Web Development

COMP 431 / COMP 531

End-to-End Testing

Scott E Pollack, PhD

October 6, 2016

Frontend Recap

- HTML and HTML5, Storage, Canvas
- JavaScript and Scope
- Forms, CSS, Events
- jQuery, AJAX, and fetch
- Modern JS
- MVC, React and Redux   => Testing!

~~Testing~~
Mocking
End-To-End
FALL BREAK

Homework Assignment 4
(JavaScript Game)
Due TONIGHT by 2AM

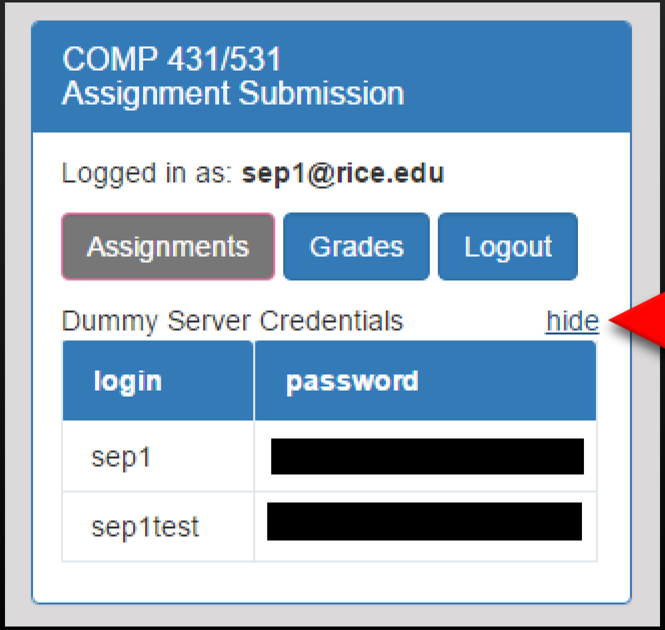
Homework Assignment 5
(Front-End App)
Due Tuesday 10/18

HW 5 Front-End Web Application

- TDD = *Write tests first!*
- Implement:
  - User login / logout – routing from landing to main to profile views
  - Status headline update
  - Filter the posts by author/body (not id)
  - Add a post
  - Edit a post
  - Comment on a post
  - Edit a comment
- Using the Dummy Server

COMP 431/531 Dummy Data Server

- We are writing separate frontend and backend applications.
  Why?
  https://www.clear.rice.edu/comp431/data/api.html

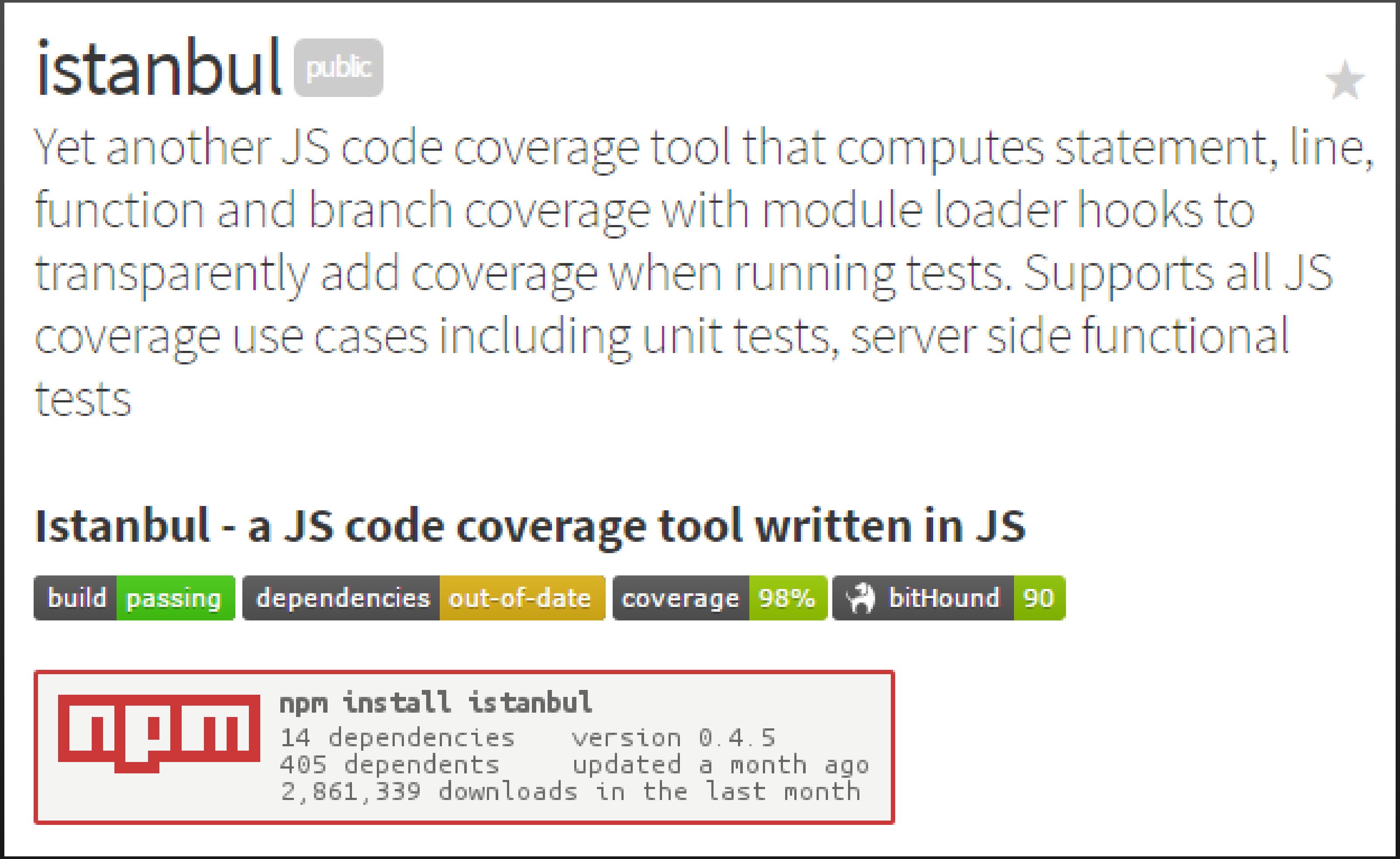


COMP 431/531
Assignment Submission

Logged in as: **sep1@rice.edu**

Dummy Server Credentials hide

login	password
sep1	[redacted]
sep1test	[redacted]


istanbul public

Yet another JS code coverage tool that computes statement, line, function and branch coverage with module loader hooks to transparently add coverage when running tests. Supports all JS coverage use cases including unit tests, server side functional tests

