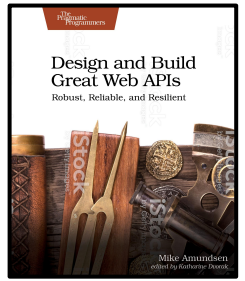
Building Great APIs
from the Ground Up

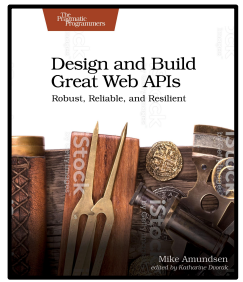
Afternoon

Mike Amundsen
@mamund



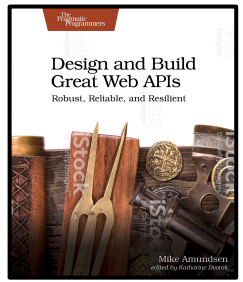
Morning Summary

- Session One
  - Web APIs (HTTP, WWW, REST)
  - Exploring APIs (curl)
  - Tracking your Project (git)
  - Managing your Project (npm)
- Session Two
  - Designing APIs (Design Process)
  - Diagramming (wsd)
  - Describing (alps)
  - Sharing your Project (github)

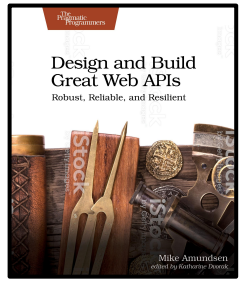


Afternoon Preview

- Session Three
  - Coding APIs
  - Sketching (blueprint)
  - Prototyping (swagger)
  - Building (expressjs)
- Session Four
  - Testing APIs (postman)
  - Securing APIs (auth0)
  - Deploying APIs (heroku)

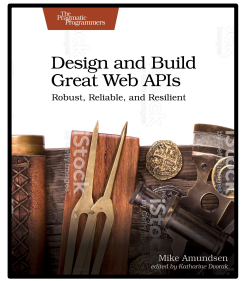


Afternoon


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

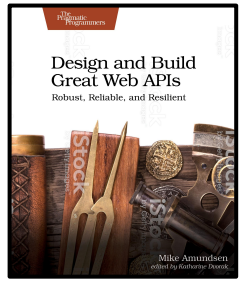
Afternoon -- Session Three

- Coding APIs
- Sketching (blueprint)
- Prototyping (swagger)
- Building (expressjs)



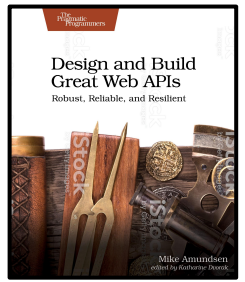
Coding APIs

- APIs are just Interfaces
- Translating the Design



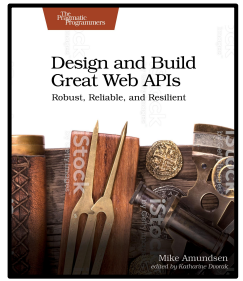
APIs are interfaces

- You're not designing the functionality of a service
- You MAY already have that functionality somewhere
- You MAY need to create the functionality
- Focus on the "API-First"



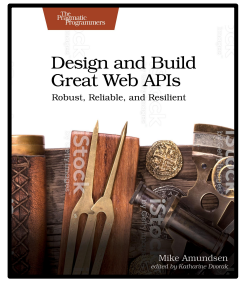
You may already have the functionality

- Your job is to act as a "proxy" between the interface design and the existing functionality
- Identify the existing functionality (the service(s)
- "Do the Work"
  - Convert interface inputs into service inputs
  - Convert service outputs into interface outputs



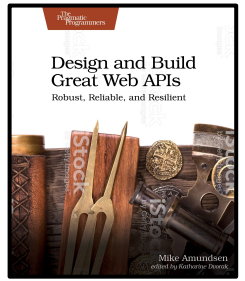
You may need to create the functionality

- Your job is to act as a "guide" for the new functionality
- Offer the interface as a "shell" for future functionality
- Be prepared to do conversions
  - Convert the inputs to match the new functionality
  - Convert the new outputs to match the interface



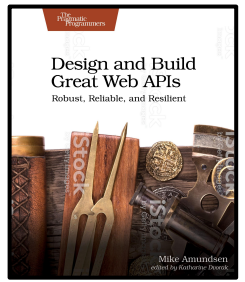
Focus on an "API First" Approach

- Stick to the "API-First" view
- Put on your "API" hat when reviewing implementations
- Assume the API will not change, but the implementation details will
- Once released to production, it is easier to modify functionality than interfaces



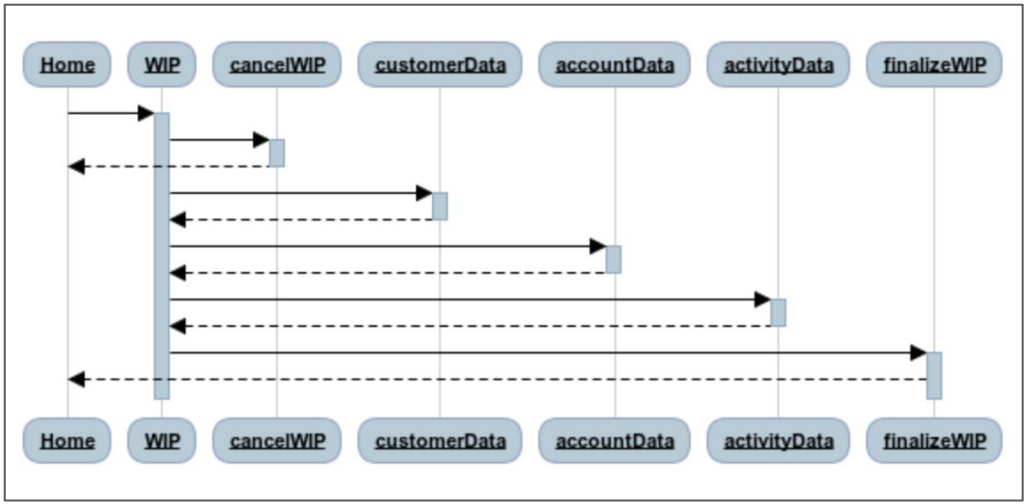
Translating the Design

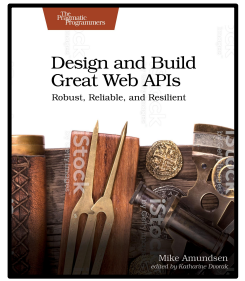
- Assets
  - User Story
  - Sequence Diagram
  - ALPS Profile
- Implementation
  - HTTP
  - Resources
  - Messages



Translating the Design -- Assets

- User Story
- Sequence Diagram
- ALPS Profile

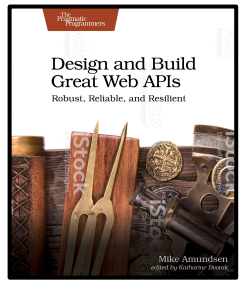




Translating the Design -- Implementation

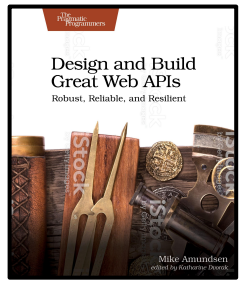
- HTTP
- Resources
- Messages

```yaml
swagger: '2.0'
info:
  title: Onboarding API
  version: ''
  description: Polls is a simple API allowing consumers to view polls and
host: polls.apiblueprint.org
basePath: /
schemes:
  - http
paths:
  /questions:
    get:
      responses:
        '200':
          description: OK
          headers: {}
          examples:
            application/json:
```

