Lab session 7 – JavaScript (part 2)

Unit	Programming languages: principles and design (6G6Z1110) Programming languages – SE frameworks (6G6Z1115)
Lecturer	Rob Frampton
Week	8
Portfolio element	This lab is aimed at helping you with the JavaScript portfolio element.

Description

The aim with this lab exercises is to practise the several JavaScript elements as studied in the lecture "JavaScript – part 2". By the end of this lab session, you should have learnt how to implement JavaScript objects and classes.

Exercises

Exercise 1

Copy this code which corresponds to the definition of a JavaScript object:

```
let user = {
  name: "John",
  age: 30,
  // method "speak" here!
};
```

Add the missing method "`speak`" to above object such that executing the following code:

```
user.speak()
```

Will result in the following output in the console:

```
Hello, my name is John and I am 30 years old.
```

Exercise 2a

The following code creates a JavaScript object with two data properties, "`name`" and "`surname`". An *accessor* property called "`email`" should be added to the object which returns an email address in the format: name.surname@mmu.ac.uk:

```
let user = {
  name: "John",
  surname: "Smith"
};
```

Such that the following code:

```
console.log(user.email)
```

Will result in the following output:

```
John.Smith@mmu.ac.uk
```

Exercise 2b

Using the Exercise 2a's code, add the required code such that when *assigning* to an email address, it should extract the first name and surname from the email address and use them to update the name and surname fields. You will need to use the split function from the String type to separate the email address components.

For example, the following code:

```
user.email = "Albert.Einstein@mmu.ac.uk"
console.log(user.name)
console.log(user.surname)
```

Should result in the following console output:

```
Albert
Einstein
```

Exercise 3a

Implement Exercise 2b using a class called "`User`" instead of an object. Use a `constructor` to assign initial values to the properties "`name`" and "`surname`".

You can test your class with the following code:

```
let user = new User("Paul", "Rosenberg")
console.log(user.name)
console.log(user.surname)
console.log(user.email)

user.email = "David.Haig@mmu.ac.uk"
console.log(user.name)
console.log(user.surname)
```

Exercise 3b

Modify the code for Exercise 3a by creating `get` and `set` accessors for the `name` and `surname` fields too, which wrap properties named `_name` and `_surname`, respectively. Change the setters so that surnames must be at least three characters long, or the they will not be updated.

For example, the following code:

```
let user = new User("Rick", "DeVoe")
user.surname = "Astley"
user.surname = "A"
console.log(user.name)
console.log(user.surname)
```

Should result in the following console output:

```
Rick
Astley
```

Exercise 4

Extend the code from Exercise 3b by writing a subclass of `User` named `StaffUser`. `StaffUser` should override the email `get` accessor to return email addresses in the form name.surname@staff.mmu.ac.uk.

For example, the following code:

```
let user = new User("Nick", "Drake")
console.log(user.email)

let staffUser = new StaffUser("Bernie", "Taupin")
console.log(staffUser.email)
```

Should result in the following console output:

```
Nick.Drake@mmu.ac.uk
Bernie.Taupin@staff.mmu.ac.uk
```