


( / )





Curriculum

**Short Specializations** Average: 17.07% 

# 0x00. ES6 Basics

JavaScript

ES6

 By: Johann Kerbrat, Engineering Manager at Uber Works Weight: 1 Ongoing second chance project - started Mar 4, 2024 6:00 AM, must end by Mar 9, 2024 6:00 AM☒ An auto review will be launched at the deadline

## In a nutshell...

- **Auto QA review:** 7.0/41 mandatory & 0.0/4 optional
- **Altogether: 17.07%**
  - Mandatory: 17.07%
  - Optional: 0.0%
  - Calculation:  $17.07\% + (17.07\% * 0.0\%) == 17.07\%$

## Concepts

*For this project, we expect you to look at these concepts:*

- JavaScript programming (/concepts/852)
- Software Linter (/concepts/542)

 Help



## ES6 arrow functions

## Resources

### Read or watch:

- ECMAScript 6 - ECMAScript 2015 (/rltoken/NW1dFLFExQ12\_hD8yvkV3A)
- Statements and declarations (/rltoken/sroRUsUvOZV28V99MHDenw)
- Arrow functions (/rltoken/N2WLylppCtkkX3YFFtyUHw)
- Default parameters (/rltoken/kbw9gMO6sdeOKAY23SYVgA)
- Rest parameter (/rltoken/erZfCvacuGVk9z1CQIJvYQ)
- Javascript ES6 — Iterables and Iterators (/rltoken/kdF078LS2vjT-\_PickEr7Q)

## Learning Objectives

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

- What ES6 is
- New features introduced in ES6
- The difference between a constant and a variable
- Block-scoped variables
- Arrow functions and function parameters default to them
- Rest and spread function parameters
- String templating in ES6
- Object creation and their properties in ES6
- Iterators and for-of loops

## Requirements

### General

- All your files will be executed on Ubuntu 18.04 LTS using NodeJS 12.11.x



- Allowed editors: `vi`, `vim`, `emacs`, `Visual Studio Code`
- (/). All your files should end with a new line
- A `README.md` file, at the root of the folder of the project, is mandatory
- Your code should use the `js` extension
- Your code will be tested using the Jest Testing Framework (/rltoken/ECZpKsJ3fm1qRA7IDyhd\_Q)
- Your code will be analyzed using the linter ESLint (/rltoken/Ttd9w5jERwTErJW3DDbVoQ) along with specific rules that we'll provide
- All of your functions must be exported

## Setup

### Install NodeJS 12.11.x

(in your home directory):

```
curl -sL https://deb.nodesource.com/setup_12.x -o nodesource_setup.sh
sudo bash nodesource_setup.sh
sudo apt install nodejs -y
```

```
$ nodejs -v
v12.11.1
$ npm -v
6.11.3
```

### Install Jest, Babel, and ESLint

in your project directory, install Jest, Babel and ESLint by using the supplied `package.json` and run `npm install`.

## Configuration files

Add the files below to your project directory

### `package.json`

[Click here to show/hide file contents](#)



```
(/)  
{  
  "scripts": {  
    "lint": "./node_modules/.bin/eslint",  
    "check-lint": "lint [0-9]*.js",  
    "dev": "npx babel-node",  
    "test": "jest",  
    "full-test": "./node_modules/.bin/eslint [0-9]*.js && jest"  
  },  
  "devDependencies": {  
    "@babel/core": "^7.6.0",  
    "@babel/node": "^7.8.0",  
    "@babel/preset-env": "^7.6.0",  
    "eslint": "^6.4.0",  
    "eslint-config-airbnb-base": "^14.0.0",  
    "eslint-plugin-import": "^2.18.2",  
    "eslint-plugin-jest": "^22.17.0",  
    "jest": "^24.9.0"  
  }  
}
```

