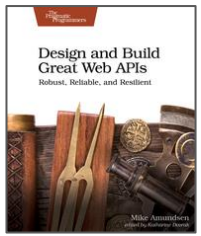
Testing your APIs with Postman and Newman

Mike Amundsen
@mamund
youtube.com/mamund

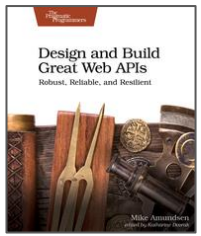


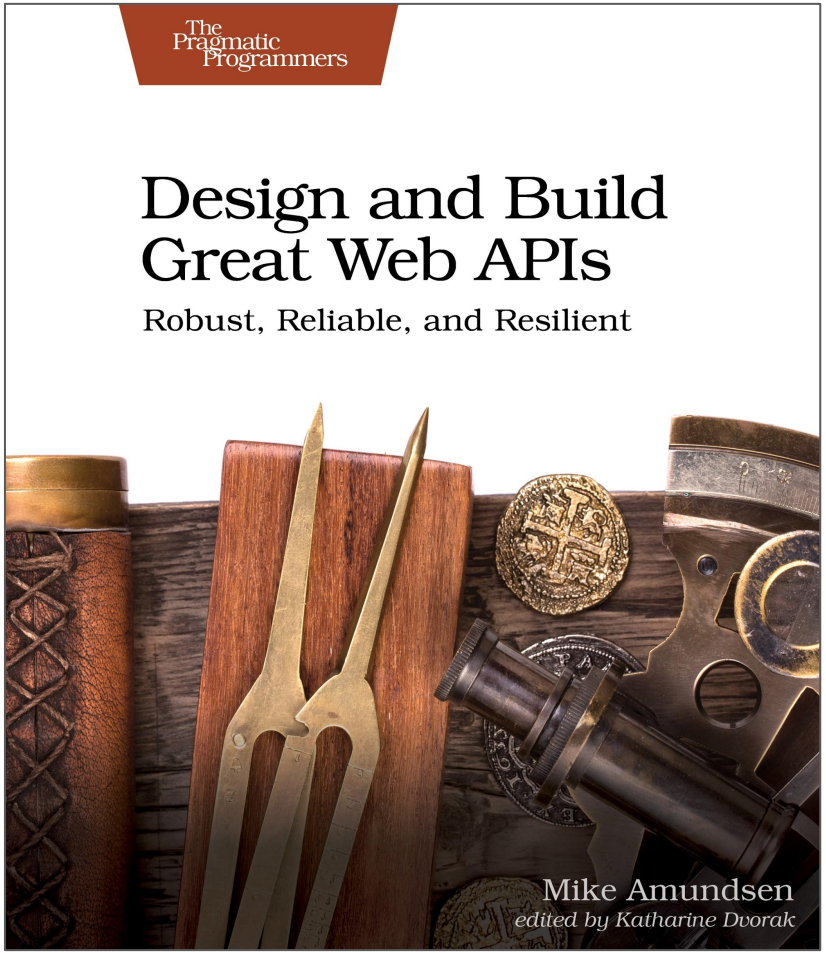

Mike Amundsen
@mamund

g.mamund.com/GreatWebAPIs

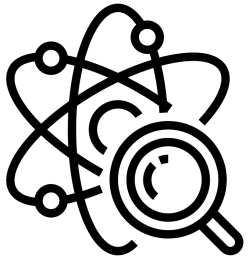
"From design to code to test to deployment, unlock hidden business value and release stable and scalable web APIs that meet customer needs and solve important business problems in a consistent and reliable manner."

-- Pragmatic Publishers

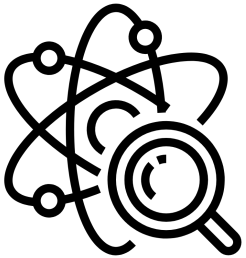




Overview

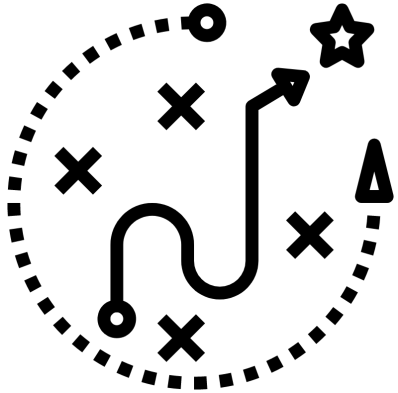


- The Goals of API Testing
- Simple Request Testing with SRTs
- Testing with Postman UI and ChaiJS Assertions
- Automate Testing with Newman CLI
- Summary



