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

Web Infrastructure

Week 2

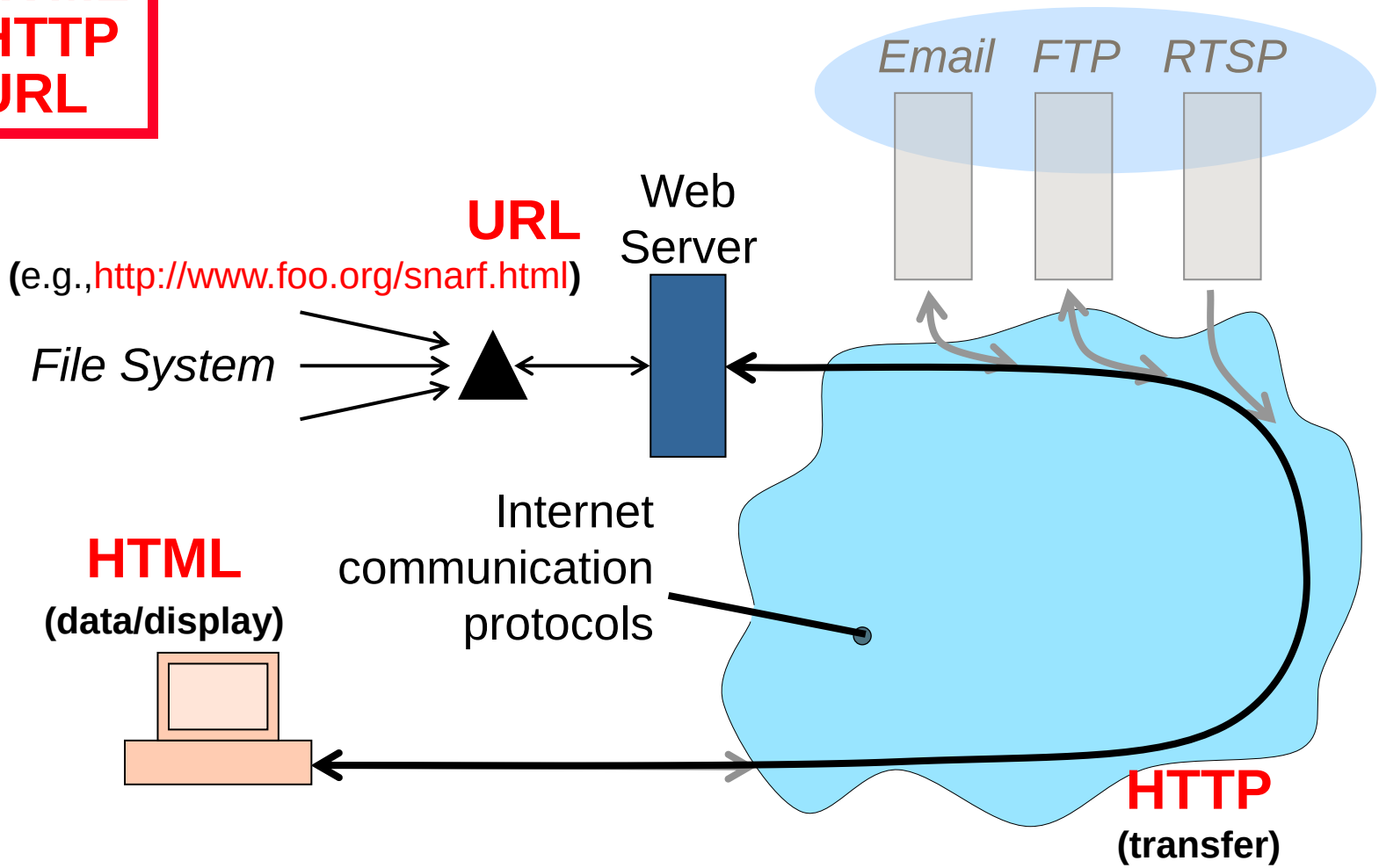
INFM 603

Agenda

- Questions
- XHTML
- CSS
- JavaScript

"The Web"

HTML
HTTP
URL



Hypertext “Anchors”

