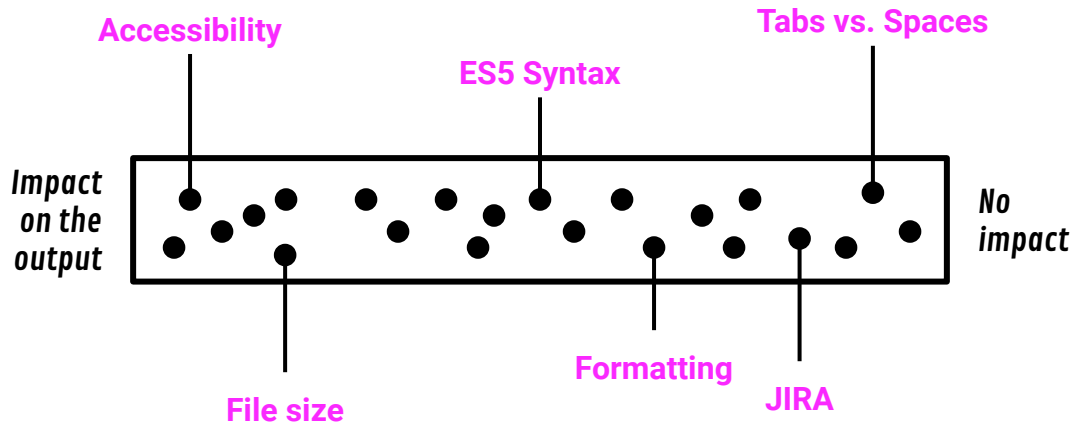




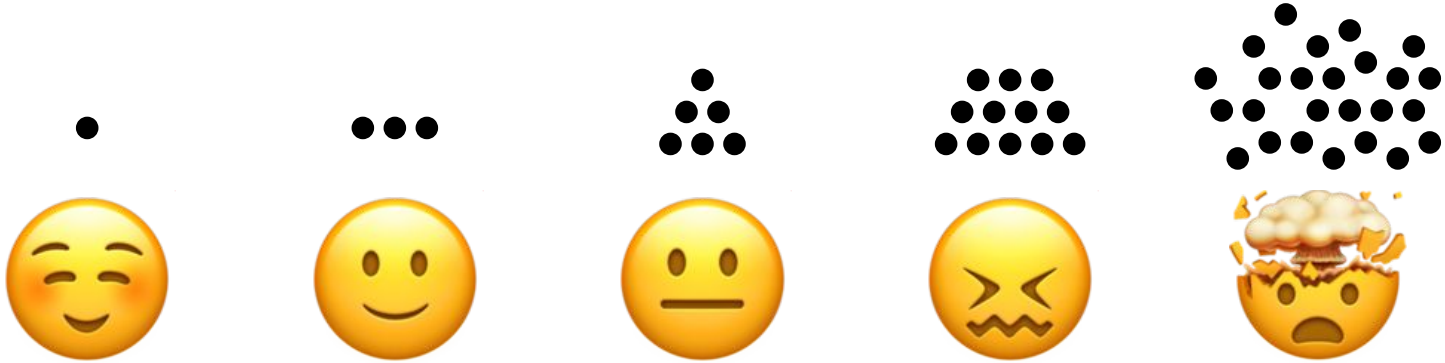
*Make your
tools work.*

Background

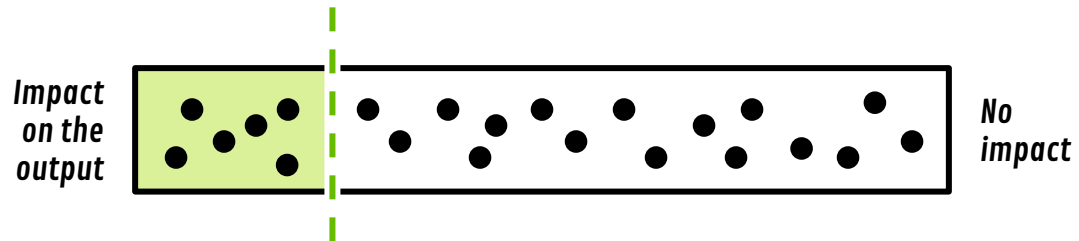
Things to do



Cognitive overload

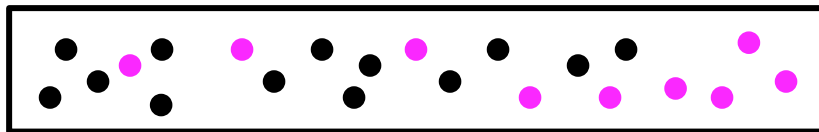


Prioritise



Automate

*Impact
on the
output*



*No
impact*

AUTOMATE



*Let your tools do
the stuff you don't
need to do.*

The background is a detailed illustration of a workshop. A wooden pegboard is covered with numerous tools, including hammers, wrenches, pliers, and screwdrivers, all rendered in a light brown, hand-drawn style. Below the pegboard, a workbench is cluttered with various items: a large metal vise, a saw, a drill, and other miscellaneous tools and materials. The overall color palette is warm, dominated by shades of brown and tan.

