Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections/to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

0x01. ES6 Promises

JavaScript ES6

Weight: 1

Project over - took place from Jun 25, 2024 6:00 AM to Jun 27, 2024 6:00 AM

An auto review will be launched at the deadline

In a nutshell…

- **Auto QA review:** 24.0/26 mandatory & 4.0/4 optional
- **Altogether: 184.62%**
  - Mandatory: 92.31%
  - Optional: 100.0%
  - Calculation: 92.31% + (92.31% * 100.0%) == **184.62%**

ONE SIMPLY DOES NOT USE ASYNC/AWAIT
WITHOUT KNOWING PROMISES!
imgflip.com

Resources

Read or watch:

- Promise (/rltoken/j_0FTFbkTg42JMcAbNPOVQ)
- JavaScript Promise: An introduction (/rltoken/2Q2LzNFokcUwpA2u3FKG6Q)
- Await (/rltoken/0Xb3S2PMBe-SLJ55isMcow)

- Async (/rltoken/_K0C7pgEjwaIzU9RpwCb8g)
- Throw / Try (/rltoken/UTjDgvKk5l892Xslh0vqcQ)

(/)

Home(/)

My Planning(/planning/me)

Projects(/projects/current)

QA Reviews I can make(/corrections/to_review)

Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)

Curriculums(/dashboards/my_curriculums)

Concepts(/concepts)

Conference rooms(/dashboards/video_rooms)

Servers(/servers)

Sandboxes(/user_containers/current)

Tools(/dashboards/my_tools)

Video on demand(/dashboards/videos)

Peers(/users/peers)

Discord(https://discord.com/app)

My Profile(/users/my_profile)

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/Z4xW7_BFaRcrHxfDySjKuQ), **without the help of Google**:

- Promises (how, why, and what)
- How to use the `then`, `resolve`, `catch` methods
- How to use every method of the Promise object
- Throw / Try
- The await operator
- How to use an `async` function

Requirements

- 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 `Jest` and the command `npm run test`
- Your code will be verified against lint using ESLint
- 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 ESList by using the supplied `package.json` and run `npm install`.

Configuration Files

Add the files below to your project directory

(/)
Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections/to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

`package.json`

Click to show/hide file contents

`babel.config.js`

Click to show/hide file contents

`utils.js`

Use when you get to tasks requiring `uploadPhoto` and `createUser`.

Click to show/hide file contents

`.eslintrc.js`

Click to show/hide file contents

and…

Don't forget to run `$ npm install` when you have the `package.json`

Response Data Format

`uploadPhoto` returns a response with the format

```
{
  status: 200,
  body: 'photo-profile-1',
}
```

`createUser` returns a response with the format

```
{
  firstName: 'Guillaume',
  lastName: 'Salva',
}
```

Tasks

0. Keep every promise you make and only make promises you can keep

mandatory

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

Return a Promise using this prototype `function getResponseFromAPI()`

```
bob@dylan:~$ cat 0-main.js
import getResponseFromAPI from "./0-promise.js";

const response = getResponseFromAPI();
console.log(response instanceof Promise);

bob@dylan:~$
bob@dylan:~$ npm run dev 0-main.js
true
bob@dylan:~$
```

Repo:

- GitHub repository: `alx-backend-javascript`
- Directory: `0x01-ES6_promise`
- File: `0-promise.js`

Check submission | >_ Get a sandbox | View results

1. Don't make a promise...if you know you can't keep it

mandatory

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

Using the prototype below, return a `promise`. The parameter is a `boolean`.

```
getFullResponseFromAPI(success)
```

When the argument is:

- `true`
  - resolve the promise by passing an object with 2 attributes:
    - `status`: `200`
    - `body`: `'Success'`
- `false`
  - reject the promise with an error object with the message `The fake API is not working currently`

Try testing it out for yourself

(/)
Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections/to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

```
bob@dylan:~$ cat 1-main.js
import getFullResponseFromAPI from './1-promise';

console.log(getFullResponseFromAPI(true));
console.log(getFullResponseFromAPI(false));

bob@dylan:~$
bob@dylan:~$ npm run dev 1-main.js
Promise { { status: 200, body: 'Success' } }
Promise {
  <rejected> Error: The fake API is not working currently
    ...
    ...
bob@dylan:~$
```

Repo:

- GitHub repository: `alx-backend-javascript`
- Directory: `0x01-ES6_promise`
- File: `1-promise.js`

Check submission | >_ Get a sandbox | View results

2. Catch me if you can!

mandatory

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

Using the function prototype below

```
function handleResponseFromAPI(promise)
```

Append three handlers to the function:

- When the Promise resolves, return an object with the following attributes
  - `status`: `200`
  - `body`: `success`
- When the Promise rejects, return an empty `Error` object
- For every resolution, log `Got a response from the API` to the console

```
bob@dylan:~$ cat 2-main.js
import handleResponseFromAPI from "./2-then";

const promise = Promise.resolve();
handleResponseFromAPI(promise);

bob@dylan:~$
bob@dylan:~$ npm run dev 2-main.js
Got a response from the API
bob@dylan:~$
```

(/)
Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections/to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

(/)
Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections/to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

Repo:

- GitHub repository: `alx-backend-javascript`
- Directory: `0x01-ES6_promise`
- File: `2-then.js`

Check submission | Get a sandbox | View results

3. Handle multiple successful promises

mandatory

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

In this file, import `uploadPhoto` and `createUser` from `utils.js`

Knowing that the functions in `utils.js` return promises, use the prototype below to collectively resolve all promises and log `body firstName lastName` to the console.

```
function handleProfileSignup()
```

In the event of an error, log `Signup system offline` to the console

```
bob@dylan:~$ cat 3-main.js
import handleProfileSignup from "./3-all";

handleProfileSignup();

bob@dylan:~$ 
bob@dylan:~$ npm run dev 3-main.js 
photo-profile-1 Guillaume Salva
bob@dylan:~$ 
```

Repo:

- GitHub repository: `alx-backend-javascript`
- Directory: `0x01-ES6_promise`
- File: `3-all.js`

Check submission | Get a sandbox | View results

4. Simple promise

mandatory

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

Using the following prototype

```
function signUpUser(firstName, lastName) {
}
```

That returns a resolved promise with this object:

```
{
  firstName: value,
  lastName: value,
}
```

```
bob@dylan:~$ cat 4-main.js
import signUpUser from "./4-user-promise";

console.log(signUpUser("Bob", "Dylan"));

bob@dylan:~$
bob@dylan:~$ npm run dev 4-main.js
Promise { { firstName: 'Bob', lastName: 'Dylan' } }
bob@dylan:~$
```

Repo:

- GitHub repository: `alx-backend-javascript`
- Directory: `0x01-ES6_promise`
- File: `4-user-promise.js`

Check submission | >_ Get a sandbox | View results

5. Reject the promises

mandatory

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

Write and export a function named `uploadPhoto`. It should accept one argument `fileName` (string).

The function should return a Promise rejecting with an Error and the string `$fileName cannot be processed`

```
export default function uploadPhoto(filename) {
}
```

(/)
Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections_to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

```
bob@dylan:~$ cat 5-main.js
import uploadPhoto from './5-photo-reject';

console.log(uploadPhoto('guillaume.jpg'));

bob@dylan:~$
bob@dylan:~$ npm run dev 5-main.js
Promise {
  <rejected> Error: guillaume.jpg cannot be processed
  ..
    ..
bob@dylan:~$
```

Repo:

- GitHub repository: `alx-backend-javascript`
- Directory: `0x01-ES6_promise`
- File: `5-photo-reject.js`

Check submission | Get a sandbox | View results

6. Handle multiple promises

mandatory

Score: 50.0% (*Checks completed: 50.0%*)

Import `signUpUser` from `4-user-promise.js` and `uploadPhoto` from `5-photo-reject.js`.

Write and export a function named `handleProfileSignup`. It should accept three arguments `firstName` (string), `lastName` (string), and `fileName` (string). The function should call the two other functions. When the promises are all settled it should return an array with the following structure:

```
[
    {
      status: status_of_the_promise,
      value: value or error returned by the Promise
    },
    ...
  ]
```

```
bob@dylan:~$ cat 6-main.js
import handleProfileSignup from './6-final-user';

console.log(handleProfileSignup("Bob", "Dylan", "bob_dylan.jpg"));

bob@dylan:~$
bob@dylan:~$ npm run dev 6-main.js
Promise { <pending> }
bob@dylan:~$
```

(/)
Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections/to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

(/)
Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections/to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

Repo:

- GitHub repository: `alx-backend-javascript`
- Directory: `0x01-ES6_promise`
- File: `6-final-user.js`

Check submission | Mark submission | >_ Get a sandbox | View results

7. Load balancer

mandatory

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

Write and export a function named `loadBalancer`. It should accept two arguments `chinaDownload` (Promise) and `USDownload` (Promise).