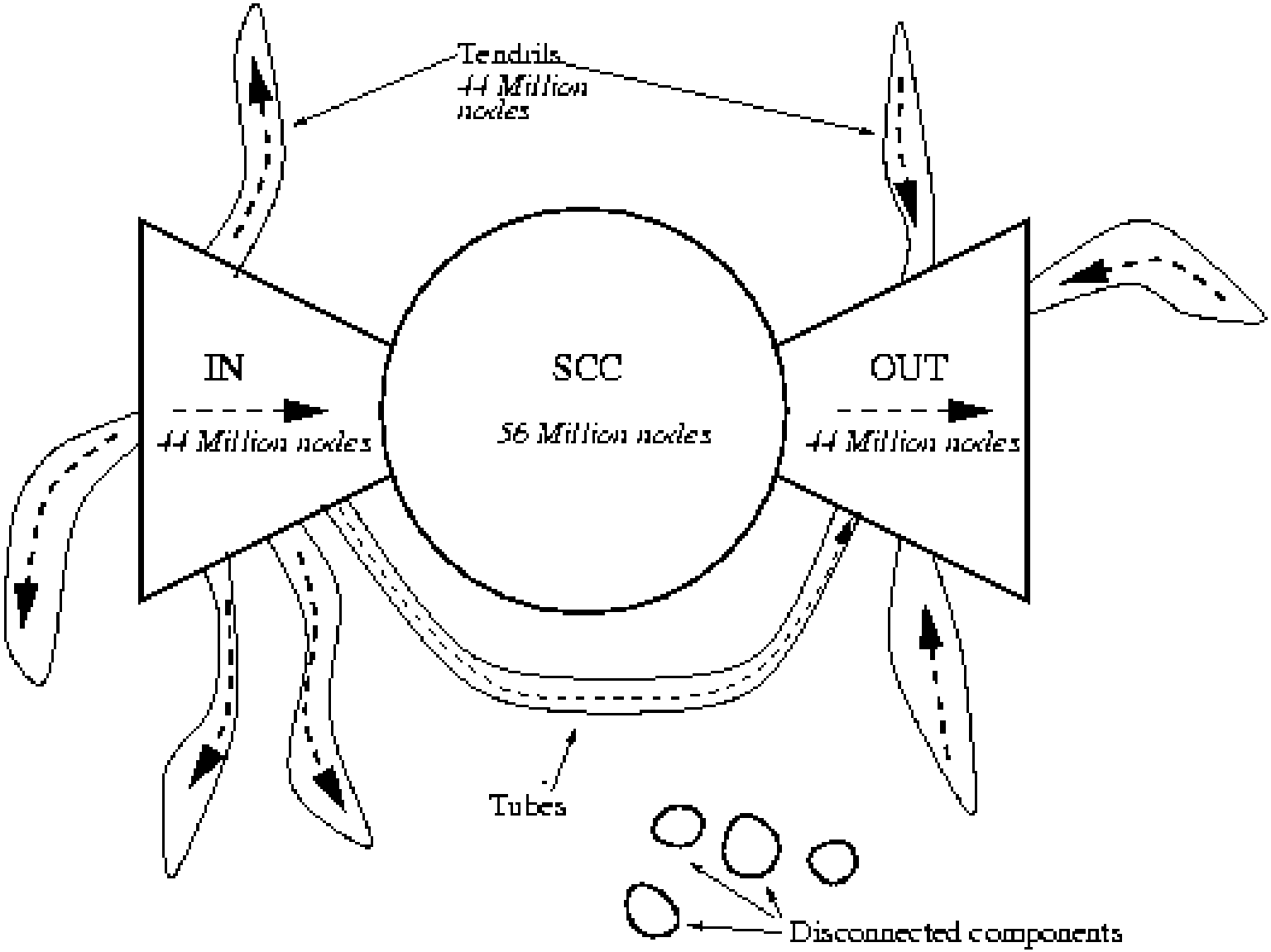
- Internal anchors: somewhere on the same page
  - Students
    - Links to: Student Information

- External anchors: to another page
  - iSchool
  - iSchool students

- URL may be complete, or relative to current page
  - 2

- File name part of URL is case sensitive (on Unix servers)
  - Protocol and domain name are not case sensitive

Link Structure of the Web



What’s a Document?

