APPENDIX C

Answers to Tutorial Exercises

This appendix provides answers to the tutorial exercises that appear in Part II of this book (Chapters 4 through 12).

Chapter 4 Answers

1. The music jukebox (a) and temperature calculator (d) are good client-side JavaScript applications. Even though the jukebox relies on server storage of the music files, you can create a more engaging and responsive user interface of buttons, swappable images, and information from a plug-in, such as LiveAudio. The temperature calculator is a natural, because all processing is done instantaneously on the client, rather than having to access the server for each conversion.

The Web site visit counter (b) that accumulates the number of different visitors to a Web site is a server-side CGI application, because the count must be updated and maintained on the server. At best, a client-side counter could keep track of the number of visits the user has made to a site and report to the user how many times he or she has been to the site. The storage would require scripting the cookie (Chapter 16). A chat room application (c) done properly requires server facilities to open up communication channels among all users connected simultaneously. Client-side scripting by itself cannot create a live chat environment.

2. The first task is to determine a valid identifier for the General Motors location in the hierarchy. Then “connect the dots”:

a. General_Motors.Chevrolet.Malibu

b. General_Motors.Pontiac.Firebird

c. General_Motors.Pontiac.Bonneville

3. a. Valid, because it is one contiguous word. InterCap spelling is fine.

b. Valid, because an underscore character is acceptable between words.

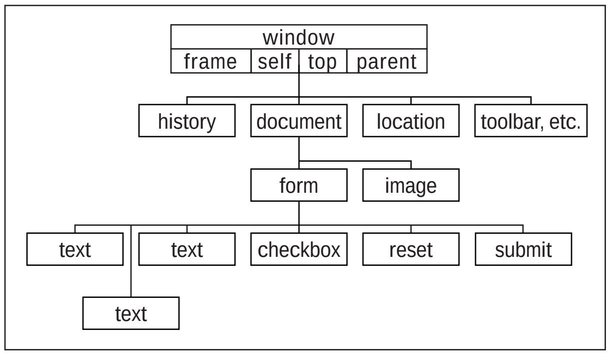
c. Not valid, because an identifier cannot begin with a numeral.

d. Not valid, because no spaces are allowed.

e. Not valid, because apostrophes and most other punctuation are not allowed.

4. The names I assign here are arbitrary, but the paths are not.

```
document.myPicture
document.entryForm
document.entryForm.nameField
document.entryForm.addressField
document.entryForm.phoneField
document.entryForm.noArchiveBox
```



5. `<INPUT TYPE="button" NAME="Hi" VALUE="Howdy" onClick="alert('Hello to you, too!')">`

Chapter 5 Answers

1.
```
<SCRIPT LANGUAGE="JavaScript">
<!--
document.write("Hello, world.")
// -->
</SCRIPT>
```

2.
```
<HTML>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
<!--
document.write("Hello, world.")
// -->
</SCRIPT>
</BODY>
</HTML>
```

3.
```
<HTML>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
<!--
// write a welcome message to the world
document.write("Hello, world.")
// -->
</SCRIPT>
</BODY>
</HTML>
```

4. My answer is written so that both event handlers call separate functions. You can also have each event handler invoke the `alert()` method inline.

```
<HTML>
<HEAD>
<TITLE>An onLoad= script</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!--
function done() {
     alert("The page has finished loading.")
}
function alertUser() {
     alert("Ouch!")
}
// -->
</SCRIPT>
</HEAD>
<BODY onLoad="done()">
Here is some body text.
<FORM>
     <INPUT TYPE="button" NAME="oneButton" VALUE="Press Me!"
     onClick="alertUser()">
</FORM>
</BODY>
</HTML>
```

5. a. The page displays two text fields.

b. The user enters text into the first field and either clicks or tabs out of the field to trigger the onChange= event handler.

c. The function displays an all-uppercase version of one field into the other.