Istanbul - a JS code coverage tool written in JS

```
npm install istanbul
14 dependencies     version 0.4.5
405 dependents      updated a month ago
2,861,339 downloads in the last month
```

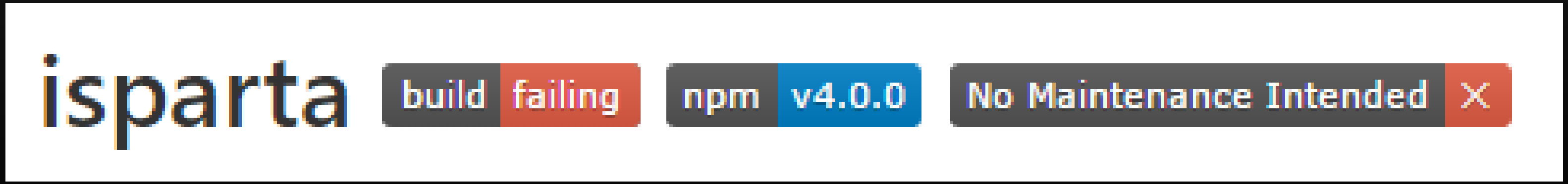
- Instruments code
- Generates coverage report

Two Approaches

Istanbul with mocha

- ES6 requires isparta

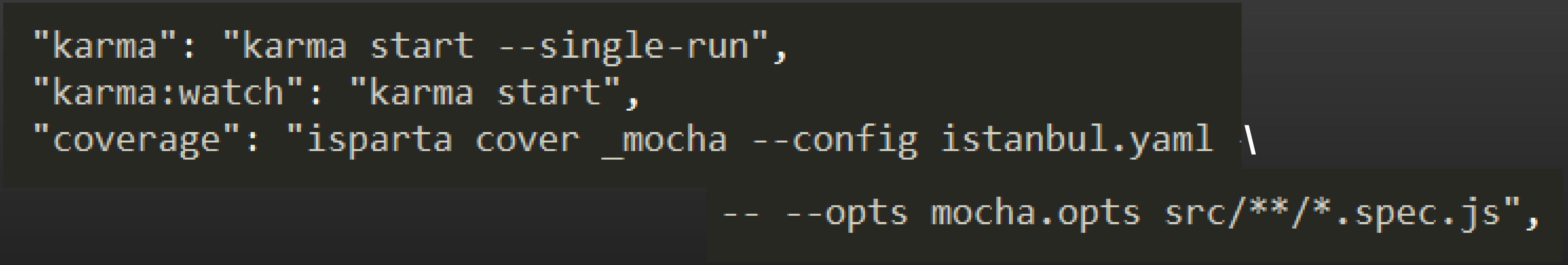
- isparta is no longer supported


isparta build failing npm v4.0.0 No Maintenance Intended X

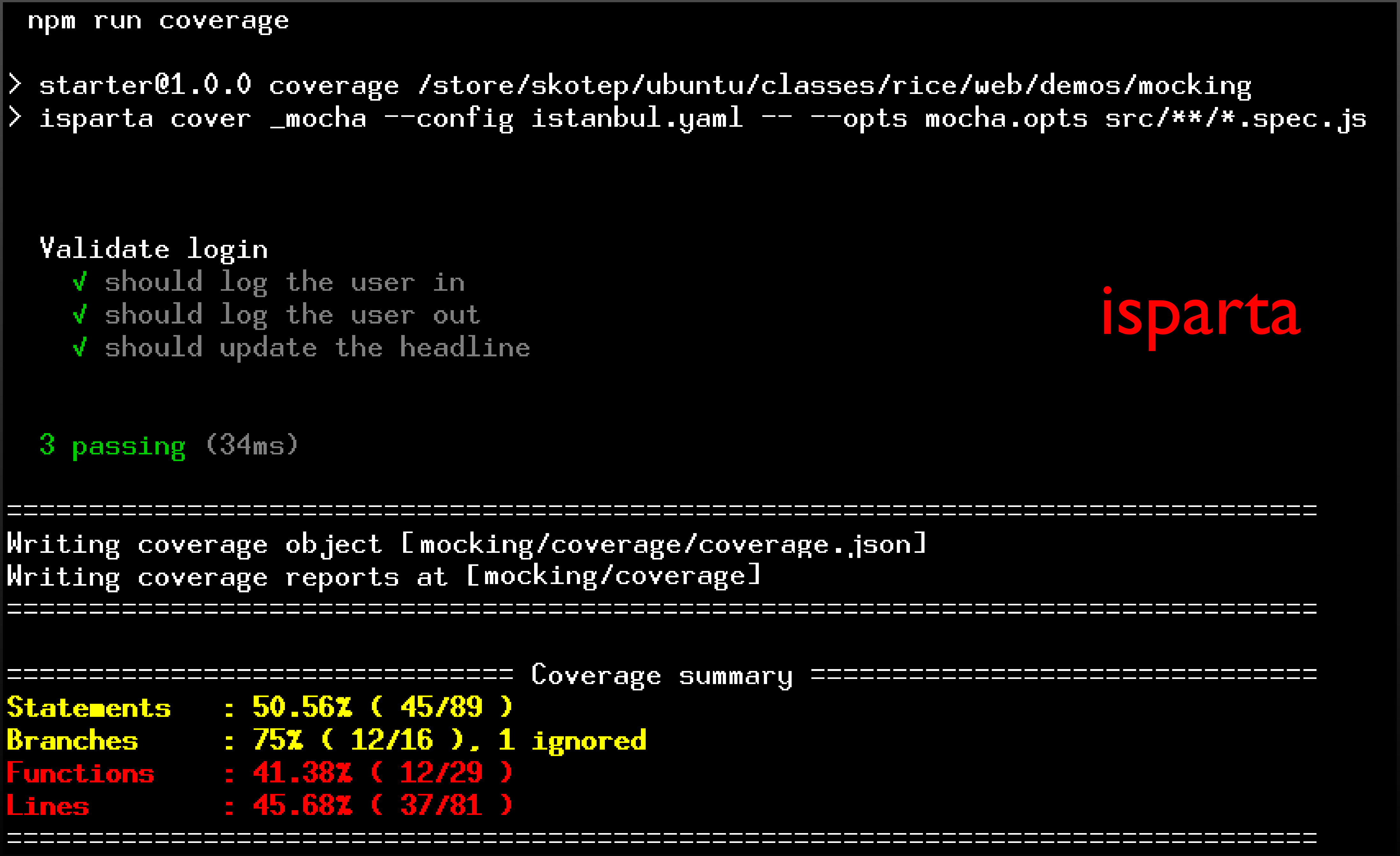
Karma with mocha

- Karma is a test runner
- Launches a browser
- Can use PhantomJS (more later)

npm scripts



```
"karma": "karma start --single-run",
"karma:watch": "karma start",
"coverage": "isparta cover _mocha --config istanbul.yaml \
                                -- --opts mocha.opts src/**/*.spec.js",
```

