Angular Unit Testing

An unusually fun workshop

Kiffin Gish

Devops

April 4, 2019

Agenda

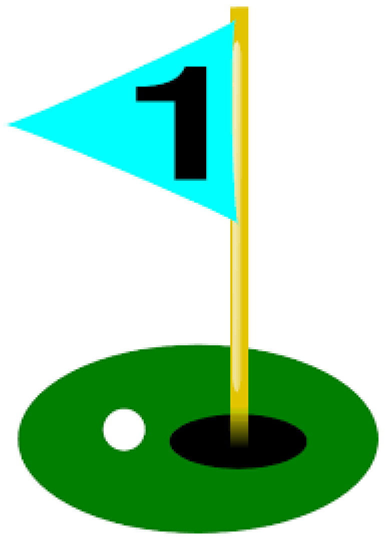
- Part I - General
  - Angular architecture
  - Testing principles
  - What is a good test?

Agenda

- Part I - General
  - Angular architecture
  - Testing principles
  - What is a good test?
- Part II – Deep dive
  - Complicated stuff

Goal

- Learn unit testing with Angular
- Write good unit tests
- Understand Isolated vs integration tests

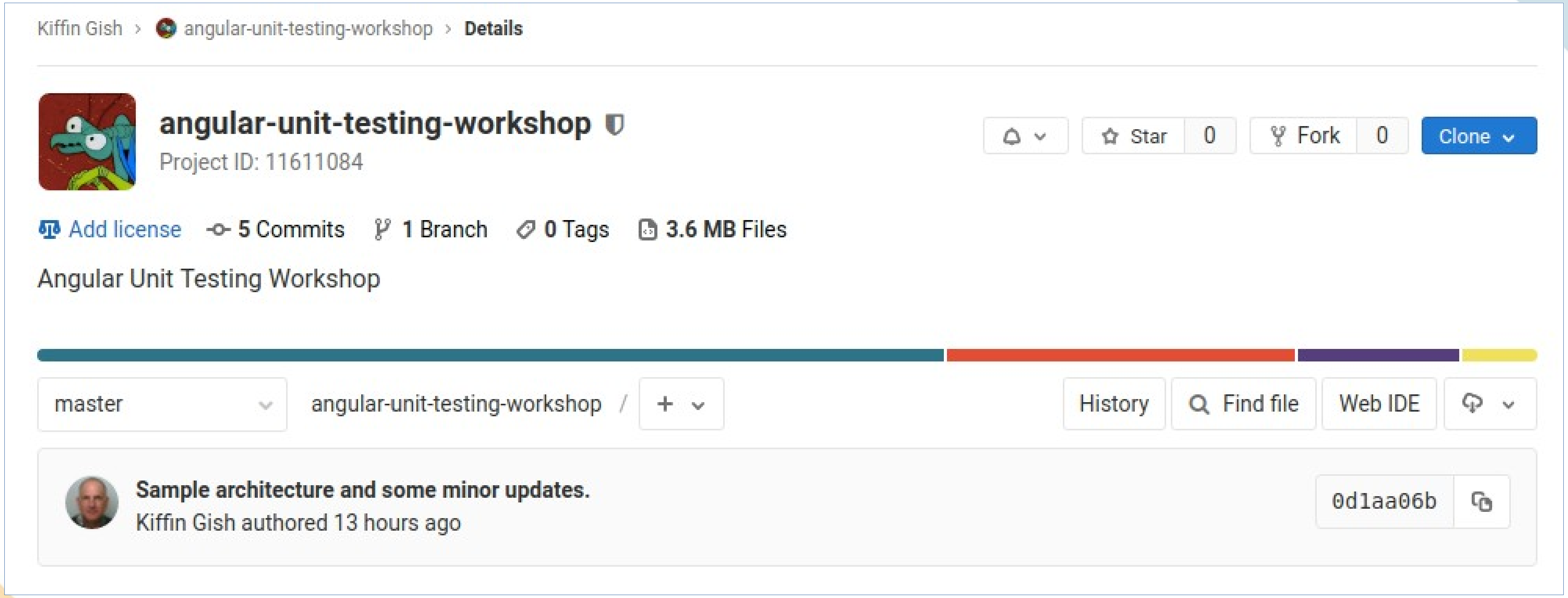


Not about

- End to end testing
- Testing tools
- Test Driven Development (TDD)

Repository

https://gitlab.com/kiffin/angular-unit-testing-workshop


Kiffin Gish > angular-unit-testing-workshop > **Details**

angular-unit-testing-workshop
Project ID: 11611084

Add license · **5** Commits · **1** Branch · **0** Tags · **3.6 MB** Files

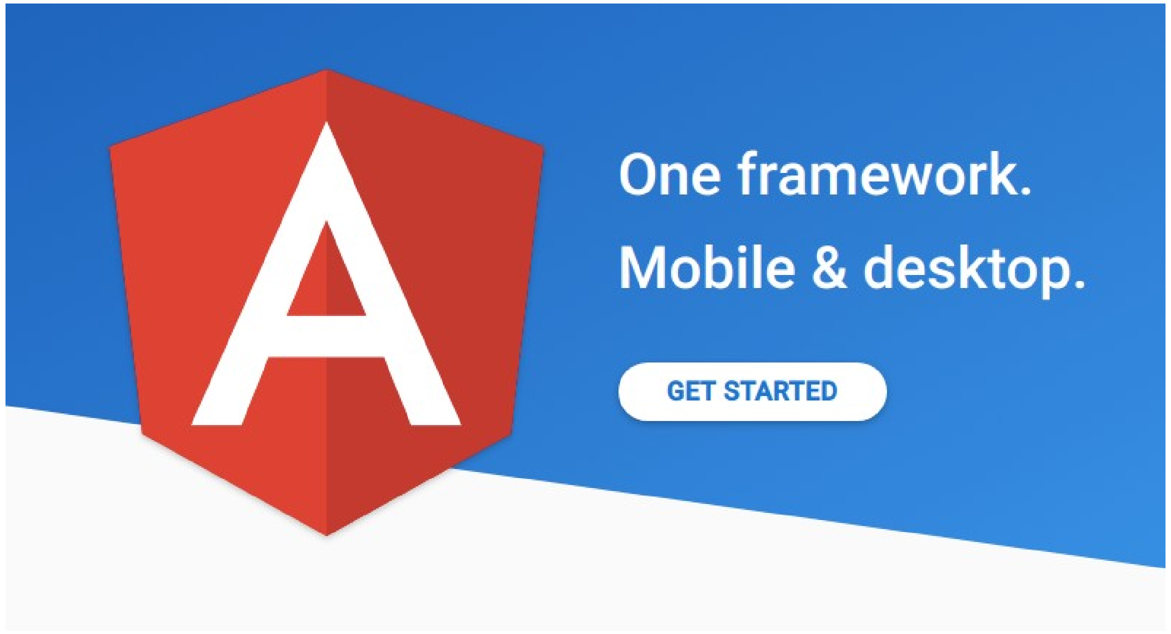
Angular Unit Testing Workshop

Sample architecture and some minor updates.
Kiffin Gish authored 13 hours ago

0d1aa06b

Angular framework

- Single page application (SPA)
- Responsive
- Theming (Material Design)
- Advanced testing



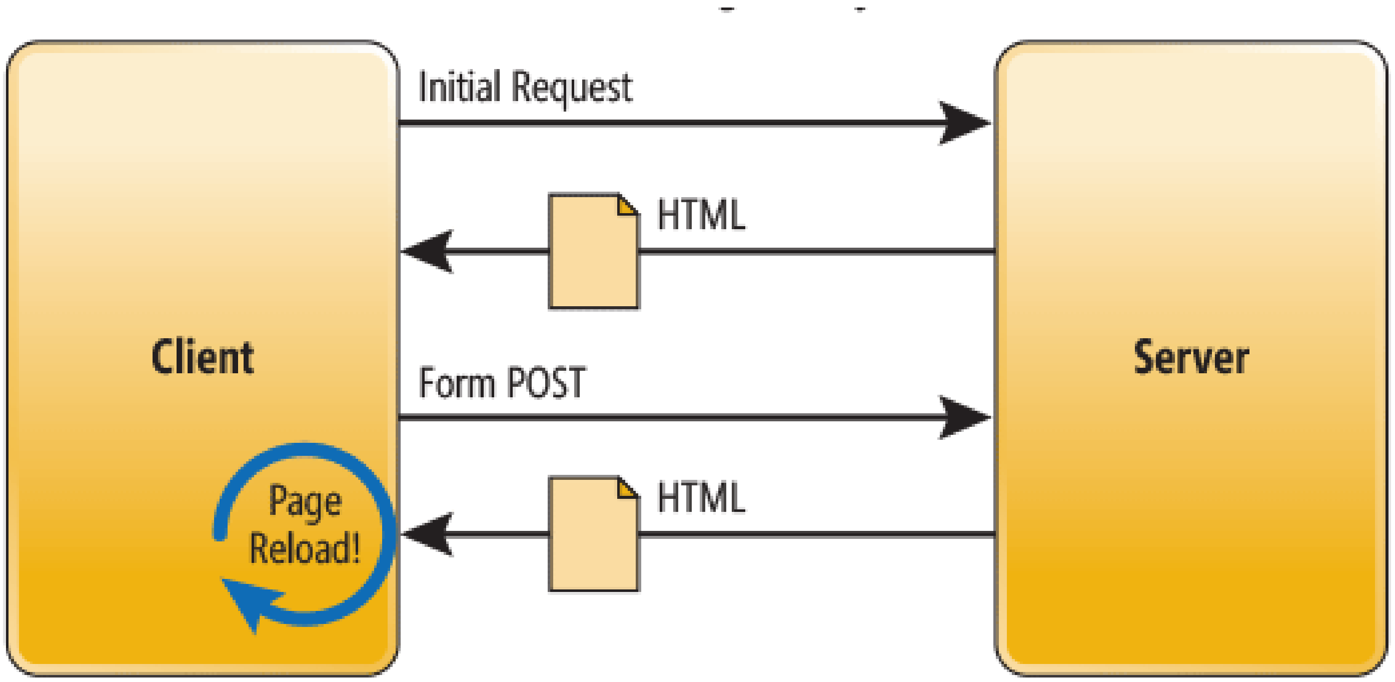
Single page application (SPA)

- HTML templates served statically
- Client retrieves data from REST services
- Views dynamically rendered
- Application bootstrapped (runs) on the client
- Interactive, quick and snappy

