



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Angular Testing

5 - Component & Integration Tests

End-to-End (E2E) Tests

**Integration &
Component Tests**

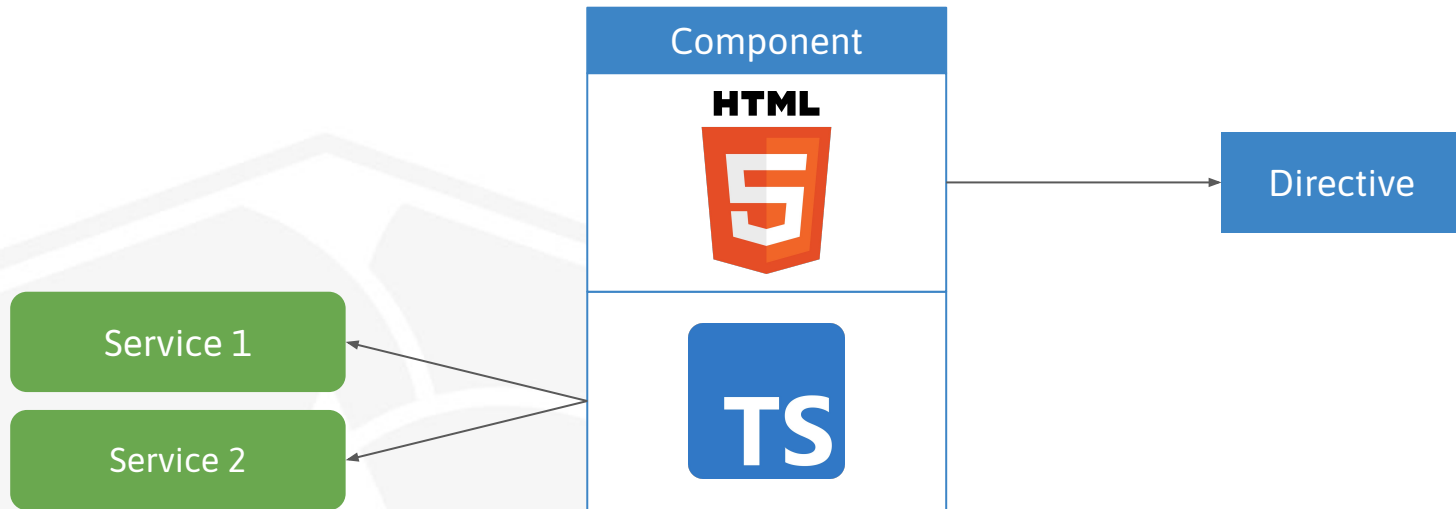
Unit Tests

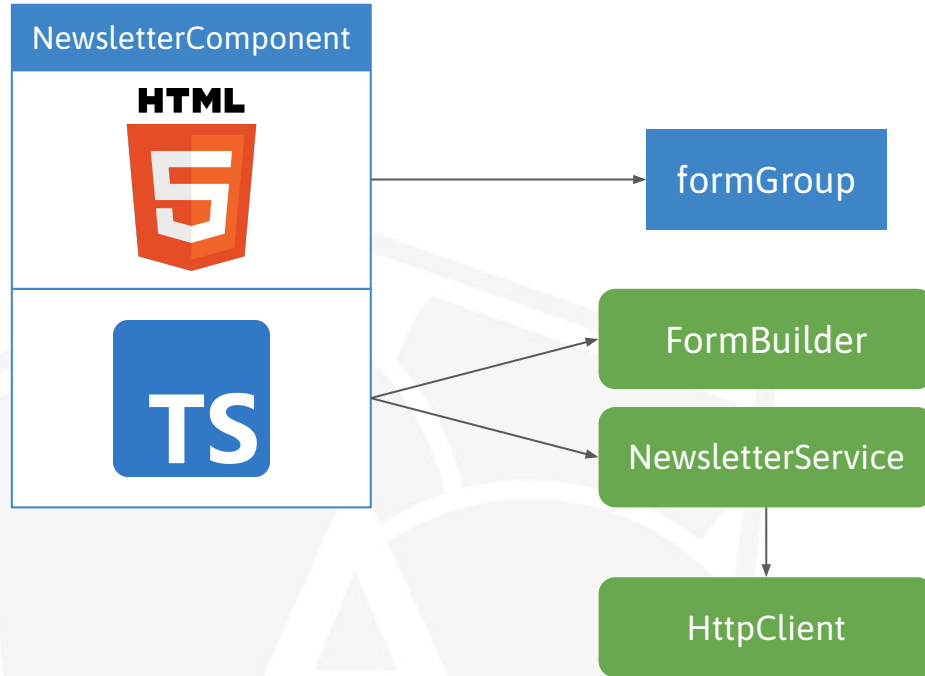


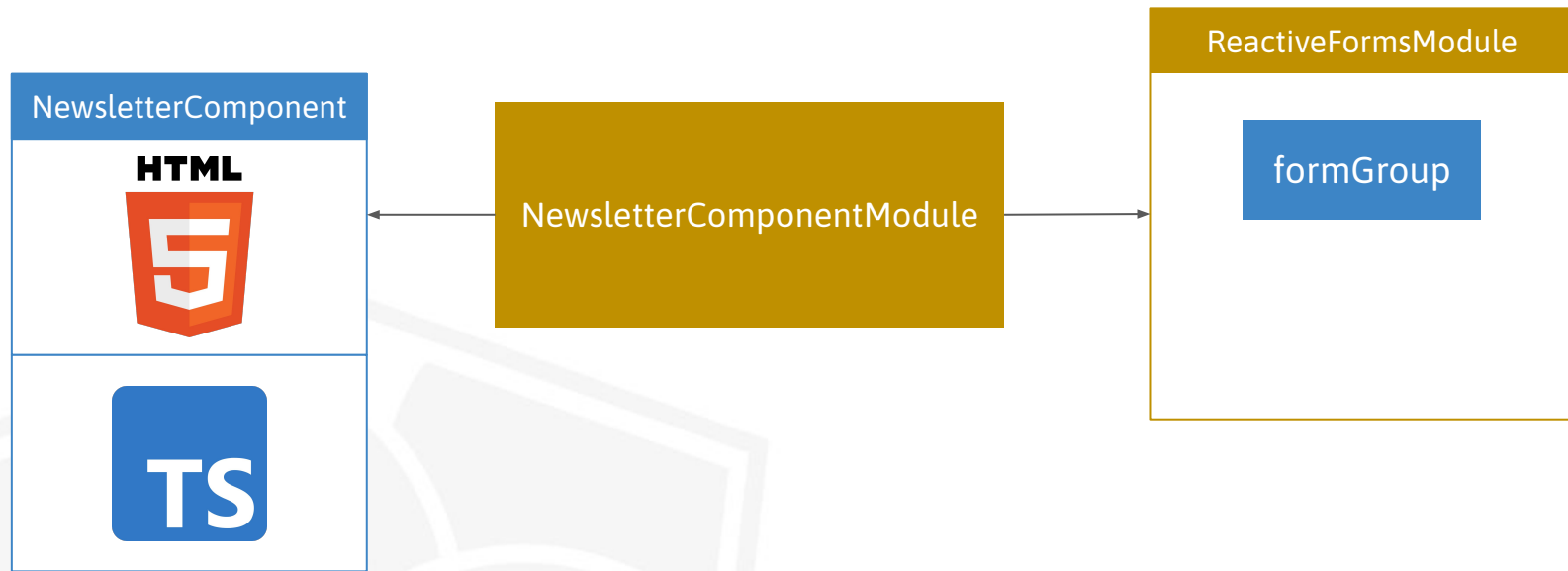
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Configuring the TestingModule





