



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

# Unit & Component Tests

## Jest, Puppeteer & Storybook

Alex Thalhammer

# Outline

- Motivation
- Jest Unit Tests & Component Tests
- Jest Unit Test Lab
- Component Tests with Puppeteer
- Component Tests with Storybook



# Motivation Testing

- Enforce Code Quality / prevent bugs / find bugs quickly
- No issues with code coverage
- Documentation
- But
  - Writing Tests needs to be learned
  - Must be backed by Devs (require discipline)
  - 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 11)

End-to-End (E2E) Tests



Component Tests



Unit Tests

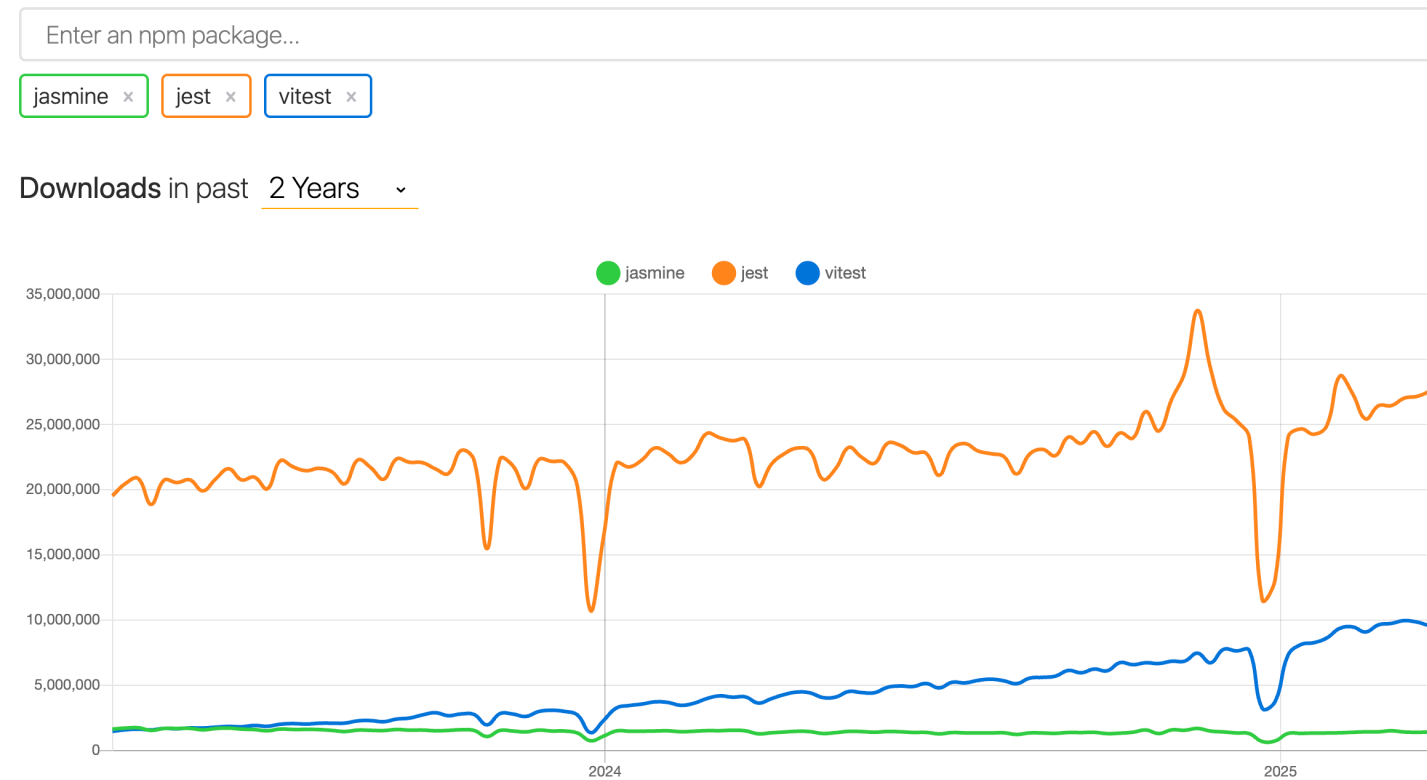


# Our recommendation



# NPM trends

jasmine vs jest vs vitest





# 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 @bribug/jest-schematic
  - Remove all karma, jasmine, protractor deps
  - Make sure tsconfig(s) & angular.json (project.json) are using jest
- NX
  - Support out-of-the-box

