



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Testing Angular Jest & Cypress

Alex Thalhammer

Outline

- Motivation
- Unit Tests & Component Tests
 - Jest
 - Puppeteer
 - Story Book
- End-to-End Tests with Cypress



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

Unit tests vs. E2E tests

- Unit tests
 - Isolated component tests
 - Should be the majority of the tests
 - Quickly executable
- E2E tests
 - Cross-component tests
 - Simulation of user inputs



Testing pyramid

End-to-End (E2E) Tests

Component Tests
(Functional & Visual)

Unit Tests



Official version (until NG 12)

End-to-End (E2E) Tests



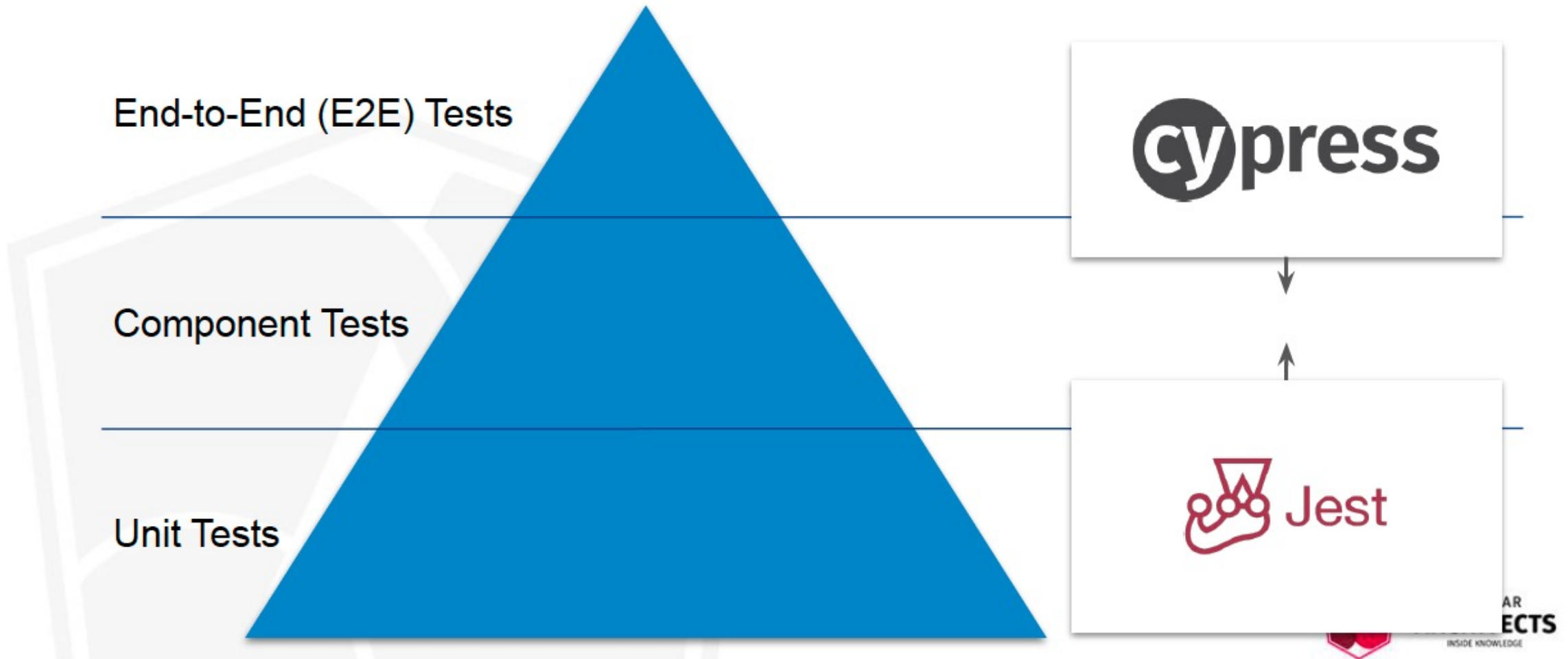
Component Tests



Unit Tests



Our recommendation



jasmine vs jest vs mocha

Enter an npm package...

