Continuous Accessibility

AMBITIOUS FOR ALL!

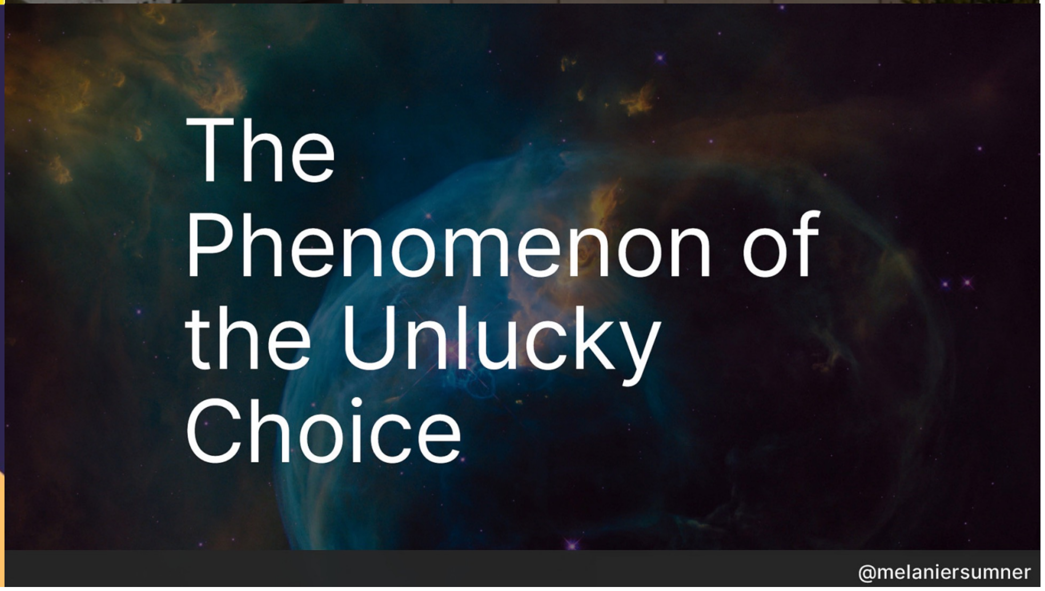
@MELANIERSUMNER

DON'T BREAK THE WEB

@melaniersumner

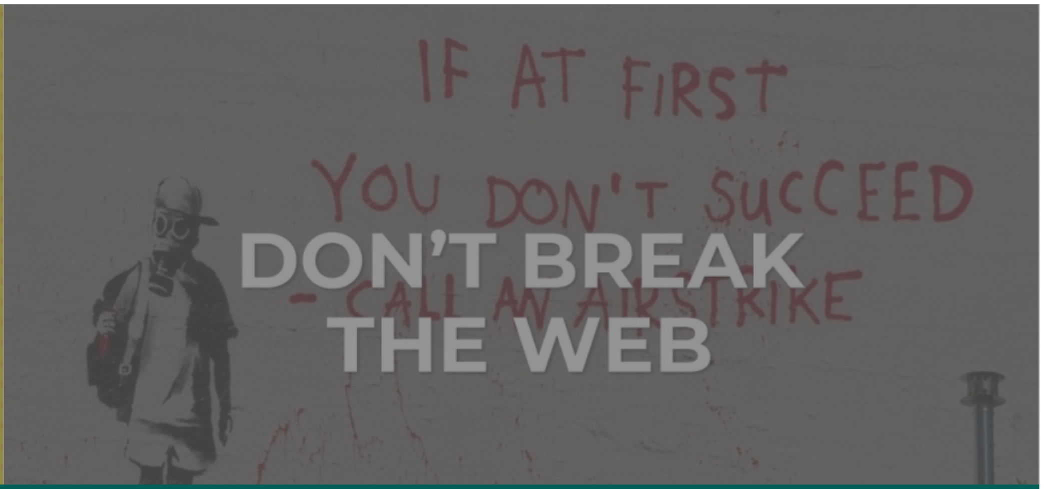
SHIFT LEFT

Melanie Sumner

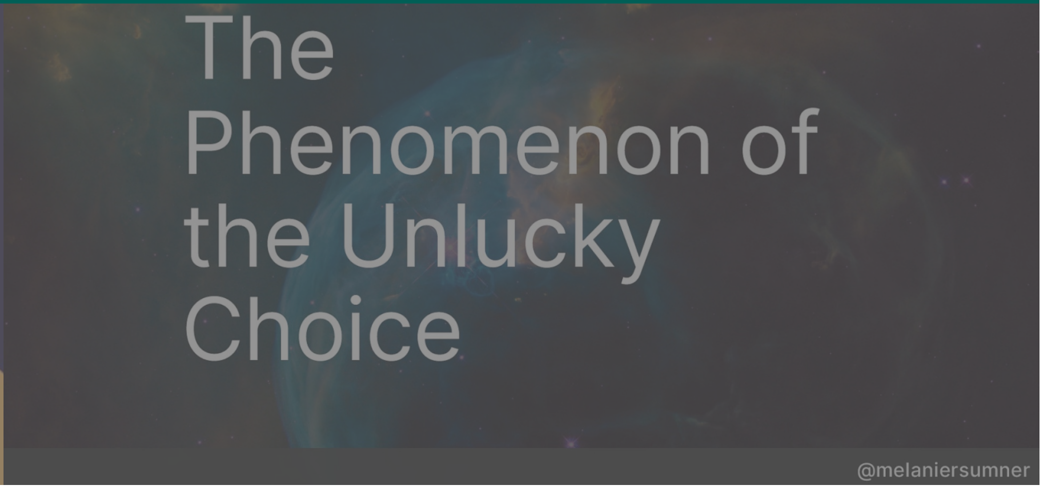


The Phenomenon of the Unlucky Choice

@melaniersumner



https://noti.st/melsumner



@melaniersumner

Accessibility Primer

- Digital accessibility for web-base sites and applications
- Accessibility -> A11y
- Web Content Accessibility Guidelines (WCAG)
- Assistive Technology

Continuous Accessibility

WHY DON'T WE JUMP RIGHT IN?
#SCHITTSCREEK
Pop TV

Imagine…

- Greater confidence in the accessibility of your code.

