

Clean `<code/>`
is no longer a myth!

Karan Balkar

About Me

Developer | Author | Technophile

karanbalkar@gmail.com

- Android mobile app developer for the past 5+ years
- Worked primarily on developing B2C and B2B based applications
- Worked on cutting edge technologies including AI, AR & VR
- Involved in developing web applications using React & Angular
- Currently working on developing mobile applications using React Native
- Passionate about technology, meeting new people and sharing ideas! 😊

“Stay Hungry ! Stay Foolish !”



Background

Agile based software development

- Result oriented
- Continuous improvement
- Small, cross functional teams

Software design principles

- DRY
- KISS
- SOLID
- SoC

Age of automation

- Using more tools & accelerators
- Reducing overall cost and effort

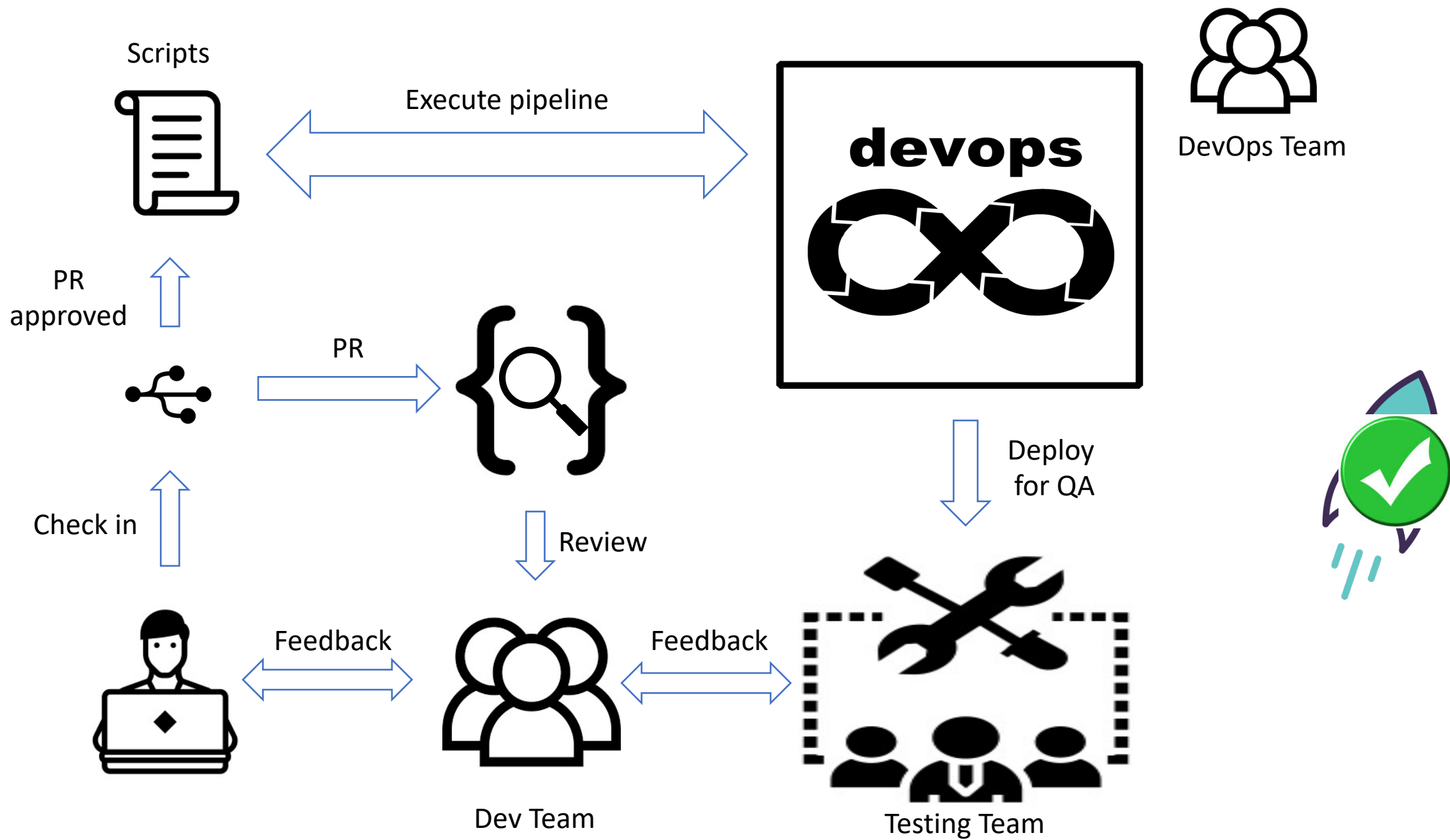
This session is not about ..

