CSC 551: Web Programming

Spring 2004

Event-driven programs and HTML form elements

- event-driven programs
  - ONLOAD, ONUNLOAD
- HTML forms & attributes
  - button, text box, text area
  - selection list, radio button, check box, password, hidden, …
- JavaScript form events
  - properties: name, type, value, …
  - methods: blur(), focus(), click(), …
  - event handlers: onBlur(), onFocus(), onChange(), onClick(), …
- advanced features & techniques
  - windows & frames, timeouts, cookies

Event-driven programs

in C++, programs are serially executed

- start with main function, execute sequentially from first statement
- may loop or skip sections of code, but the program proceeds step-by-step

the programmer specifies the sequence in which execution occurs (with some variability due to input values)

there is a beginning and an end to program execution

computation within a Web page is rarely serial

instead, the page *reacts* to events such as mouse clicks, buttons, …

- much of JavaScript's utility is in specifying actions that are to occur in the page as a result of some event

the programmer may have little or no control over when code will (if ever) be executed, e.g., code that reacts to a button click

there is no set sequence, the page waits for events and reacts

OnLoad & OnUnload

```
<html>
 <!-- Dave Reed  form01.html  2/10/04 -->

 <head>
  <title>Hello/Goodbye page</title>

  <script type="text/javascript">
     function Hello()
     {
       globalName=prompt("Welcome to my page. " +
                         "What is your name?","");
     }

     function Goodbye()
     {
       alert("So long, " + globalName +
             " come back real soon.");
     }
  </script>
 </head>

 <body onLoad="Hello();" onUnload="Goodbye();">
    Whatever text appears in the page.
 </body>
</html>
```

view page in browser

the simplest events are when the page is loaded or unloaded

- the ONLOAD attribute of the BODY tag specifies JavaScript code that is automatically executed when the page is loaded
- the ONUNLOAD attribute similarly specifies JavaScript code that is automatically executed when the browser leaves the page

HTML forms

most event-handling in JavaScript is associated with form elements

an HTML form is a collection of elements for handling input, output, and events in a page

```
<form name="FormName">
…
</form>
```

form elements include:

for input: button, selection list, radio button, check box, password, …
for input/output: text box, text area, …

we will revisit forms when we consider CGI programming

- a form groups together elements, whose contents are submitted as one

Button element

the simplest form element is a button

- analogous to a real-world button, can click to trigger events

```
<input type="button" value="LABEL" onClick="JAVASCRIPT_CODE" />
```

```
<html>
 <!-- Dave Reed   form02.html   2/10/04 -->

 <head>
  <title> Fun with Buttons</title>

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

 <body>
  <form name="ButtonForm">
    <input type="button" value="Click for Lucky Number"
           onClick="num = RandomInt(1, 100);
                    alert('The lucky number for the day is ' + num);" />
  </form>
 </body>
</html>
```

view page in browser


```
<html>
 <!-- Dave Reed   form03.html   2/10/04 -->

 <head>
  <title> Fun with Buttons</title>

  <script type="text/javascript">
      function Greeting()
      // Results: displays a time-sensitive greeting
      {
        var now = new Date();
        if (now.getHours() < 12) {
          alert("Good morning");
        }
        else if (now.getHours() < 18) {
          alert("Good afternoon");
        }
        else {
          alert("Good evening");
        }
      }
  </script>
 </head>

 <body>
  <form name="ButtonForm">
    <input type="button" value="Click for Greeting"
           onClick="Greeting();" />
  </form>
 </body>
</html>
```

view page in browser

Buttons & functions

for complex tasks,

should define function(s) and have the ONCLICK event trigger a function call

Buttons & windows

alert boxes are fine for displaying short, infrequent messages

- not well-suited for displaying longer, formatted text
- not integrated into the page, requires the user to explicitly close the box

QUESTION: could we instead use document.write ?

