COMP303
Internet Computing

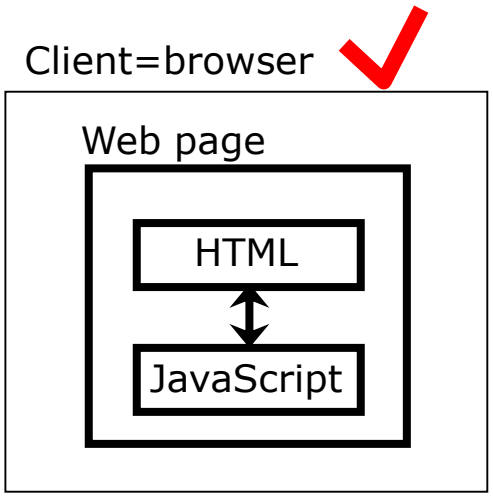
JavaScript 1

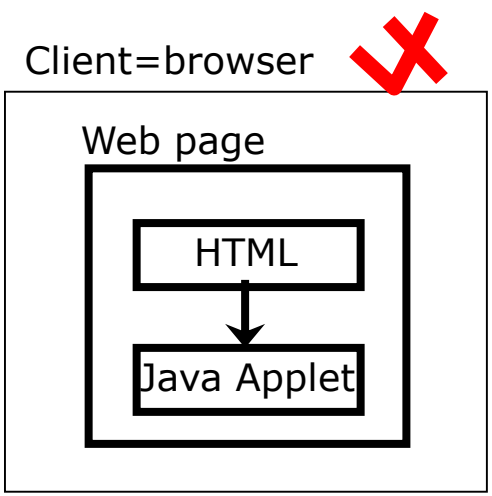
David Rossiter

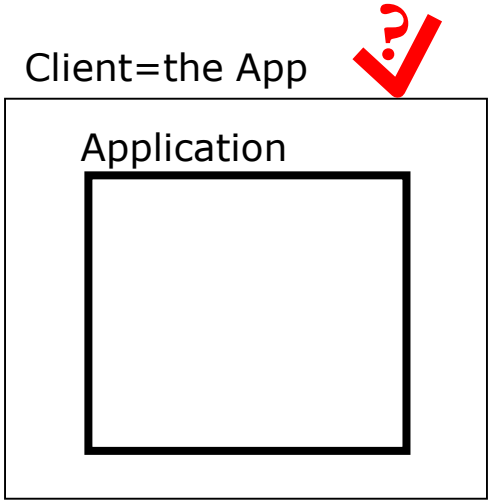
Client-Side Scripting Languages

- Scripting languages are used to control a browser's behavior
  - VBScript: a subset of VB from Microsoft
  - JavaScript: Sun/Netscape
  - ActionScript: Flash; very much like JavaScript
- Use search engine to test popularity; JavaScript is 100 times more popular than VBScript

JavaScript/ Java







Using JavaScript

HTML objects have events which can trigger JavaScript code. JavaScript can generate and control HTML objects.

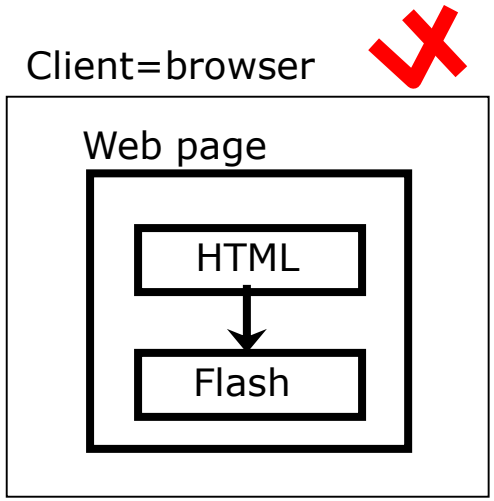
A Java Applet

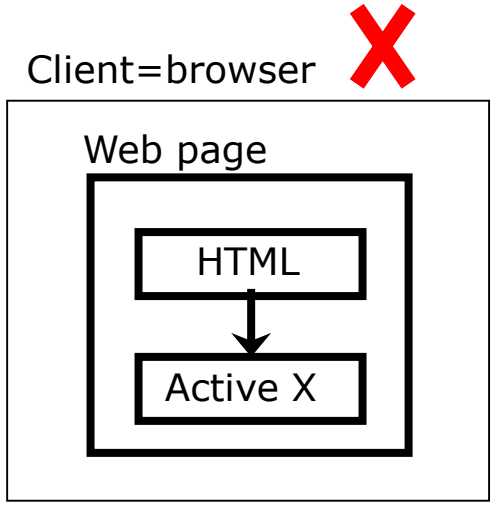
Parameters are typically passed from the HTML to the applet, once only, when the applet starts.

