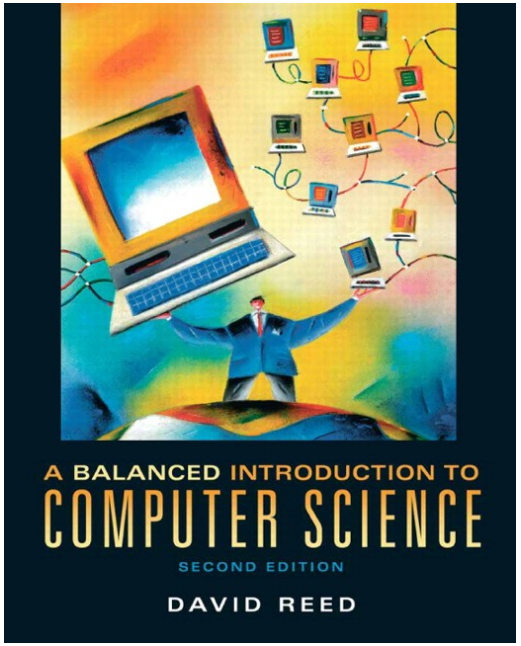
A Balanced Introduction to Computer Science, 2/E

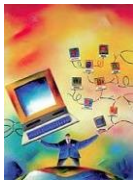
David Reed, Creighton University

ISBN 978-0-13-601722-6



Chapter 7
Event-Driven Pages

Event-driven Pages



one popular feature of the Web is its interactive nature

- e.g., you click on buttons to make windows appear
- e.g., you enter credit card information in a form and submit it

pages that respond to user actions such as mouse clicks or form entries are known as *event-driven pages*

- JavaScript code can be combined with HTML elements such as buttons, text fields, and text areas to produce event-driven pages

an *event handler* is an HTML element that can be programmed to respond to a user's actions

- the simplest event handler is a button
- a button can be associated with JavaScript code that will execute when the button is clicked

Buttons and Forms

general form of a button element:

```
<input type="button" value="BUTTON_LABEL" onclick="JAVASCRIPT_CODE" />
```

- the TYPE attribute of the INPUT element identifies the element to be a button
- the VALUE attribute specifies the text label that appears on the button
- the ONCLICK attribute specifies the action to take place
  - any JavaScript statement(s) can be assigned to the ONCLICK attribute
  - this can be (and frequently is) a call to a JavaScript function

for example,

```
<input type="button" value="Click for Free Money"
                     onclick="alert('Yeah, right.');" />
```

- the predefined `alert` function displays a message in a new window
  - here, the message 'Yeah, right.' is displayed at the click of the button
- a string can be denoted using either double("...") or single ('...') quotes
  - here, single quotes must be used to avoid confusion with the ONCLICK quotes

Random Number Example

recall the task of generating random numbers

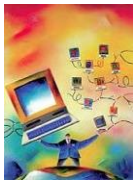
- earlier, we did this by embedding JavaScript code in SCRIPT tags
- each time the page was loaded in the browser, the code was executed and the random number was written into the HTML text using `document.write`

DRAWBACK: the user had to reload for each random number

ALTERNATIVE: place a button in the page with associated code for generating and displaying the random number

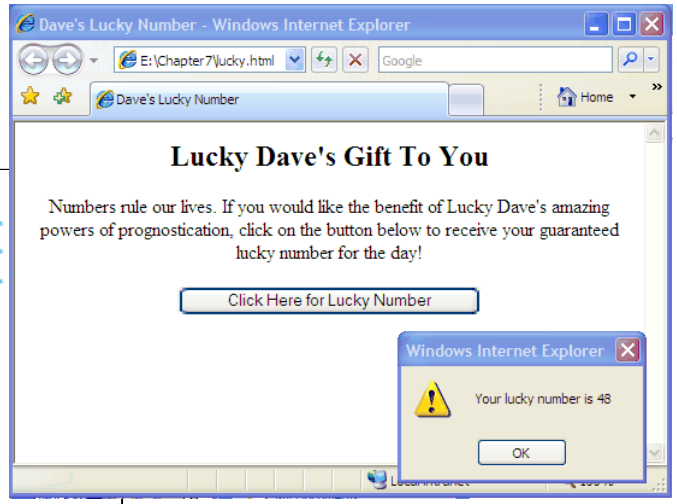
- each time the user clicks the button, the code for generating and displaying the number is executed

