Inclusive Linting

As software professionals we should strive to promote an inclusive work environment, which naturally extends to the code and documentation we write. It's important to keep the use of inclusive language consistent across an entire project or repository.

To achieve this, we recommend using a text file analysis tool such as an inclusive linter and including this as a step in your CI pipelines.

What to Lint for

The primary goal of an inclusive linter is to flag any occurrences of non-inclusive language within source code (and optionally suggest some alternatives). Non-inclusive words or phrases in a project can be found anywhere from comments and documentation to variable names.

An inclusive linter may include its own dictionary of "default" non-inclusive words and phrases to run against as a good starting point. These tools can also be customizable, oftentimes offering the ability to omit some terms and/or add your own.

The ability to add additional terms to your linter has the added benefit of enabling linting of sensitive language on top of inclusive linting. This can prevent things such as customer names or other non-public information from making it into your git history, for instance.

Getting Started with an Inclusive Linter

woke

One inclusive linter we recommend is `woke`. It is a language-agnostic CLI tool that detects non-inclusive language in your source code and recommends alternatives. While `woke` automatically applies a default ruleset with non-inclusive terms to lint for, you can also apply a custom rule config (via a yaml file) with additional terms to lint for.

Running the tool locally on a file or directory is relatively straightforward:

```
$ woke test.txt

test.txt:2:2-6: `guys` may be insensitive, use `folks`, `people` instead
```

```
(warning)
* guys
  ^
```

`woke` can be run locally on your machine or CI/CD system via CLI and is also available as a two GitHub Actions:

- Run woke
- Run woke with Reviewdog

To use the standard "Run woke" GitHub Action with the default ruleset in a CI pipeline:

1. Add the `woke` action as a step in your project's CI pipeline yaml:

```yaml
name: ci
on:
  - pull_request
jobs:
  woke:
    name: woke
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v2

      - name: woke
        uses: get-woke/woke-action@v0
        with:
          # Cause the check to fail on any broke rules
          fail-on-error: true
```

2. Run your pipeline
3. View the output in the "Actions" tab in the main repository view

Resources

- woke
- default ruleset
- example.yaml
- Run woke
- Run woke with reviewdog
- docs

Last update: August 22, 2024