**1. Write a JavaScript program to convert degrees centigrade into degrees Fahrenheit, and to write the**

**result to the page in a descriptive sentence. The JavaScript equation for Fahrenheit to centigrade is**

**as follows:**

degFahren = 9 / 5 \* degCent + 32

**Solution:**

=======

<html>

<body>

<script language=”JavaScript” type=”text/javascript”>

var degCent = prompt(“Enter the degrees in centigrade”,0);

var degFahren = 9 / 5 \* degCent + 32;

document.write(degCent + “ degrees centigrade is “ + degFahren +

“ degrees Fahrenheit”);

</script>

</body>

</html>

You get the degrees centigrade the user wants to convert by using the prompt() function, and store it

inside the degCent variable.

You then do your calculation, which uses the data stored in degCent and converts them to Fahrenheit.

The result is assigned to the degFahren variable.

Finally, you write the results to the web page, building it up in a sentence using the concatenation operator +. Note how JavaScript knows in the calculation that degCent is to be treated as a number, but in the document.write() it knows that it should be treated as text for concatenation. So how does it know? Simple, it looks at the context. In the calculation, degCent is surrounded by numbers and numerical only operators, such as \* and /. In the document.write(), degCent is surrounded by strings, hence JavaScript assumes the + means concatenate.

**2. The following code uses the prompt() function to get two numbers from the user. It then adds those**

**two numbers together and writes the result to the page:**

<html>

<body>

<script language=”JavaScript” type=”text/javascript”>

var firstNumber = prompt(“Enter the first number”,””);

var secondNumber = prompt(“Enter the second number”,””);

var theTotal = firstNumber + secondNumber;

document.write(firstNumber + “ added to “ + secondNumber + “ equals “ +

theTotal);

</script>

</body>

</html>

**However, if you try the code out, you’ll discover that it doesn’t work. Why not?**

**Change the code so that it does work.**

**Solution:**

=======

The data that the prompt() actually obtains is a string. So both firstNumber and secondNumber contain

text that happens to be number characters. When you use the + symbol to add the two variables

together, JavaScript assumes that since it’s string data, you must want to concatenate the two together

and not sum them.

To make it explicit to JavaScript that you want to add the numbers together, you need to convert the data to numbers using the parseFloat() function.

<html>

<body>

<script language=”JavaScript” type=”text/javascript”>

var firstNumber = parseFloat(prompt(“Enter the first number”,””));

var secondNumber = parseFloat(prompt(“Enter the second number”,””));

var theTotal = firstNumber + secondNumber;

document.write(firstNumber + “ added to “ + secondNumber + “ equals “ + theTotal);

</script>

</body>

</html>

Now the data returned by the prompt() function are converted to floating-point numbers before being

stored in the firstNumber and secondNumber variables. Then, when we do the addition that is stored

in theTotal, JavaScript makes the correct assumption that, because both the variables are numbers, we

must mean to add them up and not concatenate them.

The general rule is that where we have expressions with only numerical data, the + operator means do

addition. If there are any string data, the + will mean concatenate.

**3. A junior programmer comes to you with some code that appears not to work. Can you spot where he went wrong? Give him a hand and correct the mistakes.**

var userAge = prompt(“Please enter your age”);

if (userAge = 0);

{

alert(“So you’re a baby!”);

}

else if ( userAge < 0 | userAge > 200)

alert(“I think you may be lying about your age”);

else

{

alert(“That’s a good age”);

}

**Solution:**

=======

Oh dear, our junior programmer is having a bad day! There are two mistakes on the following line:

if (userAge = 0);

First, he has only one equals sign instead of two in the if’s condition, which means userAge will be

assigned the value of 0 rather than userAge being compared to 0. The second fault is the semicolon at the end of the line—statements such as if and loops such as for and while don’t require semicolons. The

general rule is that if the statement has an associated block (that is, code in curly braces) then no semicolon is needed. So the line should be as follows:

if (userAge == 0)

The next fault is with these lines:

else if ( userAge < 0 | userAge > 200)

alert(“I think you may be lying about your age”);

else

The junior programmer’s condition is asking if userAge is less than 0 OR userAge is greater than 200.

The correct operator for a Boolean OR is ||, but the programmer has only used one |.

else if ( userAge < 0 || userAge > 200)

{

alert(“I think you may be lying about your age”);

}

else

**4. Using document.write(), write code that displays the results of the 12 times table. Its output should**

**be the results of the calculations.**

12 \* 1 = 12

12 \* 2 = 24

12 \* 3 = 36

.....

12 \* 11 = 132

12 \* 12 = 144

**Solution:**

=======

<html>

<body>

<script language=”JavaScript” type=”text/javascript”>

var timesTable = 12;

var timesBy;

for (timesBy = 1; timesBy < 13; timesBy++)

{

document.write(timesTable + “ \* “ + timesBy + “ = “ + timesBy \* timesTable +

“<br>”);

}

</script>

</body>

</html>

You use a for loop to calculate from 1 \* 12 up to 12 \* 12. The results are written to the page with document.write(). What’s important to note here is the effect of the order of precedence; the concatenation operator (the +) has a lower order of precedence than the multiplication operator, \*. This means that the timesBy \* timesTable is done before the concatenation, which is the result you want. If this were not the case, you’d have to put the calculation in parentheses to raise its order of precedence.

