Refactoring Legacy Code

@sugendran

Yammer Frontend is a busy place

24 Engineers across 3 offices

5 years of experimentation

Mix of open source and custom frameworks

Refactoring the Thread List

Core component of Yammer

Had the most amount of experimentation

Hard to understand and slow to work with

Don’t hate past team mates

Past decisions were made for good reason

Rewrite vs Refactor

Time to ship is the most important

Many years of changes and bug fixes

Consider the costs to other projects

Working alone is dangerous

No one works alone at Yammer

On our own we don't have all the answers

A stubborn person will help keep you honest

Understand the domain

Do you know all the use cases?

Not everything is written down

Limit the scope after the research

Release early, release often

Get into master as fast as possible

Use a feature gate

Allows the team to be aware of changes

Fork the tests

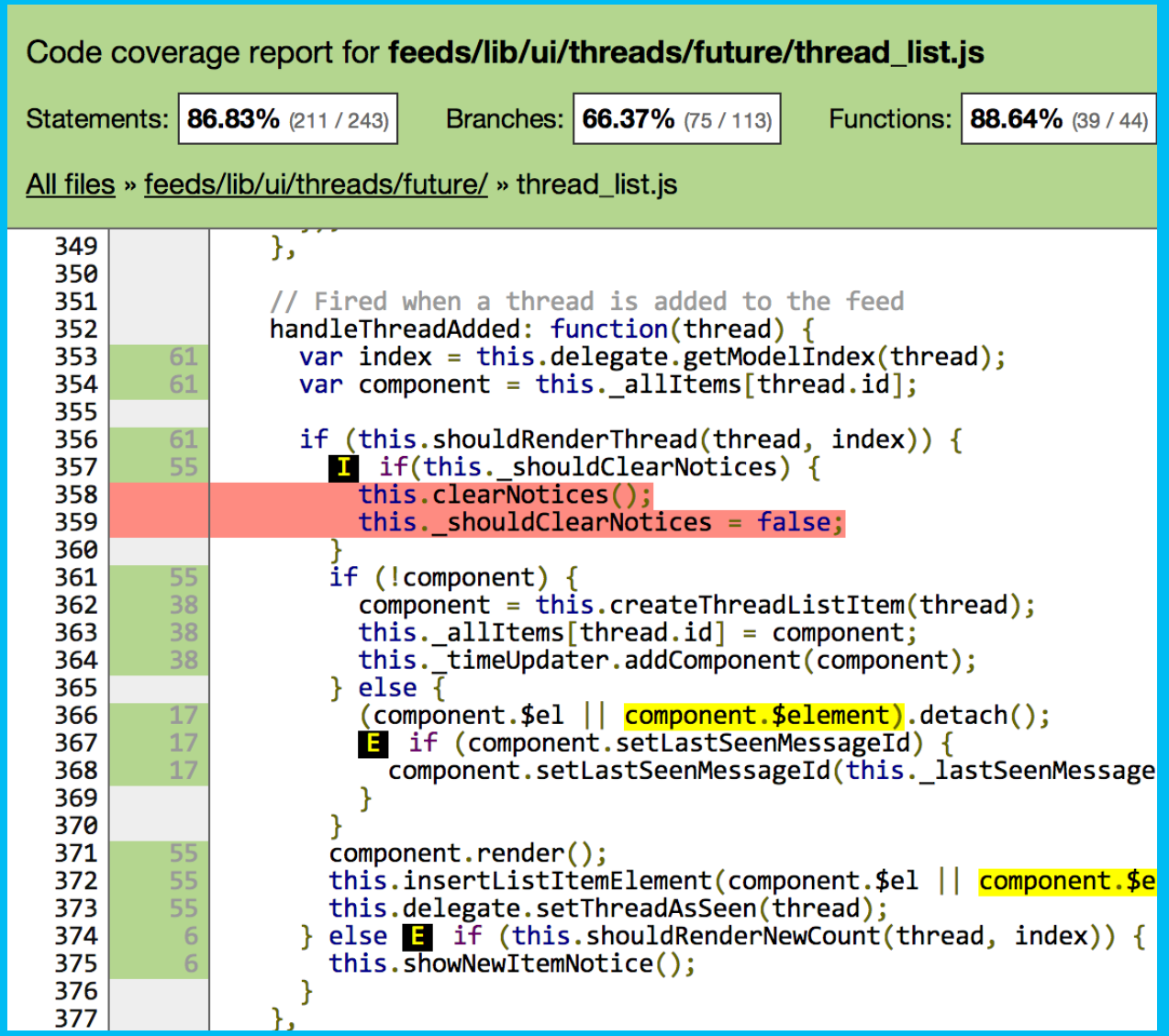
Tests give you confidence

