Code Evolution at Scale

Jake Lane

Jake Lane

Senior Software Engineer @ Atlassian

Agenda

- Problem space
- Approaches
- Risk appetite

ALVIN AND THE CHIPMUNKS™ THE SQUEAKQUEL

jQuery
write less, do more.

left-pad

Moment.js

mootools

{{current year}}

How do we migrate commonly used code to something new?

Approaches

- Manual
- Enforcing code standards
- Transforms

Manual

- Refers to manually updating code
- Human error is a big factor
    - Code review helps but is impractical at scale
- Heavy reliance on validation tools

Manual

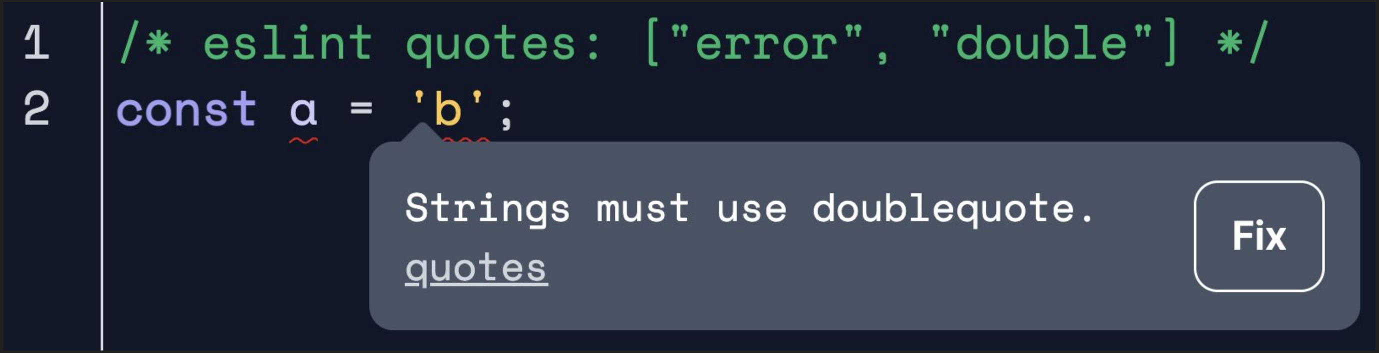
- Easy to do
- Fast to get started
- In the general case, **very slow and risky!**
- May be quicker than automation for smaller tasks

Enforcing code standards

- Use tooling to point people towards the code you want
- Discourage usage of patterns you don't want
- Eventually, ban new usages all together

ESLint

- Static code analysis tool for **identifying bad patterns** in JavaScript
- Provides feedback to the developer
- Can also provide **auto-fixing**



```
/* eslint quotes: ["error", "double"] */
const a = 'b';
```

Strings must use doublequote.
quotes

Using ESLint for code evolution

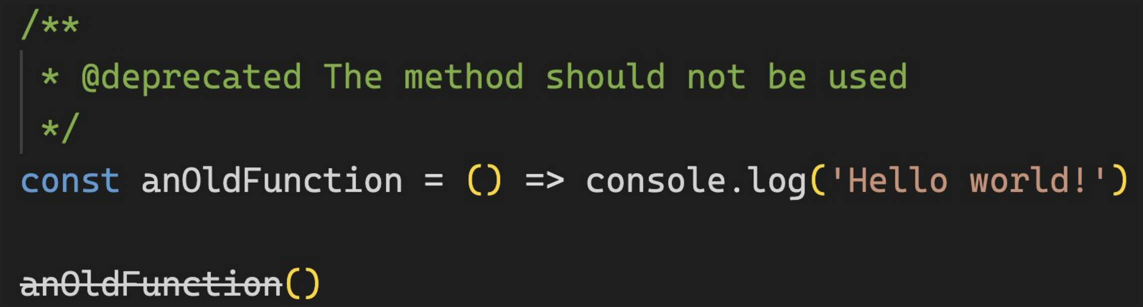
- Use a modern ruleset (like `eslint:recommended`)
- Restrict imports you don't want
- Use more detailed plugins where possible for guidance
  - eslint-plugin-you-dont-need-momentjs
  - eslint-plugin-jquery
- Write your own plugins!

