ANGULAR UNIT TESTING FROM THE TRENCHES

Justin James
DevOps Evangelist
Microsoft MVP
Professional Speaker

Unit Tests Are ***Proof*** That My Code Does What I ***Think*** It Does

Tenants

- Automated
- Isolated
- Consistent
- Easy to Read and Write
- Failures Point to Problems
- Fast

Unit Testing
Will ***Not*** Fix
Bad Practices

Angular Patterns

Modules	Components	Services
Routes	Pipes	Guards

Angular CLI Setup

Node JS (6.9+)

http://nodejs.org

NPM Global Packages

```
npm install -g @angular/cli
```

Create New Project

```
ng  new AppName --style scss --routing
```

Unit Test

```
npm test
```

Tools

Test Runner	Test Framework	Testing Utilities
Karma	Jasmine	Angular

```
>npm test

> ngws@0.0.0 test C:\personal\temp\ngws
> ng test

 10% building modules 1/1 modules 0 active22 07 2017 15:31:39.233:WARN [karma]: No captured browser, open http://
localhost:9876/
22 07 2017 15:31:39.250:INFO [karma]: Karma v1.7.0 server started at http://0.0.0.0:9876/
22 07 2017 15:31:39.251:INFO [launcher]: Launching browser Chrome with unlimited concurrency
22 07 2017 15:31:39.264:INFO [launcher]: Starting browser Chrome       22 07 2017 15:31:54.722:WARN [karma]: No c
aptured browser, open http://localhost:9876/
22 07 2017 15:31:55.716:INFO [Chrome 59.0.3071 (Windows 10 0.0.0)]: Connected on socket _jzsz5yCgFjnuSX1AAAA with
 id 12356624
Chrome 59.0.3071 (Windows 10 0.0.0): Executed 46 of 46 SUCCESS (5.547 secs / 5.511 secs)
```

Karma v1.7.0 - connected

DEBUG

Chrome 59.0.3071 (Windows 10 0.0.0) is idle

Jasmine 2.6.4 finished in 6.354s

46 specs, 0 failures raise exceptions

```
AppComponent
  should create the app
  should have as title 'Our Awesome Todo App!'
  should render title in a h1 tag

LoginComponent
  should be created
  should have 1 routerLink in template
  signup link should go to signup route
  login should redirect to home page

NotFoundComponent
  should be created

FooterComponent
  should be created

IsLoggedInGuard
  should ...

HeaderComponent

  Create Test
    should be created
```

```
describe('First spec', () => {
    it('should pass', () => {
        expect(true).toBeTruthy();
    });
});
```

Simple Passing Test

```
describe('First spec', () => {
    it('should pass', () => {
        expect(true).toBeTruthy();
    });
});
```

Simple Passing Test

```
describe('First spec', () => {
    it('should pass', () => {
        expect(true).toBeTruthy();
    });
});
```

Simple Passing Test

```
describe('First spec', () => {
    it('should pass', () => {
        expect(true).toBeTruthy();
    });
});
```

Simple Passing Test

```
describe('First spec', () => {
    it('should pass', () => {
        expect(true).toBeTruthy();
    });
});
```

Simple Passing Test

```
describe('First spec', () => {
    it('should pass', () => {
        expect(true).toBeTruthy();
    });
});
```

Simple Passing Test

```
describe('First spec', () => {
    it('should pass', () => {
        expect(true).toBeTruthy();
    });
});
```

Simple Passing Test

```
describe('First spec', () => {
    it('should pass', () => {
        expect(true).toBeTruthy();
    });
});
```

Simple Passing Test

```
describe('First spec', () => {
    it('should pass', () => {
        expect(true).toBeTruthy();
    });
});
```

```
1 spec, 0 failures

  First spec
    should pass
```

Simple Passing Test

```
expect(false).toBeTruthy();
```

1 spec, 1 failure

Spec List | Failures

First failing spec should fail

Expected false to be truthy.

```
Error: Expected false to be truthy.
    at stack (http://localhost:9876/base/node_modules/jasmine-core/lib/jasmine-core/jasmine.js?da99c5b057693d025fad
    at buildExpectationResult (http://localhost:9876/base/node_modules/jasmine-core/lib/jasmine-core/jasmine.js?da9
    at Spec.expectationResultFactory (http://localhost:9876/base/node_modules/jasmine-core/lib/jasmine-core/jasmine
    at Spec.addExpectationResult (http://localhost:9876/base/node_modules/jasmine-core/lib/jasmine-core/jasmine.js?
    at Expectation.addExpectationResult (http://localhost:9876/base/node_modules/jasmine-core/lib/jasmine-core/jasm
    at Expectation.toBeTruthy (http://localhost:9876/base/node_modules/jasmine-core/lib/jasmine-core/jasmine.js?da9
    at Object.<anonymous> (http://localhost:9876/_karma_webpack_/webpack:/C:/personal/angular-tutorial-code/src/app
    at ZoneDelegate.webpackJsonp.../../../../zone.js/dist/zone.js.ZoneDelegate.invoke (http://localhost:9876/_karma
    at ProxyZoneSpec.webpackJsonp.../../../../zone.js/dist/proxy.js.ProxyZoneSpec.onInvoke (http://localhost:9876/_
    at ZoneDelegate.webpackJsonp.../../../../zone.js/dist/zone.js.ZoneDelegate.invoke (http://localhost:9876/_karma
```

Simple Failing Test

COMPONENT TESTS

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

@Component({
    selector: 'app-simple',
    templateUrl: './simple.component.html',
    styleUrls: ['./simple.component.scss']
})
export class SimpleComponent implements OnInit {
    subject = 'World';

    constructor() { }

    ngOnInit() {
    }
}
```

Component

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

@Component({
    selector: 'app-simple',
    templateUrl: './simple.component.html',
    styleUrls: ['./simple.component.scss']
})
export class SimpleComponent implements OnInit {
    subject = 'World';

    constructor() { }

    ngOnInit() {
    }
}
```

Component

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

@Component({
    selector: 'app-simple',
    templateUrl: './simple.component.html',
    styleUrls: ['./simple.component.scss']
})
export class SimpleComponent implements OnInit {
    subject = 'World';

    constructor() { }

    ngOnInit() {
    }
}
```

Component

Configure Test Module

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';
import { SimpleComponent } from './simple.component';

describe('SimpleComponent', () => {
    beforeEach(async(() => {
        TestBed.configureTestingModule({
            declarations: [ SimpleComponent ]
        })
        .compileComponents();
    }));
});
```

TestBed

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';
import { SimpleComponent } from './simple.component';

describe('SimpleComponent', () => {
    beforeEach(async(() => {
        TestBed.configureTestingModule({
            declarations: [ SimpleComponent ]
        })
        .compileComponents();
    }));
});
```

TestBed

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';
import { SimpleComponent } from './simple.component';

describe('SimpleComponent', () => {
    beforeEach(async(() => {
        TestBed.configureTestingModule({
            declarations: [ SimpleComponent ]
        })
        .compileComponents();
    }));
});
```

TestBed

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';
import { SimpleComponent } from './simple.component';

describe('SimpleComponent', () => {
    beforeEach(async(() => {
        TestBed.configureTestingModule({
            declarations: [ SimpleComponent ]
        })
        .compileComponents();
    }));
});
```

TestBed

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';
import { SimpleComponent } from './simple.component';

describe('SimpleComponent', () => {
    beforeEach(async(() => {
        TestBed.configureTestingModule({
            declarations: [ SimpleComponent ]
        })
        .compileComponents();
    }));
});
```

