前端脚手架相关

一、竞品分析：

	定位	功能	配置	扩展	源码工程	其他

vue-cli	vue 构建工具	• **vue create** 初始化工程
		  • 根据现有的preset
		  • 手动选择后保存为preset到.vuerc
		• **vue add** 添加一个plugin
		• **vue config** 查看全局preset配置.vuerc
		• **vue plugin** 查看当前工程安装的插件plugins
		• **vue help** 查看可用的全部命令
		• **vue-cli-service** 核心service
		  • vue-cli-service serve
		  • vue-cli-service build
		• 可构建单页或者多页工程
		• 构建库、现代模式 ES module
			• 全局.vuerc
			  • 全局的配置，如使用的npm or yarn
			  • 自己保存的新建过的工程的preset，preset只包含极少工具的一些配置，在手动create项目选择的一些选项。
			• 工程下vue.config.js
			  • 工程配置项，插件在启动的时候可以加载。
			• env环境变量配置。
			• webpack
			  • 提供了对外暴露的若干高频修改的配置项
			  • 也提供基于webpack-chain的配置。
			• 模式mode和环境变量env
				• 插件
				• vue add或者create项目的时候，插件可以修改工程文件
				• vue-cli-service插件可以添加新命令
				• 插件指定默认的模式mode
					• ES
					• @vue/cli bin: vue
					• @vue/cli-service bin: vue-cli-service
						1. 由于只是构建工具，所以只针对工程中的工具，webpack，babel，postcss等做了配置，工程结构随意。也不能指定拉取自定义好的工程模板，预设集合preset只针对工具集。
						2. 保存preset后，如果在工程里添加了新的plugin，不能自动同步到.vuerc中的preset，只能手动去同步更新。
						3. 预设的配置，包括插件plugin配置和其他配置叫做一组预设，供初始化工程使用。

飞冰	react 前端框架	• 完整的前端框架
		  • 路由配置或约定式
		  • 只支持单页应用
		• 根目录下.ice临时生成文件
		• 但工程入口文件还是src/app.tsx，不是自动生成
		• 可以添加约定形式的物料
		• icework配套的vscode可视化工具
			• build.json
			  • 工程配置
			• src/route.ts
			  • 路由配置
			• config.ts
			  • 环境配置
			  • 和环境相关的应用配置，比如接口地址前缀等
			• src/app.ts
			  • 应用功能启动项配置
				• 插件
				• index.ts 工程构建配置
				  • start/build等命令框架会加载每个插件的index.ts
				• runtime.ts 运行时配置
				  • src/app.ts中的runApp()方法会加载并执行所有插件的runtime.ts
					• TS
					  • create-ice
					  • ice.js
						• 基于插件式的前端框架，整体和umijs类似
						• 有icework vscode插件

Umi JS	react 前端框架	• 完整的前端框架
		  • 路由配置或者约定式
		  • 只支持单页应用
		• 约定式路由决定了工程目录的固定，所以说它是一个框架
		• 当然也有webpack等构建功能
		• .umi临时生成文件夹
		  • 整个工程的核心，包括工程的构建入口文件，入口文件是生成的
		• npm init @umijs/umi-app
		• 可以添加antd-pro定义的区块之类的
			• 根目录下.umirc.tc
			  • 路由（应用层面）
			  • 工程插件（工程层面）
			• 环境变量
			  • cli中设置
			  • .env中设置
			  • 只能设置已有的环境变量，不能新增，所以也有不能有多个.env环境变量的文件
			• 可以根据UMI_ENV设置不同环境变量，同时也可以根据不同环境变量有不同的.umirc.[env].tc配置文件
			• 运行时配置
				• 插件
				• 可以注册命令
				• 可以有限按照规则修改源文件
				• 可以向框架umij或者pluginsAPI上导出插件的功能api，供其他插件使用或功能调用
				• 可以注册配置项（.umrc.js中配置）
				• 可以暴露运行时配置???
				• 插件hooks回调
					• TS
					• create-umi-app bin: create-umi-app
					• umi
						• 一切都是插件，基础内置都是插件构成。
						• presets插件集，像babel
						• 也借鉴了webpack的插件hooks
						• 项目的入口文件是代码生成的，有点迷，暴露性不强
						• 配置分为工程配置，构建使用
						• 运行时配置的使用场景是什么
						• 未来规划的可视化是类似vue ui的web可视化，不是和vscode插件集成模式

megvii-fe	基于模块化和默认webpack构建的前端框架	• 初始化基于模块的单页系统工程
		  • 路由可以通过配置
		  • 路由可以兼容约定式（基于模块的目录约定）
		• 初始化基于模块和异步加载的工程
		• 根目录下提供一个打包临时构建的.megvii-cli文件夹
		• main.js保留为约定的入口
		• 提供webpack等为默认基础的构建工具
		• 命令行工具cli
		• 对外暴露供ui调用的接口
		• 提供默认模块化封装的工程结构和框架mevii-fe
			• 工程配置文件
			• env config 环境变量、和环境变量有关的应用变量
				• 插件
				• 插件集preset和插件plugin
				  • 根据支持的业务功能类型和工程种类不同默认配置不同的preset和plugin
				• service和pluginAPi参考umijs
				• 运行时配置等参考飞冰等
					• TS+JS	