**5. Change the code of Question 4 so that it’s a function that takes as parameters the times table required and the values at which it should start and end. For example, you might try the four times table displayed starting with 4 \* 4 and ending at 4 \* 9.**

**Solution:**

=======

<html>

<body>

<script language=”JavaScript” type=”text/javascript”>

function writeTimesTable(timesTable, timesByStart, timesByEnd)

{

for (;timesByStart <= timesByEnd; timesByStart++)

{

document.write(timesTable + “ \* “ + timesByStart + “ = “ +

timesByStart \* timesTable + “<br>”);

}

}

writeTimesTable(4,4,9);

</script>

</body>

</html>

You’ve declared your function, calling it writeTimesTable(), and given it three parameters. The first is

the times table you want to write, the second is the start point, and the third is the number it should go

up to.

You’ve modified your for loop. First you don’t need to initialize any variables, so the initialization part

is left blank—you still need to put a semicolon in, but there’s no code before it. The for loop continues

while the timesByStart parameter is less than or equal to the timesByEnd parameter. You can see that,

as with a variable, you can modify parameters—in this case, timesByStart is incremented by one for

each iteration through the loop.

The code to display the times table is much the same. For the function’s code to be executed, you now

actually need to call it, which you do in the following line:

writeTimesTable(4,4,9);

This will write the four times table, starting at 4 times 4 and ending at 9 times 4.

**6. Modify the code of Question 5 to request the times table to be displayed from the user; the code should continue to request and display times tables until the user enters -1. Additionally, do a check to make sure that the user is entering a valid number; if the number is not valid, ask her to re-enter it.**

**Solution:**

=======

<html>

<body>

<script language=”JavaScript” type=”text/javascript”>

function writeTimesTable(timesTable, timesByStart, timesByEnd)

{

for (;timesByStart <= timesByEnd; timesByStart++)

{

document.write(timesTable + “ \* “ + timesByStart + “ = “ +

timesByStart \* timesTable + “<br>”);

}

}

var timesTable;

while ( (timesTable = prompt(“Enter the times table”,-1)) != -1)

{

while (isNaN(timesTable) == true)

{

timesTable = prompt(timesTable + “ is not a valid number, please retry”,-1);

}

if (timesTable == -1)

{

break;

}

document.write(“<br>The “ + timesTable + “ times table<br>”);

writeTimesTable(timesTable,1,12);

}

</script>

</body>

</html>

The function remains the same, so let’s look at the new code. The first change from Question 3 is that

you declare a variable, timesTable, and then initialize it in the condition of the first while loop. This

may seem like a strange thing to do at first, but it does work. The code in parentheses inside the while

loop’s condition

(timesTable = prompt(“Enter the times table”,-1))

is executed first because its order of precedence has been raised by the parentheses. This will return a

value, and it is this value that is compared to -1. If it’s not –1, then the while condition is true, and the

body of the loop executes. Otherwise it’s skipped over, and nothing else happens in this page.

In a second while loop nested inside the first, you check to see that the value the user has entered is

actually a number using the function isNaN(). If it’s not, then you prompt the user to try again, and this

will continue until a valid number is entered.

If the user had entered an invalid value initially, then in the second while loop she may have entered –1,

so following the while is an if statement that checks to see if -1 has been entered. If it has, you break

out of the while loop; otherwise the writeTimesTable() function is called.

**7. Using the Date object, calculate the date 12 months from now and write this into a web page.**

**Solution:**

=======

<html>

<body>

<script language=”JavaScript” type=”text/javascript”>

var months = new Array(“Jan”,”Feb”,”Mar”,”Apr”,”May”,”Jun”,”Jul”,”Aug”,“Sep”,”Oct”,”Nov”,”Dec”);

var nowDate = new Date();

nowDate.setMonth(nowDate.getMonth() + 12);

document.write(“Date 12 months ahead is “ + nowDate.getDate());

document.write(“ “ + months[nowDate.getMonth()]);

document.write(“ “ + nowDate.getFullYear());

</script>

</body>

</html>

Because the getMonth() method returns a number between 0 and 11 for the month rather than its

name, an array called months has been created that stores the name of each month. You can use

getMonth() to get the array index for the correct month name.

The variable nowDate is initialized to a new Date object. Because no initial value is specified, the new

Date object will contain today’s date.

To add 12 months to the current date you simply use setMonth(). You get the current month value with

getMonth(), and then add 12 to it.

Finally you write the result out to the page.

**8. Obtain a list of names from the user, storing each name entered in an array. Keep getting another name until the user enters nothing. Sort the names in ascending order, and then write them out to the page, with each name on its own line.**

**Solution:**

=======

<html>

<body>

<script language=”JavaScript” type=”text/javascript”>

var inputName = “”;

var namesArray = new Array();

while ( (inputName = prompt(“Enter a name”,””)) != “” )

{

namesArray[namesArray.length] = inputName;

}

namesArray.sort();

var namesList = namesArray.join(“<br>”)

document.write(namesList);

</script>

</body>

</html>

First you declare two variables: inputName, which will hold the name entered by the user, and

namesArray, which holds an Array object that stores each of the names entered.

