QCon+ 案例研习社

淘系中后台的前端架构演进之路

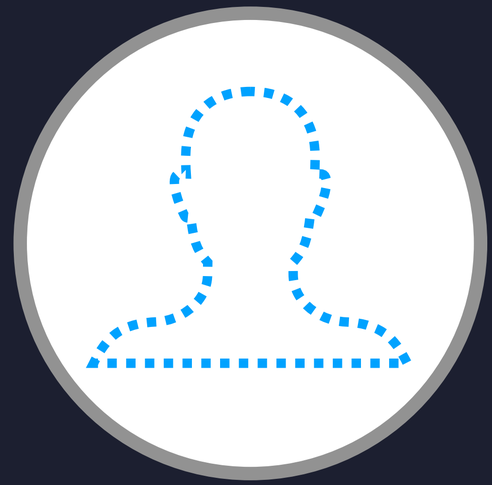
大果

关于我

大果 http://github.com/imsobear

淘系技术部 | 前端架构 | 飞冰（ICE）

资深答疑专家 | 跨端解决方案 Rax



1 飞冰（ICE）

2 Rax

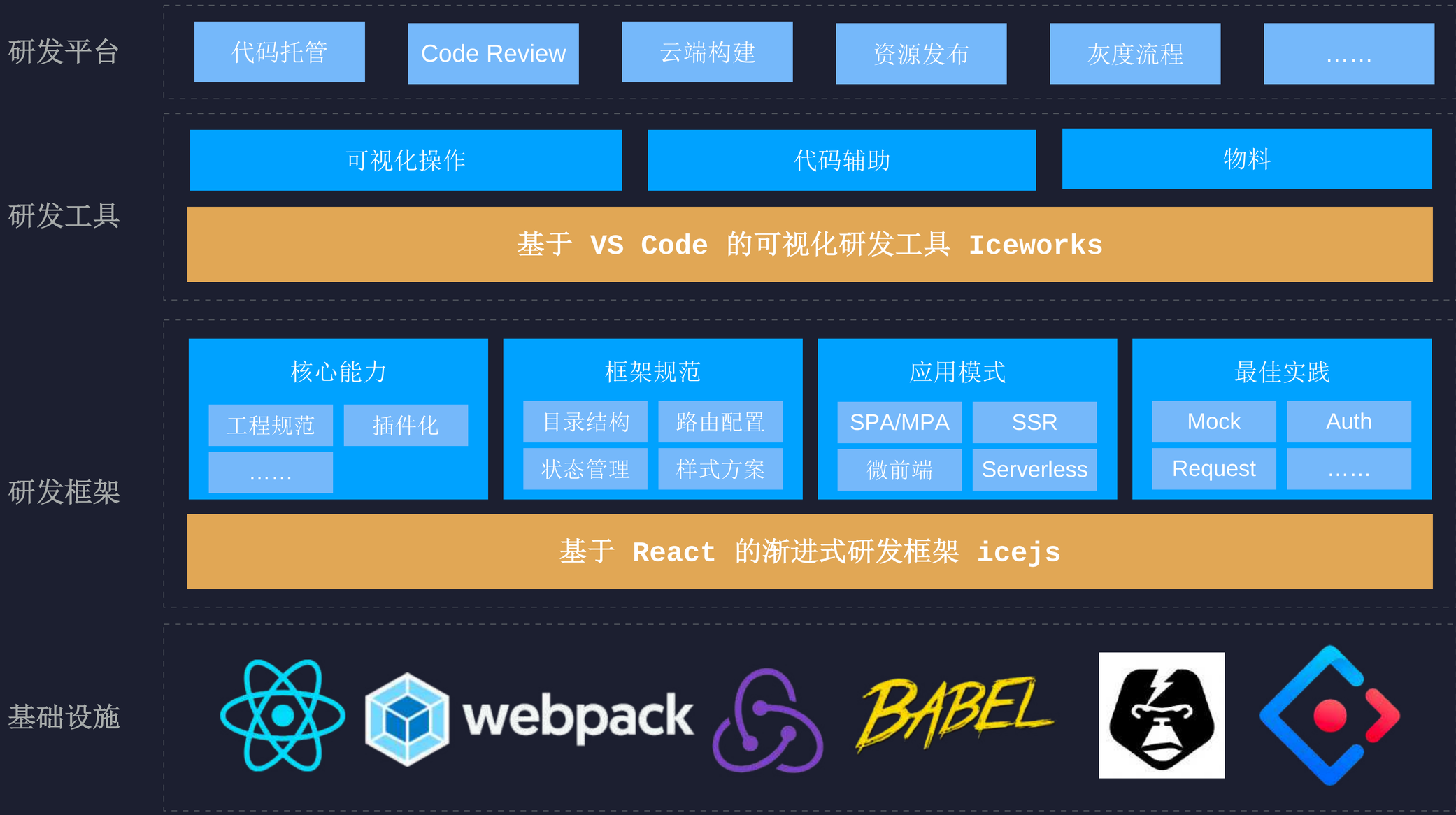
目录

体系介绍：中后台与消费端

	中后台	消费端
用户群体	小二/商家/...	消费者
技术诉求	研发效率	用户体验
技术难点	研发能力偏低	渲染环境复杂

