

拉勾教育

— 互联网人实战大学 —

《前端高手进阶》

朱德龙 前中兴软创主任工程师

— 拉勾教育出品 —

第22讲：如何合理搭建前端项目

上一课时分析了前端构建工具 webpack 的底层原理

在理解原理之后再来探索构建工具的具体应用——如何合理搭建前端项目

本课时我们将从项目组织、代码规范 2 个方面来进行分析

场景描述

开发项目 `projectA` 的时候

发现其中的 `codeX` 也可以用于项目 `projectB`

最简单直接的处理方式就是把 `codeX` 的代码直接复制到 `projectB` 下

按照“三次原则”处理方式没什么问题

但如果此时项目 `projectC` 和 `projectD` 也会用到 `codeX`

那么这种方式维护起来会很麻烦

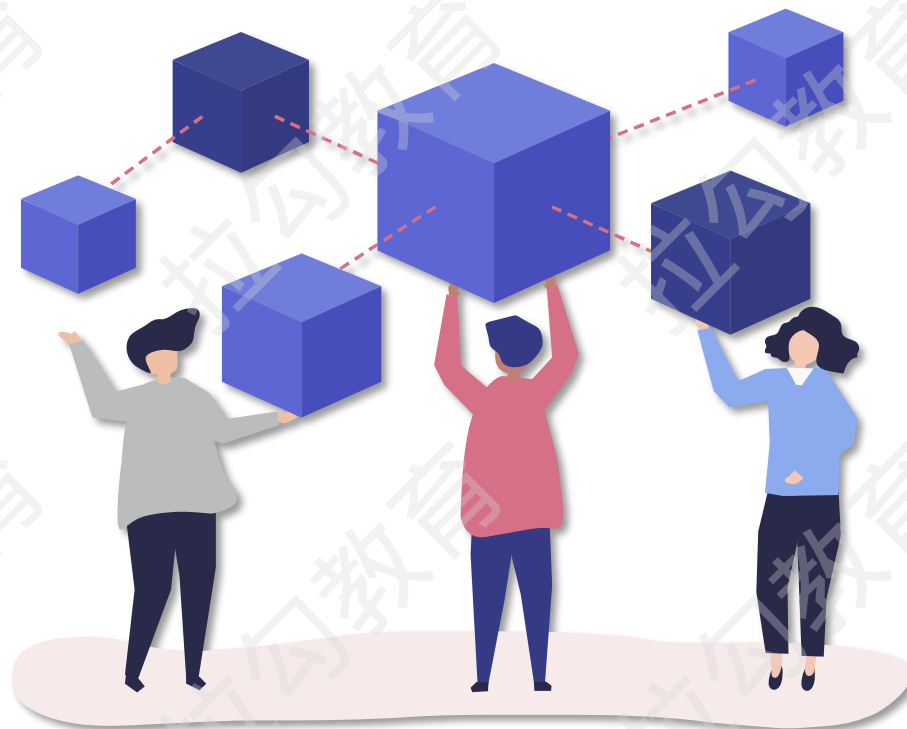
项目组织

multirepo

multirepo 就是将项目中的模块拆分出来
放在不同的仓库中进行**独立管理**

拉勾教育

— 互联网人实战大学 —

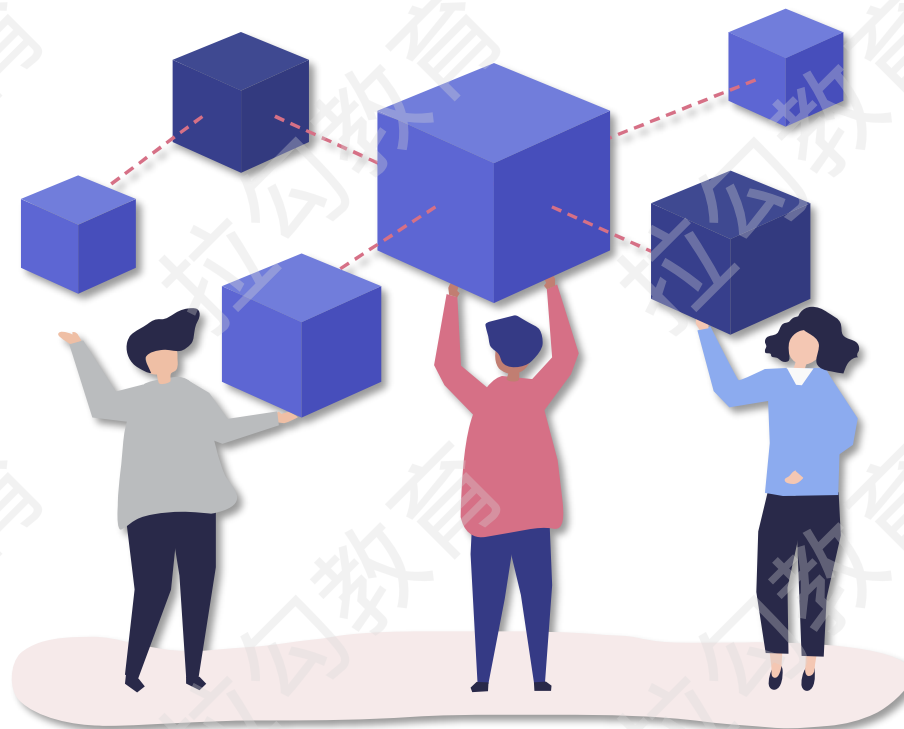


multirepo

这种方式可以保证仓库的独立性

方便**不同团队**维护对应的仓库代码