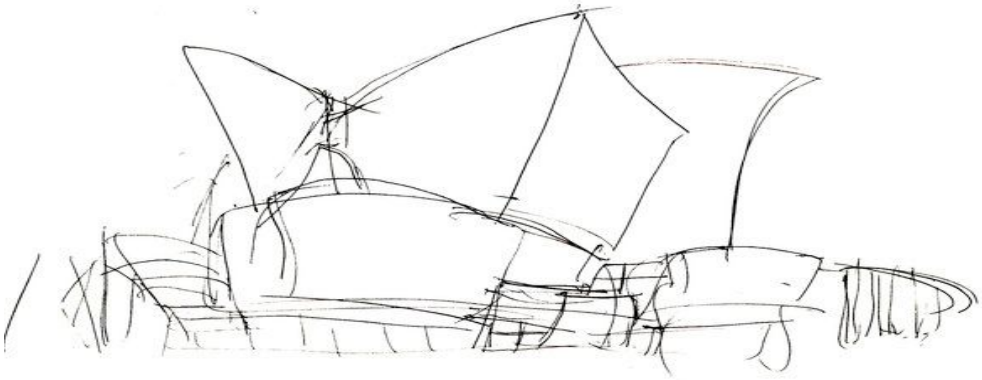


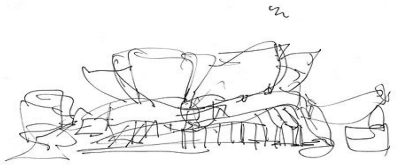
Sketching APIs

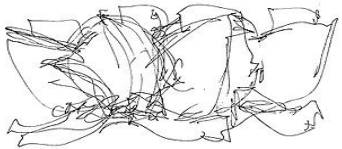
- Sketching
- Using **Blueprint**

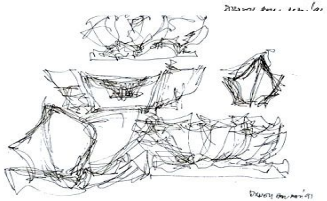


Frank Gehry Sketches







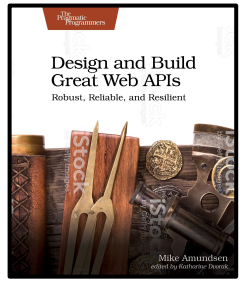


Frank Gehry on Design

An architect is given a program, budget, place, and schedule. Sometimes the end product rises to art

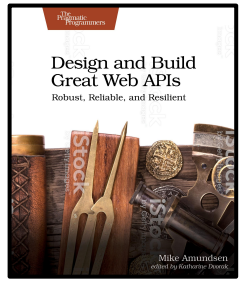
Sketching APIs -- Sketching

- Sketches are terse, rough drawings
- They give the general idea of a thing but lack important details.
- Usually, one can glean the basics from a sketch but
- Sketches usually are just explorations of ideas, not fully-formed items.



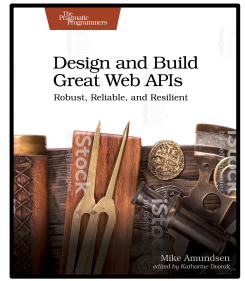
Sketching APIs -- Sketching

- Create a sketch (using **Blueprint**).
- Show it to others (devs, stakeholders) and get their feedback.
- If possible use simple API consumer tools (curl, NodeJS, etc.) to test.
- Continue to modify the simple sketches as needed



Sketching APIs -- Using Blueprint

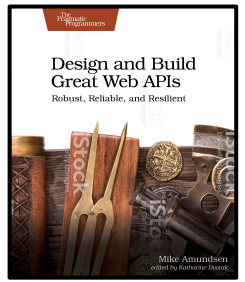
- Created in 2013 by Jakub Nesetril
- Focused on quickly mocking API request/response
- Based on Markdown
- Sold to Oracle in 2017



Sketching APIs -- Using Blueprint

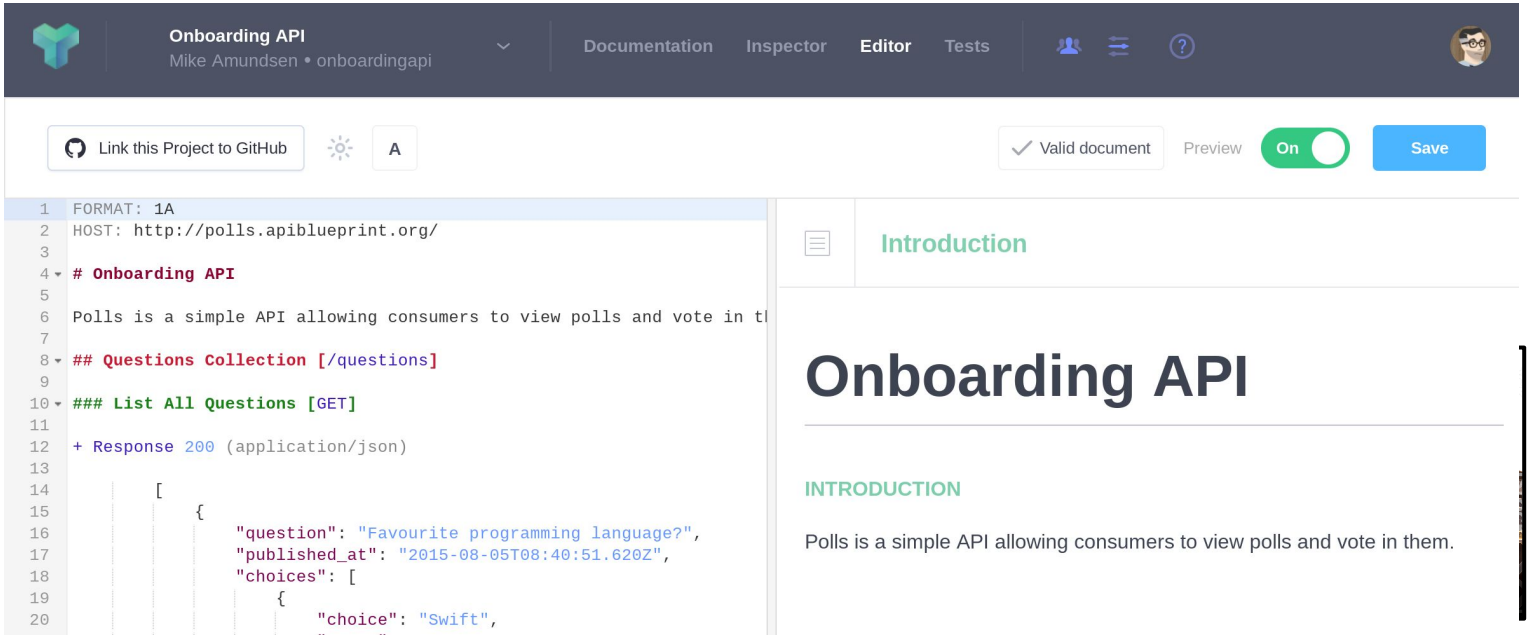
- No download needed
  https://app.apiary.io/onboardingapi/editor
- Documentation
  https://help.apiary.io/tools/apiary-editor/
- Create Account (optional)
  https://login.apiary.io/

apiary

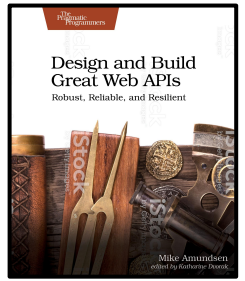


Sketching APIs -- Using Blueprint

- Write (or copy/paste) APIB doc into Editor
- Copy/Paste into local doc to save to disk

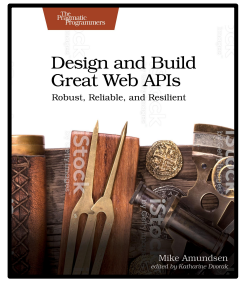


Sketches are made to be thrown away.



