

INTRO TO TESTING

minimizing mistakes

QUESTIONS YOU MIGHT HAVE

- Why test?
- How to test?
- What to test?

WHY DO I NEED TO WRITE TESTS?

WHY

- Reliability
- Refactorability
- Documentation
- Accuracy
- Value in industry

HOW TO WRITE MY TESTS?

ANATOMY OF A SPEC

Describe blocks: contains

[sub-groups of] specs

1) descriptive labels of
entity to be tested

2) function to nest further
describes (sub-entities)
or its

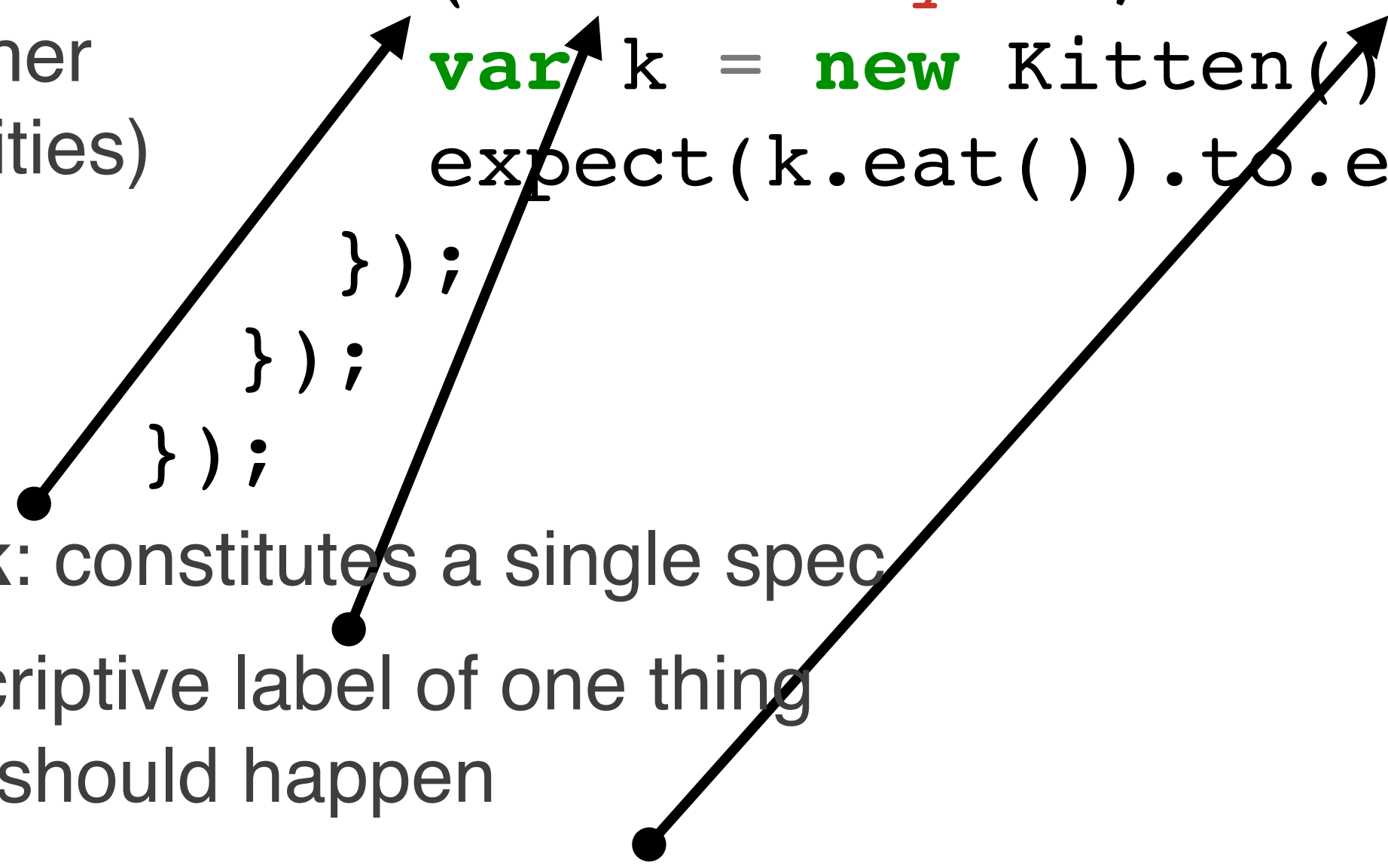
```
describe('Kittens', function() {  
  describe('eat', function() {  
    it('returns yum', function() {  
      var k = new Kitten();  
      expect(k.eat()).to.equal('yum');  
    });  
  });  
});
```

ANATOMY OF A SPEC

Describe blocks: contains
[sub-groups of] specs

- 1) descriptive labels of entity to be tested
- 2) function to nest further describes (sub-entities) or its

```
describe('Kittens', function() {  
  describe('eat', function() {  
    it('returns yum', function() {  
      var k = new Kitten();  
      expect(k.eat()).to.equal('yum');  
    });  
  });  
});
```



