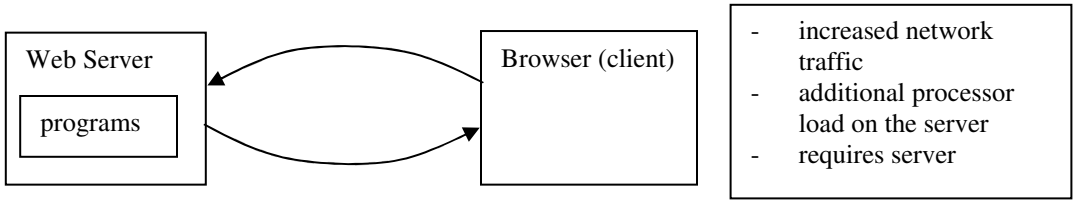
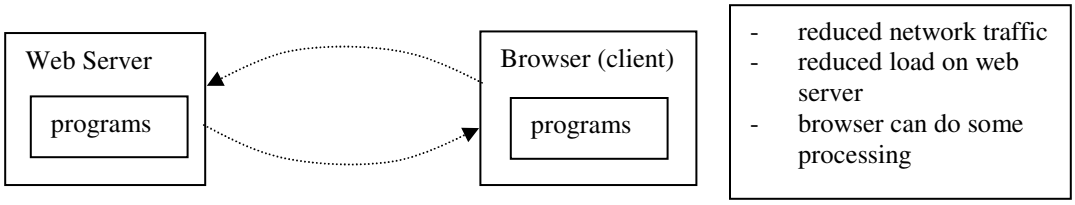
Introduction to Javascript

- static versus dynamic web pages
  - basic HTML allows us to prepare the appearance of a web page
  - if you want the layout and content of the page to respond to user events dynamically, then additional features are needed in the web page more than basic HTML
  - approaches to provide this include dynamic HTML (DHTML), Java, Javascript, VBScript, Perl scripts
- server-side and client-side programs

  Server-side: all program functionality resides on the web server; the browser sends data to the server which must process the information and send it back to the browser



  Client-side: push as much program functionality down to the browser as possible; the browser handles as much information processing as it can



- problems with server-side programs
  - users had to be connected to the web server to run the CGI script
  - only the programmer could alter the script
  - problems for the system administrator
  - load on the server

- development of Java and Javascript
  - early 1990's before the development of the WWW, programmers at Sun Microsystems were looking for a system so that all consumer and household devices could be all networked together and controlled as a unit
  - development lead to Java and HotJava
  - applets nestled into web pages
  - problems with Java
    - complex and difficult for new programmers
    - requires the JDK (Java Developer's Kit) to create applets
    - programs must be compiled and saved as separate files before they can be used
    - Java programs tend to be slow

- solution: simplify Java to a form called Javascript
  - a simple Java-like language to be used only within HTML documents
  - no developer's kit is required
  - scripts are written directly into the HTML file and require no compiling
  - is interpreted not compiled
  - Javascript scripts are executed on the fly
  - JavaScript has limitations as far as extended the language and client-server interaction designed for simple, small programs like calculators for example

- can respond easily to user events like mouse clicks and data entry in forms

JavaScript - Strengths

- quick development
- no compilation
- interface features such as dialog boxes handled by the browser and HTML code
- easy to learn
- doesn't share the more complex syntax of Java
- platform independence
- small overhead

JavaScript - Weaknesses

- limited range of built-in methods (functions)
- might change with next release
- no code hiding
  - JavaScript visible within the HTML
  - consensus is that JavaScripts are freeware
  - lack of debugging and development tools

JavaScript and HTML

HTML tags `<script> </script>` contain the JavaScript portion within the HTML document

```
<script type="text/javascript" language="JavaScript">
      …JavaScript program goes here
</script>
```

Hiding scripts from browsers that do not support JavaScript

```
<script type="text/javascript" language="JavaScript">
<!-- Hide the Script from other browsers

      …JavaScript program

// Stop hiding from other browsers -->
</script>
```

Where to place the JavaScript code ?