可以根据团队情况选择擅长的工具、工作流等



项目组织

multirepo

- 开发调试及版本**更新效率低下**
- 团队**技术选型分散**

拉勾教育

— 互联网人实战大学 —



项目组织

拉勾教育

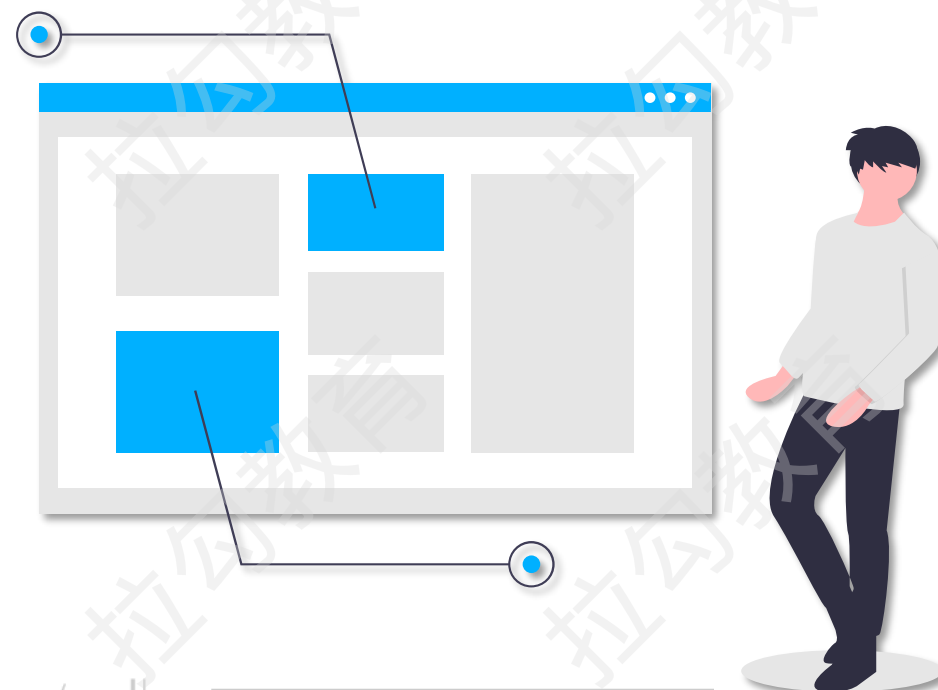
— 互联网人实战大学 —

monorepo

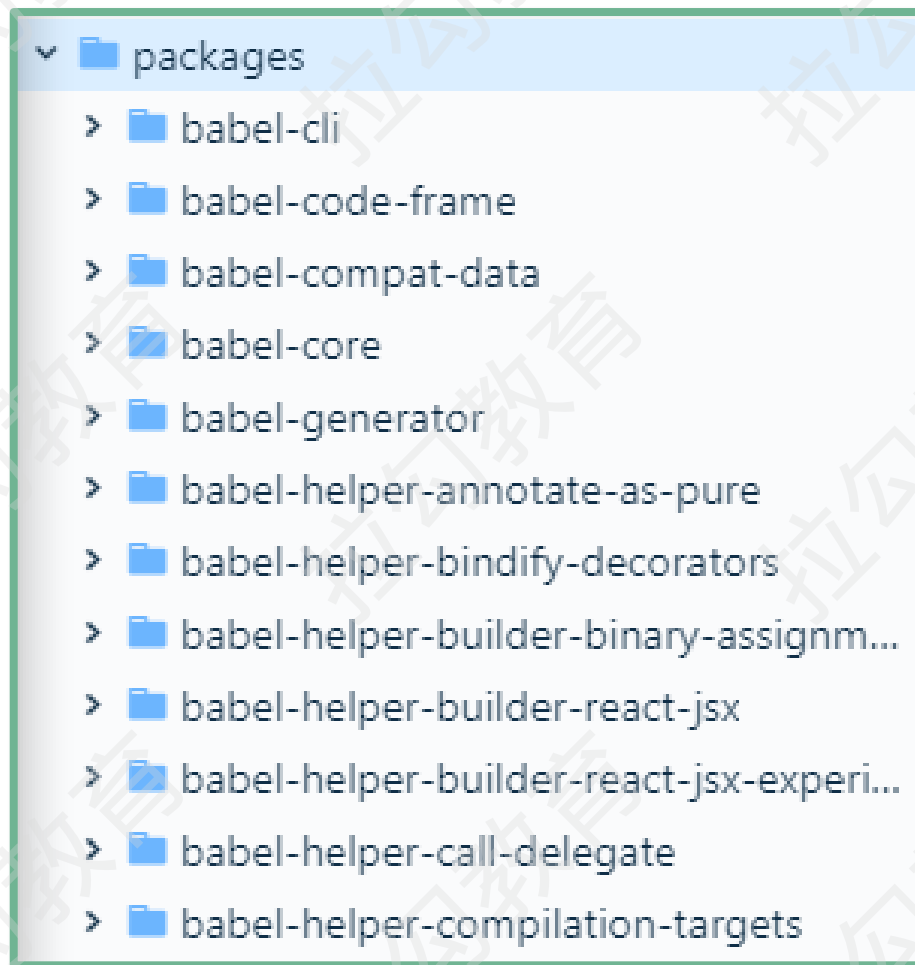
monorepo 将所有相关的模块放在同一个项目仓库中

这种方式在管理上会更加方便

项目所有代码可以使用统一的规范及构建、测试、发布流程
















monorepo



babel 的依赖模块

monorepo

- >  babel-helper-builder-react-jsx
- >  babel-helper-builder-react-jsx-experi...
- >  babel-helper-call-delegate
- >  babel-helper-compilation-targets
- >  babel-helper-create-class-features-pl...
- >  babel-helper-create-regexp-features-...
- >  babel-helper-define-map
- >  babel-helper-explode-assignable-exp...
- >  babel-helper-explode-class
- >  babel-helper-fixtures
- >  babel-helper-function-name
- >  babel-helper-get-function-arity
- >  babel-helper-hoist-variables

babel 的依赖模块

monorepo

开源工具 **lerna** 是一个用于管理带有多个包的 JavaScript 项目工具

- **packages** 目录
- **lerna.json** 文件
- **package.json** 文件



monorepo

Fixed/Locked 模式为默认模式

- 开发者执行 lerna publish 后
lerna 会在 lerna.json 中找到指定 version 版本号
- 如果这一次发布包含某个项目的更新
会自动更新 version 版本号
- 任何一个项目大版本升级，其他项目的大版本号也会更新



