Client-Side Scripting

- So far, the browser has only passively displayed content.
- It is also possible to download a program to the browser, and have it execute on the client browser.
  - JavaScript / Jscript / ECMAScript
  - VBScript
  - TCL
- Example: JavaScript fact.html

JavaScript

- a.k.a. ECMAScript
  - http://www.ecma.ch/ecma1/STAND/ECMA-262.HTM
  - an OO programming language for performing computations and manipulating computational objects within a host environment
    - not intended to be computationally self-sufficient
    - A web browser provides an ECMAScript host environment for client-side computation including, for instance, objects that represent windows, menus, pop-ups, dialog boxes, text areas, anchors, frames, history, cookies, and input/output.
      - http://developer.netscape.com/docs/manuals/communicator/jsref/contents.htm
    - Further, the host environment provides a means to attach scripting code to events such as change of focus, page and image loading, unloading, error and abort, selection, form submission, and mouse actions.

Relationship to Java

- Netscape originated *LiveScript*, later renamed to JavaScript at the last minute.
  - Invented by Brendan Eich at Netscape
  - Perceived competition with Sun Microsystems Java for client-side scripting
- Microsoft has a similar thing called JScript
- Complementary
  - JavaScript
    - Cannot draw, multi-thread, network or do I/O
  - Java
    - Cannot interact with Browser or control content

Client-Side JavaScript

- To interact with the host environment, there needs to be an API
  - DOM
    - http://www.w3.org/TR/REC-DOM-Level-1
  - Browser-specific
    - no standards
    - Netscape & Microsoft stay somehwat in-step
      - need a reference that distinguishes NN, IE, DOM support
        - » e.g., O'Reilly Dynamic HTML
        - » use only common features
    - http://developer.netscape.com/docs/manuals/communicator/jsref/contents.htm

JavaScript Security

- Language/API limitations:
  - No file/directory access defined in the language
  - No network access either
    - load URLs
    - send HTML form data to
      - web servers, CGI scripts, e-mail addresses
- Leaves privacy issues:
  - Many restrictions:
    - cannot read history
    - cannot hide/show menubar, status line, scrollbars, …
    - cannot close a window not opened by itself
    - 'same origin policy'
      - can only read props of documents and windows from the same place: host, port, protocol
    - …

Signed Scripts

- Latter set of restrictions can be removed:
  - in certain cases explicit dialog boxes will ask for user confirmation
    - e.g., close() other window
  - In other cases user can grant privileges to signed scripts (more on this later in the course).
    - UniversalBrowserRead/Write
    - UniversalFileread,
    - UniversalSendMail
    - UniversalPreferencesRead/Write

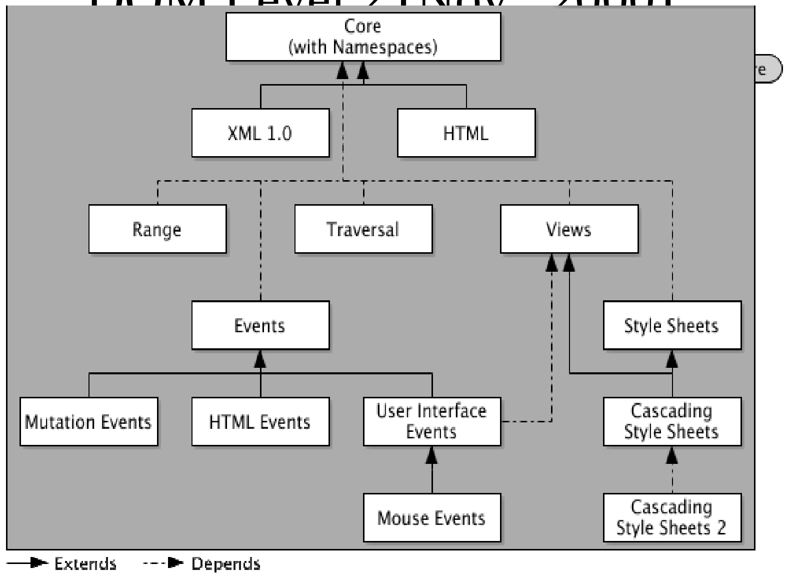
Document Object Model

- W3C Standard
- The Document Object Model is a platform- and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure and style of documents.
- The document can be further processed and the results of that processing can be incorporated back into (or even replace) the presented page.