Single page application (SPA)

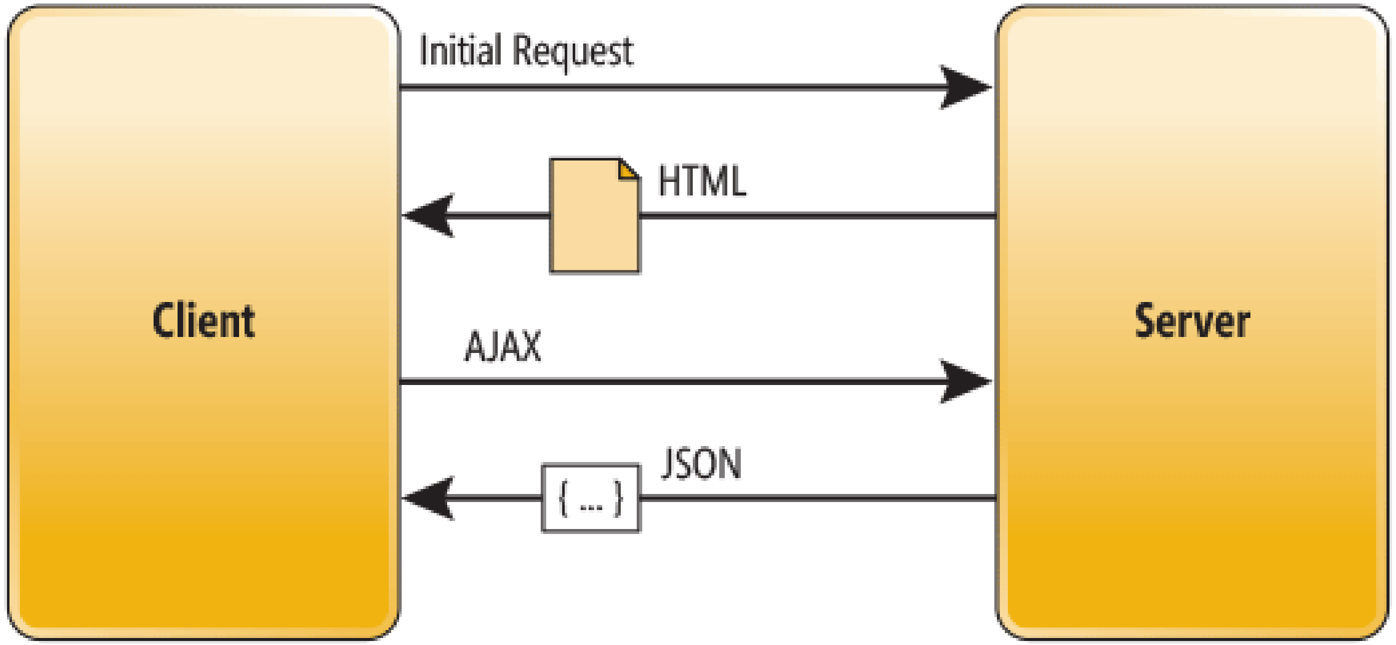
- Mobile-friendly
- More secure (HTTPS, HMAC and JWT)
- Deployment greatly simplified
- Easier to test!

Traditional page lifecycle

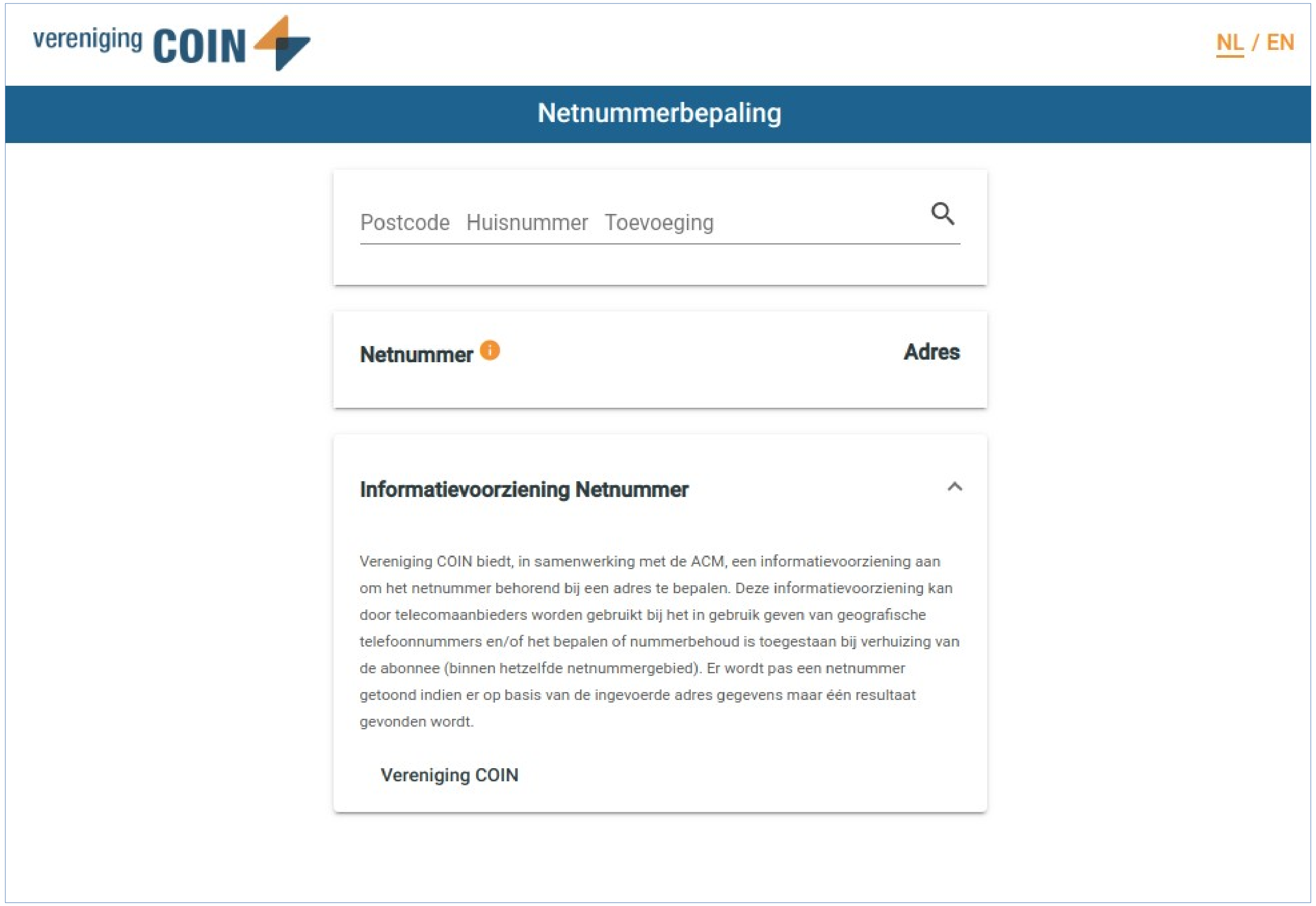


vereniging COIN

SPA lifecycle



Responsive - desktop



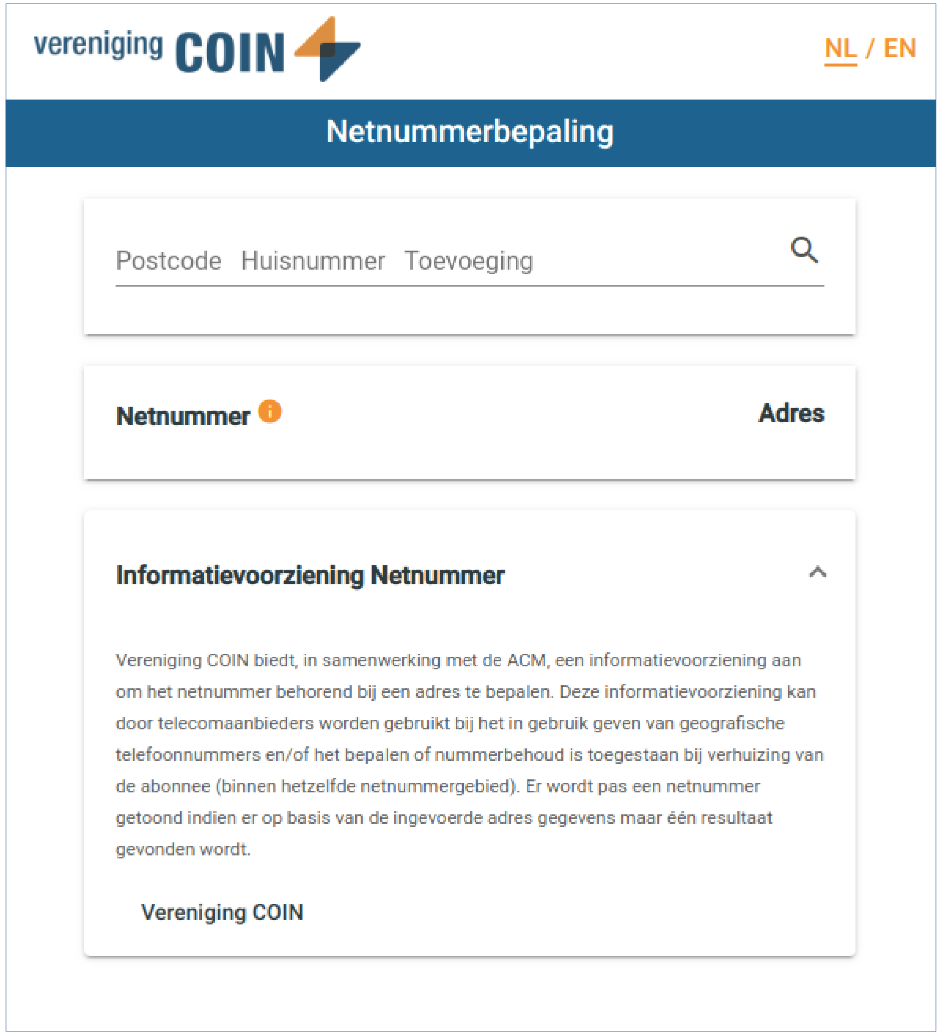
Netnummerbepaling

Informatievoorziening Netnummer

Vereniging COIN biedt, in samenwerking met de ACM, een informatievoorziening aan om het netnummer behorend bij een adres te bepalen. Deze informatievoorziening kan door telecomaanbieders worden gebruikt bij het in gebruik geven van geografische telefoonnummers en/of het bepalen of nummerbehoud is toegestaan bij verhuizing van de abonnee (binnen hetzelfde netnummergebied). Er wordt pas een netnummer getoond indien er op basis van de ingevoerde adres gegevens maar één resultaat gevonden wordt.