A Java Application

(stand-alone program)

Flash/ Active X





These are the most common approaches to scripting/programming inside a web page, but there are some others.

Flash

If any parameters are passed to Flash then they are typically passed from the HTML to the flash program, once only, when it starts.

An ActiveX component in a web page

Active X components can have a lot of access to Windows procedures. However, there have been security issues. Also, they are not 'naturally' supported by browsers except IE.

JavaScript Vs. Java

- 'JavaScript' sounds like it must be similar to 'Java'…
- No - it is very different
- JavaScript is a scripting language embedded in an html file
  - Java can be used for 'stand alone' applets which are embedded inside the page
- JavaScript can manipulate HTML elements
  - Java applets are usually isolated from the web page
- JavaScript - no 3D graphics library/ threading/ networking etc
  - Java - has lots of things like that

JavaScript Engines

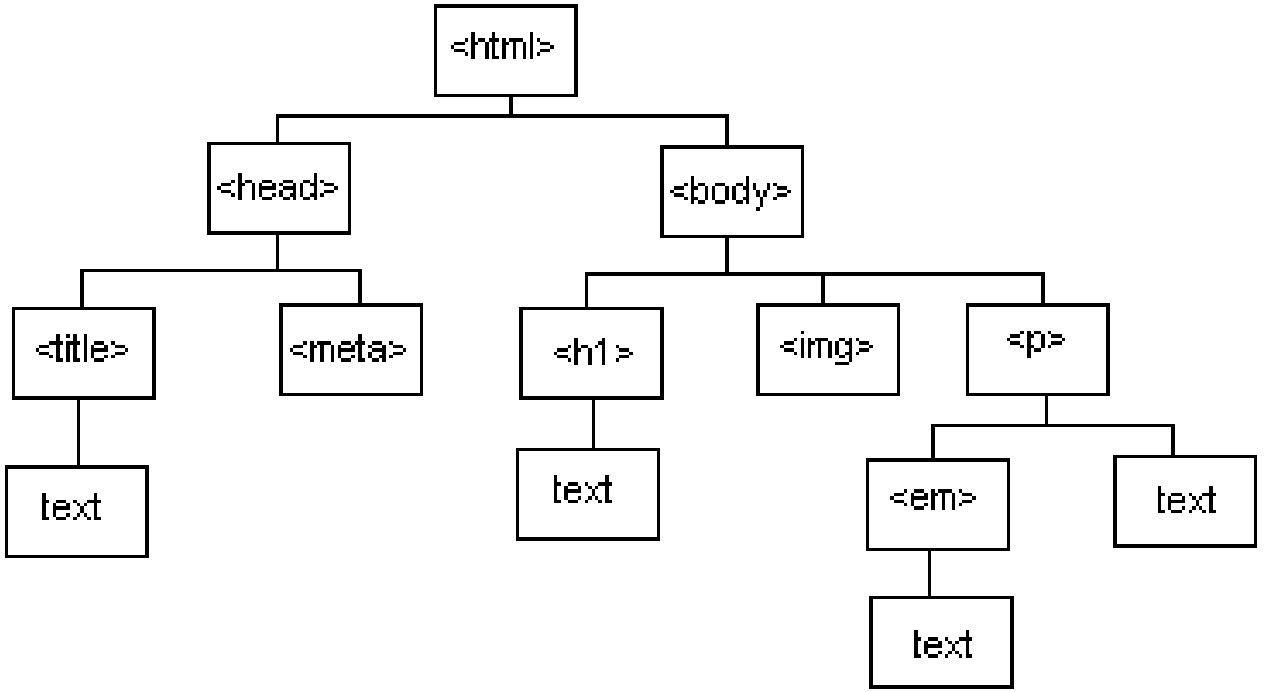
- A JavaScript *engine* is the software inside the browser which runs the JavaScript code
- The speed of the JavaScript engine determine (almost) the speed of a browser as JavaScript programs are getting larger and larger and more and more sophisticated (e.g., gmail)
- Two open-source examples: SpiderMonkey and Rhino
- Chrome v8 open-source JavaScript Engine

