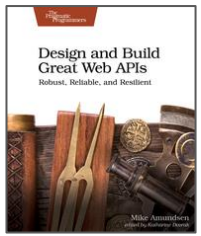
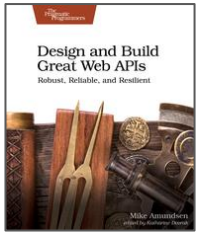
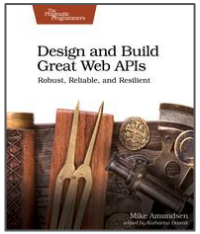
GraphQL, gRPC and REST, Oh My!

A unified API design method

Mike Amundsen
@mamund
youtube.com/mamund





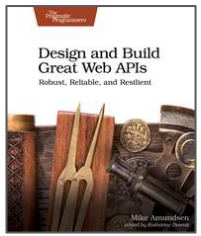


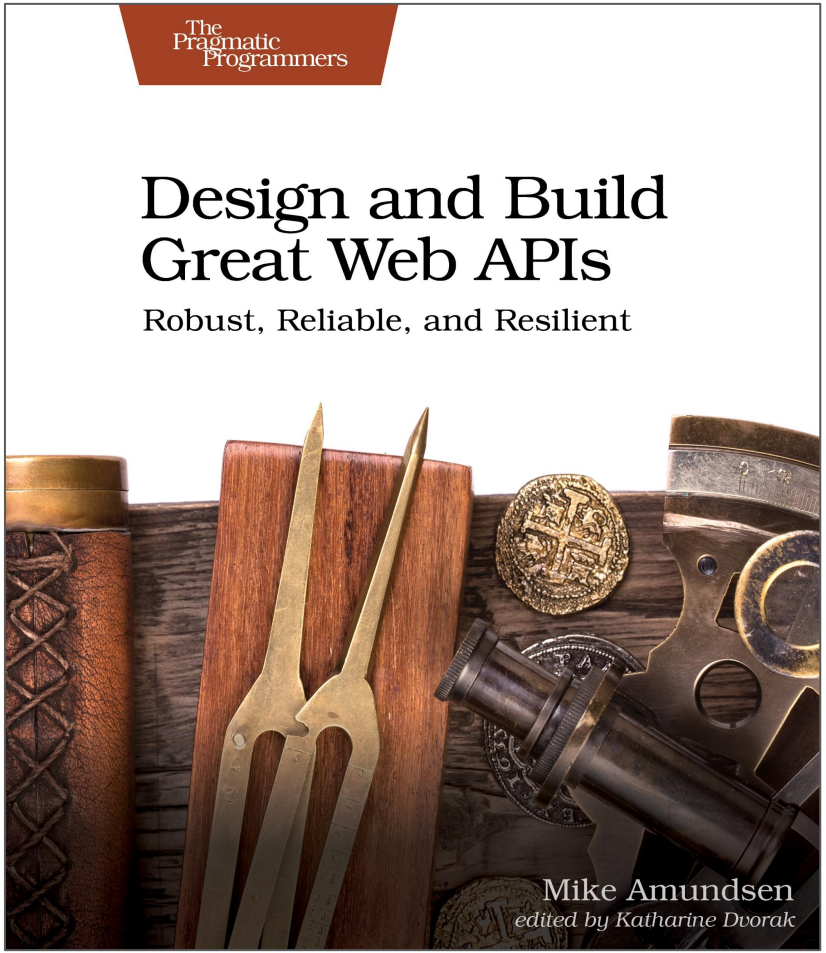

Mike Amundsen
@mamund

g.mamund.com/GreatWebAPIs

"From design to code to test to deployment, unlock hidden business value and release stable and scalable web APIs that meet customer needs and solve important business problems in a consistent and reliable manner."