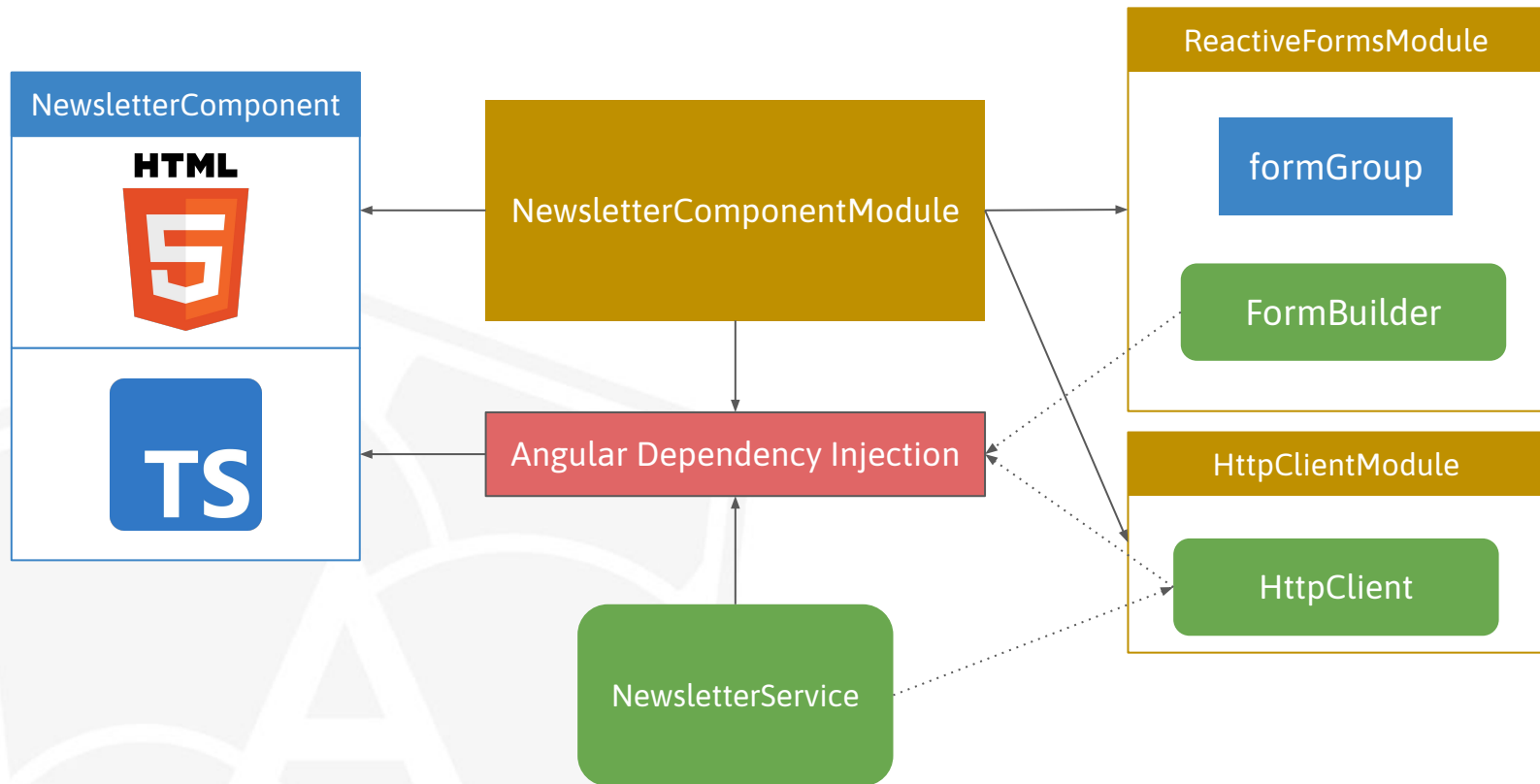




Different Ways of providing Services

1. `@Injectable({providedIn: 'root'})`
2. `providers` property of an `NgModule`
3. `providers` property of a `Component/Pipe/Directive`