jasmine x

jest x

mocha x

+ qunit

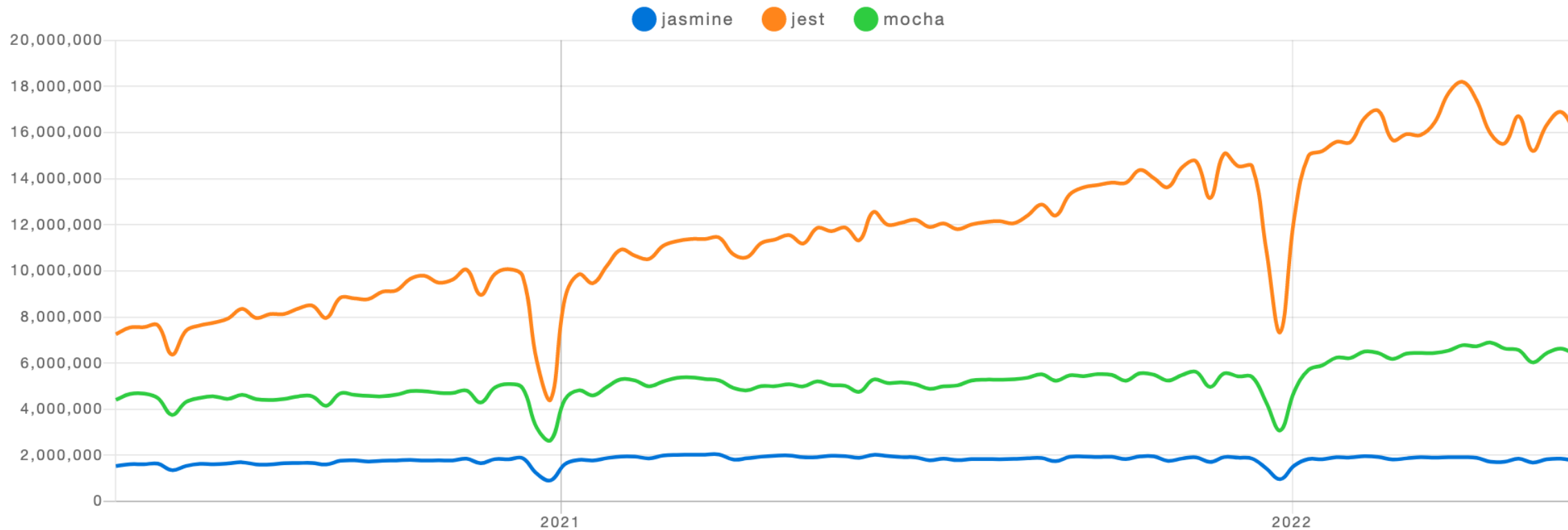
+ ava

+ chai

+ expect

+ should

Downloads in past 2 Years ▾



cypress vs protractor

Enter an npm package...

