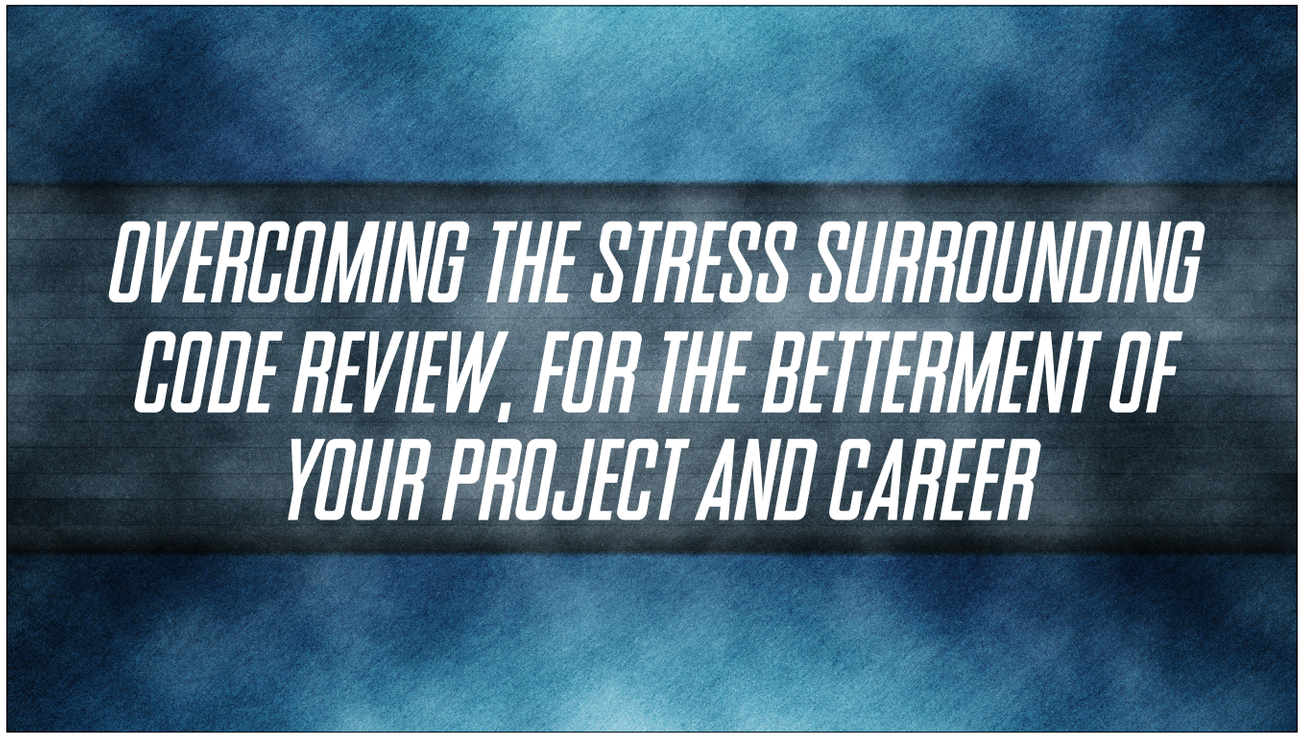
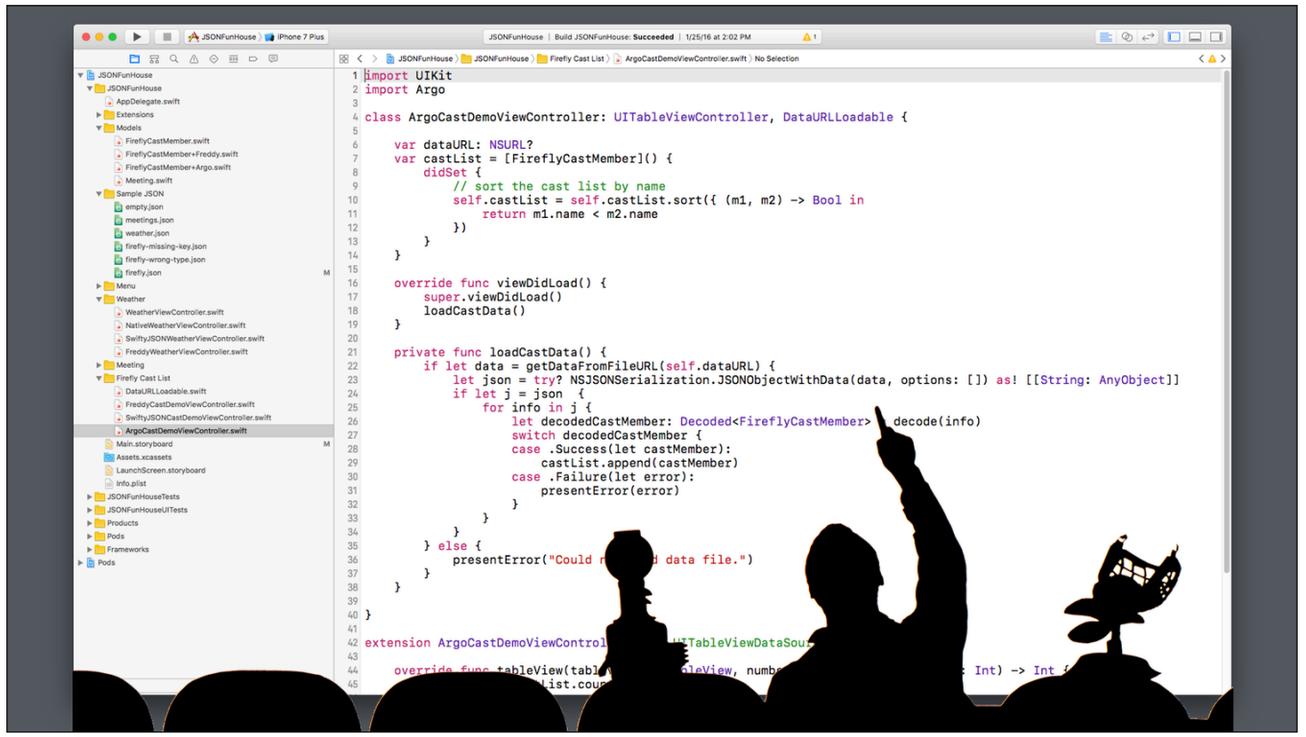
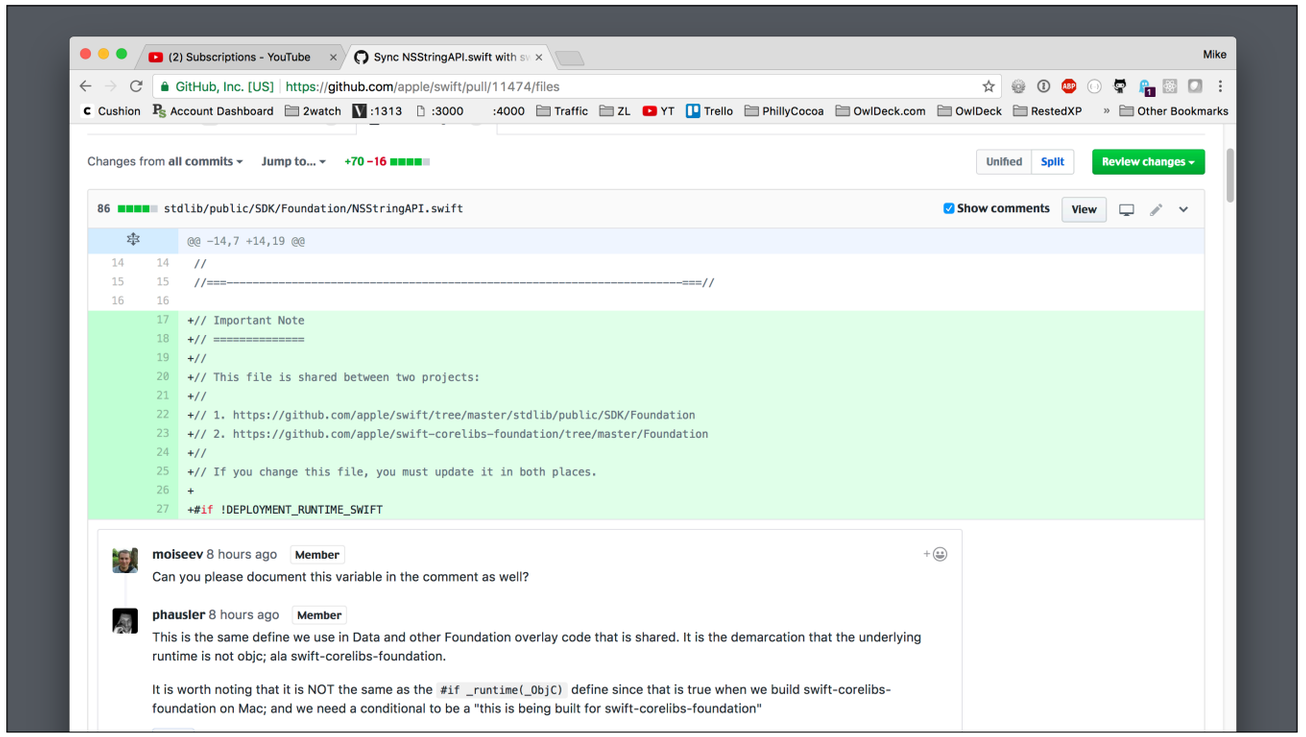