- Used in the Firefox browser
- Used by various programs

The DOM

- JavaScript can be used to control all the browser components, which includes the web page, through a memory structure called the DOM
- DOM=Document Object Model
- The DOM is a tree structure

Example Tree Structure DOM



- This image gives a rough idea of a DOM structure for a browser after loading a web page
- We will look more at the DOM later

Example of a Simple Web Page Using JavaScript

```
<html>
<head>
  <title>A Simple Example Of Using JavaScript
   </title>

  <script language="javascript"
   type="text/javascript">

    <!--

    document.write("Welcome!");

    // -->

  </script>

</head>

<body>
<p>
This is the first text line of the HTML page.
The browser will look at JavaScript which is in the head
part of the page before it looks at the first line of html text.
</p>
</body>
</html>
```

Enclose script content as a comment to hide it from browsers that don't understand JavaScript

- JavaScript needs to be put inside <script> … </script>
- The script is executed in place; it is equivalent to putting the string "Welcome!" inside the head tag; note: it is not a good practice to output a message in the header

- <!-- signifies a one-line comment for JavaScript, same as // …
- Pretend you are a browser not supporting JavaScript, what do you see?

Order of Assessment

```
<html>
<head>
  <title>A Simple Example Of Using JavaScript
   </title>

  <script language="javascript"
   type="text/javascript">

    <!--

   document.write("Welcome!");

    // -->

  </script>

</head>

<body>
<p>
This is the first text line of the HTML page.
The browser will look at JavaScript which is in the head
part of the page before it looks at the first line of html text.
</p>
</body>
</html>
```

- *This part of the web page was assessed by the browser first*
- *The browser saw there was a direct instruction to do something, so it did it*
- *Then this part of the web page was assessed*

Note: the script is executed in place; thus, it is equivalent to putting the string “Welcome!” inside the head tag, which is not a good way to output a message on the page

document.write()

- The instruction `document.write("Welcome!")` tells the browser to write the word to the document, meaning the web page
- The result is that those words are added to the web page at the point where the JavaScript is executed
- The user can immediately see the words in the web page

Simple Text Output – alert()

- For showing text to the user, a quick and easy solution is to use alert( ) i.e.

```
alert("Welcome!");
```

- Alert is one of the three dialog boxes supplied by JavaScript:

```
alert()
prompt()
confirm()
```

Basic Example

```
<html>
<head>
  <title>A Simple Example Of Using JavaScript
   </title>

  <script language="javascript" type="text/javascript">

    alert("Welcome!");

  </script>

</head>

<body>
<p>
This is the first text line of the HTML page.
The browser will look at JavaScript which is in the head
part of the page before it looks at the first line of html text.
</p>
</body>
</html>
```

- *Only this part has changed - now we are using alert()*

Simple Text Input – prompt()

- For getting input from the user, one easy-to-handle way is to use prompt(), for example:

```
var user_name;  // Declare the variable

user_name=prompt("What is your name?", "" );
```

Default string shown in the prompt box

- In JavaScript it is not actually required to declare a variable before you use it
- However, it is good programming practice

Combining Both Together

- Simple text input and text output

```
var user_name;
user_name=prompt("What is your name?",  ""  );
alert("Welcome " + user_name + "!");
```

- Here the '+' means string concatenation, not numerical addition
- The choice between string or numerical handling is automatically made by the JavaScript engine

Simple Selection – Confirm()

```
if (confirm("Select OK to continue, Cancel to abort"))
{
    document.write("OK, I will continue");
    …
} else {
  document.write("Operation cancelled…");
}
```

Be Careful When Programming

```
var user_identifier="";
user_identifer=prompt("What is your account name?", "" );
alert(user_identifier);
```

- The above code would be happily executed, with no execution error, or any other kind of error shown
- However, the alert box would always display an empty string regardless of what the user enters

About Semicolons in JavaScript

In JavaScript, this works:

```
function do_numbers(){
var result_str = "";
for (var i = 0 ; i < 10; ++i)
   result_str += i + " ";
return result_str;
}

document.write(do_numbers());
```

This also works:

```
function do_numbers(){
var result_str = ""
for (var i = 0 ; i < 10; ++i)
   result_str += i + " "
return result_str
}
document.write(do_numbers())
```

About Semicolons in JavaScript, Cont.