cypress x

protractor x

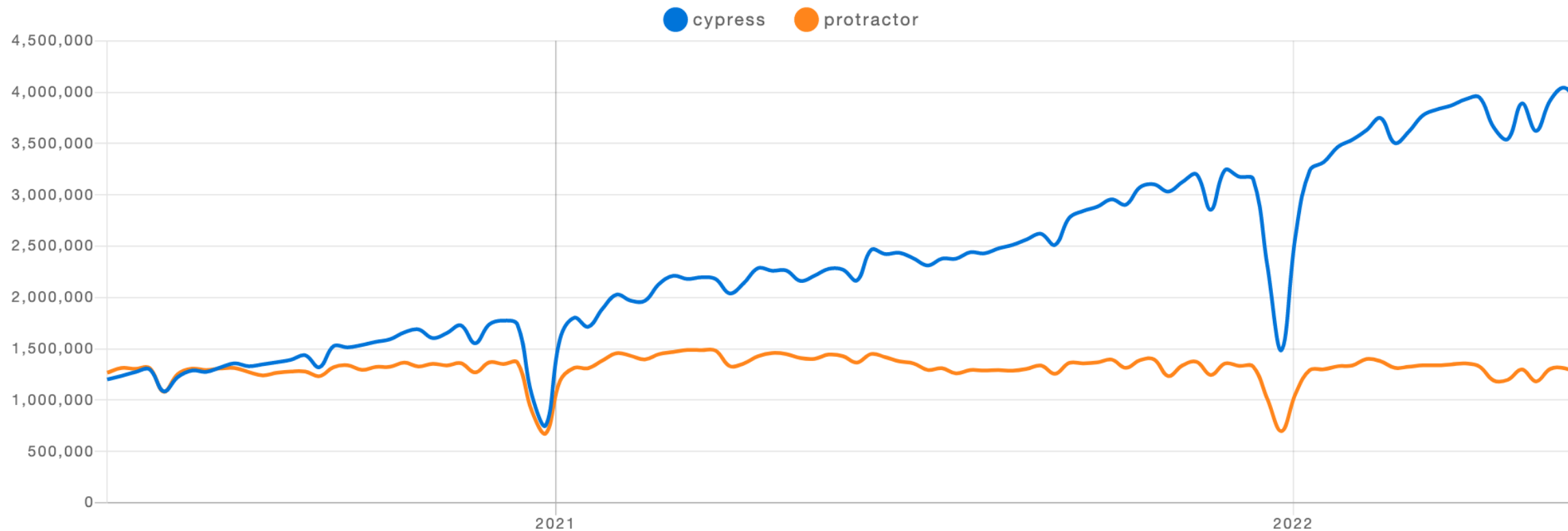
+ nightwatch

+ webdriverio

+ testcafe

+ puppeteer

Downloads in past 2 Years v



Unit Tests



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
- nrwl 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');`