OVERCOMING THE STRESS SURROUNDING
CODE REVIEW, FOR THE BETTERMENT OF
YOUR PROJECT AND CAREER

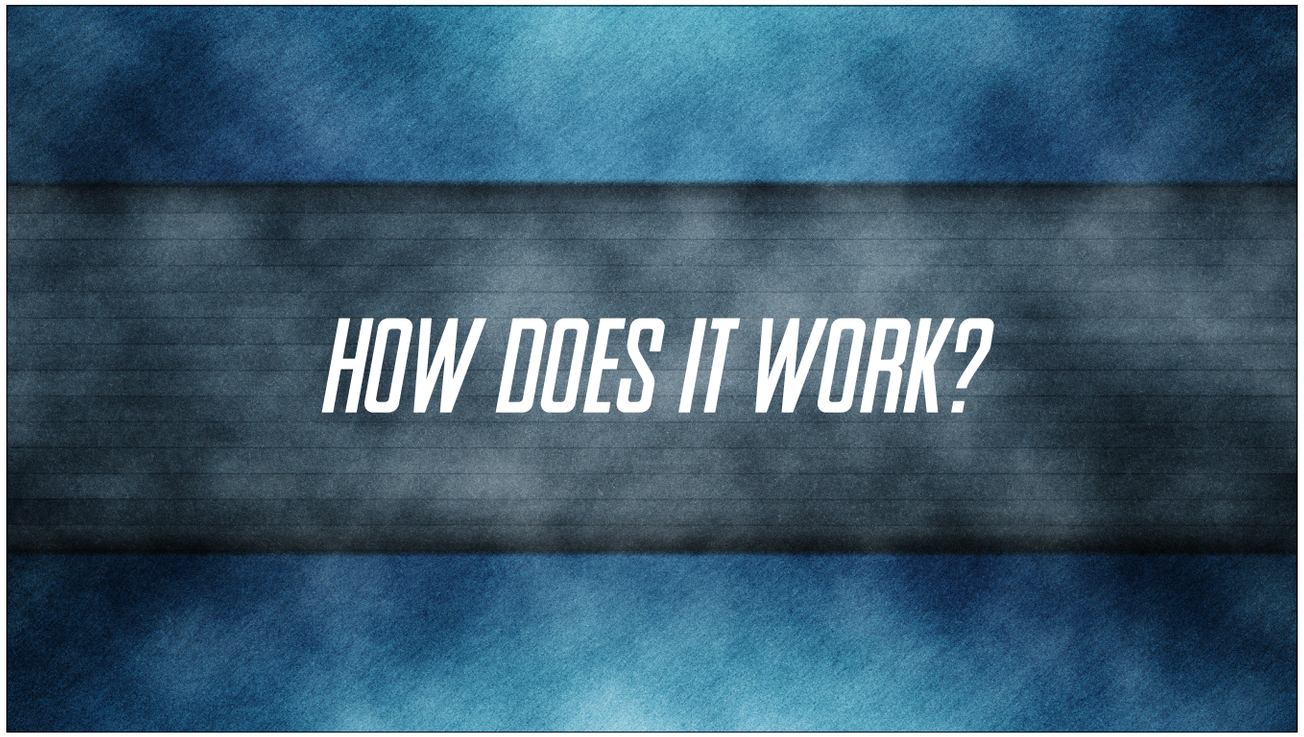
My name is Michael Zornek.
Long time Apple Developer, first on Mac and then iOS.
Currently I’m self employed and do a mixture of consulting, education and product development.
When I’m not working, I like video games.
Questions at end

Code Review is a practice where a code change or addition is first assessed by one's peers for correctness and style before being added to the main codebase. In GitHub parlance we call this a Pull Request.