NO -- would overwrite the current page, including form elements

but could open a new browser window and write there

```
var OutputWindow = window.open();              // open window and assign
                                               //   a name to that object
                                               //   (first arg is an HREF)
OutputWindow.document.open();                  // open that window for
                                               //   writing
OutputWindow.document.write("WHATEVER");       // write text to that
                                               //   window as before
OutputWindow.document.close();                 // close the window
```


Window example

```
<html>
 <!-- Dave Reed   form04.html   2/10/04 -->

 <head>
  <title> Fun with Buttons </title>
  <script type="text/javascript">
    function Help()
    // Results: displays a help message in a separate window
    {
      var OutputWindow = window.open();
      OutputWindow.document.open();

      OutputWindow.document.write("This might be a context-" +
                "sensitive help message, depending on the " +
                "application and state of the page.");

      OutputWindow.document.close();
    }
  </script>
 </head>

 <body>
  <form name="ButtonForm">
    <input type="button" value="Click for Help"
           onClick="Help();" />
  </form>
 </body>
</html>
```

view page in browser

```
<html>
 <!-- Dave Reed   form05.html   2/10/04 -->

 <head>
  <title> Fun with Buttons </title>
  <script type="text/javascript">
    function Help()
    // Results: displays a help message in a separate window
    {
      var OutputWindow =
          window.open("", "",
                      "status=0,menubar=0,height=200,width=200");
      OutputWindow.document.open();

      OutputWindow.document.write("This might be a context-" +
               "sensitive help message, depending on the " +
               "application and state of the page.");

      OutputWindow.document.close();
    }
  </script>
 </head>

 <body>
  <form name="ButtonForm">
    <input type="button" value="Click for Help"
           onClick="Help();" />
  </form>
 </body>
</html>
```

view page in browser

Window example refined

can have arguments to `window.open`

1st arg specifies HREF

2nd arg specifies internal name

3rd arg specifies window properties (e.g., size)

Text boxes

a text box allows for user input

- unlike prompt, user input persists on the page & can be edited

`<input type="text" name="BOX_NAME"…>`

optional attributes: `SIZE` : width of the box (number of characters)
`VALUE` : initial contents of the box

JavaScript code can access the contents as `document.FormName.BoxName.value`

```
<html>
 <!-- Dave Reed   form06.html   2/10/04 -->

 <head>  <title> Fun with Text Boxes </title>  </head>

 <body>
  <form name="BoxForm">
    Enter your name here:
    <input type="text"  name="userName" size=12 value="" />
    <br /><br />
    <input type="button" value="Click Me"
       onClick="alert('Thanks, ' + document.BoxForm.userName.value +
                      ', I needed that.');" />
  </form>
 </body>
</html>
```

view page in browser

Read/write text boxes

similarly, can change the contents with an assignment

Note: the contents are raw text, no HTML formatting

Also: contents are accessed as a string, must parseFloat if want a number

```
<html>
 <!-- Dave Reed   form07.html   2/10/04 -->

 <head>
  <title> Fun with Text Boxes </title>
 </head>

 <body>
   <form name="BoxForm">
     Enter a number here:
     <input type="text" size=12 name="number" value=2 />
     <br /><br />
     <input type="button" value="Double"
            onClick="document.BoxForm.number.value=
                       parseFloat(document.BoxForm.number.value) * 2;" />
   </form>
 </body>
</html>
```

view page in browser


```
<html>
 <!-- Dave Reed   form08.html   2/10/04 -->

 <head>
  <title> Fun with Text Boxes </title>
  <script type="text/javascript">
    function FahrToCelsius(tempInFahr)
    // Assumes: tempInFahr is a number (degrees Fahrenheit)
    // Returns: corresponding temperature in degrees Celsius
    {
      return (5/9)*(tempInFahr - 32);
    }
  </script>
 </head>

 <body>
  <form name="BoxForm">
    Temperature in Fahrenheit:
    <input type="text" name="Fahr" size=10 value="0"
     onChange="document.BoxForm.Celsius.value =
       FahrToCelsius(parseFloat(document.BoxForm.Fahr.value));" />
      <tt>----></tt>  
    <input type="text" name="Celsius" size=10 value=""
           onFocus="blur();" />
    in Celsius
  </form>
 </body>
</html>
```

view page in browser

Text box events

ONCHANGE
triggered when the contents of the box are changed

ONFOCUS
triggered when the mouse clicks in the box

`blur()`
removes focus

Text box validation

what if the user enters a non-number in the Fahrenheit box?