体系介绍：中后台架构分层

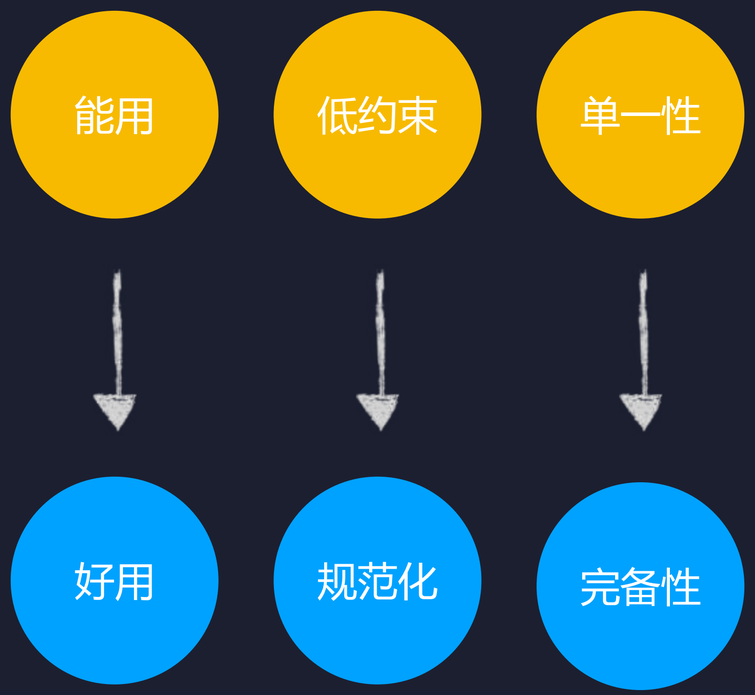
体系介绍：淘系中后台前端架构大图



研发框架

为什么需要研发框架

研发框架的发展之路



青铜段位

裸写工程配置

钻石段位

工程能力封装，插件化

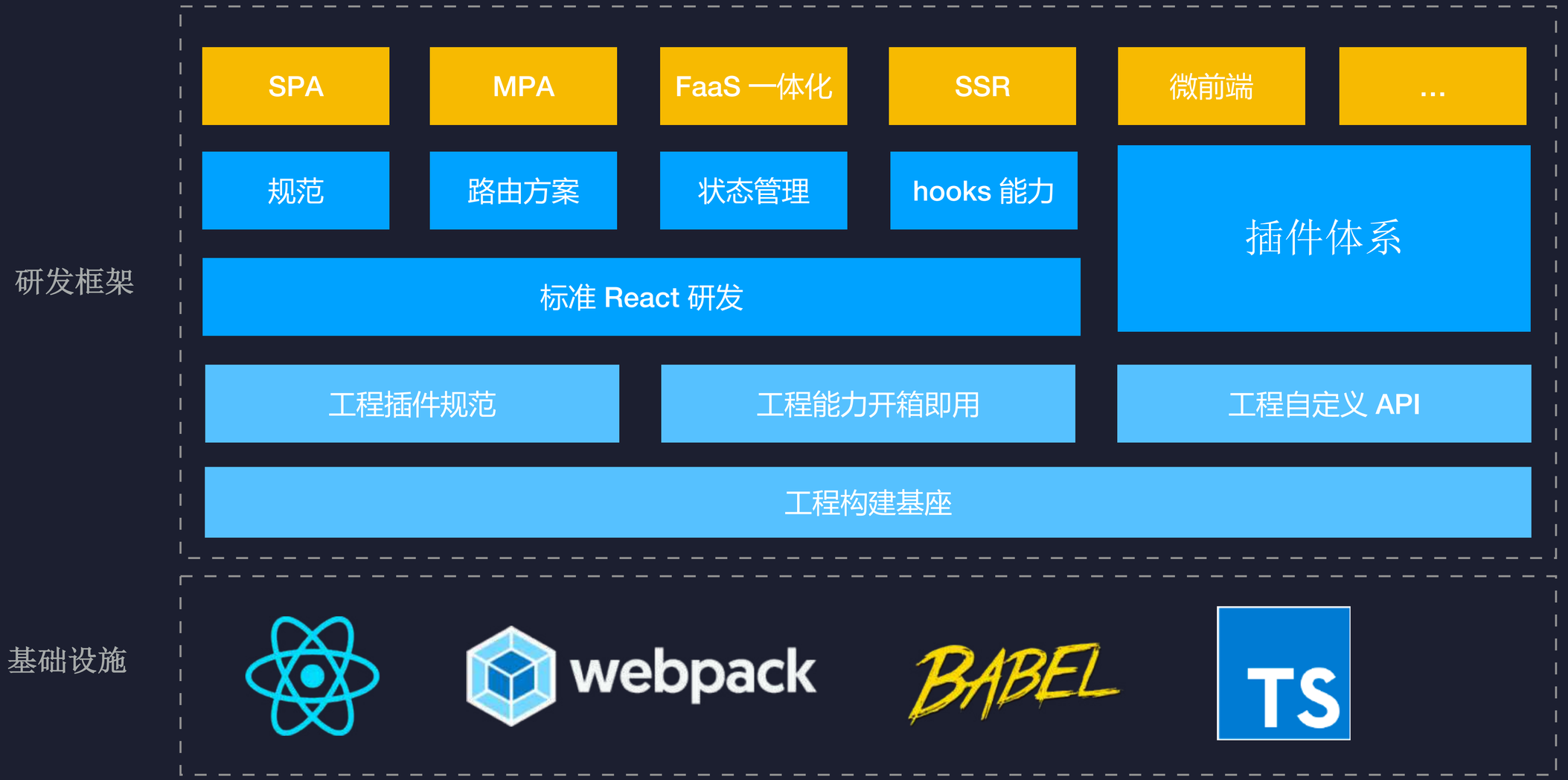
研发框架

开箱即用+最佳实践

为什么不是……？

- 飞冰（ICE）从 2016 年开始发展，早于部分框架
- 结合业务场景提供深度定制的解决方案

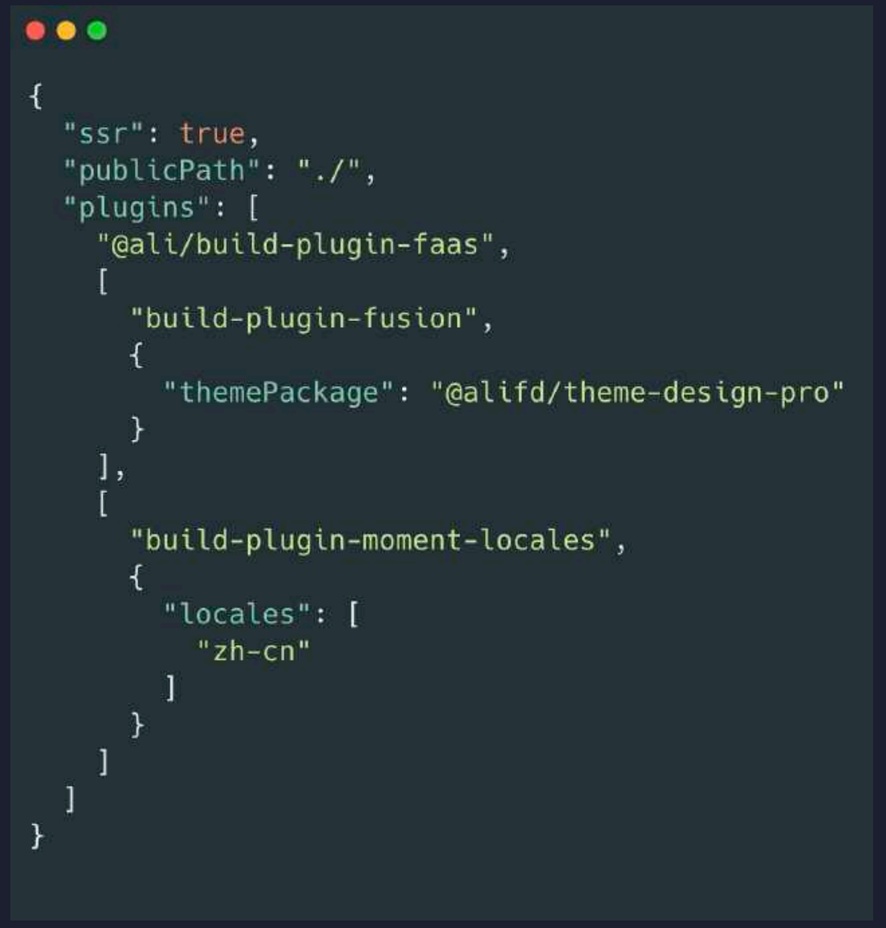
icejs: 基于 React 的渐进式研发框架



快速认识 icejs

```
├── .ice/
├── public/
├── src/
│   ├── components/
│   ├── layouts/
│   ├── pages/
│   │   └── Home/
│   │       └── index.tsx
│   ├── models/
│   ├── global.scss
│   ├── routes.ts
│   └── app.ts
├── build.json
├── package.json
└── tsconfig.json
```