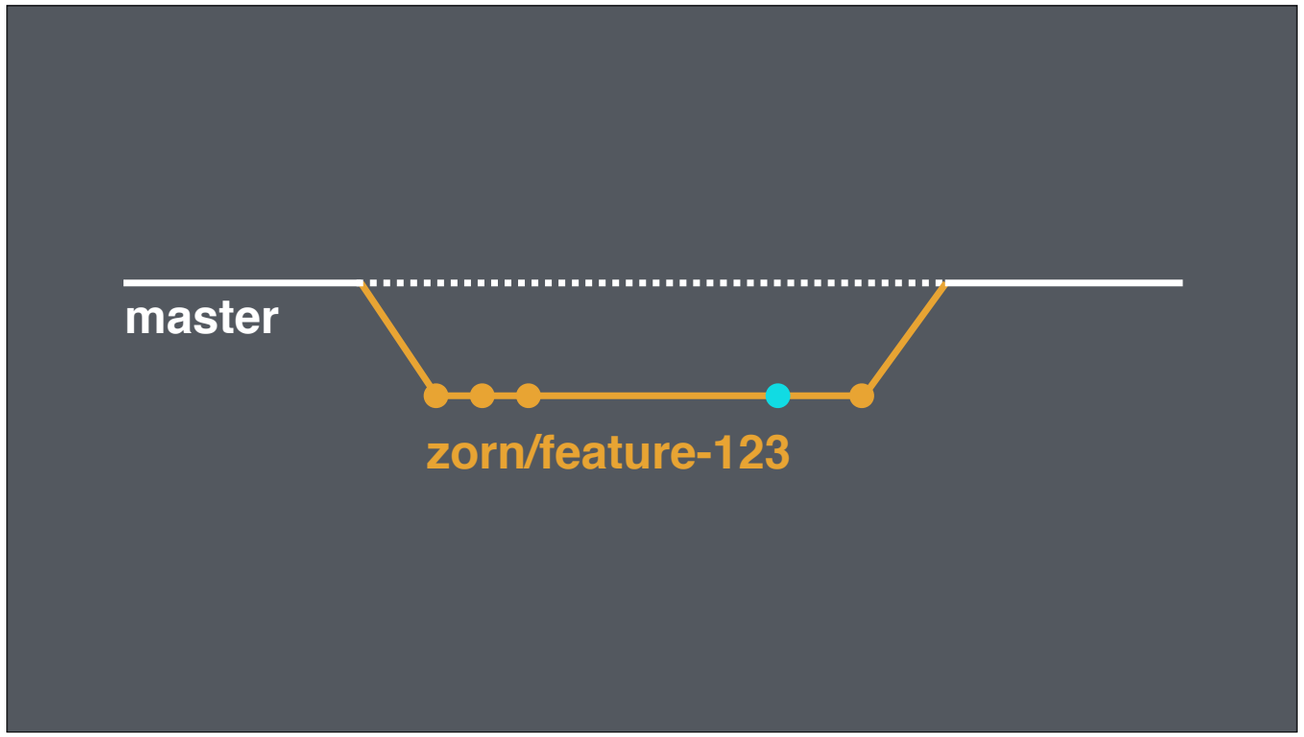


Some might visualize it like this…


(2) Subscriptions - YouTube
Sync NSStringAPI.swift with
Mike
GitHub, Inc. [US] https://github.com/apple/swift/pull/11474/files
Cushion
Account Dashboard
2watch
:1313
:3000
:4000
Traffic
ZL
YT
Trello
PhillyCocoa
OwlDeck.com
OwlDeck
RestedXP
Other Bookmarks
Changes from all commits
Jump to...
+70 −16
Unified
Split
Review changes
86 stdlib/public/SDK/Foundation/NSStringAPI.swift
Show comments
View
@@ -14,7 +14,19 @@
//
//===--===//
+// Important Note
+// ==============
+//
+// This file is shared between two projects:
+//
+// 1. https://github.com/apple/swift/tree/master/stdlib/public/SDK/Foundation
+// 2. https://github.com/apple/swift-corelibs-foundation/tree/master/Foundation
+//
+// If you change this file, you must update it in both places.
+
+#if !DEPLOYMENT_RUNTIME_SWIFT
moiseev 8 hours ago
Member
Can you please document this variable in the comment as well?
phausler 8 hours ago
Member
This is the same define we use in Data and other Foundation overlay code that is shared. It is the demarcation that the underlying runtime is not objc; ala swift-corelibs-foundation.
It is worth noting that it is NOT the same as the #if _runtime(_ObjC) define since that is true when we build swift-corelibs-foundation on Mac; and we need a conditional to be a "this is being built for swift-corelibs-foundation"