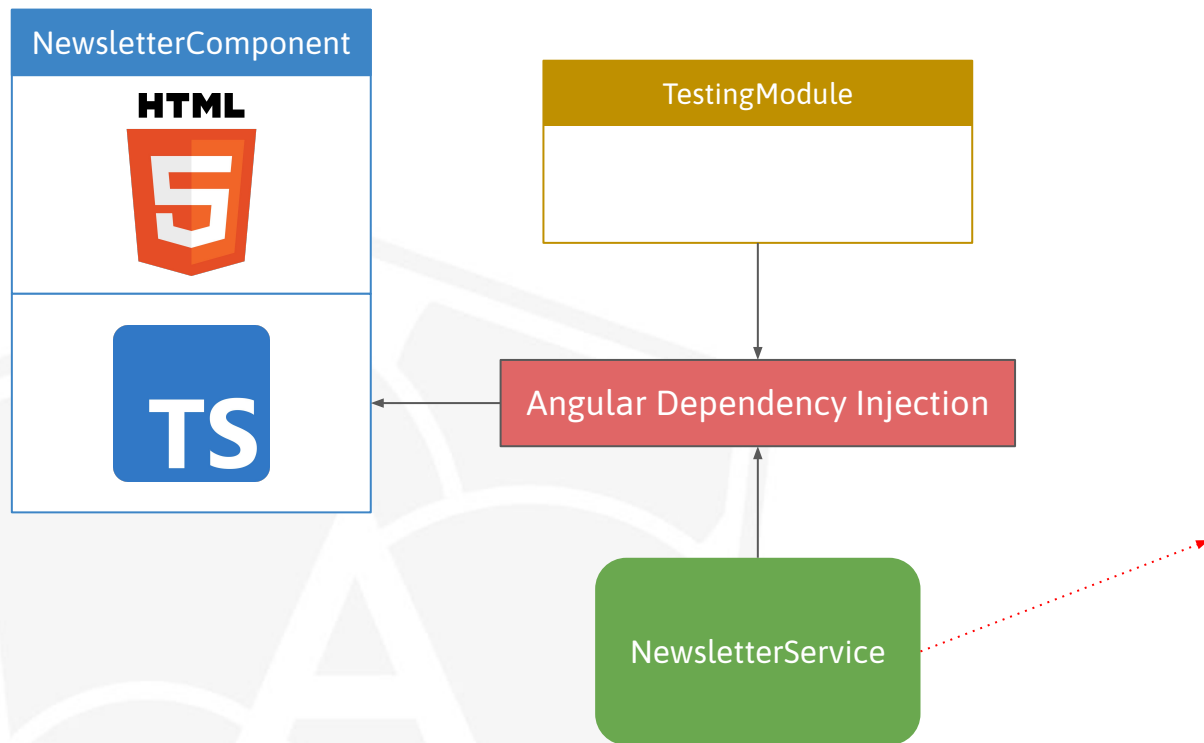


Testing Module

```
const fixture = TestBed.configureTestingModule({})
```



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

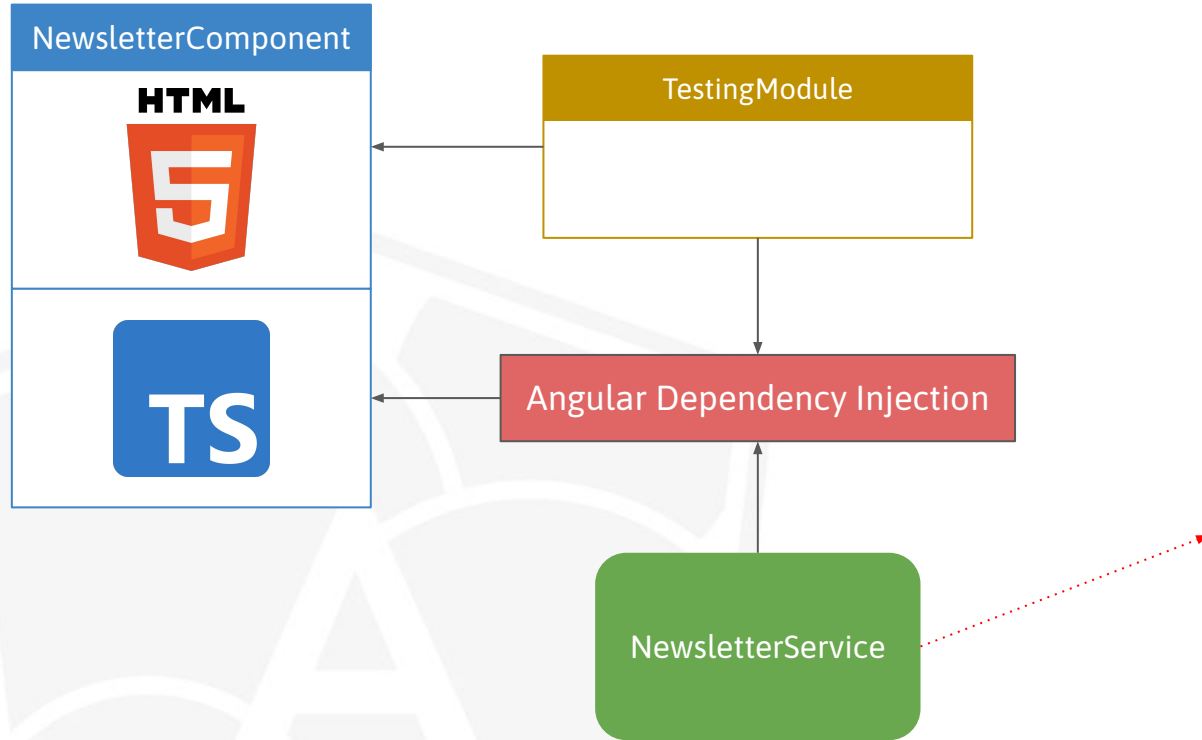