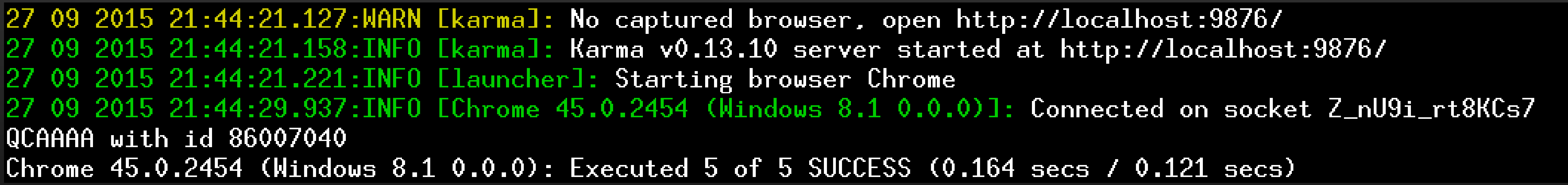


```
 npm run coverage

> starter@1.0.0 coverage /store/skotep/ubuntu/classes/rice/web/demos/mocking
> isparta cover _mocha --config istanbul.yaml -- --opts mocha.opts src/**/*.spec.js



  Validate login
    √ should log the user in
    √ should log the user out
    √ should update the headline


  3 passing (34ms)

=============================================================================
Writing coverage object [mocking/coverage/coverage.json]
Writing coverage reports at [mocking/coverage]
=============================================================================

=============================== Coverage summary ===============================
Statements   : 50.56% ( 45/89 )
Branches     : 75% ( 12/16 ), 1 ignored
Functions    : 41.38% ( 12/29 )
Lines        : 45.68% ( 37/81 )
=============================================================================
```

isparta

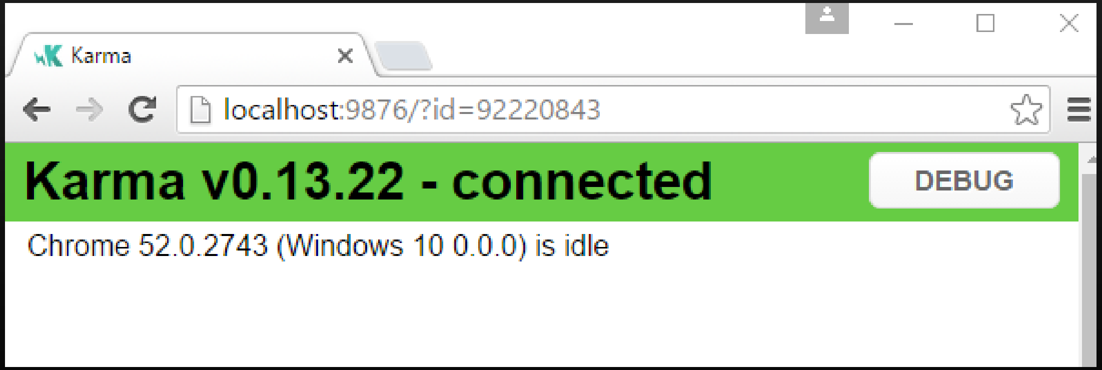
- Karma is a test runner used to run tests in a browser environment