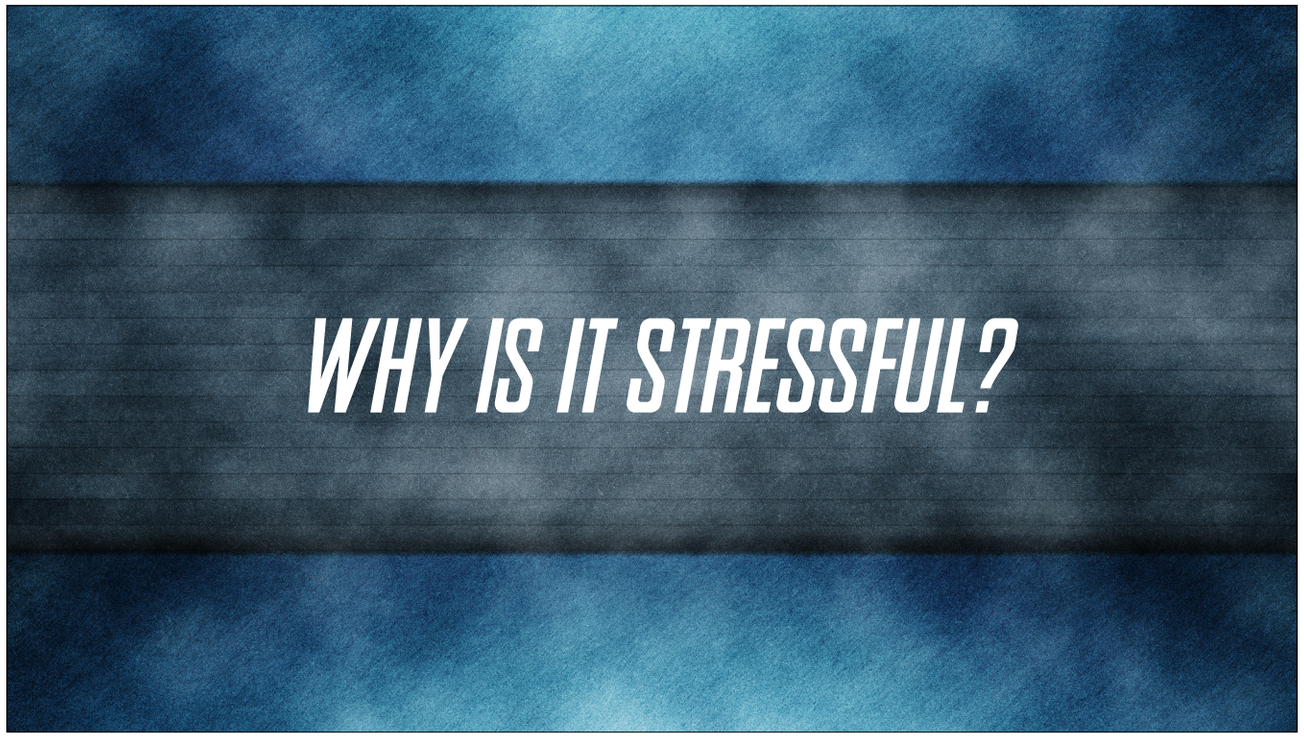
Depending on the nature of the project your Pull Request may be asking to merge a git branch from a single repo into it's own master or from a repo fork, in the case of many open source projects.



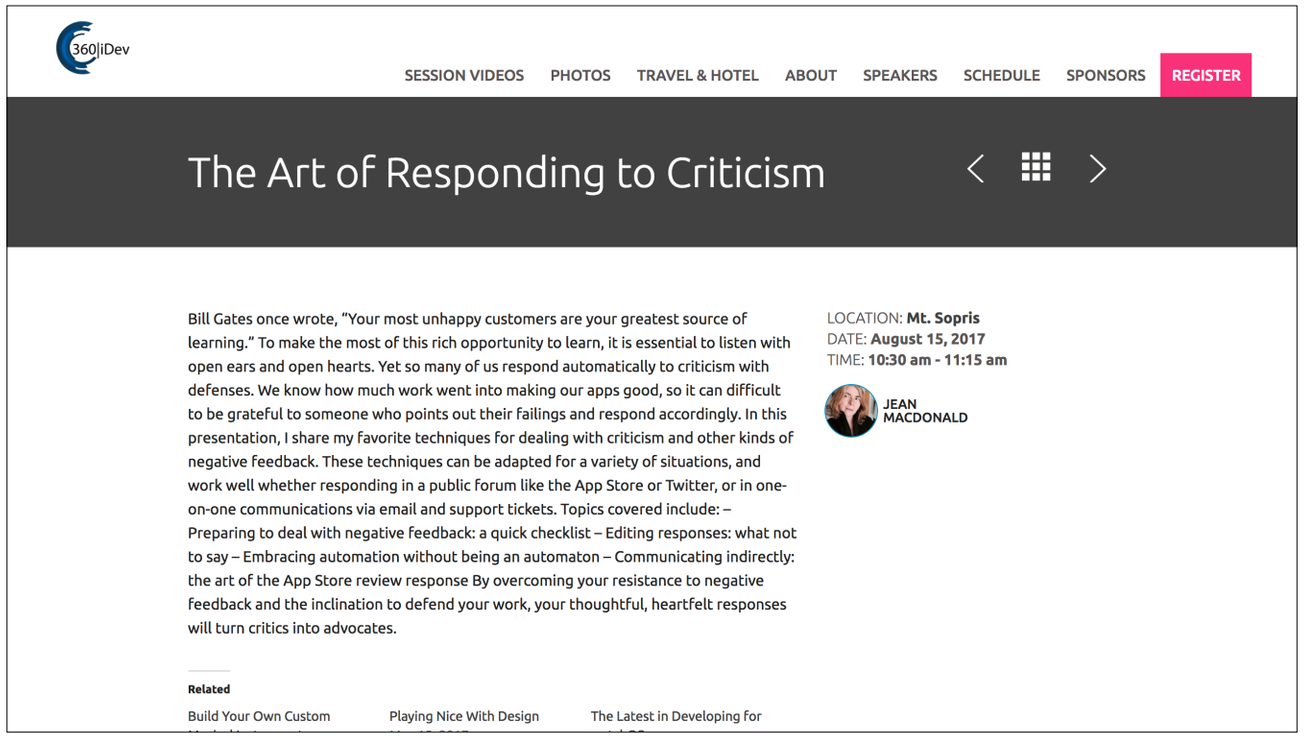
In the simplest of cases, where everything is inside a single repo…
You branch off of master, naming your branch with the author, a short description and the id of the story this work is being done for.
You do your work and when you'd like it to be merged back into master you'll make a pull request.
From the PR you'll get some feedback and make some more changes and then the merge is ready.

Beneficial for the project...
* Verifies the **new behavior does what it says** with **no other negative impact** to the project.
* Makes sure **new code follows a project's design rules** and is otherwise community standards for correctness.

Beneficial for the programmers...
* Having others give you **regular feedback on your code/solutions will help you grow** and become a better programmer.



Giving good criticism is hard.
Accepting and learning to use criticism is hard.
It’s a very human thing that will not be taught as part of a comp sci degree.



Consider checking out this video as a supplemental to this talk.

RESPONSIBILITIES OF
THE POSTER

Ready for master integration.
Work In Progress (WIP) PR are fine form comments, I usually delete them in favor of a clean review PR.
For large work, consider a feature branch and then branch off of that in small increments.