- Content
- Structure
- Appearance
- Behavior

“Hello World” HTML

This is the header

```
<html>
<head>
<title>Hello World!</title>
</head>

<body>

<p>Hello world! This is my first webpage!</p>

</body>
</html>
```

This is the actual content of the HTML document

Rendering

- Different devices have different capabilities
  - Desktop or laptop computer
  - Handheld device

- Rendering maps logical tags to physical layout
  - Controls line wrap, size, font…
    - Place the title in the page border
    - Render <h1> as 24pt Times
    - Render as bold

Logical Structure Tags

- Head
  - Title
- Body
  - Headers: <h1> <h2> <h3> <h4> <h5>
  - Lists: <ol>, <ul> (can be nested)
  - Paragraphs:<p>
  - Definitions: <dt><dd>
  - Tables: <table> <tr> <td> </td> </tr> </table>
  - Role: <cite>, <address>, , …

Physical Structure Tags

- Font
  - Typeface:
  - Size:
  - Color:
    - http://webmonkey.wired.com/webmonkey/reference/color_codes/Emphasis
  - Bold:
  - Italics:

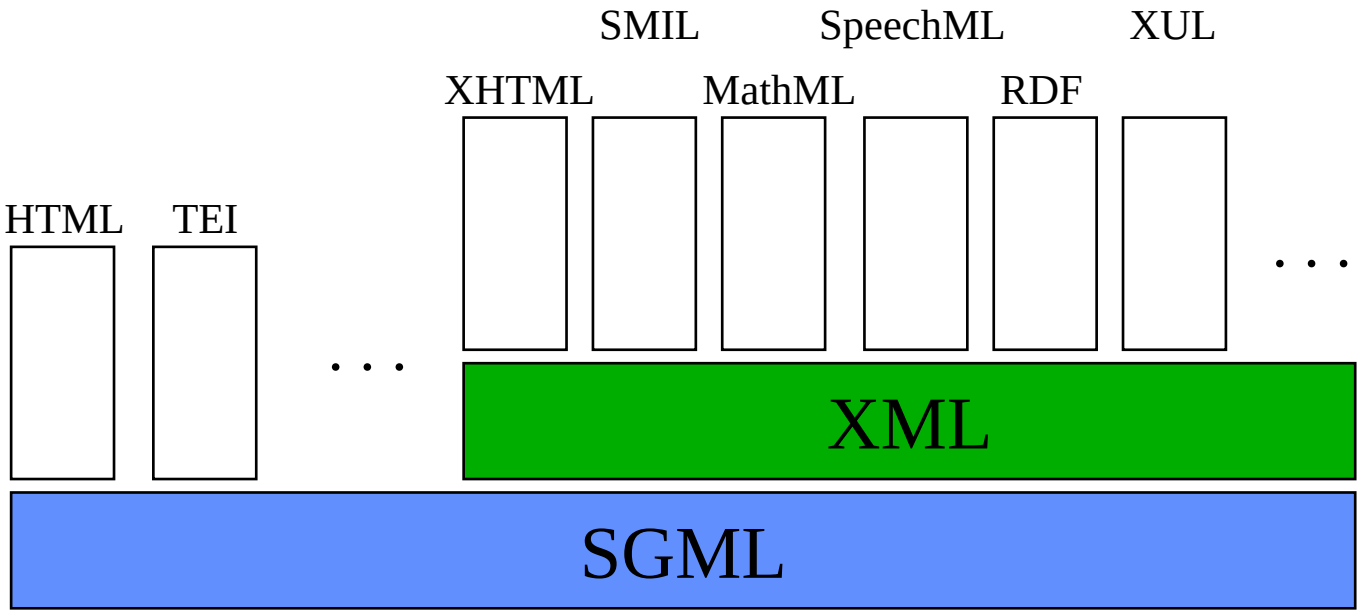
What’s Wrong with the Web?