npm run karma



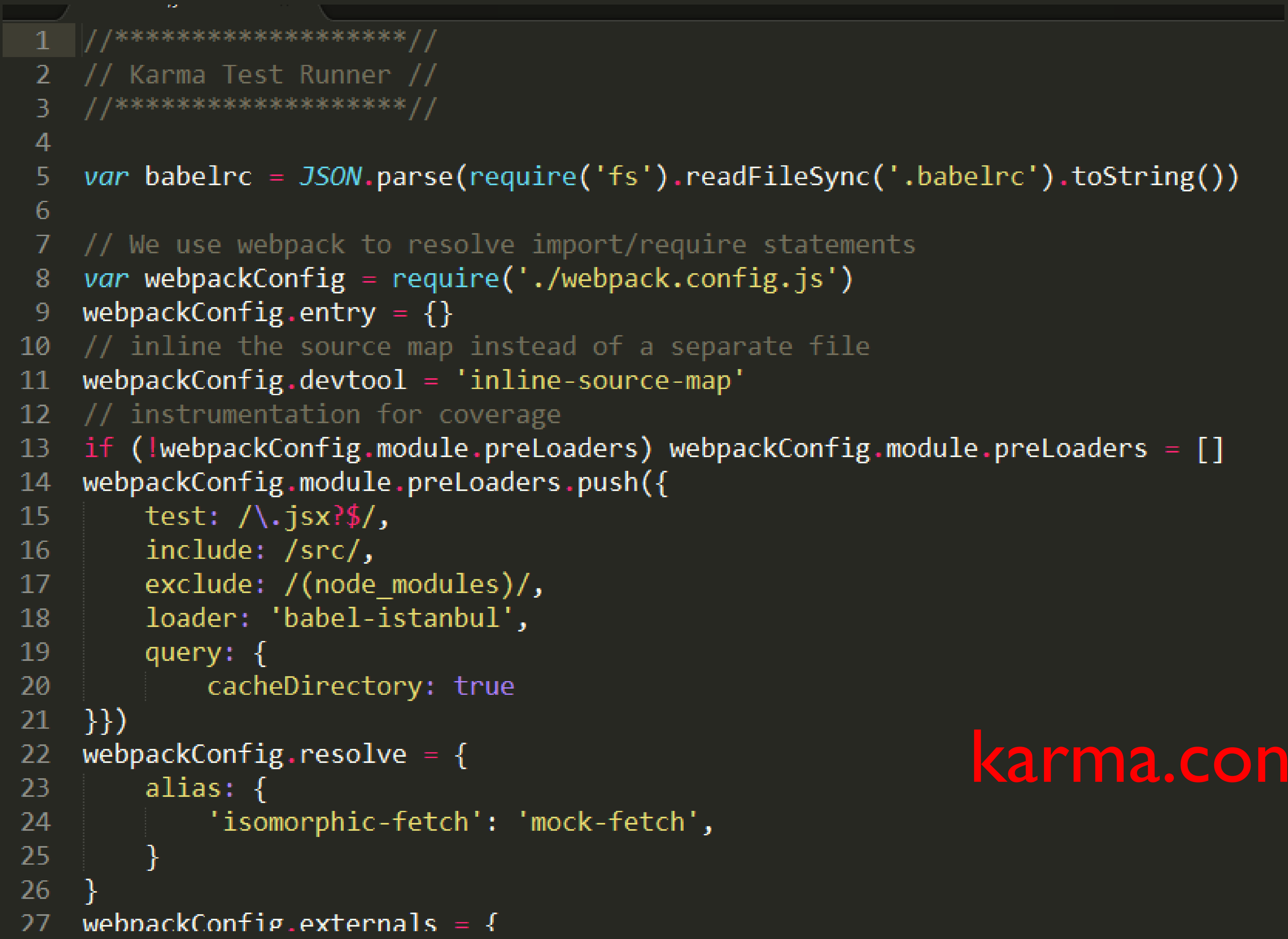
```
27 09 2015 21:44:21.127:WARN [karma]: No captured browser, open http://localhost:9876/
27 09 2015 21:44:21.158:INFO [karma]: Karma v0.13.10 server started at http://localhost:9876/
27 09 2015 21:44:21.221:INFO [launcher]: Starting browser Chrome
27 09 2015 21:44:29.937:INFO [Chrome 45.0.2454 (Windows 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