The reviewer is not your first pass of QA. You should have high confidence that this new code achieves the desired behavior change without negative side effects before making a pull request.
Finding problems in the behavior should be the exception and rare. If it happens often it will come off as if you don't respect your code reviewer's time.

The PR should have all the information a person needs to accomplish the review.

```
<Story Title> [Delivers #xxxxx]

## Finishes

https://www.pivotaltracker.com/story/show/xxxxx

## Additions/Changes

* One
* Two
* Three

## Testing

Verify this. Use real hardware because of x.
```

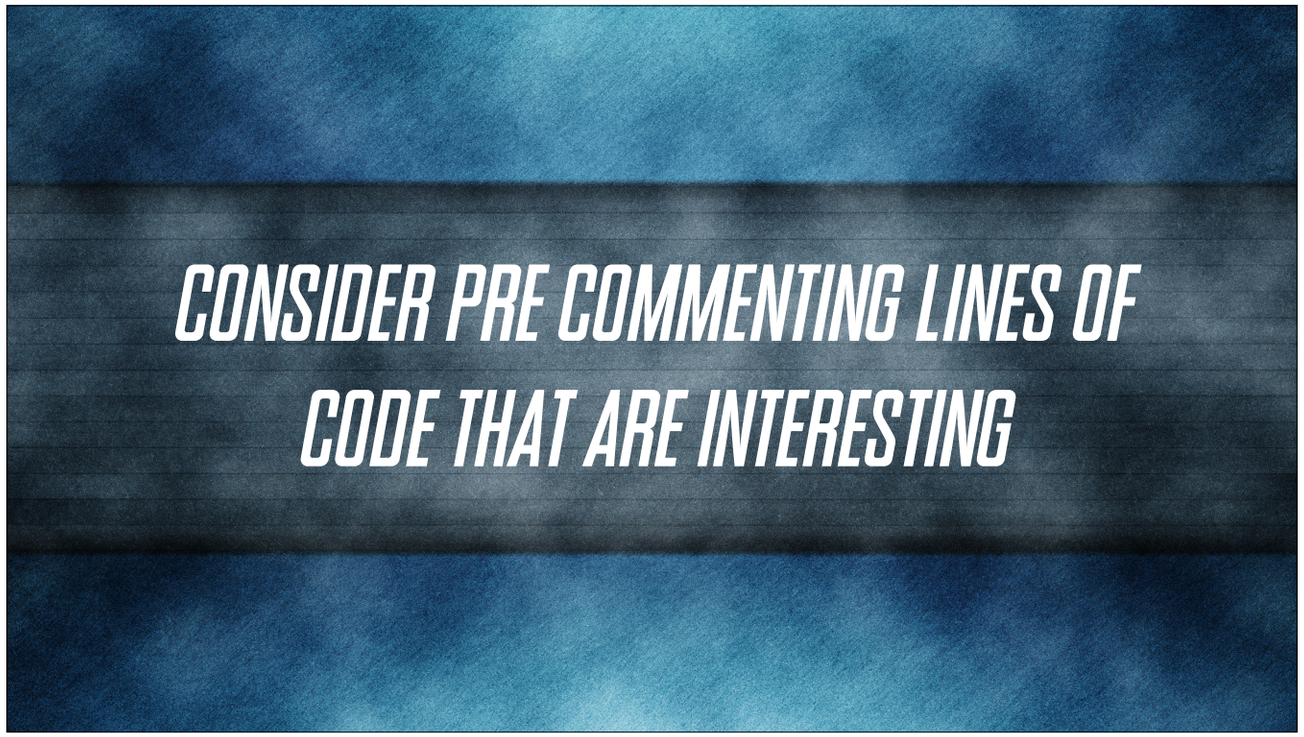
My Format
Make it clear what behaviors need to be tested. Don't assume the reviewer knows the dependancies of your app.
“I edited the import code which effects features A, B and C."

```
<Story Title>

As a user,
I want FEATURE,
So that I ACCOMPLISH_GOAL.

Acceptance Criteria:

* One
* Two
* Three
```

```
<Story Title>

As a user,
I want FEATURE,
So that I ACCOMPLISH_GOAL.

Acceptance Criteria:

* One
* ~~Two~~ Moved: #1234
* Three
```

Don’t be afraid to move behaviors if needed.

The PR should be **based off of master and be kept up to date**.
It is **not the responsibility of the reviewer** to merge in master and update merge conflicts.
If you must make a PR based off a branch in review **make sure it's clear that the parent branch needs to be reviewed/merged first**.

