Advanced Testing Recipes (v6+)

Google Developer Expert

International Speaker

Spoken at 83 events in 24 countries

Blogger

Writer

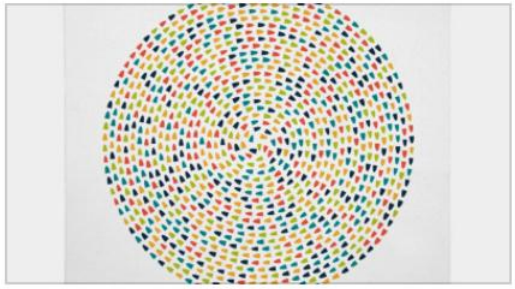
Applause from Houssein Djirdeh, Chau Nguyen, and 273 others

Angular—Testing Guide (v4+)

Nine easy-to-follow examples using TestBed,

Applause from Seyed Mostafa Meshkati, Coskun Deniz, and 141 others

Angular — Supercharge your Router transitions using new animation features (v4.3+)



Applause from Aaron Frost, Angular Academy, and 61 others

Angular — Applying Motion principles to a listing

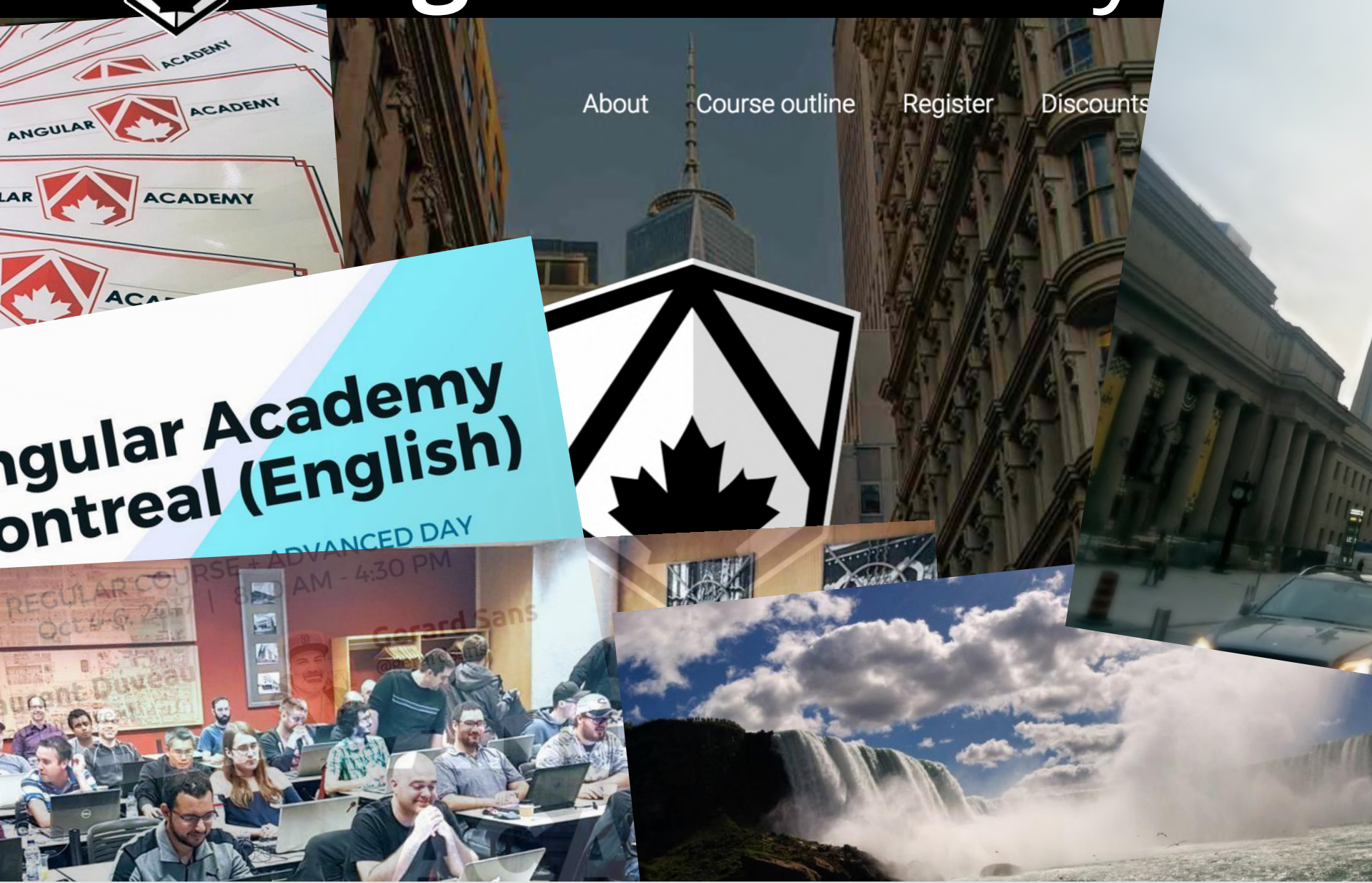
Community Leader

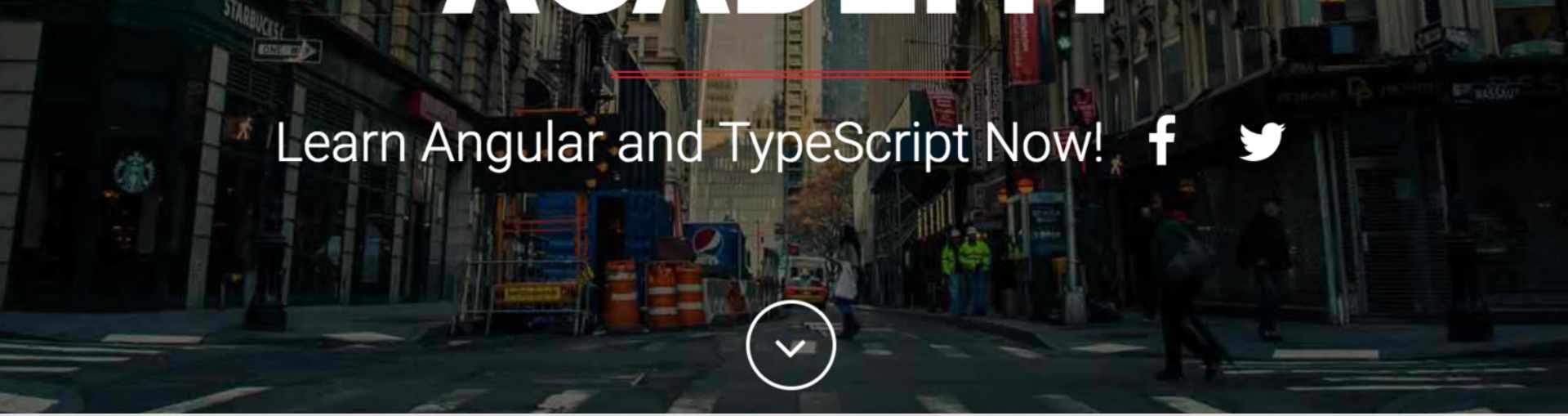
Angular Trainer

Master of Ceremonies


NG-CRUISE
— PRESENTED BY NG-CONF —
NG-CRUISE
evanue

Angular Academy





Introduction

Testing Architecture

Overview

- Does this method work?

Unit tests

- Does this feature work?

e2e Tests

- Does this product work?

Acceptance Tests

Angular CLI

ANGULAR CLI DOCUMENTATION GITHUB GET ST

```
> npm install -g @angular/cli
> ng new my-dream-app
> cd my-dream-app
> ng serve
```

Angular CLI

A command line interface for Angular

GET STARTED

ng new

The Angular CLI makes it easy to create an application that already works, right out of the box. It already follows our best practices!

