launch_code

- Codergirl – Frontend
- Unit 2 - Class 15
- Jan 27, 2021

AGENDA

- Consuming a service
- SASS
- Section 28 – Unit Testing.
- Studio – practice unit testing.
- Project requirements
  https://docs.google.com/document/d/1W3o35917j_T1qd8UlKgAkLz-23d07x90ayhXoKsRb0o/edit

CONSUMING A SERVICE

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

@Injectable({
  providedIn: 'root'
})

export class WeatherService {


  constructor(public httpClient: HttpClient) {
  }

  getWeather() {
    return
this.httpClient.get("http://api.weatherstack.com/current?access_key=04bd8fafbb30ee63c16f382a1d6aa404&units=f&
query=63303")
  }
}
```

SASS

- package.json
- scss - https://www.digitalocean.com/community/tutorials/using-sass-with-the-angular-cli
- https://netbasal.com/angular-cli-and-global-sass-variables-a1b92d8ca9b7

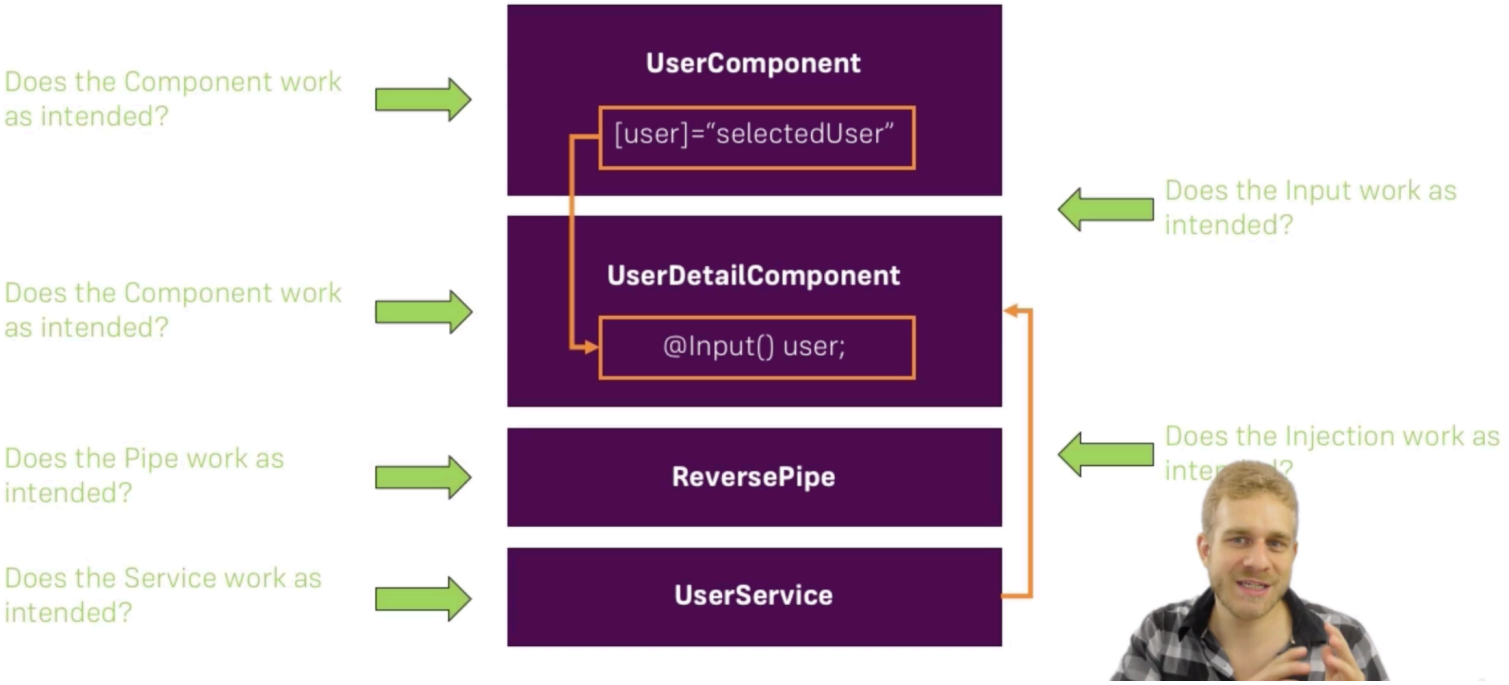
Why Unit Tests?

Guard against Breaking Changes

Analyze Code Behavior (Expected and Unexpected)

Reveal Design Mistakes

Why Unit Tests?



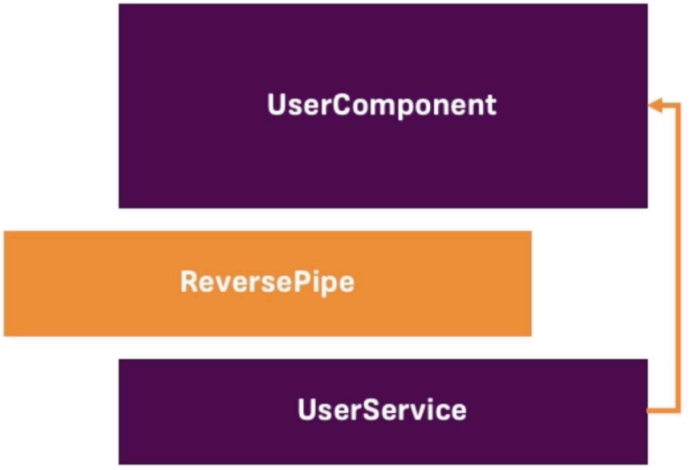
UNIT TESTING RESOURCES

- https://angular.io/guide/testing
  - The CLI takes care of Jasmine and Karma configuration for you.
  - The test file extension must be .spec.ts so that tooling can identify it as a file with tests (AKA, a *spec* file).
  - CI
  - https://jasmine.github.io/pages/getting_started.html
  - https://jasmine.github.io/tutorials/your_first_suite
- https://semaphoreci.com/community/tutorials/testing-components-in-angular-2-with-jasmine
- https://github.com/angular/angular-cli

TEST UTILITIES

- Need to get test utility setup via tutorial.
- Test Runner – Karma.
- Describe the to be tested unit.
- Unit test- Each 'it' block is a test.
- beforeEach() runs before each test.
- Test logic – Angular app with app module and app component and runs in browser.
- Testing environment is the test running it. We need to bootstrap our application. Setup app module and execute tests to what user sees.

Isolated and Non-Isolated Tests



TESTING PIPE FUNCTION