TestBed

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';
import { SimpleComponent } from './simple.component';

describe('SimpleComponent', () => {
    beforeEach(async(() => {
        TestBed.configureTestingModule({
            declarations: [ SimpleComponent ]
        })
        .compileComponents();
    }));
});
```

TestBed

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';
import { SimpleComponent } from './simple.component';

describe('SimpleComponent', () => {
    beforeEach(async(() => {
        TestBed.configureTestingModule({
            declarations: [ SimpleComponent ]
        })
        .compileComponents();
    }));
});
```

TestBed

Configure Test Module

Create Fixture

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';
import { SimpleComponent } from './simple.component';

describe('SimpleComponent', () => {
    let fixture: ComponentFixture<SimpleComponent>;

    beforeEach(async(() => {
        TestBed.configureTestingModule({
            declarations: [ SimpleComponent ]
        })
        .compileComponents();
    }));

    beforeEach(() => {
        fixture = TestBed.createComponent(SimpleComponent);
    });
});
```

Create Component

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';
import { SimpleComponent } from './simple.component';

describe('SimpleComponent', () => {
    let fixture: ComponentFixture<SimpleComponent>;

    beforeEach(async(() => {
        TestBed.configureTestingModule({
            declarations: [ SimpleComponent ]
        })
        .compileComponents();
    }));

    beforeEach(() => {
        fixture = TestBed.createComponent(SimpleComponent);
    });
});
```

Create Component

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';
import { SimpleComponent } from './simple.component';

describe('SimpleComponent', () => {
    let fixture: ComponentFixture<SimpleComponent>;

    beforeEach(async(() => {
        TestBed.configureTestingModule({
            declarations: [ SimpleComponent ]
        })
        .compileComponents();
    }));

    beforeEach(() => {
        fixture = TestBed.createComponent(SimpleComponent);
    });
});
```

Create Component

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';
import { SimpleComponent } from './simple.component';

describe('SimpleComponent', () => {
    let fixture: ComponentFixture<SimpleComponent>;

    beforeEach(async(() => {
        TestBed.configureTestingModule({
            declarations: [ SimpleComponent ]
        })
        .compileComponents();
    }));

    beforeEach(() => {
        fixture = TestBed.createComponent(SimpleComponent);
    });
});
```

Create Component

Configure Test Module
Create Fixture
Get Component Instance

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';
import { SimpleComponent } from './simple.component';

describe('SimpleComponent', () => {
    let fixture: ComponentFixture<SimpleComponent>;
    let component: SimpleComponent;

    beforeEach(async(() => {
        TestBed.configureTestingModule({
            declarations: [ SimpleComponent ]
        })
        .compileComponents();
    }));

    beforeEach(() => {
        fixture = TestBed.createComponent(SimpleComponent);
        component = fixture.componentInstance;
    });
});
```

Create Component

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';
import { SimpleComponent } from './simple.component';

describe('SimpleComponent', () => {
    let fixture: ComponentFixture<SimpleComponent>;
    let component: SimpleComponent;

    beforeEach(async(() => {
        TestBed.configureTestingModule({
            declarations: [ SimpleComponent ]
        })
        .compileComponents();
    }));

    beforeEach(() => {
        fixture = TestBed.createComponent(SimpleComponent);
        component = fixture.componentInstance;
    });
});
```

Create Component

```
it('should be created', () => {
    expect(component).toBeTruthy();
});
```

Component Created Test

```
it('should be created', () => {
    expect(component).toBeTruthy();
});
```

Component Created Test

Configure Test Module
Create Fixture
Get Component Instance
Trigger Change Detection

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';
import { SimpleComponent } from './simple.component';

describe('SimpleComponent', () => {
    let component: SimpleComponent;
    let fixture: ComponentFixture<SimpleComponent>;

    beforeEach(async(() => {
        TestBed.configureTestingModule({
            declarations: [ SimpleComponent ]
        })
        .compileComponents();
    }));

    beforeEach(() => {
        fixture = TestBed.createComponent(SimpleComponent);
        component = fixture.componentInstance;
        fixture.detectChanges();
    });
});
```

Create Component

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';
import { SimpleComponent } from './simple.component';

describe('SimpleComponent', () => {
    let component: SimpleComponent;
    let fixture: ComponentFixture<SimpleComponent>;

    beforeEach(async(() => {
        TestBed.configureTestingModule({
            declarations: [ SimpleComponent ]
        })
        .compileComponents();
    }));

    beforeEach(() => {
        fixture = TestBed.createComponent(SimpleComponent);
        component = fixture.componentInstance;
        fixture.detectChanges();
    });
});
```

Create Component

Configure Test Module
Create Fixture
Get Component Instance
Trigger Change Detection
Access Component's DOM

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';
import { SimpleComponent } from './simple.component';
import { By } from '@angular/platform-browser';
import { DebugElement } from '@angular/core';
describe('SimpleComponent', () => {
    let component: SimpleComponent;
    let fixture: ComponentFixture<SimpleComponent>;
    let element: DebugElement;

    beforeEach(async(() => {
        TestBed.configureTestingModule({   declarations: [ SimpleComponent ]  })
        .compileComponents();
    }));
    beforeEach(() => {
        fixture = TestBed.createComponent(SimpleComponent);
        component = fixture.componentInstance;
        element = fixture.debugElement;
        fixture.detectChanges();
    });
});
```

Element Debug

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';
import { SimpleComponent } from './simple.component';
import { By } from '@angular/platform-browser';
import { DebugElement } from '@angular/core';
describe('SimpleComponent', () => {
    let component: SimpleComponent;
    let fixture: ComponentFixture<SimpleComponent>;
    let element: DebugElement;

    beforeEach(async(() => {
        TestBed.configureTestingModule({   declarations: [ SimpleComponent ]  })
        .compileComponents();
    }));
    beforeEach(() => {
        fixture = TestBed.createComponent(SimpleComponent);
        component = fixture.componentInstance;
        element = fixture.debugElement;
        fixture.detectChanges();
    });
});
```

Element Debug

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';
import { SimpleComponent } from './simple.component';
import { By } from '@angular/platform-browser';
import { DebugElement } from '@angular/core';
describe('SimpleComponent', () => {
    let component: SimpleComponent;
    let fixture: ComponentFixture<SimpleComponent>;
    let element: DebugElement;

    beforeEach(async(() => {
        TestBed.configureTestingModule({   declarations: [ SimpleComponent ]  })
        .compileComponents();
    }));
    beforeEach(() => {
        fixture = TestBed.createComponent(SimpleComponent);
        component = fixture.componentInstance;
        element = fixture.debugElement;
        fixture.detectChanges();
    });
});
```

Element Debug

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';
import { SimpleComponent } from './simple.component';
import { By } from '@angular/platform-browser';
import { DebugElement } from '@angular/core';
describe('SimpleComponent', () => {
    let component: SimpleComponent;
    let fixture: ComponentFixture<SimpleComponent>;
    let element: DebugElement;

    beforeEach(async(() => {
        TestBed.configureTestingModule({   declarations: [ SimpleComponent ]  })
        .compileComponents();
    }));
    beforeEach(() => {
        fixture = TestBed.createComponent(SimpleComponent);
        component = fixture.componentInstance;
        element = fixture.debugElement;
        fixture.detectChanges();
    });
});
```

Element Debug

```
<h1>Hello {{ subject }}!</h1>
```

Component Html

```
it('greets the subject', () => {
    const h1 = element.query(By.css('h1'));
    expect(h1.nativeElement.innerText).toBe('Hello world!');
});
```

Test Element Value

```
it('greets the subject', () => {
    const h1 = element.query(By.css('h1'));
    expect(h1.nativeElement.innerText).toBe('Hello world!');
});
```

