CIS 339
Section 2

Introduction to JavaScript

What is JavaScript

• An interpreted programming language with object-oriented capabilities.
• Shares its syntax with Java, C++, and C.
• Is neither a subset nor a superset of Java.
• Is a weakly typed language.
• The interpreter is embedded within web browsers, and the browser interprets the JavaScript code along with the HTML.
Types

- Numbers
- Strings
- Boolean
- Functions
- Objects
- Arrays

Numbers

- No distinction between fixed and float
- Literals use the same syntax as C and Java.
- Special numeric constants to represent ±infinity, NaN, and min/max numbers.
Strings

- Sequences of characters.
- Literals are enclosed in either single ('') or double ("""") quotes, but otherwise use the same syntax as C and Java.
- Since HTML uses double quotes, it is common to use single quotes for JavaScript strings.

Boolean

- Literals: the reserved words
  - true
  - false
- Automatic conversion to/from numbers, false is zero, and true is one.
- Result of comparisons
- Used in if, while, and for statements the same as in Java and C.
Functions

• A function is a piece of JavaScript code that can be defined once and executed, or invoked, many times by the program.
• Functions take arguments and return values.
• Alternative syntax:
  - function square(x) {return x*x;}
  - var square = function(x) {return x*x;}
  - var square = new Function("x", "return x*x");
  - All three create a function object and assign it to the variable square.

Objects

• An object is a collection of named pieces of data, called properties.
• Properties may be accessed using the dot notation of Java and C.
  - image.width
• Properties may also be accessed as if the object were an associative array.
  - image["width"]
• Objects may be created using the statement:
  - var o = new Object();
Object Properties

- Object properties can hold any type.
- Objects may be assigned new properties merely by assigning a value to them.
  - var o = new Object();
  - // o is an empty object
  - o.x = 1.2;
  - // o now has a property x
- If reference is made to an undefined object property, the value is a special value known as undefined.

Object Literals

- An object literal is a comma-separated list of colon-separated property/value pairs, all enclosed in curly brackets.
  - var point = {x:2.3, y:-1.2};
Arrays

- Arrays are collections of values accessed by an index, starting at zero.
- The size of an array grows to whatever is necessary.
- Elements of an array can be of any type.
  - `var = new Array();`
  - `a[0] = 1.2;`
  - `a[1] = "JavaScript";`
  - `a[2] = {x:1, y:3};`

Variables

- Variables can hold values of any type.
  - `var i;`
  - `i = 10;`
  - `i = "ten";`
- Variable are either of function or global scope, but there is no block scope.
- Assigning a value to an undeclared variable creates that variable in global scope.
JavaScript Program to Calculate Loan Payments

```javascript
function calculate()
{
    var principal = document.loandata.principal.value;
    var interest = document.loandata.interest.value / 1200;
    var payments = document.loandata.years.value * 12;
    var x = Math.pow(1 + interest, payments);
    var monthly = (principal * interest) / (x - 1);
    function round(x) { return Math.round(x * 100) / 100; }

    if (isNaN(monthly) || Number.POSITIVE_INFINITY || Number.NEGATIVE_INFINITY)
    {
        document.loandata.payment.value = "";
        document.loandata.total.value = "";
        document.loandata.totalinterest.value = "";
    }
    else
    {
        document.loandata.payment.value = round(monthly);
        document.loandata.total.value = round(monthly * payments);
        document.loandata.totalinterest.value = round((monthly * payments) - principal);
    }
}
```

Form for Loan Calculator

```html
<form name="loandata">
  <table>
    <tr><td colspan=3><big><b>Enter loan Information:</b></big></td></tr>
    <tr><td>1) </td><td>Amount of the loan</td><td><input type="text" name="principal" size=12 onChange="calculate()" /></td></tr>
    <tr><td>2) </td><td>Annual percentage rate</td><td><input type="text" name="interest" size=12 onChange="calculate()" /></td></tr>
    <tr><td>3) </td><td>Repayment period in years</td><td><input type="text" name="years" size=12 onChange="calculate()" /></td></tr>
    <tr><td colspan=3><big><input type="button" value="Compute" onClick="calculate()" /></big></td></tr>
    <tr><td>4) </td><td>Your payment will be</td><td><input type="text" name="payment" size=12 /></td></tr>
    <tr><td>5) </td><td>Your total payment</td><td><input type="text" name="total" size=12 /></td></tr>
    <tr><td>6) </td><td>Your total interest payments</td><td><input type="text" name="totalinterest" size=12 /></td></tr>
  </table>
</form>
```
Form as Initially Displayed

Form with Data
How JavaScript Accesses the Document

• There is a global object called `document`.
• Each form in the document is contained within an array called `document.forms`.
• A named form is an object named `document.name`.
• The elements of a form are in an array `form.elements`.
• A named element can be accessed via `form.name`.

