College of Information Studies

University of Maryland Hornbake Library Building College Park, MD 20742-4345

Programming

Week 11

LBSC 690

Information Technology

Outline

- Programming
- Javascript
- PHP

Software

- Software models aspects of reality
  - Input and output represent the state of the world
  - Software describes how the two are related

- Examples
  - Ballistic computations
  - Google
  - Microsoft Word

Types of Software

- Application programs (e.g., Powerpoint)
  - What you normally think of as a “program”
- Compilers and interpreters
  - Programs used to write other programs
- Operating system (e.g., Windows Vista)
  - Manages display, CPU, memory, disk, tape,
- Embedded program (e.g., TiVO)
  - Permanent software inside some device

Programming Languages

- Used to specify every detail of the model

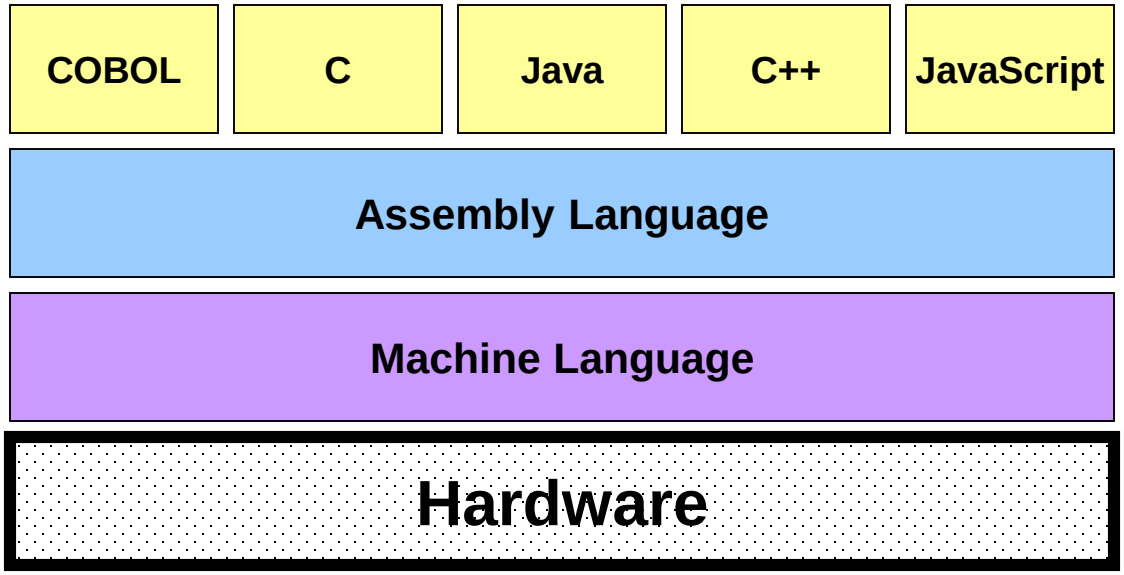
- Special purpose
  - Able to specify an entire class of models
    - Spreadsheets (Excel, ...)
    - Databases (Access, Oracle, ...)

- General purpose
  - Able to specify any possible model
    - JavaScript, Java, Perl, C, C++, ...

History of Programming

- Machine language
  - Language that machine can understand
- Assembly language
  - Assembler changes names to machine code
- High-level languages
  - Compiler/Interpreter translates to machine language
  - FORTRAN, COBOL, C, C++, Javascript
- Visual programming language
  - Visually arrange the interface components
  - Visual Basic, …

Programming Languages



Machine Language

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

```
00001000   00010101   01010110
```

00001000	ADD
00010101	first number (21)
01010110	second number (86)

Assembly Language

- Symbolic instruction codes and addresses
  - Symbolic instruction code “ADD”
  - Symbolic address “SUM1”
- For instance

```
ADD          21, SUM1
```

High level Languages

- Procedural (modular) Programming
  - Group instructions into meaningful abstractions
  - C, Pascal, Perl
- Object oriented programming
  - Group “data” and “methods” into “objects”
  - Naturally represents the world around us
  - C++, Java, JavaScript

Programming for the Web

- PHP [Server side]
  - Forms encode field values into a URL
  - Web server passes field values to a PHP program
  - Program 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

Variables

- Data types
  - Boolean: true, false
  - Number: 5, 9, 3.1415926
  - String: “Hello World”
- A “variable” holds a value of a specific data type
  - Represented as symbols: x, celsius
- 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”

Operators

• -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
• x = 5	set the value of x to be 5
• x += y	x = x + y
• x *= 5	x = x * 5

Statements

- In JavaScript, instructions end with a semicolon
  - If missing at end of line, it is automatically inserted
- Simple assignment statements

  ```
  celsius = 5/9 * (f-32);
  ```