- HTML
  - Confounds structure and appearance (XML)
- HTTP
  - Can’t recognize related transactions (Cookies)
- URL
  - Links breaks when you move a file (PURL)

History of Structured Documents

- Early standards were “typesetting languages”
  - NROFF, TeX, LaTeX, SGML
- HTML was developed for the Web
- Specialized standards met other needs
  - Change tracking in Word, annotating manuscripts, …
- XML seeks to unify these threads
  - One standard format for printing, viewing, processing

The XML Family Tree



Some Basic Rules for All XML

- XML is case sensitive
- XML declaration is the first statement
  - <?xml version="1.0"?>
- An XML document is a “tree”
  - Must contain one root element
  - Other elements must be properly nested
- **All** start tags must have end tags
- Attribute values must have quotation marks
  - <item id=“33905”>
- Certain characters are “reserved”
  - For example: < is used to represent <

XHTML: Cleaning up HTML

```
<?xml version="1.0" encoding="iso-8859-1"?>
<html xmlns="http://www.w3.org/TR/xhtml1" >
<head>
  <title> Title of text XHTML Document </title>
</head>
<body>
<div class="myDiv">
   <h1> Heading of Page </h1>
    <p> here is a paragraph of text. I will include  inside this paragraph
          a bunch of wonky text so that it looks fancy. </p>
    <p>Here is another paragraph with  <em>inline emphasized</em>
          text, and <b> absolutely no</b> sense of humor. </p>
    <p>And another paragraph, this one  with an <img src="image.gif"
         alt="waste of time" />   image, and a <br /> line break. </p>
</div>
</body></html>
```

Defining Blocks of Text

- …
  - Named region
  - Implies a paragraph break,
  - Can include multiple paragraphs
- …
  - Individual paragraph
- …
  - Any region
  - Does not create a paragraph break

CSS

- Separating content and structure from appearance
- Rules for defining styles “cascade” from broad to narrow:
  - Browser default
  - External style sheet
  - Internal style sheet
  - Inline style

Basics of CSS

- Basic syntax:

selector {property: value}

HTML tag you want to modify…

The property you want to change…

The value you want the property to take

- Example:

```
p { text-align: center;
    color: black;
    font-family: arial }
```

Causes

- Font to be center-aligned
- Font to be Arial and black

Different Ways of Using CSS

- Inline style:
  - Causes only this tag to have the desired properties

```
<p style="font-family:arial; color:blue“>…</p>
```

- Internal stylesheet:
  - Causes *all* tags to have the desired properties

```
…
<head>…
<style type="text/css" >
   p { font-family:arial; color:blue}
</style>
</head>
<body>
<p>…</p>
…
```

Customizing Classes

- Ability to define customized styles for standard HTML tags:

```
…
<head>…
<style type="text/css">
   p.style1 { font-family:arial; color:blue}
   p.style2 { font-family:serif; color:red}
</style>
</head>
<body>
<p class=“style1“>…</p>
<p class=“style2“>…</p>
…
```

External Style Sheets

- Store formatting metadata in a separate file

mystyle.css

```
p.style1 { font-family:arial; color:blue}
p.style2 { font-family:serif; color:red}
```

```
…
<head>…
<link rel="stylesheet" href="mystyle.css" type="text/css" />
</head>
<body>
<p class=“style1">…</p>
<p class=“style2">…</p>
…
```

HTML Editors

- Several are available
  - Macromedia Dreamweaver available commercially
  - Microsoft Word (Page->”Edit with Word” in IE)
- You may still need to edit the HTML file
  - Some editors use browser-specific features
  - Some HTML features may be unavailable
  - File names may be butchered when you upload
- Detailed patterns can make hand-editing difficult

Some Style Guidelines

- Provide appropriate “access points”
  - Users’ navigation strategies differ
- Design useful navigational aids
  - Search may lead users to the middle of a site
- Include some indication of recency
  - Date of last update, “new” icons, etc.
- Indicate who is responsible for the content
  - Helps readers assess authority

Some Accessibility Guidelines

- Design for device independence
- Maintain compatibility with earlier browsers
  - Provide alternative pages if necessary
- Provide alternatives to aural and visual content
  - Alt tags for images, transcripts for audio
- Make is easy for assistive devices to work
  - Give a title to each frame
  - Use tables only for data, not to control layout

Section 508 (Federal Web pages)