solution: have the text box validate its own contents

- start with legal value
- at ONCHANGE, verify that new value is legal (otherwise, reset)
- the `verify.js` library defines several functions for validating text boxes

```
function VerifyNum(textBox)
// Assumes: textBox is a text box
// Returns: true if textBox contains a number, else false + alert
{
  var boxValue = parseFloat(textBox.value);
  if ( isNaN(boxValue) ) {
    alert("You must enter a number value!");
    textBox.value = "";
    return false;
  }
  return true;
}
```


Validation example

```
<html>
 <!-- Dave Reed   form09.html   2/10/04 -->

 <head>
  <title> Fun with Text Boxes </title>
  <script type="text/javascript"
      src="http://www.creighton.edu/~davereed/csc551/JavaScript/verify.js">
  </script>

  <script type="text/javascript">
    function FahrToCelsius(tempInFahr)
    {
      return (5/9)*(tempInFahr - 32);
    }
  </script>
 </head>

 <body>
  <form name="BoxForm">
    Temperature in Fahrenheit:
    <input type="text" name="Fahr" size=10 value=0
       onChange="if (VerifyNum(this)) {  // this refers to current element
                 document.BoxForm.Celsius.value =
                     FahrToCelsius(parseFloat(this.value));
                 }" />
      <tt>----></tt>  
    <input type="text" name="Celsius" size=10 value="" onFocus="blur();" />
    in Celsius
  </form>
 </body>
</html>
```

view page in browser

Text areas

a TEXT box is limited to one line of input/output

a TEXTAREA is similar to a text box in functionality, but can specify any number of rows and columns

```
<textarea name="TextAreaName" rows=NumRows cols=NumCols wrap="virtual">
Initial Text
</textarea>
```

- *Note:* unlike a text box, a TEXTAREA has closing tag
  initial contents of the TEXTAREA appear between the tags
- WRAP="virtual" specifies that text in the box will wrap lines as needed
- as with a text box, no HTML formatting of TEXTAREA contents

Textarea example

```
<html> <!-- Dave Reed   form10.html   2/10/04 -->
 <head>  <title> Fun with Textareas </title>
  <script type="text/javascript"
    src="http://www.creighton.edu/~davereed/csc551/JavaScript/verify.js">
  </script>

  <script type="text/javascript">
    function Table(low, high, power)
    // Results: displays table of numbers between low & high, raised to power
    {
      var message = "i: i^" + power + "\n-------\n";
      for (var i = low; i <= high; i++) {
        message = message + i + ": " + Math.pow(i, power) + "\n";
      }
      document.AreaForm.Output.value = message;
    }
  </script>
 </head>

 <body>
  <form name="AreaForm">
    <div style="text-align:center">
      Show the numbers from <input type="text" name="lowRange" size=4 value=1
                                   onChange="VerifyInt(this);" />
      to <input type="text" name="highRange" size=4 value=10
                onChange="VerifyInt(this);" />
      raised to the power of <input type="text" name="power" size=3 value=2
                                    onChange="VerifyInt(this);" />
      <br /> <br />
      <input type="button" value="Generate Table"
             onClick="Table(parseFloat(document.AreaForm.lowRange.value),
                            parseFloat(document.AreaForm.highRange.value),
                            parseFloat(document.AreaForm.power.value));" />
      <br /> <br />
      <textarea name="Output" rows=20 cols=15 wrap="virtual"></textarea>
    </div>
  </form>
 </body>
</html>
```

view page in browser

More examples

Hoops tournament
- text boxes in a table to form brackets
- users selects teams by clicking on text boxes, automatically filling in brackets