ESLint

```
// .eslintrc.json

{

  "no-restricted-imports":
      ["error", "moment"]

}
```

Types

- Using a Type system like TypeScript or Flow can help you reduce usages of deprecated code
- Mark old code as @deprecated so devs don’t accidentally use it
- You can detect and ban certain things with conditional types and the never type



```
/**
 * @deprecated The method should not be used
 */
const anOldFunction = () => console.log('Hello world!')

~~anOldFunction~~()
```

Caveat with using types

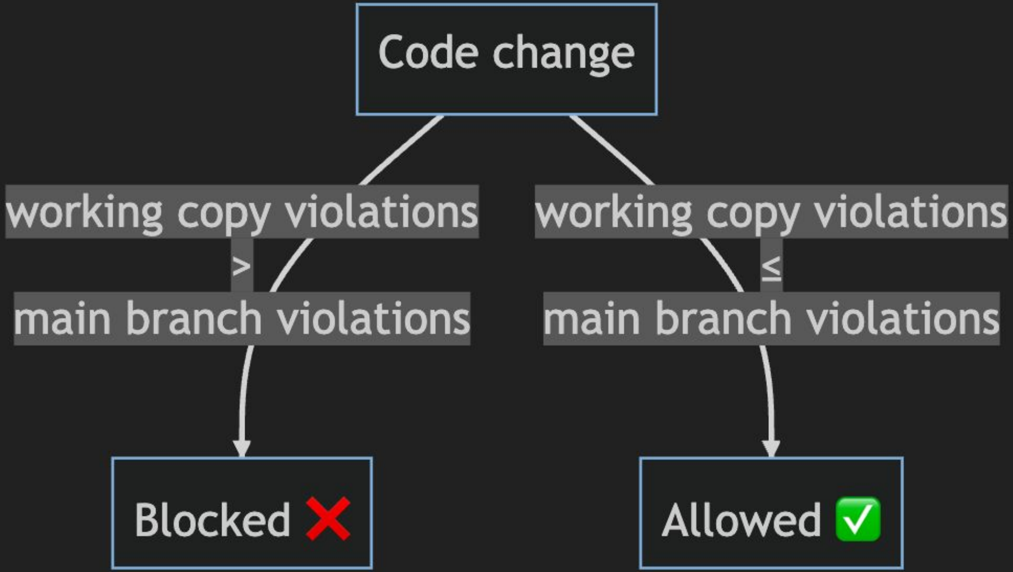
- Be careful with type coverage!
  - `never` will be ignored if the type is any
- `// @ts-expect-error` causes any types
- Untyped packages also cause any types
- Very impactful to improve your type coverage

```
type IfAny<T, Y, N> = 0 extends
(1 & T) ? Y : N;
```

```
<P,}>(component: P extends {
__BANNED_TYPE?: true} ? never :
React.ComponentType<P>):
React.ComponentType<P>
```