- Statements that invoke a method

  ```
  Temperature.toCelsius(104);
  ```

- Return a value from a method

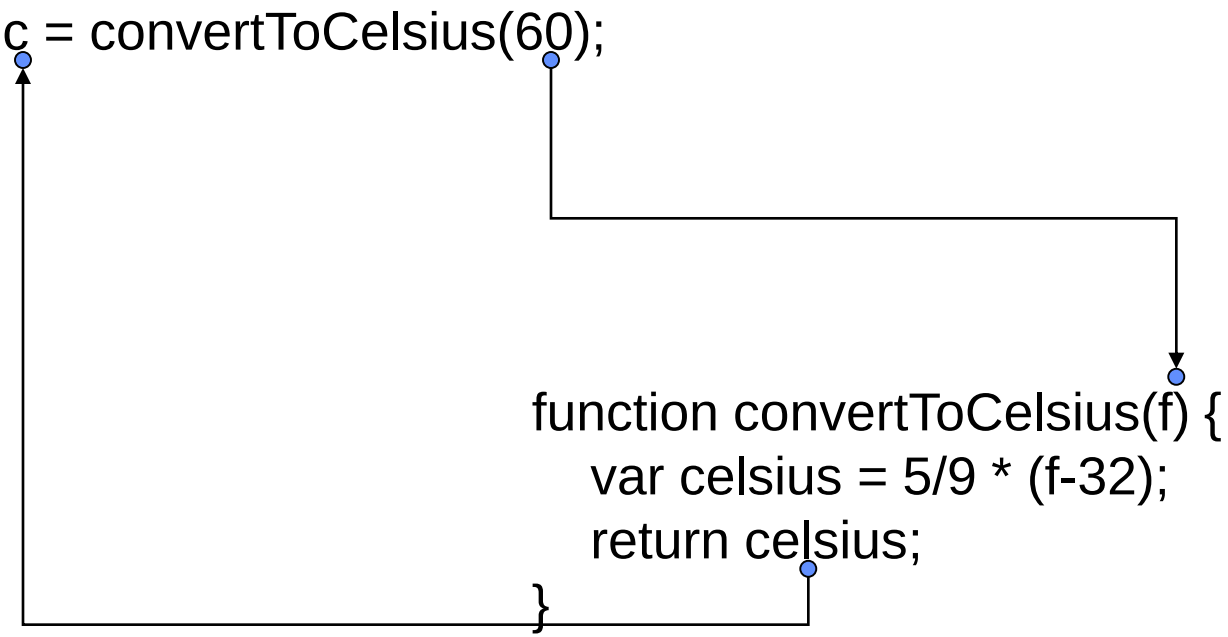
  ```
  return celsius;
  ```

Functions

- Reusable code for complex “statements”
  - Takes one or more values as “parameters”
  - Returns at most one value as the “result”

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

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



Algorithms

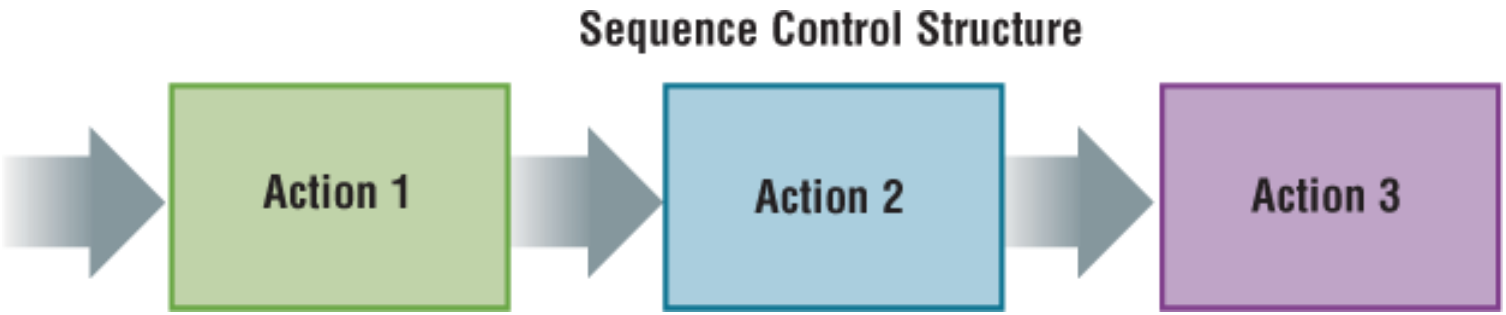
- A sequence of well-defined instructions designed to accomplish a certain task
- Derived from the name of the Persian mathematician Al-Khwarizmi

Basic Control Structures

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

Not much different from cooking recipes!