Imagine…

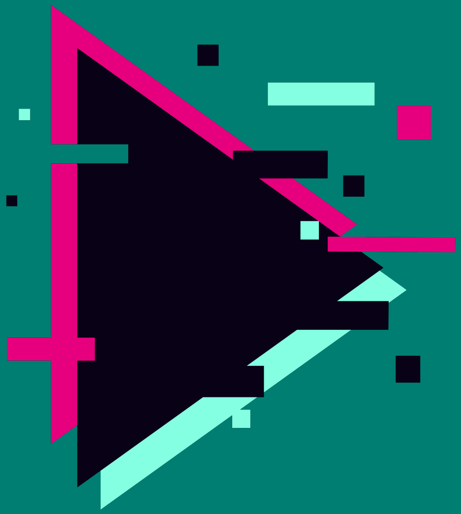
- Greater confidence in the accessibility of your code.
- More easily delivering accessible experiences at scale.

Imagine…

- Greater confidence in the accessibility of your code.
- More easily delivering accessible experiences at scale.
- No fear.

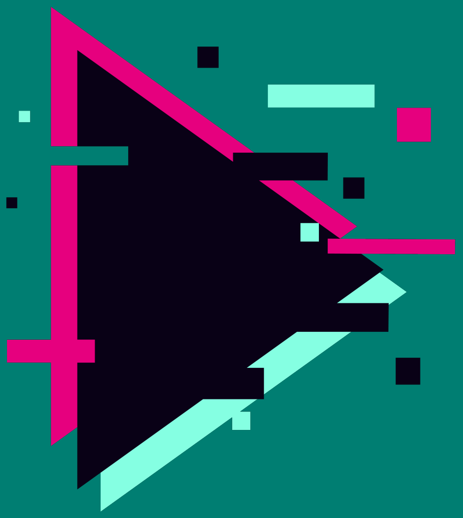
Continuous Delivery

Continuous
Delivery



Continuous
Integration

Continuous Delivery



Continuous Integration

Continuous Accessibility

Principles of Continuous Software Engineering

- Build quality in
- Work in small batches
- Let computers perform repetitive tasks so people can solve hard problems
- Relentlessly pursue improvement
- Everyone is responsible

Principles of Continuous Software Engineering

- Build quality in
- Work in small batches
- Let computers perform repetitive tasks so people can solve hard problems
- Relentlessly pursue improvement
- Everyone is responsible

Principles of Continuous Software Engineering

- Build quality in
- Work in small batches
- Let computers perform repetitive tasks so people can solve hard problems
- Relentlessly pursue improvement
- Everyone is responsible

Principles of Continuous Software Engineering

- Build quality in
- Work in small batches
- Let computers perform repetitive tasks so people can solve hard problems
- Relentlessly pursue improvement
- Everyone is responsible

Principles of Continuous Software Engineering

- Build quality in
- Work in small batches
- Let computers perform repetitive tasks so people can solve hard problems
- Relentlessly pursue improvement
- Everyone is responsible

In this talk, you will learn how to strategically think about your codebase from a continuous accessibility perspective.

Strategy

Strategy

Plan for the code we already have

Strategy

Plan for the code we already have

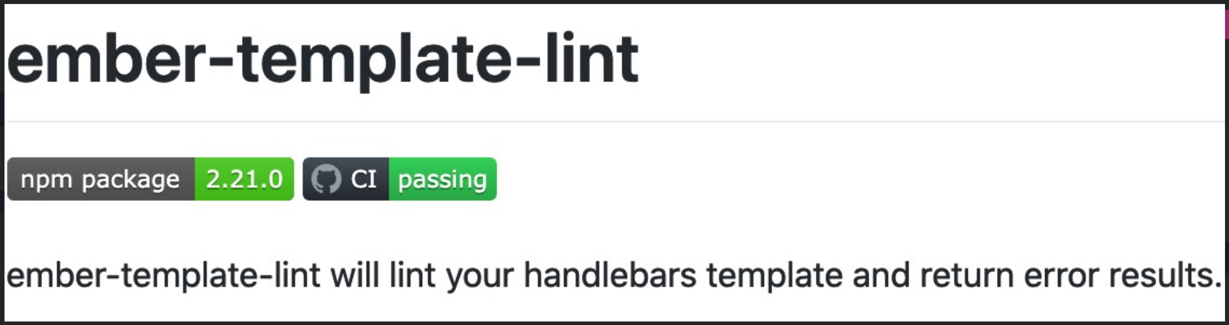
Plan for the code we will create

Strategy

Plan for the code we already have

Plan for the code we will create

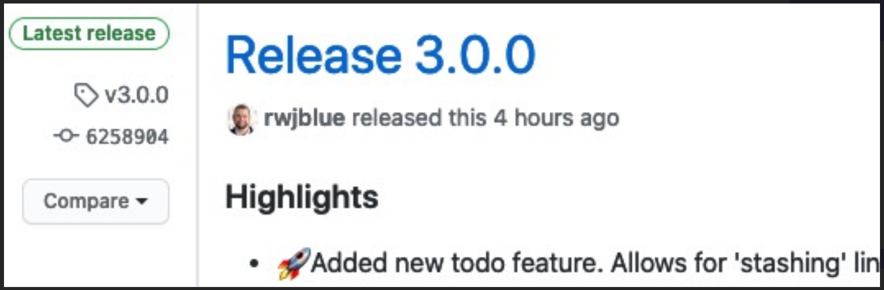
Plan to measure our progress



ember-template-lint

npm package 2.21.0 | CI passing

ember-template-lint will lint your handlebars template and return error results.



Latest release

v3.0.0

6258904

Compare

Release 3.0.0

rwjblue released this 4 hours ago

Highlights

- 🚀Added new todo feature. Allows for 'stashing' lin

- Rolling out this new feature to our apps
- Automated rollout tooling
- Configuration trackers

The Code We Have

- How long has that code been around?
- How do we plan for library upgrades?
- How difficult is it to give developers new tools?