- A **text equivalent** for every non-text element shall be provided.
- Equivalent **alternatives for any multimedia** presentation shall be synchronized with the presentation.
- Web pages shall be designed so that all information conveyed with color is also **available without color**.
- Documents shall be organized so they are **readable without requiring an associated style sheet**.
- Redundant text links shall be provided for each active region of a server-side image map.
- **Client-side image maps** shall be provided instead of server-side image maps except where the regions cannot be defined with an available geometric shape.
- **Row and column headers shall be identified for data tables**.
- Markup shall be used to **associate data cells and header cells** for data tables that have two or more logical levels of row or column headers.
- **Frames shall be titled** with text that facilitates frame identification and navigation.
- Pages shall be designed to **avoid causing the screen to flicker** with a frequency >2 Hz and <55 Hz.
- A **text-only page**, with equivalent information or functionality, shall be provided when compliance cannot be accomplished in any other way. The content shall be updated when the primary page changes
- When pages use **scripting languages** to display content or to create interface elements, the information provided by the script shall be identified with functional text that can be read by assistive technology.
- When a web page requires that an **applet**, plug-in or other application be present on the client system to interpret page content, the page must provide a link to a plug-in or applet that complies with the above.
- When electronic **forms** are designed to be completed on-line, the form shall allow people using assistive technology to access the information, field elements, and functionality required.
- A method shall be provided that permits users to **skip repetitive navigation links**.
- When a timed response is required, the user shall be alerted and **given sufficient time** to indicate more time is required.

Validation Services

- HTML cross-browser compatibility
  - http://validator.w3.org
- CSS cross-browser compatibility
  - http://jigsaw.w3.org/css-validator/
- Section 508 compliance
  - http://www.cynthiasays.com/
- Try them on http://www.umd.edu ☹

Programming for the Web

- JavaScript [Client-side]
  - Server embeds a program in HTML
  - Browser runs the program when it gets to it
- PHP “Common Gateway Interface” [Server-side]
  - HTML form sends field values to the server
  - Server passes field values to a program
  - Program generates a Web page as a response
- Ruby on Rails [Ajax]
  - Server sends browser a generic program to run
  - Browser and server programs exchange XML-encoded data

Software

- Software **models** some 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

Programming Languages

- Used to specify every detail of the model

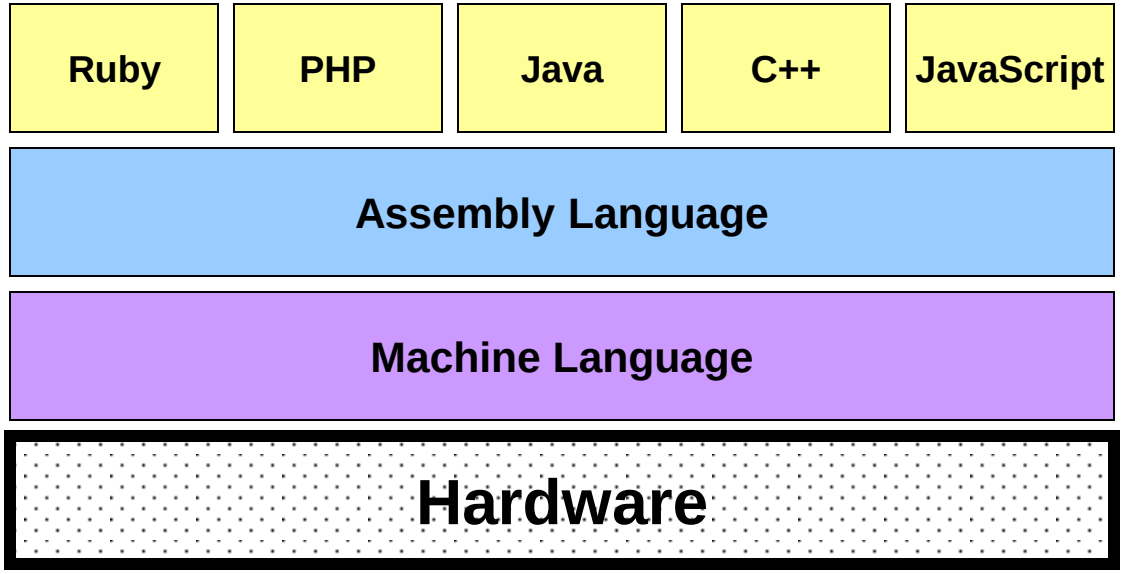
- Special purpose
  - Able to specify an entire class of models
    - Spreadsheets (Excel, ...)
    - Databases (Access, Oracle, ...)