Test Element Value

```
it('greets the subject', () => {
    const h1 = element.query(By.css('h1'));
    expect(h1.nativeElement.innerText).toBe('Hello world!');
});
```

Test Element Value

COMPONENT WITH SERVICE DEPENDENCY

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

@Injectable()
    export class GreetingService {
    subject = 'world';
    constructor() { }
}
```

Simple Service

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

@Injectable()
    export class GreetingService {
    subject = 'world';
    constructor() { }
}
```

Simple Service

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

@Injectable()
    export class GreetingService {
    subject = 'world';
    constructor() { }
}
```

Simple Service

```
import { Component, OnInit } from '@angular/core';
import { GreetingService } from '../shared/services/greeting.service';

@Component({
    selector: 'app-simple',
    templateUrl: './simple.component.html',
    styleUrls: ['./simple.component.scss']
})
export class SimpleComponent implements OnInit {
    subject = this.service.subject;

    constructor(private service: GreetingService) { }

    ngOnInit() { }
}
```

Component with Service

```
import { Component, OnInit } from '@angular/core';
import { GreetingService } from '../shared/services/greeting.service';

@Component({
    selector: 'app-simple',
    templateUrl: './simple.component.html',
    styleUrls: ['./simple.component.scss']
})
export class SimpleComponent implements OnInit {
    subject = this.service.subject;

    constructor(private service: GreetingService) { }

    ngOnInit() { }
}
```

Component with Service

```
import { Component, OnInit } from '@angular/core';
import { GreetingService } from '../shared/services/greeting.service';

@Component({
    selector: 'app-simple',
    templateUrl: './simple.component.html',
    styleUrls: ['./simple.component.scss']
})
export class SimpleComponent implements OnInit {
    subject = this.service.subject;

    constructor(private service: GreetingService) { }

    ngOnInit() { }
}
```

Component with Service

```
import { Component, OnInit } from '@angular/core';
import { GreetingService } from '../shared/services/greeting.service';

@Component({
    selector: 'app-simple',
    templateUrl: './simple.component.html',
    styleUrls: ['./simple.component.scss']
})
export class SimpleComponent implements OnInit {
    subject = this.service.subject;

    constructor(private service: GreetingService) { }

    ngOnInit() { }
}
```

Component with Service

Don't Use Real Services

Don't Use Real Services
Use a Stub

Don't Use Real Services

Use a Stub

Use a Spy

Service Stub

```
export class GreetingServiceStub {
    public subject = 'Test World';
}
```

Greeting Service Stub

```
describe('SimpleComponentWithServiceComponent', () => {
    let component: SimpleComponentWithServiceComponent;
    let fixture: ComponentFixture<SimpleComponentWithServiceComponent>;
    let element: DebugElement;
    beforeEach(async(() => {
        TestBed.configureTestingModule({
            declarations: [SimpleComponentWithServiceComponent],
            providers: [{ provide: GreetingService, useClass: GreetingServiceStub }]
        })
        .compileComponents();
    }));
    beforeEach(() => {
        fixture = TestBed.createComponent(SimpleComponentWithServiceComponent);
        component = fixture.componentInstance;
        element = fixture.debugElement;
        fixture.detectChanges();
    });
});
```

Component With Service Setup

```
describe('SimpleComponentWithServiceComponent', () => {
    let component: SimpleComponentWithServiceComponent;
    let fixture: ComponentFixture<SimpleComponentWithServiceComponent>;
    let element: DebugElement;
    beforeEach(async(() => {
        TestBed.configureTestingModule({
            declarations: [SimpleComponentWithServiceComponent],
            providers: [{ provide: GreetingService, useClass: GreetingServiceStub }]
        })
        .compileComponents();
    }));
    beforeEach(() => {
        fixture = TestBed.createComponent(SimpleComponentWithServiceComponent);
        component = fixture.componentInstance;
        element = fixture.debugElement;
        fixture.detectChanges();
    });
});
```

Component With Service Setup

```
it('greets the subject', () => {
    const h1 = element.query(By.css('h1'));
    expect(h1.nativeElement.innerText).toBe('Hello Test World!');
});
```

Same Greet Test but With Service

```
it('greets the subject', () => {
    const h1 = element.query(By.css('h1'));
    expect(h1.nativeElement.innerText).toBe('Hello Test World!');
});
```

Same Greet Test but With Service

```
it('greets the subject', () => {
    const h1 = element.query(By.css('h1'));
    expect(h1.nativeElement.innerText).toBe('Hello Test World!');
});

export class GreetingServiceStub {
    public subject = 'Test World';
}
```

Same Greet Test but With Service

Service Spying

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

@Injectable()
export class GreetingService {
    constructor() { }

    getGreeting(): Observable<string> { return Observable.of('Hello'); }

    getSubject(): Observable<string> { return Observable.of('World'); }

    getPunctuation(): Observable<string> { return Observable.of('!'); }
}
```

Service

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

@Injectable()
export class GreetingService {
    constructor() { }

    getGreeting(): Observable<string> { return Observable.of('Hello'); }

    getSubject(): Observable<string> { return Observable.of('World'); }

    getPunctuation(): Observable<string> { return Observable.of('!'); }
}
```

Service

```
export class SimpleComponentWithServiceComponent implements OnInit {
    subject: string;
    greeting: string;
    punctuation: string;
    constructor(private greetingService: GreetingService) { }

    ngOnInit() {
        this.greetingService.getSubject().subscribe((result) => {
            this.subject = result;
        });

        this.greetingService.getGreeting().subscribe((result) => {
            this.greeting = result;
        });

        this.greetingService.getPunctuation().subscribe((result) => {
            this.punctuation = result;
        });
    }
}
```

Component

```
export class SimpleComponentWithServiceComponent implements OnInit {
    subject: string;
    greeting: string;
    punctuation: string;
    constructor(private greetingService: GreetingService) { }

    ngOnInit() {
        this.greetingService.getSubject().subscribe((result) => {
            this.subject = result;
        });

        this.greetingService.getGreeting().subscribe((result) => {
            this.greeting = result;
        });

        this.greetingService.getPunctuation().subscribe((result) => {
            this.punctuation = result;
        });
    }
}
```

Component

```
export class SimpleComponentWithServiceComponent implements OnInit {
    subject: string;
    greeting: string;
    punctuation: string;
    constructor(private greetingService: GreetingService) { }

    ngOnInit() {
        this.greetingService.getSubject().subscribe((result) => {
            this.subject = result;
        });

        this.greetingService.getGreeting().subscribe((result) => {
            this.greeting = result;
        });

        this.greetingService.getPunctuation().subscribe((result) => {
            this.punctuation = result;
        });
    }
}
```

Component

```
let component: SimpleComponentWithServiceComponent;
let fixture: ComponentFixture<SimpleComponentWithServiceComponent>;
let element: DebugElement;
let greetingService: GreetingService;

beforeEach(async(() => {
    TestBed.configureTestingModule({
        declarations: [SimpleComponentWithServiceComponent],
        providers: [GreetingService]
    })
    .compileComponents();
}));

beforeEach(() => {
    fixture = TestBed.createComponent(SimpleComponentWithServiceComponent);
    comonent = fixture.componentInstance;
    element = fixture.debugElement;
    greetingService = element.injector.get(GreetingService);
});
```

Setup

```
let component: SimpleComponentWithServiceComponent;
let fixture: ComponentFixture<SimpleComponentWithServiceComponent>;
let element: DebugElement;
let greetingService: GreetingService;

beforeEach(async(() => {
    TestBed.configureTestingModule({
        declarations: [SimpleComponentWithServiceComponent],
        providers: [GreetingService]
    })
    .compileComponents();
}));

beforeEach(() => {
    fixture = TestBed.createComponent(SimpleComponentWithServiceComponent);
    comonent = fixture.componentInstance;
    element = fixture.debugElement;
    greetingService = element.injector.get(GreetingService);
});
```

Setup

