基于微模块

构建大型web前端应用

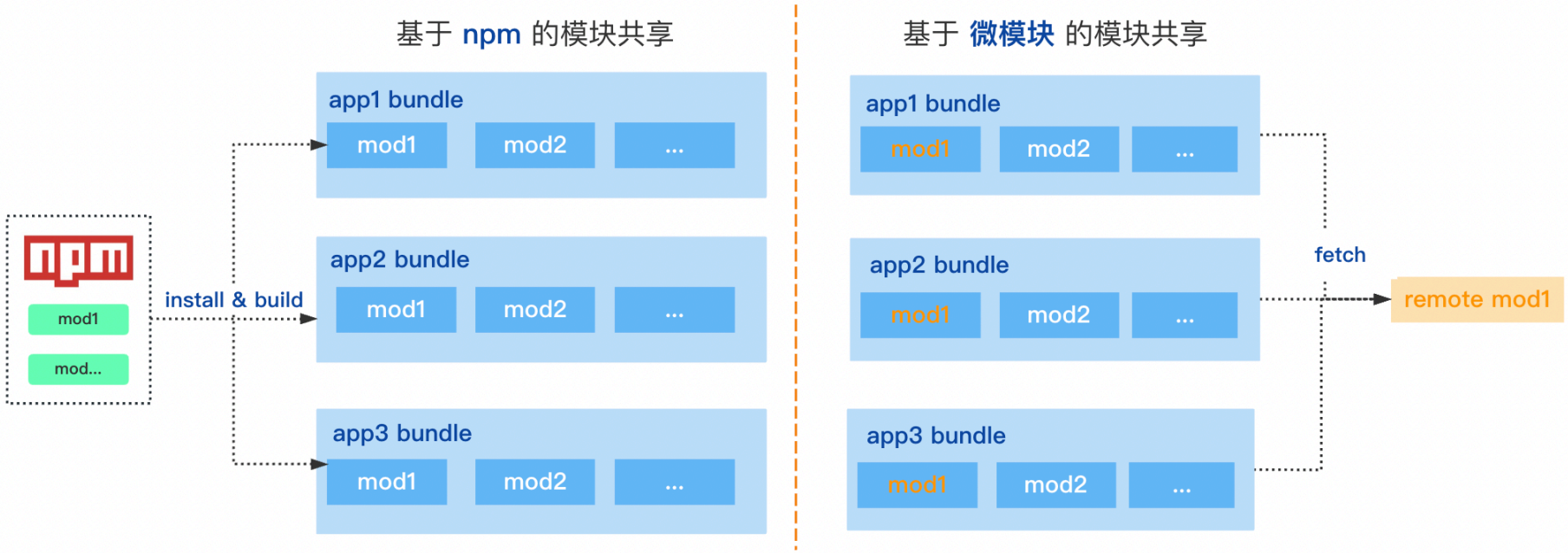
钟正楷

2023年06月19日

目录

1 微模块之进化

微模块：一种支持模块**独立开发**与**独立部署**，并在多个项目间 **运行时共享**的技术方案。

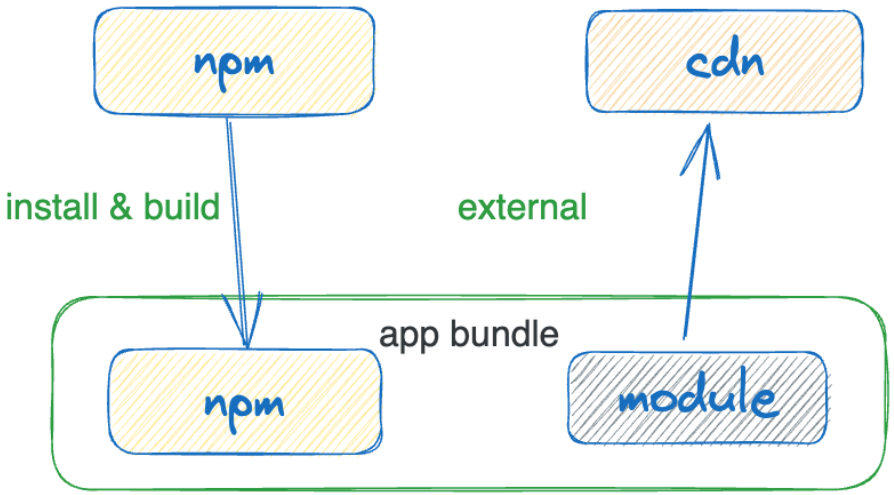


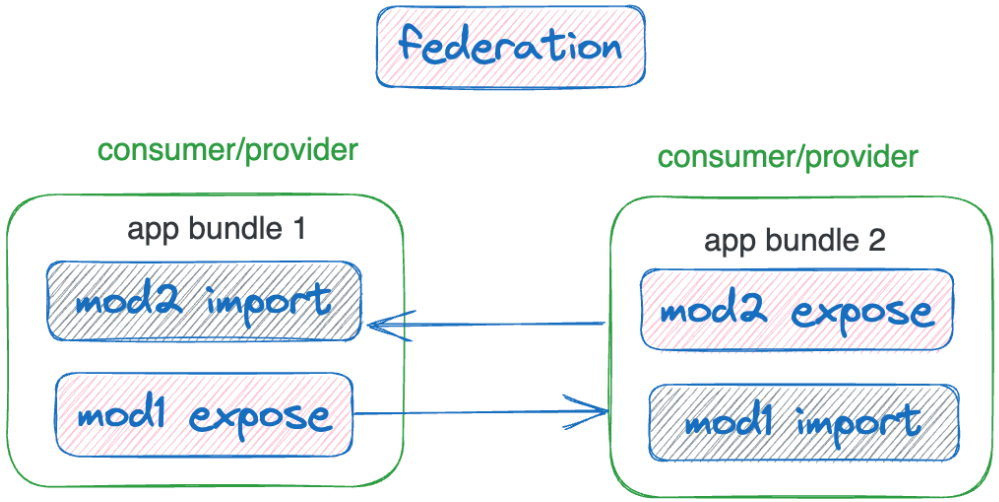
- 高效的模块分发效率
- 分离编译、构建提速
- 宿主包体积减少

cdn（umd）：

1 **模块抽象**复杂度

2 **依赖时序**问题



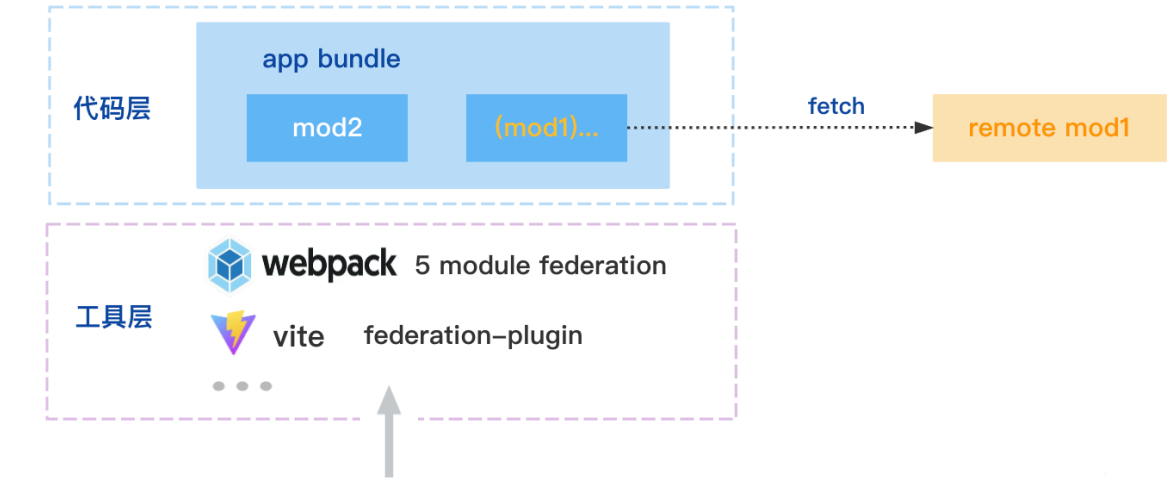


模块联邦（跨应用共享模块）：

1 便捷的**模块供需**关系

2 稳定的**模块依赖**关系

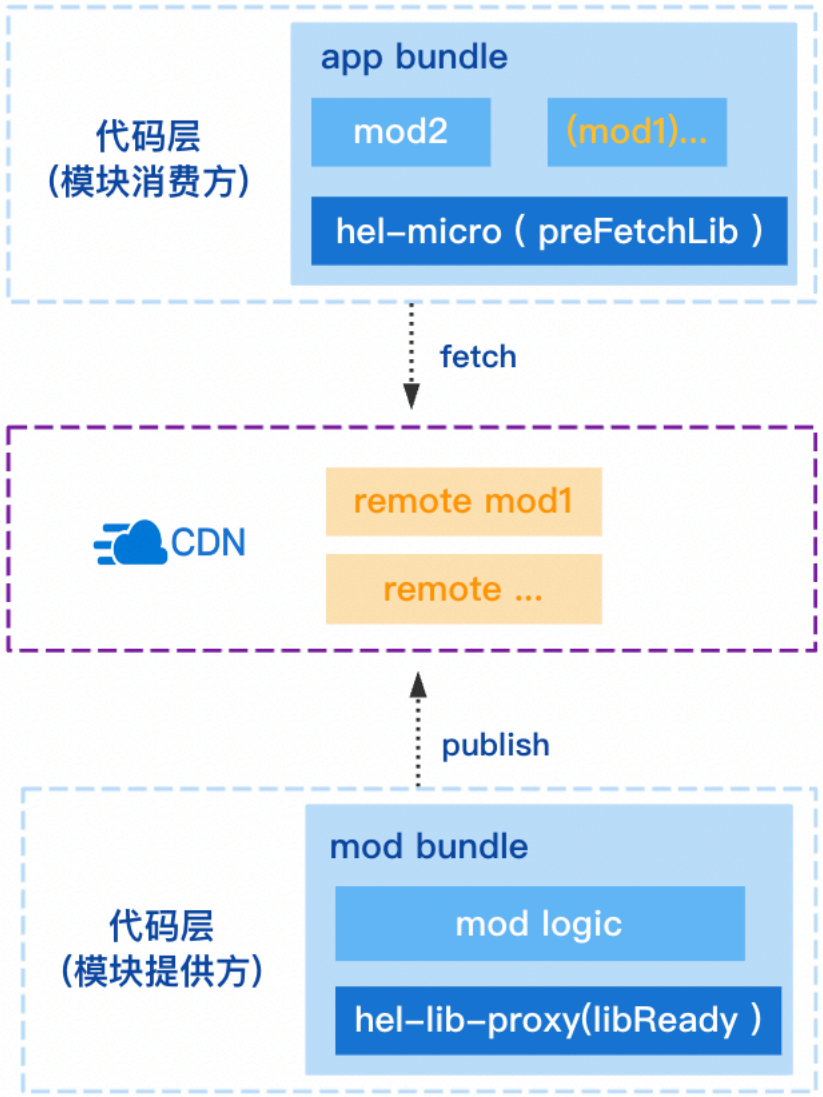
模块联邦的困境（插件）