*Make your
tools work.*

Core Principles

Our tools should:

- *Help us*
- *Improve our output*
- *Be satisfying*

Off the shelf tools

- We have VSCode
- Syntax highlighting
- Custom themes
- Inline syntax validation
- Integrated Git + Diffs

```
import React from 'react';

// This is comment...
class MyComponent extends React.Component {
  constructor() {
    super();

    this.state = {
      title: 'World'
    };
  }

  componentDidMount() {
    console.log('MyComponent is mounted!');
  }

  clickHandler(title) {
    this.setState({ title });
  }

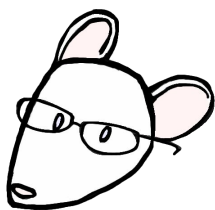
  render() {
    let { title } = this.state;

    return (
      <div>
        <h1>Hello, {title}!</h1>
        <button onClick={() => this.clickHandler('React')}>
          Change title
        </button>
      </div>
    );
  }
}

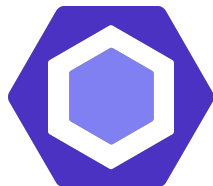
export default MyComponent;
```

Automated Formatting

- VSCode plugins and 'Format on Save'
- Automated, not enforced



+



+

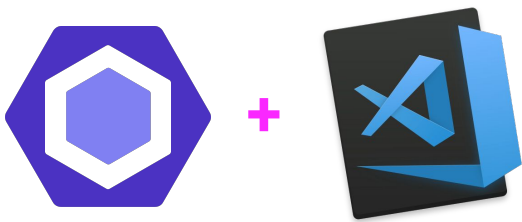


+



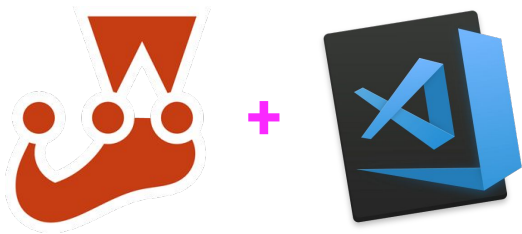
Inline Best Practice hints

- ESLint plugins:
 - react-a11y (accessibility best practices)
 - jest/recommended
 - airbnb/recommended
- As-you-type, not afterwards or one-off audit tasks



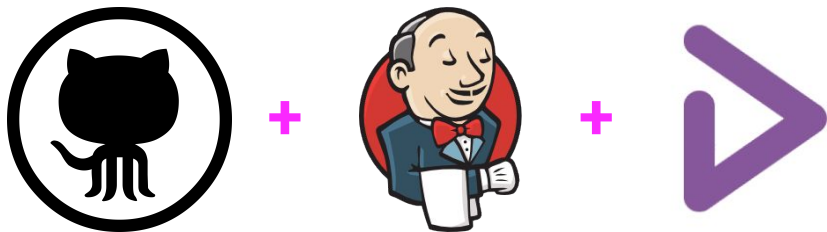
Inline / Integrated Unit Testing

- VSCode plugin
- Continual background testing
- Editor panel integration
- Inline test results, code coverage & error display



GitHub Integration for Jenkins / GoCD

- Automated Unit Tests, code checks in Docker
- Report results in GitHub
- Test for each commit, branch, PR



The screenshot shows a GitHub pull request interface. At the top, there's a green checkmark icon and the text "Changes approved" with a link to "Show all reviewers". Below this, a green circle icon and the text "Some checks haven't completed yet" with a link to "Hide all checks". The interface lists several checks:

- WIP** WIP In progress — Work in progress (Details)
- TaskLint** — All tasks in comments complete
- continuous-integration/jenkins/pr-merge** — This commit looks good (Details)
- meerstrap-universe-pull-request-v3/build** (Required) (Details)
- meerstrap-universe-pull-request-v3/tests** (Required) (Details)

At the bottom, there's a warning icon and the text "This branch is out-of-date with the base branch" with a link to "Update branch". Below this, there's a red text line: "As an administrator, you may still merge this pull request." and a button labeled "Squash and merge" with a dropdown arrow. To the right of the button, there's a link: "You can also open this in GitHub Desktop or view command line instructions."

Automated Refactoring

- Command-line refactoring
 - codemod
 - jscodeshift
- Inline refactoring

The screenshot shows the GitHub repository for 'facebook / codemod'. At the top, there's a navigation bar with links for Pull requests, Issues, Marketplace, and Explore. Below this, the repository name 'facebook / codemod' is displayed along with statistics: 59 Watchers, 2,622 Stars, 137 Forks, and 137 Issues. The main content area shows a list of files and their commit history. A table lists files like .codemod, .gitignore, .travis.yml, LICENSE, Makefile, README.md, dev-requirements.txt, pytest.ini, and setup.py, along with their commit messages and dates. Below the file list, there's a section for the README.md file, which includes a title 'codemod', a badge for 'pypi v1.0.0', and a description of the tool. The README also includes an 'Overview' section and a code snippet for the command-line interface.

