Client-Side Web Programming (Part 1)

Robert M. Dondero, Ph.D.
Princeton University
Objectives

- You will learn about:
  - Client-side web programming, alias...
  - Programming the browser, via...
  - Java Applets
  - JavaScript
Motivating Problem

Problem:
- In Pennypack app, missing author name is an error
- In principle, browser could:
  - Detect missing author name
  - Warn user w/o contacting server
- And thereby:
  - Provide user feedback more quickly
  - Reduce server load

Solution:
- Java Applets or JavaScript
Part 1:
Java Applets
Java Applets

- **Applet**
  - Java code that is executed by a browser
  - A subclass of **JApplet**
    - Which is a subclass of **Applet**
      - Which is a subclass of **Panel**
Java Applets History

- Time line:
  - Applets were part of initial version of JDK
  - Applets ran in Sun's HotJava browser
  - Applets ran in Netscape Navigator
    - Java became popular
    - "The language of the Internet"
Java Applets History

- Time line (cont.):
  - Netscape support lagged
  - Microsoft adoption lagged
    - IE sometimes supported old versions
    - IE sometimes didn't support Applets at all
  - Sun released browser "Java Plug-In"
    - Enhanced many browsers to handle Applets
PennypackAppletValidate App

- See **PennypackAppletValidate** application
  - book.py, database.py, common.py
  - index.html
  - searchform.cgi, searchform.py
  - SearchApplet.java
  - searchresults.cgi, searchresults.py

- Some notes...
PennypackAppletValidate App

- **Server side:**
  - Written in CGI/Python
    - Arbitrary
    - Could have used CGI/Java, PHP, …
  - **Still must validate form data!!!**
    - User might browse to searchresults.cgi directly
searchform.py

- Generates page containing `<applet>` tag
- Applet tag commands browser to load specified Applet class and create Applet object
- Must specify height and width of Applet
- Cannot resize Applet!!! Take your best guess!!!
SearchApplet.java

- Similar to Swing application
- `init()` method instead of `main()` method
- `applet.getAppletContext()`
  - Returns the "context" of the Applet object; essentially the browser
- `applet.getAppletContext().showDocument(url)`
  - Uses URL object to command browser to fetch page with given url
- Validates author upon event
Part 2: JavaScript
JavaScript

- **Who**: Brendan Eich of Netscape
- **When**: 1995
- **Why**: client-side scripting language for web pages
JavaScript Overview

- Client-side scripting language
  - Embedded into HTML code
  - Interpreted by Netscape browser, then all popular browsers
- Usage:
  - `<script> JavaScriptCode </script>`
  - `<script src="url"></script>`
  - `<sometag onSomeEvent ="JavaScriptCode">`
JavaScript vs. Java

- JavaScript name was originally **LiveScript**
  - Later changed to JavaScript to capitalize on popularity of Java
- JavaScript is related to Java only superficially
Yet another programming language!!!

- No time (and inappropriate) to cover thoroughly
- See my personal "Subset of JavaScript"
  - Link on web pages
- Some simple examples that might help...
JavaScript Examples

- See hello1.html
  - The <script> tag
- See hello2.html
  - Functions and function calls
- See hello3.html
  - Event handling
  - Works partially in Firefox!
  - Fails in Epiphany!!
  - Welcome to client-side web programming!!!
JavaScript Examples

- See hello4.html and hello4.js
  - JavaScript code stored externally
  - Referenced by URL
    - Here a simple file name

- See helloerror.html
  - Error reporting
  - In Firefox: Error Console (Tools → Error Console)
JavaScript Examples

- See `control.html`
  - Control statements
- See `datastructures.html`
  - Data structures
    - Arrays
    - Associative arrays
- See `objects.html`
  - Classes, object references, objects
  - Unusual!
JavaScript Global Objects

- Constructors
  - Boolean(), Number(), String(), Array(), Object(), Function(), RegExp(), Date()