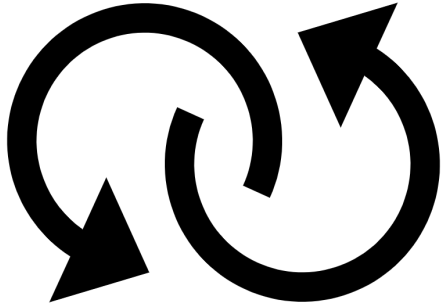
The Goals of API Testing

Goals of API Testing



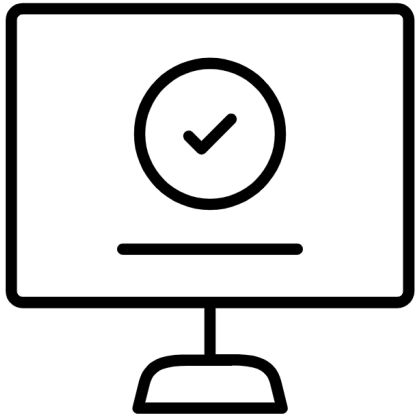
- "Outside-in" approach
- Validating resource inputs/outputs
- Confirming interface behavior

Goals of API Testing

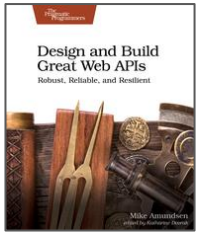


- "Outside-in" approach
  - We can only "see" the API
  - Your tests are for the interface, not the code
- Validating resource inputs/outputs
- Confirming interface behavior

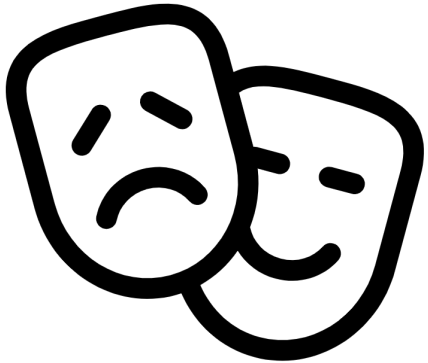
Goals of API Testing



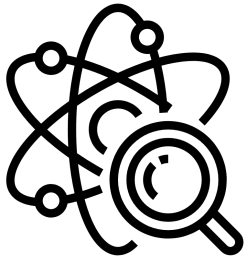
- "Outside-in" approach
- Validating resource inputs/outputs
  - Confirm the URL exists, responds
  - Validate the methods, parameters, and response bodies
- Confirming interface behavior



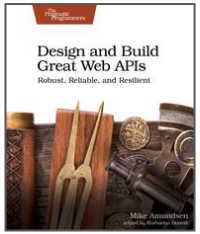
Goals of API Testing

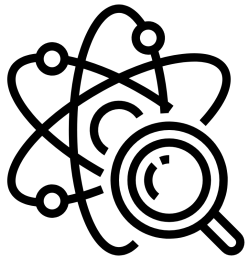


- "Outside-in" approach
- Validating resource inputs/outputs
- Confirming interface behavior
  - Validate expected behaviors
  - Happy-path tests (200 OK)
  - Sad-path tests (400 Bad Request)



Let's see how that works...

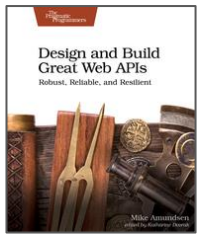




Simple Request Testing with SRTs

Simple Request Testing (SRTs)

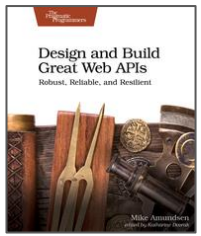
- SRTs are a great way to validate endpoints
- Use `curl` to execute simple requests against your API
- "Eyeball" the results to make sure it works as expected



Simple Request Testing (SRTs)

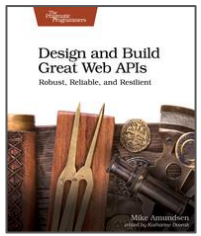
- SRTs are a great way to validate endpoints
  - Confirm URLs respond
  - Validate 200 OK & body
- Use **`curl`** to execute simple requests against your API
- "Eyeball" the results to make sure it works as expected

```
# person api test requests
# 2020-02 mamund

http://localhost:8181/
http://localhost:8181/list/
http://localhost:8181/filter?status=active
http://localhost:8181/ -X POST -d id=q1w2e3r4&status=pending&email=test@example.org
http://localhost:8181/q1w2e3r4 -X PUT -d givenName=Mike&familyName=Mork&telephone=123-456-7890
http://localhost:8181/status/q1w2e3r4 -X PATCH -d status=active
http://localhost:8181/q1w2e3r4 -X DELETE

# EOF
```



