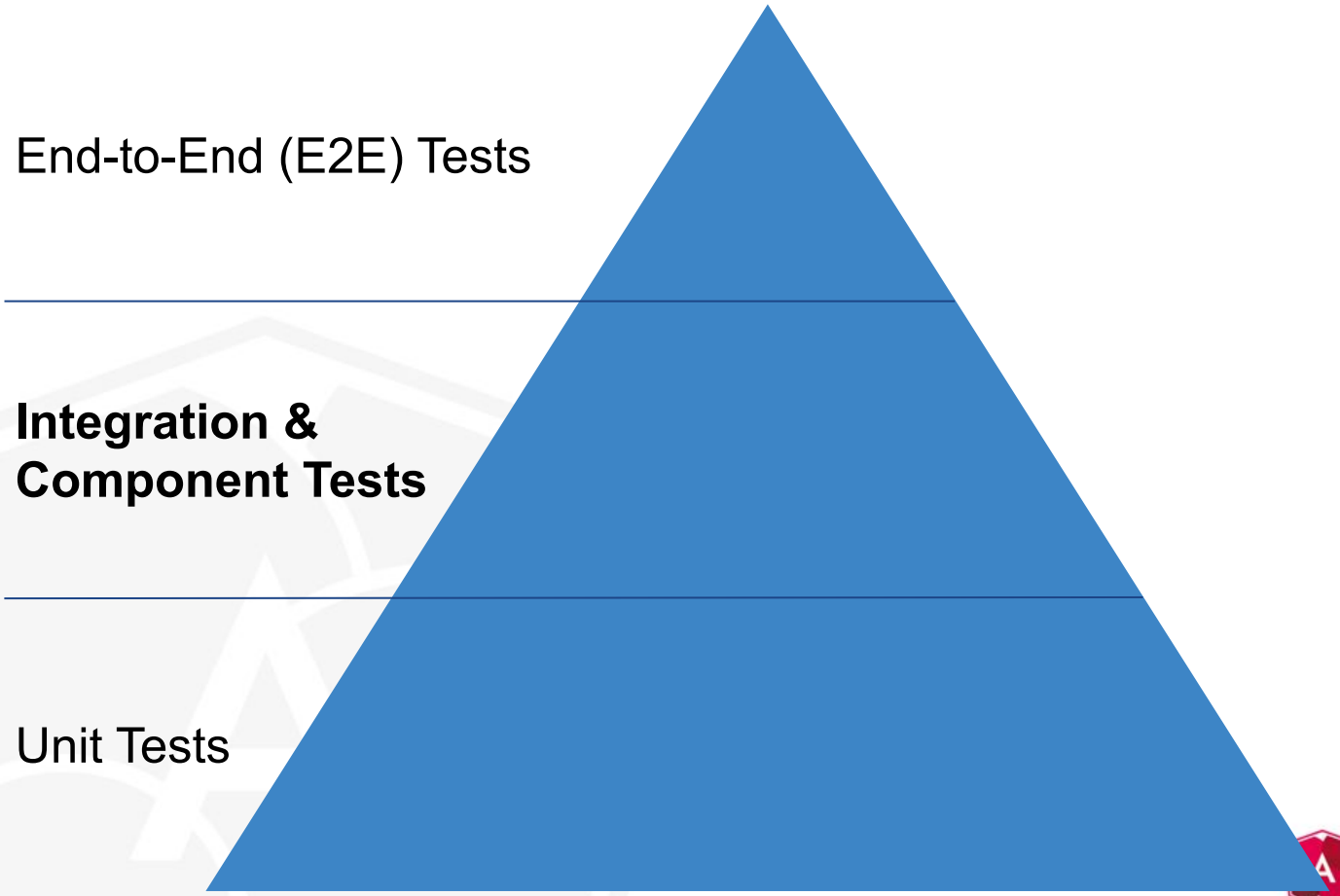
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Angular Testing

5 - Component Tests - Functional



ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Test Specs

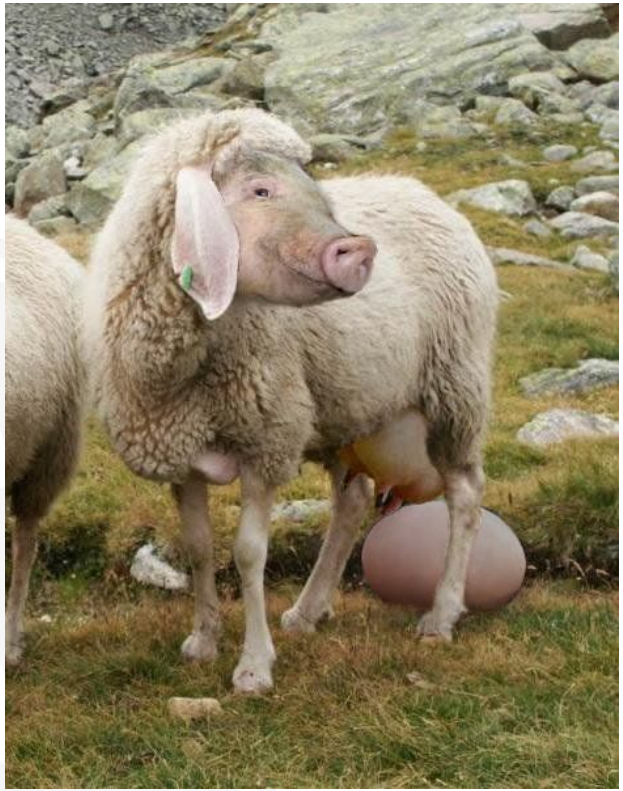
Application Code

JSDom

@angular/testing

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

TestBed



ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

TestModule

```
const fixture = TestBed.configureTestingModule({
  declarations: [RequestInfoComponent],
  imports: [ReactiveFormsModule],
  providers: [{ provide: AddressLookuper, useValue: null }]
}).createComponent(AddressComponent);


const component = fixture.componentInstance;
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Interacting with the DOM

```
fixture.detectChanges();

const button = fixture.debugElement

  .query(By.css('button'))

  .nativeElement as HTMLButtonElement;

button.click();
```

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Remember fakeAsync?

Proper Handling of Event Listeners

```
it("should trigger search on click", fakeAsync(() => {
  fixture.detectChanges();
  const button = fixture.debugElement.query(By.css("button"))
    .nativeElement as HTMLButtonElement;
  button.click();
  tick();

  expect(lookuper.lookup).toHaveBeenCalled();
}));
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Snapshot

- Serialization Feature
- Good fit when it comes to rough template checks
- No CSS
- Is not an image, e.g. png or jpg.