Sequential Control Structure

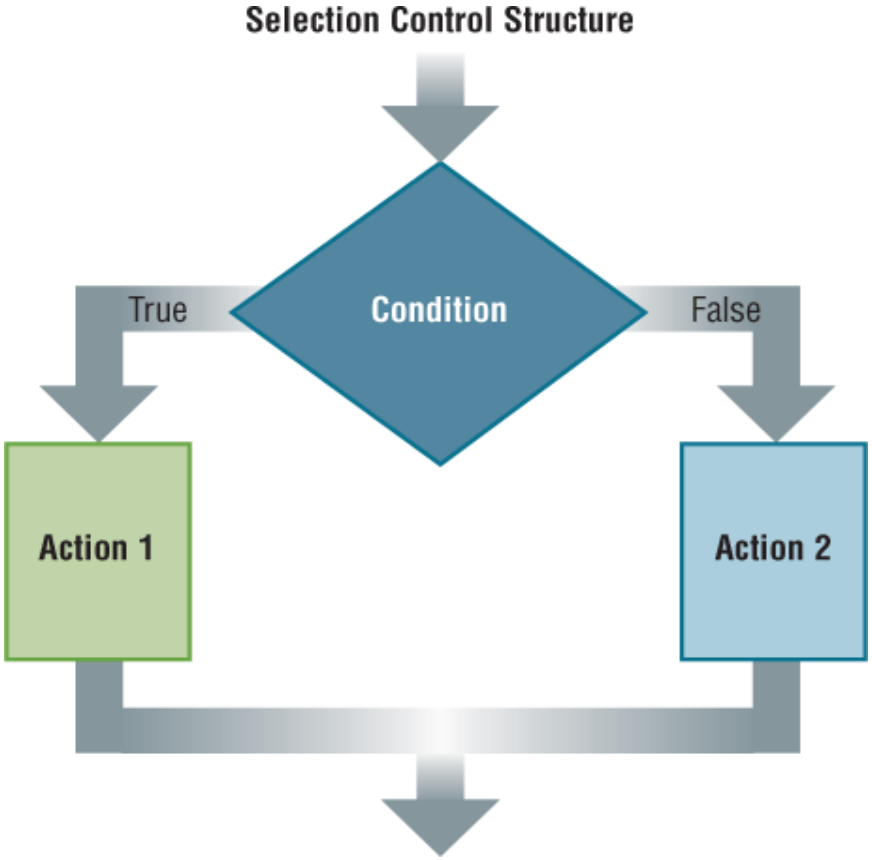


a = 2

b = 3

c = a * b

Conditional Selection Control Structure

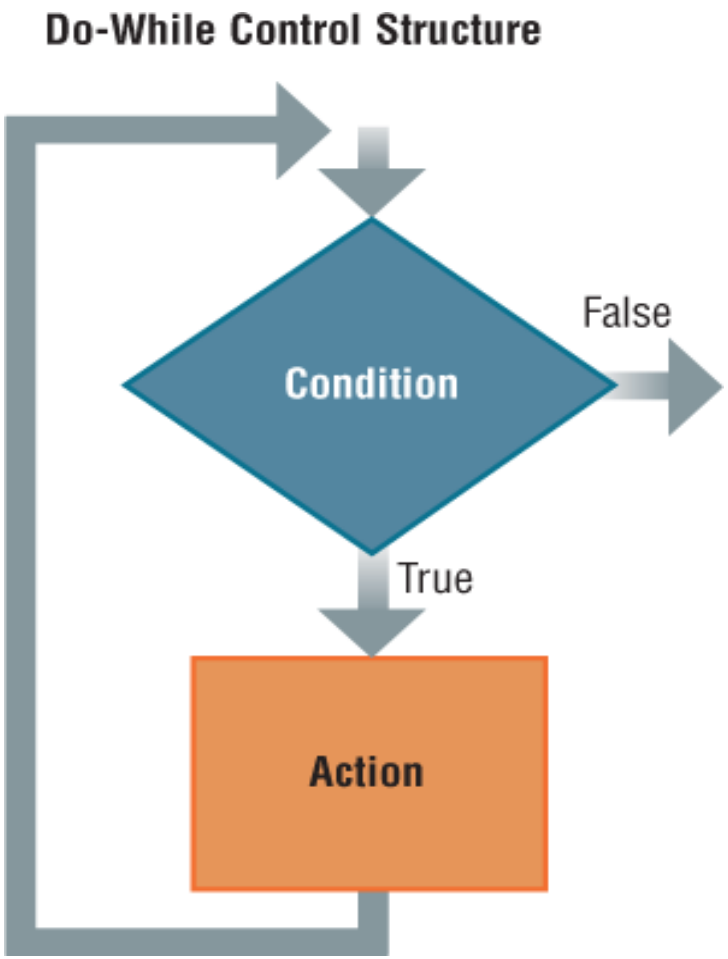



```
if (gender == “male”) {
    greeting = “Hello, Sir”
}
else {
    greeting = “Hello, Madam”
}
```

Generating Boolean Results

- 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 2:

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

Arrays

- A set of elements
  - 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

JavaScript

```
<HTML>
<HEAD>
    <TITLE>My first script</TITLE>
</HEAD>
<BODY BGCOLOR=WHITE>
<H1>
    <SCRIPT LANGUAGE=JAVASCRIPT TYPE="TEXT/JAVASCRIPT">
        document.write("Hello, world!")
    </SCRIPT>
</H1>
</BODY></HTML>
```

Try it at http://www.umiacs.umd.edu/~oard/teaching/690/fall08/notes/9/firstscript.html

Placement

- JavaScript is usually in the <head> section

```
…
<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

- Events:
  - Actions that users perform while visiting a page
- Use event handlers to response events
  - Event handlers triggered by 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

HTML “Forms”

- Accept input and display output for JavaScript

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>
```

Reads in a value

eval function turns it into a number

JavaScript code

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

Changes the value in the textbox

Hands On:
Adopt a JavaScript Program

- Launch a Web browser
  - http://www.umiacs.umd.edu/~oard/teaching/690/spring13/notes/11/selector.htm
- See how it behaves if you are 13 (or 65)
- View source and read the program
- Save a local copy
- Make some changes and see how it works

Programming Tips

- Attention to detail!
  - Careful where you place that comma, semi-colon, etc.
- Write a little bit of code at a time
  - Add some functionality, make sure it works, move on
  - Don’t try to write a large program all at once
- Debug by viewing the “state” of your program
  - Print values of variables using document.write
  - Is the value what you expected?

