Web Development

COMP 431 / COMP 531

More Testing and Headless Browsing

Scott E Pollack, PhD

February 25, 2016

Front-End Recap

- HTML and HTML5
- JavaScript and JS Libraries
- Forms and Events
- CSS and Style Frameworks
- MVC and HTML Templates
- Homework Assignment 5 (Front-End App)
  - **Due Thursday 3/10**

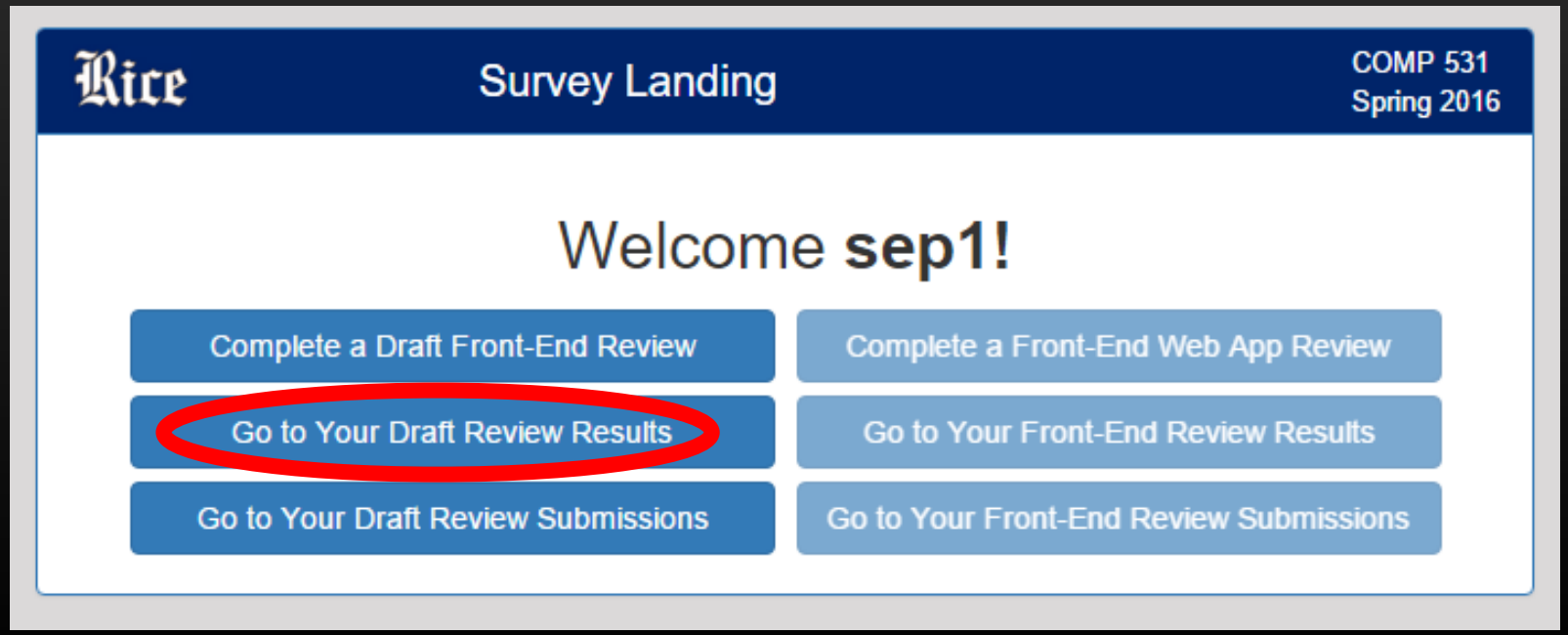
~~Unit Testing~~
~~Angular Services~~
Headless Browsing
SPRING BREAK