Vereniging COIN

Responsive - tablet



vereniging COIN

NL / EN

Netnummerbepaling

Postcode Huisnummer Toevoeging

Netnummer

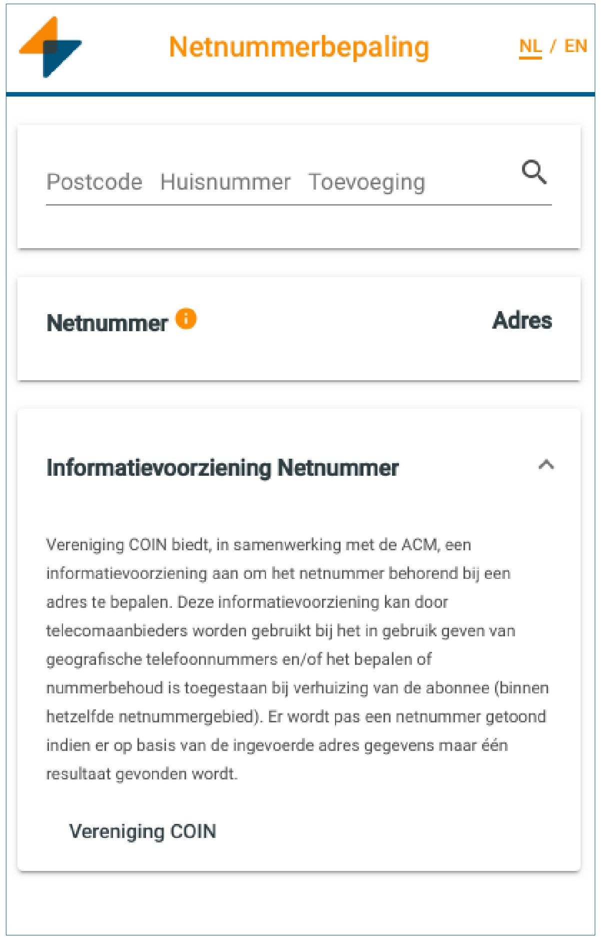
Adres

Informatievoorziening Netnummer

Vereniging COIN biedt, in samenwerking met de ACM, een informatievoorziening aan om het netnummer behorend bij een adres te bepalen. Deze informatievoorziening kan door telecomaanbieders worden gebruikt bij het in gebruik geven van geografische telefoonnummers en/of het bepalen of nummerbehoud is toegestaan bij verhuizing van de abonnee (binnen hetzelfde netnummergebied). Er wordt pas een netnummer getoond indien er op basis van de ingevoerde adres gegevens maar één resultaat gevonden wordt.

Vereniging COIN

Responsive - smartphone



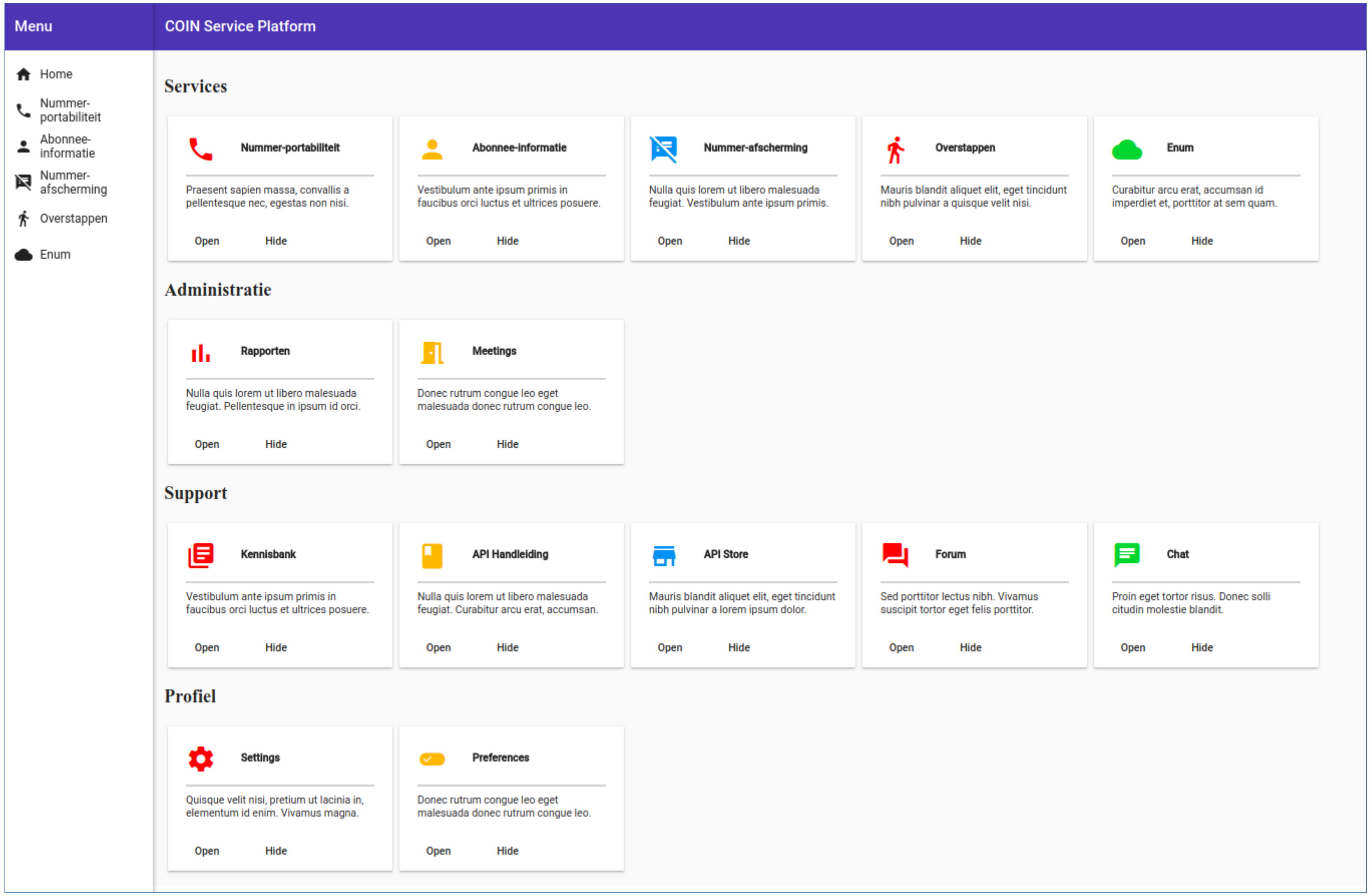
Informatievoorziening Netnummer

Vereniging COIN biedt, in samenwerking met de ACM, een informatievoorziening aan om het netnummer behorend bij een adres te bepalen. Deze informatievoorziening kan door telecomaanbieders worden gebruikt bij het in gebruik geven van geografische telefoonnummers en/of het bepalen of nummerbehoud is toegestaan bij verhuizing van de abonnee (binnen hetzelfde netnummergebied). Er wordt pas een netnummer getoond indien er op basis van de ingevoerde adres gegevens maar één resultaat gevonden wordt.

Vereniging COIN

vereniging COIN

COIN Portal



Menu

COIN Service Platform

Home
Nummer-portabiliteit
Abonnee-informatie
Nummer-afscherming
Overstappen
Enum

Services

Nummer-portabiliteit
Praesent sapien massa, convallis a pellentesque nec, egestas non nisi.
Open Hide

Abonnee-informatie
Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere.
Open Hide

Nummer-afscherming
Nulla quis lorem ut libero malesuada feugiat. Vestibulum ante ipsum primis.
Open Hide

Overstappen
Mauris blandit aliquet elit, eget tincidunt nibh pulvinar a quisque velit nisi.
Open Hide

Enum
Curabitur arcu erat, accumsan id imperdiet et, porttitor at sem quam.
Open Hide

Administratie

Rapporten
Nulla quis lorem ut libero malesuada feugiat. Pellentesque in ipsum id orci.
Open Hide

Meetings
Donec rutrum congue leo eget malesuada donec rutrum congue leo.
Open Hide

Support

Kennisbank
Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere.
Open Hide

API Handleiding
Nulla quis lorem ut libero malesuada feugiat. Curabitur arcu erat, accumsan.
Open Hide

API Store
Mauris blandit aliquet elit, eget tincidunt nibh pulvinar a lorem ipsum dolor.
Open Hide

Forum
Sed porttitor lectus nibh. Vivamus suscipit tortor eget felis porttitor.
Open Hide

Chat
Proin eget tortor risus. Donec solli citudin molestie blandit.
Open Hide

Profiel

Settings
Quisque velit nisi, pretium ut lacinia in, elementum id enim. Vivamus magna.
Open Hide

