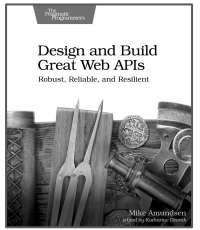
Design and Build Great Web APIs

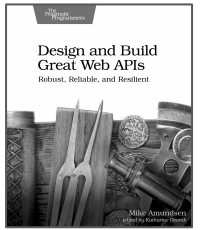
Principles

@mamund
Mike Amundsen
training.amundsen.com

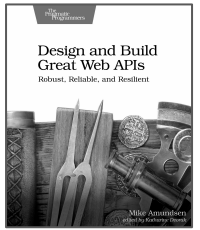


Principles

- API-First
  - You have to start somewhere...
- HTTP, the Web, and REST
  - Why do these matter?
- Modeling Interactions
  - Life is a loop.

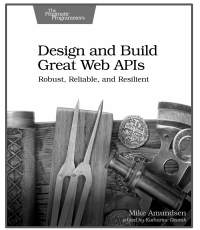


API-First


Design and Build
Great Web APIs
Robust, Reliable, and Resilient
Mike Amundsen

Principles : API-First

- Kas Thomas & Kin Lane
- Business Focus
- People Centric

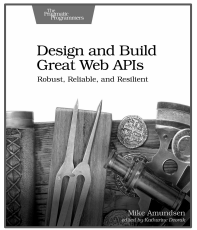


Principles : API-First

"API-first design means identifying and/or defining key actors and personas, determining what those actors and personas expect to be able to do with APIs."

-- Kas Thomas, 2009

http://asserttrue.blogspot.com/2009/04/api-first-design.html

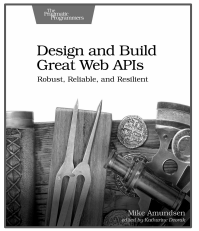


Principles : API-First

"Before you build your website, web, mobile or single page application you develop an API first."

-- Kin Lane

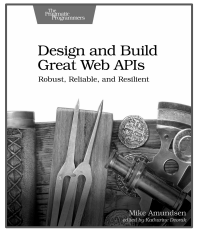
https://apievangelist.com/2014/08/11/what-is-an-api-first-strategy-adding-some-dimensions-to-this-new-question/



Principles : API-First : Business Focus

- Reduce the time/cost of getting something done
- Increase the ease of use or improve likelihood
- Solve a problem no one else has been able to solve

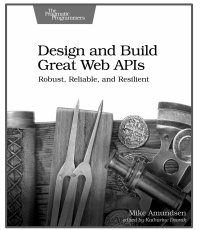
Good APIs have a purpose. What's yours?



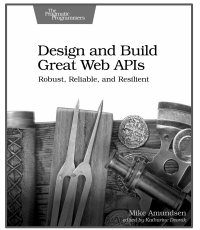
Principles : API-First : People-Centric

- Know your audience/customer
- Internal developer with a deadline?
- Partner with an SLA?
- Third-party you will never actually meet?

Solve your customer's problem, not your own.

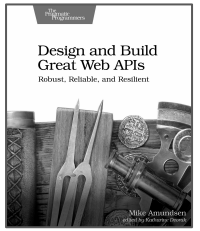


HTTP, the Web, and REST



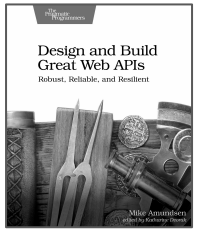
Principles : HTTP, the Web, and REST

- HTTP is a protocol
- The Web is a set of common practices
- REST is a specific style



Principles : HTTP

- Messages
- Methods
- Safety & Idempotence



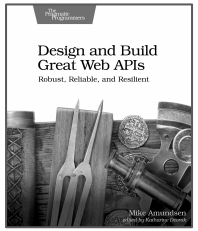
Principles : HTTP : Messages carry state

REQUEST

```
GET onboarding HTTP/1.1
User-Agent:
Host: apis.example.org
Accept: application/json
Accept-Language: en-us
Accept-Encoding: gzip, deflate
```