Simple Request Testing (SRTs)

- SRTs are a great way to validate endpoints
- Use **`curl`** to execute simple requests against your API
  - **`curl`** or **`wget`** or any other simple HTTP CLI client works fine
  - Pipe results to a file (**`output.txt`**) and check into source control
- "Eyeball" the results to make sure it works as expected

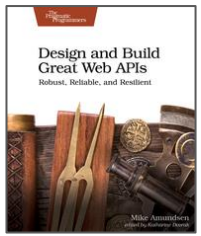



```
##################################################
# run requests
echo
echo start request run...
while IFS= read -r line
do
  if [ ! -z "$line" ] && [ ${line:0:1} != "#" ]
  then
    echo
    echo "$line"
    if [ -z "$outfile" ]
    then
      curl $line
    else
      echo "$line" >> $outfile
      curl --silent --show-error --fail $line >> $outfile
    fi
  fi
done < $infile
```

Simple Request Testing (SRTs)

- SRTs are a great way to validate endpoints
- Use **`curl`** to execute simple requests against your API
- "Eyeball" the results to make sure it works as expected
  - Does the request fail?
  - Does the body "look ok?"

```
http://localhost:8181/23456 -X GET
Time:  1584657412556 : localhost:8181/23456 : GET : {}
{
  "error": [
    {
      "type": "error",
      "title": "SimpleStorage: [api]",
      "detail": "Not Found [23456]",
      "status": "400",
      "instance": "http://localhost:8181/23456"
    }
  ]
}
```



SRT Demo

jam - mca@mamund-ws: ~/Dropbox/Private/Projects

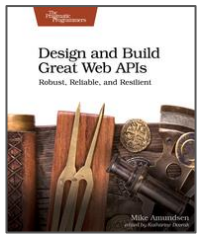
File Edit View Terminal Tabs Help

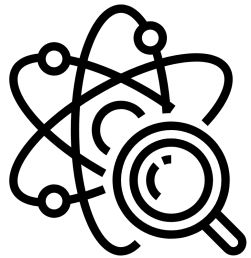
```
#####################################
# run requests
echo
echo start request run...
while IFS= read -r line
do
  if [ ! -z "$line" ] && [ ${line:0:1
  then
    echo
    echo "$line"
    if [ -z "$outfile" ]
    then
      curl $line
    else
      echo "$line" >> $outfile
      curl --silent --show-error --fa
    fi
  fi
done < $infile

#####################################
# all done
echo
echo "job completed."
echo

#####################################
# EOF
```

jam - mca@mamund-ws: ~/Dropbox/Private/Projects/api-starter/tests

File Edit View Terminal Tabs Help

```
# ******************************************************
# person api
# simple test requests (SRTs)
# 2020-02 mamund
# ******************************************************

happy path
http://localhost:8181/
http://localhost:8181/list/
http://localhost:8181/filter?status=active
http://localhost:8181/ -X POST -d id=q1w2e3r4&status=pending&email=test@example.org
http://localhost:8181/q1w2e3r4 -X PUT -d givenName=Mike&familyName=Mork&telephone=123-456-7890
http://localhost:8181/status/q1w2e3r4 -X PATCH -d status=active
http://localhost:8181/q1w2e3r4 -X DELETE

# sad path
# http://localhost:8181/12345 -X DELETE
# http://localhost:8181/ -X POST -d id=12345
# http://localhost:8181/ -X POST -d id=12345&email=sample@example.org
# http://localhost:8181/ -X POST -d id=12345&email=sample@example.org
# http://localhost:8181/12345 -X PUT -d email=updated@example.org&hatSize=12.5
# http://localhost:8181/23456 -X GET

#
# EOF
#
```