You use a while loop to keep getting another name from the user as long as the user hasn’t left the

prompt box blank. Note that the use of parentheses in the while condition is essential. By placing the

following code inside parentheses, you ensure that this is executed first and that a name is obtained

from the user and stored in the inputName variable.

(inputName = prompt(“Enter a name”,””))

Then you compare the value returned inside the parentheses—whatever was entered by the user—

with an empty string (denoted by “”). If they are not equal—that is, if the user did enter a value, you

loop around again.

Now, to sort the array into order, you use the sort() method of the Array object.

namesArray.sort();

Finally, to create a string containing all values contained in the array elements with each being on a new

line, you use the HTML <br> tag and write the following:

var namesList = namesArray.join(“<br>”)

document.write(namesList);

The code namesArray.join(“<br>”) creates the string of array elements with a <br> between each.

Finally, you write the string into the page with document.write().

**9. Create a page with a number of links. Then write code that fires on the window onload event, displaying the href of each of the links on the page.**

**Solution:**

=======

<html>

<head>

<script language=”JavaScript” type=”text/javascript”>

function displayLinks()

{

var linksCounter;

for (linksCounter = 0; linksCounter < document.links.length; linksCounter++)

{

alert(document.links[linksCounter].href);

}

}

</script>

</head>

<body onload=”displayLinks()”>

<A href=”link0.htm” >Link 0</A>

<A href=”link1.htm”>Link 2</A>

<A href=”link2.htm”>Link 2</A>

</body>

</html>

You connect to the window object’s onload event handler by adding an attribute to the <body> tag.

<body onload=”displayLinks()”>

On the onload event firing, this will run the script in quotes calling the displayLinks() function.

In this function you use a for loop to cycle through each A object in the document object’s links array.

function displayLinks()

{

var linksCounter

for (linksCounter = 0; linksCounter < document.links.length; linksCounter++)

{

alert(document.links[linksCounter].href);

}

}

You used the length property of the links array in your condition to determine how many times you need to loop. Then, using an alert box, you display each A object’s href property. You can’t use document. write() in the onload event, because it occurs when the page has finished loading.

**10. Create two pages, one called IEOnly.htm and the other called FFOnly.htm. Each page should have a heading telling you what page is loaded, for example:**

**<H2>Welcome to the Internet Explorer only page</H2>**

**Using the functions for checking browser type, connect to the window object’s onload event handler and detect what browser the user has. Then if it’s the wrong page for that browser, redirect to the other page.**

**Solution:**

=======

The FFOnly.htm page is as follows:

<html>

<head>

<script language=”JavaScript” type=”text/javascript”>

function getBrowserName()

{

var lsBrowser = navigator.userAgent;

if (lsBrowser.indexOf(“MSIE”) >= 0)

{

lsBrowser = “MSIE”;

}

else if (lsBrowser.indexOf(“Netscape”) >= 0)

{

lsBrowser = “Netscape”;

}

else if (lsBrowser.indexOf(“Firefox”) >= 0)

{

lsBrowser = “Firefox”;

}

else if (lsBrowser.indexOf(“Safari”) >= 0)

{

lsBrowser = “Safari”;

}

else if (lsBrowser.indexOf(“Opera”) >= 0)

{

lsBrowser = “Opera”;

}

else

{

lsBrowser = “UNKNOWN”;

}

return lsBrowser;

}

function checkBrowser()

{

if (getBrowserName() == “MSIE”)

{

window.location.replace(“IEOnly.htm”);

}

}

</script>

</head>

<body onload=”checkBrowser()”>

<H2>Welcome to the Firefox only page</H2>

</body>

</html>

The IEOnly.htm page is very similar:

<html>

<head>

<script language=”JavaScript” type=”text/javascript”>

function getBrowserName()

{

var lsBrowser = navigator.userAgent;

if (lsBrowser.indexOf(“MSIE”) >= 0)

{

lsBrowser = “MSIE”;

}

else if (lsBrowser.indexOf(“Netscape”) >= 0)

{

lsBrowser = “Netscape”;

}

else if (lsBrowser.indexOf(“Firefox”) >= 0)

{

lsBrowser = “Firefox”;

}

else if (lsBrowser.indexOf(“Safari”) >= 0)

{

lsBrowser = “Safari”;

}

else if (lsBrowser.indexOf(“Opera”) >= 0)

{

lsBrowser = “Opera”

}

else

{

lsBrowser = “UNKNOWN”

}

return lsBrowser;

}

function checkBrowser()

{

if (getBrowserName() == “Firefox”)

{

window.location.replace(“FFOnly.htm”);

}

}

</script>

</head>

<body onload=”checkBrowser()”>

<H2>Welcome to the Internet Explorer only page</H2>

</body>

</html>

Starting with the IEOnly.htm page, first you add an onload event handler, so that on loading of the

page, your checkBrowser() function is called.

<body onload=”checkBrowser()”>

Then, in checkBrowser(), you use your getBrowserName() function to tell you which browser the

user has. If it’s Firefox, you replace the page loaded with the FFOnly.htm page. Note that you use

replace() rather than href, because you don’t want the user to be able to hit the browser’s Back button.

