1

CSC 551: Web Programming

Spring 2004

client-side programming with JavaScript

- scripts vs. programs
  - JavaScript vs. JScript vs. VBScript
  - common tasks for client-side scripts
- JavaScript
  - data types & expressions
  - control statements
  - functions & libraries
  - strings & arrays
  - Date, document, navigator, user-defined classes

2

Client-side programming

- recall: HTML is good for developing *static* pages
  - can specify text/image layout, presentation, links, …
  - Web page looks the same each time it is accessed
  - in order to develop interactive/reactive pages, must integrate programming

client-side programming

- programs are written in a separate programming language

  e.g., JavaScript, JScript, VBScript
- programs are embedded in the HTML of a Web page, with tags to identify the program component

  e.g., <script type="text/javascript"> … </script>
- the browser executes the program as it loads the page, integrating the dynamic output of the program with the static content of HTML

3

Scripts vs. programs

- a scripting language is a simple, interpreted programming language
  - scripts are embedded as plain text, interpreted by application
  - *simpler execution model:* don't need compiler or development environment
  - *saves bandwidth:* source code is downloaded, not compiled executable
  - *platform-independence:* code interpreted by any script-enabled browser
  - *but:* slower than compiled code, not as powerful/full-featured

JavaScript: the first Web scripting language, developed by Netscape in 1995
syntactic similarities to Java/C++, but simpler & more flexible
(loose typing, dynamic variables, simple objects)

JScript: Microsoft version of JavaScript, introduced in 1996
same core language, but some browser-specific differences
fortunately, IE & Netscape can (mostly) handle both JavaScript & JScript

JavaScript 1.5 & JScript 5.0 cores conform to ECMAScript standard

VBScript: client-side scripting version of Microsoft Visual Basic

4

Common scripting tasks

- adding dynamic features to Web pages
  - validation of form data
  - image rollovers
  - time-sensitive or random page elements
  - handling cookies
- defining programs with Web interfaces
  - utilize buttons, text boxes, clickable images, prompts, frames

limitations of client-side scripting

- since script code is embedded in the page, viewable to the world
- for security reasons, scripts are limited in what they can do
  e.g., can't access the client's hard drive
- since designed to run on any machine platform, scripts do not contain platform specific commands
- script languages are not full-featured
  e.g., JavaScript objects are crude, not good for large project development

5

JavaScript

- JavaScript code can be embedded in a Web page using SCRIPT tags
  - the output of JavaScript code is displayed as if directly entered in HTML

```
<html>
<!-- Dave Reed  js01.html  2/01/04 -->

<head>
  <title>JavaScript Page</title>
</head>

<body>
  <script type="text/javascript">
    // silly code to demonstrate output

    document.write("Hello world!");

    document.write("<p>How are <br />" +
                   "<i>you</i>?</p>");
  </script>

  <p>Here is some static text as well.
  </p>
</body>
</html>
```

view page in browser

`document.write` displays text in page

text to be displayed can include HTML tags

the tags are interpreted by the browser when the text is displayed

as in C++/Java, statements end with `;`

JavaScript comments similar to C++/Java

`//` starts a single line comment

`/*...*/` enclose multi-line comments

6

JavaScript data types & variables

- JavaScript has only three primitive data types
  - *strings* : `"Howdy Pal"` `"I said 'hi'."` `""`
  - *numbers* : `12` `3.14159` `1.5E6`
  - *booleans* : `true` `false`

```
<html>
<!-- Dave Reed  js02.html  2/01/04 -->

<head>
  <title>Data Types and Variables</title>
</head>

<body>
  <script type="text/javascript">
    x = 1024;
    document.write("<p>x = " + x + "</p>");

    x = "foobar";
    document.write("<p>x = " + x + "</p>");
  </script>
</body>
</html>
```

view page in browser

assignments are as in C++/Java

```
message = "howdy";
pi = 3.14159;
```

variable names are sequences of letters, digits, and underscores: *start with a letter*

variables names are case-sensitive

you don't have to declare variables, will be created the first time used

variables are loosely typed, can assign different types of values

7

JavaScript operators & control statements

```
<html>
<!-- Dave Reed  js03.html  2/01/04 -->

<head>
  <title>Folding Puzzle</title>
</head>

<body>
 <script type="text/javascript">
    distanceToSun = 93.3e6*5280*12;
    thickness = .002;

    foldCount = 0;
    while (thickness < distanceToSun) {
        thickness *= 2;
        foldCount++;
    }
    document.write("Number of folds = " +
                   foldCount);
 </script>
</body>
</html>
```

view page in browser

standard C++/Java operators & control statements are provided in JavaScript

- `+, -, *, /, %, ++, --, …`
- `==, !=, <, >, <=, >=`
- `&&, ||, !`
- if, if-else, while, do, …

