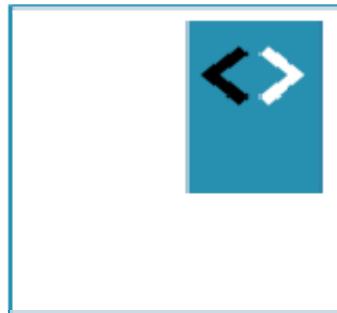




Angular 2

Module 11 - Testing - WIP



Peter Kassenaar –
info@kassenaar.com



Testing - some shortcuts

Goed introductie artikel

The screenshot shows a DZone article page. At the top, there's a navigation bar with links for REF CARDZ, GUIDES, ZONES, and various technology categories like AGILE, BIG DATA, CLOUD, DATABASE, DEVOPS, INTEGRATION, IOT, JAVA, MOBILE, PERFORMANCE, SECURITY, and WEB DEV. There's also a search icon and a user profile icon.

Testing With Angular 2: Some Recipes (Talk and Slides)

Juri Stumpflohner reflects on his recent talk about diving deeper into testing Angular 2 apps. He also links to a dedicated code repository on GitHub with the purpose of collecting testing recipes for various scenarios one might encounter while testing Angular applications.

by Juri Strumpflohner MVB · Jan. 16, 17 · Web Dev Zone

Like (-2) Comment (0) Save Tweet 4,327 Views

Join the DZone community and get the full member experience. JOIN FOR FREE

Start coding today to experience the powerful engine that drives data application's development, brought to you in partnership with Qlik.

I recently wanted to dive deeper into testing Angular applications, in specific on how to write proper unit tests for some common scenarios you might encounter.

Dave, the organizer of the Angular Hamburg Meetup group, asked me whether I'd be interested in

Subscribe

<https://dzone.com/articles/talk-testing-with-angular-some-recipes>

Tooling

The image is a collage of screenshots for three testing tools: Jasmine, Karma, and Protractor.

- Jasmine:** Write Unit Tests & Specs
- Karma:** Test Runner
- Protractor:** End to end testing

Jasmine: Write Unit Tests & Specs

The Jasmine homepage features its logo and the text "Behavior-Driven Javascript". Below the logo is a sidebar with the word "FAST" and the text "Low overhead, no external dependencies." A code snippet shows a simple test:

```
describe("A suite is just a function", function() {
  var a;
  it("and so is a spec", function() {
    a = true;
  });
});
```

Karma: Test Runner

The Karma homepage features its logo and the text "KARMA". It includes a quote from the AngularJS team: "On the [AngularJS](#) team, we rely on testing and we always seek better tools to make our life easier. That's why we created Karma - a test runner that fits all our needs." It also has links to "View project on GitHub" and "npm install karma".

Protractor: End to end testing

The Protractor homepage features its logo and the text "end to end testing for AngularJS". It includes a "View on GitHub" button and a "Follow @ProtractorTest" button. A screenshot of a terminal window shows a "Testacular - JavaScript Test Runner" session with a file tree and some test code. The text below describes Protractor as an end-to-end test framework for Angular and AngularJS applications.

Protractor is an end-to-end test framework for Angular and AngularJS applications. Protractor runs tests against your application running in a real browser, interacting with it as a user would.

Test Like a User
Protractor is built on top of WebDriverJS, which uses native events and browser-specific drivers to interact with your application as a user would.

For Angular Apps
Protractor supports Angular-specific locator strategies, which allows you to test Angular-specific elements without any setup effort on your part.

Automatic Waiting
You no longer need to add waits and sleeps to your test. Protractor can automatically execute the next step in your test the moment the webpage finishes pending tasks, so you don't have to worry about waiting for your test and webpage to sync.

- https://twitter.com/_victormejia/status/871971522033979392
- New, as of June 6, 2017 - Testing in Headless Chrome

<https://jasmine.github.io/>

<https://karma-runner.github.io/1.0/index.html>

<http://www.protractortest.org/#/>

General testing pattern/syntax

- `import {Person} from "./person.model";`
- `// Person.model.ts`
`export class Person {`
 `constructor(public name?: string,`
 `public email?: string) {`
 `}`

 `sayHello(): string {`
 `return `Hi, ${this.name}`;`
 `}`
}

General unit test pattern

```
// Generic testing pattern

// 1. Describe block for every test suite
describe('The Person', () => {

    // 2. Variables used by this test suite
    let aPerson;

    // 3. Setup block, run before every individual test
    beforeEach(() => {
        aPerson = new Person('Peter');
    });

    // 4. Clean up after every individual test
    afterEach(() => {
        aPerson = null;
    });

    // 5. Perform each test in an it()-block
    it('should say Hello', () => {
        let msg = aPerson.sayHello();
        expect(msg).toBe('Hi, Peter');
    });

    // 6. More it()-blocks...
});

});
```