LuckyForm Example



when the button is clicked, two JavaScript statements are executed

- a number in the range 1..100 is randomly selected
- that number is displayed in an alert window




```
1.  <html>
2.    <!-- lucky.html                                          Dave Reed -->
3.    <!-- This page displays a lucky number at the click of a button. -->
4.    <!-- ================================================== -->
5.
6.    <head>
7.      <title> Dave's Lucky Number </title>
8.    </head>
9.
10.   <body>
11.     <div style="text-align:center">
12.       <h2>Lucky Dave's Gift To You</h2>
13.
14.       <p>Numbers rule our lives. If you would like the benefit of Lucky Dave's
15.       amazing powers of prognostication, click on the button below to receive
16.       your guaranteed lucky number for the day!</p>
17.
18.       <input type="button" value="Click Here for Lucky Number"
19.              onclick="luckyNum = Math.floor(Math.random()*100)+1;
20.                       alert('Your lucky number is ' + luckyNum);" />
21.     </div>
22.   </body>
23. </html>
```

Output via Text Boxes

a button provides a simple mechanism for user interaction in a Web page

- by clicking the button, the user initiates some action

a *text box* is an event-handler that can display text (a word or phrase)

- unlike an alert window, the text box appears as a box embedded in the page
- text can be written to the box by JavaScript code (i.e., the box displays output)

for example, we could reimplement the lucky number page using a text box

the text box containing the random number is embedded in the page

doesn't require the user to close the alert window after each number

Output via Text Boxes

general form of a text box element:

```
<input type="text" id="BOX_NAME" size="NUM_CHARS" value="INITIAL_TEXT" />
```

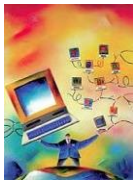
- the TYPE attribute of the INPUT element identifies the element to be a text box
- the ID attribute gives the element an identifier so that it can be referenced
- the SIZE attribute specifies the size of the box (number of characters that fit)
- the VALUE attribute specifies text that initially appears in the box

to display text in a text box, a JavaScript assignment is used to assign to its value attribute

- as part of the assignment, must specify the *absolute name* of the box
- the general form is:

```
document.getElementById('BOX_NAME').value = VALUE_TO_BE_DISPLAYED;
```

Text Box for Displaying Output



```
1.  <html>
2.   <!-- lucky1.html                                          Dave Reed -->
3.   <!-- This page displays a lucky number at the click of a button. -->
4.   <!-- ================================================== -->
5.
6.   <head>
7.     <title> Dave's Lucky Number </title>
8.   </head>
9.
10.  <body>
11.   <div style="text-align:center">
12.     <h2>Lucky Dave's Gift To You</h2>
13.
14.     <p>Numbers rule our lives. If you would like the benefit of Lucky Dave's
15.     amazing powers of prognostication, click on the button below to receive
16.     your guaranteed lucky number for the day!</p>
17.
18.     <input type="button" value="Click Here for Lucky Number"
19.            onclick="luckyNum = Math.floor(Math.random()*100) +1;
20.                     document.getElementById('numberBox').value = luckyNum;" />
21.     <hr />
22.     <p>
23.     Your lucky number is: <input type="text" id="numberBox" size="4" value="" />
24.     </p>
25.   </div>
26.  </body>
27. </html>
```

when the button is clicked, the function call `DisplayNumber();` is executed

the function generates the random number and assigns it to the text box

as a result, each button click yields a new number in the box

Input via Text Boxes

text boxes can also be used for receiving user input

- the user can enter text directly into the box
- that text can then be accessed by JavaScript code via the absolute name of the box

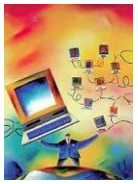
```
document.geElementById('BOX_NAME').value
```

