(/)

0x13. JavaScript - Objects, Scopes and Closures

JavaScript

By: Guillaume

Weight: 1

Project over - took place from Mar 14, 2023 3:00 AM to Mar 15, 2023 3:00 AM

An auto review will be launched at the deadline

In a nutshell…

- **Auto QA review:** 116.0/116 mandatory & 29.0/29 optional
- **Altogether: 200.0%**
  - Mandatory: 100.0%
  - Optional: 100.0%
  - Calculation: 100.0% + (100.0% * 100.0%) == **200.0%**

Resources

Read or watch:

- JavaScript object basics (/rltoken/dsSkBB-Cj0tqUFL8eOZLLQ)
- Object-oriented JavaScript (/rltoken/qqgqdyHPzUZkKQ5UMnw2MQ) (***read all examples!***)
- Class - ES6 (/rltoken/NEm-UViCThD5hfq_3Lj9Hg)
- super - ES6 (/rltoken/_cxdVKsdqPWbbp2cHtQSbQ)
- extends - ES6 (/rltoken/6wdl6Bc5yjBplpiZKmr6Zw)
- Object prototypes (/rltoken/NiBbDiOlfhfUf4eliggIlw)
- Inheritance in JavaScript (/rltoken/qqgqdyHPzUZkKQ5UMnw2MQ)
- Closures (/rltoken/CybTMKEDNdTdU99kx_OXgQ)
- this/self (/rltoken/XcOkisoKPud4faDDkLMABw)
- Modern JS (/rltoken/rU_q2J3qGWfvTYNlIW8JnA)

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/Eo6JxX0bkDywq4IxT8wRew), **without the help of Google**:

General

- Why JavaScript programming is amazing
- How to create an object in JavaScript
- What `this` means
- What `undefined` means
- Why the variable type and scope is important
- What is a closure
- What is a prototype
- How to inherit an object from another

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be interpreted on Ubuntu 20.04 LTS using node (version 14.x)
- All your files should end with a new line
- The first line of all your files should be exactly `#!/usr/bin/node`
- A `README.md` file, at the root of the folder of the project, is mandatory
- Your code should be `semistandard` compliant. Rules of Standard (/rltoken/CAKkGG6pUDtpu3T2rn4MXw) + semicolons on top (/rltoken/oc1-9XTUtCilyZkdAFvoUQ). Also as reference: AirBnB style (/rltoken/JvqqQQrEPtGjP-57CZSEaQ)
- All your files must be executable
- The length of your files will be tested using `wc`
- You are not allowed to use `var`

More Info

Install Node 14

```
$ curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash -
$ sudo apt-get install -y nodejs
```

Install semi-standard

Documentation (/rltoken/oc1-9XTUtCilyZkdAFvoUQ)

```
$ sudo npm install semistandard --global
```

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. Rectangle #0

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write an empty class `Rectangle` that defines a rectangle:

- You must use the `class` notation for defining your class

```
guillaume@ubuntu:~/0x13$ cat 0-main.js
#!/usr/bin/node
const Rectangle = require('./0-rectangle');

const r1 = new Rectangle();
console.log(r1);
console.log(r1.constructor);

guillaume@ubuntu:~/0x13$ ./0-main.js
Rectangle {}
[Class: Rectangle]
guillaume@ubuntu:~/0x13$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x13-javascript_objects_scopes_closures`
- File: `0-rectangle.js`

Done! | Help | Check your code | >_ Get a sandbox | QA Review

1. Rectangle #1

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a class `Rectangle` that defines a rectangle:

- You must use the `class` notation for defining your class

- The constructor must take 2 arguments `w` and `h`
- Initialize the instance attribute `width` with the value of `w`
- Initialize the instance attribute `height` with the value of `h`

