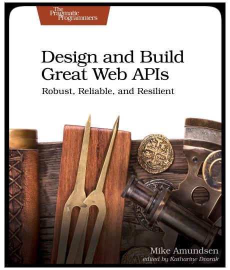
Building Great Web APIs
Part One

@mamund
Mike Amundsen

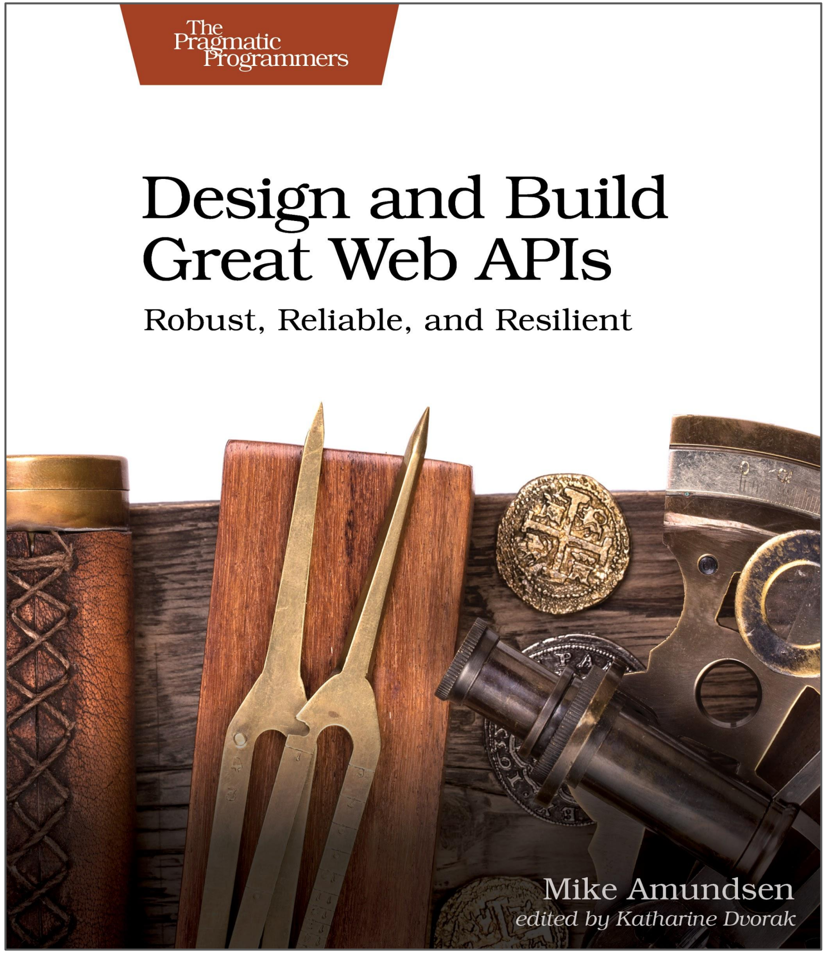

The Pragmatic Programmers
Design and Build Great Web APIs
Robust, Reliable, and Resilient
Mike Amundsen
edited by Katharine Dvorak

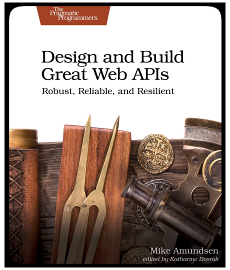
Mike Amundsen
@mamund

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."

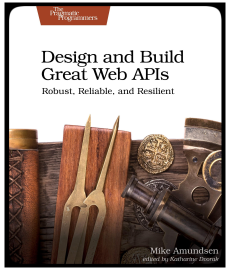
-- Pragmatic Publishers


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


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

Logistics and Preparation

- Introductions
- Workshop Outline
- Zooming


The Pragmatic Programmers

Design and Build Great Web APIs

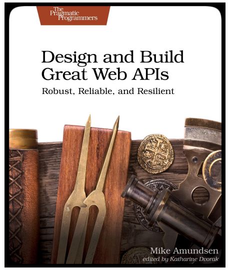
Robust, Reliable, and Resilient

Mike Amundsen

edited by Katharine Dvorak

Introductions

- Name
- Current work
- What you're hoping to learn


The Pragmatic Programmers

Design and Build Great Web APIs

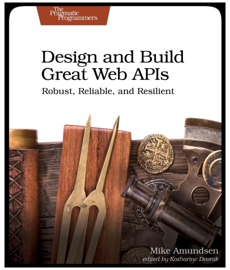
Robust, Reliable, and Resilient

Mike Amundsen

edited by Katharine Dvorak

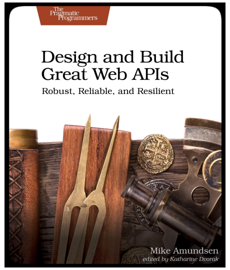
API Design Workshop

- Part One (today)
  - Three-Phase Implementation
  - Sketching and Prototyping
  - Building APIs w/ NodeJS/Express/DARRT
  - Overnight Assignment
- Part Two (tomorrow)
  - Assignment Review
  - Deploying APIs w/ Heroku
  - Open Question Time

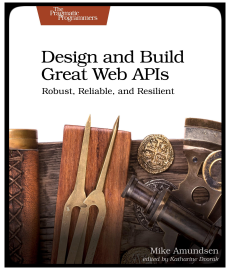

The Pragmatic Programmers
Design and Build Great Web APIs
Robust, Reliable, and Resilient
Mike Amundsen
edited by Katharine Dvorak

Zooming

- Share video feed on whenever possible
- Mute your microphone when not talking
- Raise your hand to share, ask questions, etc.
- Add background questions/comments in chat window
- If you need to leave your desk, turn video off

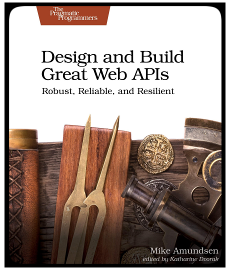

The Pragmatic Programmers
Design and Build Great Web APIs
Robust, Reliable, and Resilient
Mike Amundsen
edited by Katharine Dvorak

Three-Phase Implementation


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

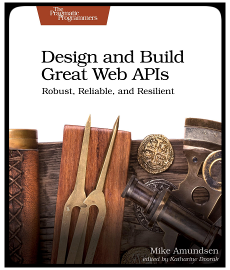
Three-Phase API Implementation

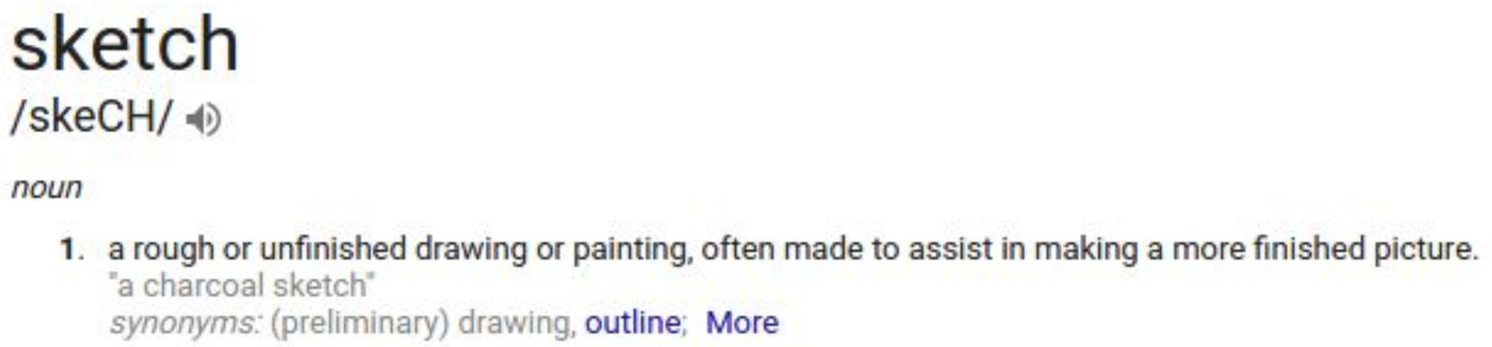
- To reduce cost and risk, take a three-phase approach
- **Sketching** - disposable experiments
- **Prototyping** - testable examples
- **Building** - production implementation


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

Reduce Cost and Risk in API Implementation

- Implementation can be costly
- Mistakes may be uncovered along the way
- Uncover mistakes early when they are inexpensive to fix
- Put off writing code for as long as possible.