JavaScript programs can be included anywhere in the header or body of an HTML file but the preferred location is in the header just after the </title> tag.

long and complex scripts can be placed in a separate file (must have a .js file extension) and included in the HTML file:

```
<script type="text/javascript" language="JavaScript"
                              src="http://www.you.com/JavaScript.js"
</script>
```

JavaScript - Sample 1

```
<html>
<head>
<title>JavaScript Sample </title>
</head>
```

```
<body>
  Here's a sample JavaScript:
  <script type="text/javascript" language="JavaScript">
  <!-- Hide from old browsers

  // Display a message
  document.writeln("Have a nice day!< /br>");

  // Stop hiding from other browsers -->
  </script>
</body>
</html>
```

Javascript command syntax

Syntax is the set of rules that determine the correct formation of a language statement.

Since Javascript program code is "free form", you need to tell the interpreter where the end of the statement is.

Syntax rule #1 in Javascript: All Javascript statements end with a semicolon.
Syntax rule #2 in Javascript: Comments begin with a double / and continue to the end of the line.
Syntax rule #2 in Javascript: Javascript keywords (like document, writeln) are in lowercase.

```
document.writeln("Hello");
```

this JavaScript statement invokes the method writeln(), which is part of the document object
in JavaScript, as in Java, everything is case-sensitive.

```
document.writeln("Hello< /br>");
```

outputs the text Hello< /br> to the browser which interprets it as HTML information
writeln is one of many methods or functions associated with the object document
in JavaScript, methods are called by combining the object name with the method
object-name.method-name.

Data that the method needs to perform is provided as an argument in the parentheses

```
document.write("Welcome !");
document.writeln("Have a great day!");
```

The "write" sends the text on the current line in the browser. The "writeln" sends the text on a new line in the browser. The SCRIPT container does not affect the HTML structures where it occurs, so any format tags or elements in the HTML file will affect the text produced by write()

```
<html>
<head>
<title>JavaScript Sample </title>
</head>

<body>
Here's a sample JavaScript <b>
<script type="text/javascript" language="JavaScript">
<!-- Hide from other browsers
```

```
// Display a message
document.writeln("This is in bold.</b>");

// Stop hiding from other browsers -->
</script> </body> </html>
```

Interacting with the user

- simplest way is with the prompt() method
- prompt displays the first argument as the prompt text
- the second argument is displayed as the default value within the dialog box
- no special programming required

```
<script type="text/javascript" language="JavaScript">
<!-- Hide from other browsers

document.write('<img src="welcome.gif">');
document.write("<h1>Greetings, ");
document.write(prompt("Enter your name:", "Name"));
document.write(". Welcome! </h1>");
// Stop hiding from other browsers -->
</script>
```

- the prompt method doesn't need to be prefixed by document because it is part of the window object
- if the object name is missing, then window object is assumed

alert dialog box

- useful to show some information in a dialog box
- **alert("Click OK to continue.");**
- useful to point out incorrect information in a form or invalid result from a calculation, other messages

```
<script type="text/javascript" language="JavaScript">
<!-- Hide from other browsers

document.write('<IMG SRC="welcome.gif">');
alert("Welcome to my page !");
document.write("<H1>Greetings, ");
document.write(prompt("Enter your name:", "Name"));
document.write(". Welcome to Netscape! </H1>");

// Stop hiding from other browsers -->
</script>
```

JavaScript Data Types

JavaScript provides four basic data types

Numbers as in 21 and 32.62

- integers and floating point numbers

Strings as in "Hello" and 'There'

- strings must be enclosed with a pair of single or double quotation marks

Boolean either true or false
undefined – used to detect if a value is assigned to a variable
null is a special keyword for a nothing value but not empty string or zero

JavaScript – Casting

JavaScript is a loosely typed programming language

- the type of a literal or a variable is not defined when a variable is created and can, at times, change based on context
- you can get away with mixing data of different types

Java and C are strongly typed .
When combining literals of different types, the first type is used
for example "Count to " + 10 becomes "Count to 10"
cannot do 2.5 + "10" and get 12.5 (you get "2.510") BUT "25" – 10 returns 15

use parseInt() and parseFloat() functions
parseInt("12") returns the integer 12
parseFloat("33.23") returns 33.23

JavaScript - Variables

Variables in JavaScript can be of any data type but you must precede with the "var" keyword.