Letter sequence generator
- text boxes to input sequence length, number of sequences, letter choices
- button to initiate generation of sequences
- text area to display sequences

Substitution cipher
- text box to enter substitution key
- text areas for message & code, generates code at ONCHANGE event

Prisoner's Dilemma simulation
- select boxes for choosing strategies to compete
- text boxes for results of each round, scores
- buttons to play a single round, complete all rounds, reset

Random walk simulator
- text box to display position of walker, number of steps
- button to initiate a step

JavaScript & frames

alternatives for program output:

1. alert box : good for small messages
2. separate window : good for longer text, outside of page
3. text box / text area : integrated into page, but awkward & no formatting
4. frames : can easily write lots of output, integrated & fully formattable

```
<html>
 <!-- Dave Reed  frame11.html  2/10/04 -->

 <head>
   <title>Table of Squares</title>
 </head>

 <frameset rows="20%,*">
   <frame name="Input" src="form11.html">
   <frame name="Output" src="about:blank">
 </frameset>

</html>
```

`src="about:blank"` loads a blank page into the frame (ready to be written to)

Frame example

```
<html>
 <!-- Dave Reed   form11.html   2/10/04 -->

 <head>
  <title> Fun with Frames</title>

  <script type="text/javascript">
    function Help()
    // Results: displays a help message in a separate frame
    {
      parent.Output.document.open();
      parent.Output.document.write("This might be a context-" +
               "sensitive help message, depending on the " +
               "application and state of the page.");
      parent.Output.document.close();
    }
  </script>
 </head>

 <body>
  <form name="ButtonForm">
    <input type="button" value="Click for Help" onClick="Help();" />
  </form>
 </body>
</html>
```

view page in browser

Better example

```
<html> <!-- Dave Reed   form12.html   2/10/04 -->

 <head> <title>Fun with Frames</title>
  <script type="text/javascript"
          src="http://www.creighton.edu/~davereed/csc551/JavaScript/verify.js">
  </script>
  <script type="text/javascript">
      function Table(low, high, power)
      {
        parent.Output.document.open();
        parent.Output.document.write("<table border=1><tr><th>i</th>" +
                                     "<th>i<sup>" + power + "</sup></th></tr>");
        for (var i = low; i <= high; i++) {
            parent.Output.document.write("<tr><td align='right'>" + i + "</td>" +
                    "<td align='right'>" + Math.pow(i, power) + "</td></tr>");
        }
        parent.Output.document.write("</table>");
        parent.Output.document.close();
      }
  </script>
 </head>

 <body>
  <form name="ButtonForm">
    <div style="text-align:center">
      Show the numbers from <input type="text" name="lowRange" size=4 value=1
                                   onChange="VerifyInt(this);" />
      to <input type="text" name="highRange" size=4 value=10
                onChange="VerifyInt(this);" />
      raised to the power of <input type="text" name="power" size=3 value=2
                                    onChange="VerifyInt(this);" />
      <br /><br />
      <input type="button" value="Generate Table"
             onClick="Table(parseFloat(document.ButtonForm.lowRange.value),
                            parseFloat(document.ButtonForm.highRange.value),
                            parseFloat(document.ButtonForm.power.value));" />
    </div>
  </form>
 </body>
</html>
```

view page in browser

JavaScript & timeouts

the setTimeout function can be used to execute code at a later time

setTimeout(*JavaScriptCodeToBeExecuted*, *MillisecondsUntilExecution*)

example: forward link to a moved page

```
<html>
 <!-- Dave Reed   form13.html   2/10/04 -->

 <head>
  <title> Fun with Timeouts </title>
  <script type="text/javascript">
    function Move()
    // Results: sets the current page contents to be newhome.html
    {
        self.location.href = "newhome.html";
    }
  </script>
 </head>

 <body onLoad="setTimeout('Move()', 3000);">
   This page has moved to <a href="newhome.html">newhome.html</a>.
 </body>
</html>
```