```
let component: SimpleComponentWithServiceComponent;
let fixture: ComponentFixture<SimpleComponentWithServiceComponent>;
let element: DebugElement;
let greetingService: GreetingService;

beforeEach(async(() => {
    TestBed.configureTestingModule({
        declarations: [SimpleComponentWithServiceComponent],
        providers: [GreetingService]
    })
    .compileComponents();
}));

beforeEach(() => {
    fixture = TestBed.createComponent(SimpleComponentWithServiceComponent);
    comonent = fixture.componentInstance;
    element = fixture.debugElement;
    greetingService = element.injector.get(GreetingService);
});
```

Setup

```
let component: SimpleComponentWithServiceComponent;
let fixture: ComponentFixture<SimpleComponentWithServiceComponent>;
let element: DebugElement;
let greetingService: GreetingService;

beforeEach(async(() => {
    TestBed.configureTestingModule({
        declarations: [SimpleComponentWithServiceComponent],
        providers: [GreetingService]
    })
    .compileComponents();
}));

beforeEach(() => {
    fixture = TestBed.createComponent(SimpleComponentWithServiceComponent);
    comonent = fixture.componentInstance;
    element = fixture.debugElement;
    greetingService = element.injector.get(GreetingService);
});
```

Setup

```
it('spying on greets', async(() => {
    spyOn(greetingService, 'getGreeting').and.returnValue(Observable.of('hello'));
    fixture.detectChanges();
    fixture.whenStable().then(() => {
        fixture.detectChanges();
        expect(component.greeting).toBe('hello');
    });
}));
```

Component Test with SpyOn

```
it('spying on greets', async(() => {
    spyOn(greetingService, 'getGreeting').and.returnValue(Observable.of('hello'));
    fixture.detectChanges();
    fixture.whenStable().then(() => {
        fixture.detectChanges();
        expect(component.greeting).toBe('hello');
    });
}));
```

Component Test with SpyOn

```
it('spying on greets', async(() => {
    spyOn(greetingService, 'getGreeting').and.returnValue(Observable.of('hello'));
    fixture.detectChanges();
    fixture.whenStable().then(() => {
        fixture.detectChanges();
        expect(component.greeting).toBe('hello');
    });
}));
```

Component Test with SpyOn

```
it('spying on greets', async(() => {
    spyOn(greetingService, 'getGreeting').and.returnValue(Observable.of('hello'));
    fixture.detectChanges();
    fixture.whenStable().then(() => {
        fixture.detectChanges();
        expect(component.greeting).toBe('hello');
    });
}));
```

Component Test with SpyOn

```
it('spying on greets', async(() => {
    spyOn(greetingService, 'getGreeting').and.returnValue(Observable.of('hello'));
    fixture.detectChanges();
    fixture.whenStable().then(() => {
        fixture.detectChanges();
        expect(component.greeting).toBe('hello');
    });
}));
```

Component Test with SpyOn

```
it('spying on greets', async(() => {
    spyOn(greetingService, 'getGreeting').and.returnValue(Observable.of('hello'));
    fixture.detectChanges();
    fixture.whenStable().then(() => {
        fixture.detectChanges();
        expect(component.greeting).toBe('hello');
    });
}));
```

Component Test with SpyOn

```
it('spying on greets', async(() => {
    spyOn(greetingService, 'getGreeting').and.returnValue(Observable.of('hello'));
    fixture.detectChanges();
    fixture.whenStable().then(() => {
        fixture.detectChanges();
        expect(component.greeting).toBe('hello');
    });
}));
```

Component Test with SpyOn

Make Test Look More Synchronous Code

```
import { tick, fakeAsync } from '@angular/core/testing';

it('fakeasync test', fakeAsync(() => {
    spyOn(greetingService, 'getGreeting').and.returnValue(Observable.of('hello'));
    fixture.detectChanges();
    tick();
    fixture.detectChanges();
    expect(component.greeting).toBe('hello');
}));
```

Test Using FakeAsync

```
import { tick, fakeAsync } from '@angular/core/testing';

it('fakeasync test', fakeAsync(() => {
    spyOn(greetingService, 'getGreeting').and.returnValue(Observable.of('hello'));
    fixture.detectChanges();
    tick();
    fixture.detectChanges();
    expect(component.greeting).toBe('hello');
}));
```

Test Using FakeAsync

```
import { tick, fakeAsync } from '@angular/core/testing';

it('fakeasync test', fakeAsync(() => {
    spyOn(greetingService, 'getGreeting').and.returnValue(Observable.of('hello'));
    fixture.detectChanges();
    tick();
    fixture.detectChanges();
    expect(component.greeting).toBe('hello');
}));
```

Test Using FakeAsync

```
import { tick, fakeAsync } from '@angular/core/testing';

it('fakeasync test', fakeAsync(() => {
    spyOn(greetingService, 'getGreeting').and.returnValue(Observable.of('hello'));
    fixture.detectChanges();
    tick();
    fixture.detectChanges();
    expect(component.greeting).toBe('hello');
}));
```

Test Using FakeAsync

```
import { tick, fakeAsync } from '@angular/core/testing';

it('fakeasync test', fakeAsync(() => {
    spyOn(greetingService, 'getGreeting').and.returnValue(Observable.of('hello'));
    fixture.detectChanges();
    tick();
    fixture.detectChanges();
    expect(component.greeting).toBe('hello');
}));
```

Test Using FakeAsync

```
import { tick, fakeAsync } from '@angular/core/testing';

it('fakeasync test', fakeAsync(() => {
    spyOn(greetingService, 'getGreeting').and.returnValue(Observable.of('hello'));
    fixture.detectChanges();
    tick();
    fixture.detectChanges();
    expect(component.gretting).toBe('hello');
}));
```

Test Using FakeAsync

```
import { tick, fakeAsync } from '@angular/core/testing';

it('fakeasync test', fakeAsync(() => {
    spyOn(greetingService, 'getGreeting').and.returnValue(Observable.of('hello'));
    fixture.detectChanges();
    tick();
    fixture.detectChanges();
    expect(component.greeting).toBe('hello');
}));
```

Test Using FakeAsync

REACTIVE FORMS

Reactive Forms Tests

Validation Errors

Field Validity

Form Validity

Error Messages

Error Message Display

```
import { FormGroup, FormBuilder, Validators } from '@angular/forms';
import 'rxjs/add/operator/debounceTime';

export class TodoComponent implements OnInit {
    addForm: FormGroup;
    constructor(private formBuilder: FormBuilder) { }

    ngOnInit() {
        this.addForm = this.formBuilder.group({
            'item': ['', [Validators.required, Validators.minLength(3)]]
        });

        this.addForm.valueChanges
            .debounceTime(1000)
            .subscribe(data =>
                this.onValueChanged(data)
            );
    }
}
```

Reactive Form Setup

```
import { FormGroup, FormBuilder, Validators } from '@angular/forms';
import 'rxjs/add/operator/debounceTime';

export class TodoComponent implements OnInit {
    addForm: FormGroup;
    constructor(private formBuilder: FormBuilder) { }

    ngOnInit() {
        this.addForm = this.formBuilder.group({
            'item': ['', [Validators.required, Validators.minLength(3)]]
        });

        this.addForm.valueChanges
            .debounceTime(1000)
            .subscribe(data =>
                this.onValueChanged(data)
            );
    }
}
```

Reactive Form Setup