This way it’s less easy to spot that a new page is being loaded.

function checkBrowser()

{

if (getBrowserName() == “Firefox”)

{

window.location.replace(“FFOnly.htm”);

}

}

The FFOnly.htm page is identical, except that in your if statement you check for MSIE and redirect to

IEOnly.htm if it is MSIE.

function checkBrowser()

{

if (getBrowserName() == “MSIE”)

{

window.location.replace(“IEOnly.htm”);

}

}

**11. Insert an image in the page with the <img> tag. When the mouse pointer rolls over the image, it should switch to a different image. When the mouse pointer rolls out (leaves the image), it should swap back again.**

**Solution:**

**=======**

<html>

<head>

<script language=”JavaScript” type=”text/javascript”>

function mouseOver()

{

document.images[“myImage”].src = “Img2.jpg”;

}

function mouseOut()

{

document.images[“myImage”].src = “Img1.jpg”;

}

</script>

</head>

<body>

<img src=”Img1.jpg”

name=”myImage”

onmouseover=”mouseOver()”

onmouseout=”mouseOut()”>

</body>

</html>

At the top of the page you define your two functions to handle the onmouseover and onmouseout

events.

function mouseOver()

{

document.images[“myImage”].src = “Img2.jpg”;

}

function mouseOut()

{

document.images[“myImage”].src = “Img1.jpg”;

}

The function names tell you what events they will be handling. You access the img object for your <img>

tag using the document.images array and putting the name in square brackets. In the onmouseover

event you change the src property of the image to Img2.jpg, and in the onmouseout event you change

it back to img1.jpg, the image you specified when the page was loaded.

In the page itself you have your <img> tag.

<img src=”Img1.jpg”

name=”myImage”

onmouseover=”mouseOver()”

onmouseout=”mouseOut()”>

**12. Create a user interface for temperature converter and connect it to the existing code so that the user can enter a value in degrees Fahrenheit and convert it to centigrade.**

**Solution:**

**=======**

<html>

<head>

<script language=”JavaScript” type=”text/javascript”>

function convertToCentigrade(degFahren)

{

var degCent;

degCent = 5/9 \* (degFahren - 32);

return degCent;

}

function butToCent\_onclick()

{

var CalcBox = document.form1.txtCalcBox;

if (isNaN(CalcBox.value) == true || CalcBox.value == “”)

{

CalcBox.value = “Error Invalid Value”;

}

else

{

CalcBox.value = convertToCentigrade(CalcBox.value);

}

}

</script>

</head>

<body>

<form name=form1>

<P>

<input type=”text” name=txtCalcBox value=”0.0”>

</P>

<input type=”button”

value=”Convert to centigrade”

name=butToCent

onclick=”butToCent\_onclick()”>

</form>

</body>

</html>

The interface part is simply a form containing a text box into which users enter the Fahrenheit value and

a button they click to convert that value to centigrade. The button has its onclick event handler set to

call a function called butToCent\_onclick().

The first line of butToCent\_onclick() declares a variable and sets it to reference the object representing

the text box.

var CalcBox = document.form1.txtCalcBox;

Why do this? Well, in your code when you want to use document.form1.txtCalcBox, you can now

just use the much shorter CalcBox; it saves typing and keeps your code shorter and easier to read.

So

alert(document.form1.txtCalcBox.value);

is the same as

alert(CalcBox.value);

In the remaining part of the function you do a sanity check—if what the user has entered is a number

(that is, it is not NotANumber) and the text box does contain a value, you use the Fahrenheit-to-centigrade conversion function to do the conversion, the results of which are used to set the text box’s value.

**13. Create a user interface that allows the user to pick the computer system of her dreams, similar in principle to the e-commerce sites selling computers over the Internet. For example, she could be given a choice of processor type, speed, memory, and hard drive size, and the option to add additional components like a DVD-ROM drive, a sound card, and so on. As the user changes her selections, the price of the system should update automatically and notify her of the cost of the system as she has specified it, either by using an alert box or by updating the contents of a text box.**

**Solution:**

**=======**

<html>

<head>

<script language=”JavaScript” type=”text/javascript”>

var CompItems = new Array();

CompItems[100] = 1000;

CompItems[101] = 1250;

CompItems[102] = 1500;

CompItems[200] = 35;

CompItems[201] = 65;

CompItems[202] = 95;

CompItems[300] = 50;

CompItems[301] = 75;

CompItems[302] = 100;

CompItems[400] = 10;

CompItems[401] = 15;

CompItems[402] = 25;

function updateOrderDetails()

{

var total = 0;

var orderDetails = “”;

var formElement;

formElement =

document.form1.cboProcessor[document.form1.cboProcessor.selectedIndex];

total = parseFloat(CompItems[formElement.value]);

orderDetails = “Processor : “ + formElement.text;

orderDetails = orderDetails + “ $” + CompItems[formElement.value] + “\n”;

formElement =

document.form1.cboHardDrive[document.form1.cboHardDrive.selectedIndex];

total = total + parseFloat(CompItems[formElement.value]);

orderDetails = orderDetails + “Hard Drive : “ + formElement.text;

orderDetails = orderDetails + “ $” + CompItems[formElement.value] + “\n”;

formElement = document.form1.chkCDROM;

if (formElement.checked == true)