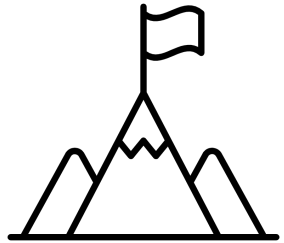
-- Pragmatic Publishers

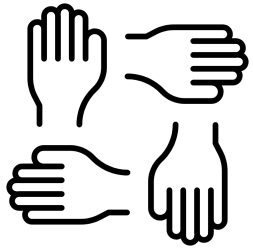


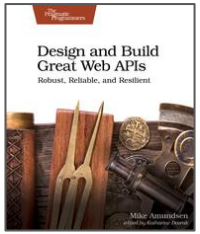


Overview

- A Story of API Design and Governance
- The Challenge of HTTP-centric API Design
- A Unified Method for API Design



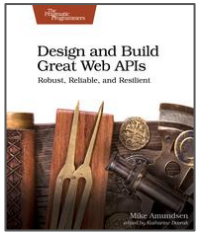




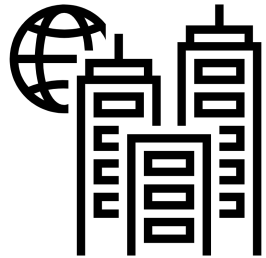
A Story of API Design and Governance

A Story of API Design and Governance

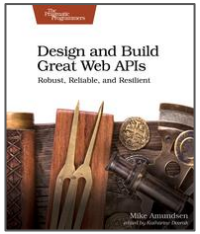
- A large company committed to strong API Design practice
- They determined OpenAPI as the backbone of the practice
- Investing training, tooling, and consistent design reviews
- All was going fine until....
- Now they have to commit to multiple parallel practices



A Story of API Design and Governance



- **A large company committed to strong API Design practice**
  - ***Needed a consistent practice in order to scale up API community***
  - ***Made design the responsibility of a central, enterprise-level body***
- They determined OpenAPI as the backbone of the practice
- Investing training, tooling, and consistent design reviews
- All was going fine until....
- Now they have to commit to multiple parallel practices

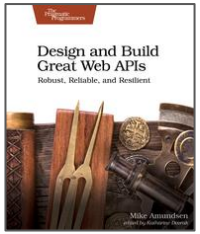


A Story of API Design and Governance

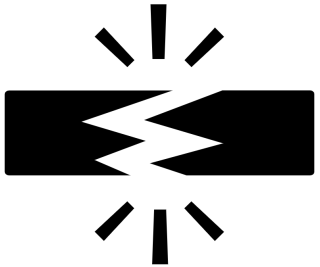
- A large company committed to strong API Design practice
- **They determined OpenAPI as the backbone of the practice**
  - ***Researched options, common usage, general guidance***
  - ***Mapped out holistic approach to API design***
- Investing training, tooling, and consistent design reviews
- All was going fine until....
- Now they have to commit to multiple parallel practices

A Story of API Design and Governance

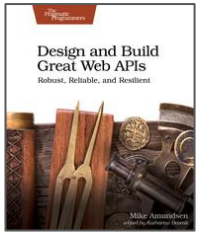
- A large company committed to strong API Design practice
- They determined OpenAPI as the backbone of the practice
- **Investing training, tooling, and consistent design reviews**
  - ***Created courses and wrote guidance documents***
  - ***Built a custom OpenAPI editor/linter/catalog system***
  - ***Hired/trained full-time design review teams***
- All was going fine until....
- Now they have to commit to multiple parallel practices



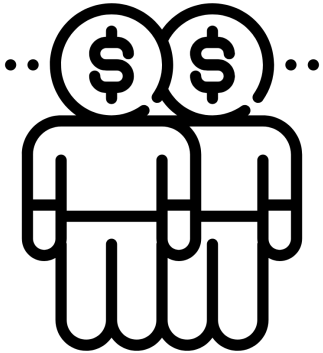
A Story of API Design and Governance



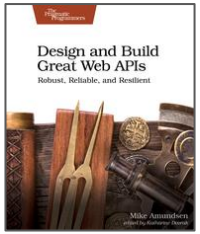
- A large company committed to strong API Design practice
- They determined OpenAPI as the backbone of the practice
- Investing training, tooling, and consistent design reviews
- **All was going fine until....**
  - ***They wanted to start using GraphQL***
  - ***None of their training, tools, or processes applied anymore***
- Now they have to commit to multiple parallel practices



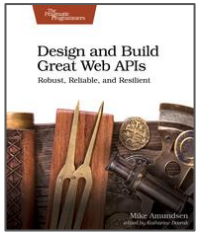
A Story of API Design and Governance

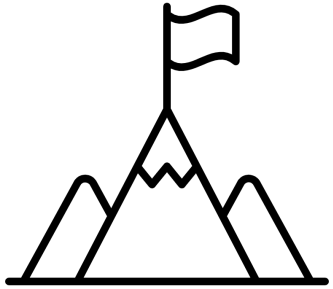


- A large company committed to strong API Design practice
- They determined OpenAPI as the backbone of the practice
- Investing training, tooling, and consistent design reviews
- All was going fine until....
- **Now they have to commit to multiple parallel practices**
  - ***Added training, tools, review teams, etc.***
  - ***Each new style/tech means another process track***

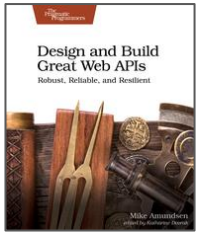


What's going on here?

