ANGULAR ARCHITECTS
INSIDE KNOWLEDGE
SOFTWARE ARCHITECT

# Testing

Alex Thalhammer

# Outline

- Motivation
- Unit Tests & Component Tests incl. demos
- End-to-End Tests with Cypress incl. demo &
- Labs

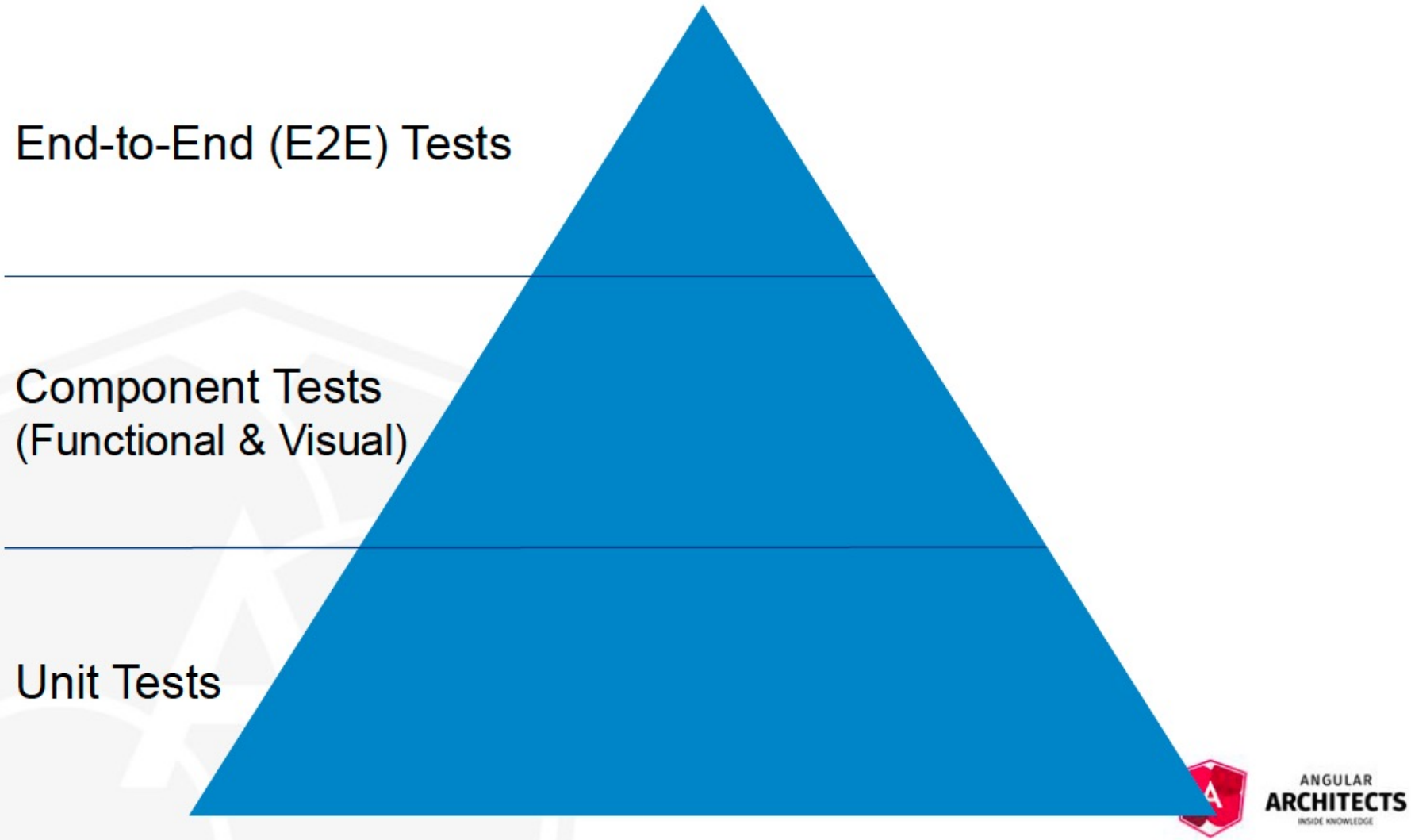


ANGULAR ARCHITECTS INSIDE KNOWLEDGE
SOFTWARE ARCHITECT

# Motivation Testing

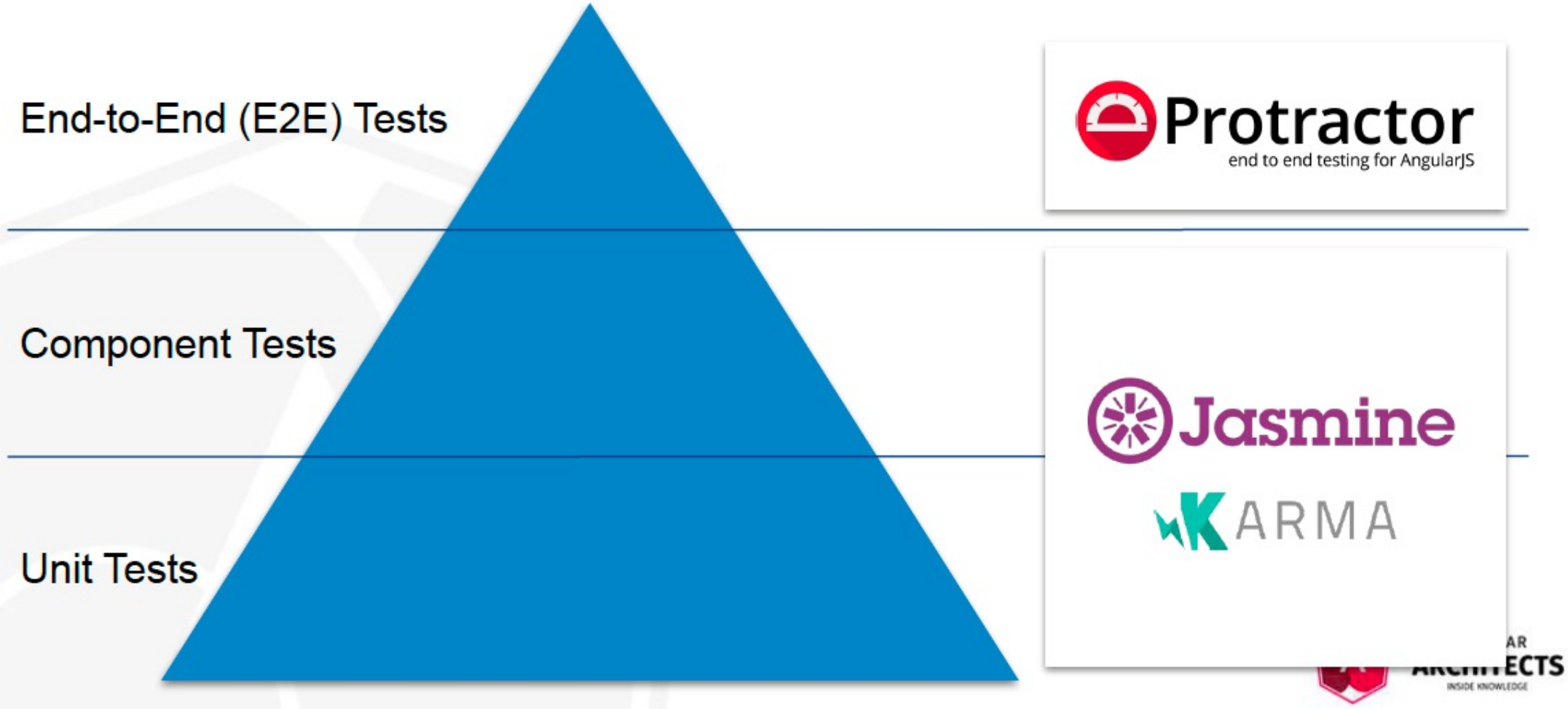
- Prevent bugs
- Enforce code quality
- Tests must be backed by Devs (require discipline)
- Writing Tests needs to be learned
- Tests must run fast, each has its own universe

