A Promise-Based Worldview

Once the Node.js developer's victim of choice, `Promise` *now enjoys healthy adoption on both sides of the stack, with prominent open-source contributors like @sindresorhus visibly leveraging them.*
What's changed?

Nicolás Bevacqua

Published 8 months ago | *2 minute read* | 💬 *0*

Just a thought.

What was once considered a mistake by many – inclusion of `Promise` as a native language API – is now seeing increasingly high rates of adoption. I'd say there's three reasons for this.

The most obvious one is – recursively – the fact that `Promise` is now in the language. `Promise` being in the language drives adoption by itself, through the official endorsement of the spec-writing JavaScript gods. Furthermore, libraries like Babel and `bluebird` making it all too simple to include spec-compliant `Promise` -based solutions in any app at very little cost. Anyone writing code for modern browsers or using a modern development toolchain leverages Babel and/or `bluebird` .

Another reason is that **developers are increasingly comfortable with ES6**. There have been plenty of tutorials, a couple of books, and many conference talks describing ES6. It's been roughly plenty of tutorials, a couple of books, and many conference talks describing ES6. It's been roughly a year since the specification was finalized. People now roughly understand the `Promise` API, and what's better: the API isn't changing anymore. Recently, jQuery 3 was released into *Promises/A+* compliance, a huge win for native `Promise`.

Third, there's `async` / `await`. While not the most heavily utilized JavaScript API, `async` / `await` is already a stage 3 proposal, and at this point I think we can state confidently and without hesitation that it'll someday be an official language feature.

The synergy between `Promise`, `async` / `await`, and generators, is just too good to pass up!