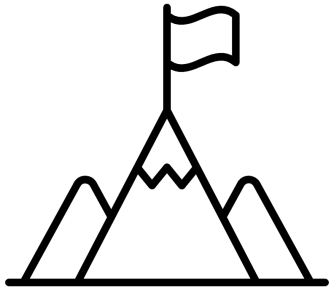




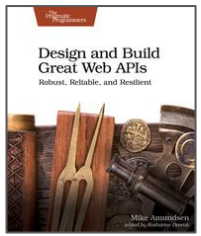
The Challenge of HTTP-centric API Design



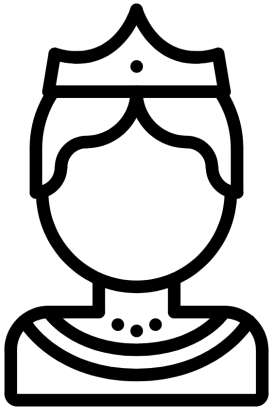
The Challenge



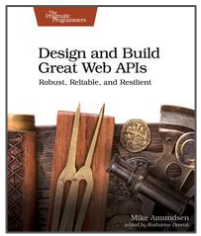
- HTTP reigns supreme
- HTTP-centric implementation leads to HTTP-centric design
- HTTP-centric design leads to HTTP-centric definitions
- HTTP-centric definitions lead to HTTP-centric governance
- Introducing other implementations (graphQL, etc.) breaks everything



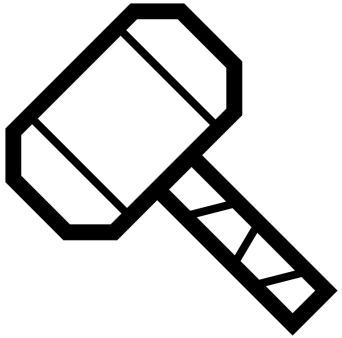
The Challenge



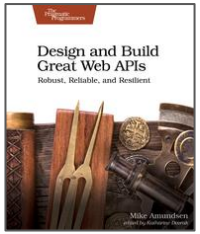
- **HTTP reigns supreme**
  - ***Most people start w/ HTTP-based APIs***
  - ***Lots of tools and training focuses on HTTP-based APIs***
- HTTP-centric implementation leads to HTTP-centric design
- HTTP-centric design leads to HTTP-centric definitions
- HTTP-centric definitions lead to HTTP-centric governance
- Introducing other implementations (graphQL, etc.) breaks everything



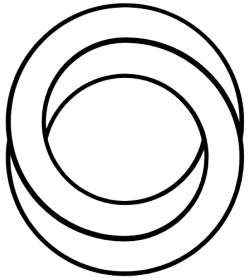
The Challenge



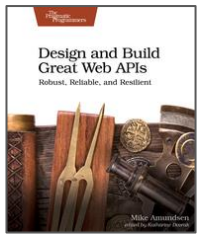
- HTTP reigns supreme
- **HTTP-centric implementation leads to HTTP-centric design**
  - ***"When all you have is a hammer…"***
  - ***HTTP-elements become design-elements (URIs, Methods, Headers, etc.)***
- HTTP-centric design leads to HTTP-centric definitions
- HTTP-centric definitions lead to HTTP-centric governance
- Introducing other implementations (graphQL, etc.) breaks everything