# Running Tests

- Running all tests
  - `jest` or `ng / nx 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');`



# DEMO



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**

# LAB



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**

# Component Tests



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**



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



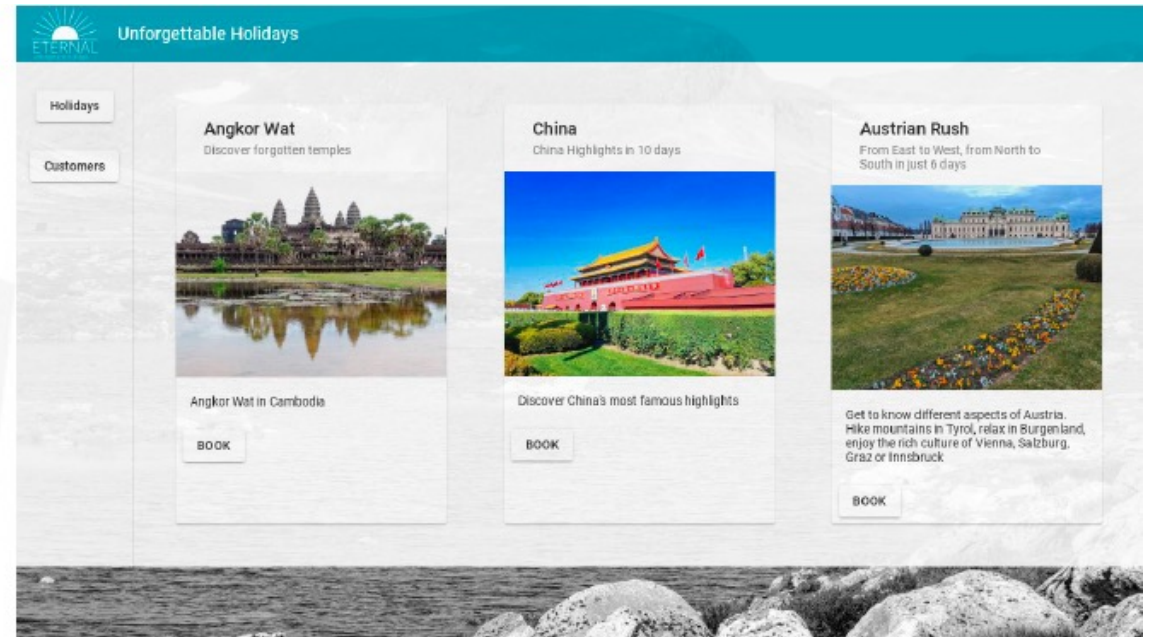
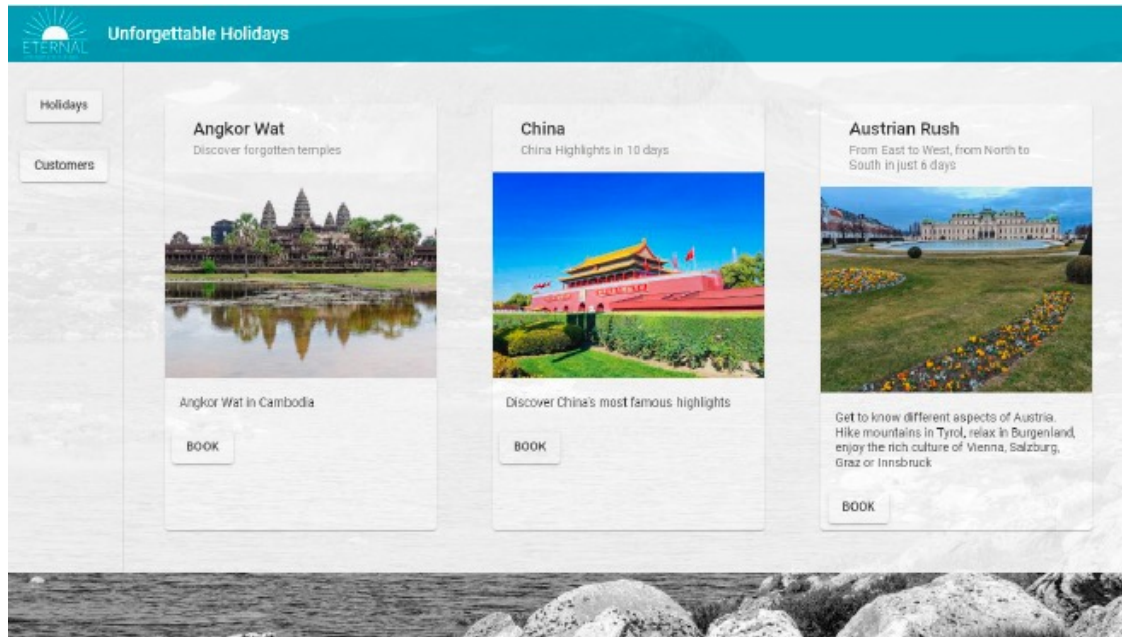
ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**

