EFFECTIVE CODE REVIEW

JUSTIN SPAHR-SUMMERS

@JSPAHRSUMMERS

CODE REVIEW PUTS THE ENGINEERING INTO SOFTWARE ENGINEERING

POOR QUALITY **REVIEWS...**

POOR QUALITY **REVIEWS...**

- **WASTE EVERYONE'S TIME TODAY**

POOR QUALITY **REVIEWS...**

- **WASTE EVERYONE'S TIME TODAY**
- **WASTE EVERYONE'S TIME IN THE FUTURE**

POOR QUALITY **REVIEWS...**

- WASTE EVERYONE'S TIME TODAY
- WASTE EVERYONE'S TIME IN THE FUTURE
- PROVIDE A FALSE SENSE OF SECURITY

GREAT REVIEWS...

GREAT REVIEWS...

- MINIMIZE TECHNICAL DEBT

GREAT REVIEWS...

- MINIMIZE TECHNICAL DEBT
- IMPROVE THE ARCHITECTURE

GREAT REVIEWS...

- MINIMIZE TECHNICAL DEBT
- IMPROVE THE ARCHITECTURE
- SHARE DOMAIN KNOWLEDGE

GREAT REVIEWS...

- MINIMIZE TECHNICAL DEBT
- IMPROVE THE ARCHITECTURE
- SHARE DOMAIN KNOWLEDGE
- PROVIDE TEACHING OPPORTUNITIES

FIRST PRINCIPLES

GOOD CODE REVIEW
STARTS FROM THE SAME PERSPECTIVE AS
WRITING GOOD CODE

IF WE WERE TO WRITE THE BEST VERSION OF THIS, WHAT WOULD IT BE?

CRITIQUE THE CODE

(NOT THE PERSON)

UNBLOCK OTHERS

INCREASE YOUR TEAM'S PRODUCTIVITY

DON'T ASSUME IT'S OBVIOUS

CODE CHANGES AND FEEDBACK BOTH NEED EXPLANATION

THE PROCESS

START AT THE HIGHEST LEVEL, THEN DIVE DEEPER...

START AT THE HIGHEST LEVEL, THEN DIVE DEEPER...

1. **INTENT**

START AT THE HIGHEST LEVEL, THEN DIVE DEEPER...

1. **INTENT**
2. **DESIGN**

START AT THE HIGHEST LEVEL, THEN DIVE DEEPER...

1. **INTENT**
2. **DESIGN**
3. **BEHAVIOR**

1. INTENT

WHAT IS THE GOAL?

IS THE EXPLANATION CLEAR?

mplement `jest-haste-map` instead of `node-haste` #896

Closed **cpojer** wants to merge 10 commits into `facebook:master` from `cpojer:jest-haste-map`

Conversation 93 | Commits 10 | Checks 0 | Files changed 47

cpojer commented on Apr 15, 2016 • edited ▾ Collaborator

This is a new haste map implementation for Jest which is much more scalable than node-haste.

node-haste2 isn't well designed and not scalable for short-lived services like Jest. The startup time of node-haste1 vs. node-haste2 on www is almost the same, both between 6 and 8 seconds which is not acceptable for our engineers. This implementation is attempting to accomplish a much reduced and more scalable startup time. It also has reduced scope – the goal of this is to only build a haste map and provide a way to resolve a one-level deep dependency tree, which is all that Jest really needs from node-haste. `jest-haste-map` can serve as an ideal basis for rewriting node-haste2 (into node-haste3!).