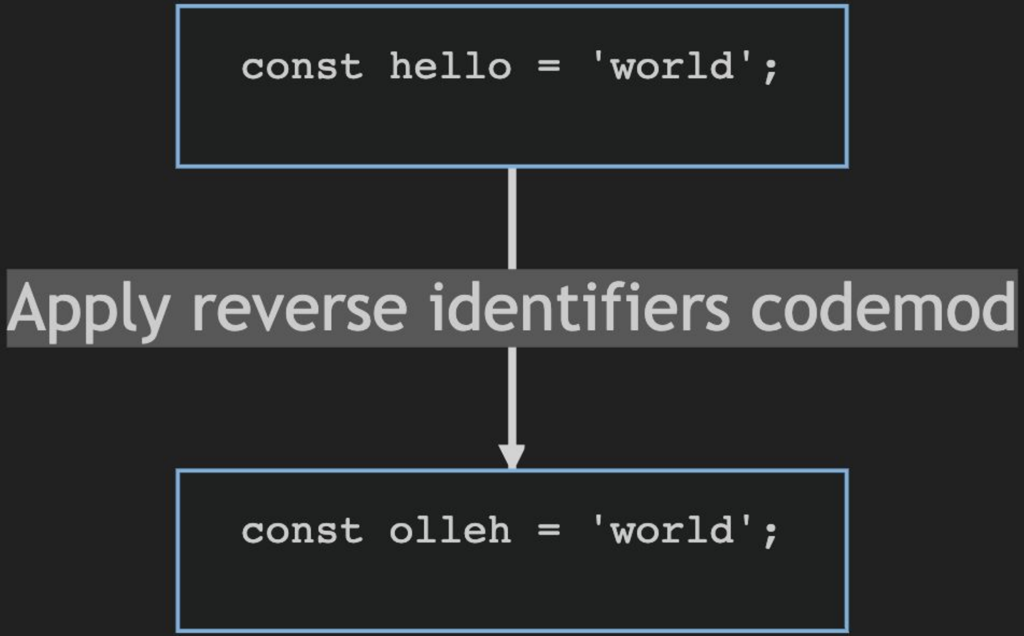
Ratcheting

- **Do not allow any new usages of a pattern**
- The usage count can only go down or stay the same
- Beneficial to make this fast and simple
  - grep works well
- Too complicated to grep?
  `grep '// eslint-disable …'`


Code change
working copy violations > main branch violations
working copy violations ≤ main branch violations
Blocked ❌
Allowed ✅

Transforms

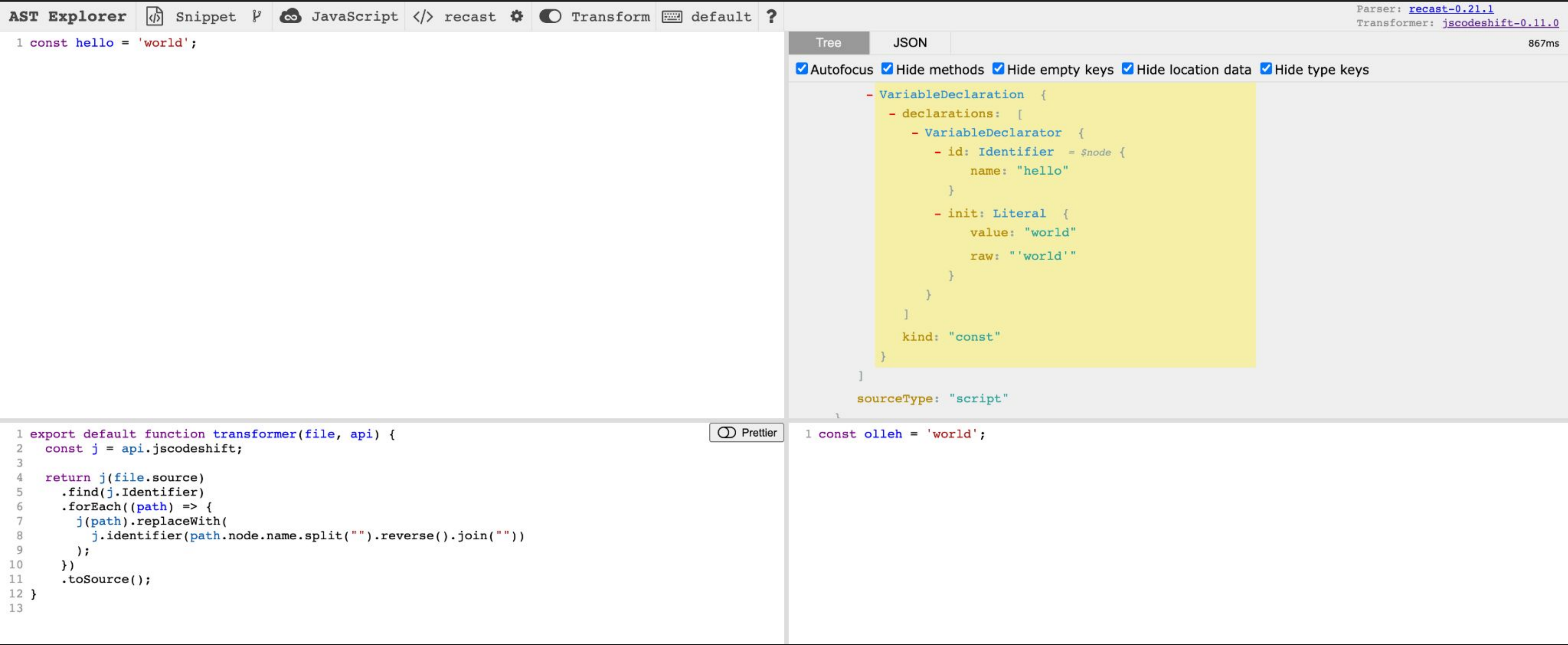
- Transforms or codemods are used to change code from one state to another
- facebook/jscodeshift is a great tool for this
- Uses Abstract Syntax Trees as the data structure



```
const hello = 'world';
```