```
new ModuleFederationPlugin({
  "name": "main_app",
  "filename": "remoteEntry.js",
  "exposes": {
    "./search": "./src/search/search.vue"
  },
  "remotes": {
    lib_remote: "lib_remote@http://xxx.yyy/remoteEntry.js",
  }
});
```

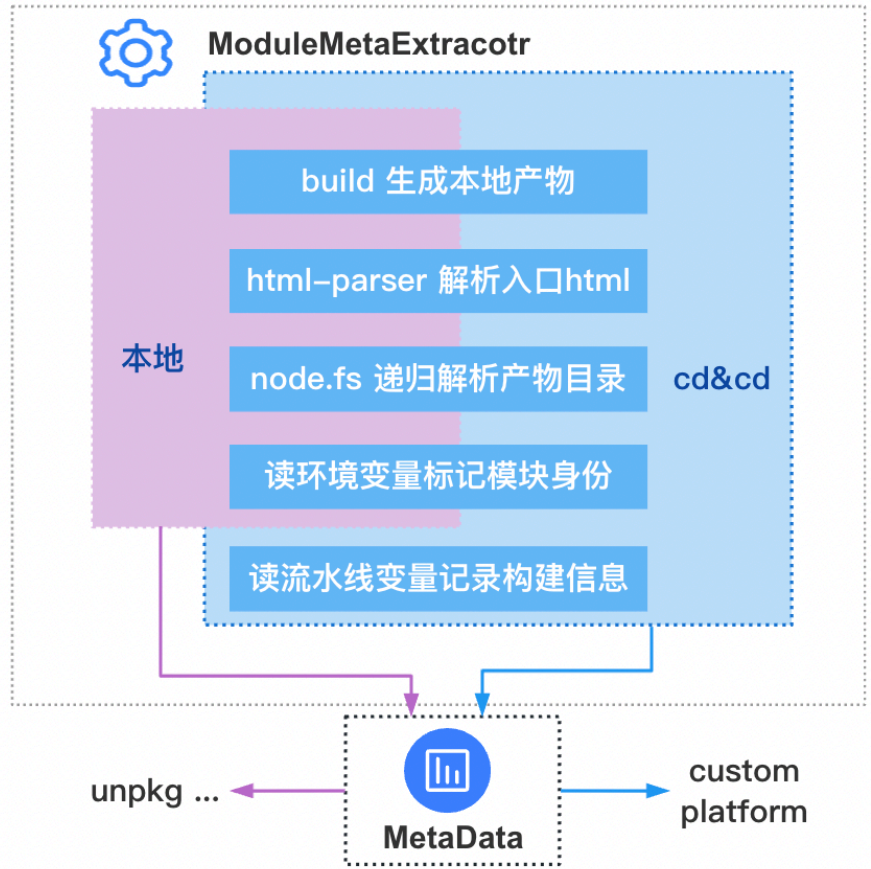
工具链绑定

构建时声明远程模块

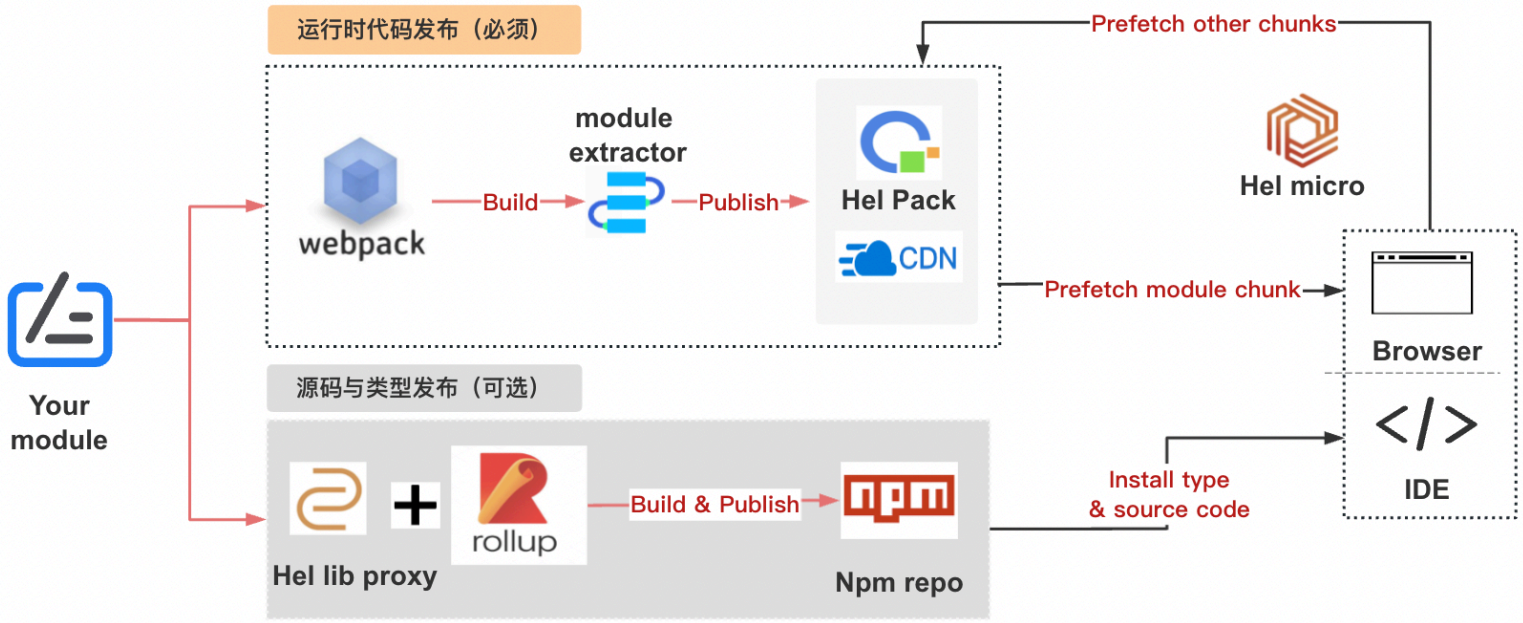


HEL MICRO

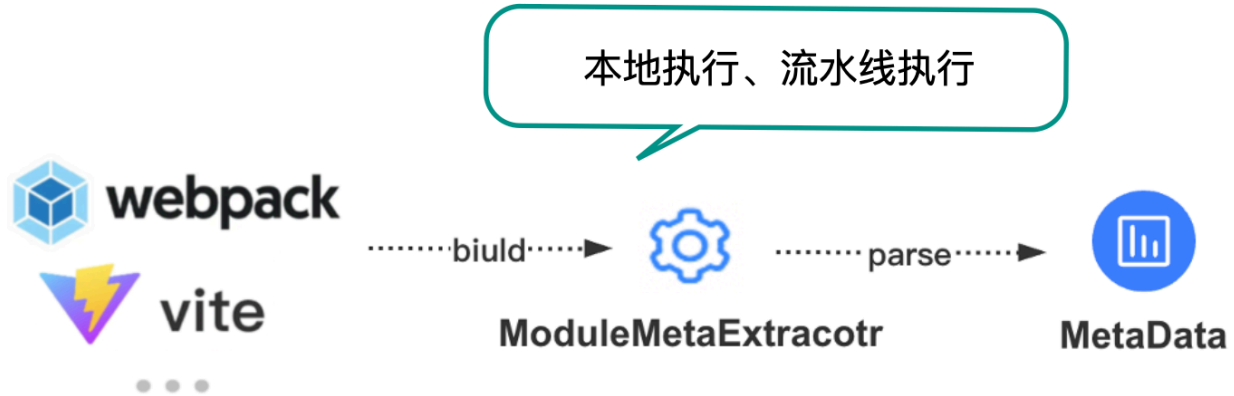
一种与**构建工具无关**，基于**构建产物元数据**提取而实现的模块联邦sdk化的技术方案。