Form Elements

• There are various elements associated with each type of form element as identified by its HTML tag.
• All elements have some common properties:
  - `name`
  - `type`
  - `value`
Events

- Associated with each form element are one or more event handlers.
- An event handler is string of one or more JavaScript statements separated by semicolons.
- An event handler is executed whenever the event occurs.
  - onChange: when a text input value changes
  - onClick: when a button, checkbox, or radio button is selected

Checkbox vs. Radio Button

- **Radio Button**
  - Mutually exclusive selections.
  - Selection of one deselects the other.
  - Cannot deselect such that none are selected once one has been selected.
  - Sent to the server as `name=value`.
  - Selected value available to JavaScript as `boolean name.value.checked`

- **Checkbox**
  - Any number may be selected.
  - Clicking an unselected value selects it.
  - Clicking a selected value deselects it.
  - Sent to the server as `name=array of selected values`.
  - Available in JavaScript as the array `name`.
  - Selection status available as `name[i].checked`
Checkbox Example

The following HTML:

```html
<input type="checkbox" name="color" value="red">
<input type="checkbox" name="color" value="green">
<input type="checkbox" name="color" value="blue">
```

Creates the array `color` with three elements.

- `color[i].value` will be the value.
- `color[i].checked` will indicate if this item is selected.

JavaScript to Verify Checkboxes

```javascript
function validate(group, element, maxSelections) {
  var count = 0;
  for (var i = 0; i < group.length; i++) {
    if (group[i].checked) count++;
  }
  if (count > maxSelections) {
    element.checked = false;
    alert("You may pick at most " + maxSelections);
  }
}
```
Form with Checkboxes

<form name="f" action="verifyform.jsp" method=POST>
Select your favorite color:
<table>
<tr><td><input type="checkbox" name="color" value="red" onClick="validate(f.color,this,1)"></td>
<td>Red</td></tr>
<tr><td><input type="checkbox" name="color" value="yellow" onClick="validate(f.color,this,1)></td>
<td>Yellow</td></tr>
<tr><td><input type="checkbox" name="color" value="green" onClick="validate(f.color,this,1)"></td>
<td>Green</td></tr>
<tr><td><input type="checkbox" name="color" value="blue" onClick="validate(f.color,this,1)"></td>
<td>Blue</td></tr>
</table>
</form>

Form with Checkboxes (2)

Select at most 2 fruits:
<table>
<tr><td><input type="checkbox" name="fruit" value="A" onClick="validate(f.fruit,this,2)"></td>
<td>Apple</td></tr>
<tr><td><input type="checkbox" name="fruit" value="O" onClick="validate(f.fruit,this,2)"></td>
<td>Orange</td></tr>
<tr><td><input type="checkbox" name="fruit" value="G" onClick="validate(f.fruit,this,2)"></td>
<td>Grape</td></tr>
</table>
<input type="submit" value="Submit"><input type="reset">
</form>
Empty Form

