Lecture 4: Document Object Model (DOM)

Wendy Liu

CSC309F – Fall 2007

1

Outline

- What is DOM
- Traversal and Modification
- Events and Event Handling

2

Document Object Model (DOM)

- An defined application programming interface (API) between XHTML documents and application programs
  - An abstract model
- In fact, it is a collection of interfaces
  - Including one for each document tree node type
  - Platform-neutral and language-neutral
  - Support a variety of application programming languages (JavaScript, C++, Java, VB)
  - Can be used in programming to
    - Create documents, navigate the structure, and change/add/delete elements and their content
- Documents in the DOM have a treelike structure
  - There can be more than one tree in a document (but is unusual)
- W3C Standard (DOM 1, DOM 2)
  - Implementation is browser dependent

3

Language Binding

- A support language must have a binding to the DOM constructs
- Correspondence between constructs in the language and elements in the DOM
- JavaScript binding
  - Elements of a document are objects
  - Attributes are represented by properties

4

Example: Element Binding

```
function Element() {
   //  properties and methods of the Node object
   ...
   // properties and methods of Element object
   var tagName ;
   getAttribute = elementGetAttribute(name);
   setAttribute = elementSetAttribute(name, value);
   removeAttribute = elementRemoveAttribute(name);
   getAtrributeNode = elementGetAttributeNode(name);
   setAtrributeNode = elementSetAttributeNode(newAttr);
   ...
}
```

5

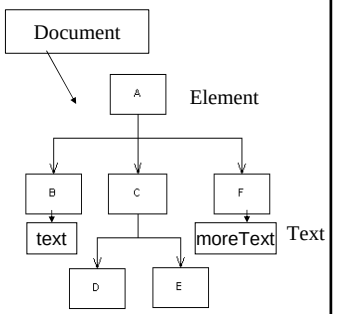
Traversal and Modification

Window, Document, and Element

6

DOM Representation of XML

```
<A>
  <B>text</B>
  <C>
    <D>child of C</D>
    <E>another child of C</E>
  </C>
  <F>moreText</F>
</A>
```



7

Window

- The `Window` object represents the window that displays the document
  - Properties of the `Window` object are visible to all JavaScript scripts that appear in the window's XHTML document
  - The `Window` object include all of the global variables as properties
  - It is the unique *global object* which is created before control enters any execution context
  - Has a self referential property: window
- All JavaScript variables are properties of some object

8

Document

- Represent the displayed XHTML document
  - A central interface
  - Create new elements and text nodes

9

Element

- Elements of a document are Element objects
- Can be used to
  - Navigate document
  - Change document tree structure

10

Identifying Existing Elements

```
<body>
   <form name="myForm">
         <p><input type="button" name="turnItOn" id="onButton"/></p>
   <form>
</body>
```

- Location in tree
  - document.forms[0].element[0]
- ID
  - document.getElementById("onButton")
- Tag name
  - document.getElementsByTagName("input")
    - Returns a node list (could be empty)
- Attribute name
  - document.getElementByName("turnItOn")
    - Discouraged in XHTML
  - document.myForm.turnItOn
    - HTML usage; deprecated in XHTML

11

Caution: Inconsistency in the Book

- Sebesta P89 – first paragraph
  - "(XHTML 1.1) form elements must still use the name attribute because it is used in processing form data."
- Sebesta P194 – middle two paragraphs
  - "…the XHTML 1.1 standard does not allow the name attribute in the form element, even though the attribute is now legal for form elements.
    … Although name attributes are not allowed on form elements, name attributes are often required on the elements in a form."
- The official version (W3C DOM 2 HTML Specification)
  - "getElementsByName
    With [HTML 4.01] documents, this method returns the (possibly empty) collection of elements whose name value is given by elementName.
    In [XHTML 1.0] documents, this method only returns the (possibly empty) collection of form controls with matching name."

12

Example: DOM Tree

A Table

1	2

Print DOM

```
<html xmlns="http://www.w3.org/
   1999/xhtml">
<head><title>Navigating DOM</title>
<script type"text/javascript">...
</script>
</head>
<body><h1>A Nice Table</h1>
   <table id="mTable“ border="1">
   <tr><td>1</td><td>2</td></tr>
   </table><form><input
   type="button"onclick="showDOM()"
   value="Print DOM" />
   </form></body></html>
```

HTML						
	HEAD					
		TITLE				
			#text			
		SCRIPT				
			#text			
	BODY					
		H1				
			#text			
		TABLE				
			TBODY			
				TR		
					TD	
						#text
					TD	
						#text
		FORM				
			P			
				INPUT		

Modifying DOM

- Node
  - appendChild(newChild)
    - Add node newChild to the end of child list
  - replaceChild(newChild, oldChild)
    - Replace node oldChild with newChild
  - insertBefore(newChild, oldChild)
    - Insert node newChild in child list before node oldChild
  - cloneNode(deep)
    - Return a copy of the node including all descendents if deep==true
  - removeChild(aChild)
    - Remove node aChild from child list including all descendents
  - (all methods in Node are available to Document and Element)
- Document
  - createElement(tagName)
    - Create an Element object of type tagName
  - createTextNode(data)
    - Create a Text object with data
- Element
  - removeAttribute(name)
  - setAttribute(name, value)
- http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113/ecma-script-binding.html

Example: Adding Table Rows

User Table

Add a new row to table.

