Web Development

COMP 431 / COMP 531

Lecture 17: Back End Testing

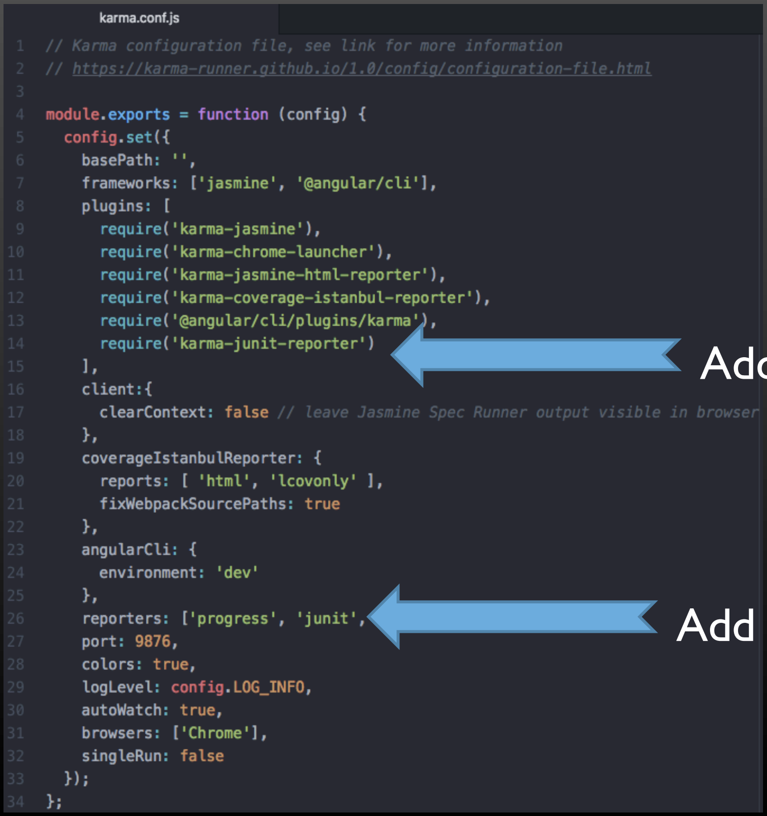
Instructor: Mack Joyner

Department of Computer Science, Rice University

mjoyner@rice.edu

http://www.clear.rice.edu/comp431

Coverage Testing and JUnit Reporting



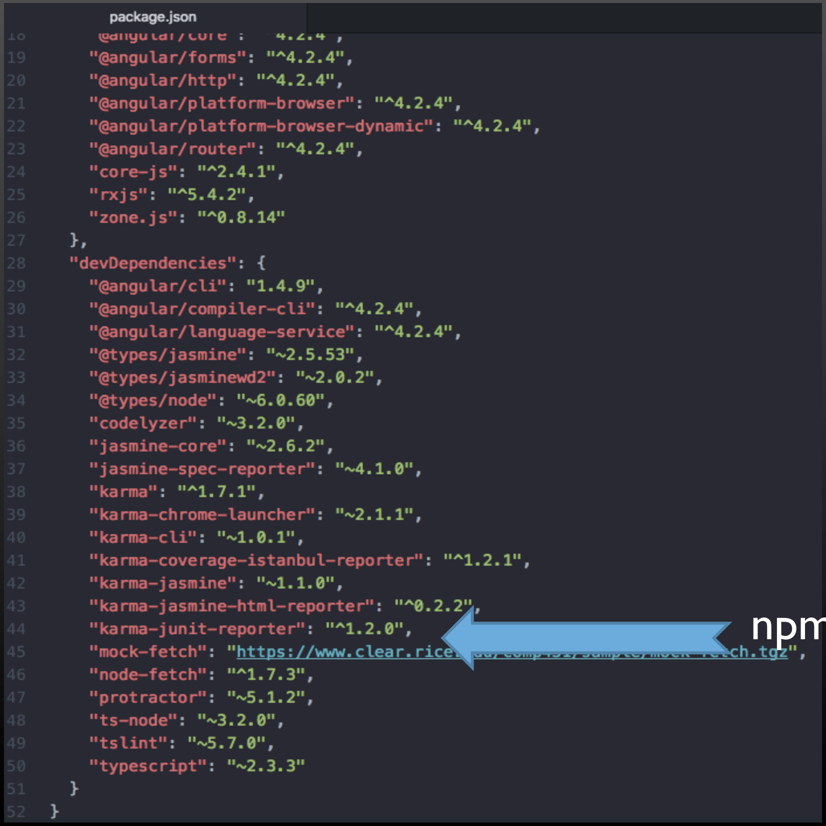
karma.conf.js

```
// Karma configuration file, see link for more information
// https://karma-runner.github.io/1.0/config/configuration-file.html

module.exports = function (config) {
  config.set({
    basePath: '',
    frameworks: ['jasmine', '@angular/cli'],
    plugins: [
      require('karma-jasmine'),
      require('karma-chrome-launcher'),
      require('karma-jasmine-html-reporter'),
      require('karma-coverage-istanbul-reporter'),
      require('@angular/cli/plugins/karma'),
      require('karma-junit-reporter')
    ],
    client:{
      clearContext: false // leave Jasmine Spec Runner output visible in browser
    },
    coverageIstanbulReporter: {
      reports: [ 'html', 'lcovonly' ],
      fixWebpackSourcePaths: true
    },
    angularCli: {
      environment: 'dev'
    },
    reporters: ['progress', 'junit',
    port: 9876,
    colors: true,
    logLevel: config.LOG_INFO,
    autoWatch: true,
    browsers: ['Chrome'],
    singleRun: false
  });
};
```