双构建机制，无缝整合npm生态



完美解决远程模块的 类型提示问题，让用户像使用本地模块一样使用远程模块




```
- app: {
    name: "hel-lodash",
    app_group_name: "hel-lodash",
    git_repo_url: "",
    extract_mode: "build",
    online_version: "1.2.10",
    build_version: "1.2.10",
},
- version: {
    sub_app_name: "hel-lodash",
    sub_app_version: "1.2.10",
  - src_map: {
      webDirPath: "https://unpkg.com/hel
      htmlIndexSrc: "https://unpkg.com/h
      iframeSrc: "",
      chunkCssSrcList: [ ],
      privCssSrcList: [ ],
    + headAssetList: [1],
    + bodyAssetList: [2],
    },
```

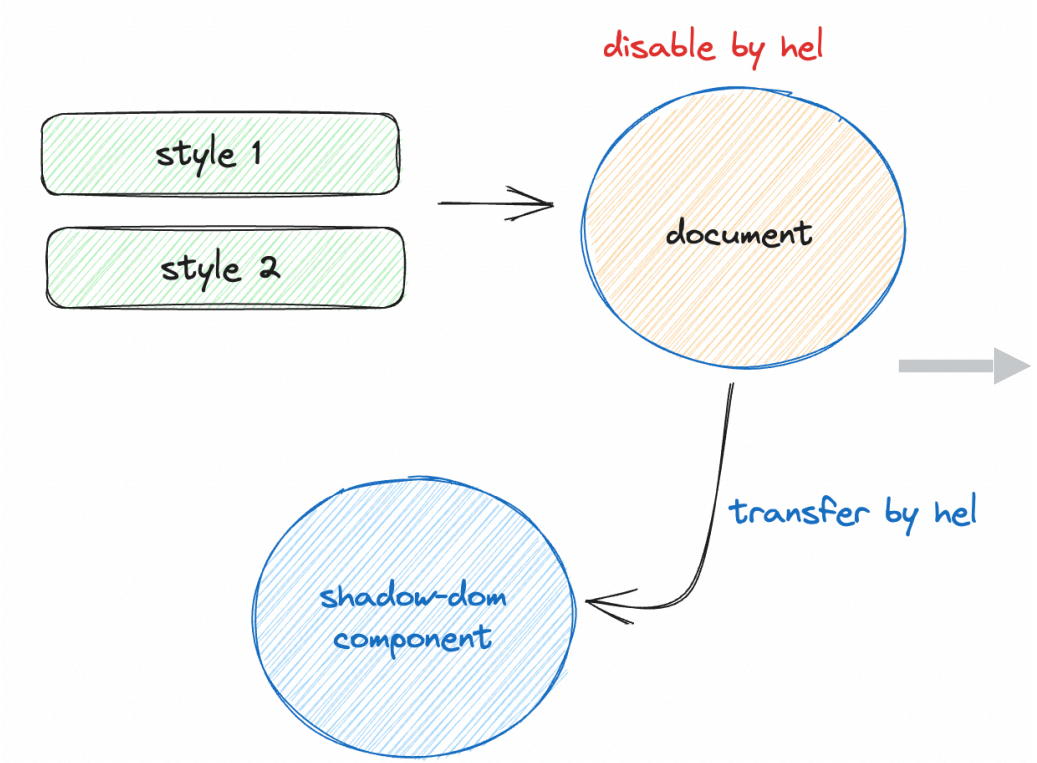
https://unpkg.com/hel-lodash/hel_dist/hel-meta.json

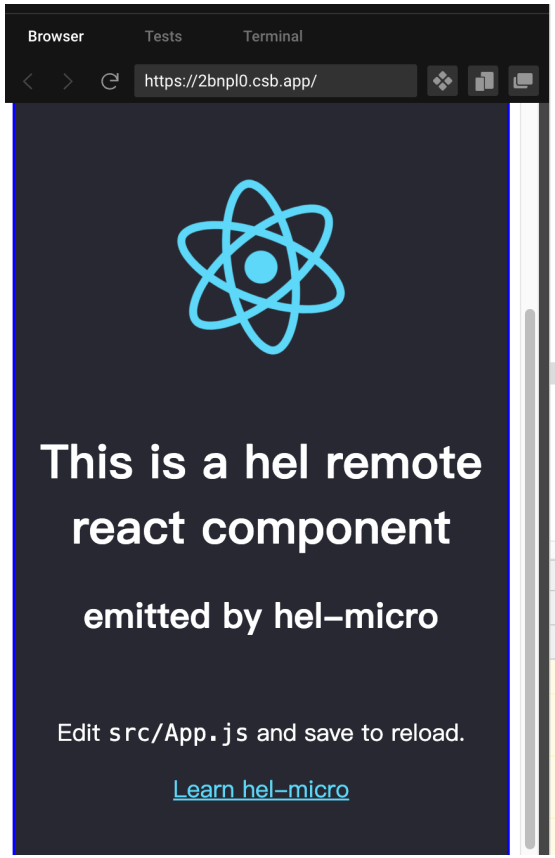
Css files

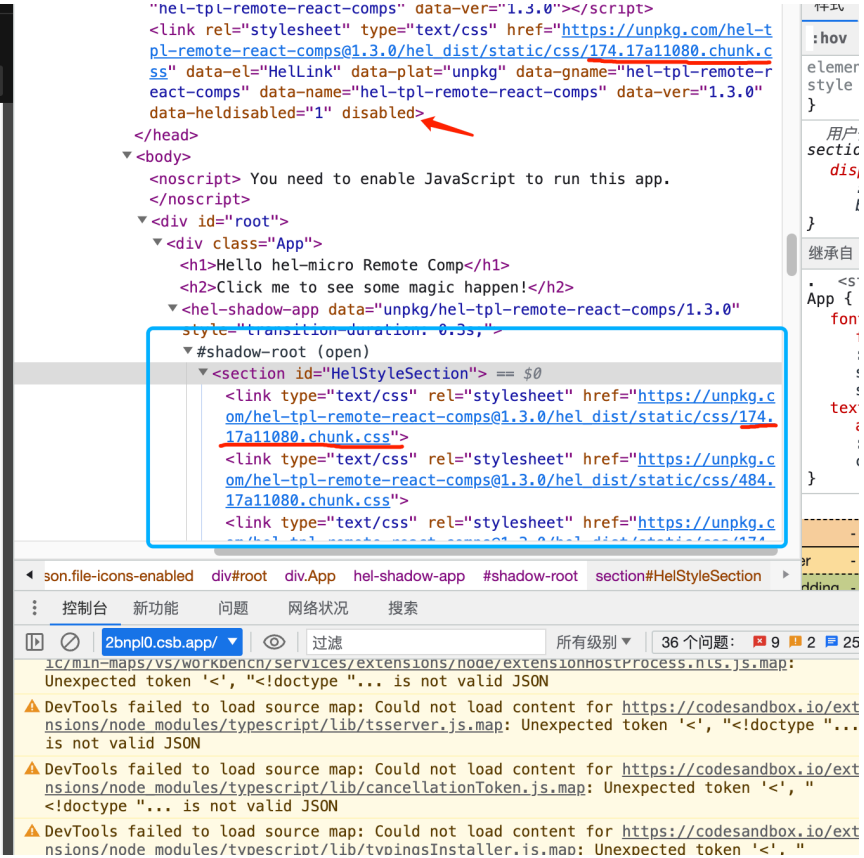
Js files

Js entry file

基于 **MutationObserver** 监听非首屏加载的css资源，为 shadowdom 组件做好更彻底的样式隔离







简单接入(consumer)

```
// await preFetchLib("hel-lodash", "2.1.0");
// await preFetchLib("hel-lodash", "2.1.1");
await preFetchLib("hel-lodash"); // latest
```

```
import { preFetchLib } from 'hel-micro';

export async function callRemoteMethod() {
  const m = await preFetchLib('hel-lodash');
  m.myUtils.myMod.sayHelloToHel();
}
```

懒加载，使用时注入

预加载，提前注入后启动应用

```
import { preFetchLib } from "hel-micro";

async function main() {
  await preFetchLib("hel-lodash"); // default latest
  await import("./loadApp");
}

main().catch(console.error);
```

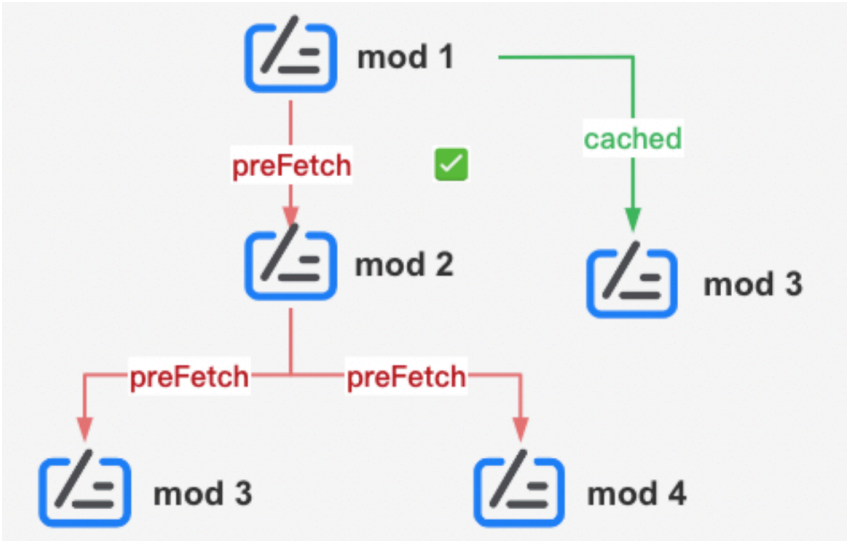


```
import m from "hel-lodash";

m.myUtils.myMod.sayHelloToHel(); // call remote safel
```