With this, the cold start time (building the entire haste map) is now about 14 seconds on www (that's acceptable) but the incremental invocation is only 2 seconds (**@kyldvs** will love me). I haven't heavily micro-optimized the JavaScript in `packages/jest-haste-map/src/index.js` and there is a bunch of data copying with `HasteResolver.js` – I believe I can get close to 1 second with these optimizations once I'm done. One of the goals I have is to allow tacking on data (list of mocks) to the haste map and serialize the haste map and read that one in directly in the workers instead of keeping two caches.

mplement `jest-haste-map` instead of `node-haste` #896

Closed **cpojer** wants to merge 10 commits into `facebook:master` from `cpojer:jest-haste-map`

Conversation 93 | Commits 10 | Checks 0 | Files changed 47

cpojer commented on Apr 15, 2016 • edited ▾ Collaborator

This is a new haste map implementation for Jest which is much more scalable than node-haste.

node-haste2 isn't well designed and not scalable for short-lived services like Jest. The startup time of node-haste1 vs. node-haste2 on www is almost the same, both between 6 and 8 seconds which is not acceptable for our engineers. This implementation is attempting to accomplish a much reduced and more scalable startup time. It also has reduced scope – the goal of this is to only build a haste map and provide a way to resolve a one-level deep dependency tree, which is all that Jest really needs from node-haste. `jest-haste-map` can serve as an ideal basis for rewriting node-haste2 (into node-haste3!).

With this, the cold start time (building the entire haste map) is now about 14 seconds on www (that's acceptable) but the incremental invocation is only 2 seconds (**@kyldvs** will love me). I haven't heavily micro-optimized the JavaScript in `packages/jest-haste-map/src/index.js` and there is a bunch of data copying with `HasteResolver.js` – I believe I can get close to 1 second with these optimizations once I'm done. One of the goals I have is to allow tacking on data (list of mocks) to the haste map and serialize the haste map and read that one in directly in the workers instead of keeping two caches.

mplement `jest-haste-map` instead of `node-haste` #896

Closed **cpojer** wants to merge 10 commits into `facebook:master` from `cpojer:jest-haste-map`

Conversation 93 | Commits 10 | Checks 0 | Files changed 47

cpojer commented on Apr 15, 2016 • edited ▾ Collaborator

This is a new haste map implementation for Jest which is much more scalable than node-haste.

node-haste2 isn't well designed and not scalable for short-lived services like Jest. The startup time of node-haste1 vs. node-haste2 on www is almost the same, both between 6 and 8 seconds which is not acceptable for our engineers. This implementation is attempting to accomplish a much reduced and more scalable startup time. It also has reduced scope – the goal of this is to only build a haste map and provide a way to resolve a one-level deep dependency tree, which is all that Jest really needs from node-haste. `jest-haste-map` can serve as an ideal basis for rewriting node-haste2 (into node-haste3!).

With this, the cold start time (building the entire haste map) is now about 14 seconds on www (that's acceptable) but the incremental invocation is only 2 seconds (**@kyldvs** will love me). I haven't heavily micro-optimized the JavaScript in `packages/jest-haste-map/src/index.js` and there is a bunch of data copying with `HasteResolver.js` – I believe I can get close to 1 second with these optimizations once I'm done. One of the goals I have is to allow tacking on data (list of mocks) to the haste map and serialize the haste map and read that one in directly in the workers instead of keeping two caches.

DOES IT SUCCEED?

HOW DO YOU KNOW?

nitial version of jest-worker #4497

Merged **cpojer** merged 8 commits into `facebook:master` from `mjesun:jest-parallel` on Oct 4, 2017

Conversation 85 | Commits 8 | Checks 0 | Files changed 14

mjesun commented on Sep 17, 2017 • edited

Contributor

This PR introduces a new module, `jest-worker`, intended to allow heavy task parallelization over multiple workers.

The module has a few advantages over the currently one used both in `jest` and `metro-bundler`:

- 100% `flow`-ified.
- 100% test coverage on it, all statements, methods and branches.
- Slightly faster than the currenly used one.
- Natively provides a `Promise` based interface, which allow us to avoid the extra wrapping layer in order to be used with `async`/`await.
- It only has one single dependency (`merge-stream`), which we could also remove.