- Use the browser to execute code
  - Runs all the tests
- Collected results from the browser
- Presents the test results in the terminal



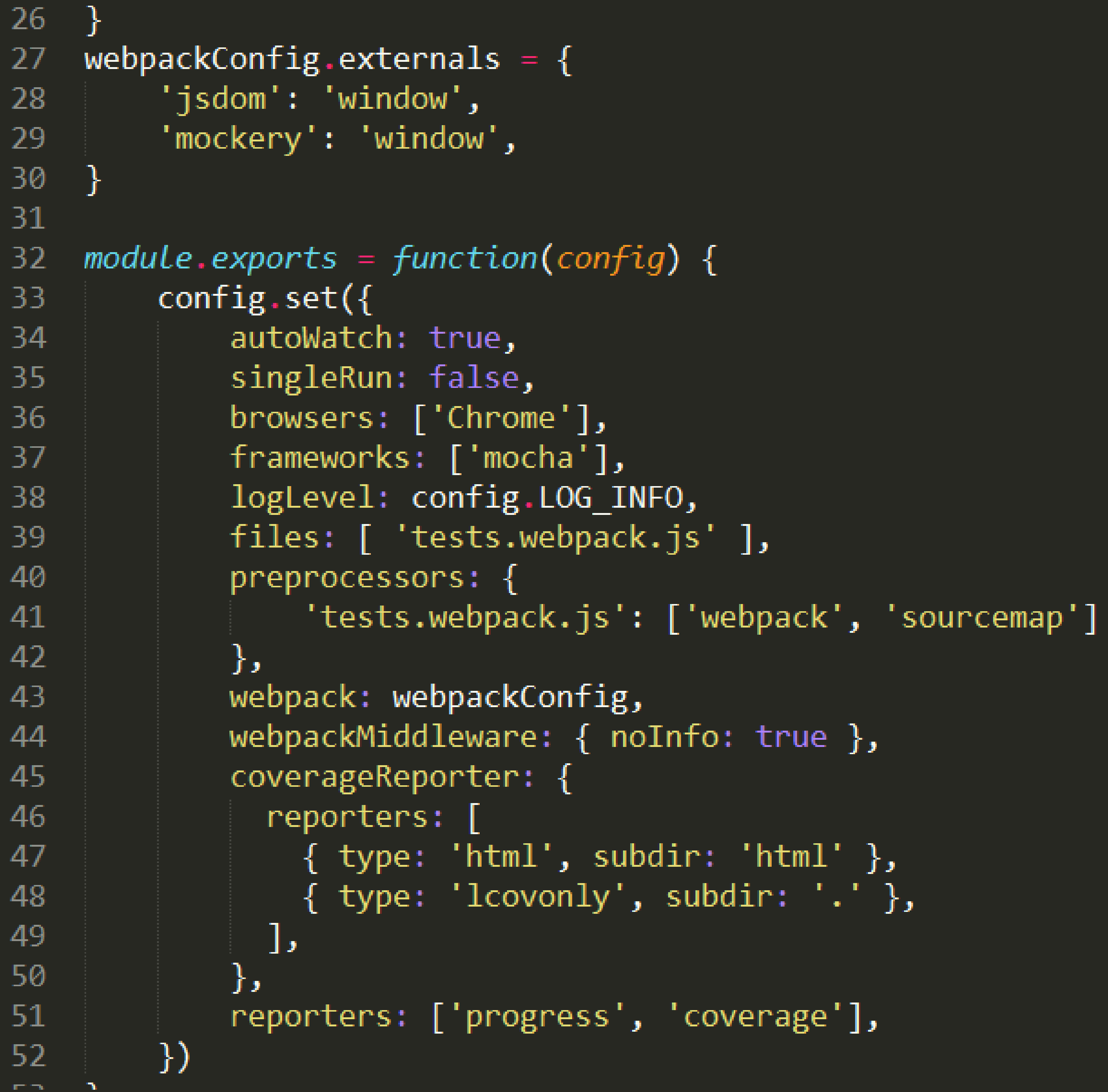
Karma v0.13.22 - connected

Chrome 52.0.2743 (Windows 10 0.0.0) is idle



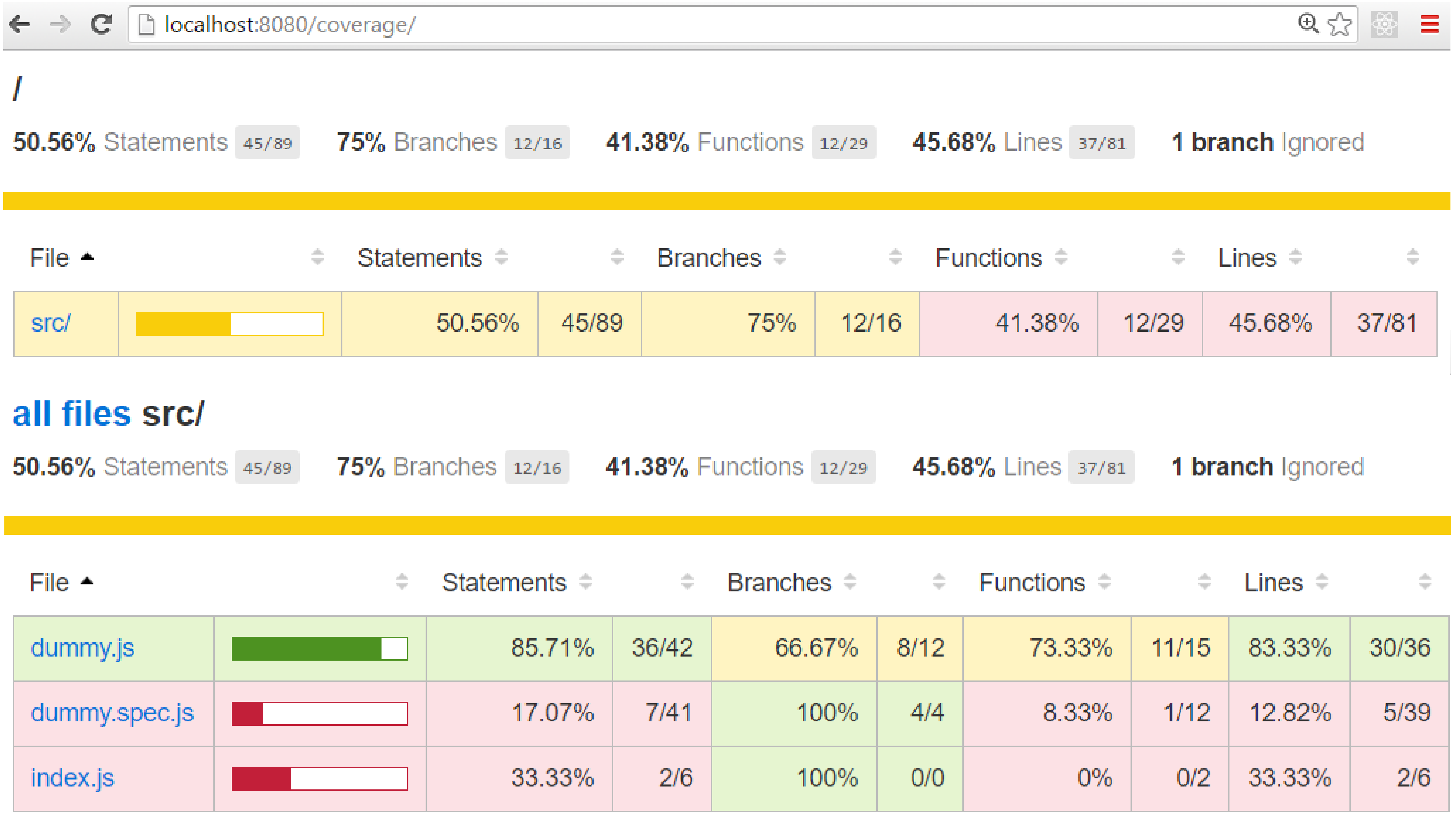
```
//********************//
// Karma Test Runner //
//********************//