静态导入，同步调用远程方法

1 **html**和**meta**双入口，确定是宿主还是子模块

2 **模块依赖**指定与复用



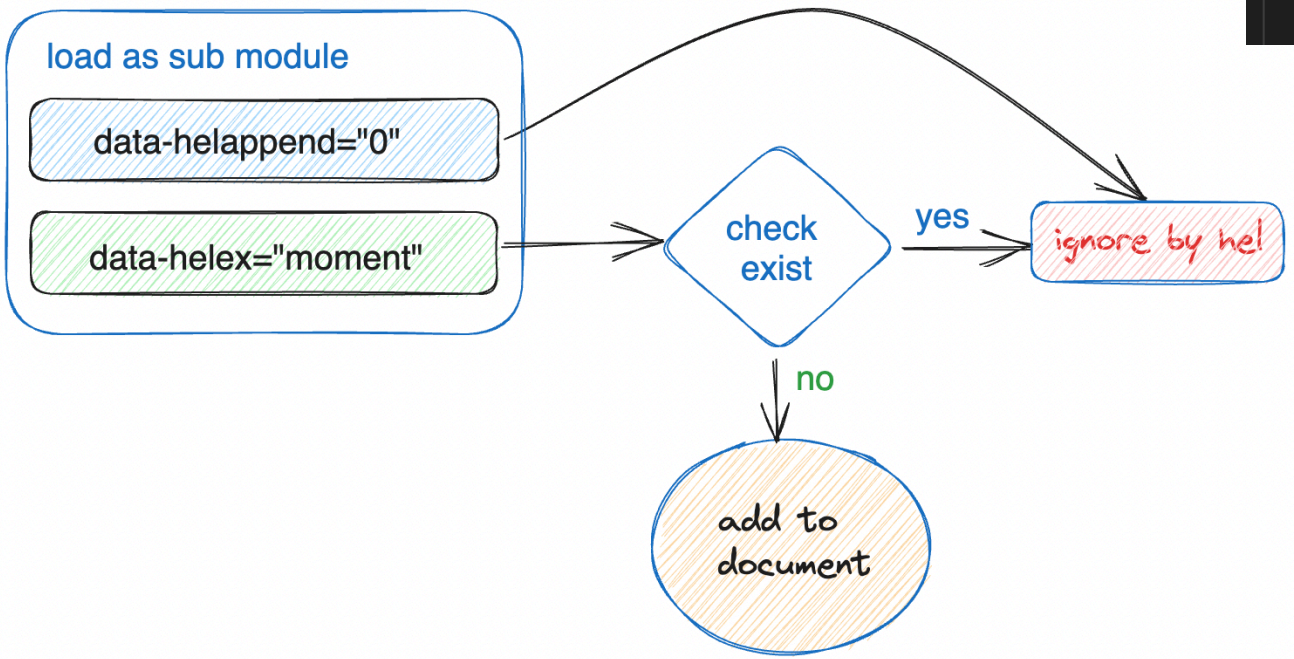

```
async function main() {
  const { isMasterApp } = await import('hel-iso');
  const { libReady } = await import('hel-lib-proxy');
  const { LIB_NAME } = await import('./configs/subApp');
  // 依赖其他模块，可使用 preFetchLib 来加载这些包体
  // const { preFetchLib } = await import('hel-micro');
  // await preFetchLib('other-lib');

  const libProperties = await import('./entrance/libProperties');
  // 发射模块
  libReady(LIB_NAME, libProperties.default);

  // 当前模块作为宿主被加载时
  if (isMasterApp()) {
    await import('./loadApp');
  }
};
```

html入口里标记构建原语，精细化控制资源加载

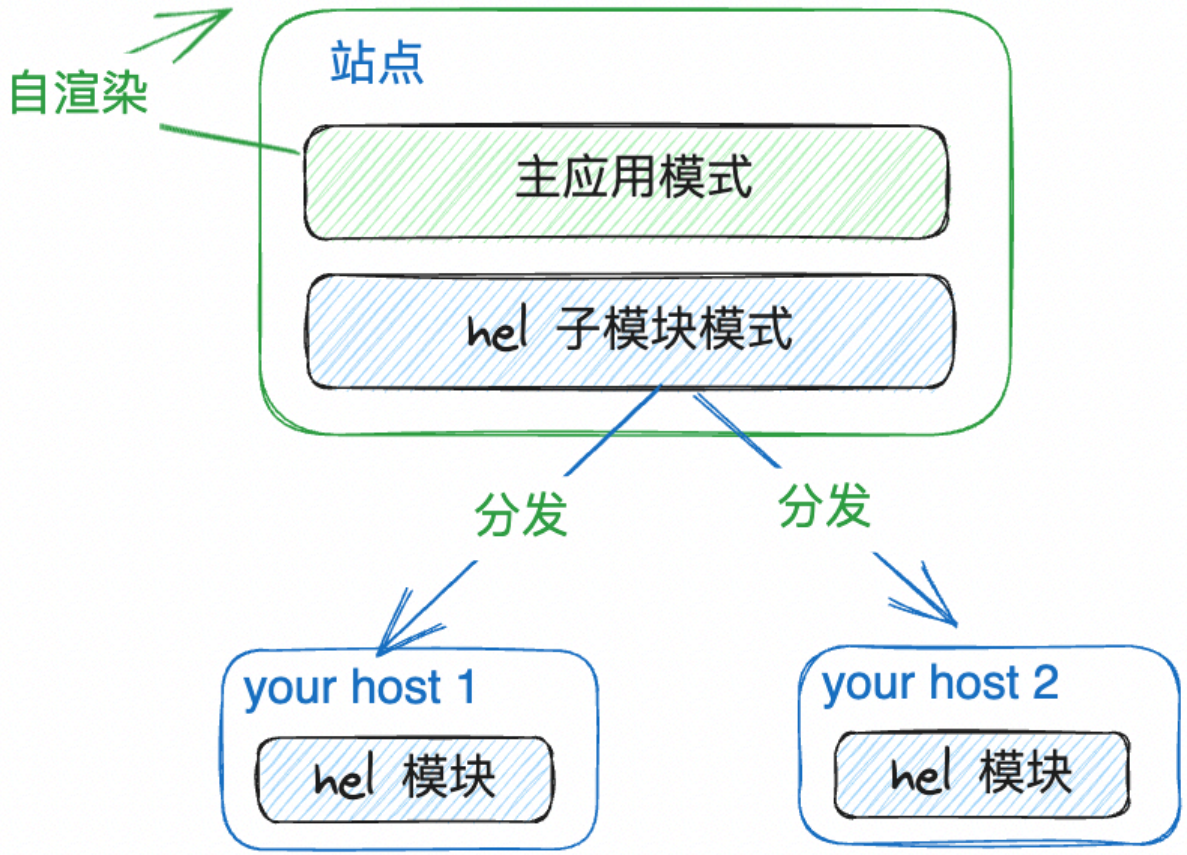
data-helappend 控制非独立加载时需禁用的资源，

data-helex 控制可延迟加载的共享模块

```
  },
  {
    "tag": "relativeScript",
    "append": true,
    "attrs": {
      "src": "/oss/watermark.js?app=osstool&show=1",
      "async": ""
    }
  },
  {
    "tag": "relativeScript",
    "append": true,
    "attrs": {
      "src": "/osscapi/rainbow/group?group=osstool&callback=__OSSTOOL
      "data-helex": "rainbow"
    }
  },
  {
```



双入口加载

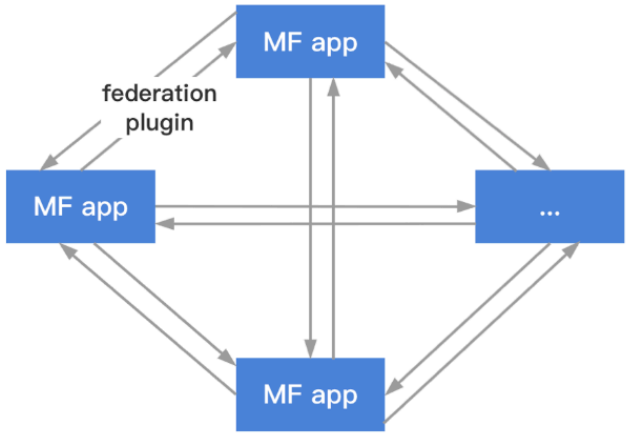



```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width
  <meta name="theme-color" content="#000000" />
  <meta name="description" content="Create Remote R
  <title>Remote React Comp</title>
  <script data-helex="moment" src="https://tnfe.gti
  <script src="https://tnfe.gtimg.com/hel-external/
</head>

<body>
  <noscript>You need to enable JavaScript to run th
  <div id="root"></div>
</body>      You, 2个月前 • feat: bump hel …

</html>
```

插件 vs sdk

插件

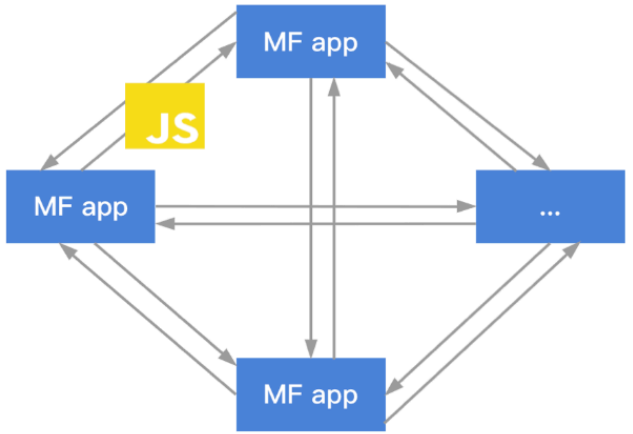
webpack vite ...