Prototype APIs

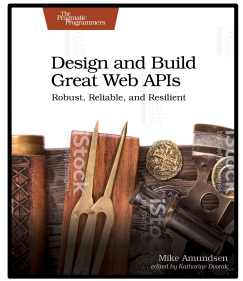
- Prototyping
- Using **Swagger**



Prototyping in Sculpture

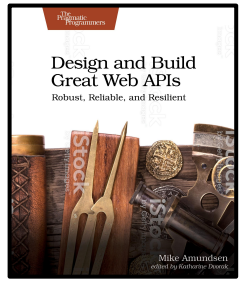
Prototype APIs -- Prototyping

- Prototypes look like the real thing, but are not. They’re "fakes."
- They let you work up something with all the details of a real API, but without the actual functionality behind it.
- They’re an inexpensive way to work out the details
- Use them to discover challenges before you go into production.



Prototype APIs -- Prototyping

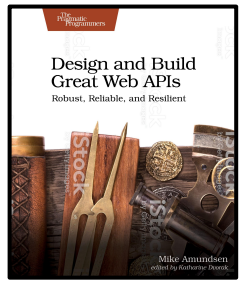
- Select a likely sketch
- Create a prototype of it (using **Swagger**).
- Show it to others (devs, stakeholders) and get their feedback.
- If possible, use production-level API consumer tools to test.
- Continue to modify the simple sketches as needed



Prototype APIs -- Using Swagger

- Swagger (AKA OpenAPI)
- Created in 2011 by Tony Tam (Wordnik)
- Focused on auto-generated Docs & SDKs
- Open API Initiative created in 2015 (Linux Foundation)

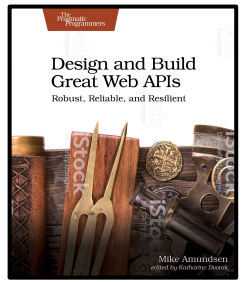
OPENAPI
INITIATIVE



Prototype APIs -- Using Swagger

- No download needed
  https://editor.swagger.io/
- Documentation
  https://swagger.io/docs/
- Create an account (optional)
  https://app.swaggerhub.com

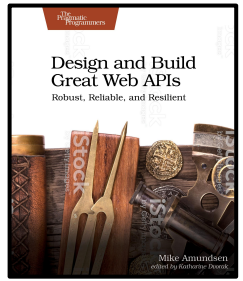
OPENAPI INITIATIVE



Prototype APIs -- Using Swagger

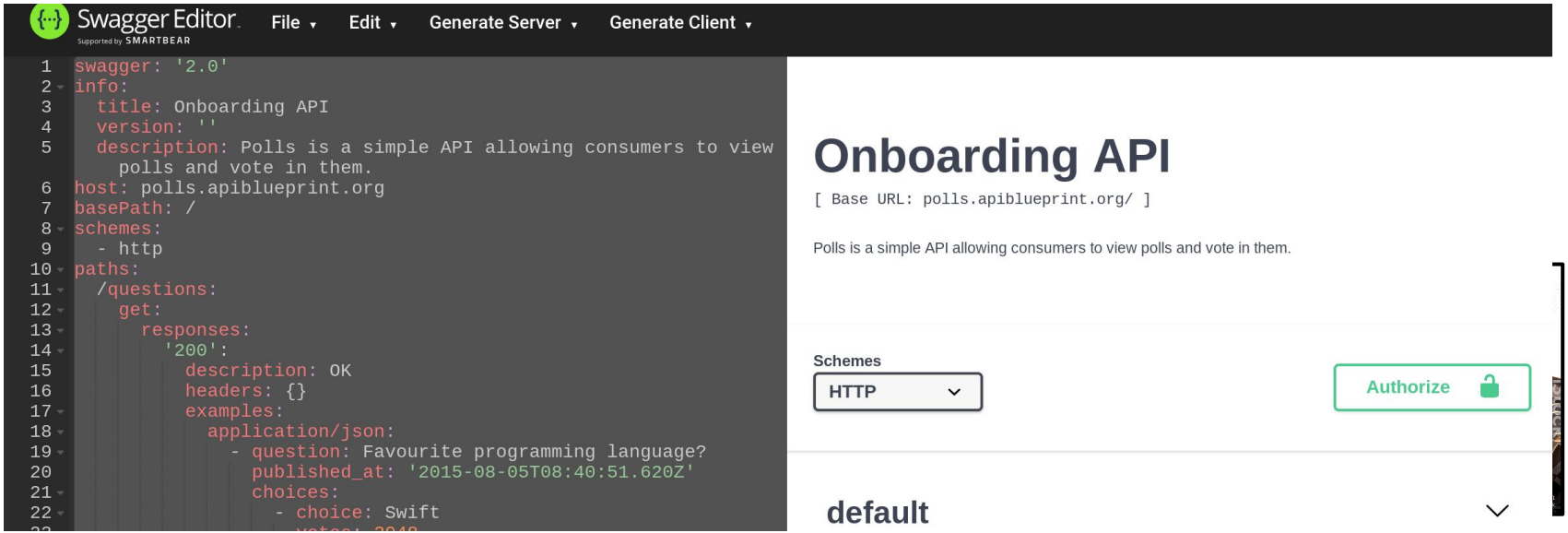
- Convert APIB to Swagger
  https://github.com/kminami/apib2swagger

```
apib2swagger -i onboardingAPI.apib -o onboarding.json

apib2swagger -i onboardingAPI.apib --yaml -o onboarding.json
```



Prototype APIs -- Using Swagger

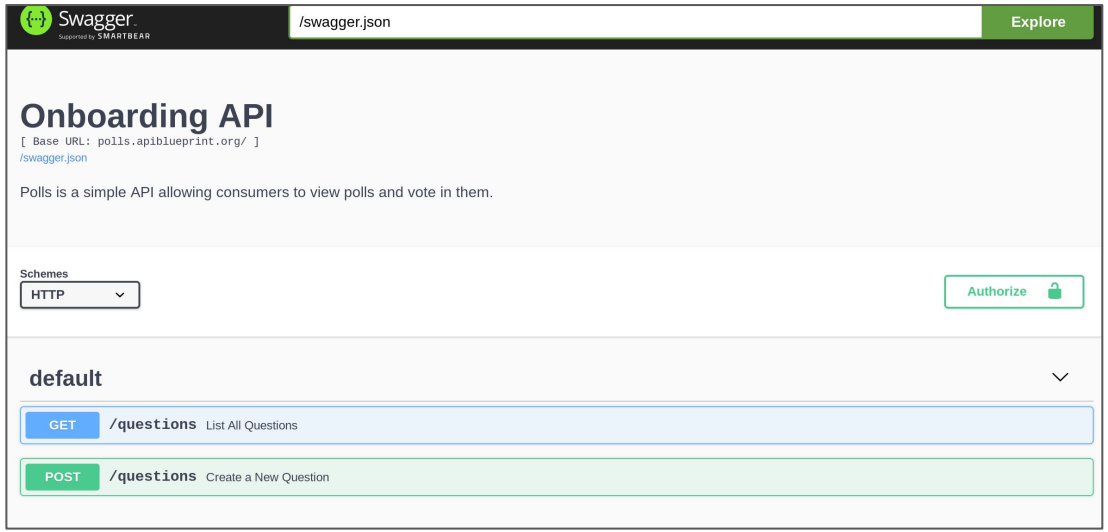
- Copy/Paste into Swagger Editor



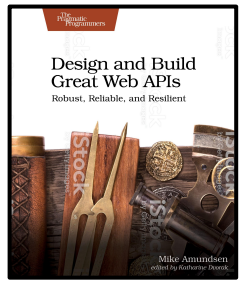
Prototype APIs -- Using Swagger

- Generate SwaggerUI documentation

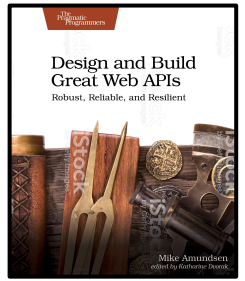
```
apib2swagger -i onboarding.apib -s -p 3000
```