It block: constitutes a single spec

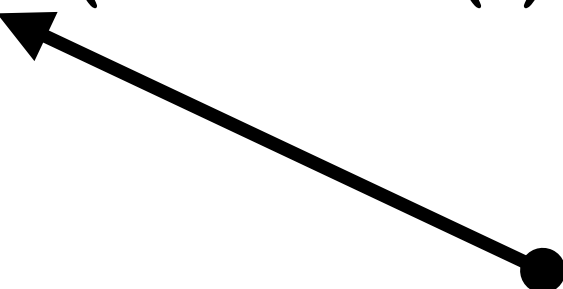
- 1) descriptive label of one thing that should happen
- 2) function for testing that actually happens

ANATOMY OF A SPEC

Describe blocks: contains
[sub-groups of] specs

- 1) descriptive labels of entity to be tested
- 2) function to nest further describes (sub-entities) or its

```
describe('Kittens', function() {  
  describe('eat', function() {  
    it('returns yum', function() {  
      var k = new Kitten();  
      expect(k.eat()).to.equal('yum');  
    });  
  });  
});
```



It block: constitutes a single spec

- 1) descriptive label of one thing that should happen
- 2) function for testing that actually happens

● **Assertion:** making your expectations

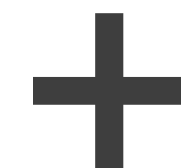
- 1) There can be many assertions within one it block
- 2) If something is thrown *at any point* in an it block, the spec stops and **fails**

TOOLS

- ◉ Jasmine and Mocha/Chai are two popular testing frameworks in the JavaScript ecosystem
- ◉ We chose mocha/chai for the popularity and flexibility
- ◉ However, there are many other options out there!



simple, flexible, fun



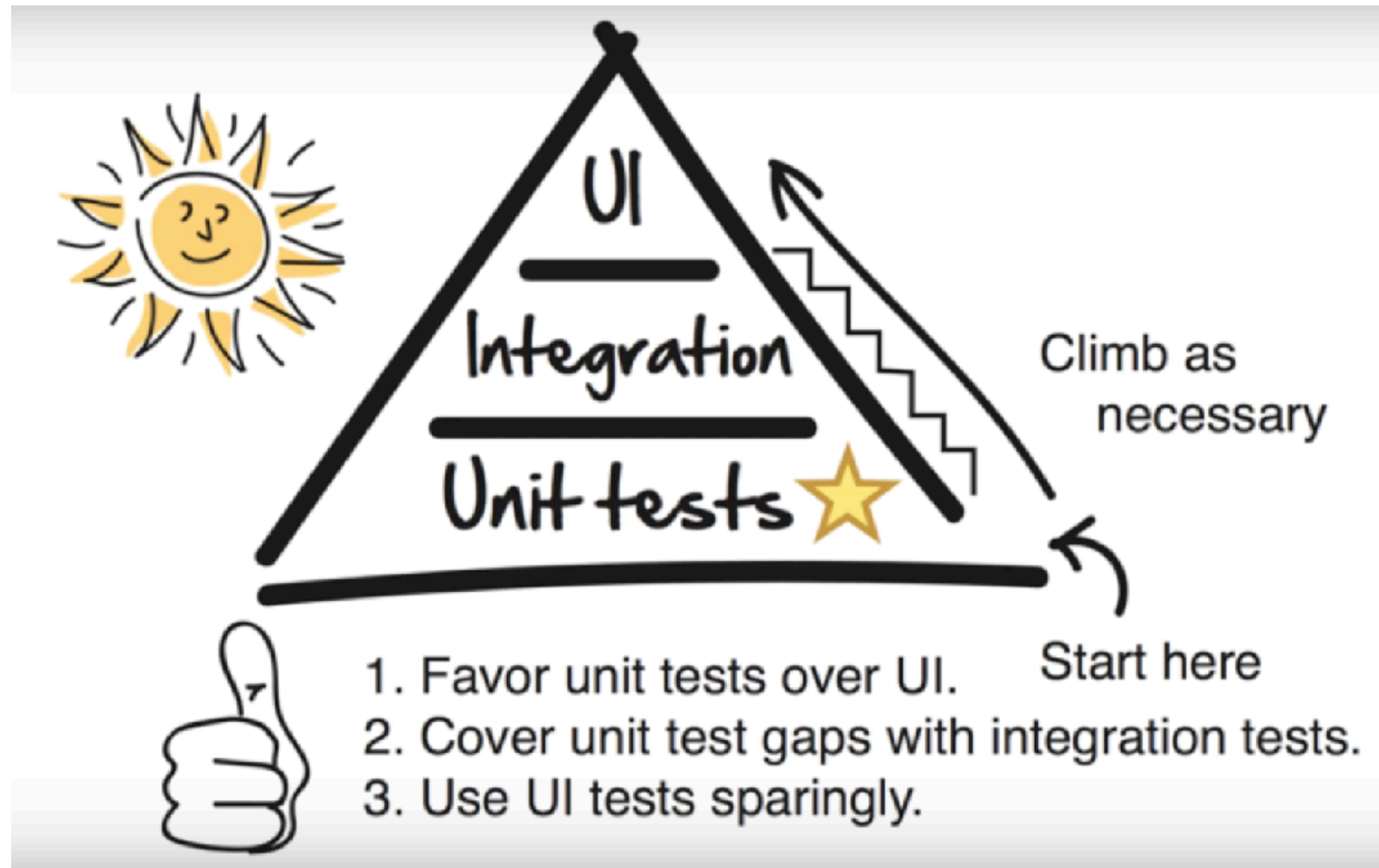
Chai Assertion Library

WHAT DO I TEST FOR?