var babelrc = JSON.parse(require('fs').readFileSync('.babelrc').toString())

// We use webpack to resolve import/require statements
var webpackConfig = require('./webpack.config.js')
webpackConfig.entry = {}
// inline the source map instead of a separate file
webpackConfig.devtool = 'inline-source-map'
// instrumentation for coverage
if (!webpackConfig.module.preLoaders) webpackConfig.module.preLoaders = []
webpackConfig.module.preLoaders.push({
    test: /\.jsx?$/,
    include: /src/,
    exclude: /(node_modules)/,
    loader: 'babel-istanbul',
    query: {
        cacheDirectory: true
}})
webpackConfig.resolve = {
    alias: {
        'isomorphic-fetch': 'mock-fetch',
    }
}
webpackConfig.externals = {
```

karma.conf.js



```
}
webpackConfig.externals = {
    'jsdom': 'window',
    'mockery': 'window',
}

module.exports = function(config) {
    config.set({
        autoWatch: true,
        singleRun: false,
        browsers: ['Chrome'],
        frameworks: ['mocha'],
        logLevel: config.LOG_INFO,
        files: [ 'tests.webpack.js' ],
        preprocessors: {
            'tests.webpack.js': ['webpack', 'sourcemap']
        },
        webpack: webpackConfig,
        webpackMiddleware: { noInfo: true },
        coverageReporter: {
          reporters: [
            { type: 'html', subdir: 'html' },
            { type: 'lcovonly', subdir: '.' },
          ],
        },
        reporters: ['progress', 'coverage'],
    })
```

karma.conf.js


localhost:8080/coverage/

/

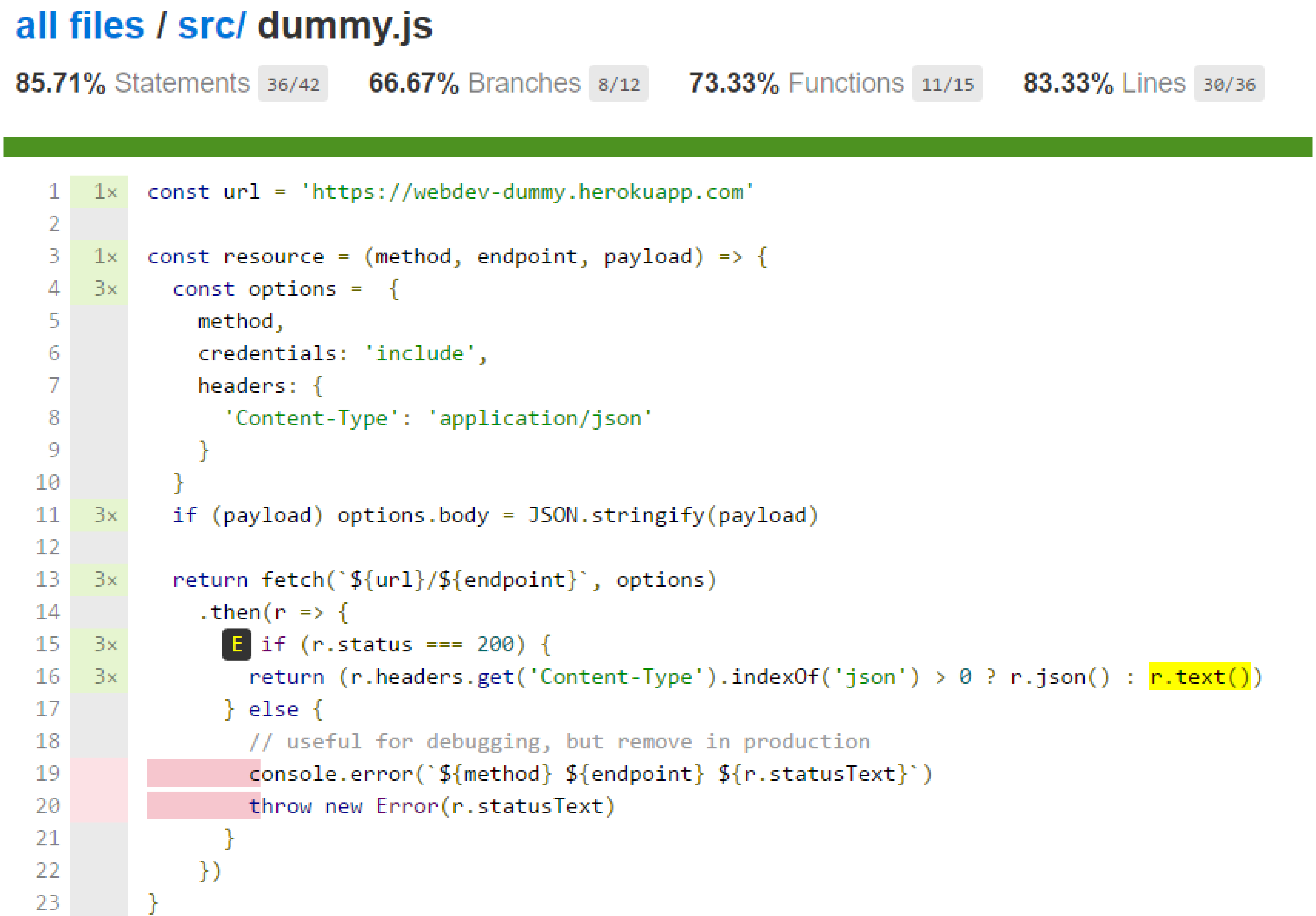
50.56% Statements 45/89 **75%** Branches 12/16 **41.38%** Functions 12/29 **45.68%** Lines 37/81 **1 branch** Ignored

File		Statements		Branches		Functions		Lines	
src/		50.56%	45/89	75%	12/16	41.38%	12/29	45.68%	37/81

all files src/

50.56% Statements 45/89 **75%** Branches 12/16 **41.38%** Functions 12/29 **45.68%** Lines 37/81 **1 branch** Ignored

File		Statements		Branches		Functions		Lines	
dummy.js		85.71%	36/42	66.67%	8/12	73.33%	11/15	83.33%	30/36
dummy.spec.js		17.07%	7/41	100%	4/4	8.33%	1/12	12.82%	5/39
index.js		33.33%	2/6	100%	0/0	0%	0/2	33.33%	2/6


all files / src/ dummy.js

85.71% Statements 36/42 66.67% Branches 8/12 73.33% Functions 11/15 83.33% Lines 30/36

```
1x  const url = 'https://webdev-dummy.herokuapp.com'

1x  const resource = (method, endpoint, payload) => {
3x    const options =  {
        method,
        credentials: 'include',
        headers: {
          'Content-Type': 'application/json'
        }
      }
3x    if (payload) options.body = JSON.stringify(payload)

