



---

# Angular 2

## Module 11 - Testing - WIP



Peter Kassenaar –  
[info@kassenaar.com](mailto:info@kassenaar.com)



# Testing - some shortcuts

# Goed introductie artikel



The screenshot shows the DZone / Web Dev Zone website. The header includes the DZone logo, navigation links (REFCARDZ, GUIDES, ZONES, etc.), and a search bar. The article title is "Testing With Angular 2: Some Recipes (Talk and Slides)". The author is Juri Strumpflohner, with a bio mentioning his recent talk about testing Angular 2 apps. The article has 4,327 views and a "Like (-2)" button. A promotional banner for DZone membership is visible, along with a "Subscribe" button.

**DZone / Web Dev Zone** 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

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

Juri Strumpflohner 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.*

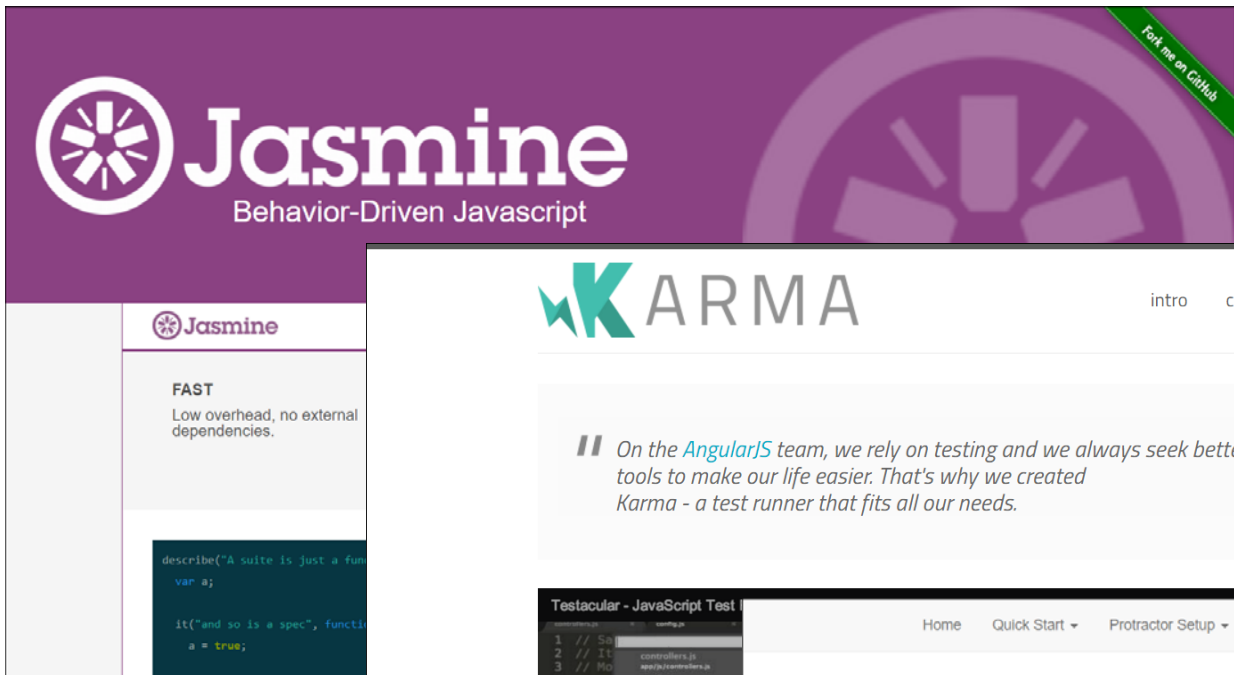
I recently wanted to dive deeper into testing Angular applications, in specific on how to write proper unit tests for some common scenarios you might encounter.