PUZZLE: Suppose you took a piece of paper and folded it in half, then in half again, and so on.

How many folds before the thickness of the paper reaches from the earth to the sun?

8

JavaScript Math routines

```
<html>
<!-- Dave Reed  js04.html  2/01/04 -->

<head>
  <title>Random Dice Rolls</title>
</head>

<body>
  <div style="text-align:center">
    <script type="text/javascript">
      roll1 = Math.floor(Math.random()*6) + 1;
      roll2 = Math.floor(Math.random()*6) + 1;

      document.write("<img src='http://www.creighton.edu/"+
                     "~davereed/csc551/Images/die" +
                     roll1 + ".gif' />");
      document.write("  ");
      document.write("<img src='http://www.creighton.edu/"+
                     "~davereed/csc551/Images/die" +
                     roll2 + ".gif' />");
    </script>
  </div>
</body>
</html>
```

view page in browser

the Math object contains functions and constants

```
Math.sqrt
Math.pow
Math.abs
Math.max
Math.min
Math.floor
Math.ceil
Math.round

Math.PI
Math.E
```

`Math.random` function returns number in [0..1)

9

Interactive pages using prompt

```
<html>
<!-- Dave Reed  js05.html  2/01/04 -->

<head>
  <title>Interactive page</title>
</head>

<body>
 <script type="text/javascript">
    userName = prompt("What is your name?", "");

    userAge = prompt("Your age?", "");
    userAge = parseFloat(userAge);

    document.write("Hello " + userName + ".")
    if (userAge < 18) {
      document.write("  Do your parents know " +
                     "you are online?");
    }
 </script>

  <p>The rest of the page...
</body>
</html>
```

view page in browser

crude user interaction can take place using prompt

1st argument: the prompt message that appears in the dialog box

2nd argument: a default value that will appear in the box (in case the user enters nothing)

the function returns the value entered by the user in the dialog box (a string)

if value is a number, must use parseFloat to convert

forms will provide a better interface for interaction *(later)*

10

User-defined functions

- function definitions are similar to C++/Java, except:
  - no return type for the function (since variables are loosely typed)
  - no types for parameters (since variables are loosely typed)
  - by-value parameter passing only (parameter gets copy of argument)

```
function isPrime(n)
// Assumes: n > 0
// Returns: true if n is prime, else false
{
  if (n < 2) {
    return false;
  }
  else if (n == 2) {
    return true;
  }
  else {
    for (var i = 2; i <= Math.sqrt(n); i++) {
      if (n % i == 0) {
        return false;
      }
    }
    return true;
  }
}
```

can limit variable scope

if the first use of a variable is preceded with `var`, then that variable is local to the function

for modularity, should make all variables in a function local

Function example

11

```
<html>
<!-- Dave Reed  js06.html  2/01/04 -->

<head>
  <title>Prime Tester</title>

  <script type="text/javascript">
    function isPrime(n)
    // Assumes: n > 0
    // Returns: true if n is prime
    {
      // CODE AS SHOWN ON PREVIOUS SLIDE
    }
  </script>
</head>

<body>
  <script type="text/javascript">
    testNum = parseFloat(prompt("Enter a positive integer", "7"));

    if (isPrime(testNum)) {
      document.write(testNum + " <b>is</b> a prime number.");
    }
    else {
      document.write(testNum + " <b>is not</b> a prime number.");
    }
  </script>
</body>
</html>
```

function definitions go in the HEAD

HEAD is loaded first, so the function is defined before code in the BODY is executed