Apply reverse identifiers codemod

```
const olleh = 'world';
```



```
const hello = 'world';
```

```
- VariableDeclaration {
  - declarations: [
    - VariableDeclarator {
      - id: Identifier = $node {
          name: "hello"
        }
      - init: Literal {
          value: "world"
          raw: "'world'"
        }
      }
    ]
    kind: "const"
  }
]
sourceType: "script"
```

```
export default function transformer(file, api) {
  const j = api.jscodeshift;

  return j(file.source)
    .find(j.Identifier)
    .forEach((path) => {
      j(path).replaceWith(
        j.identifier(path.node.name.split("").reverse().join(""))
      );
    })
    .toSource();
}
```

```
const olleh = 'world';
```

https://astexplorer.net/

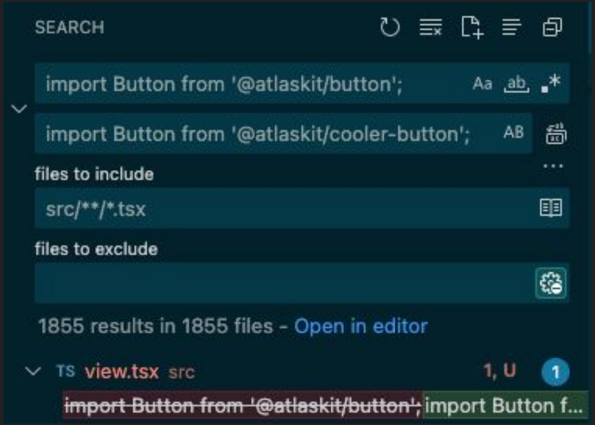
Transforms

- Pretty easy!
- Flexible
  - Don’t have to care about formatting
- Repeatable
  - Easy to rerun
  - Merge conflicts don’t matter
  - Can be part of developer tooling

https://www.hlj.com/transformers-sports-label-megatron-feat-nike-free-tkt76003

Find and replace

- ASTs are cool but not always needed
- Simple find and replace works well!



SEARCH

import Button from '@atlaskit/button';

import Button from '@atlaskit/cooler-button';