```
import { Pipe } from "@angular/core";

@Pipe({
  name: 'reverse'
})
export class ReversePipe {
  transform(value: string) {
    return value.split("").reverse().join("");
  }
}
```

```
/* tslint:disable:no-unused-variable */

import { ReversePipe } from "./reverse.pipe";
describe('Pipe: ReversePipe', () => {
  it('should reverse the inputs', () => {
    let reversePipe = new ReversePipe();
    expect(reversePipe.transform('hello')).toEqual('olleh');
  });

});
```

TEST UTILITIES

- Test for component created
- Fixture holds created component – truthy means existent
- debugElement lets us examine the component
- expect is a testing package provided by testing package such as karma or Jasmine
- Expect app to have a property of a title with a value of the existing title
- detectChanges triggers change detection
- We can expect certain elements such as `h1` element and containing some specific text value
- ng test will run the test cli

CREATING UNIT TEST

- Create a new component
- ng g c user

```
<div *ngIf="isLoggedIn">
  <h1>User logged in</h1>
  <p>User is: {{ user.name }}</p>
</div>
<div *ngIf="!isLoggedIn">
  <h1>User not logged in</h1>
  <p>Please log in first</p>
</div>
```

CREATING UNIT TEST

```
/* tslint:disable:no-unused-variable */

import { TestBed, async, fakeAsync, tick } from '@angular/core/testing';
import { UserComponent } from './user.component';
import { UserService } from "./user.service";
import { DataService } from "../shared/data.service";

describe('Component: User', () => {
  beforeEach(() => {
    TestBed.configureTestingModule({
      declarations: [UserComponent]
    });
  });

  it('should create the app', () => {
    let fixture = TestBed.createComponent(UserComponent);
    let app = fixture.debugElement.componentInstance;
    expect(app).toBeTruthy();
  });

  it('should use the user name from the service', () => {
    let fixture = TestBed.createComponent(UserComponent);
    let app = fixture.debugElement.componentInstance;
    let userService = fixture.debugElement.injector.get(UserService);
    fixture.detectChanges();
    expect(userService.user.name).toEqual(app.user.name);
  });

  it('should display the user name if user is logged in', () => {
    let fixture = TestBed.createComponent(UserComponent);
    let app = fixture.debugElement.componentInstance;
    app.isLoggedIn = true;
    fixture.detectChanges();
    let compiled = fixture.debugElement.nativeElement;
    expect(compiled.querySelector('p').textContent).toContain(app.user.name);
  });

  it('shouldn\'t display the user name if user is not logged in', () => {
    let fixture = TestBed.createComponent(UserComponent);
    let app = fixture.debugElement.componentInstance;
    fixture.detectChanges();
    let compiled = fixture.debugElement.nativeElement;
    expect(compiled.querySelector('p').textContent).not.toContain(app.user.name);
  });

  it('shouldn\'t fetch data successfully if not called asynchronously', () => {
    let fixture = TestBed.createComponent(UserComponent);
    let app = fixture.debugElement.componentInstance;
```

CREATING USER SERVICE TEST

```
export class UserService {
  user = {
    name: 'Max'
  };
}
```

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

import { UserService } from "./user.service";
import { DataService } from "../shared/data.service";

@Component({
  selector: 'app-user',
  templateUrl: './user.component.html',
  styleUrls: ['./user.component.css'],
  providers: [UserService, DataService]
})
export class UserComponent implements OnInit {
  user: {name: string};
  isLoggedIn = false;
  data: string;

  constructor(private userService: UserService, private dataService: DataService) { }

  ngOnInit() {
    this.user = this.userService.user;
    this.dataService.getDetails().then((data: string) => this.data = data);
  }

}
```

SIMULATING ASYNC TASKS

```
export class UserService {
  user = {
    name: 'Max'
  };
}
```

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

import { UserService } from "./user.service";
import { DataService } from "../shared/data.service";

@Component({
  selector: 'app-user',
  templateUrl: './user.component.html',
  styleUrls: ['./user.component.css'],
  providers: [UserService, DataService]
})
export class UserComponent implements OnInit {
  user: {name: string};
  isLoggedIn = false;
  data: string;

  constructor(private userService: UserService, private dataService: DataService) { }

  ngOnInit() {
    this.user = this.userService.user;
    this.dataService.getDetails().then((data: string) => this.data = data);
  }

}
```

QUESTIONS?

STUDIO

Course Project – Unit Testing