```
guillaume@ubuntu:~/0x13$ cat 1-main.js
#!/usr/bin/node
const Rectangle = require('./1-rectangle');

const r1 = new Rectangle(2, 3);
console.log(r1);
console.log(r1.width);
console.log(r1.height);

const r2 = new Rectangle(2, -3);
console.log(r2);
console.log(r2.width);
console.log(r2.height);

const r3 = new Rectangle(2);
console.log(r3);
console.log(r3.width);
console.log(r3.height);

guillaume@ubuntu:~/0x13$ ./1-main.js
Rectangle { width: 2, height: 3 }
2
3
Rectangle { width: 2, height: -3 }
2
-3
Rectangle { width: 2, height: undefined }
2
undefined
guillaume@ubuntu:~/0x13$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x13-javascript_objects_scopes_closures`
- File: `1-rectangle.js`

☑ Done! | Help | Check your code | >_ Get a sandbox | QA Review

2. Rectangle #2

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a class `Rectangle` that defines a rectangle:

- You must use the `class` notation for defining your class
- The constructor must take 2 arguments `w` and `h`
- Initialize the instance attribute `width` with the value of `w`

- Initialize the instance attribute `height` with the value of `h`
- If `w` or `h` is equal to 0 or not a positive integer, create an empty object

```
guillaume@ubuntu:~/0x13$ cat 2-main.js
#!/usr/bin/node
const Rectangle = require('./2-rectangle');

const r1 = new Rectangle(2, 3);
console.log(r1);
console.log(r1.width);
console.log(r1.height);

const r2 = new Rectangle(2, -3);
console.log(r2);
console.log(r2.width);
console.log(r2.height);

const r3 = new Rectangle(2);
console.log(r3);
console.log(r3.width);
console.log(r3.height);

const r4 = new Rectangle(2, 0);
console.log(r4);
console.log(r4.width);
console.log(r4.height);

guillaume@ubuntu:~/0x13$ ./2-main.js
Rectangle { width: 2, height: 3 }
2
3
Rectangle {}
undefined
undefined
Rectangle {}
undefined
undefined
Rectangle {}
undefined
undefined
guillaume@ubuntu:~/0x13$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x13-javascript_objects_scopes_closures`
- File: `2-rectangle.js`

Done! | Help | Check your code | >_ Get a sandbox | QA Review

3. Rectangle #3

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

(/)

Write a class `Rectangle` that defines a rectangle:

- You must use the `class` notation for defining your class
- The constructor must take 2 arguments: `w` and `h`
- Initialize the instance attribute `width` with the value of `w`
- Initialize the instance attribute `height` with the value of `h`
- If `w` or `h` is equal to 0 or not a positive integer, create an empty object
- Create an instance method called `print()` that prints the rectangle using the character `X`

```
guillaume@ubuntu:~/0x13$ cat 3-main.js
#!/usr/bin/node
const Rectangle = require('./3-rectangle');

const r1 = new Rectangle(2, 3);
r1.print();

const r2 = new Rectangle(10, 5);
r2.print();

guillaume@ubuntu:~/0x13$ ./3-main.js
XX
XX
XX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
guillaume@ubuntu:~/0x13$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x13-javascript_objects_scopes_closures`
- File: `3-rectangle.js`

☑ Done! | Help | Check your code | >_ Get a sandbox | QA Review

4. Rectangle #4

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a class `Rectangle` that defines a rectangle:

- You must use the `class` notation for defining your class
- The constructor must take 2 arguments: `w` and `h`
- Initialize the instance attribute `width` with the value of `w`
- Initialize the instance attribute `height` with the value of `h`
- If `w` or `h` is equal to 0 or not a positive integer, create an empty object
- Create an instance method called `print()` that prints the rectangle using the character `X`

