ITEC 136
Business Programming Concepts

Week 1
Module 1: Overview of HTML and Introduction to JavaScript

Agenda

• Course overview
• This week’s expected outcomes
• This week’s topics
• This week’s homework
• Upcoming deadlines
• Questions and answers
Course Overview

- Course Outcomes
  - Explain the stages of the software lifecycle
  - Design solutions using top-down stepwise refinement
  - Implement solution algorithms using selection and repetition control structures
  - Translate algorithms into clearly documented and modularized programming language code
  - Document and Debug Code
  - Generate and execute test plans
Course Overview

• Books
  • Primary: JavaScript by Gosselin
  • Secondary: Learning JavaScript by Powers

• Additional
  • Key points on web site
  • Web, tutorials, etc.

• Why is this important?
  • Solid grounding in programming logic and design
  • Understanding of the software development lifecycle
  • Automation of tasks
  • Pre-requisite for other language-based coursework
Why JavaScript

- Ubiquitous on systems since 1996
- Syntax derived from Java/C++ but is neither!
- Dynamic language – change easily
- Key Component for Web 2.0 interfaces
- Can be embedded in other programs
- It can be fun!

JavaScript for Coding

- Why not language X or Y?
  - Not a specific language-focused course
  - Runs Within Internet Browser
  - Principles apply to most languages
    - Variables
    - Functions
    - Control structures
    - Testing
Course Overview

• Course Structure
  • Lots of practice (“shampoo” method)
    • Reading
    • FranklinLive presentations
    • Homework Exercises
    • Lab Exercises
    • Two Exams
    • Final exam

• Tools you will need
  • Aptana Integrated Development Environment (IDE)
  • A standards-compliant web browser (Mozilla Firefox for use with Aptana)
  • Firebug
  • Your textbooks
  • Patience and experimentation! 😊
Other Useful Resources

- Safari Books Online
  - http://www.safaribooksonline.com
  - Also accessible through Franklin library online
- Google (or preferred search engine)
- Any number of web sites
  - http://www.pageresource.com/ (tutorial)
  - http://jennifermadden.com/ (tutorial)
  - http://www.javascriptmall.com/learn/contents.htm (tutorial)
  - http://www.squarefree.com/shell/ (sandbox)
  - http://www.webreference.com/ (reference tutorial)
  - http://www.visibone.com/ (fact sheets $)

VisiBone

- Information packed
- spiral bound
- Covers
  - HTML/XHTML tags
  - JavaScript
  - CSS and Styles
  - Fonts and Colors
  - Document Object Model
  - Regular Expressions
Week 1 Outcomes

- Create and display simple syntactically correct HTML documents
- Describe the execution a JavaScript program in a web browser

Week 1 Preparation

- Have You Already?
  - Read Key Points on Course Web Site
  - Read Chapter 1 in Both Books
  - Reviewed the HTML Primer
HTML Primer

- How does the web work?
  - Request and response cycle
  - Text and binary transfer over HTTP

---

HTML Primer

- How does the web work?
  - Browser sends HTTP request as text

```
GET /article.pl?sid=06/12/19/2256259/ HTTP/1.1
Host: it.slashdot.org
```
HTML Primer

• How does the web work?

• Server sends back HTTP response

HTTP/1.1 200 OK
Date: Thu, 21 Dec 2006 21:31:30 GMT
Content-Type: text/html
Content-Length: 1354

<html>
<body>
<h1>Should JavaScript Get More Respect?</h1>
(more file contents)
</body>
</html>
HTTP and the Web

- Web browser renders the text as a formatted document based on HTML

---

HTML Primer

- HTML: **HyperText Markup Language**
- Use `<` and `>` to delimit an HTML element – usually called a tag
- Open & Close tags around info
  - `<x>info</x>`
- No element? Can use `/` in single tag, **must** have space prior to `/`
  - `<x />`
HTML Examples

- `<title>JavaScript 101</title>`
- `<img src="pic.jpg" />
- `<img src="pic.jpg"></img>`
- `<head><title>Hello World</title></head>`

Two contrasting `<img>` tags. First is becoming more popular.

Can be nested. The `<title>` tag is inside the `<head>` tag.