Size

Scale

Breaking Changes

Devs

Priorities

User-Reported Issues

Audit-Reported Issues

Automation Reported Issues

Automation

Audits

Users

Our Future Code

Principles of Continuous Software Engineering

- Build quality in
- Work in small batches
- Let computers perform repetitive tasks so people can solve hard problems
- Relentlessly pursue improvement
- Everyone is responsible

A11y Automation Now

- Lighthouse
- Microsoft Accessibility Insights
- ember-a11y-testing
- axe-core

(Dynamic Analysis)

A11y Automation Now

- ember-template-lint

(Static Analysis)

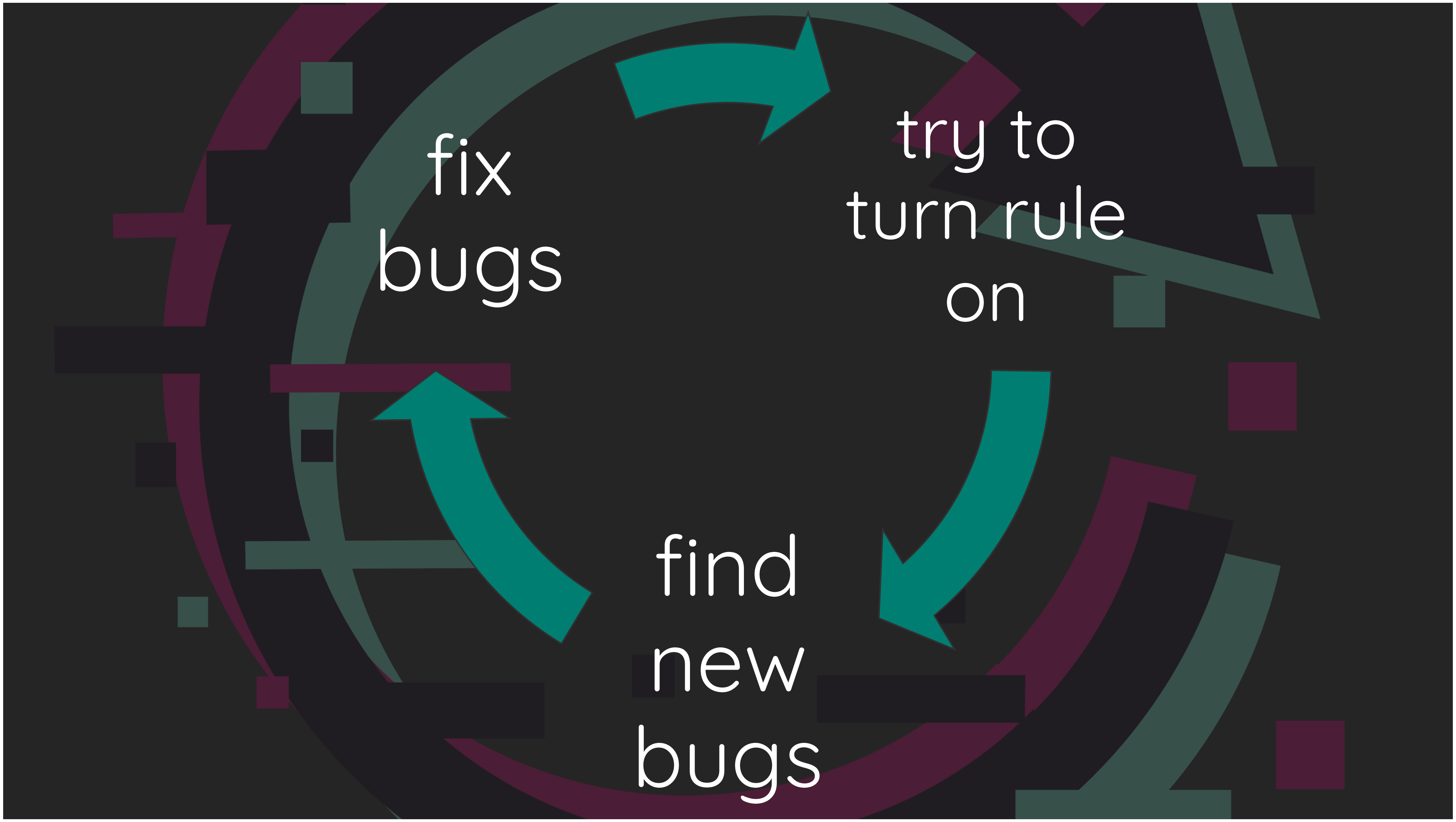
Ember Template Lint

- Real-time feedback
- Custom rules
- Automatic fixes
- Shareable configs

(Static Analysis)