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

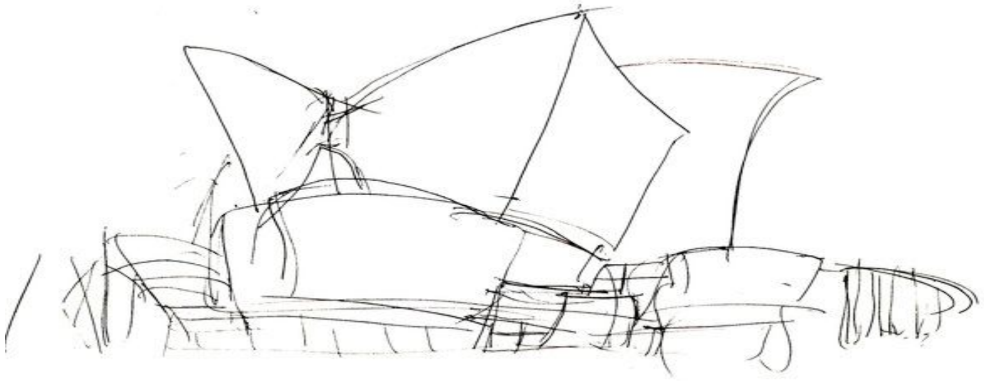


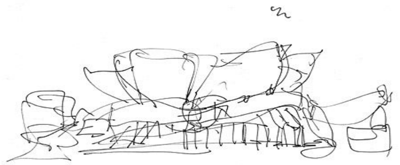
sketch

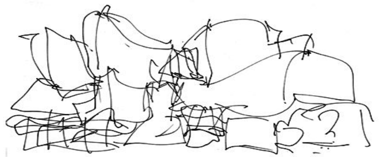
/skeCH/

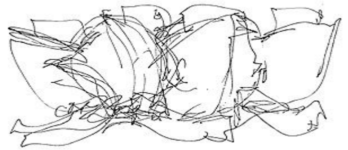
noun

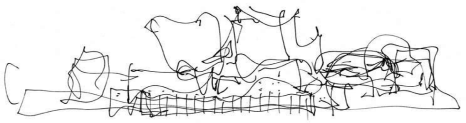
1. a rough or unfinished drawing or painting, often made to assist in making a more finished picture.
   "a charcoal sketch"
   synonyms: (preliminary) drawing, outline; More

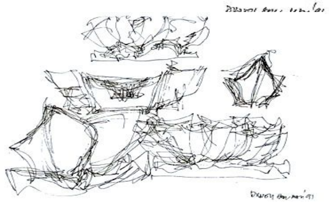





F Gehry Disney Hall Midblock

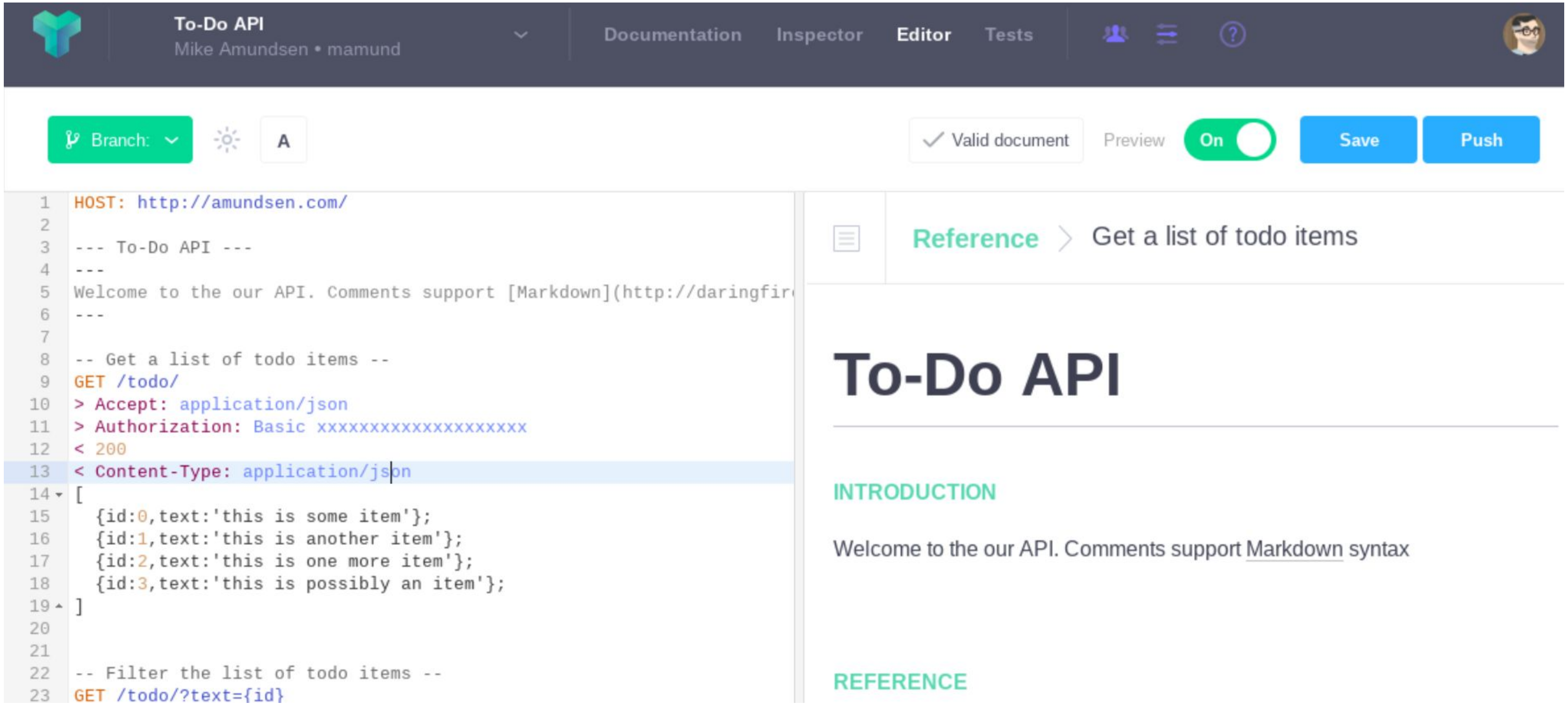



F Gehry Disney Hall Oct '91.



Sketching APIs

- 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.



```
HOST: http://amundsen.com/

--- To-Do API ---
---
Welcome to the our API. Comments support [Markdown](http://daringfir
---

-- Get a list of todo items --
GET /todo/
> Accept: application/json
> Authorization: Basic xxxxxxxxxxxxxxxxxxxx
< 200
< Content-Type: application/json
[
  {id:0,text:'this is some item'};
  {id:1,text:'this is another item'};
  {id:2,text:'this is one more item'};
  {id:3,text:'this is possibly an item'};
]


-- Filter the list of todo items --
GET /todo/?text={id}
```

Reference > Get a list of todo items

To-Do API

INTRODUCTION

Welcome to the our API. Comments support Markdown syntax

REFERENCE

Sketches are made to be thrown away.