编译时

sdk

hel-micro



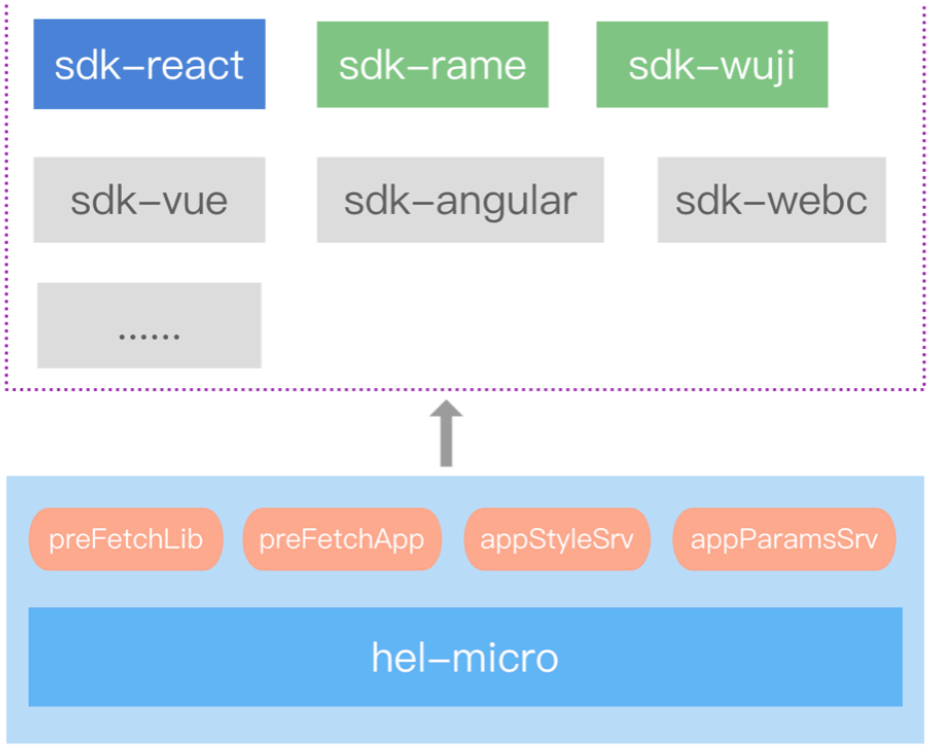
运行时

- 接入更快
- 使用更灵活
- 模块流通性更好

基于核心做更多的扩展

已实现
正在实现
等待共建

通过 hel-micro 核心接口对接具体的框架、场景、或ui适配层。




```
// fetch HelloRemoteReactComp
const compsMod = await preFetchLib("hel-tpl-remote-react-comps");
const { HelloRemoteReactComp } = compsMod;
```

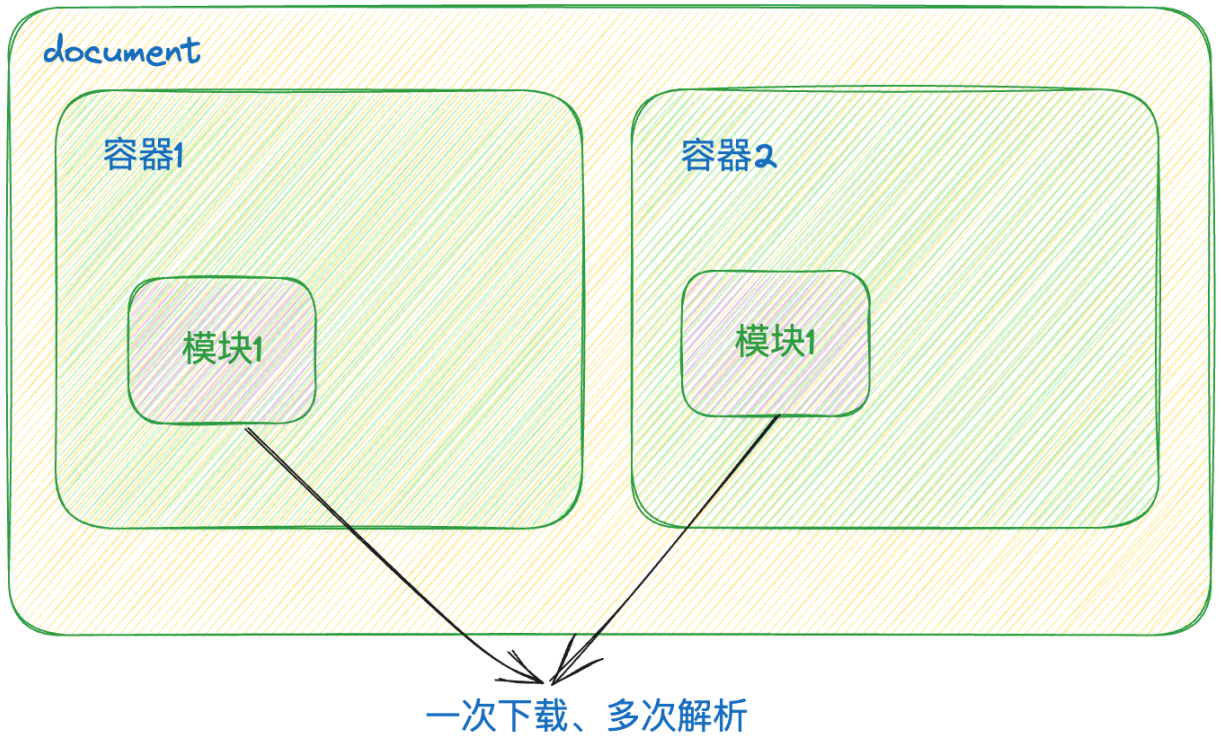
```
import "./styles.css";
import React from "react";
import { useRemoteComp } from "hel-micro-react";

export default function App() {
  const ref = React.useRef(null);
  const onClick = () => {
    ref.current?.sayHello();
  };

  const Comp = useRemoteComp(
    "hel-tpl-remote-react-comps",
    "HelloRemoteReactComp"
  );

  return (
    <div className="App">
      <h1 onClick={onClick}>Hello hel-micro Remote Comp</h1>
      <Comp ref={ref} />
    </div>
  );
}
```