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Lab Time

Nested Components

1. "Three Monkeys"
2. Component Stubs
3. ng-mocks
4. Import all Modules

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Nested Components - Three Monkeys

```
it('should mock components in 🙈🙉🙊 style, () => {
  const fixture = TestBed.configureTestingModule({
    declarations: [RequestInfoComponent],
    schemas: [NO_ERRORS_SCHEMA]
  }).createComponent(RequestInfoComponent);
  fixture.detectChanges();
  expect(true).toBe(true);
});
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Nested Components - Stubs

```
it('should stub the components', () => {
  @Component({ selector: 'mat-form-field', template: '' })
  class MatFormField {}

  const fixture = TestBed.configureTestingModule({
    declarations: [RequestInfoComponent, MatFormField],
    imports: [ReactiveFormsModule]
  }).createComponent(RequestInfoComponent);

  fixture.detectChanges();
  expect(true).toBe(true);
});
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Nested Components - Mocked

```
it('should stub the components', () => {
  const fixture = TestBed.configureTestingModule({
    declarations: [RequestInfoComponent, MockComponent(MatFormField)],
    imports: [ReactiveFormsModule]
  }).createComponent(RequestInfoComponent);

  fixture.detectChanges();
  expect(true).toBe(true);
});
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Nested Components - Import Modules

```
it('should import the modules', () => {
  const fixture = TestBed.configureTestingModule({
    declarations: [RequestInfoComponent],
    imports: [ReactiveFormsModule, MatFormFieldModule, MatHintModule, MatLabelModule]
}).createComponent(RequestInfoComponent);

  fixture.detectChanges();

  expect(true).toBe(true);
});
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Reducing Boilerplate

Customisable Factory Methods

Libraries

Testing Library

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Spectator

- Wrapper for TestBed
  - Queries
  - Factory method for component
  - Host
  - Eventtriggering
- Special Cases
  - Pipes
  - Directives

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Spectator: Pipe Test

```
describe('Customer Pipe', () => {
  const customer: Customer = {
    id: 1, firstname: 'Karl', name: 'Feiersberger', birthdate: '1991-01-20', country: 'AT'
  };

  @Component({template: '<p>{{ customer | customer}}</p>'})
  class CustomerPipeTestComponent {
    @Input() customer: Customer = customer;
  }

  const createPipe = createPipeFactory({
    pipe: CustomerPipe,
    host: CustomerPipeTestComponent
  });

  it("should print the user's name", () => {
    const spectator = createPipe();
    expect(spectator.element).toHaveText('Feiersberger, Karl');
  });
});
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Spectator: Directive Test

```
describe("Don't leave me", () => {
  const createDirective = createDirectiveFactory(DontLeaveMeDirective);


  it('should show the dog image', () => {
    const spectator = createDirective('<img appDontLeaveMe src="/dummy.jpg">');
    spectator.dispatchMouseEvent(spectator.element, 'mouseleave');


    expect(spectator.element).toHaveAttribute('src', '/assets/dontleaveme.jpg');
  });
});
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

