JavaScript and HTML Documents

1 JavaScript Execution Environment

- **JavaScript executing in a browser**
- **The Window object represents the window displaying a document**
  - **All properties are visible to all scripts**
  - **Global variables are properties of the Window object**
  - **There can be more than one Window object**
    - **Global variables depend on which Window is the context**
- **The Document object represents the document displayed**
  - **The document property of Window**
  - **Property arrays for forms, links, anchors, …**
- **The frames property of Window**

2 Document Object Model

- **DOM Levels**
  - **DOM 0: informal, early browsers**
  - **DOM 1: XHMTL/XML structure**
  - **DOM 2: event model, style interface, traversal**
  - **DOM 3: content model, validation**
- **DOM specifications describe an abstract model of a document**
  - **Interfaces describe methods and properties**
  - **The interfaces describe a tree structure**
  - **Different languages will *bind* the interfaces to specific implementations**
    - **The internal representation may not be tree-like**
    - **In JavaScript, data are represented as properties and operations as methods**

2 Example

- **The HTML document on page 192 is shown as a conceptual tree**
- **Nodes of the tree will be JavaScript objects**
- **Attributes of elements become named properties of element node objects**
  - **<input type="text" name="address">**
  - **The object representing this node will have two properties**
    - **type property will have value "text"**
    - **name property will have value "address"**

3 Element Access in JavaScript

- **Elements in XHTML document correspond to objects in JavaScript**

- **Objects can be addressed in several ways:**
  - `forms` and `elements` array defined in DOM 0
    - **Individual elements are specified by index**
    - **The index may change when the form changes**
  - Using the name attributes for form and form elements
    - **A name on the form element causes validation problems**
    - **Names are required on form elements providing data to the server**
  - Using getElementById with id attributes
    - **id attribute value must be unique for an element**

3 Using forms array

- **Consider this simple form:**

```
<form action = "">
    <input type = "button"  name = "pushMe">
 </form>
```

- **The input element can be referenced as**

```
document.forms[0].element[0]
```

3 Using name Attributes

- **All elements from the reference element up to, but not including, the body must have a name attribute**
- **This violates XHTML standards in some cases**
- **Example**

```
<form name = "myForm"  action = "">
    <input type = "button"  name = "pushMe">
</form>
```

- **Referencing the input**

```
document.myForm.pushMe
```

3 Using id Attribute

- **Set the id attribute of the input element**

```
<form action = "">
    <input type="button"  id="turnItOn">
</form>
```

- **Then use getElementById**

```
document.getElementById("turnItOn")
```

4 Events and Event Handling

- *Event-driven programming* is a style of programming in which pieces of code, *event handlers*, are written to be activated when certain *events* occur

- Events represent activity in the environment including, especially, user actions such as moving the mouse or typing on the keyboard

- An *event handler* is a program segment designed to execute when a certain event occurs

- Events are represented by JavaScript objects

- *Registration* is the activity of connecting a script to a type of event
  - Assign an event attribute an event handler
  - Assign a DOM node an event handler

4 Events, Attributes and Tags

Event	*Tag Attribute*
blur	onblur
change	onchange
click	onclick
focus	onfocus
load	onload
mousedown	onmousedown
mousemove	onmousemove
mouseout	onmouseout
mouseover	onmouseover
mouseup	onmouseup
select	onselect
submit	onsubmit
unload	onunload

4 Events, Attributes and Tags

- **Particular events are associated to certain attributes**
- **The attribute for one kind of event may appear on different tags allowing the program to react to events affecting different components**
- **A text element gets focus in three ways:**
  1. When the user puts the mouse cursor over it and presses the left button
  2. When the user tabs to the element
  3. By executing the `focus` method
- **Losing the focus is *blur*ring**

4 Setting a Handler

- **Using a an attribute, a JavaScript command can be specified:**

  ```
  <input type="button" name="myButton"
     onclick=
     "alert('You clicked the button!')"/>
  ```

- **A function call can be used if the handler is longer than a single statement**

  ```
  <input type="button" name="myButton"
     onclick="myHandler()"/>
  ```

5 Handling Events from Body Elements

- **See the load.html example**
- **This example illustrates a script that is run when the page first loads**

6 Handling Events from Button Elements

- **An event can be registered for this tag in two ways**

  ```
  <input type="button" name="freeOffer"
      id="freeButton"/>
  ```

- **Using an event attribute**

  ```
  <input type="button" name="freeOffer"
     id="freeButton"
     onclick="freebuttonHandler()"/>
  ```

- **Assigning to a property of the element node**

  ```
  document.getElementById("freeButton").onclick =
      freeButtonHandler
  ```

  - **Note that the function name, a reference to the function, is assigned**
  - **Writing freeButtonHandler() would assign the return value of the function call as the handler (possible, but unlikely)**

6 Checkboxes and Radio Buttons

- **Example radio_click.html illustrates a script that displays an alert when a radio button is clicked**
  - Note that a parameter is passed to the handler function
- **In example radio_click2.html, a reference to the handler function is assigned to the appropriate property of each element node**
  - Note that the no parameters are passed to the function when called by the JavaScript system
  - The handler code must identify the element that caused the call
- **The handler call can be enclosed in an anonymous function**
  - `dom.elements[0].onclick = function() {planeChoice(152)};`
  - `Example radio_click2x.html`