We're using angular-cli here

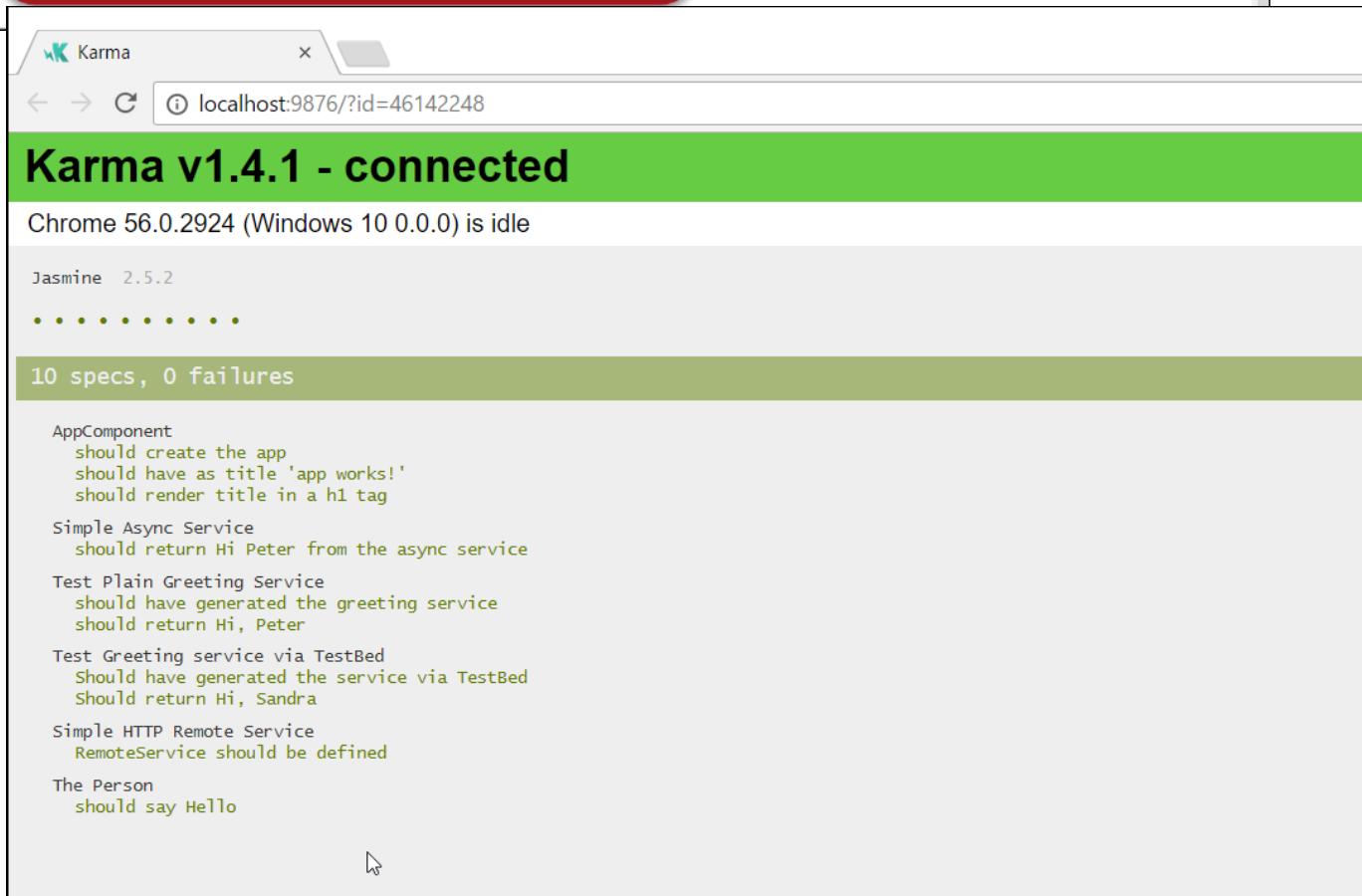
- Angular-cli: all dependencies already installed and configured.
 - Command: `ng test`
- Manual project? Install & setup karma and jasmine yourself
 - Create and adapt `karma.conf.js`
 - `karma --init`
 - Install and setup jasmine reporters
 - We're not doing that here.
 - See documentation



```
...
},
"devDependencies": {
  ...
  "@types/jasmine": "2.5.38",
  "@types/node": "~6.0.60",
  "codelyzer": "~2.0.0",
  "jasmine-core": "~2.5.2",
  "jasmine-spec-reporter": "~3.2.0",
  "karma": "~1.4.1",
  "karma-chrome-launcher": "~2.0.0",
  "karma-cli": "~1.0.1",
  "karma-jasmine": "~1.1.0",
  "karma-jasmine-html-reporter": "^0.2.2",
  "karma-coverage-istanbul-reporter": "^0.2.0",
  "protractor": "~5.1.0",
  ...
}
```