Why to write clean code?

What is clean code ?

What is a clean architecture?

Why don't software engineers write clean code?





Best moment

Drawbacks

Developers focus on making the solution “work”

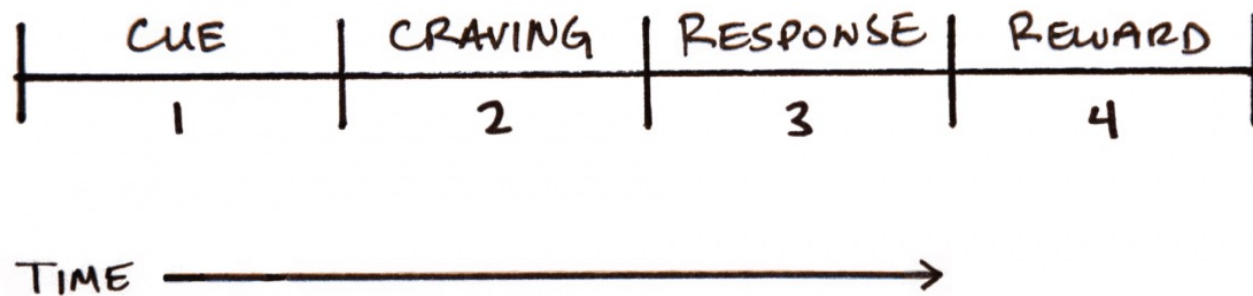
Less amount of time spent on refactoring

No or limited pre-commit checks in place

Working Code Vs Maintainable Code

A bit about human psychology

THE FOUR STAGES OF HABIT



- Writing clean code needs to become a habit
- Four stages of a building a habit
 - Cue
 - Craving
 - Response
 - Reward

Writing clean code – Developing the habit!

Cue

- Remove dead code
- Find unused variables
- Automate formatting
- Apply custom rules

Craving

- Write maintainable code
- Maximize refactoring
- Minimize bugs

Response

- Using Git hooks
 - Pre-commit
 - Pre-push
- Linting
 - Rules

Rewards

- Consistency
- Effort and cost savings
- Code quality

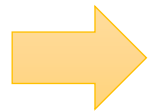
Using Husky and Git Hooks

- Husky (<https://www.npmjs.com/package/husky>)
 - It allows developers to run scripts when doing specific actions
 - For example: Pre-commit -> lint -> commit; Pre-push -> lint -> Push
- Git hooks
 - Scripts that run automatically every time a particular event occurs in a Git repository. These events can be:
 - ☐ pre-commit
 - ☐ pre-push
 - ☐ pre-rebase
 - ☐ post-update

Husky - Setup

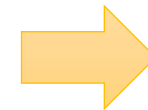
- There are significant changes in the Husky tooling if one is migrating from husky v4 to v6. The new Husky approach tends to keep the Husky away from the JS eco-system, instead making it hybrid.

`npx husky-init`



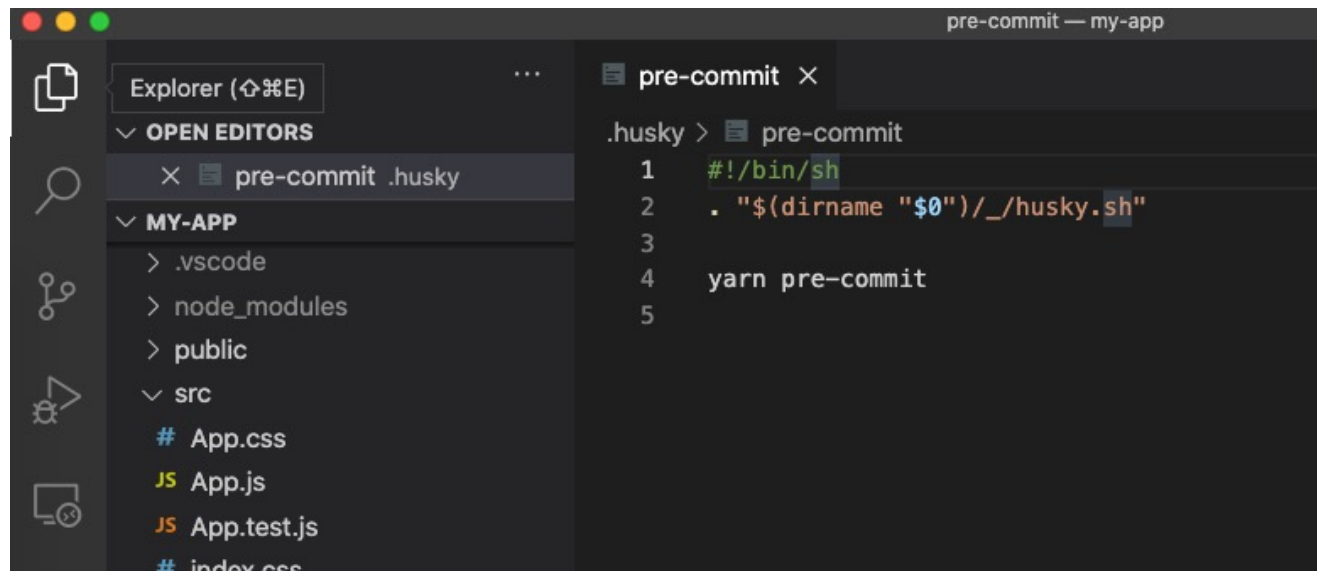
```
{  
  "scripts": {  
    ...  
    "prepare": "husky install"  
  },  
  "devDependencies": {  
    ...  
    "husky": "^6.0.0"  
  }  
}
```