6 Comparing Registration Methods

- **Assigning to an attribute is more flexible, allowing passing parameters without having to create an anonymous function**

- **Assigning to a node property helps separate HTML and code**

- **Assigning to a node property allows reassignment later if the handler needs to be changed**

7 Handling Events from Text Box and Password Elements

- **By manipulating the focus event the user can be prevented from changing the amount in a text input field**
  - Example nochange.html illustrates 'blurring' a field whenever it gains focus
- **This is possible to work around**
  - Copy the page but leave out the validation code
  - Simulate an HTTP request directly with socket-level programming
  - If the validity of data is important, the server needs to check it

7 Validating Form Input

- **Validating data using JavaScript provides quicker interaction for the user**

- **Validity checking on the server requires a round-trip for the server to check the data and then to respond with an appropriate error page**

- **Handling a data validity error**

  - **Put the focus in the field in question**

  - **Highlight the text for easier editing**

- **If an event handler returns false, default actions are not taken by the browser**

  - **This can be used in a Submit button event handler to check validity and not submit if there are problems**

- **Example pswd_chk.html illustrates validity checking**

  - **Does not work properly in FireFox**

7 Validating Input

- **The validator.html example demonstrates using regular expressions to validate text input**

- **The name is First, Last, Middle-Initial, each part capitialized**
  - **/^[A-Z][a-z]+, ?[A-Z][a-z]+, ?[A-Z]\.?$/**

- **The phone is ddd-ddd-dddd where d is a digit**
  - **/^\d{3}-\d{3}-\d{4}$/**

- **Each pattern uses the ^ and $ anchors to make sure the entire string matches**

8 DOM 2 Event Model

- **DOM 2 is defined in modules**
- **The Events module defines several submodules**
  - **HTMLEvents and MouseEvents aer common**
- **An event object is passed as a parameter to an event handler**
  - **Properties of this object provide information about the event**
  - **Some event types will extend the interface to include information relevant to the subtype. For example, a mouse event will include the location of the mouse at the time of the event**

8 Event Flow

- **DOM 2 defines a process for determining which handlers to execute for a particular event**
- **The process has three phases**
  - **Different event types**
- **The event object representing the event is created at a particular node called the *target node***

8 Event Propagation (continued)

- **Three traversal phases then occur**
- **In the *capturing phase* each node from the document root to the target node, in order, is examined.**
  - If the node is not the target node and there is a handler for that event at the node and the handler is enabled for capture for the node, the handler is executed
- **Then all handlers registered for the target node, if any, are executed**
- **In the *bubbling phase* each node from the parent of the target node to the root node, in order, is examined**
  - If there is a handler for that event at the node and the handler is *not* enabled for capture for the node, the handler is executed
  - Some event types are not allowed to bubble: load, unload, blur and focus among the HTML event types

8 Event Propagation

- **As each handler is executed, properties of the event provide context**
    - The `currentTarget` property is the node to which the handler is registered
    - The `target` property is the node to which the event was originally directed
- **One major advantage of this scheme over DOM 0 is that event handling can be centralized in an ancestor node**
- **For example, a calculator keyboard will have a number of digit buttons**
    - In some GUI frameworks, a handler must be added to each button separately
    - In DOM 2, the buttons could be organized under a single node and the handler placed on the node

8 Event Handler Registration

- **Handlers are called *listeners* in DOM 2**

- **addEventListener is used to register a handler, it takes three parameters**

  - **A string naming the event type**

  - **The handler**

  - **A boolean specifying whether the handler is enabled for the capture phase or not**

8 Example of DOM 2 Event Model

- **The validator2.html example modifies the validation example to use DOM 2 style event handling**

9 The `navigator` Object

- **Properties of the `navigator` object allow the script to determine characteristics of the browser in which the script is executing**
- **The `appName` property gives the name of the browser**
- **The `appVersion` gives the browser version**

9 Output From `navigate.html`

- **Note that the browser is actually FireFox and the version is 2.0.0.4**

[JavaScript Application]

The browser is: Netscape
The version number is: 5.0 (Windows; en-US)

OK

10 DOM Tree Traversal and Modification

- **Each element in an XHTML document has a corresponding ELEMENT object in the DOM representation**

- **The ELEMENT object has methods to support**
  - **Traversing the document, that is, visiting each of the document nodes**
  - **Modifying the document**
    - **For example, removing and inserting child nodes**

10 DOM Tree Traversal

- **Various properties of `Element` objects are related nodes**
- `parentNode` **references the parent node of the** `Element`
- `previousSibling` **and** `nextSibling` **connect the children of a node into a list**
- `firstChild` **and** `lastChild` **reference children of an Element**
  - These would be text nodes or element nodes contained in the Element

10 DOM Tree Modification

- The `insertBefore` method inserts a new child of the target node

- `replaceChild` will replace a child node with a new node

- `removeChild` removes a child node

- `appendChild` adds a node as a child node at the end of the children