- Errors
  - Error(), EvalError(), RangeError(), ReferenceError(), SyntaxError(), TypeError(), URIError()
Non-Constructor Functions
- `decodeURI()`, `decodeURIComponent()`, `encodeURI()`, `encodeURIComponent()`, `eval()`, `isFinite()`, `isNaN()`, `parseFloat()`, `parseInt()`

Other
- `Infinity`, `Math`, `NaN`, `undefined`
Document Object Model (DOM)

- A programmatic **model** of the current **document**
- Each element and attribute is represented as an **object**
- Objects arranged in a tree
  - **window**
    - **document**
    - **Element**
- (Sadly) Browser dependent
  - There is a standard, but some vendors don't observe
JavaScript DOM (Gecko Engine)

- **window** object
  - Represents the browser window
  - Important property: `document`
  - Event handlers: `onabort`, `onbeforeunload`, `onblur`, `onchange`, `onclick`, `onclose`, `oncontextmenu`, `ondragdrop`, `onerror`, `onfocus`, `onhashchange`, `onkeydown`, `onkeypress`, `onkeyup`, `onload`, `onmousedown`, `onmousemove`, `onmouseout`, `onmouseover`, `onmouseup`, `onbeforepaint`, `onpaint`, `onpopstate`, `onreset`, `onresize`, `onscroll`, `onselect`, `onsubmit`, `onunload`, `onpageshow`, `onpagehide`
**document** class

- Represents the current document. Contains references to element objects
- Important methods: `write()`, `getElementById()`
- Event handlers: `ononline`, `onoffline`, `onreadystatechange`
element class

- Represents an HTML element
- Contains references to attributes and children elements
- Event handlers: onafterscriptexecute, onbeforescriptexecute, oncopy, oncut, onpaste, onbeforeunload, onblur, onchange, onclick, oncontextmenu, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, onresize, onscroll
- Specific elements have additional handlers
See printdomtree.html
- Recursively traverses DOM tree
- Prints rep of current document
- (Try printing contents of #text and #comment nodes)
See events.html

- Event handlers on <body> tag
- Event handlers on <form> tag
- Event handlers on <input> tag
- Ways to reference an object within the DOM
JavaScript for Web Programming

- Common uses of JavaScript in Web programming:
  - Form input validation (now)
  - AJAX (later)
  - ...

See **PennypackJavaScriptValidate** application

- book.py, database.py, common.py
- index.html
- searchform.cgi, **searchform.py**
- searchresults.cgi, searchresults.py

Some notes...
Server side:
- Written in CGI/Python
  - Arbitrary
  - Could have used CGI/Java, PHP, …
- Still must validate form data!!!
  - User might browse to searchresults.cgi directly

searchform.py
- Python multi-line string ("" … ")
- Validates author upon form submission
JavaScript Parting Thoughts

- C/Java-like syntax
- Weakly typed
- Object-oriented
  - Unusual object model; very dynamic
- Many incompatibilities among browsers
  - DOM may vary
  - JavaScript may vary!
  - (HTML may vary!!)
How to handle incompatibilities?
- Keep it simple
- Use a JavaScript generator
  - E.g. Google Web Toolkit
- Use a JavaScript library
  - E.g. JQuery
Security in Client-Side Web Pgms

- Java Applets and JavaScript:
  - Run in a sandbox
    - Can perform only web-related tasks
    - Cannot interact with local file system
  - Enforce same origin policy
    - Code from one website cannot interact with another website
  - Both allow exceptions to those policies
Applets vs. JavaScript

- **Applet pros:**
  - Portable across (supporting) browsers
  - More powerful
    - Java lang & libraries are more powerful than JavaScript lang & predefined objects & predefined classes

- **JavaScript pros:**
  - Launches faster
  - Simpler
  - Better integrated into browser (e.g. resizing)
Summary

- We have covered:
  - Client-side web programming, alias...
  - Programming the browser, via...
  - Java Applets
  - JavaScript