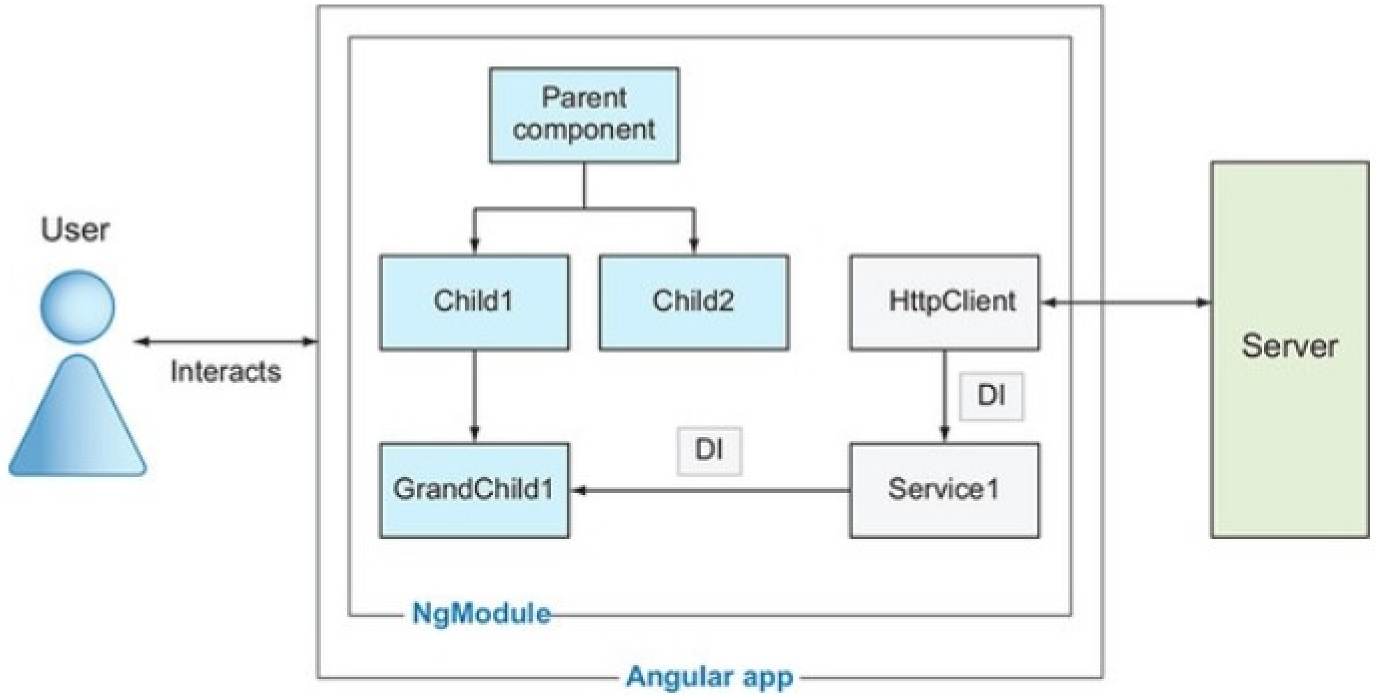
Preferences
Donec rutrum congue leo eget malesuada donec rutrum congue leo.
Open Hide

vereniging COIN

Angular architecture

- Layered (separation of concerns)
- Modules and components
- Web components
- Templates, directives and data-binding
- Data down ↓, actions up ↑
- Services and Dependency Injection (DI)

Modules and components



Web components

- Custom elements
- Shadow DOM[1]
- HTML Templates
- Composition and slots

```
<html>
    <my-component>
        <h1><slot name="title"></slot></h1>
        <p><slot name="description"></slot></p>
    </my-component>
</html>
```

[1]DOM - The Document Object Model is a programming interface for HTML and XML documents.

vereniging COIN

Web components

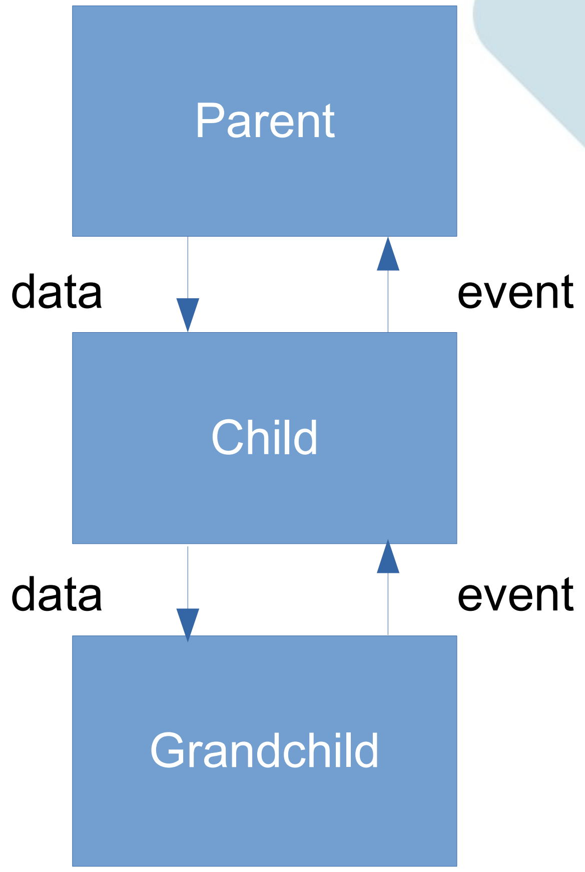
- Sub-class of HTML Element
- Constructor
- Properties (state)
- Methods

```
class MyComponent extends HTMLElement {

  count = 0

  constructor() {}

  addBy(n) {
    count += n
  }

}
```

Data down ↓ actions up↑

- Consistent interfaces
- Reusable tests

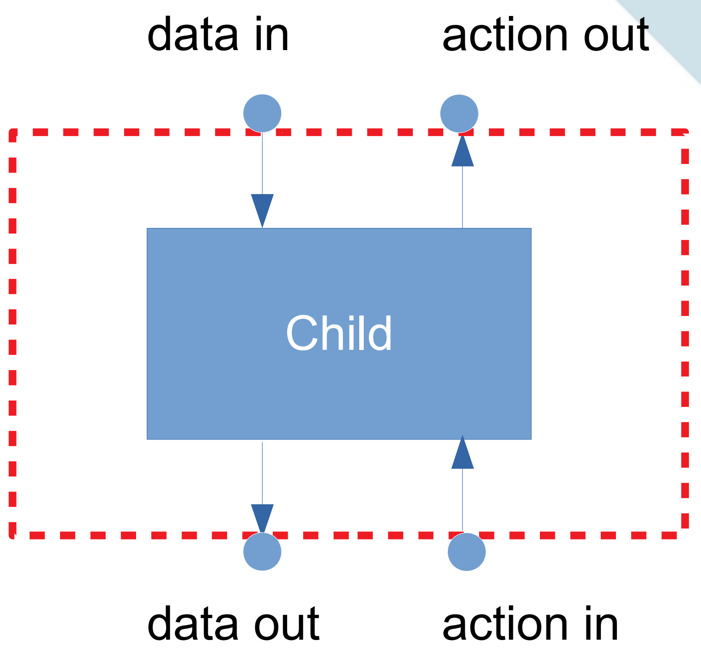
```
class MyComponent extends HTMLElement {

  @Input stock
  @Output stockValueChange = new EventEmitter()

  add(n) {
    stock += n
    StockValueChange.emit(stock)
  }
}
```



Data down ↓ actions up↑

- Consistent interfaces
- Reusable tests

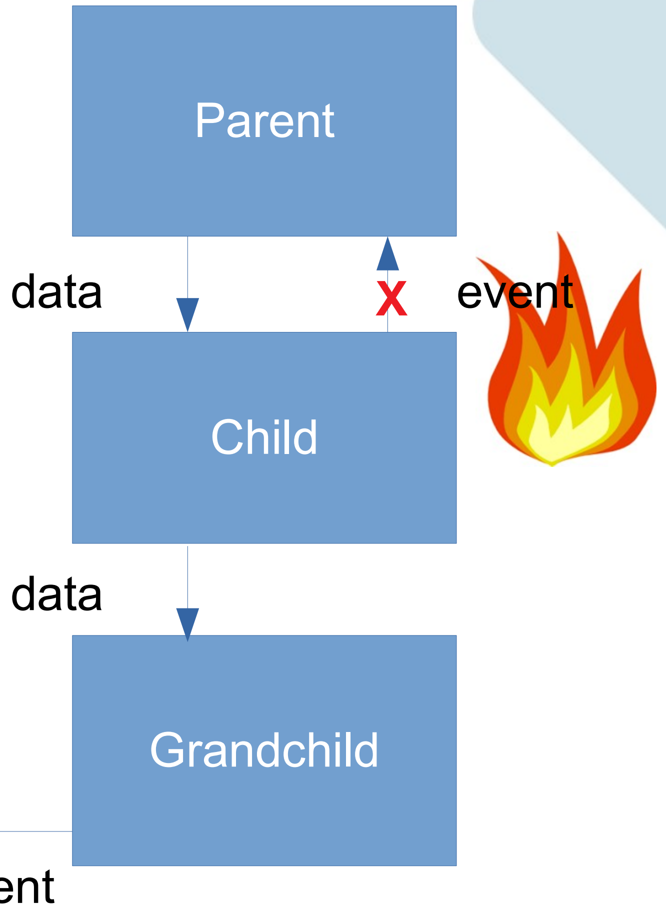
```
class MyComponent extends HTMLElement {

  @Input stock
  @Output stockValueChange = new EventEmitter()

  add(n) {
    stock += n
    StockValueChange.emit(stock)
  }
}
```



vereniging COIN

Data down ↓ actions up↑

- Breaking this is like playing with fire.
- You will be punished!



Dependency injection (DI)

- Angular has its own DI framework
- Coding pattern for testing and maintainability
- Service injection, where service is:
  - Application state
  - Storage
  - Auth0 (refresh)
  - Messaging
  - HTTP

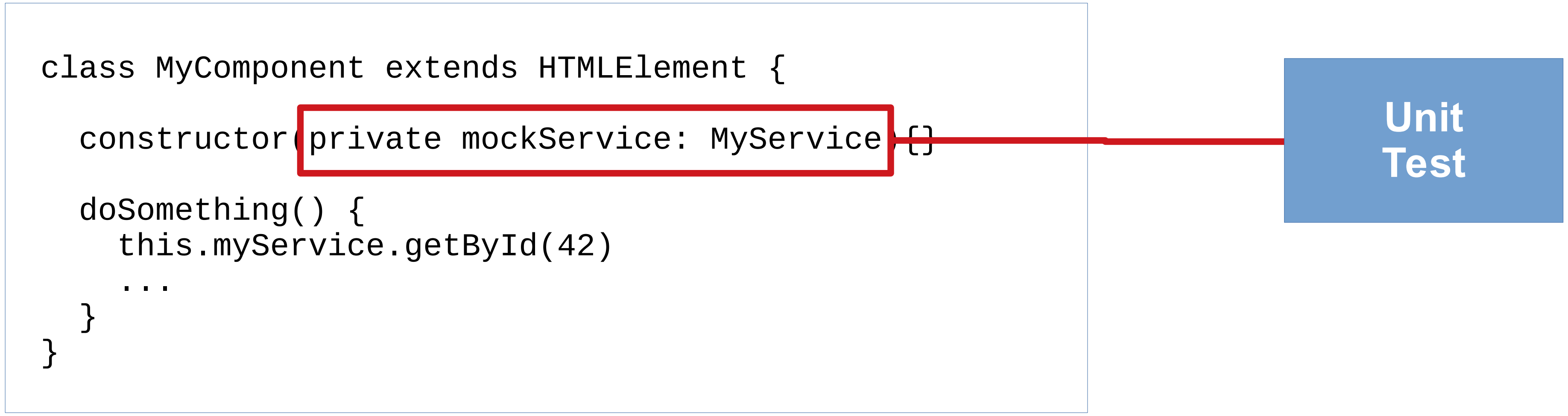
Dependency injection (DI)