Performance test

It can be run by doing `node --expose-gc test.js` under `__performance_tests__`. Note that the percentage improvement shown (~ 10%) applies to 10,000 calls, meaning the performance improvement per single call is negligible. The test implements a `Promise` wrapper over the current implementation, so we can equivalently test both implementations as we use them in real scenarios.

```
------------------------------------------------------------------------
jest-worker: { globalTime: 738, processingTime: 707 }
worker-farm: { globalTime: 885, processingTime: 866 }
------------------------------------------------------------------------
jest-worker: { globalTime: 738, processingTime: 718 }
worker-farm: { globalTime: 865, processingTime: 849 }
------------------------------------------------------------------------
jest-worker: { globalTime: 708, processingTime: 685 }
```

Coverage

File		Statements		Branches		Functions		Lines	
child.js		100%	27/27	100%	20/20	100%	4/4	100%	27/27
index.js		100%	67/67	100%	29/29	100%	11/11	100%	67/67
types.js		100%	5/5	100%	0/0	100%	0/0	100%	5/5
worker.js		100%	44/44	100%	13/13	100%	8/8	100%	44/44

A GOOD SUMMARY SHOULD INCLUDE...

A GOOD SUMMARY SHOULD INCLUDE...

- **BACKGROUND CONTEXT**

A GOOD SUMMARY SHOULD INCLUDE...

- **BACKGROUND CONTEXT**
- **WHAT THE BUG OR FEATURE IS**

A GOOD SUMMARY SHOULD INCLUDE...

- BACKGROUND CONTEXT
- WHAT THE BUG OR FEATURE IS
- WHAT THE CHANGE ACHIEVES

A GOOD SUMMARY SHOULD INCLUDE...

- BACKGROUND CONTEXT
- WHAT THE BUG OR FEATURE IS
- WHAT THE CHANGE ACHIEVES
- HOW THE CHANGE IS TESTED

A GOOD SUMMARY SHOULD INCLUDE...

- BACKGROUND CONTEXT
- WHAT THE BUG OR FEATURE IS
- WHAT THE CHANGE ACHIEVES
- HOW THE CHANGE IS TESTED
- KNOWN LIMITATIONS OR ANYTHING STILL MISSING

A GOOD SUMMARY SHOULD INCLUDE...

- BACKGROUND CONTEXT
- WHAT THE BUG OR FEATURE IS
- WHAT THE CHANGE ACHIEVES
- HOW THE CHANGE IS TESTED
- KNOWN LIMITATIONS OR ANYTHING STILL MISSING
- REQUEST CHANGES IF THE SUMMARY IS INCOMPLETE!

Allow creating an alias for repositories #12000

Merged sergiou87 merged 8 commits into `development` from `repository-alias` 25 days ago