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

Subscribe

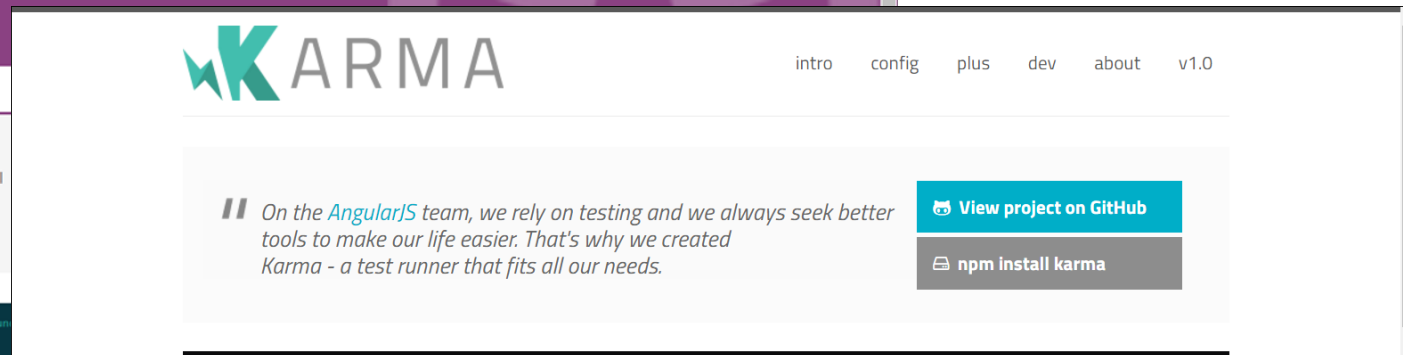
<https://dzone.com/articles/talk-testing-with-angular-some-recipes>