3x    return fetch(`${url}/${endpoint}`, options)
        .then(r => {
3x        E if (r.status === 200) {
3x          return (r.headers.get('Content-Type').indexOf('json') > 0 ? r.json() : r.text())
          } else {
            // useful for debugging, but remove in production
            console.error(`${method} ${endpoint} ${r.statusText}`)
            throw new Error(r.statusText)
          }
        })
    }
```

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 executes “unit tests”

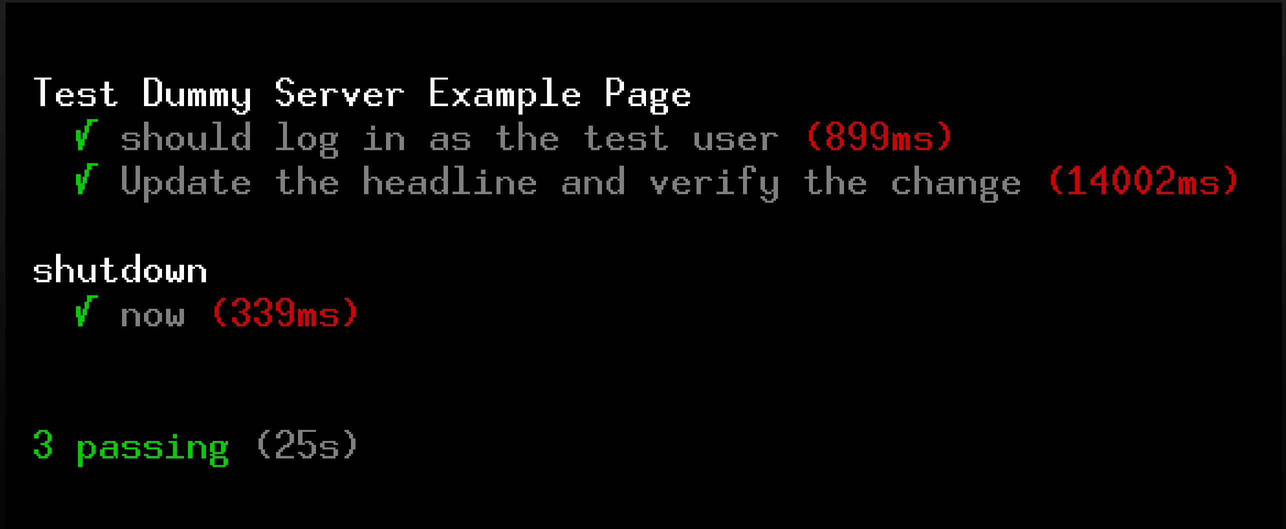
- Integration, or End-to-End tests, require a full browsing experience

  We want a real browser and drive that browser, imitating a user

SeleniumHQ
Browser Automation

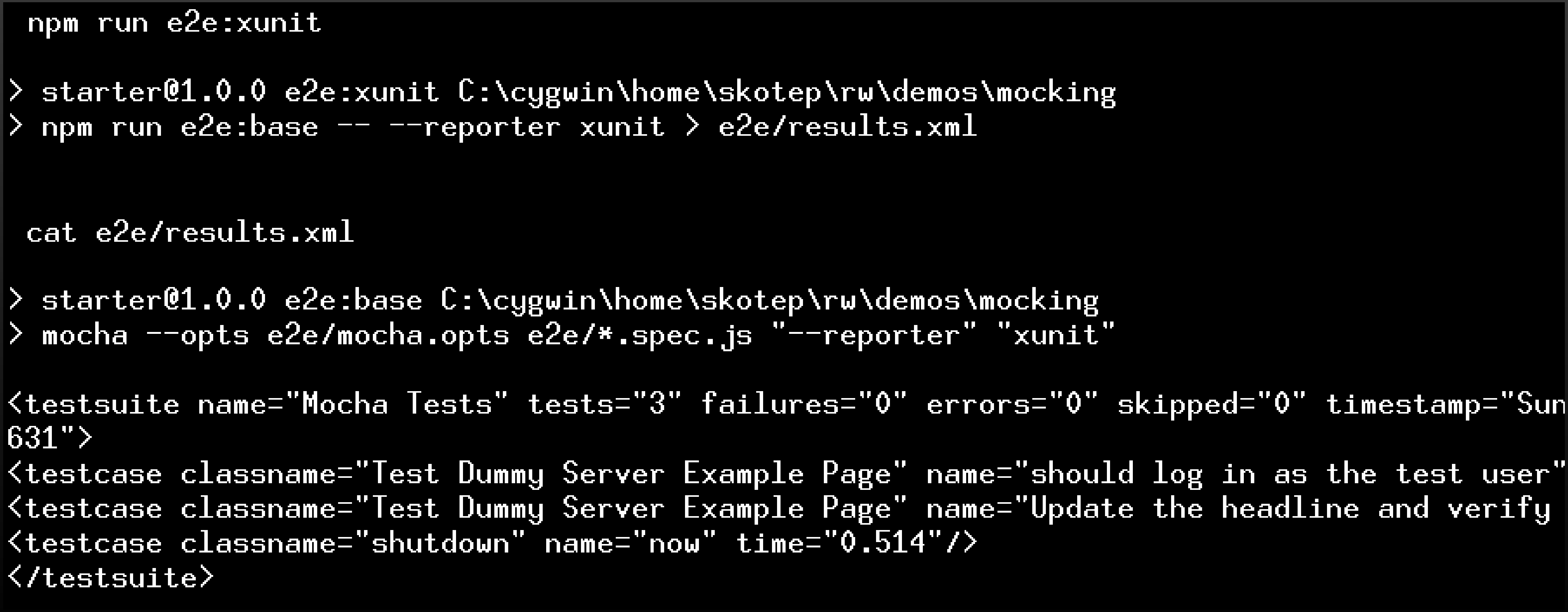
You’ll need to download the ***chromedriver***
The driver needs to be in your path to execute.

```
npm install -g webdriver-manager
webdriver-manager update --chrome –standalone
which chromedriver

npm start &
npm run e2e
```



```
Test Dummy Server Example Page
  √ should log in as the test user (899ms)
  √ Update the headline and verify the change (14002ms)

shutdown
  √ now (339ms)


3 passing (25s)
```

End-to-End Test Report



```
 npm run e2e:xunit

> starter@1.0.0 e2e:xunit C:\cygwin\home\skotep\rw\demos\mocking
> npm run e2e:base -- --reporter xunit > e2e/results.xml


 cat e2e/results.xml

> starter@1.0.0 e2e:base C:\cygwin\home\skotep\rw\demos\mocking
> mocha --opts e2e/mocha.opts e2e/*.spec.js "--reporter" "xunit"

<testsuite name="Mocha Tests" tests="3" failures="0" errors="0" skipped="0" timestamp="Sun
631">
<testcase classname="Test Dummy Server Example Page" name="should log in as the test user"
<testcase classname="Test Dummy Server Example Page" name="Update the headline and verify
<testcase classname="shutdown" name="now" time="0.514"/>
</testsuite>
```

Headless Browsing

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

What did we just do with Selenium?

We controlled a browser programmatically!

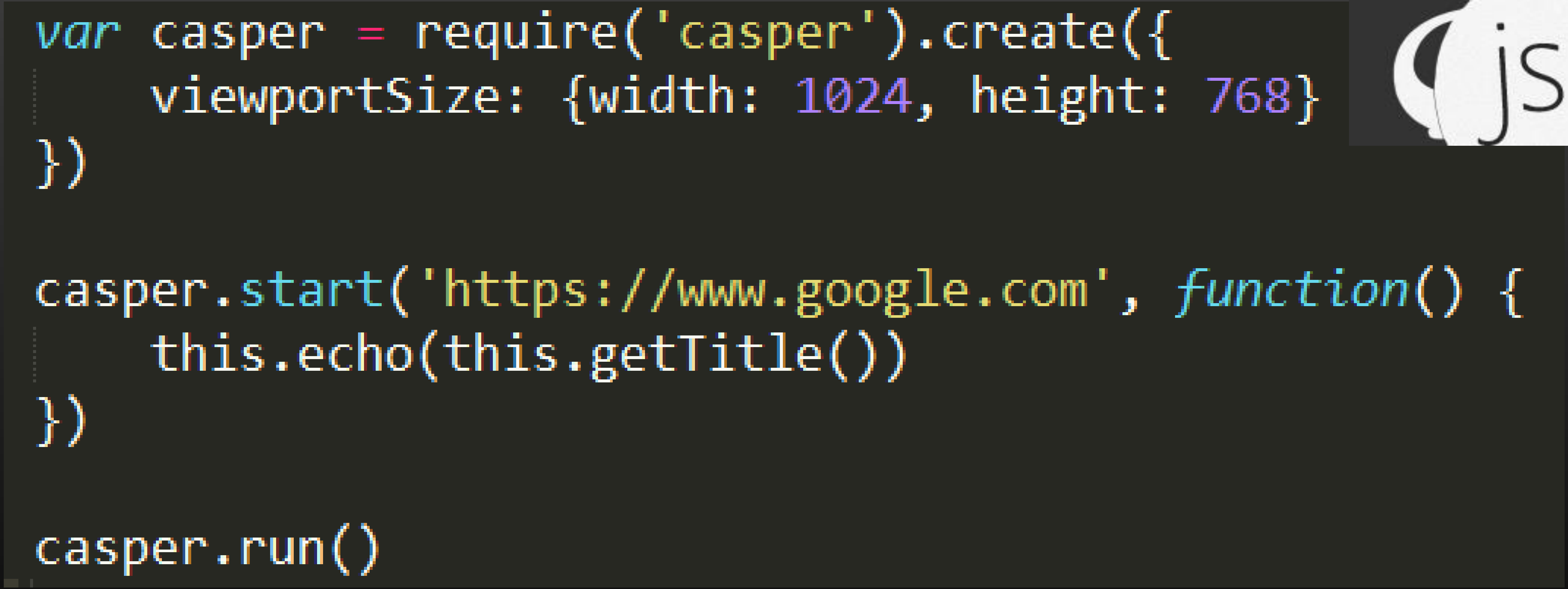
… why do we need the GUI?

PhantomJS

Karma and Selenium are set to use Chrome but we can have them use PhantomJS for headless testing instead.

Headless Browsing with Casper