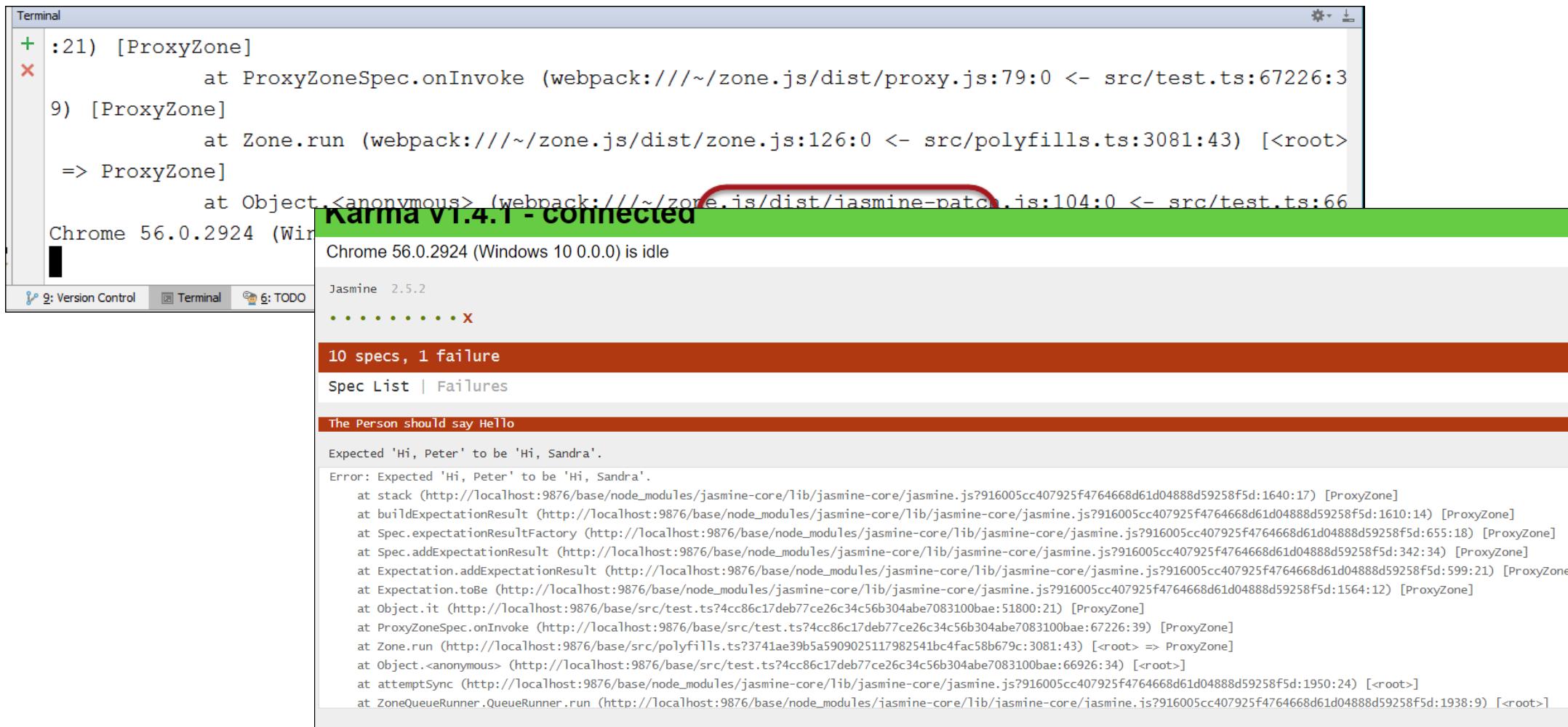
Browser like...

```
C:\Users\Peter Kassenaar\Desktop\ng-testing>ng test
10 03 2017 16:50:40.157:WARN [karma]: No captured browser, open http://localhost:9876/
10 03 2017 16:50:40.170:INFO [karma]: Karma v1.4.1 server started at http://0.0.0.0:9876/
10 03 2017 16:50:40.171:INFO [launcher]: Launching browser Chrome with unlimited concurrency
10 03 2017 16:50:40.310:INFO [launcher]: Starting browser Chrome
10 03 2017 16:50:42.220:INFO [Chrome 56.0.2924 (Windows 10 0.0.0)]: Connected on socket 30x3NIH4ohFYk
MAuAAAA with id 46142248
Chrome 56.0.2924 (Windows 10 0.0.0): Executed 10 of 10 SUCCESS (0.461 secs / 0.439 secs)
```



