VALENTINOGAGLIARDI

LAST UPDATED 18TH APRIL 2018 BY VALENTINO GAGLIARDI

How to Throw Errors From Async Functions in Javascript? (and how to test them)

It is possible to **throw errors from async functions in Javascript**?

```
async function whatever() {
  try {
    const valentinogagliardi = new Person("valentinogagliardi");
    await valentinogagliardi.getData();
    // do stuff with the eventual result and return something
  } catch (error) {
    throw Error(error);
  }
}

whatever().catch(err => console.error(err));
```

The topic has been covered hundred of times but let's see it from a **TDD** standpoint.

Answer the question without looking at Stackoverflow.

If you know the answer, well I'm impressed.

If not that's cool too. Keep reading and you'll find it!

Table of Contents

1. How to Throw Errors From Async Functions in Javascript: what you will learn
2. How to Throw Errors From Async Functions in Javascript: requirements
3. How to Throw Errors From Regular Functions in Javascript
4. How to Throw Errors From Async Functions in Javascript: testing exceptions
5. How to Throw Errors From Async Functions in Javascript: catch me if you can
6. How to Throw Errors From Async Functions in Javascript: wrapping up

How to Throw Errors From Async Functions in Javascript: what you will learn

In the following post you'll learn:

- how to throw errors from async functions in Javascript
- how to test exception from async functions with Jest

How to Throw Errors From Async Functions in Javascript: requirements

To follow along you should have:

- a basic understanding of **Javascript and ES6**
- a working installation of Node.Js and Jest

How to Throw Errors From Regular Functions in Javascript

Use exceptions rather than return codes (Clean code).

Throwing errors is a best practice for dealing with unknowns.

The same rule applies for every modern language: Java, Javascript, Python, Ruby.

You can throw errors from a function, consider the following example in Javascript:

```
function upperCase(name) {
  if (typeof name !== "string") {
    throw TypeError("name must be a string");
  }
  return name.toUpperCase();
}

module.exports = upperCase;
```

And here is the test for it (I'm using Jest):

```
"use strict";

const assert = require("assert");
const upperCase = require("../function");

describe("upperCase function", () => {
  test("it throws when name is not provided", () => {
    assert.throws(() => upperCase());
  });
  test("it throws when name is not a string", () => {
    assert.throws(() => upperCase(9));
  });
});
```

You can throw errors from ES6 classes too.

I always throw in the constructor for unexpected values when writing classes in Javascript.

A contrived example:

```
class Person {
  constructor(name) {
    if (typeof name !== "string") {
      throw TypeError("name must be a string");
    }

    this.name = name;

  }

  // some method here
}

module.exports = Person;
```

And here is the test for the class:

```
"use strict";

const assert = require("assert");
const Person = require("../index");

describe("Person class", () => {
  test("it throws when name is not provided", () => {
    assert.throws(() => new Person());
  });
  test("it throws when name is not a string", () => {
    assert.throws(() => new Person(9));
  });
});
```

The test indeed passes:

```
PASS  test/index.test.js
 Person class
   ✓ it throws when name is not provided (1ms)
   ✓ it throws when name is not a string
```

Neat!

So everything works as expected whether you're throwing from a regular function or from a class constructor (or from a method).

What if I want to **throw an error from an async function**?

Can I still use assert.throws in my test?

Let's find out.

How to Throw Errors From Async Functions in Javascript: testing exceptions

So you know Javascript async functions right?

Given the previous class:

```
class Person {
  constructor(name) {
    if (typeof name !== "string") {
      throw TypeError("name must be a string");
    }
```

```
6.
7.      this.name = name;
8.
9.    }
10.
11.   // some method here
12. }
13.
14. module.exports = Person;
```

suppose you want to add an **async method** for fetching data about that person.

Such method takes a url.

If the url is not a string we throw an error like we did in the previous example.

Let's update the class:

```
1.  class Person {
2.    constructor(name) {
3.      if (typeof name !== "string") {
4.        throw TypeError("name must be a string");
5.      }
6.
7.      this.name = name;
8.    }
9.
10.   async getData(url) {
11.     if (typeof url !== "string") {
12.       throw TypeError("url must be a string");
13.     }
14.     // const response = await fetch(url)
15.     // do stuff
16.   }
17. }
18.
19. module.exports = Person;
```

What will happen if I run the code?