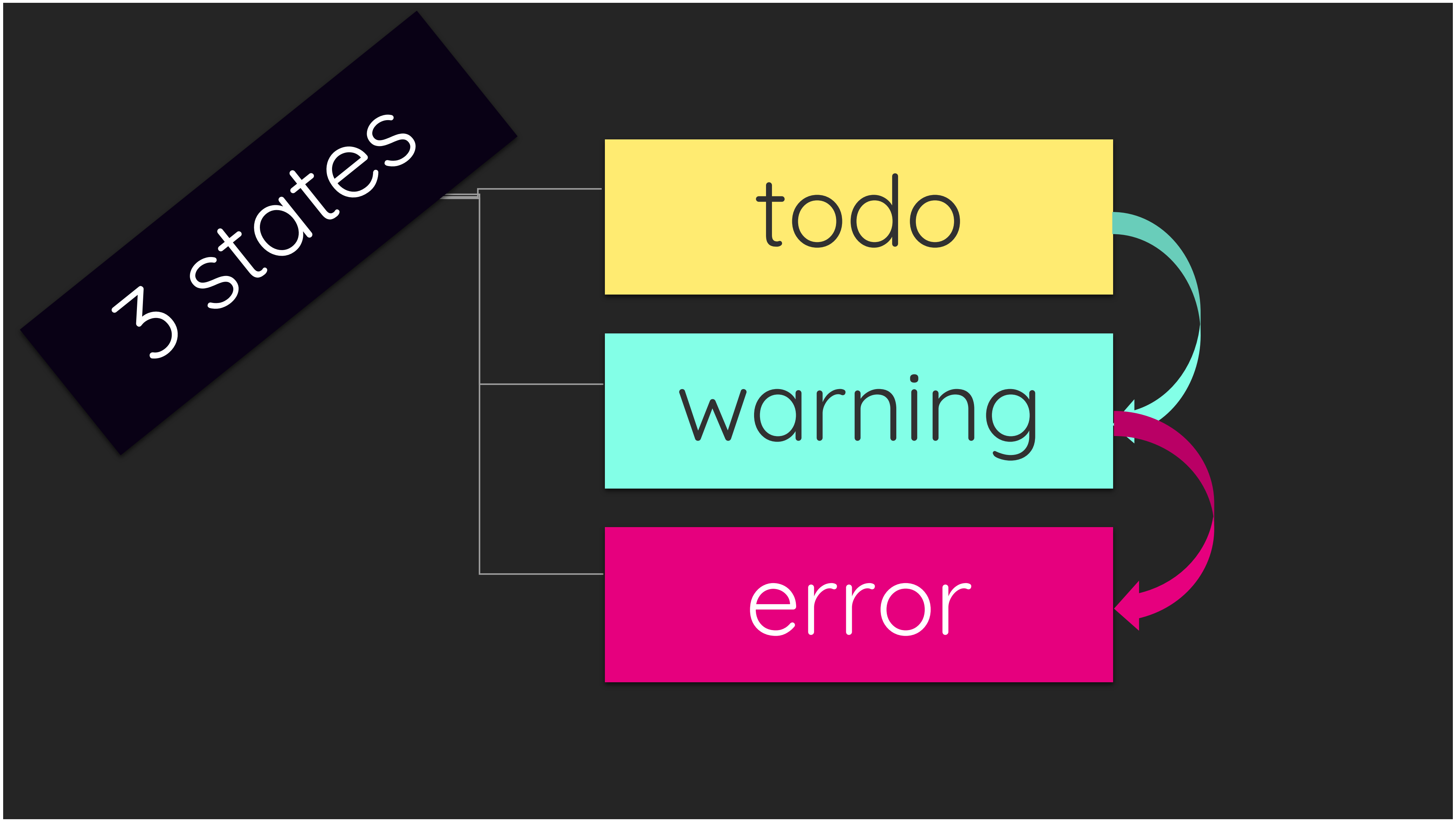
Pending

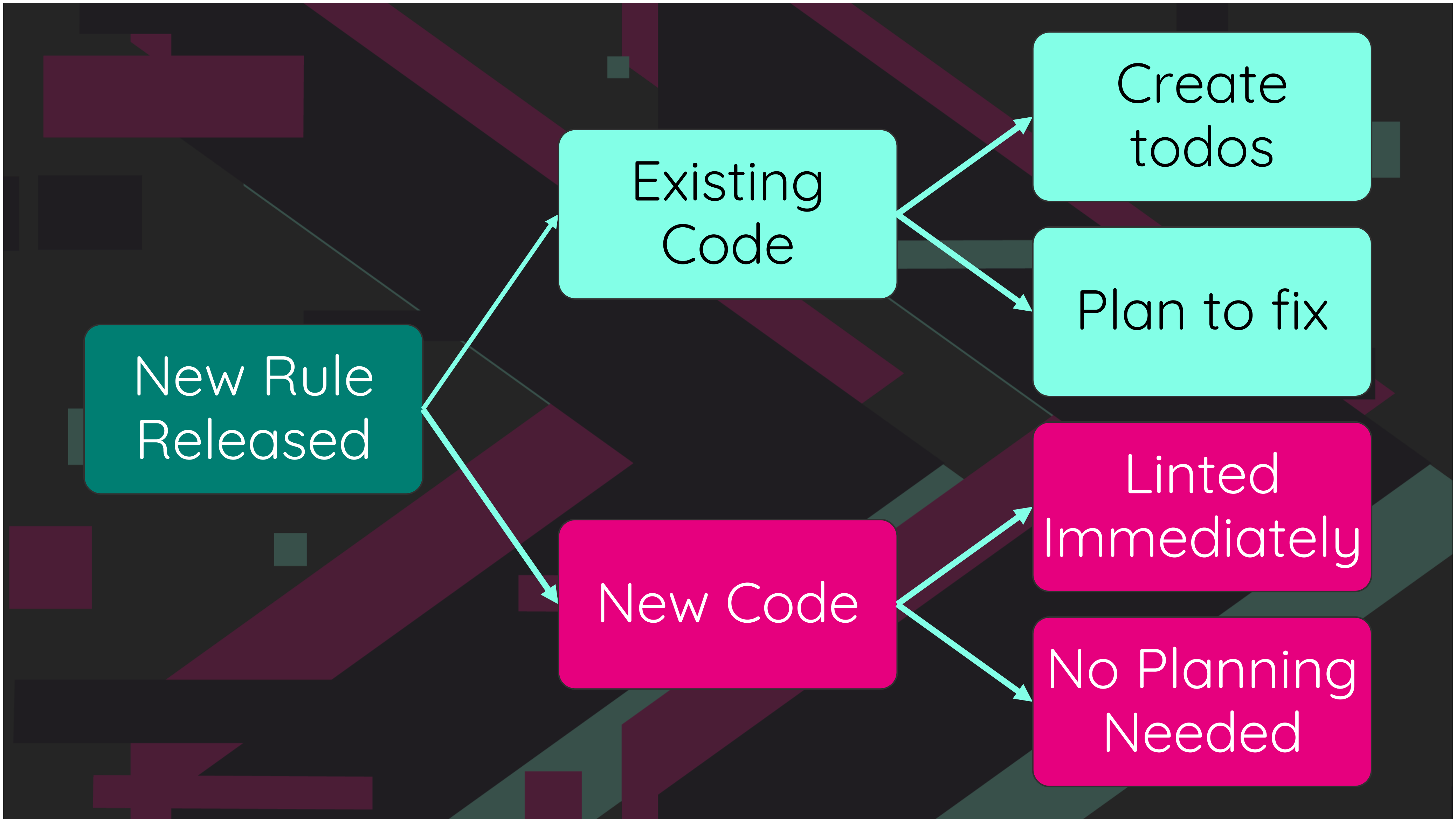
- `--print-pending`
- AKA "ignore forever" list


try to
turn rule
on
find
new
bugs
fix
bugs

WE NEED TO
DO SOMETHING
#SCHITTSCREEK
Pop TV

New Process: Todo!