package.json



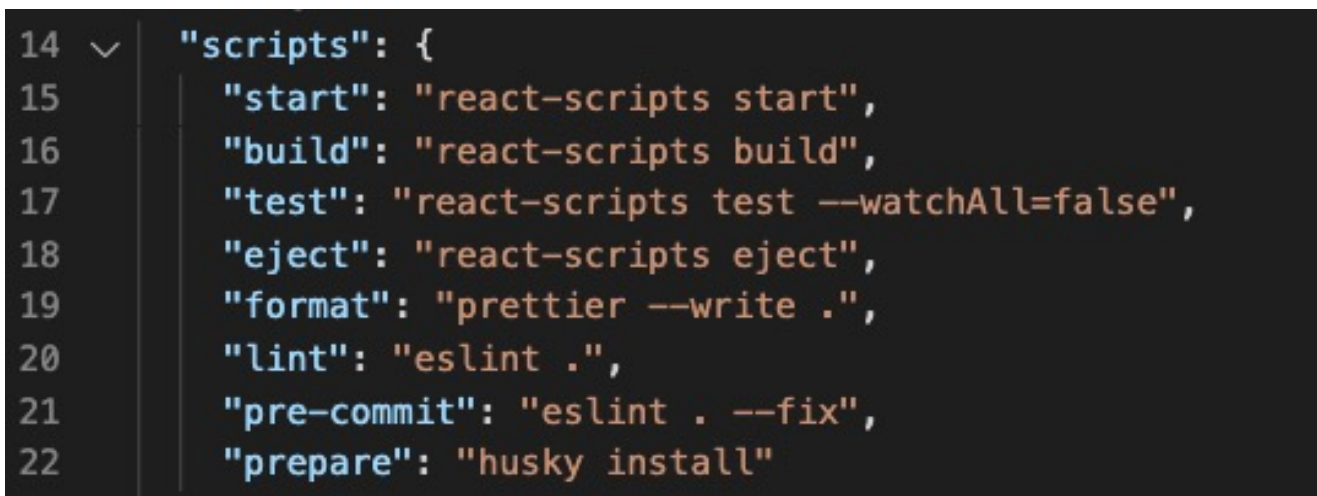
`npm install`

Using pre-commit hooks



A screenshot of the Visual Studio Code interface. The Explorer sidebar on the left shows the project structure with folders like .vscode, node_modules, public, and src, and files like App.css, App.js, App.test.js, and index.css. The main editor area shows a file named pre-commit inside the .husky directory. The content of the pre-commit file is as follows:

```
.husky > pre-commit
1  #!/bin/sh
2  . "$(dirname "$0")/_/husky.sh"
3
4  yarn pre-commit
5
```



A screenshot of a code editor showing a JSON configuration, likely package.json. The 'scripts' section is expanded, showing the following configuration:

```
14  "scripts": {
15    "start": "react-scripts start",
16    "build": "react-scripts build",
17    "test": "react-scripts test --watchAll=false",
18    "eject": "react-scripts eject",
19    "format": "prettier --write .",
20    "lint": "eslint .",
21    "pre-commit": "eslint . --fix",
22    "prepare": "husky install"
```

Using ESLint & Prettier

- ESLint helps developers to find problems with their code without executing it.
- Prettier is a code-formatter that helps you format the code in a specific way.
- When creating React JS based applications, *create-react-app* comes with **eslint** config.

```
"eslint": "^7.31.0",  
"eslint-config-airbnb": "^18.2.1",  
"eslint-config-prettier": "^8.3.0",  
"eslint-plugin-import": "^2.23.4",  
"eslint-plugin-jsx-a11y": "^6.4.1",  
"eslint-plugin-prettier": "^3.4.0",  
"eslint-plugin-react": "^7.24.0",  
"eslint-plugin-react-hooks": "^4.2.0",  
"husky": "^7.0.0",  
"prettier": "^2.3.2"
```

yarn add --dev --exact prettier

Using ESLint & Prettier (contd..)