Testing Module

```
const fixture = TestBed.configureTestingModule({  
  declarations: [NewsletterComponent]  
})
```



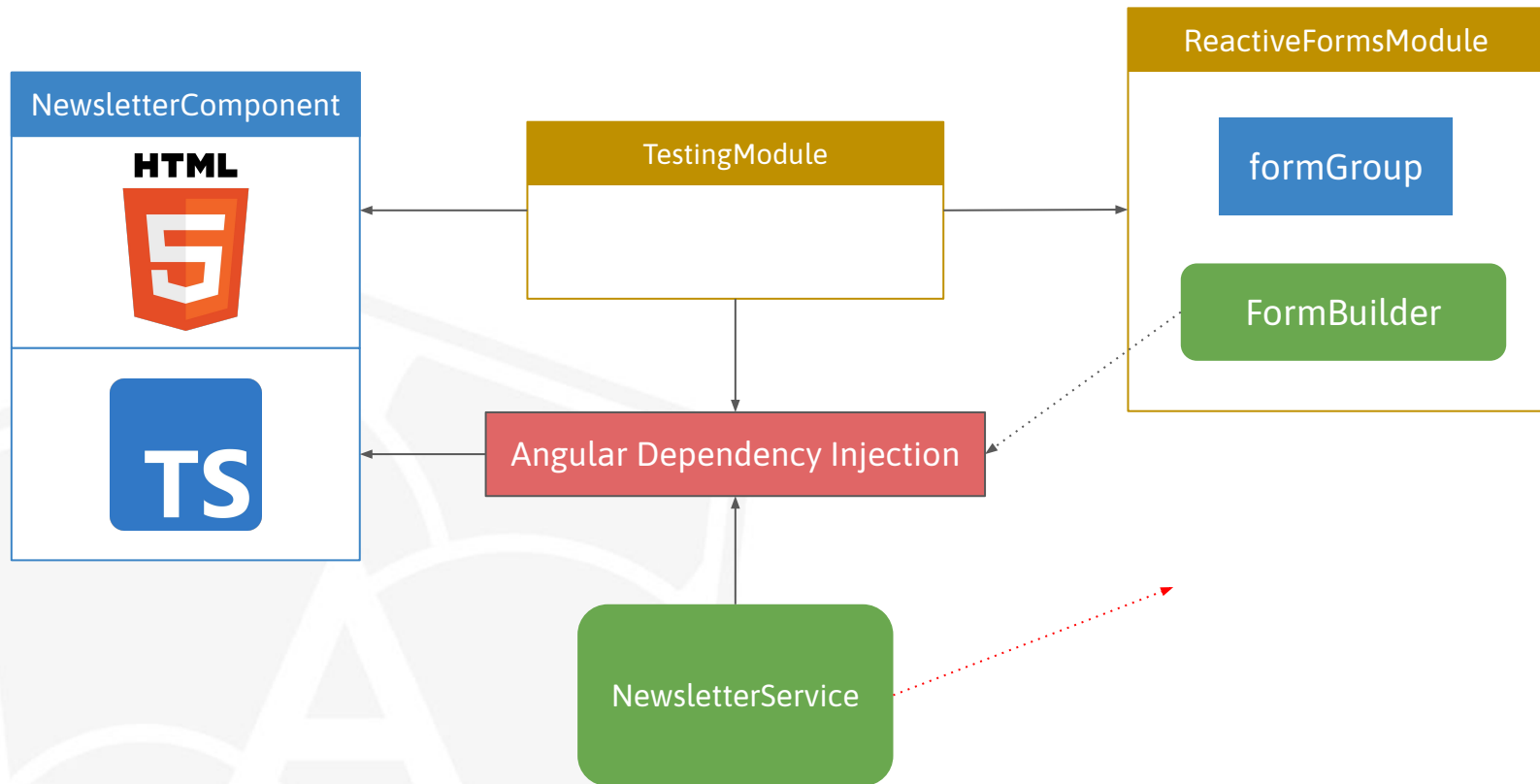
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE



