COMP519 Practical 6
JavaScript (1)

Introduction

- This worksheet contains 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.
- You might proceed more quickly if you cut-and-paste code from this 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.
- 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 by re-creating the 'Hello World' JavaScript example that you have seen in the lectures.

   It is assumed that you have already created a sub-directory called `public_html` in your home directory and that the directory is both readable and executable by everyone. Make sure that this is so before you proceed.

   a. Create a file name `jsDemo06A.html` in `$HOME/public_html/` containing the following HTML markup and JavaScript code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <script id="s1"> // Script with id s1
    </script>
    <p>Our first JavaScript script</p>
    <script id="s2"> // Script with id s2
      user = "<your name>"
      document.writeln("<p><b>Hello " + user +
                       "<br>\nHello World!</b></p>")
    </script>
    <script id="s3"> // Script with id s3
    </script>
    <p>This is the rest of the page</p>
    <noscript>JavaScript not supported or disabled</noscript>
  </body>
```

```
</html>
```

Replace *<your name>* with your own name.

The 'command' `document.writeln` writes a string of text to a document stream, here, the HTML document the JavaScript scripts are part of, at the *insertion point*. Basically, the first `document.writeln` in a script would insert its string argument where the script itself is placed in the markup and all consecutive `document.writeln` commands would place their arguments after that.

Note that there are three JavaScript scripts in this code. In order to be able to refer to each of these three scripts they have been given the ids `s1`, `s2` and `s3`. We will use these ids in the instructions below.

b. Open a terminal, go to the directory in which the file has been stored and make sure that 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/jsDemo06A.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.

c. Now open a web browser and access the URL

`https://student.csc.liv.ac.uk/~`*<user>*`/jsDemo06A.html`

where *<user>* should be replaced by your username.

Check that the page looks as expected and also that the HTML code producing the web page you are shown is exactly the HTML code you have seen in Exercise 1a above, including the JavaScript code.

d. If already at this point or in one of the later exercises the web browser does not show the expected result, you will have to debug your code. For that you will need access to some diagnostic output for the JavaScript code that you have written.

A lot of web browsers provide developer tools including a 'console' on which errors are shown. For example, in Firefox the console can be found in the main menu under 'Web Developer→Web Console'. The shortcut CTRL+SHIFT+I in Firefox also opens the error console. See [1, 2] for further information on the console in Google Chrome and Mozilla Firefox.

Figure out how to access the console in your web browser and open it. If the web browser you use does not have a console (or you cannot figure out how to open it), switch to a better browser.

e. To see how the console works let us introduce an error into our JavaScript code.

At the end of the script with id `s2` insert

```
document.writeln("(1) The value of userAge is: " +
                 userAge + "<br>")
document.writeln("(2) This statement is executed<br>")
```

into `jsDemo06A.html`. Save the file again and refresh the page in your web browser.

Note that the web page appears unchanged. In particular, the statements introduced in Exercise 1e seem to produce no output. However, in the console you should now see an error message indicating that `userAge` has not been defined/declared.

f. We should correct the error by adding a declaration for `userAge`.

First, try the following: Add

```
var userAge = "27"
document.writeln("(3) The value of userAge is: " +
                 userAge + "<br>")
```

at the end of the script with id `s2` in `jsDemo06A.html` (that is, after the `document.writeln` statement that already uses `userAge`). Save the file again. Clear the output shown in the console, then refresh the page in your web browser.

The console should no longer show an error message and the web page should include the lines:

```
(1) The value of userAge is: undefined
(2) This statement is executed
(3) The value of userAge is: 27
```

This shows you that in JavaScript a declaration of a variable does not have to precede its use in order to prevent a reference error, however the initialisation of a variable only affects code that is executed later (even if the initialisation is done as part of a declaration).

But this principle only applies to a particular JavaScript code fragment enclosed in the tags `<script>` and `</script>` as we will see next.

g. Move the declaration

```
var userAge = "27"
```

from the script with id `s2` to the script with id `s3`.

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

Is the code correct now or does it produce an error? What does that tell you about the scope of a variable declaration?

h. Move the declaration

```
var userAge = "27"
```

from the script with id `s3` to the script with id `s1`.

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

Is the code correct now or does it produce an error? Is the output the same as in Exercise 1f? If not, why not?

i. We need to consider one more possible way in which the scope of variable declarations might work. In a lot of programming languages, the scope of a variable is restricted to the *block* in which it is declared. For example, a variable declared in the then-branch of a conditional statement has the then-branch as its scope. On the else-branch of the same conditional statement (or anywhere else), the variable would be undeclared. Let us see whether this is the case for JavaScript. Add the following code **before** the noscript-element in `jsDemo06A.html`:

```
<!-- Exercise 1j -->
<script id="s4">
if (false) {
  var x = 519
} else {
  document.writeln("The value of x is " + x + "<br>")
}
</script>
```

(Note the syntax used for the comment. The comment is not within the script but within the surrounding HTML markup. So, the syntax for HTML comments is used.)

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

What output or errors does the script produce now?

j. In the script with id `s4`, replace the declaration

