Introduction to RSpec

Our Goals

- Understand Test-Driven Development as a concept
- Understand the benefits and importance of TDD
- Understand the TDD process
- To be able to write basic RSpec tests

History of TDD

- Mercury Space Program - punch cards from 1959 - 1963
- Gerry Weinberg - punch cards 👇
- Kent Beck - SUnit, jUnit, Extreme Programming, Test Driven Development by Example
- 2004 - Rails!

Punchcard Lyfe

Seibel: How did you learn to program? When did it all start?

Armstrong: When I was at school. I was born in 1950 so there weren't many computers around then. The final year of school, I suppose I must have been 17, the local council had a mainframe computer—probably an IBM. We could write Fortran on it. It was the usual thing—you wrote your programs on coding sheets and you sent them off. A week later the coding sheets and the punch cards came back and you had to approve them. But the people who made the punch cards would make mistakes. So it might go backwards and forwards one or two times. And then it would finally go to the computer center.

Then it went to the computer center and came back and the Fortran compiler had stopped at the first syntactic error in the program. It didn't even process the remainder of the program. It was something like three months to run your first program.

Why do we need it?

- We are in a battle
- We need certainty
- We need it to help us find bugs
- We need it to be able to refactor with confidence
- We need it to help us work in teams
    - The Agile methodology all but relies on it

The benefits

- Less software defects
- Helps reveal design flaws
- 👉 Immediate feedback to the programmer 👈
- Increased confidence
- Assists the refactoring process
- Cleaner and simpler designs

Types of Tests

- **Unit testing**
  - Tests one piece at a time
- **Integration testing**
  - Tests the way that pieces work together
- **Regression testing**
  - Check that new features don't break existing ones
- **Performance testing**
  - Does the system do its job quickly enough

Is there only TDD?

No!

- **TDD**
- **ATDD** (Acceptance test driven development)
- **BDD** (Behavior driven development)
- **MDD** (Mortgage driven development)

The Approach

- Write a failing test
- Run the test to make sure it fails
- Write the code
- Run the test to make sure it passes
- Refactor the code
- Run the test again to make sure it still passes
- Repeat as necessary

The Red, Green, Refactor Cycle

Ruby and Testing

- Ruby developers are very into testing
  - Rails was one of the first frameworks to actually include it by default
- RSpec is the most popular testing library in Ruby

Conventions

- Test the smallest possible pieces of granularity
    - A single function or class for example
- Write as little code as possible to make the test pass
- Seperate common logic
- Treat your tests with respect
- Make your tests expressive
    - Aim to make them self-documenting

Test Structure

- Setup
- Execution
- Validation
- Cleanup

Important Links

- The RSpec Website
- RSpec on Github
- RSpec Documentation
- BetterSpecs

Naming conventions

- Class methods
  - `describe ".new" do ... end`
- Instance methods
  - `describe "#admin?" do ... end`

Let's have a go of RSpec

```
gem install rspec

...

rspec --init
```

How to break tests up

- ***Describe*** - Creates a group of examples
- ***Context*** - Creates a state
- ***It*** - Creates a single example