JavaScript Resources

- Google “javascript”
  - Tutorials: to learn to write programs
  - Code: to do things you want to do

- Engineering and Physical Sciences Library

Some Details About Access

- Joins are automatic if field names are same
  - Otherwise, drag a line between the fields
- Sort order is easy to specify
  - Use the menu
- Queries form the basis for reports
  - Reports give good control over layout
  - Use the report wizard - the formats are complex
- Forms manage input better than raw tables
  - Invalid data can be identified when input
  - Graphics can be incorporated

Programming Tips

- Attention to detail!
  - Careful where you place that comma, semi-colon, etc.
- Write a little bit of code at a time
  - Add some functionality, make sure it works, move on
  - Don’t try to write a large program all at once
- Debug by viewing the “state” of your program
  - Print values of variables using document.write
  - Is the value what you expected?

JavaScript Resources

- Google “javascript”
  - Tutorials: to learn to write programs
  - Code: to do things you want to do

- Engineering and Physical Sciences Library

Ways of Generating Web Pages

- Static: Written in a markup language
  - HTML, XML
- Dynamic: Generated using a program
  - Common Gateway Interface [Perl] (.cgi)
  - Java servlets
- Dynamic: Generated from a database
  - Cold Fusion (.cfm)
  - PHP (.php)

Why Database-Generated Pages?

- Remote access to a database
  - Client does not need the database software
- Serve rapidly changing information
  - e.g., Airline reservation systems
- Provide multiple “access points”
  - By subject, by date, by author, …
- Record user responses in the database

Issues to Consider

- Benefits
  - Multiple views
  - Data reuse
  - Scalable
  - Access control
- Costs
  - Formal modeling
  - Complex (learn, design, implement, debug)
  - Brittle (relies on multiple communicating servers)
  - Not crawlable

Downside

- Brittle
  - Depends on multiple servers
- Complex
  - Learning, design, implementation, debugging
- Formally modeled

- Relational normalization
- Structured programming
- Software patterns
- Object-oriented design
- Functional decomposition

Interface Design

Interaction Design

Business rules

Client Hardware	(PC)
Web Browser	(IE, Firefox)
Client-side Programming	(JavaScript)
Interchange Language	(HTML, XML)
Server-side Programming	(PHP)
Database	(MySQL)
Server Hardware	(PC, Unix)

PHP Programming Environments

- You need three systems on the same server:
  - **P**HP (programming language)
  - **M**ySQL (DBMS)
  - **A**pache (Web server)

- XAMPP Server
  - Includes GUI tools

- OTAL (Sun Unix) supports Web deployment
  - Requires a text editor (e.g., emacs) or FTP

Making PHP

----- HTML stuff -----

<?php

    ----- PHP stuff -----

?>

----- HTML stuff -----

---

http://---URL stuff---/xxxxx.php

- Download and install XAMPP
  - http://www.apachefriends.org/en/xampp.html
  - Check to install Apache and MySQL as services
  - Check the Web server at http://localhost/
- Download and install Joomla!
  - http://www.joomla.org
  - Unzip to c:\xampp\htdocs\joomla15
  - Configure it at http://localhost/joomla15/
    - Host Name: localhost
    - User Name: root
    - Database Name: joomla15
    - DO NOT install sample data
  - Delete c:\xampp\htdocs\joomla15\installation

Connecting PHP to MySQL

- On XAMPP:

$dbc=mysql_connect (‘localhost’, ‘userid’, ‘password’);

- On OTAL:

$dbc=mysql_connect(‘:/export/software/otal/mysql/run/mysqld.sock’,
‘userid’, ‘password’);

Create a MySQL Database

- "root" user creates database + grants permissions
  - Using the XAMPP console (or mysql –u root –p)
    - root has no initial password; just hit <enter> when asked
  - By the system administrator on OTAL (otal.umd.edu)

```
CREATE DATABASE project;
GRANT SELECT, INSERT, UPDATE, DELETE, INDEX, ALTER, CREATE, DROP ON project.* TO 'foo'@'localhost' IDENTIFIED BY 'bar';
FLUSH PRIVILEGES;
```

- Start mysql
  - MySQL console for XAMPP, ssh for OTAL

```
mysql –u foo –p bar
```

- Connect to your database

```
USE project;
```

Creating Tables

```
CREATE TABLE contacts (
  ckey      MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,
  id        MEDIUMINT UNSIGNED NOT NULL,
  ctype     SMALLINT UNSIGNED NOT NULL,
  cstring   VARCHAR(40) NOT NULL,
  FOREIGN KEY (id) REFERENCES persons(id) ON DELETE CASCADE,
  FOREIGN KEY (ctype) REFERENCES ctlabels(ctype) ON DELETE RESTRICT,
  PRIMARY KEY (ckey)
) ENGINE=INNODB;
```