If anything goes wrong...

Hard to read output:



The screenshot shows a terminal window with the following content:

```
+ :21) [ProxyZone]
      at ProxyZoneSpec.onInvoke (webpack:///~/zone.js/dist/proxy.js:79:0 <- src/test.ts:67226:3
  9) [ProxyZone]
      at Zone.run (webpack:///~/zone.js/dist/zone.js:126:0 <- src/polyfills.ts:3081:43) [<root>
=> ProxyZone]
      at Object.<anonymous> (webpack:///~/zone.js/dist/jasmine-patch.js:104:0 <- src/test.ts:66
Chrome 56.0.2924 (Win
```

A green banner at the top of the terminal window reads "Karma v1.4.1 - connected". Below it, the terminal displays the following message:

Chrome 56.0.2924 (Windows 10 0.0.0) is idle

Jasmine 2.5.2

• • • • • x

10 specs, 1 failure

[Spec List](#) | [Failures](#)

The Person should say Hello

Expected 'Hi, Peter' to be 'Hi, Sandra'.

Error: Expected 'Hi, Peter' to be 'Hi, Sandra'.
at stack (http://localhost:9876/base/node_modules/jasmine-core/lib/jasmine-core/jasmine.js?916005cc407925f4764668d61d04888d59258f5d:1640:17) [ProxyZone]
at buildExpectationResult (http://localhost:9876/base/node_modules/jasmine-core/lib/jasmine-core/jasmine.js?916005cc407925f4764668d61d04888d59258f5d:1610:14) [ProxyZone]
at Spec.expectationResultFactory (http://localhost:9876/base/node_modules/jasmine-core/lib/jasmine-core/jasmine.js?916005cc407925f4764668d61d04888d59258f5d:655:18) [ProxyZone]
at Spec.addExpectationResult (http://localhost:9876/base/node_modules/jasmine-core/lib/jasmine-core/jasmine.js?916005cc407925f4764668d61d04888d59258f5d:342:34) [ProxyZone]
at Expectation.addExpectationResult (http://localhost:9876/base/node_modules/jasmine-core/lib/jasmine-core/jasmine.js?916005cc407925f4764668d61d04888d59258f5d:599:21) [ProxyZone]
at Expectation.toBe (http://localhost:9876/base/node_modules/jasmine-core/lib/jasmine-core/jasmine.js?916005cc407925f4764668d61d04888d59258f5d:1564:12) [ProxyZone]
at Object.it (http://localhost:9876/base/src/test.ts?4cc86c17deb77ce26c34c56b304abe7083100bae:51800:21) [ProxyZone]
at ProxyZoneSpec.onInvoke (http://localhost:9876/base/src/test.ts?4cc86c17deb77ce26c34c56b304abe7083100bae:67226:39) [ProxyZone]
at Zone.run (http://localhost:9876/base/src/polyfills.ts?3741ae39b5a5909025117982541bc4fac58b679c:3081:43) [<root> => ProxyZone]
at Object.<anonymous> (http://localhost:9876/base/src/test.ts?4cc86c17deb77ce26c34c56b304abe7083100bae:66926:34) [<root>]
at attemptSync (http://localhost:9876/base/node_modules/jasmine-core/lib/jasmine-core/jasmine.js?916005cc407925f4764668d61d04888d59258f5d:1950:24) [<root>]
at ZoneQueueRunner.QueueRunner.run_ (http://localhost:9876/base/node_modules/jasmine-core/lib/jasmine-core/jasmine.js?916005cc407925f4764668d61d04888d59258f5d:1938:9) [<root>]

Solution: install custom reporters

Custom reporters

- Choices available.
- Editors choice: Karma Mocha Reporter
 - `npm install --save-dev karma-mocha-reporter`
- Edit `karma.conf.js`
 - `plugins: require('karma-mocha-reporter')`
 - `reporters: ? ['mocha', 'karma-remap-istanbul']`
`: ['mocha' , 'kjhtml'] ,`
 - instead of progress

