COMP519 Practical 7
JavaScript (3)

Introduction

- This worksheet contains further exercises that are intended to familiarise you with JavaScript Programming. While you work through the tasks below compare your results with those of your fellow students and ask for help and comments if required.
- This worksheet can be found at

  http://cgi.csc.liv.ac.uk/~ullrich/COMP519/notes/practical07.pdf

  and you might proceed more quickly if you cut-and-paste code from the PDF file. Note that a cut-and-paste operation may introduce extra spaces into your code. It is important that those are removed and that your code exactly matches that shown in this worksheet.
- The exercises and instructions in this worksheet assume that you use the Department's Linux systems to experiment with JavaScript.

  If you want to use the Department's Windows systems instead, then you can do so.
- To keep things simple, we will just use a text editor, a terminal, and a web browser. You can use whatever text editor and web browser you are most familiar or comfortable with.
- If you do not manage to get through all the exercises during this practical session, please complete them in your own time before the next practical takes place.

Exercises

1. Let us start with an exercise related to *functions* in JavaScript.

   a. Open a text editor and enter the following HTML markup and JavaScript code:

```
<!DOCTYPE html>
<html lang="en-GB">
  <head>
    <title>JavaScript 07A</title>
    <script>
    // Function sumAll has no minimum number of arguments,
    // but accepts an arbitrary number of arguments
    function sumAll() {
      sum = 0
      for (var i=0; i<arguments.length; i++)
          sum = sum + arguments[i]
      return sum
    }
    </script>
  </head>
  <body>
    <script>
```

```
    document.writeln("sumAll() = "+sumAll()+"<br>")
    document.writeln("sumAll(2) = "+sumAll(2)+"<br>")
    document.writeln("sumAll(2,3) = "+sumAll(2,3)+"<br>")
    document.writeln("sumAll(2,3,4) = "+sumAll(2,3,4)+"<br>")
    </script>
    <noscript>JavaScript not supported or enabled</noscript>
  </body>
</html>
```

b. Save the code to a file named `jsDemo07A.html` in `$HOME/public_html/`.

c. Open a terminal, go to the directory in which the file has been stored and make sure the file is only readable by yourself and nobody else can read, write or execute the file:

```
chmod u+r,og-rwx $HOME/public_html/jsDemo07A.html
```

(Note: No space after the comma!) You will only have to do so once. File permissions should not change while you continue to edit the file.

d. Now open a web browser and access the URL

```
http://student.csc.liv.ac.uk/~<user>/jsDemo07A.html
```

where *<user>* should be replaced by your departmental user name.

If everything is working correctly, then you should be shown the following in your web browser:

```
JavaScript Functions

Output:
sumAll() = 0
sumAll(2) = 2
sumAll(2,3) = 5
sumAll(2,3,4) = 9
```

If not, open the console in your web browser and track down the errors in your code.

e. In the definition of `sumAll` we have used a variable `sum`. We have learned that one of the distinctions that JavaScript makes with respect to variables that are introduced in a function is whether they are *local* (only accessible from within the function) or *global* (accessible from elsewhere in the code).

Is `sum` local or global?

Let us test that by adding the following code before the last `</script>`-tag in the file:

```
document.writeln("sum = "+sum+"<br>")
```

Save the file again. Clear the output shown in the error console, then refresh the page in your web browser.

What does the additional output tell you about whether `sum` is local or global?

f. If your conclusion is that `sum` is global, then change the code for the function `sumAll` so that it will be local.

Test that any change you make has the desired effect. Once you have confirmed that `sum` is local, comment out the line of code introduced in Exercise 1e.

2. We now move on to a more complex function, the `bubble_sort` function we discussed in the lectures.

   a. Open a text editor and enter the following HTML markup and JavaScript code:

```
<!DOCTYPE html>
<html lang="en-GB">
  <head>
    <title>JavaScript 07A</title>
    <style>
      table {
        border:     1px;
        cellpadding: 5px;
      }
    </style>
    <script>
    function bubble_sort(array) {
      function swap(array,i, j) {
        // swap can change array because array is
        // a local variable of the outer function bubble_sort
        var tmp = array[i]; array[i] = array[j]; array[j] = tmp;
      }
      if (!(array && array.constructor == Array))
        throw("Argument not an array")
      for (var i=0; i<array.length; i++)
        for (var j=0; j<array.length-i; j++)
          if (array[j+1] < array[j]) swap(array,j, j+1)
      return array
    }
    </script>
  </head>
  <body>
    <h1>Sorting Arrays</h1>
    <script>
    array = [20,4,3,9,6,8,5,10]
    document.writeln("<h2>bubble_sort function</h2>")
    document.writeln("<table>\n<tbody>")
    document.writeln("<tr><td>array before sorting"+
                     "</td><td>"+array.join(", ")+"</td></tr>")
    // Sort a copy of array
    sorted = bubble_sort(array.slice())
    document.writeln("<tr><td>array after sorting of copy"+
                     "</td><td>"+array.join(", ")+"</td></tr>")
    sorted = bubble_sort(array)
    document.writeln("<tr><td>array after sorting of itself"+
                     "</td><td>"+array.join(", ")+"</td></tr>")
    document.writeln("<tr><td>sorted array</td><td>"+
                     sorted.join(", ")+"</td></tr>")
    document.writeln("</tbody>\n</table>")
    </script>
  </body>
</html>
```

b. Save the file as `jsDemo07B.html` in `$HOME/public_html`. Make sure that the access rights for the file are set correctly, then open it in a web browser.

c. Compare the output produced by the script in with that in the lecture notes.