3 states
todo
warning
error

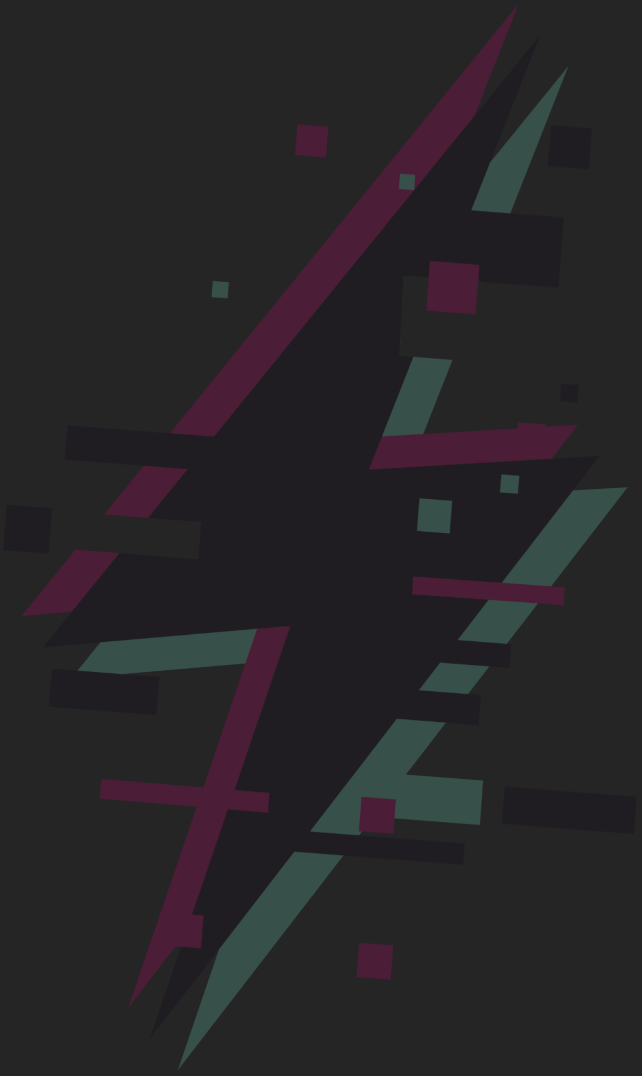

New Rule Released
Existing Code
New Code
Create todos
Plan to fix
Linted Immediately
No Planning Needed

Try it out!

```
$ ember-template-lint . --update-todo

$ ember-template-lint . --include-todo

$ ember-template-lint . --fix
```



ember®

Metrics for Measure

Metrics TL;DR

- Meaningful
- Controllable
- Easy to Access
- Actionable

Types of Actionable Outcomes

- Problem diagnosis
- Process improvement
- Goal setting
- Trend development

“When a measure becomes a target, it ceases to be a good measure.”

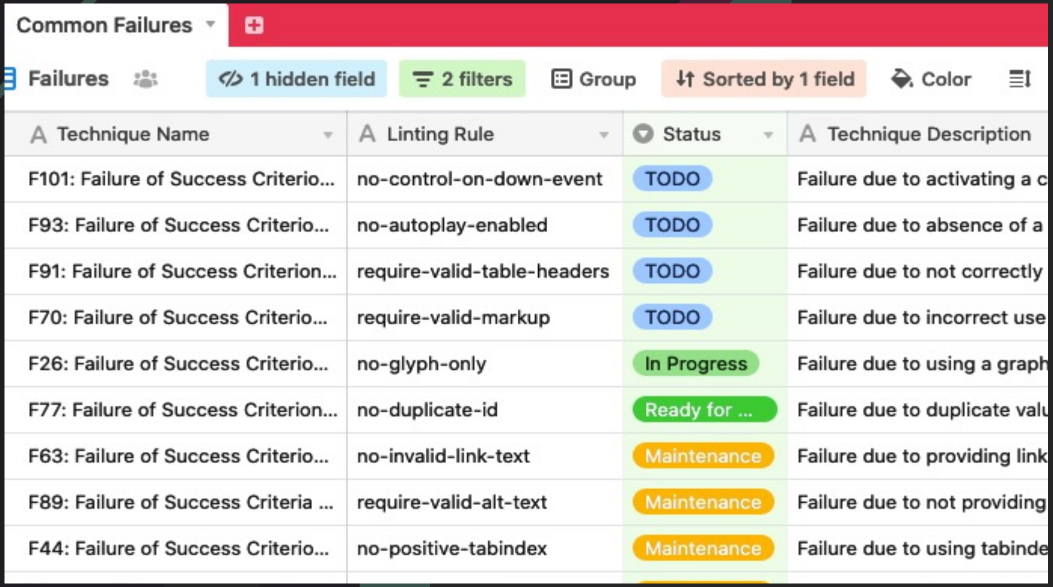
- Goodhart’s Law

Potential Violation Count

- Web Standards (WCAG)
- Location Legal Standards
- Failures identified by audit findings