RESPONSE

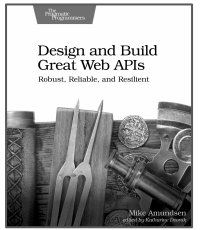
```
HTTP/2.0 200 OK
Date: Mon, 27 Jul 2019 12:28:53 GMT
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: application/json
Connection: Closed
```



Principles : HTTP : Methods express intent

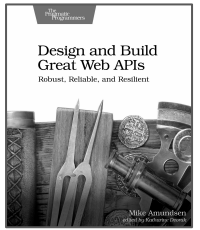
- HTTP clients use **Methods** to communicate intention
- GET, PUT, POST, DELETE ….
- Methods are NOT functions
- There are lots of registered methods (40+)

https://www.iana.org/assignments/http-methods/http-methods.xhtml



Principles : HTTP : Safety & Idempotence

- HTTP offers two powerful assurances:
- **Safety** (GET vs. DELETE)
- **Idempotency** (PUT vs. POST)

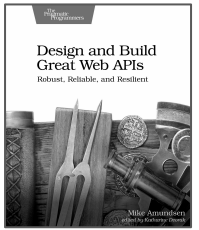


Principles : HTTP : Safety & Idempotence

- **Safety** (GET vs. DELETE)

```
GET /company/delete?id=21
```

What's wrong with this HTTP request?



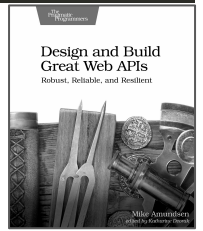
Principles : HTTP : Safety & Idempotence

- **Idempotence** (PUT vs. POST)

The bank-transfer dilemma

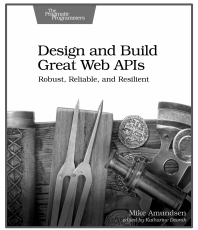
```
POST /bank-transfer HTTP/2.0
Host: apis.example.org
Accept: application/json
Content-Type: application/x-www-form-urnencoded

amount=1000&from-account=q1w2e3&to-account=zaxscd
```

What if I never get a response?



Principles : HTTP : To summarize

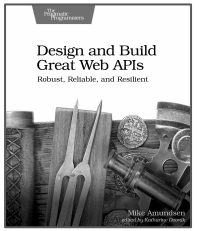
- Messages
  - To carry state
- Methods
  - To express intent
- Safety & Idempotence
  - To ensure success



Principles : The Web

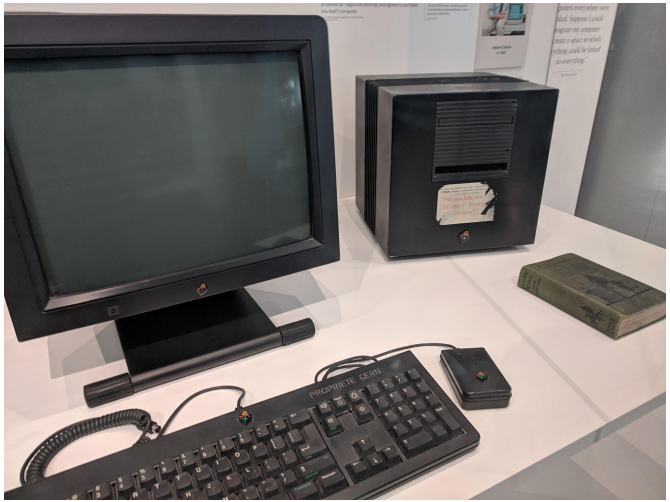
- Tim Berners-Lee built the first Web server & client in 1989
- He had to create HTTP & HTML to do it!
- CSS and Javascript was added by others later

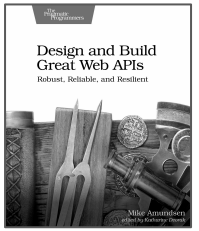
https://www.w3.org/History/1989/proposal-msw.html



