Chapter 19: The Bean Counter: A JavaScript Program

Fluency with Information Technology
Third Edition

by
Lawrence Snyder

Preliminaries

- Use a basic text editor; fancy formatting will confuse the web browser
- File format is text; filename extension should be .html
  - The operating system knows the file will be processed by a web browser
- To create program, start a file whose first line is <html> and last line is </html>

Preliminaries (cont'd)

- Enclose JavaScript text in <script> tags

  <script language="JavaScript">

  </script>
- When program is written, save it, then find file on computer and double click it
  - Web browser should open file and display the page you created

```
<html>
  <head><title>Version 0</title></head>
  <body>
    <h1>Here's Version 0 of the Bean Counter</h1>
    <script language = "JavaScript">
      var drink = "latte"; //Temporary Decl. for Version 0;
                           //to be removed
      var shots = 2;       //Temporary Decl. for Version 0;
                           //to be removed
      var ounce = 12;      //Temporary Decl. for Version 0;
                           //to be removed
      var price;
      var caffeine = 0.095;
      if (drink == "espresso")
          price = 1.50;
      if (drink == "latte" || drink == "cappuccino") {
        if (ounce == 8)
            price = 2.25;
        if (ounce == 12)
            price = 2.75;
        if (ounce == 16)
            price = 2.75;
      }
      if (drink == "americano")
        price = 2.10 + .30 * (ounce/8);
      price = price + (shots - 1) * .75;
      price = price + price * caffeine;
      alert(price);    //Temporary statement to print result;
                       //to be changed
    </script>
  </body>
</html>
```

Figure 19.1. Version 0 of the Bean Counter program without the user interface, and with fixed inputs: drink="latte", shots=2, ounce=12.

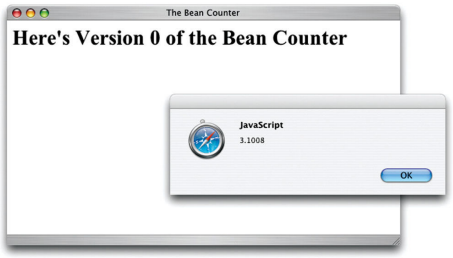


Figure 19.2. *Web page displayed by Version 0 of the Bean Counter program of Figure 19.1.*

Background for the GUI

- Review HTML Basics

```
<head>
    <title> The Bean Counter </title>
</head>
<body>
</body>
```

- All programming in this example is concentrated in the body of the document
- Tables give us control over where information is displayed on a page

```
<table>
  <tr>
    <td> </td>
    …
  </tr>
  …
</table>
```

Background for the GUI (cont'd)

- Interacting with a GUI
  - Input facilities like buttons and checkboxes are known as *elements of forms*
  - Form tags <form> and </form> must surround all input elements
  - Form tag has name attribute:

    <form name = "Bean">

    </form>
- File ends with

  </body>

  </html>


```
<html>
   <head>
      <title>The Bean Counter</title>
   </head>
   <body bgcolor="#804000" text="#FF9900" align="center">
      <font face="Helvetica", "Arial">
      <h1 align="center">
         <font color="#FFFFFF">the bean counter</font></h1>
      <hr width=50%>
      <p align="center"><b>figuring the price of espresso
         drinks<br> so baristas can have time to chat</b></p>
      <form name = "Bean">
      </form>
   </body>
</html>
```

Figure 19.4. The Bean Counter interface to this point, and the HTML that produced it.

Events and Event Handlers

- GUI inputs being used cause an event to occur
  - *Event* is an indication from the computer (Operating System) that something just happened
  - User clicking on command button causes "click event"
- *Event handler* is program that performs task in response to event

Three Input Elements

- **Button**
  - <input type=button value="label" name="identifier" onClick="*event_handler*" />
    - *value* gives text to be printed on the button
    - *identifier* is the name of the element
    - *onClick* gives event handler (JavaScript instructions)
    - Button image is placed in next position in text of HTML document

Three Input Elements

- **Text Box**
  - Used to input or output numbers or words
  - <input type=text name="*identifier*" size=6 onChange="*event_handler*" />
  - *identifier* is the name of the element
  - *onChange* gives the event handler's JavaScript instructions
    - These program instructions are performed after user enters text
  - Text input image is placed in next position in text of HTML program

