Announcements

- Due dates extended:
  - Project 1B—Wednesday by 10pm
  - 1-1-1 rule Thursday by 10pm
  - Lab 5—Friday by 10pm

- Vocabulary for the week has been posted in GoPost
- Reading
  - Ch 18 for today and Wednesday
  - Ch 21 for Friday

Programs Defined

- A program is an algorithm written for a specific programming language and specific circumstances

Types of Programming Languages

High- vs. Low-Level Languages

| HIGH-LEVEL | Translated | LOW-LEVEL |
| High-Level Language | Computers Understand |
| Fortran | C, C++, C# | JavaScript |
| Assembly Language |
| Machine Language |

Videos

- High- and low-level programming languages
**Human-Understandable Code**

- Today’s programs are written in “high-level” language (HLL) that we can understand (and debug)
- HLL use “real” words—`if`, `while`, `when`, `until`, `push`, `pop`, `print`, `set`, etc.
- Words look like English
- Have a precisely defined meaning for the computer
- Make it easier for us to understand (and troubleshoot)
- For example:
  ```java
  if (today=='Wednesday')
  print "I have lecture today!"
  else
  print "Tonight is time to study!"
  ```

**High-Level Languages (HLL)**

- Video: High-Level Programming Languages

**Assembly language**

- The lowest level language humans can understand
- Example
  ```assembly
  LOOP: MOV.B r0, #80 ;initialize counter
  ```

**What computers understand**

- Machine code
  - Assembly code is translated to binary:
    - 0011 0000 1000 0000
    - Binary is how computer stores information
    - all zeroes and ones
    - Magnetized or not
    - Off or on
    - Bumpy surface on the CD or not

**Translating human to machine**

```
0110111000101110110011101110110011000110011
001111011010100100011011101101100101110110
011011001110110011000111011001110011000110
1000011011100110110101110100110011101111
10110101101100011010111010011000101101100
110001110110110000110110101011011011010110
101101010110011001101111101011100011111101
110011011011010110110011001100011011101100
110011011011000110110001101111101011101101
101101010110011001101111101011100011111101
100110110111011011101110111110110011011111
001100111010110011101011101010111011001111
```
Compiled Languages

- Compiled languages are translated to machine code (assembly language) before they are run.
- Whenever you make changes to your program, you have to recompile the program again.
- Because they already speak the computer’s language, they run faster.
- Sometimes, they run by themselves—.exe files (NotePad2.exe)—or run with an engine (the Java virtual engine).

Examples:
- Java
- C family
- Visual Basic
- COBOL
- ForTRAN
- many others

Interpreted Languages

Also called scripting languages

INTERPRETED LANGUAGES

An interpreter translates from JavaScript to machine language while the Web browser renders the page
- The interpreter is part of the Web browser.
  - The JavaScript interpreter is available in all major Web browsers

How the Interpreter Works

- The interpreter translates the script to machine language while the program runs!
  - Two tasks at once—translating and running the program!
  - Scripts run slower than compiled programs

The Advantages…

- Scripted languages are interpreted on the fly—while they are running
  - Make changes while the program is running!
  - Provides a dynamic, responsive, and interactive experience for the user

Font Size

normal | large | largest
JavaScript

- Java was developed by Sun Microsystems and is seen on the Web mostly in Java Applets.

JavaScript is not Java!

Brief History of JavaScript

- Released with Netscape Navigator in 1995.
- Ratified by the European Equipment Manufacturer's Association (ECMA)
- Result:
  - ECMAScript is the core spec for the JavaScript language
  - Netscape, MS, and the others try to conform to the spec

Divergence from standards….

<table>
<thead>
<tr>
<th>Developer</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Netscape (now Mozilla)</td>
<td>JavaScript</td>
</tr>
<tr>
<td>Microsoft</td>
<td>JScript</td>
</tr>
</tbody>
</table>

- Programmers call both JavaScript.
- Both comply differently with the standards

Objectives

- Understand how JavaScript and HTML interact
- Understand where to place JavaScripts on the HTML page
Programming Concepts

- Programming: Act of formulating an algorithm or program
- Basic concepts have been developed over last 50 years to simplify common programming tasks
- Concepts will be expressed in JavaScript

The Web page

- Without JavaScript the Web page is like a brick; it just sits there!

Scripts

- Client-side scripts
  - Your Web browser on your computer is the client
- Server-side scripts
  - Web server
  - Database server
  - File server

Web browser and JavaScript

- The major Web browsers have a built-in interpreter that parses JavaScript
- Parses: breaks into smaller pieces that can be translated into machine code

Placing JavaScripts on a Web page

Types of scripts:
- Body scripts
- Header scripts
- External scripts

Body Script

```html
<html>
<head>
<title>Name of Page</title>
</head>
<body>
<script type="text/javascript">
//JavaScript goes here
</script>
</body>
</html>
```
**Header Script**

```html
<html>
<head>
<title>Name of Page</title>
<script type="text/javascript" />
</head>
<body>
Body content goes here
</body>
</html>
```

**Linking to External JavaScripts**

- Linked in the `<head>`
- `src` gives path to file and its name

```html
<html>
<head>
<title>Name of Page</title>
<script type="text/javascript" src="basic.js"></script>
</head>
<body>
Body content goes here
</body>
</html>
```

**External JavaScripts**

- Make changes to scripts in one place
- Reusable
  - Link to any page, every page, or many sites

**Best Practice**

- Best practice to separate Content from Action from Appearance
- Put styles in external CSS
- Put scripts in external JavaScript files
- Leave only the HTML markup and content on the page
- Accomplishing that goal takes more experience....

**Summary**

- Understand how JavaScript and HTML interact
- Understand where to place JavaScripts on the HTML page