- To delete: DROP TABLE contacts;

Populating Tables

```
INSERT INTO ctlabels
  (string) VALUES
  ('primary email'),
  ('alternate email'),
  ('home phone'),
  ('cell phone'),
  ('work phone'),
  ('AOL IM'),
  ('Yahoo Chat'),
  ('MSN Messenger'),
  (‘other’);
```

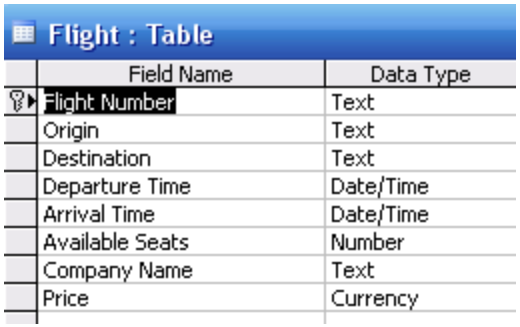
- To empty a table: DELETE FROM ctlabels;

“Looking Around” in MySQL

- SHOW DATABASES;
- SHOW TABLES;
- DESCRIBE tablename;
- SELECT * FROM tablename;

Structured Query Language

DESCRIBE Flight;



Flight : Table

Field Name	Data Type
Flight Number	Text
Origin	Text
Destination	Text
Departure Time	Date/Time
Arrival Time	Date/Time
Available Seats	Number
Company Name	Text
Price	Currency

Structured Query Language

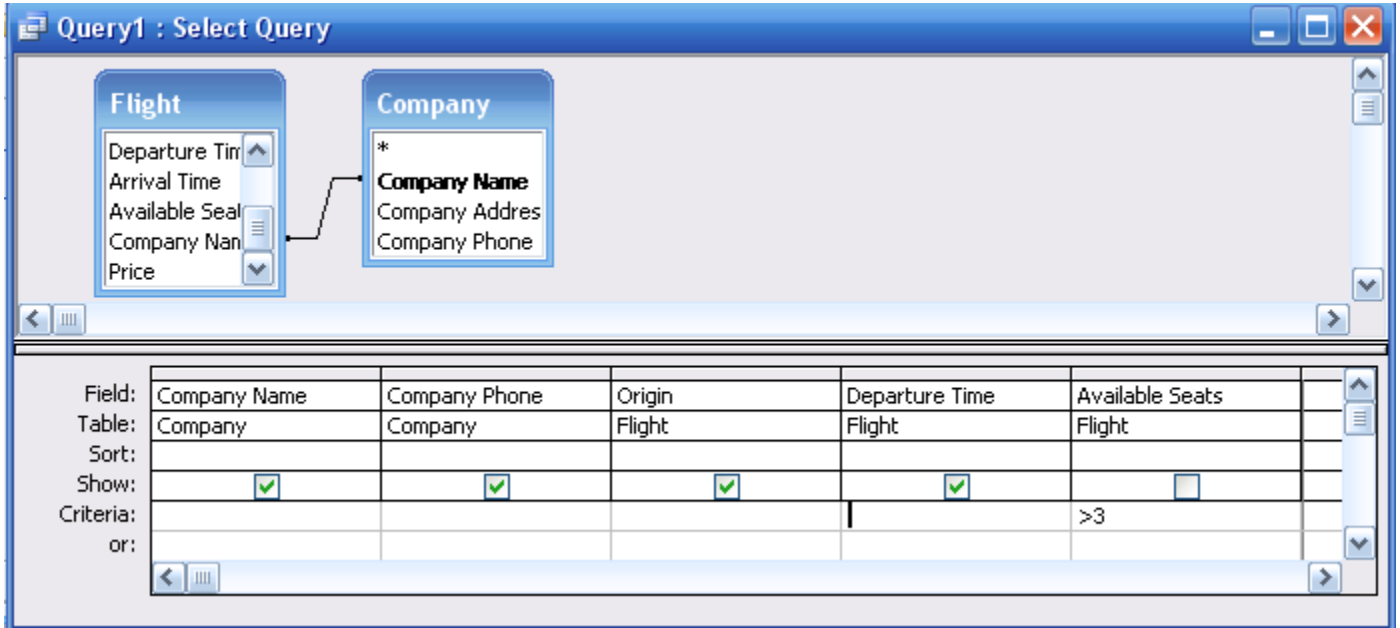
SELECT * FROM Flight;

Flight : Table

Flight Number	Origin	Destination	Departure Time	Arrival Time	Available Seats	Company Name	Price
CA210	DC	Austin	6:00:00 AM	11:00:00 AM	0	Cal Air	$200.00
CA345	San Jose	San Diego	9:00:00 AM	10:30:00 AM	20	Cal Air	$100.00
FT900	Chicago	New York	2:00:00 PM	5:00:00 PM	1	Fancy Trans	$200.00
GJ405	DC	San Jose	12:30:00 PM	8:45:00 PM	10	Green Jet	$340.00
GJ908	New York	Austin	8:00:00 AM	12:00:00 PM	2	Green Jet	$250.00
TP123	New York	San Jose	7:00:00 AM	11:00:00 AM	2	Trans Planet	$400.00
					0		$0.00