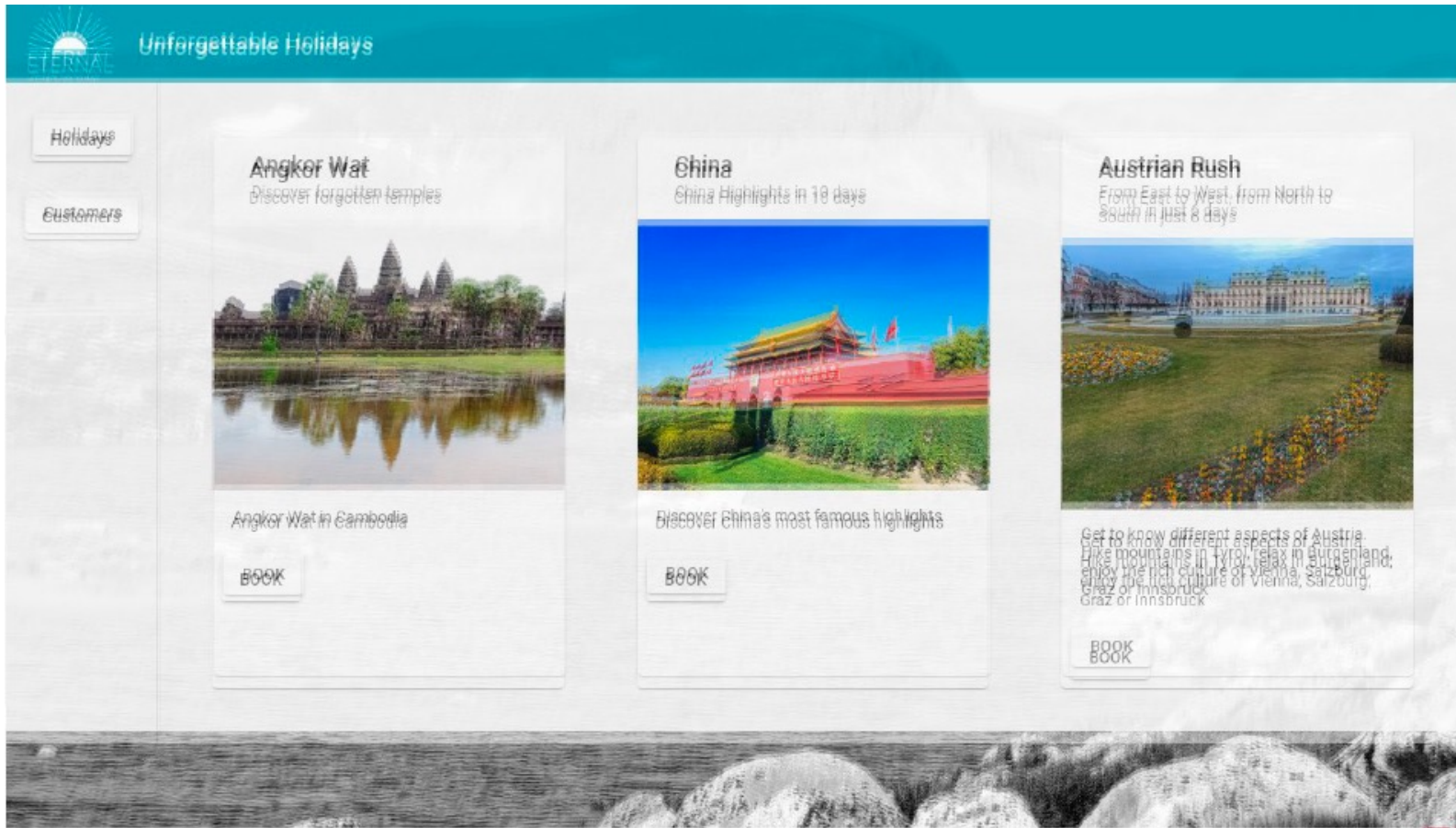
# Component Tests - Puppeteer I

- Puppeteer
  - Headless Browser
  - Spot the difference

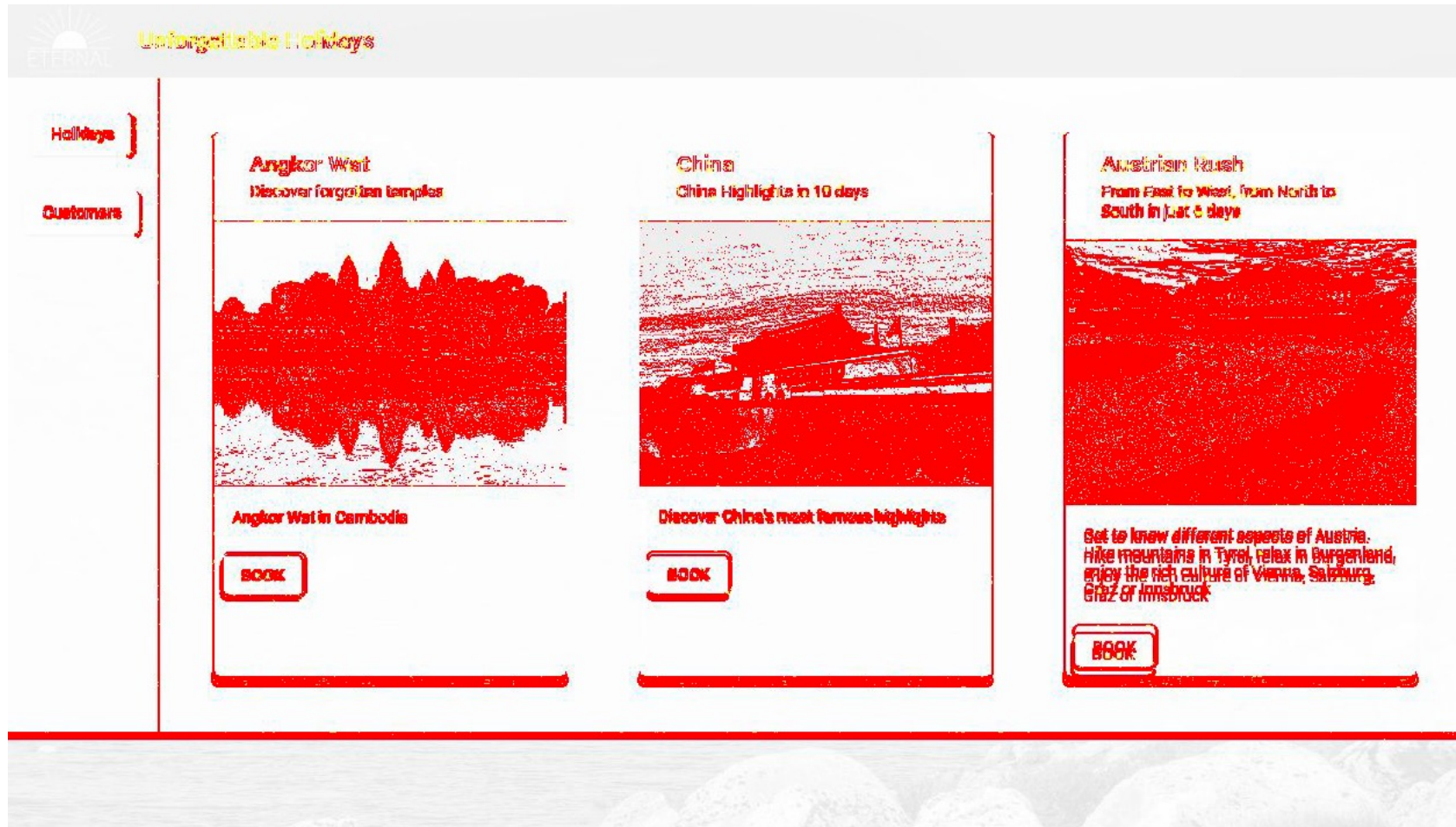