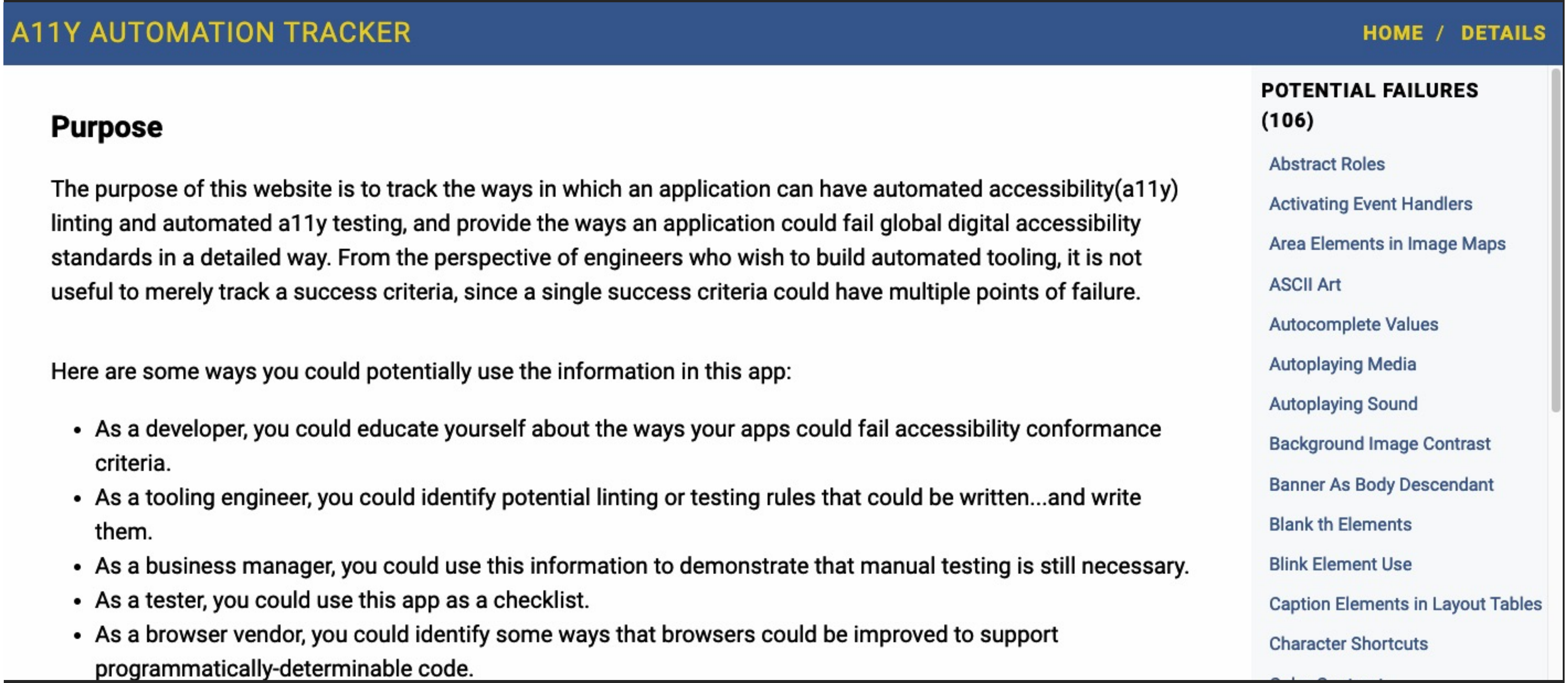
Potential Violation Count: Itemized List

- Turn ambiguity into clarity
- Make the unknown known
- Increase confidence



Technique Name	Linting Rule	Status	Technique Description
F101: Failure of Success Criterio...	no-control-on-down-event	TODO	Failure due to activating a c
F93: Failure of Success Criterio...	no-autoplay-enabled	TODO	Failure due to absence of a
F91: Failure of Success Criterion...	require-valid-table-headers	TODO	Failure due to not correctly
F70: Failure of Success Criterio...	require-valid-markup	TODO	Failure due to incorrect use
F26: Failure of Success Criterio...	no-glyph-only	In Progress	Failure due to using a graph
F77: Failure of Success Criterion...	no-duplicate-id	Ready for ...	Failure due to duplicate valu
F63: Failure of Success Criterio...	no-invalid-link-text	Maintenance	Failure due to providing link
F89: Failure of Success Criteria ...	require-valid-alt-text	Maintenance	Failure due to not providing
F44: Failure of Success Criterio...	no-positive-tabindex	Maintenance	Failure due to using tabinde

https://a11y-automation.dev/



A11Y AUTOMATION TRACKER

HOME / DETAILS

Purpose

The purpose of this website is to track the ways in which an application can have automated accessibility(a11y) linting and automated a11y testing, and provide the ways an application could fail global digital accessibility standards in a detailed way. From the perspective of engineers who wish to build automated tooling, it is not useful to merely track a success criteria, since a single success criteria could have multiple points of failure.