```
import { FormGroup, FormBuilder, Validators } from '@angular/forms';
import 'rxjs/add/operator/debounceTime';

export class TodoComponent implements OnInit {
    addForm: FormGroup;
    constructor(private formBuilder: FormBuilder) { }

    ngOnInit() {
        this.addForm = this.formBuilder.group({
            'item': ['', [Validators.required, Validators.minLength(3)]]
        });

        this.addForm.valueChanges
            .debounceTime(1000)
            .subscribe(data =>
                this.onValueChanged(data)
            );
    }
}
```

Reactive Form Setup

```
import { FormGroup, FormBuilder, Validators } from '@angular/forms';
import 'rxjs/add/operator/debounceTime';

export class TodoComponent implements OnInit {
    addForm: FormGroup;
    constructor(private formBuilder: FormBuilder) { }

    ngOnInit() {
        this.addForm = this.formBuilder.group({
            'item': ['', [Validators.required, Validators.minLength(3)]]
        });

        this.addForm.valueChanges
            .debounceTime(1000)
            .subscribe(data =>
                this.onValueChanged(data)
            );
    }
}
```

Reactive Form Setup

```
import { FormGroup, FormBuilder, Validators } from '@angular/forms';
import 'rxjs/add/operator/debounceTime';

export class TodoComponent implements OnInit {
    addForm: FormGroup;
    constructor(private formBuilder: FormBuilder) { }

    ngOnInit() {
        this.addForm = this.formBuilder.group({
            'item': ['', [Validators.required, Validators.minLength(3)]]
        });

        this.addForm.valueChanges
            .debounceTime(1000)
            .subscribe(data =>
                this.onValueChanged(data)
            );
    }
}
```

Reactive Form Setup

```
import { ReactiveFormsModule } from '@angular/forms';
import { AbstractControl, Validators } from '@angular/forms';

beforeEach(async(() => {
    TestBed.configureTestingModule({
        imports: [ReactiveFormsModule],
        declarations: [ TodoComponent  ],
        providers: [ { provide: TodoService, useClass: MockTodoService } ]
    })
    .compileComponents();
}));

beforeEach(() => {
    fixture = TestBed.createComponent(TodoComponent);
    element = fixture.debugElement;
    component = element.componentInstance;
    itemField = component.addForm.controls['item'];
    fixture.detectChanges();
});
```

TestBed Setup

```
import { ReactiveFormsModule } from '@angular/forms';
import { AbstractControl, Validators } from '@angular/forms';

beforeEach(async(() => {
    TestBed.configureTestingModule({
        imports: [ReactiveFormsModule],
        declarations: [ TodoComponent  ],
        providers: [ { provide: TodoService, useClass: MockTodoService } ]
    })
    .compileComponents();
}));

beforeEach(() => {
    fixture = TestBed.createComponent(TodoComponent);
    element = fixture.debugElement;
    component = element.componentInstance;
    itemField = component.addForm.controls['item'];
    fixture.detectChanges();
});
```

TestBed Setup

```
import { ReactiveFormsModule } from '@angular/forms';
import { AbstractControl, Validators } from '@angular/forms';

beforeEach(async(() => {
    TestBed.configureTestingModule({
        imports: [ReactiveFormsModule],
        declarations: [ TodoComponent  ],
        providers: [ { provide: TodoService, useClass: MockTodoService } ]
    })
    .compileComponents();
}));

beforeEach(() => {
    fixture = TestBed.createComponent(TodoComponent);
    element = fixture.debugElement;
    component = element.componentInstance;
    itemField = component.addForm.controls['item'];
    fixture.detectChanges();
});
```

TestBed Setup

```
import { ReactiveFormsModule } from '@angular/forms';
import { AbstractControl, Validators } from '@angular/forms';

beforeEach(async(() => {
    TestBed.configureTestingModule({
        imports: [ReactiveFormsModule],
        declarations: [ TodoComponent  ],
        providers: [ { provide: TodoService, useClass: MockTodoService } ]
    })
    .compileComponents();
}));

beforeEach(() => {
    fixture = TestBed.createComponent(TodoComponent);
    element = fixture.debugElement;
    component = element.componentInstance;
    itemField = component.addForm.controls['item'];
    fixture.detectChanges();
});
```

TestBed Setup

```
it('item field required with blank validity', () => {
    itemField.setValue('');
    errors = itemField.errors || {};
    expect(errors['required']).toBeDefined('required validator not triggered');
    expect(errors['minlength']).toBeUndefined('min length validator was triggered');
    expect(itemField.valid).toBeFalsy();
    expect(component.addForm.valid).toBeFalsy();
});
```

Test Field Validity

```
it('item field required with blank validity', () => {
    itemField.setValue('');
    errors = itemField.errors || {};
    expect(errors['required']).toBeDefined('required validator not triggered');
    expect(errors['minlength']).toBeUndefined('min length validator was triggered');
    expect(itemField.valid).toBeFalsy();
    expect(component.addForm.valid).toBeFalsy();
});
```

Test Field Validity

```
it('item field required with blank validity', () => {
    itemField.setValue('');
    errors = itemField.errors || {};
    expect(errors['required']).toBeDefined('required validator not triggered');
    expect(errors['minlength']).toBeUndefined('min length validator was triggered');
    expect(itemField.valid).toBeFalsy();
    expect(component.addForm.valid).toBeFalsy();
});
```

Test Field Validity

```
it('item field required with blank validity', () => {
    itemField.setValue('');
    errors = itemField.errors || {};
    expect(errors['required']).toBeDefined('required validator not triggered');
    expect(errors['minlength']).toBeUndefined('min length validator was triggered');
    expect(itemField.valid).toBeFalsy();
    expect(component.addForm.valid).toBeFalsy();
});
```

Test Field Validity

```
it('item field required with blank validity', () => {
    itemField.setValue('');
    errors = itemField.errors || {};
    expect(errors['required']).toBeDefined('required validator not triggered');
    expect(errors['minlength']).toBeUndefined('min length validator was triggered');
    expect(itemField.valid).toBeFalsy();
    expect(component.addForm.valid).toBeFalsy();
});
```

Test Field Validity

```
it('item field min length too short validity', () => {
    itemField.setValue('1');
    errors = itemField.errors || {};
    expect(errors['minlength']).toBeDefined();
    expect(errors['required']).toBeUndefined();
    expect(itemField.valid).toBeFalsy();
    expect(component.addForm.valid).toBeFalsy();
});
```

Min Length Validator

```
it('item field min length too short validity', () => {
    itemField.setValue('1');
    errors = itemField.errors || {};
    expect(errors['minlength']).toBeDefined();
    expect(errors['required']).toBeUndefined();
    expect(itemField.valid).toBeFalsy();
    expect(component.addForm.valid).toBeFalsy();
});
```

Min Length Validator

```
it('item field min length too short validity', () => {
    itemField.setValue('1');
    errors = itemField.errors || {};
    expect(errors['minlength']).toBeDefined();
    expect(errors['required']).toBeUndefined();
    expect(itemField.valid).toBeFalsy();
    expect(component.addForm.valid).toBeFalsy();
});
```

Min Length Validator

```
it('item field min length too short validity', () => {
    itemField.setValue('1');
    errors = itemField.errors || {};
    expect(errors['minlength']).toBeDefined();
    expect(errors['required']).toBeUndefined();
    expect(itemField.valid).toBeFalsy();
    expect(component.addForm.valid).toBeFalsy();
});
```

Min Length Validator

REACTIVE FORMS GOTCHAS

Field Not Set as Dirty When Calling setvalue

```
beforeEach(() => {
    itemField.markAsDirty();
    expect(itemField.dirty).toBeTruthy('field should be dirty');
    fixture.detectChanges();
});
```

Mark Field as Dirty