Chapter 6 Answers

1. a. Valid.

b. Not valid. The variable needs to be a single word, such as `howMany` or `how_many`.

c. Valid.

d. Not valid. The variable name cannot begin with a numeral. If the variable needs a number to help distinguish it from other similar variables, then put the numeral at the end: `address1`.

2. a. 4

b. 40

c. "4020"

d. "Robert"

3. The functions are `parseInt()` and `parseFloat()`. Strings to be converted are passed as parameters to the functions: `parseInt(document.forms[0].entry.value)`.

4. Both text field values are strings that must be converted to numbers before they can be arithmetically added together. You can use the `parseFloat()` functions either on the variable assignment expressions (for example, `var value1 = parseFloat(document.adder.inputA.value)`) or in the addition expression (`document.adder.output.value = parseFloat(value1) + parseFloat(value2)`).

5. Concatenate means to join together two strings to become one string.

Chapter 7 Answers

1. Because the references in the function point to a text field named `entry` inside a form named `entryForm`, be sure to assign those names to the `NAME` attributes in the respective HTML tags.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
var USStates = new Array(51)
USStates[0] = "Alabama"
USStates[1] = "Alaska"
USStates[2] = "Arizona"
USStates[3] = "Arkansas"
USStates[4] = "California"
USStates[5] = "Colorado"
USStates[6] = "Connecticut"
USStates[7] = "Delaware"
USStates[8] = "District of Columbia"
```

```
USStates[9] = "Florida"
USStates[10] = "Georgia"
USStates[11] = "Hawaii"
USStates[12] = "Idaho"
USStates[13] = "Illinois"
USStates[14] = "Indiana"
USStates[15] = "Iowa"
USStates[16] = "Kansas"
USStates[17] = "Kentucky"
USStates[18] = "Louisiana"
USStates[19] = "Maine"
USStates[20] = "Maryland"
USStates[21] = "Massachusetts"
USStates[22] = "Michigan"
USStates[23] = "Minnesota"
USStates[24] = "Mississippi"
USStates[25] = "Missouri"
USStates[26] = "Montana"
USStates[27] = "Nebraska"
USStates[28] = "Nevada"
USStates[29] = "New Hampshire"
USStates[30] = "New Jersey"
USStates[31] = "New Mexico"
USStates[32] = "New York"
USStates[33] = "North Carolina"
USStates[34] = "North Dakota"
USStates[35] = "Ohio"
USStates[36] = "Oklahoma"
USStates[37] = "Oregon"
USStates[38] = "Pennsylvania"
USStates[39] = "Rhode Island"
USStates[40] = "South Carolina"
USStates[41] = "South Dakota"
USStates[42] = "Tennessee"
USStates[43] = "Texas"
USStates[44] = "Utah"
USStates[45] = "Vermont"
USStates[46] = "Virginia"
USStates[47] = "Washington"
USStates[48] = "West Virginia"
USStates[49] = "Wisconsin"
USStates[50] = "Wyoming"

var stateEntered = new Array(51)
stateEntered[0] = 1819
stateEntered[1] = 1959
stateEntered[2] = 1912
stateEntered[3] = 1836
stateEntered[4] = 1850
stateEntered[5] = 1876
stateEntered[6] = 1788
stateEntered[7] = 1787
stateEntered[8] = 0000
```

