

CSC 533: Organization of Programming Languages

Spring 2007

Java vs. JavaScript

- JavaScript design goals
- examples
 - input/output, functions, numbers & expressions, control statements
 - dynamic images, buttons, text boxes, text areas
 - strings, arrays
 - special purpose functions, timeouts
- Java vs. JavaScript

1

Java vs. JavaScript

recall: Java took many features from C++, but removed/added features due to different design goals

- e.g., platform independence → interpreted+compiled execution model
- ease of development over efficiency → dynamic binding, garbage collection
- simplicity over expressivity → no goto, no implicit coercions, no operator overload
- security → no pointer access, byte code verifier

interesting to consider a third variant: JavaScript

- designed to be a scripting language for execution in and enhancements of a Web browser
- developed at Netscape in 1995, integrated into Navigator
 - close variant added to IE under the name JScript
 - the core of both languages were standardized as ECMAScript
- as with Java, chose to keep basic syntax of C++ to aid learning
- different design goals yield different features

2

JavaScript design

intended to be a scripting language for Web pages

- JavaScript code is embedded directly into HTML, interpreted by browser
- a.k.a., client-side scripting

scripting applications are more quick-and-dirty, relatively small

- variables are bound to type and address dynamically for flexibility
- do not need to declare variables, functions are not typed either
- code size is limited by the browser

not expected to develop large applications

- object-based: lots of useful classes predefined (Array, String, Math, ...) can define new classes but awkward, no info hiding

user security is important, script code security isn't

- like Java, JavaScript code can't access local files
- no way to hide the JavaScript source when download Web page

3

JavaScript input/output

```
<html>
<!-- oldmac.html                      Dave Reed -->
<!-- Web page that displays a verse of Old MacDonald. -->
<!------->

<head>
<title> Old MacDonald </title>
</head>

<body>
<h3 style="text-align:center">Old MacDonald Had a Farm</h3>

<script type="text/javascript">
  animal = prompt("Enter a kind of animal:", "cow");
  sound = prompt("What kind of sound does it make?", "moo");

  document.write("<p>Old MacDonald had a farm, E-I-E-I-O.<br />");
  document.write("And on that farm he had a " + animal +
    ", E-I-E-I-O.<br />");
  document.write("With a " + sound + "-" + sound + " here, and a " +
    sound + "-" + sound + " there,<br />");
  document.write("  here a " + sound + ", there a " + sound +
    ", everywhere a " + sound + "-" + sound +
    ".<br />");
  document.write("Old MacDonald had a farm, E-I-E-I-O.</p>");
</script>
</body>
</html>
```

JavaScript code can be embedded in the page using SCRIPT tags

code is executed by the browser, output is inserted into the page

can prompt the user for input via the prompt function

document.write is JavaScript's output routine

4

JavaScript functions

```
<html>
<!-- newmac.html                                Dave Reed -->
<!-- This page uses a function to display Old MacDonald verses. -->
<!------->

<head>
<title> New Improved Old MacDonald </title>
<script type="text/javascript">
function OldMacVerse(animal, sound)
// Assumes: animal and sound are strings
// Results: displays corresponding Old MacDonald verse
{
  document.write("<p>Old MacDonald had a farm, E-I-E-I-O.<br />");
  document.write("And on that farm he had a " + animal +
    ", E-I-E-I-O.<br />");
  document.write("With a " + sound + "-" + sound +
    " here, and a " + sound + "-" + sound +
    " there,<br />");
  document.write("&nbsp;&nbsp;&nbsp; here a " + sound + ", there a " +
    sound + ", everywhere a " + sound + "-" +
    sound + "<br />");
  document.write("Old MacDonald had a farm, E-I-E-I-O.</p>");
}
</script>
</head>

<body>
<script type="text/javascript">
OldMacVerse("cow", "moo");
OldMacVerse("pig", "oink");
OldMacVerse("duck", "quack");
</script>
</body>
</html>
```

variables are not explicitly typed

neither are functions

similar to Java, comments can be written using //

by convention, functions are defined in the HEAD, executable statements in the BODY

5

JavaScript numbers & expressions

```
<html>
<!-- convert.html                                Dave Reed -->
<!-- This page converts a temperature from Fahrenheit to Celsius. -->
<!------->

<head>
<title> New Improved Temperature Conversion </title>
<script type="text/javascript">
function FahrToCelsius(tempInFahr)
// Assumes: tempInFahr is a temperature in Fahrenheit
// Returns: the equivalent temperature in Celsius
{
  return (5/9) * (tempInFahr - 32);
}
</script>
</head>

<body>
<script type="text/javascript">
tempF = prompt("Enter the temperature (in Fahrenheit):", "32");
tempF = parseFloat(tempF);

tempC = FahrToCelsius(tempF);

document.write("You entered " + tempF +
  " degrees Fahrenheit.<br />");
document.write("That's equivalent to " + tempC +
  " degrees Celsius.");
</script>
</body>
</html>
```

there is only a single number type (that represents both integers and reals)