{

orderDetails = orderDetails + “CD-ROM : $” +

CompItems[formElement.value] + “\n”;

total = total + parseFloat(CompItems[formElement.value]);

}

formElement = document.form1.chkDVD

if (formElement.checked == true)

{

orderDetails = orderDetails + “DVD-ROM : $” +

CompItems[formElement.value] + “\n”;

total = total + parseFloat(CompItems[formElement.value]);

}

formElement = document.form1.chkScanner

if (formElement.checked == true)

{

orderDetails = orderDetails + “Scanner : $” +

CompItems[formElement.value] + “\n”;

total = total + parseFloat(CompItems[formElement.value]);

}

formElement = document.form1.radCase

if (formElement[0].checked == true)

{

orderDetails = orderDetails + “Desktop Case : $” +

CompItems[formElement[0].value];

total = total + parseFloat(CompItems[formElement[0].value]);

}

else if (formElement[1].checked == true)

{

orderDetails = orderDetails + “Mini Tower Case : $” +

CompItems[formElement[1].value];

total = total + parseFloat(CompItems[formElement[1].value]);

}

else

{

orderDetails = orderDetails + “Full Tower Case : $” +

CompItems[formElement[2].value]

total = total + parseFloat(CompItems[formElement[2].value]);

}

orderDetails = orderDetails + “\n\nTotal Order Cost is $” + total;

document.form1.txtOrder.value = orderDetails;

}

</script>

</head>

<body>

<form name=”form1”>

<table>

<TR>

<TD width=”300”>

Processor

<br>

<select name=cboProcessor>

<option value=”100”>MegaPro 10ghz</option>

<option value=”101”>MegaPro 12</option>

<option value=”102”>MegaPro 15ghz</option>

</select>

<br><br>

Hard drive

<br>

<select name=cboHardDrive>

<option value=”200”>30tb</option>

<option value=”201”>40tb</option>

<option value=”202”>60tb</option>

</select>

<br><br>

CD-ROM

<input type=”checkbox” name=chkCDROM value=”300”>

<br>

DVD-ROM

<input type=”checkbox” name=chkDVD value=”301”>

<br>

Scanner

<input type=”checkbox” name=chkScanner value=”302”>

<br><br>

Desktop Case

<input type=”radio” name=radCase checked value=”400”>

<br>

Mini Tower

<input type=”radio” name=radCase value=”401”>

<br>

Full Tower

<input type=”radio” name=radCase value=”402”>

<P>

<input type=”button” value=”Update” name=butUpdate onclick=”updateOrderDetails()”>

</P>

</TD>

<TD>

<textarea rows=”20” cols=”35” id=txtOrder name=”txtOrder”>

</textarea>

</TD>

</TR>

</table>

</form>

</body>

</html>

This is just one of many ways to tackle this question—you may well have thought of a better way.

Here we are displaying the results of the user’s selection as text in a textarea box, with each item and

its cost displayed on separate lines and a final total at the end.

Each form element has a value set to hold a stock ID number. For example, a full tower case is stock ID

402. The actual cost of the item is held in arrays defined at the beginning of the page. Why not just store

the price in the value attribute of each form element? Well, this way is more flexible. Currently your

array just holds price details for each item, but we could modify it that so it holds more data—for example price, description, number in stock, and so on. Also, if this form is posted to a server the values

passed will be stock IDs, which we could then use for a lookup in a stock database. If the values were set

to prices and the form were posted, we’d have no way of telling what the customer ordered—all we’d

know is how much it all cost.

This solution includes an Update button which, when clicked, updates the order details in the textarea

box. However, you may want to add event handlers to each form element and update when anything

changes.

Turning to the function that actually displays the order summary, updateOrderDetails(), we can see

that there is a lot of code, and although it looks complex, it’s actually fairly simple. A lot of it is repeated

with slight modification.

To save on typing and make the code a little more readable, this solution declares a variable,

formElement, which will be set to each element on the form in turn and used to extract the stock ID

and, from that, the price. After the variable’s declaration, we then find out which processor has been

selected, calculate the cost, and add the details to the textarea.

formElement =document.form1.cboProcessor[document.form1.cboProcessor.selectedIndex];

total = parseFloat(CompItems[formElement.value]);

orderDetails = “Processor : “ + formElement.text;

orderDetails = orderDetails + “ $” + CompItems[formElement.value] + “\n”;

The selectedIndex property tells us which Option object inside the select control has been selected by

the user, and we set our formElement variable to reference that.

The same principle applies when we find the hard drive size selected, so let’s turn next to the check

boxes for the optional extra items, looking first at the CD-ROM check box.

formElement = document.form1.chkCDROM

if (formElement.checked == true)

{

orderDetails = orderDetails + “CD-ROM : $” +

CompItems[formElement.value] + “\n”;

total = total + parseFloat(CompItems[formElement.value]);

}

Again, we set the formElement variable to now reference the chkCDROM check box object. Then, if the

check box is checked, we add a CD-ROM to the order details and update the running total. The same

principle applies for the DVD and scanner check boxes.