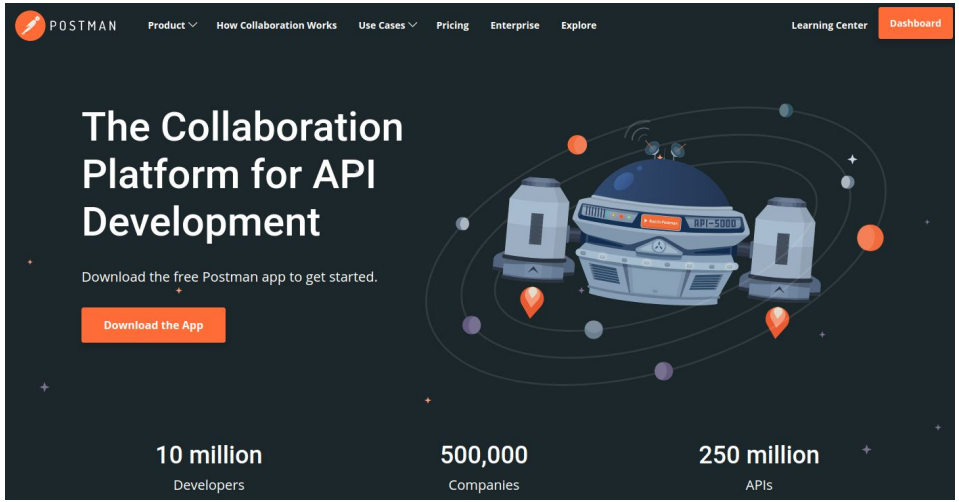


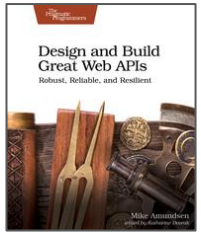


Testing with Postman UI and ChaiJS

Postman UI and ChaiJS

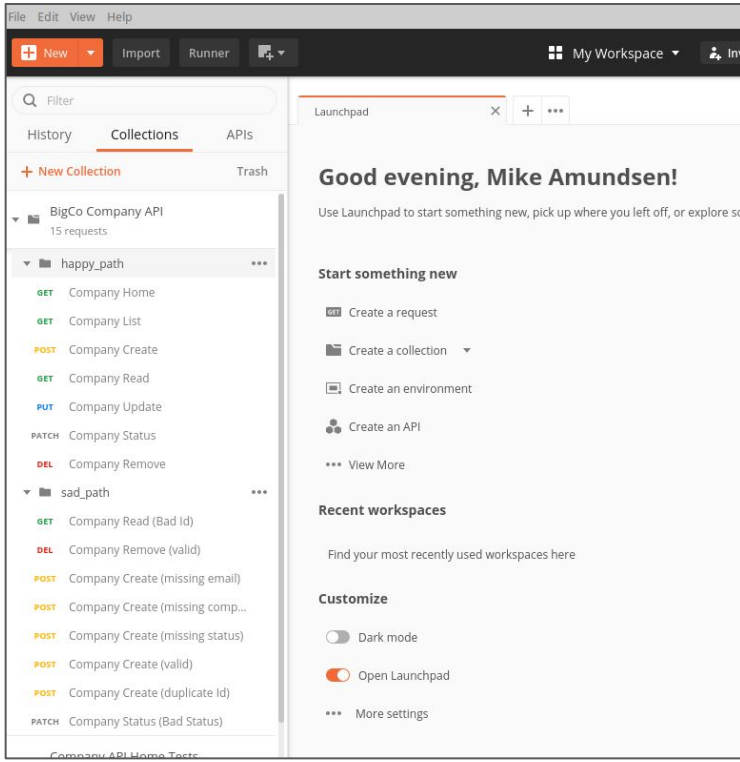
- Postman API platform (2014)
- Collections -> Folders -> Requests
- Embedded libraries for scripting tests
- ChaiJS assertions

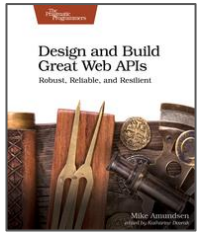




Postman UI and ChaiJS

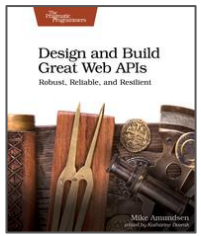
- Postman API platform (2014)
  - Design, build, mock, test, etc.
  - May require a client-side install
- Collections -> Folders -> Requests
- Embedded libraries for scripting tests
- ChaiJS assertions

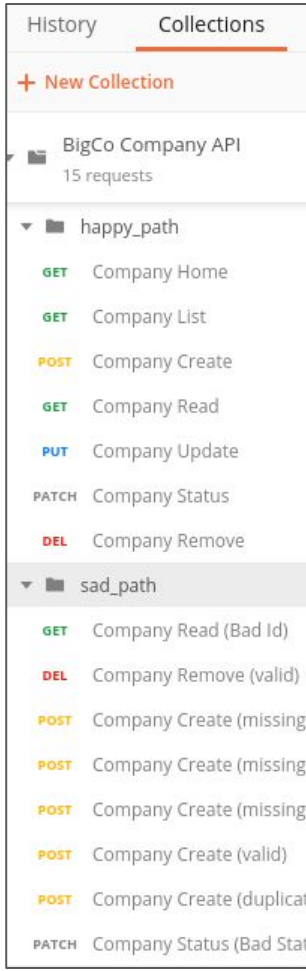




Postman UI and ChaiJS

- Postman API platform (2014)
- Collections -> Folders -> Requests
  - You can import/export collections
  - Supports global, collection, and environment memory sharing
- Embedded libraries for scripting tests
- ChaiJS assertions





Postman UI and ChaiJS

- Postman API platform (2014)
- Collections -> Folders -> Requests
- Embedded libraries for scripting tests
  - 12+ client-side libraries
  - 10+ NodeJS libraries
- ChaiJS assertions

Global functions (pm.*)

require

```
require(moduleName:String):function → *
```

The `require` function allows you to use the sandbox built-in library modu
corresponding documentation.