files to include

src/**/*.tsx

files to exclude

1855 results in 1855 files - Open in editor

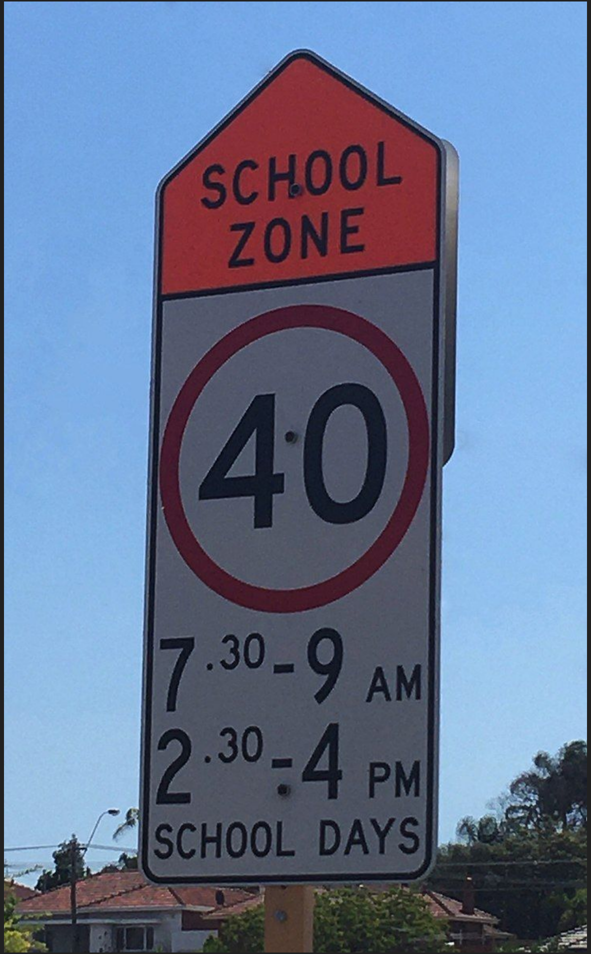
TS view.tsx src 1, U 1

~~import Button from '@atlaskit/button';~~ import Button f...

```
git grep -l "import Button from '@atlaskit\\/button';"
| xargs sed -i '' "s/import Button from
'@atlaskit\\/button';/import Button from
'@atlaskit\\/cooler-button';/g"
```

Risk appetite

Real world example



Real world example

Major three factors of risk

Risk of tests missing errors

- Confidence in your test suite
  - Confidence in visual changes (VR tests)
  - Confidence in functional changes (integration tests)
- Comes down to stuff you’ve decided is correct

Risk of incorrect changes

- What if your choices are incorrect?
- Confidence in your changes being equivalent/correct
- For example:
  - Moving to a new modal library
    - What about mobile?
    - What about keyboard shortcuts?

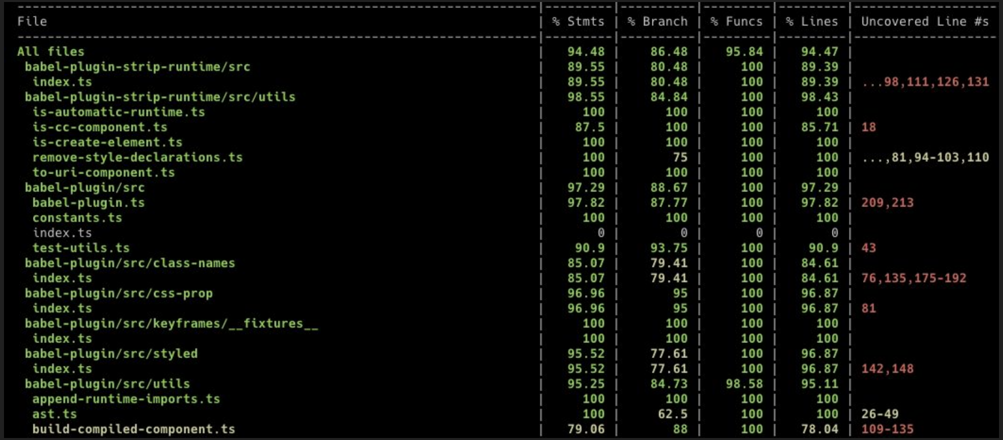
Risk of impact to customers

- Time to Recovery
  - How long does it take for you to resolve the issue?
- Blast radius
  - How many of your customers will be affected?
- Measurement
  - Did you introduce a performance regression?

What can we do to reduce risk?

Reducing risk of tests missing errors

- Improve test coverage
  - Unit tests
  - Integration tests
  - VR tests
- Only migrate what has coverage



```
------------------------------------------------------|---------|----------|---------|---------|-------------------
File                                                  | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
------------------------------------------------------|---------|----------|---------|---------|-------------------
All files                                             |   94.48 |    86.48 |   95.84 |   94.47 |
 babel-plugin-strip-runtime/src                       |   89.55 |    80.48 |     100 |   89.39 |
  index.ts                                            |   89.55 |    80.48 |     100 |   89.39 | ...98,111,126,131
 babel-plugin-strip-runtime/src/utils                 |   98.55 |    84.84 |     100 |   98.43 |
  is-automatic-runtime.ts                             |     100 |      100 |     100 |     100 |
  is-cc-component.ts                                  |    87.5 |      100 |     100 |   85.71 | 18
  is-create-element.ts                                |     100 |      100 |     100 |     100 |
  remove-style-declarations.ts                        |     100 |       75 |     100 |     100 | ...,81,94-103,110
  to-uri-component.ts                                 |     100 |      100 |     100 |     100 |
 babel-plugin/src                                     |   97.29 |    88.67 |     100 |   97.29 |
  babel-plugin.ts                                     |   97.82 |    87.77 |     100 |   97.82 | 209,213
  constants.ts                                        |     100 |      100 |     100 |     100 |
  index.ts                                            |       0 |        0 |       0 |       0 |
  test-utils.ts                                       |    90.9 |    93.75 |     100 |    90.9 | 43
 babel-plugin/src/class-names                         |   85.07 |    79.41 |     100 |   84.61 |
  index.ts                                            |   85.07 |    79.41 |     100 |   84.61 | 76,135,175-192
 babel-plugin/src/css-prop                            |   96.96 |       95 |     100 |   96.87 |
  index.ts                                            |   96.96 |       95 |     100 |   96.87 | 81
 babel-plugin/src/keyframes/__fixtures__              |     100 |      100 |     100 |     100 |
  index.ts                                            |     100 |      100 |     100 |     100 |
 babel-plugin/src/styled                              |   95.52 |    77.61 |     100 |   96.87 |
  index.ts                                            |   95.52 |    77.61 |     100 |   96.87 | 142,148
 babel-plugin/src/utils                               |   95.25 |    84.73 |   98.58 |   95.11 |
  append-runtime-imports.ts                           |     100 |      100 |     100 |     100 |
  ast.ts                                              |     100 |     62.5 |     100 |     100 | 26-49
  build-compiled-component.ts                         |   79.06 |       88 |     100 |   78.04 | 109-135
```