PART II
Web Servers
CGI Gateway
Frameworks
Web Hosting

Draft Front-End Reviews *are available!*

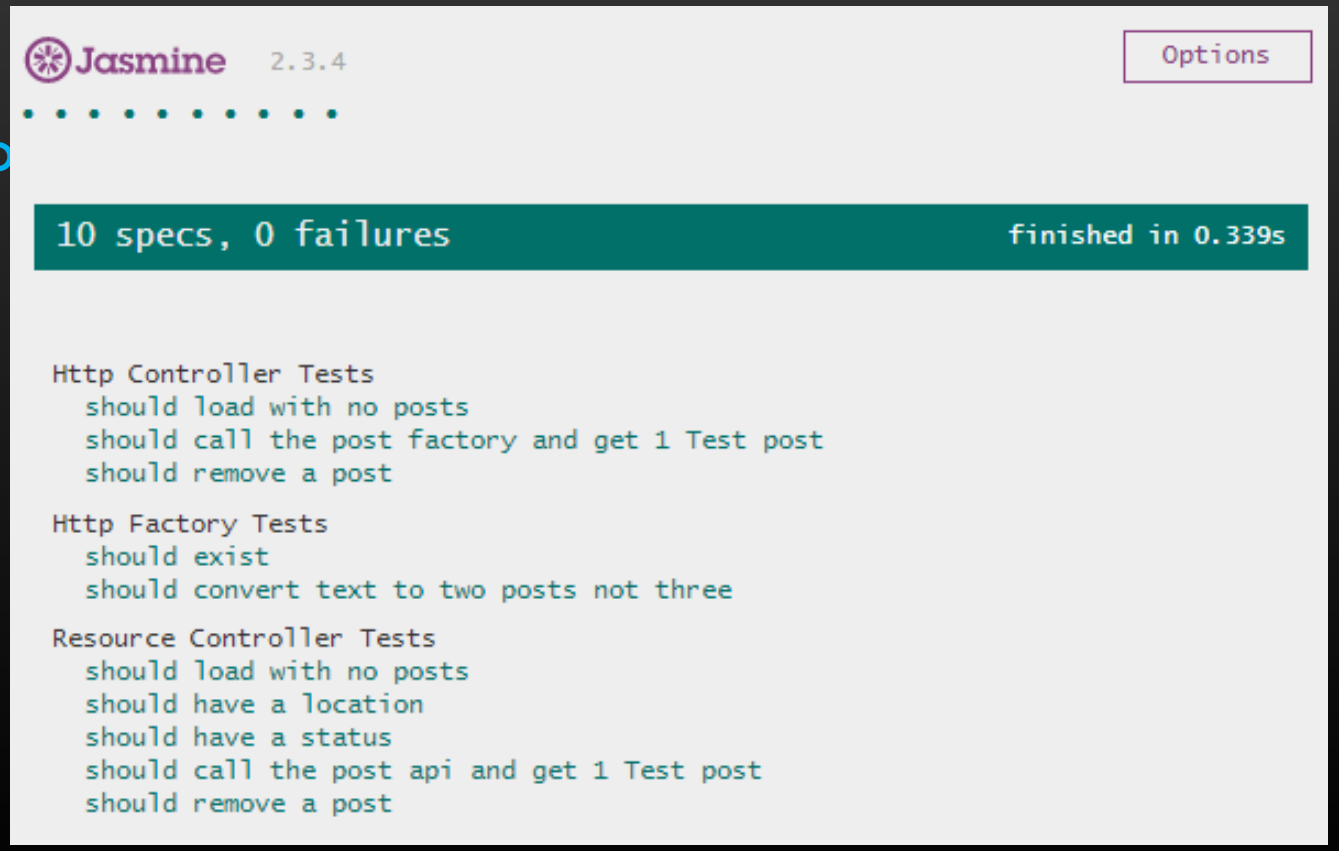
http://webdev-dummy.herokuapp.com/survey

- Go to this address and log in using your netid and 3-word password supplied to you by email for the dummy server



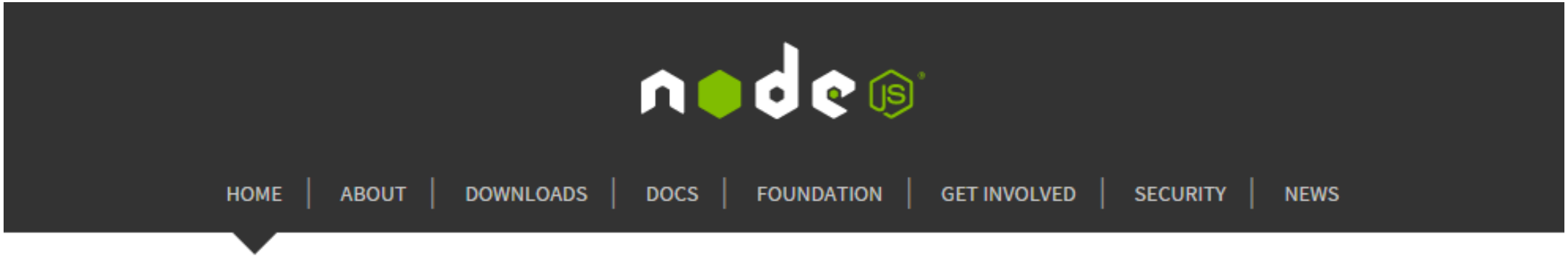
~~Pieces of Testing~~ → Automated Testing Continuous Integration

