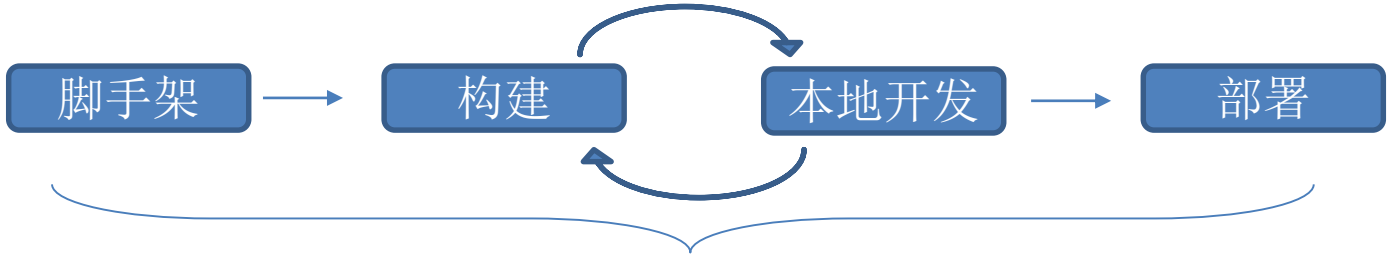
前端工程化介绍

汇报人：云程

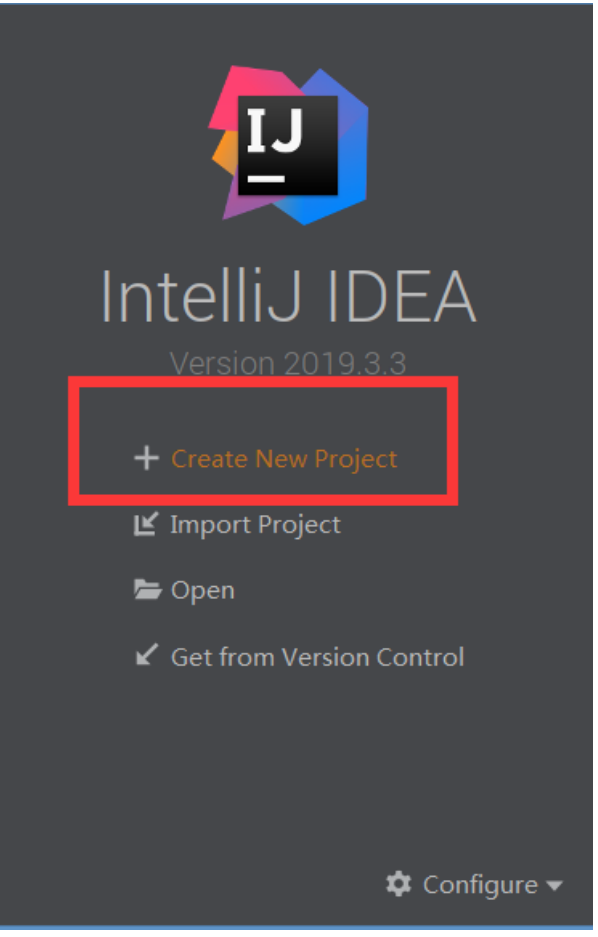
前言 Preface

1. 本次介绍的上下文是**”前端”**

2. 围绕工程化的主题, 主要论述如下几个方面:

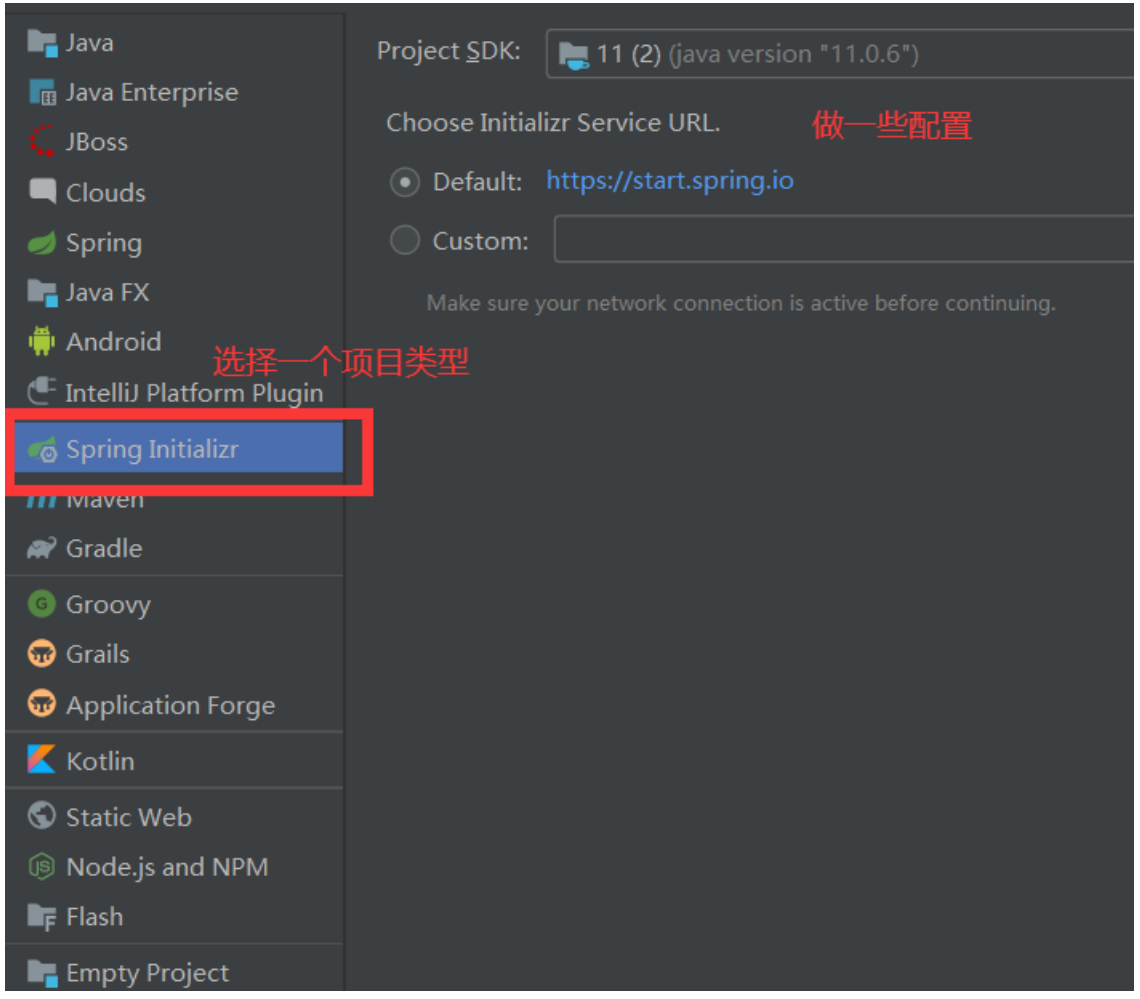


什么是脚手架