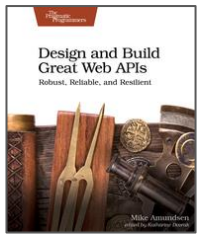
- ajv → v6.6.2
- atob → v2.1.2
- btoa → v1.2.1
- chai → v4.2.0
- cheerio → v0.22.0
- crypto-js → v3.1.9-1
- csv-parse/lib/sync → v1.2.4
- lodash → v4.17.11 (when used with require, the inbuilt `_` object is for
- moment → v2.22.2 (sans locales)
- postman-collection → v3.4.0
- tv4 → v1.3.0
- uuid → (the module loaded is a shim for original module)
- xml2js → v0.4.19

A number of NodeJS modules are also available to use in the sandbox:

- path
- assert
- buffer
- util
- url
- punycode
- querystring
- string-decoder
- stream
- timers
- events

Postman UI and ChaiJS

- Postman API platform (2014)
- Collections -> Folders -> Requests
- Embedded libraries for scripting tests
- ChaiJS assertions
  - BDD-style assertions
  - Fluent interface
  - `should(...), expect(...), assert(...)`



Postman UI Demo

Company Home

GET http://company-atk.herokuapp.com...

Params Authorization Headers Body Pre-request Script

```
/***********************
 * COMPANY HOME
 ***********************/

// PROTOCOL
pm.test('Status is 200', function() {
    pm.expect(pm.response.code).to.equal(200);
});

pm.test('Content-Type header is application/json', function() {
    var hdr = pm.response.headers.get('content-type');
    pm.expect(hdr).to.include('application/json');
});
// STRUCTURE
var body = pm.response.json();

// expect body be valid home object
pm.test('Response body contains a valid home object', function()
    pm.expect(body.home).to.be.an('array');
    coll = body.home;
    coll.forEach(function(item) {
        pm.expect(item).to.have.property('id');
        pm.expect(item).to.have.property('name');
        pm.expect(item).to.have.property('rel');
        pm.expect(item).to.have.property('href');
    });
});
// VALUES
var item = body.home.find(x => x.id = 'list');

pm.test('home.id is set to list', function() {
    pm.expect(item.id).to.include('list');
});
```

Collection Runner | Run Results | My Workspace

19 PASSED | 0 FAILED

Company API Home Tests company-localhost

just now

Run Summary | Export Resu

Iteration 1

GET Company Home Starter http://company-atk.hero... ...e Tests / Company Home Starter 200 OK 33 ms 563 B

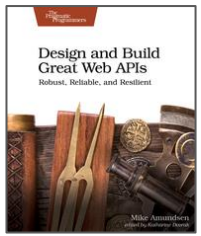
This request does not have any tests.

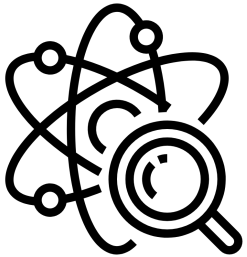
GET Company Home http://company-atk.hero... ...API Home Tests / Company Home 200 OK 22 ms 563 B

- Status is 200
- Content-Type header is application/json
- Response body contains a valid home object
- home.id is set to list
- home.name is set to company
- home.rel contains company and collection

GET Company Home Utils http://company-atk.hero... ...ome Tests / Company Home Utils 200 OK 24 ms 563 B

- Status is 200
- Header content-type contains application/json
- Valid home object
- home has property id set to list
- home has property name set to company



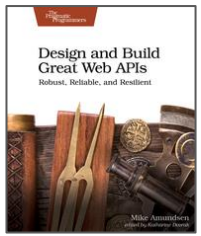


Automate Testing with Newman CLI

Automate Testing with Newman CLI

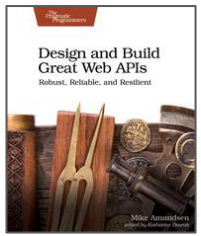
- Newman (2016) is the command-line client for Postman
- Use Postman API for managing collections & environments
- Newman + API = custom test-run scripts
- Add -reporters option for improved output

	executed	failed
iterations	1	0
requests	15	0
test-scripts	30	0
prerequest-scripts	30	0
assertions	163	0
total run duration: 8s		
total data received: 103.87KB (approx)		
average response time: 445ms		