Testing Module

```
const fixture = TestBed.configureTestingModule({  
  declarations: [NewsletterComponent],  
  imports: [ReactiveFormsModule]  
})
```

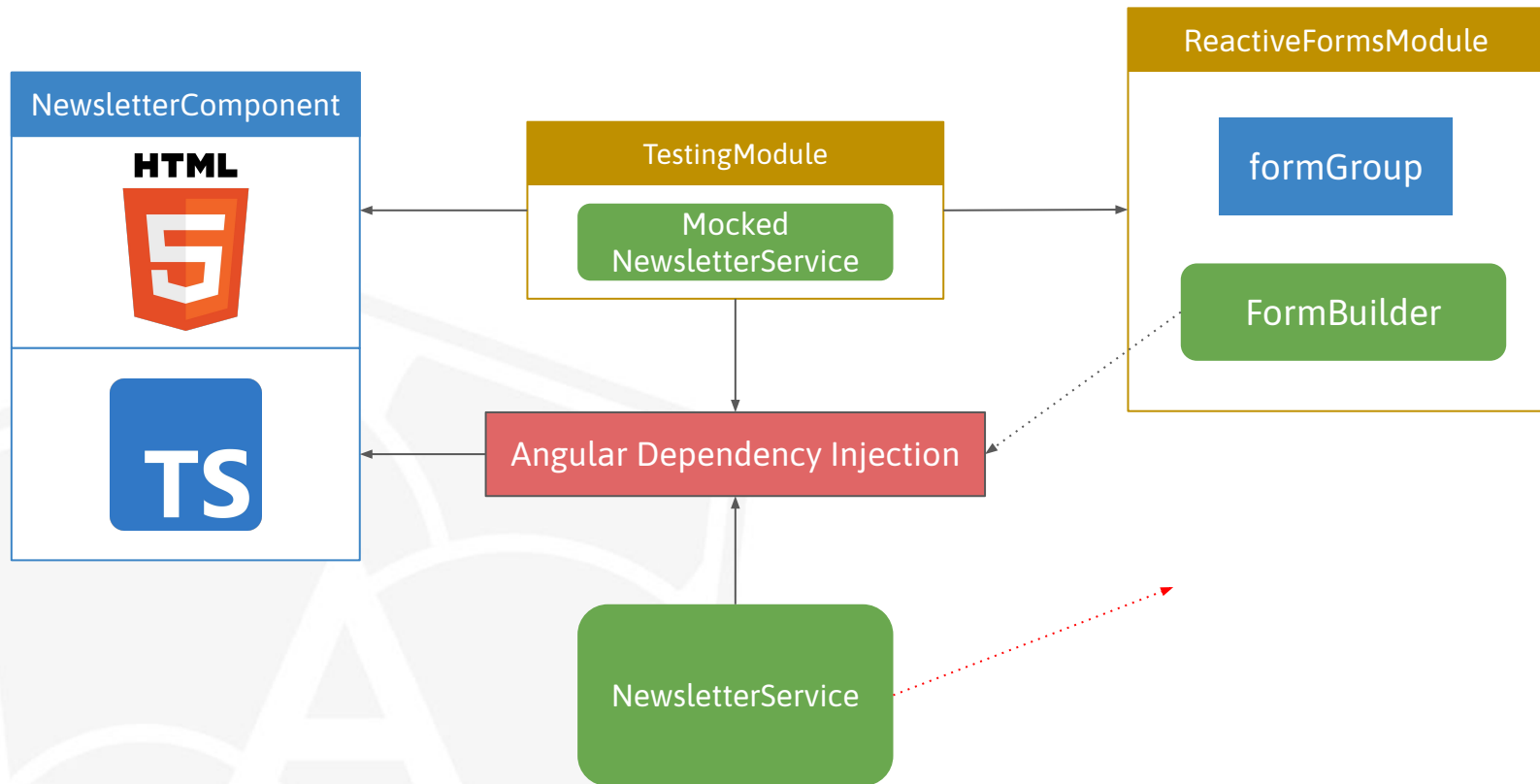


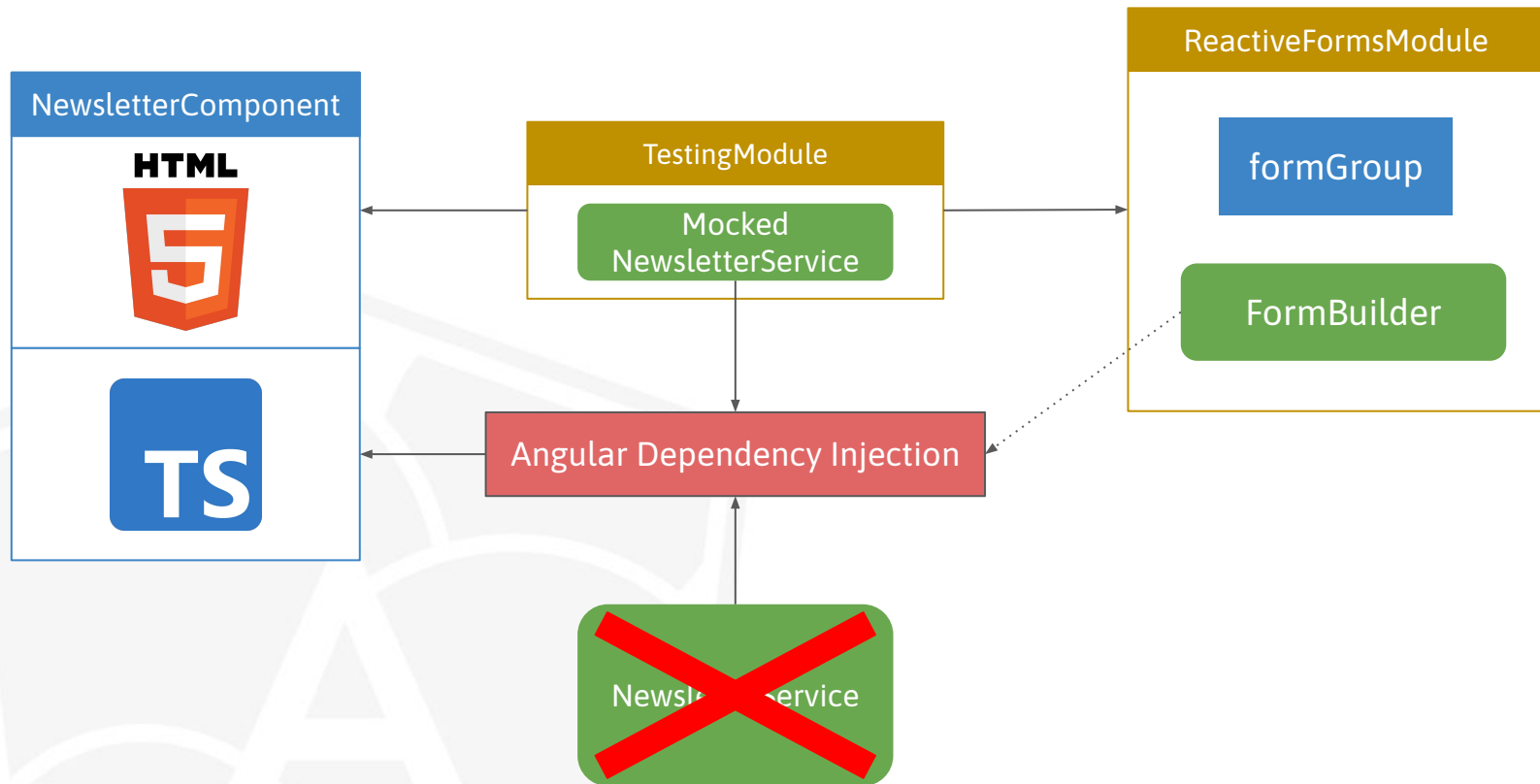


Adding Dependency Injection

