# 测试相关文档

1. 什么是单元测试

单元是指在软件开发过程中，可以独立运行的最小单元，这个部分的测试被称为单元测试。单元测试通常是自动进行的，虽然它也可以手动进行。

2. 为什么单元测试很重要？

（1）单元测试会分析代码的行为；

（2）单元测试是代码无意更改的一个安全守卫；

（3）让程序员觉得，随意的代码更改是不应该的（考虑到还得注意单元测试，在写代码时会考虑一次性搞定）；

（4）单元测试还能够揭露应用设计的缺陷。

3. 测试相关配置、调试及代码覆盖率

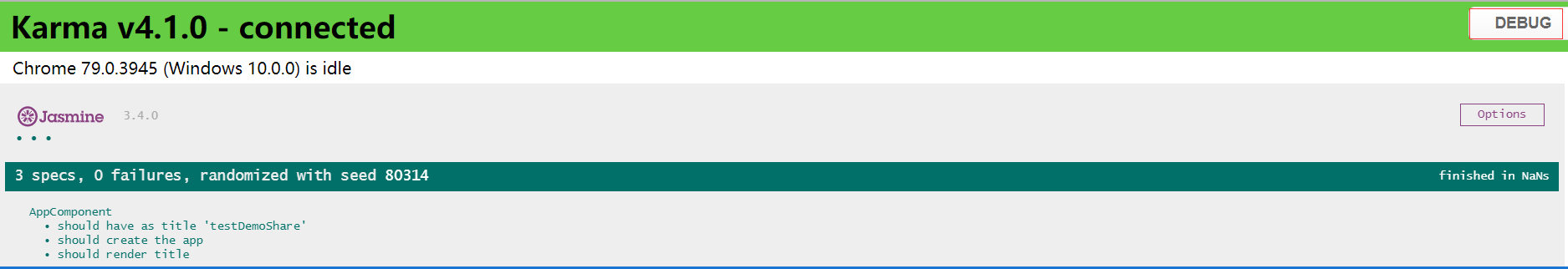
3.1 配置

如果使用cli创建的项目，可以直接使用ng test命令启动测试

3.2 调试

使用ng test命令启动测试项目后，会自动打开配置的浏览器。点击debug后会弹出另外一个标签页，按F12或者Ctrl + Shift + I可以打开调试框，在源码中设置断点/debugger进行调试。

实际上，不点击debug的页面也可以直接按照上述方式进行调试。



3.3 测试覆盖率

需要设置两个地方：

（1）angluar.json文件中设置：

"test":{

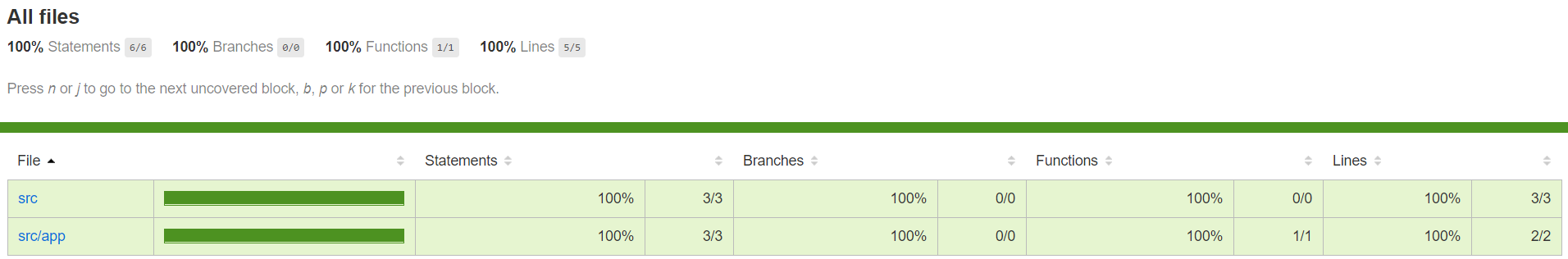
"options":{

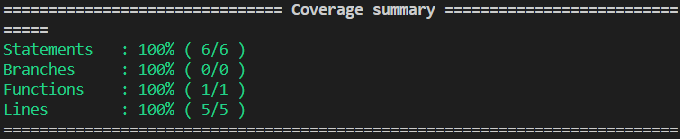
"codeCoverage": true

}

}

（2）ng test **–code-coverage**命令会生成测试覆盖率报告。测试覆盖率报告存储于项目中的coverage文件夹下，打开文件夹下index.html，可以查看详细的覆盖率报告。在命令的下方也会有一个总结性的覆盖率报告。





