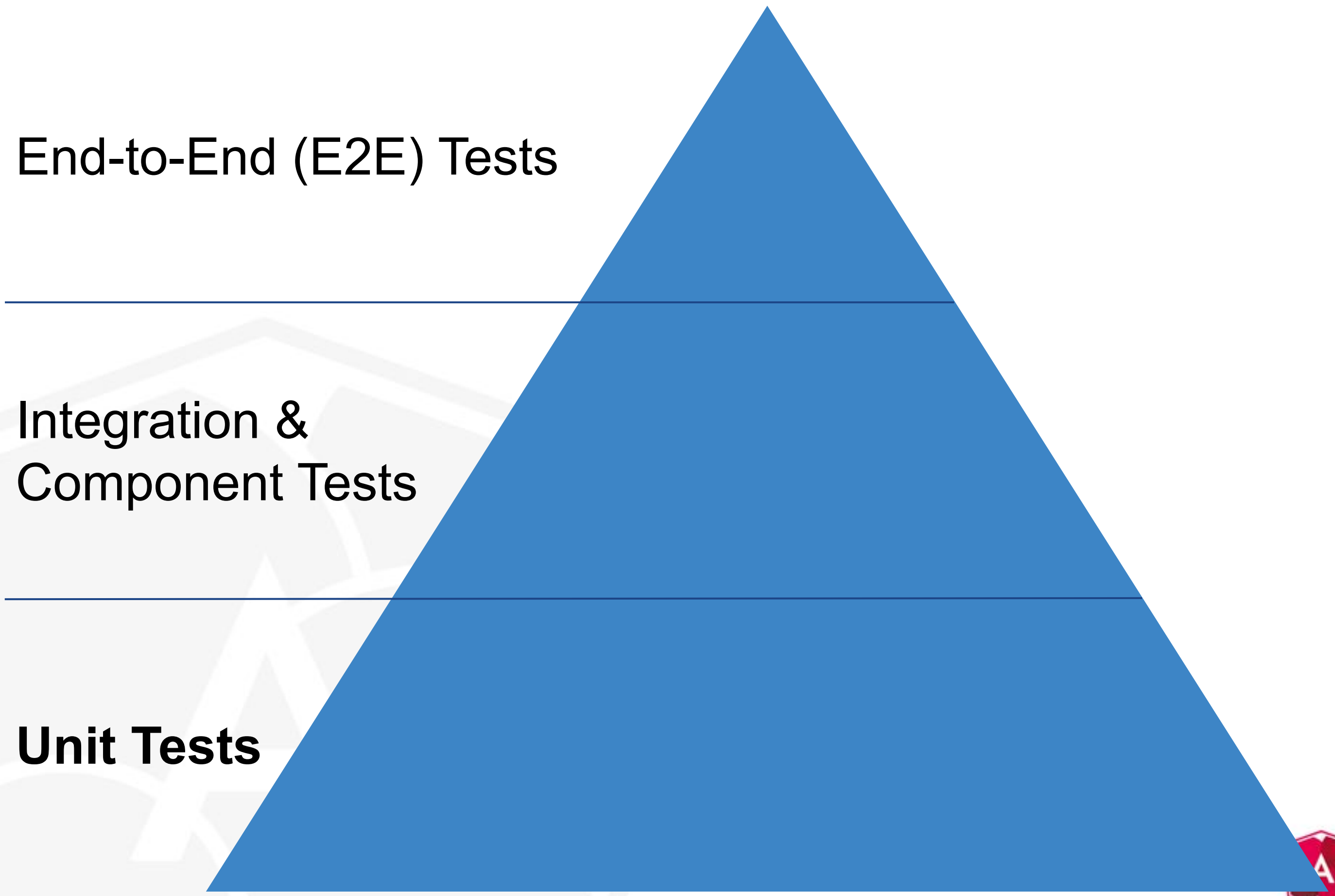
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Angular Testing

2 - Unit Tests Basics


End-to-End (E2E) Tests
Integration & Component Tests
Unit Tests

… the smallest piece of code that can be **logically isolated** in a system.