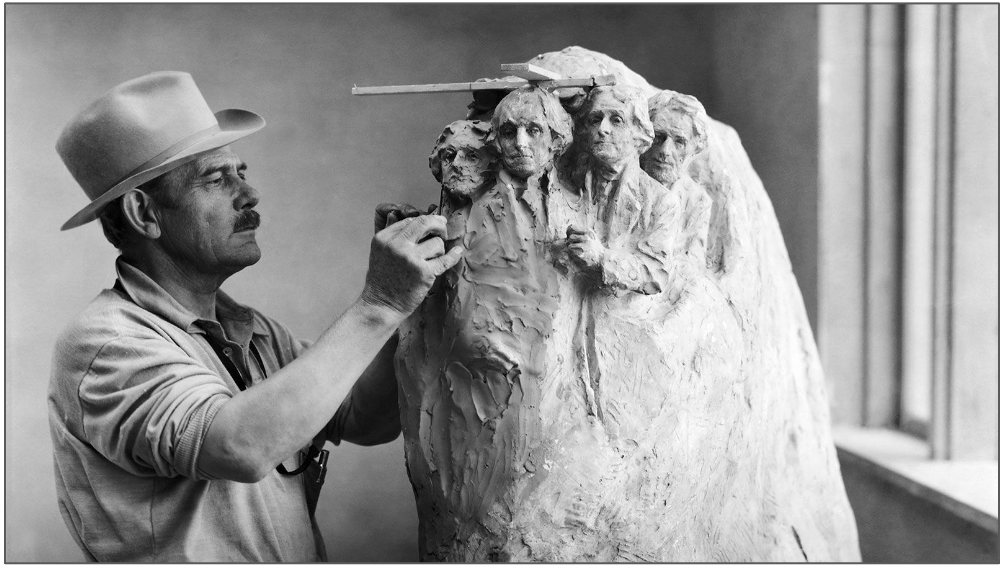
pro·to·type

/ˈprōdəˌtīp/

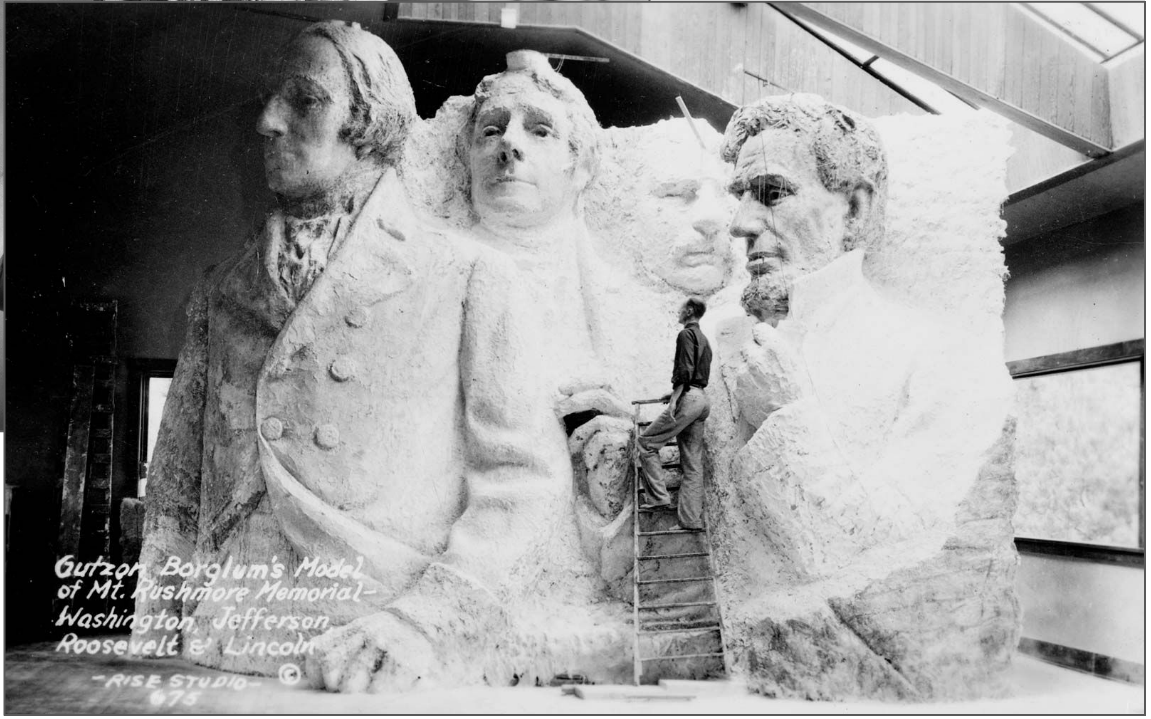
noun

1. a first, typical or preliminary model of something, especially a machine, from which other forms are developed or copied.
   "the firm is testing a prototype of the weapon"

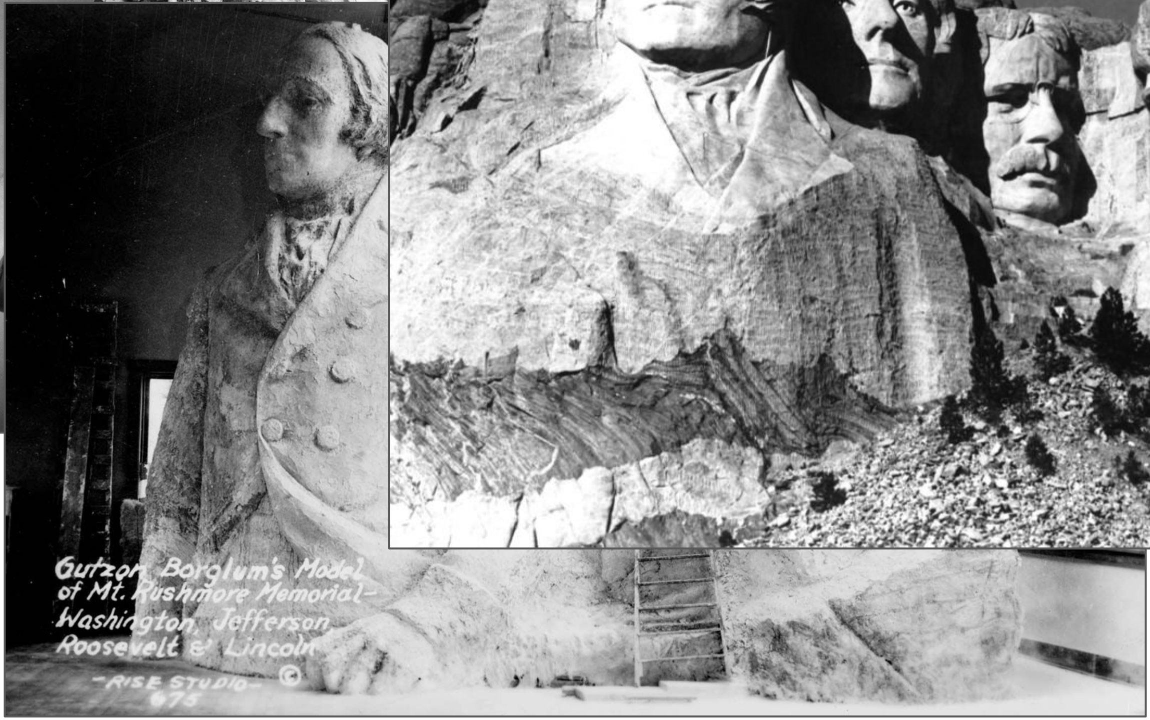
Borglum's Prototypes

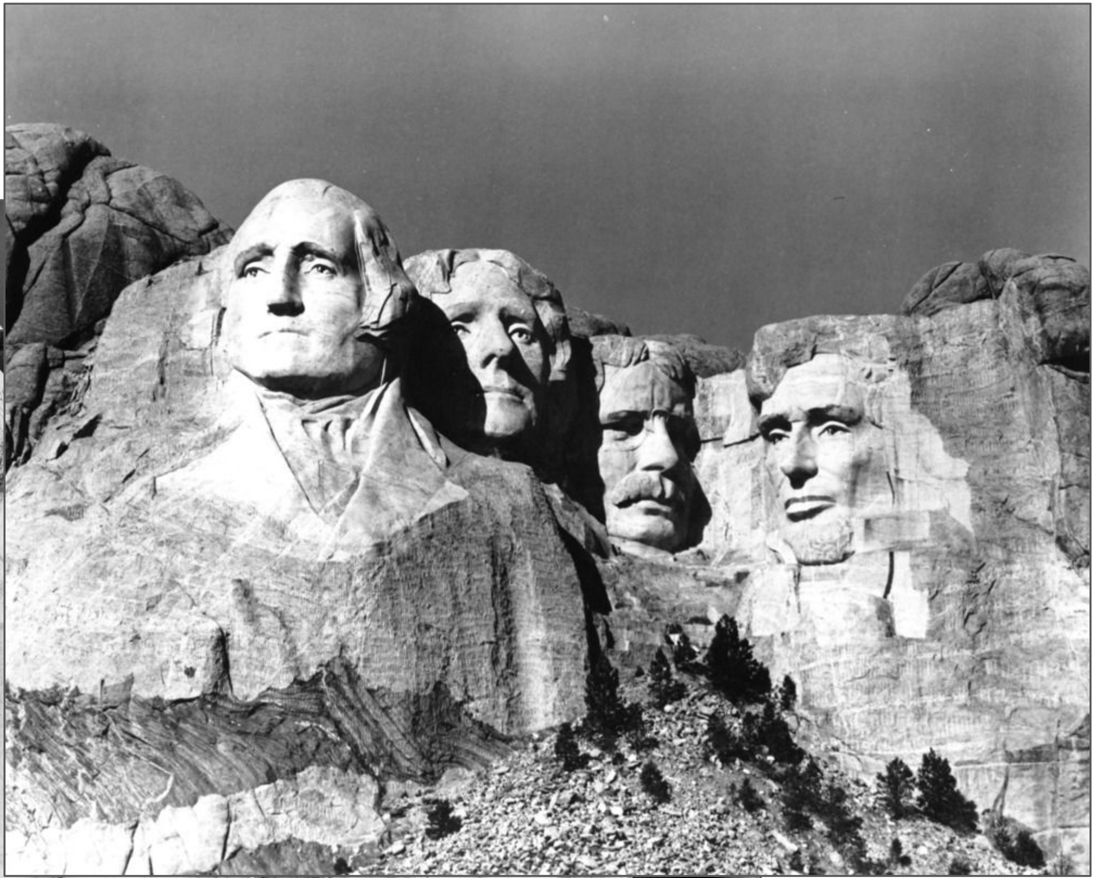


Borglum's Prototypes


Gutzon Borglum's Model of Mt. Rushmore Memorial - Washington, Jefferson Roosevelt & Lincoln
-RISE STUDIO- ©
675

