

Due in the Digital Drop Box on Wednesday, Sept. 20 at 11:00 PM

1. The HTML document listed below with gaps adds as matrices two user-supplied two-dimensional arrays. It then outputs to the browser the two input arrays/matrices and the sum array/matrix as tables with identifying captions. The screen capture at right shows the result when two 2×2 matrices are added. Recall that, to be compatible for addition, two matrices must have the same dimensions, and the sum array has these common dimensions. The JavaScript program first prompts for and inputs the common numbers of rows and columns. It then allocates the three arrays, prompts for and inputs integer values for the summand arrays, adds these arrays, and outputs the tables (as shown at right) to the browser. An example of a prompt for an element of an input array is

The entry for array 1 at indices 0, 1

This is prompting for the element at row 0, column 1 in the first input array.

The JavaScript code consists of four functions. Function `start` is invoked when the body finishes loading. All the other functions are called from it. Function `start` inputs the numbers of rows and columns, creates the three arrays, and allocates elements for their rows. In a loop, it creates row arrays for each cell of these arrays and allocates elements. I have had problems using the `Array` constructor in the form

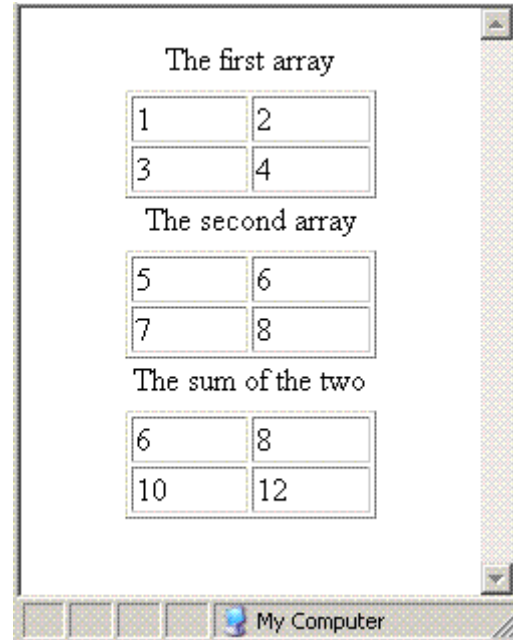
```
ar = new Array( x );
```

where `x` is a variable or a property (e.g., `ar1.length`) since I installed Windows XP Service Pack 2. Rather than creating an array of size `x`, it creates an array with one element, whose value is `x`. The solution is to use, instead of the above form, two steps:

```
ar = new Array();
ar.length = x;
```

After allocating for the arrays, function `start` calls function `inputArray` twice, once for each input array. It has two formal parameters, one, `ar`, is the output parameter for the array, and the other, `num`, is the input parameter for the array number used in the prompt. It uses nested `for` loops to range over all the cells in the array. Inside the nested `for` loops, you need a loop in which the user is prompted for the value of the cell. If the value (which is returned as a string) is not numerical (use the `isNaN` function), then an alert box reminds the user that the value must be a number, and the user is prompted again. Note that numerical strings must be converted to integers with `parseInt`.

After the calls to `inputArray`, function `start` calls function `addArrays`, which has three formal parameters: `ar1` and `ar2` are input parameters for the two summand arrays, and `sumAr` is an output parameter for their sum. The body of



`addArrays` consists of two nested `for` loops within which the sum of the elements of `ar1` and `ar2` at indices `i, j` is assigned to the corresponding element of `sumAR`.

Finally, function `start` calls `outArray` three times, to output tables for the three arrays. Function `outArray` has two formal parameters: `ar` is the input parameter for the array and `caption` is the input parameter for the string that will appear in the table's caption. The table sent to the browser has a `tbody` element but no `thead` element. This function also has nested `for` loops to range over all the cells of the array. `td` elements contain the values of the individual array cells, and `tr` elements correspond to rows in the array.

The gaps in the listing are labeled with Greek letters. After the listing, these letters are repeated along with descriptions of the code for the corresponding gaps. You can download the file with gaps as part of the zipped file accompanying this assignment.