2

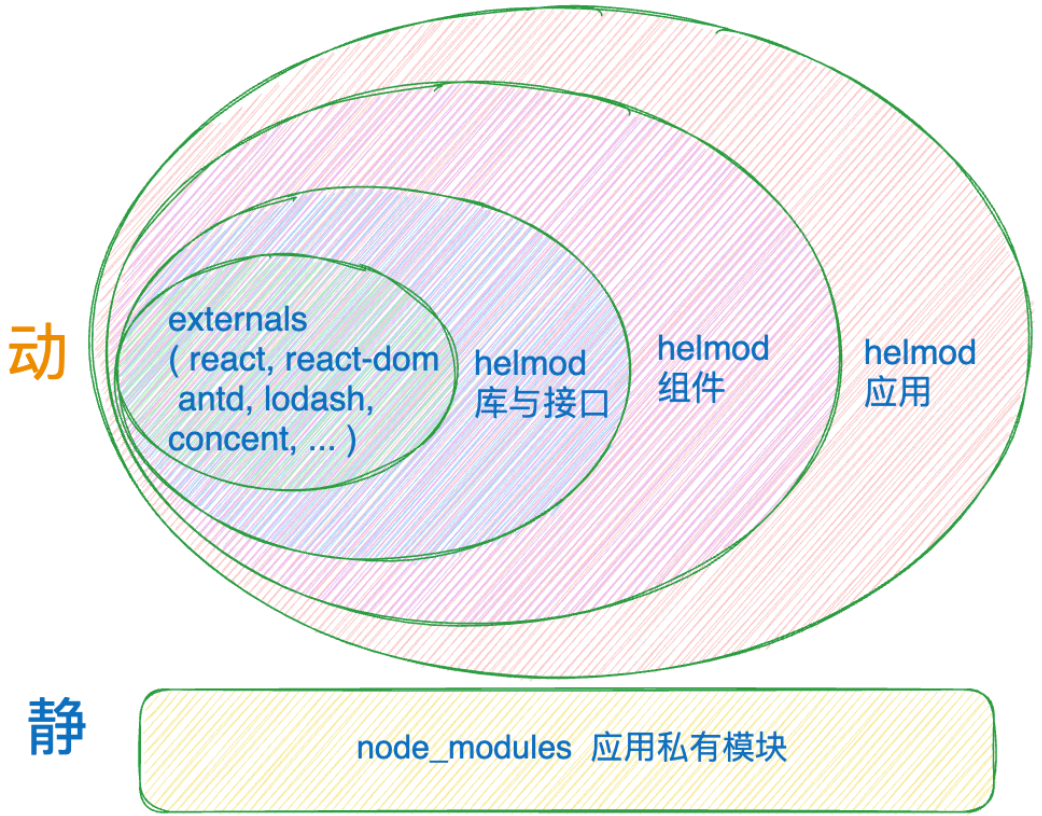
基于hel微模块的

大型web前端项目设计与架构

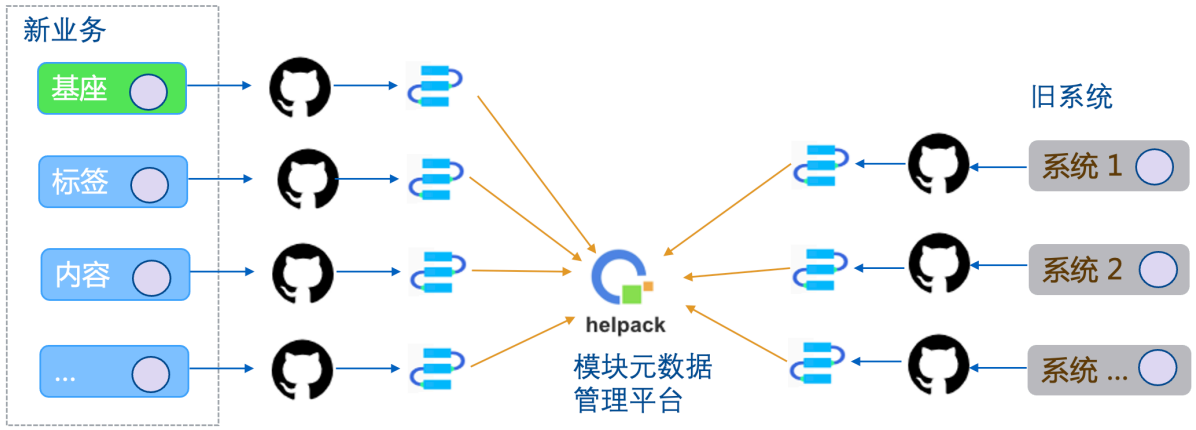
基于 **先微模 再容器** 的原则



externals 管理底层公共模块
hel-micro 管理业务公共模块
node-modules 管理私有模块

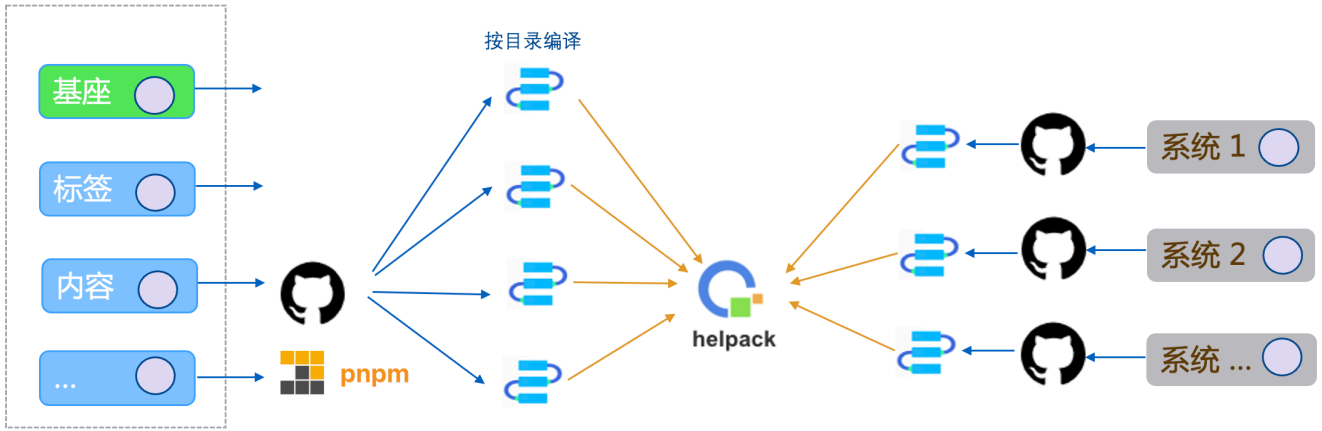


大仓化（pnpm + hel-micro）

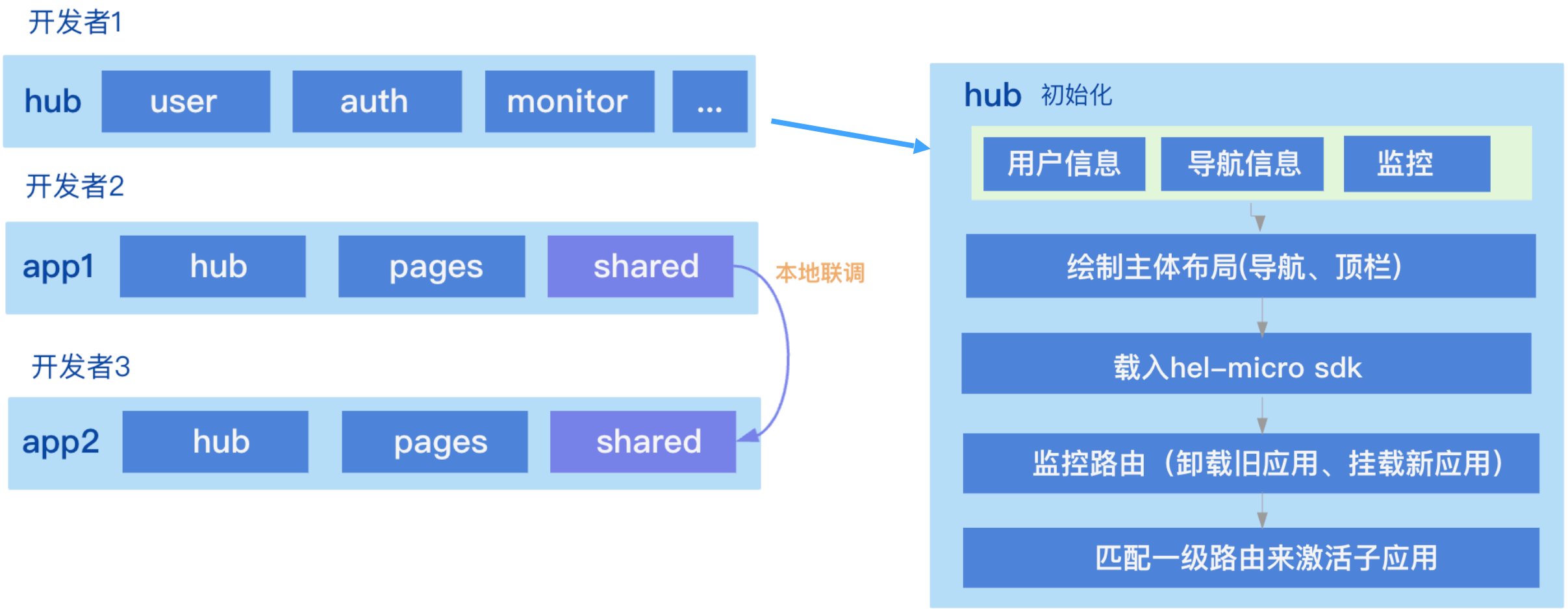


阶段1：
接入hel-micro sdk，按功能切分模块，采用**一主多子**的运行模式搭建起**基于微模块**的微前端架构。

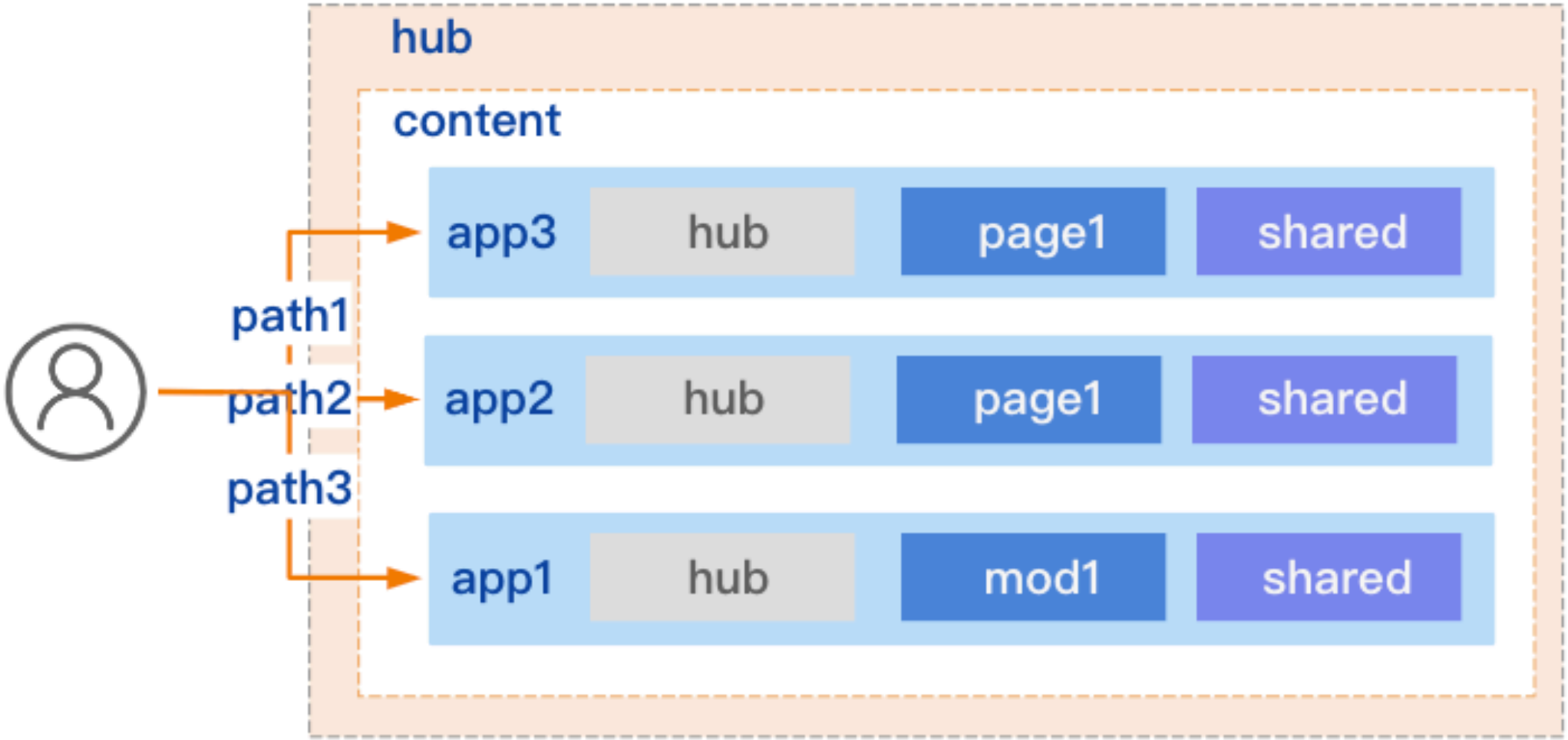
阶段2：
引入pnpm workspace，将项目**大仓化**，方便快速开发多个应用，统一管理手脚架基础配置，提速本地项目的初始化速度。