## babel.config.js

[Click here to show/hide file contents](#)

```
module.exports = {  
  presets: [  
    [  
      '@babel/preset-env',  
      {  
        targets: {  
          node: 'current',  
        },  
      },  
    ],  
  ],  
};
```

## .eslintrc.js

[Click here to show/hide file contents](#)



```
(/)  
module.exports = {  
  env: {  
    browser: false,  
    es6: true,  
    jest: true,  
  },  
  extends: [  
    'airbnb-base',  
    'plugin:jest/all',  
  ],  
  globals: {  
    Atomics: 'readonly',  
    SharedArrayBuffer: 'readonly',  
  },  
  parserOptions: {  
    ecmaVersion: 2018,  
    sourceType: 'module',  
  },  
  plugins: ['jest'],  
  rules: {  
    'no-console': 'off',  
    'no-shadow': 'off',  
    'no-restricted-syntax': [  
      'error',  
      'LabeledStatement',  
      'WithStatement',  
    ],  
  },  
  overrides:[  
    {  
      files: ['*.js'],  
      excludedFiles: 'babel.config.js',  
    }  
  ]  
};
```

## Finally...

Don't forget to run `npm install` from the terminal of your project folder to install all necessary project dependencies.

# Tasks



## 0. Const or let?

mandatory

Score: 100.0% (Checks completed: 100.0%)

(1)

## Modify

- function `taskFirst` to instantiate variables using `const`
- function `taskNext` to instantiate variables using `let`

```
export function taskFirst() {
  var task = 'I prefer const when I can.';
  return task;
}

export function getLast() {
  return ' is okay';
}

export function taskNext() {
  var combination = 'But sometimes let';
  combination += getLast();

  return combination;
}
```

## Execution example:

```
bob@dylan:~$ cat 0-main.js
import { taskFirst, taskNext } from './0-constants.js';

console.log(`${taskFirst()} ${taskNext()}`);

bob@dylan:~$
bob@dylan:~$ npm run dev 0-main.js
I prefer const when I can. But sometimes let is okay
bob@dylan:~$
```

## Repo:

- GitHub repository: `alx-backend-javascript`
- Directory: `0x00-ES6_basic`
- File: `0-constants.js`

☒ Done![Help](#)[Check your code](#)[Get a sandbox](#)[QA Review](#)

## 1. Block Scope

mandatory



Score: 66.67% (Checks completed: 66.67%)

Given what you've read about `var` and hoisting, modify the variables inside the function `taskBlock` so that the variables aren't overwritten inside the conditional block.

```
export default function taskBlock(trueOrFalse) {  
  var task = false;  
  var task2 = true;  
  
  if (trueOrFalse) {  
    var task = true;  
    var task2 = false;  
  }  
  
  return [task, task2];  
}
```

Execution:

```
bob@dylan:~$ cat 1-main.js  
import taskBlock from './1-block-scoped.js';  
  
console.log(taskBlock(true));  
console.log(taskBlock(false));  
bob@dylan:~$  
bob@dylan:~$ npm run dev 1-main.js  
[ false, true ]  
[ false, true ]  
bob@dylan:~$
```

### Repo:

- GitHub repository: `alx-backend-javascript`
- Directory: `0x00-ES6_basic`
- File: `1-block-scoped.js`

☒ Done![Help](#)[Check your code](#)[Get a sandbox](#)[QA Review](#)

## 2. Arrow functions

**mandatory**

Score: 0.0% (Checks completed: 0.0%)

Rewrite the following standard function to use ES6's arrow syntax of the function `add` (it will be an anonymous function after)



```
export default function getNeighborhoodsList() {  
  this.sanFranciscoNeighborhoods = ['SOMA', 'Union Square'];  
  
  const self = this;  
  this.addNeighborhood = function add(newNeighborhood) {  
    self.sanFranciscoNeighborhoods.push(newNeighborhood);  
    return self.sanFranciscoNeighborhoods;  
  };  
}
```

