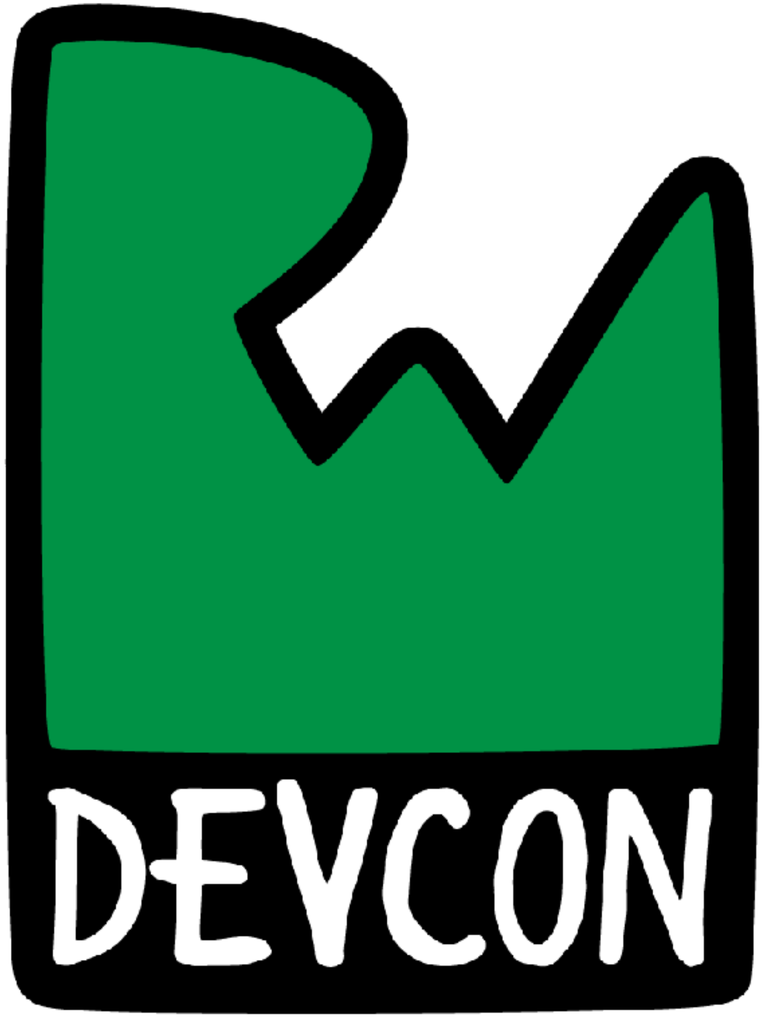
Session 10: Improving App Quality with TDD