monorepo

Independent 模式各个项目都相互独立

开发者需要独立管理多个包的版本更新

每次发布时 lerna 会配合 Git 检查相关包文件的变动

只发布有改动的 package



monorepo

- 导致仓库**体积变大**
- 目录结构也会变得更**复杂**
- 需要**额外的工具**来管理各个模块



代码规范

拉勾教育

— 互联网人实战大学 —

- 少用全局变量
- 高内聚、低耦合
- 遵循单一原则
- 拥有注释说明



“风格一致”就是让参与项目开发的工程师形成一种开发上的契约
从而降低维护成本

可以从代码编写和代码管理两个方向入手
分别对应编写规范和提交规范

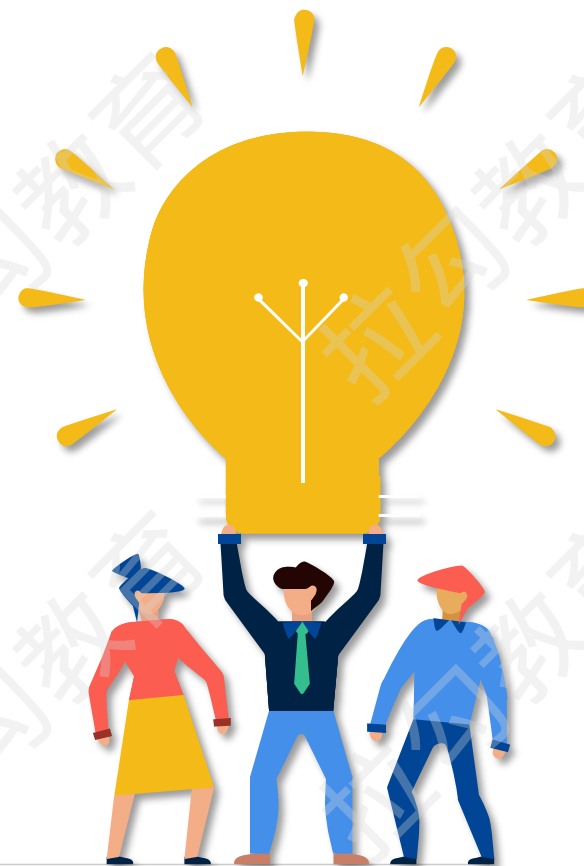


一般大型互联网公司都会制定自己的编写规范

比如 Google 的 JavaScript 风格指南、Airbnb 风格指南

以 JavaScript 为例

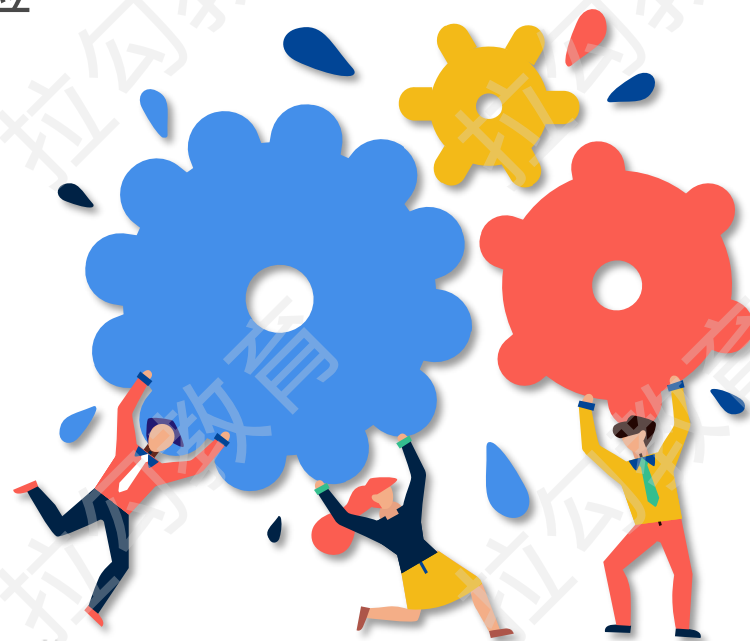
比如 JSLint、JSHint、JSCS、ESLint 等多种规则校验工具



- 可执行

制定编写规范首先要保证的就是规范的可执行性

建议编写规范中的每一条规则都能有对应的校验工具规则与之对应



• 可配置

代码的可读性有时候是一个比较主观的问题

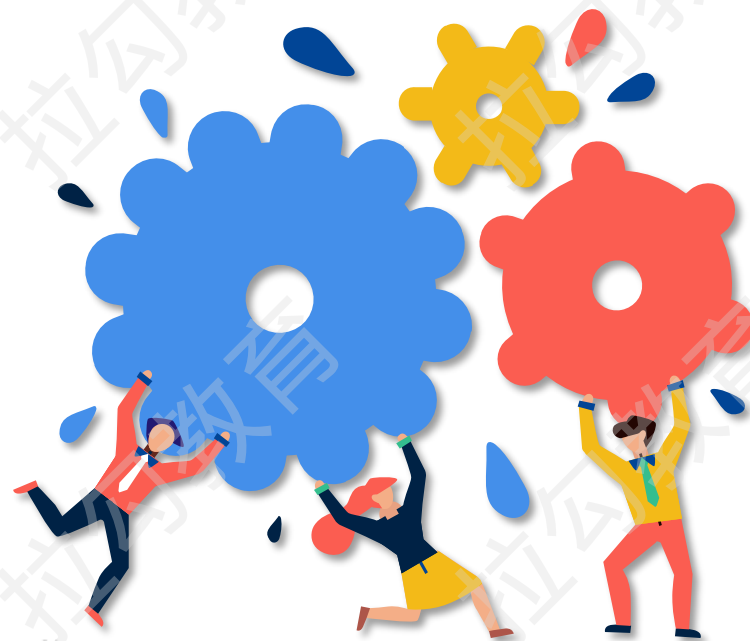
使用具有丰富配置项的代码校验工具就可以很轻松地解决这些分歧



- 可扩展

当校验工具的已有配置规则无法支持项目需求时

可以自行编写插件来扩展校验规则



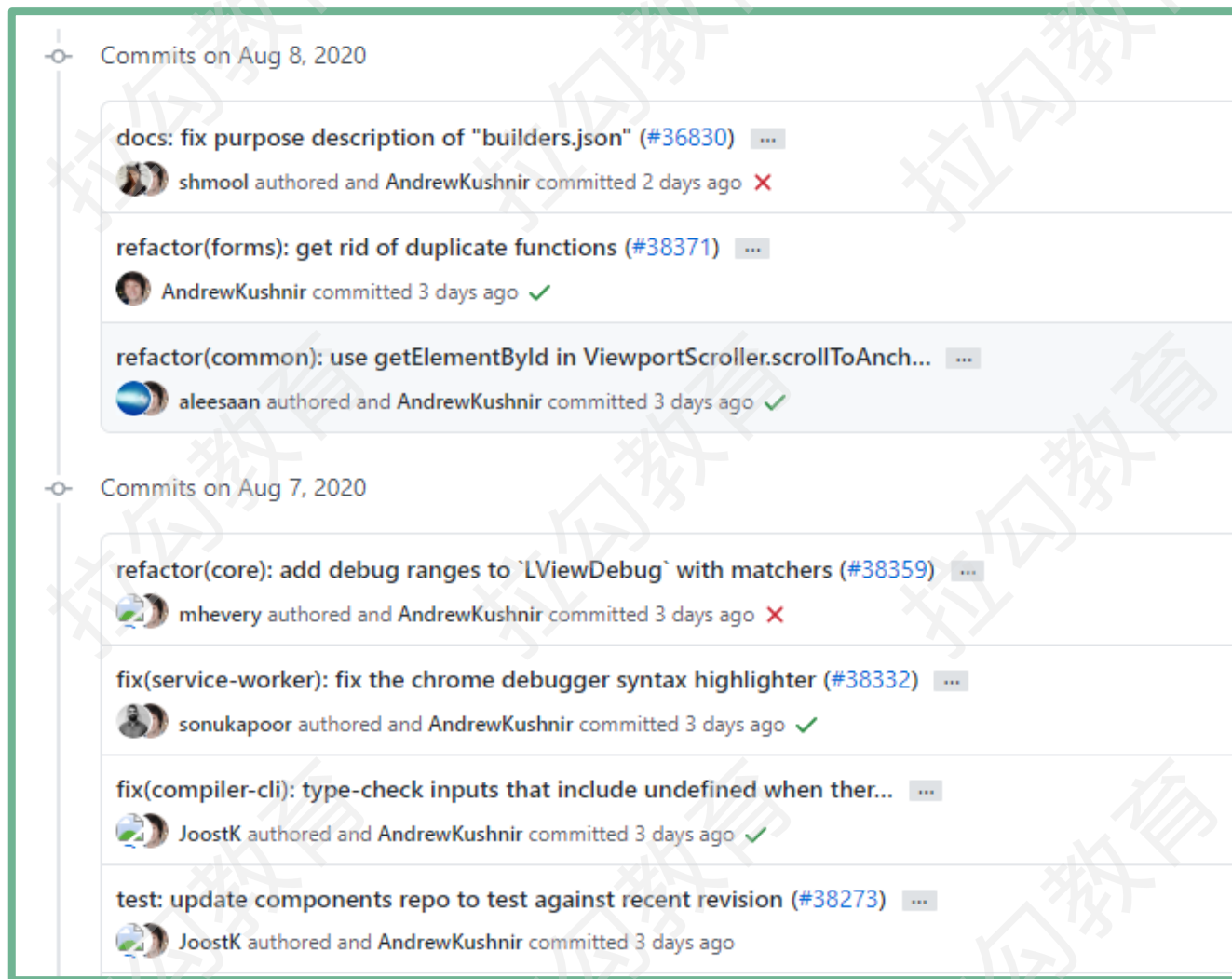
ESlint 就可以满足可配置、可扩展的原则

它的核心功能是通过一个叫 `verify()` 的函数来实现的










该函数有两个必传参数：

要验证的源码文本和一个配置对象





Angular 的提交日志

	JoostK authored and AndrewKushnir committed 3 days ago