```
beforeEach(() => {
    itemField.markAsDirty();
    expect(itemField.dirty).toBeTruthy('field should be dirty');
    fixture.detectChanges();
});
```

Mark Field as Dirty

```
beforeEach(() => {
    itemField.markAsDirty();
    expect(itemField.dirty).toBeTruthy('field should be dirty');
    fixture.detectChanges();
});
```

Mark Field as Dirty

```
beforeEach(() => {
    itemField.markAsDirty();
    expect(itemField.dirty).toBeTruthy('field should be dirty');
    fixture.detectChanges();
});
```

Mark Field as Dirty

Debounce Time
Has to be Overridden

```
import { FormGroup, FormBuilder, Validators } from '@angular/forms';
import 'rxjs/add/operator/debounceTime';

export class TodoComponent implements OnInit {
    addForm: FormGroup;
    constructor(private formBuilder: FormBuilder) { }

    ngOnInit() {
        this.addForm = this.formBuilder.group({
            'item': ['', [Validators.required, Validators.minLength(3)]]
        });

        this.addForm.valueChanges
            .debounceTime(1000)
            .subscribe(data =>
                this.onValueChanged(data)
            );
    }
}
```

Reactive Form Setup

```
beforeAll(() => {
    // monkey patches debounce time to make field validation errors test past.
    Observable.prototype.debounceTime = function () { return this; };
});
```

Monkey Patch Debounce

HTTP SERVICES

Don't Call Actual Backend
Use MockBackend

```
import { BaseRequestOptions, Http, Response, ResponseOptions } from '@angular/http';
import { MockBackend } from '@angular/http/testing';
let service: TodoService;
let mockBackend: MockBackend;
beforeEach(() => {
    const bed = TestBed.configureTestingModule({
        providers: [
            TodoService,
            MockBackend,
            BaseRequestOptions,
            {
                provide: Http,
                useFactory: (backend, options) => new Http(backend, options),
                deps: [MockBackend, BaseRequestOptions]
            }
        ]
    });
    service = bed.get(TodoService);
    mockBackend = bed.get(MockBackend);
});
```

MockBackend

```
import { BaseRequestOptions, Http, Response, ResponseOptions } from '@angular/http';
import { MockBackend } from '@angular/http/testing';
let service: TodoService;
let mockBackend: MockBackend;
beforeEach(() => {
    const bed = TestBed.configureTestingModule({
        providers: [
            TodoService,
            MockBackend,
            BaseRequestOptions,
            {
                provide: Http,
                useFactory: (backend, options) => new Http(backend, options),
                deps: [MockBackend, BaseRequestOptions]
            }
        ]
    });
    service = bed.get(TodoService);
    mockBackend = bed.get(MockBackend);
});
```

MockBackend

```
import { BaseRequestOptions, Http, Response, ResponseOptions } from '@angular/http';
import { MockBackend } from '@angular/http/testing';
let service: TodoService;
let mockBackend: MockBackend;
beforeEach(() => {
    const bed = TestBed.configureTestingModule({
        providers: [
            TodoService,
            MockBackend,
            BaseRequestOptions,
            {
                provide: Http,
                useFactory: (backend, options) => new Http(backend, options),
                deps: [MockBackend, BaseRequestOptions]
            }
        ]
    });
    service = bed.get(TodoService);
    mockBackend = bed.get(MockBackend);
});
```

MockBackend

```
import { BaseRequestOptions, Http, Response, ResponseOptions } from '@angular/http';
import { MockBackend } from '@angular/http/testing';
let service: TodoService;
let mockBackend: MockBackend;
beforeEach(() => {
    const bed = TestBed.configureTestingModule({
        providers: [
            TodoService,
            MockBackend,
            BaseRequestOptions,
            {
                provide: Http,
                useFactory: (backend, options) => new Http(backend, options),
                deps: [MockBackend, BaseRequestOptions]
            }
        ]
    });
    service = bed.get(TodoService);
    mockBackend = bed.get(MockBackend);
});
```

MockBackend

```
import { BaseRequestOptions, Http, Response, ResponseOptions } from '@angular/http';
import { MockBackend } from '@angular/http/testing';
let service: TodoService;
let mockBackend: MockBackend;
beforeEach(() => {
    const bed = TestBed.configureTestingModule({
        providers: [
            TodoService,
            MockBackend,
            BaseRequestOptions,
            {
                provide: Http,
                useFactory: (backend, options) => new Http(backend, options),
                deps: [MockBackend, BaseRequestOptions]
            }
        ]
    });
    service = bed.get(TodoService);
    mockBackend = bed.get(MockBackend);
});
```

MockBackend

```
import { BaseRequestOptions, Http, Response, ResponseOptions } from '@angular/http';
import { MockBackend } from '@angular/http/testing';
let service: TodoService;
let mockBackend: MockBackend;
beforeEach(() => {
    const bed = TestBed.configureTestingModule({
        providers: [
            TodoService,
            MockBackend,
            BaseRequestOptions,
            {
                provide: Http,
                useFactory: (backend, options) => new Http(backend, options),
                deps: [MockBackend, BaseRequestOptions]
            }
        ]
    });
    service = bed.get(TodoService);
    mockBackend = bed.get(MockBackend);
});
```

MockBackend

```
it('should get todo items', async(() => {
    mockBackend.connections.subscribe(conn => {
        conn.mockRespond(new Response(new ResponseOptions({
            body: mockTodoData.todoItems
        })));
    });

    service.getAll()
        .subscribe((result => {
            expect(result.length).toBe(mockTodoData.todoItems.length);
            expect(result).toEqual(mockTodoData.todoItems);
        }));
}));
```

Using MockBackend in Test

```
it('should get todo items', async(() => {
    mockBackend.connections.subscribe(conn => {
        conn.mockRespond(new Response(new ResponseOptions({
            body: mockTodoData.todoItems
        })));
    });

    service.getAll()
        .subscribe((result => {
            expect(result.length).toBe(mockTodoData.todoItems.length);
            expect(result).toEqual(mockTodoData.todoItems);
        }));
}));
```

Using MockBackend in Test

```
it('should get todo items', async(() => {
    mockBackend.connections.subscribe(conn => {
        conn.mockRespond(new Response(new ResponseOptions({
            body: mockTodoData.todoItems
        })));
    });

    service.getAll()
        .subscribe((result => {
            expect(result.length).toBe(mockTodoData.todoItems.length);
            expect(result).toEqual(mockTodoData.todoItems);
        }));
}));
```

Using MockBackend in Test

```
it('should get todo items', async(() => {
    mockBackend.connections.subscribe(conn => {
        conn.mockRespond(new Response(new ResponseOptions({
            body: mockTodoData.todoItems
        })));
    });

    service.getAll()
        .subscribe((result => {
            expect(result.length).toBe(mockTodoData.todoItems.length);
            expect(result).toEqual(mockTodoData.todoItems);
        }));
}));
```

Using MockBackend in Test

ROUTING VALIDATION

Goal Is To Test Navigation

Not Angular Router

Abstract Router Complexity

```
import { RouterTestingModule } from '@angular/router/testing';

beforeEach(async(() => {
    TestBed.configureTestingModule({
        imports: [
            RouterTestingModule.withRoutes([
                { path: '', component: HeaderComponent },
                { path: 'login', children: [], component: HeaderComponent },
                { path: 'signup', component: HeaderComponent },
                { path: '**', component: HeaderComponent }
            ]),
        ],
        declarations: [
            HeaderComponent,
        ]
    })
    .compileComponents();
}));
```

RouterTestingModule Setup Part 1 of 2