Running test again

Much nicer output and better error reports

```
START:  
  AppComponent  
    ✓ should create the app  
    ✓ should have as title 'app works!'  
    ✓ should render title in a h1 tag  
  Simple Async Service  
    ✓ should return Hi Peter from the async service  
  Test Plain Greeting Service  
    ✓ should have generated the greeting service  
    ✓ should return Hi, Peter  
  Test Greeting service via TestBed  
    ✓ Should have generated the service via TestBed  
    ✓ Should return Hi, Sandra  
  Simple HTTP Remote Service  
    ✓ RemoteService should be defined  
  The Person  
    ✓ should say Hello  
  
Finished in 0.485 secs / 0.46 secs @ 17:49:52 GMT+0100 (West-Europa (standaardtijd))  
  
SUMMARY:  
✓ 10 tests completed
```

...or, when errors occur

SUMMARY:

```
✓ 9 tests completed
✗ 1 test failed
```

FAILED TESTS:

The Person

✗ should say Hello

Chrome 56.0.2924 (Windows 10 0.0.0)

Expected 'Hi, Peter' to be 'Hi, Sandra'.

at Object.it (webpack:///src/app/tests/generic-pattern.spec.ts:23:14 <- src/test.ts:51800:21

[ProxyZone]

Terms and conditions...

- TestBed
- inject
- async
- fakeAsync
- ComponentFixture
- DebugElement
- configureTestingModule

Testing a single service

One of the more easier concepts to test. So let's start there.

```
// greeting.service.ts

import {Injectable} from '@angular/core';

@Injectable()
export class GreetingService {

  constructor() {}

  greet(name: string): string {
    return `Hi, ${name}`;
  }
}
```

```
// greeting.service.spec.ts

import {GreetingService} from './greeting.service';

describe('Test Plain Greeting Service', () => {
  let greetingService;

  beforeEach(() => {
    greetingService = new GreetingService();
  });

  it('should have generated the greeting service', () => {
    expect(greetingService).toBeTruthy()
  });

  it('should return Hi, Peter', () => {
    let msg = greetingService.greet('Peter');
    expect(msg).toEqual('Hi, Peter');
  });
});
```

Output

```
Terminal
+ START:
✖ Test Plain Greeting Service
  ✓ should have generated the greeting service
  ✓ should return Hi, Peter
The Person
  ✓ should say Hello
```

But what about DI?

// HIER VERDER

But what about DI?

- Most of the time you don't test a single service.
- Angular relies on DI for your services and uses them in the context of an `@NgModule` – show simple Module.
- In testing this is referred to as the TestBed.
- A TestBed takes a similar configuration object to mimic a module
- `TestBed.configureTestingModule({, providers: [<services>] })`
- In TestBed you don't need to configure the complete module. Only the parts you need. In this case the greetingService.
- Then instantiate the service from the TestBed. All possible other dependencies are resolved. `service = TestBed.get(GreetingService);`
- Alternative notation: via injector, instead of `TestBed.get()`.
- `beforeEach(inject([GreetingService], (s: GreetingService) => { service = s }));`

Async behavior

- If the services uses async calls, for instance promises or Observables
- Create service with Async call, for instance new Promise which resolves after 100ms.
- In spec-file, it always returns true, because the expect-statement is not run in the .then()-clause
- ```
service.greetAsync('Peter').then((result)=>{
 expect(result).toEqual('Hi, Peter12234');
}) // test passed
```
- Solution: wrap it inside Angular async-construct, like
- ```
it('should return Hi Peter from the async service', async(()=>{
  service.greetAsync('Peter').then((result)=>{
    expect(result).toEqual('Hi, Peter12234');
  })
})) // Test failed.
```
- Also available : fakeAsync. More control. Less often used

Testing XHR calls

- Setup remote service to fetch some data over http
- *// Get fake people*
getPeople():Observable<any>{
 return
 this.http.get('http://someEndPoint/somePeople.json')
 .map(result =>result.json());
}