Search or jump to... / Pull requests Issues Marketplace Explore

facebook / codemod Watch 59 Star 2,622 Fork 137

Code Issues 10 Pull requests 2 Projects 0 Insights

112 commits 1 branch 1 release 29 contributors Apache-2.0

Branch: master New pull request Create new file Upload files Find file Clone or download

modocache Merge pull request #104 from sbz/unicode-fix Latest commit 5ddf246 on 4 Oct

codemod	Fix test suite handle by doctest for python3.x	2 months ago
.gitignore	Add .gitignore	4 years ago
.travis.yml	Add python3.x stable version in travis ci integration	2 months ago
LICENSE	codemod: initial codemod release checkin.	10 years ago
Makefile	Refactor, PEP8 Fixes, script "codemod", release PyPi	3 years ago
README.md	corrected minor typos	a year ago
dev-requirements.txt	Add python3.x stable version in travis ci integration	2 months ago
pytest.ini	Refactor, PEP8 Fixes, script "codemod", release PyPi	3 years ago
setup.py	Bump version to 1.0, fixes #81	a year ago

README.md

codemod

pypi v1.0.0 downloads 35/week build passing Code Health

Overview

codemod is a tool/library to assist you with large-scale codebase refactors that can be partially automated but still require human oversight and occasional intervention.

Example: Let's say you're deprecating your use of the `` tag. From the command line, you might make progress by running:

```
codemod -m -d /home/jrosenstein/www --extensions php,html \  
'<font *color="(.*?)">(.*?)</font>' \  
'<span style="color: \1;"></span>'
```

For each match of the regex, you'll be shown a colored diff, and asked if you want to accept the change (the replacement of the `` tag with a `` tag), reject it, or edit the line in question in your \$EDITOR of choice.

Install

In a virtual environment or as admin user

The background of the image is a workshop or garage. On the left, a large hand saw with a wooden handle and a metal blade is leaning against a workbench. In the center, a wooden pegboard is mounted on the wall, holding several tools including wrenches, pliers, and screwdrivers. To the right, a shelf holds various small tools and containers. The overall scene is brightly lit, with a warm, yellowish-orange glow. The text "Build your own tools" is overlaid in the center, written in a white, cursive font with a blue drop shadow.

*Build your
own tools*

Automated deployment scripts

- Automated repetitive steps:
 - Preflight checks (clean workspace)
 - Build packages
 - Test packages
 - Regenerate docs
 - Tag in git
 - Publish to npm

GitHub-JIRA integration

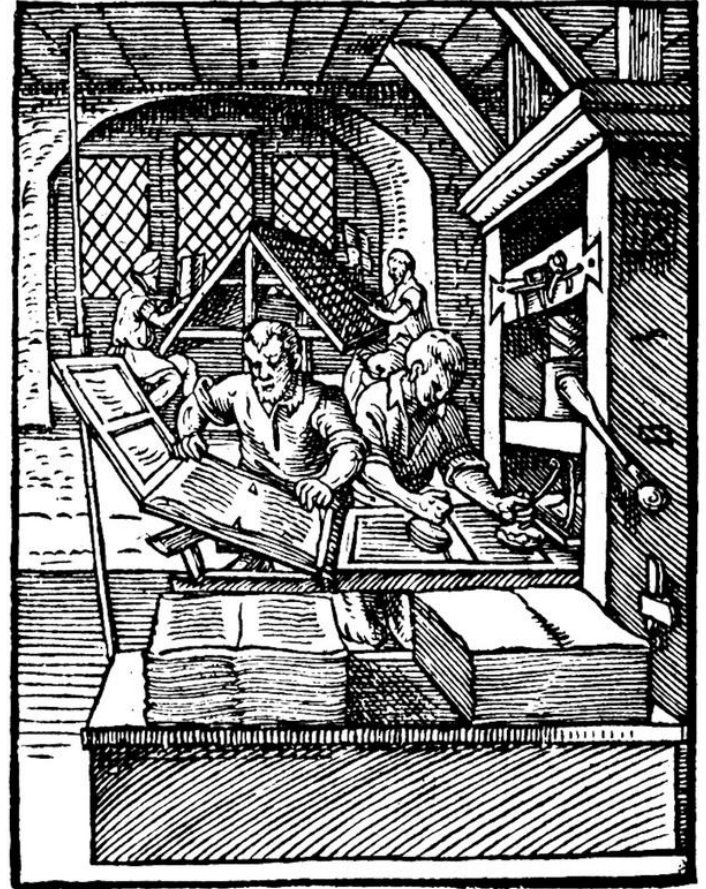
- Creates tickets in JIRA for GitHub Issues & Pull Requests
- Transition tickets in reaction to GitHub webhook events
- Assign users, mirror comments & updates

Craftsmanship

Software development is a comparatively new profession.

Building your own tools is not a new idea - it's an old one, actually.

Mass produced tools work for common tasks, but for in-depth bespoke work, it's worthwhile investing in custom tools.



Summary

Make your tools work for you.