Borglum's Prototypes

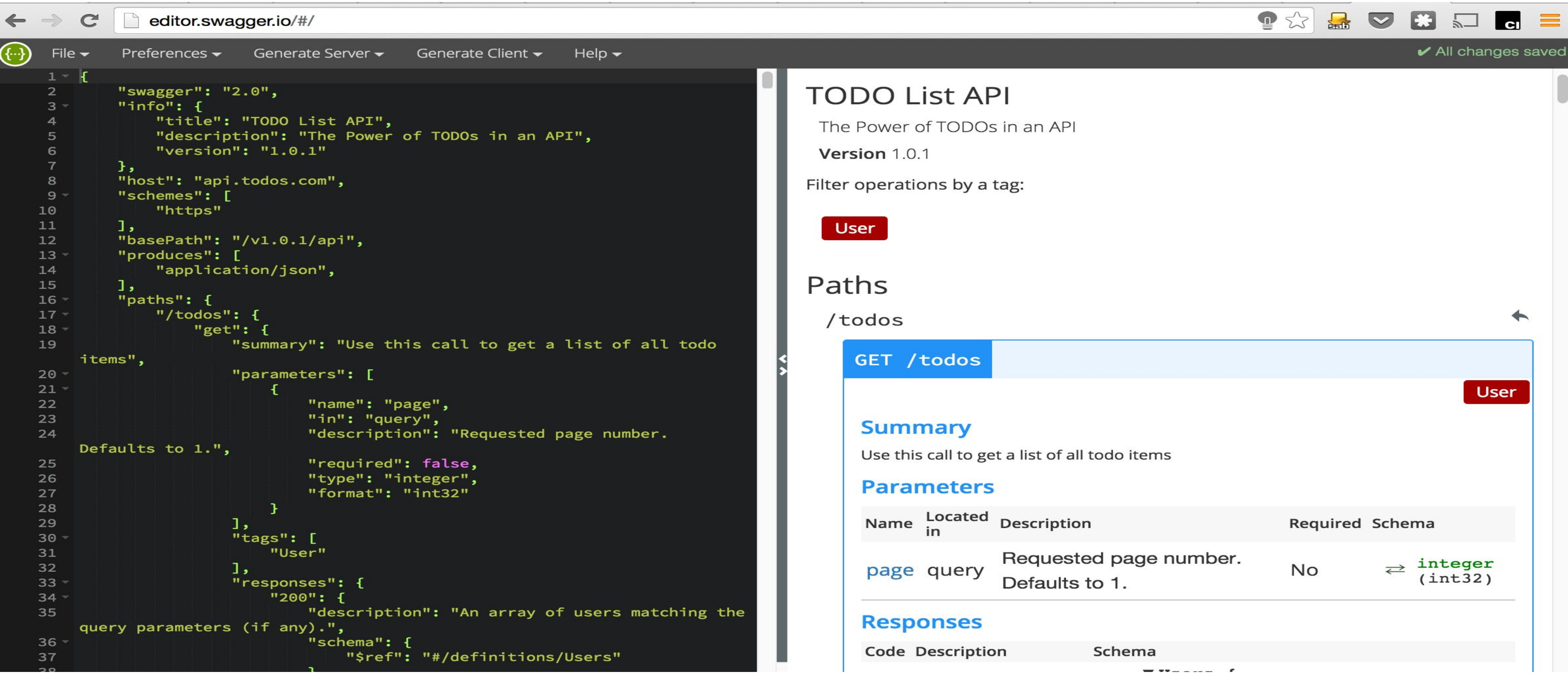




Prototyping APIs

- 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.

Swagger Editor



```
{
    "swagger": "2.0",
    "info": {
        "title": "TODO List API",
        "description": "The Power of TODOs in an API",
        "version": "1.0.1"
    },
    "host": "api.todos.com",
    "schemes": [
        "https"
    ],
    "basePath": "/v1.0.1/api",
    "produces": [
        "application/json",
    ],
    "paths": {
        "/todos": {
            "get": {
                "summary": "Use this call to get a list of all todo
items",
                "parameters": [
                    {
                        "name": "page",
                        "in": "query",
                        "description": "Requested page number.
Defaults to 1.",
                        "required": false,
                        "type": "integer",
                        "format": "int32"
                    }
                ],
                "tags": [
                    "User"
                ],
                "responses": {
                    "200": {
                        "description": "An array of users matching the
query parameters (if any).",
                        "schema": {
                            "$ref": "#/definitions/Users"
```

TODO List API

The Power of TODOs in an API

Version 1.0.1

Filter operations by a tag:

User

Paths

/todos

GET /todos

User

Summary

Use this call to get a list of all todo items

Parameters

Name	Located in	Description	Required	Schema
page	query	Requested page number. Defaults to 1.	No	integer (int32)

Responses

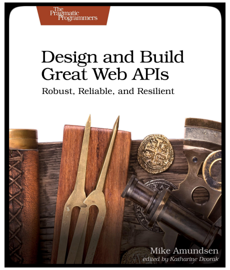
Code	Description	Schema

Prototyping APIs

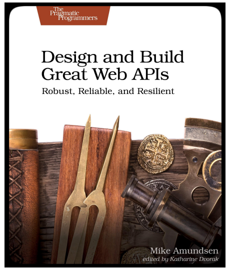
- Use tools like NodeJS express,
  or other code-generating platforms.
- It’s also a good idea to use service-virtualization
frameworks to mock up the response data.
- If possible, include access-control checking when
  running tests against the prototype.
- If possible use existing production-level API
  consumers to test out the prototype.

Prototypes are made to be tested.

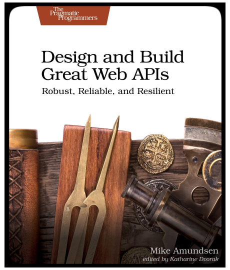
Let's Discuss!


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

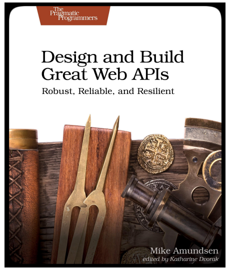
BREAK


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

Prototyping with OpenAPI

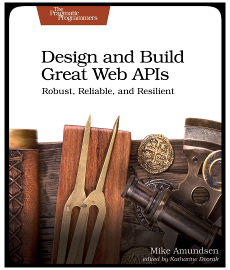

The Pragmatic Programmers
Design and Build Great Web APIs
Robust, Reliable, and Resilient
Mike Amundsen
edited by Katharine Dvorak

Exercise: Prototyping your API

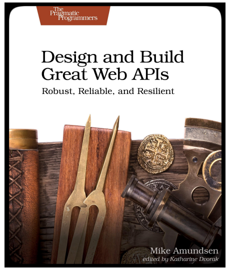

The Pragmatic Programmers
Design and Build Great Web APIs
Robust, Reliable, and Resilient
Mike Amundsen
edited by Katharine Dvorak

Exercise: Prototyping your API

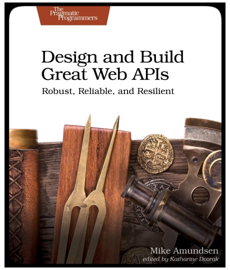
- Start with your ALPS API Description (from repo)
  - https://github.com/mamund/2020-04-goto-chicago-api-build
- Hand-Roll your OpenAPI Document
  - `info`
  - `paths`
  - `components`
- Validate your OpenAPI Document w/ SwaggerHub
  - https://app.swaggerhub.com/home
- Generate Docs using Redoc HTML page (from the repo)
  -


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

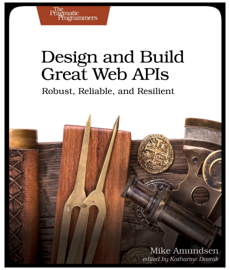
Exercise: Stand-Up

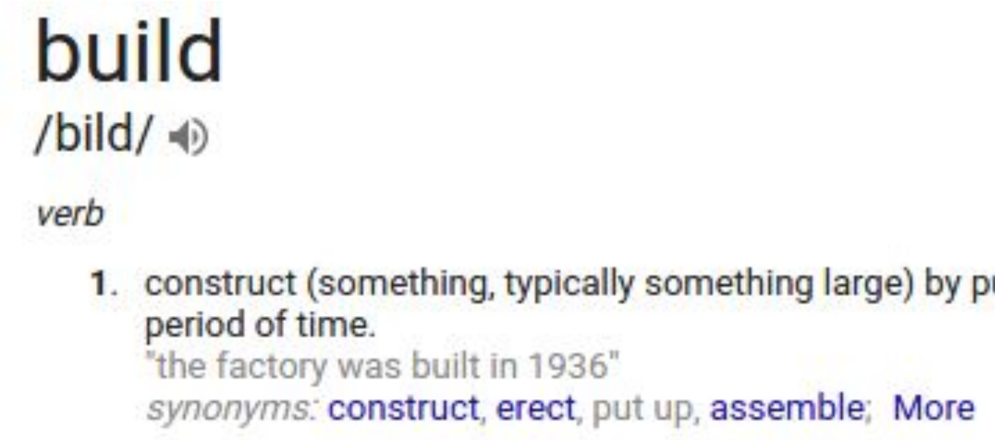

The Pragmatic Programmers
Design and Build Great Web APIs
Robust, Reliable, and Resilient
Mike Amundsen
edited by Katharine Dvorak

BREAK


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

Building w/ NodeJS/Express/DARRT


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

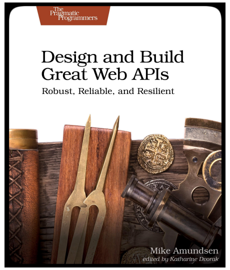


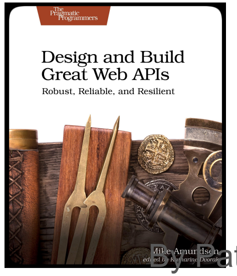
build

/bild/

verb

1. construct (something, typically something large) by putting parts or material together over a period of time.
"the factory was built in 1936"
synonyms: construct, erect, put up, assemble; More


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



The Pragmatic Programmers

Design and Build Great Web APIs

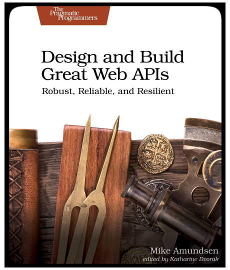
Robust, Reliable, and Resilient

Mike Amundsen
edited by Katharine Dvorak

By Patrick Creighton - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=49014485

Building APIs

- API builds are the real thing
- Production-ready, access-controlled, resilient, scalable.
- Building the production implementation means
  - Working out all the kinks
  - Supporting all the use-cases identified during the sketch and prototype phases.