Execution:

```
bob@dylan:~$ cat 2-main.js  
import getNeighborhoodsList from './2-arrow.js';  
  
const neighborhoodsList = new getNeighborhoodsList();  
const res = neighborhoodsList.addNeighborhood('Noe Valley');  
console.log(res);  
bob@dylan:~$  
bob@dylan:~$ npm run dev 2-main.js  
[ 'SOMA', 'Union Square', 'Noe Valley' ]  
bob@dylan:~$
```

### Repo:

- GitHub repository: alx-backend-javascript
- Directory: 0x00-ES6\_basic
- File: 2-arrow.js

☒ Done![Help](#)[Check your code](#)[Get a sandbox](#)[QA Review](#)

## 3. Parameter defaults

**mandatory**

Score: 0.0% (Checks completed: 0.0%)

Condense the internals of the following function to 1 line - without changing the name of each function/variable.

*Hint:* The key here to define default parameter values for the function parameters.





```
export default function getSumOfHoods(initialNumber, expansion1989, expansion2019) {  
  if (expansion1989 === undefined) {  
    expansion1989 = 89;  
  }  
  
  if (expansion2019 === undefined) {  
    expansion2019 = 19;  
  }  
  return initialNumber + expansion1989 + expansion2019;  
}
```

Execution:

```
bob@dylan:~$ cat 3-main.js  
import getSumOfHoods from './3-default-parameter.js';  
  
console.log(getSumOfHoods(34));  
console.log(getSumOfHoods(34, 3));  
console.log(getSumOfHoods(34, 3, 4));  
bob@dylan:~$  
bob@dylan:~$ npm run dev 3-main.js  
142  
56  
41  
bob@dylan:~$
```

#### Repo:

- GitHub repository: alx-backend-javascript
- Directory: 0x00-ES6\_basic
- File: 3-default-parameter.js

☒ Done![Help](#)[Check your code](#)[Get a sandbox](#)[QA Review](#)

## 4. Rest parameter syntax for functions

**mandatory**

Score: 0.0% (Checks completed: 0.0%)

Modify the following function to return the number of arguments passed to it using the rest parameter syntax

```
export default function returnHowManyArguments() {  
  
}
```



Example:

```
returnHowManyArguments("Hello", "Holberton", 2020);  
3  
>
```

Execution:

```
bob@dylan:~$ cat 4-main.js  
import returnHowManyArguments from './4-rest-parameter.js';  
  
console.log(returnHowManyArguments("one"));  
console.log(returnHowManyArguments("one", "two", 3, "4th"));  
bob@dylan:~$  
bob@dylan:~$ npm run dev 4-main.js  
1  
4  
bob@dylan:~$
```

### Repo:

- GitHub repository: alx-backend-javascript
- Directory: 0x00-ES6\_basic
- File: 4-rest-parameter.js

☒ Done![Help](#)[Check your code](#)[Get a sandbox](#)[QA Review](#)

## 5. The wonders of spread syntax

**mandatory**

Score: 0.0% (Checks completed: 0.0%)

Using spread syntax, concatenate 2 arrays and each character of a string by modifying the function below. Your function body should be one line long.

```
export default function concatArrays(array1, array2, string) {  
}
```

Execution:



```
bob@dylan:~$ cat 5-main.js
import concatArrays from './5-spread-operator.js';

console.log(concatArrays(['a', 'b'], ['c', 'd'], 'Hello'));

bob@dylan:~$
bob@dylan:~$ npm run dev 5-main.js
[
  'a', 'b', 'c',
  'd', 'H', 'e',
  'l', 'l', 'o'
]
bob@dylan:~$
```

**Repo:**

- GitHub repository: alx-backend-javascript
- Directory: 0x00-ES6\_basic
- File: 5-spread-operator.js

☒ Done!