Filename conventions

- app.component.ts
- app.component.spec.ts
- app.e2e.ts

```
$ npm run tests
$ npm run e2e
```

Tools Online

Plunker v0.11.4 | Save | New | Stop | gsans

▼ FILES
- config.js
- index.html
- README.md
- src/app.ts
- src/main.ts
- style.css
- New file

▼ PLUNK

Description:

Angular2 + Typescript Demo Plunk

Tags:

Enter tags

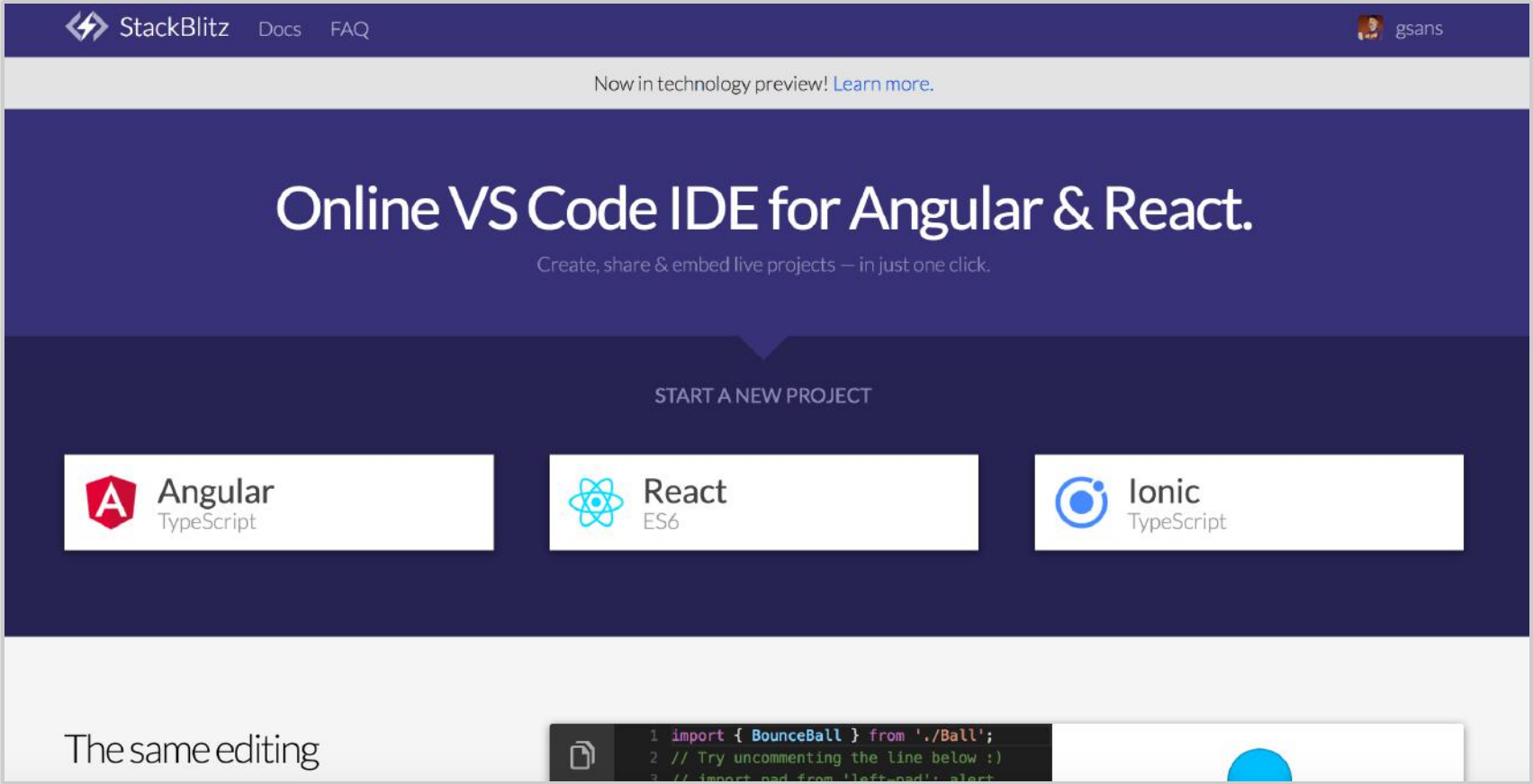
Privacy:

☑ private plunk

```
1  <!DOCTYPE html>
2  <html>
3
4    <head>
5      <base href="." />
6      <title>angular2 playground</titl
7      <link rel="stylesheet" href="styl
8      <script src="https://unpkg.com/zo
9      <script src="https://unpkg.com/re
10     <script src="https://unpkg.com/sy
11     <script src="https://unpkg.com/ty
12     <script src="config.js"></script>
13     <script>
14     System.import('app')
15       .catch(console.error.bind(consc
16   </script>
17   </head>
18
19   <body>
20     <my-app>
21     loading...
22   </my-app>
23   </body>
24
25 </html>
26
```

Hello Angular2

Tools Online



Mocks vs Stubs

Mocks

- Used to replace Complex Objects/APIs
- Examples:
  - *MockBackend*
  - *MockEventEmitter*
  - *MockLocationStrategy*

Stubs

- Used to cherry pick calls and change their behaviour for a single test
- When to use:
  - *control behaviour to favour/avoid certain path*

Jasmine

Jasmine

Main Concepts

- Suites ***describe('', function)***
- Specs ***it('', function)***
- Expectations and Matchers
  - ***expect(x).toBe(expected)***
  - ***expect(x).toEqual(expected)***

Basic Test

```
let calculator = {
  add: (a, b) => a + b
};

describe('Calculator', () => {
  it('should add two numbers', () => {
    expect(calculator.add(1,1)).toBe(2);
  })
})
```

Setup and teardown

- *beforeAll (once)*
  - *beforeEach (many)*
  - *afterEach (many)*
- *afterAll (once)*

Useful techniques

- Nesting suites and using scopes
- Utility APIs
  - *`fail(msg), pending(msg)`*
- Disable
  - *`xdescribe, xit`*
- Focused
  - `fdescribe, fit`

Jasmine Spies

Test double functions that record calls, arguments and return values

Tracking Calls

```
describe('Spies', () => {
  let calculator = { add: (a,b) => a+b };

  it('should track calls but NOT call through', () => {
    spyOn(calculator, 'add');
    let result = calculator.add(1,1);
    expect(calculator.add).toHaveBeenCalled();
    expect(calculator.add).toHaveBeenCalledTimes(1);
    expect(calculator.add).toHaveBeenCalledWith(1,1);
    expect(result).not.toEqual(2);
  })
})
```

Calling Through

```
describe('Spies', () => {
  it('should call through', () => {
    spyOn(calculator, 'add').and.callThrough();
    let result = calculator.add(1,1);
    expect(result).toEqual(2);

    //restore stub behaviour
    calculator.add.and.stub();
    expect(calculator.add(1,1)).not.toEqual(2);
  })
})
```

Set return values

```
describe('Spies', () => {
  it('should return value with 42', () => {
    spyOn(calculator, 'add').and.returnValue(42);
    let result = calculator.add(1,1);
    expect(result).toEqual(42);
  })

  it('should return values 1, 2, 3', () => {
    spyOn(calculator, 'add').and.returnValues(1, 2, 3);
    expect(calculator.add(1,1)).toEqual(1);
    expect(calculator.add(1,1)).toEqual(2);
    expect(calculator.add(1,1)).toEqual(3);
  })
})
```

Set fake function

```
describe('Spies', () => {
  it('should call fake function returning 42', () => {
    spyOn(calculator, 'add').and.callFake((a,b) => 42);
    expect(calculator.add(1,1)).toEqual(42);
  })
})
```

Error handling

```
describe('Spies', () => {
  it('should throw with error', () => {
    spyOn(calculator, 'add').and.throwError("Ups");
    expect(() => calculator.add(1,1)).toThrowError("Ups")
  })
})
```

Creating Spies

```
describe('Spies', () => {
  it('should be able to create a spy manually', () => {
    let add = jasmine.createSpy('add');
    add();
    expect(add).toHaveBeenCalled();
  })
})

// usage: create spy to use as a callback
//  setTimeout(add, 100);
```

Creating Spies

```
describe('Spies', () => {
  it('should be able to create multiple spies manually',
    let calculator = jasmine.createSpyObj('calculator', [
    calculator.add.and.returnValue(42);

    let result = calculator.add(1,1);
    expect(calculator.add).toHaveBeenCalled();
    expect(result).toEqual(42);
  })
})
```

