Stefano Schmidt
Launch School
Introduction to Programming With JavaScript
Objects – Exercises

1: This exercise asks us how we can access the name property of the person object:

```
let person = {
  name:       'Bob',
  occupation: 'web developer',
  hobbies:    'painting',
};
```

Two ways of doing this are the following:

```
person['name'];
```

```
person.name;
```

2: This exercise asks us to explain which of the following are valid keys for an object:

1: this is a valid key for an object; it will be coerced to the value '1'.
'1': this is a valid key for an object.
undefined: this is a valid key for an object; it will be coerced to the value 'undefined'.
'hello world': this is a valid key for an object.
true: this is a valid key for an object; it will be coerced to the value 'true'.
'true': this is a valid key for an object.

3: This exercise asks us to create an object that behaves like an array in a loop using object literal syntax. I did this as follows:

```
let myArray = {
  length: 5,
  0: 'dog',
  1: 'cat',
  2: 'turtle',
  3: 'llama',
  4: 'giraffe'
}
```

4: This exercise asks us to create an array with the keys of an object named obj in uppercase. I did this as follows:

```
let keysUpperCase = Object.keys(obj).map(key => key.toUpperCase());
```

5: This exercises asks us to create a new object myObj that uses myProtoObj as prototype. I did this as follows:

```
let myProtoObj = {
  foo: 1,
  bar: 2,
};

let myObj = Object.create(myProtoObj);
```

6: This exercises asks us to add a qux property to myObj with value 3. I did this as follows:

```
myObj['qux'] = 3;
```

We are also asked to examine the following two snippets of code and determine whether they will produce the same output:

```
Object.keys(myObj).forEach(function(key) {
  console.log(key);
});
```

```
for (let key in myObj) {
  console.log(key);
}
```

The answer is that they will not produce the same output. The for loop will iterate through all the keys in myObj including those inherited from myProtoObj the for loop will only iterate through the keys defined explicitly for myObj and not through those inherited from myProtoObj. Hence the first snippet will log ‘foo’, ‘bar’, and ‘qux’ while the second snippet will log ‘qux’.

7: This exercise asks us to create a function that returns a copy of an object. It should both take either one or two arguments, an object and/or an object and an array of keys, returning a full copy of the object of the first case and an object containing only the key-value pairs corresponding to the keys in the array. I did this as follows:

```
function copyObj(object, array) {
  if (array) {
    let entries = [];
    for (let i = 0; i < array.length; i += 1) {
      entries.push([array[i], object[array[i]]]);
    }
    return Object.fromEntries(entries);
  } else{
    return Object.assign({},object);
  }
}

let objToCopy = {
  foo: 1,
  bar: 2,
};
```

8: This exercise asks us to explain what the following program logs to the console:

```
let foo = {
  a: 'hello',
  b: 'world',
};

let qux = 'hello';

function bar(argument1, argument2) {
  argument1.a = 'hi';
  argument2 = 'hi';
}

bar(foo, qux);

console.log(foo.a);
console.log(qux);
```

The answer is that it will log { a: ‘hi’, b: ‘world’ } and ‘hello’. This is because when bar(foo, qux) is called, foo is being passed as a reference while qux is being passed as a value. Thus what the function body does is change the a property of the object referenced by foo and reassing the value ‘hi’ to the variable argument2 which had initially been assigned the value ‘hello’; qux variable is never reassigned.