Reducing the risk of incorrect changes

- Manual testing!
  - You don’t know what’s different unless you try it like a user

Reducing the risk of impact to customers

- Feature flagging
  - Instant recovery when toggled
- Rollback strategies
  - Have a runbook
  - Be ready to hot fix
- Have monitoring
  - Set up alerts for metrics you have

Opsgenie

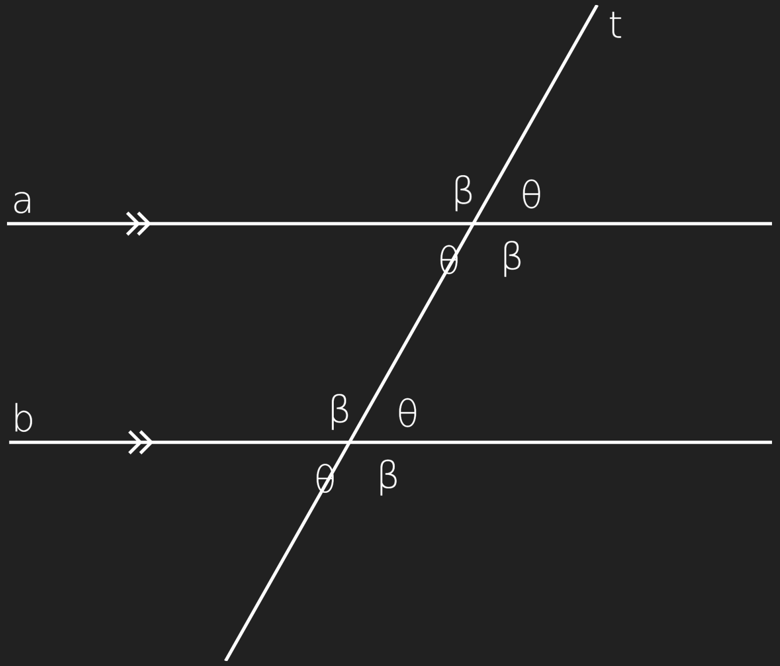
These are all very slow

How do we speed this up?

- Can we make the change without making the behaviour different?
- The easiest thing to release is a release with no changes
- We can build axioms/assumptions

What’s an axiom?

- **a statement that is taken to be true**
- We have to make these to solve problems all the time

