The Evolving API: Designing While Requirements Are Still In Flux
Shaun Gallagher, Monetate
When are requirements "in flux"?

- Uncertainty
  - Not having enough information to discern present or future requirements
- Change
  - New information, circumstances, or goals affect existing requirements
Examples of uncertainty

• You know some of your users' needs and desires, but you don't yet know how all of your users will use your API *in practice*.

• Architectural decisions on which your API will depend are still being made.

• Business decisions that will affect how your API is implemented are still being made.
Examples of change

• Your users are using your API in ways you did not anticipate.
• Underlying services on which your API depends have changed.
• You want to support new features.
• You need to address bugs.
It's all about contracts

- All functions have contracts.
- Your API is a function.
- Therefore ... your API has a contract.
What does the contract cover?

- Who can access this function?
  - public or private
- Are there side effects?
  - will same input always lead to same result?
- Will this function change?
Will this function change?

• If you can change the callers, you can (usually) change the function.
• If you can't change the callers, you need a stricter contract around change.
Deprecation policy: part of the contract

- Distinguish between tracks
  - Alpha, beta, GA
- Distinguish between types of changes
  - Minor, disruptive, emergency
- Distinguish between sunset stages
  - Deprecated, retired, decommissioned
Deprecation policy: part of the contract

- How much notice will consumers receive before deprecation?
- How will they be notified?
  - Release notes, emails, in API
- How much time will they have to address breaking changes?
Versioning: yup, part of the contract

- Common versioning policy
  - Compatible change: no uprev
  - Breaking change: uprev minor version
  - Overhaul: uprev major version
- Semantic Versioning policy
  - Uprev on even compatible changes
Versioning: caveats

- "Some people, when confronted with a problem, think, 'I know, I’ll use versioning.' Now they have 2.1.0 problems." – Brandon Byers
- "Every conditional is death." – Jeff Persch
Versioning: caveats

switch (version) {
    case 1: do_this();
        break;
    case 2: do_this_other_thing();
        break;
    case 3: do_it_differently();
        break;
    case 4: do_it_some_other_way();
        break;
}
Versioning: advice

• Use judiciously, when unavoidable.
• Look for ways to transform, rather than diverge.
Release notes for in-flux APIs

• The more in-flux the API, the more developer-driven documentation and release notes should be.
• Cover the contract:
  • What has changed?
  • Whom does it affect?
  • What actions need to be taken?
  • When do those actions need to be taken?
  • Where can I find out more?
Getting feedback

• APIs need UX designers just as much as GUIs do.
• Dare to cross frontend-backend boundaries for design feedback.
Naming

• Naming things is hard, and all the more so in the context of APIs.
• Naming conventions are useful, and all the more so in the context of APIs.
• Write tests around the stupid stuff.
• Renaming: handle as early as possible.
Flexible functions

• Consider flexible data structures for inputs and output
• Do you actually need positional arguments?
Declarative APIs

- Users specify the outcome they want, not the actions they want to take.
- Change can still happen, but it's hidden in the application layer.
- GraphQL and other *QL-based APIs tend to be declarative.
Permissive APIs

- Can you ignore unrecognized parameters?
  
  /api/endpoint?count=10&order=desc&mayonnaise=no

- If you do ignore, surface the fact that you've ignored.
Discoverable APIs

• In a mature/strict REST API, hypermedia controls are included in responses.
• A new kind of contract: API consumers agree to query for a resource's hypermedia controls, rather than looking them up in the docs. This allows you to update your API scheme without introducing a breaking change.
Shaun Gallagher

Web: shaungallagher.pressbin.com

Twitter: @shaun_gallagher
Monetate

We're hiring: monetate.com/jobs

We use AI to personalize customer experiences.
Experimenting With Babies: 50 Amazing Science Projects You Can Perform on Your Kid

experimentingwithbabies.com

In bookstores and online retailers
Experiments for Newlyweds: 50 Amazing Science Projects You Can Perform With Your Spouse

newlywed.science

On sale April 2019
Correlated:
Surprising Connections Between Seemingly Unrelated Things

correlated.org

Not in bookstores anymore, but still available online