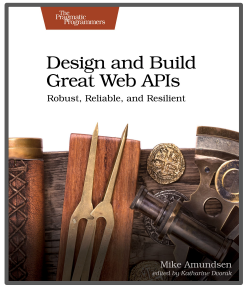
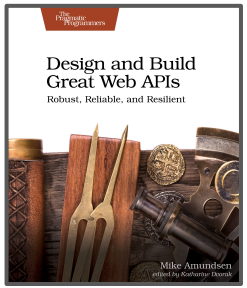
Building Great APIs
from the Ground Up

GOTO Amsterdam

Mike Amundsen
@mamund

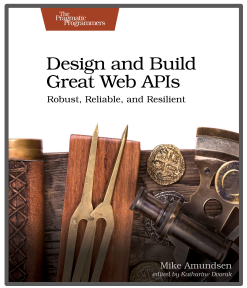


Afternoon


The Pragmatic Programmers
Design and Build Great Web APIs
Robust, Reliable, and Resilient
Mike Amundsen
edited by Katharine Dvorak

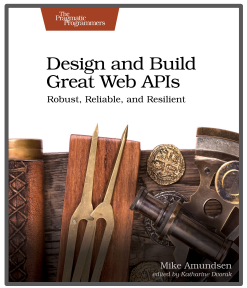
Afternoon -- Developing

- Interfaces
- Translating
- Developing APIs



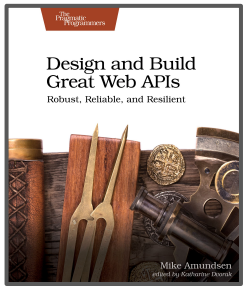
APIs are interfaces

- 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"



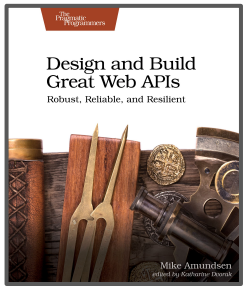
You may already have the functionality

- 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



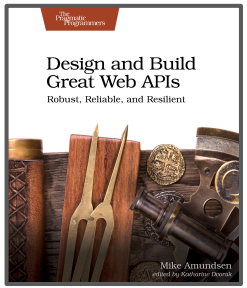
You may need to create the functionality

- 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



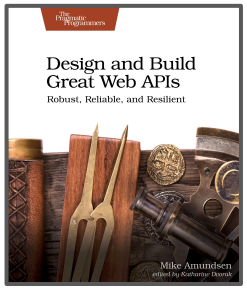
Focus on an "API First" Approach

- 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



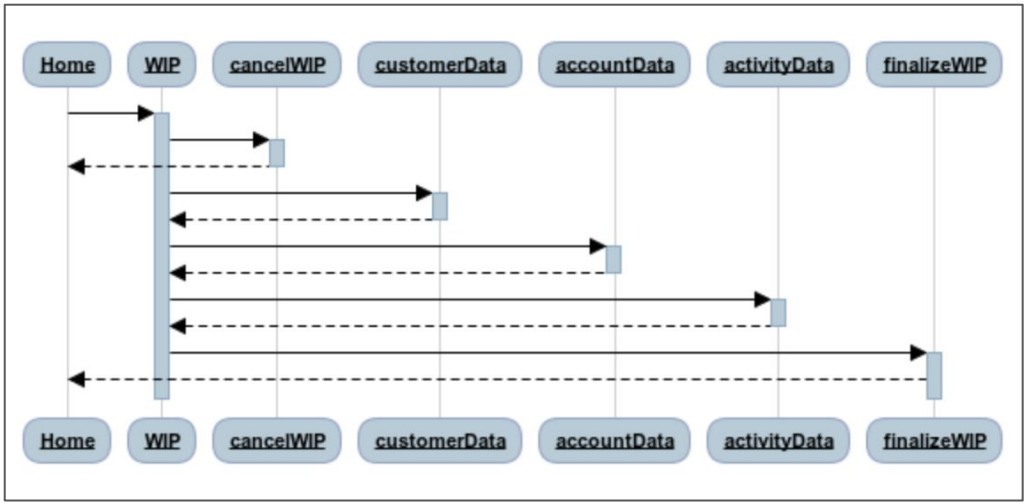
Translating the Design

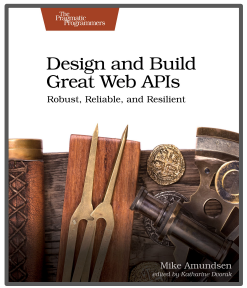
- Assets
  - User Story
  - Sequence Diagram
  - ALPS Profile
- Implementation
  - HTTP
  - Resources
  - Messages



Translating the Design -- Assets

- User Story
- Sequence Diagram
- ALPS Profile

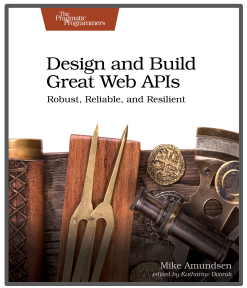




Translating the Design -- Implementation

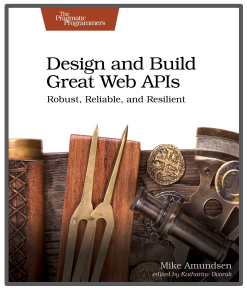
- HTTP
- Resources
- Messages

```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:
```