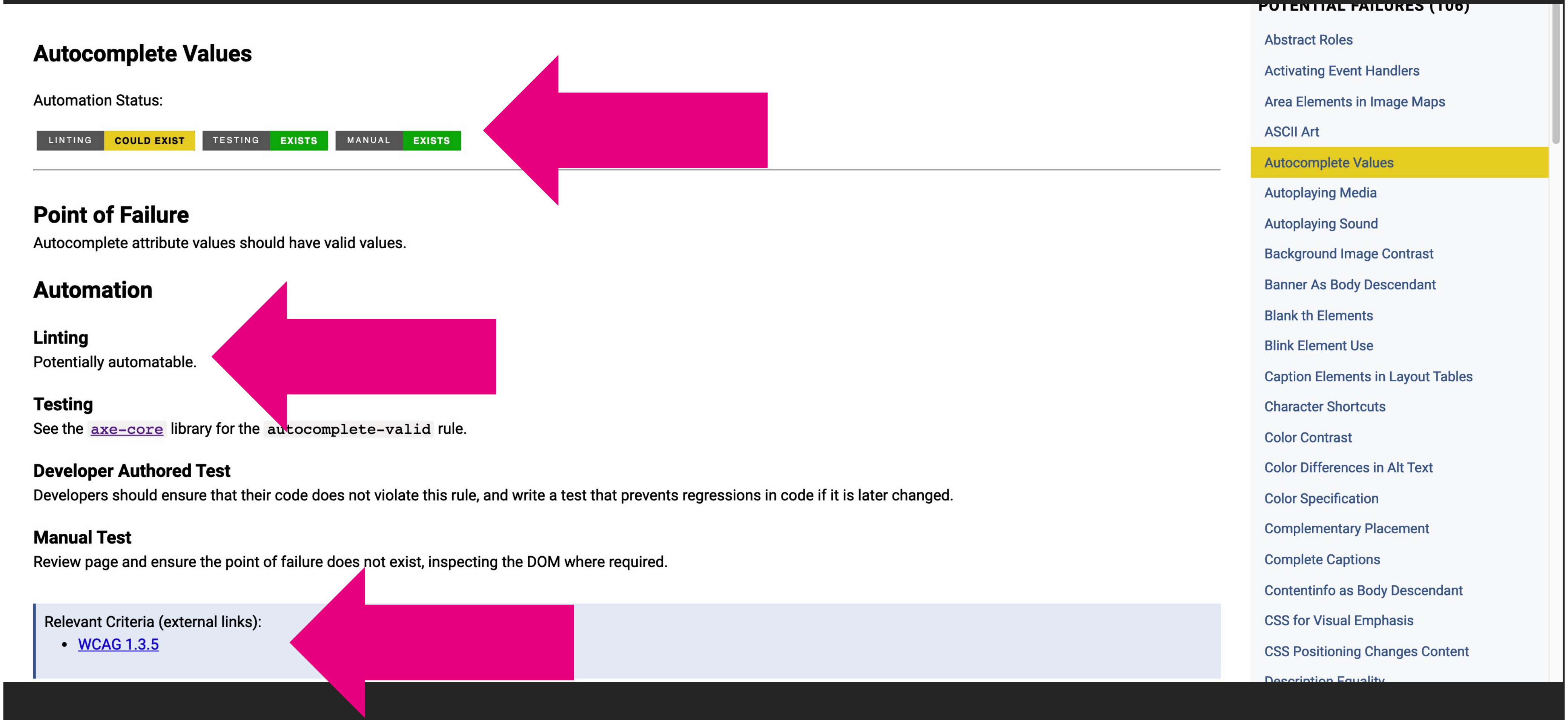
Here are some ways you could potentially use the information in this app:

- As a developer, you could educate yourself about the ways your apps could fail accessibility conformance criteria.
- As a tooling engineer, you could identify potential linting or testing rules that could be written...and write them.
- As a business manager, you could use this information to demonstrate that manual testing is still necessary.
- As a tester, you could use this app as a checklist.
- As a browser vendor, you could identify some ways that browsers could be improved to support programmatically-determinable code.

POTENTIAL FAILURES (106)

- Abstract Roles
- Activating Event Handlers
- Area Elements in Image Maps
- ASCII Art
- Autocomplete Values
- Autoplaying Media
- Autoplaying Sound
- Background Image Contrast
- Banner As Body Descendant
- Blank th Elements
- Blink Element Use
- Caption Elements in Layout Tables
- Character Shortcuts

https://a11y-automation.dev/



Autocomplete Values

Automation Status:

LINTING COULD EXIST | TESTING EXISTS | MANUAL EXISTS

Point of Failure

Autocomplete attribute values should have valid values.

Automation

Linting

Potentially automatable.

Testing

See the `axe-core` library for the `autocomplete-valid` rule.

Developer Authored Test

Developers should ensure that their code does not violate this rule, and write a test that prevents regressions in code if it is later changed.

Manual Test

Review page and ensure the point of failure does not exist, inspecting the DOM where required.

Relevant Criteria (external links):

- WCAG 1.3.5

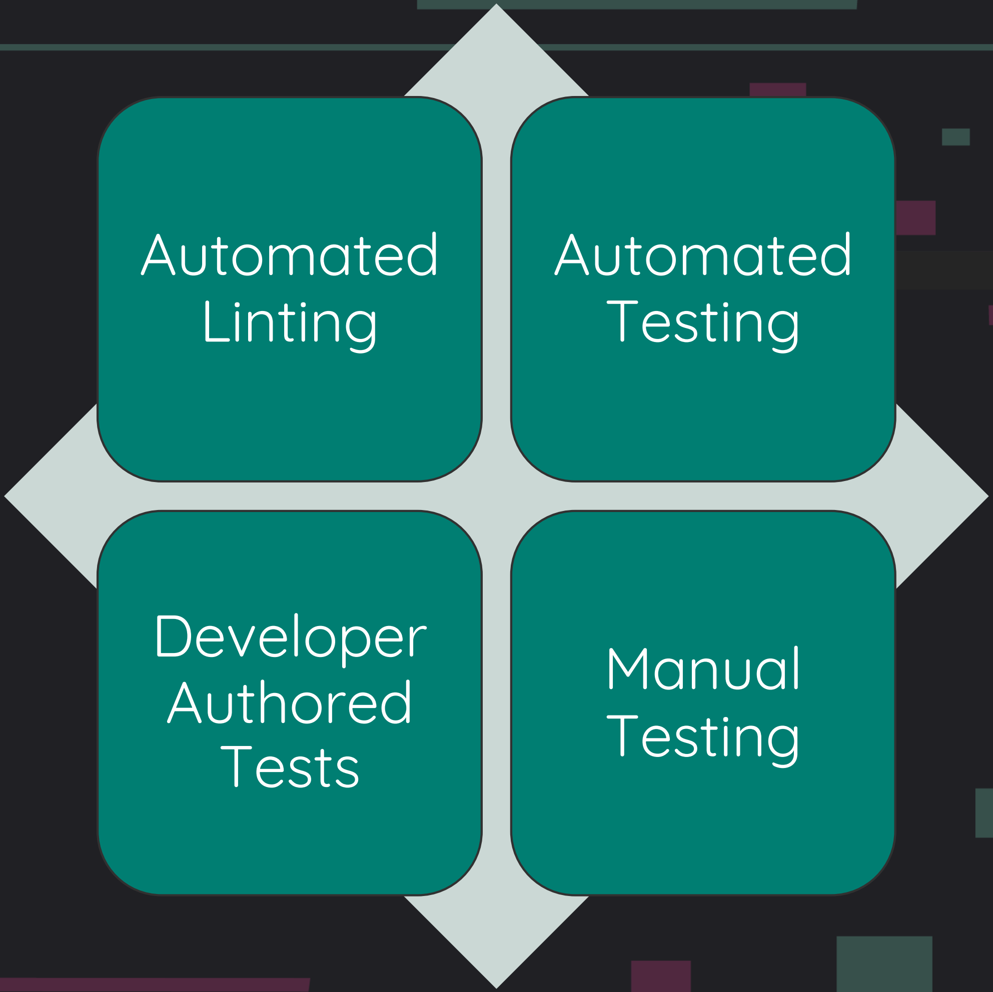
This is why you still need manual accessibility testers.