开发时每个应用都**自带基座**，上线后子应用的**基座不加载**

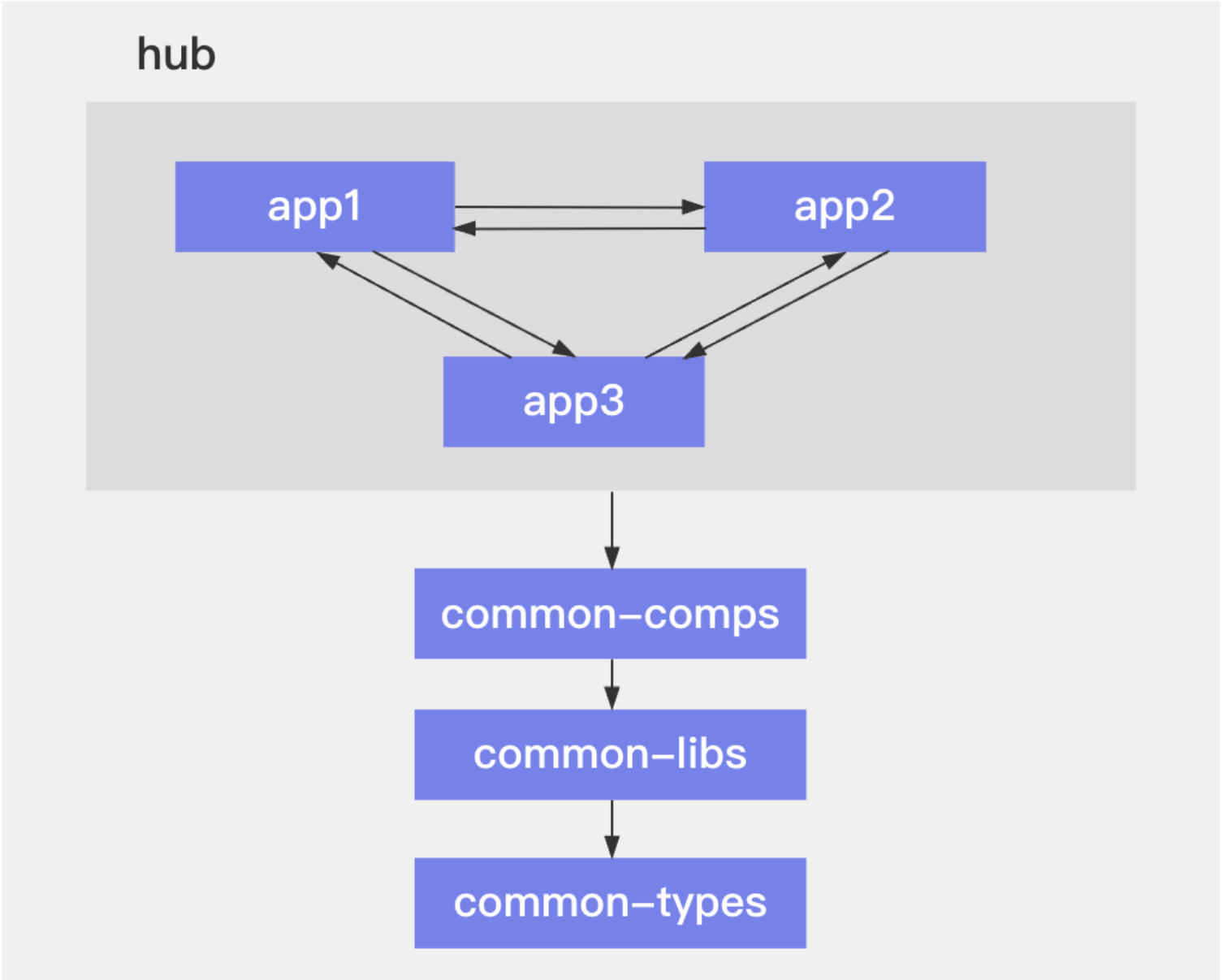


线上运行时，先载入基座，通过**一级路由激活匹配**的子应用，子应用在自己固定的一级路由下注册其他**二级路由相关页面**

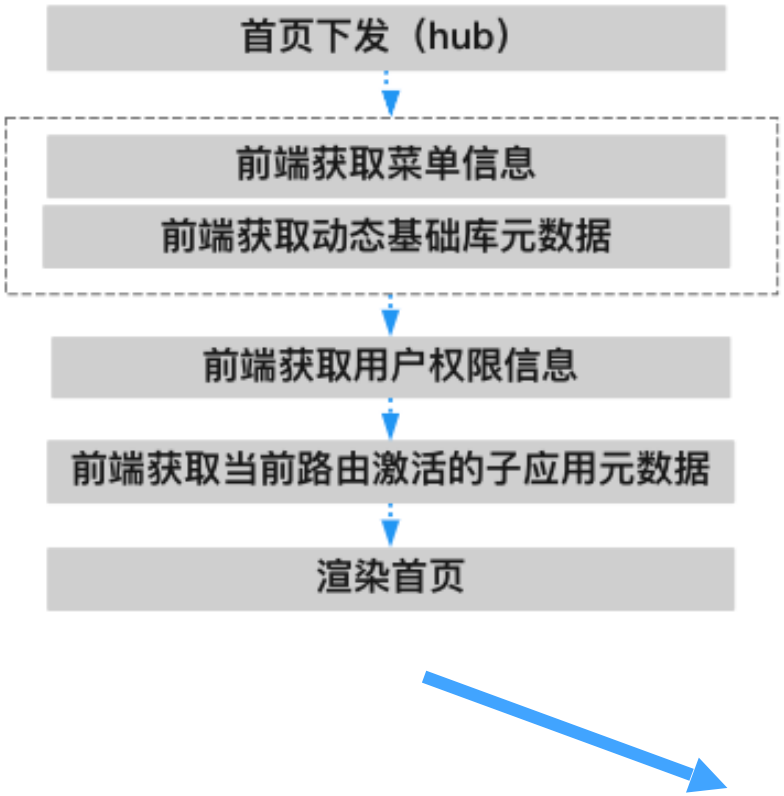


1. 需在各个子应用间复用的小组件，直接**利用sdk共享出去**，方便随取随用

2. 可以剥离出子应用上下文的组件、函数等，再进一步抽到**公共模块层**，（common-comps 或 common-libs）



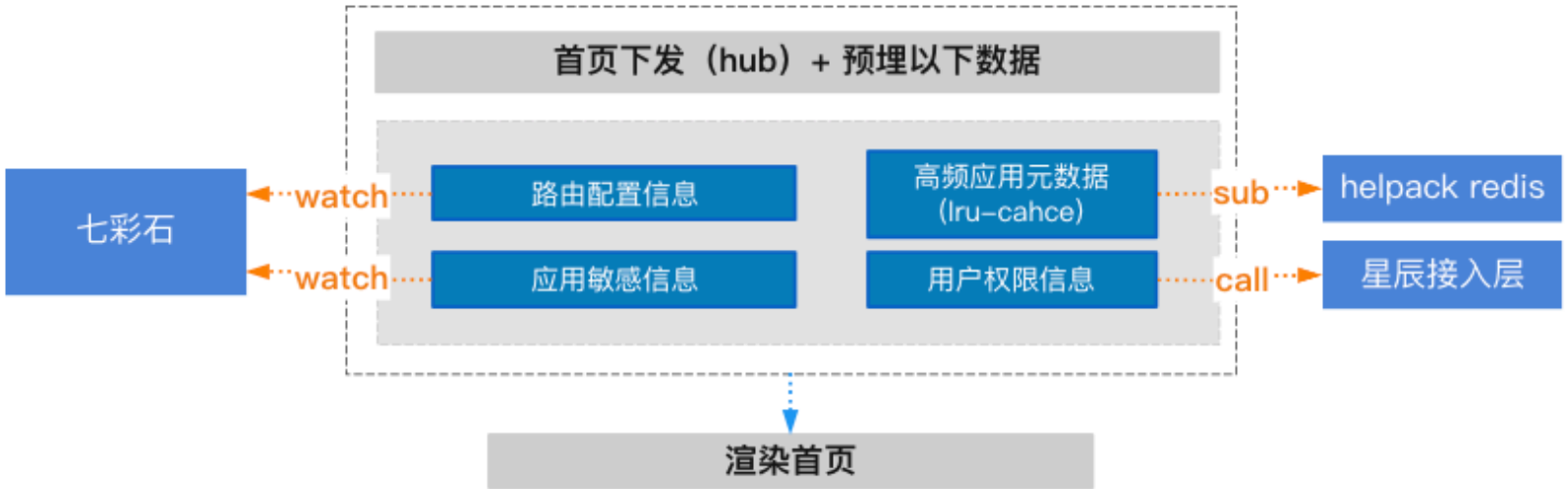
首页提速（元数据预埋）



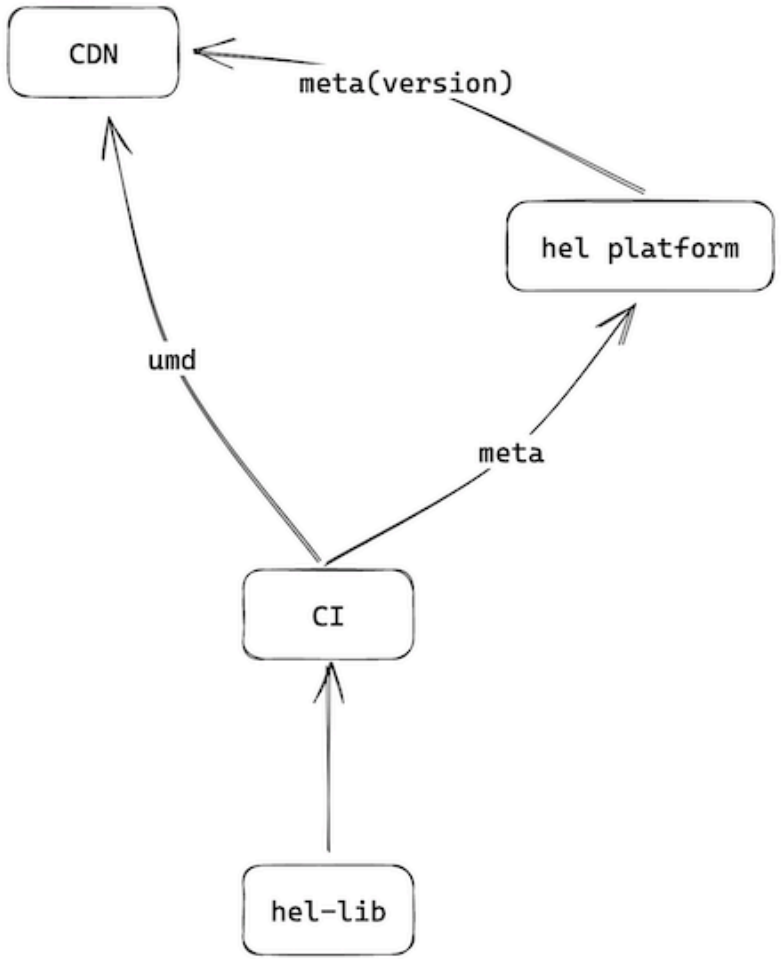
1 首页**预埋元数据**，提速首屏模块加载速度

2 首页**预埋元数据版本**，前端缓存元数据信息做diff，减少首屏html大小

3 requestIdleCallback **预热高频应用**，应用切换秒开



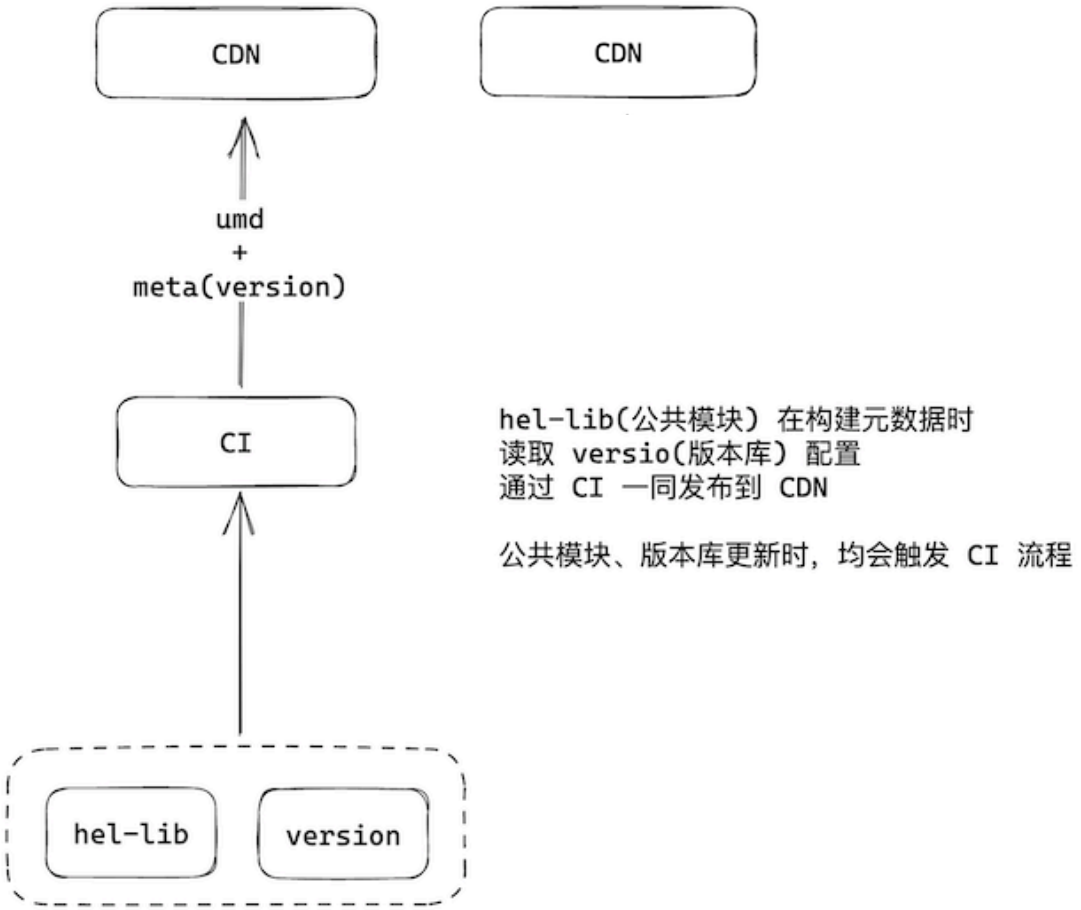
平台化与纯cdn架构


CDN
meta(version)
hel platform
umd
meta
CI
hel-lib


应用 remote-react-comps-tpl
编辑应用
版本列表
项目与版本 new
访问应用
灰度名单:
全部
已标记
灰度通过
切换灰度版本
切换在线版本
刷 新
线上
版本号: remote-react-comps-tpl_20230520181828
触发者: fancyzhong
git 提交 hash: fbdd13df00d7e179d91275850f290dee75d0fb60