Component Tests





Test Specs



Application Code



JSDom

@angular/testing



Component Tests powered by Angular

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

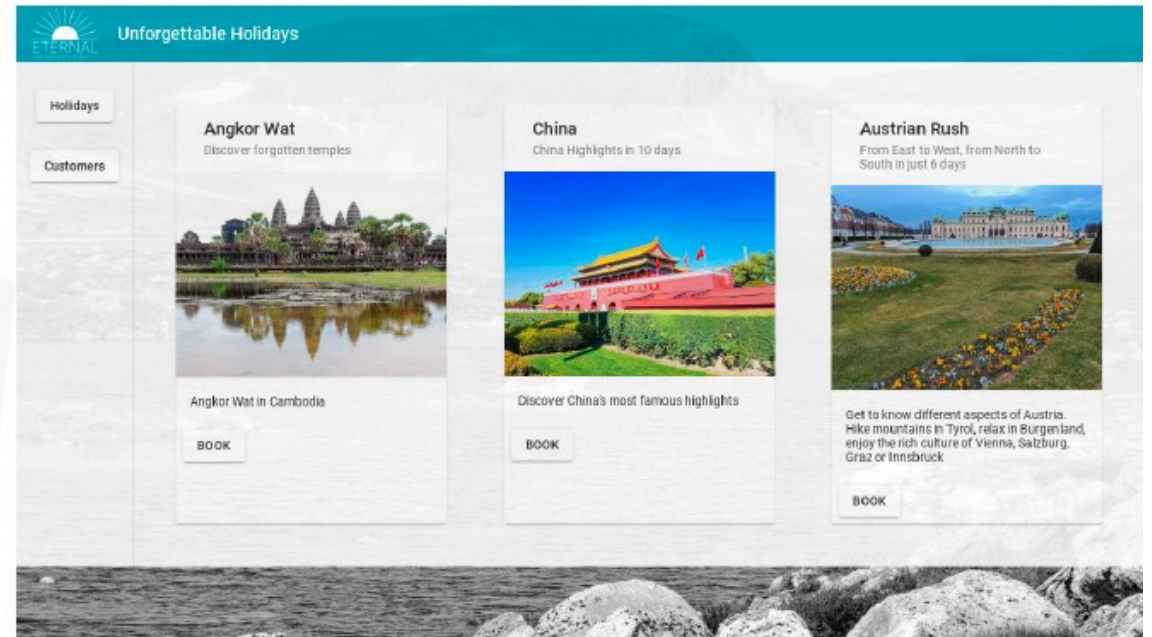
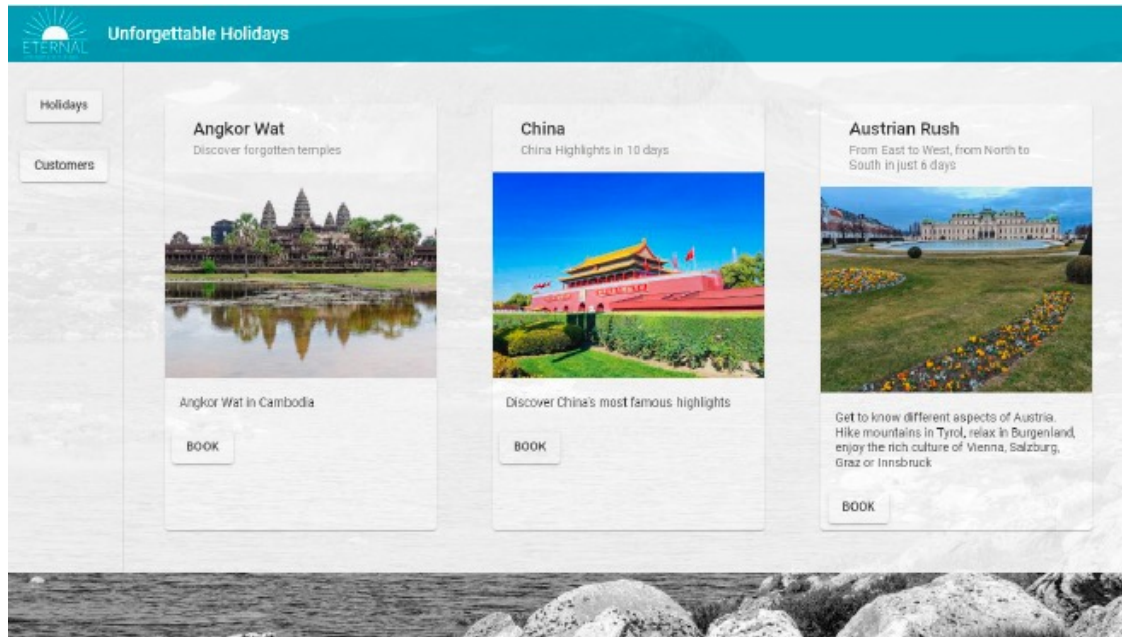
- TestModule