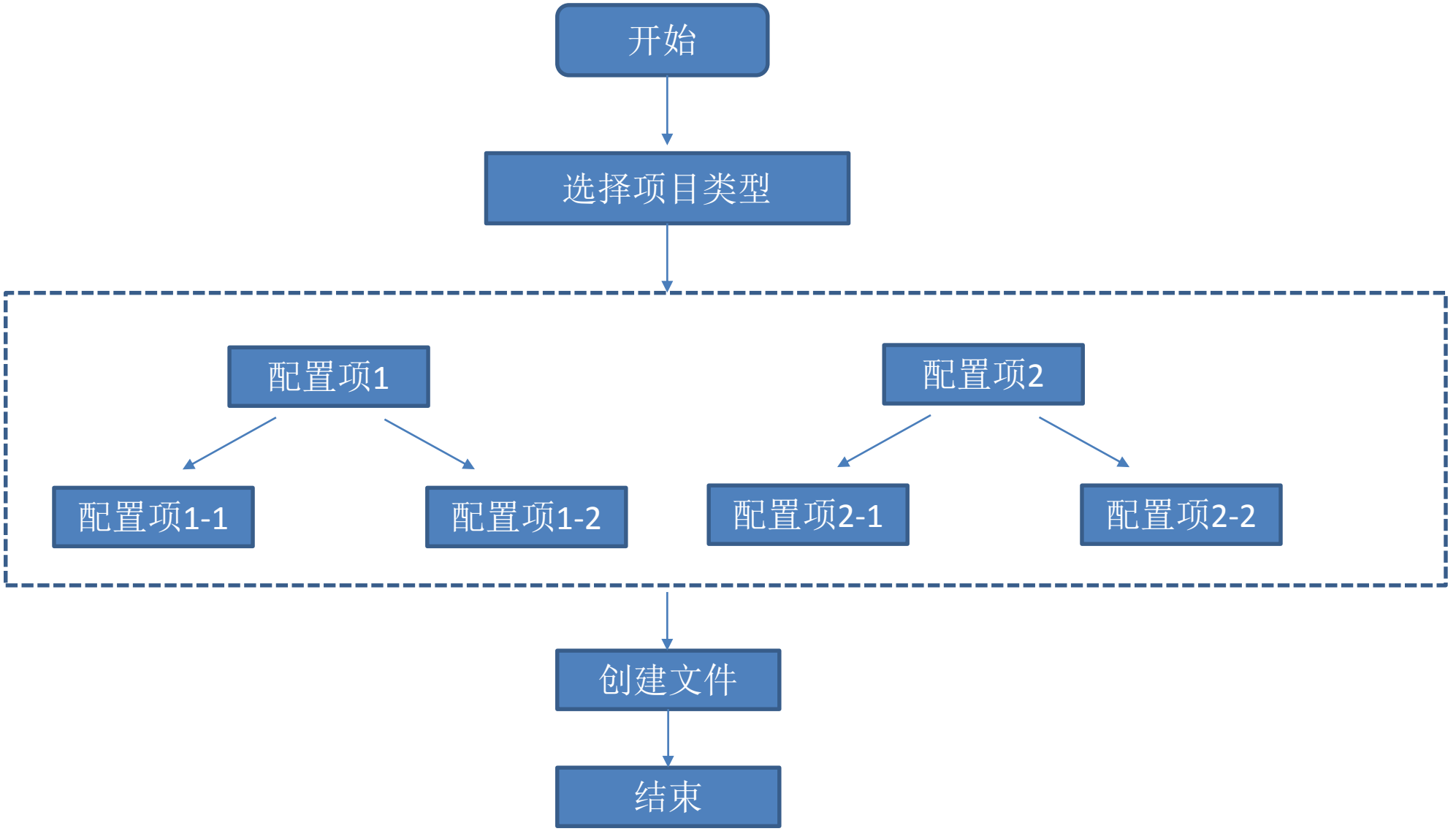


Hello world!

