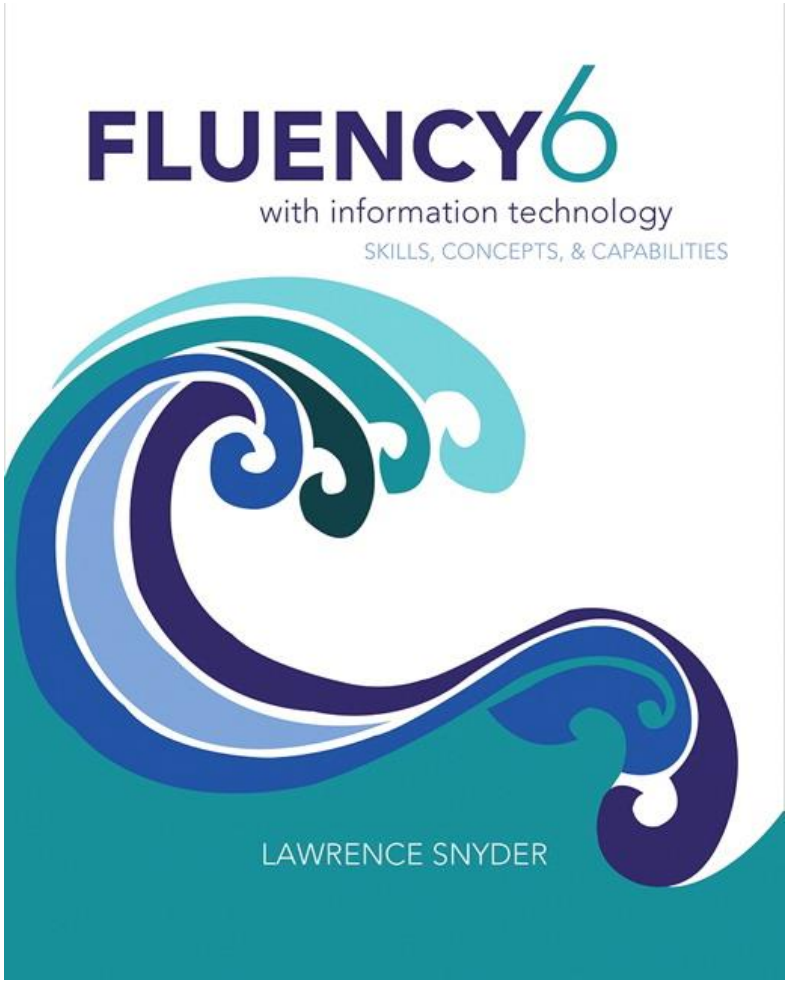


Chapter 18

A JavaScript Program

Table of Contents

Learning Objectives

- Use the Bean Counter application as a model to do the following:
  - Write input elements
  - Create a button table
  - Write an event handler in JavaScript
  - Produce a UI similar to that of the Bean Counter
- In chapter 17, we just built the computation part of the Bean counter and now we want to have an interactive version of the Bean counter
- Trace the execution of the Bean Counter, saying what output is produced by a given input
- Explain event-based programming in JavaScript and the use of event handlers

Basic JavaScript Syntax

JAVASCRIPT STRUCTURE

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>Hello</title>
   </head>
   <body>
     <div>
         <p>regular body text</p>
     </div>
     <script>
        alert("hello");
     </script>
   </body>
</html>
```

JavaScript is case SENSITIVE!

```
<html>
  <head>
     <title>Simple Page</title>
  </head>
  <body>
    <div id="main">
       <h1>Today's Headline</h1>
       <p>Regular paragraph text goes here.</P>
    </div>
  </body>
</html>
```

```
alert("Hello world");
Alert("Hello world");
```

```
alert("Hello world"); alert("Another message");
```

```
alert("Hello world");
alert("Another message");
```

```
do really really really really really really comple
```

. . . . . .

. . .

```
do really really really
really really really complex calculation;
```

JAVASCRIPT WILL FORGIVE SOME OMISSIONS

```
alert("Hello world");
```

JAVASCRIPT IS WHITESPACE INSENSITIVE

```
alert("Hello world");
alert  (  "Hello world"  )  ;
alert
(  "Hello world"
);
alert("Hello      world");
```

JAVASCRIPT COMMENTS

```
// this is a comment
alert("hello"); // can go here

/* this is a
   multiple
   line
   comment
*/
```

JAVASCRIPT STRUCTURE

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>Hello</title>
   </head>
   <body>
     <div>
         <p>regular body text</p>
     </div>
     <script>
        alert("hello");
     </script>
   </body>
</html>
```

simple.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
     <title>Hello</title>
  </head>
  <body>
   <div>
     <p>regular body text</p>
   </div>
   <script src="myscript.js">
   </script>
  </body>
</html>
```

myscript.js

```
alert("hello");
```

보조자료

LOCATION OF THE SCRIPT TAG

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Hello</title>
    <script src="myscript.js"></script>
  </head>
  <body>
    <div>
       <p>regular body text</p>
    </div>
  </body>
</html>
```


LOCATION OF THE SCRIPT TAG

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Hello</title>
  </head>
  <body>
    <script src="myscript.js"></script>
    <div>
       <p>regular body text</p>
    </div>
  </body>
</html>
```

보조자료

Data Values

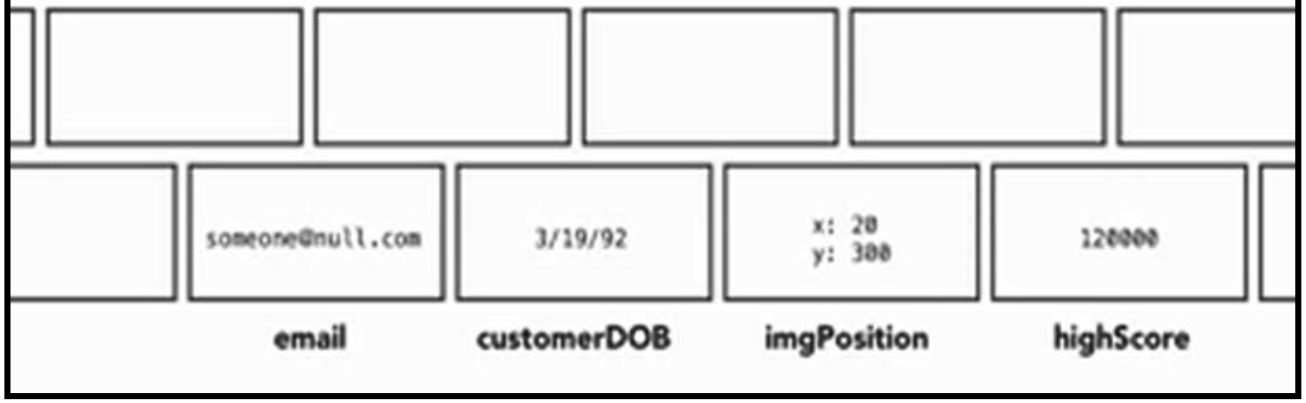
someone@null.com

3/19/92

x: 30
y: 200

1200000

VARIABLES



CREATING VARIABLES