Java-like numeric operations are provided (+, -, *, /, %)

a function can include a return statement to return a value

the `parseFloat` function converts a String into a number

6

JavaScript control statements

```
<html>
<!-- convert.html                                Dave Reed -->
<!-- Silly page that demos JavaScript control statements. -->
<!------->

<head>
  <title> Dave's Hello World Page </title>

  <script type="text/javascript">
    function SayHello(name)
    {
      if (name == "Dave") {
        document.write("It's an honor, Dave!");
      }
      else {
        for (var i = 0; i < 10; i++) {
          document.write("Hello <i>" + name +
            "</i>, glad to meet you! <br />");
        }
      }
    }
  </script>
</head>

<body>
  <script type="text/javascript">
    userName = prompt("Enter your name:", "Your name");
    SayHello(userName);
  </script>
</body>
</html>
```

control structures are similar to Java

- if, switch
- while, do, for

by default, a variable's scope is the entire page

you can restrict a variable to be local to a function by preceding its first use with "var"

7

JavaScript objects

many useful objects w/ methods are predefined

e.g., Math.abs
document.write

can define new classes
can even utilize inheritance

- very awkward syntax
- no info hiding

can add data/methods dynamically

- die1.owner = "Dave"

also, can put useful code in a separate file, load using SCRIPT tag with SRC attribute

```
// Die.js
function Die(sides) {
  this.numSides = sides;
  this.numRolls = 0;
  this.Roll =
    function () {
      this.numRolls++;
      return Math.floor(Math.random()*this.numSides)+1;
    }
}

function ColoredDie(sides, color) {
  this.inheritFrom = Die;
  this.inheritFrom(sides);
  this.dieColor = color;
}
```

```
<html>
<head>
  <title> Roll two dice</title>
  <script type="text/javascript" src="Die.js"></script>
</head>
<body>
  <script type="text/javascript">
    die1 = new Die(6);
    die2 = new ColoredDie(6, "blue");

    roll1 = die1.Roll();
    roll2 = die2.Roll();

    document.write("I rolled " + roll1 + " and " +
      roll2 + " ==> " + (roll1+roll2) + "<br />");
  </script>
</body>
</html>
```

8

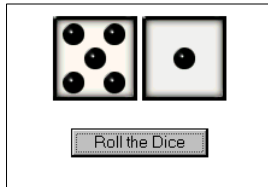
JavaScript event-handling: buttons & images

main uses of JavaScript

- processing form elements (e.g., text box)
- reacting to user-controlled events (e.g., button click)

the `onclick` attribute of a button specifies JavaScript code

can change image source using the `src` attribute



```
<html>
<head>
  <title> Dice Stats </title>
  <script type="text/javascript" src="Die.js"></script>
  <script type="text/javascript">
    function DoIt()
    {
      var die1 = new Die(6);
      var die2 = new Die(6);

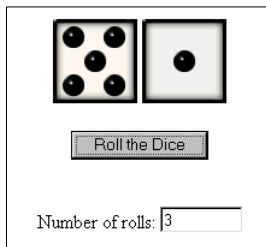
      document.getElementById('die1').src =
        "http://www.dave-reed.com/book/Images/die" +
        die1.Roll() + ".gif";
      document.getElementById('die2').src =
        "http://www.dave-reed.com/book/Images/die" +
        die2.Roll() + ".gif";
    }
  </script>
</head>
<body>
  <div style="text-align:center">
    
    
    <p>
      <input type="button" value="Roll the Dice"
        onclick="DoIt();" />
    </p>
  </div>
</body>
</html>
```

9

JavaScript event-handling: text boxes

a text box can be used for user input, or to display output

access contents using `value` attribute of the element



```
<html>
<head>
  <title> Dice Stats </title>
  <script type="text/javascript" src="Die.js"></script>
  <script type="text/javascript">
    function DoIt()
    {
      var die1 = new Die(6);
      var die2 = new Die(6);

      document.getElementById('die1').src =
        "http://www.dave-reed.com/book/Images/die" +
        die1.Roll() + ".gif";
      document.getElementById('die2').src =
        "http://www.dave-reed.com/book/Images/die" +
        die2.Roll() + ".gif";

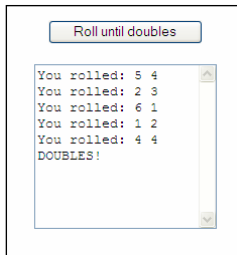
      document.getElementById('numRolls').value++;
    }
  </script>
</head>
<body>
  <div style="text-align:center">
    
    
    <p>
      <input type="button" value="Roll the Dice" onclick="DoIt();" />
    </p>
    <p>
      Number of rolls:
      <input type="text" id="numRolls" size=8 value=0>
    </p>
  </div>
</body>
</html>
```

