# BACKEND TESTING

Fullstack Academy of Code

# War Stories from Production

- Common Scenarios You'll Find in the "Real World"
  - *Legacy Code:* "Someone dumber, sloppier and less good looking than me wrote that code"
  - *Emergency Push:* "It needs to go out right now because the CMO said so"
  - *Rush to finish:* "We'll do our testing in the three months before launch."
  - *Production Destruction:* "It's just a small fix to the database update code."
  - *Spray/Pray:* "Don't worry, QA will find it"
  - *Maintenance Nightmare:* "Only Roy in the basement knows how that module works."

# What are our goals?

- Ensure code is working
- Ensure code will continue to work after someone changes it
- Document what the code actually does
- Make software development more accurate, more professional: “This software works”

# History of Testing

- Pre-1970s: Developers tested their own code
- 1970s: Dedicated Testers following written scripts
- 1980s: Capture/Replay Testing
- 1990s: Scriptable Capture/Replay Testing – rise of Unit Testing frameworks
- 2000s: Unit Testing and the Test Pyramid
- 2000s: Test-Driven Development
- 2015 and Beyond: Death of TDD?

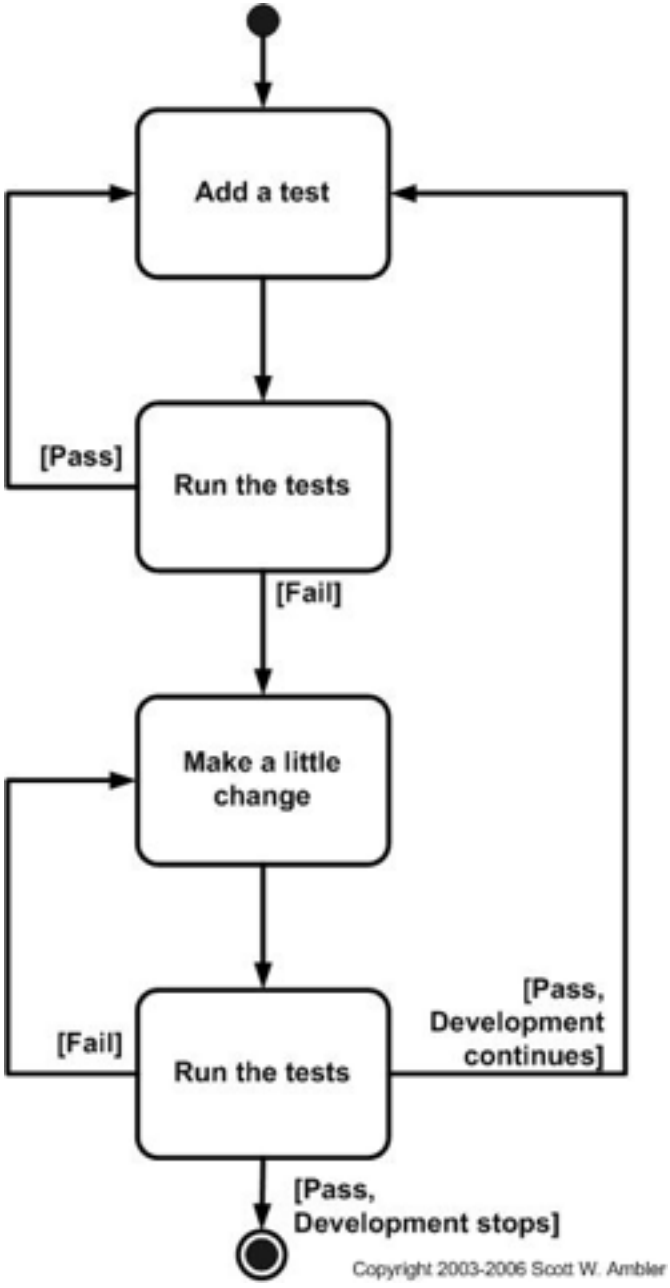
Automated Testing ≠ Test Driven Development

# Test-Driven Development

- What is Test-Driven Development?
  - A practice where you write your automated unit tests BEFORE you write your implementation code