Angular Testing

Testing APIs

- *inject,TestBed*
- *async*
- *fakeAsync/tick*

Setup

```
import { TestBed } from '@angular/core/testing';
import {
  BrowserDynamicTestingModule,
  platformBrowserDynamicTesting
} from '@angular/platform-browser-dynamic/testing';

TestBed.initTestEnvironment(
  BrowserDynamicTestingModule,
  platformBrowserDynamicTesting()
);
```

Testing a Service

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

@Injectable()
export class LanguagesService {
  get() {
    return ['en', 'es', 'fr'];
  }
}
```

Testing a Service

```
describe('Service: LanguagesService', () => {
  //setup
  beforeEach(() => TestBed.configureTestingModule({
    providers: [ LanguagesService ]
  }));

  //specs
  it('should return available languages', inject([LanguagesService],
    let languages = service.get();
    expect(languages).toContain('en');
    expect(languages).toContain('es');
    expect(languages).toContain('fr');
    expect(languages.length).toEqual(3);
  });
});
```

refactoring inject

```
describe('Service: LanguagesService', () => {
  let service;

  beforeEach(() => TestBed.configureTestingModule({
    providers: [ LanguagesService ]
  }));
  beforeEach(inject([LanguagesService], s => {
    service = s;
  }));

  it('should return available languages', () => {
    let languages = service.get();
    expect(languages).toContain('en');
    expect(languages).toContain('es');
    expect(languages).toContain('fr');
    expect(languages.length).toEqual(3);
  });
});
```

Asynchronous Testing

Asynchronous APIs

- *Jasmine.done*
- *async*
- *fakeAsync/tick*

Http Service

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import 'rxjs/add/operator/map';

@Injectable()
export class UsersService {
  constructor(private http: HttpClient) { }

  public get() {
    return this.http.get('./src/assets/users.json')
      .map(response => response.users);
  }
}
```

Testing Real Service 1/2

```
describe('Service: UsersService', () => {
  let service, http;

  beforeEach(() => TestBed.configureTestingModule({
    imports: [ HttpClientModule ],
    providers: [ UsersService ]
  }));

  beforeEach(inject([UsersService, HttpClient], (s, h) => {
    service = s;
    http = h;
  }));

  [...]
```

Testing Real Service 2/2

```
describe('Service: UsersService', () => {
  [...]

  it('should return available users (LIVE)', done => {
    service.get()
      .subscribe({
        next: res => {
          expect(res.users).toBe(USERS);
          expect(res.users.length).toEqual(2);
          done();
        }
      });
  });

});
```

Testing HttpMock 1/2

```
describe('Service: UsersService', () => {
  let service, httpMock;

  beforeEach(() => TestBed.configureTestingModule({
    imports: [ HttpClientTestingModule ],
    providers: [ UsersService ]
  }));

  beforeEach(inject([UsersService, HttpTestingController], (s, h) =>
    service = s;
    httpMock = h;
  }));

  afterEach(httpMock.verify);

  [...]
```

Testing HttpMock 2/2

```
describe('Service: UsersService', () => {
  [...]

  it('should return available users', done => {
    service.get()
      .subscribe({
        next: res => {
          expect(res.users).toBe(USERS);
          expect(res.users.length).toEqual(2);
          done();
        }
      });
    httpMock.expectOne('./src/assets/users.json')
      .flush(USERS);
  });

});
```

Components Testing

Greeter Component

```
import {Component, Input} from '@angular/core';

@Component({
  selector: 'greeter',    // <greeter name="Igor"></greeter>
  template: `<h1>Hello {{name}}!</h1>`
})
export class Greeter {
  @Input() name;
}
```

Testing Fixtures (sync)

```
describe('Component: Greeter', () => {
  let fixture, greeter, element, de;

  //setup
  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [ Greeter ]
    });

    fixture = TestBed.createComponent(Greeter);
    greeter = fixture.componentInstance;
    element = fixture.nativeElement;
    de = fixture.debugElement;
  });
}
```