First Name: Wendy

Last Name: Liu

Add to Table

FIRST NAME	LAST NAME
Wendy	Liu

Events and Event Handling

Event-driven Programming

"Most, if not all, GUI systems and toolkits are designed to be event driven, meaning that the main flow of your program is not sequential from beginning to end."

Robin Dunn
Speech at GUI programming at OSCON2004

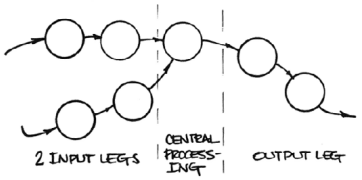
"Hollywood Principle: "Don't call us; we'll call you." ... You implement the interfaces, you get registered. You get called when the time is right."

Dafydd Rees
http://c2.com/cgi/wiki?HollywoodPrinciple

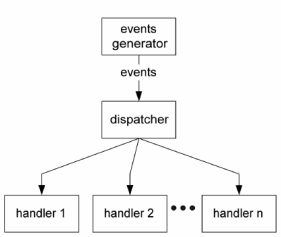
17

Structured vs. Event Driven

Structured Program



Event Driven Architecture



18

Event-Driven Execution

- JavaScript programs are typically event-driven
- Execution can be triggered by events that occur on the Web page, usually as a result of user actions
  - onclick, ondblclick, onkeydown, onload, onmouseover, onsubmit, onresize, …
- Events are associated with XHTML tag attributes
  - The onclick event can be associated with <a> and form <input> tags

19

Partial List of Events

- Clipboard
  - oncopy, oncut, onpaste
- Keyboard
  - onkeydown, onkeyup, onkeypress
- Mouse
  - onmousedown, onmouseup, onmousemove
- Other
  - onfocus, onblur, load, unload

20

Registering Event Handlers

- By assigning the event handler script to an event tag attribute
  - …
- By assigning the event handler script to an event property of an object
  - document.getElementById("myLink").onmouseover = popup;

21

DOM 2 Event Model

- Modularized interface
  - HTMLEvents
    - abort, blur, change, error, focus, load
  - MouseEvents
    - click, mousedown, mousemove, mouseover
- Support event propagation
  - Not available in DOM 0 (i.e. HTML event model)

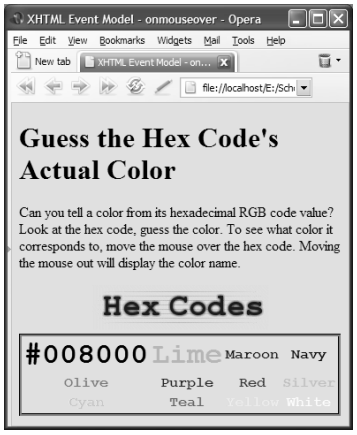
22

Event Propagation

- Basic flow
  - Capturing phase
    - The process by which an event can be handled by one of the event's target's ancestors before being handled by the event's target
    - Capture operates from the top of the tree, generally the Document, dispatching the event downward to the target node
  - At target
    - When the event reaches the target, any event listeners registered on the EventTarget are triggered
  - Bubbling phase
    - The process by which an event propagates upward through its ancestors after being handled by the event's target
    - Bubbling events will then trigger any additional event listeners found by following the EventTarget's parent chain upward, until Document
      - Some events are non-bubbling: load, unload
- Cancel further propagation
  - Event.stopPropagation();
  - Can be used in the capture and bubbling phase

23

Example: Event Propagation



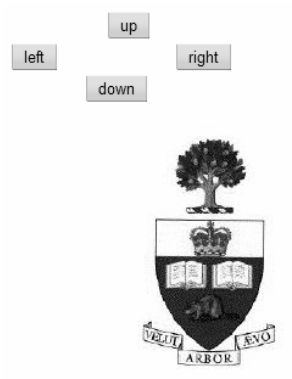
24

Changing Style Attributes

- CSS1 and CSS-P (CSS - Positioning) are scriptable from JavaScript
  - Allow dynamic style update on XHTML elements
- The ***style*** property
  - position
  - top
  - left
  - visibility
  - backgroundColor
  - color
  - zIndex
    - The effect of the 3rd dimension
    - The largest value is on the top

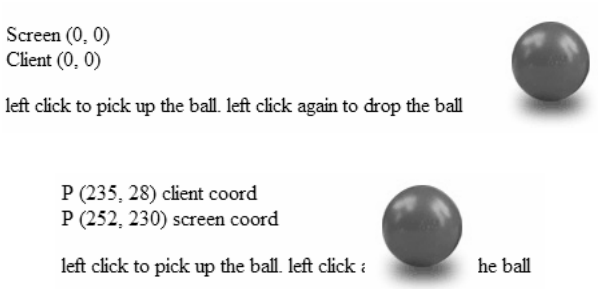
25

Example: Move Image on Screen



26

Tracking Mouse Movements



- Track mouse position on screen
- Drag and drop ball on click
- Events onmousemove and onclick

27

Slow Movement of Element

- Window methods
  - setTimeout
    - setTimeout("move()", 20);
    - Delays for 20 ms before move() is called
  - setInterval
    - setInterval("move()", 20);
    - Cause move() to be repeatedly executed at 20 ms intervals

28