B211 Internet Computing

Web Scripting Languages

Lecture Outline

- Client-side Scripting
  - JavaScript and Jscript
  - VBScript
  - Dynamic HTML (DHTML)
- Server-side Scripting
  - ASP
  - PHP
  - Perl and CGI
  - JSP
  - ColdFusion

JavaScript

- JavaScript was originally created by Netscape to increase the capabilities of browsers and add interactivity to web sites.
- Today, JavaScript is used to
  - process data collected in HTML forms right on the user's computer, without involving a server
  - add interactivity to graphics;
  - change page elements on the fly based on user input; and
  - integrate HTML data more tightly with other Web technologies (for example, Java applets and ActiveX controls).

Before JavaScript

- Before JavaScript (and other similar technologies) came along, web site processing was done through CGI.
  - Involves uneccessary network traffic
  - Places heavy burden on servers
  - … especially on processing which is only relevant to the client-side.

Versions of "JavaScript”

- JavaScript 1.0 (then called LiveScript) originally introduced by Netscape as part of Navigator 2.0
- JavaScript 1.1 came with Navigator 3.0
- JavaScript 1.2 came with Navigator 4.0 (packaged with Netscape Communicator 4.0).
- JavaScript 1.3 - Netscape Communicator 4.06 or later

Versions of "JavaScript”

- Microsoft implemented its own version of JavaScript called *JScript* version 1
  - came with MS Internet Explorer (MSIE) 3.0 - except for the Mac version!
  - JScript 1 **more or less** compatible with JavaScript 1.0
- JScript version 2 came with some (not all) copies of MSIE 3.02, and all copies of MSIE 4.0
- JScript 3 came with MSIE 4.0
  - JScript 3 **more or less** compatible with JavaScript 1.2

The ECMAScript Standard

- Due to pressures from developers who want standardized JavaScript:
  - Netscape proposed their JavaScript language specifications as a standard to ECMA (European Computer Manufacturers' Association)
  - Microsoft added their suggestions to the proposed standard using features of Jscript
- In June 1997, ECMA released a standard officially known as ECMA-262 for a language standard called ECMAScript.
- ECMA-262 closely resembles JavaScript 1.1, **but not exactly the same**.

Support for ECMAScript

- Netscape and Miscrosoft have pledged to support ECMA-262 in all future versions of their browsers.
- MSIE 4.0 and above claims to be 100% ECMAScript-compliant, **with added features**
  - ie. scripts written which conform to ECMA-262 should run under MSIE 4.0

Support for ECMAScript (cont'd)

- ECMAScript is hardly ever mentioned except in technical circles.
- Since JavaScript/Jscript/ECMAScript is not a technology actively developed by non-proprietary bodies like W3C, its direction is completely determined by Netscape and Microsoft and what they choose to support in their browsers.

VBScript

- VBScript stands for *Visual Basic Scripting Edition*
  - a subset of the Microsoft Visual Basic programming language
- Supported and executable only in Microsoft Internet Explorer.
- Used mainly for adding interactivity to web pages, especially for linking to ActiveX controls.

Dynamic HTML (DHTML)

- *Dynamic HTML* is a term used to refer to technologies which allows a web page to change after it is downloaded by a browser.
- It is commonly a combination of some of the technologies we have already seen:
  - HTML
  - style sheets
  - DOM
  - scripting languages/technologies like JavaScript, VBScript, etc.

DHTML (cont'd)

- The two main aspects of DHTML:

1. Style modifications
   - dynamically on-the-fly altering the positioning and layout of content on a page.
2. Event handling
   - linking user events (eg. mouse clicks) to changes in page.

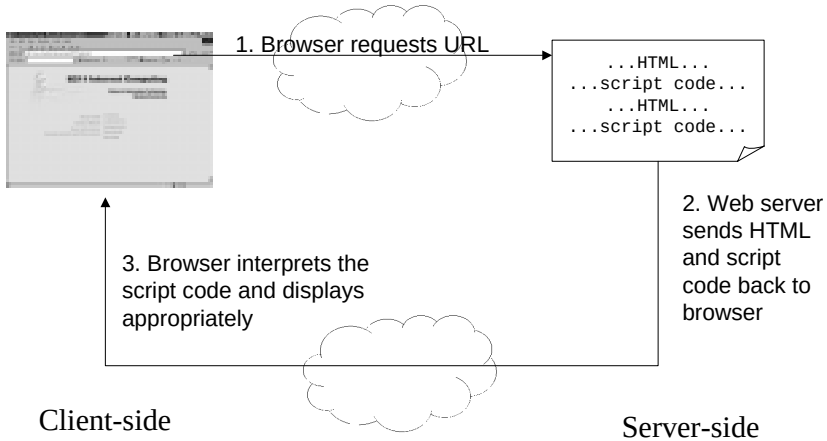
Support for DHTML

- Microsoft and Netscape are concentrating on different ways to implement DHTML in their browsers.
- Netscape is concentrating on the new HTML/CSS tag <LAYER>
  - The <LAYER> and associated tags were designed to mark-up elements which can be positioned anywhere in a window, and overlapping each other.
- Microsoft is concentrating on using scripts (VBScript and JScript) to process CSS elements, and using its proprietary ActiveX controls.