Testing Fixtures (async)

```
describe('Component: Greeter', () => {
  let fixture, greeter, element, de;

  //setup
  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [ Greeter ],
    })
    .compileComponents() // compile external templates and css
    .then(() => {
      fixture = TestBed.createComponent(Greeter);
      greeter = fixture.componentInstance;
      element = fixture.nativeElement;
      de = fixture.debugElement;
    });
 ))
});
```

Using Change Detection

```
describe('Component: Greeter', () => {
  it('should render `Hello World!`', async(() => {
    greeter.name = 'World';
    //trigger change detection
    fixture.detectChanges();
    fixture.whenStable().then(() => {
      expect(element.querySelector('h1').innerText).toBe('Hello World
      expect(de.query(By.css('h1')).nativeElement.innerText).toBe('He
    });
  }));
}
```

Using fakeAsync

```
describe('Component: Greeter', () => {
  it('should render `Hello World!`', fakeAsync(() => {
    greeter.name = 'World';
    //trigger change detection
    fixture.detectChanges();
    //execute all pending asynchronous calls
    tick();
    expect(element.querySelector('h1').innerText).toBe('Hello World!'
    expect(de.query(By.css('h1')).nativeElement.innerText).toBe('Hell
  }));
}
```

Override Template

```
describe('Component: Greeter', () => {
  let fixture, greeter, element, de;

  //setup
  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [ Greeter ],
    })
    .compileComponents() // compile external templates and css
    .then(() => {
      TestBed.overrideTemplate(Greeter, '<h1>Hi</h1>');
      fixture = TestBed.createComponent(Greeter);
      greeter = fixture.componentInstance;
      element = fixture.nativeElement;
      de = fixture.debugElement;
    });
 ))
});
```

Shallow Testing

NO_ERRORS_SCHEMA

```
beforeEach(() => {
  TestBed.configureTestingModule({
    declarations: [ MyComponent ],
    schemas: [ NO_ERRORS_SCHEMA ]
  })
});
```

E2E Testing

End to End Testing

- Test features instead of methods
- Test as final user no Mocking
- Run on multiple browsers
- Complex to create/debug
- Resource intensive (slow)

Protractor

Protractor

- Automate browser testing
- WebDriverJS Wrapper
- ControlFlow
  - Deals with async code (zones)

Protractor

- Browser
  - browser.driver
  - browser.get(url)
- DOM
  - by.id('user')
  - element(selector).getText()

Timeouts

browser.get(url)

```
//  Error: Timed out waiting for page to load after

getPageTimeout: NEW_TIMEOUT_MS

browser.get(url, NEW_TIMEOUT_MS)
```

Angular timeout

```
// Timed out waiting for asynchronous Angular tasks
//  to finish after 11 seconds.

allScriptsTimeout: NEW_TIMEOUT_MS

this.ngZone.runOutsideAngular(() => {
  setTimeout(() => {
    // Changes here will not propagate into your vi
    this.ngZone.run(() => {
      // Run inside the ngZone to trigger change de
    });
  }, REALLY_LONG_DELAY);
});
```

disable wait

```
browser.waitForAngularEnabled(false);
browser.get('/non-angular-page.html');

browser.waitForAngularEnabled(true);
browser.get('/angular-page.html');
```

spec timeout

```
// timeout: timed out waiting for spec to complete

jasmineNodeOpts: {
  defaultTimeoutInterval: NEW_TIMEOUT_MS
}

// it(title, fn, timeout)
it('should work with long timeout', () => {
  service.isOnline().then(online => {
   expect(online).toBe(true)
  })
}, NEW_TIMEOUT_MS)
```

More?

Blog Post

Gerard Sans
Angular GDE | Coding is fun | Coded something awesome today? | Blogger Speaker Trainer Comm..
Dec 5, 2016 · 9 min read

Angular 2—Testing Guide

Nine easy-to-follow examples using TestBed, fixtures, async and fakeAsync/tick.

Examples covering

- Components, Directives, Pipes
- Services, Http, MockBackend
- Router, Observables
- Spies

stackblitz