```
stateEntered[9] = 1845
stateEntered[10] = 1788
stateEntered[11] = 1959
stateEntered[12] = 1890
stateEntered[13] = 1818
stateEntered[14] = 1816
stateEntered[15] = 1846
stateEntered[16] = 1861
stateEntered[17] = 1792
stateEntered[18] = 1812
stateEntered[19] = 1820
stateEntered[20] = 1788
stateEntered[21] = 1788
stateEntered[22] = 1837
stateEntered[23] = 1858
stateEntered[24] = 1817
stateEntered[25] = 1821
stateEntered[26] = 1889
stateEntered[27] = 1867
stateEntered[28] = 1864
stateEntered[29] = 1788
stateEntered[30] = 1787
stateEntered[31] = 1912
stateEntered[32] = 1788
stateEntered[33] = 1789
stateEntered[34] = 1889
stateEntered[35] = 1803
stateEntered[36] = 1907
stateEntered[37] = 1859
stateEntered[38] = 1787
stateEntered[39] = 1790
stateEntered[40] = 1788
stateEntered[41] = 1889
stateEntered[42] = 1796
stateEntered[43] = 1845
stateEntered[44] = 1896
stateEntered[45] = 1791
stateEntered[46] = 1788
stateEntered[47] = 1889
stateEntered[48] = 1863
stateEntered[49] = 1848
stateEntered[50] = 1890

function getStateDate() {
     var selectedState = document.entryForm.entry.value
     for ( var i = 0; i < USStates.length; i++) {
        if (USStates[i] == selectedState) {
           break
        }
     }
     alert("That state entered the Union in " + stateEntered[i] +
".")
}
```

```
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME="entryForm">
Enter the name of a state:
<INPUT TYPE="text" NAME="entry">
<INPUT TYPE="button" VALUE="Look Up Entry Date"
onClick="getStateDate()">
</BODY>
</HTML>
```

2. Several problems plague this function definition. Parentheses are missing from the first `if` construction's condition statement. Curly braces are missing from the second nested `if...else` construction. A mismatch of curly braces also exists for the entire function. The following is the correct form (changes and additions in boldface):

```
function format(ohmage) {
     var result
     if (ohmage >= 10e6) {
        ohmage = ohmage / 10e5
        result = ohmage + " Mohms"
     } else {
        if (ohmage >= 10e3) {
           ohmage = ohmage / 10e2
           result = ohmage + " Kohms"
        } else {
           result = ohmage + " ohms"
        }
     }
     alert(result)
}
```

3. Here is one possibility:

```
for (var i = 1; i < tomatoes.length; i++) {
     if (tomatoes[i].looks == "mighty tasty") {
        break
     }
}
var myTomato = tomatoes[i]
```

4. The new version defines a different local variable name for the dog.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
var aBoy = "Charlie Brown"   // global
var hisDog = "Snoopy"        // global
function sampleFunction() {
     // using improper design to demonstrate a point
     var WallacesDog = "Gromit"      // local version of hisDog
     var output = WallacesDog + " does not belong to " + aBoy +
".<BR>"
     document.write(output)
}
</SCRIPT>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
sampleFunction()       // runs as document loads
document.write(hisDog + " belongs to " + aBoy + ".")
</SCRIPT>
</BODY>
</HTML>
```

5. The application uses three parallel arrays and is structured very much like the solution to question 1. Learn to reuse code whenever you can.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
var planets = new Array(4)
planets[0] = "Mercury"
planets[1] = "Venus"
planets[2] = "Earth"
planets[3] = "Mars"

var distance = new Array(4)
distance[0] = "36 million miles"
distance[1] = "67 million miles"
distance[2] = "93 million miles"
distance[3] = "141 million miles"

var diameter = new Array(4)
diameter[0] = "3100 miles"
diameter[1] = "7700 miles"
diameter[2] = "7920 miles"
diameter[3] = "4200 miles"

function getPlanetData() {
     var selectedPlanet = document.entryForm.entry.value
     for ( var i = 0; i < planets.length; i++) {
        if (planets[i] == selectedPlanet) {
           break
        }
     }
     var msg = planets[i] + " is " + distance[i] + " from the Sun
and "
     msg += diameter[i] + " in diameter."
     document.entryForm.output.value = msg
}
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME="entryForm">
Enter the name of a planet:
<INPUT TYPE="text" NAME="entry">
<INPUT TYPE="button" VALUE="Look Up a Planet"
onClick="getPlanetData()">
```

