COMP519 Practical 9
JavaScript (4)

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.
- 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.

Exercises

1. In the lectures we have considered how objects are defined in JavaScript and how, in the absence of classes, we can establish an inheritance relationship.

   The following exercises is intended to reinforce those considerations.

   a. Create a file `jsDemo09A.html` in your `public_html` directory with the following content:

```
<!DOCTYPE html>
<html lang="en-GB">
  <head>
    <title>JavaScript 09A: Inheritance</title>
    <script id="j1">
      function Rectangle(width, height) {
        this.width  = width
        this.height = height
        this.type   = 'Rectangle'
      }
      Rectangle.prototype.area =
        function() { return this.width * this.height }

      function Square(length) {
        this.width = this.height = length
        this.type  = 'Square'
      }

      // Square inherits from Rectangle
```

```
      Square.prototype = new Rectangle()
    </script>
  </head>
  <body>
    <script id="j2">
      var rc1 = new Rectangle(2,3)
      var sq1 = new Square(5)
      document.writeln("The area of rc1 is ",rc1.area(),"<br>")
      document.writeln("The area of sq1 is ",sq1.area(),"<br>")
      document.writeln("The area of ",rc1," is ",rc1.area(),"<br>")
      document.writeln("The area of ",sq1," is ",sq1.area(),"<br>")
    </script>
  </body>
</html>
```

b. Make sure that the access rights for `jsDemo9A.html` are set correctly.

c. Open `jsDemo09A.html` in a web browser. You should see the following:

```
The area of rc1 is 6
The area of sq1 is 25
The area of [object Object] is 6
The area of [object Object] is 25
```

Note that whenever you print out an object, JavaScript uses the object's `toString` method to get a string representation of the object. If the `toString` is inherited from `Object` without being overwritten, then this string representation is `"[object type]"`, where *type* is the object type.

d. Change the string representation of a `Rectangle` object with width *wd* and height *ht* to `Rectangle[`*wd*`,`*ht*`]`. Note that `this.`*type* already gives you the string `'Rectangle'`. If done correctly, the last two lines of output of your script change to

```
The area of Rectangle[2,3] is 6
The area of Square[5,5] is 25
```

e. That the string representation of `Square` objects is now `Square[`*wd*`,`*ht*`]` is a bit sub-optimal. It should really be `Square[`*wd*`]`. Change the `toString` method for `Square` to achieve the desired representation. The last line of the output should now be

```
The area of Square[5] is 25
```

f. In analogy to `Rectangle` and `Square` add an object constructor `Polygon` for regular polygons (the code for this should be added to the script element with id `j1`). The `Polygon` constructor should have two parameters for the number of sides *n* and the length *s* of each side. The string representation should be *n*`-Polygon[`*s*`]`. The area of a polygon should be computed using the formula given in [2], rounded to two digits after the decimal point. Hint: The formula in [2] uses a tangent function that takes degrees as argument while JavaScript uses a tangent function that takes radians as argument. See [1] for how to deal with the conversion.

Test your solution with a five-sided polygon where each side has a length of 6. The output you produce for this test case should be

```
The area of 5-Polygon[6] is 61.94
```

g. Extend the code in the script element with id `j2` so that (i) 10 shapes are randomly generated and stored in an array and (ii) the string representation of the shapes and their area are printed out.

In more detail, to generate a shape, first generate a random natural number r between 0 and 2.

- If r is equal to 0, generate another random natural number s between 2 and 9, construct a square object O with s as the length of its sides, and add O to your array.
- If r is equal to 1, generate two another random natural number s and t between 2 and 9, construct a rectangle object O with sides s and t, and add O to your array.
- If r is equal to 2, generate a random number n between 5 and 8 as well as a random natural number s between 2 and 9, construct a polygon object with n sides of length s, and add O to your array.

Then iterate over all elements in the array, print out the string representation and their area like we have already done for `rc1` and `sq1`.

2. In the lectures we have also considered various aspects of objects in JavaScript, in particular, the way *instance variables* and *'class' variables* are declared and how these can be made *public* or *private*.