# Why Test-Driven Development

- Focus on what code is supposed to do
- Have a goal
- Ensure you don't blow off automated testing
- Improves design and modularity of code
- "Refactorability"



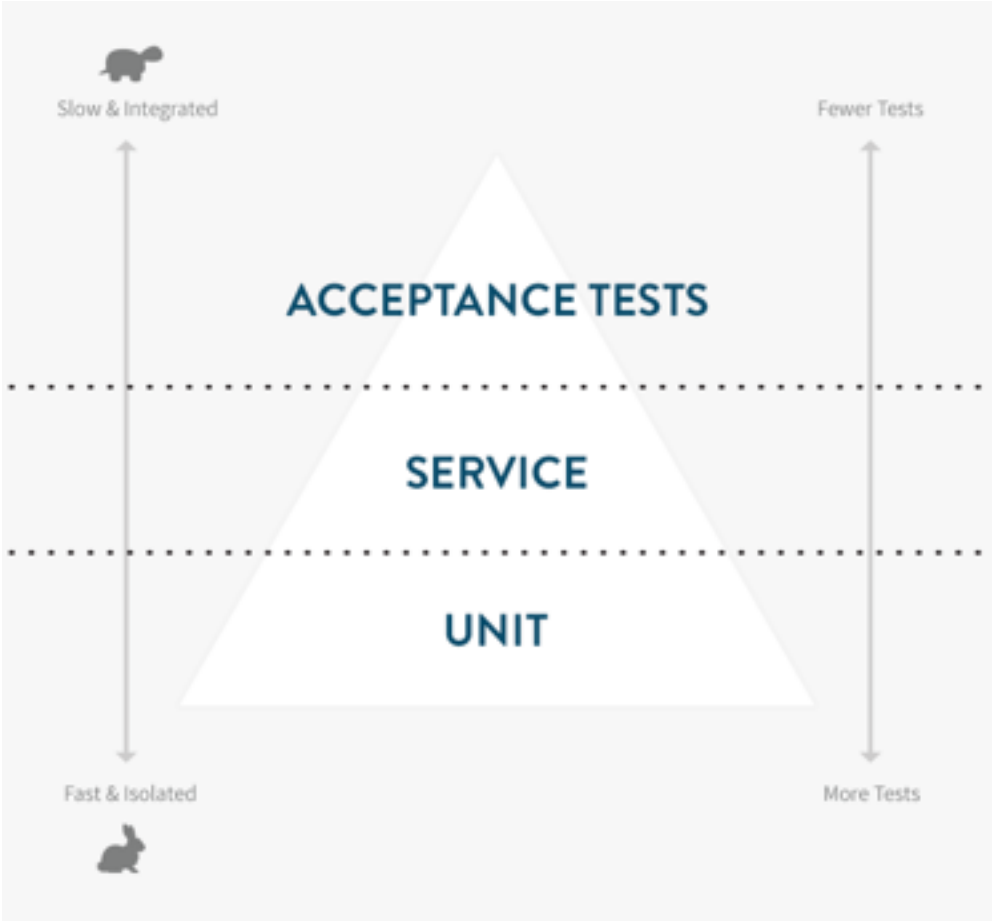
# Getting Started

- Don't overcomplicate testing in your mind – the concept is very simple
  - Code that runs your code
  - Code that makes assertions about what you expect to happen

