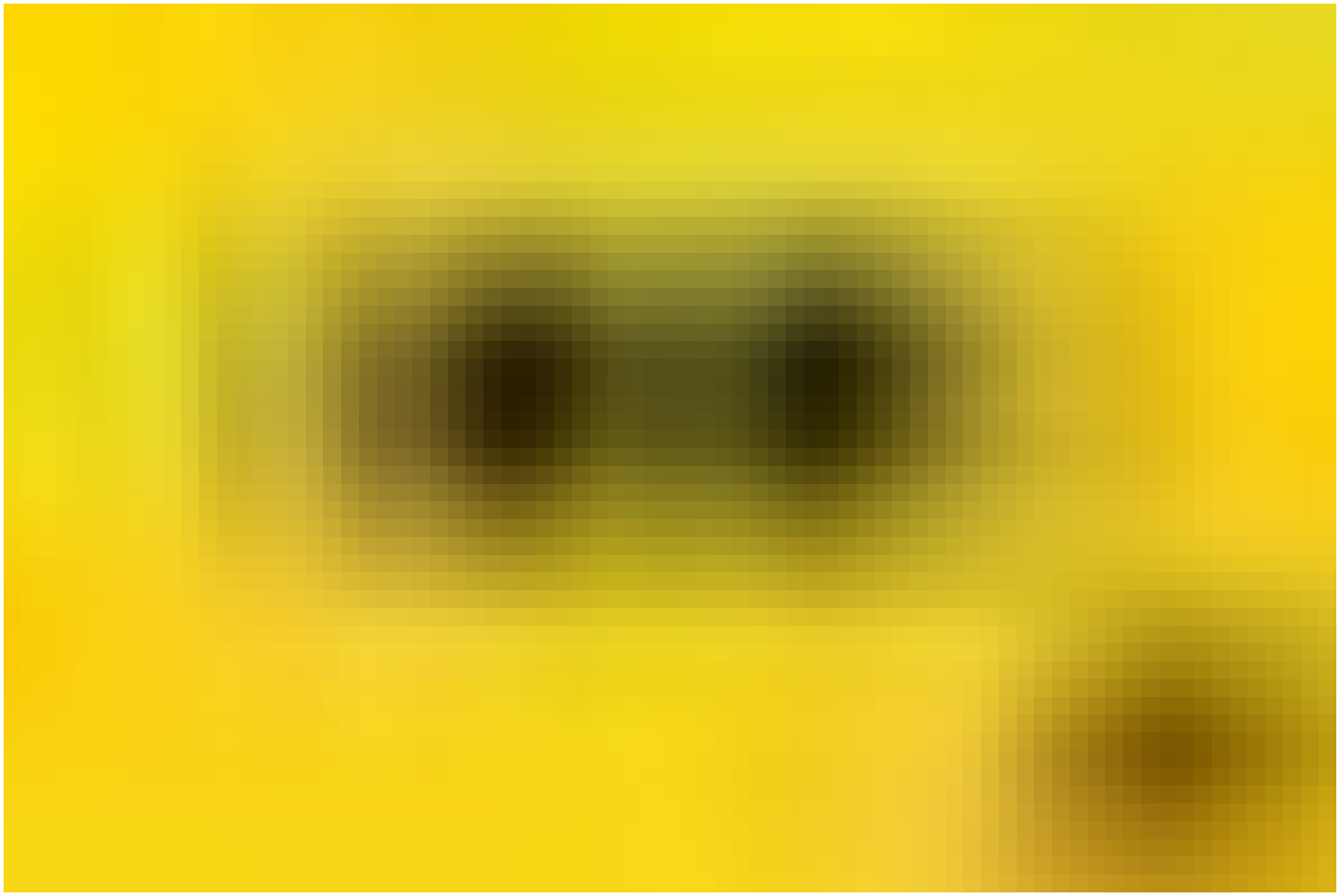
Max Silva Follow
Web / Dev / Design / Blockchain
Apr 6 · 2 min read



# Javascript ES8 asynchronous await service + ES6 Modules

ES8 asynchronous fetch from different modules.

> *Click here to share this article on LinkedIn »*

. . .

I recommend to get familiar with the latest versions of ECMAScript or “ECMA-262". Really good resources that helped me can be found

here. Jeffery Way has an excellent series on ES2015+ for beginners here.

Imagine we have an ES6 class file containing methods which all need to fetch data from different sources?

```
(function () {
   'use strict';
}());


export default class someClass {
    constructor() {
        this.users =
'https://jsonplaceholder.typicode.com/users';
        this.posts
='https://jsonplaceholder.typicode.com/posts';


        this.getUsers();
        this.getPosts();
    }


    getUsers() {
        fetch(this.posts)
        .then(response => response.json())
        .then(data => console.log(data))
        .catch(error => console.log('Error:', error));
    }


    getPosts() {
        fetch(this.posts)
        .then(response => response.json())
        .then(data => console.log(data))
        .catch(error => console.log('Error:', error));
    }
}
```

As you can see above getUsers() and getPosts() contain virtually the same logic a fetch request. We are duplicating code here and this can be wrapped in a ES8 sugar the asynchronous await function. Lets begin by changing the class code to something more concise:-

```
(function () {
   'use strict';
}());


import { getData } from './GetDataHelper.js';
```

```
export default class someClass {
    constructor() {
        this.users =
'https://jsonplaceholder.typicode.com/users';
        this.posts
='https://jsonplaceholder.typicode.com/posts';

        this.getUsers();
        this.getPosts();
    }

    getUsers() {
        getData(this.users);
    }

    getPosts() {
        getData(this.posts);
    }
}
```

We can now create a new file GetDataHelper.js:-

```
(function () {
   'use strict';
}());

// async function
export async function getData(url) {
    return await (await(fetch(url))).json();
}
```

So in summary we are now importing getData into the class as a module. This module could contain multiple methods if we wanted to use more methods within our class we can change the import to:-

```
import { getData, someFunc, anotherFunc } from
'./GetDataHelper.js';
```

or all using an asterisk.

```
import { * } from './GetDataHelper.js';
```

We are awaiting for the promise to fetch the JSON, using our new async method in the new module. This can be made more complicated if you require but as you can see this is now just very concise and clean. All is required is to transpile the ES2015+ code using a converter such as babel. A more complicated example of me needing to use the ES8 async function with HTML5's local storage functionality can be seen on my latest work.

The class file is here.

The caching service (HTML5—local storage is here.

The async request service (getData helper) is here.