Let's try:

```
1.  const Person = require("../index");
2.  const valentinogagliardi = new Person("valentinogagliardi");
3.  valentinogagliardi.getData();
```

And here it is:

```
1.  UnhandledPromiseRejectionWarning: Unhandled promise rejection (rejection id: 1):
    TypeError: name must be a string
```

```
2.
3.  DeprecationWarning: Unhandled promise rejections are deprecated. In the future,
    promise rejections that are not handled will terminate the Node.js process with a
    non-zero exit code.
```

Unsurprisingly the **async method returns a Promise rejection** but it doesn't throw in the strict sense.

The error is wrapped inside a Promise rejection.

In other words **I cannot use assert.throws** for testing it.

Let's confirm with a test:

```
1.  "use strict";
2.
3.  const assert = require("assert");
4.  const Person = require("../index");
5.
6.  describe("Person methods", () => {
7.    test("it throws when url is not a string", () => {
8.      const valentinogagliardi = new Person("valentinogagliardi");
9.      assert.throws(() => valentinogagliardi.getData());
10.   });
11. });
```

The test fails as expected!

```
1.  FAIL  test/index.test.js
2.    Person methods › it throws when url is not a string
3.
4.     assert.throws(function)
5.
6.     Expected the function to throw an error.
7.     But it didn't throw anything.
8.
9.     Message:
10.      Missing expected exception.
```

So? What's the **catch**? (No pun intended).

Drum roll...

How to Throw Errors From Async Functions in Javascript: catch me if you can

Async functions and async methods do not throw errors in a strict sense.

Async functions and async methods **always return a Promise, either resolved or rejected.**

You must attach **then() and catch()**, no matter what.

(Or wrap the method inside try/catch).

A **rejected Promise will propagate up in the stack unless you catch it**.

As for the test here's how it should be:

```
"use strict";

const assert = require("assert");
const Person = require("../index");

describe("Person methods", () => {
  test("it rejects when url is not a string", async () => {
    expect.assertions(1);
    const valentinogagliardi = new Person("valentinogagliardi");
    await expect(valentinogagliardi.getData()).rejects.toEqual(
      TypeError("url must be a string")
    );
  });
});
```

We must test not the plain exception but the **rejects** with a TypeError.

Now the test passes:

```
PASS  test/index.test.js
 Person methods
   ✓ it rejects when url is not a string
```

And how about the code?

To **catch the error** you would refactor like so:

```
const Person = require("../index");

const valentinogagliardi = new Person("valentinogagliardi");
valentinogagliardi
  .getData()
  .then(res => res)
  .catch(err => console.error(err));
```

Now the exception will show up in the console:

```
TypeError: url must be a string
    at Person.getData (/home/valentino/Documenti/articles-and-broadcasts/throw-from-async-functions-2018-04-02/index.js:12:13)
    at Object.<anonymous> (/home/valentino/Documenti/articles-and-broadcasts/throw-from-async-functions-2018-04-02/index.js:22:4)
    // ...
```

There is an important thing to note if you like more try/catch.

The following code won't catch the error:

```
const Person = require("../index");

async function whatever() {
  try {
    const valentinogagliardi = new Person("valentinogagliardi");
    await valentinogagliardi.getData();
    // do stuff with the eventual result and return something
  } catch (error) {
    throw Error(error);
  }
}

whatever();
```

Remember: a rejected Promise will propagate up in the stack unless you **catch** it.

To catch the error properly in try/catch you would refactor like so:

```
async function whatever() {
  try {
    const valentinogagliardi = new Person("valentinogagliardi");
    await valentinogagliardi.getData();
    // do stuff with the eventual result and return something
  } catch (error) {
    throw Error(error);
  }
}

whatever().catch(err => console.error(err));
```

That's how it works.

How to Throw Errors From Async Functions in Javascript: wrapping up

To recap:

Throwing error from an async function **won't spit out** a "**plain exception**".

Async functions and async methods always return a Promise, either resolved or rejected.

To intercept exceptions from async functions you must use **catch()**.

And here are the rules for **testing exceptions in Jest**:

- use assert.throws for testing exceptions in normal functions and methods
- use expect + rejects for testing exceptions in async functions and async methods

If you're interested in testing Koa 2 with Jest check out this concise introduction to testing with Jest and Supertest.

Thanks for reading!

About Latest Posts

Valentino Gagliardi

Consultant, Developer Coach. Are you stuck on a project? Let's talk!

JAVASCRIPT

One Reply to “How to Throw Errors From Async Functions in Javascript? (and how to test them)”

Pingback: How To Use Async Await in React? (componentDidMount Async)