

LBSC 690: Session 10

Programming, JavaScript



Jimmy Lin
College of Information Studies
University of Maryland

Monday, November 12, 2007

Software

- Software "does something"
 - Tells the machine how to operate on some data
- Software models aspects of reality
 - Input and output represent the state of the world
 - Software describes how the two are related
- Examples:
 - Ballistic computations
 - Visa's credit card verification system
 - Google
 - Microsoft Word

Types of Software

- Application programs (e.g., PowerPoint)
 - What you normally think of as a "software"
- Compilers and interpreters
 - Programs used to write other programs
- Operating system (e.g., Windows XP)
 - Manages computing resources
- Embedded software (e.g., TiVO)
 - Programs permanently embedded inside some physical device

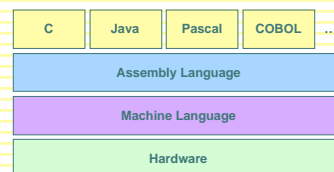
Programming Languages

- Software "does something"
 - Programming languages tell the machine "what to do"
- Special purpose: geared towards specific tasks
 - Complex math (e.g., Matlab)
 - Databases (e.g., SQL)
- General purpose: able to accomplish anything
 - Examples: Java, JavaScript, Perl, C, C++, ...

Types of Programming

- Machine language
 - Language that the machine can directly understand
- Assembly language
 - Directly specifies actions of the machine
 - Assembler changes instructions to machine code
- High-level languages
 - Specifies machine instructions at a more abstract level
 - Compiler/interpreter translates to machine language
 - Examples: C/C++, Java, JavaScript
- Visual programming languages
 - Visually arrange interface components
 - Example: Visual Basic

Types of Languages



Machine Language

- Everything is a binary number
 - Operations
 - Data
 - Memory locations
- For instance

```
00001000 00010101 01010110 0010
```

- 00001000 ADD
- 00010101 first number (21)
- 01010110 second number (86)
- 0010 register 2

Assembly Language

- One level up from machine language
 - Symbolic instructions
 - Symbols representing memory locations
- For instance

```
ADD 21, 86, R2
```
- Assembly code is directly translatable into machine language

High-Level Languages

- Instructions represent higher-level constructs
 - Closer to how humans approach problems
 - Must be converted into machine code by a compiler or interpreter

Programming: Overview

- A program consists of a sequence of instructions
- Basic concepts:
 - Data types
 - Variables
 - Basic operations
 - Instructions
- Structures for controlling how instructions are executed:
 - Sequential
 - Conditional
 - Repetition

Programming: Foundations

- Data types = things that you can operate on
 - Boolean: true, false
 - Number: 5, 9, 3.1415926
 - String: "Hello World"
- Variables hold values of a particular data type
 - Represented as symbols (e.g., x)
- Operations = things that you can do
 - $-x$ reverse the sign of x (negation)
 - $6+5$ Add 6 and 5 (numeric)
 - "Hello" + "World" Concatenate two strings
 - $2.1 * 3$ Multiply two values
 - $x++$ increase value of x by 1

Basic Instructions

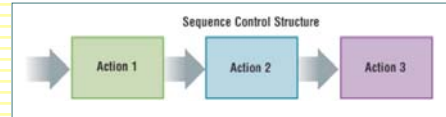
- Assignment = store the result of an operation
 - $x = 5$ set the value of x to be 5
 - $x += y$ $x = x + y$
 - $x *= 5$ $x = x * 5$
- In JavaScript, *var* declares a variable
 - `var b = true;` create a boolean b and set it to true
 - `var n = 1;` create a number n and set it to 1
 - `var s = "hello";` create a string s and set it to "hello"
- In JavaScript, all instructions end with a semicolon (;)

Basic Control Structures

- Sequential
 - Perform instructions one after another
- Conditional
 - Perform instructions contingent on something else being true
- Repetition
 - Repeat instructions until a condition is met

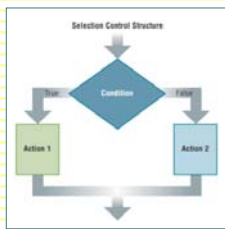
Not much different from cooking recipes!