Conversation 12 | Commits 8 | Checks 6 | Files changed 13

sergiou87 commented 26 days ago — Member

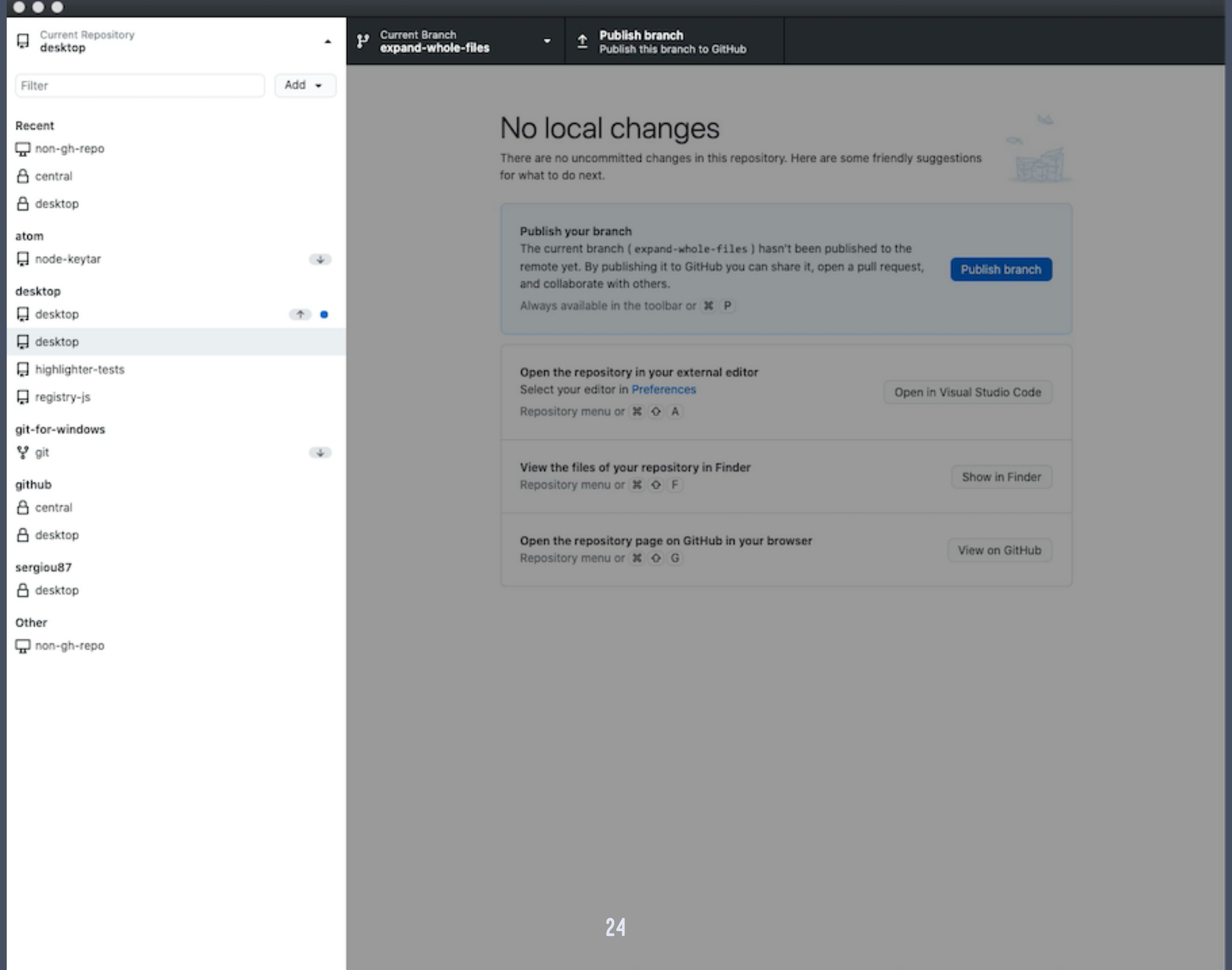
This is part of #7856

Description

This PR adds initial support to create alias for repositories:

- New context menu actions to create, change or remove the alias of a repository from the repository list. Any character is valid.
- Filtering repositories now takes the alias into account.
- The original repository name is still visible by hovering over it, in the tooltip.

Screenshots


Current Repository
desktop
Filter
Add
Recent
non-gh-repo
central
desktop
atom
node-keytar
desktop
desktop
desktop
highlighter-tests
registry-js
git-for-windows
git
github
central
desktop
sergiou87
desktop
Other
non-gh-repo
Current Branch
expand-whole-files
Publish branch
Publish this branch to GitHub
No local changes
There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.
Publish your branch
The current branch (expand-whole-files) hasn't been published to the remote yet. By publishing it to GitHub you can share it, open a pull request, and collaborate with others.
Always available in the toolbar or ⌘ P
Publish branch
Open the repository in your external editor
Select your editor in Preferences
Repository menu or ⌘ ⌥ A
Open in Visual Studio Code
View the files of your repository in Finder
Repository menu or ⌘ ⌥ F
Show in Finder
Open the repository page on GitHub in your browser
Repository menu or ⌘ ⌥ G
View on GitHub

ARE THE CHANGES APPROPRIATE?

