CSC 551: Web Programming

Spring 2004

Emerging technologies

- ➢ Dynamic HTML
  - ✓ DOM1, attributes & methods
  - ✓ event handling, visibility, drag&drop
- ➢ XML
  - ✓ XML & CSS, XSL, DTD
- ➢ ActiveX

Dynamic HTML

chapters 5 & 6 describe more dynamic features of HTML

- possible to change the appearance of HTML elements after loading

Domain Object Model (DOM)

- Level 0 (pre 1988): Netscape & Microsoft had different models for accessing page elements and modifying their attributes
- Level 1 (1988): standard defined by World Wide Web Consortium (W3C)

  DOM1 consists of a collection of interfaces, similar to C++/Java API

  supported by all modern browsers
- Level 2 (2000): specifies a style sheet object model & defines how style is handled

  mostly supported by Netscape/Mozilla, spotty coverage by IE
- Level 3 (2004): in development but not yet adopted

DOM1

Document Object Model – Level 1

document attributes

▪ `document.title`	title of the page
▪ `document.referrer`	URL of document that linked to the page
▪ `document.domain`	domain name of server for the page
▪ `document.URL`	full URL of the page
▪ `document.lastModified`	data the page was last edited
▪ `document.cookie`	cookies associated with the page
▪ `document.images`	array of all image elements in the page
▪ `document.applets`	array of all applet elements in the page
▪ `document.anchors`	array of all anchor elements in the page
▪ `document.forms`	array of all forms in the page

can access individual elements of these arrays using:

```
document.images.card1.src = "as.gif";
document.images["card1"].src = "as.gif";
```

DOM1 methods

document methods

▪ `document.open`	opens document stream for writing
▪ `document.write`	writes text to document stream
▪ `document.close`	closes document stream
▪ `document.getElementById`	accesses arbitrary page element, identified by its ID attribute

```
<img id="card" src="b.gif" />


<input type="button" value="flip"
       onClick="var c = document.getElementById('card');
                if (c.src.indexOf('b.gif') != -1) {
                    c.src = 'as.gif';
                }
                else {
                    c.src = 'b.gif';
                }" />
```

view page in browser

Dynamic style modifications

more useful: can change the appearance of page elements after loading

- use `getElementById` to get a handle on the element
- access the (CSS) style attribute by name
  (if style attribute has hyphen, remove and capitalize next letter)

```
<img id="card" src="b.gif"
     style="border-style:solid; border-color:black" />


<input type="button" value="border"
       onClick="var c = document.getElementById('card');
                if (c.style.borderColor == 'black') {
                  c.style.borderColor = 'red';
                }
                else {
                  c.style.borderColor = 'black';
                }" />
```

view page in browser

Dynamic text

```
<p id="changeme" style="text-align:left; color:black; font-weight:normal">
Here is some plain text.</p>

<form name="textStuff">
  <input type="button" value="align"
         onClick="var t = document.getElementById('changeme');
                  if (t.style.textAlign == 'left') {
                    t.style.textAlign = 'center';
                  }
                  else if (t.style.textAlign == 'center') {
                    t.style.textAlign = 'right';
                  }
                  else {
                    t.style.textAlign = 'left';
                  }">
  <input type="button" value="bold"
         onClick="var t = document.getElementById('changeme');
                  if (t.style.fontWeight == 'normal') {
                    t.style.fontWeight = 'bold';
                  }
                  else {
                    t.style.fontWeight = 'normal';
                  }">
  <input type="button" value="color"
         onClick="var t = document.getElementById('changeme');
                  if (t.style.color == 'black') {
                    t.style.color = 'blue';
                  }
                  else {
                    t.style.color = 'black';
                  }">
</form>
```

style attributes can be dynamically altered for any kind of page element

view page in browser

Event handling and dynamic style attributes

can combine dynamic style with event handling such as

`onClick`, `onFocus`, `onMouseOver`, `onMouseOut`, …

```
<img name="card" id="card" src="b.gif"
     style="border-style:solid; border-color:black;border-width=2"
     onMouseOver="this.style.borderColor='red';"
     onMouseOut="this.style.borderColor='black';" />



<p id="changeme" style="text-align:left; color:black; font-weight:normal"
   onMouseOver="this.style.color='blue';"
   onMouseOut="this.style.color='black';">
Here is some plain text.</p>
```

view page in browser

Visibility and button labels

it is possible to hide an element altogether using the visibility attribute

can also change the label on buttons

```
<img name="card" id="card" src="b.gif"
     style="border-style:solid; border-color:black;
            border-width=2; visibility:visible"
     onMouseOver="this.style.borderColor='red';"
     onMouseOut="this.style.borderColor='black';" />


<input type="button" value="hide"
       onClick="var c = document.getElementById('card');
                if (c.style.visibility == 'visible') {
                    c.style.visibility = 'hidden';
                    this.value = 'show'
                }
                else {
                    c.style.visibility = 'visible';
                    this.value = 'hide';
                }" />
```

view page in browser

Other common uses of DHTML

pull-down menus, collapsing lists:

- cs.creighton.edu

drag-and-drop (but cannot be done cross-platform without duplication)

- www.jsmadeeasy.com/javascripts/IE5%20Scripts/alienhead/template.htm

Extensible Markup Language (XML)

protocol for representing structured data in text files

- can represent arbitrary structures, define own abstractions
- since raw text, easy to peruse/edit & platform-independent

XML is a meta-language for designing your own markup language

- like HTML, utilizes tags and attributes (e.g., `<p name="foo">`)
  however, HTML specifies what each tag & attribute means
  whereas, XML simply delimits pieces of data, no interpretation implied

XML is meant to be read by applications, not people

- formatting rules are very strict (missing tag or quotes invalidates file)
- many applications have been developed that utilize XML as data format

note: representing data as XML text is not the most efficient bitwise

- disk space is cheap; compression tools can alleviate

XML & HTML

HTML has been reformulated in XML (now known as XHTML 1.0)

- an existing HTML document is valid XML as long as
  - first line is of form: `<?XML version="1.0" ?>`
  - tags & attributes are lower-case, attribute values are in quotes
  - every opening tag has a closing tag (or else ends with />)

    e.g., `<p></p>`   `<img src="foo.gif" />`

can define own tags for structuring data

```
<question>
  Where was the Web invented?
  <answer> Microsoft </answer>
  <answer> Sun Microsystems </answer>
  <answer correct="true"> CERN </answer>
  <answer> IBM </answer>
</question>
```

XML & CSS

for simple viewing in a browser, can utilize a style sheet to specify format

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/css" href="question.css" ?>

<question>
  Where was the Web invented?
  <answer> Microsoft </answer>
  <answer> Sun Microsystems </answer>
  <answer class="correct"> CERN </answer>
  <answer> IBM </answer>
</question>
```

`question.xml:`

?xml tags identify content type, load CSS stylesheet

```
question {display:block; font-size:120%;
          font-weight:bold;}
question answer {display:block; margin-left:40px;
                 font-weight:normal}
question answer.correct {color:red}
```

`question.css:`

defines format for new elements

view page in browser

More advanced features

can use a separate program to transform XML text into HTML

can utilize Extensible Stylesheet Language (XSL)
- similar to CSS, defines how XML elements are to be formatted
- more powerful than CSS, customized for XML

can use Document Type Declaration (DTD) to specify the element structure

```
<!element question (#PCDATA, answer+)>
<!element answer (#PCDATA)>
<!attlist answer class CDATA #IMPLIED>
```

- include the DTD with the XML document
  automatically used to validate the element structure in the document

```
<?XML version="1.0" rmd="all">
<!DOCTYPE test SYSTEM "test.dtd"
<question>
  ...
</question>
```

ActiveX

ActiveX is a set of technologies from Microsoft
- provides tools for linking (Windows) desktop applications to the Web.
- ActiveX controls are the building block of applications
  e.g., text control to read user ID & password, button control to submit
- similar to applets, but have full access to Windows OS
- once downloaded to the client, the control automatically registers itself & becomes available to the browser
- can automatically trigger a self-update if newer version is available

thousands of ActiveX controls are available for download
- can develop your own using Microsoft programming tools
  e.g., Visual Basic, Visual C++, Visual J++

ActiveX controls are integrated into Microsoft products

e.g, can allow users to view Word and Excel documents directly in a browser

FINAL EXAM

similar format to previous tests

- true or false
- discussion/short answer
- explain or modify code (HTML, JavaScript, Java, CGI)

cumulative, but will emphasize material since Test 2

designed to take 60-75 minutes, will allow full 100 minutes

study hints:

- review lecture notes
- review text
- look for supplementary materials where needed (e.g., Web search)
- think big picture -- assimilate the material!
- use online review sheet as a study guide, *but not exhaustive*