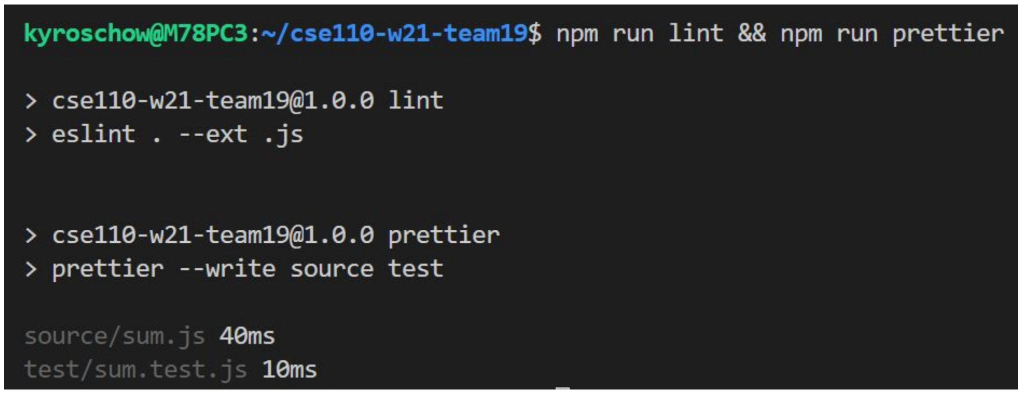
CI Pipeline Phase 1

Linting and Code Style Enforcement

ESLint and Prettier are used for linting and code style reinforcement.
This happens via a GitHub Action, when a pull request is made. The workflow is as follows:

1. ESLint checks for linting errors.
2. Run Prettier to format the code.
3. Commit correctly formatted code to the branch initiating the pull request.

Code style is not reinforced in the code editor, because every developer may prefer different IDEs, and it is difficult to create some configuration that can be set across multiple IDEs. However, before committing any code, developers should run “npm run lint” and “npm run prettier” to check if the code passes ESLint and is formatted correctly by Prettier. Running these commands will ensure the workflow completes successfully on GitHub.
Working: Yes



```
kyroschow@M78PC3:~/cse110-w21-team19$ npm run lint && npm run prettier

> cse110-w21-team19@1.0.0 lint
> eslint . --ext .js


> cse110-w21-team19@1.0.0 prettier
> prettier --write source test

source/sum.js 40ms
test/sum.test.js 10ms
```

Code Quality via Tool

Currently exploring to use Code Climate to automate ensuring code quality.
This happens via a GitHub action, when any commit is pushed to a remote branch. This allows developers to frequently evaluate their code quality.
Working: No

Code Quality via Human Review

Code quality via human review is done through pull requests. The “main” and “dev” branches are protected, and changes can only be made via pull requests. The workflow is as follows:

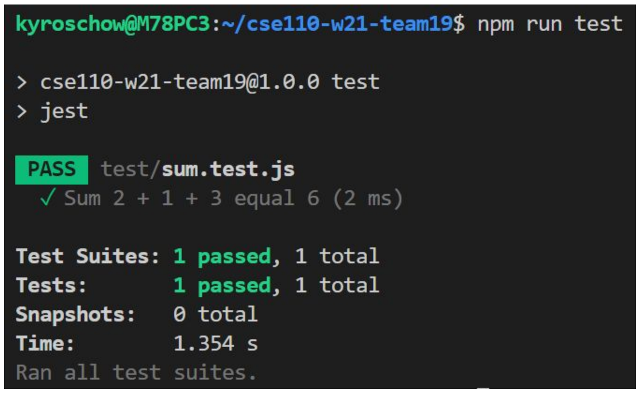
1. Developer checks out the repository.
2. Developer checks out a new branch called “new feature”, and works on “new feature”.
3. Developer commits and pushes changes to “new feature”.
4. If “new feature” is completed, then a pull request is made to merge with the “dev” branch.

5. Code is merged to the “dev” branch after review and passing all tests.
6. If the code in “dev” is ready for deployment, merge code to the “main” branch and the code will be deployed automatically.

Currently, there is no protection set on the “main” and “dev” branches. However, the team has already adopted the workflow of creating pull requests after working in some feature branch.
Working: Yes

Unit Tests via Automation

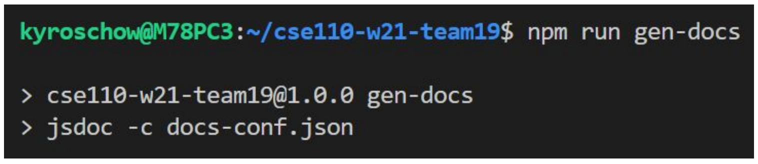
Unit tests are done through GitHub actions, when a pull request is made. Jest is used to perform tests on the test files under the directory called “tests. Before making a pull request to the “dev” branch, developers should run “npm run test” to test if all the unit tests are passed. This will ensure that the unit tests run by GitHub actions will be passed as well.
Working: Yes



```
kyroschow@M78PC3:~/cse110-w21-team19$ npm run test

> cse110-w21-team19@1.0.0 test
> jest

 PASS  test/sum.test.js
  ✓ Sum 2 + 1 + 3 equal 6 (2 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        1.354 s
Ran all test suites.
```

Documentation Generation via Automation

Documentation generation is done through GitHub actions, when a push is made to the “main” branch. JSDoc is used to generate documentation, and outputs the resulting documentation files in the “docs” directory. Normally, the “docs” directory is ignored by git and only generated when the application is deployed, but developers can run “npm run gen-docs” to preview their documentation before pushing or creating pull requests.
Working: Yes



```
kyroschow@M78PC3:~/cse110-w21-team19$ npm run gen-docs

> cse110-w21-team19@1.0.0 gen-docs
> jsdoc -c docs-conf.json
```