Structured Query Language

```
SELECT Company.CompanyName, Company.CompanyPhone,
    Flight.Origin, Flight.DepartureTime
FROM Flight,Company
WHERE Flight.CompanyName=Company.CompanyName
  AND Flight.AvailableSeats>3;
```



Statements in PHP

- Sequential

  {…; …;…;}

  Semicolons are required at the end of every statement
- Conditional

  if (3==i) {…} else {…}
- Loop

  for ($i=0; $i<10; $i++) {…}

  while ($row=mysql_fetch_array(…)) {…}

  foreach ($array as $key => $value) {…}
- Braces are optional around a single statement

Variables

- Name starts with a $
  - Case sensitive (assume everything could be!)
- Hold a value
  - Number (integer, float)
  - String (double quotes, \ escape character)
  - TRUE, FLASE
  - NULL
- Need not be declared (automatically “cast”)

Operators in PHP

- Arithmetic operators

  + - * /

- Logical operators

  < <= == != >= > && || !

- String operator

  .

Arrays in PHP

- A set of key-element pairs

  $days = array(“Jan”->31, “Feb”=>28, …);

  $months = explode(“/”, “Jan/Feb/Mar/…/Dec”);

  $_POST

- Each element is accessed by the key
  - {$days[“Jan”]}
  - $months[0];

Functions in PHP

- Declaration

  ```
  function multiply($a, $b=3){return $a*$b;}
  ```

- Invoking a method

  ```
  $b = multiply($b, 7);
  ```

- All variables in a function have only local scope
  - Unless declared as global in the function

Using PHP with (X)HTML Forms

```
<form action="formResponseDemo.php", method="post">
    email: <input type="text", name="email", value="<?php echo $email ?>", size=30 />
    <input type="radio", name="sure", value="yes" /> Yes
    <input type="radio", name="sure", value="no" /> No
    <input type="submit", name="submit", value="Submit" />
    <input type="hidden", name="submitted", value="TRUE" />
</form>
```

```
if (isset($_POST["submitted"])) {
    echo "Your email address is $email.";
} else {
    echo "Error: page reached without proper form submission!";
}
```

```
<?php # Script 8.1 - mysql_connect.php
// Set the database access information as constants.
DEFINE ('DB_USER', 'tester');
DEFINE ('DB_PASSWORD', 'tester');
DEFINE ('DB_HOST', 'localhost');
DEFINE ('DB_NAME', 'sitename');

// Make the connection.
$dbc = @mysql_connect (DB_HOST, DB_USER, DB_PASSWORD) OR die ('Could not connect to
MySQL: ' . mysql_error() );

// Select the database.
@mysql_select_db (DB_NAME) OR die ('Could not select the database: ' . mysql_error() );

// Create a function for escaping the data.
function escape_data ($data) {
        // Address Magic Quotes.
        if (ini_get('magic_quotes_gpc')) {
                $data = stripslashes($data);
        }
        // Check for mysql_real_escape_string() support.
        if (function_exists('mysql_real_escape_string')) {
                global $dbc; // Need the connection.
                $data = mysql_real_escape_string (trim($data), $dbc);
        } else {
                $data = mysql_escape_string (trim($data));
        }
        // Return the escaped value.
        return $data;
} // End of function.
?>
```

```php
<?php # login.php
// Send NOTHING to the Web browser prior to the session_start() line!
// Check if the form has been submitted.

if (isset($_POST['submitted'])) {
   require_once ('../mysql_connect.php'); // Connect to the db.
   $errors = array(); // Initialize error array.

   // Check for an email address.
   if (empty($_POST['email'])) {
      $errors[] = 'You forgot to enter your email address.';
   } else {
      $e = escape_data($_POST['email']);
   }

   // Check for a password.
   if (empty($_POST['password'])) {
      $errors[] = 'You forgot to enter your password.';
   } else {
      $p = escape_data($_POST['password']);
   }
```

```
if (empty($errors)) { // If everything's OK.
    /* Retrieve the user_id and first_name for that email/password combination. */
    $query = "SELECT user_id, first_name FROM users WHERE email='$e' AND password=SHA('$p')";
    $result = @mysql_query ($query); // Run the query.
    $row = mysql_fetch_array ($result, MYSQL_NUM); // Return a record, if applicable.
    if ($row) { // A record was pulled from the database.
      // Set the session data & redirect.
      session_name ('YourVisitID');
      session_start();
      $_SESSION['user_id'] = $row[0];
      $_SESSION['first_name'] = $row[1];
      $_SESSION['agent'] = md5($_SERVER['HTTP_USER_AGENT']);
      // Redirect the user to the loggedin.php page.
      // Start defining the URL.
      $url = 'http://' . $_SERVER['HTTP_HOST'] . dirname($_SERVER['PHP_SELF']);
      // Check for a trailing slash.
      if ((substr($url, -1) == '/') OR (substr($url, -1) == '\\') ) {
        $url = substr ($url, 0, -1); // Chop off the slash.
      }
      // Add the page.
      $url .= '/loggedin.php';
      header("Location: $url");
      exit(); // Quit the script.
    } else { // No record matched the query.
      $errors[] = 'The email address and password entered do not match those on file.'; // Public message.
      $errors[] = mysql_error() . '<br /><br />Query: ' . $query; // Debugging message.
    }
  } // End of if (empty($errors)) IF.
  mysql_close(); // Close the database connection.
} else { // Form has not been submitted.
  $errors = NULL;
} // End of the main Submit conditional.
```

```
// Begin the page now.
$page_title = 'Login';
include ('./includes/header.html');

if (!empty($errors)) { // Print any error messages.
   echo '<h1 id="mainhead">Error!</h1>
   <p class="error">The following error(s) occurred:<br />';
   foreach ($errors as $msg) { // Print each error.
      echo " - $msg<br />\n";
   }
   echo '</p><p>Please try again.</p>';
}

// Create the form.
?>

<h2>Login</h2>
<form action="login.php" method="post">
   <p>Email Address: <input type="text" name="email" size="20" maxlength="40" /> </p>
   <p>Password: <input type="password" name="password" size="20" maxlength="20" /></p>
   <p><input type="submit" name="submit" value="Login" /></p>
   <input type="hidden" name="submitted" value="TRUE" />
</form>

<?php
include ('./includes/footer.html');
?>
```