```
import { RouterTestingModule } from '@angular/router/testing';

beforeEach(async(() => {
    TestBed.configureTestingModule({
        imports: [
            RouterTestingModule.withRoutes([
                { path: '', component: HeaderComponent },
                { path: 'login', children: [], component: HeaderComponent },
                { path: 'signup', component: HeaderComponent },
                { path: '**', component: HeaderComponent }
            ]),
        ],
        declarations: [
            HeaderComponent,
        ]
    })
    .compileComponents();
}));
```

RouterTestingModule Setup Part 1 of 2

```
import { RouterTestingModule } from '@angular/router/testing';

beforeEach(async(() => {
    TestBed.configureTestingModule({
        imports: [
            RouterTestingModule.withRoutes([
                { path: '', component: HeaderComponent },
                { path: 'login', children: [], component: HeaderComponent },
                { path: 'signup', component: HeaderComponent },
                { path: '**', component: HeaderComponent }
            ]),
        ],
        declarations: [
            HeaderComponent,
        ]
    })
    .compileComponents();
}));
```

RouterTestingModule Setup Part 1 of 2

```
import { RouterTestingModule } from '@angular/router/testing';

beforeEach(async(() => {
    TestBed.configureTestingModule({
        imports: [
            RouterTestingModule.withRoutes([
                { path: '', component: HeaderComponent },
                { path: 'login', children: [], component: HeaderComponent },
                { path: 'signup', component: HeaderComponent },
                { path: '**', component: HeaderComponent }
            ]),
        ],
        declarations: [
            HeaderComponent,
        ]
    })
    .compileComponents();
}));
```

RouterTestingModule Setup Part 1 of 2

```
import { RouterTestingModule } from '@angular/router/testing';

beforeEach(async(() => {
    TestBed.configureTestingModule({
        imports: [
            RouterTestingModule.withRoutes([
                { path: '', component: HeaderComponent },
                { path: 'login', children: [], component: HeaderComponent },
                { path: 'signup', component: HeaderComponent },
                { path: '**', component: HeaderComponent }
            ]),
        ],
        declarations: [
            HeaderComponent,
        ]
    })
    .compileComponents();
}));
```

RouterTestingModule Setup Part 1 of 2

```
import { SpyLocation } from '@angular/common/testing';
import { Location } from '@angular/common';
import { RouterLinkWithHref } from '@angular/router';

let location: SpyLocation;
let allLinkDes: DebugElement[];

beforeEach(() => {
    fixture = TestBed.createComponent(HeaderComponent);
    element = fixture.debugElement;
    component = element.componentInstance;

    const injector = fixture.debugElement.injector;
    location = injector.get(Location) as SpyLocation;

    allLinkDes = fixture.debugElement.queryAll(By.directive(RouterLinkWithHref));

    fixture.detectChanges();
});
```

RouterTestingModule Setup Part 2 of 2

```
import { SpyLocation } from '@angular/common/testing';
import { Location } from '@angular/common';
import { RouterLinkWithHref } from '@angular/router';

let location: SpyLocation;
let allLinkDes: DebugElement[];

beforeEach(() => {
    fixture = TestBed.createComponent(HeaderComponent);
    element = fixture.debugElement;
    component = element.componentInstance;

    const injector = fixture.debugElement.injector;
    location = injector.get(Location) as SpyLocation;

    allLinkDes = fixture.debugElement.queryAll(By.directive(RouterLinkWithHref));

    fixture.detectChanges();
});
```

RouterTestingModule Setup Part 2 of 2

```
import { SpyLocation } from '@angular/common/testing';
import { Location } from '@angular/common';
import { RouterLinkWithHref } from '@angular/router';

let location: SpyLocation;
let allLinkDes: DebugElement[];

beforeEach(() => {
    fixture = TestBed.createComponent(HeaderComponent);
    element = fixture.debugElement;
    component = element.componentInstance;

    const injector = fixture.debugElement.injector;
    location = injector.get(Location) as SpyLocation;

    allLinkDes = fixture.debugElement.queryAll(By.directive(RouterLinkWithHref));

    fixture.detectChanges();
});
```

RouterTestingModule Setup Part 2 of 2

```
import { SpyLocation } from '@angular/common/testing';
import { Location } from '@angular/common';
import { RouterLinkWithHref } from '@angular/router';

let location: SpyLocation;
let allLinkDes: DebugElement[];

beforeEach(() => {
    fixture = TestBed.createComponent(HeaderComponent);
    element = fixture.debugElement;
    component = element.componentInstance;

    const injector = fixture.debugElement.injector;
    location = injector.get(Location) as SpyLocation;

    allLinkDes = fixture.debugElement.queryAll(By.directive(RouterLinkWithHref));

    fixture.detectChanges();
});
```

RouterTestingModule Setup Part 2 of 2

```
import { SpyLocation } from '@angular/common/testing';
import { Location } from '@angular/common';
import { RouterLinkWithHref } from '@angular/router';

let location: SpyLocation;
let allLinkDes: DebugElement[];

beforeEach(() => {
    fixture = TestBed.createComponent(HeaderComponent);
    element = fixture.debugElement;
    component = element.componentInstance;

    const injector = fixture.debugElement.injector;
    location = injector.get(Location) as SpyLocation;

    allLinkDes = fixture.debugElement.queryAll(By.directive(RouterLinkWithHref));

    fixture.detectChanges();
});
```

RouterTestingModule Setup Part 2 of 2

```
import { SpyLocation } from '@angular/common/testing';
import { Location } from '@angular/common';
import { RouterLinkWithHref } from '@angular/router';

let location: SpyLocation;
let allLinkDes: DebugElement[];

beforeEach(() => {
    fixture = TestBed.createComponent(HeaderComponent);
    element = fixture.debugElement;
    component = element.componentInstance;

    const injector = fixture.debugElement.injector;
    location = injector.get(Location) as SpyLocation;

    allLinkDes = fixture.debugElement.queryAll(By.directive(RouterLinkWithHref));

    fixture.detectChanges();
});
```

RouterTestingModule Setup Part 2 of 2

```
import {advance, expectPathToBe, click } from '../../../testing';

it('all items takes me home', fakeAsync(() => {
    const linkDes = allLinkDes[4];
    expectPathToBe(location, '', 'link should not have navigated yet');

    click(linkDes);
    advance(fixture);

    expectPathToBe(location, '/signup');
}));
```

Test Route Worked

```
import {advance, expectPathToBe, click } from '../../../testing';

it('all items takes me home', fakeAsync(() => {
    const linkDes = allLinkDes[4];
    expectPathToBe(location, '', 'link should not have navigated yet');

    click(linkDes);
    advance(fixture);

    expectPathToBe(location, '/signup');
}));
```

Test Route Worked

```
import {advance, expectPathToBe, click } from '../../../testing';

it('all items takes me home', fakeAsync(() => {
    const linkDes = allLinkDes[4];
    expectPathToBe(location, '', 'link should not have navigated yet');

    click(linkDes);
    advance(fixture);

    expectPathToBe(location, '/signup');
}));
```

Test Route Worked

```
import {advance, expectPathToBe, click } from '../../../testing';

it('all items takes me home', fakeAsync(() => {
    const linkDes = allLinkDes[4];
    expectPathToBe(location, '', 'link should not have navigated yet');

    click(linkDes);
    advance(fixture);

    expectPathToBe(location, '/signup');
}));
```

Test Route Worked

```
import {advance, expectPathToBe, click } from '../../../testing';

it('all items takes me home', fakeAsync(() => {
    const linkDes = allLinkDes[4];
    expectPathToBe(location, '', 'link should not have navigated yet');

    click(linkDes);
    advance(fixture);

    expectPathToBe(location, '/signup');
}));
```

Test Route Worked

```
import {advance, expectPathToBe, click } from '../../../testing';

it('all items takes me home', fakeAsync(() => {
    const linkDes = allLinkDes[4];
    expectPathToBe(location, '', 'link should not have navigated yet');

    click(linkDes);
    advance(fixture);

    expectPathToBe(location, '/signup');
}));
```

Test Route Worked

```
import {advance, expectPathToBe, click } from '../../../testing';

it('all items takes me home', fakeAsync(() => {
    const linkDes = allLinkDes[4];
    expectPathToBe(location, '', 'link should not have navigated yet');

    click(linkDes);
    advance(fixture);

    expectPathToBe(location, '/signup');
}));
```

Test Route Worked

```
export const ButtonClickEvents = {
    left: { button: 0 },
    right: { button: 2 }
};

export function click(
    el: DebugElement | HTMLElement,
    eventObj: any =  ButtonClickEvents.left): void
{
    if (el instanceof HTMLElement) {
        el.click();
    } else {
        el.triggerEventHandler('click', eventObj);
    }
}
```

Helper: Click Function