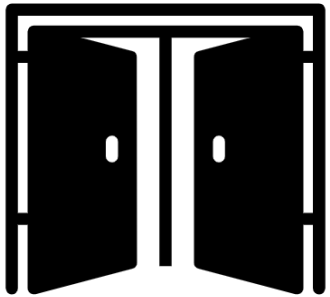
Building APIs

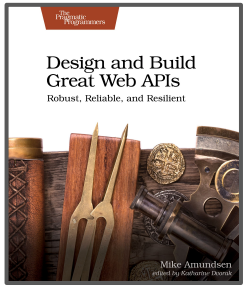
- The DORR Model
- Using NodeJS & ExpressJS



Building APIs -- DORR Model

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

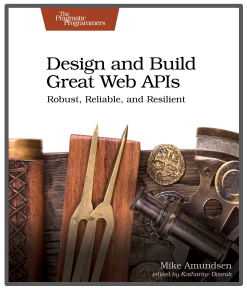




Building APIs -- DORR Model (Data)

simple-storage.js

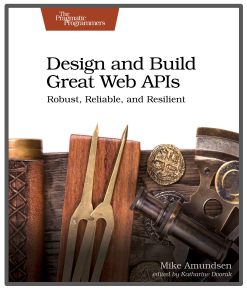
```
/*
 * args is a hash table of possible arguments
 * {object:"",action:"",filter:"",id:"",item:objItem}
 */
function main(args) {
  var rtn;

  // 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

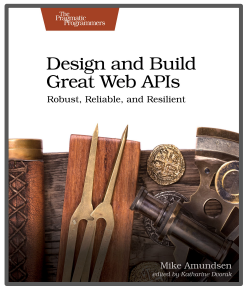
```
// app-level actions for tasks
// args: name, props, reqd, action, id, filter, item
function main(args) {
  var name, rtn, props;
  var conn, action, id, filter, item;

  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)

resources.js

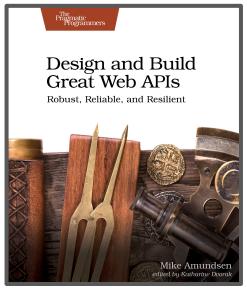
```
/*************************************
 * handle request events
 *************************************/
// home
router.get('/', function (req, res) {
  res.send('{"home" : "{"name":"customer", "rel" : "c
})

// 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)

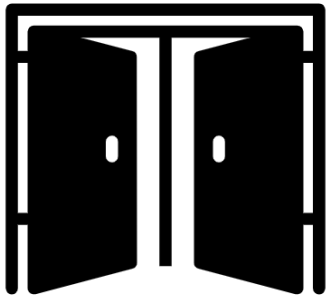
ejs templates

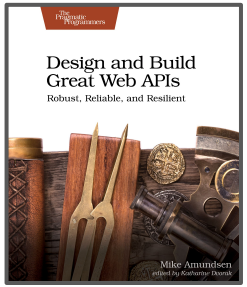
```
var ejs = require('ejs');
var jsonView = '<%= body %>';

// 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

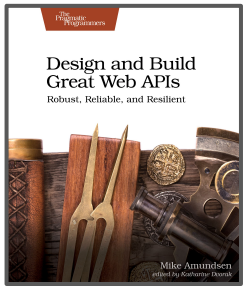
- Use the API-STARTER repo
  https://github.com/mamund/api-starter

- Work through the DORRs
  - Properties.js
  - Actions.js
  - Resources.js
  - (representations)

```
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;
```

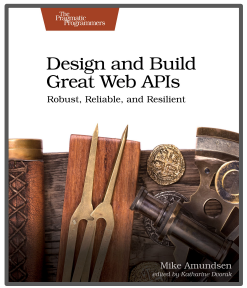
Developing APIs

- APIs are just Interfaces
- Translating the Design
- DORRs



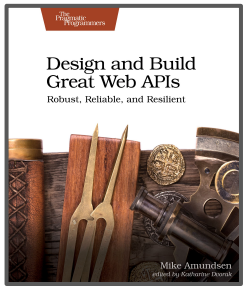
Afternoon -- Session Four

- The Power of Pipelines
- Deploying APIs (heroku CLI)



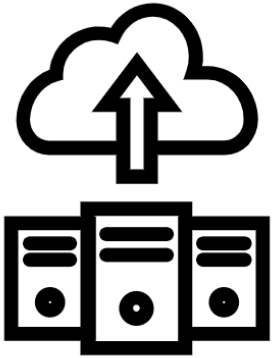
Deploying APIs

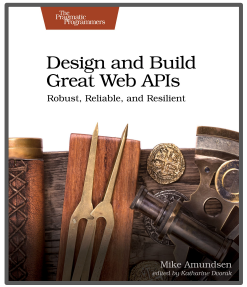
- Git-based Deployment
- Using **Heroku**