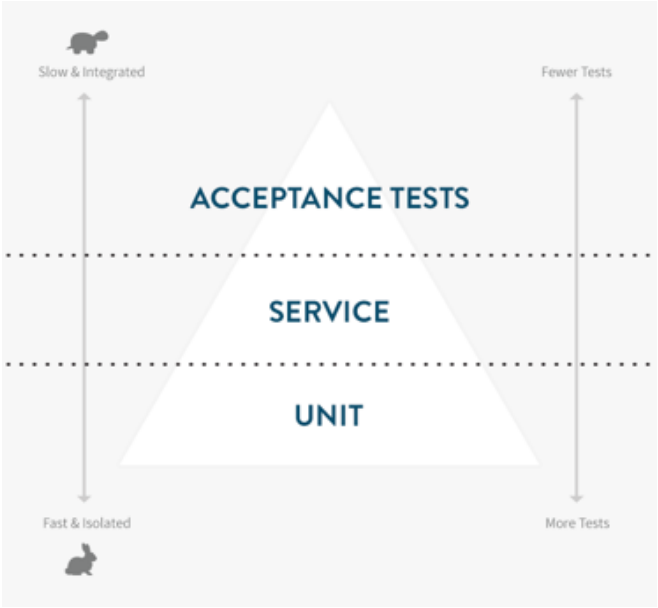
```
/* Our testing library */
 assert = function(result) {
   if (!result) {
     throw new Error("A test failed");
   }
 };
 /* end of testing library */


 /* tests */
 result = MyMathLibrary.add(1, 2);
 assert(result === 3)
```

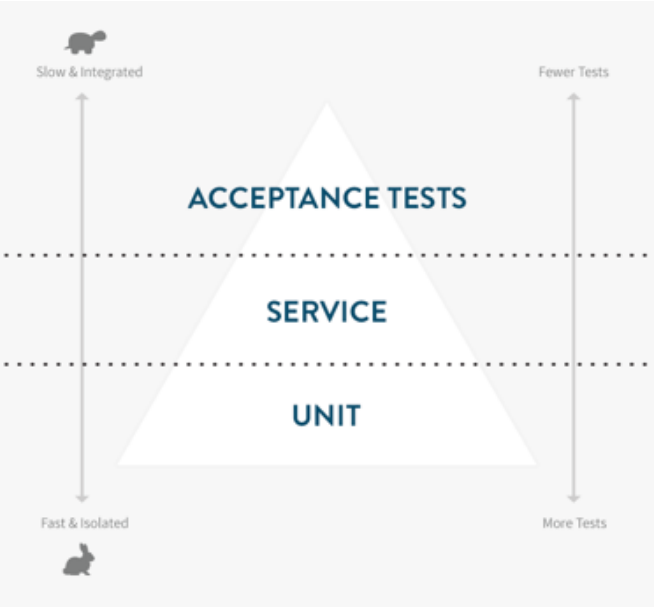
# The Testing Pyramid



# The Testing Pyramid



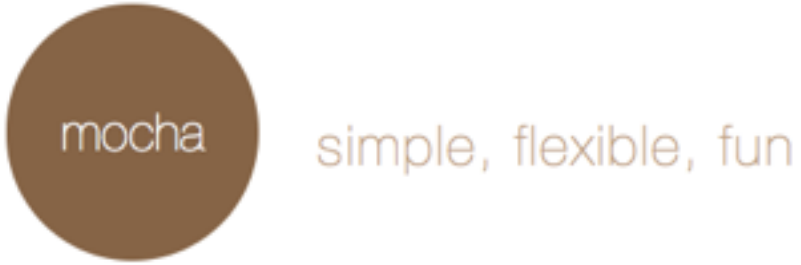

Front-end




Back-end

# Backend Unit: Mocha/Chai

- In JavaScript – the two contenders for most popular testing framework are Jasmine by Pivotal and Mocha/Chai by TJ Holowaychuk

# Testing Models

# Backend Service: SuperTest

- SuperTest runs HTTP requests against your Express server's API routes

```
describe('Trip API', function() {
  describe('GET /trips', function() {
    it('should return an array of trips',
function(done) {
      request(app)
      .get('/trips')
      .expect(200, done);
    });
  });
});
```

# Isolate Tests

- Highly intertwined tests are brittle – change one thing and the whole thing will break
- Reduce brittleness by isolating tests:
  - Reduce your dependence on state
  - Reduce dependence on things running
  - Reduce dependence on everything else working

# Spies, Stubs and Mocks

- Spies
  - Just functions that record how and when they were called
- Stubs
  - Spies + can return preset values
- Mocks
  - Stubs + expectations on how it will be called

Sinon.JS