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

History of Programming

- Machine language
  - Language that machine can understand
- Assembly language
  - Assembler translates “symbolic” references to machine instructions and memory locations into machine code
- High-level languages
  - Compiler rewrites everything in machine code OR
  - **<u>Interpreter</u>** performs the specified actions at “run time”

Programming Languages



Machine Language

- **<u>Everything</u>** is a binary number
  - Operations
  - Data

00001000	00010101	01010110

00001000	ADD
00010101	number to be added (21)
01010110	memory location to add it to (86)

Assembly Language

- **<u>Symbolic</u>** instructions and addresses
  - Symbolic instruction “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, PHP, Ruby

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>
```

Variables

- Data types:
  - Boolean: true, false
  - Number: 5 9 3.1415926
  - String: “Hello World”

- A “variable” holds a value (of some data type)
  - Represented as “variable names”: x celsiusTemp
  - Variables are “dynamically typed” at run time
    - If you use it as a string, it’s a string …
  - Variable names are case sensitive in JavaScript

Operators

-x	reverse the sign of x (negation)
6+5	Addition (numeric)
“Hello” + “ World”	Concatenation (strings) [note the space]
2.1 * 3	Multiply (treats the int as a float)
4 + “ Horsemen”	Concatenation (treats 4 as a string)

Assignment Statements

- Assignment **<u>sets</u>** the value of a variable

x = 5	set the value of x to be 5
	[a command, not an assertion!!]
x = 5*x	Multiply x by 5 and save the result in x
x = 5+8/4*2	Set x to 5+((8/4)*2) [precedence rules!]
x += y	x = x + y
x++	x = x + 1

- JavaScript statements end with a semicolon
  - Optional at the end of a line

Some Useful Predefined “Methods”

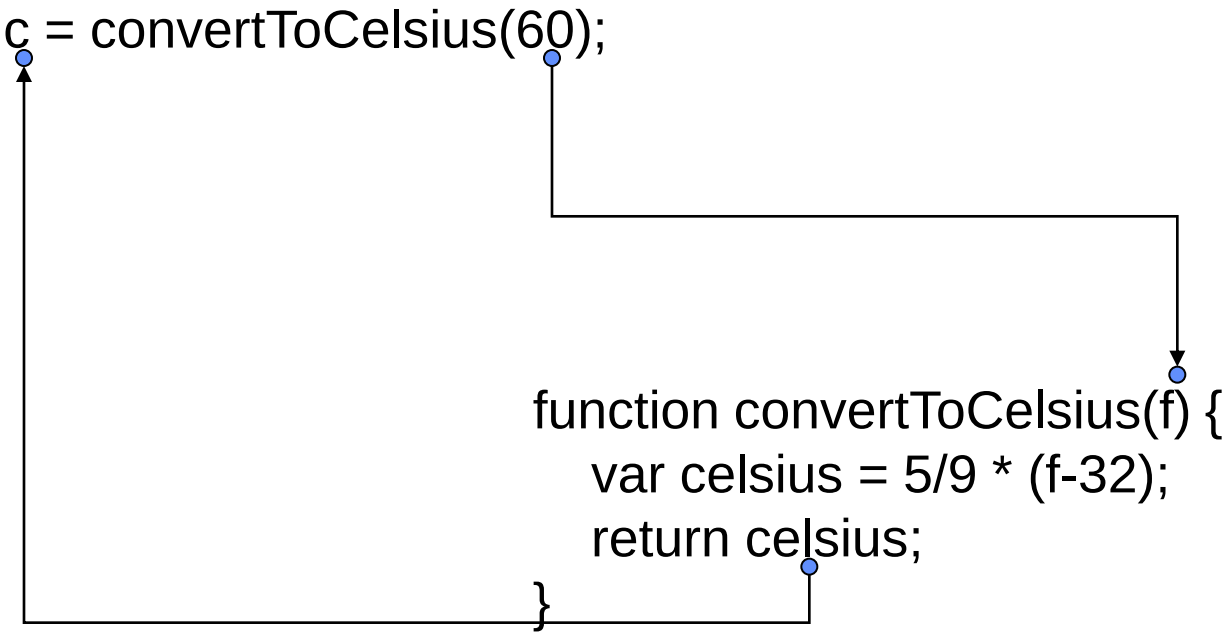
- document.writeln(“…”);
  - String gets **<u>rendered</u>** as (X)HTML
  - Include “<br />” to force a line break
- window.alert(“…”);
  - String is **<u>written verbatim</u>** as text
  - Include “\n” to force a line break
- foo = window.prompt(“…”);
  - String is **<u>shown verbatim</u>** as text
  - Result is whatever string the user enters

Functions (non-object "Methods")

- 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);
```



