Document Object Model (DOM)

INFO/CSE 100, Spring 2006
Fluency in Information Technology

http://www.cs.washington.edu/100
References

• References
  » *JavaScript, The Definitive Guide*
    • by David Flanagan. Publisher O'Reilly

» W3C Document Object Model
  • http://www.w3.org/DOM/
  • http://www.w3.org/2003/02/06-dom-support.html

» Document Object Model in Mozilla
  • http://www.mozilla.org/docs/dom/
What the heck is the DOM?

• Document Object Model
  » Your web browser builds a *model* of the web page (the *document*) that includes all the *objects* in the page (tags, text, etc)
  » All of the properties, methods, and events available to the web developer for manipulating and creating web pages are organized into objects
  » Those objects are accessible via scripting languages in modern web browsers
This is what the browser reads (sampleDOM.html).

```html
<html>
  <head>
    <title>Sample DOM Document</title>
  </head>
  <body>
    <h1>An HTML Document</h1>
    <p>This is a <i>simple</i> document.</p>
  </body>
</html>
```

This is what the browser displays on screen.

An HTML Document

This is a simple document.
This is a drawing of the model that the browser is working with for the page.

Figure 17-1. The tree representation of an HTML document
Copied from JavaScript by Flanagan.
Why is this useful?

• Because we can access the model too!
  » the model is made available to scripts running in the browser, not just the browser itself
    • A script can find things out about the state of the page
    • A script can change things in response to events, including user requests
  » We have already used this capability in the GUI programming that we've done
Recall our simple GUI example

This GUI has several simple controls.

- Two buttons to control the results
- One text field to display the results
- One pair of radio buttons to control the display
- One button to reinitialize

setResults(resultString)

```javascript
<script type="text/javascript">
function setResults(resultString) {
    var tempString = resultString;
    if (document.getElementById("radioLC").checked) {
        tempString = tempString.toLowerCase();
    } else if (document.getElementById("radioUC").checked) {
        tempString = tempString.toUpperCase();
    }
    document.getElementById("resultField").value = tempString;
}
</script>
```

The highlighted script above makes reference to several objects in the document object model.
Reference to several nodes in the model of the page that the browser constructed

- `document.getElementById("radioLC").checked`

- The root of the tree is an object of type `HTMLDocument`

- Using the global variable `document`, we can access all the nodes in the tree, as well as useful functions and other global information
  - title, referrer, domain, URL, body, images, links, forms, ...
  - open, write, close, `getElementById`, ...
Some information from a document

```html
<html>
  <head>
    <title>DOM Sample A</title>
  </head>
  <body>
    Information about this document.<br>
    <script type="text/javascript">
        document.write("<br>Title: ",document.title);
        document.write("<br>Referrer: ",document.referrer);
        document.write("<br>Domain: ",document.domain);
        document.write("<br>URL: ",document.URL);
    </script>
  </body>
</html>
```

Information about this document.

title: DOM Sample A
referrer: http://www.cs.washington.edu/education/courses/cse100/04au/calendar100.html
domain: www.cs.washington.edu
document.getElementById("radioLC").checked

• `getElementById("radioLC")`
  » This is a predefined function that makes use of the `id` that can be defined for any element in the page
  » An `id` must be unique in the page, so only one element is ever returned by this function
  » The argument to `getElementById` specifies which element is being requested
Some information about elements

<html>
<head>
<title>DOM Sample B</title>
<script type="text/javascript">
function showInfo() {
  var element = document.getElementById("opener");
  var buffer = element.id + " tag is " + element.tagName;
  alert(buffer);
  element = document.getElementById("actionItem");
  buffer = element.id + " tag is " + element.tagName;
  buffer += ", type is "+element.type;
  alert(buffer);
}
</script>
</head>
<body>
<p id="opener">The id attribute is very helpful.</p>
<p id="closer">This is the closing paragraph.</p>
<form>
<button id="actionItem" type="button" onclick="showInfo()">Show Info</button>
</form>
</body>
</html>
The id attribute is very helpful.