2. DESIGN

ASK YOURSELF:
HOW WOULD YOU DO IT?

REVIEW THE ARCHITECTURE

(THE COMPONENTS AND HOW THEY RELATE TO ONE ANOTHER)

REVIEW THE ARCHITECTURE

(THE COMPONENTS AND HOW THEY RELATE TO ONE ANOTHER)

- **DO YOU UNDERSTAND IT WELL ENOUGH TO USE OR EXTEND?**

REVIEW THE ARCHITECTURE

(THE COMPONENTS AND HOW THEY RELATE TO ONE ANOTHER)

- DO YOU UNDERSTAND IT WELL ENOUGH TO USE OR EXTEND?
  - ARE THE ARCHITECTURAL CHOICES JUSTIFIED?

REVIEW THE ARCHITECTURE

(THE COMPONENTS AND HOW THEY RELATE TO ONE ANOTHER)

- DO YOU UNDERSTAND IT WELL ENOUGH TO USE OR EXTEND?
- ARE THE ARCHITECTURAL CHOICES JUSTIFIED?
- WOULD EVERYONE ELSE BE HAPPY TO MAINTAIN THIS?

REVIEW THE API

(THE CONTRACT FOR USING EACH COMPONENT)

REVIEW THE API

(THE CONTRACT FOR USING EACH COMPONENT)

- **IS THE API UNDERSTANDABLE WITHOUT THE PR?**

REVIEW THE API

(THE CONTRACT FOR USING EACH COMPONENT)

- IS THE API UNDERSTANDABLE WITHOUT THE PR?
- DOES THE DOCUMENTATION TEACH THE READER HOW TO USE IT?

REVIEW THE API

(THE CONTRACT FOR USING EACH COMPONENT)

- IS THE API UNDERSTANDABLE WITHOUT THE PR?
- DOES THE DOCUMENTATION TEACH THE READER HOW TO USE IT?
- IS THE API CONVENTIONAL?

IS THE DESIGN GOOD?

YAGNI

YOU AREN'T GONNA NEED IT

SIMPLE VS. EASY

PIT OF SUCCESS

MAKE THE RIGHT THINGS EASY
& THE WRONG THINGS POSSIBLE

SOLID PRINCIPLES

SOLID PRINCIPLES

- **SINGLE-RESPONSIBILITY PRINCIPLE**

EACH THING SHOULD HAVE ONLY ONE RESPONSIBILITY

SOLID PRINCIPLES

- **SINGLE-RESPONSIBILITY PRINCIPLE**

  EACH THING SHOULD HAVE ONLY ONE RESPONSIBILITY

- **OPEN-CLOSED PRINCIPLE**

  BEHAVIOR SHOULD BE EXTENSIBLE WITHOUT MODIFYING CODE

SOLID PRINCIPLES

- **SINGLE-RESPONSIBILITY PRINCIPLE**
  EACH THING SHOULD HAVE ONLY ONE RESPONSIBILITY
- **OPEN-CLOSED PRINCIPLE**
  BEHAVIOR SHOULD BE EXTENSIBLE WITHOUT MODIFYING CODE
- **LISKOV SUBSTITUTION PRINCIPLE**
  TYPES SHOULD BE REPLACEABLE WITH SUBTYPES

SOLID PRINCIPLES

- **SINGLE-RESPONSIBILITY PRINCIPLE**
  EACH THING SHOULD HAVE ONLY ONE RESPONSIBILITY
- **OPEN-CLOSED PRINCIPLE**
  BEHAVIOR SHOULD BE EXTENSIBLE WITHOUT MODIFYING CODE
- **LISKOV SUBSTITUTION PRINCIPLE**
  TYPES SHOULD BE REPLACEABLE WITH SUBTYPES