启动~



一个项目的创建

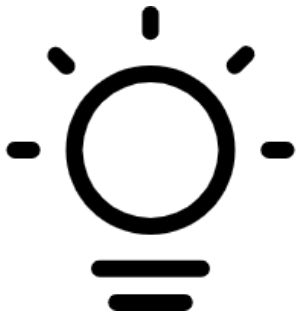


脚手架的本质

创建项目的初始文件

为什么使用脚手架

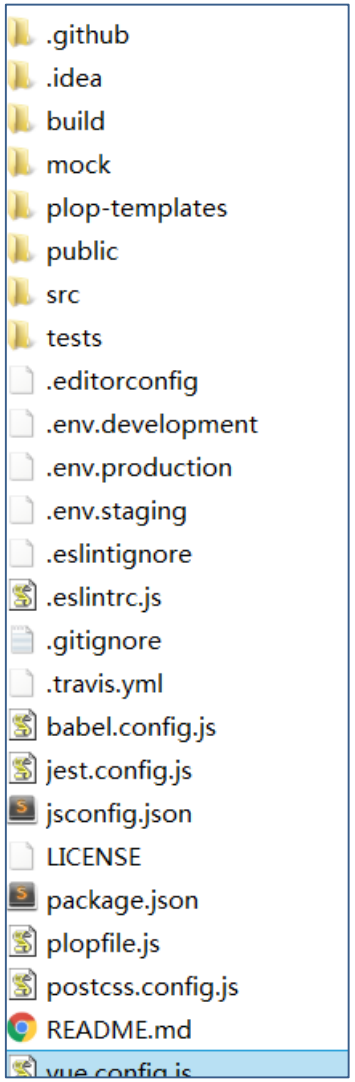
一个栗子

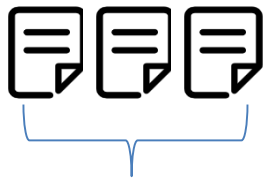


Hello world

最潮的框架

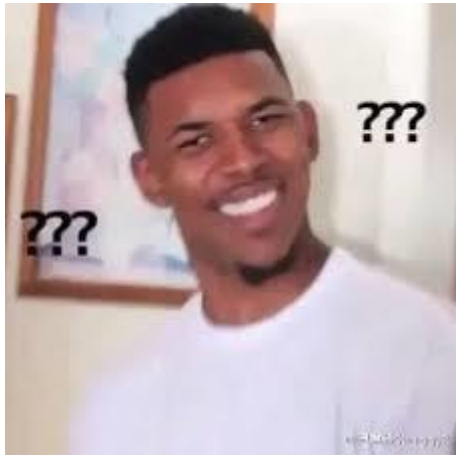
最靓的UI





10G doc

无题



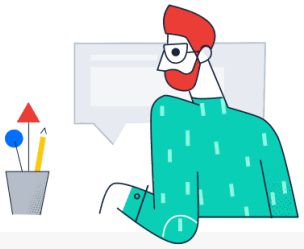
我只想写一个hello world呀

我这有本武林秘籍

CLI

- 一键启动
- 目录清爽
- 配置少
- 快速开始

脚手架工具哪里来



资源汇总

官方工具

- vue-cli
- create-react-app

git repo

你可以新建一个git仓库,依据你自己的经验组织好目录, 配置好初始化文件,别人git clone就好

★ Vue-admin-template

生成器

- cookiecutter
- yeoman

总结

脚手架是一个单独衡量并没有很大价值, 但却是前端工程体系不可或缺的功能模块, 它可以降低工程体系,框架的学习成本, 统一小组成员的开发入口(规范), 方便我们集中精力完成业务逻辑,而不是陷入各种技术细节框架杂糅的汪洋大海