- Create an instance method called `rotate()` that exchanges the `width` and the `height` of the rectangle
- Create an instance method called `double()` that multiples the `width` and the `height` of the rectangle by 2

```
guillaume@ubuntu:~/0x13$ cat 4-main.js
#!/usr/bin/node
const Rectangle = require('./4-rectangle');

const r1 = new Rectangle(2, 3);
console.log('Normal:');
r1.print();

console.log('Double:');
r1.double();
r1.print();

console.log('Rotate:');
r1.rotate();
r1.print();

guillaume@ubuntu:~/0x13$ ./4-main.js
Normal:
XX
XX
XX
Double:
XXXX
XXXX
XXXX
XXXX
XXXX
XXXX
Rotate:
XXXXXX
XXXXXX
XXXXXX
XXXXXX
guillaume@ubuntu:~/0x13$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x13-javascript_objects_scopes_closures`
- File: `4-rectangle.js`

Done! | Help | Check your code | >_ Get a sandbox | QA Review

5. Square #0

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a class `Square` that defines a square and inherits from `Rectangle` of `4-rectangle.js`:

(/)

- You must use the `class` notation for defining your class and `extends`
- The constructor must take 1 argument: `size`
- The constructor of `Rectangle` must be called (by using `super()`)

```
guillaume@ubuntu:~/0x13$ cat 5-main.js
#!/usr/bin/node
const Square = require('./5-square');

const s1 = new Square(4);
s1.print();
s1.double();
s1.print();

guillaume@ubuntu:~/0x13$ ./5-main.js
XXXX
XXXX
XXXX
XXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
XXXXXXXX
guillaume@ubuntu:~/0x13$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x13-javascript_objects_scopes_closures`
- File: `5-square.js`

☑ Done! | Help | Check your code | >_ Get a sandbox | QA Review

6. Square #1

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a class `Square` that defines a square and inherits from `Square` of `5-square.js`:

- You must use the `class` notation for defining your class and `extends`
- Create an instance method called `charPrint(c)` that prints the rectangle using the character `c`
  - If `c` is `undefined`, use the character `X`

```
guillaume@ubuntu:~/0x13$ cat 6-main.js
#!/usr/bin/node
const Square = require('./6-square');

const s1 = new Square(4);
s1.charPrint();

s1.charPrint('C');

guillaume@ubuntu:~/0x13$ ./6-main.js
XXXX
XXXX
XXXX
XXXX
CCCC
CCCC
CCCC
CCCC
guillaume@ubuntu:~/0x13$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x13-javascript_objects_scopes_closures`
- File: `6-square.js`

Done! | Help | Check your code | >_ Get a sandbox | QA Review

7. Occurrences

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that returns the number of occurrences in a list:

- Prototype: `exports.nbOccurences = function (list, searchElement)`

```
guillaume@ubuntu:~/0x13$ cat 7-main.js
#!/usr/bin/node
const nbOccurences = require('./7-occurrences').nbOccurences;

console.log(nbOccurences([1, 2, 3, 4, 5, 6], 3));
console.log(nbOccurences([3, 2, 3, 4, 5, 3, 3], 3));
console.log(nbOccurences(["S", 12, "c", "S", "School", 8], "S"));

guillaume@ubuntu:~/0x13$ ./7-main.js
1
4
2
guillaume@ubuntu:~/0x13$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x13-javascript_objects_scopes_closures`
- File: `7-occurrences.js`

Done! | Help | Check your code | QA Review

8. Esrever

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that returns the reversed version of a list:

- Prototype: `exports.esrever = function (list)`
- You are not allow to use the built-in method `reverse`

```
guillaume@ubuntu:~/0x13$ cat 8-main.js
#!/usr/bin/node
const esrever = require('./8-esrever').esrever;

console.log(esrever([1, 2, 3, 4, 5]));
console.log(esrever(["School", 89, { id: 12 }, "String"]));

guillaume@ubuntu:~/0x13$ ./8-main.js
[ 5, 4, 3, 2, 1 ]
[ 'String', { id: 12 }, 89, 'School' ]
guillaume@ubuntu:~/0x13$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x13-javascript_objects_scopes_closures`
- File: `8-esrever.js`

Done! | Help | Check your code | >_ Get a sandbox | QA Review

9. Log me

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that prints the number of arguments already printed and the new argument value. (see example below)

- Prototype: `exports.logMe = function (item)`
- Output format: `<number arguments already printed>: <current argument value>`

0)