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

<!-- Hide from other browsers

var name=prompt("Enter your name:", "Name");

document.write('<IMG SRC="welcome.gif">');

document.write("<h1> Greetings, " + name +
               ".   Welcome to my page! </h1>");

// Stop hiding from other browsers -->

</script>
```

The following assignment statements are equivalent:

```
var example;
example = "Here is an example";
```

```
var example = "Here is an example";
```

```
document.write( example );
```

variable names are case sensitive and must start with a letter or an underscore

JavaScript – Expressions

Expressions in JavaScript come in four types
- Assignment which assigns a value to a variable
- Arithmetic evaluates to a number

```
var x = 10;
var y = 5;

x += y;  // x is now 15 (10 + 5)
x *= y;  // x is now 75 (15 * 5)
x /= y;  // x is now 15 (75 / 5)
x %= y;  // x is now 0  (15 / 5 leaves 0 remainder )

x = 10;
y = 5;

x++;  // increment operator; x is now 11
y--;  // decrement operator; y is now 4
z = ++y;  // z is 5 and y is now 5
z = --x;  // z is 10 and x is now 10
```

String evaluates to a string

```
A = "Today" + " is Friday";   // A is "Today is Friday"
```

Logical evaluates to a boolean value

logical AND operator is `&&` (the double ampersand – no spaces)
logical OR operator is `||` (the double pipe or vertical bar – no spaces)
logical not operator is ! (the exclamation mark)

comparison operators

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to (note - no spaces)
!=	not equal to

```
x = 10;
y = 5;

( x < y ) && ( x == 5 )  is false
( x > y ) || ( x < 5 )  is true
```

truth tables

A	B	A and B	A or B	!A and !B
True	True	True	True	False
True	False	False	True	False
False	True	False	True	False
False	False	False	False	True

Special operators

=== returns true if both operands are equal and of the same type (e.g. both are strings or both are numbers)
!== returns true if the operands are not equal and/or not of the same type.

JavaScript – Conditional

as in C, C++

```
(condition) ?  value1 : value2;
```

if (condition) evaluates to true, then value1 is returned; otherwise, value2 is returned
can be cryptic, so don't overuse

```
var response = (answer ==”Y”) ? “Yes” : “No”;
```

JavaScript – String

concatenation operator is +
“Welcome to ” + “my site!” becomes “Welcome to my site!”
welcome += “ Thank-you.” adds the string “Thank-you.” to the end of the string variable called welcome

JavaScript - IF construct

Syntax:
```
if (condition is true) {
        Javascript commands if true
} else {
        Javascript commands if false
}
```

```
if ( day == “Saturday”) {
  document.writeln(“It’s the weekend!”);
} else {
  document.writeln(“It’s not Saturday.”);
}
```

JavaScript - Confirm

confirm method allows the user to select an OK button or a Cancel button
confirm returns true if OK clicked, false if Cancel clicked

```
if (confirm(“Press OK to retry.”))
  response = prompt(“What is 2+2 ?”, “3”);
```

The “3” will appear in the input prompt box as the default.

JavaScript - Sample

mathtest.html demonstrates the confirm method in action

```
<script type="text/javascript" language="JavaScript">
   <!-- hide from older browsers
   // define some variables
   var question = "What is 10+10?";
   var answer = 20;
   var correct =  '<img src="correct.gif">';
```

```
        var incorrect ='<img src="incorrect.gif">';
        // ask the question
        var response = prompt(question,"0");

        // check the answer
        if (response != answer) {

        // wrong answer; retry once

       if (confirm("Wrong! Press OK for a second chance."))

       response = prompt(question,"0");
        }
     // check the answer
       var output = (response == answer) ? correct : incorrect;

// stop hiding from older browsers -->
</script>
</head>

<body>
<script language="JavaScript">
<!-- hide from older browsers

document.write(output);

// stop hiding from older browsers -->
</script>
```

JavaScript - Functions

defined using the function statement

function function_name(parameters, arguments) { command block
}

e.g.