# Testing pyramid

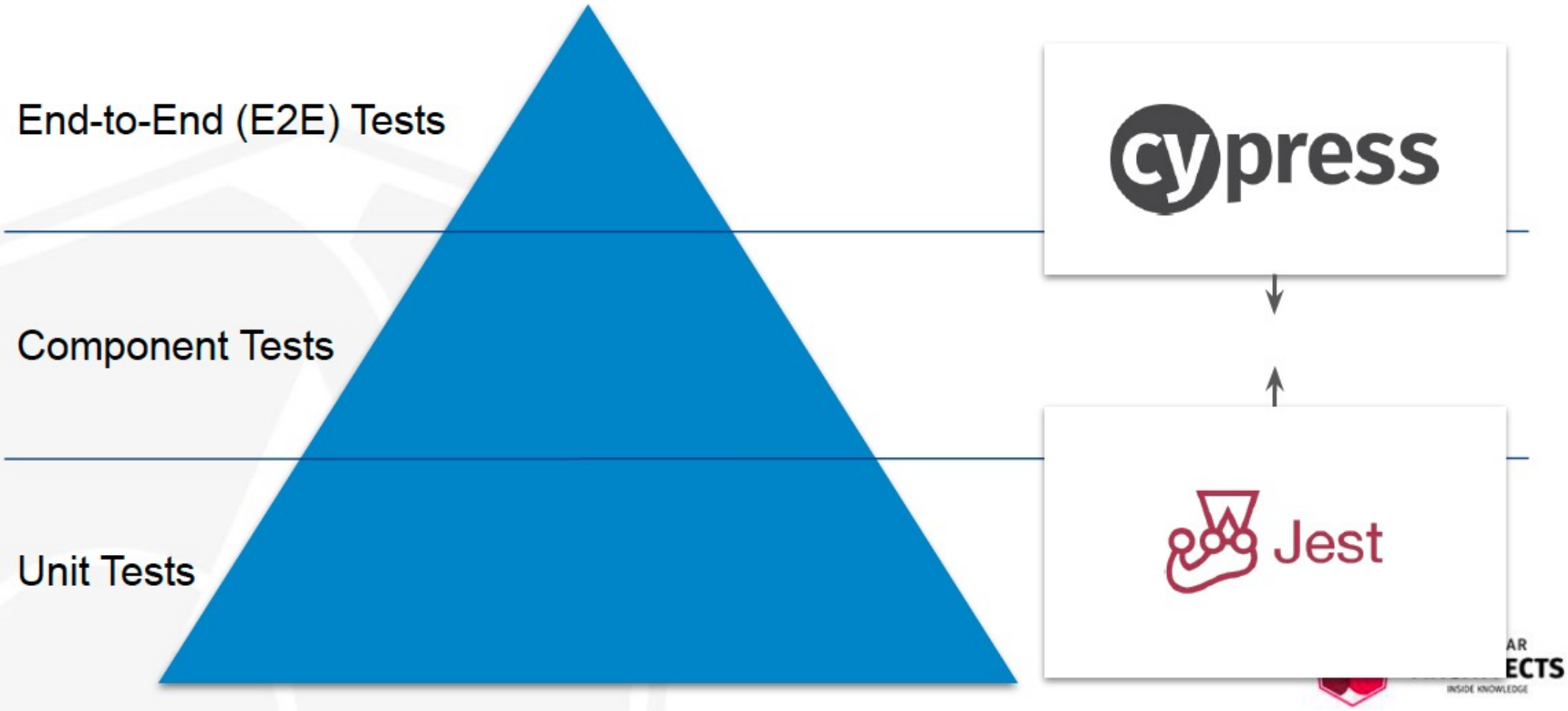



ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

# Official version (until NG 12)



# Our recommendation



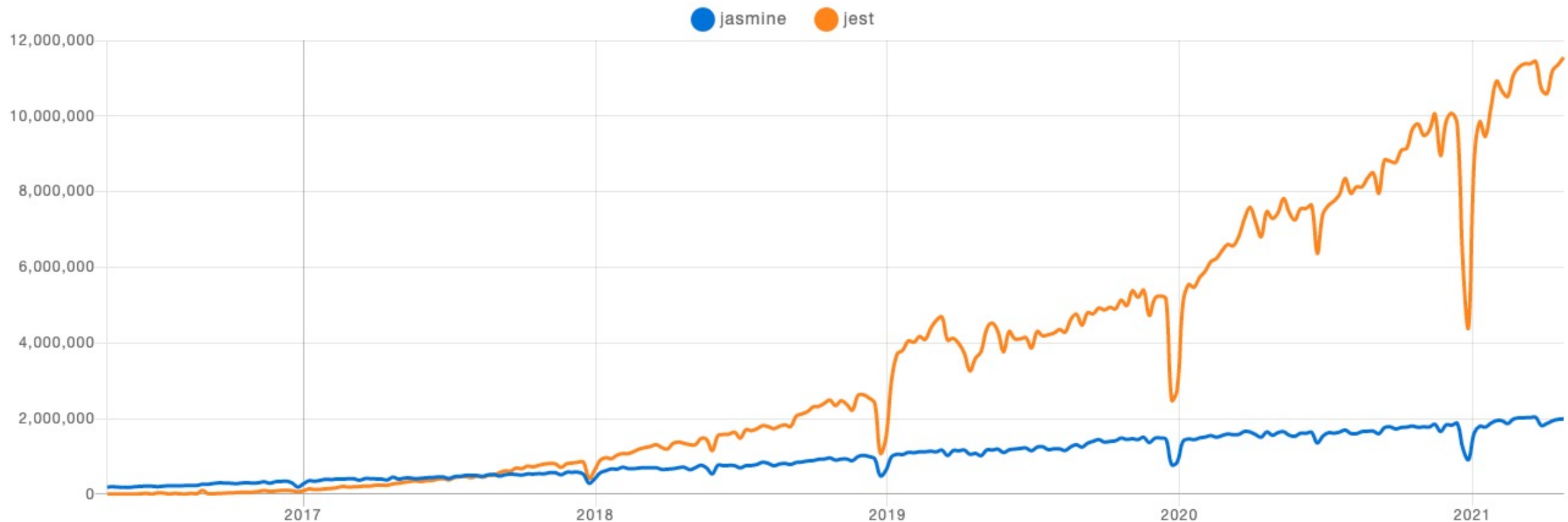
npm trends

# jasmine vs jest

Enter an npm package...