- note that the value retrieved from a text box is always a string
  - if the user enters a number, say 93, then the absolute name will access "93"
  - similar to `prompt`, you must use `parseFloat` to convert the string to its numeric value

example: we can revisit our temperature conversion page

- the user enters the Fahrenheit temperature in a text box
- at the click of a button, the input is accessed and converted to Celsius
- another text box is used to display the converted temperature

Text Boxes for Input

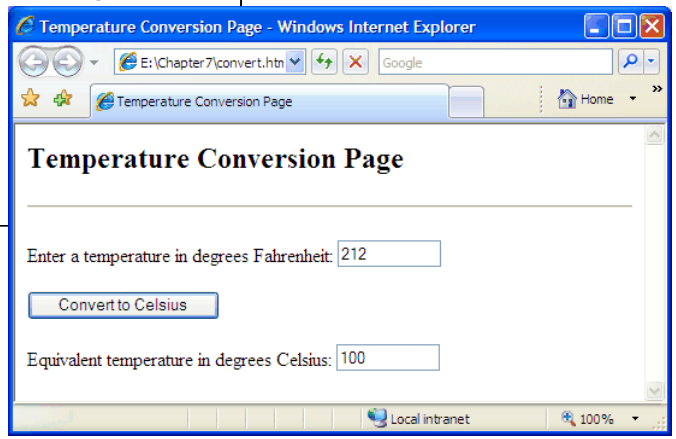


```
1. <html>
2.   <!-- convert.html                                        Dave Reed -->
3.   <!-- This page converts temperatures from Fahrenheit to Celsius. -->
4.   <!-- ========================================================== -->
5.
6.   <head>
7.     <title>Temperature Conversion Page</title>
8.   </head>
9.
10.  <body>
11.    <h2>Temperature Conversion Page</h2>
12.    <hr />
13.    <p>
14.      Enter a temperature in degrees Fahrenheit:
15.      <input type="text" id="fahrBox" size="10" value="" />
16.    </p>
17.    <p>
18.      <input type="button" value="Convert to Celsius"
19.             onclick="tempInF = document.getElementById('fahrBox').value;
20.                      tempInF = parseFloat(tempInF);
21.                      tempInC = (5/9) * (tempInF - 32);
22.                      document.getElementById('celsiusBox').value = tempInC;" />
23.    </p>
24.    <p>
25.      Equivalent temperature in degrees Celsius:
26.      <input type="text" id="celsiusBox" size="10" value="" />
27.    </p>
28.   </body>
29. </html>
```