10

JavaScript event-handling: text areas

a textarea is similar to a text box, but can span multiple lines

access contents using value attribute of the element



```
<html>
<head>
  <title> Dice Stats </title>
  <script type="text/javascript" src="Die.js"></script>
  <script type="text/javascript">
    function RollUntilDoubles()
    {
      var d1 = new Die(6);
      var d2 = new Die(6);

      var roll1 = d1.Roll();
      var roll2 = d2.Roll();
      document.getElementById('Output').value =
        "You rolled: " + roll1 + " " + roll2 + "\n";

      while (roll1 != roll2) {
        roll1 = d1.Roll();
        roll2 = d2.Roll();
        document.getElementById('Output').value +=
          "You rolled: " + roll1 + " " + roll2 + "\n";
      }
      document.getElementById('Output').value += "DOUBLES!";
    }
  </script>
</head>
<body>
  <input type="button" value="Roll until doubles"
    onclick="RollUntilDoubles();" />
  <br /><br />
  <textarea name="Output" rows=20 cols=20></textarea>
</body>
</html>
```

11

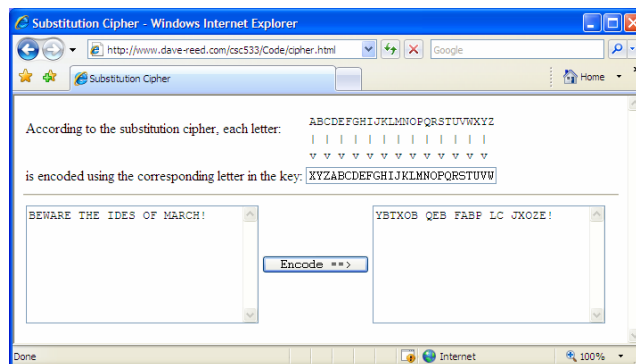
JavaScript strings

JavaScript has Java-like strings, with all of the corresponding methods

- length
- charAt
- substring
- ...

Cipher.html performs a simple substitution cipher

- uses text box for the code key; text areas for the message & its encoding
- substitutes letter by letter



12

JavaScript arrays

JavaScript has flexible arrays – non-typed so can store any sequence of values

Pirate Translator

- stores English-Pirate dictionary as an array of arrays
- utilizes regular expressions to search for words in context



13

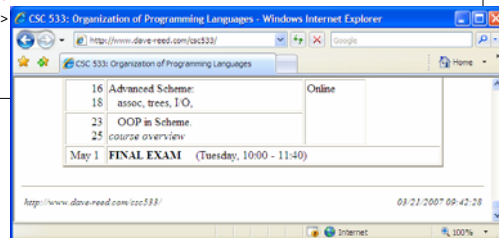
JavaScript special-purpose functions

JavaScript has many predefined functions for interacting with the HTML document

- `document.location` gives the URL of the current page
- `document.lastModified` gives the date when the page was last changed

can write JavaScript code to store and access cookies on the client computer

```
...
<p><hr />
<table border=0 width="100%">
<tr>
<td style="text-align:left">
<script type="text/javascript">
document.write("<font size=-1><i>"+
document.location+"</i></font>");
</script>
</td>
<td style="text-align:right">
<script type="text/javascript">
document.write("<font size=-1><i>"+
document.lastModified+"</i></font>");
</script>
</td></tr>
</table>
</body>
</html>
```



14

JavaScript form verification

when submitting online forms, JavaScript is commonly used to verify the contents

NCAA contest

- when the SUBMIT button is clicked, JavaScript code checks to make sure that all boxes are filled in



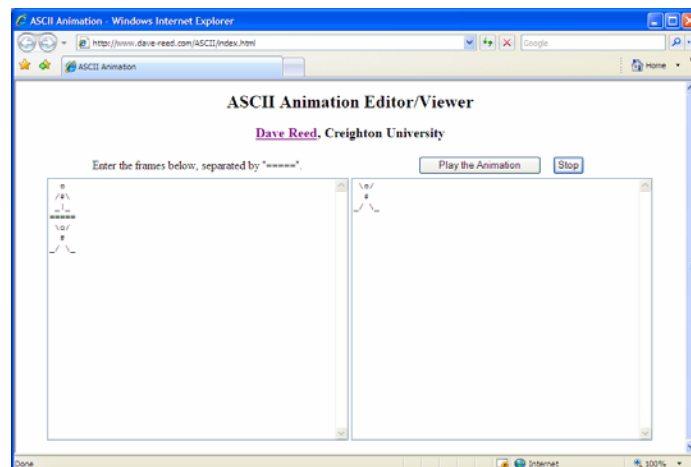
15

JavaScript timeouts

JavaScript has very simple threading – can set a timeout in order to execute code at a time in the future

ASCII animator

- uses string `split` method to separate the animation into frames (stored in an array)
- repeatedly sets a timeout to display the next frame



the [official contest version](#) of this page integrates PHP code for uploading/downloading animations

16

Java vs. JavaScript

compare this JavaScript-based animation viewer with the Java application you wrote in CSC 427

JavaScript is much simpler

- timeout feature is more straightforward than Java TimerTask
- can utilize HTML elements (textarea, button) instead of creating Swing components

Java version is more powerful/reliable/extensible

- can access files (but not if written as an applet)
- can add additional GUI elements easily, e.g., progress bar, speed slider, ...
- OOP techniques & strong type checking manage complexity better; easy reuse

general rule:

- ✓ for simple tasks where JavaScript suffices, it tends to be simpler and faster to develop
- ✓ if advanced features are desired or future extensibility is expected, then the overhead of Java is worth it