jasmine × jest × + mocha + qunit + ava + chai + expect

**Downloads** in past 5 Years



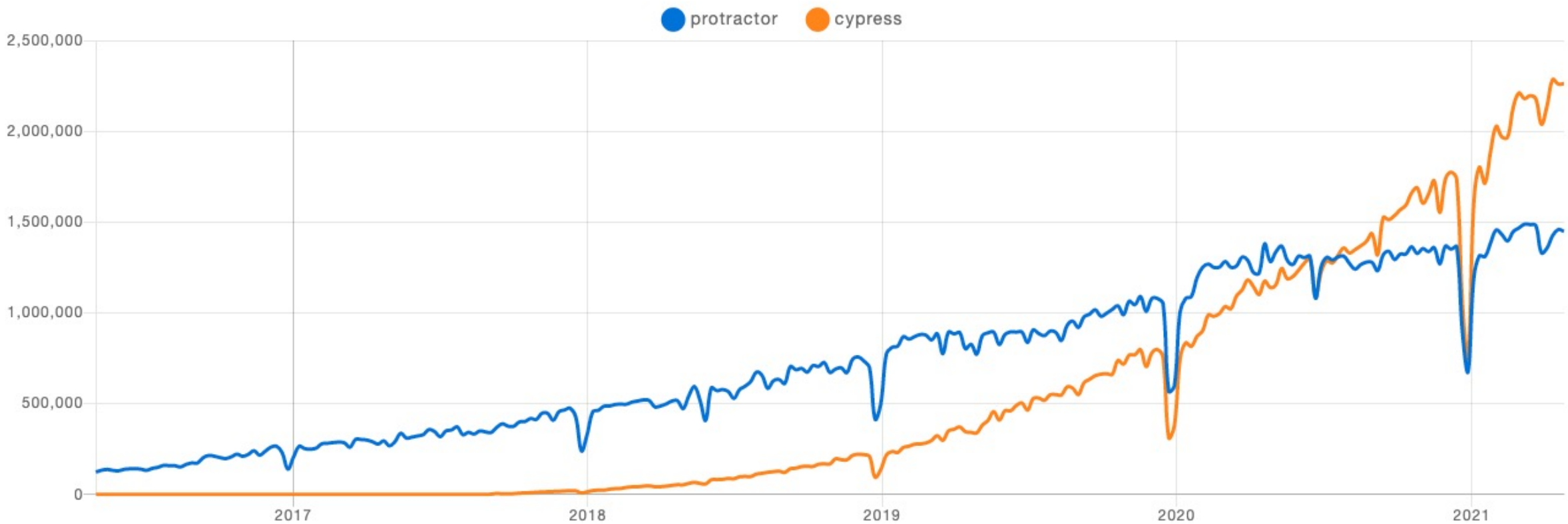
npm trends

# protractor vs cypress

Enter an npm package...

protractor × cypress × + nightwatch + webdriverio + testcafe + puppeteer

## Downloads in past 5 Years



# Unit Tests

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

# Testdriven development process (ideal world)

- Start with a Test
- Define how you would like to use the functionality
- Make sure it fails
- Implement it
- For next use case, define test

# Setup

- Angular CLI
  - ng add @briebug/jest-schematic
  - Remove all karma, jasmine, protractor deps
  - Make sure tsconfig is using jest types

- NX
  - Support out-of-the-box

# Motivation

- Superior Code Quality
- Documentation
- Find bugs quickly
- No issues with code coverage

# Running Tests

- Running all tests
  - jest or ng test
- Running specific ones
  - jest -t [namePattern]
- Running interactively (Developer Mode)
  - jest --watch

# A basic test

```
describe('Initial Tests', () => {
        it('should work', () => {
                expect(true).toBe(true);
        });
});
```

# Basic Expects

- expect(true).not.toBe(false);

- expect(true).toBeTruthy();
- expect({}).toBeTruthy();
- expect('').toBeFalsy();

- expect('').toBeDefined();
- expect(null).toBeNull();
- expect(null).toBeDefined();

# Data-Type Expects

- string & number
  - expect('hallo').toMatch(/l/);
  - expect(5).toBeGreaterThan(2);
  - expect(0.2 + 0.1).toBeCloseTo(0.3);

- arrays
  - expect([]).toHaveLength(0);
  - expect([1, 2, 3]).toContain(1);

- types
  - expect(new Date()).toBeInstanceOf(Date);
  - expect(new A()).toBeInstanceOf(A);
  - expect(() => true).toBeInstanceOf(Function);

# Object Expects

```
const address = {
        street: 'Domgasse’,
        streetNumber: '5’,
        zip: '1010’,
        city: 'Vienna’
};
const clone = { ...address };
```

- expect(address).toBe(clone); // fails
- expect(address).toEqual(clone); // succeeds
- expect(address).toMatchObject({ street: 'Domgasse', city: 'Vienna' }); // succeeds
- expect(address).toMatchObject({ city: expect.stringMatching(/Vienna|Wien/) }); // succeeds