DOM

- DOM Level 1 now being fully implemented in IE6.
  - Two parts:
    - Core
      - a minimal set of objects and interfaces for accessing and manipulating document objects (mainly XML oriented)
    - HTML
      - extends the Level 1 Core API to describe objects and methods specific to HTML documents
  - The key differences between the core DOM and the HTML application of DOM is
    - the HTML Document Object Model exposes a number of convenience methods and properties that are consistent with the existing models and are more appropriate to script writers.
    - In many cases, these enhancements are not applicable to a general DOM because they rely on the presence of a predefined DTD.

DOM Level 2 (Nov. 2000)



How the DOM is Defined

- The DOM is defined using IDL (Interface Definition Language)
  – IDL was originally defined for use in CORBA
  – Java RMI and Microsoft DCOM can also use IDL
- IDL is then mapped into
  – Java
  – ECMAScript (JavaScript to the rest of us)

Example IDL Definition

```
interface HTMLTitleElement: HTMLElement {
  attribute DOMString text;
};
```

Example Java Language Binding

```
package org.w3c.dom.html;
public interface HTMLTitleElement
  extends HTMLElement {
    public String getText();
    public void setText(String text);
}
```

Example JavaScript Language Binding

- **Object HTMLTitleElement**
  - **HTMLTitleElement has the all the properties and methods of the HTMLElement object as well as the properties and methods defined below.**
  - **The HTMLTitleElement object has the following properties:**
    - **text**
      - This property is of type **String**.

Core - Document

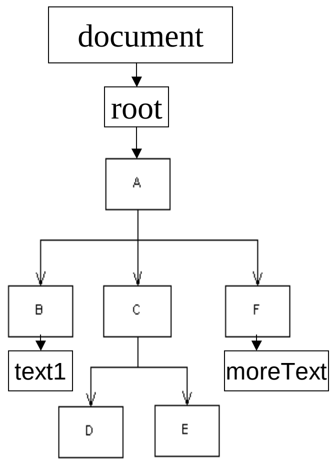
- The central interface is Document
  - Contains a number of factory methods for creating elements, text nodes, comments, CDATA sections, PI's attributes, and entity references.
  - To traverse the tree structure, start by
    - `Element documentElement;`
    - `NodeList getElementsByTagName(name);`

Core - Node

- Represents the various types of nodes found in a document
  - – methods for traversing to nearby nodes
  - – methods for adding new nodes
- Various specializations for the various types of nodes.
  - – ProcessingInstruction, Comment, CDATA, Notation, …

DOM Representation of HTML

```
<A>
  <B>text1</B>
  <C>
    <D>child of C</D>
    <E>another child of C</E>
  </C>
  <F>moreText</F>
</A>
```



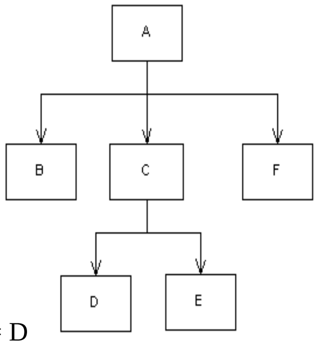
DOM representation

DOM Representation of HTML

Node object properties:

A.childNodes[0] = B
A.childNodes[2] = F
A.firstChild = B
A.childNodes.length = 3
B.parentNode = A
B.nextSibling = C
F.nextSibling = null
A.childNodes[1].firstChild = D
E.parentNode.parentNode = A

getElementById(“id")



Modifying:

insertBefore()
replaceChild()
removeChild()
appendChild()
cloneNode()

Client Environment

- DOM is still somewhat theoretical
  - extremely limited support for the standard
  - IE6 is a big improvement
- Most uses of JavaScript make use of the client-side environment objects supplied (somewhat incompatibly) by the browsers

Scripting Languages in HTML pages

- Intrinsic event scripts

```
<img src="image.gif" onclick="…" />
```

- <script> element scripts

```
<script type="text/javascript" src="script.js"/>
```