Principles : The Web

- The Web is not a standard or specification
- It is a set of **common practices**
- "A linked information system"
- Basic principles:
  - Pass messages
  - Include links to follow/read (GET)
  - Include forms to send data (POST)

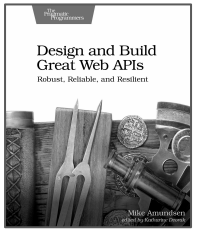




Principles : REST

- Roy Fielding created REST in 2000
- A list of properties and constraints
- REST not a standard or a common practice, it is a **style**.

https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm



Principles : REST

- List of System Properties
  - Performance
  - Scalability
  - Simplicity
  - Modifiability
  - Visibility
  - Portability
  - Reliability
- List of Implementation Constraints
  - Client-Server
  - Stateless
  - Caching
  - Uniform Interface
  - Layered System
  - Code on Demand

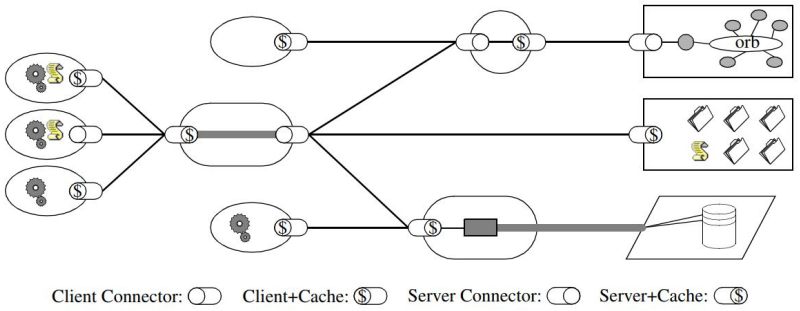
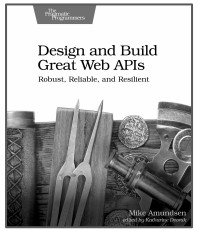
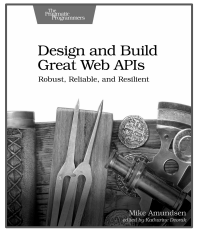


Figure 5-8. REST

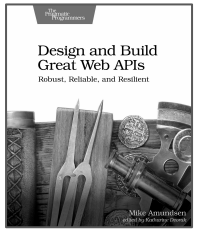


Modeling Interactions



Principles : Interactions

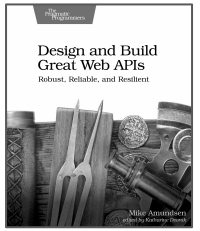
- Norman's Action Lifecycle
- Request - Parse - Wait (RPW Loop)
- Modeling Interactions



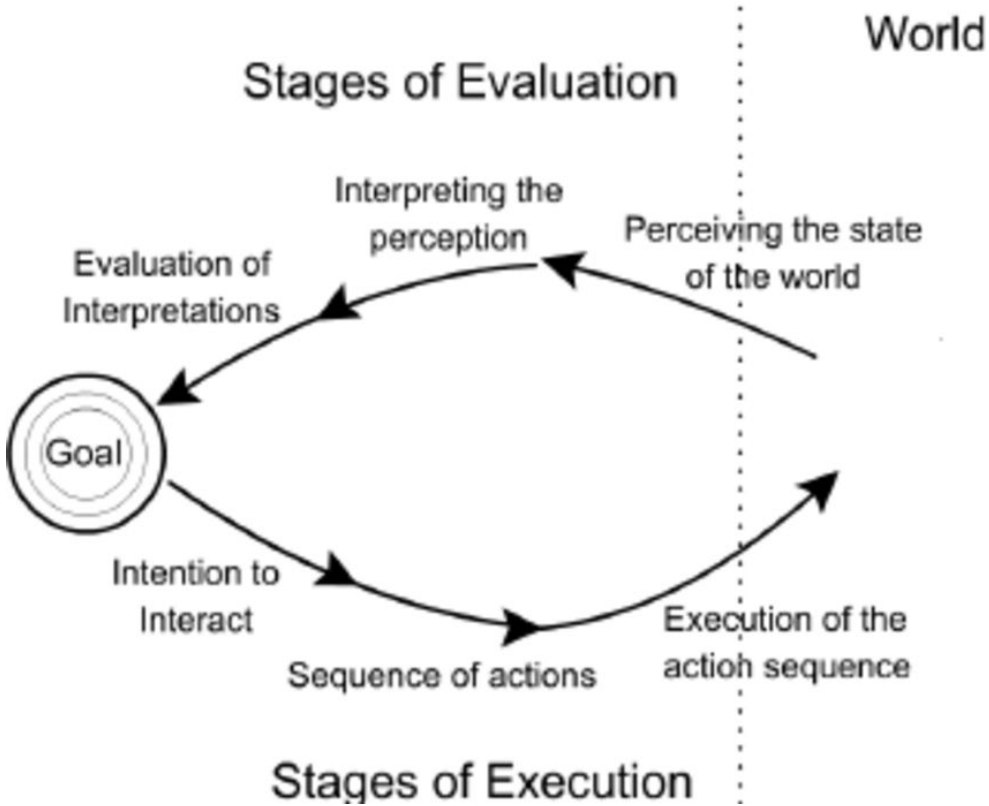
Principles : Interaction : Donald Norman

