JavaScript &
the browser

Lecture 18
CS 638 Web Programming

Lecture overview

- Browser context for JavaScript
- User interaction with JavaScript

Adding JavaScript to a page

- Using the <script> </script> tag
  - Text between tags is JavaScript program
  - Can specify external file using `src` attribute
  - Executed as the document is loading
- Value of an attribute such as `onclick`
  - This type of code is called event handler
  - Executed when event happens
- Body of a URL using the `javascript:` protocol
  - Executed when browser loads URL
The context for JavaScript

- All scripts on the same page share the same global variables, even if they are read using the src attribute
- All global variables are just properties of a "global object"
- The keyword this (unless inside object methods) and the variable window refer to the "global object"

Some properties of window

- document holds the internal representation of the document displayed in the window
- location holds the URL of the current page
  - Can instruct the browser to load a new page by assigning its URL to location
- status (a.k.a. defaultStatus) holds the current message in the status bar
  - Typically shows the URL pointed to by link under mouse
- navigator has details about the browser software (application name, version, platform, etc.)

Some methods of window

- For interacting with user through dialog boxes
  - alert() displays a message
  - confirm() asks user to confirm or cancel
  - prompt() asks user to enter string
- For manipulating timers
- For manipulating windows (opening a new one, closing, moving, resizing, printing, etc.)
- Browser configuration may disallow certain functions (or the writing of certain properties)
```html
<script language="JavaScript" type="text/javascript">
// Browser information
for(prop in navigator)
  browserInfo+=prop+":"+navigator[prop]+'
alert(browserInfo);

if(confirm("Do you like cheese?"))
  document.write("Wisconsin is a good place for you.<br>
else
  document.write("You have weird preferences.<br>
username=prompt("What should I call you?");
document.write("Bye bye "+username+"!")
</script>
```
Debugging JavaScript

- Heed messages in the error console
- Use alerts to display debugging info
- Firefox add-ons can debug JavaScript
  - Firefly, Venkman
  - Unfortunately they don’t always work

Lecture overview

- Browser context for JavaScript
- User interaction with JavaScript
Accessing forms

- `document.forms` array of objects representing the forms on the page (usually just one form present)
- `document.forms[0].elements` holds array with objects representing the HTML controls in form

Referring to controls

- `document.forms[0].xyz` – control with name xyz
- `document.forms[0].xyz[1]` – second control with name xyz (recall that radio buttons working as a group must have same name)

Form elements

- Properties of JavaScript objects for form elements
  - `type` – values such as "button", "text", "checkbox"
  - `form` – the form containing the element
  - `name` – name of element (defined by HTML attribute name)
  - `value` – the value of the element (exact interpretation depends on type of element)
    - By modifying this property you can for example change the text shown on a button or the text within an text input field
  - Many event handlers associated with each element
    - May specify JavaScript function to be invoked in response
    - HTML elements other than

Some simple JavaScript events

<table>
<thead>
<tr>
<th>Handler</th>
<th>Triggered when</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>onclick</td>
<td>Mouse click on element</td>
<td>Return <code>false</code> to cancel default action</td>
</tr>
<tr>
<td>onchange</td>
<td>The control's state changes</td>
<td></td>
</tr>
<tr>
<td>onfocus</td>
<td>The control gets focus</td>
<td></td>
</tr>
<tr>
<td>onsubmit</td>
<td>The form is submitted</td>
<td>Specific to forms</td>
</tr>
<tr>
<td>onmouseup</td>
<td>Mouse moves over el.</td>
<td></td>
</tr>
<tr>
<td>onmouseout</td>
<td>Mouse moves off el.</td>
<td></td>
</tr>
<tr>
<td>onmousedown</td>
<td>Mouse button pressed</td>
<td></td>
</tr>
<tr>
<td>onmouseup</td>
<td>Mouse button released</td>
<td></td>
</tr>
<tr>
<td>onkeydown</td>
<td>Key pressed down</td>
<td>Used for form elements and <code>&lt;body&gt;</code></td>
</tr>
<tr>
<td>onkeyup</td>
<td>Key released</td>
<td>Used for form elements and <code>&lt;body&gt;</code></td>
</tr>
<tr>
<td>onkeydown</td>
<td>Key pressed down</td>
<td>Return <code>false</code> to cancel</td>
</tr>
<tr>
<td>onkeyup</td>
<td>Key released</td>
<td></td>
</tr>
<tr>
<td>onload</td>
<td>Document load complete</td>
<td>Used for <code>&lt;body&gt;</code> and <code>&lt;img&gt;</code></td>
</tr>
<tr>
<td>onreadystatechange</td>
<td>Document load complete</td>
<td></td>
</tr>
</tbody>
</table>
Defining event handlers

- Can be defined as values of the HTML attribute with the name describing the event
- Can use arbitrary JavaScript code, often just a call to a function is used
- `<form onsubmit="return verify(this)">`
- Can be defined inside program by assigning to properties of objects representing elements
  - `document.forms[0].onsubmit=function(){alert("submission");return false}`
- When event handling code is run, `this` points to the object representing the HTML element the event applies to