Finally, we have the case type. Because only one case type out of the options can be selected, we’ve used a radio button group. Unfortunately, there is no selectedIndex for radio buttons as there is for check boxes, so we have to go through each radio button in turn and find out if it has been selected.

formElement = document.form1.radCase

if (formElement[0].checked == true)

{

orderDetails = orderDetails + “Desktop Case : $” +

CompItems[formElement[0].value];

total = total + parseFloat(CompItems[formElement[0].value]);

}

else if (formElement[1].checked == true)

{

orderDetails = orderDetails + “Mini Tower Case : $” +

CompItems[formElement[1].value];

total = total + parseFloat(CompItems[formElement[1].value]);

}

else

{

orderDetails = orderDetails + “Full Tower Case : $” +

CompItems[formElement[2].value]

total = total + parseFloat(CompItems[formElement[2].value]);

}

We check to see which radio button has been selected and add its details to the textarea and its price to

the total. If our array of stock defined at the beginning of the code block had further details, such as

description as well as price, we could have looped through the radio button array and added the details

based on the CompItems array.

Finally, we set the textarea to the details of the system the user has selected.

orderDetails = orderDetails + “\n\nTotal Order Cost is “ + total;

document.form1.txtOrder.value = orderDetails;

**14. Create a web page with an advertisement image at the top. When the page loads, select a random image for that advertisement. Every four seconds, make the image change to a different one, making sure a different advertisement is selected until all the advertisement images have been seen.**

**Solution:**

=======

<html>

<head>

<script language=”JavaScript” type=”text/javascript”>

var imagesSelected = new Array(false,false,false);

var noOfImages = 3;

var totalImagesSelected = 0;

function window\_onload()

{

setInterval(“switchImage()”,4000);

}

function switchImage()

{

var imageIndex;

if (totalImagesSelected == noOfImages)

{

for (imageIndex = 0; imageIndex < noOfImages; imageIndex++)

{

imagesSelected[imageIndex] = false;

}

totalImagesSelected = 0;

}

var selectedImage = Math.floor(Math.random() \* noOfImages) + 1;

while (imagesSelected[selectedImage - 1] == true)

{

selectedImage = Math.floor(Math.random() \* noOfImages) + 1;

}

totalImagesSelected++;

imagesSelected[selectedImage - 1] = true;

document.imgAdvert.src = “AdvertImage” + selectedImage + “.jpg”;

}

</script>

</head>

<body onload=”window\_onload()”>

<img src=”AdvertImage1.jpg” name=”imgAdvert”>

</body>

</html>

**15. Create a form that gets the user’s date of birth. Then, using that information, tell her on what day of the week she was born.**

**Solution:**

=======

<html>

<head>

<script language=”JavaScript” type=”text/javascript”>

var days = new Array();

days[0] = “Sunday”;

days[1] = “Monday”;

days[2] = “Tuesday”;

days[3] = “Wednesday”;

days[4] = “Thursday”;

days[5] = “Friday”;

days[6] = “Saturday”;

function dayOfWeek()

{

var form = document.form1;

var date = parseInt(form.txtDate.value)

var year = parseInt(form.txtYear.value)

if (isNaN(date) || isNaN(year))

{

alert(“Please enter a valid whole number”);

}

else

{

if (date < 1 || date > 31)

{

alert(“Day of the month must be between 1 and 31”);

}

else

{

userDate = date + “ “;

userDate = userDate +

form.selMonth.options[form.selMonth.selectedIndex].value;

userDate = userDate + “ “ + year;

var dateThen = new Date(userDate);

alert(days[dateThen.getDay()]);

}

}

}

</script>

</head>

<body>

<P>Find the day of your birth</P>

<P>

<form name=”form1”>

<input type=”text” name=”txtDate” size=”2” maxlength=”2”>

<select name=selMonth>

<option selected value=”Jan”>Jan</option>

<option selected value=”Feb”>Feb</option>

<option selected value=”Mar”>Mar</option>

<option selected value=”Apr”>Apr</option>

<option selected value=”May”>May</option>

<option selected value=”Jun”>Jun</option>

<option selected value=”Jul”>Jul</option>

<option selected value=”Aug”>Aug</option>

<option selected value=”Sept”>Sept</option>

<option selected value=”Oct”>Oct</option>

<option selected value=”Nov”>Nov</option>

<option selected value=”Dec”>Dec</option>

</select>

<input type=text name=txtYear size=4 maxlength=4>

<br>

<input type=”button” value=”Day of the week”

onclick=”dayOfWeek()” name=”button1”>

</form>

</P>

</body>

</html>

The solution is surprisingly simple. You create a new Date object based on the date entered by the user.

Then you get the day of the week using the Date object’s getDay() method. This returns a number, but

by defining an array of days of the week to match this number, you can use the value of getDay() as the

index to your days array.

You also do some basic sanity checking to make sure that the user has entered numbers and that in the

case of the date, the number is between 1 and 31. You could have defined a select element as the method

of getting the date and only have numbers from 1 to 31. Of course, for either way, you don’t check whether invalid dates are entered (for example, the 31st of February). You might want to try this as an

additional exercise.