view page in browser

12

```
<html>
<!-- Dave Reed  js07.html  2/01/04 -->

<head>
  <title> Random Dice Rolls Revisited</title>

  <script type="text/javascript">
    function RandomInt(low, high)
    // Assumes: low <= high
    // Returns: random integer in range [low..high]
    {
      return Math.floor(Math.random()*(high-low+1)) + low;
    }
  </script>
</head>

<body>
  <div align="center">
    <script type="text/javascript">
      roll1 = RandomInt(1, 6);
      roll2 = RandomInt(1, 6);

      document.write("<img src='http://www.creighton.edu/"+
                     "~davereed/csc551/Images/die" +
                     roll1 + ".gif' />");
      document.write("  ");
      document.write("<img src='http://www.creighton.edu/"+
                     "~davereed/csc551/Images/die" +
                     roll2 + ".gif' />");
    </script>
  </div>
</body>
</html>
```

Another example

recall the dynamic dice page

could define a function for generating random numbers in a range, then use whenever needed

easier to remember, promotes reuse

view page in browser

13

JavaScript libraries

- better still: if you define functions that may be useful to many pages, store in a separate library file and load the library when needed

- the file at
  `http://www.creighton.edu/~davereed/csc551/JavaScript/random.js`
  contains definitions of the following functions:

`RandomNum(low, high)`	returns random real in range [low..high)
`RandomInt(low, high)`	returns random integer in range [low..high)
`RandomChar(string)`	returns random character from the string
`RandomOneOf([item1,…,itemN])`	returns random item from list/array

Note: as with external style sheets, no tags in the JavaScript library file

load a library using the SRC attribute in the SCRIPT tag (nothing between the tags)

```
<script type="text/javascript"
        src="http://www.creighton.edu/~davereed/csc551/JavaScript/random.js">
</script>
```

Library example

14

```
<html>
<!-- Dave Reed  js08.html  2/01/04 -->

<head>
  <title> Random Dice Rolls Revisited</title>

  <script type="text/javascript"
    src="http://www.creighton.edu/~davereed/csc551/JavaScript/random.js">
  </script>
</head>

<body>
  <div align="center">
    <script type="text/javascript">
      roll1 = RandomInt(1, 6);
      roll2 = RandomInt(1, 6);

      document.write("<img src='http://www.creighton.edu/"+
                     "~davereed/csc551/Images/die" +
                     roll1 + ".gif' />");
      document.write("  ");
      document.write("<img src='http://www.creighton.edu/"+
                     "~davereed/csc551/Images/die" +
                     roll2 + ".gif' />");
    </script>
  </div>
</body>
</html>
```

view page in browser

15

JavaScript Strings

- a *class* defines a new type (formally, *Abstract Data Type*)
  - encapsulates data (properties) and operations on that data (methods)
- a String encapsulates a sequence of characters, enclosed in quotes

properties include

- `length` : stores the number of characters in the string

methods include

- `charAt(index)` : returns the character stored at the given index (as in C++/Java, indices start at 0)
- `substring(start, end)` : returns the part of the string between the start (inclusive) and end (exclusive) indices
- `toUpperCase()` : returns copy of string with letters uppercase
- `toLowerCase()` : returns copy of string with letters lowercase

to create a string, assign using `new` or just make a direct assignment (`new` is implicit)

```
word = new String("foo");          word = "foo";
```

properties/methods are called exactly as in C++/Java

- `word.length` `word.charAt(0)`

16

String example: palindromes

```
function Strip(str)
// Assumes: str is a string
// Returns: str with all but letters removed
{
  var copy = "";
  for (var i = 0; i < str.length; i++) {
    if ((str.charAt(i) >= "A" && str.charAt(i) <= "Z") ||
        (str.charAt(i) >= "a" && str.charAt(i) <= "z")) {
      copy += str.charAt(i);
    }
  }
  return copy;
}

function IsPalindrome(str)
// Assumes: str is a string
// Returns: true if str is a palindrome, else false
{
  str = Strip(str.toUpperCase());

  for(var i = 0; i < Math.floor(str.length/2); i++) {
    if (str.charAt(i) != str.charAt(str.length-i-1)) {
      return false;
    }
  }
  return true;
}
```

suppose we want to test whether a word or phrase is a palindrome

noon *Radar*
Madam, I'm Adam.
A man, a plan, a canal: Panama!

