



Micro-Frontend

精致化的微前端开发之旅



目录

CONTENTS

01

微前端概念解读

02

微前端快速入门

03

Web Components

04

三大框架微前端

05

关于未来的思考

微服务平台和微前端对比

	微服务 服务端	微前端 前端	备注
服务	独立服务, 比如交易服务	应用或者模块, 比如导航	服务是不会相互影响的
服务治理	服务注册/发现/依赖管理/跟踪/降级/限流/日志/监控/运维等	应用的发现/路由/监控/降级/运行/注销/聚合等	需要一个/多个系统统一处理一些上层的事情
服务通信	HTTP / RPC/ 中间件	eventBus/sharedWorker/BroadcastChannel/LocalStorage	

01

微前端概念解读

微前端 (Micro-Frontend) ，是将微服务 (Micro-Services) 理念应用于前端技术后的
相关实践，使得一个前端项目能够经由多个团队独立开发以及独立部署。

01 微前端开发的特性



技术无关

各个开发团队都可以自行选择技术栈，不受同一项目中其它团队影响；



代码独立

各个交付产物都可以被独立使用，避免和其它交付产物耦合；



样式隔离

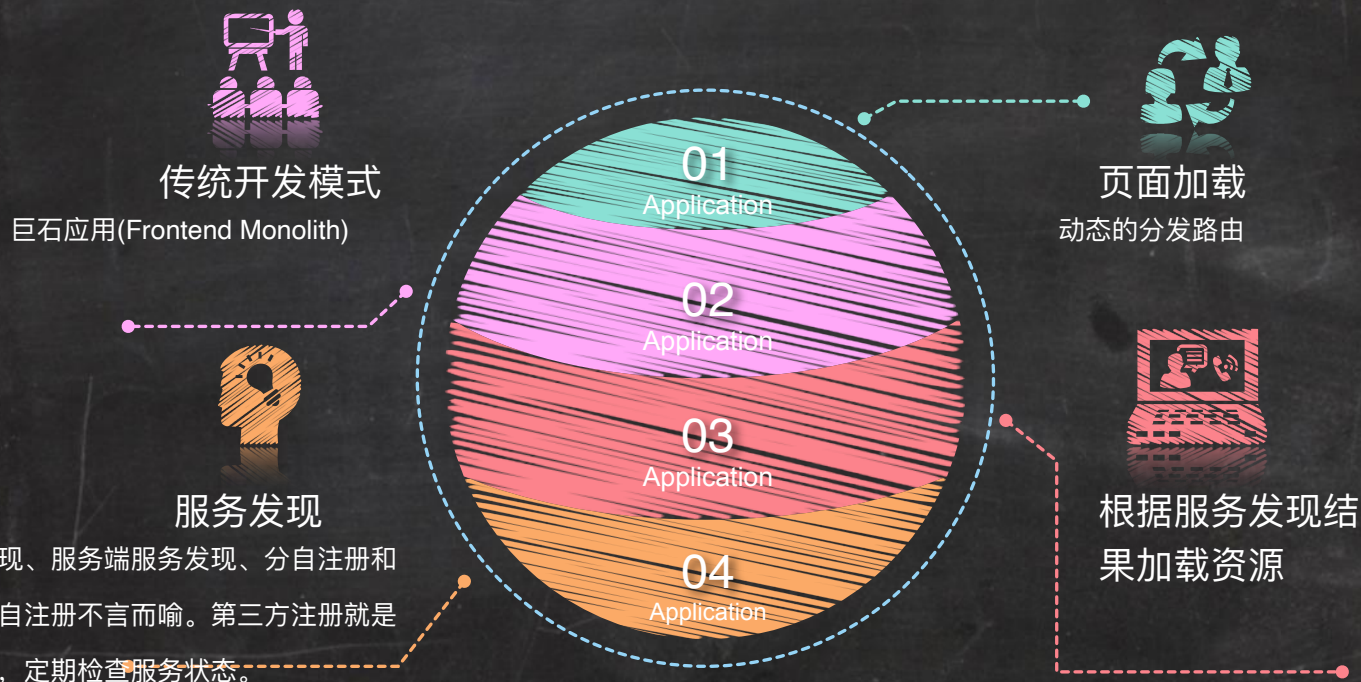
各个交付产物中的样式不会污染到其它组件；



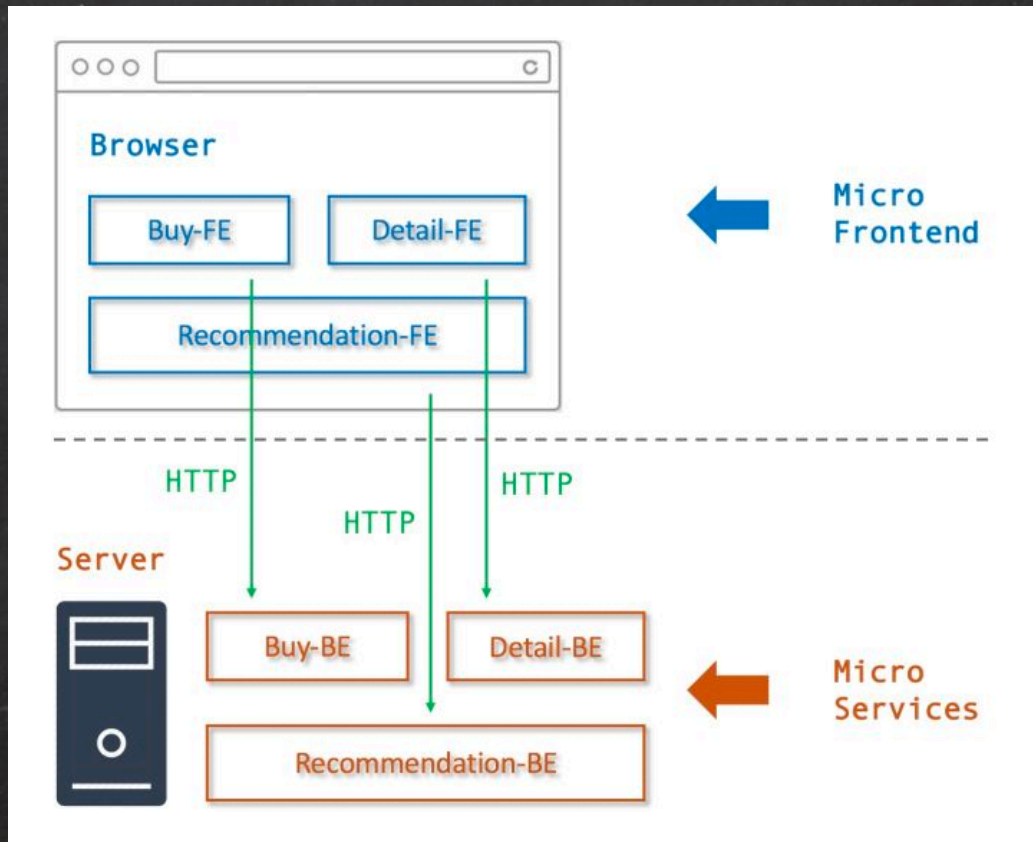
原生支持

各个交付产物都可以自由使用浏览器原生 API，而非要求使用封装后的 API；

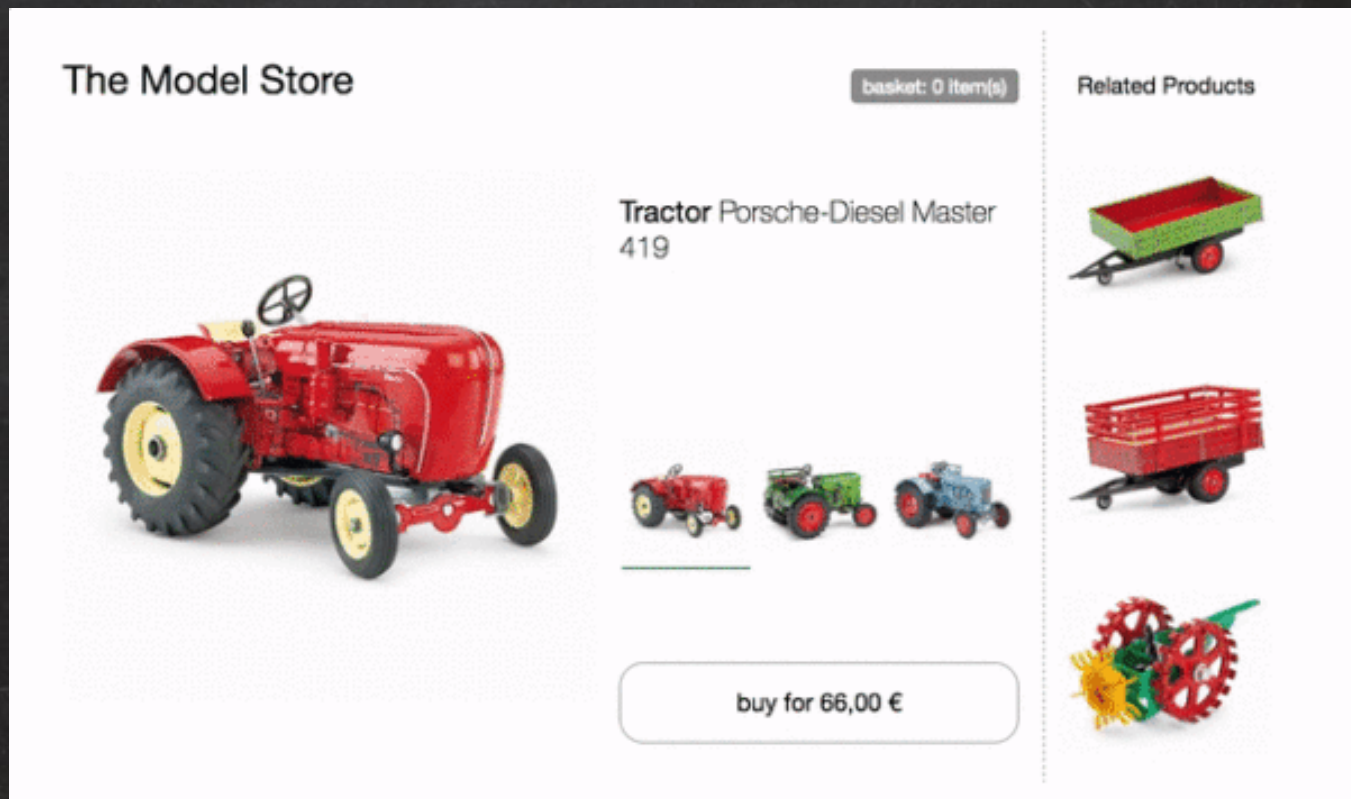