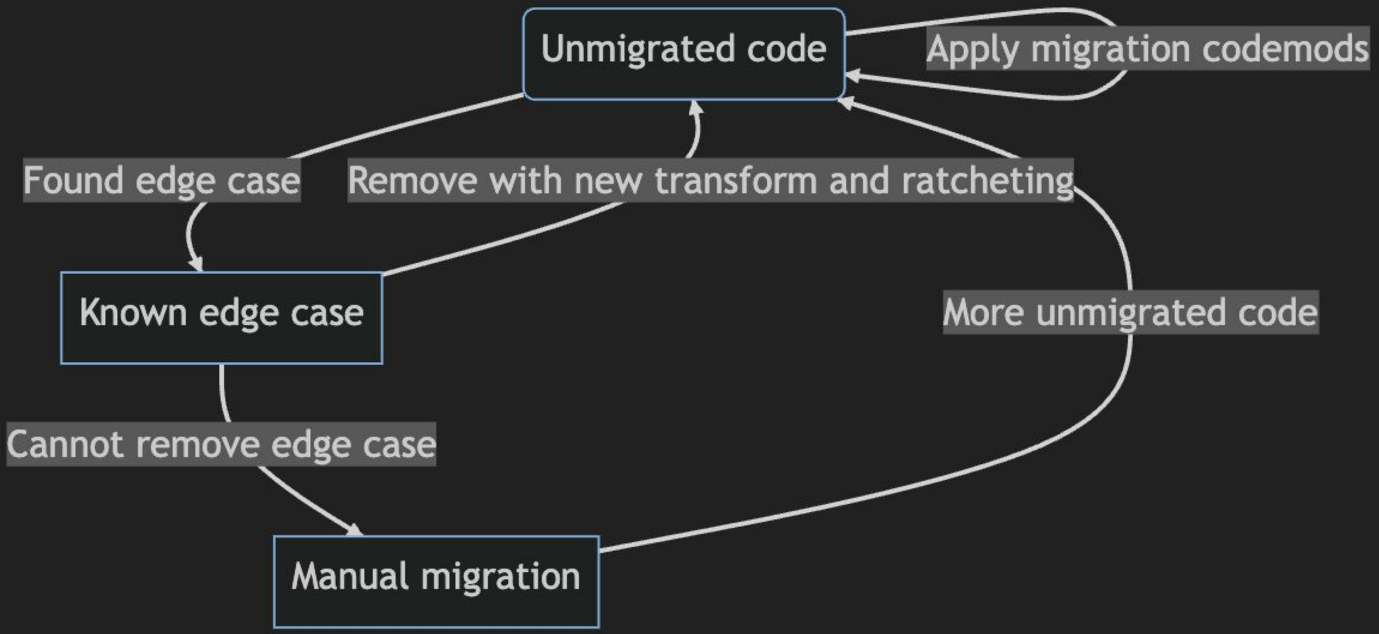


A and B will never intersect
https://commons.wikimedia.org/wiki/File:Parallel_transversal.svg

Break down the problem

- If we build axioms, we can release with much more confidence
- You can test your axioms and release them incrementally

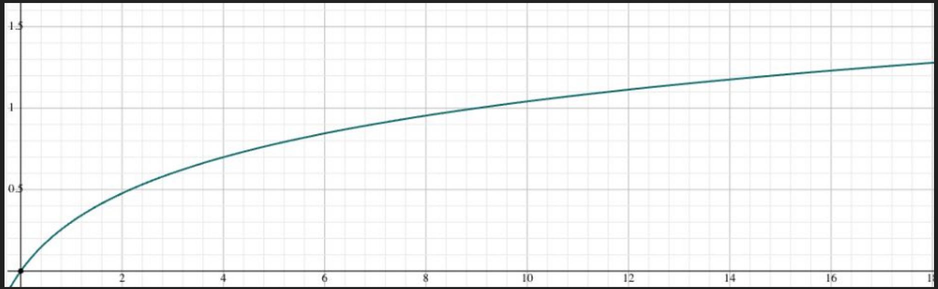
```
// Our original code
console.log("Hello");

// is the same as
sayHello();

// but NOT the same as
console.log("Hello world!");
```



Iterating on the codebase

A good approach for large changes

- Make changes in multiple passes
- The first pass should be the largest but the least complex
- The last pass should be the most complex but should be quite small
- If you encounter something more complex than expected, scope it out to the next batch



Scope creep

Scope creep

- Try not to change things unrelated to your goal
  - Don’t lose the information, put it in your backlog with sufficient detail
- If something is taking up a lot of your time
  - Drop it and come back after the easier stuff

Tools that make large changes easier

- Custom git merge driver (e.g. run a codemod on merge conflicts)
- PR generation tool
- Automated dependency updates (Renovate etc)
- Codemod CLI which targets code for you

Thank you!

linkedin.com/in/jakewlane

github.com/JakeLane