

CSC 551: Web Programming

Fall 2001

HTML forms & JavaScript events

- 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

HTML forms

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" ...>
```

attributes include: VALUE : specifies label that appears on the button
ONCLICK : specifies code to be executed when clicked

```
<html>
  <!-- Dave Reed   form01.html   9/25/01 -->

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

  <body>
    <form name="ButtonForm">
      <input type="button" value="Click Me"
        onClick="alert('Thanks, I needed that.');" />
    </form>
  </body>
</html>
```

[view page in browser](#)

Buttons & JavaScript

the ONCLICK event-handler can specify any JavaScript code

- can be a sequence of statements inside quotes, can call functions, ...

```
<html>
  <!-- Dave Reed   form02.html   9/25/01 -->

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

    <script language="JavaScript"
      src="http://www.creighton.edu/~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 9/25/01 -->
<head>
<title> Fun with Buttons</title>
<script language="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("", "WinName"); // open window and assign
// a name to that object
OutputWindow.document.open(); // (first arg is an HREF)
OutputWindow.document.write("WHATEVER"); // open that window for
// writing
OutputWindow.document.close(); // write text to that
// window as before
// close the window

```

```

<html>
  <!-- Dave Reed   form04.html   9/25/01 -->

  <head>
    <title> Fun with Buttons </title>
    <script language="JavaScript">
      function Help()
      // Results: displays a help message in a separate window
      {
        var OutputWindow = window.open("", "OutWin");
        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>

```

Window example

[view page in browser](#)

```

<html>
  <!-- Dave Reed   form05.html   9/25/01 -->

  <head>
    <title> Fun with Buttons </title>
    <script language="JavaScript">
      function Help()
      // Results: displays a help message in a separate window
      {
        var OutputWindow =
          window.open("", "OutWin",
            "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>

```

Window example refined

can have a 3rd argument to window.open

specifies window properties (e.g., size)

[view page in browser](#)

Text boxes

a text box allows for user input

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

```
<input type="text" ...>
```

attributes include: NAME : name by which its contents can be referred
SIZE : width of the box (number of characters)
VALUE : initial contents of the box

```
<html>
  <!-- Dave Reed   form06.html   9/25/01 -->
</html>
<head>
  <title> Fun with Text Boxes </title>
</head>
<body>
  <form name="BoxForm">
    Enter your name here:
    <input type="text" name="userName" size=12 value="">
    <p>
    <input type="button" name="clicker" value="Click Me"
      onClick="alert('Thanks, ' + document.BoxForm.userName.value +
        ', I needed that.');">
  </form>
</body>
</html>
```

[view page in browser](#)

Read/write text boxes

can access text box contents as `document.FormName.BoxName.value`

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   9/25/01 -->
</html>
<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>
    <p>
    <input type="button" name="double" 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   9/25/01 -->

  <head>
    <title> Fun with Text Boxes </title>
    <script language="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));">
      &nbsp; <tt>----</tt> &nbsp;
      <input type="text" name="Celsius" size=10 value=""
        onFocus="blur();">
      in Celsius
    </form>
  </body>
</html>

```

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

[view page in browser](#)

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, resetValue)
// Assumes: textBox is a text box, resetValue is optional
// Results: alert if textBox does not contain a number, resets if provided
{
  var boxValue = parseFloat(textBox.value);
  if ( isNaN(boxValue) ) {
    alert("You must enter a number value!");
    if (resetValue != null) {
      textBox.value = resetValue;
    }
  }
}

```

Validation example