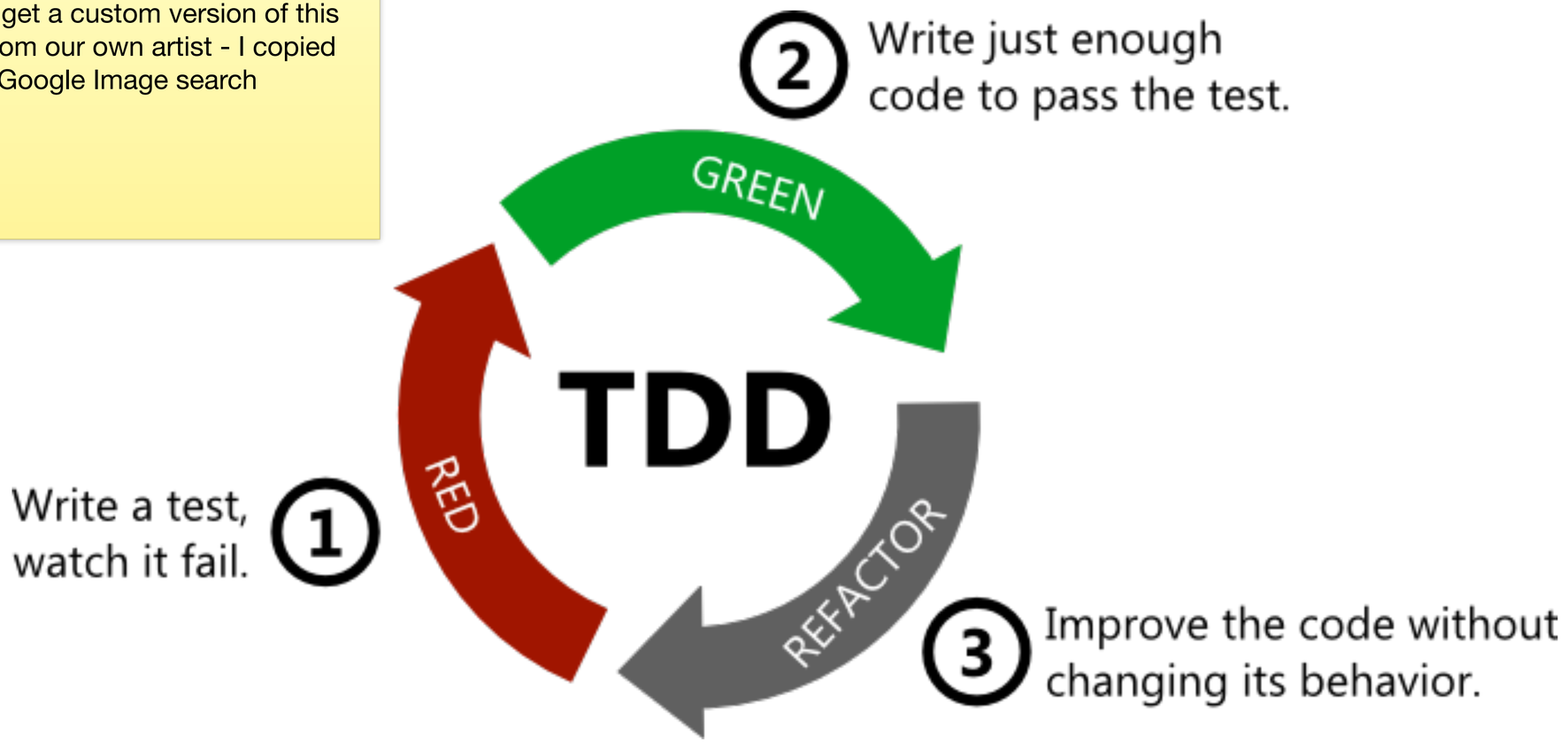


State of Automated Testing in iOS

- Unit Testing, UI testing
- Xcode support for automated testing
- Good third party tools available
- Community adoption grows everyday

What is test driven development?

I'd like to get a custom version of this graphic from our own artist - I copied this from Google Image search



Why do TDD?

- All the benefits of automated testing, plus:
- Ensures tests actually test
- You'll write code that can actually be tested
- Confidently refactor

What isn't test driven development?

- Compulsion for 100% coverage
- A hard and fast rule
- A tool to be used when appropriate

Suggestions for getting started

- Consider a PONO (Plain Old NSObject)
- Fixing a bug? Write a test first!
- Just try it!

Demo 1

More than just models...

- TDD isn't limited to your Model layer
- Also useful for both UI, and networking code

Overheard at the Water Cooler

- “You should never access the network from unit tests.”
- “Your tests will be slow!”
- “The CI server can’t access the dev server!”
- “You’ll cause unnecessary traffic on the DEV server!”
- “What test account will you use!”
- “The data will never be static!”

Well, That's All True

- But it can be really useful
- Connecting a networking layer to a new API can be challenging
- Use tests to help speed up the development cycle
- Be conscious of the long term impact of keeping these tests around

TDD Your Networking Code

- Best practices
  - Only connect to the “real web service” during development
  - Tests should run as fast as possible
  - Data should never change

Mockingjay to the Rescue

- Stub network requests from your tests
  - Makes data reliable
  - Removes network dependency from tests
- Useful for both happy and unhappy paths
- https://github.com/kylef/Mockingjay

Demo 2

And now for the UI!