构建是做什么

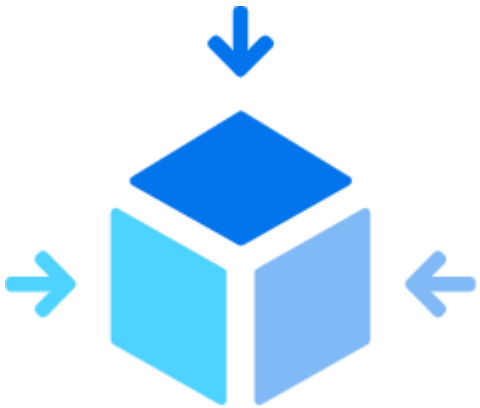
后台语义下的构建

编译源代码

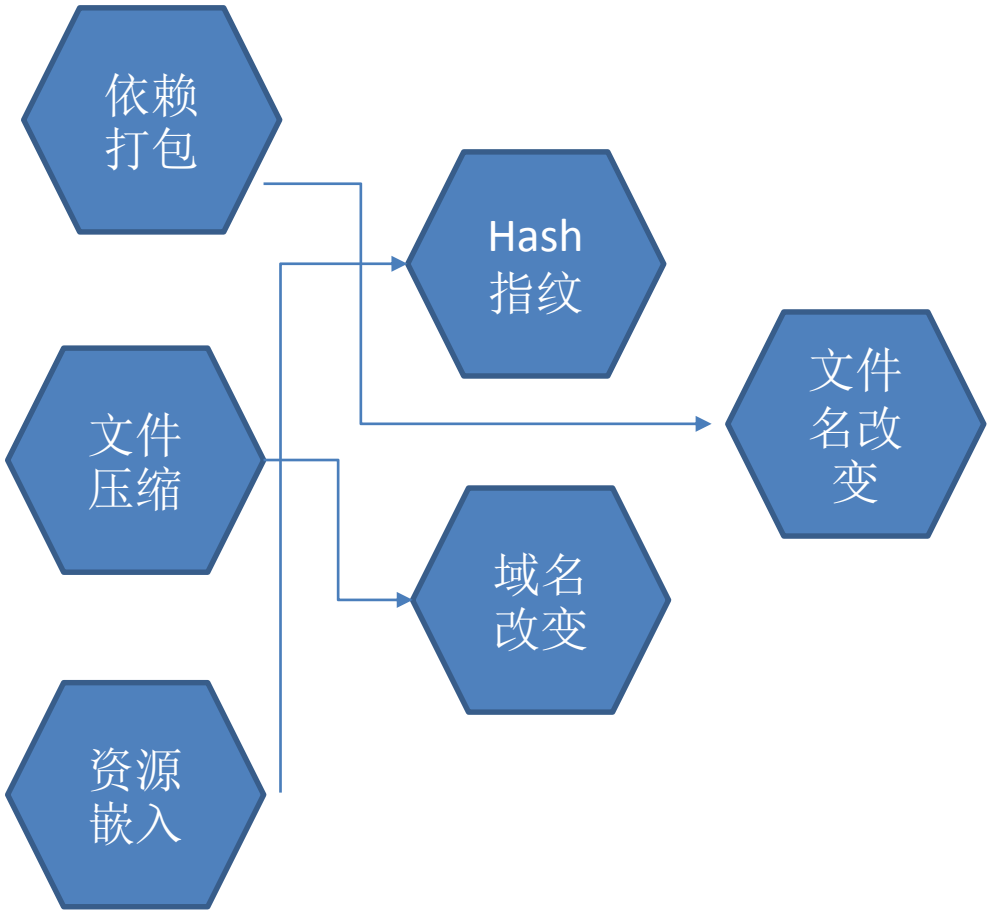
前端语义下的构建

- ES规范的转义
- CSS预编译器支持
- PostCSS处理浏览器后缀
- 代码,图片压缩
- 小体积图片的base64转码
- JavaScript模块化规范支持