What About WCAG Success Criteria?

WCAG 1.3.1 (Info and Relationships)…

…25+ different failure scenarios

Metric: Automation Counts



Metrics From Audit Results

- Total Bug Count
- Bug Severity Count
- Time To Fix
- Violation Frequency

Trend Analysis

Expected Increasing Trends

- number of automated linting rules
- number of automated tests
- number of accessibility tests written by developers

Expected Decreasing Trends

- Support requests ⬇
- A11y-related issues in new/refactored code ⬇
- Fewer issues related to new automated linting/testing rules ⬇

Trends: Thresholds

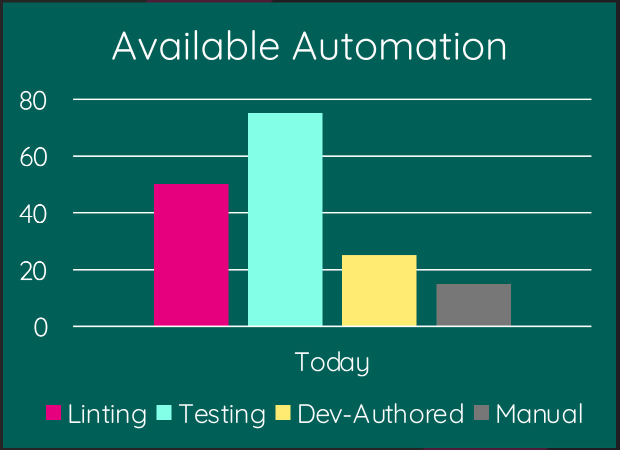
- Risk
- Conformance deterioration rate
- Learning opportunities
- App’s A11y health

Our Metrics Inform Future Work

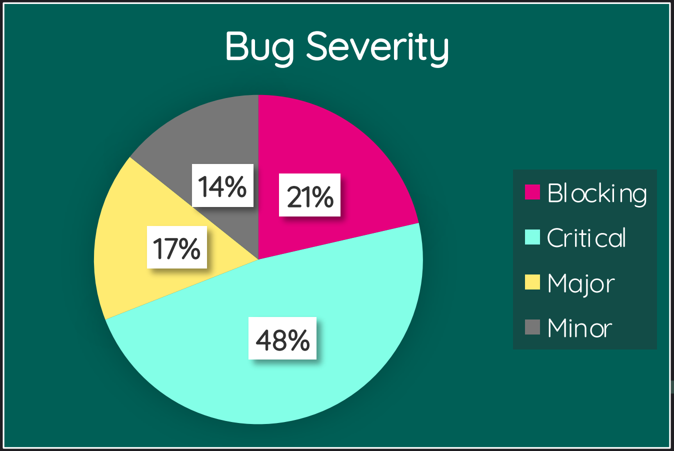
- Which potential violations could be automated?
- Which violations happen the most & how could we make that problem easier?
- How could we reduce time to fix?

A11Y METRICS DASHBOARD

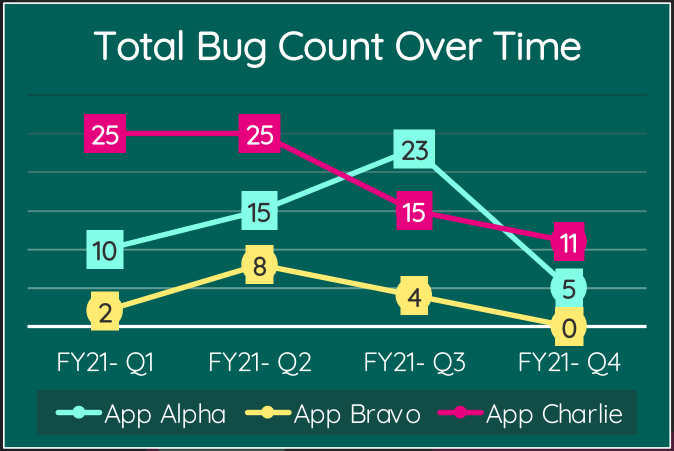
Available Automation



Bug Severity



Total Bug Count Over Time



Overall Health

Total Bug Count

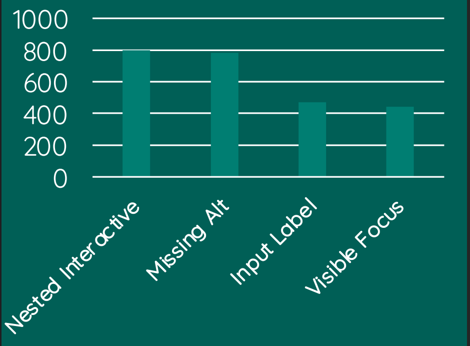
42

Avg. Time To Fix (days)

15

+5 day increase from last quarter

Incident Frequency



You have seen a path forward toward the vision of continuous accessibility.

I have inspired you to create something new or contribute to an effort already in progress.

I have empowered you to think about accessibility in your code in a new way.

Contributions

- Robert Jackson – maintains ember-template-lint and directed the idea for todos in ember-template-lint
- Steve Calvert – drove the todo implementation; designed the decay days feature
- Internal teams who agreed to be early adopters
- Joseph Temple – taught me to think like a research scientist

“When a measure becomes a target, it ceases to be a good measure.”

- Goodhart’s Law

Accessibility is for everyone.