It will fail

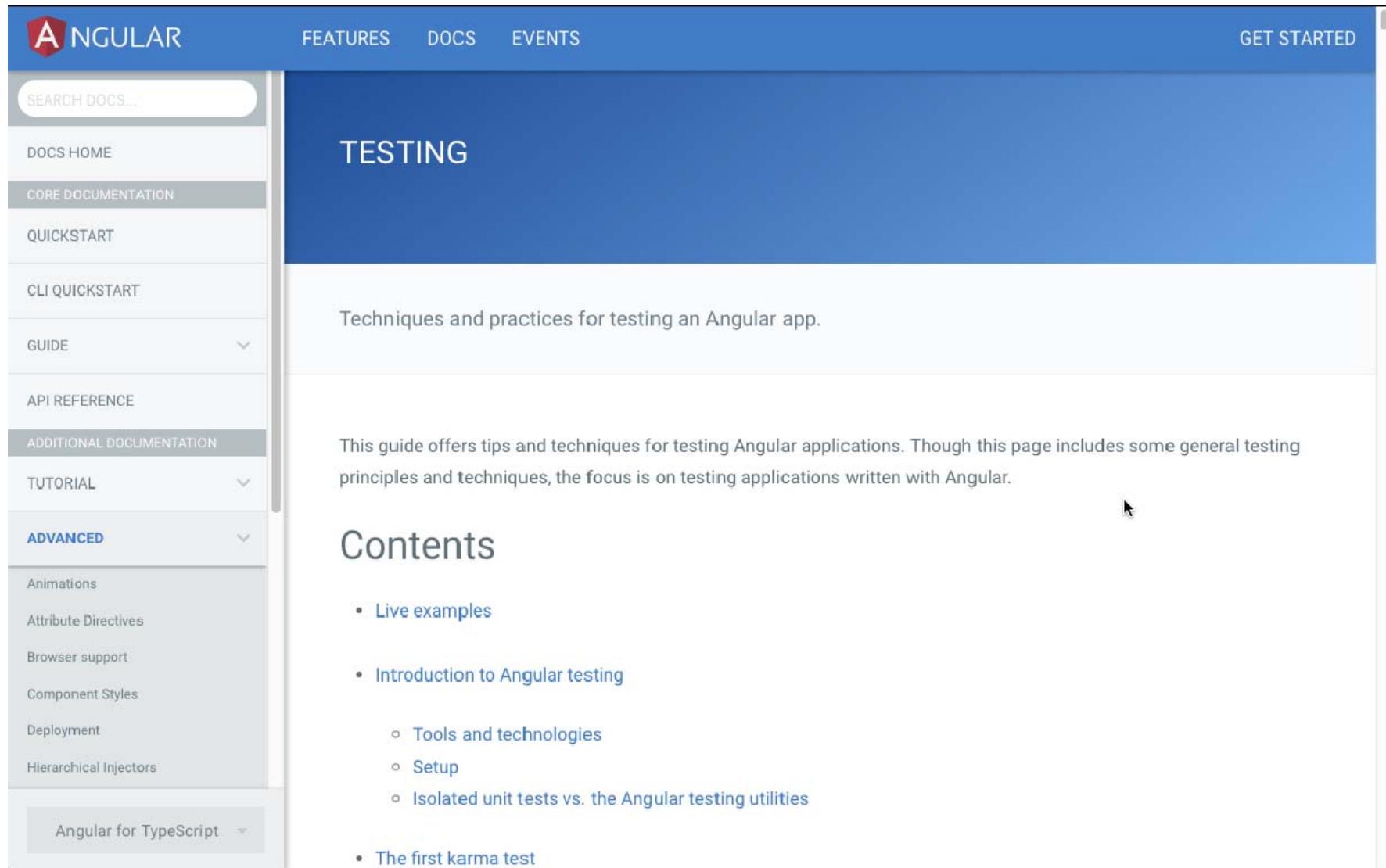
- Configure the test like so:

```
beforeEach(()=>{
    TestBed.configureTestingModule({
        imports: [HttpModule],
        providers: [RemoteService]
    })
    service = TestBed.get(RemoteService);
});
```
- **it('Should return people.json', async(()=>{
 let people;
 service.getPeople().subscribe((result)=>{
 people = result;
 });
 expect(people).toBeDefined();
}))**

Solution: create Mock Backend

- Hier verder....

Testing documentation



The screenshot shows the Angular documentation website. The top navigation bar includes the Angular logo, a search bar labeled "SEARCH DOCS...", and links for "FEATURES", "DOCS", "EVENTS", and "GET STARTED". On the left, a sidebar menu is open under the "CORE DOCUMENTATION" section, showing options like "DOCS HOME", "QUICKSTART", "CLI QUICKSTART", "GUIDE", "API REFERENCE", "ADDITIONAL DOCUMENTATION", "TUTORIAL", and "ADVANCED". The "ADVANCED" section is expanded, listing "Animations", "Attribute Directives", "Browser support", "Component Styles", "Deployment", and "Hierarchical Injectors". At the bottom of the sidebar is a link to "Angular for TypeScript". The main content area has a blue header with the word "TESTING". Below it, a sub-header reads "Techniques and practices for testing an Angular app." A paragraph explains that the guide offers tips and techniques for testing Angular applications, focusing on Angular-specific testing. The "Contents" section lists several topics: "Live examples", "Introduction to Angular testing" (with sub-points "Tools and technologies", "Setup", and "Isolated unit tests vs. the Angular testing utilities"), and "The first karma test".