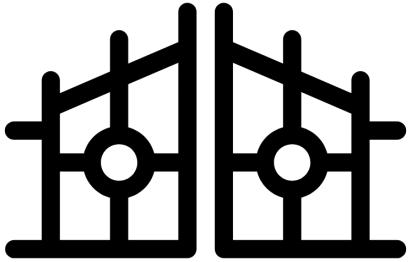
The Challenge



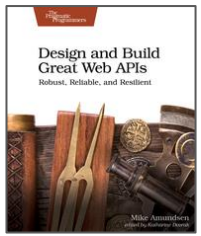
- HTTP reigns supreme
- HTTP-centric implementation leads to HTTP-centric design
- **HTTP-centric design leads to HTTP-centric definitions**
  - ***OpenAPI is HTTP-specific, but now we need lots of API definition languages***
  - ***AsyncAPI, protobuf, Scheme Definition Language, SOAP, etc.***
- HTTP-centric definitions lead to HTTP-centric governance
- Introducing other implementations (graphQL, etc.) breaks everything



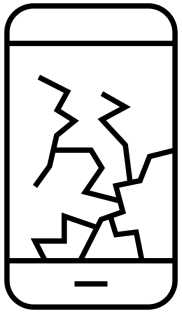
The Challenge



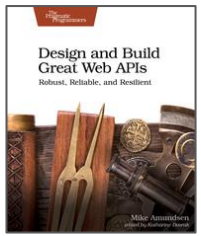
- HTTP reigns supreme
- HTTP-centric implementation leads to HTTP-centric design
- HTTP-centric design leads to HTTP-centric definitions
- **HTTP-centric definitions lead to HTTP-centric governance**
  - ***OpenAPI becomes the company's 'gatekeeper' technology***
  - ***You have to duplicate governance efforts; one for each implementation stack***
- Introducing other implementations (graphQL, etc.) breaks everything

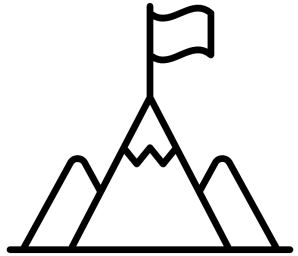


The Challenge

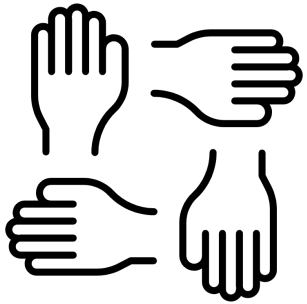


- HTTP reigns supreme
- HTTP-centric implementation leads to HTTP-centric design
- HTTP-centric design leads to HTTP-centric definitions
- HTTP-centric definitions lead to HTTP-centric governance
- **Introducing other implementations (graphQL, etc.) breaks everything**
  - ***Different design/review rules, different implementation tools, different monitoring, etc.***
  - ***Slows experimentation, exploration, and roll-out of innovative solutions***

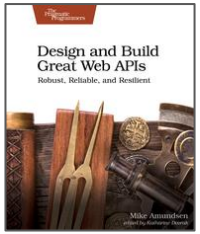




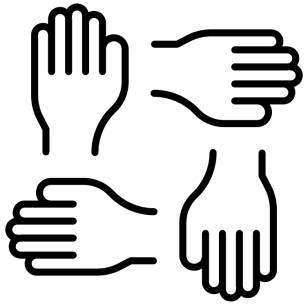
OK, how do we solve this?



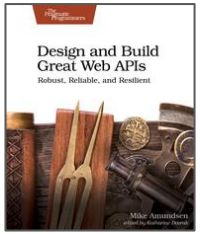
A Unified Method for API Design



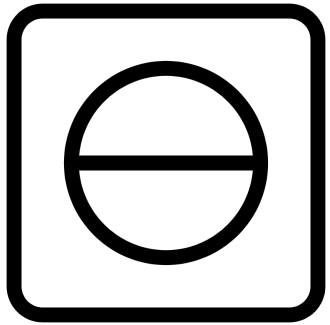
A Unified Design Solution



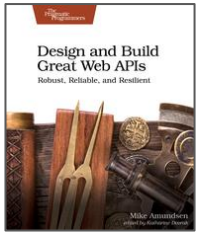
- Use design methods that don't rely on HTTP-specifics
- Focus on interface properties and actions instead
- Use an interface description language (ALPS) for designs
- Translate design language into implementation definitions (SDL, proto, etc.)



A Unified Design Solution

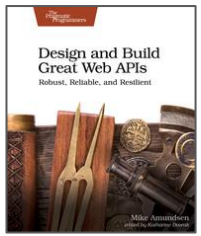


- **Use design methods that don't rely on HTTP-specifics**
  - ***Don't start with CRUD or resource-based designs***
  - ***Don't design URLs, resources, headers, status codes, methods, etc.***
- Focus on interface properties and actions instead
- Use an interface description language (ALPS) for designs
- Translate design language into implementation definitions (SDL, proto, etc.)