- Angular has its own DI framework
- Coding pattern for testing and maintainability
- Dependencies from external source rather than creating itself

```
class MyComponent extends HTMLElement {

  constructor(private myService: MyService){}

  doSomething() {
    this.myService.getById(42)
    ...
  }
}
```

Dependency injection (DI)

- Angular has its own DI framework
- Coding pattern for testing and maintainability
- Dependencies from external source rather than creating itself



```
class MyComponent extends HTMLElement {

  constructor(private mockService: MyService){}

  doSomething() {
    this.myService.getById(42)
    ...
  }
}
```

Testing with DI

- Looks like a duck, walks like a duck and quacks like a duck …

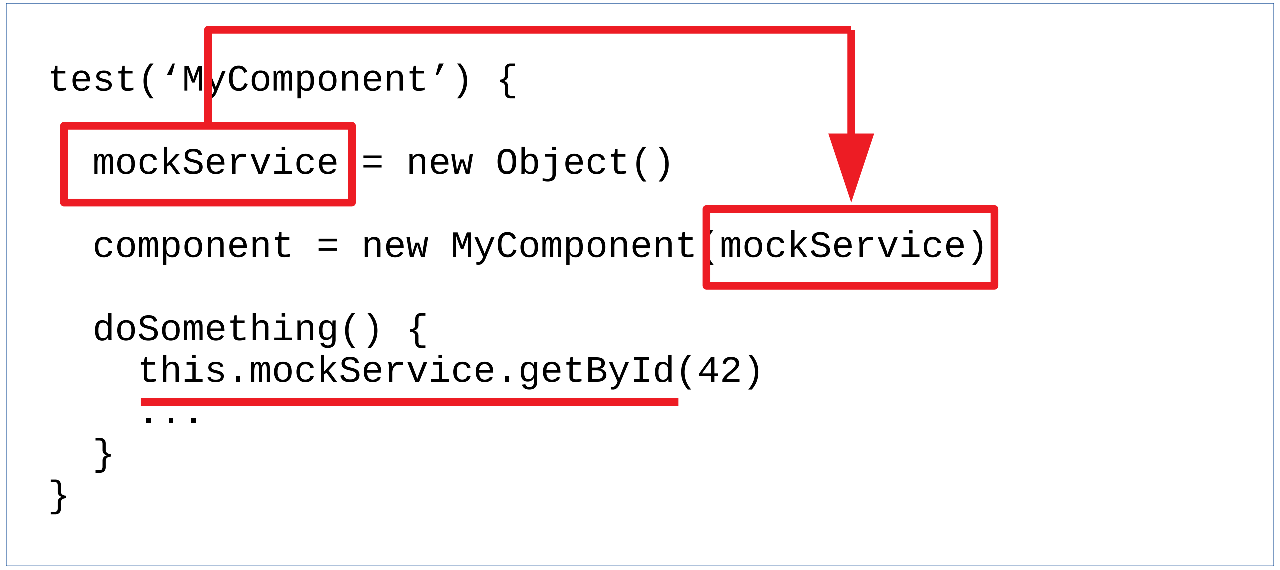
Testing with DI

- Looks like a duck, walks like a duck and quacks like a duck ...

```
test(‘MyComponent’) {

  mockService = new Object()

  component = new MyComponent(mockService)

  doSomething() {
    this.mockService.getById(42)
    ...
  }
}
```

Testing with DI

- Looks like a duck, walks like a duck and quacks like a duck ...

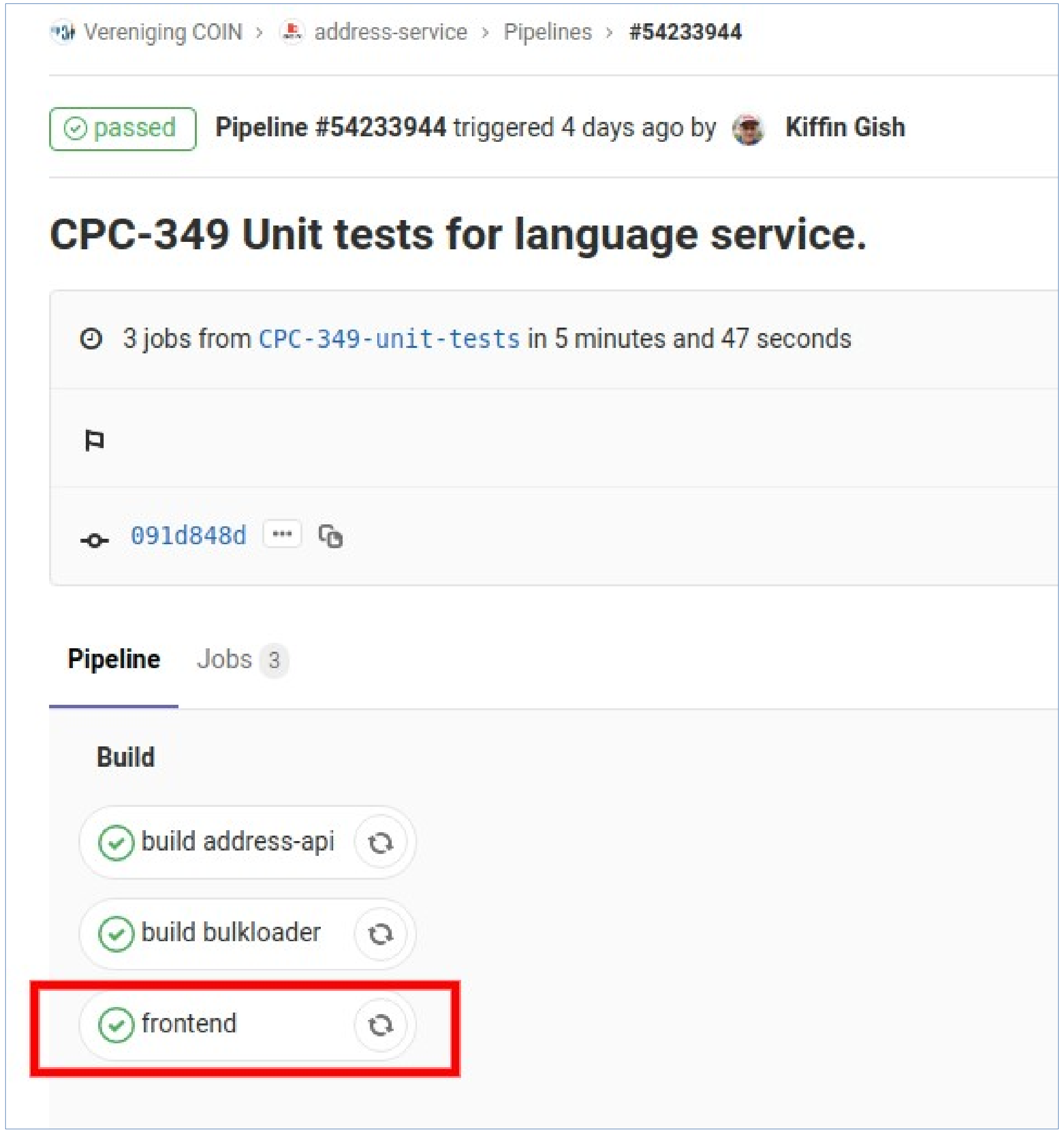


```
test(‘MyComponent’) {

  mockService = new Object()

  component = new MyComponent(mockService)

  doSomething() {
    this.mockService.getById(42)
    ...
  }
}
```

Automated testing

- Unit testing
- End to end testing
- Integration or functional testing

CI/CD pipeline


Vereniging COIN › address-service › Pipelines › **#54233944**

passed **Pipeline #54233944** triggered 4 days ago by **Kiffin Gish**

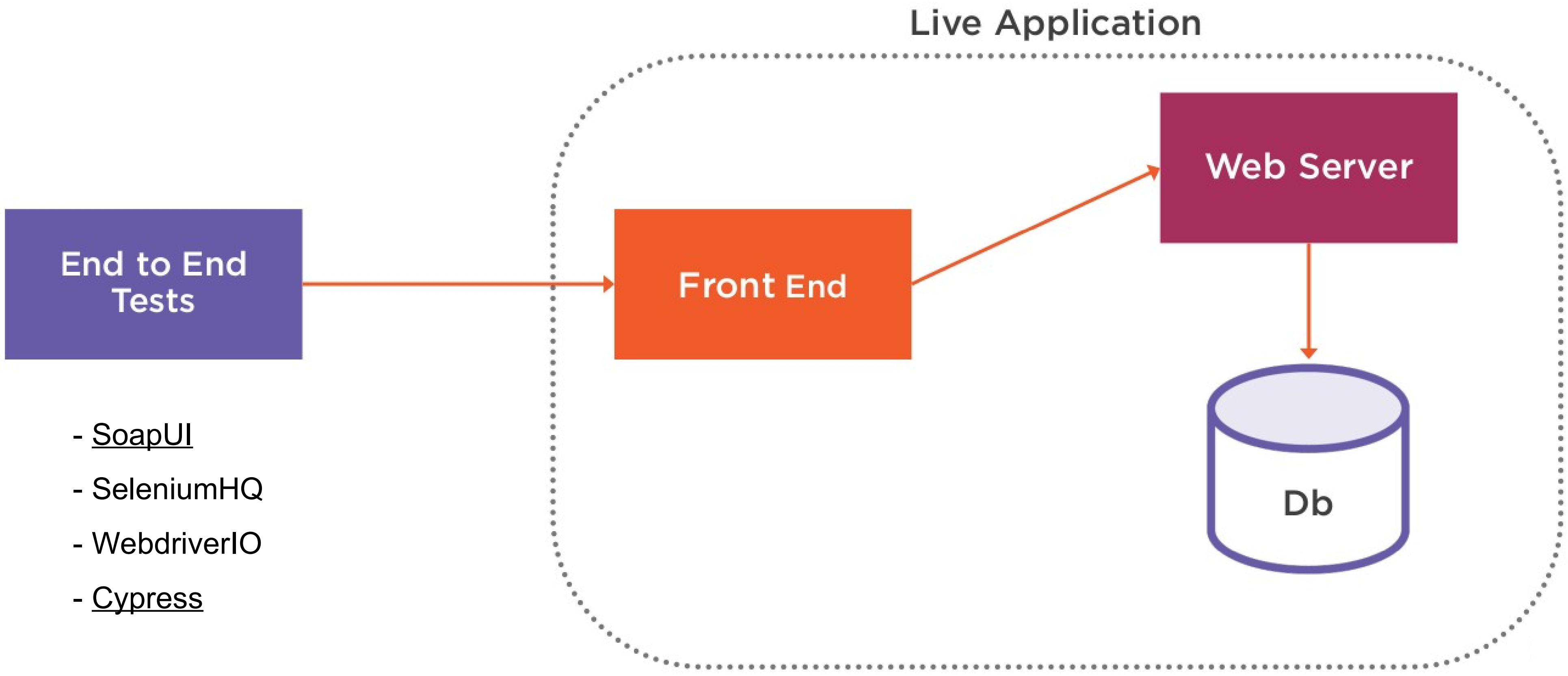
CPC-349 Unit tests for language service.