```
export const ButtonClickEvents = {
    left: { button: 0 },
    right: { button: 2 }
};

export function click(
    el: DebugElement | HTMLElement,
    eventObj: any =  ButtonClickEvents.left): void
{
    if (el instanceof HTMLElement) {
        el.click();
    } else {
        el.triggerEventHandler('click', eventObj);
    }
}
```

Helper: Click Function

```
export const ButtonClickEvents = {
    left: { button: 0 },
    right: { button: 2 }
};

export function click(
    el: DebugElement | HTMLElement,
    eventObj: any =  ButtonClickEvents.left): void
{
    if (el instanceof HTMLElement) {
        el.click();
    } else {
        el.triggerEventHandler('click', eventObj);
    }
}
```

Helper: Click Function

```
export const ButtonClickEvents = {
    left: { button: 0 },
    right: { button: 2 }
};

export function click(
    el: DebugElement | HTMLElement,
    eventObj: any =  ButtonClickEvents.left): void
{
    if (el instanceof HTMLElement) {
        el.click();
    } else {
        el.triggerEventHandler('click', eventObj);
    }
}
```

Helper: Click Function

```
import { tick, ComponentFixture } from '@angular/core/testing';

export function advance(f: ComponentFixture<any>): void {
    tick();
    f.detectChanges();
}
```

Helper: Advance Function

```
import { tick, ComponentFixture } from '@angular/core/testing';

export function advance(f: ComponentFixture<any>): void {
    tick();
    f.detectChanges();
}
```

Helper: Advance Function

```
import { tick, ComponentFixture } from '@angular/core/testing';

export function advance(f: ComponentFixture<any>): void {
    tick();
    f.detectChanges();
}
```

Helper: Advance Function

```
import { tick, ComponentFixture } from '@angular/core/testing';

export function advance(f: ComponentFixture<any>): void {
    tick();
    f.detectChanges();
}
```

Helper: Advance Function

```
import { Location } from '@angular/common';

export function expectPathToBe(
    l: Location,
    path: string,
    expectationFailOutput?: any)
{
        expect(l.path()).toEqual(path, expectationFailOutput || l.path());
}
```

Helper: expectPathToBe function

```
import { Location } from '@angular/common';

export function expectPathToBe(
    l: Location,
    path: string,
    expectationFailOutput?: any)
{
        expect(l.path()).toEqual(path, expectationFailOutput || l.path());
}
```

Helper: expectPathToBe function

```
import { Location } from '@angular/common';

export function expectPathToBe(
    l: Location,
    path: string,
    expectationFailOutput?: any)
{
        expect(l.path()).toEqual(path, expectationFailOutput || l.path());
}
```

Helper: expectPathToBe function

```
import { Location } from '@angular/common';

export function expectPathToBe(
    l: Location,
    path: string,
    expectationFailOutput?: any)
{
        expect(l.path()).toEqual(path, expectationFailOutput || l.path());
}
```

Helper: expectPathToBe function

JASMINE GOODIES

```
fdescribe('MyTest Suite', () => {

    fit('My Test', () => {
    });

});
```

Run Only This

```
fdescribe('MyTest Suite', () => {

    fit('My Test', () => {
    });

});
```

Run Only This

```
xdescribe('MyTest Suite', () => {

    xit('My Test', () => {
    });

});
```

Don't Run This

```
xdescribe('MyTest Suite', () => {

    xit('My Test', () => {
    });

});
```

Exclude Test

```
describe('HeaderComponent', () => {
    setup();
    describe('Navigation Test', navigationTests);
    describe('Create Test', createTest);
    describe('Toggle Menu Test', toggleMenuTest);
});

function setup() {
    beforeEach(async(() => {
    }));

    beforeEach(() => {
    });
}

function navigationTests() {
    beforeEach(() => {
        allLinkDes = fixture.debugElement.queryAll(By.directive(RouterLinkWithHref));
    });
}
```

Grouping Tests

```
describe('HeaderComponent', () => {
    setup();
    describe('Navigation Test', navigationTests);
    describe('Create Test', createTest);
    describe('Toggle Menu Test', toggleMenuTest);
});

function setup() {
    beforeEach(async(() => {
    }));

    beforeEach(() => {
    });
}

function navigationTests() {
    beforeEach(() => {
        allLinkDes = fixture.debugElement.queryAll(By.directive(RouterLinkWithHref));
    });
}
```

Grouping Tests

```
describe('HeaderComponent', () => {
    setup();
    describe('Navigation Test', navigationTests);
    describe('Create Test', createTest);
    describe('Toggle Menu Test', toggleMenuTest);
});

function setup() {
    beforeEach(async(() => {
    }));

    beforeEach(() => {
    });
}

function navigationTests() {
    beforeEach(() => {
        allLinkDes = fixture.debugElement.queryAll(By.directive(RouterLinkWithHref));
    });
}
```

Grouping Tests

```
describe('HeaderComponent', () => {
    setup();
    describe('Navigation Test', navigationTests);
    describe('Create Test', createTest);
    describe('Toggle Menu Test', toggleMenuTest);
});

function setup() {
    beforeEach(async(() => {
    }));

    beforeEach(() => {
    });
}

function navigationTests() {
    beforeEach(() => {
        allLinkDes = fixture.debugElement.queryAll(By.directive(RouterLinkWithHref));
    });
}
```

Grouping Tests

```
describe('HeaderComponent', () => {
    setup();
    describe('Navigation Test', navigationTests);
    describe('Create Test', createTest);
    describe('Toggle Menu Test', toggleMenuTest);
});

function setup() {
    beforeEach(async(() => {
    }));

    beforeEach(() => {
    });
}

function navigationTests() {
    beforeEach(() => {
        allLinkDes = fixture.debugElement.queryAll(By.directive(RouterLinkWithHref));
    });
}
```

Grouping Tests

Unit Tests Are ***Proof*** That My Code Does What I ***Think*** It Does

Unit Testing Will Make You Faster

Confirm Functionality

Check Regressions

Pinpoint Bugs

Document Functionality

"Don't let the fear that testing can't catch all bugs stop you from writing the tests that will catch most bugs"

Martin Fowler
Refactoring

Resources

Angular Unit Testing Guide:

angular.io/guide/testing

Jasmine Intro:

jasmine.github.io/2.0/introduction.html

Code Repository:

github.com/digitaldrummerj/angular-tutorial-code/tree/chapter-unit-test

THANK YOU

slideshare.net/digitaldrummerj

digitaldrummerj.me

digitaldrummerj

THANK YOU

slideshare.net/digitaldrummerj

digitaldrummerj.me

digitaldrummerj