```
function printName( name ) {
      document.write("<hr>Your Name is <b><i>");
      document.write( name );
      document.write("</b></i><hr>");
}
```

functions should be defined in the header
this ensures that all functions have been parsed before it is possible for user events to invoke a function
eval() method
evaluates a string to a numeric
e.g. eval("4 + 5") returns a value of 9

HTML file testing.html

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

   <!-- hide from older browsers
```

```
   // define function testQuestion()
   function testQuestion(question) {
      // define some local variables

      var answer = eval(question);
      var output = "What is " + question + "?";
      var correct = '<img src="correct.gif">';
      var incorrect = '<img src="incorrect.gif">';
      // ask the question
      var response=prompt(output,"0");
            // check the result
     return (response == answer) ? correct : incorrect;
   }
   // stop hiding from older browsers -->
   </script>
   </head>
   <body>
   <script type="text/javascript" language="JavaScript">
   <!-- hide from older browsers
   var result = testQuestion("4 + 5");
   document.write(result);
 // stop hiding from older browsers -->
   </script>
```

JavaScript - Dates

variable = new Date(parameters)

if no parameters, then the current date is used

```
var today = new Date();  // the date right now
var birthday = new Date( 1962, 8, 24);
var party = new Date( 96, 3, 23, 8, 0, 0);
var the_date = new Date("October, 15, 1998");
```

a Date object, like other objects, has methods associated with it

getDate()	returns the day of the month (e.g. 30)
getDay()	returns the day of the week (e.g. 0=Sunday, 1=Monday,…)
getHours()	returns the hours (e.g. 12)
getMinutes()	returns the minutes (e.g. 47)
getMonth()	returns the month (e.g. 0=January, 1=February,…)
getSeconds()	returns the seconds (e.g. 54)
getTime()	returns the complete time elapsed in milliseconds since January 1, 1970 (e.g. 1,081,445,128,000)
getFullYear()	returns the four digit year (e.g. 2005)

Methods must be applied to objects using the format of **`object_name.method_name()`** as in

```
var today = new Date();
var Day = Today.getDate();
```

example:

```
<head>
```

```
<title>Day Test using Javascript</title>

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

<!--Hide from non-JavaScript browsers

// This function calculates the number of days until Christmas
function XmasDays(Month, Day, Year) {

      var DayCount = (25 - Day) + 30;

      return DayCount;
}

// Stop hiding -->

</script>

</head>

<body>
<script type="text/javascript" language="JavaScript">
<!--Hide from non-JavaScript browsers
// Get date information
var Today = new Date("November 1, 2005");
var ThisDay = Today.getDate();
var ThisMonth = Today.getMonth() + 1;
var ThisYear = Today.getYear();
document.write("Today is "+ThisMonth+"/"+ThisDay+"/"+
                  ThisYear+"< /br>");
// Get number of days until Christmas
var DaysLeft = XmasDays(ThisMonth, ThisDay, ThisYear);
document.write("Only " + DaysLeft + " days until Christmas");
// Stop hiding -->
</script>
```

Arrays in JavaScript

An **array** is an ordered collection of values referenced by a single variable name.

```
Weekday[1]= "Monday";
Weekday[2]= "Tuesday";
Weekday[3]= "Wednesday";
Weekday[4]= "Thursday";
Weekday[5]= "Friday";
Weekday[6]= "Saturday";
Weekday[7]= "Sunday";
```

Each element is identified by an **index**, the number appearing in the brackets. For example, the element "Monday" has an index of 1. In the Weekday array there are actually eight elements in total. The first element is Weekday[0], and it has the null value.

Syntax for creating an array variable is

var variable = new Array(size);

specifying the size is optional. Javascript will automatically increase the size of the array as you add more elements.

```
var Month = new Array();
Month[12] = "December";
```

Javascript will create 13 elements for the Month array all having null values except for the Month[12] element.

Looping

A **loop** is a set of instructions that is run repeatedly. There are two kinds of loops: a loop that repeats a set number of times, and a loop that will continually repeat until a condition is met.

The **For loop** is the first kind

Syntax: for (start; stop; update) {
Javascript commands
}

** Note the locations of the braces { } on the lines can be anywhere but usually the closing brace } is placed so that it lines up vertically with its matching for keyword.

```
for (Num=1; Num<4; Num++) {
      document.write("The value of Num is " + Num + "<BR>");
      // this will repeat exactly three times
}

for (i=10; i>0; i--) // this loop will repeat 10 times

for (j=0; j<=36; j+=5) // this loop will repeat 8 times
```

The **While loop** is the second kind

Syntax: while (condition is true) {
Javascript commands
}

```
var Num = 1;
while (Num < 4) {
      document.write("The value of Num is " + Num);
      Num++;
}     // repeats three times
```

JavaScript Resources

http://devedge-temp.mozilla.org/library/manuals/2000/javascript/1.5/guide/