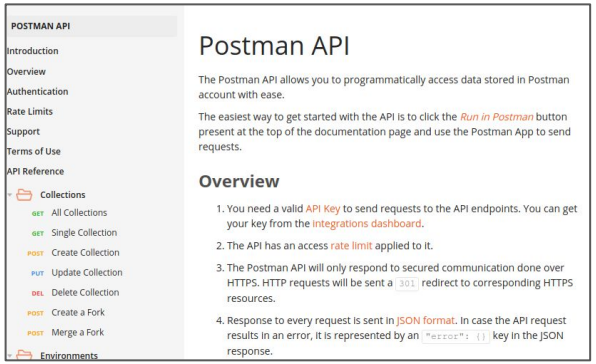
Automate Testing with Newman CLI

- Newman is the command-line client for Postman (2016)
  - `npm install -g newman`
  - `newman run <collection>`
- Postman API for managing collections & environments
- Newman + API = custom test-run scripts
- Add -reporters option for improved output



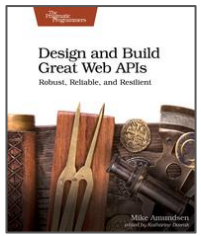
Automate Testing with Newman CLI

- Newman is the command-line client for Postman (2016)
- Postman API for managing collections & environments
  - Generate your API key
  - Use `curl` to make postman API calls for collections & environments
- Newman + API = custom test-run scripts
- Add -reporters option for improved output



Automate Testing with Newman CLI

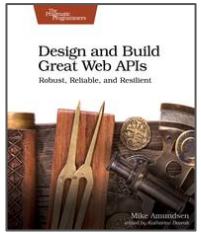
- Newman is the command-line client for Postman (2016)
- Postman API for managing collections & environments
- Newman + API = custom test-run scripts
  - Pull proper collections/environments (API)
  - Run tests and report results (Newman)
  - **`test-run.sh local`**
- Add -reporters option for improved output

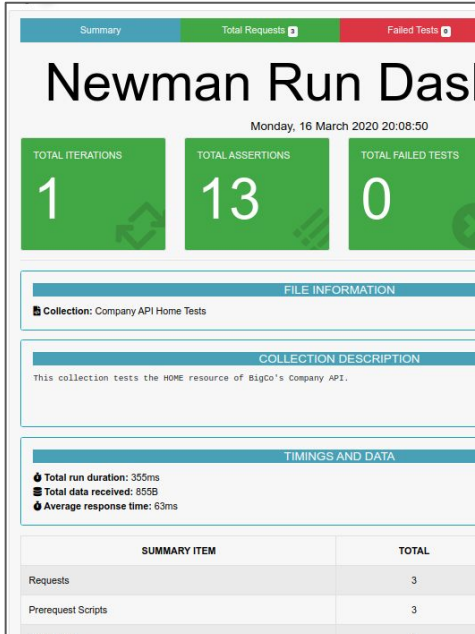



```
# ************************************
# pull collection
echo "Pulling postman data..."
curl -s -X GET $svr/collections/$collid -H "X-Api-Key:$
  -H "Cache-Control:no-cache" -o $testfile
curl -s -X GET $svr/environments/$envid -H "X-Api-Key:$
  -H "Cache-Control:no-cache" -o $envfile

# ************************************
# run the tests
echo "Running tests..."
if [ -z "$outfile" ]
then
  newman run $testfile -e $envfile --bail -r cli,htmlex
else
  newman run $testfile -e $envfile --bail -r cli,htmlex
fi
```

Automate Testing with Newman CLI

- Newman is the command-line client for Postman (2016)
- Postman API for managing collections & environments
- Newman + API = custom test-run scripts
- Add -reporters option for improved output
  - **`npm install -g newman-reporter-htmlextra`**
  - **`newman run <collection> -r htmlextra`**





Newman CLI Demo

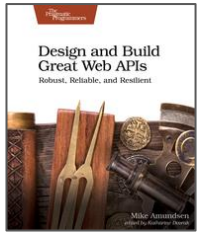
```
# ************************************
# pull collection
echo "Pulling postman data..."
curl -s -X GET $svr/collections/$collid -H "X-Api-Key:$apikey" \
  -H "Cache-Control:no-cache" -o $testfile
curl -s -X GET $svr/environments/$envi
  -H "Cache-Control:no-cache" -o $env

# ************************************
# run the tests
echo "Running tests..."
if [ -z "$outfile" ]
then
  newman run $testfile -e $envfile --
else
  newman run $testfile -e $envfile --
fi
```