must strip non-letters out of the word or phrase

make all chars uppercasein order to be case-insensitive

finally, traverse and compare chars from each end

17

```
<html>
<!-- Dave Reed  js09.html  2/01/04 -->

<head>
 <title>Palindrome Checker</title>

  <script type="text/javascript">
    function Strip(str)
    {
       // CODE AS SHOWN ON PREVIOUS SLIDE
    }

    function IsPalindrome(str)
    {
      // CODE AS SHOWN ON PREVIOUS SLIDE
    }
  </script>
</head>

<body>
  <script type="text/javascript">
    text = prompt("Enter a word or phrase", "Madam, I'm Adam");

    if (IsPalindrome(text)) {
      document.write("'" + text + "' <b>is</b> a palindrome.");
    }
    else {
      document.write("'" + text + "' <b>is not</b> a palindrome.");
    }
  </script>
</body>
</html>
```

view page in browser

18

JavaScript arrays

- arrays store a sequence of items, accessible via an index
  since JavaScript is loosely typed, elements do not have to be the same type
  - to create an array, allocate space using `new` (or can assign directly)

```
items = new Array(10);      // allocates space for 10 items

items = new Array();   // if no size, will adjust dynamically

items = [0,0,0,0,0,0,0,0,0,0]; // can assign size & values []
```

  - to access an array element, use `[]` (as in C++/Java)

```
for (i = 0; i < 10; i++) {
    items[i] = 0;          // stores 0 at each index

}
```

  - the `length` property stores the number of items in the array

```
for (i = 0; i < items.length; i++) {
    document.write(items[i] + "<br>");   // displays elements
}
```

19

Array example

```
<html>
<!-- Dave Reed  js10.html  2/01/04 -->

<head>
 <title>Die Statistics</title>

 <script type="text/javascript"
   src="http://www.creighton.edu/~davereed/csc551/JavaScript/random.js">
 </script>
</head>

<body>
 <script type="text/javascript">
    numRolls = 60000;
    dieSides = 6;

    rolls = new Array(dieSides+1);
    for (i = 1; i < rolls.length; i++) {
        rolls[i] = 0;
    }

    for(i = 1; i <= numRolls; i++) {
        rolls[RandomInt(1, dieSides)]++;
    }

    for (i = 1; i < rolls.length; i++) {
        document.write("Number of " + i + "'s = " +
                       rolls[i] + "<br />");
    }
 </script>
</body>
</html>
```

suppose we want to simulate die rolls and verify even distribution

keep an array of counters:

- initialize each count to 0
- each time you roll `X`, increment `rolls[X]`
- display each counter

view page in browser

20

Date class

- String & Array are the most commonly used classes in JavaScript
  - other, special purpose classes & objects also exist
- the Date class can be used to access the date and time
  - to create a Date object, use new & supply year/month/day/... as desired
    - `today = new Date();  // sets to current date & time`
    - `newYear = new Date(2002,0,1); //sets to Jan 1, 2002  12:00AM`
  - methods include:
    - `newYear.getYear()` — can access individual components of a date
    - `newYear.getMonth()`
    - `newYear.getDay()`
    - `newYear.getHours()`
    - `newYear.getMinutes()`
    - `newYear.getSeconds()`
    - `newYear.getMilliseconds()`

21

```
<html>
<!-- Dave Reed  js11.html  2/01/04 -->
<head>
  <title>Time page</title>
</head>

<body>
  Time when page was loaded:
  <script type="text/javascript">
    now = new Date();

    document.write("<p>" + now + "</p>");

    time = "AM";
    hours = now.getHours();
    if (hours > 12) {
        hours -= 12;
        time = "PM"
    }
    else if (hours == 0) {
        hours = 12;
    }
    document.write("<p>" + hours + ":" +
                   now.getMinutes() + ":" +
                   now.getSeconds() + " " +
                   time + "</p>");
  </script>
</body>
</html>
```