Discussion Point: Mythical Person-Month

- Why is software development different from manufacturing car?

- If it would take one person three months, why does it take four people SIX months?

Estimating Completion Time

- Rules of thumb
  - 1/3 specification
  - 1/6 coding
  - 1/2 test planning, testing, and fixing!

- Add time for coding to learn as you go, but don’t take time away from the other parts!
  - Reread the section on “gutless estimating” if you are tempted

Trading People and Months is Hard

- Sequential constraints
- Communication
- Training

- Relational normalization
- Structured programming
- Software patterns
- Object-oriented design
- Functional decomposition

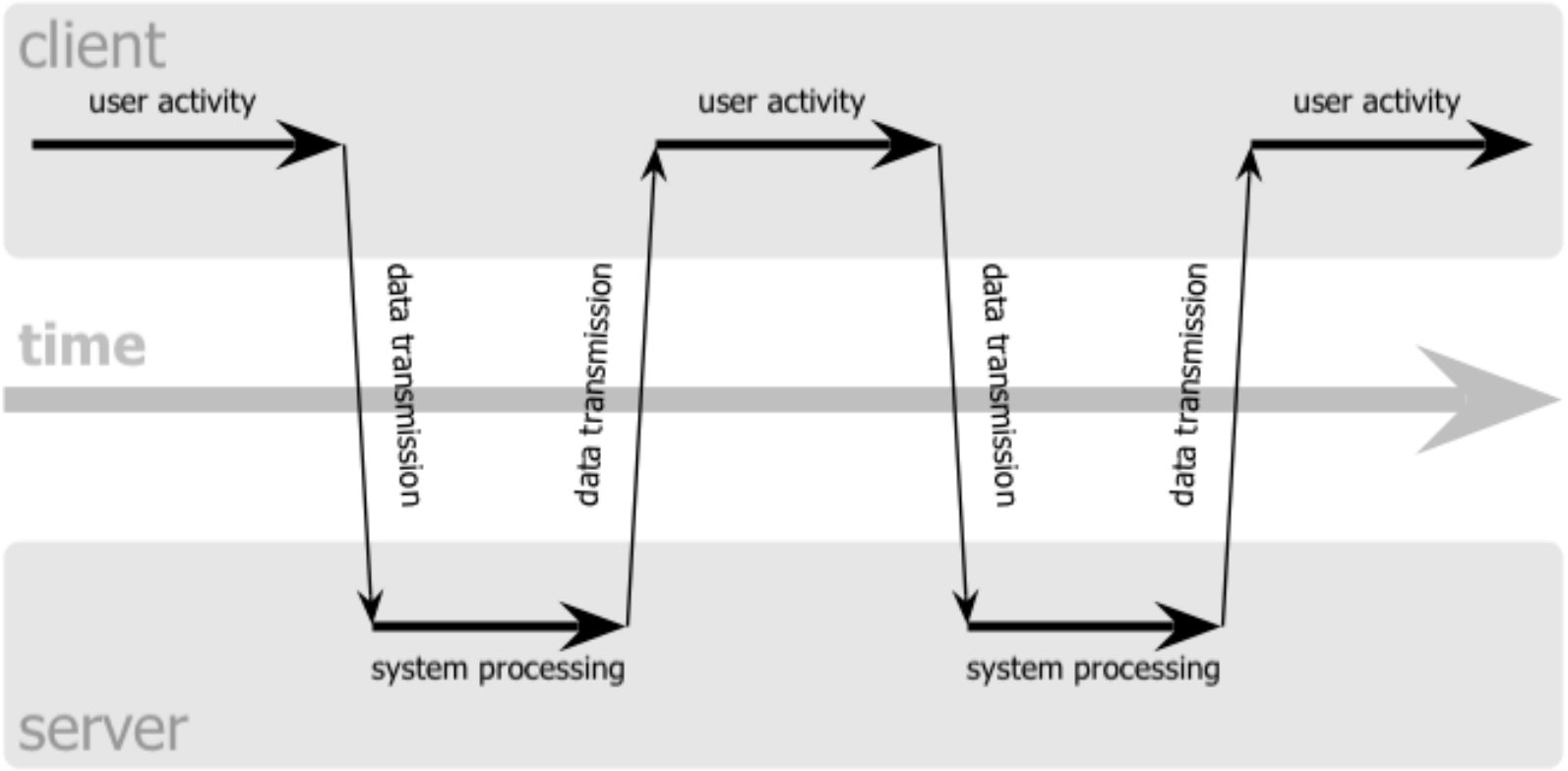