Add karma-junit reporter

Add junit reporter

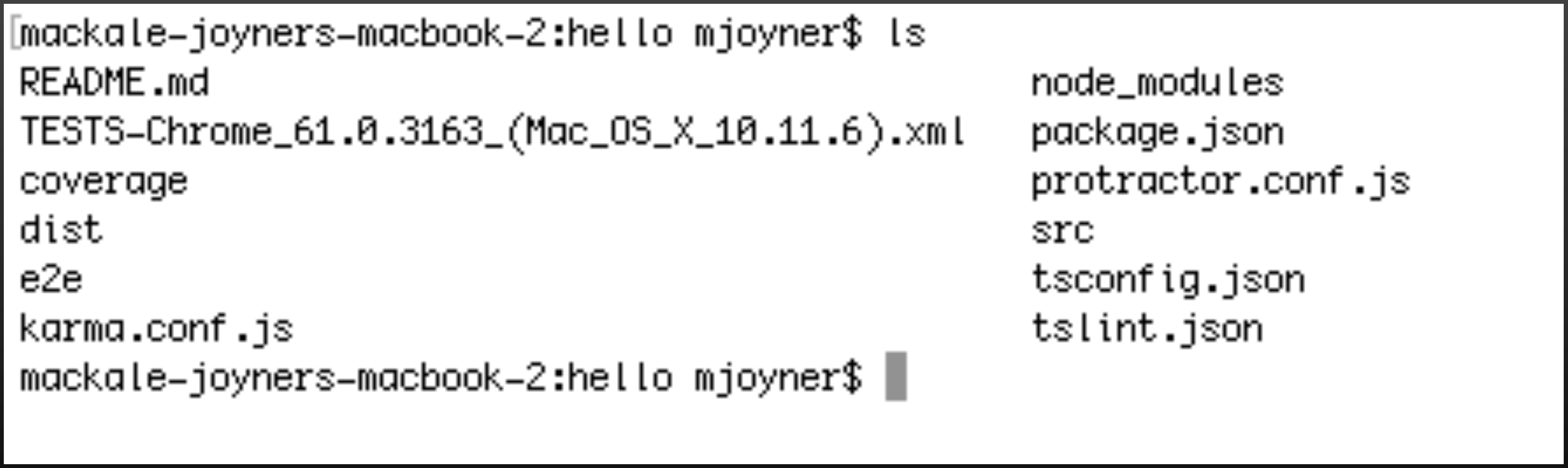
Coverage Testing and JUnit Reporting



```
    "@angular/core": "^4.2.4",
    "@angular/forms": "^4.2.4",
    "@angular/http": "^4.2.4",
    "@angular/platform-browser": "^4.2.4",
    "@angular/platform-browser-dynamic": "^4.2.4",
    "@angular/router": "^4.2.4",
    "core-js": "^2.4.1",
    "rxjs": "^5.4.2",
    "zone.js": "^0.8.14"
  },
  "devDependencies": {
    "@angular/cli": "1.4.9",
    "@angular/compiler-cli": "^4.2.4",
    "@angular/language-service": "^4.2.4",
    "@types/jasmine": "~2.5.53",
    "@types/jasminewd2": "~2.0.2",
    "@types/node": "~6.0.60",
    "codelyzer": "~3.2.0",
    "jasmine-core": "~2.6.2",
    "jasmine-spec-reporter": "~4.1.0",
    "karma": "^1.7.1",
    "karma-chrome-launcher": "~2.1.1",
    "karma-cli": "~1.0.1",
    "karma-coverage-istanbul-reporter": "^1.2.1",
    "karma-jasmine": "~1.1.0",
    "karma-jasmine-html-reporter": "^0.2.2",
    "karma-junit-reporter": "^1.2.0",
    "mock-fetch": "https://www.clear.rice.edu/comp431/sample/mock-fetch.tgz",
    "node-fetch": "^1.7.3",
    "protractor": "~5.1.2",
    "ts-node": "~3.2.0",
    "tslint": "~5.7.0",
    "typescript": "~2.3.3"
  }
}
```