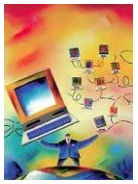
`fahrBox` is used for input

the button's ONCLICK attribute specifies the code for converting the temperature

`celsiusBox` is used for output

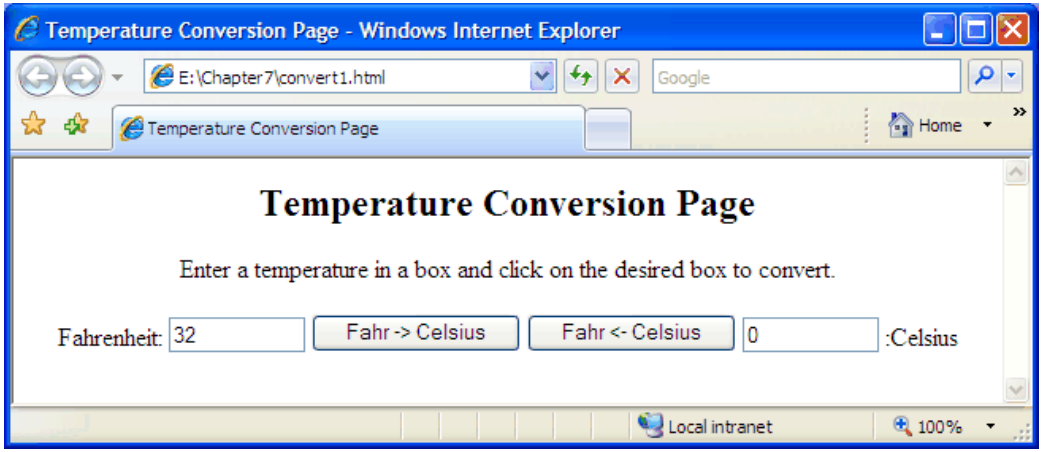


Input and Output

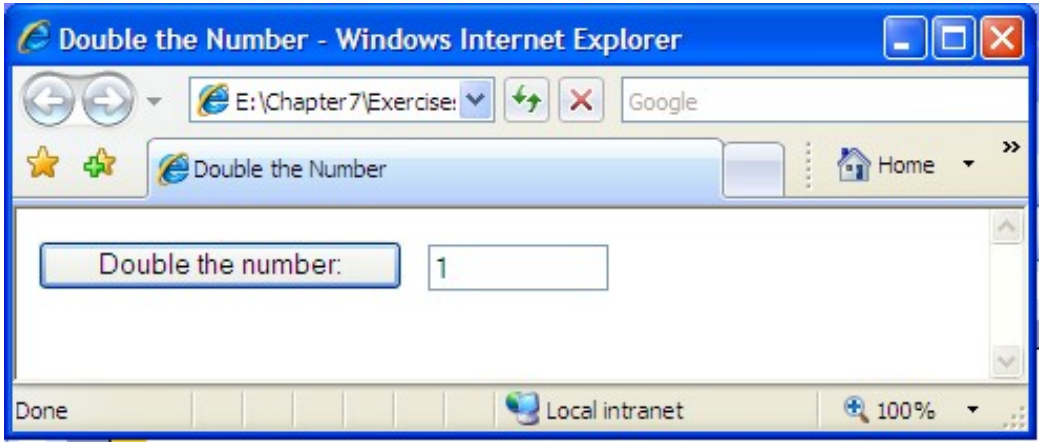


note: the same text box can be used for both input and output

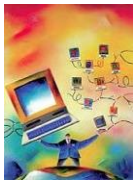
- can modify the conversion page to allow for entering a temperature in either box, then convert to the other



- can write a simple page in which the user can enter a number, then double it by clicking a button



Text Areas



a text area is similar to a text box but it can contain any number of text lines

general form of a text area element:

```
<textarea name="TEXTAREA_NAME" rows=NUM_ROWS cols=NUM_COLS wrap="virtual">
INITIAL_TEXT
</textarea>
```

- the NAME attribute gives the element a name so that it can be referenced
- the ROWS attribute specifies the height (number of text lines) of the area
- the COLS attribute specifies the width (number of characters) of the area
- the WRAP attribute ensures that the text will wrap from one line to the next instead of running off the edge of the text area

unlike a text box, opening and closing tags are used to define a text area

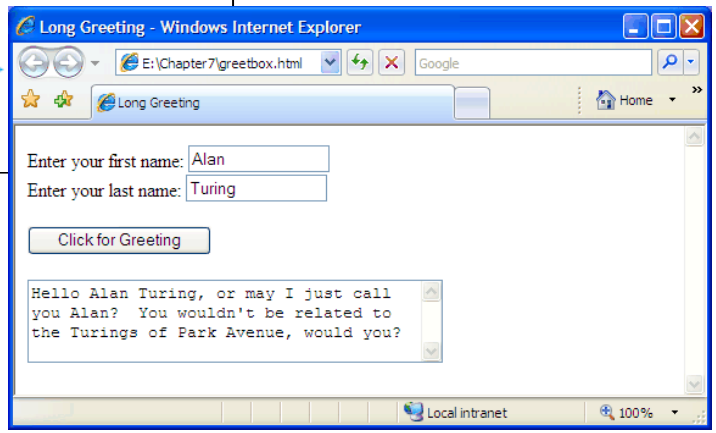
- any text appearing between the tags will be the initial text in the text area
- otherwise, the contents of a text area are accessed/assigned in the same way