Interface Design

Interaction Design

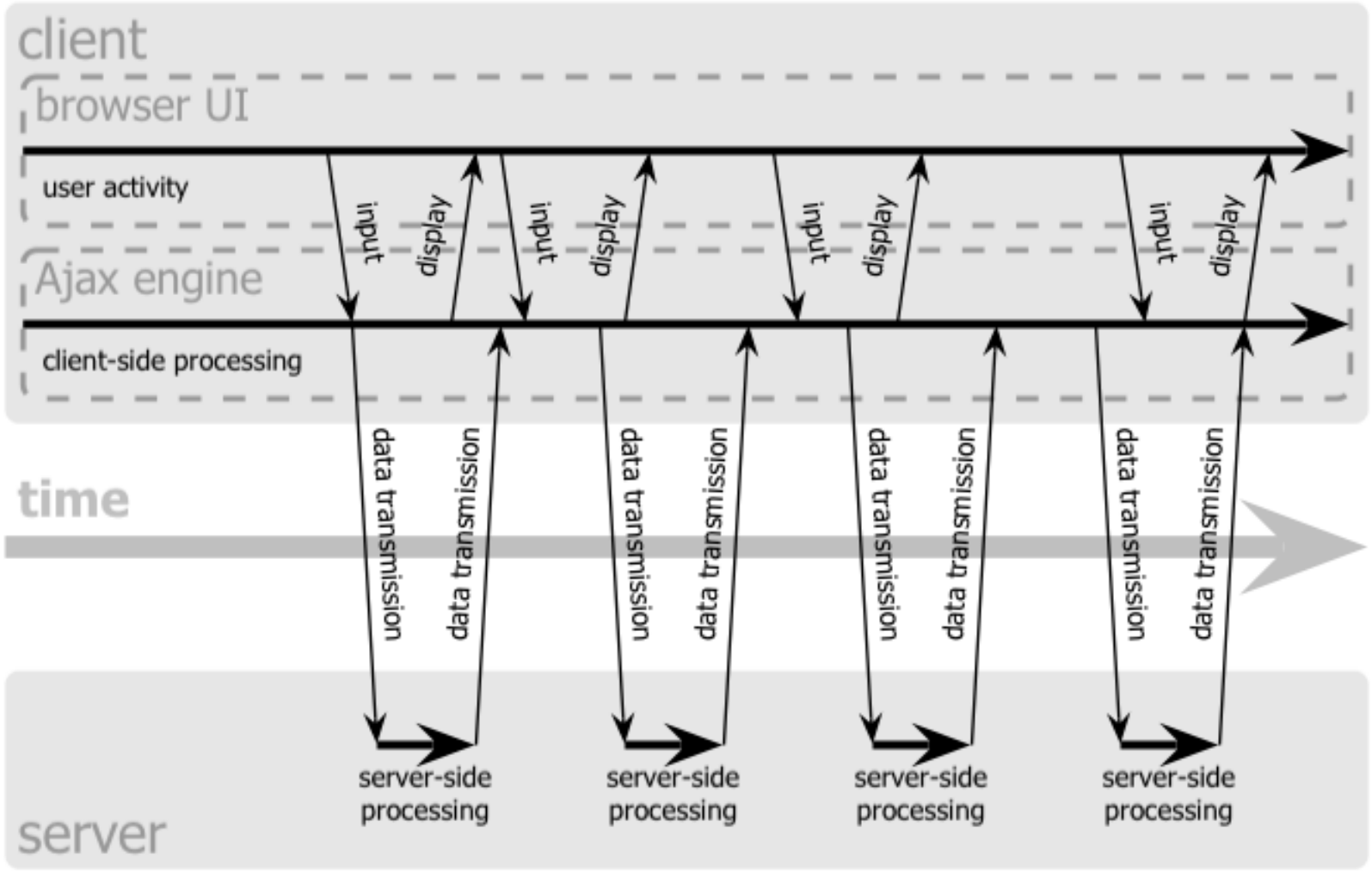
Business rules

Client Hardware	(PC)
Web Browser	(IE, Firefox)
Client-side Programming	(JavaScript)
Interchange Language	(HTML, XML)
Server-side Programming	(PHP)
Database	(MySQL)
Server Hardware	(PC, Unix)

classic web application model (synchronous)



Ajax web application model (asynchronous)



Ajax Applications

- Google Maps
  - http://maps.google.com
- Google Suggest
  - http://www.google.com/webhp?complete=1&hl=en
- Sajax Tables
  - http://labs.revision10.com/?p=5
- Sajax
  - http://www.modernmethod.com/sajax/

Before You Go

On a sheet of paper, answer the following (ungraded) question (no names, please):

What was the muddiest point in today's class?