# Tooling

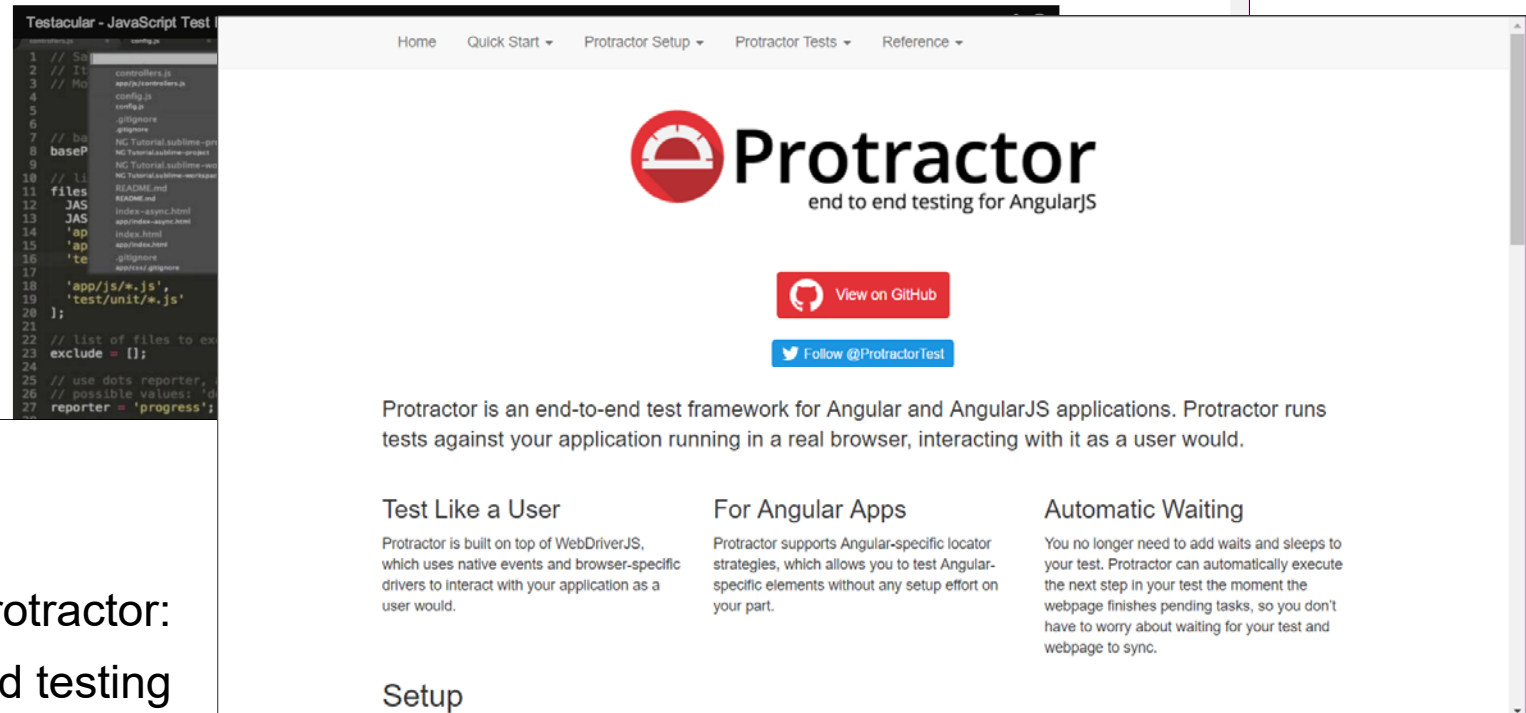


Jasmine:  
Write Unit Tests & Specs

Karma:  
Test Runner



Protractor:  
End to end testing



- [https://twitter.com/\\_victormeja/status/871971522033979392](https://twitter.com/_victormeja/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",
  "codemaker": "~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 30x3NIH4ohFYkMAuAAAA with id 46142248
Chrome 56.0.2924 (Windows 10 0.0.0): Executed 10 of 10 SUCCESS (0.461 secs / 0.439 secs)
```

Karma v1.4.1 - connected

Chrome 56.0.2924 (Windows 10 0.0.0) is idle

Jasmine 2.5.2

10 specs, 0 failures

- 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

# If anything goes wrong...

Hard to read output:

```
Terminal
+ :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/zone.js:104:0 <- src/test.ts:66
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