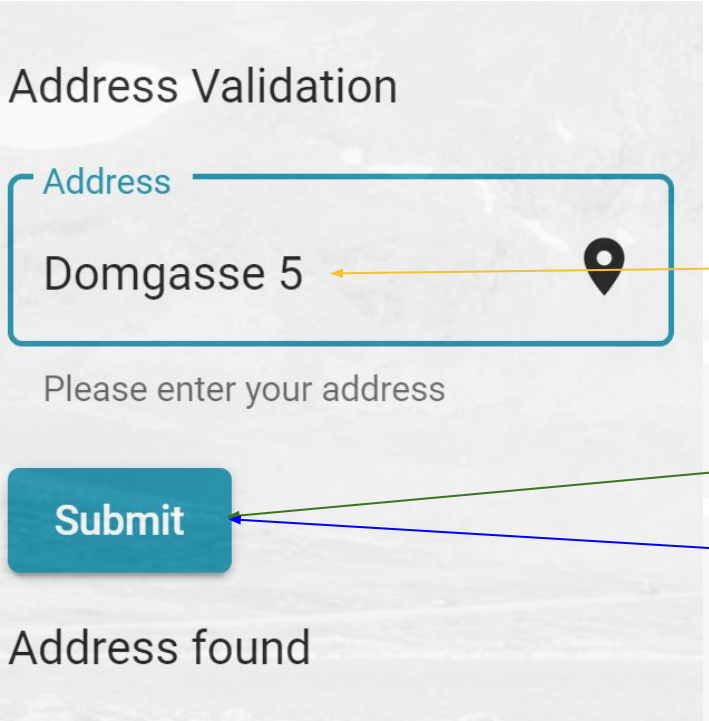
Harnesses

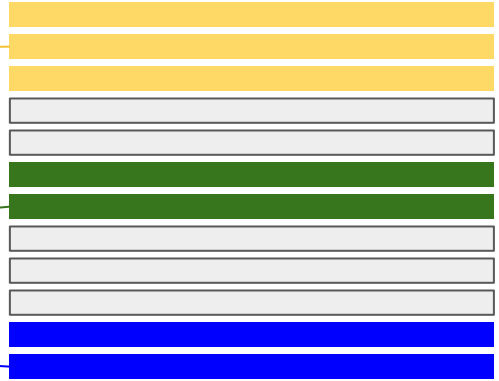
Taming the Beast…

Test Harnesses

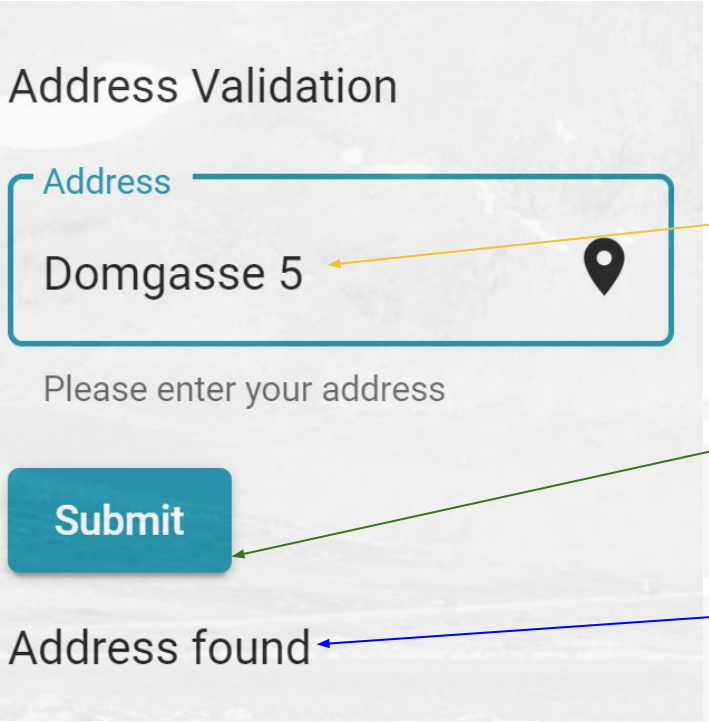
- Page Object Models for Component Tests
- Available since Angular v9
- Provide a test abstraction for components
- Developed by @angular/material
- Full coverage for material since v11
- Reduces code size significantly
  - Better Readability
  - Better Maintainability

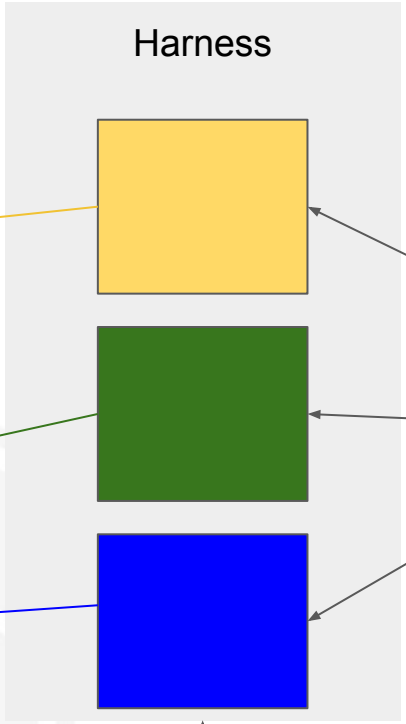
ANGULAR ARCHITECTS
INSIDE KNOWLEDGE





ANGULAR ARCHITECTS
INSIDE KNOWLEDGE





- Element Selection
- Change Detection
- Asynchronity
- Rendering

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Creating a Harness

```
export class RequestInfoComponentHarness extends ComponentHarness {

  static hostSelector = "app-address";

  protected getButton = this.locatorFor("button[type=submit]");


  async submit(): Promise<void> {

    const button = await this.getButton();

    return await button.click();

  }

}
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Using a Harness

```
it("should use the harness", async () => {
  // setup TestModule…

  const harness = await TestbedHarnessEnvironment.harnessForFixture(
    Fixture,
    RequestInfoComponentHarness
  );
  await harness.submit();
  // expect something
});
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Lab Time

Two Competing Schools of Unit Testing

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Differences

London

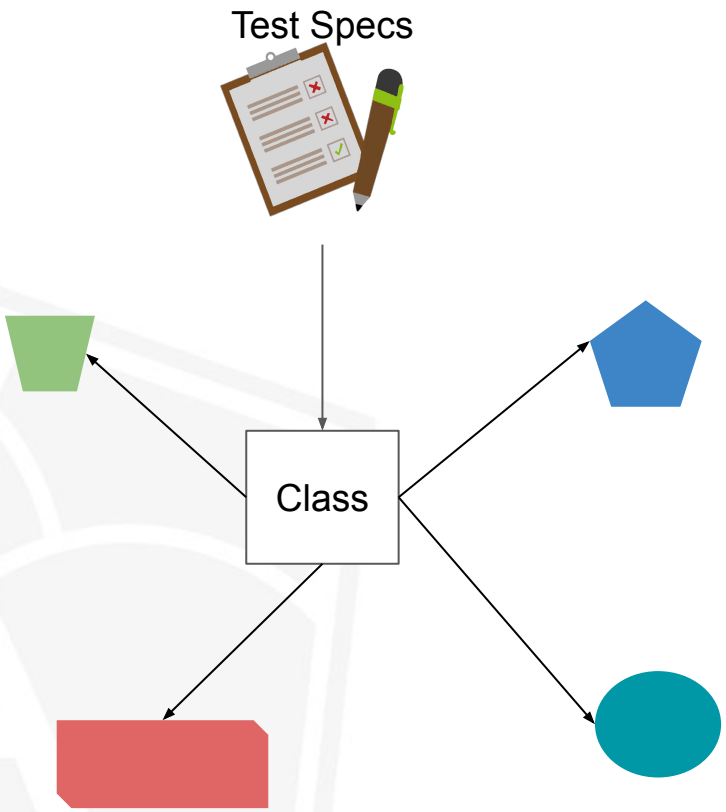
- Unit is a Class
- Mock everything except the class
  - Very tightly coupled to implementation
  - Lots of Code for Mocking
  - Refactoring not really possible
- Great Code Quality

Detroit

- Unit is Behaviour
- Mock out-of-system dependencies
- Works against a "public contract"
  - No usage of private function
  - Refactoring without changing tests