<https://angular.io/guide/testing>

Repo – Angular Testing recipes

The screenshot shows the GitHub repository page for 'juristr/angular-testing-recipes'. The repository name is at the top left, followed by 'This repository' and 'Search' buttons. To the right are 'Pull requests', 'Issues', 'Gist', and a user icon. Below the header are buttons for 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', 'Pulse', and 'Graphs'. The 'Code' button is highlighted. On the right, there are 'Watch 3', 'Star 18', 'Fork 1', and a bell icon. The main content area displays the repository's description: 'Simple testing patterns for Angular version 2+'. Below the description are tags: 'angular', 'jasmine', 'karma', and 'testing'. A summary bar shows '43 commits', '1 branch', '0 releases', and '1 contributor'. Below the bar are buttons for 'Branch: master ▾', 'New pull request', 'Create new file', 'Upload files', 'Find file', and a green 'Clone or download ▾' button. The commit list starts with a commit from 'juristr' on Jan 13, 2017, titled 'docs: adjust intro'. The list continues with commits to 'e2e', 'src', '.gitignore', '.travis.yml', 'README.md', 'angular-cli.json', 'karma.conf.js', 'package.json', and 'protractor.conf.js', all made 'a month ago'.

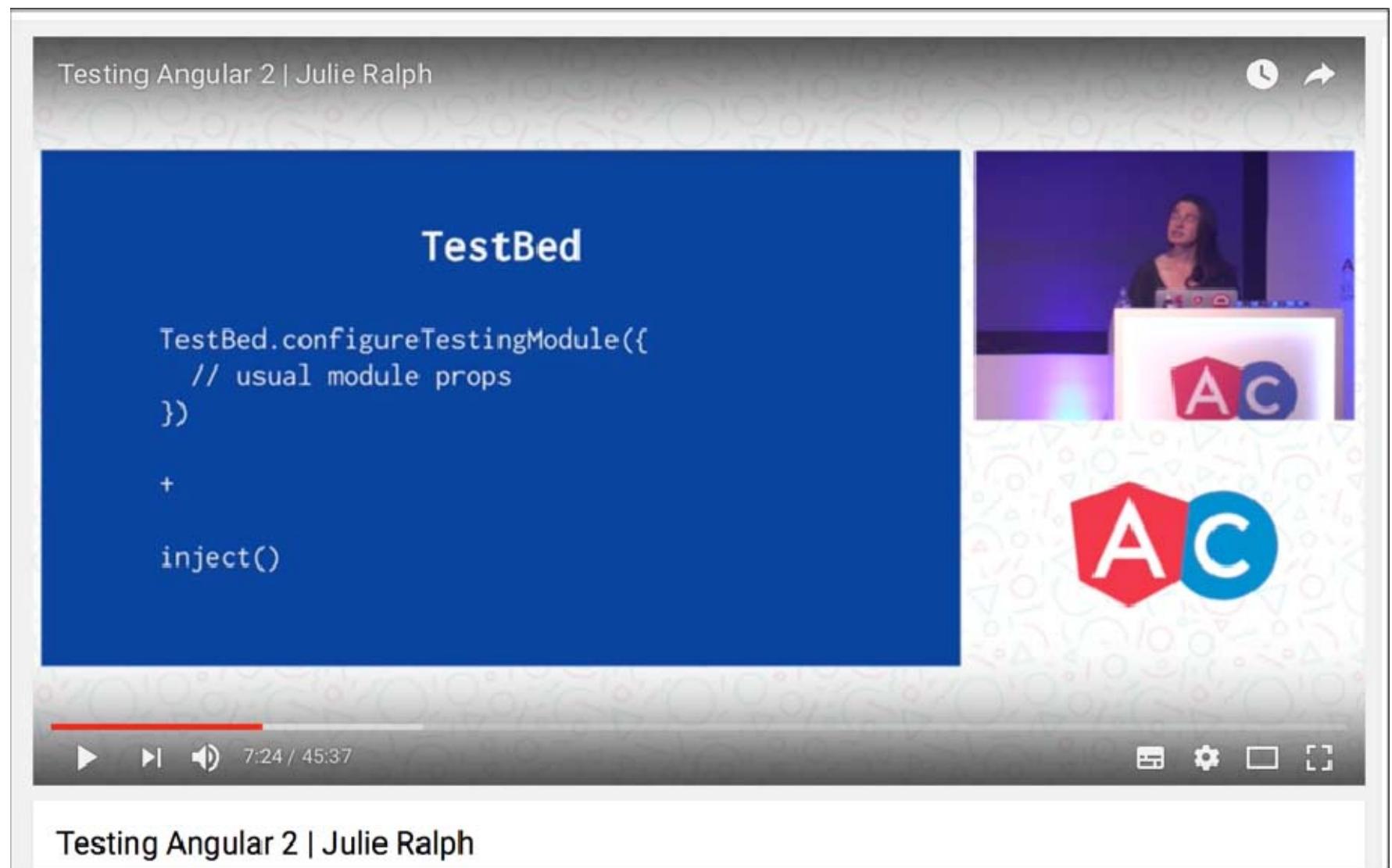
File / Commit Type	Description	Date
e2e	chore: base setup with CLI	a month ago
src	fix: missing parentheses	a month ago
.gitignore	chore: add .vscode to gitignore	a month ago
.travis.yml	chore: add travis config file	a month ago
README.md	docs: adjust intro	a month ago
angular-cli.json	chore: base setup with CLI	a month ago
karma.conf.js	feat: add karma mocha reporter	a month ago
package.json	chore(packages): upgrade TypeScript reference	a month ago
protractor.conf.js	chore: base setup with CLI	a month ago