4. angular测试相关工具集

4.1 TestBed

TestBed可以称得上是Angular测试工具集中最重要的工具了。它可以提供组件依赖的元数据。TestBed.configureTestingModule类似于@NgModule。

beforeEach(async(() => {

TestBed.configureTestingModule({

imports: [

…

],

providers: [

],

declarations: [

…

],

}).compileComponents();

}));

4.2 component fixture

**TestBed.createComponent<T>**会创建一个组件实例，这个实例被称为fixture。

fixture.debugElement使得测试程序可以不依赖浏览器获取DOM呈现。通过这个debugElement，能够遍历组件的所有元素和组件子树。

fixture.detectChanges(): 当绑定在组件上的值发生变化，需要调用该函数来使变化生效。

5. 组件和服务的相关测试

5.1 组件

组件可以认为是一种由模板和类组合在一起的。那么在测试时，往往需要测试模板和类是否能够正确地协同工作。

5.2 同步服务

当需要测试一个注入在组件中的同步服务时，注入的真正的服务可能会遇到很多问题：比如网络问题，登录问题，数据库访问问题等，不便于测试的创建和控制。所以除了使用方式1，还可以使用方式2创建间谍服务。

方式1

const userService = fixture.debugElement.injector.get(UserService);

方式2

const mockUser = { name: 'Mannie' };

spyOn(userService, 'getUser').and.returnValue(mockUser);

5.3 异步服务/操作

当服务中存在一些由Observable/Promise/Async-Await/回调等构成的异步服务/操作时，我们需要模拟一个异步环境来指定某些异步操作。有两个函数可以完成这件事情，一个是fakeAsync，一个是tick。

fakeAsync: 将it单元测试用例包裹后，就能够为测试创建一个异步环境。在这里允许运行异步任务，还能够模拟在浏览器中运行异步。

tick: 等待异步调用结束。在fakeAsync的伪造异步环境中，可以结束timer定时器，挂起的异步任务等来模拟时间的流逝。（例如一个setTimeout定时1000ms，tick(1000)可以快速结束掉这个1000ms的setTimeout）。

此外，还有一个与tick非常相似的flush函数，两者之间的区别可以用一句话来形容：

**tick()** moves time forward.

**flush()** moves time to the end.

[参考](https://stackoverflow.com/questions/56498799/what-is-the-difference-between-tick-and-flush-in-angular-testing)

tick不传入任何参数，则与flush相同

tick指定流逝的时间，flush运行到结尾，获取到流逝的时间。

it('should clear periodic timers', fakeAsync(() => {

let cycles = 0;

const id = setInterval(() => { cycles++; }, 10);

tick(10);

expect(cycles).toEqual(1);

discardPeriodicTasks();

// Tick once to clear out the timer which already started.

**tick(10);**

expect(cycles).toEqual(2);

**tick(10);**

// Nothing should change

expect(cycles).toEqual(2);

}));

it('should flush multiple tasks', fakeAsync(() => {

let ran = false;

let ran2 = false;

setTimeout(() => { ran = true; }, 10);

setTimeout(() => { ran2 = true; }, 30);

**let elapsed = flush();**

expect(ran).toEqual(true);

expect(ran2).toEqual(true);

**expect(elapsed).toEqual(30);**

}));

5.4 http相关测试

HttpClientModule

HttpClientTestingModule

（待完善）

一般在使用angular的http API时，都会使用到HttpClientModule模块，在测试时，也有相应的HttpClientTestingModule。

5.5 管道测试

在对管道之类的功能性工具进行测试时，应该提供一个隔离环境。pipe的测试一般不依赖于angular，即与angular环境无关。

6. 跳过某些测试/只运行某些测试

xit/xdescribe: 跳过测试用例/测试套件

fit/fdescribe: 运行特定测试用例/测试套件

7. 性能相关

beforeEach中的程序会运行于每个it单元测试之前，所以angular官方推荐的是使用setup函数，需要时调用，不需要时不调用。

7.1 angular官方推荐

function setup() {

const fixture = TestBed.createComponent(AppComponent);

const app = fixture.debugElement.componentInstance;

return { fixture, app };

}

7.2 hack

7.3 parallel并行插件