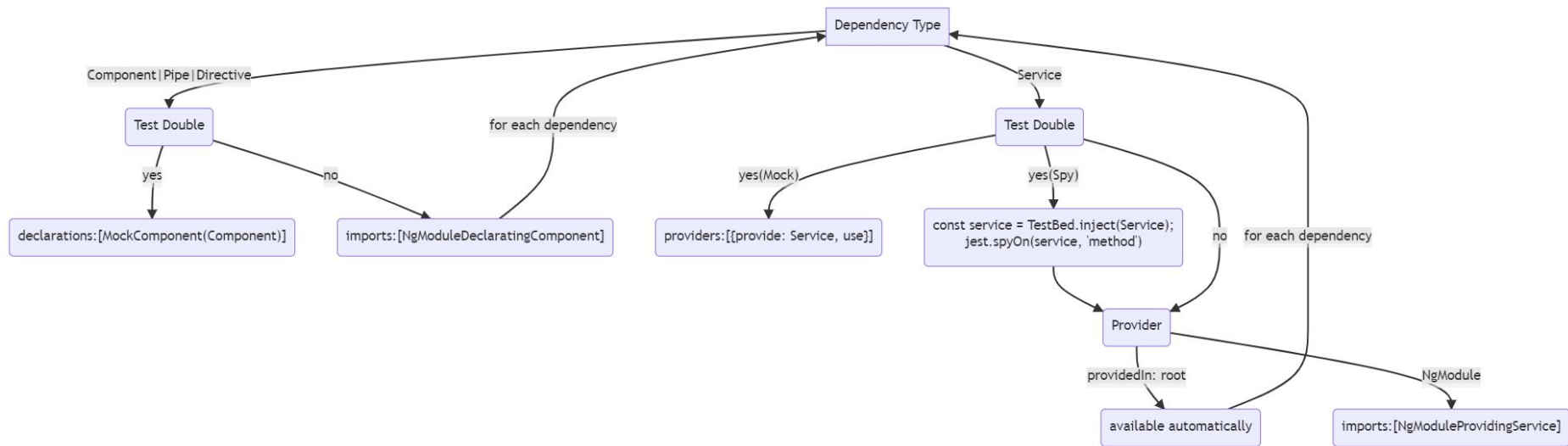
```
const fixture = TestBed.configureTestingModule({  
  declarations: [NewsletterComponent],  
  providers: [{ provide: NewsletterService, useValue: null }]  
})
```







Decision Tree: TestingModule



Mock Visual Elements