<https://github.com/juristr/angular-testing-recipes>

```
6  describe('AppComponent', () => {
7    beforeEach(() => {
8      TestBed.configureTestingModule({
9        declarations: [
10          AppComponent
11        ],
12      });
13    });
14
15    it('should create the app', async(() => {
16      let fixture = TestBed.createComponent(AppComponent);
17      let app = fixture.debugElement.componentInstance;
18      expect(app).toBeTruthy();
19    }));
20
21    it(`should have as title 'app works!'", async(() => {
22      let fixture = TestBed.createComponent(AppComponent);
23      let app = fixture.debugElement.componentInstance;
24      expect(app.title).toEqual('app works!');
25    }));

```

Videos on testing



<https://www.youtube.com/watch?v=f493Xf0F2yU>

Goed introductie artikel

The screenshot shows a DZone article page. At the top, the DZone logo and 'Web Dev Zone' are visible, along with a navigation bar for 'REFCARDZ', 'GUIDES', 'ZONES', and various technology categories like 'AGILE', 'BIG DATA', 'CLOUD', etc. A user profile icon and a search bar are also at the top right. The main title of the article is 'Testing With Angular 2: Some Recipes (Talk and Slides)' by Juri Strumpflohner. Below the title is a summary: 'Juri Stumpflohner reflects on his recent talk about diving deeper into testing Angular 2 apps. He also links to a dedicated code repository on GitHub with the purpose of collecting testing recipes for various scenarios one might encounter while testing Angular applications.' The author's profile picture and name are shown, along with the publication date ('Jan. 16, 17') and category ('Web Dev Zone'). Below the summary are social sharing buttons for 'Like', 'Comment', 'Save', and 'Tweet', and a view count of '4,327 Views'. A call-to-action button 'JOIN FOR FREE' is present. The bottom section contains a testimonial from 'Dave' about a meetup group, followed by a 'Subscribe' button.

Over a million developers have joined DZone. [Sign In / Join](#)

REFCARDZ GUIDES ZONES | AGILE BIG DATA CLOUD DATABASE DEVOPS INTEGRATION IOT JAVA MOBILE PERFORMANCE SECURITY WEB DEV

Let's be friends:

Testing With Angular 2: Some Recipes (Talk and Slides)

by Juri Strumpflohner · Jan. 16, 17 · Web Dev Zone

»

Like (-2) Comment (0) Save Tweet

4,327 Views

Join the DZone community and get the full member experience. [JOIN FOR FREE](#)

Start coding today to experience the powerful engine that drives data application's development, brought to you in partnership with [Qlik](#).

I recently wanted to dive deeper into testing Angular applications, in specific on how to write proper unit tests for some common scenarios you might encounter.

Dave, the organizer of [the Angular Hamburg Meetup group](#), asked me whether I'd be interested in

<https://dzone.com/articles/talk-testing-with-angular-some-recipes>