OPENAPI INITIATIVE

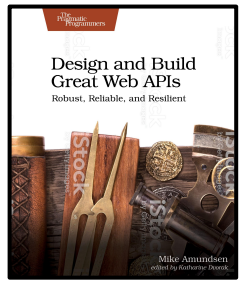


Prototypes are made to be tested.



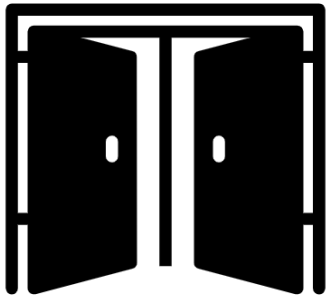
Building APIs

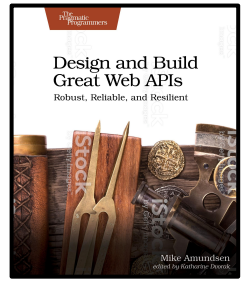
- The DORR Model
- Using NodeJS & ExpressJS



Building APIs -- DORR Model

- Data Model
- Object Model
- Resource Model
- Representation Model

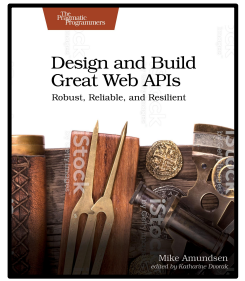




Building APIs -- DORR Model (Data)

simple-storage.js

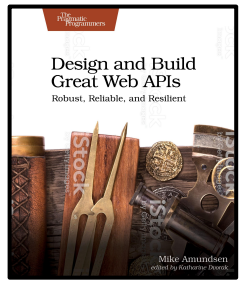
```
/*
 * args is a hash table of possible arguments
 * {object:"",action:"",filter:"",id:"",item:objItem}
 */
function main(args) {
  var rtn;

  // resolve arguments
  var action = args.action||"";
  var object = args.object||null;
  var filter = args.filter||null;
  var id = args.id||null;
  var item = args.item||{};

  switch (action) {
    case 'create':
      rtn = createObject(object);
      break;
    case 'list':
      rtn = getList(object);
      break;
    case 'filter':
      rtn = getList(object, filter);
      break;
    case 'item':
      rtn = getItem(object, id);
```



Building APIs -- DORR Model (Objects)

simple-component.js

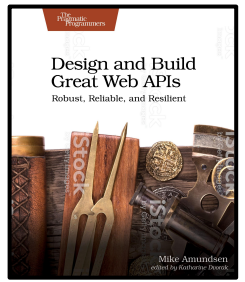
```
// app-level actions for tasks
// args: name, props, reqd, action, id, filter, item
function main(args) {
  var name, rtn, props;
  var conn, action, id, filter, item;

  elm = args.name||"";
  props = args.props||[];
  reqd = args.reqd||[];
  action = args.action||"list";
  id = args.id||"";
  filter = args.filter||"";
  item = args.item||{};

  // confirm existence of object storage
  storage({action:'create',object:elm});

  // handle action request
  switch (action) {
    case 'exists':
      rtn = (storage({object:elm, action:'item', id:id})=
      break;
    case 'props' :
      rtn = utils.setProps(item,props);
      break;
    case 'profile':
```



Building APIs -- DORR Model (Resources)

company.js

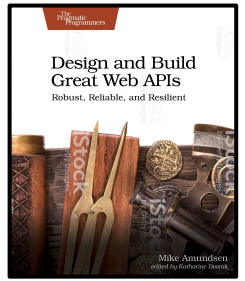
```
/*************************************
 * handle request events
 *************************************/
// home
router.get('/', function (req, res) {
  res.send('{"home" : "{"name":"customer", "rel" : "c
})

// create
router.post('/', function(req,res) {
  processPost(req,res).then(function(body) {
    res.send('{"customer" : ' + JSON.stringify(body,n
  }).catch(function(err) {
    res.send('{"error" : ' + JSON.stringify(err,null,
  });
});

// list
router.get('/list/', function(req, res) {
  processList(req,res).then(function(body) {
    res.send('{"customer":' + JSON.stringify(body,nul
  }).catch(function(err) {
    res.send('{"error" : ' + JSON.stringify(err,null,
  });
});
```



Building APIs -- DORR Model (Representations)

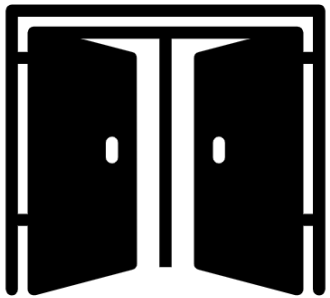
ejs templates

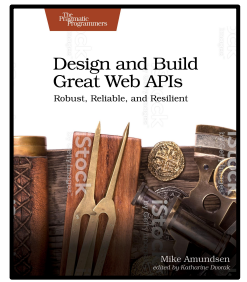
```
var ejs = require('ejs');
var jsonView = '<%= body %>';

// set up request body parsing
router.use(bodyParser.json({type:[
    "application/json",
    "application/vnd.hal+json",
    "application/vnd.siren+json",
    "application/vnd.collection+json"
    ]}));
router.use(bodyParser.urlencoded({extended:true}));
```



Building APIs -- DORR Model

- DORR Advantages
  - Maintains separation of concerns
  - Allows for easier work as a team (data owner, object owner, etc.)
  - Retains options for later modularization





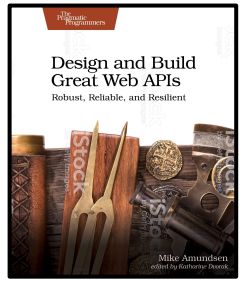
Building APIs -- Using NodeJS & ExpressJS

- Start w/ the sample onboarding-porto repo
  https://github.com/mamund/onboarding-porto
- Review Company Form
- Add Account & Activity Forms

```
function companyLinks(list) {
  var id="";
  list.links = [];
  if(list.length>0) {
    id=list[0].id;
  }
  list.links[0] = {rel:"home",href:"/onboarding/"};
  list.links[1] = {rel:"update",href:"/onboarding/wip",
    form: {
      method:"put",
      contentType:"application/x-www-form-urlencoded",
      properties: [
        {name:"onboardingId",value:id},
        {name:"companyName",value:""},
        {name:"email",value:""},
        {name:"status",value:"pending"},
      ]
    }
  };
return list;
```