```
var year;
var customerEmail;
var todaysDate;
var foo;
var x;
var 99problems; ✘
var problems99; ✔
```

letters
numbers
_
$

PEARSON

CREATING VARIABLES

```
var year;
```

CREATING VARIABLES

```
var year;
year = 2011;
```

CREATING VARIABLES

```
var year = 2011;
```

VARIABLE NAMES ARE CASE SENSITIVE

```
var x = 200;

X = 210;
```

MULTIPLE VARIABLES

```
var year, month, day;
```

MULTIPLE VARIABLES WITH VALUES

```
var year = 2011, month = 10, day = 31;
```

MULTIPLE VARIABLES WITH VALUES

```
var year = 2011;
var month = 10;
var day = 31;
```

VARIABLE DATA TYPES

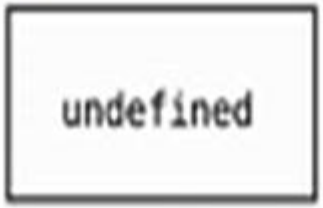
```
var myVariable;
```

myVariable

integer?
floating point number?
text string? boolean value? array? date? object?

VARIABLE DATA TYPES

```
var myVariable;
```



myVariable

VARIABLE DATA TYPES

```
var myVariable;
```

myVariable

```
myVariable = 200;
```

VARIABLE DATA TYPES

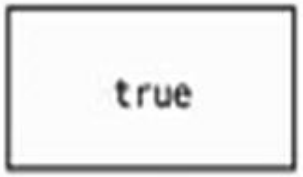
```
var myVariable;
```

myVariable

```
myVariable = 200;
myVariable = "Hello";
```

VARIABLE DATA TYPES

```
var myVariable;
```



myVariable

```
myVariable = 200;
myVariable = "Hello";
myVariable = 'Hello';
myVariable = true;
```

VARIABLE DATA TYPES

```
var myVariable;
```

myVariable

```
myVariable = 200;
myVariable = "Hello";
myVariable = 'Hello';
myVariable = true;
myVariable = false;
```

JavaScript Reserved Words

보조자료

JavaScript Reserved Words		
break	export	return
case	for	switch
comment	function	this
continue	if	typeof
default	import	var
delete	in	void
do	label	while
else	new	with

Java Keywords (Reserved by JavaScript)		
abstract	implements	protected
boolean	instanceOf	public
byte	int	short
char	interface	static
double	long	synchronized
false	native	throws
final	null	transient
float	package	true
goto	private	

ECMAScipt Reserved Words		
catch	enum	throw
class	extends	try
const	finally	
debugger	super	

Other JavaScript Keywords		
alert	isFinite	personalbar
Anchor	isNan	Plugin
Area	Java	print
arguments	JavaArray	prompt
Array	JavaClass	prototype
assign	JavaObject	Radio
blur	JavaPackage	ref
Boolean	length	RegExp
Button	Link	releaseEvents
callee	location	Reset
caller	Location	resizeBy
captureEvents	locationbar	resizeTo
Checkbox	Math	routeEvent
clearInterval	menubar	scroll
clearTimeout	MimeType	scrollbars
close	moveBy	scrollBy
closed	moveTo	scrollTo
confirm	name	Select
constructor	NaN	self
Date	navigate	setInterval
defaultStatus	navigator	setTimeout
document	Navigator	status
Document	netscape	statusbar
Element	Number	stop
Element	Number	stop
escape	Object	String
eval	onBlur	Submit
FileUpload	onError	sun
find	onFocus	taint
focus	onLoad	Text
Form	onUnload	Textarea
Frame	open	toolbar
Frames	opener	top
Function	Option	toString
getClass	outerHeight	unescape
Hidden	OuterWidth	untaint
history	Packages	unwatch
History	pageXoffset	valueOf
home	pageYoffset	watch
Image	parent	window
Infinity	parseFloat	Window
InnerHeight	parseInt	
InnerWidth	Password	

ARITHMETIC OPERATORS

```
+   -   *   /

score += 10;

+=   -=   *=   /=
```

ASSIGNMENT

```
result = a + b;
```

OPERATOR PRECEDENCE

```
result = 5 + 5 * 10;

 55       5 +   50
```

```
result = (5 + 5) * 10;

 100        10    * 10
```

CONDITIONAL CODE

```
if ( condition ) {
  // code goes here
  // ...
}
```

TERMINOLOGY

(parentheses)

[brackets]

{ braces }

CONDITIONAL CODE

```
if (   a < 50   ) {
  // code goes here
  // ...
}
```

CONDITIONAL CODE

```
if (  d != 100 ) {
  // code goes here
  // ...
}
```

CONDITIONAL CODE

```
if (            ) {
   // code goes here
   // ...
}
```

code block

CONDITIONAL CODE

```
if (            ) {
   // code goes here
   // ...
} else {
      // otherwise, different code
      if (        ) {
         // nested if
      }
}
```

EQUALITY

```
if (a == b) {

    // execute this code

}
```

ASSIGNMENT INSTEAD OF EQUALITY

```
var a = 5;
var b = 10;

if (a = b) {

    // always true!

}
```

OPERATORS WITH =

=	assignment
==	equality
===	strict equality

COMPARISON

Relational Operators

```
if (a == b) { ...
if (a != b) { ...
if (a === b) { ...
if (a !== b) { ...
if (a > b) { ...
if (a < b) { ...
if (a >= b) { ...
if (a <= b) { ...
```

LOGICAL AND / OR

Logical Operators

