

Use Promises Instead of Callbacks with `promisify-node`

OSCON, Austin, TX • May 8-11 • Save 20% PC20DWALSH

By [David Walsh](#) on October 28, 2015

1 [Twitter](#) [Facebook](#) [Google+](#) [Email](#) [Bitcoin](#)

One of the reasons we love promises so much is because they allows us to avoid the infamous callback hell that we've all experienced in these early days of Node.js. When I see an API that doesn't use the promise pattern, I get annoyed. Luckily I've found [promisify-node](#), a module that wraps functions or objects in a promise wrapper so you can avoid the callback mess!

{Track:js}



There are a few different ways to use `promisify-node`. The first is wrapping a single function in the promise:

```
var promisify = require('promisify-node');

function async(callback) {
  callback(null, true);
}

// Convert the function to return a Promise.
var wrap = promisify(async);

// Invoke the newly wrapped function.
wrap().then(function(value) {
  console.log(value === true);
});
```

You could even recursively wrap a Node.js module's functions:

```
var promisify = require('promisify-node');
var fs = promisify('fs');

// This function has been identified as an asynchronous function so it has
// been automatically wrapped.
fs.readFile('/etc/passwd').then(function(contents) {
  console.log(contents);
});
```

And then you can wrap an object's methods:

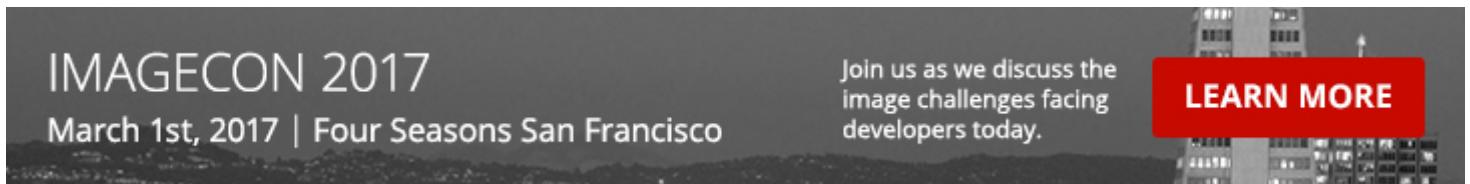
```
var promisify = require('promisify-node');

var myObj = {
  myMethod: function(a, b, cb) {
    cb(a, b);
  }
};

// No need to return anything as the methods will be replaced on the object.
promisify(myObj);

// Intentionally cause a failure by passing an object and inspect the message.
myObj.myMethod({ msg: 'Failure!' }, null).then(null, function(err) {
  console.log(err.msg);
});
```

Since many front-end APIs are moving to Promise-based APIs, it would be awesome to use something like Promisify to get into the habit of using them on both the server and client sides. Be warned, however, that this module uses a snippet of code to [detect function arguments](#). If you don't use a frequently-used callback argument name, like `callback` or `cb`, the promisify-wrapped function may not work correctly.



Recent Features



9 Mind-Blowing WebGL Demos

As much as developers now loathe Flash, we're still playing a bit of catch up to natively duplicate the animation capabilities that Adobe's old technology provided us. Of course we have [canvas](#), an awesome technology, one which I highlighted [9 mind-blowing demos](#). Another technology available...

Introducing MooTools Templated

One major problem with creating UI components with the MooTools JavaScript framework is that there isn't a great way of allowing customization of template and ease of node creation. As of today, there are two ways of creating: new Element Madness The first way to create UI-driven...

Incredible Demos



Advanced CSS Printing – Using JavaScript Double-Click To Remove Unwanted DIVs

Like any good programmer, I'm constantly searching around the internet for ideas and articles that can help me improve my code. There are thousands of talented programmers out there so I stumble upon some great articles and code snippets that I like to print out...

MooTools 1.2 OpenLinks Plugin

I often incorporate tools into my customers' websites that allow them to have some control over the content on their website. When doing so, I offer some tips to my clients to help them keep their website in good shape. One of the tips...

Discussion

John Szwarconek



Since you suggested using this on the front-end, it should be noted that production front-end code is often minified. Minification will change the variable names of the callback params. I'm posting this to prevent someone from doing a bunch of work and then hitting a gotcha going to production.

Name Email Website

Wrap your code in `<pre class="{Language}"></pre>` tags, link to a GitHub gist, JSFiddle fiddle, or CodePen pen to embed!

Continue this conversation via email

[Post Comment!](#)

[Use Code Editor](#)