# Expect Exceptions

```
const fn = () => {
        throw new Error('nothing works');
};
```

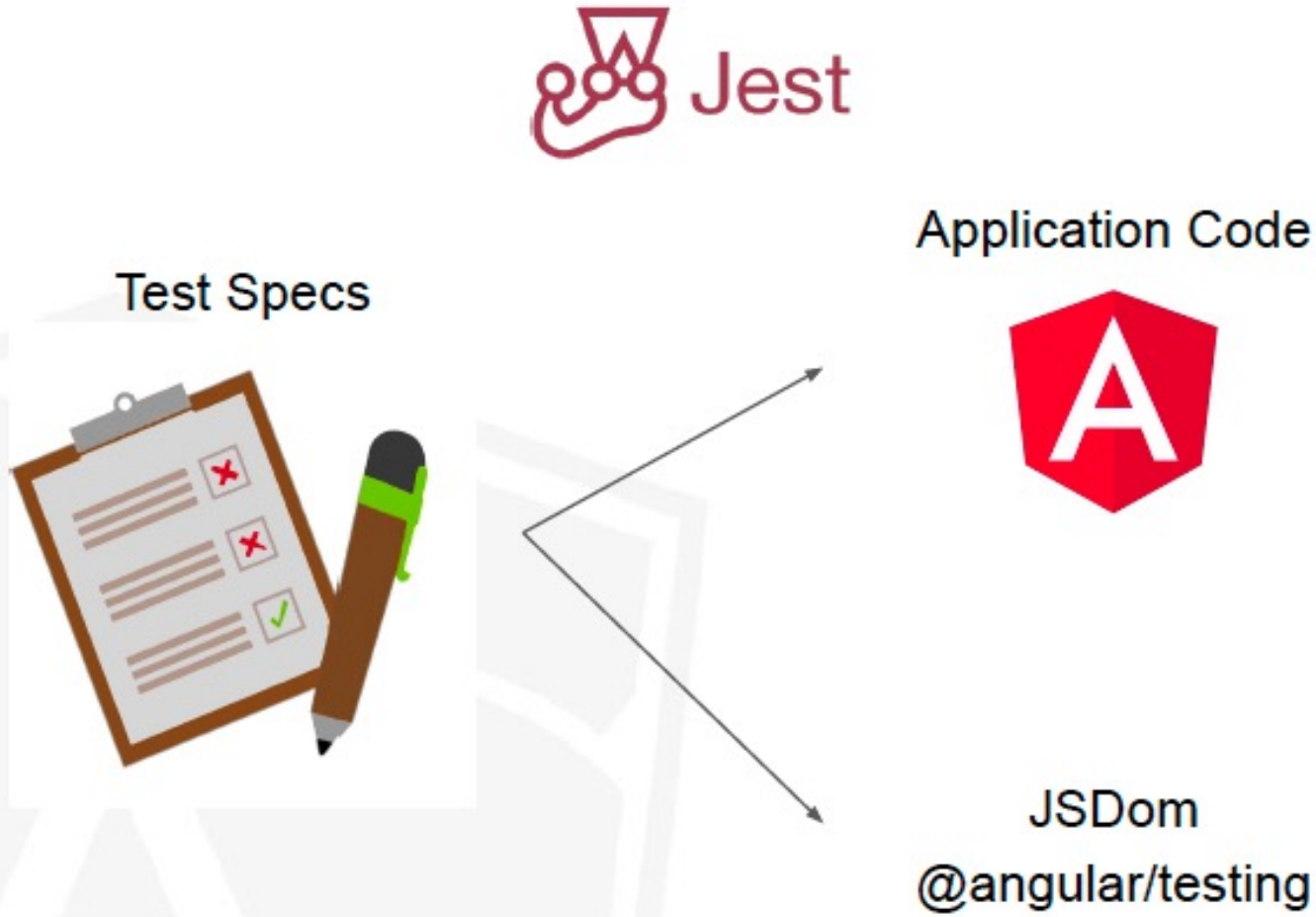
- expect(fn).toThrowError();
- expect(fn).toThrowError('nothing works');

# Unit tests with Jest – Demo

# Component Tests

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT




ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

# Component Tests powered by Angular

- TestBed
  - Configures & initializes environment for unit testing
  - Provides methods for creating components and services in unit tests.

