Boris Okunskiy Follow
Engineer @ UB
Oct 6, 2016 · 2 min read

ES7 async/await: trouble in paradise

Hi, I'm your average magpie developer. Like all other magpies super-excited about new shiny ECMAScript features I've been waiting for async/await to land into JavaScript for almost 3 years now. I've been waiting for it like it's
a holy grail, or an answer to life, or an artificial intelligence which will write all my code from now on, or a mix of all these things.

And this time has come. Yay!

Well, this only removes some transpiling from the picture. Babel made it possible for more than a year now, so we already grew accustomed to async/await.

And we learned some pitfalls, because there definitely are some.

Forgotten await of eternal regret

First thing everyone notices is how you need to train yourself to write ***await*** every so often. What happens if you don't?

You're officially screwed. Not only the things that were supposed to run in sequence are now running in parallel, you're now also have **absolutely no visibility** that it happens by looking in your code.

And yeah, your errors are now swallowed (luckily, all runtimes kindly warn you about unhandled rejections). But anyway, good luck finding it.

The easiness of accidentally forgetting ***await*** is particularly disturbing. But hey, it's roughly the same as forgetting to ***return*** a promise inside ***.then***, right?

Promise of no return

This one has struck me the other night. I woke up in cold sweat and ran to my laptop to confirm/infirm my worst nightmare.

> Is there any difference between ***return promise*** and ***return await promise*** inside ***async function***? Can they be used interchangeably?

I used to think that there is absolutely no semantic differences between the two, with following reasoning: since every async function returns a Promise, it should be irrelevant from caller's point of view whether execution pauses inside a function or not—as long as it resolves with our value.

What if I told you™ that they are NOT equivalent?

Consider this (or actually try it with Chrome 53):

```
async function asyncHello() {
    try {
        return asyncHi();
    } catch (e) {
        console.log('Caught you', e);
    }
}

async function asyncHi() {
    throw new Error('Bug!');
}
```

Now call it. WTF just happened? Where's my "Caught you" in console?

Ok, now try again with ***return await asyncHi()***. Makes sense now?

Now look at the code again (the one without ***await***). I think it works pretty much as expected, don't you? And it *should* work like that.

Don't let these fancy new keywords trick you: there is no free lunch, asynchronous code is still asynchronous, async/await is not "magical", it does not liberate you from understanding Promises and mastering your Promise-fu.

. . .

Oh, good. Looks like I still need to reason about my code (kinda decreases possibility of being replaced by AI in near future).