- Assertion
  - A single condition expected to be true
- Test Case
  - Executes an atomic component o
  - May contain multiple assertions
- Test Suite
  - Collection of Test Cases
- Test Runner
  - Executes the test suite



Install node.js

https://nodejs.org



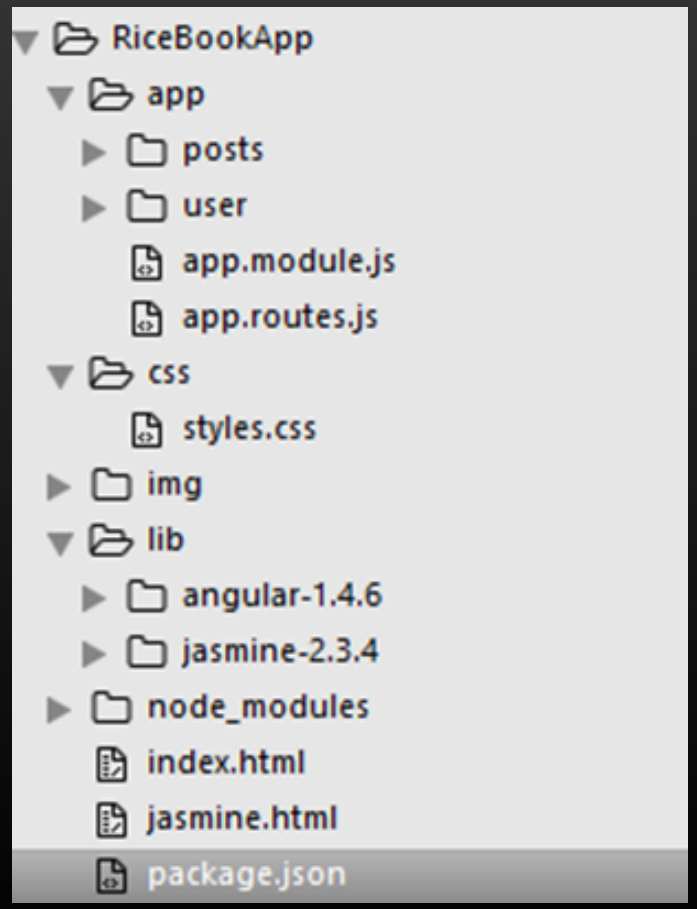
Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

Current Version: v4.1.1

Other Downloads | Changelog | API Docs

Package configuration

- First we need a package.json config file for npm to write to




```
// create a package.json file
# npm init -y

// open package.json
// and edit if you like
```


```
// install Karma using npm
# npm install karma --save-dev

// modules for jasmine and chrome
# npm install karma-jasmine \
     karma-chrome-launcher \
     karma-coverage --save-dev

// global karma in your path
# npm install karma-cli -g
```

https://www.clear.rice.edu/comp431/sample/RiceBookApp/karma.conf.js

Configure Karma to Run

```
// get angular locally for karma
# npm install angular angular-mocks --save-dev
```

```
module.exports = function(config) {
  config.set({
    basePath: '',
    frameworks: ['jasmine'],
    files: [
        'node_modules/angular/angular.js',
        'node_modules/angular-mocks/angular-mocks.js',
        'app/**/*.js'
    ],
    browsers: ['Chrome'],
    singleRun: false,
    autoWatch: true,
    reporters: ['progress', 'coverage'],
    preprocessors: { 'app/**/*.js': ['coverage'] }
  })
```

```
# karma start karma.conf.js
```

What did Karma do?

```
27 09 2015 21:44:21.127:WARN [karma]: No captured browser, open http://localhost:9876/
27 09 2015 21:44:21.158:INFO [karma]: Karma v0.13.10 server started at http://localhost:9876/
27 09 2015 21:44:21.221:INFO [launcher]: Starting browser Chrome
27 09 2015 21:44:29.937:INFO [Chrome 45.0.2454 (Windows 8.1 0.0.0)]: Connected on socket Z_nU9i_rt8KCs7
QCAAAA with id 86007040
Chrome 45.0.2454 (Windows 8.1 0.0.0): Executed 5 of 5 SUCCESS (0.164 secs / 0.121 secs)
```