Full Document HTML

- Document structure

```html
<!DOCTYPE html PUBLIC
 "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>
  <head>
    <title>Should JavaScript Get More Respect?</title>
  </head>

  <body>
    <h1>Should JavaScript Get More Respect?</h1>
    <p>An anonymous reader points out an article...<p></p>
  </p>

  <!-- (more file contents) -->
</body>
</html>
```
HTML Primer

• Elements (tags)

<table>
<thead>
<tr>
<th>Element</th>
<th>Within</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;html&gt;</td>
<td></td>
<td>HTML content</td>
</tr>
<tr>
<td>&lt;head&gt;</td>
<td>&lt;html&gt;</td>
<td>Page metadata</td>
</tr>
<tr>
<td>&lt;title&gt;</td>
<td>&lt;head&gt;</td>
<td>Title of the page</td>
</tr>
<tr>
<td>&lt;body&gt;</td>
<td>&lt;html&gt;</td>
<td>Page data</td>
</tr>
<tr>
<td>&lt;h1&gt;</td>
<td>&lt;body&gt;</td>
<td>Heading level 1</td>
</tr>
<tr>
<td>&lt;p&gt;</td>
<td>&lt;body&gt;</td>
<td>Paragraph data</td>
</tr>
<tr>
<td>&lt;div&gt;</td>
<td>&lt;html&gt;</td>
<td>Arbitrary page section</td>
</tr>
<tr>
<td>&lt;span&gt;</td>
<td></td>
<td>Arbitrary inline section</td>
</tr>
<tr>
<td>&lt;ol&gt;</td>
<td>&lt;html&gt;</td>
<td>Ordered list</td>
</tr>
<tr>
<td>&lt;ul&gt;</td>
<td>&lt;html&gt;</td>
<td>Unordered list</td>
</tr>
<tr>
<td>&lt;li&gt;</td>
<td>&lt;ol&gt;</td>
<td>Numbered list item</td>
</tr>
<tr>
<td>&lt;li&gt;</td>
<td>&lt;ul&gt;</td>
<td>Bulleted list item</td>
</tr>
</tbody>
</table>
### HTML Primer

**Elements (tags)**

<table>
<thead>
<tr>
<th>Element</th>
<th>Within</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;table&gt;</td>
<td>&lt;html&gt;</td>
<td>Table</td>
</tr>
<tr>
<td>&lt;tr&gt;</td>
<td>&lt;table&gt;</td>
<td>Table row</td>
</tr>
<tr>
<td>&lt;td&gt;</td>
<td>&lt;tr&gt;</td>
<td>Table data (cell)</td>
</tr>
<tr>
<td>&lt;dl&gt;</td>
<td>&lt;html&gt;</td>
<td>Definition list</td>
</tr>
<tr>
<td>&lt;dt&gt;</td>
<td>&lt;dl&gt;</td>
<td>Definition term</td>
</tr>
<tr>
<td>&lt;dd&gt;</td>
<td>&lt;dl&gt;</td>
<td>Definition data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Element</th>
<th>Within</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;br /&gt;</td>
<td>&lt;html&gt;</td>
<td>Line break</td>
</tr>
<tr>
<td>&lt;hr /&gt;</td>
<td>&lt;html&gt;</td>
<td>Horizontal rule</td>
</tr>
<tr>
<td>&lt;img /&gt;</td>
<td>&lt;html&gt;</td>
<td>Image (JPEG, GIF, PNG)</td>
</tr>
<tr>
<td>&lt;form&gt;</td>
<td>&lt;html&gt;</td>
<td>Form for user input</td>
</tr>
<tr>
<td>&lt;input /&gt;</td>
<td>&lt;form&gt;</td>
<td>User input field (text, button)</td>
</tr>
<tr>
<td>&lt;script&gt;</td>
<td></td>
<td>Executable code</td>
</tr>
</tbody>
</table>
HTML Entities

• Entities (special characters)

<table>
<thead>
<tr>
<th>Code</th>
<th>Render</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;</td>
<td>&quot;</td>
</tr>
<tr>
<td>&amp;</td>
<td>&amp;</td>
</tr>
<tr>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
</tr>
<tr>
<td> </td>
<td>□</td>
</tr>
<tr>
<td>©</td>
<td>©</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code</th>
<th>Render</th>
</tr>
</thead>
<tbody>
<tr>
<td>®</td>
<td>®</td>
</tr>
<tr>
<td>™</td>
<td>™</td>
</tr>
<tr>
<td>£</td>
<td>£</td>
</tr>
<tr>
<td>€</td>
<td>€</td>
</tr>
<tr>
<td>“</td>
<td>“</td>
</tr>
<tr>
<td>”</td>
<td>”</td>
</tr>
</tbody>
</table>

A space, HTML typically ignores spaces and CRs in the HTML.