Input/Output via Text Areas

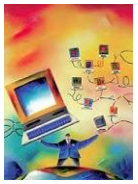
```
1. <html>
2.   <!-- greetbox.html                                        Dave Reed -->
3.   <!-- This page displays a personalized greeting in a text area. -->
4.   <!-- ============================================================= -->
5.
6.   <head>
7.     <title> Long Greeting </title>
8.   </head>
9.
10.  <body>
11.    <p>
12.      Enter your first name: <input type="text" id="firstNameBox" size="15" />
13.      <br />
14.      Enter your last name: <input type="text" id="lastNameBox" size="15" />
15.    </p>
16.    <p>
17.      <input type="button" value="Click for Greeting"
18.             onclick="firstName = document.getElementById('firstNameBox').value;
19.                      lastName = document.getElementById('lastNameBox').value;
20.                      message = 'Hello ' + firstName + ' ' + lastName +
21.                                ', or may I just call you ' + firstName +
22.                                '? You wouldn\'t be related to the ' + lastName +
23.                                's of Park Avenue, would you?';
24.                      document.getElementById('messageArea').value = message;" />
25.    </p>
26.    <p>
27.      <textarea id="messageArea" rows="4" cols="40"></textarea>
28.    </p>
29.  </body>
30. </html>
```

the user enters first and last names into text boxes

a long greeting is constructed using the names and assigned to the text area



Dynamic Images



just as you can use user-initiated events to change the contents of text areas and text boxes, you can also dynamically modify images

```
<img id="faceImg" src="happy.gif" alt="Happy Face" />
```

causes the image stored in the file `happy.gif` to appear in the page

you can change the image by reassigning its SRC attribute

- similar to the way that text boxes/areas have their VALUE attribute reassigned

```
document.getElementById('faceImg').src = "sad.gif";
```

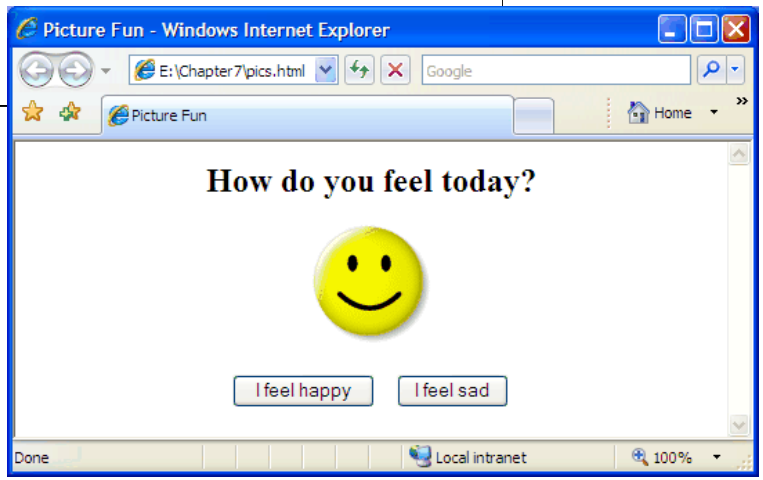
replaces `happy.gif` with `sad.gif`

Dynamic Image Example

```
 1. <html>
 2.   <!-- pics.html                                          Dave Reed -->
 3.   <!-- This page allows the user to select from among two images. -->
 4.   <!-- ============================================================ -->
 5.
 6.   <head>
 7.     <title> Picture Fun </title>
 8.   </head>
 9.
10.   <body>
11.     <div style="text-align:center">
12.       <h2>How do you feel today?</h2>
13.       <img id="faceImg" src="happy.gif" alt="face image" />
14.       <br /><br />
15.       <input type="button" value="I feel happy"
16.              onclick="document.getElementById('faceImg').src = 'happy.gif';" />
17.         
18.       <input type="button" value="I feel sad"
19.              onclick="document.getElementById('faceImg').src = 'sad.gif';" />
20.     </div>
21.   </body>
22. </html>
```

initially the image is set to `happy.gif`

when a button is clicked, images is assigned a new SRC file



Simplifying buttons

consider the button from `greetbox.html`:

```
<input type="button" value="Click for Greeting"
       onclick="firstName = document.getElementById('firstNameBox').value;
                lastName = document.getElementById('lastNameBox').value;
                message = 'Hello ' + firstName + ' ' + lastName +
                          ', or may I just call you ' + firstName +
                          '? You wouldn\'t be related to the ' + lastName +
                          's of Park Avenue, would you?';
                document.getElementById('messageArea').value = message;" />
```