- Opened a (captured) Browser
- Used the browser to execute code
  - Runs all the tests
- Collected results from the browser
- Presents the test results in the terminal

Test Coverage

- We turned on code coverage
- Navigate to `file:///${webapp}/coverage/`

Code coverage report for **All files**

Statements: **100%** (49 / 49) Branches: **50%** (1 / 2) Functions: **100%** (16 / 16) Lines: **100%** (49 / 49) Ignored: none

File	Statements		Branches		Functions		Lines	
app/	100%	(2 / 2)	100%	(0 / 0)	100%	(1 / 1)	100%	(2 / 2)
app/posts/	100%	(35 / 35)	50%	(1 / 2)	100%	(10 / 10)	100%	(35 / 35)
app/user/	100%	(12 / 12)	100%	(0 / 0)	100%	(5 / 5)	100%	(12 / 12)

Code coverage report for **app\posts\posts.js**

Statements: **100%** (18 / 18) Branches: **50%** (1 / 2) Functions: **100%** (4 / 4) Lines: **100%** (18 / 18) Ignored: none

All files » app/posts/ » posts.js

```
4          .controller('PostCtrl', PostCtrl);
5
6     1  function PostCtrl() {
7     4    var vm = this;
8
9     4    vm.posts = [
10           {'id':1, 'title':'the first', 'body':'message' },
11           {'id':2, 'title':'the second', 'body':'lorem ipsum'},
12           {'id':3, 'title':'the third', 'body':'e plurbus unum'},
13         ]
14
15    4    vm.removePost = function(postId) {
16    1      var index = -1;
17    1      var len = vm.posts.length;
18    1      for (var ii = 0; ii < len; ++ii) {
19    1        E if (vm.posts[ii].id === postId) {
20    1          index = ii;
21    1          break;
22           }
23         }
24    1      vm.posts.splice(index, 1)
25         }
26
```

Demo

- Add a new function

End-to-End Testing

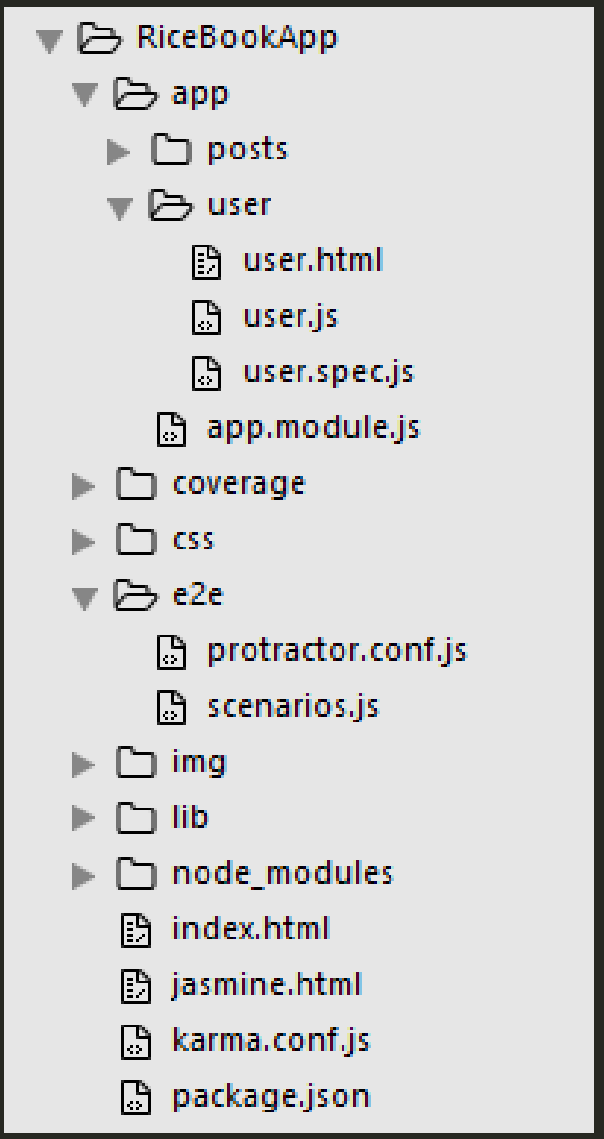
- Karma launched Chrome
- Karma can launch other browsers too!
  - and run all in parallel
- But Karma only executes "unit tests"
  - i.e., JavaScript only, no DOM