```
if (a === b                    ) { ...
if (a === b  ||  c === d ) { ...
if ( (a > b) &&  (c < d) ) { ...
if (
      (a > b)
        &&
      (c < d)    ) { ...
```

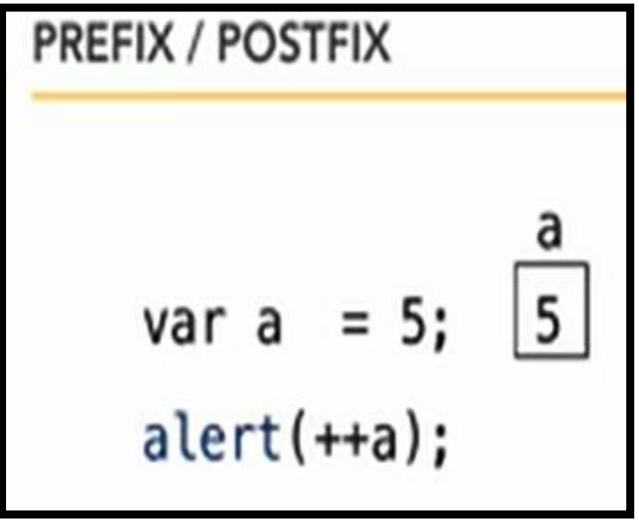
MODULUS

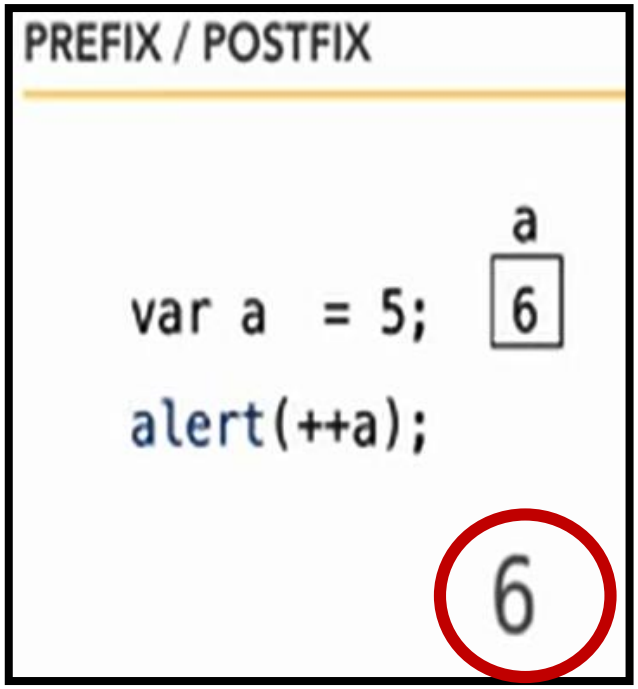
Modulo Operator

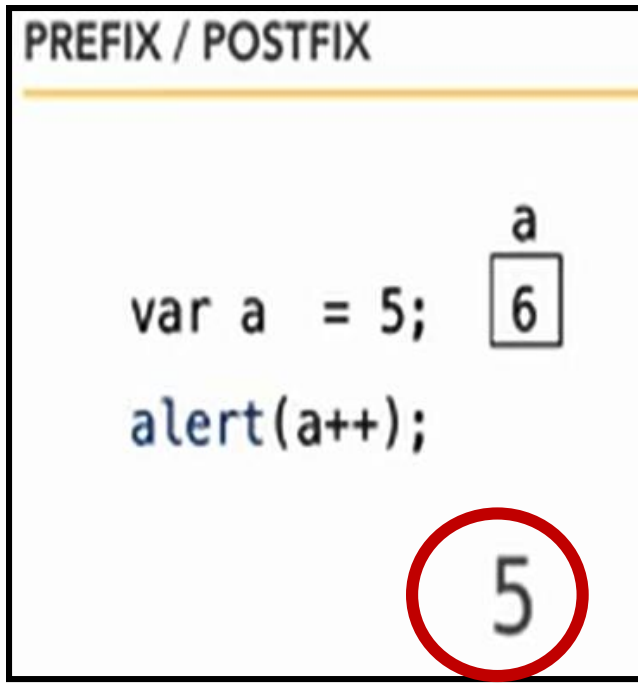
```
var year = 2003;
var remainder = year % 4;  // remainder is 3
```

INCREMENT / DECREMENT

```
a = a + 1;        a = a - 1;
a += 1;             a -= 1;
a++;                   a--;
++a;                   --a;
```


PREFIX / POSTFIX
a
var a = 5;
5
alert(++a);


PREFIX / POSTFIX
a
var a = 5;
6
alert(++a);
6


PREFIX / POSTFIX
a
var a = 5;
6
alert(a++);
5

TERNARY

```
condition ? true : false
```

TERNARY OPERATOR EXAMPLE

```
var playerOne = 500;
var playerTwo = 600;


// alternatively...  condition ? true : false
var highScore =  (playerOne > playerTwo) ? playerOne : playerTwo ;
```

TERNARY OPERATOR EXAMPLE

```
var playerOne = 500;
var playerTwo = 600;


// sometime later
var highScore;


if (playerOne > playerTwo) {
    highScore =  playerOne;
}
else {
    highScore = playerTwo;
}
```

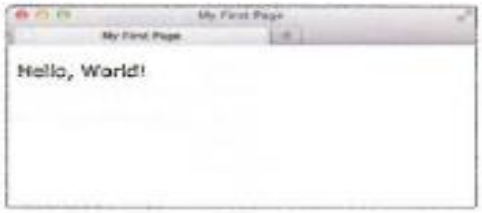
Preliminaries: reminiscing HTML

- HTML files are made of ASCII text
- Avoid word processor formatting since it confuses browsers
- We'll use the basic text editor of Chapter 4 for our JavaScript development
  - File format must be text or txt
  - File extension must be html
  - Operating system knows that an html file will be processed by a browser
- The Web page (written in HTML) is the GUI and the JavaScript code does the computing
- The Bean Counter is the GUI for the Expresso Pricing Program in JavaScript program in chap 17

Creating your JavaScript

- Open your starterPage.html in a text editor (Notepad2)

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>My First Page</title>
  </head>
  <body>
    <p>Hello, World!</p>
  </body>
</html>
```



- We can enclose the JavaScript code within <body> and </body> as follows:

  <script type="text/javascript">

  …

  </script>

- Time to test your program?
  - Save it with the file named, bean.html
  - Find the file on your computer and open it
  - JavaScript will run with HTML

Bean Counter v.0

```
<!doctype html>
<html>
 <head>
  <meta charset="UTF-8">
  <title>Bean Counter</title>
 </head>
 <body>
  <h2> Confirming that bean.html works </h2>
  <script>
    var drink = "latte";
    var ounce = 12;
    var shots = 2;
    var taxRate = 0.088;
    var price;
    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 + 0.30*(ounce/8);
    price = price + (shots - 1)*.50;
    price = price + price*taxRate;
    alert(price);
  </script>
 </body>
</html>
```

(a)

(b)

Figure 18.1 Version 0 of the Bean Counter program without the user interface, and with fixed inputs: (a) the HTML and JavaScript code, and (b) running in the Firefox browser.

Review of HTML Basics

- Remember the starterPage.html in chapter 4!
- Change the title to <title>The Bean Counter</title>
  - Replace “Hello, World” with an <h1> heading that reads “the bean counter”
- Put the Bean counter code inside the body
- Then you will get beanV0.html
- Add a global <style> section in the head
- Add an <hr /> and Next, add the paragraph below

 figuring the price of espresso drinks so baristas can have time to chat.”

- Use a
 to split the paragraph above over 2 lines
- Use saddlebrown as the background color, darkorange as the font color, and helvetica as the font family
- Make the heading appear in white
- Style the horizontal line to be 50% shorter than the full window
- Then you will get beanV1.html

So Far, So Good?

Bean Counter v0


```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8"/>
    <title>The Bean Counter</title>
    <style type="text/css">
        body {background-color : saddlebrown; color : darkorange;
              font-family : helvetica; text-align : center}
        hr   {width:50%; color: darkorange}
        h1   {color : white;}
    </style>
  </head>
  <body>
    <h1> the bean counter</h1>
    <hr/>
    <p><b>figuring the price of espresso drinks<br />
          so baristas can have time to chat</b></p>
  </body>
</html>
```

Bean Counter v1

Figure 18.3 The Bean Counter interface to this point, and the HTML that produced it.