npm install karma-junit-reporter

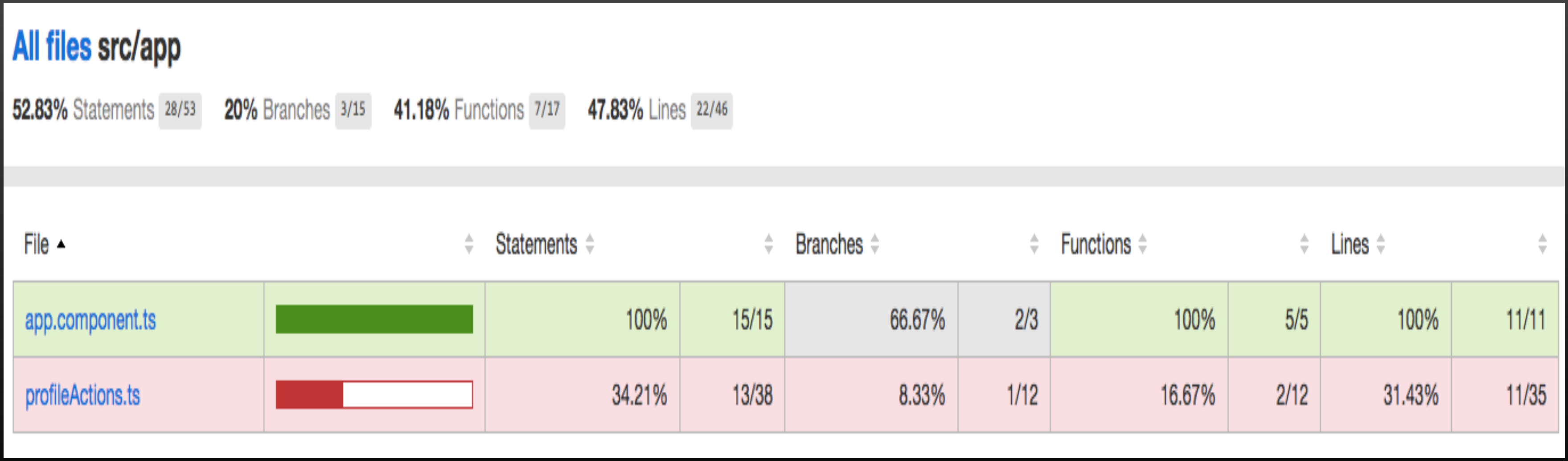
Coverage Testing and JUnit Reporting



```
mackale-joyners-macbook-2:hello mjoyner$ ls
README.md                                         node_modules
TESTS-Chrome_61.0.3163_(Mac_OS_X_10.11.6).xml     package.json
coverage                                          protractor.conf.js
dist                                              src
e2e                                               tsconfig.json
karma.conf.js                                     tslint.json
mackale-joyners-macbook-2:hello mjoyner$
```

Coverage Testing and JUnit Reporting: Demo

Coverage Testing and JUnit Reporting: Demo

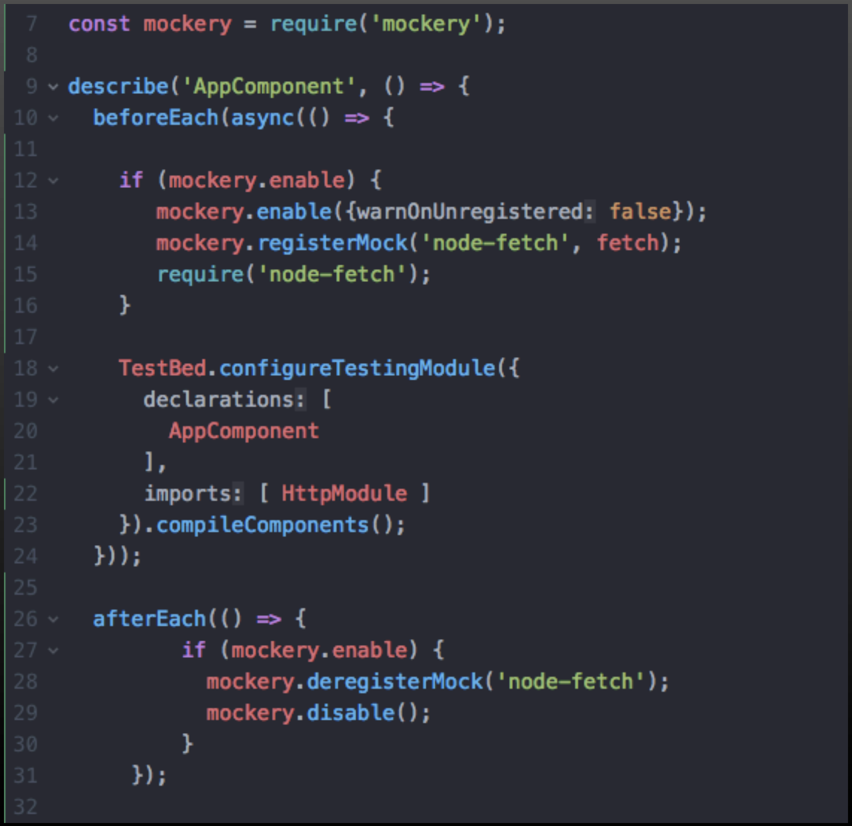


All files src/app

52.83% Statements 28/53 20% Branches 3/15 41.18% Functions 7/17 47.83% Lines 22/46

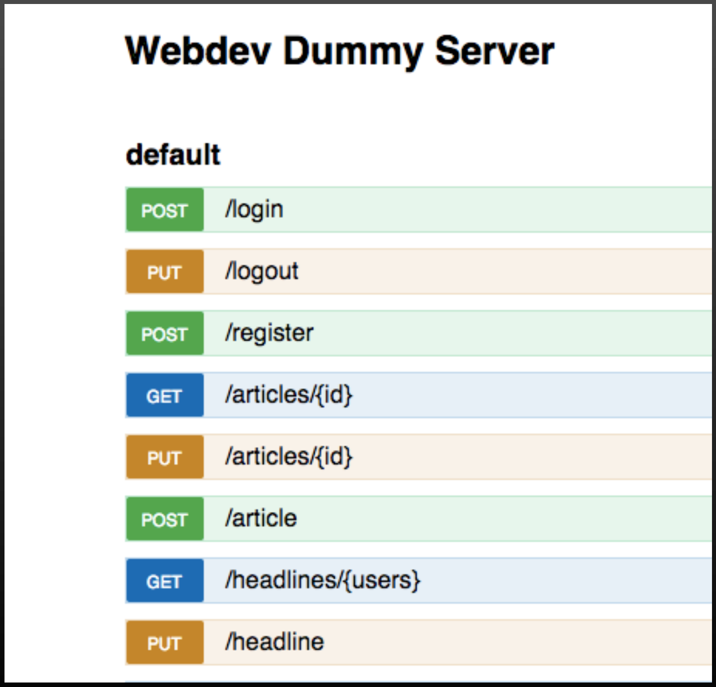
File ▲		Statements		Branches		Functions		Lines	
app.component.ts		100%	15/15	66.67%	2/3	100%	5/5	100%	11/11
profileActions.ts		34.21%	13/38	8.33%	1/12	16.67%	2/12	31.43%	11/35