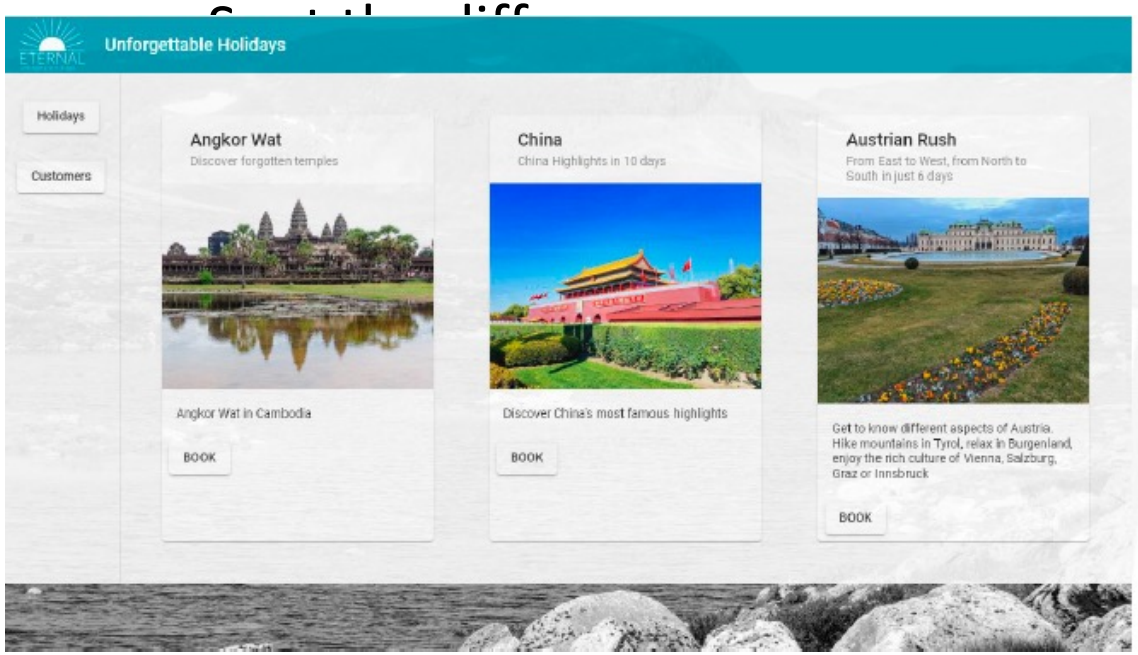
- TestModule

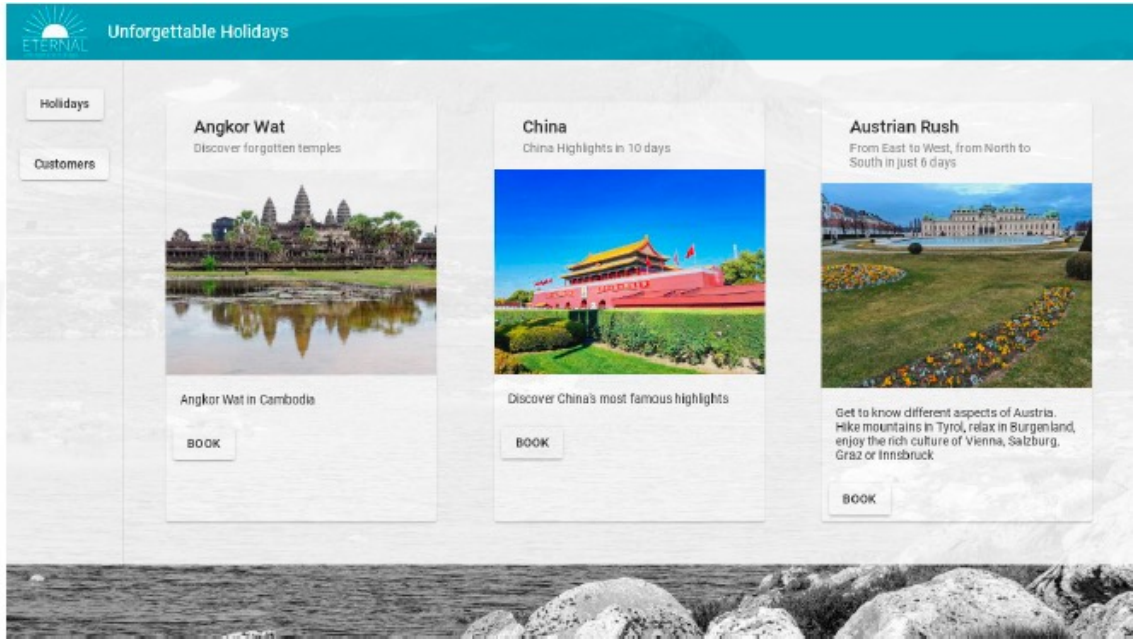
```
const fixture = TestBed.configureTestingModule({
        declarations: [AddressComponent],
        imports: [ReactiveFormsModule],
        providers: [{ provide: AddressLookuper, useValue: null }]
}).createComponent(AddressComponent);

const component = fixture.componentInstance;
```