● ● ●



构建解决的问题



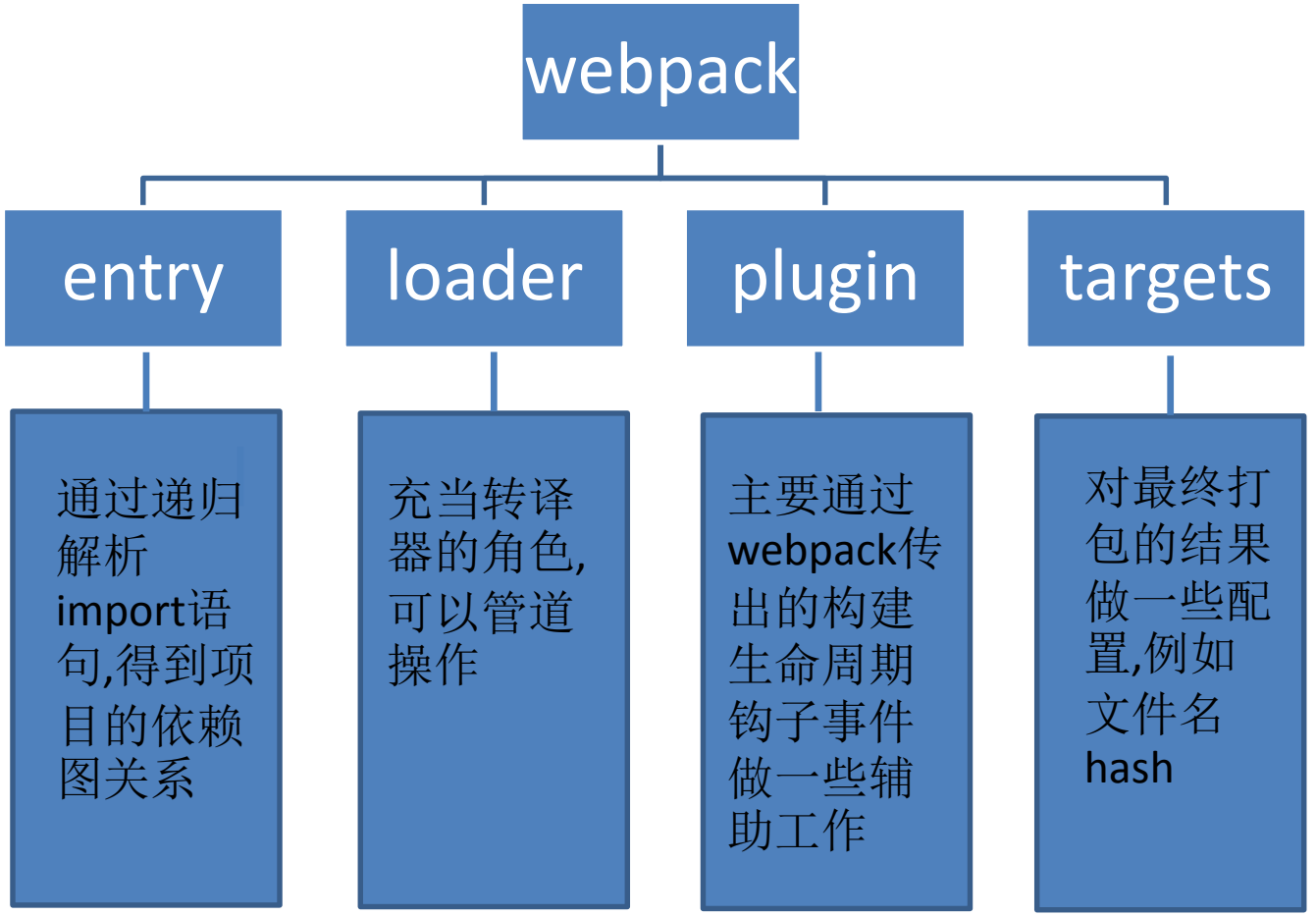
1. 面向语言
2. 面向优化
3. 面向部署

JavaScript的历史

JavaScript最早是在网景公司的netscape浏览器上运行的一门脚本语言,但是后来各浏览器公司因为商业原因争相实现自己的标准,导致JavaScript生态的大分裂,开发者常常为了兼容不同平台而疲惫不堪, 好在1996年网景公司向Ecma国际提交了js的标准化提案, 被称为ECMA-462标准, 是之后被发布的ES1标准的原型,以此来对js规范化. 时间过去, 语言也在不断发展, 现在已经有了ES10, 这些标准主要是添加一些现代化的语法特性,然而浏览器版本的大跨度, 兼容性的问题依然存在, 由此诞生了**babel,他负责将js源代码转译降级**以适应未实现新特性的浏览器

如何解决构建问题

Webpack, 让一切变简单！



本地开发

本地开发面临的问题

Mock服务

现代web开发基本是前后端分离的,为了二者进度能够独立, 前端需要自己的mock server

动态构建

前端开发主要面对的是UI需求,需要频繁的调试, 为了节省人力,需要能够监听源文件改动然后自动重新构建

Mock服务

Mock的发展进程

1.假数据

假数据的普遍用法是在业务代码中声明一个变量,代替接口返回的数据

2.客户端Mock

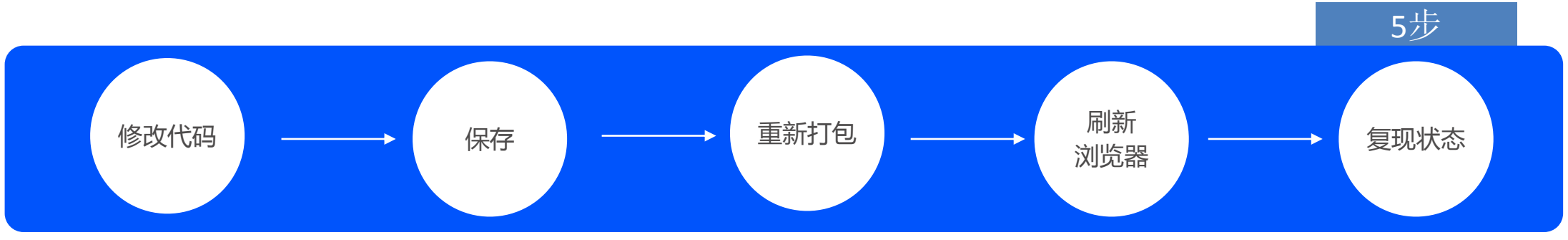
原理主要是通过重写浏览器的XHR对象来拦截ajax请求并返回自定义数据

3. Mock Server

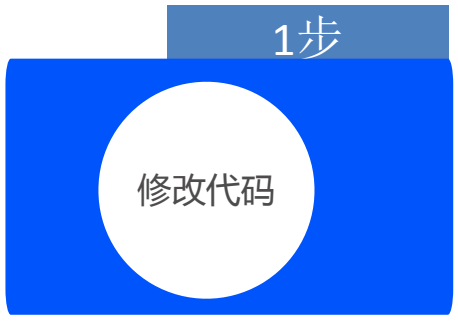
使用独立的服务器,不侵入业务代码,本质是一个简化的web server , 至于路由匹配后如何分发数据就很灵活了