Help

Check your code

&gt; Get a sandbox

QA Review

**6. Take advantage of template literals**

mandatory

Score: 0.0% (Checks completed: 0.0%)

Rewrite the return statement to use a template literal so you can the substitute the variables you've defined.

```
export default function getSanFranciscoDescription() {
  const year = 2017;
  const budget = {
    income: '$119,868',
    gdp: '$154.2 billion',
    capita: '$178,479',
  };

  return 'As of ' + year + ', it was the seventh-highest income county in the United States'
    / ', with a per capita personal income of ' + budget.income + '. As of 2015, San Fra
ncisco'
    / ' proper had a GDP of ' + budget.gdp + ', and a GDP per capita of ' + budget.capit
a + '.';
}
```

Execution:



```
bob@dylan:~$ cat 6-main.js
import getSanFranciscoDescription from './6-string-interpolation.js';

console.log(getSanFranciscoDescription());

bob@dylan:~$
bob@dylan:~$ npm run dev 6-main.js
As of 2017, it was the seventh-highest income county in the United States, with a per capita
personal income of $119,868. As of 2015, San Francisco proper had a GDP of $154.2 billion, a
nd a GDP per capita of $178,479.
bob@dylan:~$
```


**Repo:**

- GitHub repository: alx-backend-javascript
- Directory: 0x00-ES6\_basic
- File: 6-string-interpolation.js

☒ Done!

Help

Check your code

 Get a sandbox

QA Review

**7. Object property value shorthand syntax**

mandatory

Score: 0.0% (Checks completed: 0.0%)

Notice how the keys and the variable names are the same?

Modify the following function's `budget` object to simply use the keyname instead.

```
export default function getBudgetObject(income, gdp, capita) {
  const budget = {
    income: income,
    gdp: gdp,
    capita: capita,
  };

  return budget;
}
```

Execution:



```
bob@dylan:~$ cat 7-main.js
import getBudgetObject from './7-getBudgetObject.js';

console.log(getBudgetObject(400, 700, 900));

bob@dylan:~$
bob@dylan:~$ npm run dev 7-main.js
{ income: 400, gdp: 700, capita: 900 }
bob@dylan:~$
```

**Repo:**

- GitHub repository: alx-backend-javascript
- Directory: 0x00-ES6\_basic
- File: 7-getBudgetObject.js

☒ Done!

Help

Check your code

&gt; Get a sandbox

QA Review

**8. No need to create empty objects before adding in properties**

mandatory

Score: 0.0% (Checks completed: 0.0%)

Rewrite the `getBudgetForCurrentYear` function to use ES6 computed property names on the `budget` object

```
function getCurrentYear() {
  const date = new Date();
  return date.getFullYear();
}

export default function getBudgetForCurrentYear(income, gdp, capita) {
  const budget = {};

  budget[`income-${getCurrentYear()}`] = income;
  budget[`gdp-${getCurrentYear()}`] = gdp;
  budget[`capita-${getCurrentYear()}`] = capita;

  return budget;
}
```

Execution:



```
bob@dylan:~$ cat 8-main.js
import getBudgetForCurrentYear from './8-getBudgetCurrentYear.js';

console.log(getBudgetForCurrentYear(2100, 5200, 1090));

bob@dylan:~$
bob@dylan:~$ npm run dev 8-main.js
{ 'income-2021': 2100, 'gdp-2021': 5200, 'capita-2021': 1090 }
bob@dylan:~$
```


**Repo:**

- GitHub repository: alx-backend-javascript
- Directory: 0x00-ES6\_basic
- File: 8-getBudgetCurrentYear.js

☒ Done!