   The following exercises is intended to reinforce those considerations.

   a. Create a file `jsDemo09B.html` in your `public_html` directory with the following content:

```
<!DOCTYPE HTML>
<html lang="en-GB">
  <head>
    <title>JavaScript 09B: Objects</title>
    <script>
    </script>
  </head>
  <body>
    <script>
      var e = []
      e[0] = new Employee("Hal Smith", 30000)
      e[1] = new Employee("Tim Peck", 20000)
      e[2] = new Employee("Ari Bell",18000)
      // For e[0].name        we expect 'Hal Smith'
      document.writeln("e[0].name = "+e[0].name+"<br>")
      // For e[0].salary      we expect 'undefined'
      document.writeln("e[0].salary = "+e[0].salary+"<br>")
      // For e[0].getSalary() we expect 30000
      document.writeln("e[0]'s salary = "+e[0].getSalary()+"<br>")
      // For e[1].getName()   we expect 'Tim Peck'
      document.writeln("e[1]'s name = "+e[1].getName()+"<br>")
      // For e[1].getSalary() we expect 20000
      document.writeln("e[1]'s salary = "+e[1].getSalary()+"<br>")
      // We make changes to e[1]
      document.writeln("Changing e[1]'s name to 'Tom Beck' and "+
                       "salary to 25000<br>")
      e[1].name = "Tom Beck"
      e[1].setSalary(25000)
      // For e[1].getName()   we now expect 'Tom Beck'
```

```
      document.writeln("e[1]'s name = "+e[1].getName()+"<br>")
      // For e[1].getSalary() we now expect 25000
      document.writeln("e[1]'s salary = "+e[1].getSalary()+"<br>")
      // For e[1].getEmployeeCount() we expect 3
      document.writeln("Employees: "+e[1].getEmployeeCount()+"<br>")
    </script>
  </body>
</html>
```

This HTML markup will serve as testbed for Exercise 2c.

b. Make sure that the access rights for `jsDemo09B.html` are set correctly.

c. We want to define an *employee* object. To keep the exercise simple, we assume that the only attributes of an *employee* are a *name* and a *salary*. The first should be *public*, the second *private*. In addition, we need a method to obtain information on an employee's salary as well as a method that allows us to change it. Finally, we want to keep track of how many employees there are in total and we want to keep that number private. The total number of employee's should be automatically incremented each time a new employee object is created.

   Create a *constructor* for employee objects that satisfies these requirements and add it to the *head element* of `jsDemo09B.html`.

d. Test your definition of the *employee* constructor by opening `jsDemo09B.html` in your web browser and observing that the output is as expected.

e. Since the total employee count is not really an attribute of a particular employee, it is a bit odd that we obtain that count by using an expressions like `e[1].employeeCount()`.

   Is it possible to define `employeeCount()` in such a way that we could use the expression `Employee.employeeCount()` instead? If so, modify your code accordingly and test your solution by replacing

```
document.writeln("Employees: "+e[1].getEmployeeCount()+"<br>")
```

   with

```
document.writeln("Employees: "+Employee.getEmployeeCount()+"<br>")
```

   and checking that you get the correct output after you have saved the file and refreshed the page in the browser.

f. Being able to 'create' new employees is obviously nice, but sometimes we also have to 'delete' an existing employee.

   Can we extend our definition of `Employee` by a method `remove` that deletes a particular employee object and at the same time decrements the total employee count? If so, modify your code accordingly and test your solution by adding the code

```
e[2].remove()
// Expected to print the number 2 now
document.writeln("Employees: "+Employee.getEmployeeCount()+"<br>")
```

   and checking that you get the correct output after you have saved the file and refreshed the page in the browser.

   Hint: There is a related discussion [3] on StackOverflow.

   The optiomal solution would be to develop an `Employees` object that maintains the set of all `Employee` objects and would allow to remove an employee by name. However, that is beyond the scope of this practical.

References

[1] Mozilla and individual contributors. *`Math.tan()`*. MDN Web Docs. 31 December 2022. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/tan (accessed 24 January 2023).

[2] John D. Page. *Area of a regular polygon*. Math Open Reference. 2011. URL: https://www.mathopenref.com/polygonregulararea.html (accessed 24 January 2023).

[3] Wytze (https://stackoverflow.com/users/316727/wytze). *How to quickly clear a JavaScript Object?* Stack Exchange Network. 27 February 2013. URL: https://stackoverflow.com/a/15089643 (accessed 28 January 2023).