# Component Tests - Puppeteer II



# Component Tests - Puppeteer III



# Component Tests - Storybook



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE



SOFTWARE  
**ARCHITECT**

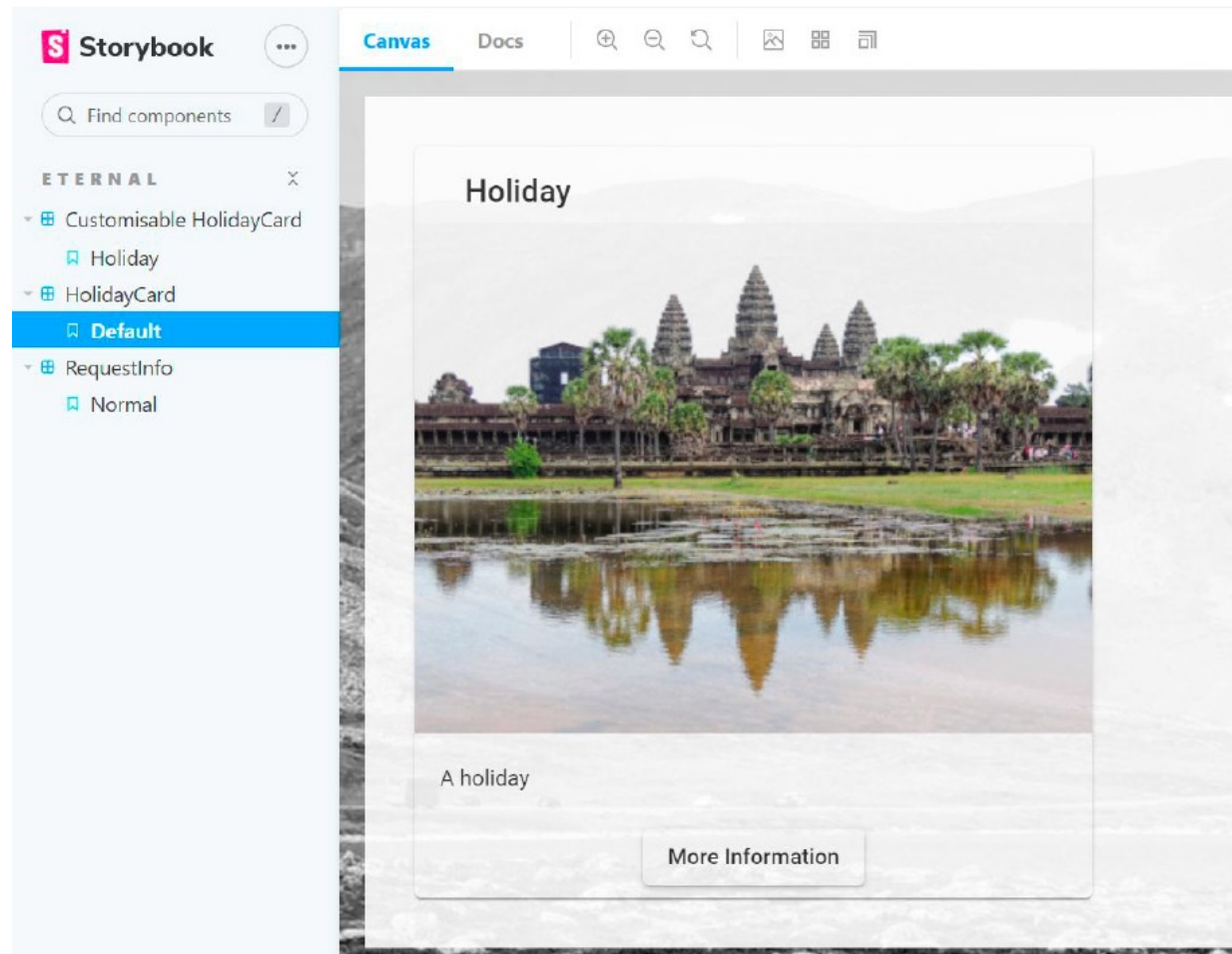
# Component Tests – Storybook I

- 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



# That's it for unit & component testing!

- What are your questions?