The function should return the value returned by the promise that resolved the first.

```
export default function loadBalancer(chinaDownload, USDownload) {

}
```

```
bob@dylan:~$ cat 7-main.js
import loadBalancer from "./7-load_balancer";

const ukSuccess = 'Downloading from UK is faster';
const frSuccess = 'Downloading from FR is faster';

const promiseUK = new Promise(function(resolve, reject) {
    setTimeout(resolve, 100, ukSuccess);
});

const promiseUKSlow = new Promise(function(resolve, reject) {
    setTimeout(resolve, 400, ukSuccess);
});

const promiseFR = new Promise(function(resolve, reject) {
    setTimeout(resolve, 200, frSuccess);
});

const test = async () => {
    console.log(await loadBalancer(promiseUK, promiseFR));
    console.log(await loadBalancer(promiseUKSlow, promiseFR));
}

test();

bob@dylan:~$
bob@dylan:~$ npm run dev 7-main.js
Downloading from UK is faster
Downloading from FR is faster
bob@dylan:~$
```

(/)
Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections/to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

Repo:

- GitHub repository: `alx-backend-javascript`
- Directory: `0x01-ES6_promise`
- File: `7-load_balancer.js`

Check submission | >_ Get a sandbox | View results

8. Throw error / try catch

mandatory

Score: 66.67% (*Checks completed: 66.67%*)

Write a function named `divideFunction` that will accept two arguments: `numerator` (Number) and `denominator` (Number).

When the `denominator` argument is equal to 0, the function should throw a new error with the message `cannot divide by 0`. Otherwise it should return the numerator divided by the denominator.

```
export default function divideFunction(numerator, denominator) {

}
```

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

console.log(divideFunction(10, 2));
console.log(divideFunction(10, 0));

bob@dylan:~$
bob@dylan:~$ npm run dev 8-main.js
5
..../8-try.js:15
    throw Error('cannot divide by 0');
    ^
.....

bob@dylan:~$
```

Repo:

- GitHub repository: `alx-backend-javascript`
- Directory: `0x01-ES6_promise`
- File: `8-try.js`

Check submission | Mark submission | >_ Get a sandbox | View results

(/)
Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections/to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)

9. Throw an error

mandatory

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

Write a function named `guardrail` that will accept one argument `mathFunction` (Function).

This function should create and return an array named `queue` .

When the `mathFunction` function is executed, the value returned by the function should be appended to the queue. If this function throws an error, the error message should be appended to the queue. In every case, the message `Guardrail was processed` should be added to the queue.

Example:

```
[
  1000,
  'Guardrail was processed',
]
```

```
bob@dylan:~$ cat 9-main.js
import guardrail from './9-try';
import divideFunction from './8-try';

console.log(guardrail(() => { return divideFunction(10, 2)}));
console.log(guardrail(() => { return divideFunction(10, 0)}));

bob@dylan:~$
bob@dylan:~$ npm run dev 9-main.js
[ 5, 'Guardrail was processed' ]
[ 'Error: cannot divide by 0', 'Guardrail was processed' ]
bob@dylan:~$
```

Repo:

- GitHub repository: `alx-backend-javascript`
- Directory: `0x01-ES6_promise`
- File: `9-try.js`

Check submission | Get a sandbox | View results

10. Await / Async

#advanced

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

Import `uploadPhoto` and `createUser` from `utils.js`

Write an async function named `asyncUploadUser` that will call these two functions and return an object with the following format:

```
{
  photo: response_from_uploadPhoto_function,
  user: response_from_createUser_function,
}
```

If one of the async function fails, return an empty object. Example:

```
{
  photo: null,
  user: null,
}
```

```
bob@dylan:~$ cat 100-main.js
import asyncUploadUser from "./100-await";

const test = async () => {
    const value = await asyncUploadUser();
    console.log(value);
};

test();

bob@dylan:~$
bob@dylan:~$ npm run dev 100-main.js
{
  photo: { status: 200, body: 'photo-profile-1' },
  user: { firstName: 'Guillaume', lastName: 'Salva' }
}
bob@dylan:~$
```

Repo:

- GitHub repository: `alx-backend-javascript`
- Directory: `0x01-ES6_promise`
- File: `100-await.js`

Check submission　Get a sandbox　View results

(/)
Home(/)
My Planning(/planning/me)
Projects(/projects/current)
QA Reviews I can make(/corrections/to_review)
Evaluation quizzes(/dashboards/my_current_evaluation_quizzes)
Curriculums(/dashboards/my_curriculums)
Concepts(/concepts)
Conference rooms(/dashboards/video_rooms)
Servers(/servers)
Sandboxes(/user_containers/current)
Tools(/dashboards/my_tools)
Video on demand(/dashboards/videos)
Peers(/users/peers)
Discord(https://discord.com/app)
My Profile(/users/my_profile)