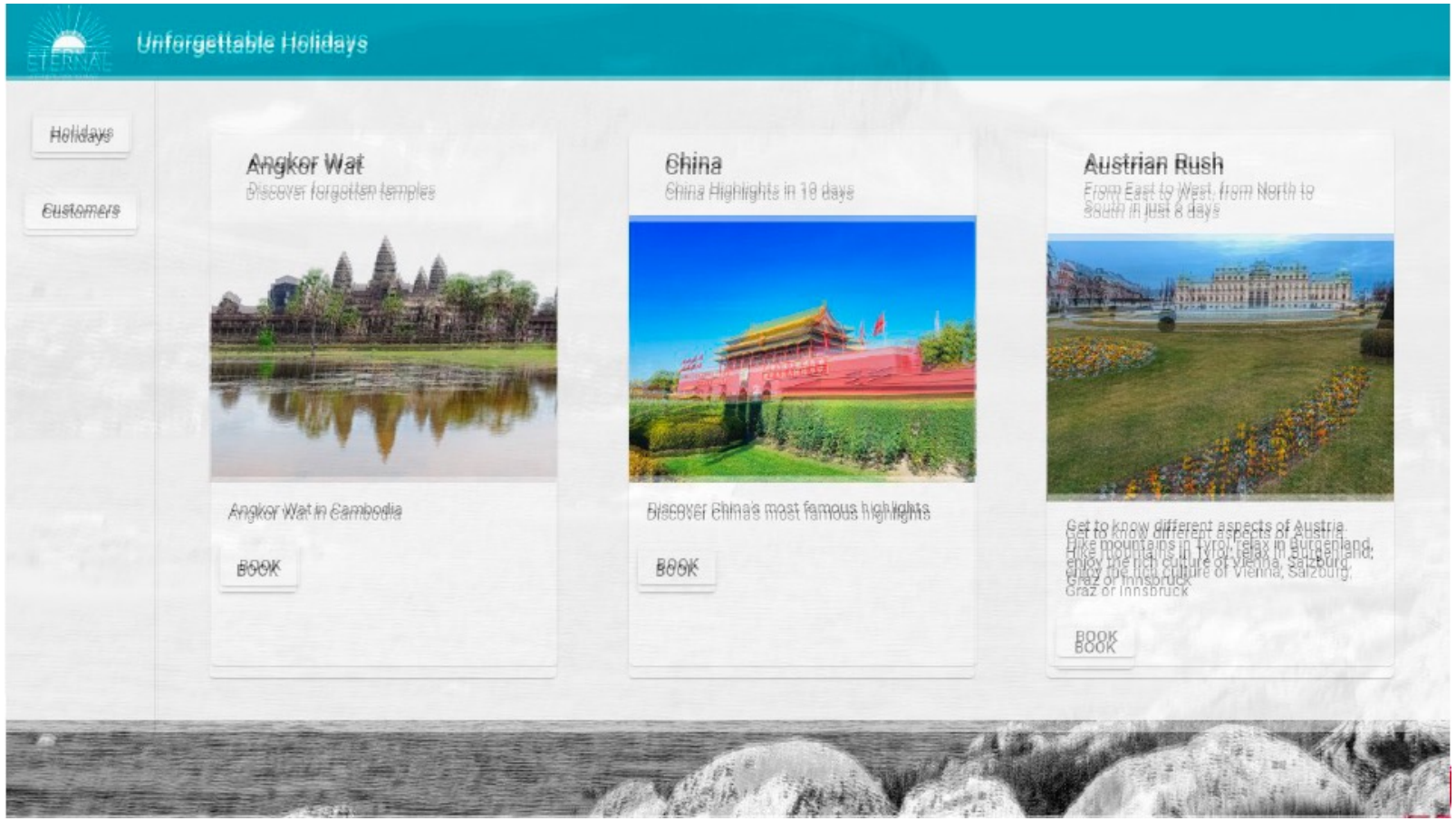
# Component Tests - Puppetteer I

- Puppetteer
  - Headless Browser

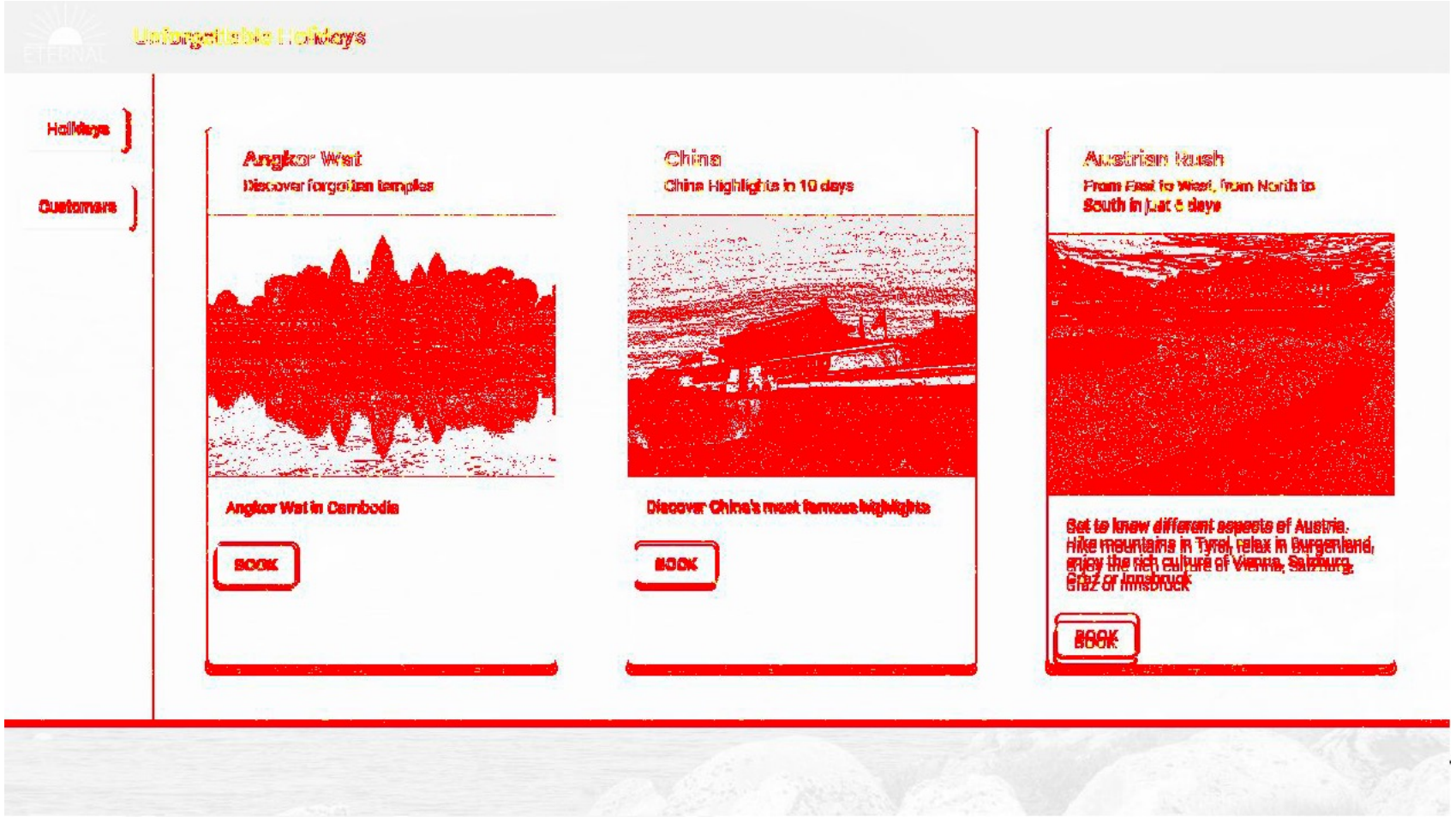


# Component Tests - Puppetteer II



# Component Tests - Puppetteer III



# Component Tests – Get Started

- yarn add puppeteer
- Decoupled from business logic
- No Dependency Injection
- If possible only primitive types as @Input