二、脚手架设计

总体：

- megvii-fe CLI
  - 工程管理CLI工具
  - megvii-fe create 初始化工程

    - megvii-fe plugin 工程中插件相关（add、list等）
    - megvii-fe upgrade 升级（升级到某一版本）
- megvii-fe-service
    - 前端框架，核心依赖（vue、webpack、babel等）
    - megvii-fe-service scripts（dev、build等，可通过插件扩展）
    - 供工程应用调用的功能
        - 内置
        - 插件扩展到框架
- 对外接口
    - 供UI和vscode开发平台调用。

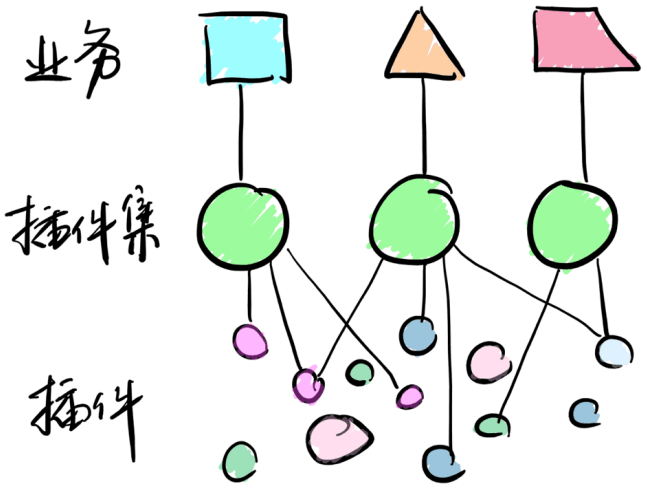
功能点：

- megvii-fe create 工程初始化：
    - 空工程
    - 简单的单页应用
    - CBG现有工程形式应用
    - 另外的cli命令，和工程管理相关，如megvii-fe add plugin、megvii-fe upgrade等

	特性	**基础依赖 built-in**	**额外安装依赖external**
空工程	• 只包含入口main.js、app.vue等工程必要部分 • 只默认安装vue、vue-loader等，不包含vue-router	vue webpack babel post-css mock lint	megvii-ui（可选） megvii-http（可选）
简单的单页应用	• 按照module封装形式初始化工程 • 默认安装vue-router • 路由配置默认集中配置，或者可支持约定式路由		vue-router megvii-ui（可选） megvii-http（可选） megvii-gis（可选）
CBG现有工程形式应用	• 当前工程能力。 • 模块单独打包导出 • auth菜单组合和路由生成 • 模块异步加载		各种能力单独封装成独立的plugin，汇总成preset-sst-fe插件集

- megvii-fe-service 框架
    - dev
    - build
    - test
    - lint
    - 应用内部可调用的service功能

- 插件
    - 业务工程、插件集合和插件
        - 空工程 preset-vue
        - 简单单页应用 preset-single-page
        - CBG应用 preset-sst-fe



- 插件对框架的扩展和使用
  - 注册新命令
  - 修改部分文件结构
  - 扩展现有的命令功能
  - 扩展框架功能，向service扩展功能
- 对外暴露的megvii-fe接口服务

工作内容：

megvii-fe-service	整个框架完整地运行周期
	
	(工程构建)
		• 内置
		  • dev、build、lint
		• 自定义
		  • test
		  • publish	

	插件扩展到serive的功能 （应用调用）	i18n、sst-fe模块异步的部分功能等	
插件	插件的初始化、集成和开发	• 内置built-in插件集和插件的梳理 • 插件的设计 PluginsAPI • 插件的开发、发布、集成和使用方式	
lint	开发规范和校验	• eslint • es、vue • prettier • stylelint • ls-lint文件目录 • 兼容现有的eslint-config-megvii • eslint和webpack的集成 • lint校验时机（lintonsave和git hooks） • 支持lint命令可扩展	
webpack	构建核心	• 最优配置 • 默认webpack 5 • webpack 4 单独plugin，兼容现有工程 • webpack配置的merge规则 • 支持webpack-chain修改webpack配置	browserslist配置
post-css	css	• 核心必要的post-css插件	

babel	js	• babel的最优配置和最佳使用方式 • babel配置的merge规则等	

config和env	配置和环境变量	• 配置方案
		  • 工程配置
		  • 应用配置
		• 环境变量方案
		• 框架和插件根据不同环境加载不同高配置	

空vue模板			mock方案 本地mockjs yapi
简单单页应用的工程模板	基于module的简单单页应用	• plugin-router • plugin-vuex • plugin-http	
CBG工程模板		• auth、startup等封装	
UI组件、业务组件、模块、业务流程组合等物料概念	各个级别物料	• 统一各个级别物料概念 • 除了基础UI组件（megvii-ui），物料仓库（公共业务组件、公共模块、高复用业务流程组合）	
其他垂直方案的整合	主题切换、国际化	• 统一的方案	

和vscode开发平台的结合	开发平台调用megvii-fe的接口，在vscode里可视化完成工程的搭建等功能	• 新建工程
		  • 和megvii-fe create一致
		  • 获取全部的模板和对应的preset
		• 新建、添加业务组件、module等	