- User interface code is also worthy of test driven development
- Many of the same concerns of the model layer apply

Types of Code in a UI layer

- Connections to Model code
  - eg. QueryService, Tune
- Code that updates the user interface
  - eg. View controllers, views, IBOutlets, Auto Layout
- Code that interacts with the system or other apps
  - eg. UIApplication, Extensions, CallKit

Challenges Testing This Code

- Objects you may not “own”
- Triggers behavior not ideal in tests
- Storyboards or XIBs
- Dependencies on external resources

Mock Objects to the Rescue

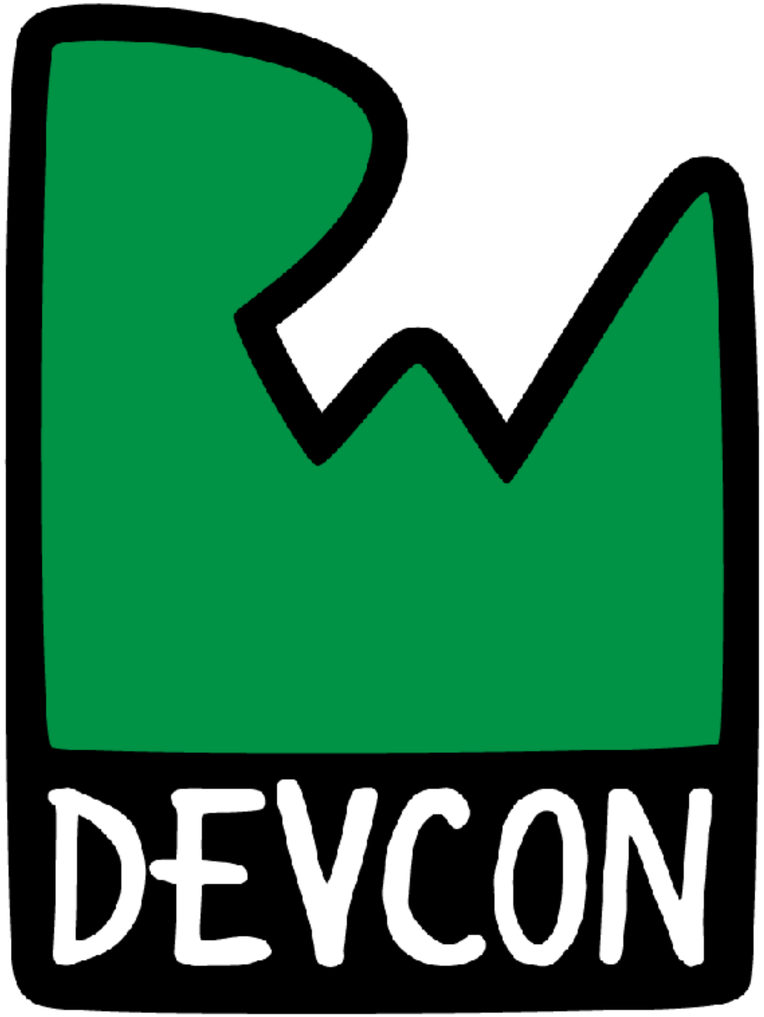
- Mock Object - An object that can be used as a:
  - Replacement in the context of a test to either suppress behavior
  - Provide a repeatable and expected outcome
  - Enable you to verify behavior in the test
- Synonyms: Fake, Stub

Mock Object Implementation Techniques

- Subclass
- Protocols
- Third Party Framework
  - eg. Mockingjay

Demo 3

Session 10: Improving App Quality with TDD



Conclusion

What You Learned

- **Demo 1**: Benefits of TDD, where to get started
- **Demo 2**: TDD techniques for networking code
- **Demo 3**: TDD techniques for user interface code

Where To Go From Here?

- Free tutorials on our site
- https://qualitycoding.org/ - Jon Reid
- “Test-Driven iOS Development with Swift 4” - Book by Dominik Hauser
- Twitter: @obusek