavaScript & XML

- In traditional Web coding, to get information from a database or a file on the server
  - make an HTML form
  - GET or POST data to the server (click the "Submit" button)
  - the browser loads a results page.

- Because the server returns a new page each time the user submits input, traditional web applications can run slowly and tend to be less user-friendly.

AJAX: Async JavaScript & XML

- AJAX does not require the page to be reloaded
  - Instead JavaScript communicates directly with the server using the *XMLHttpRequest* object
- The user will stay on the same page, and he or she will not notice that scripts request pages, or send data to a server in the background.
- Because it's asynchronous, the browser is not locked up

XMLHttpRequest or JQuery

- Methods to handle comms
  - Very tedious
- JQuery

```
$.ajax({
    url: 'document.xml',
    type: 'GET',
    dataType: 'xml',
    timeout: 1000,
    error: function(){alert('Error loading XML'); },
    success: function(xml){do something}
    });
```

Example Success Function

```
success: function(xml){
    $(xml).find('item').each(function(){
        var item_text = $(this).text();

        $('<li></li>')
            .html(item_text)
            .appendTo('ol');
    });
}
```

Simpler AJAX

- Often, you simply need to pass some parameters to a page on the server.

- `$.post('save.cgi', {text: 'my string', number: 23}, function() {alert('Your data has been saved.');});`

Simplest AJAX

- A common use of Ajax is to load a chunk of HTML into an area of the page. To do that, simply select the element you need and use the load() function.

```
$('#stats').load('stats.html');
```

Trouble with AJAX

- Two problems with AJAX
  - Security model
  - XML
- Most browsers prohibit AJAX calls to different domains
- XML is tedious to parse and use
  - Can serialise data as JavaScript objects!

JSON: JavaScript Object Notation

- Basic types are
  - **Number** integer, real, or floating point
  - **String** double-quoted Unicode with backslashes
  - **Boolean** true and false
  - **Array** ordered sequence of comma-separated values enclosed in square brackets
  - **Object** collection of comma-separated "key":value pairs enclosed in curly brackets
  - **null**

JSON: JavaScript Object Notation

```
{ "firstName": "Leslie",
  "lastName": "Carr",
  "age": 43,
  "address": {
     "streetAddress": "21 Foo Str”,
     "city": "New York"
     },
  "powers": [1, 2, 4, 8, 16, 32]
  }
```

```
var p = eval("(" + JSON_text + ")");
alert(p.age);
```

JSON-P: JavaScript Object Notation

```
foo(
  { "firstName": "Leslie",
    "lastName": "Carr",
    "age": 43,
    "powers": [1, 2, 4, 8, 16, 32]
    }
  )
```

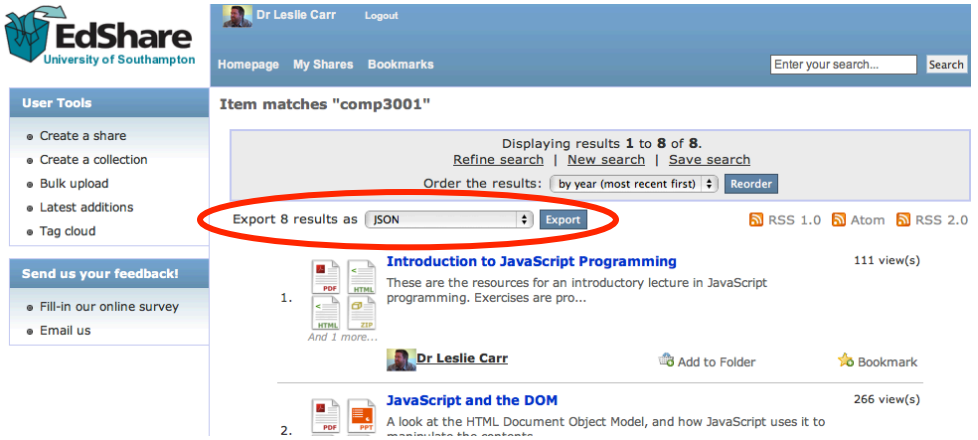
- Bypasses security by pretending to be a script
- API calls have a *callback=foo* parameter
  - *i.e.* client tells the server what callback to invoke on the returned data

JSON/JQuery Flickr Example

```
$.getJSON("http://api.flickr.com/services/feeds/photos_public.gne?
    tags=cat&tagmode=any&format=json&jsoncallback=?",
          function(data){
            $.each(data.items, function(i,item){
              $("<img/>").attr("src", item.media.m).appendTo("#images");
              if ( i == 3 ) return false;
            });
          });
```

The Latest Cat Photos from Flickr

EPrints




```
$(document).ready(function(){
   $.getJSON("http://www.edshare.soton.ac.uk/cgi/search/simple/export_edshare_JSON.js?exp=…..comp3001…&jsonp=?",
      function(data){
        $.each(data, function(i,item){
          $("<a/>").
            attr("href","http://edshare.soton.ac.uk/"+item.eprintid).append(
              $("<img/>").
                attr("src","http://edshare.soton.ac.uk/"+item.eprintid+"/thumbnails/1/preview.png").
                 bind("error",function(){$(this).addClass("hidden")})
             ).appendTo("#lectures");
        });
      });
 });
```

Recent JavaScript Lectures

```
<h1>Recent JavaScript Lectures</h1>
<div id="lectures">
</div>
```