When to use async/await

One of the first things that caught my attention when ES2015 was first released in 2015 were promises. They provide a nice alternative to callbacks and made the code easier for me to reason through.

In 2017 TC39 (the group in charge of Javascript standardization) added async/await, a way to make promise code easier to read and reason through, to tje ECMA262 specification, esentially writing promise-based asynchronous code as it was synchronous blocks of code that, as developers, we're already familiar with.

The idea sounds simple enough but I always trip on the execution. This post is my attempt at figuring out the differences between promises and async/await and how to use them.

First, let's look at a promise-based example. The code below fetches the latest post from a site's WordPress Rest API, parses it as JSON and then logs it to the console.

```
function fetchData() {
  fetch('https://publishing-project.rivendellweb.net/wp-json/wp/v2/posts?p
  .then((response) => {
    if (!response.ok) {
      throw new Error("HTTP error, status = " + response.status);
    }
    return response.json();
  })
  .then((data) => {
    this.data = data;
    console.log('Success:', this.data);
  })
  .catch((error) => {
    console.error('Error:', error);
  });
}
```

This is the code I'm most familiar with and it's the way I've done asycnrhonous code since 2015. For someone just learning Javascript this may look overtly complicated, particularly when we add `.then()` statements to do more work with the code we fetched.

The `async/await` version of the code uses try/catch blocks to separate the actions we take to fetch the data and the actions to take when there' an error.

```
async function fetchData() {
  try {
    const response = await fetch('https://publishing-project.rivendellweb
    const data = await response.json();
    console.log(data);
    return data;
  }
  catch(error) {
    console.log('fetch failed', error);
  }
};
```

The issue with this code is that data is only visible inside the `try` block in the async function. If I try to access the data outside I get an error because the data is not available.

```
(async () => {
  try {
    const response = await fetch('https://publishing-project.rivendellweb
    const data = await response.json();
    console.log(data);
  }
  catch(error) {
    console.log('fetch failed', error);
  }
})();
```

Links and Resources

- Async functions: making promises friendly — web.dev
- async function — MDN