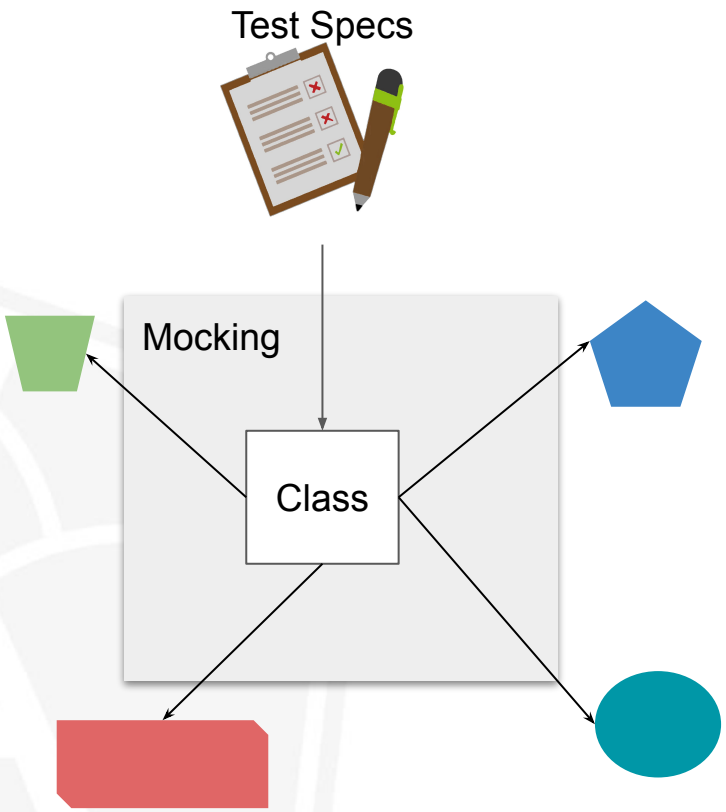
ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

London Style



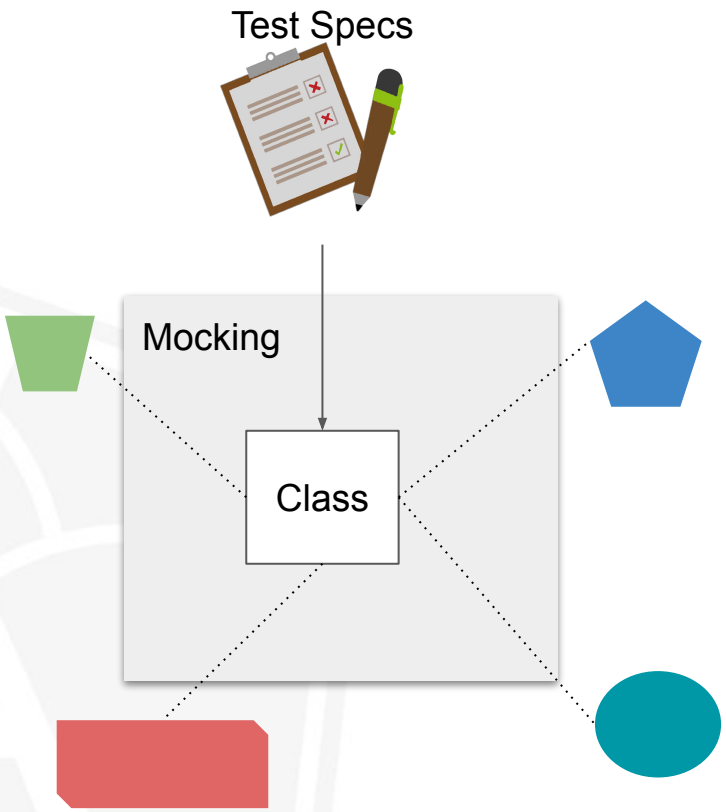
ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

London Style



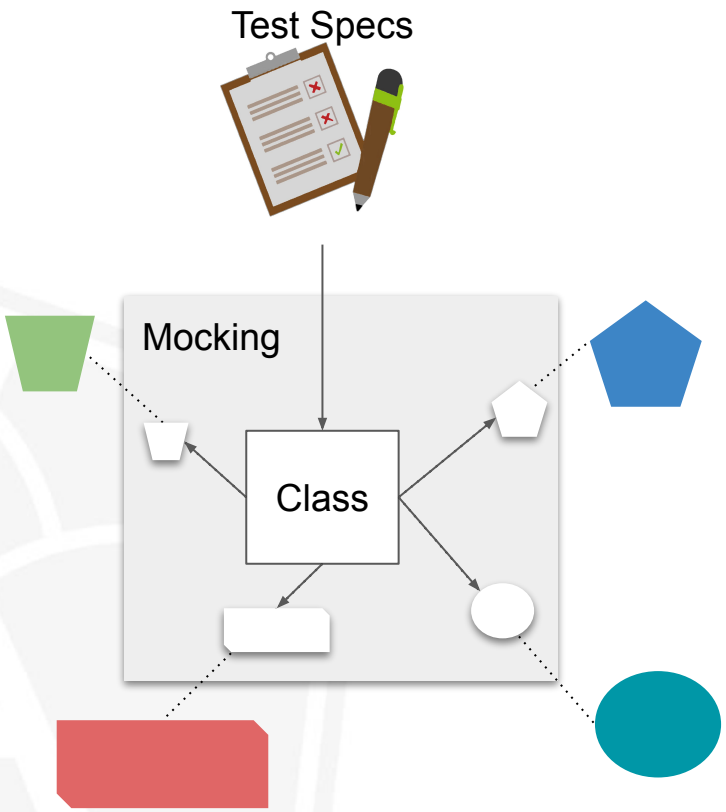
ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