```
<BR>
<INPUT TYPE="text" SIZE=70 NAME="output">
</BODY>
</HTML>
```

Chapter 8 Answers

1. a. Close, but no cigar. Array references are always plural: `window.document.forms[0]`.

b. Not valid: `self` refers to a window and `entryForm` must refer to a form. Where's the `document`? It should be `self.document.entryForm.entryField.value`.

c. Valid.

d. Not valid. The `document` reference is missing from this one.

e. Valid, assuming that `newWindow` is a variable holding a reference to a subwindow.

2. `window.status = "Welcome to my Web page."`

3. `document.write("<H1>Welcome to my Web page.</H1>")`

4. A script in the Body portion invokes a function that returns the text entered in a `prompt()` dialog box.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
function askName() {
     var name = prompt("What is your name, please?","")
     return name
}
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
document.write("Welcome to my web page, " + askName() + ".")
</SCRIPT>
</BODY>
</HTML>
```

5. The URL can be derived from the location object.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
function showLocation() {
     alert("This page is at: " + location.href)
}
</SCRIPT>
</HEAD>
<BODY onLoad="showLocation()">
```

```
Blah, blah, blah.
</BODY>
</HTML>
```

Chapter 9 Answers

1. For Listing 9-1, pass the text object because that's the only object involved in the entire transaction.

```
<HTML>
<HEAD>
<TITLE>Text Object value Property</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function upperMe(field) {
     field.value = field.value.toUpperCase()
}
</SCRIPT>
</HEAD>
<BODY>
<FORM onSubmit="return false">
<INPUT TYPE="text" NAME="convertor" VALUE="sample"
onChange="upperMe(this)">
</FORM>
</BODY>
</HTML>
```

For Listing 9-2, the button invokes a function that communicates with a different element in the form. Pass the form object.

```
<HTML>
<HEAD>
<TITLE>Checkbox Inspector</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function inspectBox(form) {
     if (form.checkThis.checked) {
         alert("The box is checked.")
     } else {
         alert("The box is not checked at the moment.")
     }
}
</SCRIPT>
</HEAD>
<BODY>
<FORM>
<INPUT TYPE="checkbox" NAME="checkThis">Check here<BR>
<INPUT TYPE="button" VALUE="Inspect Box"
onClick="inspectBox(this.form)">
</FORM>
</BODY>
</HTML>
```

For Listing 9-3, again the button invokes a function that looks at other elements in the form. Pass the form object.

```
<HTML>
<HEAD>
<TITLE>Extracting Highlighted Radio Button</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function fullName(form) {
     for (var i = 0; i < form.stooges.length; i++) {
        if (form.stooges[i].checked) {
           break
        }
     }
     alert("You chose " + form.stooges[i].value + ".")
}
</SCRIPT>
</HEAD>

<BODY>
<FORM>
<B>Select your favorite Stooge:</B>
<INPUT TYPE="radio" NAME="stooges" VALUE="Moe Howard" CHECKED>Moe
<INPUT TYPE="radio" NAME="stooges" VALUE="Larry Fine" >Larry
<INPUT TYPE="radio" NAME="stooges" VALUE="Curly Howard" >Curly<BR>
<INPUT TYPE="button" NAME="Viewer" VALUE="View Full Name..."
onClick="fullName(this.form)">
</FORM>
</BODY>
</HTML>
```

For Listing 9-4, all action is triggered by and confined to the select object. Pass only that object to the function.