Deploying APIs - Challenges

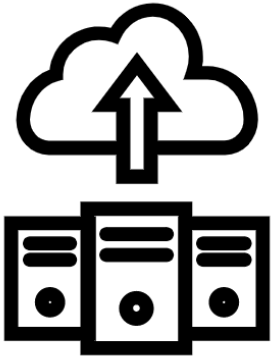
- Deploying your app can be complicated
- Compatibility
  - Hardware
  - OS
  - Platform
  - Framework
  - Dependencies

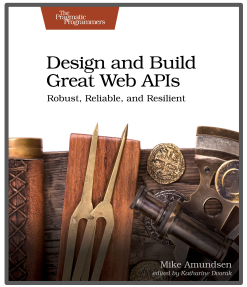




Deploying APIs - DevOps

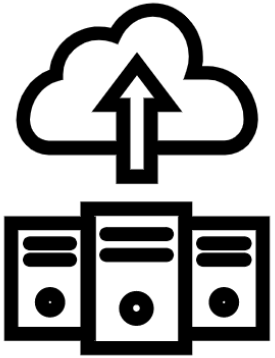
- DevOps was created to help with all this
- Developers & Operators working together
- Started as a hashtag on twitter #DevOps
- Series of small conferences started in 2009
- Emphasis on automation to improve reliability

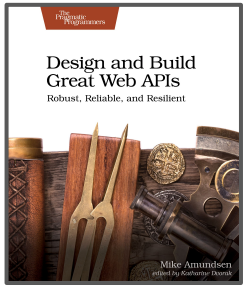




Deploying APIs - Tools

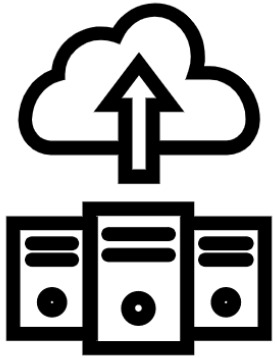
- Build tools
- CI/CD pipeline
- Docker (containers)
- Kubernetes (deployment orchestration)

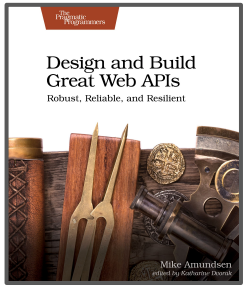




Deploying APIs - Using Heroku

- 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)

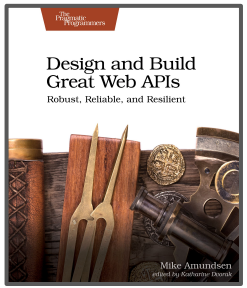




Deploying APIs - Using Heroku

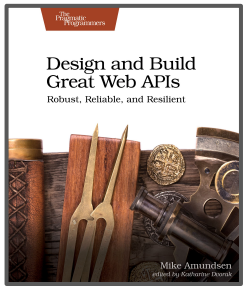
- Cloud platform (2007)
- Originally just for Ruby/Rails projects
- Now supports Java, NodeJS, Python, Go, Clojure, Scala
- Acquired by Salesforce in 2011
- Full platform w/ marketplace ecosystem
- Heroku uses proprietary container tech (Dynos)

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

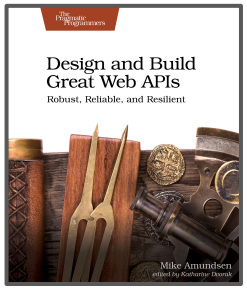
The `heroku create` CLI command creates a new empty application on Heroku, along with an associated empty Git repository. If you run this command from your app's root directory, the empty Heroku Git repository is automatically set as a remote for your local repository.

```
$ 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:

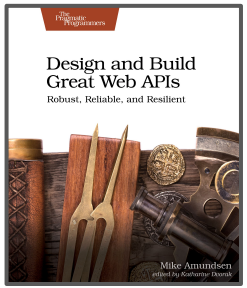
```
$ git remote -v
heroku  https://git.heroku.com/thawing-inlet-61413.git (fetch)
heroku  https://git.heroku.com/thawing-inlet-61413.git (push)
```

HEROKU



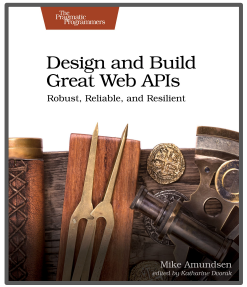
Summary

- Morning Session
  - Designing
    - Method, WSD, ALPS
  - Defining
    - APIB, Swagger
- Afternoon Session
  - Developing
    - DORR, NodeJS & Express
  - Deploying
    - Pipeline, Heroku

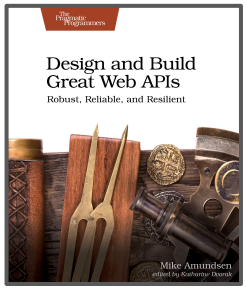


Final Notes

- I'll post all slides in the repo
  https://github.com/mamund/2019-06-goto-amsterdam
- 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
edited by Katharine Dvorak

Building Great APIs
from the Ground Up

Mike Amundsen
@mamund