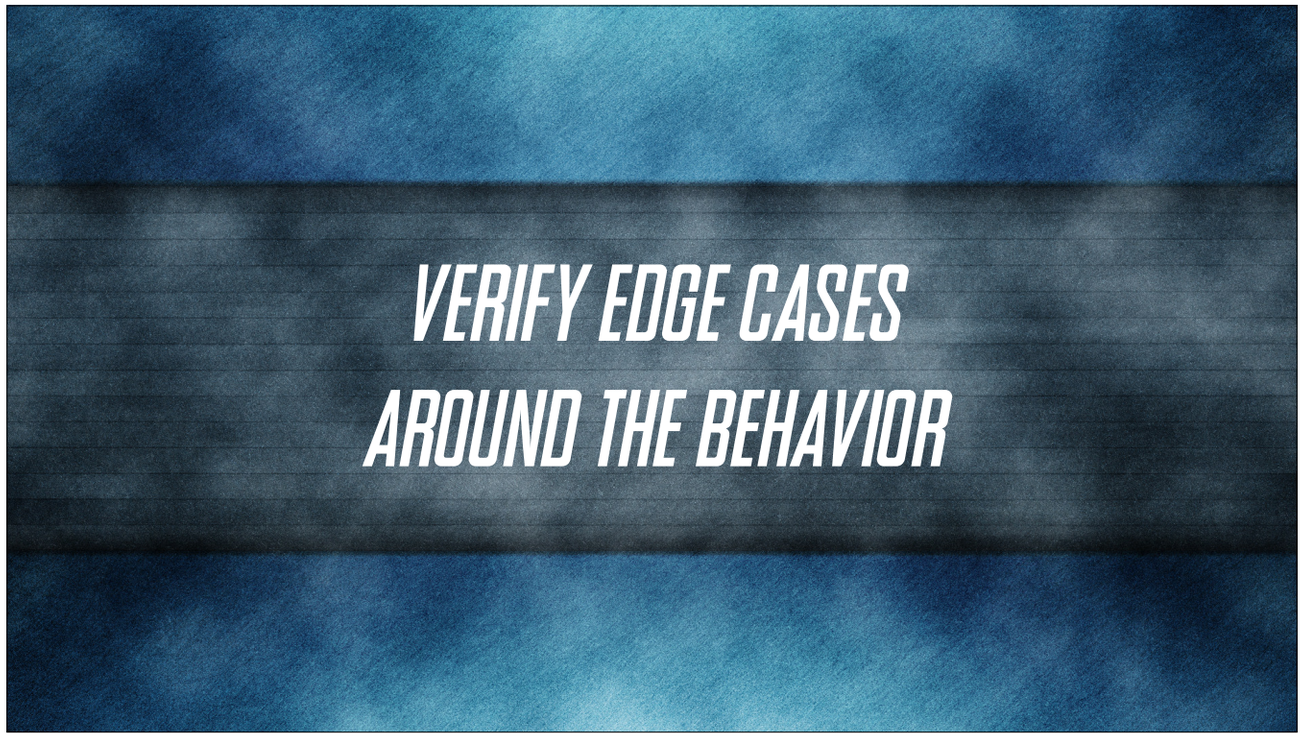


Before you hit submit, scan your code like a reviewer will.
If you see something that you think the reviewer might have an issue with, add a comment and pre-explain your thinking.

RESPONSIBILITIES OF
THE REVIEWER

Code Review should be your top priority.
Open PRs are technical debit.

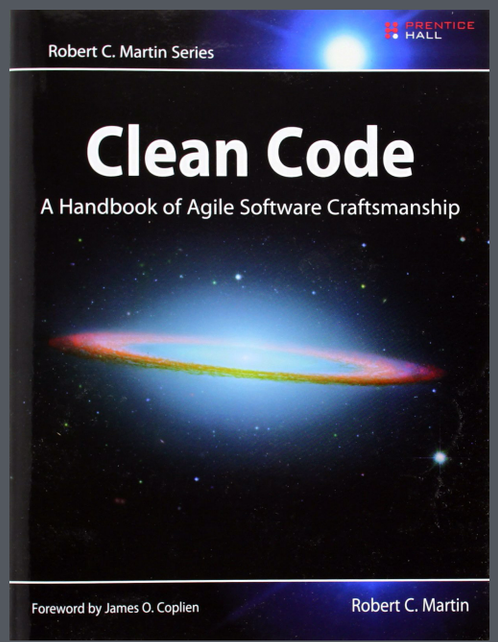
First, run the app as a user would, verifying behavior.

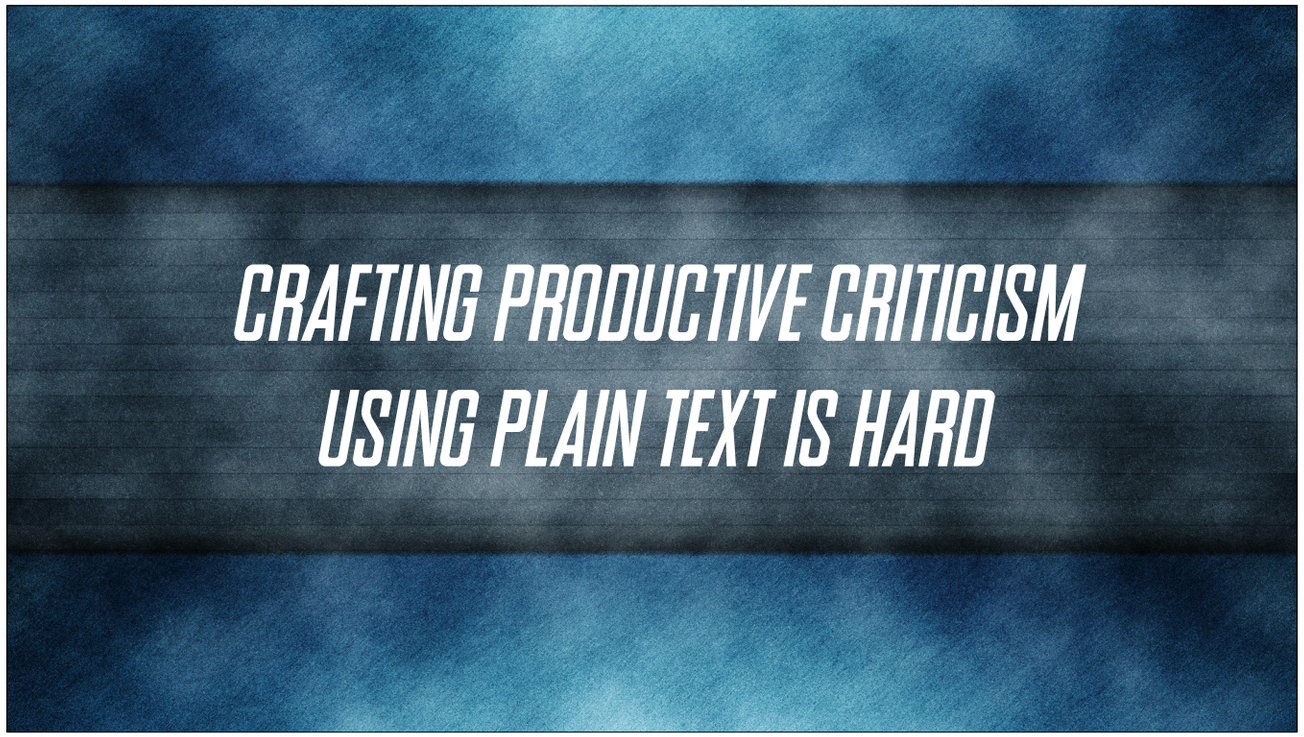


Next, run the app as a edge case user would…

Is this code something that could be put onto a projection slide and you'd be proud of it?