The Pragmatic Programmers

Design and Build Great Web APIs

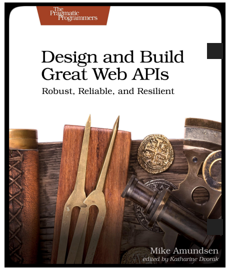
Robust, Reliable, and Resilient

Mike Amundsen

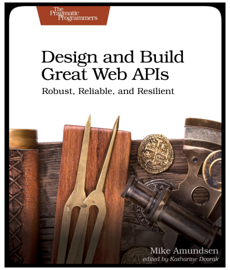
edited by Katharine Dvorak

Building APIs

- Each implementation has their own challenges to overcome.
- Each deserves their own guidance and style-guides.
  - Gateway Policies
  - ESB Rules
  - Scripting (NodeJS)
  - Code (Java/C#)
- All require exhaustive testing at the unit, acceptance, and integration levels.
- All require detailed access control.

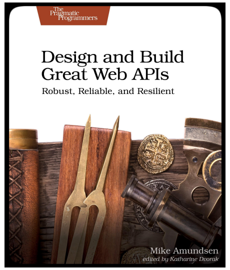

The Pragmatic Programmers
Design and Build Great Web APIs
Robust, Reliable, and Resilient
Mike Amundsen
edited by Katharine Dvorak

Production APIs are made last.


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

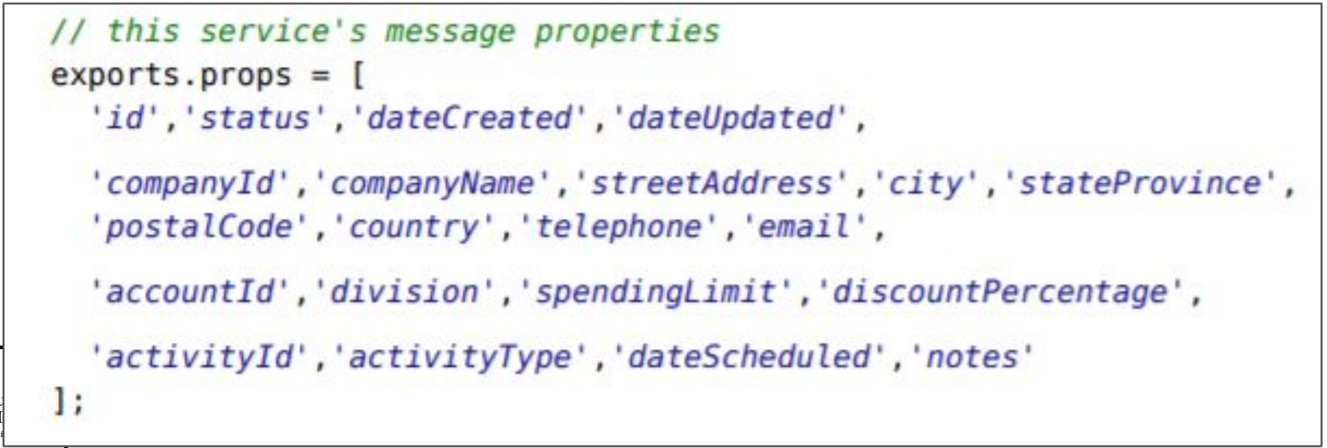
Building APIs : DARRT

- Simple process for publishing running interfaces
- Data
- Actions
- Resources
- Representations
- Transitions

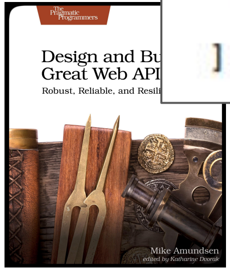

The Pragmatic Programmers
Design and Build Great Web APIs
Robust, Reliable, and Resilient
Mike Amundsen
edited by Katharine Dvorak

Building APIs : DARRT : Data

- The state properties to pass in messages
  - `properties, requireds, enums, defaults`

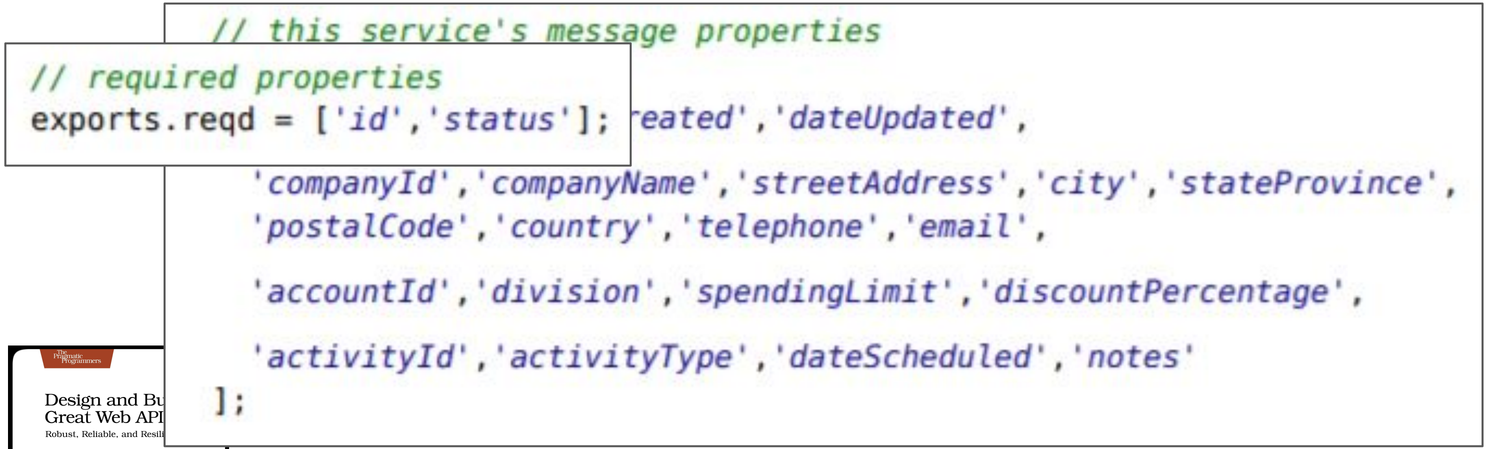


```
// this service's message properties
exports.props = [
  'id','status','dateCreated','dateUpdated',

  'companyId','companyName','streetAddress','city','stateProvince',
  'postalCode','country','telephone','email',

  'accountId','division','spendingLimit','discountPercentage',

  'activityId','activityType','dateScheduled','notes'
];
```


The Pragmatic Programmers
Design and Bu
Great Web API
Robust, Reliable, and Resili
Mike Amundsen
edited by Katharine Dvorak

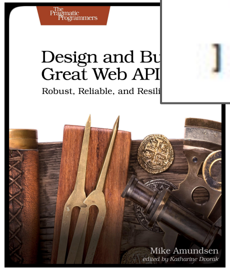
Building APIs : DARRT : Data

- The state properties to pass in messages
  - `properties, requireds, enums, defaults`



```
// this service's message properties
                                  eated','dateUpdated',
  'companyId','companyName','streetAddress','city','stateProvince',
  'postalCode','country','telephone','email',

  'accountId','division','spendingLimit','discountPercentage',

  'activityId','activityType','dateScheduled','notes'
];
```

```
// required properties
exports.reqd = ['id','status'];
```