02 微前端核心解决问题和步骤



03 开发示意图




04-1 开发效果图




04-2 开发效果图

The Model Store






Tractor Porsche-Diesel Master
419





buy for 66,00 €


basket: 0 item(s)

Related Products

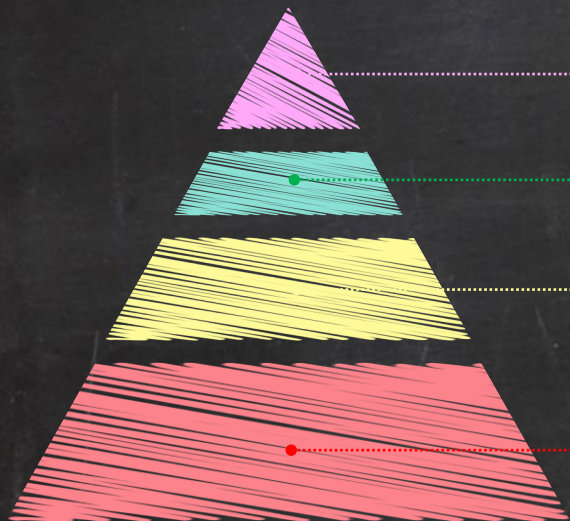


Team Product 

Team Checkout 

Team Inspire 

05 iframe现存问题 (改造成本低, 可以快速上线)



不可控制

iframe嵌入的显示区大小不容易控制, 存在一定局限性。

bfcache!