- Hard to comprehend
- Single Responsibility Principal
- Code Duplication
- Boy Scout Rule
- Off by One Errors
- Large Input Errors
- Error Handling
- Performance
- Old Devices
- Low Memory
- Method Names
- Variable Names
- Method Length
- File Length
- Docstrings
- NSLocalizedStrings
- Unit Tests
- Testing Coverage


Robert C. Martin Series
PRENTICE HALL
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Robert C. Martin



"This should be named foo." vs
"I think foo would be a better name because…", or
“why did you name this foo?”

Because code review happens over plain text it is very easy for people to take criticism too harshly or personal. If you feel like that might happen or has happened setup some face to face time to discuss the PR.

Remember it's about a code change, not a defect of the author.

My personal opinion, if it can’t be automated via a linter, I don’t comment about it.

Let people know when they are doing good work too!

1-2 reviewers

Horror story of 10+ reviewers on a new team member’s PR

RESPONSIBILITIES OF
THE POSTER


BE TIMELY IN YOUR RESPONSES

Try to separate yourself from the code.

Don't be afraid to push back a little. This isn't a must do change list. It's a conversation amongst colleges. Ask for more info, or have a talk offline about it.

Things sometimes get heated more quickly over text. Hard to read emotion, easy to be influenced but outside aggressions. Don't be afraid to talk in person.

Document the PR branch with offline discussions so others on the team know what happened.

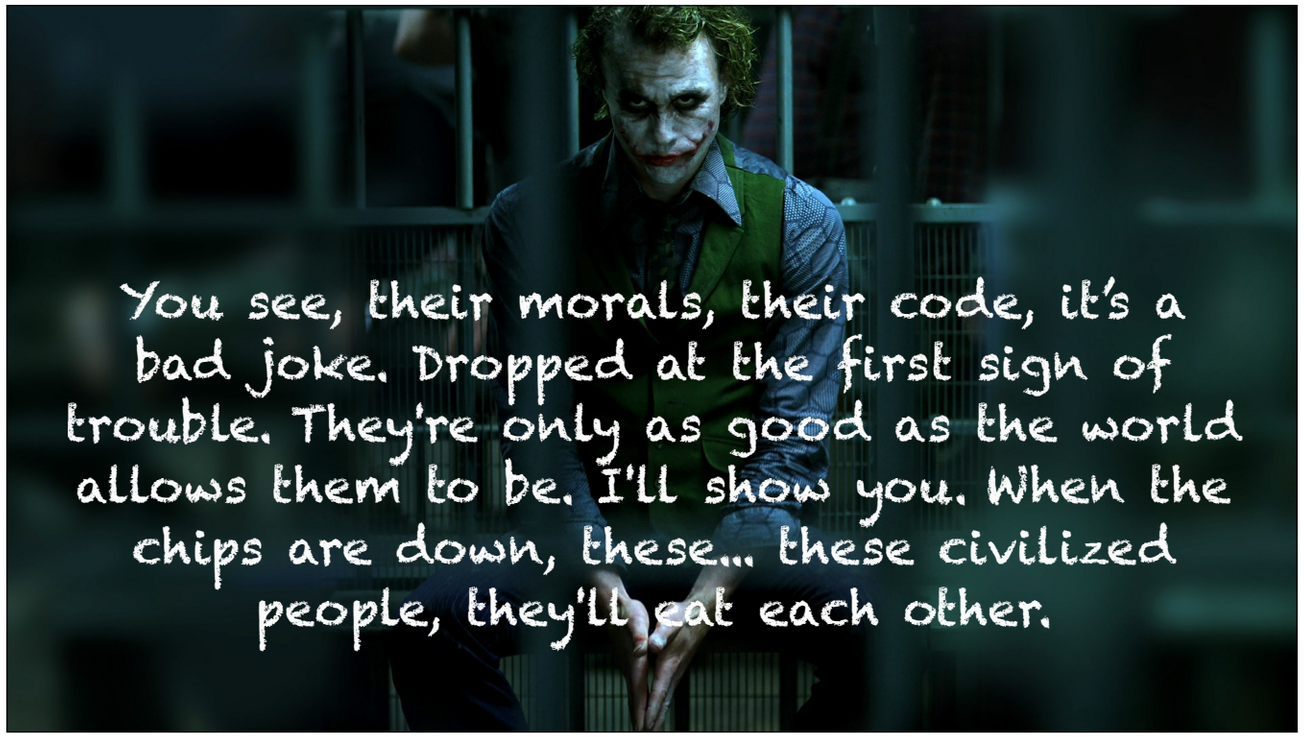
CRUNCH TIME REVIEWING


DON'T SKIP REVIEWING


DON'T SKIMP REVIEWING

This is when many "broken windows" will get added.

The theory states that maintaining and monitoring urban environments to prevent small crimes such as vandalism, public drinking, and toll-jumping helps to create an atmosphere of order and lawfulness, thereby preventing more serious crimes from happening.