London Style



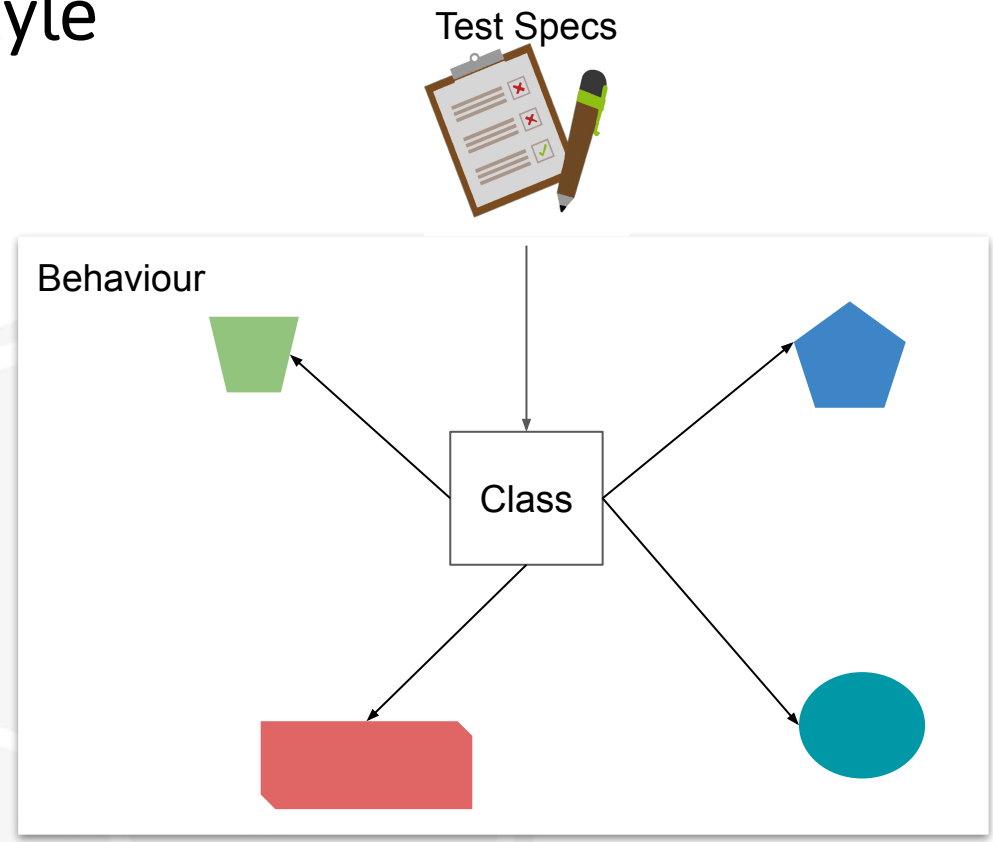
ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

London Style



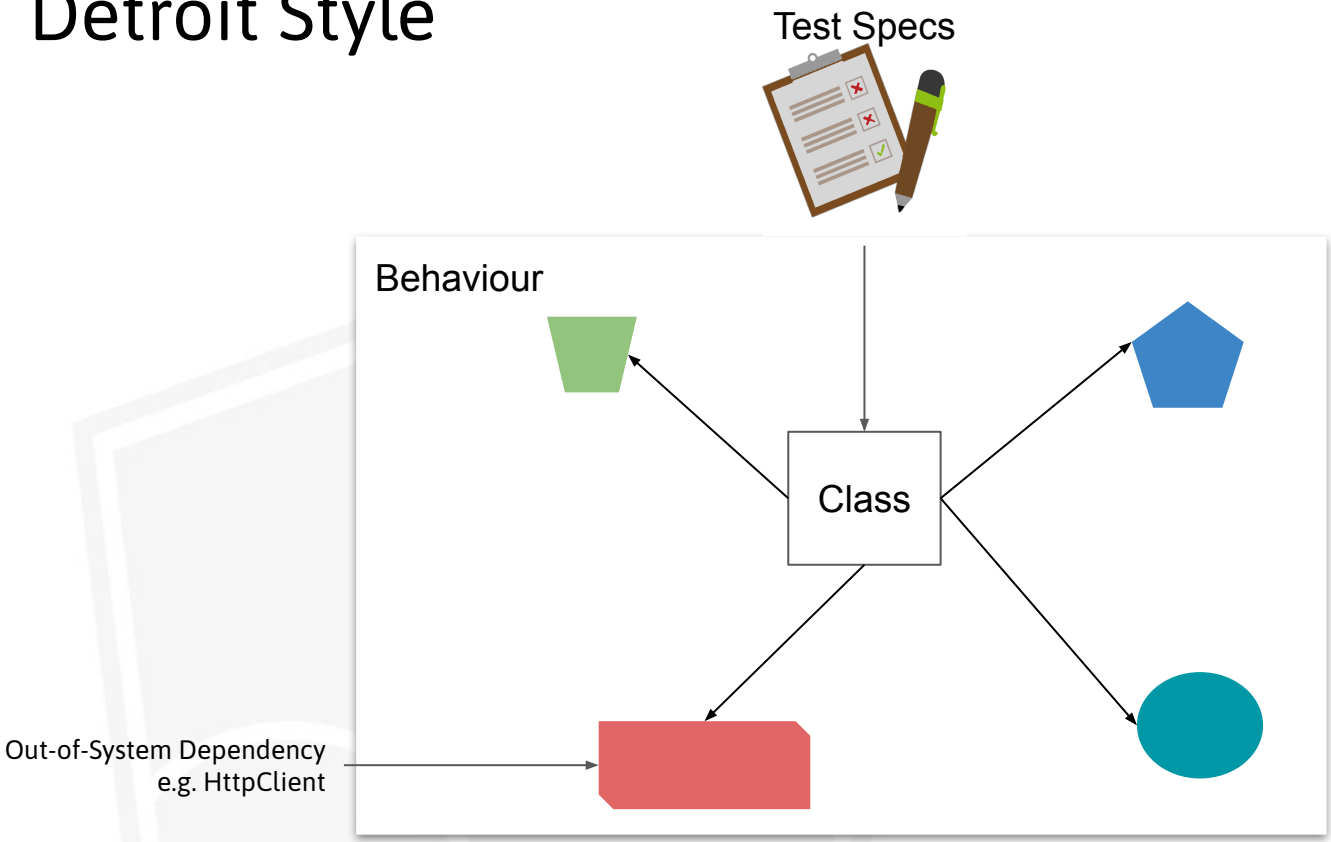
ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Detroit Style