URL的记录完全无效, 页面刷新不能够被记忆, 刷新会返回首页, iframe功能之间跳转也无效

兼容性坑

iframe的样式显示、兼容性等都具有局限性

性能开销

iframe 阻塞 onload、占用连接池、多层嵌套页面崩溃。。

06 必须要解决的问题

一个前端需要对应多个后端

01

02

提供一套应用注册机制，完成应用的无缝整合

在应用之前团队开发者要制定好使用CSR或SSR的技术方案

04

03

构建时集成应用和应用独立发布部署



KEY WORD

07 来具体说说 -微前端具体要解决好的 10 个问题

- 👁 微应用的注册、异步加载和生命周期管理；
- 👁 微应用之间、主从之间的消息机制；
- 👁 微应用之间的安全隔离措施；
- 👁 微应用的框架无关、版本无关；
- 👁 微应用之间、主从之间的公共依赖的库、业务逻辑(utils)以及版本怎么管理；
- 👁 微应用独立调试、和主应用联调的方式，快速定位报错（发射问题）；
- 👁 微应用的发布流程；
- 👁 微应用打包优化问题；
- 👁 微应用专有云场景的出包方案；
- 👁 渐进式升级：用微应用方案平滑重构老项目。

05 微前端交付产物



发布静态资源+后台路由和服务



发布组件启动时机全由父级决定



发布局部应用配置过程由自身决定

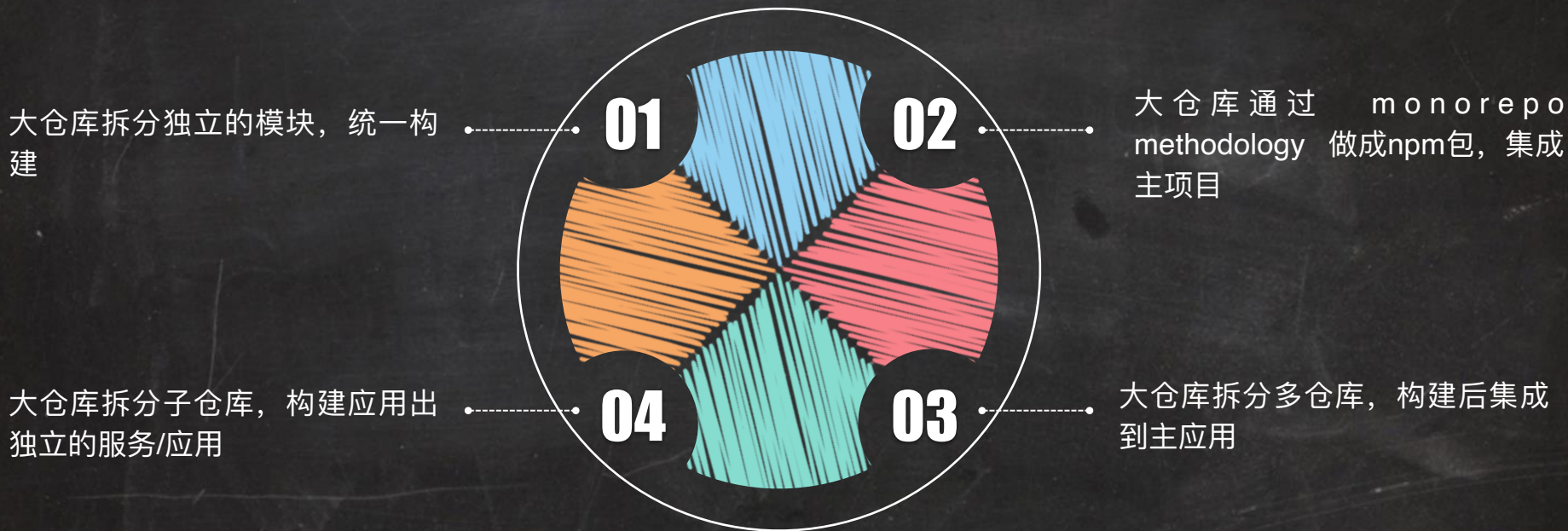
02

前端静态+后端

每个项目独立通过代码版本管理库独立分组、统一技术方案，合成整体技术架构。

<https://micro-frontends.org/>

常见的部署方式



01 FIS从入门到放弃