What We're Aiming for …

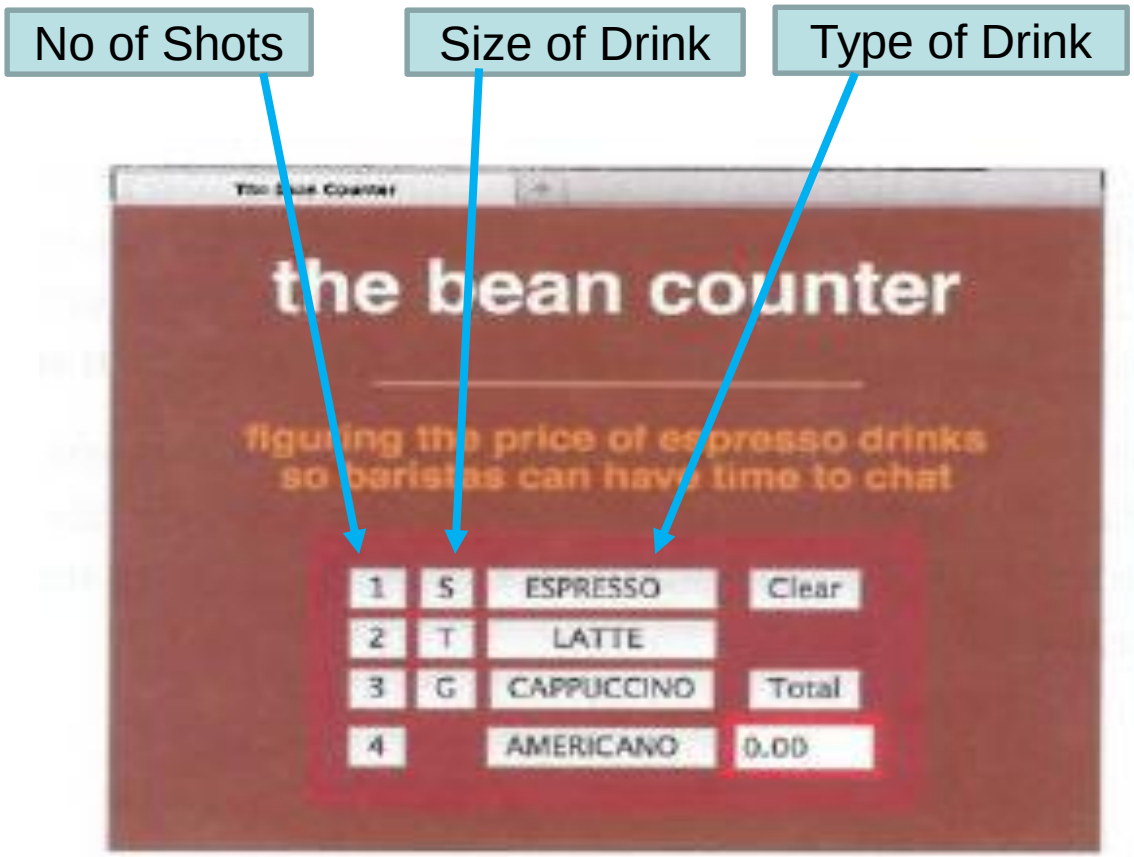


Figure 18.2 The Web interface for the Bean Counter program as it will appear when it is complete.

Interacting with a GUI

- Ordering products or answering survey questions involve user-interactions
- When the user puts some data into the form, the captured data is need to be sent to the Javascript code within HTML code or the backend-server computer

- HTML form tags <form name= “xxx” > … </form> are for user-interaction in GUI
- Various input facilities like text, buttons, checkboxes are provided with various input tag: <input ….. />
- HTML form tags are associated with various input tags

- Event Driven Programming: Event and Event Handlers
  - Users make mouse-click event and text-changing event, then the event handler (corresponding JacaScript code) are supposed to be executed

Form Tag

참고자료

- Browser에서 입력된 데이터를 capture해서 JavaScript code혹은 backend server로 보내기 위해서 사용하는 tag

- The form tags are used for grouping the input tags
- "esp"라는 이름의 form은 <input ….> 을 둘러싸고 있으며 <input…> 는 disp.value의 값을 바뀔때마다 2자리로 표현함

```
<form name= " esp" >   <input type= " text "  id= " disp "  value= " 0"  size="2" />    </form>
```

- 아래 3개의 button은 form "esp"와 연관된 button들이고 mouse-click이 되면 disp.value를 바꿔주고, disp.value가 바뀌면 바뀐값이 화면에 보여진다

```
<button form="esp" onclick='disp.value = 1' > 1 </button>
<button form="esp" onclick='disp.value = 2' > 2 </button>
<button form="esp" onclick='disp.value = 3' > 3 </button>
```

3 Input Element Types of the Input Tag

- Text Box

<input type="text“ id=“필드이름" value=“초기값" size=“n“ onchange="event_handler " />

  - identifier is the name of the element / value is the text to be placed in the box
  - onchange means that if the contents of the box(eg.value) is changed, the event handler's JavaScript instructions are executed

<input type="text“ id=“eg" value=“Initial Entry" size=“10“ onchange=“…" />

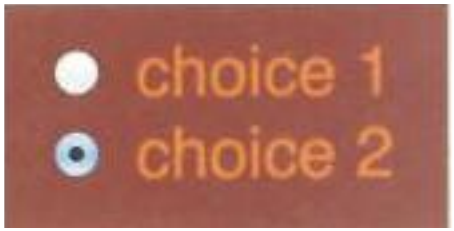
* Value값을 사용자가 변경하던, Javascript code가 변경하던 변경만되면 TextBox에 새값이 보여진다

- Radio Button

<input type="radio" name=“버튼이름“ onclick="event_handler " /> label text

  - identifier is the name of the element
  - label text is shown beside the button

<input type="radio" name=“pick“ onclick=“…" /> choice 1
<input type="radio" name=“pick“ onclick=“…" /> choice 2



3 Input Element Types of the Form tag

- Button

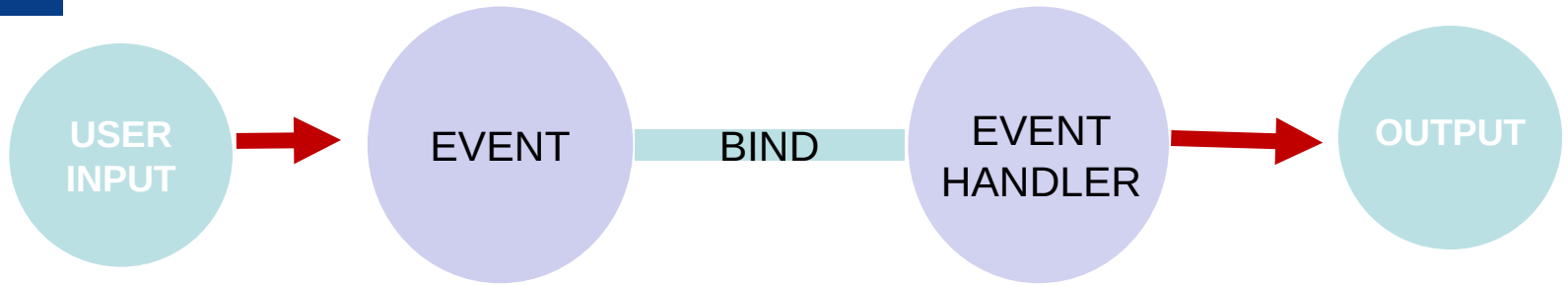
<input type="button" value= " 버튼이름" onclick="event_handler (JavaScript code)" />