- Integration, or End-to-End tests, require a full browsing experience
  - We need a DOM to manipulate


```
// install protractor
# npm install protractor -g
// for node v0.12.7: npm install protractor@2.5.1 -g
// comes with selenium
// which can run multiple browsers
# webdriver-manager update
// but we only need one.  “update” gets that for us
# webdriver-manager start // starts multiple browsers
```

https://www.clear.rice.edu/comp431/sample/RiceBookApp/e2e/protractor.conf.js

Protractor direct connect



scenarios.js | protractor.conf.js

```
exports.config = {
    allScriptsTimeout: 11000,
    specs: [ 'scenarios.js' ],
    capabilities: {
        'browserName': 'chrome'
    },

    directConnect: true,
    baseUrl: 'http://localhost:8080/',

    framework: 'jasmine2',
    jasmineNodeOpts: {
        defaultTimeoutInterval: 30000
    }
}
```

We need a LIVE web server

python -m SimpleHTTPServer 8080

python3 -m http.server 8080

https://www.clear.rice.edu/comp431/sample/RiceBookApp/e2e/scenarios.js

Protractor End-to-End Test

```
describe('RiceBookApp', function() {
    'use strict'

    beforeEach(function() {
        browser.get('/index.html')
    })

    it('should work and have header text', function() {
        expect(element.all(by.css('h2')).first().getText()).toMatch("This is RiceBook")
    })

    it('should delete a post', function() {
        expect(element.all(by.css('[value="Delete"]')).count()).toEqual(3)
        element.all(by.css('[value="Delete"]')).first().click();
        expect(element.all(by.css('[value="Delete"]')).count()).toEqual(2)
    })
```

Run Protractor

DEMO

```
# protractor e2e/protractor.conf.js
Using ChromeDriver directly...
[launcher] Running 1 instances of WebDriver
Started
.....


5 specs, 0 failures
Finished in 11.187 seconds
[launcher] 0 instance(s) of WebDriver still running
[launcher] chrome #1 passed
```

Headless Browsing

"A **headless browser** is a web browser without a graphical user interface" – Wikipedia

What did we just do with Protractor?

We controlled a browser programmatically!

… why do we need the GUI?

npm is your new best friend…

```
// install PhantomJS
# npm install phantomjs -g

// but why stop there?

// install CasperJS
# npm install ‘casperjs@1.1.0-beta3’ -g
```

Time to haunt


```
var casper = require('casper').create({
    viewportSize: {width: 1024, height: 768}
})

casper.start('https://www.google.com', function() {
    this.echo(this.getTitle())
})

casper.run()
```

```
# casperjs ./haunt.js
Google
```

Assignments: *a look ahead…*

HW5: Frontend (due 3/10)

- Angularization
- Unit tests with Coverage
- Login / Logout
- Posts and search
- Status headline

HW6: Draft Backend

- Finalize Frontend
  - Upload images
  - Edit profile
  - Followers
  - End-to-End Tests
- Draft Backend
  - /status and /post

Draft Back-End

due Thu 03/24 after class by 2 AM
Turnin repo: **COMP431-S16:hw6-draftback**

End-to-End Tests

Your end-to-end tests will run against your web app running on your local python server. Here are the tasks for your end-to-end test

- Register a new user
- Log in as your test user [Note: the dummy server has special logic for these test users]
- Create a new post and validate the post appears in the feed
- Update the status headline and verify the change
- Count the number of followed users
- Add the user "Follower" to the list of followed users and verify the count increases by one
- Remove the user "Follower" from the list of followed users and verify the count decreases by one
- Search for "Only One Post Like This" and verify only one post shows, and verify the author
- Navigate to the profile view and verify the page is loaded
- Update the user's email and verify
- Update the user's zipcode and verify
- Update the user's password, verify a "will not change" message is returned

Include a JUnitXML report of your end-to-end tests: `e2e-results/junitresults.xml`

In-Class Exercise: End-to-End Testing

Install Protractor

Download https://www.clear.rice.edu/comp431/sample/e2e-test.zip

```
# unzip e2e-test.zip
# cd e2e-test
# npm install
# npm test
// all of the tests run but are missing correct implementation
// Supply the implementation of the two tests:
      login
      validate username
      Validate status message
      update status message
```

***Turnin* scenarios.js *to* COMP431-S16:inclass-14**