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);  
});
```



Cypress Component Test Runner



Classic testing pyramide

End-to-End (E2E) Tests

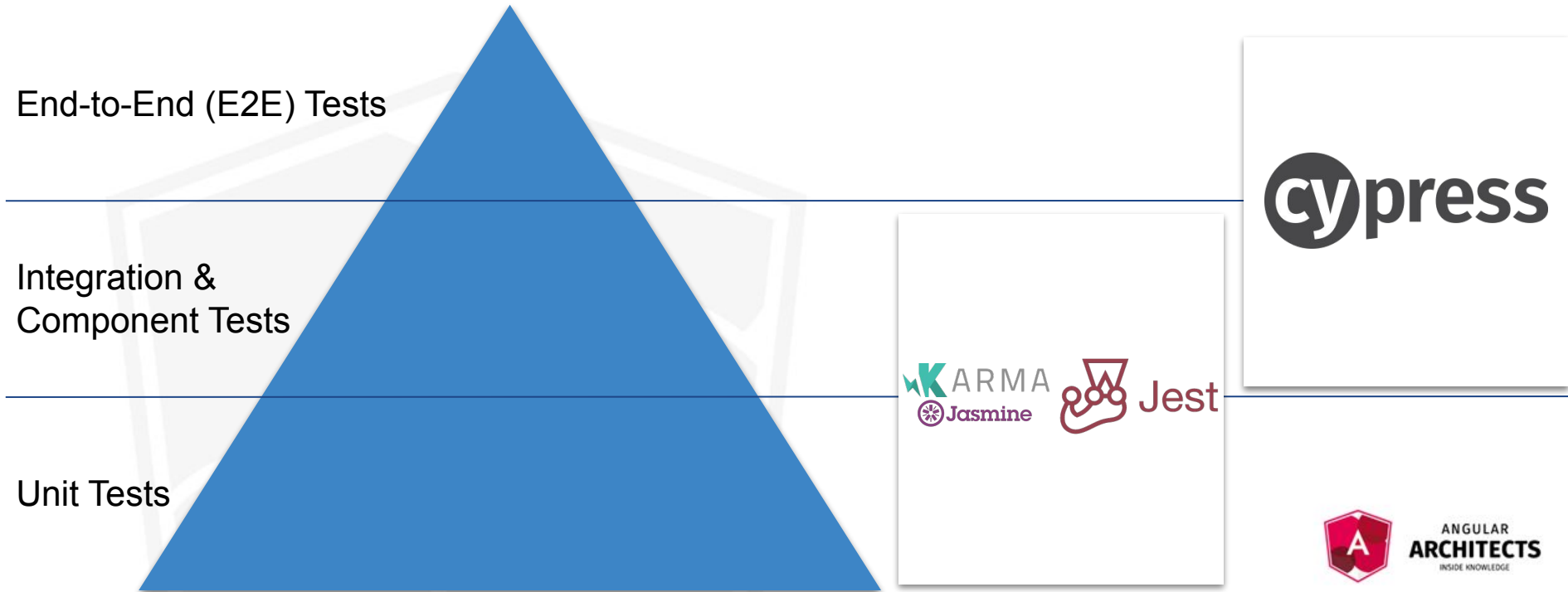


Integration &
Component Tests



Unit Tests

Testing pyramide with Cypress Component Test Runner



Classic Way 1/2