Mockery: app.component.spec.js



```
const mockery = require('mockery');

describe('AppComponent', () => {
  beforeEach(async(() => {

    if (mockery.enable) {
       mockery.enable({warnOnUnregistered: false});
       mockery.registerMock('node-fetch', fetch);
       require('node-fetch');
    }

    TestBed.configureTestingModule({
      declarations: [
        AppComponent
      ],
      imports: [ HttpModule ]
    }).compileComponents();
  }));

  afterEach(() => {
         if (mockery.enable) {
           mockery.deregisterMock('node-fetch');
           mockery.disable();
         }
     });

```

Swagger API: Webdev Dummy Server

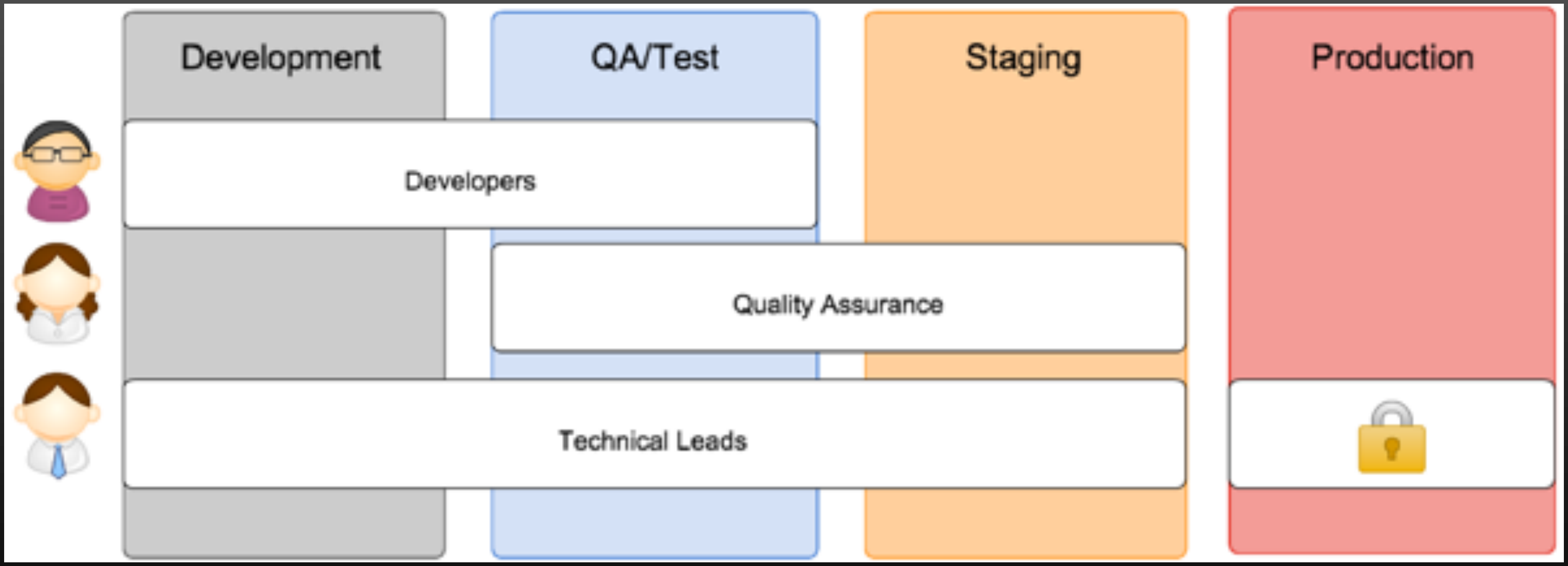


Webdev Dummy Server

default

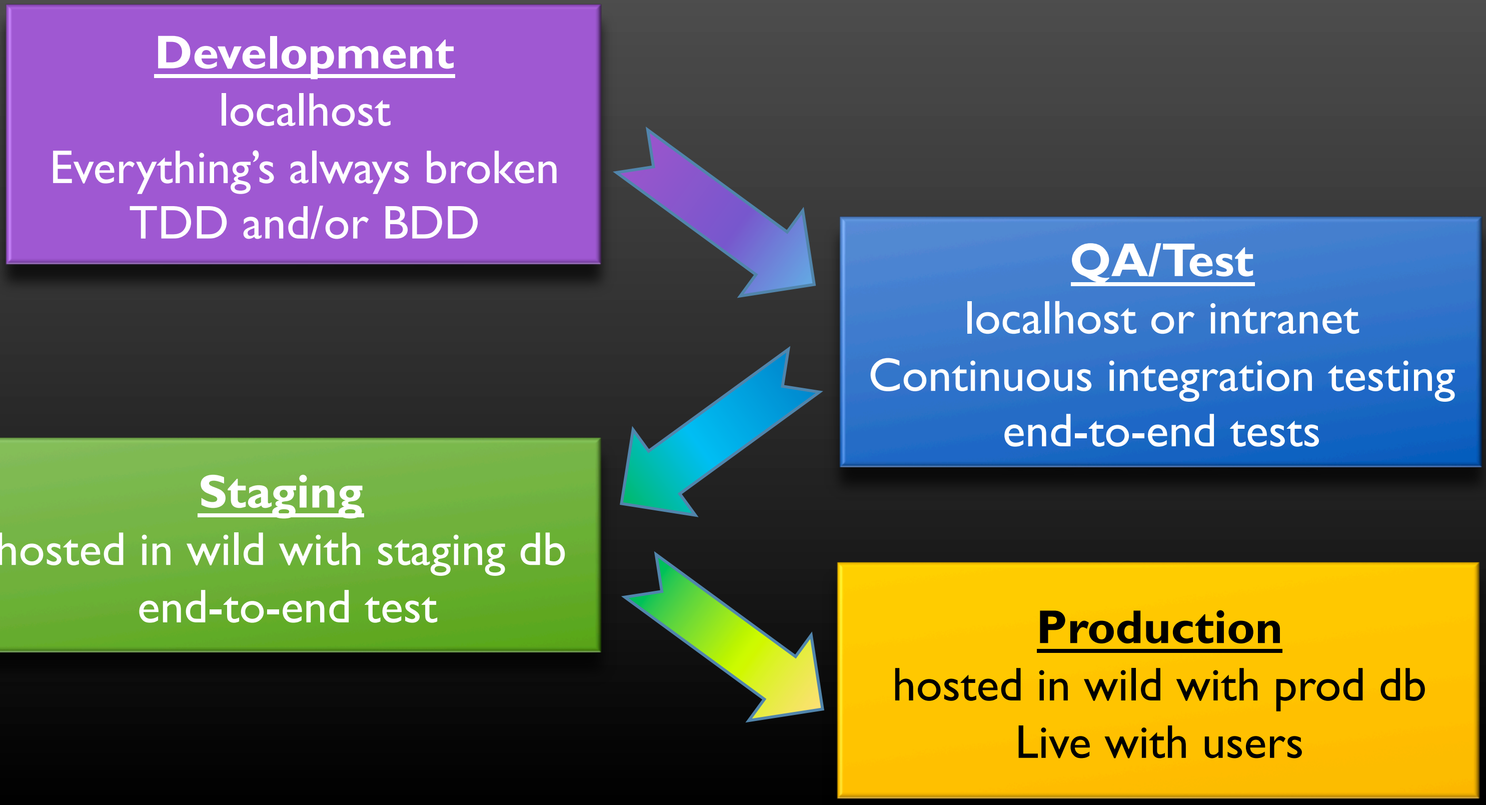
POST	/login
PUT	/logout
POST	/register
GET	/articles/{id}
PUT	/articles/{id}
POST	/article
GET	/headlines/{users}
PUT	/headline

Development to Production



http://blog.ashurex.com/2013/11/18/devops-agency/

Development to Production



Development
localhost
Everything’s always broken
TDD and/or BDD

QA/Test
localhost or intranet
Continuous integration testing
end-to-end tests

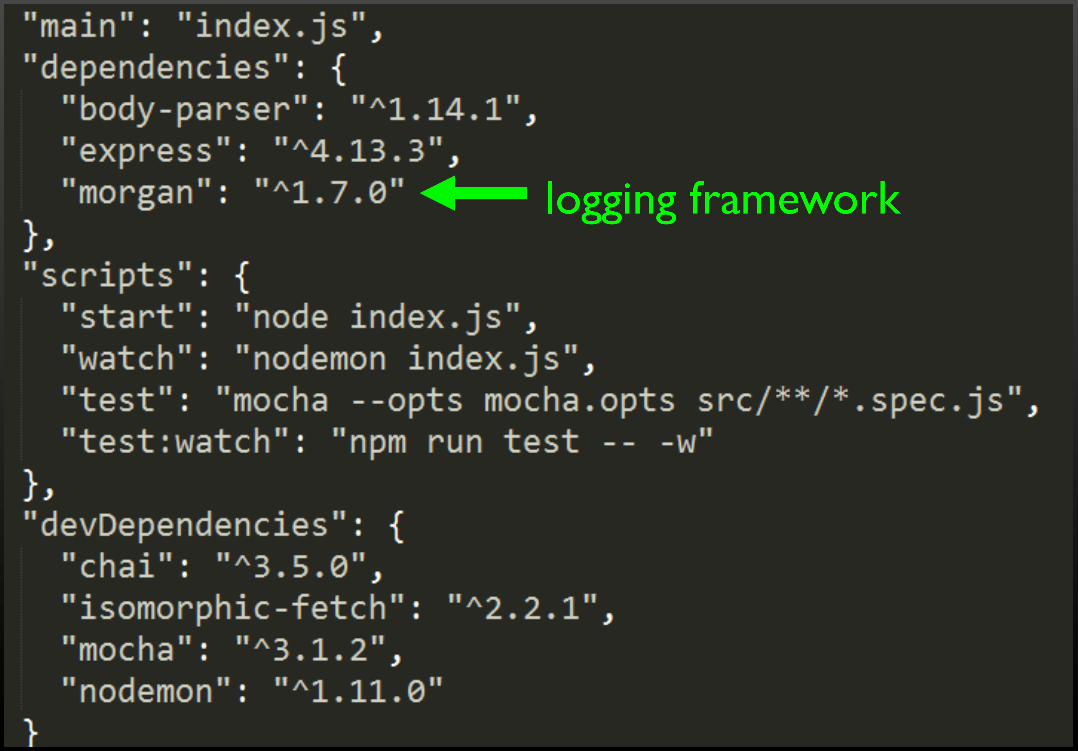
Staging
hosted in wild with staging db
end-to-end test