This is the closing paragraph.
document.getElementById("radioLC").checked

• **checked**
  » This is a particular property of the node we are looking at, in this case, a radio button
  » Each type of node has its own set of properties
    • for radio button: `checked`, `name`, ...
    • refer to the HTML DOM for specifics for each element type
  » Some properties can be both read and set
Some specific properties

```html
<head>
<title>Simple Sample GUI</title>
<script type="text/javascript">
function setResults(resultString) {
  var tempString = resultString;
  if (document.getElementById("radioLC").checked) {
    tempString = tempString.toLowerCase();
  } else if (document.getElementById("radioUC").checked) {
    tempString = tempString.toUpperCase();
  }
  document.getElementById("resultField").value = tempString;
}
</script>
</head>
```
Result: good results

- Lower case

Reset
Just the tip of the DOM

• The HTML Document Object Model is a standard for structuring data on a web page
  » The field is advancing rapidly as people recognize the benefits of standardized structure and access
  » The DOM is steadily improving to cover general purpose data structuring requirements

• XML (Extendible Markup Language) also uses the Core DOM to specify its structured data
  » similar to HTML but more carefully defined
Getting vs. Setting

```
var oldvalue = document.getElementById("resultField").value;
document.getElementById("resultField").value = "new value";
```
<table>
<thead>
<tr>
<th>DOM Module</th>
<th>DOM Level 1</th>
<th>DOM Level 2</th>
<th>DOM Level 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Core</strong>: basic methods (Level 1 and 2) and extensions for XML Namespaces (Level 2 only)</td>
<td>-</td>
<td>supported</td>
<td>2004</td>
</tr>
<tr>
<td><strong>XML</strong>: extensions for XML 1.0</td>
<td>supported</td>
<td>supported</td>
<td>2004</td>
</tr>
<tr>
<td><strong>HTML</strong>: extensions for HTML 4.0x (Level 1 and 2) and support of XHTML 1.0 (Level 2 only)</td>
<td>supported</td>
<td>supported</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Views</strong>: used with the Level 2 CSS and UIEvents DOM modules</td>
<td>N/A</td>
<td>supported</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>StyleSheets</strong>: association between a style sheet and a document</td>
<td>N/A</td>
<td>supported</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>CSS</strong>: extensions for cascading style sheets</td>
<td>N/A</td>
<td>supported</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>CSS2</strong>: extensions for Cascading Style Sheets Level 2</td>
<td>N/A</td>
<td>supported</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Events</strong>: generic events system</td>
<td>N/A</td>
<td>supported</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>UIEvents</strong>: basic user interface events</td>
<td>N/A</td>
<td>2000</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>MouseEvents</strong>: mouse device events</td>
<td>N/A</td>
<td>supported</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>MutationEvents</strong>: events for mutations in a DOM tree</td>
<td>N/A</td>
<td>2000</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>HTML Events</strong>: HTML 4.01 events</td>
<td>N/A</td>
<td>supported</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Range</strong>: extensions to manipulate a range in a DOM tree</td>
<td>N/A</td>
<td>supported</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>Traversal</strong>: Alternative traversal methods of a DOM tree</td>
<td>N/A</td>
<td>2000</td>
<td>N/A</td>
</tr>
<tr>
<td><strong>LS</strong>: Loading a document into a DOM tree</td>
<td>N/A</td>
<td>N/A</td>
<td>2004</td>
</tr>
<tr>
<td><strong>LS-Async</strong>: Asynchronous loading of a document into a DOM tree</td>
<td>N/A</td>
<td>N/A</td>
<td>2004</td>
</tr>
<tr>
<td><strong>Validation</strong>: Schema-oriented modification of a DOM tree</td>
<td>N/A</td>
<td>N/A</td>
<td>2004</td>
</tr>
</tbody>
</table>
This is what the browser reads (domC.html).

```html
<html>
  <head>
    <title>DOM Sample C</title>
    <script type="text/javascript">
      var switchCount = 0;
      var adjectives = ["simple","complex","fascinating","unique"]; function switcher() {
        switchCount = (switchCount + 1) % adjectives.length;
        var italicNode = document.getElementById("adjPhrase");
        italicNode.firstChild.nodeValue = adjectives[switchCount];
      }
    </script>
  </head>
  <body>
    <h1>An HTML Document</h1>
    <p>This is a <i id="adjPhrase">simple</i> document.
    <form>
      <button type="button" onclick="switcher()">switch</button>
    </form>
  </body>
</html>
```
This is what the browser displays on screen.