  - value gives the text to be printed on the button
  - onclick means when the button is clicked, JavaScript's event handler is executed

<input type="button" value="Click Me" onclick="…." />

- 3가지 input element type들 이외에도 Browser에서 사용자가 입력하는 데이터를 capture하는 여러가지 방식을 input tag는 지원함
  - Ex: Check-Box, Passsword, File-upload, etc

User Interaction: Events and binds



- Events: 어플리케이션에 발생하는 user interaction
  - Mouse click, Button click, Mouse dragging, Key-board typing
- Event handler: Event가 발생시 호출되는 응용 프로그램
- Binding
  - event가 발생할 때, 어플리케이션이 event handler을 호출하여 준비하도록 연결해주는 것

Events and Event Handlers

- When the GUI inputs are used they cause an event to occur
- Buttons have a "click event" (as in you click the mouse to select the button)
  - An event is an indication from the computer (operating system) that something just happened (such as mouse click, user type in)
- When JavaScript "finds out" about the event, it runs a piece of program called the event handler
  - An event handler is the program that responds to the click
- Event관련 tag에는 JavaScript코드로 된 Event Handler 를 묻는다!

```
<button form="esp" onclick = 'disp.value = 1'  > 1 </button>
<input type="text" id="textTempC" size="4"
        onchange = " textTempF.value= convertC2F(textTempC.value) )" />
```

Creating the Graphical User Interface

- All that remains is to create a table and fill in the entries
- Make sure to place the table between the form tags to ensure that the browser understands the inputs
  - The table is a four-row, four-column table with two empty cells
  - Buttons appear in all of the occupied cells but one
- Table is mostly a table of buttons
- The incremental steps for building the table :
  - Create a button table
  - Delete two buttons
  - Insert text box
  - Label the buttons
  - Primp the interface

What We're Aiming for …

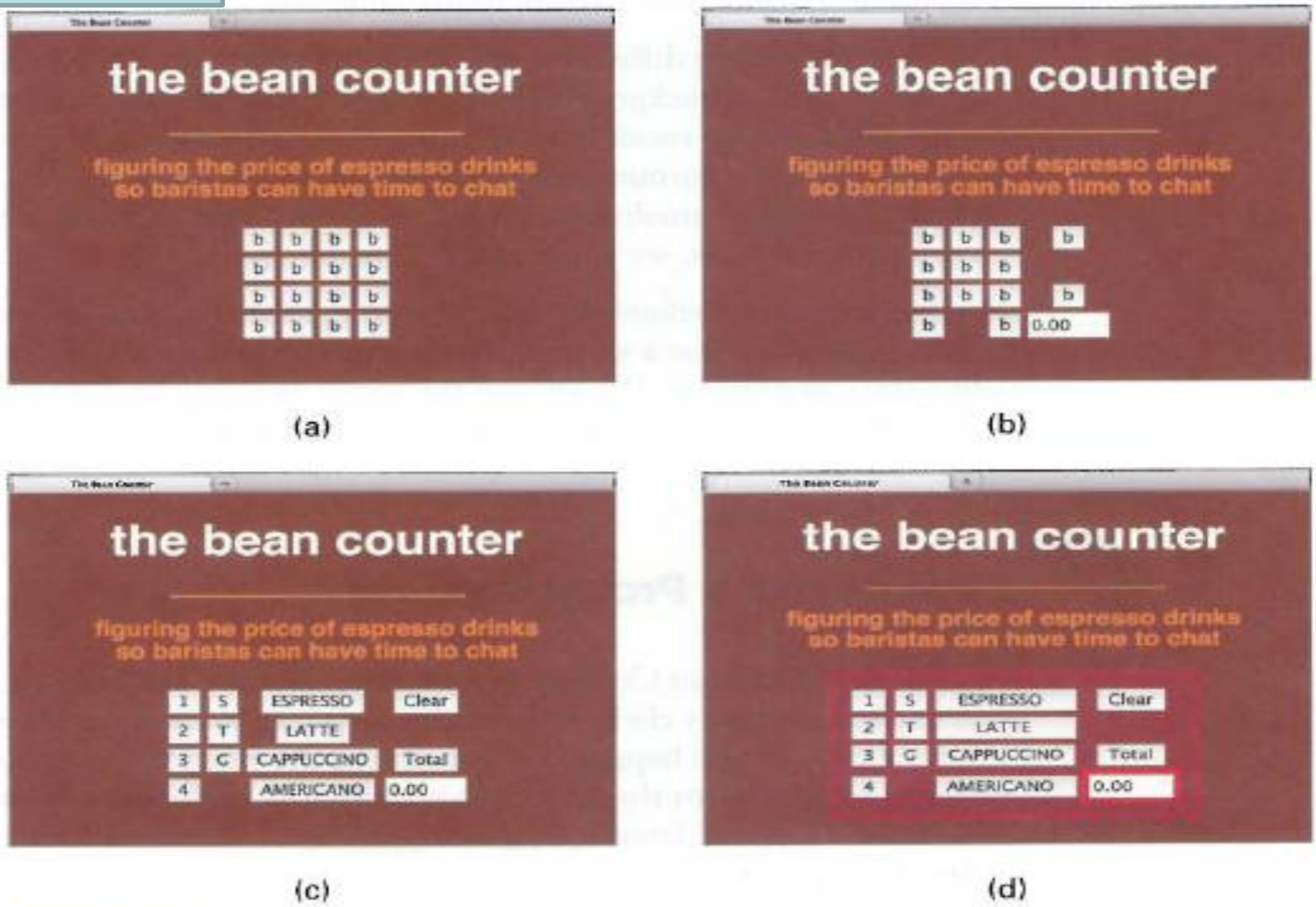


Figure 18.4 Intermediate stages in the construction of the Bean Counter interface: (a) after Step 1, (b) after Step 3, (c) after Step 4, and (d) final form.

1. Create a Button Table

In HTML, <td> < button form='esp' onclick = ' ' /> b </button> </td>

- "b" is a placeholder for the button label
- ' ' is a placeholder for the JavaScript text of the event handler

In Style, table {margin-left : auto; margin-right : auto; text-align }

- The table is centered by specifying that the left and right margins should be automatically positioned

2. Delete Two Buttons

- In row 2, cell 4, and row 4, cell 2, remove the button tage <input. . . /> because these cells must be empty
- Cells can be empty, but they still need to be surrounded with <td> and </td> tags: i.e. <td> </td>

3. Insert Text Box

```
<form name=“esp”>
  <input type="text"   id=“disp"    value="0.00"   size="5“   onchange=' ‘  />
</form>
```

- Replacing the button tag to a text input tag for making a text box in the lower right corner is needed
- Name the text box “disp" because we want to display the total price “value”
- Window is “5” characters wide since no drink inputs will result in a price of more than 4 digits plus the decimal point
- Change the onclick event handler to onchange
- We get Figure 18.4.(b)

4. Label the Buttons

- Next, go through each of the 13 table cells and change the value attribute of each button from “b” to its proper button label
  - First column is the number of shots (1, 2, 3, 4)
  - Second column is the drink sizes (S, T, G)
  - Third column is the type of drinks (Espresso, Latte, Cappuccino, Americano)
  - The two items in the last column are the controls: (Clear, Total)

Ex. <td> <button form=‘esp’ onclick = ' '/> b </button> </td>

➔ <td> <button form=‘esp’ onclick = ' '/> Latte </button> </td>

- We get Figure 18.4.(c)

5. Primp the Interface

- The buttons would look better if they were all the same width
- Simply add spaces using , which stands for non-breaking space
- Espresso vs Cappuccino

```
<td> <button form="esp" onclick='drink = "cappuccino";'>
      CAPPUCCINO   ; </button></td>
<td> <button form="esp" onclick='drink = "espresso";'>
        ESPRESSO       ; </button></td>
```

- Give the table a background color such as background-color: #993300
- Add a border to the table styling with a medium solid line colored firebrick
- Add 8 pixels of padding to the buttons
- Add a medium red border to the price text box
- Finally we get Figure 18.4 (d) (still 먹통!)