```
<HTML>
<HEAD>
<TITLE>Select Navigation</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function goThere(list) {
     location = list.options[list.selectedIndex].value
}
</SCRIPT>
</HEAD>

<BODY>
<FORM>
Choose a place to go:
<SELECT NAME="urlList" onChange="goThere(this)">
     <OPTION SELECTED VALUE="index.html">Home Page
     <OPTION VALUE="store.html">Shop Our Store
     <OPTION VALUE="policies">Shipping Policies
     <OPTION VALUE="http://www.yahoo.com">Search the Web
</SELECT>
</FORM>
</BODY>
</HTML>
```

2. This requires a bit of surgery. The Submit button is replaced with a standard button whose `VALUE` attribute is set to "Submit." The button's `onClick=` event handler calls the `checkForm()` function, which performs the validation. If an empty field exists, the function must return to bail out of the loop. Because the event handler is not expecting any returned value, you can simply issue the `return` statement to stop the function altogether. If all the tests pass, then the form is submitted with the `submit()` method. Functions that have a `return` statement inside an `if` construction must also have a `return` statement outside the construction so it always returns a value (including the null value used here). The other change is that the `onSubmit=` event handler has been removed from the `<FORM>` tag, because it is no longer needed (the `submit()` method does not trigger an `onSubmit=` event handler).

```
<HTML>
<HEAD>
<TITLE>Validator</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function checkForm(form) {
     for (var i = 0; i < form.elements.length; i++) {
        if (form.elements[i].value == "") {
           alert("Fill out ALL fields.")
           return
        }
     }
     form.submit()
     return
}
</SCRIPT>
</HEAD>

<BODY>
<FORM>
Please enter all requested information:<BR>
First Name:<INPUT TYPE="text" NAME="firstName" ><BR>
Last Name:<INPUT TYPE="text" NAME="lastName" ><BR>
Rank:<INPUT TYPE="text" NAME="rank" ><BR>
Serial Number:<INPUT TYPE="text" NAME="serialNumber" ><BR>

<INPUT TYPE="button" VALUE="Submit" onClick="checkForm(this.form)">
</FORM>
</BODY>
</HTML>
```

3. The `this` keyword refers to the text field object, so `this.value` refers to the `value` property of that object.

```
function showText(txt) {
     alert(txt)
}
```

4.
```
document.accessories.acc1.value = "Leather Carrying Case"
document.forms[1].acc.value = "Leather Carrying Case"
```

5. The select object invokes a function that does the job.

```
<HTML>
<HEAD>
<TITLE>Color Changer</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function setColor(list) {
     var newColor = list.options[list.selectedIndex].value
     document.bgColor = newColor
}
</SCRIPT>
</HEAD>

<BODY>
<FORM>
Select a background color:
<SELECT onChange="setColor(this)">
<OPTION VALUE="red">Stop
<OPTION VALUE="yellow">Caution
<OPTION VALUE="green">Go
</SELECT>
</FORM>
</BODY>
</HTML>
```

Chapter 10 Answers

1. Use `string.indexOf()` to see if the field contains the "@" symbol.

```
<HTML>
<HEAD>
<TITLE>E-mail Validator</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function checkAddress(form) {
     if (form.email.value.indexOf("@") == -1) {
        alert("Check the e-mail address for accuracy.")
        return false
     }
     return true
}
</SCRIPT>
</HEAD>

<BODY>
<FORM onSubmit="return checkAddress(this)">
Enter your e-mail address:
<INPUT TYPE="text" NAME="email" SIZE=30><BR>
<INPUT TYPE="submit">
</FORM>
</BODY>
</HTML>
```

2. Remember that the substring goes up to, but does not include, the index of the second parameter. Spaces count as characters.

```
myString.substring(0,3)      // result = "Net"
myString.substring(13,18)    // result = "gator"
myString.substring(4,12)     // result = "cape Nav"
```

3. The missing `for` loop is in boldface. You could also use the increment operator on the `count` variable (`++count`) to add 1 to it for each letter "e."

```
function countE(form) {
     var count = 0
     var inputString = form.mainstring.value.toUpperCase()
     for (var i = 0; i < inputString.length; i++) {
        if (inputStr.charAt(i) == "e") {
           count += 1
        }
     }
     alert("The string has " + count + " instances of the letter
e.")
}
```

4. The formula for the random throw of one die is in the chapter.