3 jobs from `CPC-349-unit-tests` in 5 minutes and 47 seconds

`091d848d`

End to end testing (E2E)

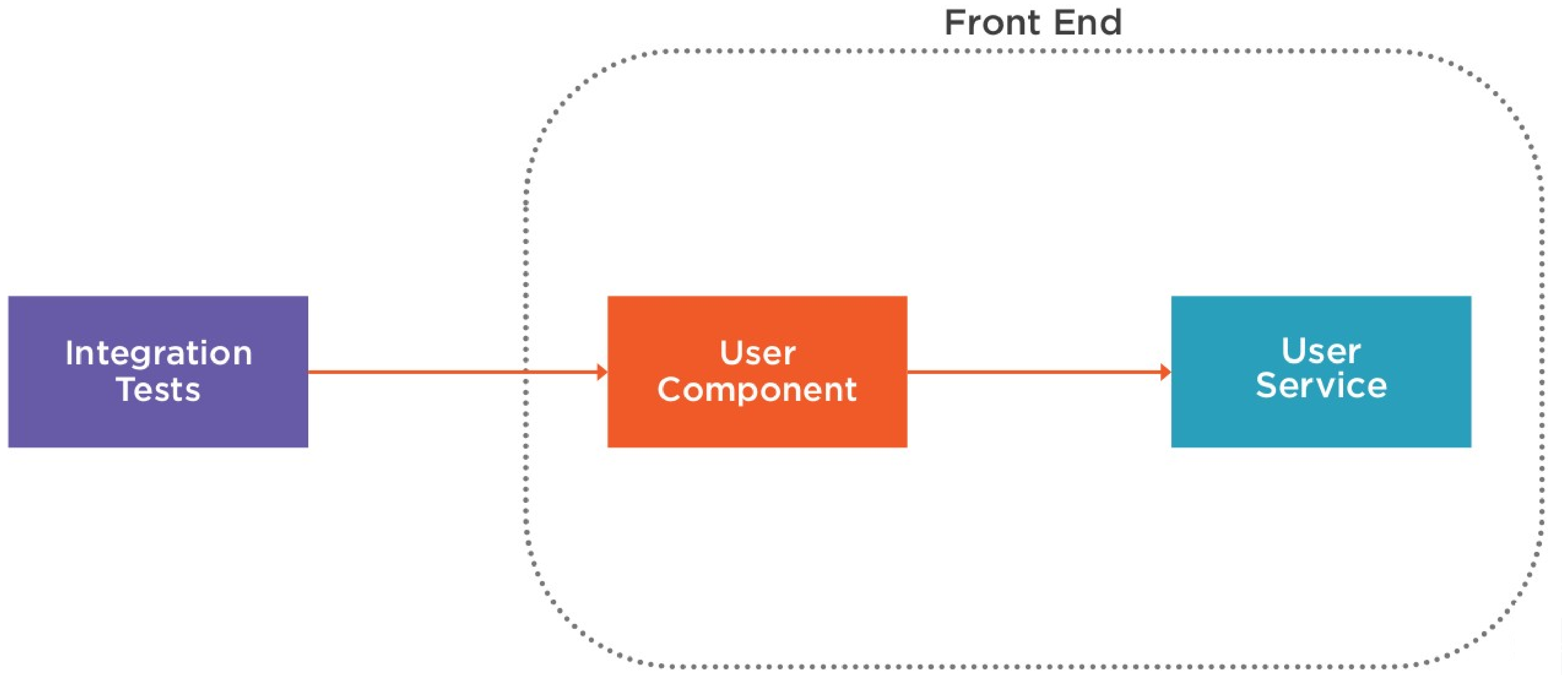
- Live running application
- Tests exercise actual application