Help

Check your code

 Get a sandbox

QA Review

**9. ES6 method properties**

mandatory

Score: 0.0% (Checks completed: 0.0%)

Rewrite `getFullBudgetObject` to use ES6 method properties in the `fullBudget` object

```
import getBudgetObject from './7-getBudgetObject.js';

export default function getFullBudgetObject(income, gdp, capita) {
  const budget = getBudgetObject(income, gdp, capita);
  const fullBudget = {
    ...budget,
    getIncomeInDollars: function (income) {
      return `$$${income}`;
    },
    getIncomeInEuros: function (income) {
      return `${income} euros`;
    },
  };

  return fullBudget;
}
```

Execution:



```
bob@dylan:~$ cat 9-main.js
import getFullBudgetObject from './9-getFullBudget.js';

const fullBudget = getFullBudgetObject(20, 50, 10);

console.log(fullBudget.getIncomeInDollars(fullBudget.income));
console.log(fullBudget.getIncomeInEuros(fullBudget.income));

bob@dylan:~$
bob@dylan:~$ npm run dev 9-main.js
$20
20 euros
bob@dylan:~$
```


**Repo:**

- GitHub repository: alx-backend-javascript
- Directory: 0x00-ES6\_basic
- File: 9-getFullBudget.js

☒ Done!

Help

Check your code

 Get a sandbox

QA Review

**10. For...of Loops**

mandatory

Score: 0.0% (Checks completed: 0.0%)

Rewrite the function `appendToEachArrayValue` to use ES6's `for...of` operator. And don't forget that `var` is not ES6-friendly.

```
export default function appendToEachArrayValue(array, appendString) {
  for (var idx in array) {
    var value = array[idx];
    array[idx] = appendString + value;
  }

  return array;
}
```

Execution:



```
bob@dylan:~$ cat 10-main.js
import appendToEachArrayValue from './10-loops.js';

console.log(appendToEachArrayValue(['appended', 'fixed', 'displayed'], 'correctly-'));

bob@dylan:~$
bob@dylan:~$ npm run dev 10-main.js
[ 'correctly-appended', 'correctly-fixed', 'correctly-displayed' ]
bob@dylan:~$
```

**Repo:**

- GitHub repository: alx-backend-javascript
- Directory: 0x00-ES6\_basic
- File: 10-loops.js

☒ Done!

Help

Check your code

&gt; Get a sandbox

QA Review

**11. Iterator**

mandatory

Score: 0.0% (Checks completed: 0.0%)

Write a function named `createEmployeesObject` that will receive two arguments:

- `departmentName` (String)
- `employees` (Array of Strings)

```
export default function createEmployeesObject(departmentName, employees) {

}
```

The function should return an object with the following format:

```
{
  $departmentName: [
    $employees,
  ],
}
```

Execution:





```
bob@dylan:~$ cat 11-main.js
import createEmployeesObject from './11-createEmployeesObject.js';

console.log(createEmployeesObject("Software", [ "Bob", "Sylvie" ]));

bob@dylan:~$
bob@dylan:~$ npm run dev 11-main.js
{ Software: [ 'Bob', 'Sylvie' ] }
bob@dylan:~$
```


**Repo:**

- GitHub repository: alx-backend-javascript
- Directory: 0x00-ES6\_basic
- File: 11-createEmployeesObject.js

☒ Done!

Help

Check your code

 Get a sandbox

QA Review

**12. Let's create a report object**

mandatory

Score: 0.0% (Checks completed: 0.0%)

Write a function named `createReportObject` whose parameter, `employeesList`, is the return value of the previous function `createEmployeesObject`.

```
export default function createReportObject(employeesList) {

}
```

`createReportObject` should return an object containing the key `allEmployees` and a method property called `getNumberOfDepartments`.

`allEmployees` is a key that maps to an object containing the department name and a list of all the employees in that department. If you're having trouble, use the spread syntax.