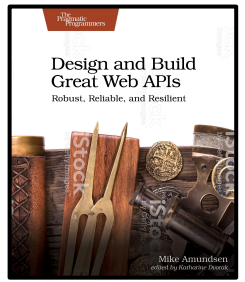
Coding APIs

- APIs are just Interfaces
- Translating the Design
- Sketch
- Prototype
- Build



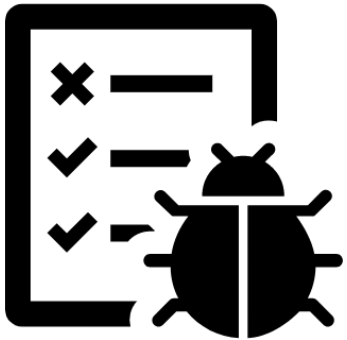
Afternoon -- Session Four

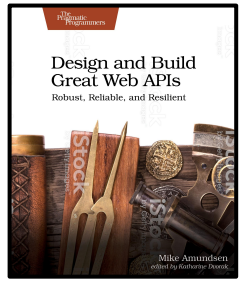
- Testing APIs (postman)
- Securing APIs (auth0)
- Deploying APIs (heroku)



Testing APIs

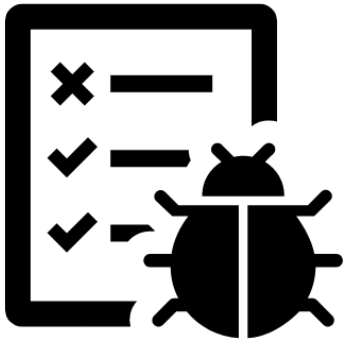
- Testing the Network
- Using **Postman/Newman**

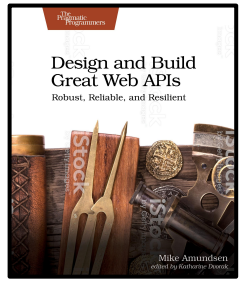




Testing APIs - Testing the Network

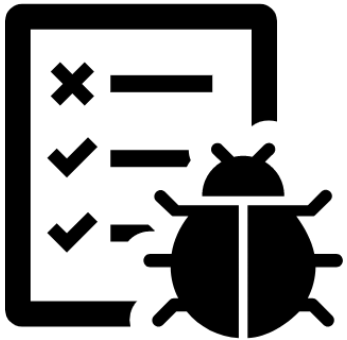
- Most of the threat for APIs is not in the code
- It's in the network itself
- And other people's APIs
- You can code and test for these cases

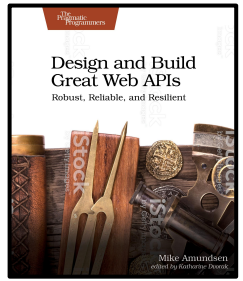




Testing APIs - Testing the Network

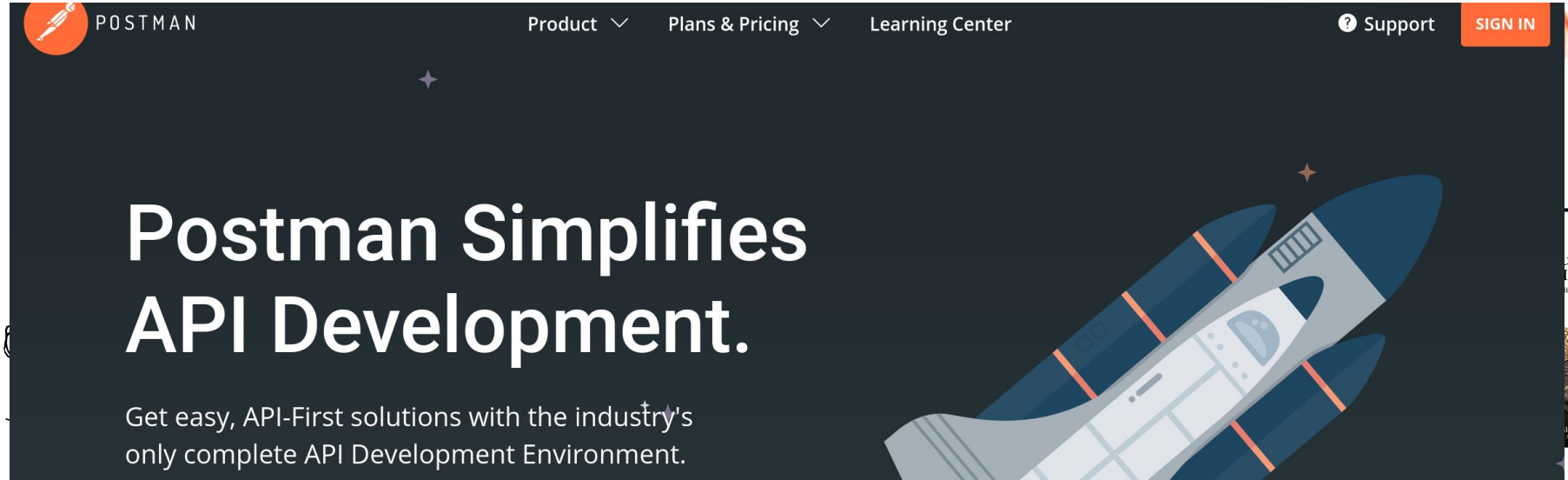
- Michael Nygard's book "Release It!" (2007, 2018)
- Stability Patterns
  - Timeout
  - Failfast
  - Bulkhead
  - Circuit-breaker
  - Handshaking
  - Steady-state





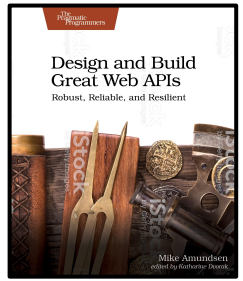
Testing APIs - Using Postman

- Postman (2014)
- Working to be a complete API platform
- Focused on testing



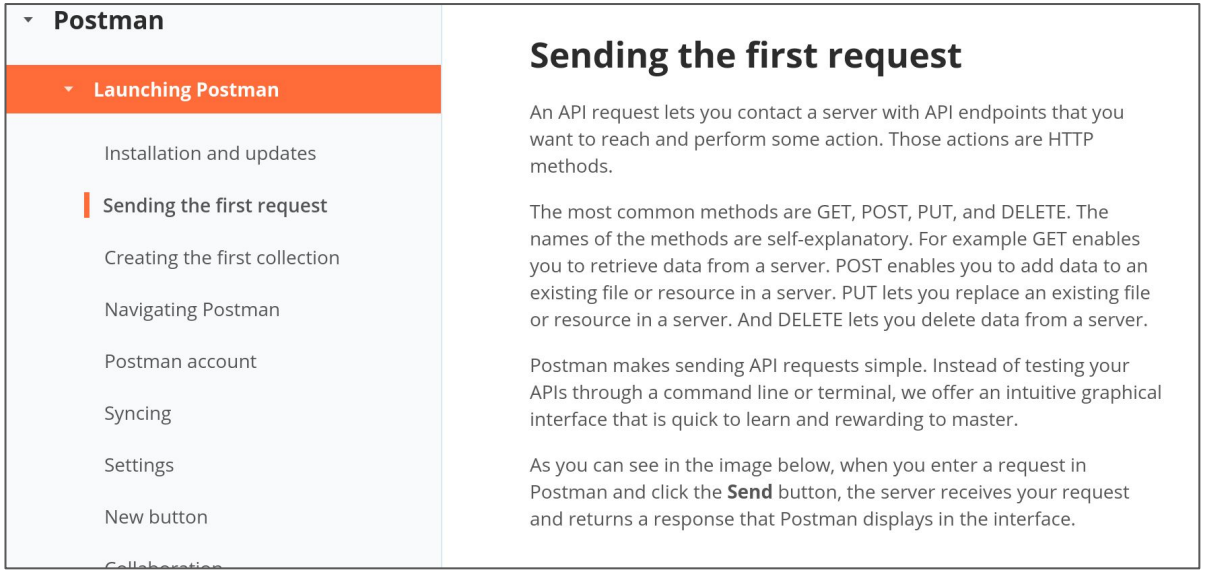
Testing APIs - Using Postman

- Download & Install
  https://www.getpostman.com/downloads/
- Documentation
  https://learning.getpostman.com/docs/postman/launching_postman/installation_and_updates/
- Create an Account (recommended)
  https://identity.getpostman.com/signup