What We're Aiming for …

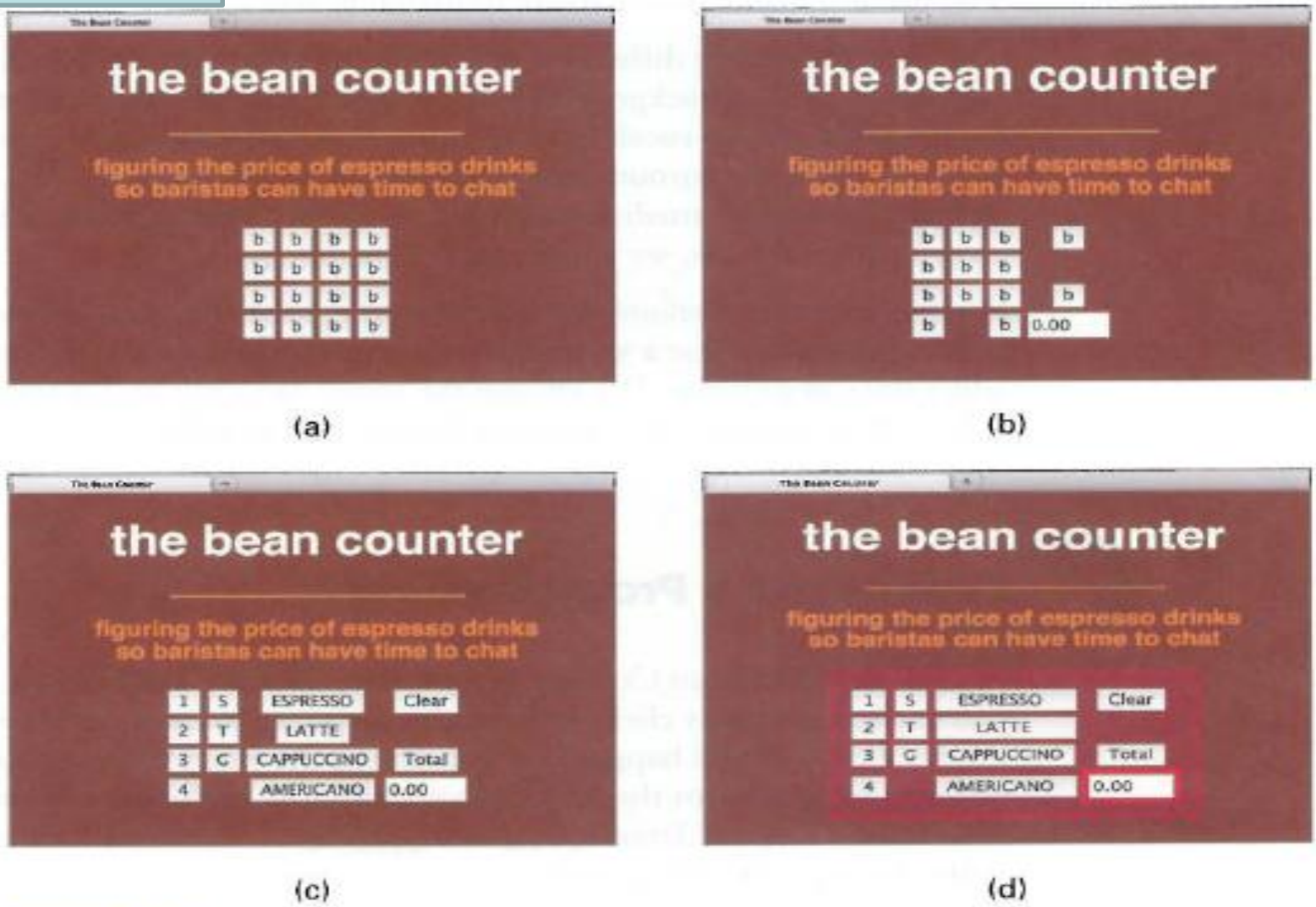


Figure 18.4 Intermediate stages in the construction of the Bean Counter interface: (a) after Step 1, (b) after Step 3, (c) after Step 4, and (d) final form.

Add JavaScript Code using Event-Based Programming

- The Bean Counter program should behave like a calculator
  - Each time a button is clicked (user-caused event), something should happen
- Programming the Bean Counter application amounts to defining the actions (the onclick event handler by JavaScript code)

```
<td> <input type = "button" value = "Total" onclick =‘ …JavaScript code… '/> </td>
```

- We insert the price computation code inside the quotes for the onclick attribute in order to use JavaScript to calculate the price
- When the Total button is clicked, the browser activates the onclick event handler
- The browser runs those instructions, which implements the action, and then waits for the next event

```
<td><button form="esp"
    onclick='
        var price = -10;
        var taxRate = 0.087;
        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 statement needed here */
    '> Total </button></td>
```

Figure 18.5 The Total button tag with the price computation inserted as the event handler. (Notice that the three temporary declarations of Figure 18.1 have been removed, as has the temporary alert( ) command at the end.)

Shots Button

<td> <button form = ‘esp’ onclick = 'shots = 1' /> 1 </button> </td>

- The number of shots the customer requests is identified by which shot button is selected
- The 2 button assigns shots the value 2, etc.

Size Buttons

<td> < button form = ‘esp’ onclick = 'ounce = 8' /> S </button> </td>

- Action to be performed on a click event for the size buttons is to assign the appropriate value to the ounce variable

Drink Buttons

<td> <button form=“esp” onclick = ‘drink = “espresso”;’>
    ESPRESSO   ; </button></td>

- The drink is assigned as "espresso", not "ESPRESSO", as written on the button
- JavaScript code behaves as though there were 52 letters in the alphabet: 26 lowercase and 26 uppercase!