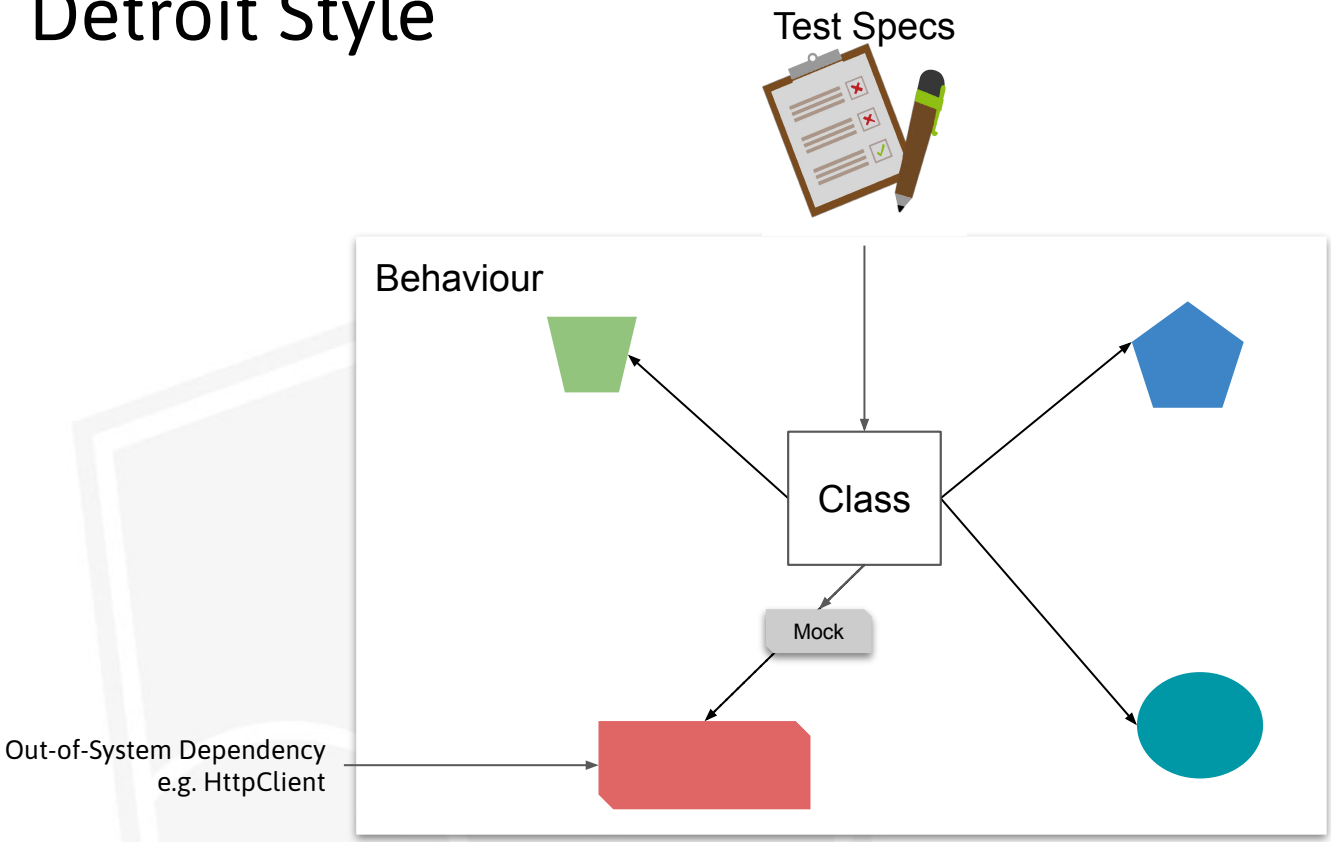
ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Detroit Style