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

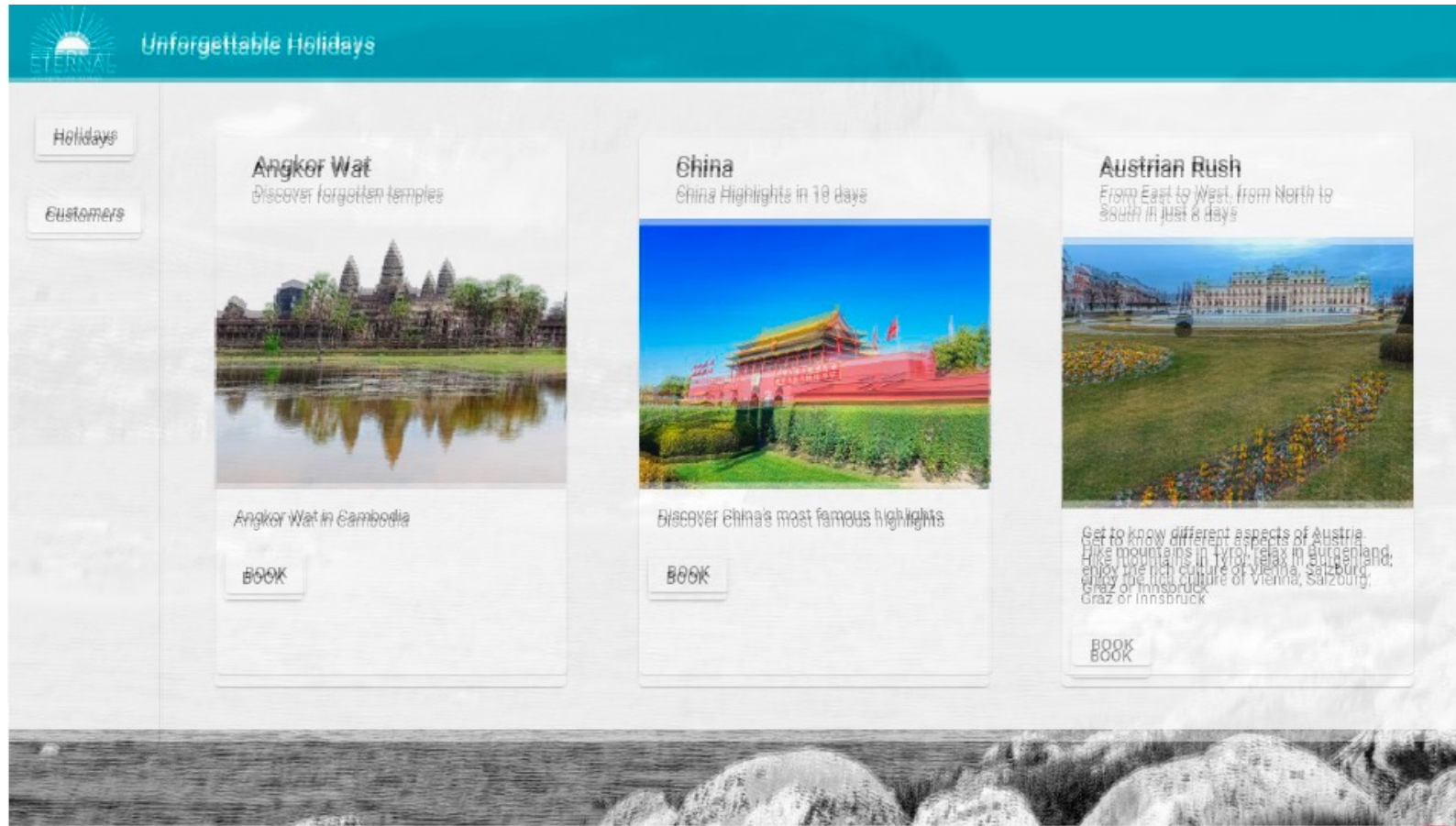
```
const component = fixture.componentInstance;
```