Production
hosted in wild with prod db
Live with users

http://blog.codeship.com/jasmine-node-js-application-testing-tutorial/

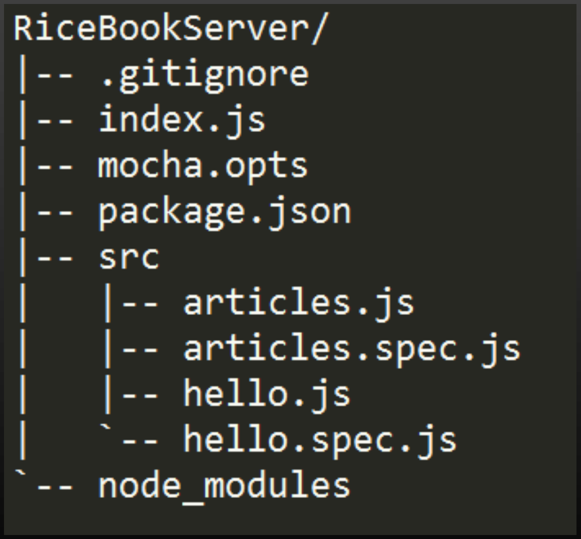
Unit Testing Node: Set Up

https://www.clear.rice.edu/comp431/sample/RiceBookServer/package.json



```
"main": "index.js",
"dependencies": {
  "body-parser": "^1.14.1",
  "express": "^4.13.3",
  "morgan": "^1.7.0"
},
"scripts": {
  "start": "node index.js",
  "watch": "nodemon index.js",
  "test": "mocha --opts mocha.opts src/**/*.spec.js",
  "test:watch": "npm run test -- -w"
},
"devDependencies": {
  "chai": "^3.5.0",
  "isomorphic-fetch": "^2.2.1",
  "mocha": "^3.1.2",
  "nodemon": "^1.11.0"
}
```

logging framework

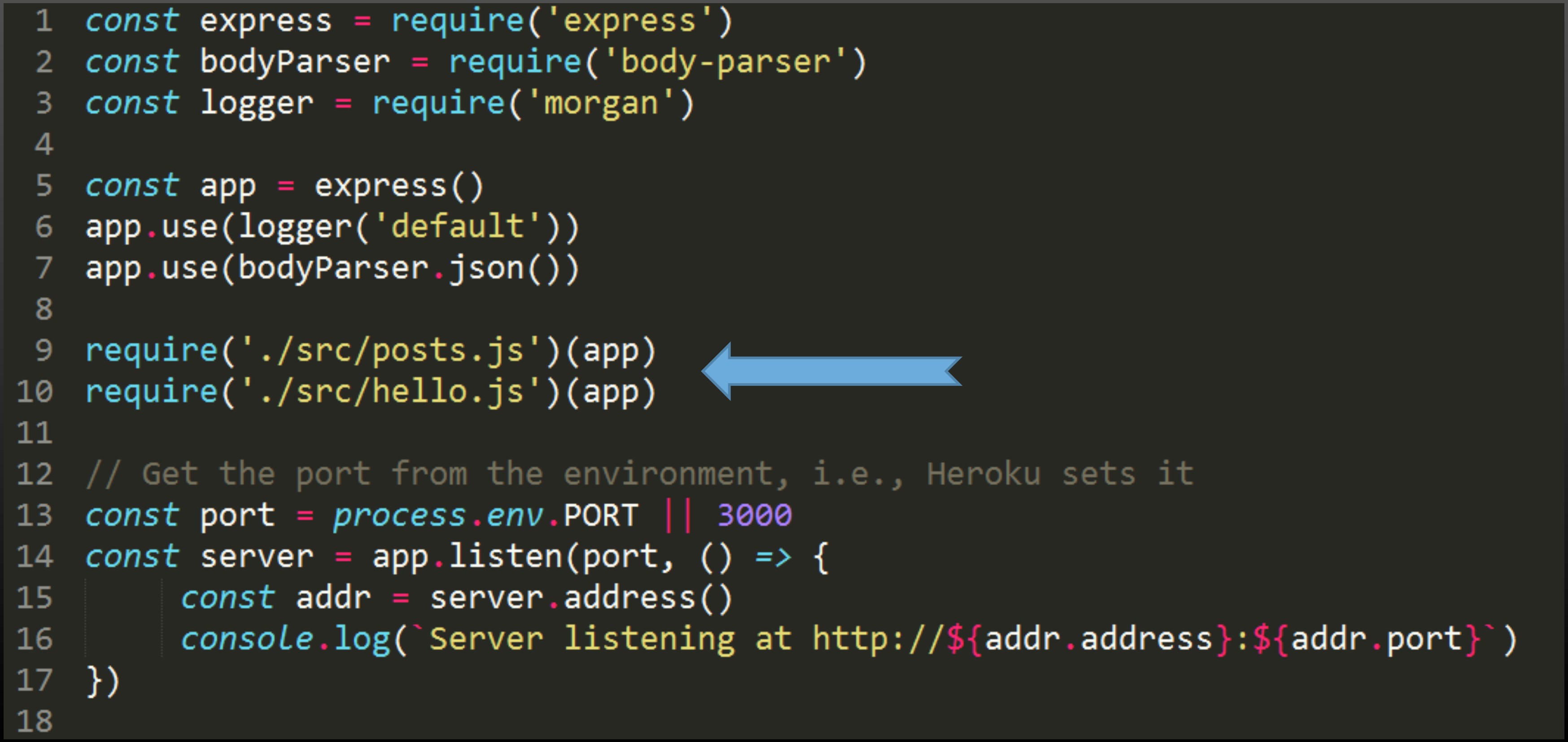


```
RiceBookServer/
|-- .gitignore
|-- index.js
|-- mocha.opts
|-- package.json
|-- src
|   |-- articles.js
|   |-- articles.spec.js
|   |-- hello.js
|   `-- hello.spec.js
`-- node_modules
```

