ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Angular Testing

4 - Component Tests Advanced

Mocking Components

1. "Three Monkeys"
2. Component Stubs
3. ng-mocks
4. Don't mock!

Mocking 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);
});
```

Mocking Components - Manually

```
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);
});
```

Mocking Components - ng-mocks

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

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

Mocking Components - Don't

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


  fixture.detectChanges();


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

Reducing Boilerplate

Test Setup

Approaches

1. beforeEach
    a. Default Configuration by Angular CLI
    b. All tests with same setup
    c. Simple situations
2. Nested describes, aka. Contexts
    a. Advanced Scenarios
    b. Limited amount of TestBed configuration
3. Factory methods
    a. The test has full control - not the Test Suite
    b. Most Flexible

Popular Libraries

<SPECTATOR>

Testing Library

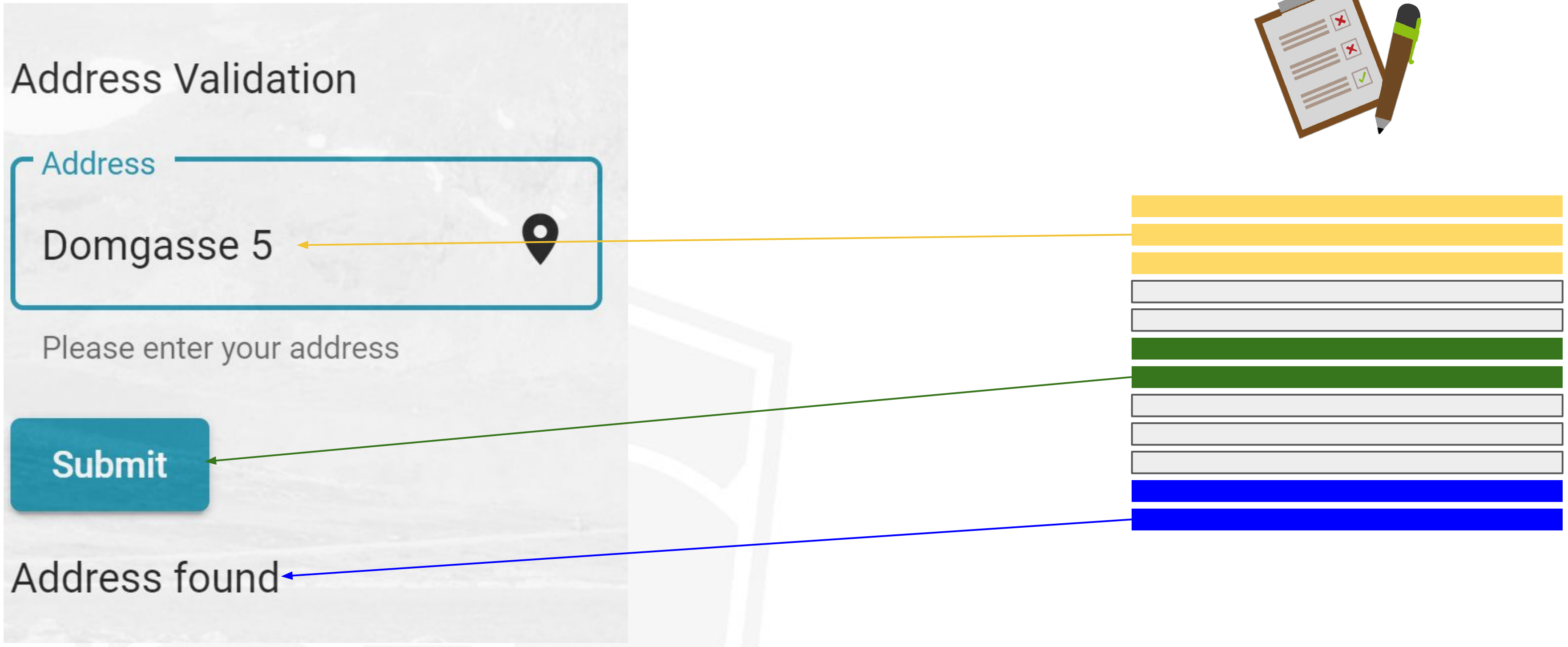
Harnesses

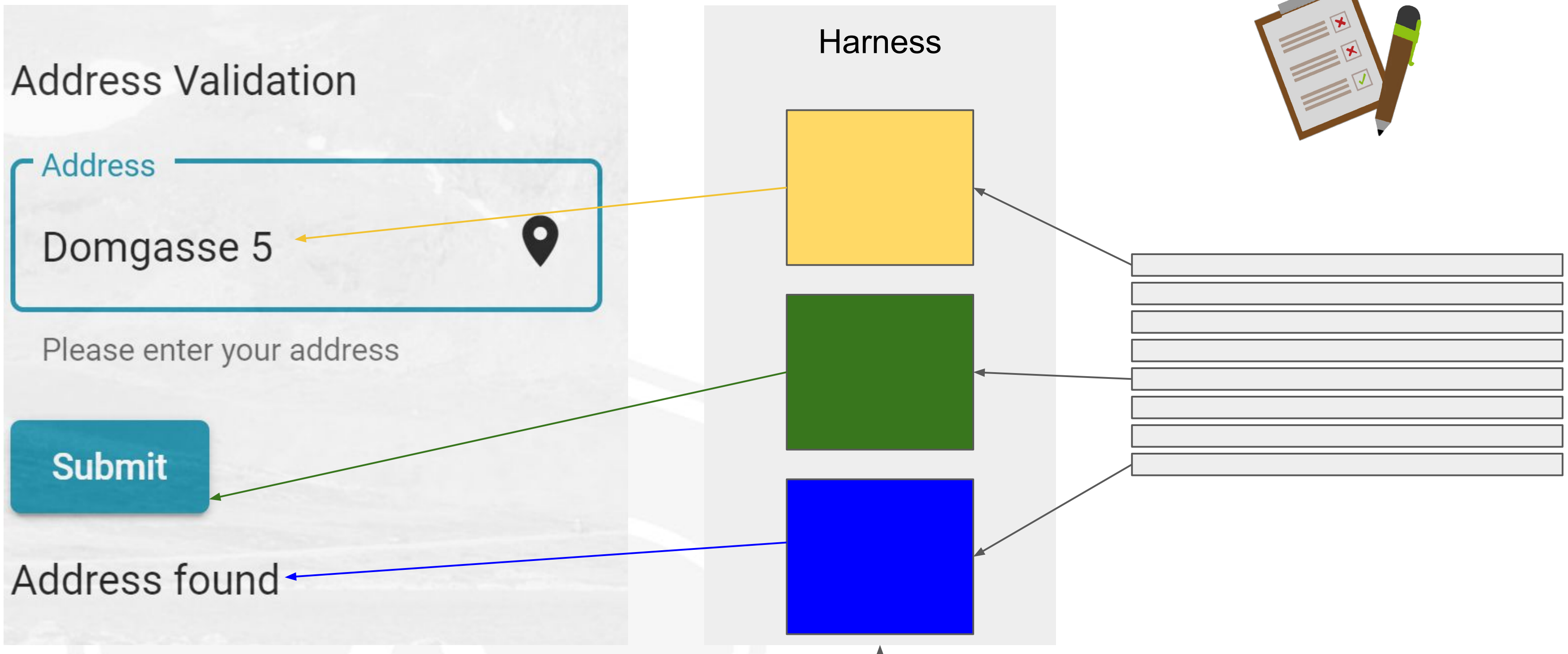
Taming the Beast...

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


Address Validation
Address
Domgasse 5
Please enter your address
Submit
Address found



- Element Selection
- Change Detection
- Asynchronity
- Rendering

Creating a Harness

```
export class RequestInfoComponentHarness extends ComponentHarness {

  static hostSelector = "app-request-info";

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


  async submit(): Promise<void> {

    const button = await this.getButton();

    return button.click();

  }

}
```

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