- SoapUI
- SeleniumHQ
- WebdriverIO
- Cypress

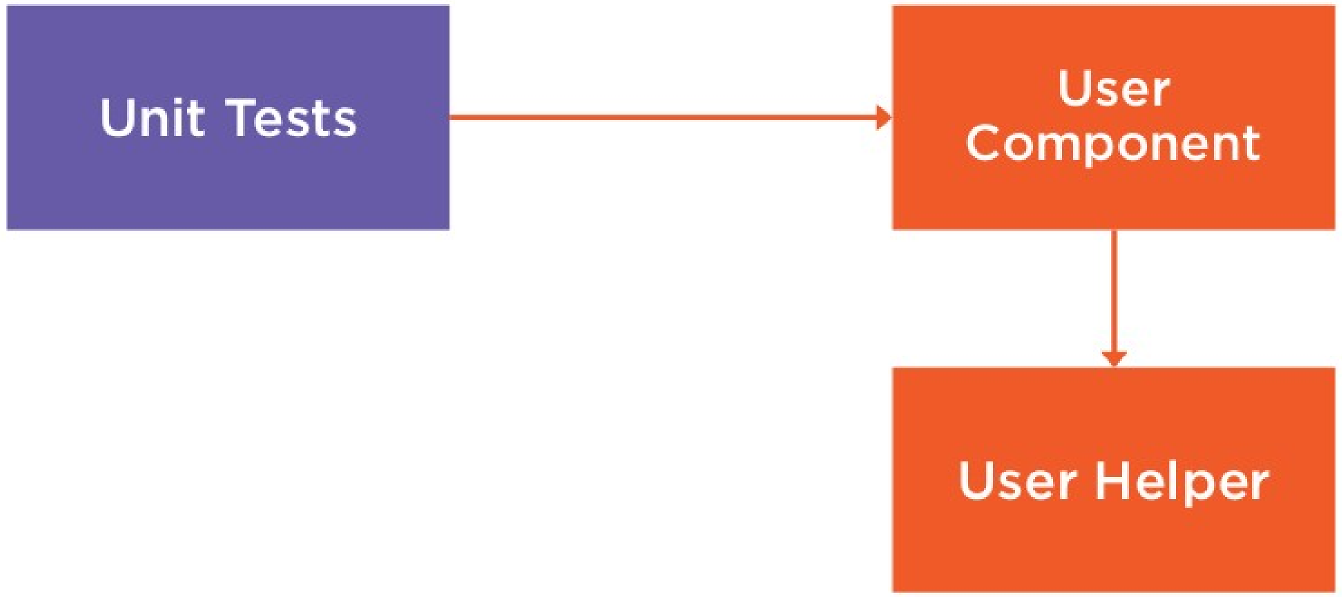
Integration and functional testing

- More than a unit, less than the complete application

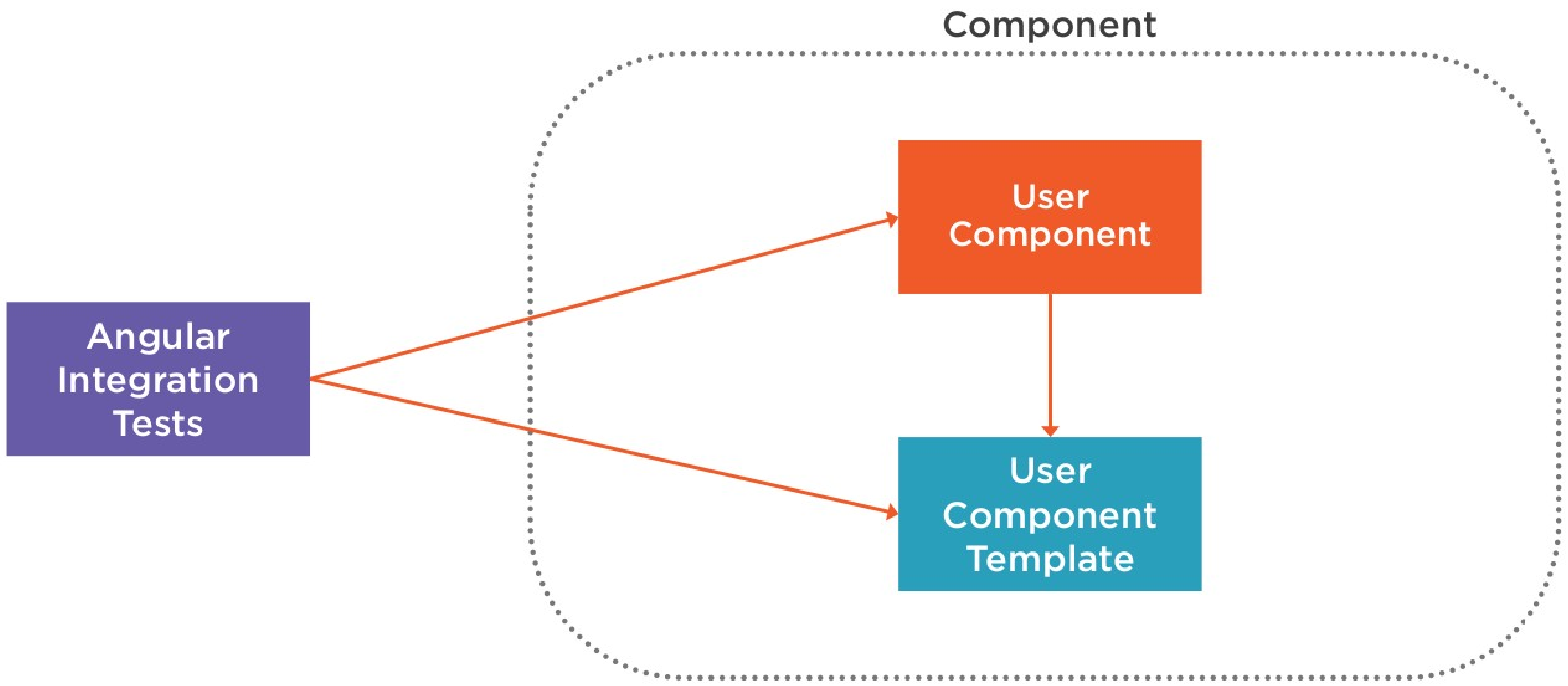


Unit testing

- A single unit of code



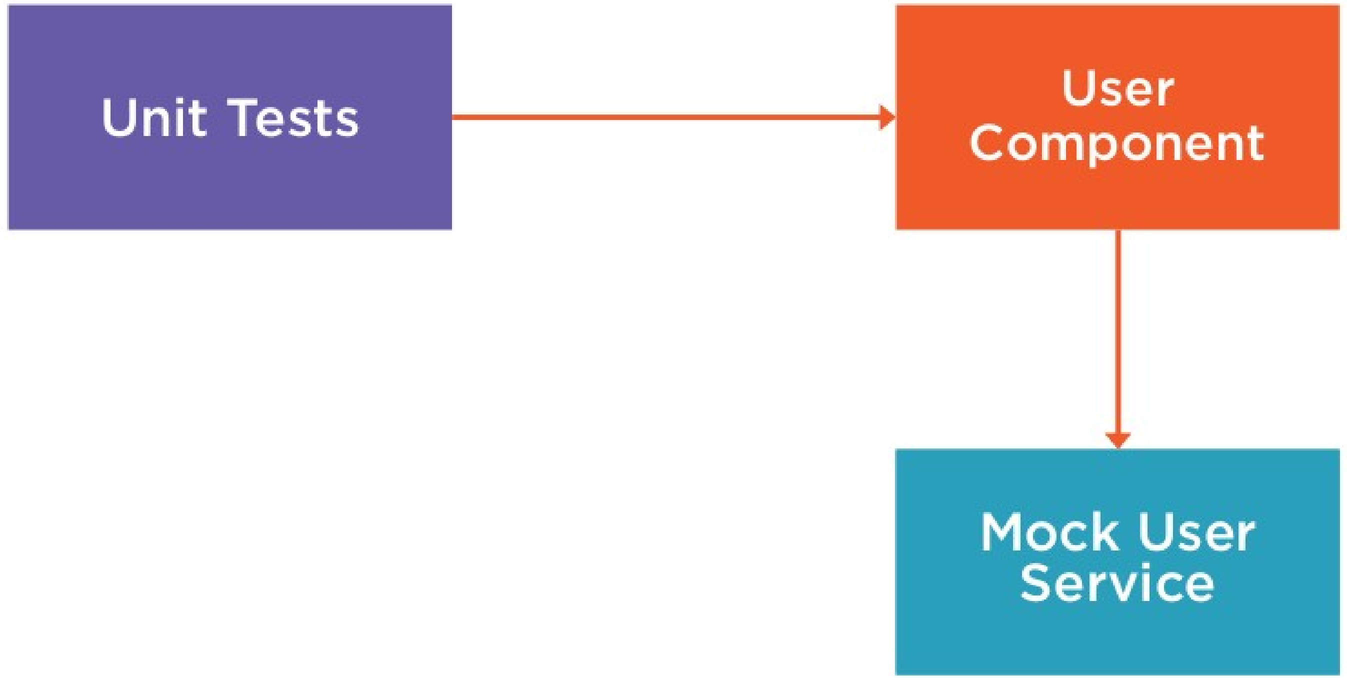
Angular integration testing



Mocking

- Instead of using the “real” service we provide a “mock” service
- A mock is a class that looks like the real class
- However, we can control what it does, what its methods return, and we can ask it questions about what methods were called during a test.

Mocking



Types of mocks

- Dummies – just objects that do nothing
- Stubs – object with controllable behavior
- Spies – keeps track of methods called
- True mocks – complex objects that verify being used exactly in a very specific way (HTTP).

Types of unit tests in Angular

- Isolated – exercise a single unit of code
- Integration – component in context of specific module
  - Shallow – single component
  - Deep – parent and child components

Angular tools

- Karma – Test runner for browsers and devices
- Jasmine – JavaScript testing framework

```
expect(array).toContain(member)
expect(fn).toThrow(string)
expect(fn).toThrowError(string)
expect(instance).toBe(instance)
expect(mixed).toBeDefined()
expect(mixed).toBeFalsy()
expect(mixed).toBeNull()
expect(mixed).toBeTruthy()
expect(mixed).toBeUndefined()
expect(mixed).toEqual(mixed)
expect(mixed).toMatch(pattern)
expect(number).toBeCloseTo(number, decimalPlaces)
expect(number).toBeGreaterThan(number)
expect(number).toBeLessThan(number)
expect(number).toBeNaN()
expect(spy).toHaveBeenCalled()
expect(spy).toHaveBeenCalledTimes(number)
expect(spy).toHaveBeenCalledWith(...arguments)
...
```

vereniging COIN

Other unit testing tools

- Jest
- Mocha, Chai
- Sinon
- TestDouble
- Wallaby
- Cypress
- Ad infinitum ...

AAA Pattern

- **<u>A</u>rrange** all necessary preconditions and inputs
- **<u>A</u>ct** on the object or class under tests
- **<u>A</u>ssert** that the expected results have occurred

Code example

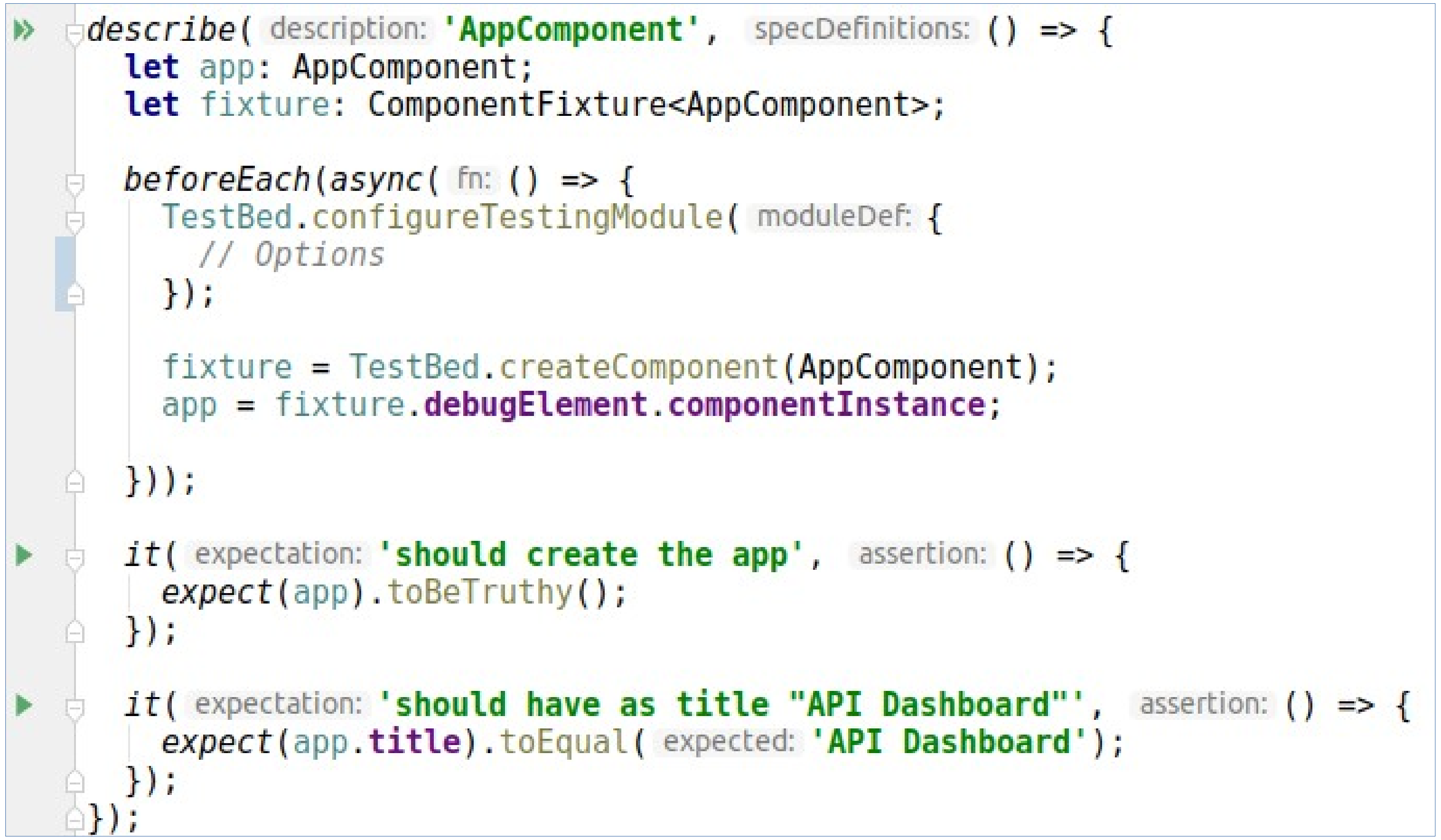
```
// Arrange
let repository = Substitute.For<IClientRepository>();
let client = new Client(repository);
client.deposit(1000);

// Act
client.charge(250);

// Assert
expect(client.amount()).toEqual(750);
```

vereniging COIN

DAMP versus DRY

- **DRY** (<u>D</u>on’t <u>R</u>epeat <u>Y</u>ourself)
  - Remove all duplication
- **DAMP** (<u>D</u>escriptive <u>A</u>nd <u>M</u>aintainable <u>P</u>rocedures)
  - Repeat yourself if necessary

App.component.spec.ts



```
describe( description: 'AppComponent',  specDefinitions: () => {
  let app: AppComponent;
  let fixture: ComponentFixture<AppComponent>;

  beforeEach(async( fn: () => {
    TestBed.configureTestingModule( moduleDef: {
      // Options
    });

    fixture = TestBed.createComponent(AppComponent);
    app = fixture.debugElement.componentInstance;

  }));

  it( expectation: 'should create the app',  assertion: () => {
    expect(app).toBeTruthy();
  });

  it( expectation: 'should have as title "API Dashboard"',  assertion: () => {
    expect(app.title).toEqual( expected: 'API Dashboard');
  });
});
```