Writing JavaScript Functions

- Convenient to put it in the <head> section
  - Use <!-- … //--> to prevent display of code

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

Scope of a Variable

- In JavaScript, *var* "declares" a variable

var mystery;	create a variable without defining its type
var b = true;	create a boolean *b* and set it to true
var n = 1;	create an integer *n* and set it to 1
var s = "hello";	create a string *s* and set it to "hello"

- Variables declared in a function are "local"
  - Same name outside function refers to **<u>different</u>** variable

- All other variables are "global"

More JavaScript Statements

- Invocation of a function

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

- Return a value from a function

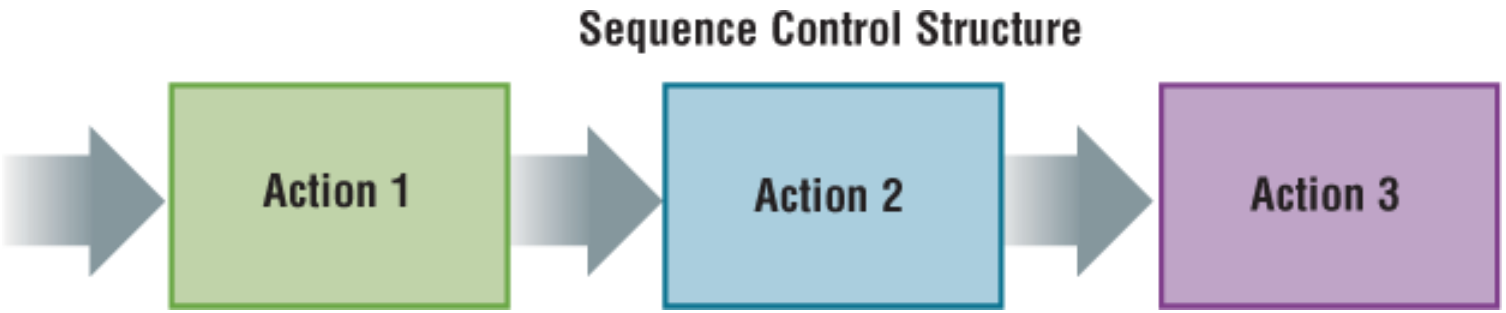
```
return celsius;
```

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

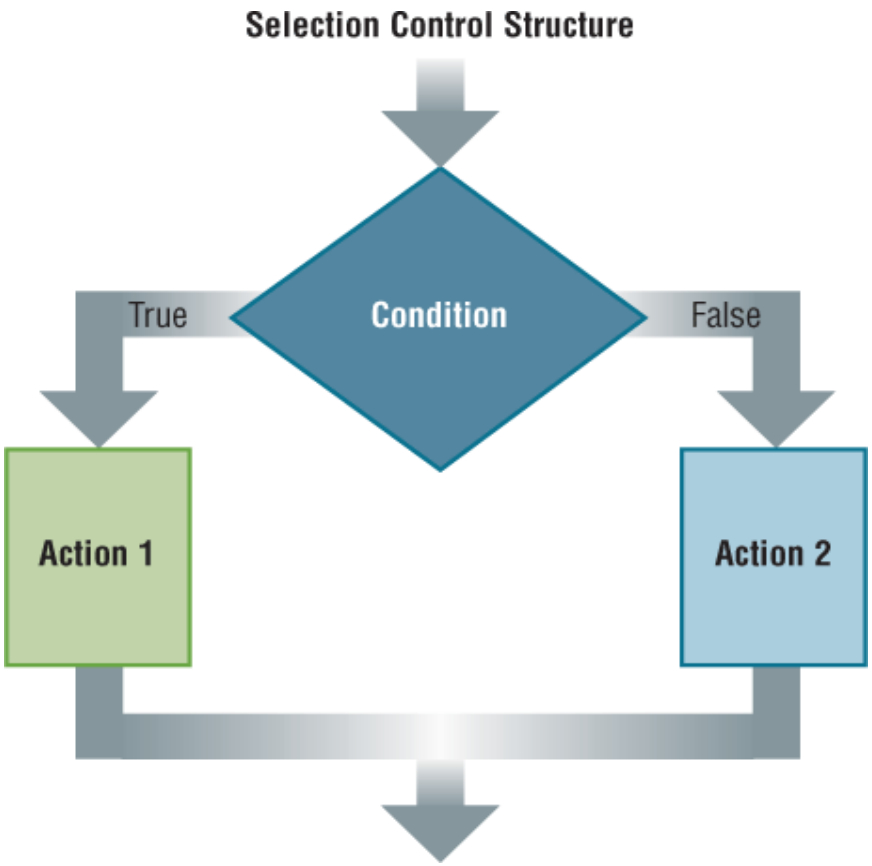


a = 2;

b = 3;

c = a * b;

Conditional Selection Control Structure

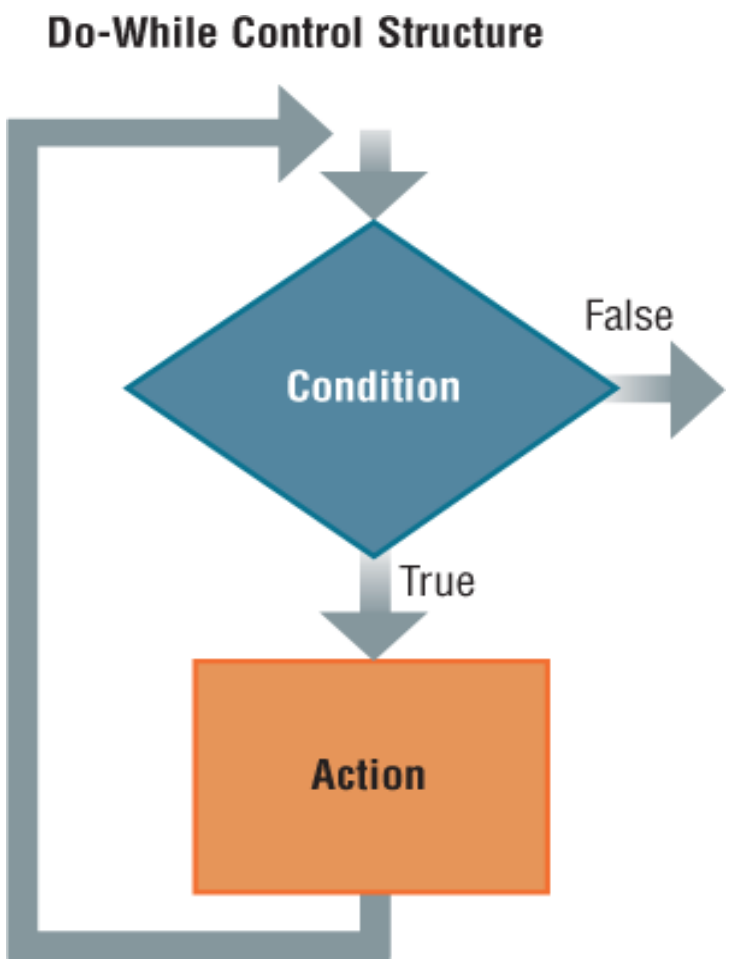



```
if (gender == “male”) {
    greeting = “Hello, Sir”;
}
else {
    greeting = “Hello, Madam”;
}
```

Boolean Operators

- x == y true if x and y are equal [use == not =]
- 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 <u>elements</u>
  - For example, the number of days in each month
- Each element is assigned an <u>index</u>
  - A number used to refer to that element
    - For example, x[4] is the <u>fifth</u> element (count from zero!)
  - Arrays and repetitions work naturally together

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

Using JavaScript with Forms

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 from the first form
(*eval* method turns it into a number)

JavaScript:

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

Changes the value in the second form

Hands On:
Adopt a JavaScript Program

- Launch a Web browser
  - http://www.umiacs.umd.edu/~oard/teaching/603/fall11/slides/2/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, semicolon, 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?

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?