Spent the last year getting coverage up

You can’t refactor without tests

Instanbul

http://gotwarlost.github.io/istanbul/



Think about the API

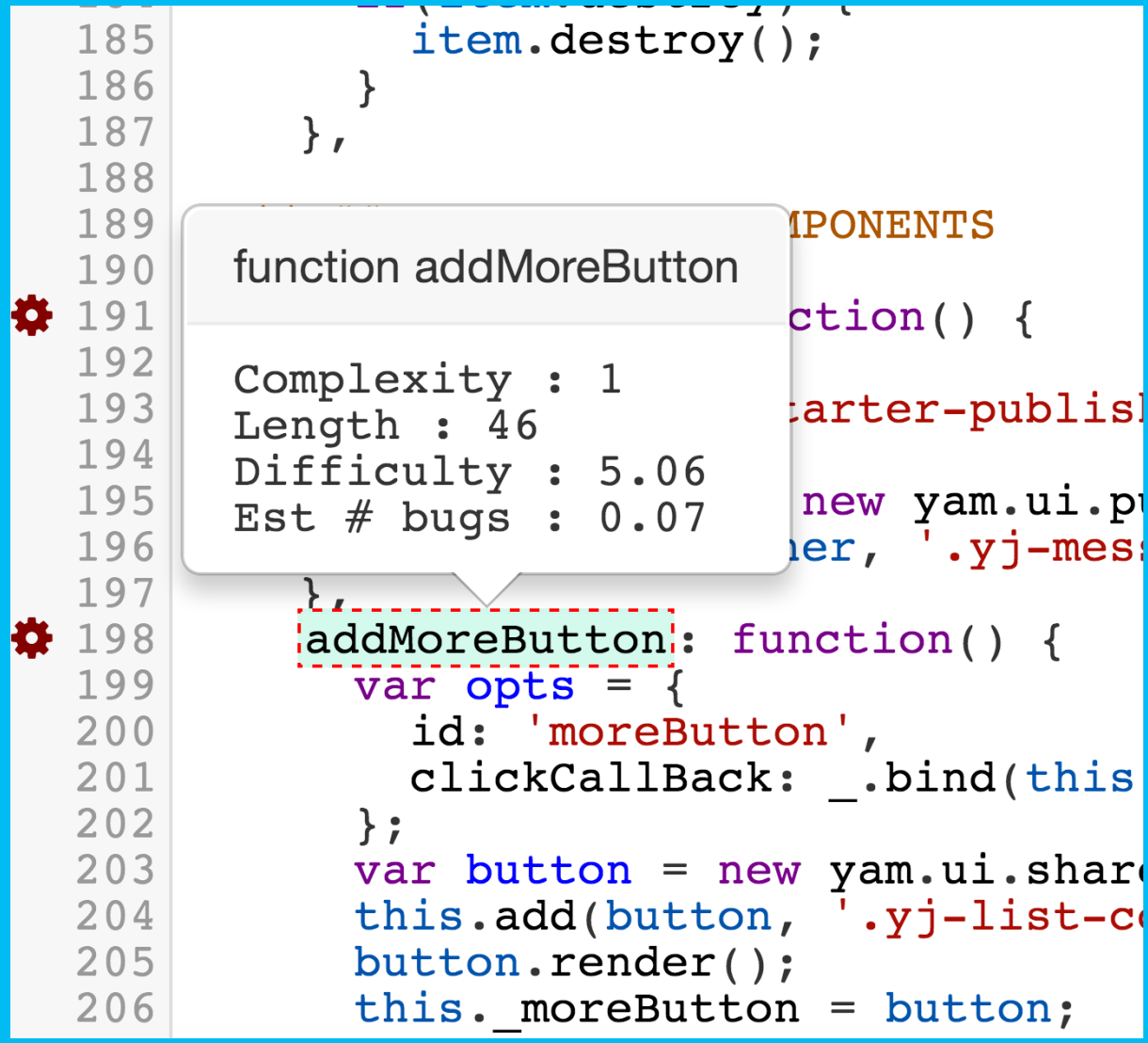
How will others use this component?

Hack up different approaches

Be explicit – don't repeat past mistakes

Plato

http://platojs.org



Keep the team in the loop

Get feedback before starting

Share the spec and answer questions

Everyone should understand the changes

When is the project finished?

Don’t let scope creep

There will be diminishing returns

Remove the feature gate

Avoid the long tail of bugs

Release early and have the team find bugs

The AB test system lets us test on Microsoft

Found features we didn’t account for

Future Works?

Write down the things outside of scope

Will there be a phase 2 and 3?

Give other teams a head start

Before and After

40% less code

Removed asynchronous rendering

Positive feedback from feed projects

One more Thing