- One can add rules for basic syntax validation, brace style, variable declaration etc. to fit the needs of the project.
- There are two primary ways to configure ESLint:
 1. Creating eslintrc.* file or
 2. Adding eslintConfig field in the package.json file

yarn run lint

```
.eslintrc.js x
.eslintrc.js > [?] <unknown> > extends
14  parserOptions: {
15    ecmaFeatures: {
16      jsx: true,
17    },
18    ecmaVersion: 12,
19    sourceType: "module",
20  },
21  plugins: ["react", "jsx-a11y"],
22  rules: {
23    "react-hooks/exhaustive-deps": "error",
24    "react/react-in-jsx-scope": "off",
25    "react/jsx-filename-extension": [1, { extensions: [".js", ".jsx"] }],
26    "no-var": "error",
27    "brace-style": "error",
28    "prefer-template": "error",
29    radix: "error",
30    "space-before-blocks": "error",
31    "import/prefer-default-export": "off",
32  },
33  overrides: [
34    {
35      files: [
36        "**/*.test.js",
37        "**/*.test.jsx",
38        "**/*.test.tsx",
39        "**/*.spec.js",
40        "**/*.spec.jsx",
41        "**/*.spec.tsx"
42      ],
43    }
44  ]
45 }
```

Commit Lint

- Adding a valid and meaningful commit message to every commit is equally important.
- It helps to track down previous issues in a comparatively less amount of time.

```
yarn add --dev  
@commitlint/config-conventional  
@commitlint/cli
```

```
JS commitlint.config.js ×  
JS commitlint.config.js > [?] <unknown> > 🔑 rules  
1  module.exports = {  
2    parserPreset: "conventional-changelog-conventionalcommits",  
3    rules: {  
4      "body-leading-blank": [1, "always"],  
5      "body-max-line-length": [2, "always", 100],  
6      "footer-leading-blank": [1, "always"],  
7      "footer-max-line-length": [2, "always", 100],  
8      "header-max-length": [2, "always", 100],  
9      "subject-case": [  
10       2,  
11       "never",  
12       ["sentence-case", "start-case", "pascal-case", "upper-case"],  
13     ],  
14     "type-enum": [  
15       2,  
16       "always",  
17       [  
18         "build",  
19         "chore",  
20         "ci",  
21         "docs",  
22         "feat",  
23         "fix",  
24         "perf",  
25         "refactor",  
26         "revert",  
27       ],  
28     ],  
29   },  
30 }
```

Environmental configurations

- Checking & managing versions of dependencies.
- Usually in the DEV environment, one does not run the “npm install” command unless one makes some version changes or are running the project for the first time.
- Install Node Version Manager (nvm) to use any number of node versions.
 - `nvm install 14.17.3` (or any version one wants)
 - `nvm list`
 - `nvm use 14.17.3`

Final thoughts..

- Writing clean code helps improve efficiency.
- Project teams need to focus on environment setup and clean code configuration before actual development starts.
- Refactoring should be made a continuous process.

“Every developer loves to write clean code, yet not every developer writes code in the same way”

Source: <https://www.devbridge.com/articles/coding-best-practices/>



References

- <https://dzone.com/articles/software-design-principles-dry-and-kiss>
- <https://digital.ai/glossary/agile-development-success>
- <https://poatek.com/2017/07/31/clean-code-what-it-is-and-why-its-important/>
- <https://www.quora.com/I-am-unable-to-write-clean-code-no-matter-how-hard-I-try-What-should-I-do>
- <https://gamedevunboxed.com/the-real-reason-its-difficult-to-write-clean-code/>
- <https://jamesclear.com/three-steps-habit-change>
- <https://www.freecodecamp.org/news/these-tools-will-help-you-write-clean-code-da4b5401f68e/>
- <https://nicolas-dmg.medium.com/clean-and-format-your-code-before-committing-with-husky-c88039b16510>
- <https://dzone.com/articles/clean-code-explanation-benefits-amp-examples>
- <https://adarshaacharya.com.np/blog/setup-husky-precommit-hooks>

Image source : Google Images



Thank you!



Any questions ?