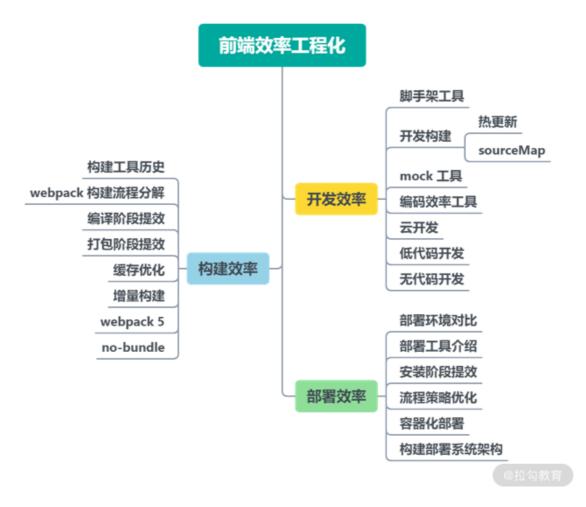
一手资源尽在:666 java.co

本文由 <u>简悦 SimpRead</u> 转码,原文地址 <u>kaiwu.lagou.com</u>

你好,我是李思嘉。

本专栏的内容到这里就结束了。我们先来简单回顾一下整个课程的主要内容,如下图:



在这个专栏中,我主要介绍且梳理了前端工程化中效率提升方向的知识,内容涵盖开发效率、构建效率 和部署效率三个方面。希望你通过这个系列课程的学习,能建立起前端效率工程化方面相对完整的知识 体系,同时在前端开发日常流程中的效率工程类问题方面,能找到分析和解决的新方向。

当然,这些方向实际涵盖的概念与技术点非常广泛,并不容易完全掌握,除了已有的概念和技术之外,新的技术和方向也在不断涌现。下面我会对前端效率工程化相关的技术做一些展望。

云工作流

开发效率方面,由 Web IDE 发展而来的云开发工具,目前正逐渐成为几个大型厂商探索的方向之一。 从开发到部署的完整云端工作流方式在未来可能成为标准。

AI 生成页面

和目前依赖设计资源与开发基建的低代码或无代码开发相比,基于 AI 的生成页面工具(例如微软的 Sketch2Code)可以进一步解放生产力。目前,无论是基于设计稿(Sketch/PSD)的精准生成方式,还是基于草图乃至描述语句的 AI 匹配生成方式,仍有各自的局限性。但是随着 AI 技术的发展,这类产品可能会替代目前前端开发过程中的一些基础重复性工作。

Go/Rust

一手资源尽在:666 java.co

在构建效率方面,Webpack 5 带来了更完整的缓存策略和代码优化策略,但是从底层性能上,构建工具本身的性能仍然受到 NodeJS 自身语言的限制。从 esbuild 工具的思路出发,基于 Go、Rust 等高性能语言的编译工具在未来或许能成为性能突破点之一。

No bundle & HTTP/3

构建效率另一个方向的发展来自无包构建。尽管无包构建工具在生产环境下仍然采用打包构建的方式,但随着网络技术(例如 HTTP/3)的发展,或许最终可以在生产环境下同样采用无包构建。此外,渐进式的使用方式(例如 vendor 部分打包而源码部分无包使用)也可能很快成为可实现的方向。

总结

希望我的专栏内容和对未来趋势的展望能对你有所帮助。学习时,你需要在日常工作中不断实践,结合具体的场景尝试可行的优化方案。

由于自身在团队中的职责不同,每个开发人员索要学习和侧重的点也不同:

- 如果你目前主要做的是具体项目的开发维护工作,那么分析现有项目的构建工具、构建配置就是一个很好的入手点。通过构建效率模块的相关课程,相信你可以比之前更全面地进行分析和优化。
- 如果你目前承担着多个项目的选型与架构工作,希望开发效率模块的一些视角可以为你带来思路。
- 如果你目前从事的是前端基础建设的相关工作,希望系列课程中提到的一些新的开发、构建和部署工具也能为你提供一些着手方向。

前端工程化是一个系统工程,不同的开发人员都可以在团队中找到适合的位置。通过不断地实战开发和经验积累,相信你可以逐步提升对技术的认识,增强技术实力。

如果你觉得课程不错,不要忘了推荐给身边的朋友。前路漫漫,一起加油!

最后,我邀请你参与对本专栏的评价,你的每一个观点对我们来说都是最重要的。<u>点击链接,即可参与评价</u>,还有机会获得惊喜奖品!