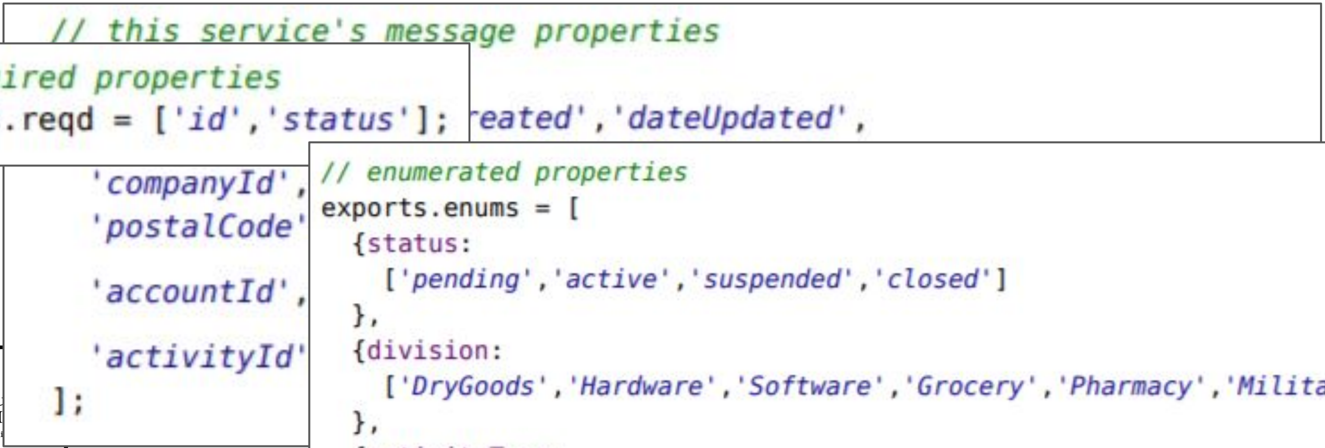
The Pragmatic Programmers

Design and Bu
Great Web API
Robust, Reliable, and Resili

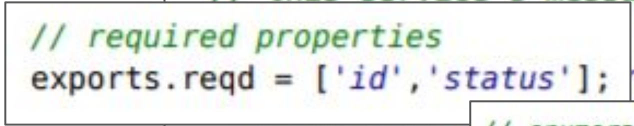
Mike Amundsen
edited by Katharine Dvorak

Building APIs : DARRT : Data

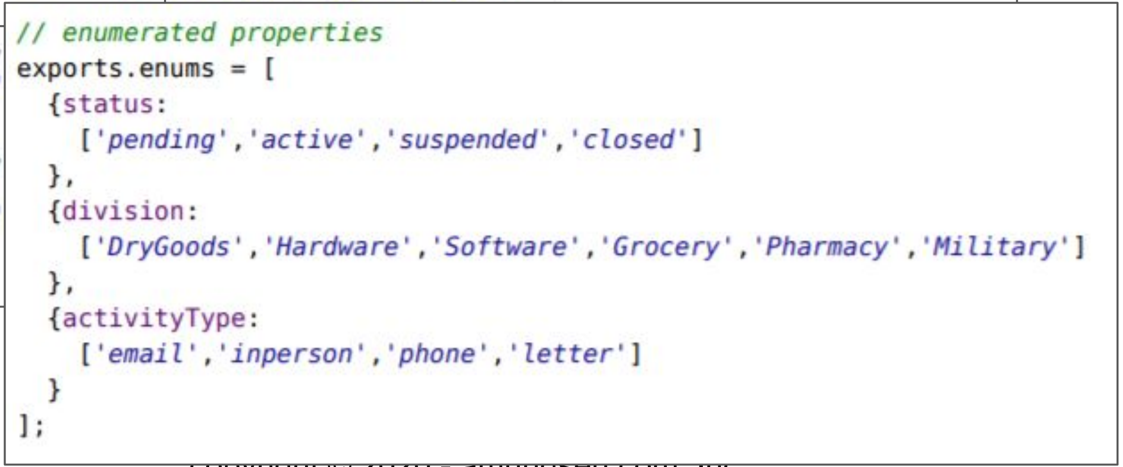
- The state properties to pass in messages
  - `properties, requireds, enums, defaults`



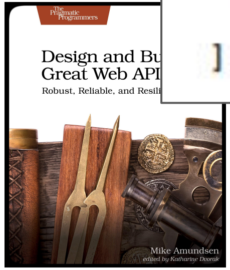
```
// this service's message properties

eated','dateUpdated',
    'companyId',
    'postalCode'

    'accountId',

    'activityId'
];
```



```
// required properties
exports.reqd = ['id','status'];
```

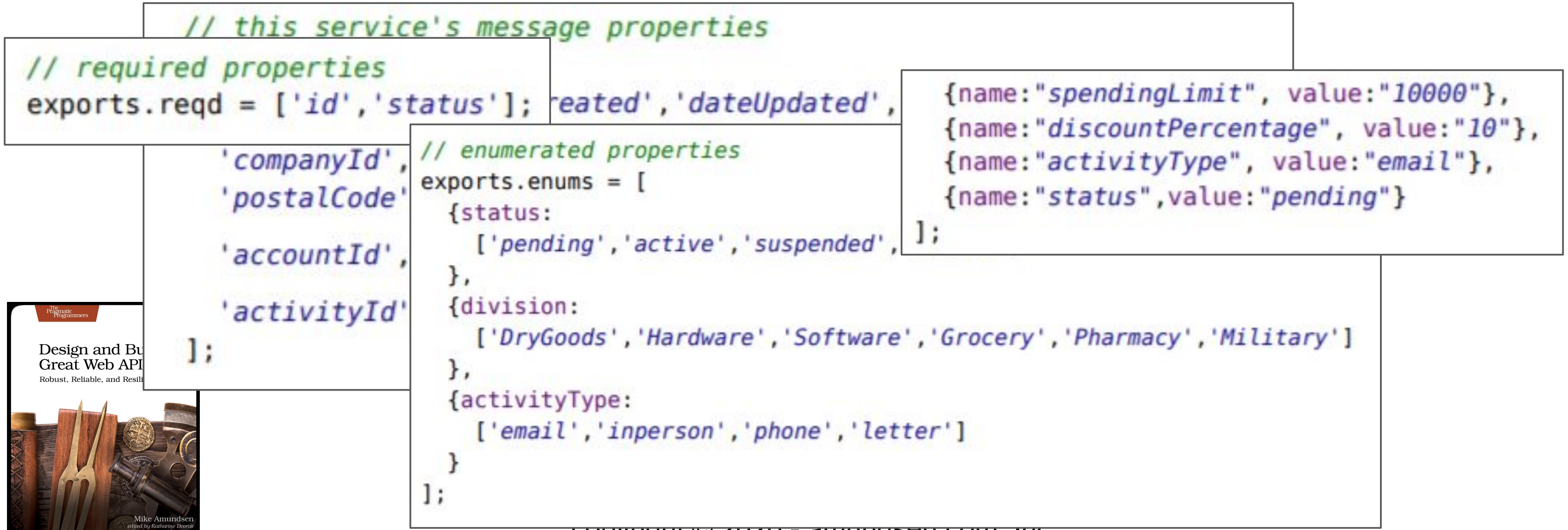


```
// enumerated properties
exports.enums = [
  {status:
    ['pending','active','suspended','closed']
  },
  {division:
    ['DryGoods','Hardware','Software','Grocery','Pharmacy','Military']
  },
  {activityType:
    ['email','inperson','phone','letter']
  }
];
```


The Pragmatic Programmers
Design and Bu
Great Web API
Robust, Reliable, and Resili
Mike Amundsen
edited by Katharine Dvorak

Building APIs : DARRT : Data

- The state properties to pass in messages
    - `properties, requireds, enums, defaults`

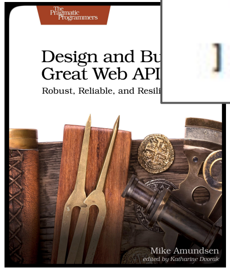


```
// this service's message properties
        reated','dateUpdated',
    'companyId',
    'postalCode'

    'accountId',

    'activityId'
];
```

```
// required properties
exports.reqd = ['id','status'];
```

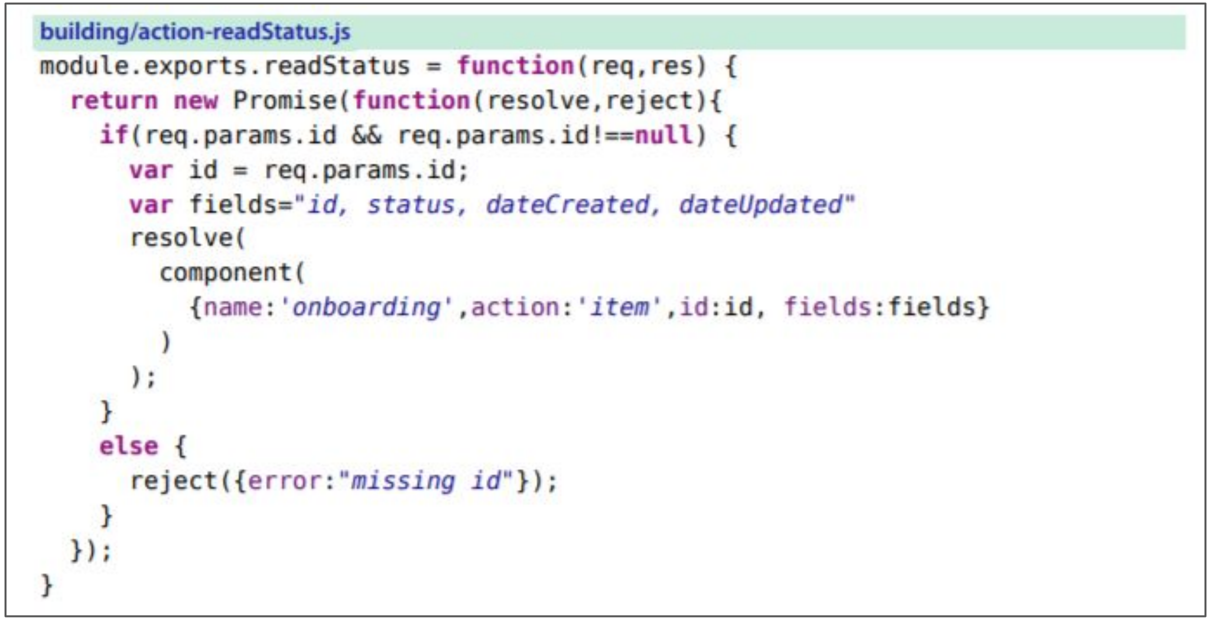
```
// enumerated properties
exports.enums = [
  {status:
    ['pending','active','suspended',
  },
  {division:
    ['DryGoods','Hardware','Software','Grocery','Pharmacy','Military']
  },
  {activityType:
    ['email','inperson','phone','letter']
  }
];
```

```
  {name:"spendingLimit", value:"10000"},
  {name:"discountPercentage", value:"10"},
  {name:"activityType", value:"email"},
  {name:"status",value:"pending"}
];
```


