前端编码规范工程化

1. 背景介绍

随着前端技术的发展，前端项目正在变得越来越复杂。`JavaScript` 作为一门弱类型解释性编程语言，其灵活多变的语法极大的提高了前端开发效率，但同时也存在一系列问题。在大型项目开发过程中，代码维护所占的时间比重往往大于新功能的开发。因此编写符合团队编码规范的代码是至关重要的。 一致性的代码规范可以增强团队开发协作效率、提高代码质量、减少遗留系统维护的负担。但是每个人的代码编写喜好不同，代码风格也会迥然不同，若要一个团队内所有的成员都能发挥最大程度的价值，一个具有普适性的标准是必不可少的。

那么，如何制定前端团队的代码规范，如何在团队内进行最小成本的推广，就是一个合格的前端架构师面临的一大难题。很多团队的基础建设都只是简单地知晓其中一部分规范工具的使用，但却没有一套完整的工程化产出来助力研发同学实现规范落地。因此，如果我们能够产出一套完整化的前端编码规范工具，都会对不仅能够解决存量项目的编码异味，还能够保证后续所有项目的编码质量。

2. 学习成果

通过本套课程的学习，我们会了解到国内外各个大厂和社区间的前端编码规范，通过前端脚手架的输出一套完整的工程化产物，掌握到：

1. 学习如何通过 `monorepo` 和 `pnpm` 的多包管理方式开发一套多 `npm` 包的管理方式，以及如何将发包流程植入 `CI` 实现自动化发布，以及 `CHANGLOG` 的自动化更新部署；
2. 学习现有前端前沿的研发流程下，我们可以通过哪些工具提升项目的编码规范，并提供配套工具的最佳实践，包括但不限于 `eslint` 、 `stylelint` 、 `commitlint` 、 `markdownlint` 、 `husky` 等，以及如何将单元测试植入配套工具的具体实现；
3. 学习如何通过脚手架的方式，以交互式形式一键接入，实现对 `JavaScript` 、 `Typescript` 、 `React` 、 `Vue` 等不同类型的前端项目下的标准的语法限制；
4. 学习如何对存量项目进行优化：对于存量代码不符合规范的问题，支持一键扫描和一键修复，一键式的修复存量问题，最小化存量代码的更新成本；
5. 学习如何对新项目添加规范：支持一键接入新增项目，通过结合git `pre-commit` 钩子，对提交文件进行编码规范的扫描；同时通过 `husky` 的 `commit-msg` 钩子，对本次代码提交 `message` 的格式进行扫描。

学完后对工作/面试的帮助：

- 面试上：在大厂面试中，不管是个人产出，还是团队的横向建设，可以当做技术亮点核心介绍；
- 后续工作上：可以快速植入到实际开发项目中，能够直接带来生产力的提效；

3. 产物

1. 前端编码规范工程化多包 `github` 地址：encode-studio-fe/fe-spec；
2. 配套的静态站点：前端编码规范工程化；
3. 编码规范 `npm` 包
   a. `encode-fe-eslint-config` 提供针对不同框架(`React` 、 `Vue`)配套的编码规范、对 `eslint` 的快速接入及 `eslint` 插件实现；
   b. `encode-fe-stylelint-config` ：提供定义样式规范及 `stylelint` 的配置；
   c. `encode-fe-commitlint-config` ：提供 `Git` 代码提交规范及对 `commit msg` 的校验配置；
   d. `encode-fe-markdownlint-config` ：提供对 `markdown` 的使用规范及 `markdownlint` 的使用配置；
4. 脚手架
   a. `encode-fe-spec-cli` ：将上述编码规范及 `lint` 工具收敛，提供简单的 `CLI` 工具及 `Node API` ，让项目能够一键接入、一键扫描、一键修复、一键升级，同时为项目配置 `git commit` 卡口，降低项目接入的成本。

4. 技术选型

- 多包管理工具： `lerna` ；
- 包管理工具： `pnpm` ；
- `lint` 规范：
  - `markdownlint` ；
  - `commitlint` ；
  - `stylelint` ；
  - `prettier` ；
  - `eslint` ；
  - `editorconfig` ；
- CLI（一键安装/一键扫描/一健修复/一键更新）： `node` 生态；

- 测试工具：`jest`；

5. 课程安排

5.1. 第一节课

> 配套代码地址：链接

1. 初始化项目：`lerna` + `pnpm` 的多包构建；
2. 梳理 `HTML` 规范、`GIT` 规范、`markdown` 规范及 `CHANGELOG` 规范；
3. 创建 `encode-fe-commitlint-config`，`encode-fe-markdownlint-config`，使用 `npm scripts` 发布 `npm` 包；
4. 使用 `vuepress` 搭建静态资源站点，创建 `markdown` 及 `commitlint` 规范文档；
5. 支持 `CHANGELOG` 自动化更新；
6. 使用 `git action` 发布静态资源站点；

5.2. 第二节课

> 配套代码地址：链接

1. 提供 `CSS`、`JavaScript`、`Typescript`、`Node` 编码规范；
2. 创建 `encode-fe-stylelint-config`，支持测试用例；
3. 创建 `encode-fe-eslint-config` 支持前端不同项目维度（`JavaScript`、`Typescript`、`React`、`Vue`）下的 `eslint` 定制化配置；
4. 支持 `prettier`；

5.3. 第三节课

> 配套代码地址：链接

1. 编码一套定制的 `eslint` 插件；
2. 基于前两节课内容，提供一套完整的 `CLI`，支持一键接入上述完整的规范内容；
3. 支持自动安装 `linter` 依赖：`eslint`、`stylelint`、`commitlint`、`markdownlint`；
4. 支持自动创建 `.eslintrc.js`、`.eslintignore`、`.stylelintrc.js`、`.stylelinti

gnore`、`commitlint.config.js`、`.markdownlint.json`、`.markdownlintignore`、`.prettierrc.js`、`.editorconfig`等；

5.4. 第四节课

> 配套代码地址：链接

1. 支持基于代码提交时使用`husky`执行前三节课的编码规范扫描；
2. 支持针对前三节课的编码规范进行一键修复；
3. 总结完整的前端编码规范工程化流程，梳理如何在简历上体现优势，以及面试中相关的常见`Q&A`；