*Hint: To get the last day of the month, get the first day of the next month and then subtract one.*

**16. The following code contains a number of common errors. See if you can spot them:**

<html>

<head>

</head>

<body>

<script language=JavaScript>

function checkForm(theForm)

{

var formValid = true;

var elementCount = 0;

while(elementCount =< theForm.length)

{

if (theForm.elements[elementcount].type == “text”)

{

if (theForm.elements[elementCount].value() = “”)

alert(“Please complete all form elements”)

theForm.elements[elementCount].focus;

formValid = false;

break;

}

}

return formValid;

}

</script>

<form name=form1 onsubmit=”return checkForm(document.form1)”>

<input type=”text” ID=text1 name=text1>

<br>

CheckBox 1<input type=”checkbox” ID=checkbox1 name=checkbox1>

<br>

CheckBox 2<input type=”checkbox” ID=checkbox2 name=checkbox2>

<br>

<input type=”text” ID=text2 name=text2>

<p>

<input type=”submit” value=”Submit” ID=submit1 name=submit1>

</p>

</form>

</body>

</html>

**Solution:**

========

The bug-free version looks like this:

<html>

<head>

</head>

<body>

<script language=”JavaScript”>

function checkForm(theForm)

{

var formValid = true;

var elementCount = 0;

while(elementCount < theForm.length)

{

if (theForm.elements[elementCount].type == “text”)

{

if (theForm.elements[elementCount].value == “”)

{

alert(“Please complete all form elements”)

theForm.elements[elementCount].focus();

formValid = false;

break;

}

}

elementCount++;

}

return formValid;

}

</script>

<form name=”form1” onsubmit=”return checkForm(document.form1)”>

<input type=”text” id=”text1” name=”text1”>

<br>

CheckBox 1<input type=”checkbox” id=”checkbox2” name=”checkbox2”>

<br>

CheckBox 1<input type=”checkbox” id=”checkbox1” name=”checkbox1”>

<br>

<input type=”text” id=”text2” name=”text2”>

<P>

<input type=”submit” value=”Submit” id=”submit1” name=”submit1”>

</P>

</form>

</body>

</html>

Let’s look at each error in turn.

The first error is a logic error.

while(elementCount =< theForm.length)

Arrays start at 0 so the first Form object is at index array 0, the second at 1, and so on. The last Form

object has an index value of 4. However, theForm.length will return 5 because there are five elements

in the form. So the while loop will continue until elementCount is less than or equal to 5, but as the

last element has an index of 4, this is one past the limit. You should write either this:

while(elementCount < theForm.length)

or this:

while(elementCount <= theForm.length - 1)

Either is fine, though the first is shorter.

You come to your second error in the following line:

if (theForm.elements[elementcount].type == “text”)

On a quick glance it looks fine, but it’s JavaScript’s strictness on case sensitivity that has caused your

downfall. The variable name is elementCount, not elementcount with a lowercase *c.* So this line

should read as follows:

if (theForm.elements[elementCount].type == “text”)

The next line with an error is this:

if (theForm.elements[elementCount].value() = “”)

This has two errors. First, value is a property and not a method, so there is no need for parentheses after

it. Second, you have the all-time classic error of one equals sign instead of two. Remember that one

equals sign means “Make it equal to,” and two equals signs mean “Check if it is equal to.” So with the

changes, the line is:

if (theForm.elements[elementCount].value == “”)

The next error is your failure to put your block of if code in curly braces. Even though JavaScript won’t

throw an error since the syntax is fine, the logic is not so fine, and you won’t get the results you expect.

With the braces, the if statement should be as follows:

if (theForm.elements[elementCount].value == “”)

{

alert(“Please complete all form elements”)

theForm.elements[elementCount].focus;

formValid = false;

break;

}

The penultimate error is in this line:

theForm.elements[elementCount].focus;

This time you have a method but with no parentheses after it. Even methods that have no parameters

must have the empty parentheses after them. So, corrected, the line is as follows:

theForm.elements[elementCount].focus();

Now you’re almost done; there is just one more error. This time it’s not something wrong with what’s

there, but rather something very important that should be there but is missing. What is it? It’s this:

elementCount++;

This line should be in your while loop, otherwise elementCount will never go above 0 and the while

loop’s condition will always be true, resulting in the loop continuing forever: a classic infinite loop.

**17. Here’s some HTML code that creates a web page. Re-create this page, using JavaScript to generate the HTML using only DOM objects, properties, and methods. Test your code in IE, Firefox, Opera, and Safari (if you have it) to make sure it works in them.**

*Hint: Comment each line as you write it to keep track of where you are in the tree structure, and create a new variable for every element on the page (for example, not just one for each of the TD cells, but nine variables).*

<html>

<head>

</head>

<body>

<table>

<thead>

<tr>

<td>Car</td>

<td>Top Speed</td>

<td>Price</td>

</tr>

</thead>

<tbody>

<tr>

<td>Chevrolet</td>

<td>120mph</td>

<td>$10,000</td>

</tr>

<tr>

<td>Pontiac</td>

<td>140mph</td>

<td>$20,000</td>

</tr>

</tbody>

</table>

</body>

</html>

**Solution:**

=======

It seems a rather daunting example, but rather than being difficult, it is just a conjunction of two areas,

one building a tree structure and the other navigating the tree structure. You start by navigating to the

<body/> element and creating a <table/> element. Now you can navigate to the new <table/> element

you’ve created and create a new <thead/> element and carry on from there. It’s a lengthy and repetitious

process, so that’s why it’s a good idea to comment your code to keep track of where you are.