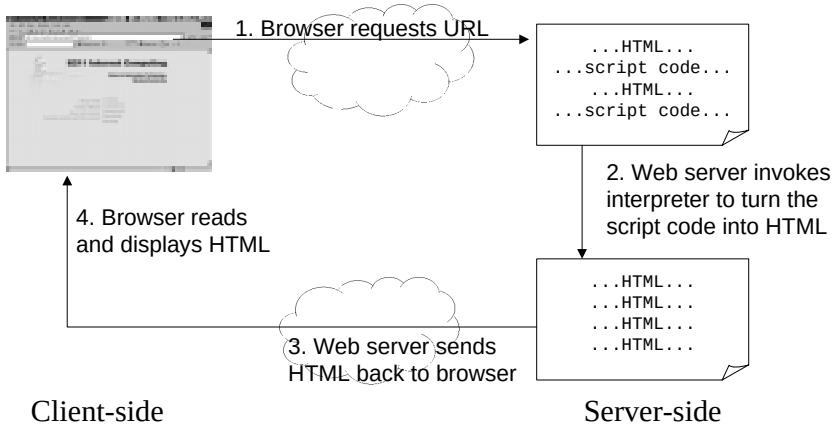
Server-side Scripting

- There are many situations where client-side technologies is not enough
  - Refer to introductory lecture on Perl/CGI for the importance of server-side scripting.
- Today, most of the powerful web scripts are written using server-side scripting languages:
  - Perl/CGI
  - PHP (PHP Hypertext Preprocessor)
  - Microsoft's ASP (Active Server Pages)
  - JSP (Java Server Pages) and Java Servlets
  - Macromedia's (previously Allaire's) ColdFusion
  - Apache's XSSI (eXtended Server-Side Includes)

How Client-side Scripting Works



How Server-side Scripting Works



How Server-side Scripting Works

- Like all technologies, all the technologies have slight variations to the above, but the basis is:
    - Processing of script code is done on the server-side, and
    - a **static** document gets sent to the client.

Recognizing Which Server-side Technology

- Most server-side technologies makes use of standard file extensions for their script files. You can recognise them when looking at the URL in your web browser's URL box.

Language	Standard file extensions
Perl/CGI	.cgi or .pl
PHP	.php
ASP	.asp
ColdFusion	.cfm or .cfml
JSP	.jsp
XSSI	.shtml

Choosing Server-side Technologies

- The following are common reasons developers adopt the different options:
    - ASP
        - familiarity with Microsoft development platform.
        - take advantage of MS-Windows technologies (eg. COM objects and ActiveX).
        - Commercial support by Microsoft Developer Network.
    - Perl/CGI
        - familiarity with Perl, and advantages of Perl language (web communication is text processing)
        - Perl's rich development community
        - open-source and completely free
    - PHP
        - open-source and completely free
        - Tight integration with MySQL (an open-source, free database)

Choosing Server-side Technologies

- JSP
    - can use the full Java language, and it's libraries.
- ColdFusion
    - Simple tag syntax (ColdFusion Markup Language CFML)
    - Powerful IDE (integrated development environment)
- XSSI
    - Natural extension to the Apache web server

- Developers preferring each platform ALL claim their respective language is easiest to learn!

Choosing Server-side Technologies

- The following are common reasons developers reject certain server-side scripting technologies:
  - ASP
    - restricted to being served through Microsoft's IIS web server (Note: there are now versions of ASP for other web servers, although not popular)
    - cost of development tools and web server.
  - CGI
    - large process and memory requirements
  - ColdFusion
    - MS-Windows platform.
    - Not built for low level programming - built to work best through using the visual IDE and CFML.
    - cost of package.

Choosing Server-side Technologies

- JSP
  - Restricted to Java programming language
- XSSI
  - A very thin language, only appropriate for small processing.

Choosing Server-side Technologies

- I make not absolute claims on which technology is best:
  - …even though I have my favourites for different purposes.
  - Note in the previous 2 pages, I use the term "common reasons" instead of just the word "reasons"
  - If and when your programming-proficiency level is good enough to need significant server-side scripting, you should decide for yourself by having a preliminary look at the details.
  - Don't make your decisions on technologies based on what other people say.

Further Reading

- Netscape's JavaScript guide and reference
  - http://developer.netscape.com/docs/manuals/js/client/jsguide/index.htm
- Microsoft's Windows Scripting Technologies
  - http://hotwired.lycos.com/webmonkey/99/46/index1a.html
- A overview tutorials on various server-side scripting options:
  - http://hotwired.lycos.com/webmonkey/99/46/index1a.html