refactor(core): extract `icuSwitchCase`, `icuUpdateCase`, `removeNest...` ...	
	mhevery authored and AndrewKushnir committed 3 days ago ✖
refactor(core): add human readable `debug` for i18n (#38154) ...	
	mhevery authored and AndrewKushnir committed 3 days ago
docs(service-worker): describe how asset-/data-group order affects re... ...	
	gkalpak authored and AndrewKushnir committed 4 days ago ✔
fix(docs-infra): correctly generate CLI commands docs when the overvi... ...	
	gkalpak authored and AndrewKushnir committed 4 days ago ✖
fix(compiler-cli): mark eager `NgModuleFactory` construction as not s... ...	
	dgp1130 authored and AndrewKushnir committed 4 days ago ✔
refactor(compiler): add `ModuleInfo` interface (#38320) ...	
	dgp1130 authored and AndrewKushnir committed 4 days ago
refactor(core): add `noSideEffects()` as private export (#38320) ...	
	dgp1130 authored and AndrewKushnir committed 4 days ago
docs: remove https://angular.io from internal links (#38360) ...	
	ajitsinghkalder authored and AndrewKushnir committed 4 days ago ✔

Angular 的提交日志

提交规范

拉勾教育

— 互联网人实战大学 —

Git 自带 template 功能

这个功能可以定义一个提交消息的模板文件
然后通过 git config 命令指向这个模板文件
但**不具有强制性**



提交规范

拉勾教育

— 互联网人实战大学 —

推荐使用工具 **@commitlint/cli** 和 **husky**

commitlint 可以设置提交消息模板并校验

husky 可以设置 pre-commit 钩子

在提交代码时调用 commitlint 进行强制校验

避免生成不符合规范的提交消息



```
// .huskyrc
{
  "hooks": {
    "pre-commit": "npm test",
    "commit-msg": "commitlint"
  }
}
```

```
// package.json
{
  ...
  "lint-staged": {
    "*.js": [
      "eslint --fix",
      "git add"
    ]
  },
  ...
}
```

- 在**项目组织**上

相关性低的模块可以采用 multirepo 方式进行独立管理

相关度高的模块则可以采用 monorepo 方式对其进行集中管理



- 在**制定代码规范**时

对于编写规范尽量做到可执行、可配置、可扩展

对于提交规范可以选择适当的工具

来保证提交消息的规范化和可读性



能对前端项目搭建有更深入的理解和思考

在空间维度和时间维度上来考虑如何组织项目代码和规范项目代码



你在开发中还用到了哪些代码校验工具

它们都有什么特点



Next：第23讲：《谈性能优化到底在谈什么》

拉勾教育

— 互联网人实战大学 —



下载「拉勾教育App」
获取更多内容