But this doesn't work:

```
function do_numbers() {
var result_str = ""
for (var i = 0 ; i < 10; ++i) result_str += i + " "      return result_str
}
document.write(do_numbers())
```

JavaScript Variable Types

- Number: an integer/ floating-point number
- String: alphabet/ numerals/ any other characters
- Boolean: *true* or *false*
- Null: Consists of the value *null*
- Undefined: Consists of the value *undefined*

- Unlike most languages, when you create a variable you don't need to specifically declare what type it is
- Like most languages, you can have global or local variables

Example JavaScript Operators

```
var x=10; result=x++;    // result is now 10, x is 11
var x=10; result=++x;    // result is now 11, x is 11
var x=10; result=x--;    // result is now 10, x is 9
var x=10; result=--x;    // result is now 9, x is 9

var x=10; result=-x;     // result is now -10, x is 10

result=10 % 3;           // result is 1
result=10 / 3;           // result is 3.333333
```

Numerical Input

- When the user enters something into a prompt( ) box, it is a string
- If you want to handle the input as if it is a number, you need to first convert the string into a number. For example:

```
var user_age_text;
var user_age;
user_age_text=prompt("What is your age?", "" );
user_age=  parseInt(user_age_text);
if (user_age<=12)
    alert("Young student!");
```

Always returns a string value

Convert the string into a number

Program Flow

- JavaScript has all the usual program flow constructs, i.e.
  - `If / else / else if`
  - `switch`
  - `while { } / do { } while`
  - `for ( )`
  - It also has: `break, continue` – discussed later

Example Else If

- The name is converted to lower case (i.e. "RoSSiteR" becomes "rossiter") before it is compared

```
var user_name;
user_name=prompt("What is your name?", "" );
if (  user_name.toLowerCase( )    == "dave")
    alert("Great name!");
else if (user_name.toLowerCase( ) == "gibson")
    alert("OK name!");
else alert("Your name isn't great… never mind");
```

Example Switch

```
var user_name;
user_name=prompt("What is your name?", "" );

switch ( user_name.toLowerCase() ) {
   case "dave":
        alert("Great name!");
        break;
   case "gibson":
        alert("OK name!");
        break;
   default:
        alert("Your name isn't great… never mind");
        break;
}
```

- Usually a 'switch' statement is more efficient than several if else statements

Example While

- The loop is terminated if the first letter is a ‘y’
- For example: yes, yea, yep, and y

```
var response;
var finished;

finished=false; // At the start, we haven't finished yet
alert("Rossiter is a great name.");

while (!finished) {
   response=prompt("Do you agree?", ""); // input from user
   if   ( response.indexOf("y")  ==  0    )          // First letter
                                                     //   must be y
            finished=true;          // loop will now finish
}
```

- Search for ‘y’ in the user’s response
- What does indexOf() do?

Break and Continue

- Useful JavaScript commands for loop control:
  - break – to stop the loop and jump to the command immediately following the loop
  - continue – to skip the rest of the current iteration

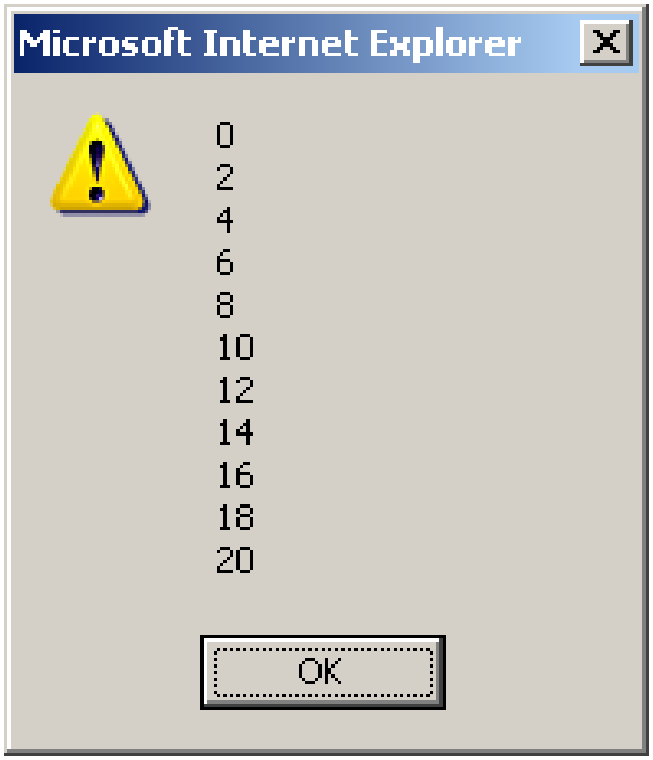
Break

```
var message = "";
var count = 1;
while (count <= 10)
{
   if (count == 8)
   {
      break;
   }
   message = message +
             count + "\n";
   count++;
}
alert(message);
```