Three Input Elements

- **Radio Button**
  - Gives a selection of preprogrammed settings
  - <input type=radio name="*identifier*" onClick="*event_handler*" /> *label text*
  - *Identifier* is the name of the element
    - A group of radio buttons use the same name
  - *label text* is shown beside the button
  - *onClick* gives the event handler
    - When user clicks button, center darkens (indicating it is set), other radio buttons are cleared, and the instructions of the event handler are performed
  - Radio button image is placed in the next position in the text of the HTML document

Create the Graphical User Interface

- Bean Counter program is mostly a table of buttons
- Algorithm for building the table:
  - Create a button table
    - Program HTML for a table with a generic button in each cell. Easy to do with Copy/Paste
  - Delete two buttons
    - Two cells should be empty. Don't delete the cells
  - Insert text box
    - Replace button for last cell with a text control
  - Label the buttons
    - Set the value attribute of each button so the label is correct
  - Primp the interface
    - Check and adjust where necessary

Create a Button Table

```
<td>
   <input type="button" value="b" onClick = ' ' />
</td>
```

- "b" is a placeholder for the button label (we'll fix in Step 4), and ' ' is a placeholder for the event handler we'll write later
- Make four copies of the cell and surround them by row tags
- Make four copies of the row and surround them by table tags
- Save the page and review
- Change default left-justification to center—surround the table with   tags

(a)

Figure 19.5. Intermediate stages in the construction of the Bean Counter interface: (a) after Step 1

(continues next page).

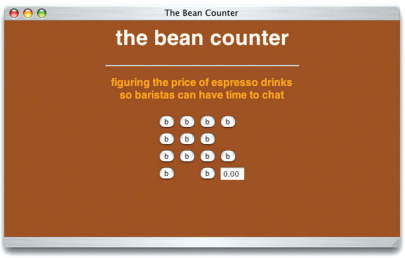
Delete the Two Buttons

- In row 2, cell 4, and row 4, cell 2, remove the <input… />
  - These cells must be empty
  - Cell can be empty but must still be surrounded by <td></td> tags to be a cell

Insert the Text Box

- Name the text box **"price"** because that's the information that will be printed
- Window should be 5 characters wide because no combination of drink inputs will result in a price of more than 4 digits plus decimal point
- onChange needs a placeholder
- Button in row 4, cell 4, should be replaced by

```
<input type="text" name="price" value="0.00" size="5" onChange=' ' />
```

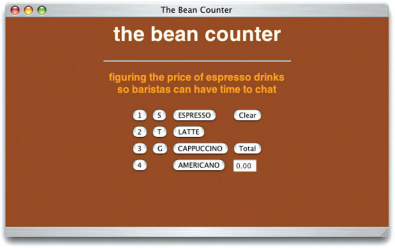
(b)

Figure 19.5. (continued). Intermediate stages in the construction of the Bean Counter interface: (b) after Step 3

(continues next page).

Label the Buttons

- Pass through cells and change the value attribute of each button from "b" to its proper label
- First column is number of shots (1, 2, 3, 4)
- Second column is sizes (S, T, G)
- Third column is the drinks (ESPRESSO, LATTE, CAPPUCCINO, AMERICANO)
- It is easiest to work row-wise rather than column-wise in HTML tables



(c)

Figure 19.5. ***(continued).*** *Intermediate stages in the construction of the Bean Counter interface: (c) after Step 4.*

Primp the Interface

- Notice that the buttons are left-justified in the columns
  - Buttons in last two columns should be centered
  - To revise ESPRESSO entry:

```
<td align="center">
  <input type="button" value=" ESPRESSO " onClick=' ' />
</td>
```

- Give table a background color

```
<table bgcolor="#993300">
```

Primp the Interface (cont'd)

- Add a border around the table as a whole
  - Make a big one-cell table and make our table the table data for that new table

```
<table border="2">
  <tr>
    <td>
      /* Existing table goes here */
    </td>
  </tr>
</table>
```

- Outline the price box with red

```
<td bgcolor="red">
```


Figure 19.3 Web interface for the Bean Counter program displayed using (a) Safari and (b) Firefox.