SOLID PRINCIPLES

- **SINGLE-RESPONSIBILITY PRINCIPLE**
  EACH THING SHOULD HAVE ONLY ONE RESPONSIBILITY
- **OPEN-CLOSED PRINCIPLE**
  BEHAVIOR SHOULD BE EXTENSIBLE WITHOUT MODIFYING CODE
- **LISKOV SUBSTITUTION PRINCIPLE**
  TYPES SHOULD BE REPLACEABLE WITH SUBTYPES
- **INTERFACE SEGREGATION PRINCIPLE**
  MANY SPECIFIC INTERFACES ARE BETTER THAN ONE ÜBER-INTERFACE

SOLID PRINCIPLES

- **SINGLE-RESPONSIBILITY PRINCIPLE**
  EACH THING SHOULD HAVE ONLY ONE RESPONSIBILITY
- **OPEN-CLOSED PRINCIPLE**
  BEHAVIOR SHOULD BE EXTENSIBLE WITHOUT MODIFYING CODE
- **LISKOV SUBSTITUTION PRINCIPLE**
  TYPES SHOULD BE REPLACEABLE WITH SUBTYPES
- **INTERFACE SEGREGATION PRINCIPLE**
  MANY SPECIFIC INTERFACES ARE BETTER THAN ONE ÜBER-INTERFACE
- **DEPENDENCY INVERSION PRINCIPLE**
  DEPEND UPON ABSTRACTIONS, NOT CONCRETE IMPLEMENTATIONS

3. BEHAVIOR

REVIEW THE TESTS

REVIEW THE TESTS

- TESTS SHOULD BE DOCUMENTATION

REVIEW THE TESTS

- TESTS SHOULD BE DOCUMENTATION
- TESTS SHOULD PROTECT AGAINST REGRESSIONS

REVIEW THE TESTS

- TESTS SHOULD BE DOCUMENTATION
- TESTS SHOULD PROTECT AGAINST REGRESSIONS
- TESTS SHOULD VALIDATE THE API CONTRACT

REVIEW THE TESTS

- TESTS SHOULD BE DOCUMENTATION
- TESTS SHOULD PROTECT AGAINST REGRESSIONS
- TESTS SHOULD VALIDATE THE API CONTRACT
- TESTS NEED TO BE UNDERSTANDABLE

REVIEW THE TESTS

- TESTS SHOULD BE DOCUMENTATION
- TESTS SHOULD PROTECT AGAINST REGRESSIONS
- TESTS SHOULD VALIDATE THE API CONTRACT
- TESTS NEED TO BE UNDERSTANDABLE
- ARE THERE MISSING TESTS?

REVIEW THE TESTS

- TESTS SHOULD BE DOCUMENTATION
- TESTS SHOULD PROTECT AGAINST REGRESSIONS
- TESTS SHOULD VALIDATE THE API CONTRACT
- TESTS NEED TO BE UNDERSTANDABLE
- ARE THERE MISSING TESTS?
- YOU CAN REQUEST CHANGES!

REVIEW THE IMPLEMENTATION

REVIEW THE IMPLEMENTATION

- THIS IS THE LEAST IMPORTANT PART TO REVIEW!

REVIEW THE IMPLEMENTATION

- THIS IS THE LEAST IMPORTANT PART TO REVIEW!
  - WOULD YOU BE ABLE TO DEBUG THIS CODE?