- the size of ONCLICK attribute makes the button complex and difficult to read
- plus, must be careful with nested quotes ("…" vs. '…')

functions provide a mechanism for simplifying complex buttons such as this

recall from Chapter 5:

- functions minimize the amount of detail that has to be considered
  - e.g., can use `Math.sqrt` without worrying about how it works
- functions reduce the length and complexity of code
  - e.g., a single call to `Math.sqrt` replaces the underlying complex algorithm

Simple user-defined functions

in addition to JavaScript's predefined functions, the user can define new functions in the HEAD section and call them within the page

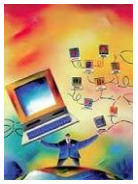
we will explore user-defined functions fully in Chapter 9

- for now, the following simple form suffices for simplifying buttons

```
function FUNCTION_NAME()
// Assumes: DESCRIPTION OF ANY ASSUMPTIONS ABOUT THE PAGE
// Results: DESCRIPTION OF THE ACTION PERFORMED BY THE FUNCTION
{
    STATEMENTS_TO_BE_EXECUTED
}
```

- a function definition begins with the word `function` followed by its name and `()`
  - a function name should be descriptive of the task being performed
- lines beginning with `//` are comments that describe the function's behavior
  - comments are ignored by the interpreter, but make code more user-readable
- the statements to be executed when the function is called are placed between the curly braces

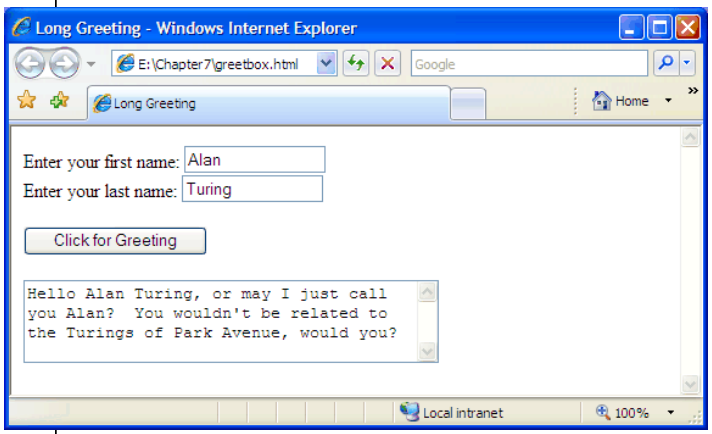
Greeting revisited



```
1.  <html>
2.    <!-- greetbox1.html                                    Dave Reed -->
3.    <!-- This page displays a personalized greeting in a text area. -->
4.    <!-- ========================================================== -->
5.
6.    <head>
7.      <title> Long Greeting </title>
8.
9.      <script type="text/javascript">
10.       function Greet()
11.       // Assumes: firstNameBox and lastNameBox contain names
12.       // Results: writes a message with those names in messageArea
13.       {
14.           firstName = document.getElementById("firstNameBox").value;
15.           lastName = document.getElementById("lastNameBox").value;
16.
17.           message = "Hello " + firstName + " " + lastName +
18.                     ", or may I just call you " + firstName +
19.                     "?  You wouldn't be related to the " + lastName +
20.                     "s of Park Avenue, would you?";
21.
22.           document.getElementById("messageArea").value = message;
23.       }
24.     </script>
25.   </head>
26.
27.   <body>
28.     <p>
29.       Enter your first name: <input type="text" id="firstNameBox" size="15" />
30.       <br />
31.       Enter your last name: <input type="text" id="lastNameBox" size="15" />
32.       <input type="text" id="fahrBox" size="10" value="" />
33.     </p>
34.     <p>
35.       <input type="button" value="Click for Greeting" onclick="Greet();" />
36.     </p>
37.     <p>
38.       <textarea id="messageArea" rows="4" cols="40"></textarea>
39.     </p>
40.   </body>
41. </html>
```

the code from the button is moved to the user-defined `Greet` function

as a result, the button is greatly simplified



GENERAL RULE: if more than one statement is to be associated with a button, define a separate function