```
<HTML>
<HEAD>
<TITLE>E-mail Validator</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function roll(form) {
     form.die1.value = Math.round(Math.random() * 5) + 1
     form.die2.value = Math.round(Math.random() * 5) + 1
}
</SCRIPT>
</HEAD>

<BODY>
<FORM>
<INPUT TYPE="text" NAME="die1" SIZE=2>
<INPUT TYPE="text" NAME="die2" SIZE=2><BR>
<INPUT TYPE="button" VALUE="Roll the Dice"
onClick="roll(this.form)">
</FORM>
</BODY>
</HTML>
```

5. If you used the `Math.round()` method in your calculations, that is fine for your current exposure to the Math object. Another method, `Math.ceil()`, may be more valuable because it rounds up any fractional value.

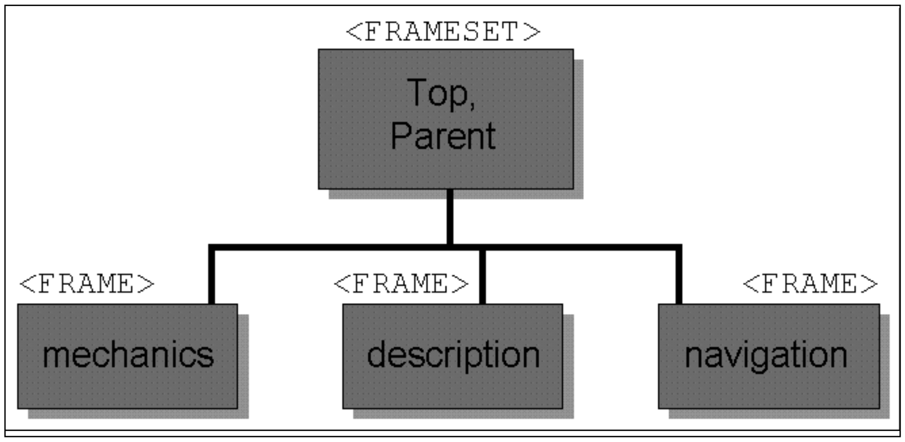
```
<HTML>
<HEAD>
<TITLE>Waiting for Santa</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function daysToXMAS() {
     var oneDay = 1000 * 60 * 60 * 24
```

```
          var today = new Date()
          var XMAS = new Date("December 25, 1998")
          var diff = XMAS.getTime() - today.getTime()
          return Math.ceil(diff/oneDay)
     }
     </SCRIPT>
     </HEAD>

     <BODY>
     <SCRIPT LANGUAGE="JavaScript">
     document.write(daysToXMAS() + " days until next Christmas.")
     </SCRIPT>
     </BODY>
     </HTML>
```

Chapter 11 Answers

1. `onLoad="parent.currCourse = 'history101'"`
2.



3. All three frames are siblings, so references include the parent.

```
parent.mechanics.location.href = "french201M.html"
parent.description.location.href = "french201D.html"
```

4. A script in one of the documents is attempting to reference the selector object in one of the frames but the document has not fully loaded, causing the object to not yet be in the browser's object model. Rearrange the script so that it fires in response to the `onLoad=` event handler of the framesetting document.

5. From the subwindow, the `opener` property refers back to the frame containing the `window.open()` method. To extend the reference to the frame's parent, the reference includes both pieces: `opener.parent.`*`ObjVarFuncName`*.

Chapter 12 Answers

1. A document image object is created by the `<IMG>` tag as the document loads. A memory image object is created with the `new Image()` constructor. Both objects have the same properties, and assigning a URL to the `src` property of a memory object loads the image into the browser's image cache.

2.
```
var janeImg = new Image(100,120)
janeImg.src = "jane.jpg"
```

3.
```
document.images["people"].src = janeImg.src
```

4. Surround `<IMG>` tags with link tags, and use the link's `onClick=`, `onMouseOver=`, and `onMouseOut=` event handlers. Set the image's `BORDER` attribute to zero if you don't want the link highlight to appear around the image.

✥ ✥ ✥