Spin up the server and mocha in watch mode

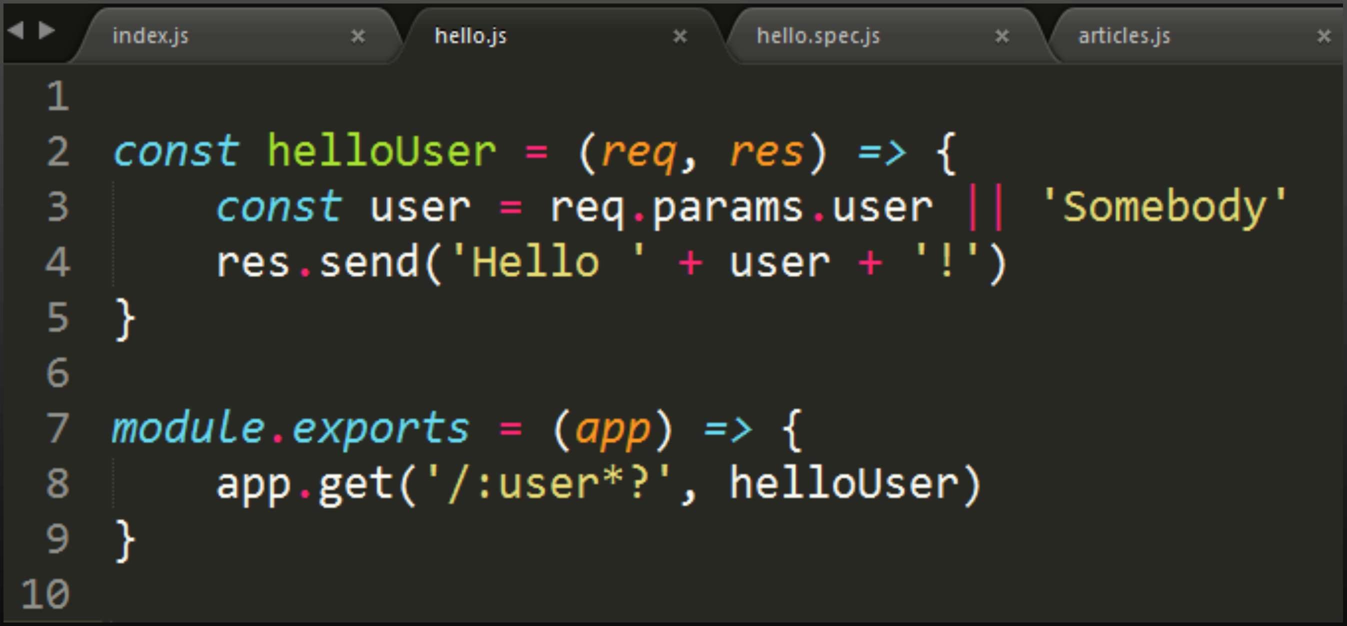
```
# Do these each in a separate window

> npm start

> npm run test:watch
```

Background…



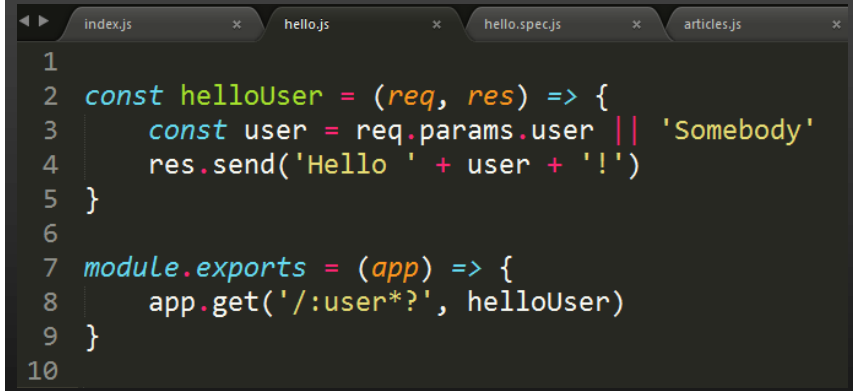
```
const express = require('express')
const bodyParser = require('body-parser')
const logger = require('morgan')

const app = express()
app.use(logger('default'))
app.use(bodyParser.json())

require('./src/posts.js')(app)
require('./src/hello.js')(app)

// Get the port from the environment, i.e., Heroku sets it
const port = process.env.PORT || 3000
const server = app.listen(port, () => {
      const addr = server.address()
      console.log(`Server listening at http://${addr.address}:${addr.port}`)
})
```