Event-Based Programming

- Something should happen as each button is clicked—only in response to user-caused events
- Define in JavaScript the actions to be performed when each button is clicked

The onClick Event Handler

- **onClick** is the event-handling attribute for the Total button
- Insert the price computation code inside the quotes for the onClick attribute
- It becomes the onClick event handler


```
<td>
<input type = button value = "Total" onClick =
'   var price;
    var taxRate = 0.088;
    if (drink == "espresso")
        price = 1.40;
    if (drink == "latte" || drink == "cappuccino") {
        if (ounce == 8)
            price = 1.95;
        if (ounce == 12)
            price = 2.35;
        if (ounce == 16)
            price = 2.75;
    }
    if (drink == "Americano")
        price = 1.20 + .30 * (ounce/8);
    price = price + (shots - 1) * .50;
    price = price + price * taxRate;
//one more assignment is required here
'>
</td>
```

Figure 19.6. The Total text box input element with the price computation inserted as the event handler. (Notice that the three temporary declarations of Figure 19.1 have been removed, as has the temporary "alert" command.)

Click Event

- When the barista clicks on the Total button, it causes a *click event* in the browser
- The browser looks for the *onClick* event handler in the Total button input tag
- Browser runs those instructions, then waits for the next event

Shots Button

- In each case, ask what action should be performed when a particular button is clicked
- For the first column, specify number of shots
  - Clicking on the 1 button should cause shots variable to have value 1

```
<td>
  <input type="button" value="1" onClick='shots = 1' />
</td>
```

  - Clicking on the 2 button assigns shots the value 2, etc.

Size and Drink Buttons

- Size buttons assign the ounce variable the appropriate value: 8, 12, or 16

```
<td>
  <input type="button" value=" S " onClick='ounce = 8' />
</td>
```

- Drink variable gets the name of the drink quoted

```
<td align="center">
  <input type="button" value=" ESPRESSO "
        onClick=' drink = "espresso" ' />
```

- Single quotes surround the assignment statement, which uses double quotes
  - Remember that string literals are case-sensitive

Clear Button and Initializations

- Clicking on Clear button should reset all variables to their initial values
- We have not declared or initialized those variables yet
- Place declarations at start of JavaScript body, enclosed in <script> tags

```
<script language="JavaScript">
    var shots = 1;
    var drinks = "none";
    var ounce = 0;
</script>
```

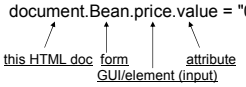
Clear Button and Initializations (cont'd)

- Clear button makes the same assignments as the initialization statements for each variable

```
<td>
   <input type="button" value="Clear" onClick='
      shots = 1; drink = "none"; ounce = 0;
      document.Bean.price.value = "0.00" ' />
</td>
```


Referencing Data across Controls

- When we want a statement in one element to change a value in another element, we must tell the browser how to navigate among the elements
- Use the *dot operator*
  - Provides a means of navigation to the proper object
    - object.property selects the property of the object
    - document.Bean.price.value = "0.00"



Changing the Window

- Input elements are for both input and output
  - When the value is reassigned, the window displays the new value, acting as output
- Displaying the Total
  - Total event handler outputs price

```
document.Bean.price.value = price;
```

Critiquing the Bean Counter

- Numbers versus Money
  - Final price is shown as number, not currency
  - Use Math.round to round to two decimal places
  - Trailing zeros will be dropped, but we won't worry about it
- Organization
  - The design and organization makes sense for its application
- Feedback
  - There is no feedback about current settings of variables
  - We could add a window above each column giving the current setting

Recap of the Bean Counter Application

- Referencing variables
  - Placed variable declarations inside the <script> and </script> tags
  - Referenced data as variables *local* to a handler (taxRate), as variables *global* to all handlers (drink), and as a variable/attribute *in another element* (document.Bean.price.value)

Recap of the Bean Counter Application (cont'd)

- Program and Test
  - Incremental process
  - Produced minimal HTML program, and tested
  - Added skeleton table, and tested
  - Improved table one feature at a time and tested
  - Wrote JavaScript to solve one event handler at a time
    - No complex tasks
    - Continual testing spots errors immediately

Recap of the Bean Counter Application (cont'd)

- Assess the Program Design
  - Critiquing how well the solution fulfilled the need for which it was written
  - Software should perfectly match the solution requirements