REVIEW THE IMPLEMENTATION

- THIS IS THE LEAST IMPORTANT PART TO REVIEW!
  - WOULD YOU BE ABLE TO DEBUG THIS CODE?
    - DRY: DON'T REPEAT YOURSELF

REVIEW THE IMPLEMENTATION

- THIS IS THE LEAST IMPORTANT PART TO REVIEW!
  - WOULD YOU BE ABLE TO DEBUG THIS CODE?
    - DRY: DON'T REPEAT YOURSELF
      - DON'T REINVENT THE WHEEL

REVIEW THE IMPLEMENTATION

- THIS IS THE LEAST IMPORTANT PART TO REVIEW!
  - WOULD YOU BE ABLE TO DEBUG THIS CODE?
    - DRY: DON'T REPEAT YOURSELF
      - DON'T REINVENT THE WHEEL
  - DON'T IGNORE LINTERS AND WARNINGS

REVIEW THE IMPLEMENTATION

- THIS IS THE LEAST IMPORTANT PART TO REVIEW!
  - WOULD YOU BE ABLE TO DEBUG THIS CODE?
    - DRY: DON'T REPEAT YOURSELF
      - DON'T REINVENT THE WHEEL
    - DON'T IGNORE LINTERS AND WARNINGS
- IF IT LOOKS CONVOLUTED, IT'S PROBABLY WRONG

BUT REMEMBER...

BUT REMEMBER...

1. **INTENT**

BUT REMEMBER...

1. **INTENT**
2. **DESIGN**

BUT REMEMBER...

1. **INTENT**
2. **DESIGN**
3. **BEHAVIOR**

OTHER PROTIPS™

GIVE EFFECTIVE FEEDBACK

GIVE EFFECTIVE FEEDBACK

> ALWAYS REQUEST CHANGES OR ACCEPT

GIVE EFFECTIVE FEEDBACK

- ALWAYS REQUEST CHANGES OR ACCEPT
- BE PRAGMATIC FOR URGENT CHANGES

GIVE EFFECTIVE FEEDBACK

- ALWAYS REQUEST CHANGES OR ACCEPT
  - BE PRAGMATIC FOR URGENT CHANGES
- IS EACH STAGE OF REVIEW IMPORTANT RIGHT NOW?

GIVE EFFECTIVE FEEDBACK

- ALWAYS REQUEST CHANGES OR ACCEPT
- BE PRAGMATIC FOR URGENT CHANGES
- IS EACH STAGE OF REVIEW IMPORTANT RIGHT NOW?
- SOLICIT SECOND OPINIONS

GIVE EFFECTIVE FEEDBACK

- ALWAYS REQUEST CHANGES OR ACCEPT
- BE PRAGMATIC FOR URGENT CHANGES
- IS EACH STAGE OF REVIEW IMPORTANT RIGHT NOW?
- SOLICIT SECOND OPINIONS
- PRIORITIZE YOUR FEEDBACK

GIVE EFFECTIVE FEEDBACK

- ALWAYS REQUEST CHANGES OR ACCEPT
- BE PRAGMATIC FOR URGENT CHANGES
- IS EACH STAGE OF REVIEW IMPORTANT RIGHT NOW?
- SOLICIT SECOND OPINIONS
- PRIORITIZE YOUR FEEDBACK
- PROVIDE CONCRETE SUGGESTIONS

TELL A STORY WITH YOUR COMMITS

TELL A STORY WITH YOUR COMMITS

- EACH COMMIT SHOULD BUILD LOGICALLY UPON THE PREVIOUS

TELL A STORY WITH YOUR COMMITS

- EACH COMMIT SHOULD BUILD LOGICALLY UPON THE PREVIOUS
  - CLEAN UP AFTER YOURSELF:

```
git rebase -i
hg histedit
```


TELL A STORY WITH YOUR COMMITS

- EACH COMMIT SHOULD BUILD LOGICALLY UPON THE PREVIOUS
  - CLEAN UP AFTER YOURSELF:

```
git rebase -i
hg histedit
```

  - STACK DEPENDENT CHANGES

QUESTIONS?

SLIDES AND NOTES ARE AVAILABLE AT:
GITHUB.COM/JSPAHRSUMMERS/EFFECTIVE-CODE-REVIEW

THANKS TO LIGHTRICKS AND BARAK YORESH FOR INVITING ME TO SPEAK!