Date example

by default, a date will be displayed in full, e.g.,

```
Sun Feb 03 22:55:20 GMT-0600
(Central Standard Time) 2002
```

can pull out portions of the date using the methods and display as desired

here, determine if "AM" or "PM" and adjust so hour between 1-12

```
10:55:20 PM
```

view page in browser

22

```
<html>
<!-- Dave Reed  js12.html  2/01/04 -->
<head>
  <title>Time page</title>
</head>

<body>
  This year:
  <script type="text/javascript">
    now = new Date();
    newYear = new Date(2004,0,1);

    secs = Math.round((now-newYear)/1000);

    days = Math.floor(secs / 86400);
    secs -= days*86400;
    hours = Math.floor(secs / 3600);
    secs -= hours*3600;
    minutes = Math.floor(secs / 60);
    secs -= minutes*60

    document.write(days + " days, " +
                   hours + " hours, " +
                   minutes + " minutes, and " +
                   secs + " seconds.");
  </script>
</body>
</html>
```

Another example

you can add and subtract Dates: the result is a number of milliseconds

here, determine the number of seconds since New Year's day

divide into number of days, hours, minutes and seconds

possible improvements?

view page in browser

23

document object

Both IE and Netscape allow you to access information about an HTML document using the `document` object *(Note: not a class!)*

```
<html>
<!-- Dave Reed  js13.html  2/01/04 -->

<head>
  <title>Documentation page</title>
</head>

<body>
  <table width="100%">
    <tr>
      <td><small><i>
        <script type="text/javascript">
            document.write(document.URL);
        </script>
      </i></small></td>
      <td align="right"><small><I>
        <script type="text/javascript">
            document.write(document.lastModified);
        </script>
      </i></small></td>
    </tr>
  </table>
</body>
</html>
```

`document.write(…)`
method that displays text in the page

`document.URL`
property that gives the location of the HTML document

`document.lastModified`
property that gives the date & time the HTML document was saved

view page in browser

24

navigator object

`navigator.appName`
property that gives the browser name

`navigator.appVersion`
property that gives the browser version

```
<!-- MSIE.css  -->

a {text-decoration:none;
   font-size:larger;
   color:red;
   font-family:Arial}
a:hover {color:blue}
```

```
<!-- Netscape.css  -->

a {font-family:Arial;
   color:white;
   background-color:red}
```

```
<html>
<!-- Dave Reed  js14.html  2/01/04 -->

<head>
  <title>Dynamic Style Page</title>

  <script type="text/javascript">
    if (navigator.appName == "Netscape") {
      document.write('<link rel=stylesheet '+
        'type="text/css" href="Netscape.css">');
    }
    else {
      document.write('<link rel=stylesheet ' +
        'type="text/css" href="MSIE.css">');
    }
  </script>
</head>

<body>
Here is some text with a
<a href="javascript:alert('GO AWAY')">link</a>.
</body>
</html>
```

view page in browser

25

User-defined classes

- can define new classes, but the notation is awkward
  - simply define a function that serves as a constructor
  - specify data fields & methods using `this`
  - no data hiding: can't protect data or methods

```
// Dave Reed       Die.js       2/01/04
//
// Die class definition
////////////////////////////////////////

function Die(sides)
{
   this.numSides = sides;
   this.numRolls = 0;
   this.Roll = Roll;
}

function Roll()
{
   this.numRolls++;
   return Math.floor(Math.random()*this.numSides) + 1;
}
```

define `Die` function (i.e., constructor)

initialize data fields in the function, preceded with this

similarly, assign method to separately defined function (which uses this to access data)

26

Class example

```
<html>
<!-- Dave Reed  js15.html  2/01/04 -->

<head>
  <title>Dice page</title>

  <script type="text/javascript"
          src="Die.js">
  </script>
</head>

<body>
 <script type="text/javascript">
    die6 = new Die(6);
    die8 = new Die(8);

    roll6 = -1;     // dummy value to start loop
    roll8 = -2;     // dummy value to start loop
    while (roll6 != roll8) {
      roll6 = die6.Roll();
      roll8 = die8.Roll();

      document.write("6-sided: " + roll6 +
                     "    " +
                     "8-sided: " + roll8 + "<br />");
    }

    document.write("<br />Number of rolls: " +
                   die6.numRolls);
 </script>
</body>
</html>
```

create a Die object using new (similar to String and Array)

here, the argument to Die initializes numSides for that particular object

each Die object has its own properties (numSides & numRolls)

Roll(), when called on a particular Die, accesses its numSides property and updates its NumRolls

view page in browser