   Make sure that you understand what the difference between `bubble_sort(array)` and `bubble_sort(array.slice())` is.

d. In the lectures we have seen how we can use the prototype property to change the methods associated with an object. This not only works for user-defined objects, but also for objects that JavaScript already provides, for example, `Array`. We will explore this further with the help of the `bubble_sort` function.

   In preparation for this exercise add the following code in the *head element* of the file `jsDemo07B.html`:

```
function bubble_sort2() {
  return this
}
```

   Also add the following code at the end of the JavaScript code in the body element of `jsDemo07B.html`:

```
Array.prototype.bubble_sort = bubble_sort2
array = [20,4,3,9,6,8,5,10]
document.writeln("<h2>bubble sort array prototype method</h2>")
document.writeln("<table>\n<tbody>");
document.writeln("<tr><td>array  before sorting"+
                 "</td><td>"+array.join(", ")+"</td></tr>")
// Sort a copy of array
sorted = array.slice().bubble_sort()
document.writeln("<tr><td>array  after  sorting of copy"+
                 "</td><td>"+array.join(", ")+"</td></tr>")
sorted = array.bubble_sort()
document.writeln("<tr><td>array  after  sorting of itself"+
                 "</td><td>"+array.join(", ")+"</td></tr>")
document.writeln("<tr><td>sorted array</td><td>"+
                 sorted.join(", ")+"</td></tr>")
document.writeln("</tbody>\n</table>")
```

   Save the file. Clear the output shown in the console, then refresh the page in your web browser.

   As the additional output indicates `array.bubble_sort()` does not (yet) return a sorted array. Make sure that you understand what the line

```
Array.prototype.bubble_sort = bubble_sort2
```

   does and why the output is as it is.

   Expand the definition of `bubble_sort2`, re-using code from `bubble_sort`, so that for any array `a0`, `a0.bubble_sort()` returns a sorted array.

   Hint: Take the code from `bubble_sort` and replace the appropriate occurrences of `array` by the keyword `this`.

e. Bubble sort is incredibly inefficient and nobody in their right mind would use it. Plus, JavaScript arrays already have a more efficient pre-defined `sort` method. Let us compare it to our `bubble_sort` method.

   Add the following code at the end of the JavaScript code in the body element of `jsDemo07B.html`:

```
array = [20,4,3,9,6,8,5,10]
document.writeln("<h2>pre-defined sort array method</h2>")
document.writeln("<table>\n<tbody>");
document.writeln("<tr><td>array  before sorting"+
                 "</td><td>"+array.join(", ")+"</td></tr>")
// Sort a copy of array
sorted = array.slice().sort()
document.writeln("<tr><td>array  after  sorting of copy"+
                 "</td><td>"+array.join(", ")+"</td></tr>")
sorted = array.sort()
document.writeln("<tr><td>array  after  sorting of itself"+
                 "</td><td>"+array.join(", ")+"</td></tr>")
document.writeln("<tr><td>sorted array</td><td>"+
                 sorted.join(", ")+"</td></tr>")
document.writeln("</tbody>\n</table>")
```

Save the file. Clear the output shown in the console, then refresh the page in your web browser.

f. You should see that the pre-defined `sort` method puts the elements of the array in a different order than out `bubble_sort` method.

Using resources on the web, try to find out why this is so.

g. Again, using resources on the web, find out how the code introduced in Exercise 2e needs to be modified so that we get numerically sorted arrays. Make sure that you document the source of your solution as expected by the COMP519 Coding Standard [1].

h. Extend `Array` with a method `sum`: `array.sum()` returns the sum of all the elements of `array`. If `array` has length 0, then the value `0` should be returned.

```
document.writeln("<h2>array prototype sum function</h2>")
// For array0 the expected result is 0
array0 = [];
document.writeln("<div>[" + array0 + "].sum() = "+array0.sum()+"</div>")
// For array1  the expected result is 6
array1 = [3,2,1];
document.writeln("<div>[" + array1 + "].sum() = "+array1.sum()+"</div>")
// For array2  the expected result is 6
array2 = [3,"2M",[1,0]];
document.writeln("<div>[" + array2 + "].sum() = "+array2.sum()+"</div>")
```

i. Extend `Array` with a method `peek`: `array.peek()` returns the first item of `array` without changing `array` if `array` has at least one element. If `array` has length 0, then the value `undefined` should be returned.

Test you method using arrays of varying length.

References

[1] U. Hustadt. *COMP519 Coding Standard*. Department of Computer Science, University of Liverpool. October 2017. URL: http://cgi.csc.liv.ac.uk/~ullrich/COMP519/notes/COMP519CodingStandard.pdf (accessed 21 October 2018).