view page in browser

Another timeout example

```
<html>
 <!-- Dave Reed   form14.html   2/10/04 -->

 <head>
  <title> Fun with Timeouts </title>
  <script type="text/javascript">
    function timeSince()
    // Assumes: document.CountForm.countdown exists in the page
    // Results: every second, recursively writes current countdown in the box
    {
      // CODE FOR DETERMINING NUMBER OF DAYS, HOURS, MINUTES, AND SECONDS
      // UNTIL GRADUATION

      document.CountForm.countdown.value=
        days + " days, " + hours + " hours, " +
        minutes + " minutes, and " + secs + " seconds";

      setTimeout("timeSince();", 1000);
    }
  </script>
 </head>

 <body onLoad="timeSince();">
  <form name="CountForm">
    <div style="text-align:center">
      Countdown to Graduation 2004 <br />
      <textarea name="countdown" rows=4 cols=15
                style="font-family:Courier" onFocus="blur();"></textarea>
    </div>
  </form>
 </body>
</html>
```

view page in browser

Cookies & JavaScript

recall that cookies are data files stored on the client machine

- can be accessed and/or modified by the server
- can also be accessed and/or modified directly by JavaScript code in a page

potential applications:

- e-commerce: remember customer name, past visits/purchases, password, …
- tutorials: remember past experience, performance on quizzes, …
- games: remember high score, best times, …

each Web page can have its own cookie

- `document.cookie` is a string of `attribute=value` pairs, separated by ;

```
"userName=Dave;score=100;expires=Mon, 21-Feb-01 00:00:01 GMT"
```


cookie.js

```
function getCookie(Attribute)
// Assumes: Attribute is a string
// Results: gets the value stored under the Attribute
{
    if (document.cookie.indexOf(Attribute+"=") == -1) {
      return "";
    }
    else {
        var begin = document.cookie.indexOf(Attribute+"=") + Attribute.length+1;
        var end = document.cookie.indexOf(";", begin);
        if (end == -1) end = document.cookie.length;
        return unescape(document.cookie.substring(begin, end));
    }
}

function setCookie(Attribute, Value)
// Assumes: Attribute is a string
// Results: stores Value under the name Attribute, expires in 30 days
{
    var ExpireDate = new Date();
    ExpireDate.setTime(ExpireDate.getTime() + (30*24*3600*1000));
    document.cookie = Attribute + "=" + escape(Value) +
                      "; expires=" + ExpireDate.toGMTString();
}

function delCookie(Attribute)
// Assumes: Attribute is a string
// Results: removes the cookie value under the name Attribute
{
    var now = new Date();
    document.cookie = Attribute + "=; expires=" + now.toGMTString();
}
```

Cookie example

```
<html>
 <!-- Dave Reed   form15.html   2/10/04 -->

 <head>
  <title> Fun with Cookies </title>
  <script type="text/javascript"
          src="http://www.creighton.edu/~davereed/csc551/JavaScript/cookie.js">
  </script>

  <script type="text/javascript">
    function Greeting()
    // Results: displays greeting using cookie
    {
      visitCount = getCookie("visits");
      if (visitCount == "") {
        alert("Welcome to my page, newbie.");
        setCookie("visits", 1);
      }
      else {
        visitCount = parseFloat(visitCount)+1;
        alert("Welcome back for visit #" + visitCount);
        setCookie("visits", visitCount);
      }
    }
  </script>
 </head>

 <body onLoad="Greeting();">
  Here is the stuff in my page.
  <form name="ClearForm" align="center">
   <div style="text-align:center">
      <input type="button" value="Clear Cookie" onClick="delCookie('visits');" />
   </div>
  </form>
 </body>
</html>
```

view page in browser