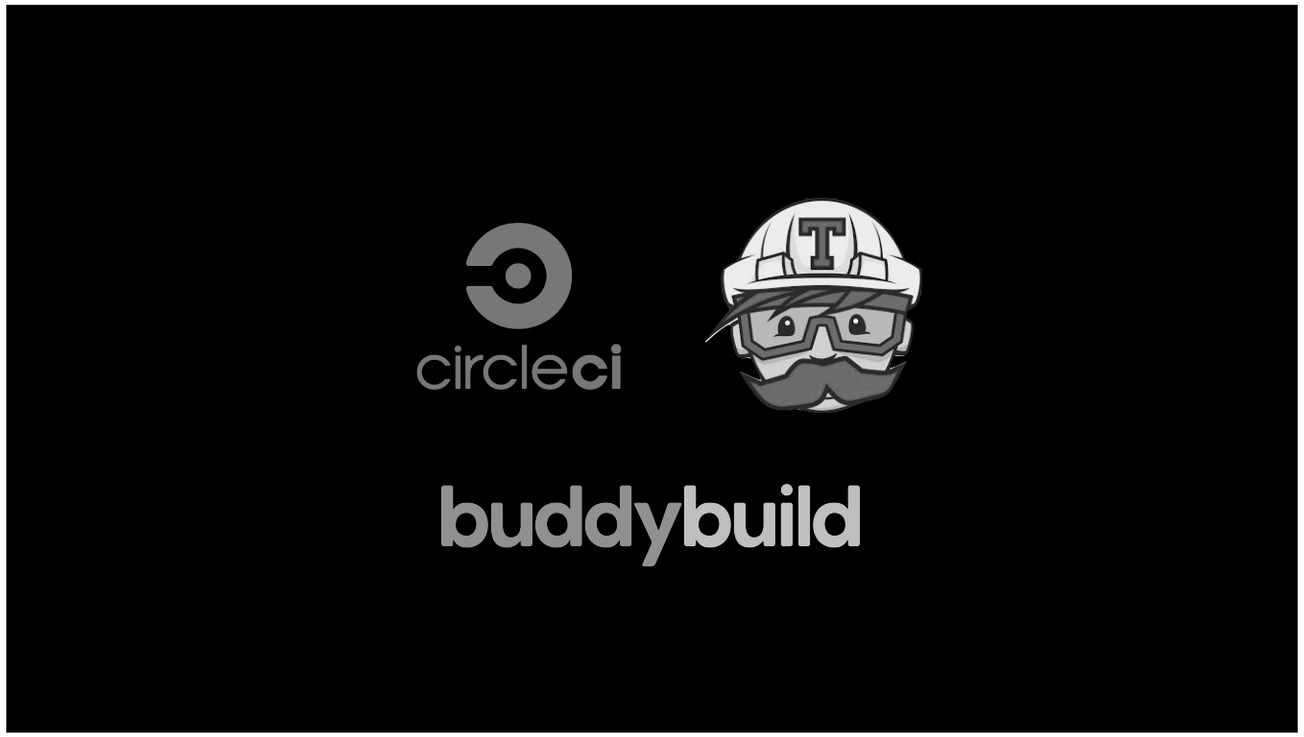

You see, their morals, their code, it's a bad joke. Dropped at the first sign of trouble. They're only as good as the world allows them to be. I'll show you. When the chips are down, these... these civilized people, they'll eat each other.

Accepts merges that…

- add errors / stops the build
- add warnings
- break user behavior
- are not associated with a ticket/story/sprint
- have not been code reviewed
- are not backed up by new tests
- does not contain inline method documentation

TIPS AND TRICKS

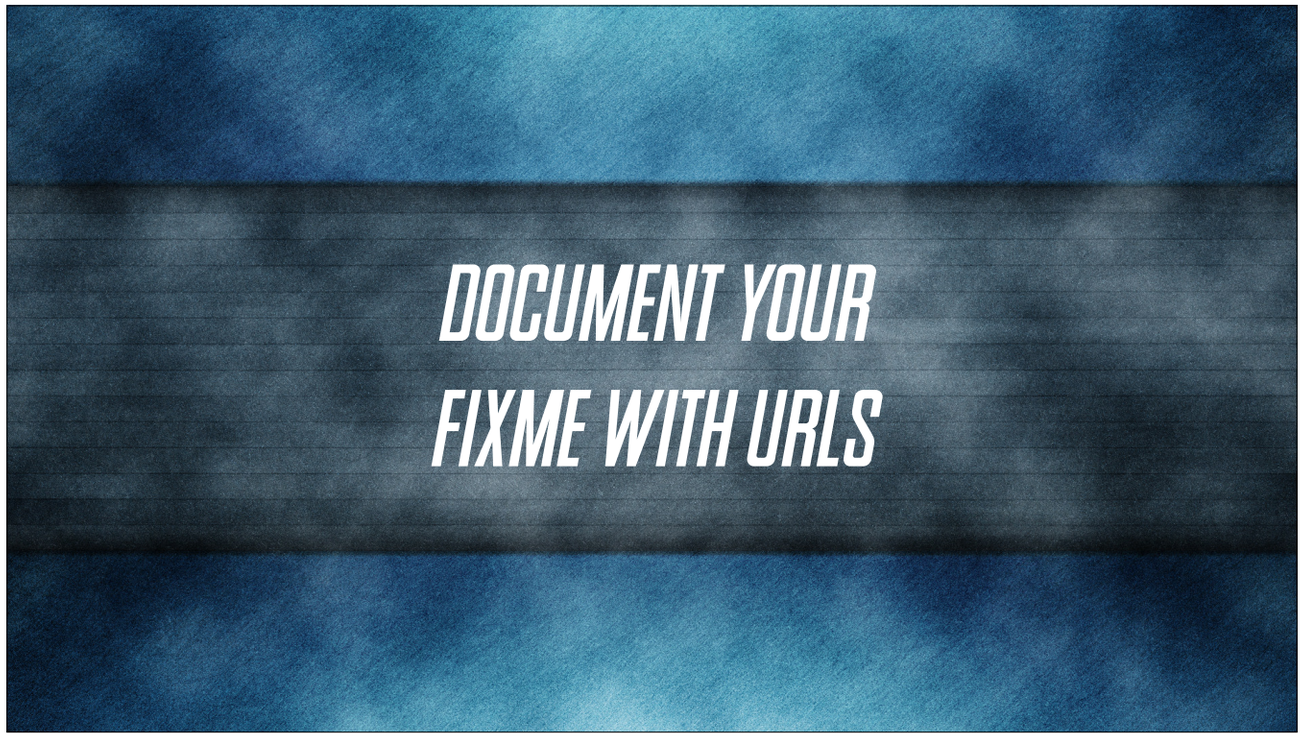
If you are getting too many rubber stamped reviews, consider a week long policy where at least 1 defect has to be commented on. Consider bringing on more experienced developers for pair review sessions.



If possible, get a CI system built to auto run the test suit on the PR branches.

New code should be a surprise. Its usually a good idea to sit down with your team and review how you plan to do something before you build it. It's a real waste to build something and then only to find out it was done the wrong way.

* Try to avoid branches that have more than a days worth of work. General good target is 1-2 PRs a day.
    * If you have a large feature that can not be intergrated into master, consider making a master-with-feature-x branch and work off of that for a week or so -- but don't get too far off master.
    * If you have a large Library update like cocoapods, do this in it's own PR.



Document your chore and known bugs in the code with links to your bug tracker.

* Try to find people who can consistently review your code (project teams greater than 1).
* Also consider new people on occasion as they will come at you with fresh ideas and no assumptions.

* Helpful as a paper trail for your work.
* Good to look at the change in isolation. (Difference between reading code in Xcode vs the web.)
* Delay solo PR merges to be 24 hours min after they are posted. Do the review with fresh eyes the next day.

THANK YOU!

MIKE ZORNEK