HTML Primer

• Attributes

• Parameters to elements

```html
<a href="http://franklin.edu/">Franklin</a>
```

• Refines or further defines the tag

```
href: hyperlink reference to another page.
```
HTML Primer

• Attributes

• Parameters to elements

    \<div id="sidebar">As an aside...</div\>

\text{id}:\ internal\ identifier\ used\ to\ programmatically\ access\ and\ alter\ the\ division.

HTML Primer

• Attributes

• Parameters to elements

    \<script type="text/javascript" language="JavaScript" src="employee.js"></script>\n
\text{type}:\ programming\ language\ the\ script\ uses.\n\text{language}:\ default\ scripting\ language\ and\ version.\n\text{src}:\ file\ from\ which\ to\ read.
Attributes

Parameters to elements

<input type="text" name="last" id="last" />

**Programming Tip**
Set id= and name= to the same value.
Ex: id="txt" name="txt"

type: kind of input control.
name: field name sent to server.

src: location of image file.
Try it out!

- Create a “hobby” page
  - Specifications
    - Title
    - Picture
    - Paragraph description
    - Bulleted list with links to other hobby pages

Introduction to JavaScript

- Dynamic web pages
  - HTML provides static content
  - Client-side scripting provides
    - Dynamic content
    - Form validation
    - Interactivity
    - Special effects
Introduction to JavaScript

- Client-side scripting restrictions
  - No access to the file system
  - No network communication outside the originating domain
- JavaScript is case dependent

The `<script>` tag

```html
1. <script type="text/javascript">
2.   //<![CDATA[
3.     document.writeln("Hello");
4.   //]]>
5. </script>

Alternative style as W3C validate dislikes the CDATA method

```html
1. <script type="text/javascript" language="JavaScript">
2.   <!-- Hide script from old browsers. -->
3.     // Put code below.
4.     document.writeln("Hello");
5.   // Stop hiding script from old browsers. -->
6. </script>
```
Introduction to JavaScript

- Hello world program

```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML...>
<html>
<head><title>Hello World</title></head>
<body>
<script type="text/javascript" language="JavaScript">
<!-- Hide script from old browsers.
    document.write("<h1>Hello World</h1>" nation="JavaScript/");
// Stop hiding script from old browsers. -->
</script>
</body>
</html>
```
**Introduction to JavaScript**

- **Comparing `writeln()` & `write()`**

  ```javascript
  document.writeln("<h1>Hello World</h1>");
  ```
  
  Appends a carriage return (in the HTML)

  ```javascript
  document.write("<h1>Hello World</h1>");
  ```
  
  Does not append a carriage return.

- **JavaScript objects**
  - **Calling object methods:**
    
    ```javascript
    noun.verb(parameters)
    ```

  ```javascript
  document.writeln("<h1>Hello World</h1>");
  ```
Introduction to JavaScript

- JavaScript objects
  - Calling object methods: `noun.verb(parameters)`

```javascript
document.writeln("<h1>Hello World</h1>");
```

Object (noun)

Method (verb)

```javascript
document.writeln("<h1>Hello World</h1>");
```
Introduction to JavaScript

- JavaScript objects
  - Calling object methods:
    `noun.verb(parameters)`

```javascript
document.writeln('<h1>Hello World</h1>');
```

Parameter (extra needed information)

- Changing object properties:
  `noun.property = new_value`

```javascript
document.bgColor = '#AABBCC';
```
Introduction to JavaScript

• JavaScript objects
  • Changing object properties:
    noun.property = new_value