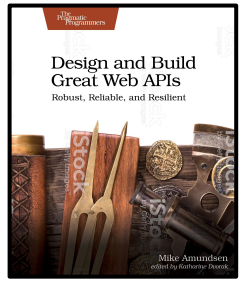


Testing APIs - Using Postman

- Install, launch, and start testing



POSTMAN



Testing APIs - Bonus Utility -- newman

- CLI for running postman tests

Getting Started

Newman is built on Node.js. To run Newman, make sure you have Node.js installed.

You can download and install Node.js on Linux, Windows, and Mac OSX.

After you install Node.js, Newman is just a command away. Install Newman from npm globally on your system, which allows you to run it from anywhere.

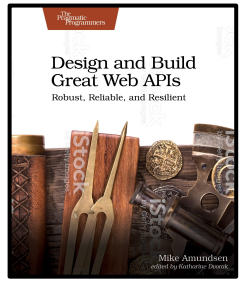
```
$ npm install -g newman
```

The easiest way to run Newman is to run it with a collection. You can run any collection file from your file system.

To learn how to export collections to share as a file, see the collection documentation.

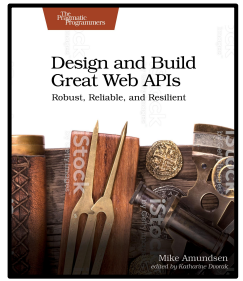
```
$ newman run mycollection.json
```

POSTMAN



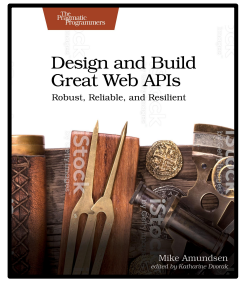
Securing APIs

- API Security Basics
- Using **Auth0**



Securing APIs - Security Basics

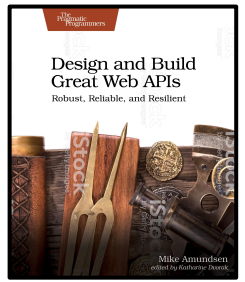
- Authentication (Identity)
- Authorization (Access Control)
- TLS/HTTPS (message encoding)
- Encryption (field-level encoding)



Securing APIs - Security Basics

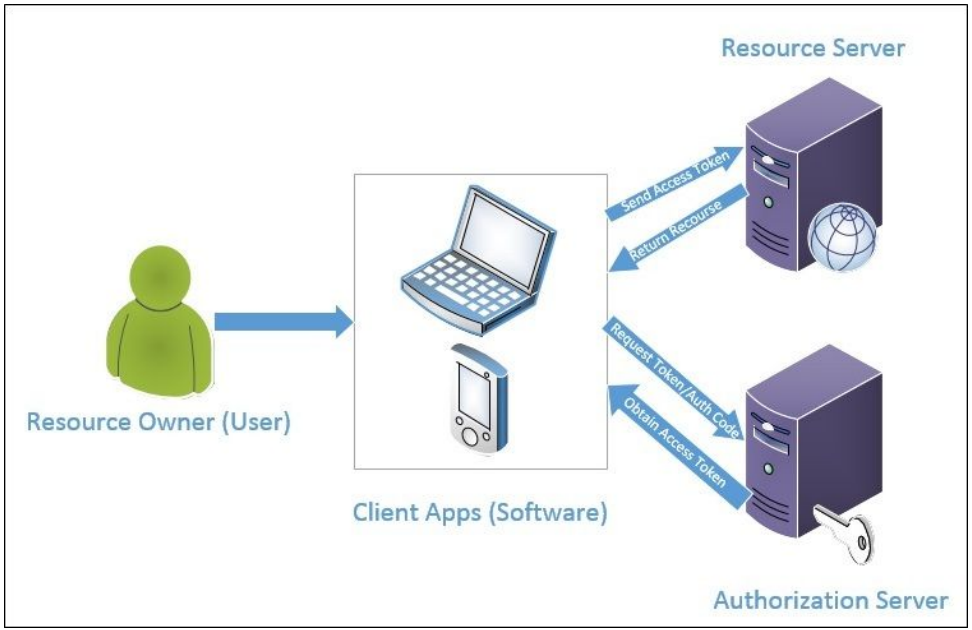
- Securing a Web API is tricky
- Who holds the list of users?
- Who holds the list of access rules?
- Who wrote the server side?
- Who wrote the client side?
- Who can see credentials?

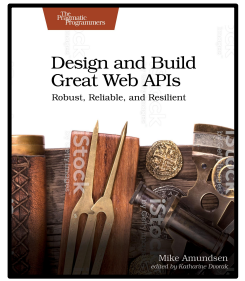
Solution: is "Three-legged Authentication"



Securing APIs - Security Basics

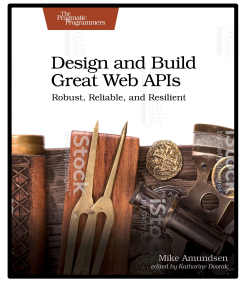
- Three-legged Authentication





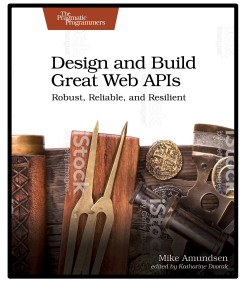
Securing APIs - OAuth

- OAuth was created to solve this problem
- Originally designed for Twitter in 2006
- Moved to IETF Standards in 2008
- OAuth 1.0 released 2010 (RFC5849)
- OAuth 2.0 released in 2012 (RFC6749 & RFC6750)