Note of the use of `for/in`

The `for/in` construction does not always work when it should (when you want to range a loop-control variable over the indices of an array). When it doesn't work, you are forced to use a C/C++-style `for` loop (usually with the length of an array as the upper bound of the loop-control variable). Try to use a `for/in` loop first, and switch to the other style only if you can't get the `for/in` loop to work. (This is a pain, but `for/in` loops are conceptually more pleasing. Hopefully, in the future, Microsoft will get the bugs out.)

```
<html>
<head>
<title>HW 2, Prob 1</title>
<script type="text/javascript">
  function start()
  {
    var rows = prompt( "Number of rows", 1 ),
        cols = prompt( "Number of columns", 1 ),
        ar1 = new Array(),
        ar2 = new Array(),
        ar3 = new Array();

    ar1.length = ar2.length = ar3.length = rows;

    for ( var i = 0; i < rows; i ++ ) {
      ar1[i] = new Array();
      ar2[i] = new Array();
      ar3[i] = new Array();
      ar1[i].length = ar2[i].length = ar3[i].length = cols;
    }

    inputArray( ar1, 1 );
    inputArray( ar2, 2 );

    addArrays( ar1, ar2, ar3 );
```

Continued

Continued from previous page

```

outputArray( ar1, "The first array" );
outputArray( ar2, "The second array" );
outputArray( ar3, "The sum of the two" );
}

function inputArray( ar, num )
{
  for ( var i in ar )
    for ( var j = 0; j < ar[i].length; j ++ )
      α
      _____
      _____
      _____
      _____
}

function addArrays( ar1, ar2, sumAr )
{
  β
  _____
  _____;
}

function outputArray( ar, caption )
{
  document.writeln( "<table border = '1' +
                    " align = 'center' width = '60%'">" );
  document.writeln( "<caption>" + caption + "</caption>" );
  document.writeln( "<tbody>" );

  γ
  _____
  _____
  _____
  _____
  _____

  document.writeln( "</tbody>" );
  document.writeln( "</table>" );
}
</script>
<body onload="start()">
</body>
</html>

```

- α : A loop in which the user is prompted for the value of `ar[i][j]`. If the value (which is returned as a string) is not numerical (use the `isNaN` function), then an alert box reminds the user that the value must be a number, and the user is prompted again. Numerical strings must be converted to integers with `parseInt`. (The lines shown do not show the structure of the code that was removed.)
- β : Two nested `for` loops within which the sum of the elements of `ar1` and `ar2` at indices `i, j` is assigned to the corresponding element of `sumAR`
- γ : Nested `for` loops to range over all the cells of array `ar`. `td` elements contain the values of the individual array cells, and `tr` elements correspond to rows in the array (The lines shown do not show the structure of the code that was removed.)

2. The HTML document listed below with gaps prompts for a string, a substring of the first that is to be replaced, and a string to replace it with. It then raises an alert box that contains the first string with the indicated substring replaced as specified. If the substring occurs more than once in the original string, only the first occurrence is replaced. If the supposed substring in fact does not occur in the original string, then the user is alerted to this fact, and nothing more is done. For example, if the first string is

It is on the desk.

the second string is

on

and the third string is

above

then the string presented in the alert box will be

It is above the desk.

The code consists of a single function, `start`, which is invoked when the body of the document is loaded. The code in this function initializes variables `str`, `target`, and `replace` to the first, second, and third strings, respectively, that are prompted for. It initializes variable `start` to the index in `str` where `target` begins and declares other variables, including `result`, which will contain the string after the replacement has been made. The code then checks the value of `start` to determine whether `target` in fact does not occur in `str`. If so, it alerts the user to this fact and returns. Otherwise, it gets the substrings of `str` before and after the (first) occurrence of `target` and concatenates them with `replace` in the middle. The resulting string is assigned to `result` and displayed in the alert box.

The gaps in the listing are labeled with Greek letters. After the listing, these letters are repeated along with descriptions of the code for the corresponding gaps. You can download the file with gaps as part of the zipped file accompanying this assignment.

```

<html>
<head>
<title>HW2, Prob2</title>
<script type="text/javascript">
  function start()
  {
    var str = prompt( "A string", "" ),
        target = prompt( "The part to replace", "" ),
        replace = prompt( "What to replace it with", "" ),
        start = str.indexOf( target ),
        α _____,
        result;

    if ( β _____ ) {
      alert( target + " does not occur in " + str );
      return;
    }

    γ _____
    _____
    _____

    alert( result );
  }
</script>
</head>
<body onload="start()">
</body>
</html>

```

α : Declare any other variables needed.

β : A condition involving `str` that is true if `target` does not occur in `str`

γ (three lines): Get the substrings of `str` before and after the (first) occurrence of `target` and concatenate them with `replace` in the middle. Assign the resulting string to `result`. **Hint:** Where `str` is a string, `str.length` is the length of `str` (the number of characters in it, or one less than the largest index). Recall that `str.substr(start, len)` returns the substring of `str` of length `len` starting at index `start`, and `str.substr(start)` returns the substring of `str` starting at index `start` and extending to the end of `str`.