```
var x = 519
```

by

```
var y = 519
```

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

- What output or errors does the script produce now?
- What do Exercises 1i and 1j tell you about the scope of variable declarations in JavaScript?
- What do Exercises 1i and 1j tell you about the way JavaScript code is executed?

If there were any errors produced in the last two steps, try to fix them now.

2. Let us compare *floating-point arithmetic* in Python and JavaScript.

a. To do so, first create a file name `MathExample.py` with the following Python program:

```
#!/usr/bin/python3

import math
import numpy as np

s = input('Input : ')
i = int(s)
d = math.sqrt(i)
e = 4 / d
f = round(e)
print("sqrt(",i,") =", d)
print("4 / ",d,"=",e)
print("round(",e,") =",f)
for d in np.arange(0.4,0.7,0.1):
  r = i + d * np.sign(i)
  print("round(",r,") =",round(r))
```

In a terminal, go to the directory in which the file has been stored and make sure that the file is executable by yourself:

```
chmod u+x ./MathExample.py
```

Then execute `MathExample.py` for the inputs 4, -4, and 0. Copy the outputs of the Python program to a text file so that you can later compare them with the corresponding outputs of a JavaScript program.

b. Create a file named `jsDemo06B.html` in `$HOME/public_html/` with the the HTML markup and JavaScript code.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Math Example</title>
  </head>
<body>
  <script>
    var i = Number(prompt("Input :"))
    /* Complete the following three lines of code with assignments that
       correspond to the computations performed in the Python program */
    var d
    var e
    var f
    document.writeln("sqrt(" + i +") = " + d + "<br>")
    document.writeln("4/" + d + " = " + e + "<br>")
    document.writeln("round(" + e + ") = " + f + "<br>")
    document.writeln(e + " > 0     : " + (e > 0) + "<br>")
    document.writeln(e + " <= 0    : " + (e <= 0) + "<br>")
    for (d = 0.4; d < 0.7; d = d + 0.1) {
      var r = i + d * Math.sign(i)
      document.writeln("round(" + r + ") = " + Math.round(r) + "<br>")
    }
  </script>
</body>
```

c. As indicated by the comment, complete the code so that the same calculatons are performed to compute `d`, `e`, and `f` as in the Python program.

d. In a web browser access the URL

   `https://student.csc.liv.ac.uk/~`*`<user>`*`/jsDemo06B.html`

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

e. A so-called prompt will appear that allows you to enter a number (or anything else). Enter the number `4` and either click the 'OK' button in the prompt or press return.

   In the brower window you should now see the results of the various computations. Are all the results the same as for the Python program produced for input `4`?

f. To repeat the computation for a different input, you need to refresh the page. Then a new prompt will appear. Enter `-4` now and press return.

   Compare the results produced by the JavaScript program with those of the Python program for input `-4`.

g. Finally, refresh the page again, enter `0`, and press return.

   Compare the results produced by the JavaScript program with those of the Python Program for input `0`.

h. Make sure that you understand what is going on for each of the three inputs.

3. The precision of 64-bit floating-point numbers is necessarily limited. This problem is aggravated by the fact that the binary representation of these floating-points numbers allows to precisely represent a lot of binary fractions, e.g. $1/8$ and $1/128$, but not the decimal fractions that humans use, e.g. $1/10$ and $1/100$.

   The following example illustrates that.

a. Create a file `jsDemo06C.html` in `$HOME/public_html/` with the following content:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Math Example 2</title>
  </head>
  <body>
    <script>
     var x = 0.3 - 0.2
     var y = 0.2 - 0.1
     document.writeln("x = " + x + "<br>")
     document.writeln("y = " + y + "<br>")
     document.writeln("(x == y) = " + (x == y) + "<br>")
     document.writeln("(x == 0.1) = " + (x == 0.1) + "<br>")
     document.writeln("(y == 0.1) = " + (y == 0.1) + "<br>")
    </script>
  </body>
</html>
```

b. Open the URL

`https://student.csc.liv.ac.uk/~`*`<user>`*`/jsDemo06C.html`

in a web browser, where *`<user>`* should be replaced by your departmental user name.

c. Have a look at the output produced by the JavaScript program. Use an on-line decimal to floating-point converter such as

http://www.binaryconvert.com/convert_double.html
http://www.exploringbinary.com/floating-point-converter/

to understand what the floating-point numbers corresponding to 0.3, 0.2, 0.1 really are and what the results of $0.3 - 0.2$ and $0.2 - 0.1$ are.

d. Write a Python program that performs the same computations and comparisons as the JavaScript code in `jsDemo06C.html`. Do you get the same results?

References

[1] Kayce Basques. *Tools for Web Developers: Console Overview*. Google. 18 April 2019. URL: https://developer.chrome.com/docs/devtools/console/ (accessed 28 January 2023).

[2] Mozilla and individual contributors. *Firefox Developer Tools: Web Console*. Firefox Source Docs. 28 January 2023. URL: https://firefox-source-docs.mozilla.org/devtools-user/web_console/index.html (accessed 28 January 2023).