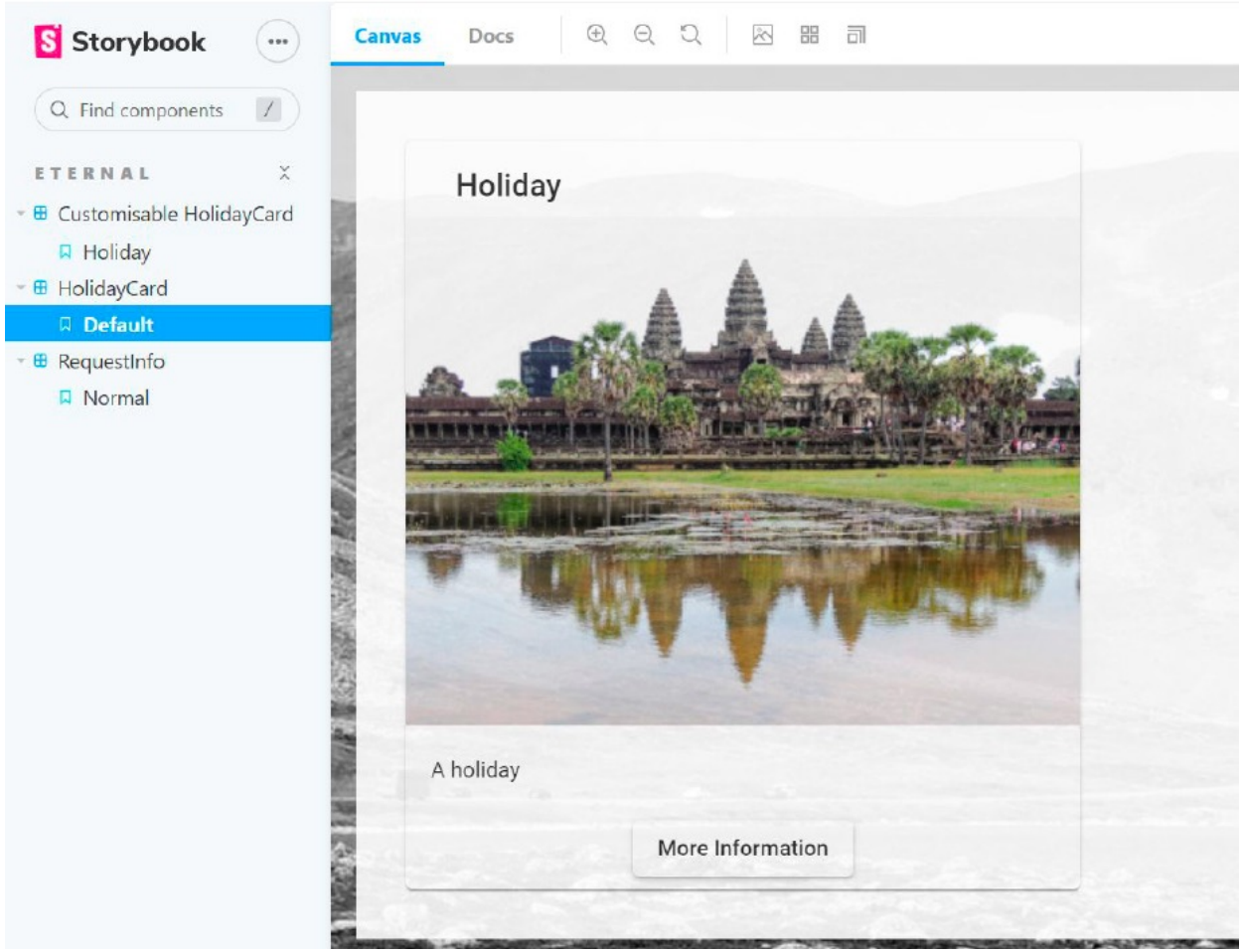
# Component Tests – Storybook I

- Allows to isolate Components
- Not Angular Specific
- Configure a Component for various states
- Can also used for visual widget library (not just testing)

# Component Tests – Storybook II

- Easy to setup in Storybook
- Decoupled from business logic
- No Dependency Injection
- If possible only primitive types as @Input

# Component Tests – Storybook III



# Storybook – Get Started

- npx sb init
- will auto generate examples
- yarn storybook

# Storybook – Conclusion

- most popular tool for UI component development & documentation
- used by GitHub, Airbnb, and Stripe
- but it has issues with not using default webpack

# Storybook – Demo

# That’s it for unit & component testing!

- Questions so far?