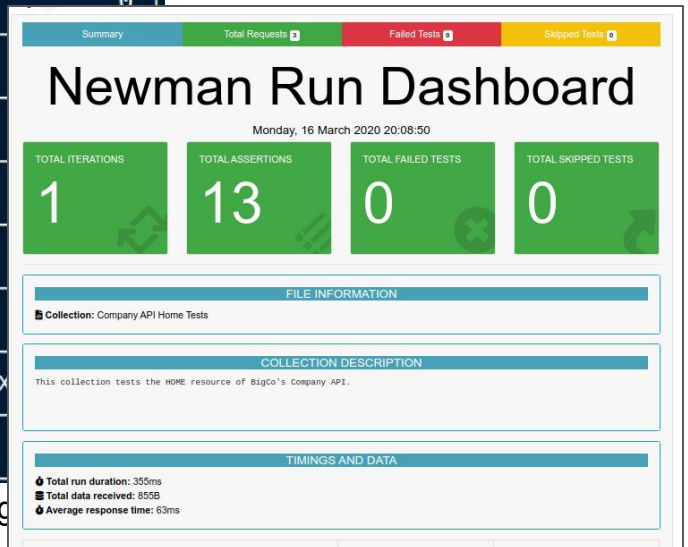
	executed	failed
iterations	1	0
requests	15	
test-scripts	30	
prerequest-scripts	30	
assertions	163	

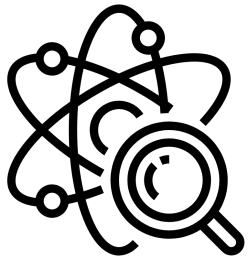
total run duration: 8s

total data received: 103.87KB (approx

average response time: 445ms



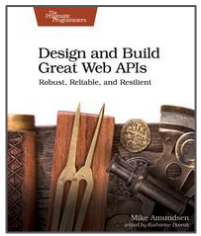




Putting it all together

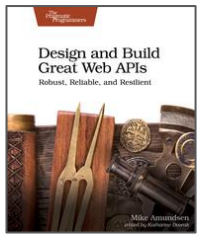
Putting It All Together w/ a "mini-pipeline"

- Build
- Local Test
- Deploy
- Remote Test



Putting It All Together w/ a "mini-pipeline"

- Build
  - `npm run dev`
- Local Test
  - `test-run.sh local`
- Deploy
  - `git push heroku master`
- Remote Test
  - `test-run.sh remote`

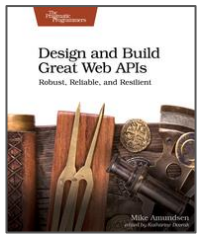


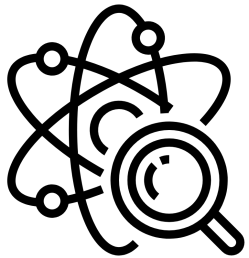
mini-pipeline Demo

```
# ************************************
# pull collection
echo "Pulling postman data..."
curl -s -X GET $svr/collections/$collid -H "X-Api-Key:$apikey" \
  -H "Cache-Control:no-cache" -o $testfile
curl -s -X GET $svr/environments/$envid -H "X-Api-Key:$apikey" \
  -H "Cache-Control:no-cache" -o $envfile

# ************************************
# run the tests
echo "Running tests..."
if [ -z "$outfile" ]
then
  newman run $testfile -e $envfile --
else
  newman run $testfile -e $envfile --
fi
```

	executed	failed
iterations	1	0
requests	15	
test-scripts	30	
prerequest-scripts	30	
assertions	163	
total run duration: 8s		
total data received: 103.87KB (approx		
average response time: 445ms		

```
##############################################
# run requests
echo
echo start request run...
while IFS= read -r line
do
  if [ ! -z "$line" ] && [ ${line:0:1} != "#" ]
  then
    echo
    echo "$line"
    if [ -z "$outfile" ]
    then
      curl $line
    else
      echo "$line" >> $outfile
      curl --silent --show-error --fail $line >> $outfile
    fi
  fi
done < $infile
```

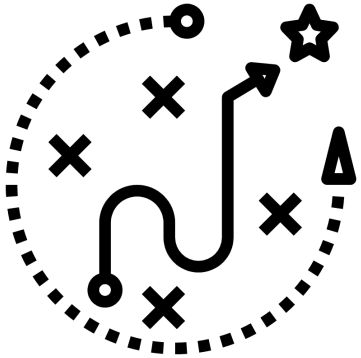


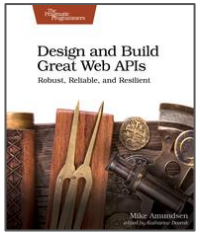


And So...