<html>

<head>

</head>

<body>

<script language=”JavaScript”>

var TableElem = document.createElement(“table”)

var THElem = document.createElement(“thead”)

var TRElem1 = document.createElement(“TR”)

var TRElem2 = document.createElement(“TR”)

var TRElem3 = document.createElement(“TR”)

var TDElem1 = document.createElement(“TD”)

var TDElem2 = document.createElement(“TD”)

var TDElem3 = document.createElement(“TD”)

var TDElem4 = document.createElement(“TD”)

var TDElem5 = document.createElement(“TD”)

var TDElem6 = document.createElement(“TD”)

var TDElem7 = document.createElement(“TD”)

var TDElem8 = document.createElement(“TD”)

var TDElem9 = document.createElement(“TD”)

var TBODYElem = document.createElement(“TBODY”)

var TextNodeA1 = document.createTextNode(“Car”)

var TextNodeA2 = document.createTextNode(“Top Speed”)

var TextNodeA3 = document.createTextNode(“Price”)

var TextNodeB1 = document.createTextNode(“Chevrolet”)

var TextNodeB2 = document.createTextNode(“120mph”)

var TextNodeB3 = document.createTextNode(“$10,000”)

var TextNodeC1 = document.createTextNode(“Pontiac”)

var TextNodeC2 = document.createTextNode(“140mph”)

var TextNodeC3 = document.createTextNode(“$14,000”)

docNavigate = document.documentElement; //Starts with HTML document

docNavigate = docNavigate.lastChild; //Moves to body element

docNavigate.appendChild(TableElem); //Adds the table element

docNavigate = docNavigate.lastChild; //Moves to the table element

docNavigate.appendChild(THElem); //Adds the thead element

docNavigate = docNavigate.firstChild; //Moves to the thead element

docNavigate.appendChild(TRElem1); //Adds the TR element

docNavigate = docNavigate.firstChild; //Moves the TR element

docNavigate.appendChild(TDElem1); //Adds the first TD element in the

// heading

docNavigate.appendChild(TDElem2); //Adds the second TD element in the

// heading

docNavigate.appendChild(TDElem3); //Adds the third TD element in the

// heading

docNavigate = docNavigate.firstChild; //Moves to the first TD element

docNavigate.appendChild(TextNodeA1); //Adds the second text node

docNavigate = docNavigate.nextSibling; //Moves to the next TD element

docNavigate.appendChild(TextNodeA2); //Adds the second text node

docNavigate = docNavigate.nextSibling; //Moves to the next TD element

docNavigate.appendChild(TextNodeA3); //Adds the third text node

docNavigate = docNavigate.parentNode; //Moves back to the TR element

docNavigate = docNavigate.parentNode; //Moves back to the thead element

docNavigate = docNavigate.parentNode; //Moves back to the table element

docNavigate.appendChild(TBODYElem); //Adds the tbody element

docNavigate = docNavigate.lastChild; //Moves to the tbody element

docNavigate.appendChild(TRElem2); //Adds the second TR element

docNavigate = docNavigate.lastChild; //Moves to the second TR element

docNavigate.appendChild(TDElem4); //Adds the TD element

docNavigate.appendChild(TDElem5); //Adds the TD element

docNavigate.appendChild(TDElem6); //Adds the TD element

docNavigate = docNavigate.firstChild; //Moves to the first TD element

docNavigate.appendChild(TextNodeB1); //Adds the first text node

docNavigate = docNavigate.nextSibling; //Moves to the next TD element

docNavigate.appendChild(TextNodeB2); //Adds the second text node

docNavigate = docNavigate.nextSibling; //Moves to the next TD element

docNavigate.appendChild(TextNodeB3); //Adds the third text node

docNavigate = docNavigate.parentNode; //Moves back to the TR element

docNavigate = docNavigate.parentNode; //Moves back to the tbody element

docNavigate.appendChild(TRElem3); //Adds the TR element

docNavigate = docNavigate.lastChild; //Moves to the TR element

docNavigate.appendChild(TDElem7); //Adds the TD element

docNavigate.appendChild(TDElem8); //Adds the TD element

docNavigate.appendChild(TDElem9); //Adds the TD element

docNavigate = docNavigate.firstChild; //Moves to the TD element

docNavigate.appendChild(TextNodeC1); //Adds the first text node

docNavigate = docNavigate.nextSibling; //Moves to the next TD element

docNavigate.appendChild(TextNodeC2); //Adds the second text node

docNavigate = docNavigate.nextSibling; //Moves to the next TD element

docNavigate.appendChild(TextNodeC3); //Adds the third text node

</script>

</body>

</html>