WHAT

- ◎ Test for behavior, *not* implementation
 - ✗ “I expect this multiply function to use the add function”
 - ✓ “I expect this multiply function to return 6 given the inputs 2 and 3”
- ◎ Implementation details change all the time, but intended behaviors generally do not

TEST PYRAMID

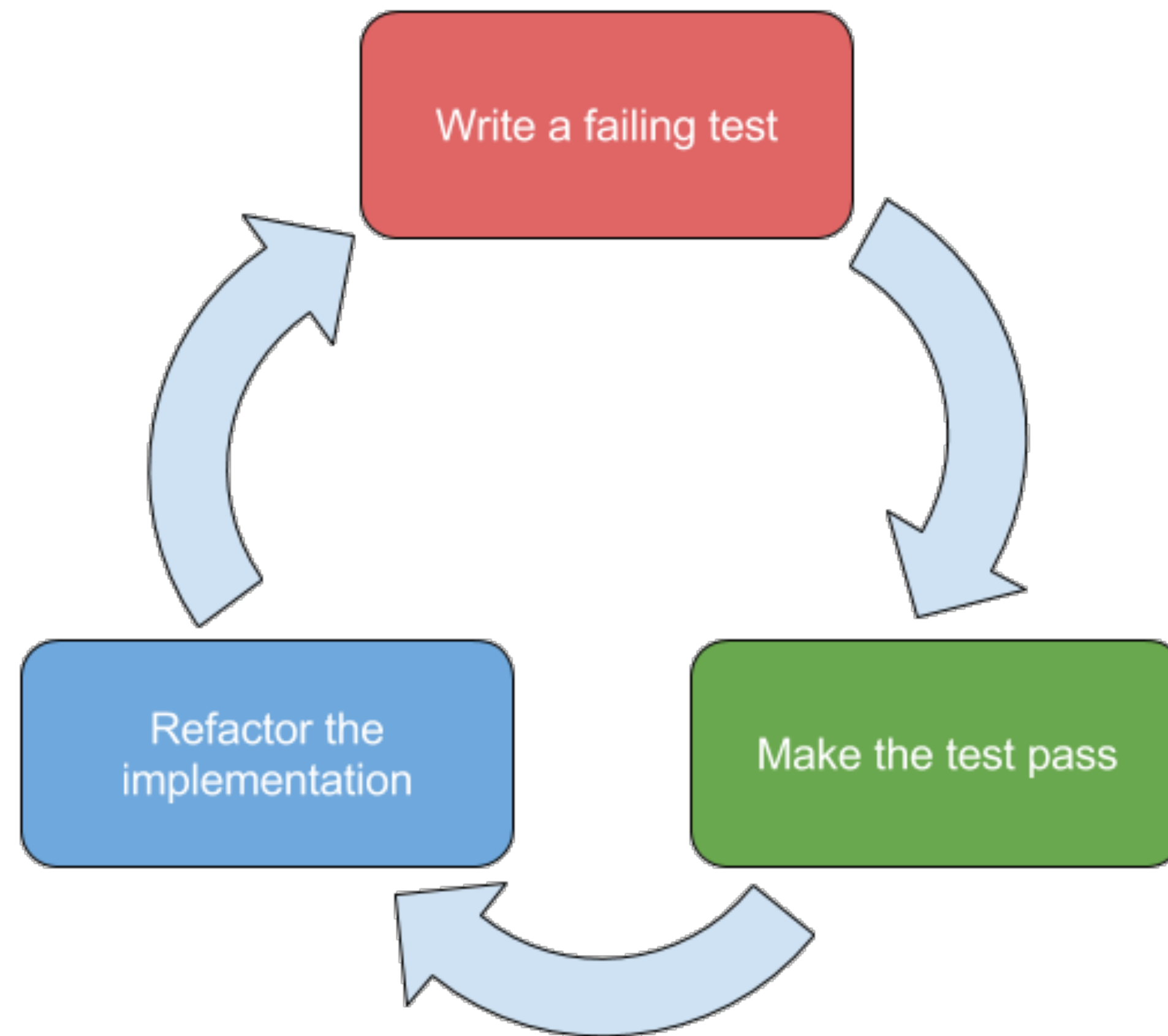


source <http://www.agilenutshell.com/>

TEST-DRIVEN DEVELOPMENT

- ◉ Write tests first, *then* write code to pass the tests
- ◉ Focus on what code does
- ◉ Have a goal
- ◉ Ensure you don't blow off automated testing
- ◉ Improve design and modularity

TEST-DRIVEN DEVELOPMENT



AUTOMATED TESTING \neq TEST-DRIVEN DEVELOPMENT