The Pragmatic Programmers
Design and Bu
Great Web API
Robust, Reliable, and Resili
Mike Amundsen
edited by Katharine Dvorak

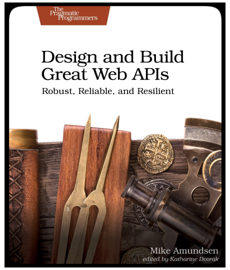
Building APIs : DARRT : Actions

- The actual operations for the interface
  - appovePayroll, updateCustomer, setStatus



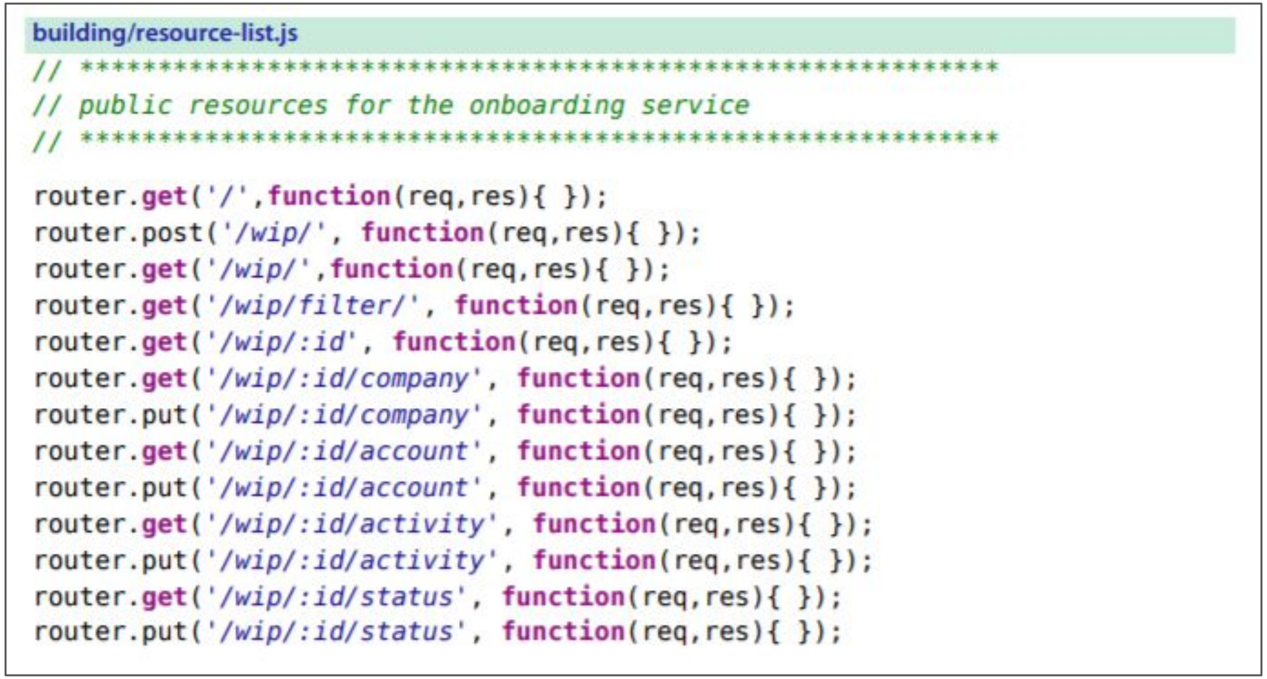
building/action-readStatus.js

```
module.exports.readStatus = function(req,res) {
  return new Promise(function(resolve,reject){
    if(req.params.id && req.params.id!==null) {
      var id = req.params.id;
      var fields="id, status, dateCreated, dateUpdated"
      resolve(
        component(
          {name:'onboarding',action:'item',id:id, fields:fields}
        )
      );
    }
    else {
      reject({error:"missing id"});
    }
  });
}
```



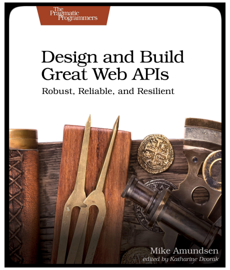
Building APIs : DARRT : Resources

- The HTTP resources to access the operations



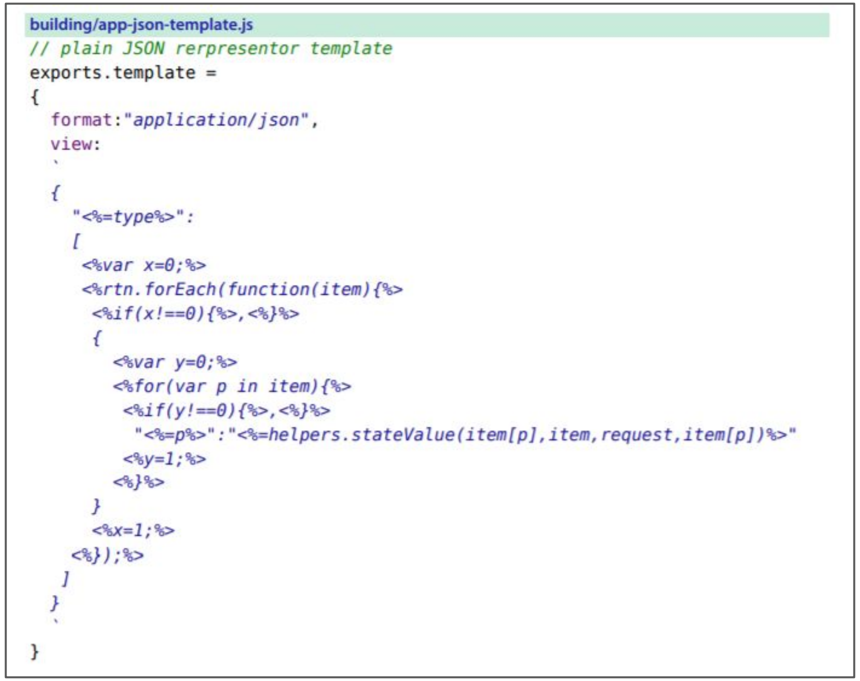
building/resource-list.js

```
// *************************************************************
// public resources for the onboarding service
// *************************************************************

router.get('/',function(req,res){ });
router.post('/wip/', function(req,res){ });
router.get('/wip/',function(req,res){ });
router.get('/wip/filter/', function(req,res){ });
router.get('/wip/:id', function(req,res){ });
router.get('/wip/:id/company', function(req,res){ });
router.put('/wip/:id/company', function(req,res){ });
router.get('/wip/:id/account', function(req,res){ });
router.put('/wip/:id/account', function(req,res){ });
router.get('/wip/:id/activity', function(req,res){ });
router.put('/wip/:id/activity', function(req,res){ });
router.get('/wip/:id/status', function(req,res){ });
router.put('/wip/:id/status', function(req,res){ });
```