Component Tests - Puppeteer I

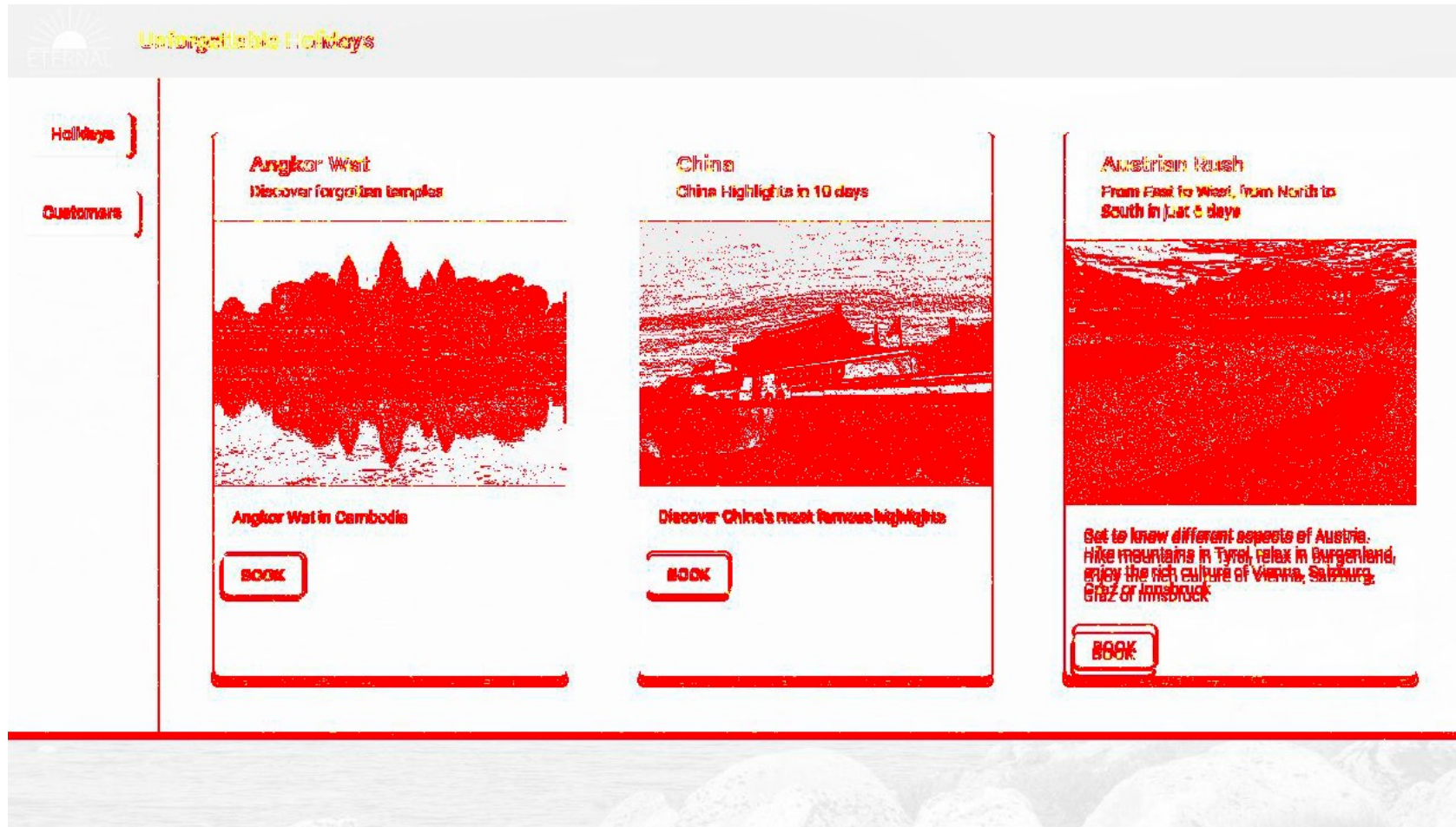
- Puppeteer
 - Headless Browser
 - Spot the difference