Form With Valid Selections
Client vs Server Validation

- Client validation has the advantage that it ensures that valid data is submitted to the server.
- No internet bandwidth is required for the validation.
- However, browser users may turn off JavaScript!
public class FormBean {
    private String[] color;
    private String[] fruit;
    private String[] validColors;
    private String[] validFruits;
    private int maxColors;
    private int maxFruits;
    public FormBean() {
        validColors = new String[]{"Red", "Yellow", "Green", "Blue"};
        maxColors = 1;
        validFruits = new String[]{"Apple", "Orange", "Grape", "Grapefruit", "Pineapple", "Banana"};
        maxFruits = 3;
    }
    public String[] getColor() {return color;}
    public void setColor(String[] color) {
        this.color = color;}
    public String[] getFruit() {return fruit;}
    public void setFruit(String[] fruit) {
        this.fruit = fruit;}
    public String[] getValidColors() {return validColors;}
    public String[] getValidFruits() {return validFruits;}
    public int getMaxColors() {return maxColors;}
    public int getMaxFruits() {return maxFruits;}
    public boolean isColorValid() {
        if (color != null)
            return color.length <= maxColors;
        else return true;
    }
    public boolean isFruitValid() {
        if (fruit != null)
            return fruit.length <= maxFruits;
        else return true;
    }
    public boolean isValid() {
        return isColorValid() && isFruitValid();
    }
    public boolean isFruitSelected(int i) {
        return isSelected(validFruits[i], fruit); }
    public boolean isColorSelected(int i) {
        return isSelected(validColors[i], color); }
    private boolean isSelected(String item, String[] choices) {
        if (choices != null)
            for (int i = 0; i < choices.length; i++)
                if (item.compareTo(choices[i]) == 0) return true;
        return false;
    }
}
FormTest.jsp

```html
<%@ page language="java" contentType="text/html" %>
<jsp:useBean id="formData" class="FormBean" scope="request" />
<html>
<head>
<Title>Form Test</Title>
</head>
<script language="JavaScript1.1">
function validate(group, element, maxSelections)
{  
    names = "";
    var count = 0;
    for (var i = 0; i < group.length; i++) {
        if (group[i].checked) count++;
    }
    if (count > maxSelections) {
        element.checked = false;
        alert("You may pick at most " + maxSelections);
    }
}
</script>
<body>
<form name="f" action="verifyform.jsp" method=POST>
Select at most <%=formData.getMaxColors()%> color(s):
<table>
 <% for (int i = 0; i < formData.getValidColors().length; i++) { %>
<tr>
<td><input type="checkbox" name="color" value="<%=formData.getValidColors()[i]%>"
<%=formData.isColorSelected(i)?"checked":""%> onClick="validate(f.color,this,"<%=formData.getMaxColors()%>)"></td>
<td><%=formData.getValidColors()[i]%></td>
</tr>
<% } %>
</table>
<% if (!formData.isColorValid()) { %>
<font color="red">
You selected too many colors!
</font>
<%}%>
</form>
</body>
```

FormTest.jsp (2)

```html
<body>
<form name="f" action="verifyform.jsp" method=POST>
Select at most <%=formData.getMaxColors()%> color(s):
<table>
  <% for (int i = 0; i < formData.getValidColors().length; i++) { %>
  <tr>
    <td><input type="checkbox" name="color" value="<%=formData.getValidColors()[i]%>"
    <%=formData.isColorSelected(i)?"checked":""%> onClick="validate(f.color,this,"<%=formData.getMaxColors()%>)"></td>
    <td><%=formData.getValidColors()[i]%></td>
  </tr>
  <% } %>
</table>
<% if (!formData.isColorValid()) { %>
  <font color="red">
  You selected too many colors!
  </font>
<%}%>
</form>
</body>
```
FormTest.jsp (3)

```html
<P>Select at most <%=formData.getMaxFruits()%> fruits:
<table>
  <% for (int i = 0; i < formData.getValidFruits().length; i++) { %>
    <tr>
      <td><input type="checkbox" name="fruit"
        value="<%=formData.getValidFruits()[i]%>"
        <%=formData.isFruitSelected(i) ? "checked":""
        onClick="validate(f.fruit,this,
        <%=formData.getMaxFruits()%>);">
      </td>
      <td><%=formData.getValidFruits()[i]%></td>
    </tr>
  <% } %>
</table>
<% if (!formData.isFruitValid()) { %>
  <font color="red">
  You selected too many fruits!
  </font>
<% } %>
<P><input type="submit" value="Submit">&nbsp;<input type="reset">
</body>
</html>
```

verifyform.jsp

```jsp
<%@ page language="java" contentType="text/html"%>
<jsp:useBean id="formData" class="FormBean" scope="request">
  <jsp:setProperty name="formData" property="*"/>
</jsp:useBean>
<% if(formData.isValid()) {%>
  <jsp:forward page="validdata.html"/>
<% } else { %>
  <jsp:forward page="formtest.jsp"/>
<% } %>
```

9/26/2001  © 2001 Paul Wolfgang
Cautions

- If the browser sees invalid JavaScript it will merely ignore it. There are no error messages!
- Since there is no type checking or verification that a variable or property is declared, some errors will not be caught and strange results will occur.