Initialization and Clear Button

- The initialization declaration should be placed at the beginning of the program just after the <body> tag

```
<script>  var shots = 1;  var drink = "none";  var ounce = 0;  </script>
```

- Clicking the Clear button resets all of the variables (drink, ounce, and shots) to their initial values

```
<td>  <input type = "button"  value = "Clear"
        onclick =  ‘ shots = 1; drink = "none";  ounce = 0;  disp.value = "0.00" ' />  </td>
```

- Initial value for shots is 1 (every espresso drink has at least one shot)
- Initial values for drink and ounce are chosen to be illegal values intentionally for the barista to receive an error message, indicating that an input is missing
- The Clear button should make these same assignments setting everything back to their initial values
- The disp.value = "0.00" statement places 0.00 in the price window

Referencing Data Across Inputs

```
<form name="esp">
  <input type="text"   id="disp"    value="0.00"   size="5"   onchange=' '/>
</form>
```

- Id is a global name
- In JavaScript, when one statement in one element to change a value in another element the browser must be told how to navigate among the elements
- Dot operator is used to aid in navigation between elements
  - object.property ➔ "property of object"
  - disp.value = "0.00" is read as "the value attribute of display is assigned 0.00."
- Whenever the value of disp.value is changed, the new value is newly written in the browser area associated with the input tag
- (Input tag의 value attribute가 reset되면 Input tag는 다시 실행된다!)

Displaying the Total

- The Total event handler must show the price as the output of “Total” button click
- It is similar to the Clear button event handler
- The final line of the Total event handler is

```
disp.value = Math.round(100 * price / 100).toFixed(2);
```

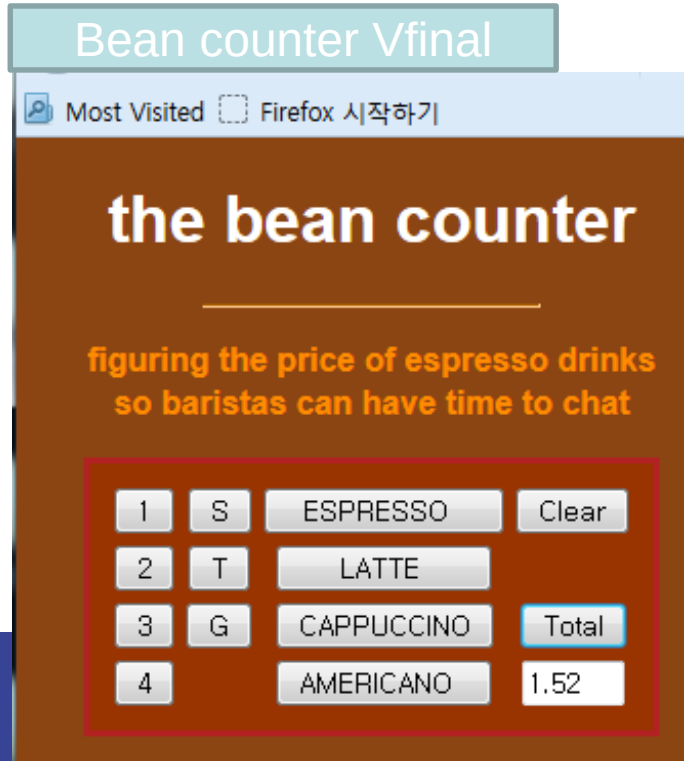
- The input tag having disp as Id is executed again

```
<form name=“esp”>
  <input type="text"   id=“disp"    value="0.00"   size="5“   onchange=' '/>
</form>
```

Referencing Variables

- Declarations are usually placed at the start of a program
- Place the declarations right after the <body> tag, inside the <script> and </script> tags
- An event handler of one element needs to place a value in the window of another element, it must describe how to navigate to the item it wants to change using dot operators
- The Bean Counter application illustrates 3 different ways to reference data values in an event-handling program:
  - as variables local to a handler (taxRate),
  - as variables global to all the handlers (drink, shot, ounce)
  - as a variable in another tag element (disp.value)

Now we get ➔



Bean counter Vfinal

```
<!doctype html>
<html>
   <head>
     <meta charset="UTF-8"/>
     <title>The Bean Counter</title>
     <script>
           var shots = 1;    var drink = "none";  var ounce = 0;
     </script>

   <style>
           body {background-color : saddlebrown; color : darkorange;
                     font-family : helvetica; text-align : center}
           hr       {width:50%; color: darkorange}
           h1      {color : white;}
           table {margin-left : auto; margin-right : auto; text-align : center;
                     background-color : #993300; border-style : solid;
                     border-color : firebrick; border-width : medium; padding : 8px }
     </style>

   </head>
   <body>
      <h1> the bean counter</h1>
      <hr/>
      <p><b>figuring the price of espresso drinks<br />
            so baristas can have time to chat</b></p>
      <table>

       <tr>
          <td><button form="esp" onclick='shots = 1' > 1 </button></td>
          <td><button form="esp" onclick='ounce = 8'> S </button></td>
          <td><button form="esp" onclick='drink = "espresso"'; >
                     ESPRESSO     </button></td>
          <td><button form="esp" onclick='
                 shots = 1;  drink = "none";
                 ounce = 0;  disp.value = "0.00"'> Clear </button></td>
        </tr>
```

Bean counter Vfinal

```
    <tr>
      <td><button form="esp" onclick='shots = 2'> 2 </button></td>
      <td><button form="esp" onclick='ounce = 12'> T </button></td>
      <td><button form="esp" onclick='drink = "latte"'>
                     LATTE         </button></td>
      <td></td>
    </tr>

    <tr>
      <td><button form='esp' onclick='shots = 3'> 3 </button></td>
      <td><button form="esp" onclick='ounce = 16'> G </button></td>
      <td><button form="esp" onclick='drink = "cappuccino"'> CAPPUCCINO </button> </td>
      <td><button form="esp"  onclick = '
      var price = -10;  var taxRate = 0.087;
      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;
      disp.value = (Math.round(100*price)/100).toFixed(2);
      '>  Total  </button></td>
    </tr>

    <tr>
      <td><button form="esp"  onclick='shots = 4'> 4 </button></td>
      <td></td>
      <td><button form="esp" onclick='drink = "Americano"'>  AMERICANO   </button> </td>
      <td>  <form name="esp"> <input type="text" id="disp" value="0.00" size="5" /> </form> </td>
    </tr>
  </table>
</body>
</html>
```

Critiquing the Bean Counter

- Every design must be critiqued to ensure that it meets the requirements:
  - Did it solve the problem?
  - Can it be improved?
- Experiment with the Bean Counter application to see how well it works.
- The most important thing! ➔ Does the design fulfill the barista's needs?