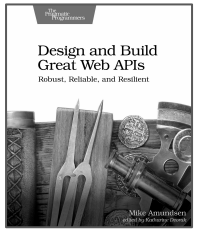
- "Design of Everyday Things"
- 'Norman Doors'
- Father of HCI
  (Human-Computer Interaction)

https://jnd.org/the-design-of-everyday-things-revised-and-expanded-edition/

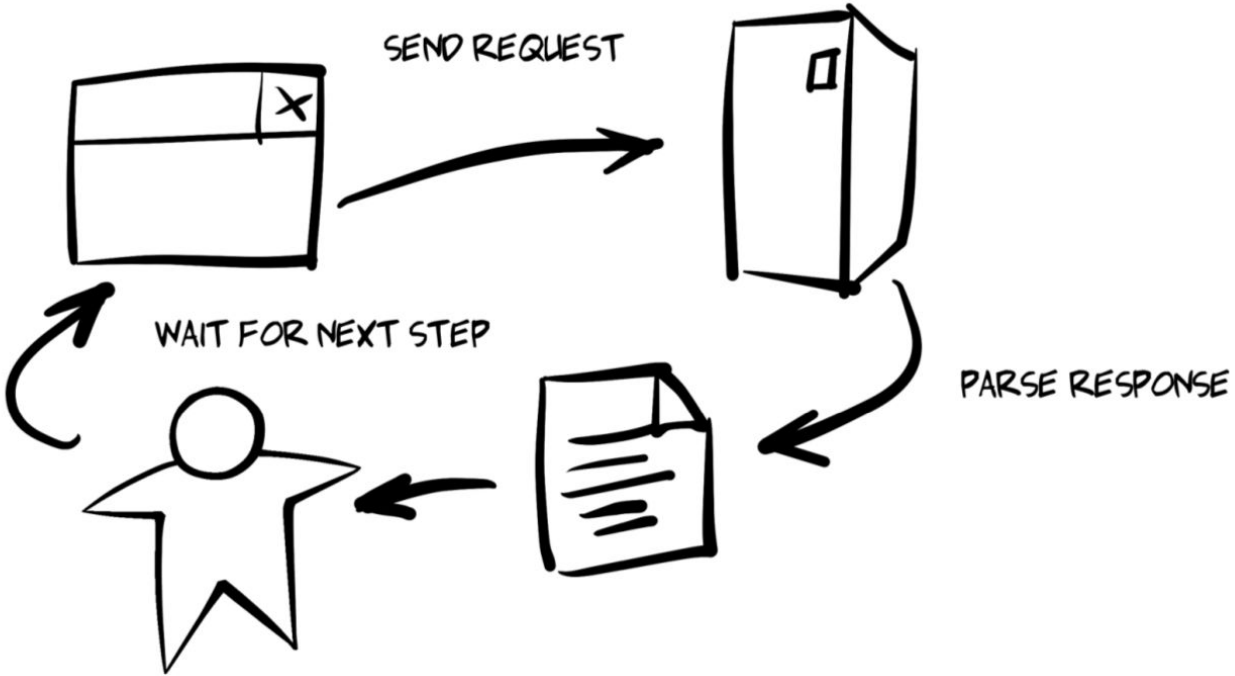


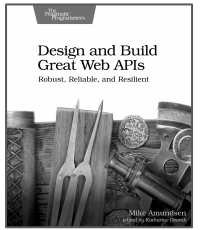
Principles : Interaction : Action Lifecycle





Principles : Interaction : RPW Loop





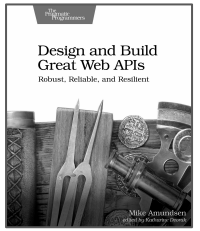
Principles : Interaction : Modeling