Sequential Control Structure



```
var a = 2;  
var b = 3;  
var c = a * b;
```

Conditional Control Structure

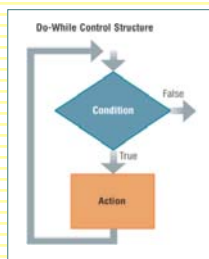


```
if (gender == "male") {  
    greeting = "Hello, Sir";  
} else {  
    greeting = "Hello, Madam";  
}
```

Test Conditions

- `x == y` true if `x` and `y` are equal
- `x != y` true if `x` and `y` are not equal
- `x > y` true if `x` is greater than `y`
- `x <= y` true if `x` is smaller than or equal to `y`
- `x && y` true if both `x` and `y` are true
- `x || y` true if either `x` or `y` is true
- `!x` true if `x` is false

Repetition Control Structure



Program Example 1:

```
n = 1;  
while ( n <= 10) {  
    document.writeln(n);  
    n++;  
}
```

Program Example 2:

```
For (n = 1; n <= 10; n++) {  
    document.writeln(n);  
}
```

Arrays

- A set of elements grouped together
 - For example, the number of days in each month
- Each element is assigned an index
 - A number used to refer to that element
 - For example, `x[4]` is the fifth element (count from zero!)
 - Arrays and repetitions work naturally together

Functions

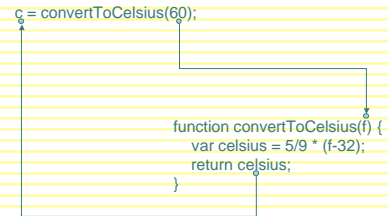
- Reusable code for doing a single task
- A function takes in one or more parameters and returns one value

```
function convertToCelsius(f) {  
  var celsius = 5/9 * (f-32);  
  return celsius;  
}
```

```
function weirdAddition(a, b) {  
  var result = a + b - 0.5;  
  return result;  
}
```

Calling Functions

- When you “call” a function, you invoke the set of instructions it represents



More Examples

```
var f = 60;  
c = convertToCelsius(f);
```

```
r = weirdAddition(2, 4);
```

```
var a = 2;  
var b = 3;  
r = weirdAddition(a, b);
```

Programming Paradigms

- Procedural Programming
 - Group instructions into meaningful functions
 - Examples: C, Pascal, Perl
- Object oriented programming
 - Group “data” and “methods” into “objects”
 - Naturally represents the world around us
 - Examples: C++, Java

Algorithms

- Derived from the name of the Persian mathematician Al-Khwarizmi
- A sequence of well-defined instructions designed to accomplish a certain task

Programming for the Web

- Common Gateway Interface (CGI) [Server-side]
 - User inputs information into a form
 - Form values passed to the sever via CGI
 - Program on the server generates a Web page as a response
- JavaScript [Client-side, interpreted]
 - Human-readable “source code” sent to the browser
 - Web browser runs the program
- Java applets [Client-side, compiled]
 - Machine-readable “bytecode” sent to browser
 - Web browser runs the program
 - Not related to JavaScript (other than similarities in syntax)

Where is the JavaScript?

- JavaScript is usually kept in the <head> section of an HTML document

```
...
<head>
<script language="JavaScript" type="text/javascript">
<!--
function calculate() {
  var num = eval(document.input.number.value);
  ...
  document.output.number.value = total;
}
//-->
</script>
</head>
...
```

Handling Events

- When does code actually get executed?
- Events:
 - User actions trigger "events"
 - Embedded in all modern GUIs
- Event handlers are used to respond to events
 - Examples of event handlers in JavaScript
 - onmouseover: the mouse moved over an object
 - onmouseout: the mouse moved off an object
 - onclick: the user clicked on an object

Input and Output

- How do you get information to/from the user?
 - Forms provide a method for accepting input and displaying output

In HTML

```
<form name="input" action="">
Please enter a number:
<input size="10" value="" name="number"/>
</form>
<form name="output" action="">
The sum of all numbers up to the number above is
<input size="10" value="" name="number" readonly="true"/>
</form>
```

JavaScript code

```
var num = eval(document.input.number.value);
document.output.number.value = 10;
```

Reads in a value
eval function turns it into a number

Changes the value in the textbox

JavaScript Resources

- Google "javascript"
 - Tutorials: to learn to write programs
 - Code: to do things you want to do ("borrow")
- Books

Programming Tips

- Details are everything!
 - Careful where you place that comma, semi-colon, etc.
- Write a little bit of code at a time
 - Add a small new functionality, make sure it works, then move on
 - Don't try to write a large program all at once
- Debug by outputting the state of the program
 - Print out the value of variables using document.write
 - Is the value what you expected?