JavaScript and AJAX

Conditions, branches, forms and frameworks

Huw Davies
daviesah7@cardiff.ac.uk

JavaScript and AJAX

- Conditional expressions
  - Evaluate to TRUE or FALSE
  - Examples:
    - x < y
      - TRUE if the value stored in x is less than the value in y
    - (x < y) && (a > b)
      - TRUE if x is less than y AND a is greater than b
  - Conditions can get quite complex
    - Use parentheses to make the meaning clear

JavaScript and AJAX

- Conditional operators
  - See www.w3schools.com/js/js_comparisons.asp
- The difference between assignment and equality:
  - '=' is the assignment operator and not a test
  - '==' tests that two values are equivalent
  - '===' tests that two values are equivalent and of the same type

JavaScript and AJAX

- The **if** statement
  - Used to perform some action(s) on the basis of some condition(s)
  - Has three forms:

    if (condition) { if-actions }

    if (condition) { if-actions } **else** { else-actions }

    if (condition) { if-actions }
    else **if** (condition) { else-if actions }
    …
    else { else-actions }

JavaScript and AJAX

The simple if statement:

```
if ( condition ) {
    statements to perform if condition is true
}
```

```
if ( x == 1) { alert( 'X equals 1' ); }
```

Note: can omit braces { } for just a single action

JavaScript and AJAX

The if – else statement:

```
if ( condition ) {
    statements to perform if condition is true
} else {
    statements  to perform if condition is false
}
```

```
if ( x == 1 ) { alert( 'one' ); } else { alert( 'not one' );
}
```

JavaScript and AJAX

The if – else if … else statement

```
If ( x == 1 ) {
    alert( 'one' );
} else if ( x == 2 ) {
    alert ( 'two' );
} else if ( x == 3 ) {
    alert ( 'three' );
} else {
    alert ( 'none of the above' ):
```

JavaScript and AJAX

- The ternary operator
  - Alternative to if – else
  - condition ? action-if-true : action-if-false;
  - Can be used to assign to a variable:

```
var elvisLives = Math.PI > 4 ? "Yep" : "Nope";
alert( elvisLives );
```

JavaScript and AJAX

- Accessing form fields
  - document.getElementById("fieldName")
    - where fieldName is the id associated with the form field
    - gets access to the field
    - **var** theField = document.getElementById("theId");
  - You can then get the field's value by doing:
    - **var** theValue = theField.value;
  - Or you can do the whole thing in one go:
    - **var** theValue = document.getElementById("theId").value;

JavaScript and AJAX

- Checkboxes
  - The value of a checkbox does not change according to whether it is checked or not!
  - So we use the property **.checked** which is **true** if it is checked or **false** otherwise
  - document.getElementById('agree').checked

JavaScript and AJAX

- Radio buttons
  - Need to identify which one
  - E.g. to check that at least one of two buttons has been checked we will use the following method:

```
x = document.forms[0]['gender'];
      // form[0] is the first form
      // 'gender' is the name given to the radio buttons
if ( !x[0].checked && !x[1].checked) {
     // do something if neither button is checked
}
```

JavaScript and AJAX

Processing events

- When an event occurs (fires) the following happens:
  - any event handler (e.g. **onclick**) is processed
  - the default action is performed unless the handler returns false
    - e.g. for a hyperlink – it goes to the location
  - if the event handler returns false the default is not performed e.g.

```
<a href="..." onclick="alert('I am on strike!'); return false;">a link</a>
```

    - clicking the link above produces the alert message and

JavaScript and AJAX

Intercepting form submission for validation

- When the submit button is pressed we trigger an onSubmit event and then send the form data only if it's valid:
  - Add an onSubmit event to the <form> tag calling a function such as “checkMyForm()”

    <form onsubmit="return checkMyForm()" … >
  - Stop the form data being sent if it's not valid by returning false from checkMyForm()
    - if we return false the default submit behaviour is stopped.

JavaScript and AJAX

Frameworks 1: the PROs

- Lots of complex JavaScripting done for us, including browser support
  - can simplify coding e.g. **document.getElementByID('x')** can be written as **$('x')** in the **prototype** framework, and **jQuery** is similar
- Lots of easy-to-add features, such as:
  - AJAX interface
  - Animations
  - Tabbed pages

JavaScript and AJAX

Frameworks 2: the CONs

- Have to learn how to use the framework
  - Not normally a big issue
- Cannot normally use more than one framework per page
- Adds extra payload (100k or more) to web page
- Debugging can be more complex
  - e.g. the bug might turn out to be in the framework code

JavaScript and AJAX

Prototype.js

- Light weight framework (142k)
- Can access elements with less typing e.g.
  - **document.getElementById('anEl')** becomes **$('anEl')**
  - **document.getElementById('x').value** becomes **$F('x')**
- Supports AJAX (wraps **xmlHTTPRequest**)
- A number of other frameworks incorporate it
  - Rico, Script.acul.us

JavaScript and AJAX

- Including the framework in your web pages
  - Download a copy to your site and reference it in a script tag:

    ```
    <script src="path/to/prototype.js"></script>
    ```

  - Download from a CDN like Google:

    ```
    <script
    src="http://ajax.googleapis.com/ajax/libs/prototype/1.6.0.2/prototype.js">
    </script>
    ```

JavaScript and AJAX

- Examples:
  - Use the result of the last example
  - Include the **prototype.js** JavaScript in a <script> tag
  - Modify the JavaScript to manipulate the DOM using **prototype.js**