FIS

FIS3, 为你定制的前端工程构建工具

解决前端开发中自动化工具、性能优化、模块化框架、开发规范、代码部署、开发流程等问题. 我们已FIS为栗子🌰切入进微前端。



03

前端独立发包

通过A站点请求B站点独立的组件，全部控制权在A站点内。



Vue

一套用于构建用户界面的渐进式框架。Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，易于上手。



React

在数据改变时 React 也可以高效地更新渲染界面。以声明式编写 UI，可以让你的代码更加可靠，且方便调试。



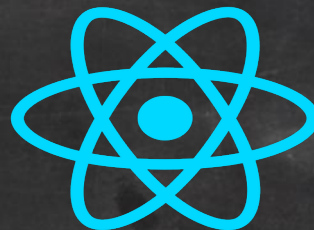
Web Components

它由四项主要技术组成，它们可以一起使用来创建封装功能的定制元素，可以在你喜欢的任何地方重用，不必担心代码冲突。



Angular

有着诸多特性，最为核心的是 MVW、模块化、自动化双向数据绑定、语义化标签、依赖注入等等。



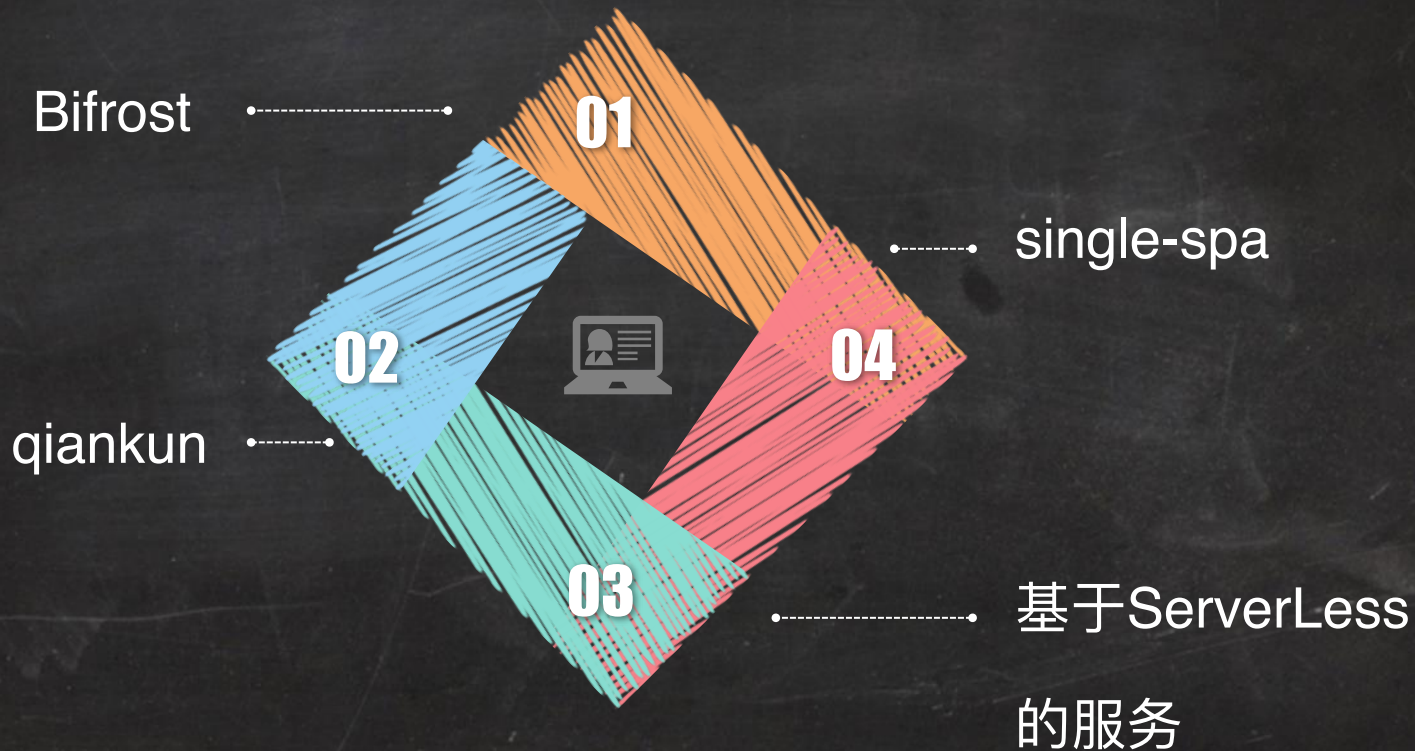
Webpack

JS应用程序的静态模块打包器。当处理应用程序时，会递归地构建一个依赖关系图，将所有这些模块打包成一个或多个 bundle。

深入理解Webpack

systemjs 是一个最小系统加载工具，用来创建插件来处理可替代的场景加载过程，包括加载 CSS 场景和图片，主要运行在浏览器和 NodeJS 中。它是 ES6 浏览器加载程序的扩展，将应用在本地图像浏览器中。通常创建的插件名称是模块本身，要是没有特意指定用途，则默认插件名是模块的扩展名称