  ```javascript
  document.bgColor = "#AABBCC";
  ```

Object (noun)

Property (noun)
Introduction to JavaScript

- JavaScript objects
- Changing object properties:
  
  ```javascript
  noun.property = new_value
  ```

  ```javascript
  document.bgColor = "#AABBCC";
  ```

- Prompting for input

  ```html
  <script type="text/javascript">
  1. <!-- Hide script from old browsers. -->
  2. var name = prompt("What is your name?");
  3. document.writeln("<h1>Hello " + name + "</h1>);
  4. // Stop hiding script from old browsers. -->
  5. </script>
  ```
Introduction to JavaScript

• Prompting for input

```html
<script type="text/javascript">
<!-- Hide script from old browsers.
var name = prompt("What is your name?");
document.writeln(<h1>Hello 
+ name + "</h1">);
// Stop hiding script from old browsers. -->
</script>
```

The variable, name, temporarily holds the user input
Introduction to JavaScript

• Prompting for input

```html
1. <script type="text/javascript">
2. <!-- Hide script from old browsers. -->
3. var name = prompt("What is your name?");
4. document.writeln("<h1>Hello " + name + "</h1>);
5. // Stop hiding script from old browsers. -->
6. </script>
```

Whatever content the user typed is substituted for the variable.
Introduction to JavaScript

• External JavaScript files

index.html
1. <script type="text/javascript" src="hello.js">
2. </script>

hello.js
1. var name = prompt("What is your name?");
2. document.writeln("<h1>Hello " + name + "</h1>");
Why use separate files?

- Keeps HTML (presentation) separated from JavaScript (programming logic)
- Allows many HTML pages to reuse a set of scripts
- Browsers can cache the files separately
- Browsers that don’t understand JavaScript ignore it rather than render

JavaScript Placement

- Where should `<script>` be put?
  - In the `<head>` section
    - To load external scripts
    - To define new functions
  - In the `<body>` section
    - To execute as or after the page loads
JavaScript Summary

- JavaScript is case dependent
- Use `<script>` for inline and external JavaScript code
- Hide script from older browsers by using HTML comments or the CDATA method around JavaScript code blocks
- Separate code into separate HTML and JS files

Try it out!

- Edit your "hobby" page to include
  - A script in the body that prompts the user for their name, and then renders a custom welcome message using their name on your page.
Self Quiz

- What is the difference between a JavaScript *method* and a JavaScript *property*?
- How are HTML tags that have no matching closing tag written? Give an example.
- What is one advantage of placing your JavaScript in a separate file versus writing it all in the HTML document?
- What does a DTD (Document Type Definition) do?
- What is the difference between an HTML *attribute* and an HTML *tag*?
Debugging Tools

• Old school approach
• Logging: debugging statements placed strategically in program code

```javascript
function log(div, message) {
    document.getElementById(div).innerHTML += message + "<br />
}

if (debug == true) {
    log("debug", "Got to here");
}
```
Debuggers

- **Debugger**: Programs that allow you to examine the state of another running program

- Built into the IDE
  - Stop your program at a particular point (*breakpoint*)
  - Inspect the contents of a variable (*inspect* or *watch*)
  - Step through a program as it executes (*step into, step over, step out*)

Typical Debugging Session

- Set a breakpoint in your code just prior to where:
  - You think a problem is occurring
  - The Debugger has shown an error

- Run the program in debug mode, Debugger will suspend program at breakpoint
Typical Debugging Session

- Examine variables at breakpoint to determine what may have gone wrong. Use the watch or inspect features.
- Step forward through the program line-by-line to examine how objects or variables change.

Firebug Tutorial & Video

- [http://getfirebug.com/](http://getfirebug.com/)
- Links in Key Points (14.2)
  - Firebug Tutorial #1
- Video Screencasts – Excellent!
  - [http://files.jnewland.com/firebug.mov](http://files.jnewland.com/firebug.mov)
Upcoming Deadlines

• Homework 1 due Jan 12
• Readings for next week
  • Chapter 2 in *JavaScript*
  • Chapter 2 and part of chapter 3 in *Learning JavaScript*
  • Module 2 Key Points - Software Life Cycle

General Q & A

• Questions?
• Comments?
• Concerns?