JavaScript and AJAX

Lambdas, Literals, JSON and AJAX

Huw Davies
daviesah7@cardiff.ac.uk

JavaScript and AJAX

Literal Objects

You have seen these in jQuery and Google Maps for defining options

```
var marker = new google.maps.Marker ({
   position: myLatLng,
   map: map,
   title: 'My Map'
});
```

JavaScript and AJAX

- Object literals are a list of property/value pairs
  - Enclosed in braces
  - Colon between the property and the value
  - Comma between the items in the list

JavaScript and AJAX

Example:

```
var employee = {
   fullname: 'Joe Bloggs',
   age: 23,
   job: 'manager'
};
```

JavaScript and AJAX

- The properties are names, like variables, so are not quoted
- The values may be any valid JavaScript type (such as **string**, **number**, **boolean** and so on – including other instances of objects)
- Values can be other (nested) arrays and even be functions

JavaScript and AJAX

Accessing properties in Object Literals

var emp = {name: 'fred', age: 23};

- Dot notation:
  - hisName = emp.name;
- Array (bracket) notation:
  - hisName = emp['name']; // Note the quotes
  - cf Perl and PHP 'associative' lists

JavaScript and AJAX

- Demonstration
  - Array and dot notation
  - literalobjects1.html

JavaScript and AJAX

- Demonstration
  - Nested arrays and objects
  - literalobjects2.html

JavaScript and AJAX

Lambda functions

- 'Anonymous' functions:

  var fun = function(x) { return 2 * x; };

- fun can now be used as a normal function

  x = fun(5) // returns 10

JavaScript and AJAX

- There is next to no difference between the following:
  - doubler = function(y) { return y * 2 };
  - function doubler(y) { return y * 2};
- Both would be called using:
  - x = doubler(5);
- The only difference is that one is a function object and the other a pointer to a function

JavaScript and AJAX

- A Lambda function can be a value in a literal object
- Demonstration: literalobjects3.html

JavaScript and AJAX

JavaScript namespaces

```
var AppSpace = AppSpace || {};
```

What does that do???

- Namespaces are a way of making sure your code doesn't clash with other people's
  - Important if you are writing code libraries that you expect other people to use

JavaScript and AJAX

- The previous code ensures that if AppSpace has already been defined it is left untouched
- If AppSpace has not been defined, it is assigned an empty object literal
- We can now assign properties and methods to it

JavaScript and AJAX

```
AppSpace.Podcast = function {
  this.title = 'Astronomy Cast';
  this.description = 'A fact-based journey through the galaxy.';
  this.link = 'http://www.astronomycast.com';
};
```

- The Podcast object is now only available in the AppSpace namespace

JavaScript and AJAX

Namespaces can be nested:

```
var MyCompany = MyCompany || {};
MyCompany.MyApplication = {};
MyCompany.MyApplication.Model = {};
// Now add properties etc...
MyCompany.MyApplication.Model.name = 'Fred';
```

JavaScript and AJAX

JSON

- JavaScript Object Notation
- Uses literal-like objects to pass data between a client and a server
- Easy to access different fields from server
- NB: properties must be passed as double-quoted strings

JavaScript and AJAX

JSON (2)

{"first": "fred", "last": "bloggs"}

Passed as text but converted to names on receipt

JavaScript and AJAX

jQuery and AJAX

```
$.ajax({type: method,
        url: url,
        data: data,
        success: suc,
        dataType: dataType,
        error: err});
```

There are some other options as well (like beforeSend), but the ones above are the most important.

JavaScript and AJAX

- type: POST or GET
- url: the address of the server
- data: the POST data for the POST method
- success: callback method on success
- dataType: the type of data being returned by the server (e.g. JSON, HTML, text, XML)
- error: callback method on error

JavaScript and AJAX

JSONP

- Normal AJAX calls using JSON only allow retrieval of data from the same server as the client
- JSONP allows you to get around that
- Basically a Cross-site Scripting hack

JavaScript and AJAX

To use JSONP client-side

- URL will be to a different server from the client
- Data is specified in the normal way
- DataType is 'jsonp'
- A 'jsonp' property is set to 'callback'
- A jsonpCallback property is set to the callback function
- The 'success' and other properties are as normal

JavaScript and AJAX

- JSONP demonstration
  - jsonp-client.html
  - jsonp-server.php