- Design Issues
  - Number vs Money
  - Organization of Buttons
  - Feedback

Numbers vs Money

- The final price is shown as a decimal number with several decimals not as currency with only two decimal points:

  $3.745 ➔ $3.75

  $4 ➔ $4.00

- disp.value = (Math.round(price*100)/100).toFixed(2);
- The built-in JavaScript function: Math.round( )
- If we want to round the 3rd position of a fraction: (Math.round(price*100)/100)
- Rational number class has several built-in functions including toFixed() which is representing the rational number upto 2 digits of fraction

Organization of Buttons

- The organization of the buttons is generally consistent with how the application will be used
- Espresso drinks are typically named with syntax of “how many shots?”, “what size?”, and “what kind of drink?” “double tall latte”

Feedback

- The form does not give a barista any feedback about the current settings of the variables
- There should always be feedback for every operation.
- Adding a window above each column of buttons that gives the current setting might be helpful

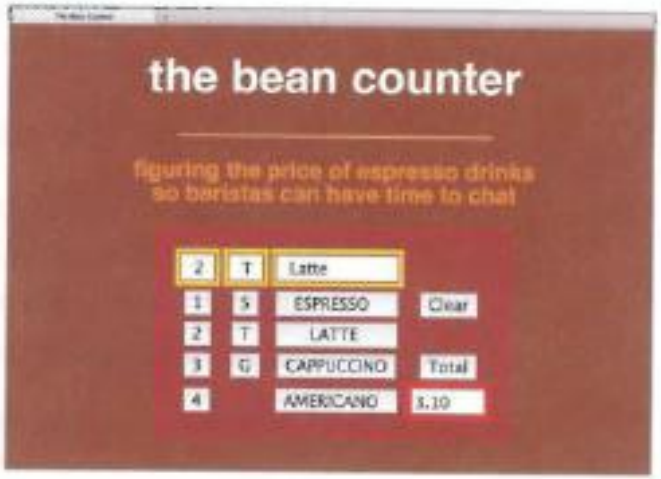
Bean counter with echo row

```
<tr>
 <td class="echo"><input type="text" form="esp"
    id="shotpic" value=" " size="1"/></td>
 <td class="echo"><input type="text" form="esp"
    id="sizepic" value=" " size="1"/></td>
 <td class="echo"><input type="text" form="esp"
    id="coffee" value=" " size="10"/></td>
 <td> </td>
</tr>
```

(a)

```
<tr>
 <td><button form="esp" onclick='shots = 1;
    shotpic.value=" 1"' > 1 </button></td>
 <td><button form="esp" onclick='ounce = 8;
    sizepic.value=" S"'> S </button></td>
 <td><button form="esp" onclick='drink = "espresso";
    coffee.value=" Espresso "'>
      ESPRESSO    </button></td>
 <td><button form="esp" onclick='
  shots = 1;
  drink = "none";
  ounce = 0;
  disp.value = "0.00"
  shotpic.value= " "; sizepic.value=" "; coffee.value=" ";
  '> Clear </button></td></tr>
```

(b)



Id 는 global variable!
Id로 지정된 variable.value가 바뀌면 화면에 보여지는 variable.value도 바뀜

Figure 18.6 The Bean Counter application improved to give the barista feedback: (a) the <button> tags for the first row, and (b) showing how the revised button event handlers assign the barista's choice to the proper text window.

Bean counter V BeanWithJuliette

Figure 18.7 Final Bean Counter page with improvements: (a) as it loads, and (b) in use.

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8"/>
    <title>The Bean Counter</title>
    <script>
        var shots = 1;
        var drink = "none";
        var ounce = 0;
    </script>
    <style>
        body {background-color : saddlebrown; color : darkorange;
              font-family : helvetica; text-align : center}
        hr    {width:50%; color: darkorange}
        h1    {color : white;}
        table {margin-left : auto; margin-right : auto; text-align : center;
               background-color : #993300; border-style : solid;
               border-color : firebrick; border-width : medium; padding : 8px }
        td.tot, td.echo {border-style : solid; border-width : medium; }
        td.tot {border-color : red;}
        td.echo {border-color:gold;}
        select {color:saddlebrown; text-align:center;   }
    </style>
  </head>
  <body>
    <h1> the bean counter</h1>
    <hr/>
    <p><b>figuring the price of espresso drinks<br />
          so baristas can have time to chat</b></p>
```

```
<table>
 <tr><td colspan="4" style="text-align:left;">
  <form name="emp">
  <select id="employee" >
  <option value="-">-Sign In-</option>
  <option value="C"> Charlie </option>
  <option value="J"> Juliette </option>
  <option value="M"> Mike </option>
  <option value="O"> Oscar </option>
  <option value="R"> Romeo </option>
  <option value="V"> Victor </option>
  </select> Is Pulling For Us
   </form>
   </td>
</tr>

 <tr>
  <td class="echo"><input type="text" form="esp" id="shotpic" value=" " size="1"/></td>
  <td class="echo"><input type="text" form="esp" id="sizepic" value=" " size="1"/></td>
  <td class="echo"><input type="text" form="esp" id="coffee" value=" " size="10"/></td>
  <td> </td>
</tr>

 <tr>
   <td><button form="esp" onclick='shots = 1; shotpic.value="  1"' > 1 </button></td>
   <td><button form="esp" onclick='ounce = 8; sizepic.value="  S"'> S </button></td>
   <td><button form="esp" onclick='drink = "espresso"; coffee.value="  Espresso  "'>
        ESPRESSO     </button></td>
  <td><button form="esp" onclick='
   shots = 1;
   drink = "none";
   ounce = 0;
   disp.value = "0.00"
   shotpic.value= " "; sizepic.value=" "; coffee.value=" ";
    '> Clear </button></td>
 </tr>
```

Pull Down Menu (혹은 Combo Box)를 구성하는 select tag와 option tag

```
<tr>
 <td><button form="esp"  onclick='shots = 2; shotpic.value="  2" '> 2 </button></td>
 <td><button form="esp" onclick='ounce = 12; sizepic.value="  T"  '> T </button></td>
 <td><button form="esp" onclick='drink = "latte"; coffee.value="  Latte     " '>
            LATTE         </button></td>
 <td></td>
</tr>

<tr>
 <td><button form='esp' onclick='shots = 3; shotpic.value = "  3" '> 3 </button></td>
 <td><button form="esp" onclick='ounce = 16; sizepic.value= "  G"'> G </button></td>
 <td><button form="esp" onclick='drink = "cappuccino";  coffee.value=" Cappucino "'>
       CAPPUCCINO  </button>
 </td>
 <td><button form="esp"
     onclick='
       var price = -10;
       var taxRate = 0.087;
       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;
       disp.value = (Math.round(100*price)/100).toFixed(2);
     '>  Total  </button></td>
</tr>
```

```
    <tr>
      <td><button form="esp"  onclick='shots = 4; shotpic.value = "  4"'> 4 </button></td>
      <td></td>
      <td><button form="esp" onclick='drink = "Americano"; coffee.value = "  Americano"'>
            AMERICANO   </button>
      </td>
      <td class="tot">
        <form name="esp">
        <input type="text" id="disp" value="0.00" size="5" />
        </form>
      </td>
    </tr>
  </table>

</body>
```

Bean Counter Recap

- Program and Test
  - Incremental approach (breaking the task into tiny pieces!)
  - Begin by producing a minimal 19-line HTML program, Improve one feature at a time and test as you go
  - Write and solve one event handler at a time (sometimes reuse similar event handlers!)
  - Continual testing meant that we immediately knew where any errors were located…the newly added code!

- Assess the Program Design
  - the program design vs the programming
  - a critique of how well our solution solved the problem (did it fulfill the barista's needs?)
  - This is an important part of any design effort, but especially so for software

Summary

- Used HTML to set up a context in which event handlers perform the actual work
  - The setup involved placing buttons and other input elements on a Web page, so a user could enter data and receive results
  - This is the input /output part of the application and it is principally written in HTML

- Wrote JavaScript code for the event handlers
  - This is the processing part of the application. We used the event-based programming style and the basic instructions of Chapter 17
  - This style will be used throughout the rest of the book
  - HTML will simply be the input / output part of a program written in JavaScript