动态构建

没有动态构建之前



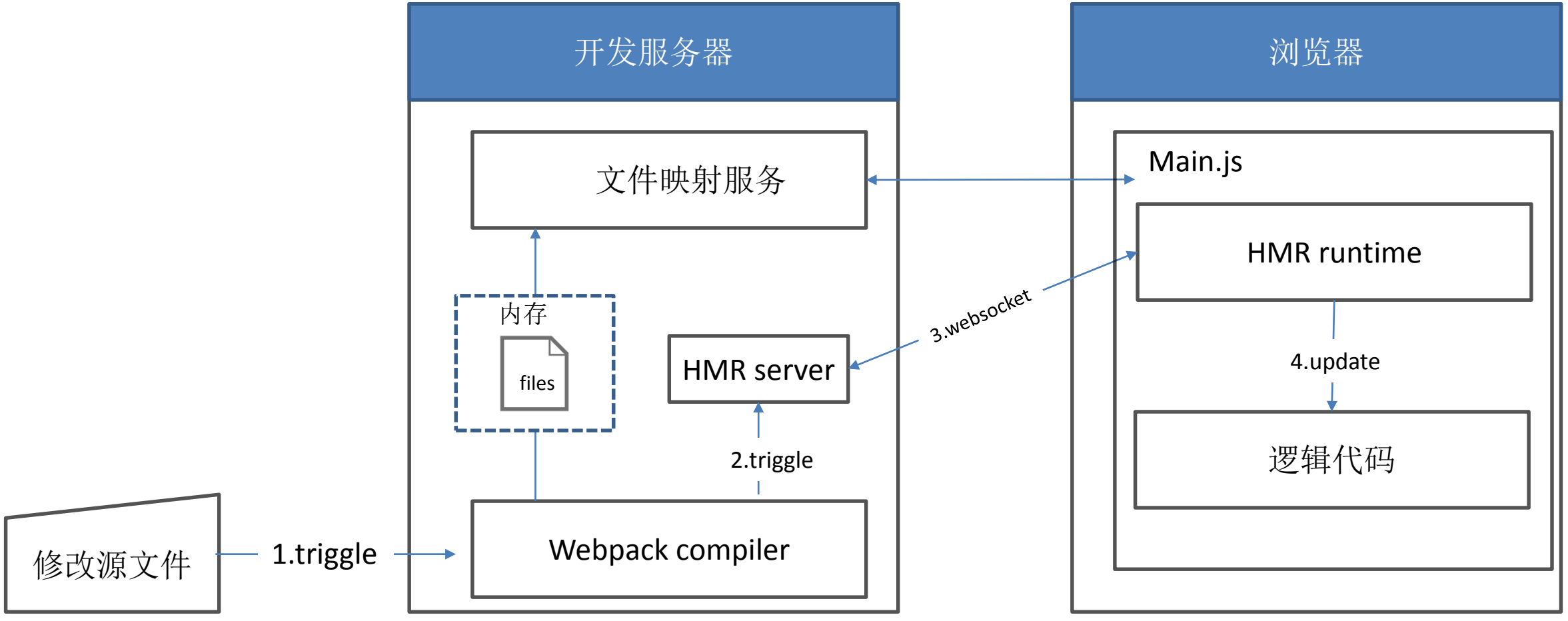
有了动态构建



模块热替换 → 所见即所得

动态构建

HMR工作原理



部署

常见部署方式

方案一

使用FTP工具

好处　工具都是现成的,不需要额外的开发学习成本

坏处　无法持续集成

方案二

jenkens配合git hook

好处　持续集成 版本控制 权限控制

坏处　需要一定的基础设施及配置

总结:

使用FTP工具比较灵活简单, 配合docker对于小项目小团队是足够的, jenkens我个人使用不多, 体验是属于一劳永逸的方案,适合多人协作

谢谢观看