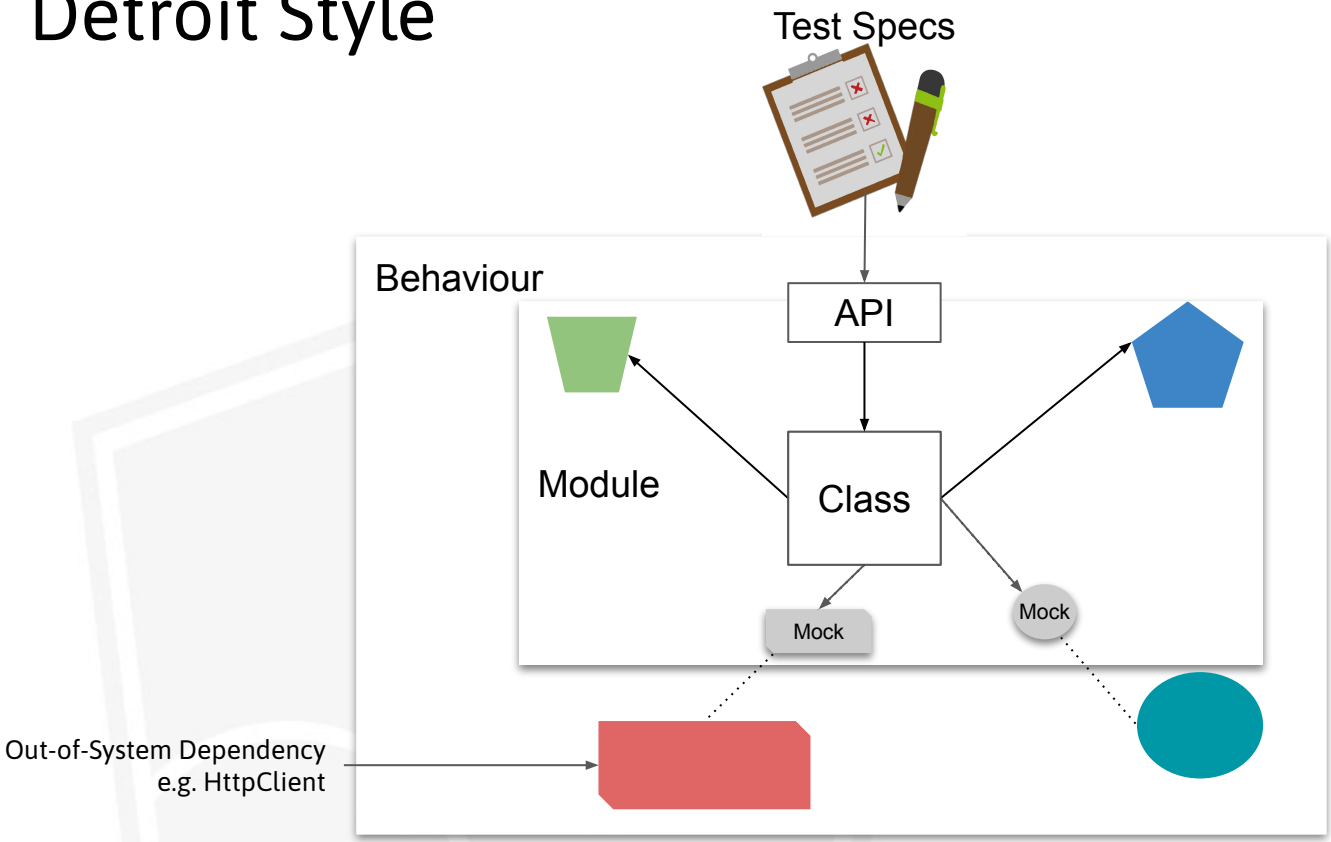
ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Detroit Style



ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Detroit Style



ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

What should it be?

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Refactoring
Precision
& Speed

HttpTest

```
it("should use Angular's http mock", () => {
  TestBed.configureTestingModule({
    declarations: [RequestInfoComponent],
    imports: [ReactiveFormsModule, HttpClientTestingModule],
  });

  const httpController = TestBed.inject(HttpTestingController);
  const fixture = TestBed.createComponent(RequestInfoComponent);
  fixture.componentInstance.search();

  const [request] = httpController.match((req) => !!req.url.match(/nominatim/));
  request.flush([{ street: "Domgasse", streetNumber: 5 }]);
  expect(lookupResult.textContent.trim()).toBe("Address found");
});
```

Instead of HttpClientModule

Runs AFTER http request

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

RoutingTest

- RouterTestingModule provides routing functionality for tests
- Location can verify the expected url
- RoutingConfiguration is required

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

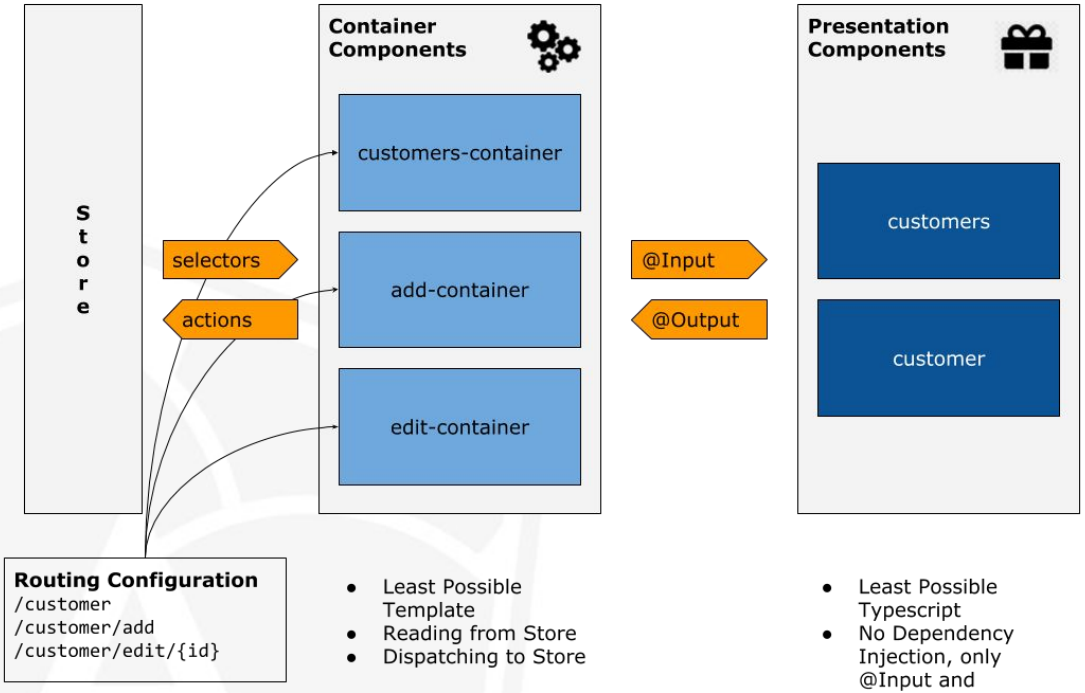
Lab Time

Towards a Testable Architecture

- Level 1 - Atomic
  - Unit Tests for complex Container Components, Services, Functions,...
  - Visual Regression
  - RxJS Marbles
- Level 2 - Logical Groups
  - Interplay between Container & Presentational Components
  - Complex Components (DataGrid)
  - State Management
- Level 3 - Integration Tests
  - Per Domain
  - Per Feature
- Level 4 - E2E Tests
  - Per User Story

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Container & Presentation Components



- Least Possible Template
- Reading from Store
- Dispatching to Store

- Least Possible Typescript
- No Dependency Injection, only @Input and @Output
- No Observables!