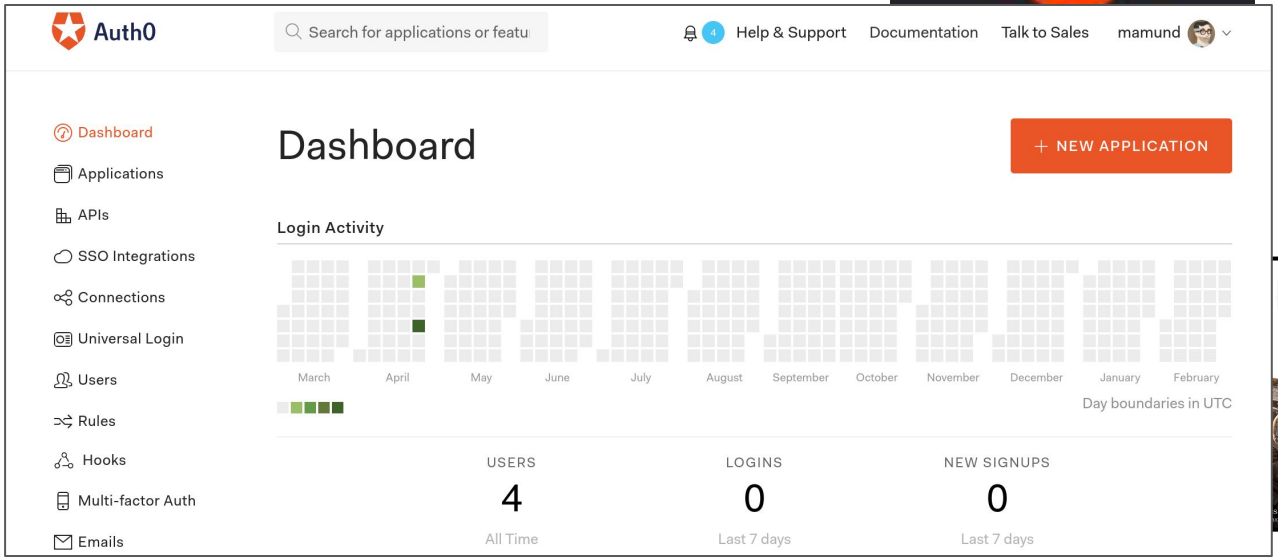
Securing APIs - Auth0

- Public OAuth Cloud Service
- Created in 2013
- Supports Mobile, Web, API scenarios
- Lots of SDKs for many platforms



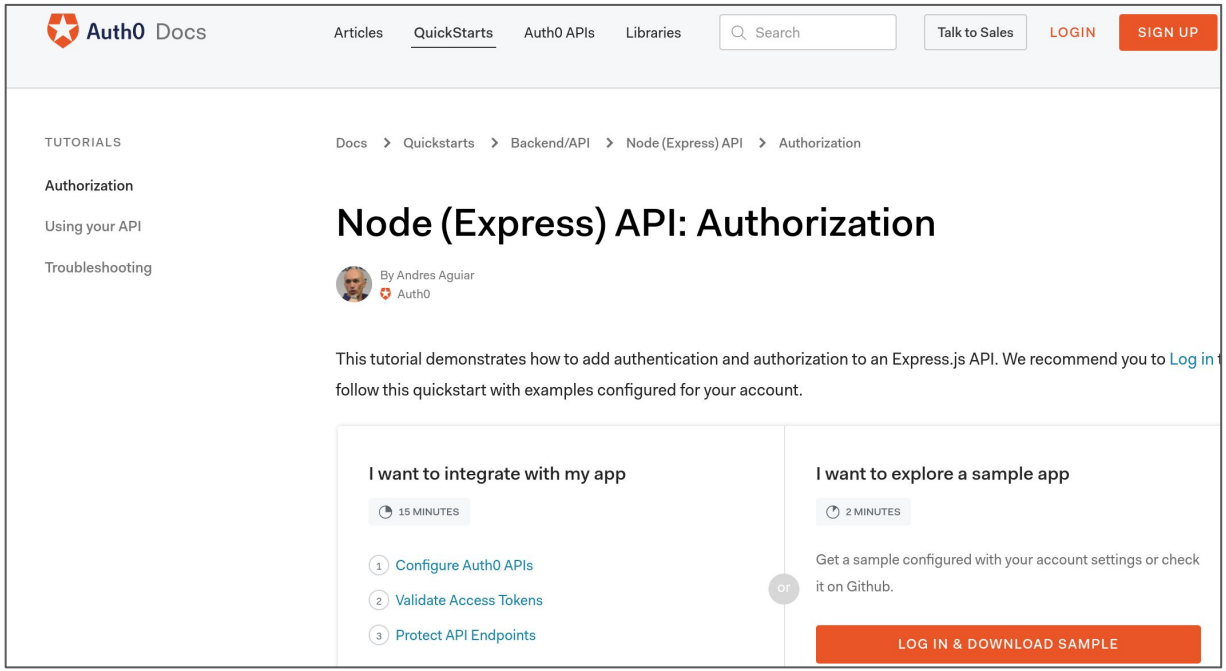
Securing APIs - Auth0

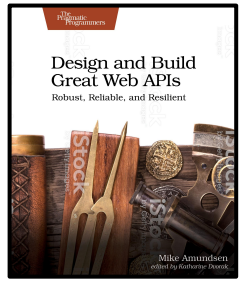
- No Download
  https://auth0.com/
- Create Account (required)
  https://auth0.com/signup
- Dashboard
  https://manage.auth0.com/#/



Securing APIs - Auth0

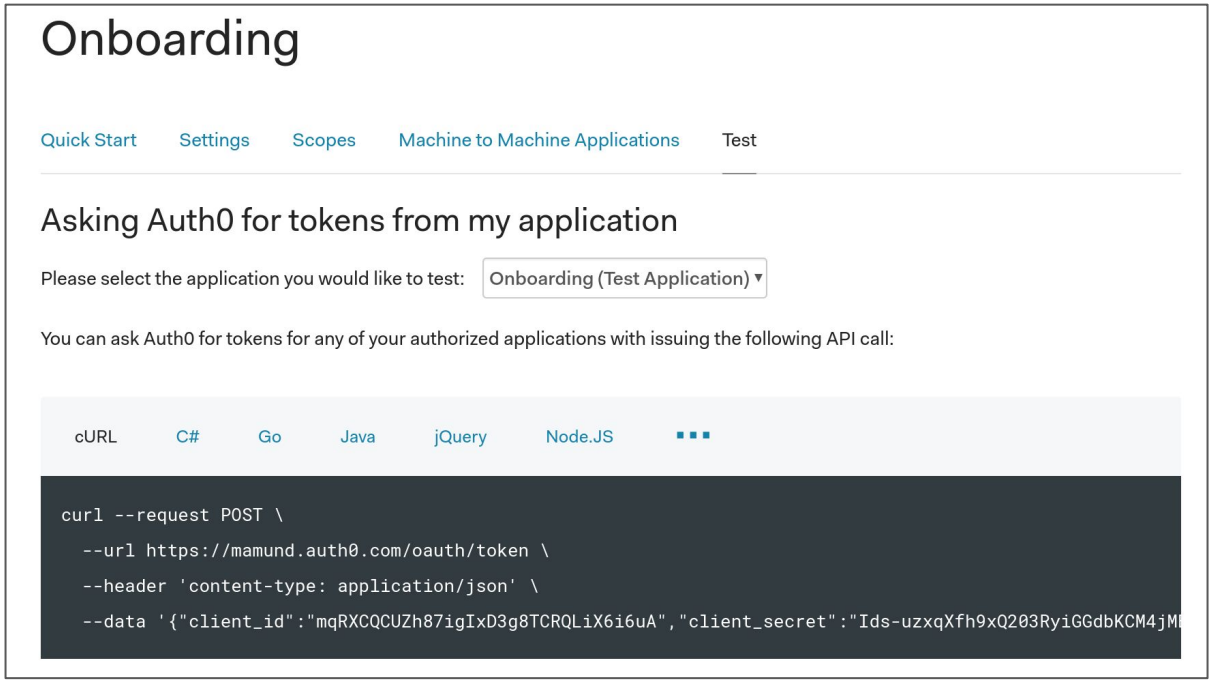
- Set up API Authentication for ExpressJS
  https://auth0.com/docs/quickstart/backend/nodejs

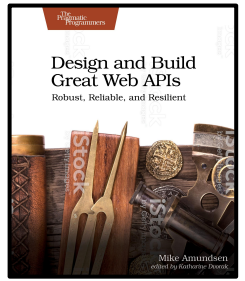




Securing APIs - Auth0

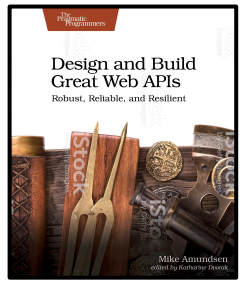
- Execute machine-to-machine (API) call via test page