https://smartbear.com/learn/automated-testing/what-is-unit-testing/

Setup

- Angular CLI
  - `npx ng add @briebug/jest-schematic`
  - `Remove all karma, jasmine, protractor dependencies`
  - `Make sure tsconfig is using jest types`
- NX
  - Support Out-of-the-Box

Running Tests

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

Our First Test

```
describe('Initial Tests', () => {

  it('should work', () => {

    expect(true).toBe(true);

  });

});
```

Our First Test

```
describe('Initial Tests', () => {

  it('should work', () => {

    expect(true).toBe(true);

  });

});
```

Our First Test

```
describe('Initial Tests', () => {

  it('should work', () => {

    expect(true).toBe(false);

  });

});
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Demo

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);
class A {}
expect(new A()).toBeInstanceOf(A);
expect(() => true).toBeInstanceOf(Function);
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

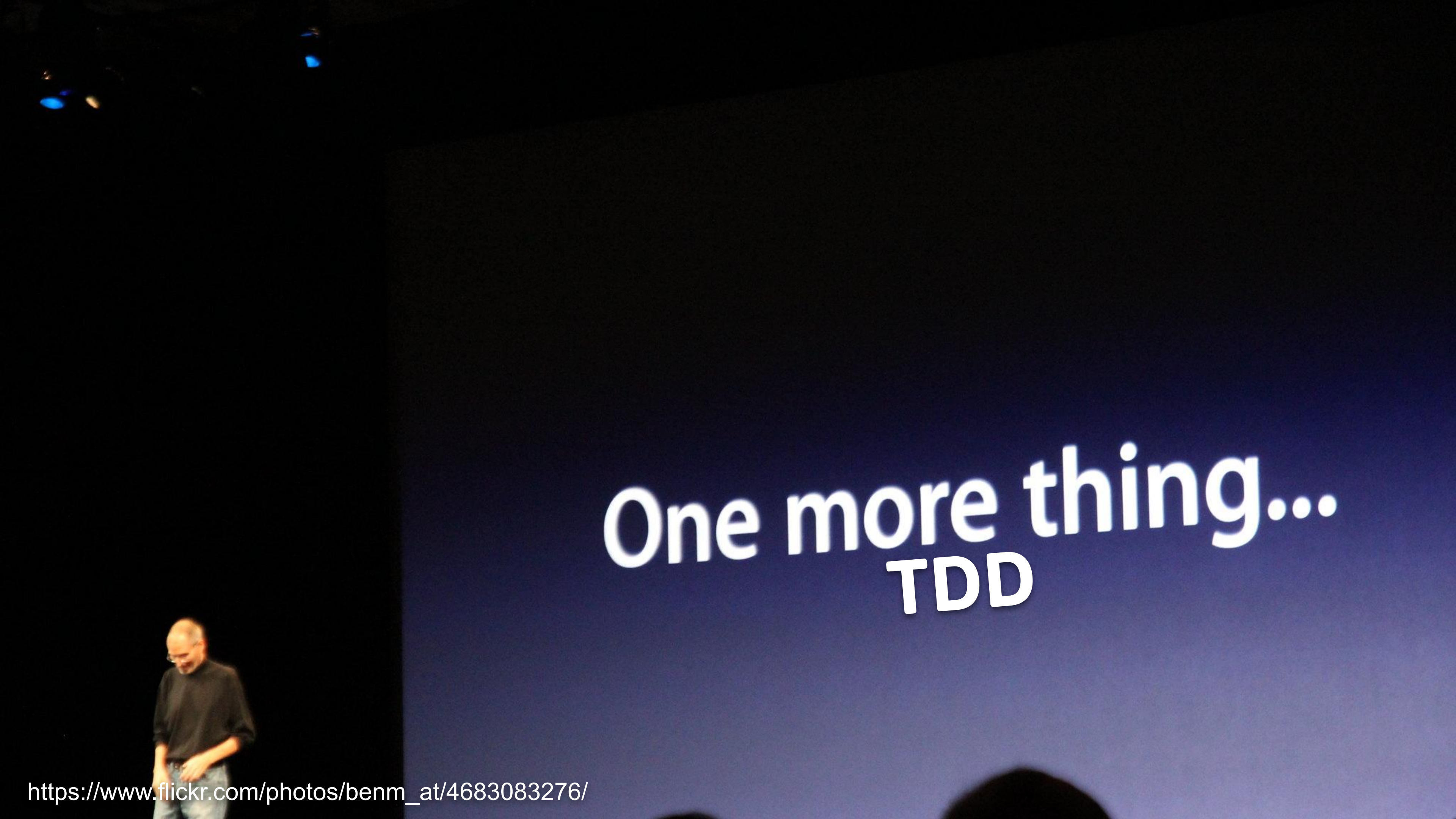
Object Expects