# Run ng 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:
x 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

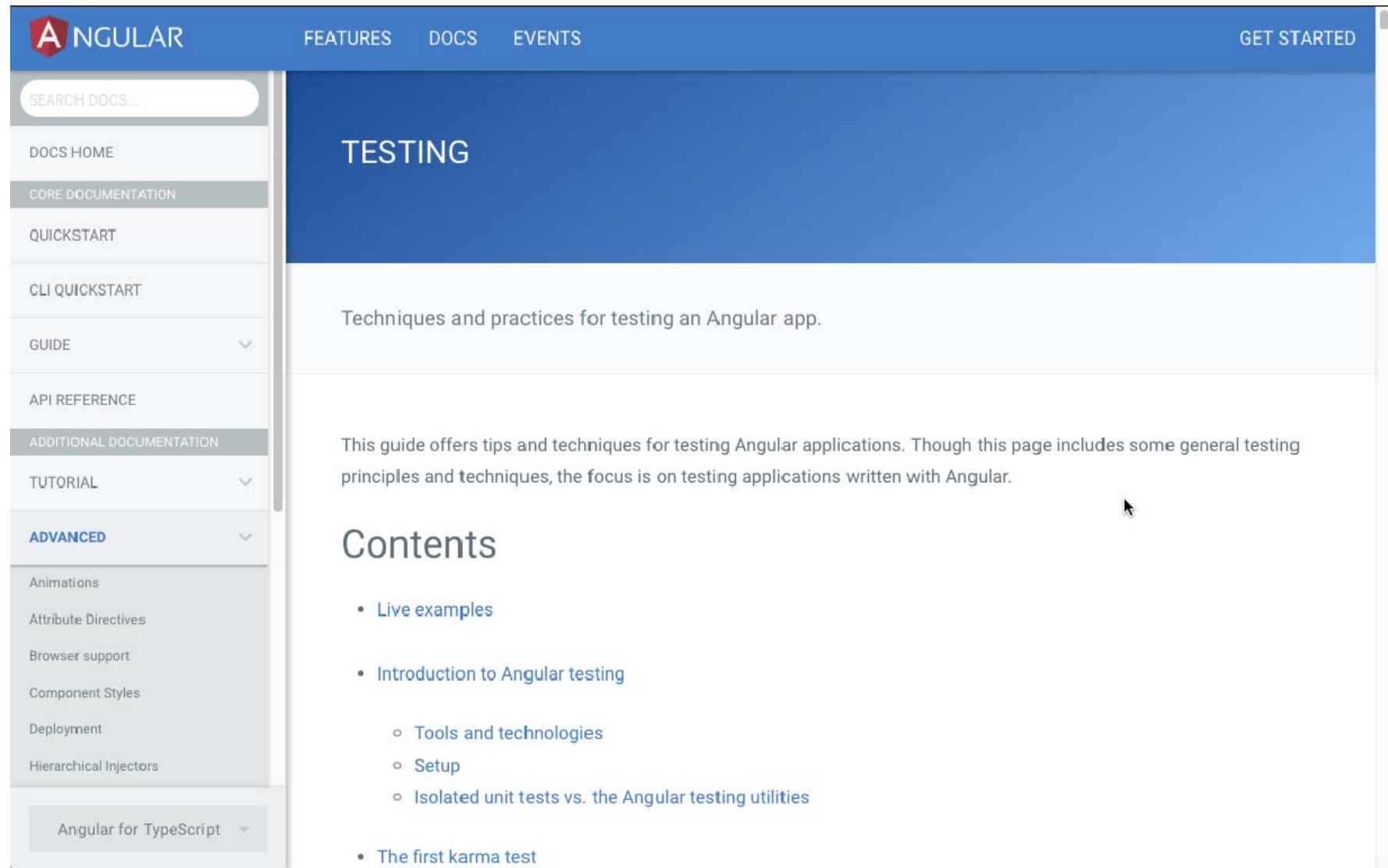
- Configure the test like so:
- ```
beforeEach(() => {  
    TestBed.configureTestingModule({  
        imports: [HttpModule],  
        providers: [RemoteService]  
    })  
    service = TestBed.get(RemoteService);  
});
```
- ```
it('Should return people.json', async(() => {  
    let people;  
    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 is blue with the Angular logo on the left and links for FEATURES, DOCS, EVENTS, and GET STARTED on the right. A search bar is located on the left side of the page. The left sidebar contains a list of documentation categories: DOCS HOME, CORE DOCUMENTATION (QUICKSTART, CLI QUICKSTART, GUIDE, API REFERENCE), ADDITIONAL DOCUMENTATION (TUTORIAL), and ADVANCED (Animations, Attribute Directives, Browser support, Component Styles, Deployment, Hierarchical Injectors, Angular for TypeScript). The main content area has a blue header with the word 'TESTING'. Below this, a subtitle reads 'Techniques and practices for testing an Angular app.' followed by a paragraph: 'This guide offers tips and techniques for testing Angular applications. Though this page includes some general testing principles and techniques, the focus is on testing applications written with Angular.' The 'Contents' section lists several topics: Live examples, Introduction to Angular testing (with sub-items: Tools and technologies, Setup, Isolated unit tests vs. the Angular testing utilities), and The first karma test.

ANGULAR

FEATURES DOCS EVENTS GET STARTED

SEARCH DOCS...

DOCS HOME

CORE DOCUMENTATION

QUICKSTART

CLI QUICKSTART

GUIDE

API REFERENCE

ADDITIONAL DOCUMENTATION

TUTORIAL

ADVANCED

Animations

Attribute Directives

Browser support