```
npm install -g casperjs@1.1.0-beta3
```



```
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
```

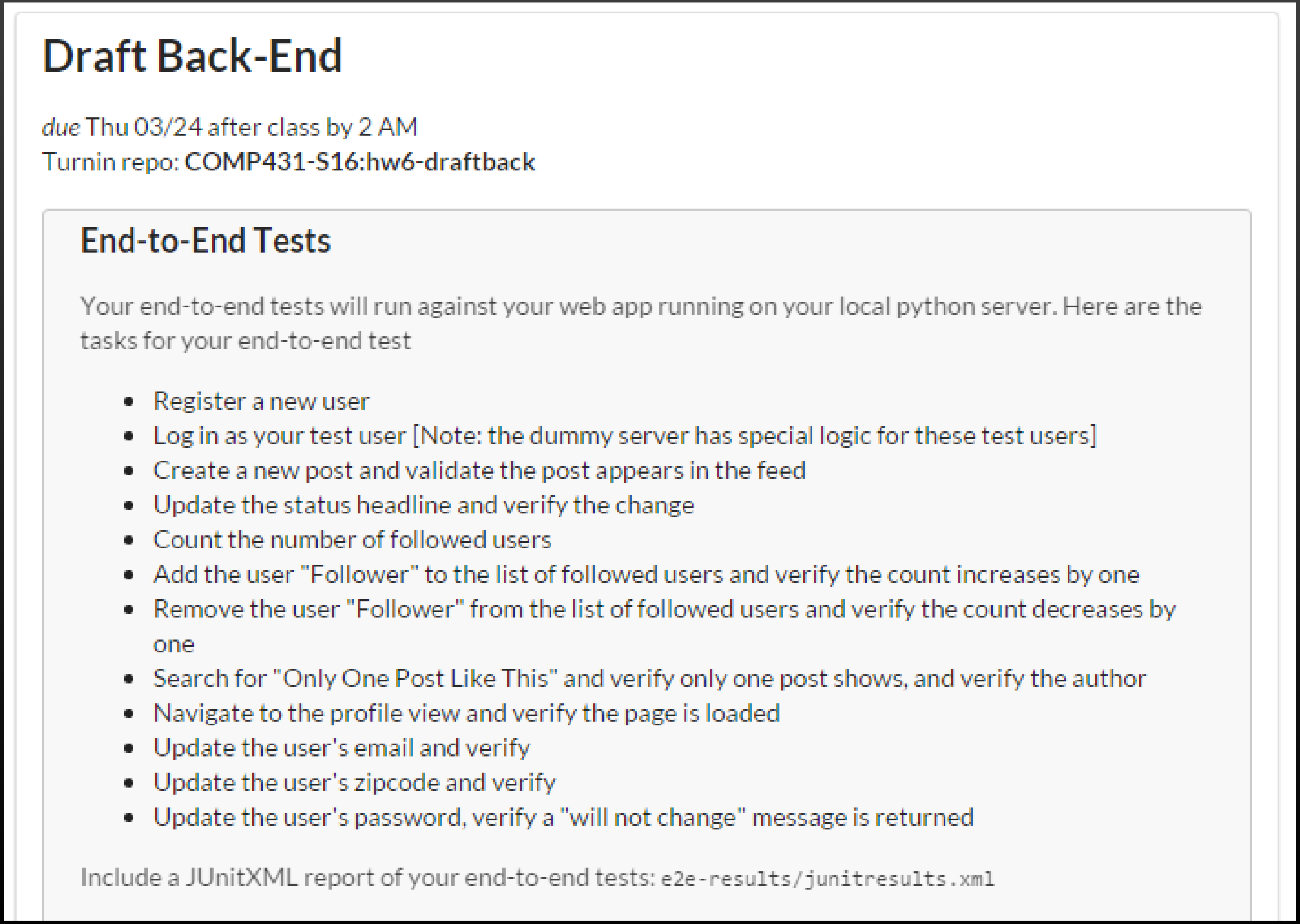
Assignments: *a look ahead…*

HW5: Frontend (due 10/18)

- Functional React+Redux
- Unit tests with Coverage
- Login / Logout
- Articles and search
- Status headline

HW6: Draft Backend

- Finalize Frontend
  - Upload images
  - Edit profile
  - Followers
  - End-to-End Tests
- Draft Backend
  - /headline 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

https://www.clear.rice.edu/comp431/sample/mocking.zip

- Download and unpack the archive

```
npm install
```

- Implement missing check in ***logout*** in ***e2e/common.js***
- Implement the ***update headline*** spec in ***e2e/login.spec.js***

- e2e testing requires your site to be served **npm start**
- During development have mocha watch **npm run e2e:watch**
- When ready, generate xunit report **npm run e2e:xunit**

/inclass-14/...

https://webdev-rice.herokuapp.com