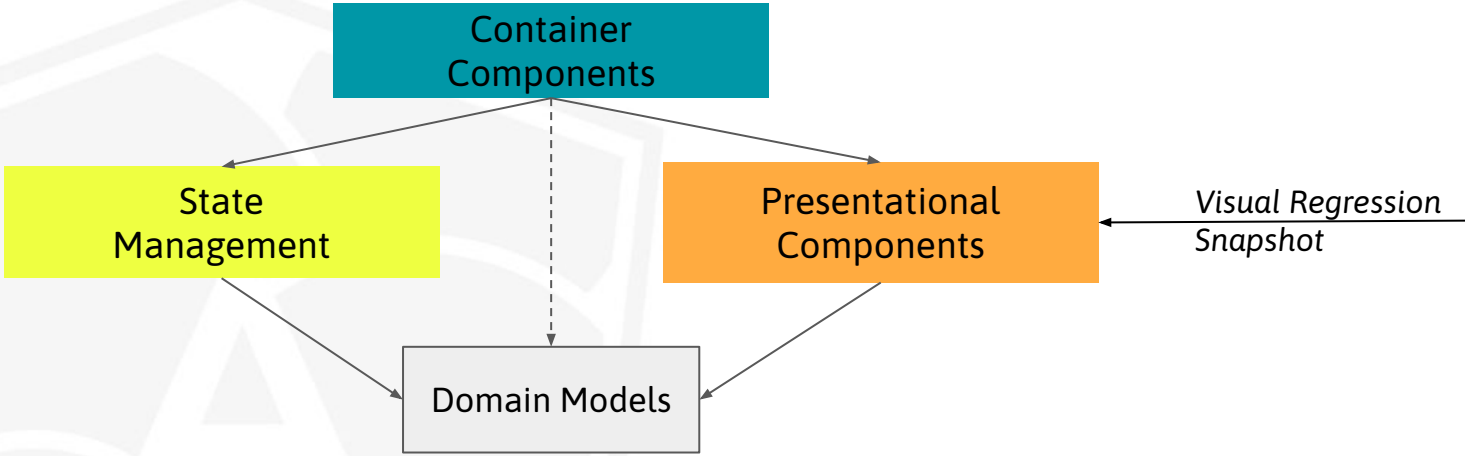
ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Presentational Component

- Easy to setup in Storybook
- Decoupled from business logic
- No Dependency Injection
- Use primitive types for @Input

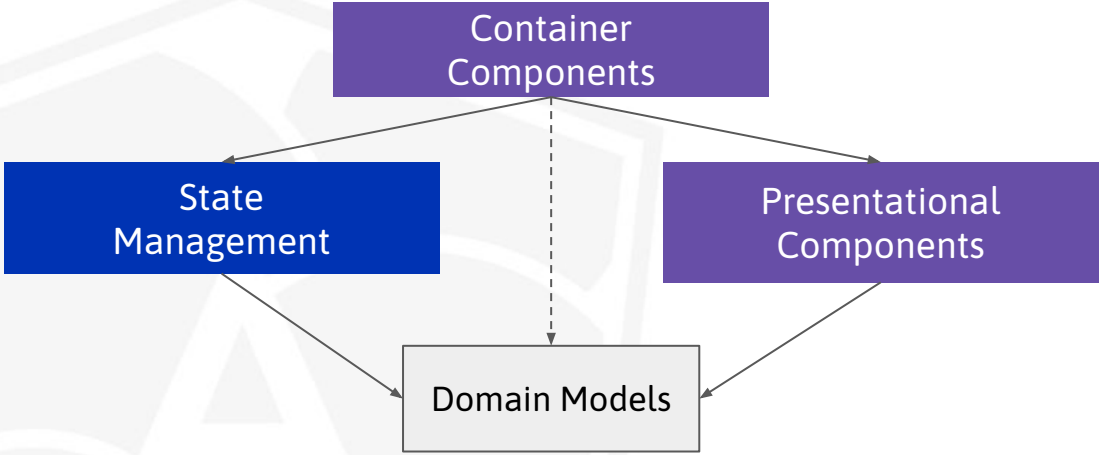
ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Testable Architecture
Level 1 - Atomic



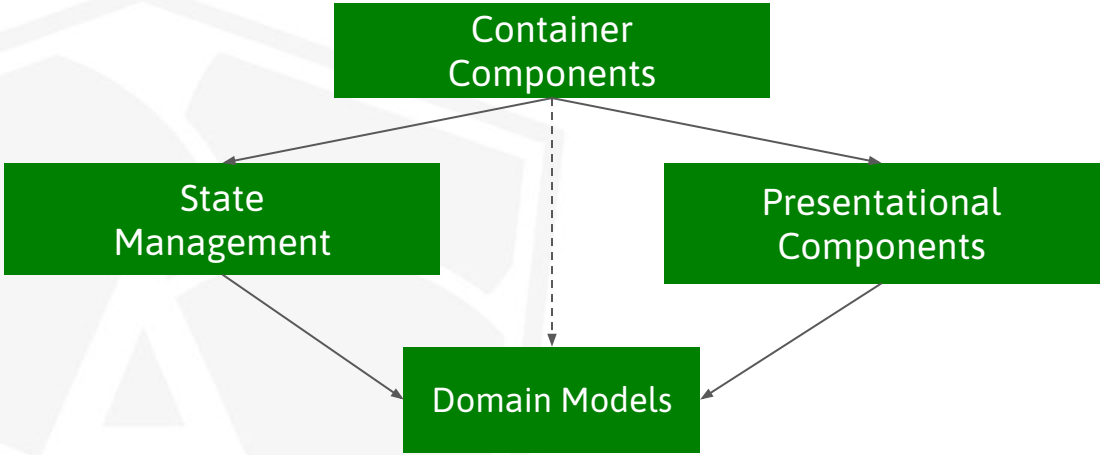
ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Testable Architecture
Level 2 - Logical Groups



ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Testable Architecture
Level 3- Integration Tests



ANGULAR ARCHITECTS
INSIDE KNOWLEDGE