Tell the story

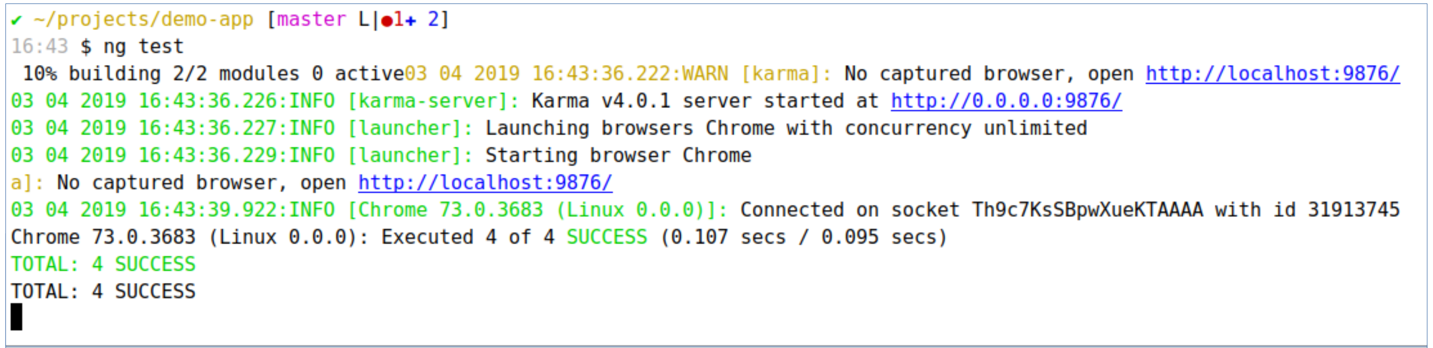
“First, we set up whatever we need to set up, then we make a change, and then we check that the change happened in the way that we expected...”

“Another way that we look at this is that we set an initial state, we change the state, and then we check to make sure that the new state is correct...”

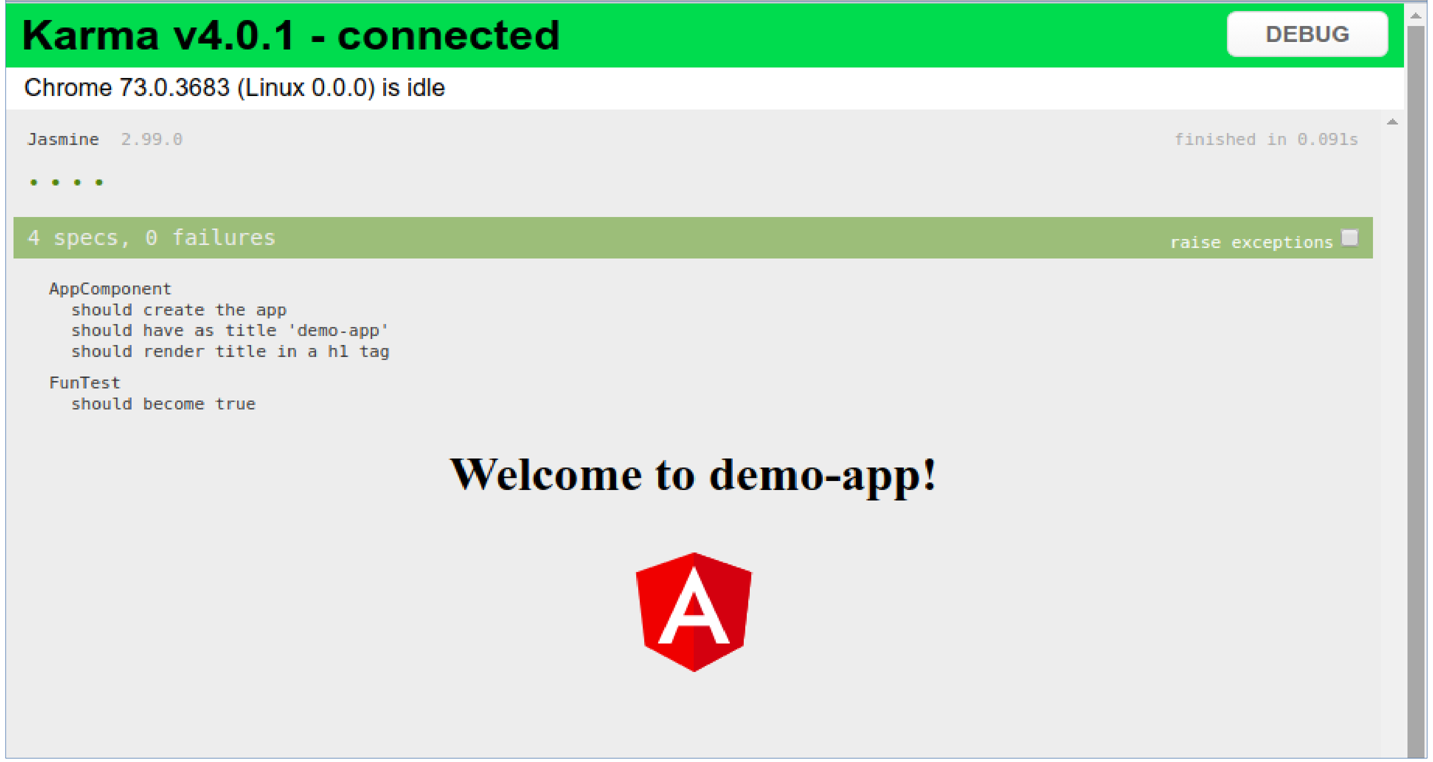
Tell the story

- A test should be a complete story within the `` `it()` ``
- Shouldn’t need to look around much to understand test
- Put less interesting setup within the `` `beforeEach()` ``
- Keep critical setup within the `` `it()` ``
- Use <u>a</u>rrange, <u>a</u>ct and <u>a</u>ssert within the `` `it()` ``

Demo



```
✔ ~/projects/demo-app [master L|●1✚ 2]
16:43 $ ng test
 10% building 2/2 modules 0 active03 04 2019 16:43:36.222:WARN [karma]: No captured browser, open http://localhost:9876/
03 04 2019 16:43:36.226:INFO [karma-server]: Karma v4.0.1 server started at http://0.0.0.0:9876/
03 04 2019 16:43:36.227:INFO [launcher]: Launching browsers Chrome with concurrency unlimited
03 04 2019 16:43:36.229:INFO [launcher]: Starting browser Chrome
a]: No captured browser, open http://localhost:9876/
03 04 2019 16:43:39.922:INFO [Chrome 73.0.3683 (Linux 0.0.0)]: Connected on socket Th9c7KsSBpwXueKTAAAA with id 31913745
Chrome 73.0.3683 (Linux 0.0.0): Executed 4 of 4 SUCCESS (0.107 secs / 0.095 secs)
TOTAL: 4 SUCCESS
TOTAL: 4 SUCCESS
```



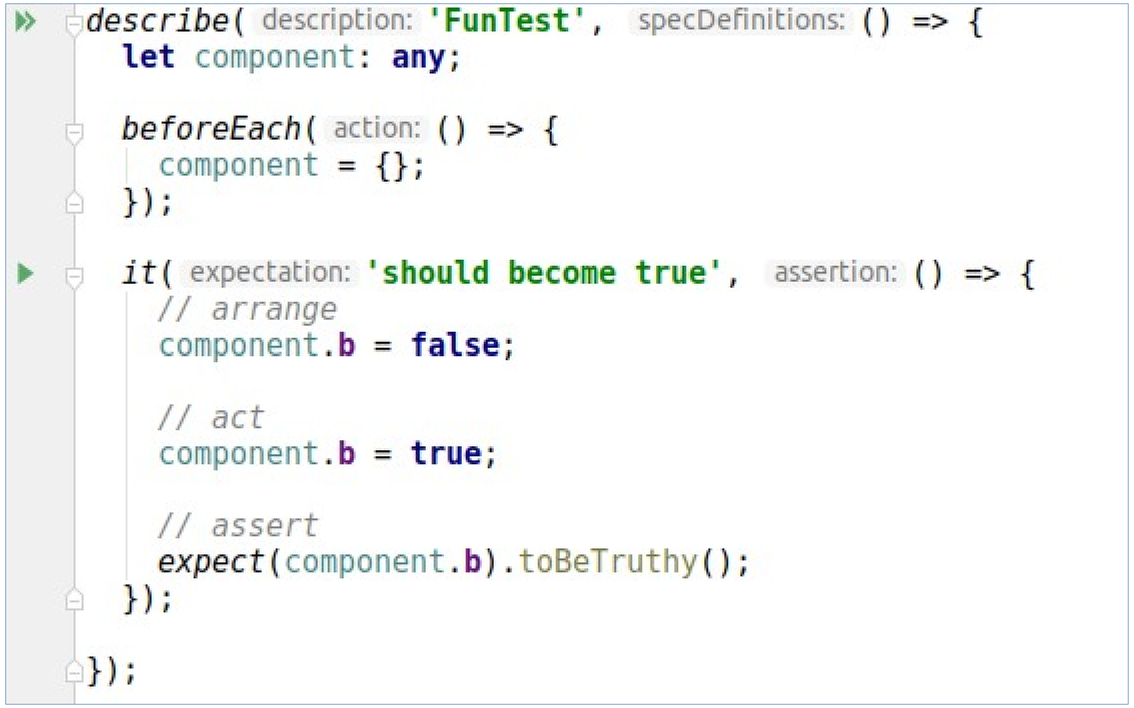
Karma v4.0.1 - connected DEBUG

Chrome 73.0.3683 (Linux 0.0.0) is idle

```
Jasmine  2.99.0                                   finished in 0.091s

• • • •

4 specs, 0 failures                               raise exceptions

  AppComponent
    should create the app
    should have as title 'demo-app'
    should render title in a h1 tag

  FunTest
    should become true
```

Welcome to demo-app!

Demo


```
describe( description: 'FunTest',  specDefinitions: () => {
  let component: any;

  beforeEach( action: () => {
    component = {};
  });

  it( expectation: 'should become true',  assertion: () => {
    // arrange
    component.b = false;

    // act
    component.b = true;

    // assert
    expect(component.b).toBeTruthy();
  });

});
```

It works !!!

Thanks and good-bye …

vereniging COIN