Component Tests - Puppeteer II



Component Tests - Puppeteer III



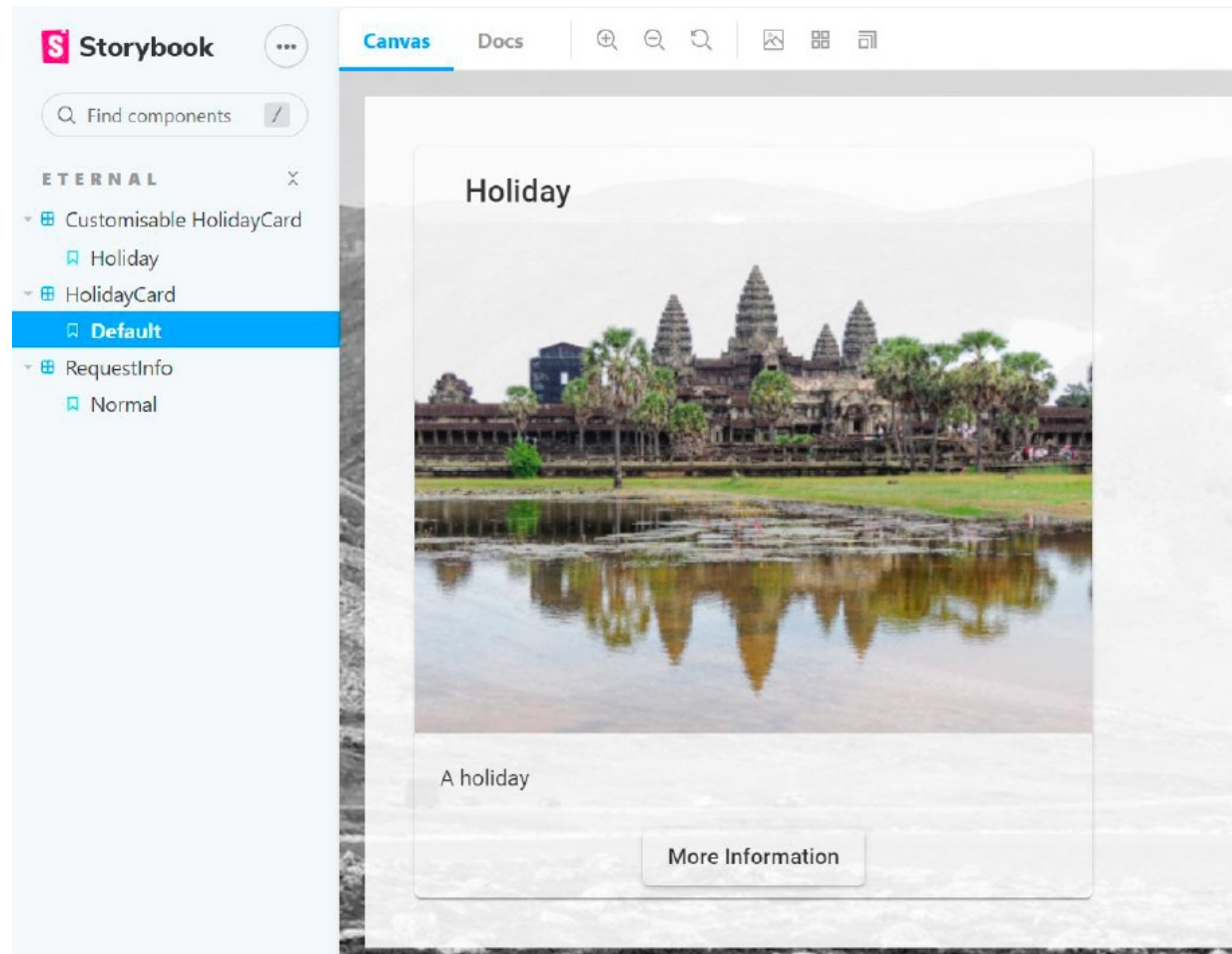
Component Tests – Storybook I

- 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 / npm run 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

Intro



- Cypress (in most cases) perfect successor
- Migration is a rewrite



Motivation

- Great Developer Experience
- Good Documentation
- Easy Setup
- Internal "IDE"
- CI Features like Videorecording and Screenshots

Cypress Setup

- Cypress has no Angular integration (yet?)
- yarn add / npm i -D cypress
- add tsconfig.json to newly cypress directory
- cypress open (open the Cypress Dashboard) or
- cypress run (just runs the tests)



Basic commands

- `cy.visit(url: string)`
 - Can only be run at the beginning (of a test)
 - Domain can't be changed
- `cy.get(selector)`
 - Uses jQuery style selectors
 - Runs asynchronously
 - Chainable
 - contains
 - click
 - type
 - ...



Assertions Implicit

- Behaves like a normal command
- Does waiting as well (asynchronicity)
- Good for single assertions

```
cy

  .get('h1')

  .should(

    'have.text',

    ' Unforgettable Holidays '

  );
```



Assertions Explicit

- More verbose
- Good when more logic is involved

```
cy.get('h1').should(($h1) => {  
  
    expect($h1).to.have  
  
        .text(' Unforgettable Holidays ');  
  
});
```



Assertions are
not required!



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Cypress – Demo



Ready for the labs!

- Questions so far?