目前比较流行的库



04

总线注册机制

通过开发期、构建期、部署期、运行期完善整体系统

01 微前端构建类单页的业务系统

父项目：映射多个后端服务、统一登录鉴权、获取全局菜单树

子项目：项目菜单、权限、角色隔离

运行期

父项目：重新生成引用文件、更新全局入口文件、重启服务

子项目：替换资源文件、调用主项目更新

部署期

父项目：生成全局路由、生成全局入口引用

子项目：替换公用库依赖、CSS添加作用域、生成项目静态资源

构建期

父项目：路由控制、模块加载、JS公用库替代、数据流注册、作用域

子项目：注册项目作用域、输出数据流、输出路由、输出功能

开发期



05

关于未来的思考

一切不已浏览器为目标的框架都是耍流氓！

01 新的前端框架一览

去万物糟粕

基于 Web Components 并使用
omio 兼容老浏览器(IE8+), 很容
易实现 MVVM 架构

<https://shoelace.style/>



详情见官网

Vue使用web-
component

JSX使用web-component

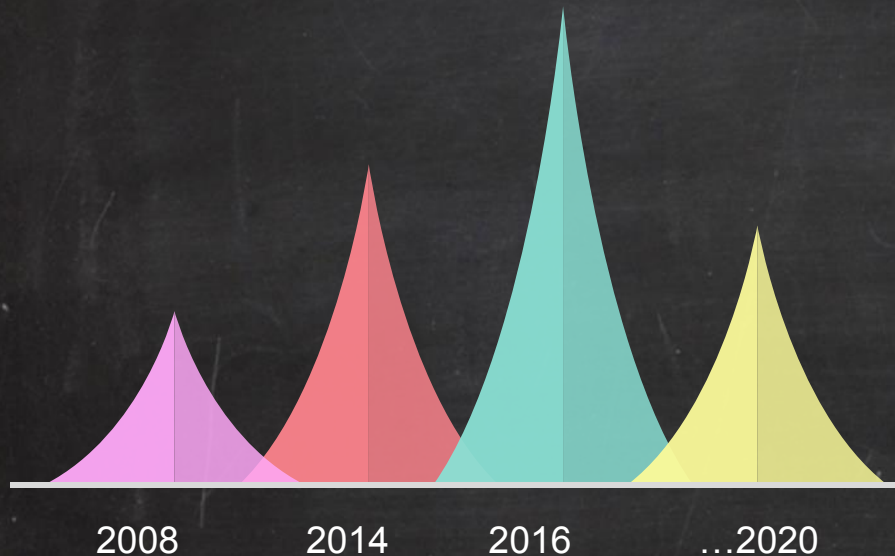
详情见官网



详情见官网

Angular使用web-
component

02 前端职业生涯



未来的产品开发主要时间是在「编排」——编排服务、编排逻辑、编排组件、编排访问策略、编排流程。到了云时代，一家企业只要招几个前端工程师就可以了，兼顾开发和运维、资产全部上云，运维任务通过控制台就能完成。开发借助 Serverless 和编排工具就能实现无服务端。在未来，无论是前端工程师还是全栈工程师，都将不复存在，应该叫端到端(F2E -> E2E)工程师了。

构建，增加微应用类型的项目构建，有动态的打包策略。传统项目管理工具通常是命令行工具，包括构建、发布、测试，会升级为项目工作台，通过 Web 界面管理项目。一个项目包括哪些微应用，版本，发布策略等在配置中心统一管理。一个大型应用被「碎片化」后，还要能做到「一目了然」。哪个模块报错，加载失败等异常发生第一时间反应在配置中心里。



3ks!

精致化的微前端开发之旅