Goals of API Testing

- Outside-in approach
- Testing the API's behavior
- Happy- and Sad-Path testing

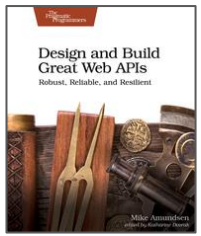




Validate Endpoints with SRTs

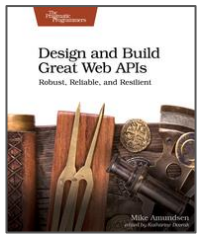
- Focus on URLs and arguments
- Use the **`srt.sh`** script with **`curl`**
- That's really "testing" yet <g>

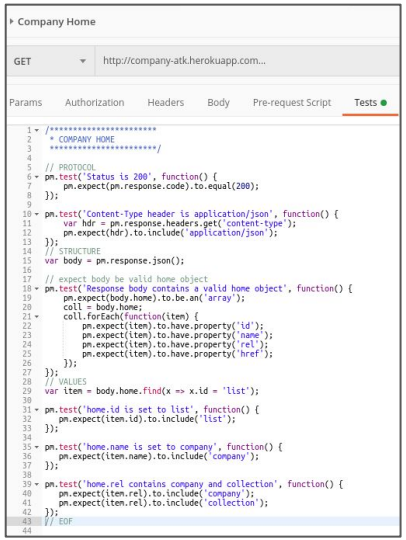
```
jam - mca@mamund-ws: ~/Dropbox/Private/Projects/api-starter/tests
File  Edit  View  Terminal  Tabs  Help
# ********************************************************
# person api
# simple test requests (SRTs)
# 2020-02 mamund
# ********************************************************

happy path
http://localhost:8181/
http://localhost:8181/list/
http://localhost:8181/filter?status=active
http://localhost:8181/ -X POST -d id=q1w2e3r4&status=pending&email=test@example.org
http://localhost:8181/q1w2e3r4 -X PUT -d givenName=Mike&familyName=Mork&telephone=123-456-7890
http://localhost:8181/status/q1w2e3r4 -X PATCH -d status=active
http://localhost:8181/q1w2e3r4 -X DELETE

# sad path
# http://localhost:8181/12345 -X DELETE
# http://localhost:8181/ -X POST -d id=12345
# http://localhost:8181/ -X POST -d id=12345&email=sample@example.org
# http://localhost:8181/ -X POST -d id=12345&email=sample@example.org
# http://localhost:8181/12345 -X PUT -d email=updated@example.org&hatSize=12.5
# http://localhost:8181/23456 -X GET

#
# EOF
#
```



Test Behavior with Postman and ChaiJS

- Create Postman test collections
- Use ChaiJS assertion library BDD-style "outside-in"
- Build up reusable test modules and share via global memory
- Be sure to use both Happy- and Sad-Path tests

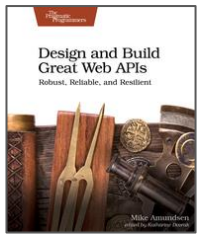


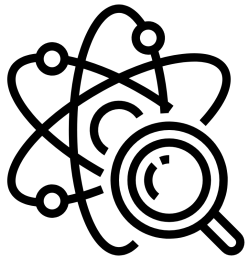


Automate Testing with Newman CLI

- Newman is the Postman CLI client
- Automate testing using **`test-run.sh`**
- Optionally, use **`-r htmlextra`** option for HTML output

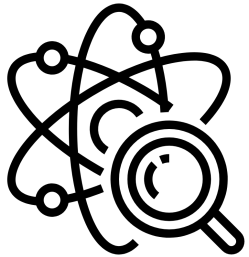
```
# ************************************
# pull collection
echo "Pulling postman data..."
curl -s -X GET $svr/collections/$collid -H "X-Api-Key:$apikey" \
  -H "Cache-Control:no-cache" -o $testfile
curl -s -X GET $svr/environments/$envid -H "X-Api-Key:$apikey" \
  -H "Cache-Control:no-cache" -o $envfile

# ************************************
# run the tests
echo "Running tests..."
if [ -z "$outfile" ]
then
  newman run $testfile -e $envfile --bail -r cli,htmlextra
else
  newman run $testfile -e $envfile --bail -r cli,htmlextra > $outfile
fi
```





That's all there is!

Resources



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

- This talk (slides, links, etc.)
  `g.mamund.com/2021-02-postman-galaxy-testing`

Testing your APIs with Postman and Newman

Mike Amundsen
@mamund
youtube.com/mamund