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

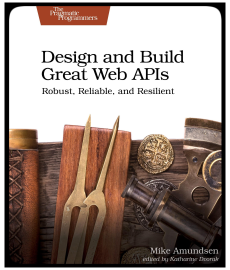
Building APIs : DARRT : Representations

- The format/media-type of resource responses



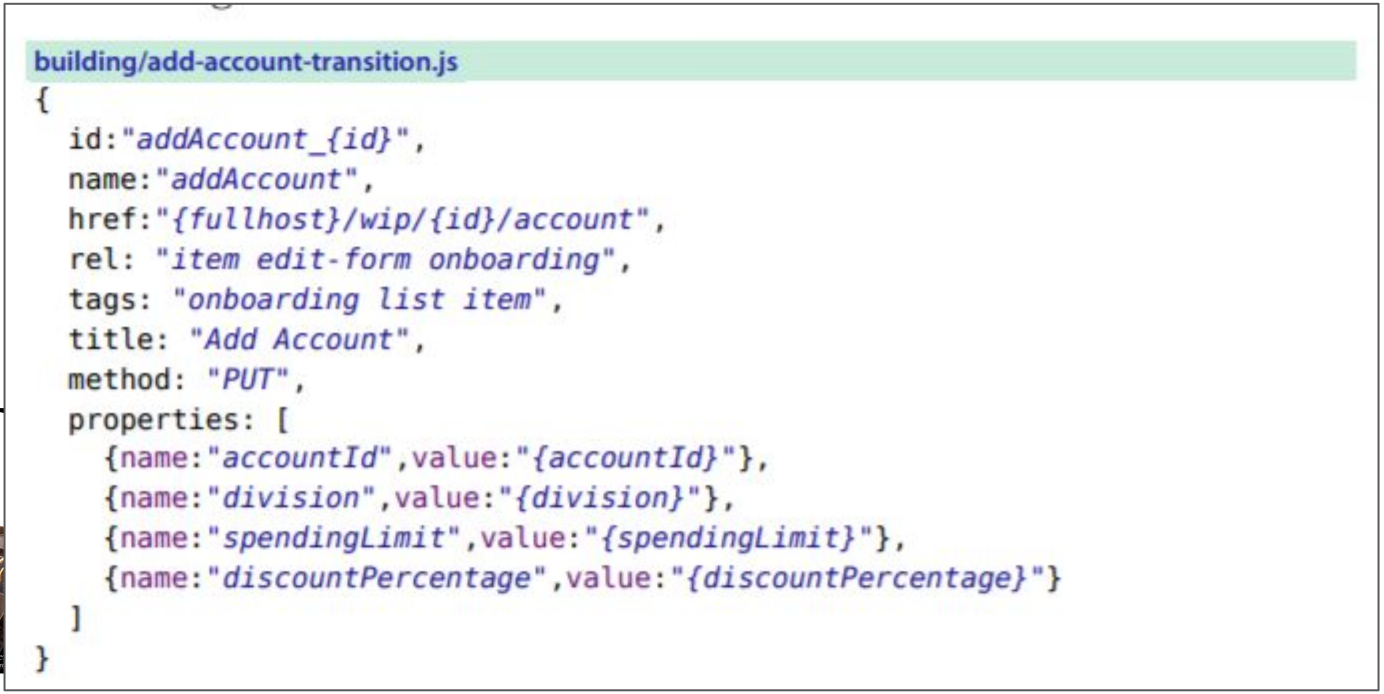
building/app-json-template.js

```
// plain JSON rerpresentor template
exports.template =
{
  format:"application/json",
  view:
  `
  {
    "<%=type%>":
    [
     <%var x=0;%>
     <%rtn.forEach(function(item){%>
      <%if(x!==0){%>,<%}%>
      {
        <%var y=0;%>
        <%for(var p in item){%>
         <%if(y!==0){%>,<%}%>
          "<%=p%>":"<%=helpers.stateValue(item[p],item,request,item[p])%>"
         <%y=1;%>
        <%}%>
      }
      <%x=1;%>
    <%});%>
   ]
  }
  `
}
```


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

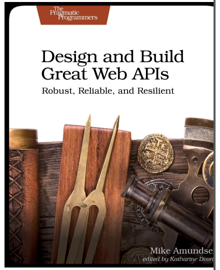
Building APIs : DARRT : Transitions

- The public expression of actions



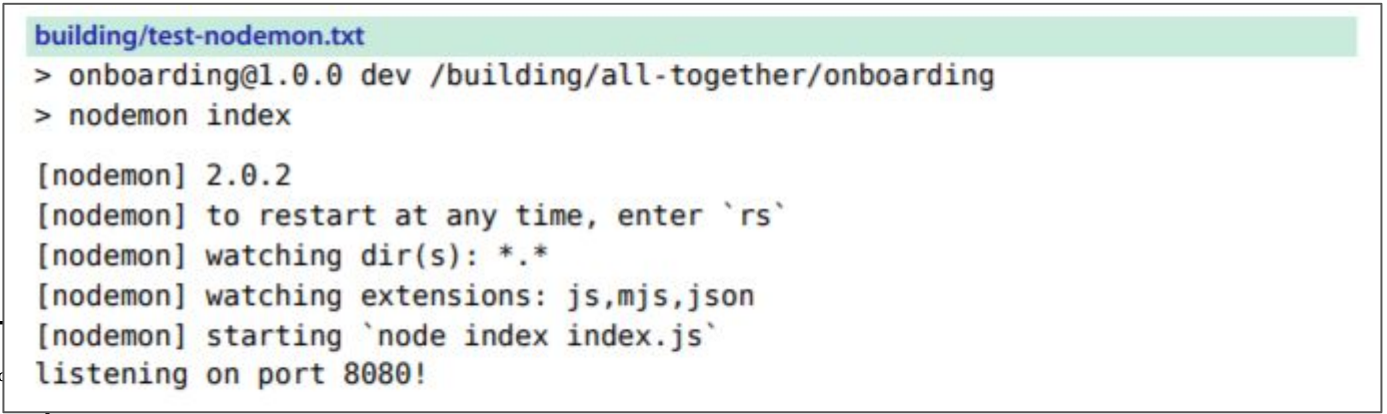
building/add-account-transition.js

```
{
  id:"addAccount_{id}",
  name:"addAccount",
  href:"{fullhost}/wip/{id}/account",
  rel: "item edit-form onboarding",
  tags: "onboarding list item",
  title: "Add Account",
  method: "PUT",
  properties: [
    {name:"accountId",value:"{accountId}"},
    {name:"division",value:"{division}"},
    {name:"spendingLimit",value:"{spendingLimit}"},
    {name:"discountPercentage",value:"{discountPercentage}"}
  ]
}
```


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

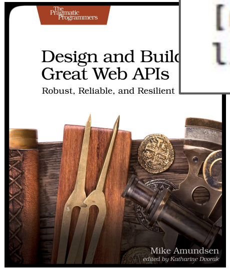
Building APIs : Putting it all together

- Use **`nodemon`** when testing your service locally

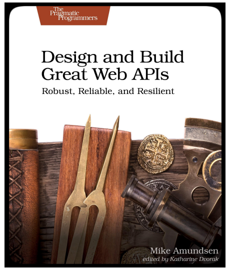


building/test-nodemon.txt

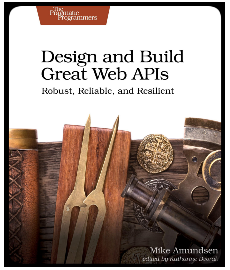
```
> onboarding@1.0.0 dev /building/all-together/onboarding
> nodemon index

[nodemon] 2.0.2
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index index.js`
listening on port 8080!
```


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

Let's Discuss!

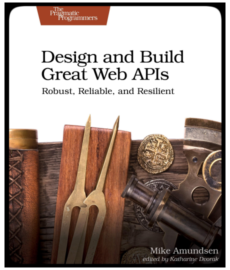

The Pragmatic Programmers
Design and Build Great Web APIs
Robust, Reliable, and Resilient
Mike Amundsen
edited by Katharine Dvorak

Exercise: Installing Tools

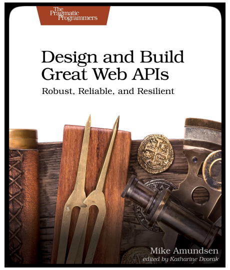

The Pragmatic Programmers
Design and Build Great Web APIs
Robust, Reliable, and Resilient
Mike Amundsen
edited by Katharine Dvorak

Exercise: Installing Tools

- Use the Installing document in the course repo as a guide
  - Skip Postman/Newman for this course
- Curl
- Git
- Github/SSH
- NodeJS/npm
  - `npm install -g nodemon`
- Heroku Client
  - You'll need to start an account
- FORK the API-Starter kit
  - https://github.com/api-tool-kit/api-starter-kit

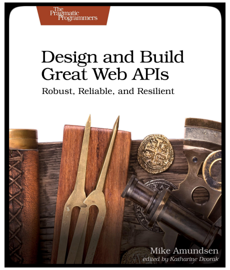

The Pragmatic Programmers
Design and Build Great Web APIs
Robust, Reliable, and Resilient
Mike Amundsen
edited by Katharine Dvorak

Your Assignment


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

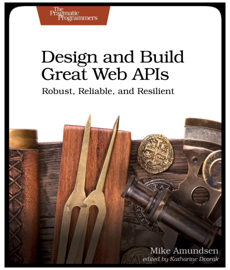
Overnight Assignment for API Design

- Start from your ALPS Description
- Using the API Starter Kit…
  - Update `data.js`, `actions.js`, & `resources.js` as needed
- Use `npm run dev` to validate the API Project


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

Building Great Web APIs
Part One

@mamund
Mike Amundsen


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