git提交描述: feat: bump hel
构建时间: 2023-5-20 18:19:45
构建分支: master
蓝盾项目空间: hel-app-store
访问应用
查看流水线
查看git
切为线上
切为灰度
更多
灰度
版本号: remote-react-comps-tpl_20230503175534
触发者: fancyzhong
git 提交 hash: 695e3b8a22b9682114bf980e379f30d67bcc3baf
git提交描述: feat: bump hel
构建时间: 2023-5-3 17:56:52
构建分支: master
蓝盾项目空间: hel-app-store
个人
版本号: remote-react-comps-tpl_20230430121702
触发者: fancyzhong
git 提交 hash: 695e3b8a22b9682114bf980e379f30d67bcc3baf
git提交描述: feat: bump hel
个人标记描述: 某一个特殊版本 (2023-06-19 10:46:48)
构建时间: 2023-4-30 12:18:19
构建分支: master
蓝盾项目空间: hel-app-store

1 **纯cdn架构**，去平台管理

2 **多cdn容灾 + asset-retry**，确保模块成功加载



```
npm install hel-micro
```

start your micro module journey

谢谢聆听，欢迎了解hel-micro微模块

项目git：https://github.com/tnfe/hel （by fantasticsoul）

HEL MICRO

幻魂

fantasticsoul