Your API Is Bad

(And You Should Feel Bad)

Paddy Foran

Your API Is Bad

(And You Should Feel Bad)

Paddy Foran

This book is for sale at http://leanpub.com/yourapiisbad

This version was published on 2014-06-21

Tweet This Book!

Please help Paddy Foran by spreading the word about this book on Twitter!

The suggested tweet for this book is:

My API is bad, and I feel bad. Get the free book: http://yourapiisbad.com #yourapiisbad

The suggested hashtag for this book is #yourapiisbad.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search?q=#yourapiisbad

Dedicated to all the innocent people I've accidentally hit with a flipped table while working with APIs. I'm sorry.

Contents

1 Introduction

I love APIs. I've been working with them for years now; writing them, writing against them, and discussing them. When APIs are done right, they're powerful, semantic definitions of a useful piece of functionality.

Let's try something. I want you to think of a definition of a chair. It should encompass everything that *is* a chair and nothing that *is not* a chair. Does it fit swivel chairs? Stools? Chairs with three legs? Does it include a bench? Do you accidentally include a table? What makes a chair a chair?

Defining things is incredibly hard.

This is why most APIs aren't done right.

Why Are APIs Important?

A lot of companies don't offer APIs that any third-party developer can write software against. And that's okay. A lot of companies *shouldn't* provide such an API.

But every company should have an API.

The era of single-device computing is over. Whether it's tablets or phones or some new category of device that we haven't seen yet, it is no longer reasonable for you to expect that everyone will use your website. It would be nice if it were; the web was designed to work on a variety of devices. But the capabilities are too varied and the pace of standardisation too slow to incorporate these capabilities into the web natively. As much as I love the web and what it represents, to offer the best user experience on mobile devices, you can't just rely on the web interface.

You need an API your mobile apps can talk to.

And if your mobile apps require an API, your website should be an unprivileged consumer of that API. That is, your website should not offer any access to your service not exposed by your API. Otherwise, you're going to have A Bad Time™ replicating that feature in your mobile clients.

APIs are the write-once-run-everywhere of today. They're the abstraction that allows you to draw your business logic into a single place.

They're not going anywhere, so making them easy to use is worth the effort.

Your API Is Bad (and You Should Feel Bad)

I've written client libraries for more APIs than I care to think about. Sometimes against APIs I wrote and defined myself. More often than not, it was an incredibly painful experience.

Why?

Most API providers don't want using the API to be a painful experience. On the contrary, they want the experience to be as pleasant and simple as possible. So why are the majority of APIs painful to use?

Because it's hard to write an API that is a pleasure to use. It's exceedingly tricky. You need to resist the temptation to constrain your API to a set of predefined use cases, or you've just written an application with a terrible interface. An API should be simple definition of building blocks that you build an application out of. But these building blocks need to be fundamental enough that you can build wildly different experiences from them, to best take advantage of each platform you're building for. And defining things at that level means reducing them to their core components, and deeply understanding those components.

A lot of APIs also don't take into account what it's like to actually use that API. How hard is it to figure out what caused an error and what the error means? What useful information is being provided? How do I need to contort my code to make sense of the data being returned by the API? What extra work is being created for me that could be avoided by a little forethought from the API designers?

Errors are my personal pet peeve. A lot of the time, an error tells you nothing more than "your request is bad and you should feel bad". There's no useful information at all.

My response is "your API is bad and you should feel bad". And that's why I'm writing this—I don't want to have to work with bad APIs.

This book is not a description of a good API. That is extremely subjective, and specifying Properties Of A Good API would only encourage people to blindly follow the "rules". *Protip*: if you're blindly following *anything*, your API is bad and you should feel bad. Think about each decision completely. Consider the effects on your API's usability. Consider what each decision *means*, semantically.

Instead of giving you rules, I'm going to take you on a tour of things that make me flip a table in rage when I work with APIs. Things that API designers didn't think about or consider, things that people change when I point them out. So I'm pointing them out here. Consider them ahead of time, and you'll be on your way to a truly elegant API.

And you won't have to feel bad.