```
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
```

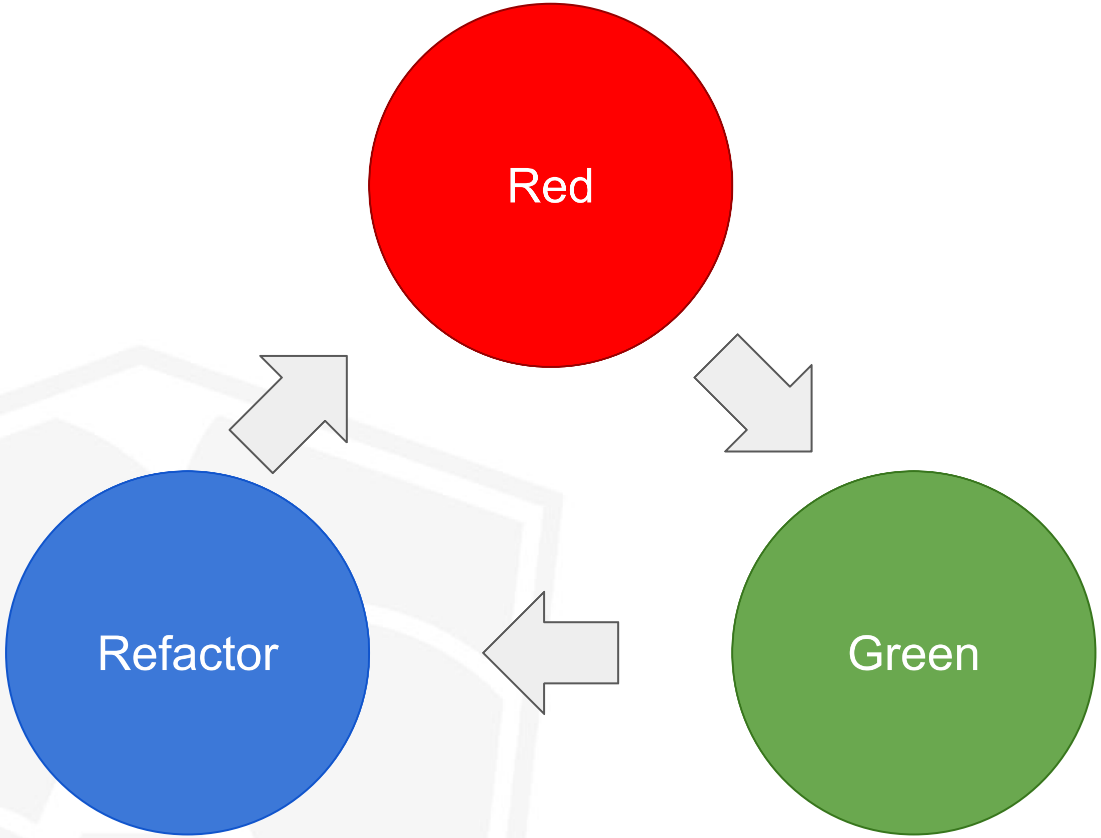
Expect Exceptions

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



TDD

https://www.flickr.com/photos/benm_at/4683083276/


Red
Green
Refactor

Process

1. Start with a Test
2. Define how you would like to use the functionality
3. Make sure it fails
4. Implement it
5. For next use case, define test

Each line of code must be **justified** by a test

Advantages

- Superior Code Quality
- Documentation
- No issues with code coverage
- Find bugs quickly
- Changes based on strong Footing

Lab Time