The onboarding process is a series of steps where we collect important data, e
generates key data records that we use throughout the life of our relationship v
is something that has been around a long time -- all done by hand -- and now v
process as possible.

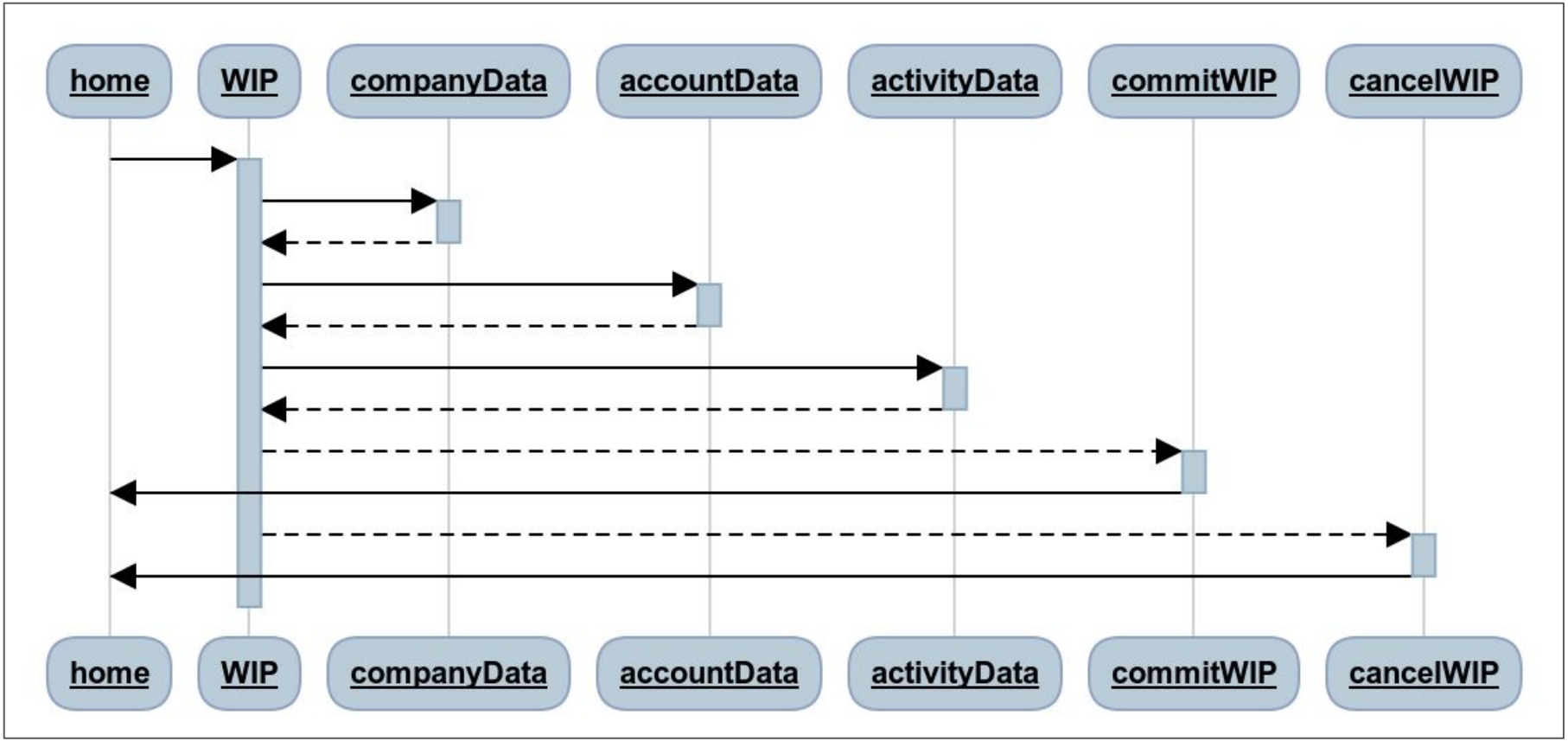
Each *onboarding experience* starts by creating an onboarding record. Currentl
important information and we pass this folder around to all parties involved in t
API, we expect this onboarding record to be something the new API will create