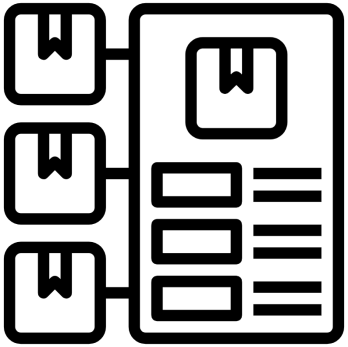


A Unified Design Solution

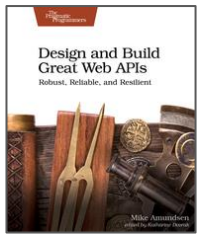
- Use design methods that don't rely on HTTP-specifics
- **Focus on interface properties and actions instead**
  - ***Define properties (***`givenName`***,*** `smsNumber`***, etc.), not objects***
  - ***Define actions (***`input-transform-output`***) , not HTTP resources and methods***
- Use an interface description language (ALPS) for designs
- Translate design language into implementation definitions (SDL, proto, etc.)



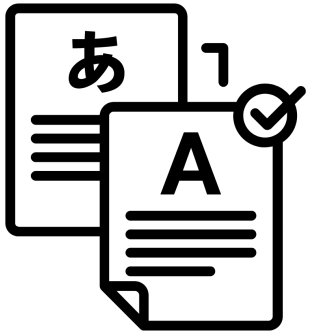
A Unified Design Solution



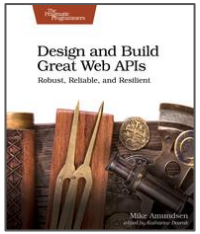
- Use design methods that don't rely on HTTP-specifics
- Focus on interface properties and actions instead
- **Use an interface description language (ALPS) for designs**
  - ***Dublin Core Application Profiles (2005)***
  - ***Application-Level Profile Semantics (2015)***
- Translate design language into implementation definitions (SDL, proto, etc.)

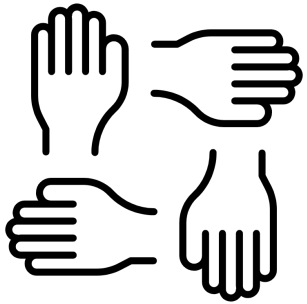


A Unified Design Solution

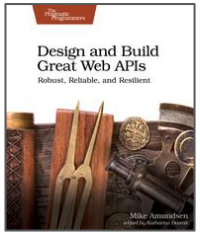


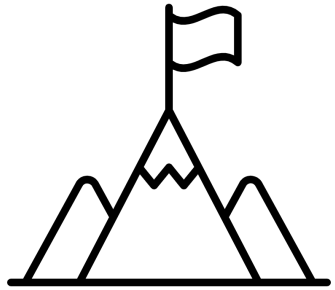
- Use design methods that don't rely on HTTP-specifics
- Focus on interface properties and actions instead
- Use an interface description language (ALPS) for designs
- **Translate designs into implementation definitions (SDL, proto, etc.)**
  - ***ALPS --> OpenAPI***
  - ***ALPS --> AsyncAPI***
  - ***ALPS --> protobuf***
  - ***ALPS --> SDL***
  - ***etc...***





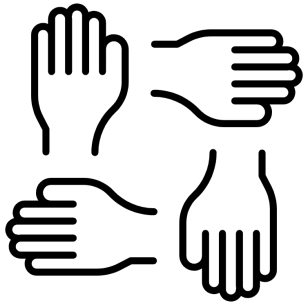
Let's see some examples....



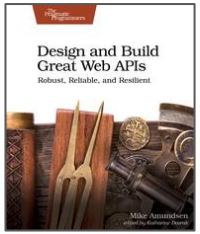


So...