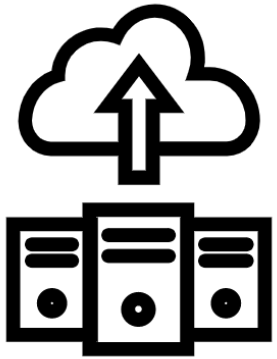
Deploying APIs

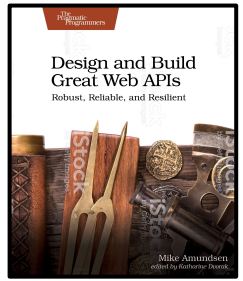
- Git-based Deployment
- Using **Heroku**



Deploying APIs - Challenges

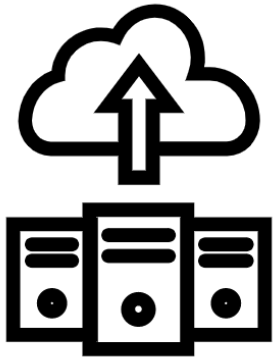
- Deploying your app can be complicated
- Compatibility
  - Hardware
  - OS
  - Platform
  - Framework
  - Dependencies

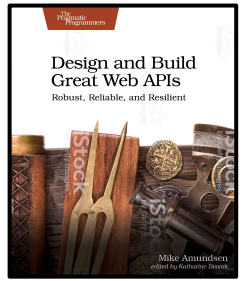




Deploying APIs - DevOps

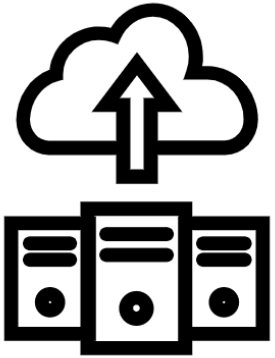
- DevOps was created to help with all this
- Developers & Operators working together
- Started as a hashtag on twitter #DevOps
- Series of small conferences started in 2009
- Emphasis on automation to improve reliability

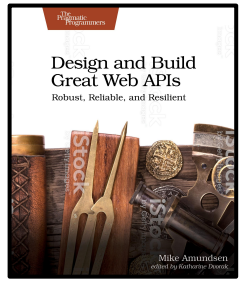




Deploying APIs - Tools

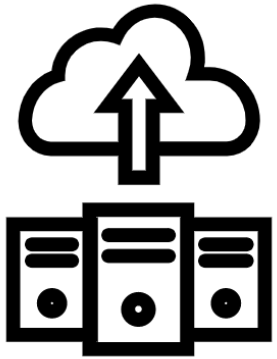
- Build tools
- CI/CD pipeline
- Docker (containers)
- Kubernetes (deployment orchestration)

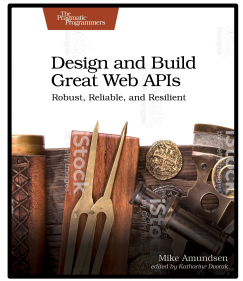




Deploying APIs - Using Heroku

- Cloud platform (2007)
- Originally just for Ruby/Rails projects
- Now supports Java, NodeJS, Python, Go, Clojure, Scala
- Acquired by Salesforce in 2011
- Full platform w/ marketplace ecosystem
- Heroku uses proprietary container tech (Dynos)

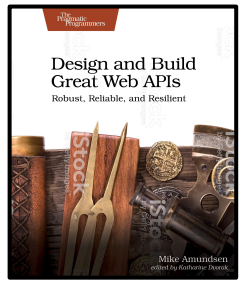




Deploying APIs - Using Heroku

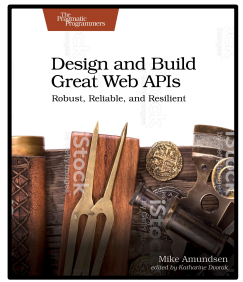
- Cloud platform (2007)
- Originally just for Ruby/Rails projects
- Now supports Java, NodeJS, Python, Go, Clojure, Scala
- Acquired by Salesforce in 2011
- Full platform w/ marketplace ecosystem
- Heroku uses proprietary container tech (Dynos)

HEROKU



Deploying APIs - Using Heroku

- Download CLI
  https://devcenter.heroku.com/articles/heroku-cli
- Documentation
  https://devcenter.heroku.com/articles/using-the-cli
- Create an Account (required)
  https://signup.heroku.com/



Deploying APIs - Using Heroku

- Git deploy tutorial
  https://devcenter.heroku.com/articles/git

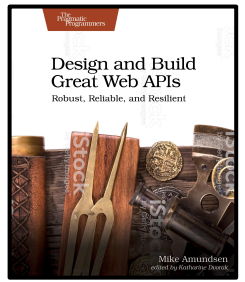
The `heroku create` CLI command creates a new empty application on Heroku, along with an associated empty Git repository. If you run this command from your app's root directory, the empty Heroku Git repository is automatically set as a remote for your local repository.

```
$ heroku create
Creating app... done, ⬢ thawing-inlet-61413
https://thawing-inlet-61413.herokuapp.com/ | https://git.heroku.com/thawing-inlet-61
```

You can use the `git remote` command to confirm that a remote named `heroku` has been set for your app:

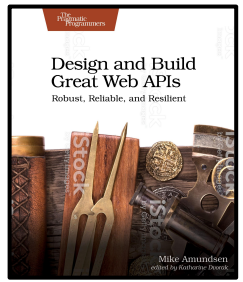
```
$ git remote -v
heroku  https://git.heroku.com/thawing-inlet-61413.git (fetch)
heroku  https://git.heroku.com/thawing-inlet-61413.git (push)
```

HEROKU



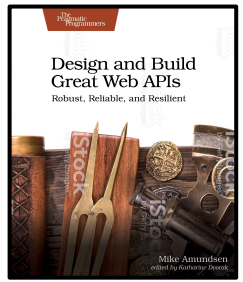
Afternoon Summary

- Session Three
  - Coding APIs
  - Sketching (blueprint)
  - Prototyping (swagger)
  - Building (expressjs)
- Session Four
  - Testing APIs (postman)
  - Securing APIs (auth0)
  - Deploying APIs (heroku)



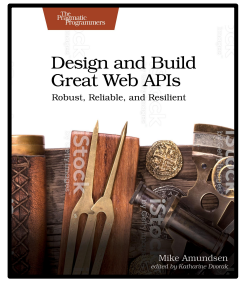
Morning - Summary

- Session One
  - Web APIs (HTTP, WWW, REST)
  - Exploring APIs (curl)
  - Tracking your Project (git)
  - Managing your Project (npm)
- Session Two
  - Designing APIs (Design Process)
  - Diagramming (wsd)
  - Describing (alps)
  - Sharing your Project (github)

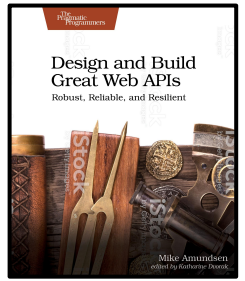


Final Notes

- I'll post all slides in the repo
  https://github.com/mamund/2019-04-goto-chicago-great-apis
- I'll include all example repos, too
- Newsletter
  http://g.mamund.com/newsletter
- Please keep in touch (@mamund)



Thanks!


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

Building Great APIs
from the Ground Up

Mike Amundsen
@mamund