```
guillaume@ubuntu:~/0x13$ cat 9-main.js
#!/usr/bin/node
const logMe = require('./9-logme').logMe;

logMe("Hello");
logMe("Best");
logMe("School");

guillaume@ubuntu:~/0x13$ ./9-main.js
0: Hello
1: Best
2: School
guillaume@ubuntu:~/0x13$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x13-javascript_objects_scopes_closures`
- File: `9-logme.js`

Done! | Help | Check your code | QA Review

10. Number conversion

mandatory

Score: 100.0% (*Checks completed: 100.0%*)

Write a function that converts a number from base 10 to another base passed as argument:

- Prototype: `exports.converter = function (base)`
- You are not allowed to import any file
- You are not allowed to declare any new variable (`var` , `let` , etc..)

```
guillaume@ubuntu:~/0x13$ cat 10-main.js
#!/usr/bin/node
const converter = require('./10-converter').converter;

let myConverter = converter(10);

console.log(myConverter(2));
console.log(myConverter(12));
console.log(myConverter(89));


myConverter = converter(16);

console.log(myConverter(2));
console.log(myConverter(12));
console.log(myConverter(89));

guillaume@ubuntu:~/0x13$ ./10-main.js
2
12
89
2
c
59
guillaume@ubuntu:~/0x13$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x13-javascript_objects_scopes_closures`
- File: `10-converter.js`

Done! | Help | Check your code | >_ Get a sandbox | QA Review

11. Factor index

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that imports an array and computes a new array.

- Your script must import `list` from the file `100-data.js`
- You must use a `map` . Tips (/rltoken/LOEW51ZbYDjO4KZCFevzNQ)
- A new list must be created with each value equal to the value of the initial list, multipled by the index in the list
- Print both the initial list and the new list

```
guillaume@ubuntu:~/0x13$ cat 100-data.js
#!/usr/bin/node
exports.list = [1, 2, 3, 4, 5];
guillaume@ubuntu:~/0x13$ ./100-map.js
[ 1, 2, 3, 4, 5 ]
[ 0, 2, 6, 12, 20 ]
guillaume@ubuntu:~/0x13$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x13-javascript_objects_scopes_closures`
- File: `100-map.js`

☑ Done! | Help | Check your code | >_ Get a sandbox | QA Review

12. Sorted occurences

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that imports a dictionary of occurrences by user id and computes a dictionary of user ids by occurrence.

- Your script must import `dict` from the file `101-data.js`
- In the new dictionary:
  - A key is a number of occurrences
  - A value is the list of user ids
- Print the new dictionary at the end

```
guillaume@ubuntu:~/0x13$ cat 101-data.js
#!/usr/bin/node
exports.dict = {
  89: 1,
  90: 2,
  91: 1,
  92: 3,
  93: 1,
  94: 2
};
guillaume@ubuntu:~/0x13$ ./101-sorted.js
{ '1': [ '89', '91', '93' ], '2': [ '90', '94' ], '3': [ '92' ] }
guillaume@ubuntu:~/0x13$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x13-javascript_objects_scopes_closures`
- File: `101-sorted.js`

Done! | Help | Check your code | >_ Get a sandbox | QA Review

13. Concat files

#advanced

Score: 100.0% (*Checks completed: 100.0%*)

Write a script that concats 2 files.

- The first argument is the file path of the first source file
- The second argument is the file path of the second source file
- The third argument is the file path of the destination

```
guillaume@ubuntu:~/0x13$ cat fileA
C is fun!
guillaume@ubuntu:~/0x13$ cat fileB
Python is Cool!!!
guillaume@ubuntu:~/0x13$ ./102-concat.js fileA fileB fileC
guillaume@ubuntu:~/0x13$ cat fileC
C is fun!
Python is Cool!!!
guillaume@ubuntu:~/0x13$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x13-javascript_objects_scopes_closures`
- File: `102-concat.js`

Done! | Help | Check your code | >_ Get a sandbox | QA Review