Component Styles

Deployment

Hierarchical Injectors

Angular for TypeScript

## TESTING

Techniques and practices for testing an Angular app.

This guide offers tips and techniques for testing Angular applications. Though this page includes some general testing principles and techniques, the focus is on testing applications written with Angular.

### Contents

- [Live examples](#)
- [Introduction to Angular testing](#)
  - [Tools and technologies](#)
  - [Setup](#)
  - [Isolated unit tests vs. the Angular testing utilities](#)
- [The first karma test](#)

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

# Repo – Angular Testing recipes

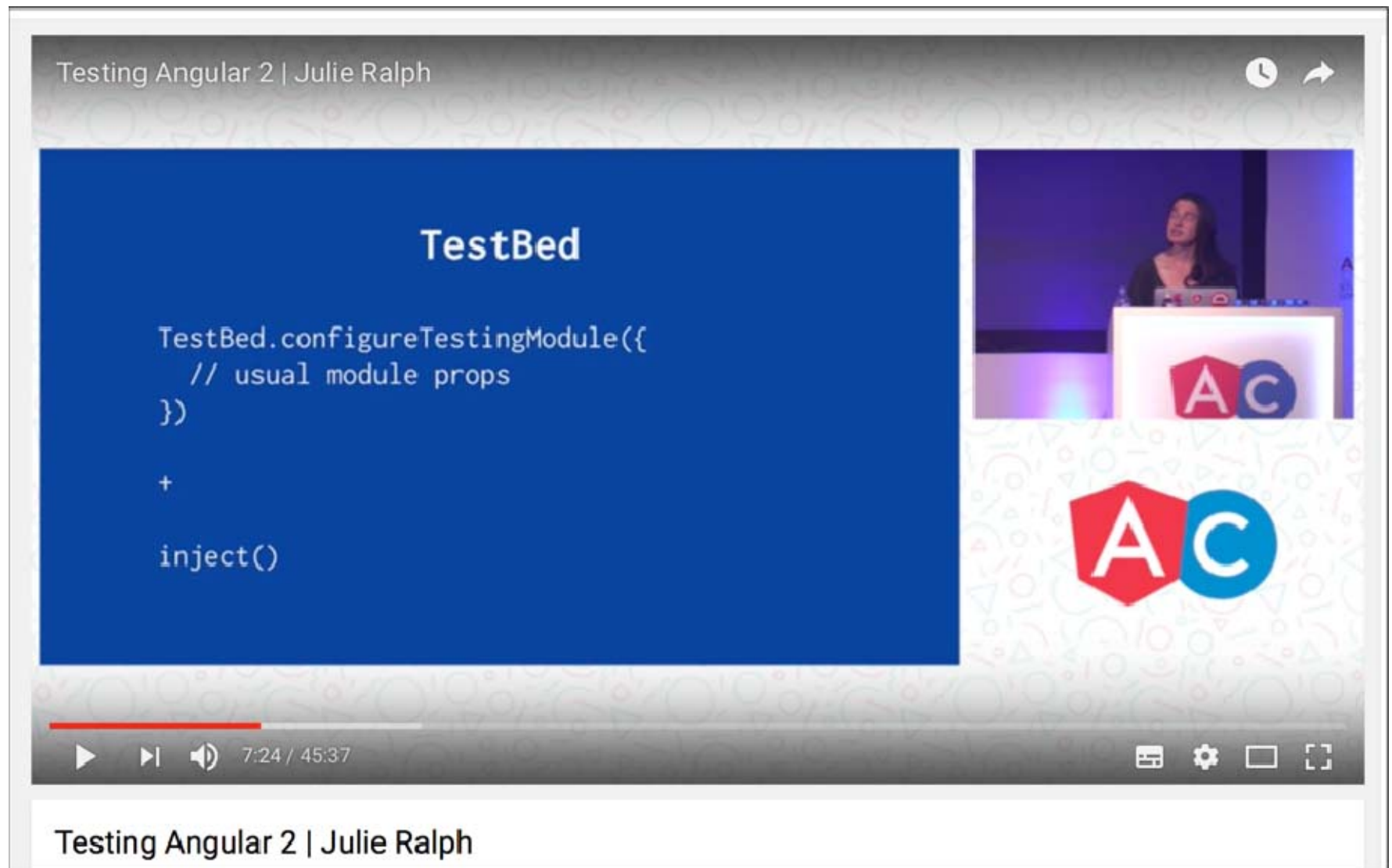
The screenshot shows the GitHub repository page for 'juristr/angular-testing-recipes'. The repository is described as 'Simple testing patterns for Angular version 2+'. It has 43 commits, 1 branch, 0 releases, and 1 contributor. The repository is on the 'master' branch. The file list includes:

File	Commit Message	Time
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 the DZone / Web Dev Zone website. The header includes the DZone logo, navigation links (REFCARDZ, GUIDES, ZONES, etc.), and a search bar. The article title is "Testing With Angular 2: Some Recipes (Talk and Slides)". The author is Juri Strumpflohner, and the article was published on Jan. 16, 17. The article text discusses testing Angular 2 apps and mentions a GitHub repository. There are social media sharing options (Like, Comment, Save, Tweet) and a view count of 4,327. A promotional banner for DZone membership is visible, along with a "Subscribe" button.

**DZone / Web Dev Zone** 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

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

Juri Strumpflohner 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>