规范化标准

规范化是我们践行前端工程化中重要的一部分

规范化介绍

- 为什么要有规范标准
- 哪里需要规范化标准
- 实施规范化的方法

为什么要有规范化标准

- 软件开发需要多人协同
- 不同开发者具有不同的编码习惯和喜好
- 不同的喜好增加项目维护成本
- 每个项目或者团队需要明确统一的标准

哪里需要规范化标准

- 代码、文档、甚至是提交日志
- 开发过程中人为编写的成果物
- 代码标准化规范最为重要

实施规范化的方法

- 编码前人为的标准约定
- 通过工具实现 Lint

常见的规范化实现方式

- ESLint 工具使用
- 定制 ESLint 校验规则
- ESLint 对 TypeScript 的支持
- ESLint 结合自动化工具或者 Webpack
- 基于 ESLint 的衍生工具
- Stylelint 工具的使用

EsLint 介绍

ESLint 介绍

- 最为主流的 JavaScript Lint 工具 监测 JS 代码质量
- ESLint 很容易统一开发者的编码风格
- ESLint 可以帮助开发者提升编码能力

ESLint 安装

ESLint 安装步骤

- 初始化项目
- 安装 ESLint 模块为开发依赖
- 通过 CLI 命令验证安装结果

ESLint 快速上手

ESLint 检查步骤

- 编写“问题”代码
- 使用 eslint 执行检测
- 完成 eslint 使用配置

ESLint 配置文件解析

browser - 浏览器环境中的全局变量。
node - **Node.js** 全局变量和 **Node.js** 作用域。
commonjs - **CommonJS** 全局变量和 **CommonJS** 作用域 (用于 **Browserify/WebPack** 打包的只在浏览器中运行的代码)。
shared-node-browser - **Node.js** 和 **Browser** 通用全局变量。
es6 - 启用除了 **modules** 以外的所有 **ECMAScript 6** 特性（该选项会自动设置 **ecmaVersion** 解析器选项为 **6**）。
worker - **Web Workers** 全局变量。
amd - 将 **require()** 和 **define()** 定义为像 **amd** 一样的全局变量。
mocha - 添加所有的 **Mocha** 测试全局变量。
jasmine - 添加所有的 **Jasmine** 版本 **1.3** 和 **2.0** 的测试全局变量。
jest - **Jest** 全局变量。
phantomjs - **PhantomJS** 全局变量。
protractor - **Protractor** 全局变量。
qunit - **QUnit** 全局变量。
jquery - **jQuery** 全局变量。
prototypejs - **Prototype.js** 全局变量。
shelljs - **ShellJS** 全局变量。
meteor - **Meteor** 全局变量。
mongo - **MongoDB** 全局变量。
applescript - **AppleScript** 全局变量。
nashorn - **Java 8 Nashorn** 全局变量。
serviceworker - **Service Worker** 全局变量。
atomtest - **Atom** 测试全局变量。
embertest - **Ember** 测试全局变量。
webextensions - **WebExtensions** 全局变量。
greasemonkey - **GreaseMonkey** 全局变量

ESLint 配置注释

http://eslint.cn/docs/user-guide/configuring#configuring-rules

ESLint 结合自动化工具

- 集成之后， ESLint 一定会工作
- 与项目统一，管理更加方便

- https://github.com/zce/zce-gulp-demo.git
- 完成相应的依赖安装
- 完成 eslint 模块安装
- 完成 gulp-eslint 模块安装

ESLint 结合 Webpack

前置工作

- https://github.com/zce/zce-react-app.git
- 安装对应模块
- 安装 eslint 模块
- 安装 eslint-loader 模块
- 初始化 .eslintrc.js 配置文件

ESLint 结合 Webpack 后续配置

现代化项目集成 ESLint

ESLint 检查 TypeScript

衍生工具之 Standard

https://github.com/standard/standard

Stylelint 的认识

Stylelint 使用介绍

- 提供默认的代码检查规则
- 提供 CLI 工具，快速调用
- 通过插件支持 Sass Less PostCSS
- 支持 Gulp 或 Webpack 集成

Prettier 的使用

Git Hooks 介绍

代码提交至仓库之前未执行 lint 工作

通过 Git Hooks 在代码提交前强制 lint

Git Hooks 介绍

- Git Hook 也称之为 git 钩子，每个钩子都对应一个任务

Git Hooks 介绍

- Git Hook 也称之为 git 钩子，每个钩子都对应一个任务
- 通过 shell 脚本可以编写钩子任务触发时要具体执行的操作

ESLint 结合 Git Hooks

很多前端开发者并不擅长使用 shell

Husky 可以实现 Git Hooks 的使用需求

扫码联系老师

技能评估、福利资料、课程优惠

Made with ❤️ by LagouFed