Hello…

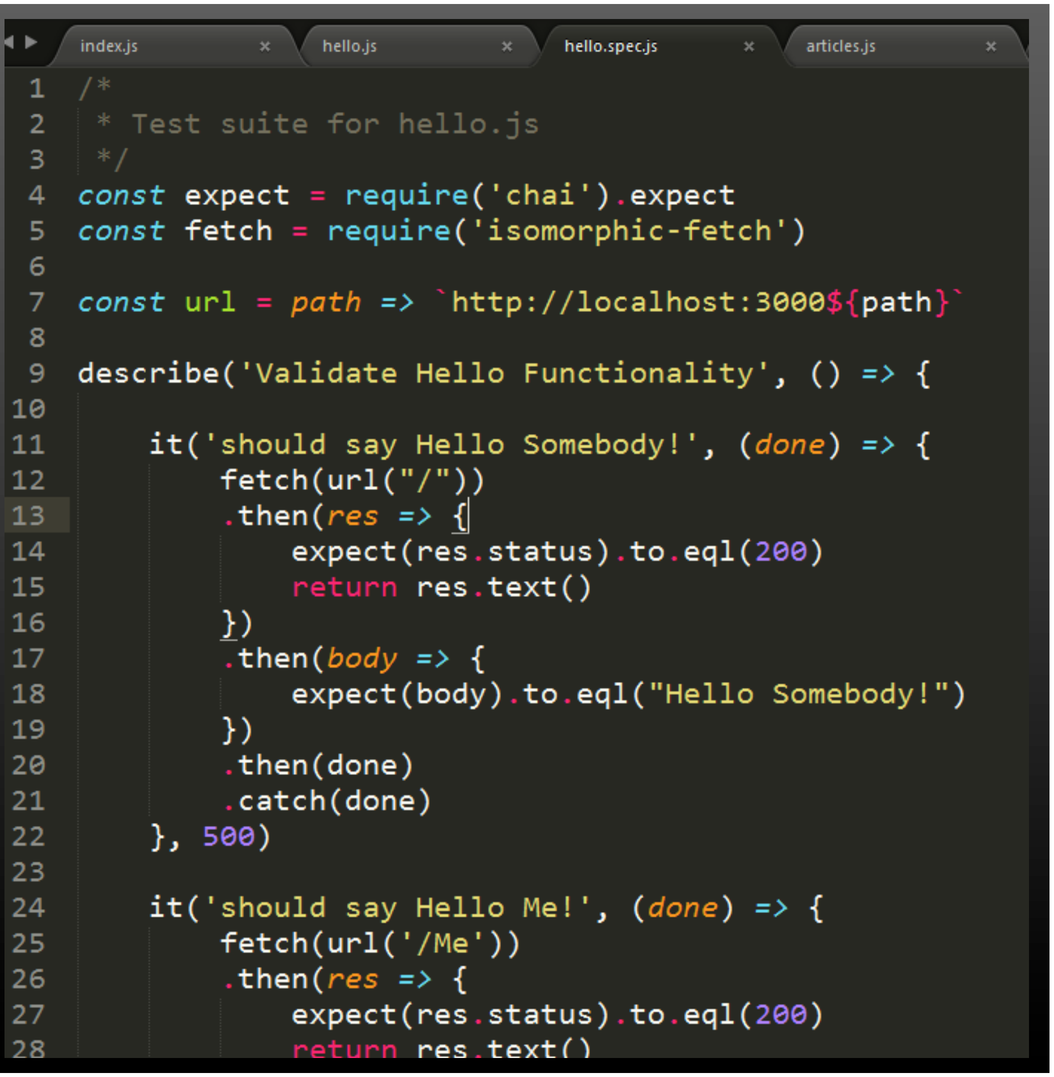


```javascript

const helloUser = (req, res) => {
    const user = req.params.user || 'Somebody'
    res.send('Hello ' + user + '!')
}

module.exports = (app) => {
    app.get('/:user*?', helloUser)
}

```

Hello.spec!



```javascript

const helloUser = (req, res) => {
    const user = req.params.user || 'Somebody'
    res.send('Hello ' + user + '!')
}

module.exports = (app) => {
    app.get('/:user*?', helloUser)
}

```



```javascript
/*
 * Test suite for hello.js
 */
const expect = require('chai').expect
const fetch = require('isomorphic-fetch')

const url = path => `http://localhost:3000${path}`

describe('Validate Hello Functionality', () => {

    it('should say Hello Somebody!', (done) => {
        fetch(url("/"))
        .then(res => {
            expect(res.status).to.eql(200)
            return res.text()
        })
        .then(body => {
            expect(body).to.eql("Hello Somebody!")
        })
        .then(done)
        .catch(done)
    }, 500)

    it('should say Hello Me!', (done) => {
        fetch(url('/Me'))
        .then(res => {
            expect(res.status).to.eql(200)
            return res.text()
```

Sorry! WE'RE
UNIT TESTING