Continue

```
var message = "";
for (var x = 0; x <=20; x++)
{
   if (x%2)
   {
       continue;
   }
   message = message + x + "\n";
}
alert(message);
```

- True if x=1, 3, 5, …



Examples of Logical Operators

- Example of ! Not

```
if (!parseInt(prompt("At what age were
  you born?",""))==0) alert("Crazy!")
```

- Example of && And

```
var response=prompt("Male or female?");
if ((response!="male")&&(response!="female"))
  alert ("Huh??!");
```

- Example of || Or

```
var response=prompt("good, great or bad?");
if ((response=="good")||(response=="great"))
  alert ("Me too!");
```

JavaScript Events

- Typically, events are caused by user interaction
- For example, the following would each cause an event:
  - moving the mouse over an image
  - clicking on a button
  - changing a value in a textbox

Example Events

- For keyboard input:
  - onkeypress
  - onkeydown
  - onkeyup

- For mouse input:
  - onclick
  - onmousedown
  - onmouseup
  - onmousemove

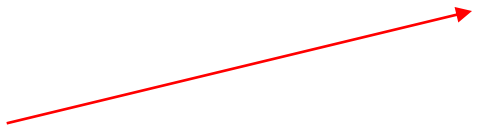
- When an object is loaded by the browser:
  - onload

Functions

```
<html> <head>
<script language="javascript"
                type="text/javascript">
<!--
function check_user_age( ) {
   if ( age_of_user( ) < 18 )
        alert("Please go to another web page.");
}

function age_of_user( ) {
   var age_text, age;
   age_text=prompt("What is your age?", "");
   age=parseInt(age_text);
   return(age);
}
//-->

</script>
</head>
```



```
<body onload=" check_user_age()   ">

<h1>This is my
   naughty home page….</h1>

</body>
</html>
```

Referring to Strings

- Use double quotes `" "` or single quotes `' '`
- When you refer to a string within a string, you have to use the other type of quotes for the inner reference
- For example:

```
<body onload="alert( 'Welcome!' )">
```

Handling Random Numbers

- Generate a random number like this:
  - `random_number=Math.random();`
  - This produces a floating point value in the range 0 to 1
- The resulting range is [0, 1) - in other words, the value of 1 will not be generated
- Multiply in order to get the range you want, i.e.
  - `random_number=Math.random() * max_value;`
- Truncate the decimal place – i.e. 12.97 -> 12
  - `Math.floor( random_number )`
- We now have an random integer in the range [0, max_value)

JavaScript Objects

- An object has some fields (or properties)
- Example 1:

```
var this_thing= new Object();
this_thing.name = "demo field";
this_thing.value = 303;
```

(No need to declare fields!)
(This field is dynamically added)
(This field is dynamically added)

- Example 2:

```
another_thing = { x:3, y:4, z: 5};
```

does the same as

```
another_thing = new Object();
another_thing.x=3; another_thing.y=4; another_thing.z=5;
```

- Accessing all fields/properties in an object:

```
for (field in another_thing)
  document.writeln(another_thing[field]);
```

another_thing.field

JavaScript Arrays

- Example:

```
var squares = new Array(5);
for (var i=0; i<squares.length; i++)
   squares[i]=i*i;
```

has the same result as

```
var squares = new Array(0, 1, 4, 9, 16);
```

- Examples of assigning values:

```
var arrayobj = new Object();
arrayObj[0] = "index zero";
arrayObj[10] = "index ten";
arrayObj.field1 = "field one";
arrayObj["field2"] = "field two";
```

- Arrays can be used like hashes

Take Home Message

- JavaScript is a powerful object-oriented language
  - Processing capability not much different from Java, C++
  - With more emphasis on user interaction (control of display, mouse and keyboards) and security (restricted access to client storage)
  - No fancy features like multithreading, networking, etc.
- JavaScript runs on browsers and enables them to do wonderful things
- We learnt simple user interaction (prompt and alert), string manipulation, math (Math.random(), important for games), simple event handling
- DOM is the common data model for client and server (will be covered more in next set of slides)