The method property receives `employeesList` and returns the number of departments. I would suggest thinking back to the ES6 method property syntax.

```
{
  allEmployees: {
    engineering: [
      'John Doe',
      'Guillaume Salva',
    ],
  },
};
```



Execution:

```
bob@dylan:~$ cat 12-main.js
import createEmployeesObject from './11-createEmployeesObject.js';
import createReportObject from './12-createReportObject.js';

const employees = {
  ...createEmployeesObject('engineering', ['Bob', 'Jane']),
  ...createEmployeesObject('marketing', ['Sylvie'])
};

const report = createReportObject(employees);
console.log(report.allEmployees);
console.log(report.getNumberOfDepartments(report.allEmployees));

bob@dylan:~$
bob@dylan:~$ npm run dev 12-main.js
{ engineering: [ 'Bob', 'Jane' ], marketing: [ 'Sylvie' ] }
2
bob@dylan:~$
```


**Repo:**

- GitHub repository: alx-backend-javascript
- Directory: 0x00-ES6\_basic
- File: 12-createReportObject.js

☒ Done!

Help

Check your code

 Get a sandbox

QA Review

## 13. Iterating through report objects

#advanced

Score: 0.0% (Checks completed: 0.0%)

Write a function named `createIteratorObject`, that will take into argument a report Object created with the previous function `createReportObject`.

This function will return an iterator to go through every employee in every department.

```
export default function createIteratorObject(report) {

}
```

Execution:



```
bob@dylan:~$ cat 100-main.js
import createIteratorObject from './100-createIteratorObject.js';

import createEmployeesObject from './11-createEmployeesObject.js';
import createReportObject from './12-createReportObject.js';

const employees = {
  ...createEmployeesObject('engineering', ['Bob', 'Jane']),
  ...createEmployeesObject('marketing', ['Sylvie'])
};

const report = createReportObject(employees);

const reportWithIterator = createIteratorObject(report);

for (const item of reportWithIterator) {
  console.log(item);
}

bob@dylan:~$
bob@dylan:~$ npm run dev 100-main.js
Bob
Jane
Sylvie
bob@dylan:~$
```


**Repo:**

- GitHub repository: alx-backend-javascript
- Directory: 0x00-ES6\_basic
- File: 100-createIteratorObject.js

☒ Done!

Help

Check your code

 Get a sandbox

QA Review

**14. Iterate through object**

#advanced

Score: 0.0% (Checks completed: 0.0%)

Finally, write a function named `iterateThroughObject`. The function's parameter `reportWithIterator` is the return value from `createIteratorObject`.

```
export default function iterateThroughObject(reportWithIterator) {

}
```



It should return every employee name in a string, separated by |

```
(/
  allEmployees: {
    engineering: [
      'John Doe',
      'Guillaume Salva',
    ],
  },
  ...
};
```

Should return John Doe | Guillaume Salva

Reminder - the functions will be *imported* by the test suite.

Full example:

```
> employees = {
  ...createEmployeesObject('engineering', engineering),
  ...createEmployeesObject('design', design),
};
>
> const report = createReportObject(employees);
> const reportWithIterator = createIteratorObject(report);
> iterateThroughObject(reportWithIterator)
'John Doe | Guillaume Salva | Kanye East | Jay Li'
>
```

Execution:

```
bob@dylan:~$ cat 101-main.js
import createEmployeesObject from './11-createEmployeesObject.js';
import createReportObject from './12-createReportObject.js';
import createIteratorObject from './100-createIteratorObject.js';
import iterateThroughObject from './101-iterateThroughObject.js';

const employees = {
  ...createEmployeesObject('engineering', ['Bob', 'Jane']),
  ...createEmployeesObject('marketing', ['Sylvie'])
};

const report = createReportObject(employees);
const reportWithIterator = createIteratorObject(report);

console.log(iterateThroughObject(reportWithIterator));

bob@dylan:~$
bob@dylan:~$ npm run dev 101-main.js
Bob | Jane | Sylvie
bob@dylan:~$
```



**Repo:**

- GitHub repository: alx-backend-javascript
- Directory: 0x00-ES6\_basic
- File: 101-iterateThroughObject.js

☒ Done![Help](#)[Check your code](#)[>\\_ Get a sandbox](#)[QA Review](#)

Copyright © 2024 ALX, All rights reserved.