```
<html>
  <!-- Dave Reed   form09.html   9/25/01 -->
<head>
  <title> Fun with Text Boxes </title>
  <script language="JavaScript"
    src="http://www.creighton.edu/~csc551/JavaScript/verify.js">
  </script>
  <script language="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="verifyNum(this, 0); // this refers to current element
      document.BoxForm.Celsius.value =
        FahrToCelsius(parseFloat(this.value));">
    &nbsp; <tt>----</tt> &nbsp; &nbsp;
    <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

```

<html>
  <!-- Dave Reed   form10.html   9/25/01 -->
</html>
<head>
  <title> Fun with Textareas </title>
  <script language="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 align="center">
      Show the numbers from
      <input type="text" name="lowRange" size=4 value=1> to
      <input type="text" name="highRange" size=4 value=10>
      raised to the power of <input type="text" name="power" size=3 value=2>

      <p><input type="button" value="Generate Table"
        onClick="Table(parseFloat(document.AreaForm.lowRange.value),
          parseFloat(document.AreaForm.highRange.value),
          parseFloat(document.AreaForm.power.value));">

      <p><textarea name="Output" rows=20 cols=15 wrap="virtual"></textarea>
    </div>
  </form>
</body>
</html>

```

Textarea example

[view page in browser](#)

```

<html>
  <!-- Dave Reed   form11.html   9/25/01 -->
</html>
<head>
  <title> Fun with Frames </title>
  <script language="JavaScript"
    src="http://www.creighton.edu/~csc551/JavaScript/verify.js">
  </script>

  <script language="JavaScript">
    function Table(low, high, power) { /* AS BEFORE */ }
  </script>
</head>
<body>
  <form name="AreaForm">
    <div align="center">
      Show the numbers from
      <input type="text" name="lowRange" size=4 value=1
        onChange="verifyInt(this, 1);"> to
      <input type="text" name="highRange" size=4 value=10
        onChange="verifyInt(this, 10);">
      raised to the power of
      <input type="text" name="power" size=3 value=2
        onChange="verifyInt(this, 2);">

      <p><input type="button" value="Generate Table"
        onClick="Table(parseFloat(document.AreaForm.lowRange.value),
          parseFloat(document.AreaForm.highRange.value),
          parseFloat(document.AreaForm.power.value));">

      <p><textarea name="Output" rows=20 cols=15 wrap="virtual" onFocus="blur();">
        </textarea>
    </div>
  </form>
</body>
</html>

```

Textarea example refined

[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  frame12.html  9/25/01 -->

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

<frameset rows="20%,*">
  <frame name="Input" src="form12.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   form12.html   9/25/01 -->

<head>
  <title> Fun with Frames</title>
  <script language="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](#)

```
<html>
<!-- Dave Reed   form13.html   9/25/01 -->

<head>
  <title>Fun with Frames</title>
  <script language="JavaScript">
    function Table(low, high, power)
      {
        parent.Output.document.open();
        parent.Output.document.write("<table border=1><tr><th>i</th> " +
          "<th>i<sup>2</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 align="center">
      Show the numbers from
      <input type="text" name="lowRange" size=4 value=1> to
      <input type="text" name="highRange" size=4 value=10>
      raised to the power of <input type="text" name="power" size=3 value=2>

      <p><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>
```

**Better
example**

[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   form14.html   9/25/01 -->

<head>
  <title> Fun with Timeouts </title>
  <script language="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   form15.html   9/25/01 -->

<head>
  <title> Fun with Timeouts </title>
  <script language="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 align="center">
      Countdown to Graduation 2002 <br>
      <input type="text" name="countdown" size=50 value="" onFocus="blur();">
    </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=Thu, 29-Feb-01 00:00:01 GMT"
```

```
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 ExpiresDate = new Date();
  ExpiresDate.setTime(ExpiresDate.getTime() + (30*24*3600*1000));
  document.cookie = Attribute + "=" + escape(Value) +
    "; expires=" + ExpiresDate.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.js

```

<html>
<!-- Dave Reed  form16.html  9/25/01  -->
<head>
<title> Fun with Cookies </title>
<script language="JavaScript"
  src="http://www.creighton.edu/~csc551/JavaScript/cookie.js">
</script>
<script language="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 align="center">
      <input type="button" value="Clear Cookie" onClick="delCookie('visits');">
    </div>
  </form>
</body>
</html>

```

Cookie
example

[view page in browser](#)

Next week...

Java programming for the Web

- design goals and language features
- data types, control statements, program structure
- classes & objects
- applications vs. applets

read Chapters 19 & 20

as always, be prepared for a quiz on

- today's lecture (moderately thorough)
- the reading (superficial)