The onboarding experience currently has the following key steps:

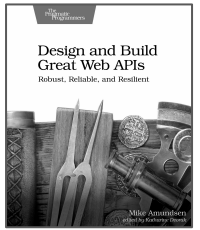
1. Create a new Onboarding Record
2. Collect and store Company information
3. Collect and store Account information
4. Collect and store Activity information
5. After review, either *accept* or *reject* the completed record.



Principles : Interaction : Modeling

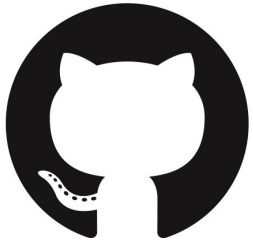


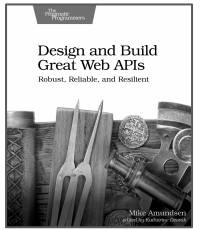
Principles Exercise



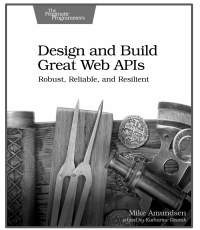
Principles Exercise

- NodeJS & npm
- Heroku account & CLI
- Git & github install (w/ ssh)
- Pull github.com/mamund/api-starter





Summary

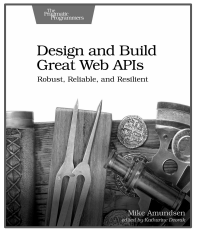

Design and Build
Great Web APIs
Robust, Reliable, and Resilient
Mike Amundsen

Principles : API-First

"API-first design means identifying and/or defining key actors and personas, determining what those actors and personas expect to be able to do with APIs."

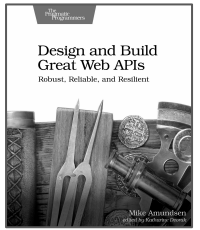
-- Kas Thomas, 2009

http://asserttrue.blogspot.com/2009/04/api-first-design.html



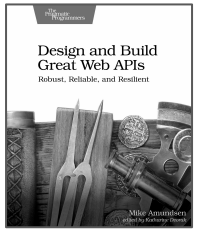
Principles : HTTP, the Web, and REST

- HTTP is a protocol
- The Web is a set of common practices
- REST is a specific style



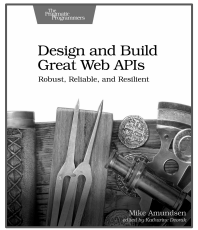
Principles : Interactions

- Norman's Action Lifecycle
- Request - Parse - Wait (RPW Loop)
- Modeling Interactions



Principles

- API-First
  - Solve your API consumer's problem, not yours
- HTTP, the Web, and REST
  - Rely upon standards, common practices, & style
- Modeling Interactions
  - Use loops to allow API consumers to control interactions



Design and Build Great Web APIs

Principles

@mamund
Mike Amundsen
training.amundsen.com