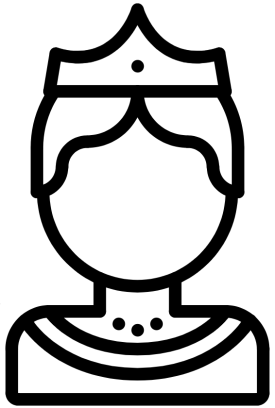
A Method for Unified API Design



- Break the HTTP-centric grip on your API design process
- Embrace interface descriptions (ALPS)
- Enable translations (OpenAPI, AsyncAPI, SDL, proto, etc.)
- Future-proof your design process

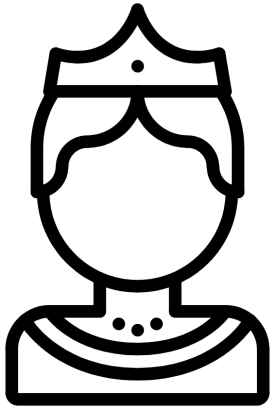


A Method for Unified API Design



- **Break the HTTP-centric grip on your API design process**
  - ***Stop using URLs, Methods, Resources, & Status Codes as design elements***
- Embrace interface descriptions (ALPS)
- Enable translations (OpenAPI, AsyncAPI, SDL, proto, etc.)
- Future-proof your design process

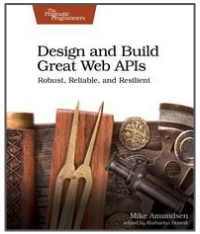
A Method for Unified API Design



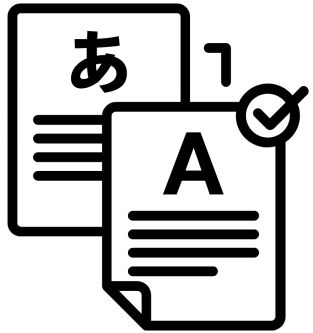
- **Break the HTTP-centric grip on your API design process**
  - ***Stop using URLs, Methods, Resources, & Status Codes as design elements***
- Embrace interface descriptions (ALPS)
- Enable translations (OpenAPI, AsyncAPI, SDL, proto, etc.)
- Future-proof your design process

A Method for Unified API Design

- Break the HTTP-centric grip on your API design process
- **Embrace interface descriptions (ALPS)**
  - ***Stick to using properties and actions to describe your API designs***
- Enable translations (OpenAPI, AsyncAPI, SDL, proto, etc.)
- Future-proof your design process

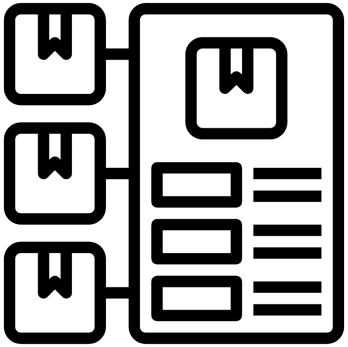


A Method for Unified API Design

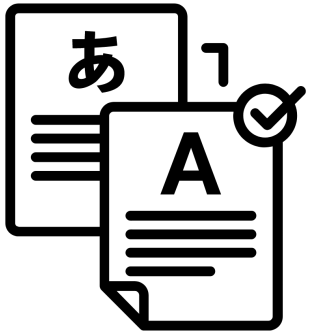


- Break the HTTP-centric grip on your API design process
- Embrace interface descriptions (ALPS)
- **Enable translations (OpenAPI, AsyncAPI, SDL, proto, etc.)**
  - ***Translate your unified design documents into implementation-specific definitions***
- Future-proof your design process

A Method for Unified API Design

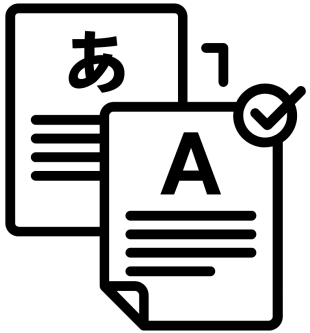


- Break the HTTP-centric grip on your API design process
- Embrace interface descriptions (ALPS)
- Enable translations (OpenAPI, AsyncAPI, SDL, proto, etc.)
- **Future-proof your design process**
  - ***Even if you use only one API style today, prepare for supporting others in the future***



That's all there is!

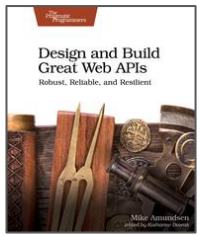
Resources



- "Design and Build Great Web APIs"
  g.mamund.com/greatwebapis

- This talk (slides, examples, etc.)
  g.mamund.com/unified-api-design

- More related content:
  g.mamund.com/youtube



GraphQL, gRPC and REST, Oh My!

A unified API design method

Mike Amundsen
@mamund
youtube.com/mamund