  – deprecated

```
<script language="javascript" src="script.js">
```

```
<script type="text/javascript"> <!--
    … javascript code …
    // end of script -->
</script>
<noscript>
    … alternate content …
</noscript>
```

Default Scripting Language

- Browser must determine the default scripting language for intrinsic event scripts
  - As a META declaration
    - <META http-equiv="Content-Script-Type" content="text/javascript">
  - In HTTP header:
    - Content-Script-Type: text/javascript
  - otherwise: document is incorrect
    - although browsers may interpret them nonetheless but are not required to
- For <script> element scripts
  - must specify a scripting language in each one
  - if you don't browser may interpret them nonetheless

Script URL

- May also embed a script in a URL:
  - javascript: alert("hello")

Document Replacement

- Scripts that are executed when a document is loaded may be able to modify the document's contents dynamically.
  1. All <script> elements are evaluated in order as the document is loaded.
  2. All script constructs within a given <script> element that generate SGML CDATA are evaluated. Their combined generated text is inserted in the document in place of the <script> element.
  3. The generated CDATA is re-evaluated.
- *"document.write" or equivalent statements invoked in intrinsic event handlers create and write to a new document rather than modifying the current one.*

Document Replacement example

- HTML documents are constrained to conform to the HTML DTD both before and after processing any <script> elements.

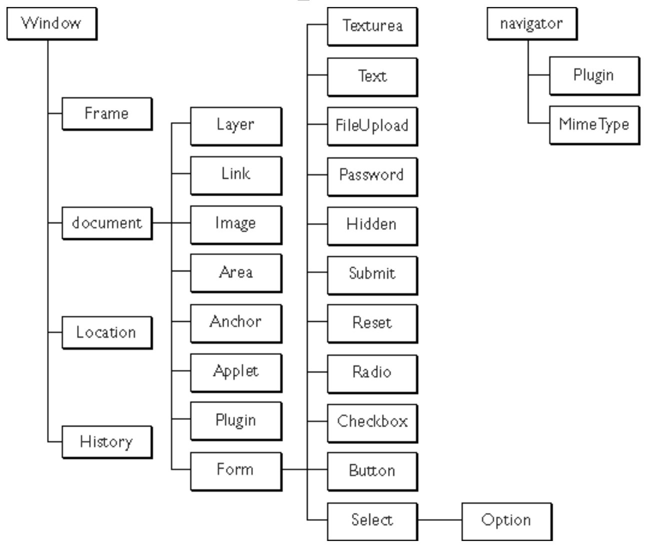
```
<title>Test Document</title>
<script type="text/javascript">
   document.write("<p><b>Hello World!<\/b>")
</script>
```

Has the same effect as this HTML markup:

```
<title>Test Document</title>
<P><B>Hello World!</B>
```

- e.g.2: replace.html

Netscape DOM



Window Object

- The Window object represents the window (or frame) in which a JavaScript program executes.
  - is the global object
    - e.g., the following mean the same
      - `var foobar = 42;`
      - `window.foobar = 42;`
      - `self.foobar = 42;`
  - However, can access the Window object of other windows/frames.

Sample Window Properties

- document
  - refers to the HTML currently displayed.
- location
  - refers to the URL from whence it came.
- window, self, parent, top, opener, frames[]
  - refers to other windows in the hierarchy
- navigator
  - information about the browser and computer
- screen
  - info about the display on which the window is shown
- history
  - history of URLs visited

Sample Window Methods

- alert(), confirm(), prompt()
- focus(), blur(), close(), open()
- moveTo(), moveBy()
- setInterval(), setTimeout()

- Example: alert.html

Document Object

- The Document object is a DOM object that represents the currently displayed page.
- dom.html
  - First set of colour buttons use deprecated DOM API
  - Second set of buttons set stylesheet colour

Dynamic HTML

- CSS1 and CSS-P (CSS - Positioning) are scriptable from JavaScript
  - allows HTML elements to float around and grow and shrink.
  - Merged in the CSS2 specification
- dynamic.html
- Browser issues
  - Netscape prematurely supported the <layer> tag in NN4. Not included in the final standard.
  - IE4 supports CSS-P quite well
  - Expose slightly different APIs to JavaScript