```
it('should find an address', fakeAsync(() => {  
  const fixture = TestBed.configureTestingModule({  
    declarations: [RequestInfoComponent],  
    imports: [  
      NoopAnimationsModule,  
      HttpClientTestingModule,  
      CommonModule,  
      ReactiveFormsModule,  
      MatButtonModule,  
      MatFormFieldModule,  
      MatInputModule,  
      MatIconModule  
    ]  
  }).createComponent(RequestInfoComponent);  
  fixture.detectChanges();  
  
  // ...  
})
```



Classic Way 2/2

```
it('should find an address', fakeAsync(() => {  
  // ...  
  
  const input = fixture.debugElement.query(By.css('[data-testid=address]'))  
    .nativeElement as HTMLInputElement;  
  const button = fixture.debugElement.query(By.css('[data-testid=btn-search]'))  
    .nativeElement as HTMLButtonElement;  
  
  input.value = 'Domgasse 5';  
  input.dispatchEvent(new CustomEvent('input'));  
  button.click();  
  
  TestBed.inject(HttpTestingController)  
    .expectOne((req) => !!req.url.match(/nominatim/))  
    .flush([true]);  
  tick(1000);  
  fixture.detectChanges();  
  
  const message = fixture.debugElement.query(By.css('[data-testid=lookup-result]'))  
    .nativeElement as HTMLParagraphElement;  
  
  expect(message.textContent).toBe('Brochure sent');  
}));
```



E2E

```
it('should test the request info', () => {  
  cy.visit('');  
  cy.testid('btn-holidays').click();  
  cy.get('app-holiday-card').first().find('a').click();  
  cy.testid('address').type('Domgasse 5');  
  cy.testid('btn-search').click();  
  cy.testid('lookup-result').should('have.text', 'Brochure sent');  
});
```



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Current situation

- Component/Integration tests are too hard
 - Managing asynchrony
 - DOM interaction
 - Setup TestBed
- Mitigation via libraries like testing-library, etc. possible
- Developers favour E2E ⇒ absence of Component tests



Integration/Component vs. E2E Tests

Integration/Component Tests

- ✓ Test components in isolation
- ✓ Precision & Control
- ✓ Fast
- ✗ TestBed Setup
- ✗ Manage asynchrony
- ✗ Manage change detection
- ✗ "DOM interaction"

E2E Tests

- ✓ No asynchrony management
- ✓ No change detection management
- ✓ Developer experience
 - Browser feedback
 - Screenshots
 - Video recording
 - Tasks, network stubbing,...
- ✗ Infrastructure setup required
- ✗ Slow



Component/Integration in Cypress

- ✓ Test components in isolation
- ✓ Precision & Control
- ✓ No asynchrony management
- ✓ No change detection management
- ✓ Developer experience
- TestBed Setup
- Speed???



Setup

```
const setup = () => {  
  const lookuper = {  
    lookup(address: string) {  
      return scheduled([address === "Domgasse 5"], asyncScheduler);  
    },  
  };  
  mount(RequestInfoComponent, {  
    imports: [NoopAnimationsModule],  
    providers: [{ provide: AddressLookuper, useValue: lookuper }],  
  });  
};
```



Tests

```
it(`should show ${message} for ${address}`, () => {  
  setup();  
  cy.get("[data-testid=ri-address]").type(address);  
  cy.get("[data-testid=ri-search]").click();  
  cy.get("[data-testid=ri-message]").should("have.text", message);  
});
```



Limitations

- Only direct dependencies of a component can be mocked
- `cy.clock/tick` is not working
- `inject()` & `private`
- Missing mocking libraries
 - sinon instead of jest
- Technology in alpha stadium



Miscellaneous



HttpTest

```
it("should use Angular's http mock", () => {  
  TestBed.configureTestingModule({  
    declarations: [RequestInfoComponent],  
    imports: [ReactiveFormsModule, HttpClientTestingModule],  
  });  
  const fixture = TestBed.createComponent(RequestInfoComponent);  
  // queries for DOM Elements button and messageBox  
  const httpController = TestBed.inject(HttpTestingController);  
  
  button.click()  
  
  const request = httpController.expectOne((req) => !!req.url.match(/nominatim/));  
  request.flush([ { street: "Domgasse", streetNumber: 5 } ]);  
  expect(lookupResult.textContent).toBe("Address found");  
});
```

Instead of HttpClientModule

Runs AFTER http request



Routing Tests 1/2

- RouterTestingModule provides routing functionality for tests
- Location can verify the expected url
- RoutingConfiguration is required
- Documentation not really great



Routing Tests 2/2

- @ngworker/spectacular
- Dedicated library for routing-based tests



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Lab Time