目录结构



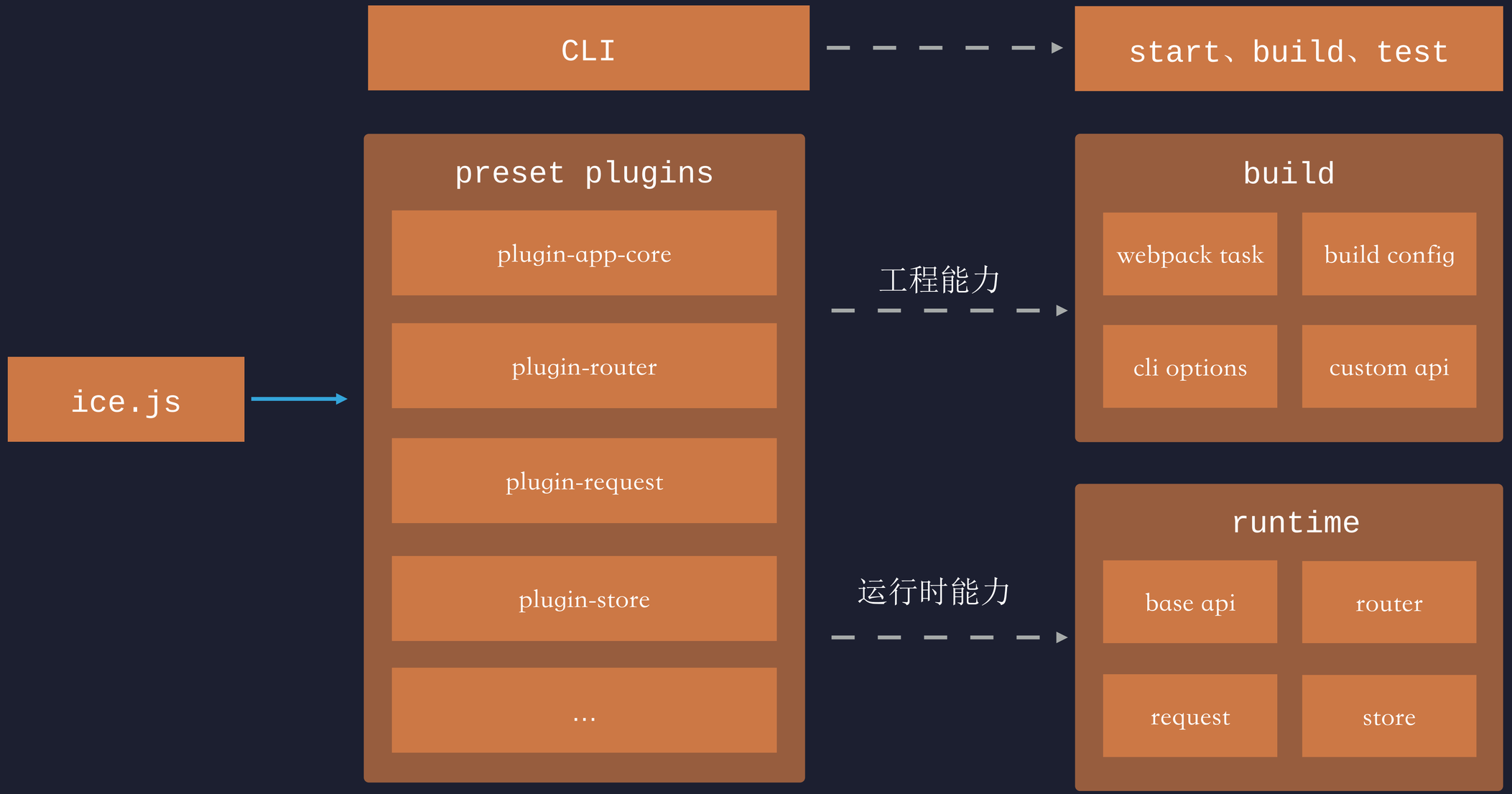
```
{
  "ssr": true,
  "publicPath": "./",
  "plugins": [
    "@ali/build-plugin-faas",
    [
      "build-plugin-fusion",
      {
        "themePackage": "@alifd/theme-design-pro"
      }
    ],
    [
      "build-plugin-moment-locales",
      {
        "locales": [
          "zh-cn"
        ]
      }
    ]
  ]
}
```

工程配置 build.json

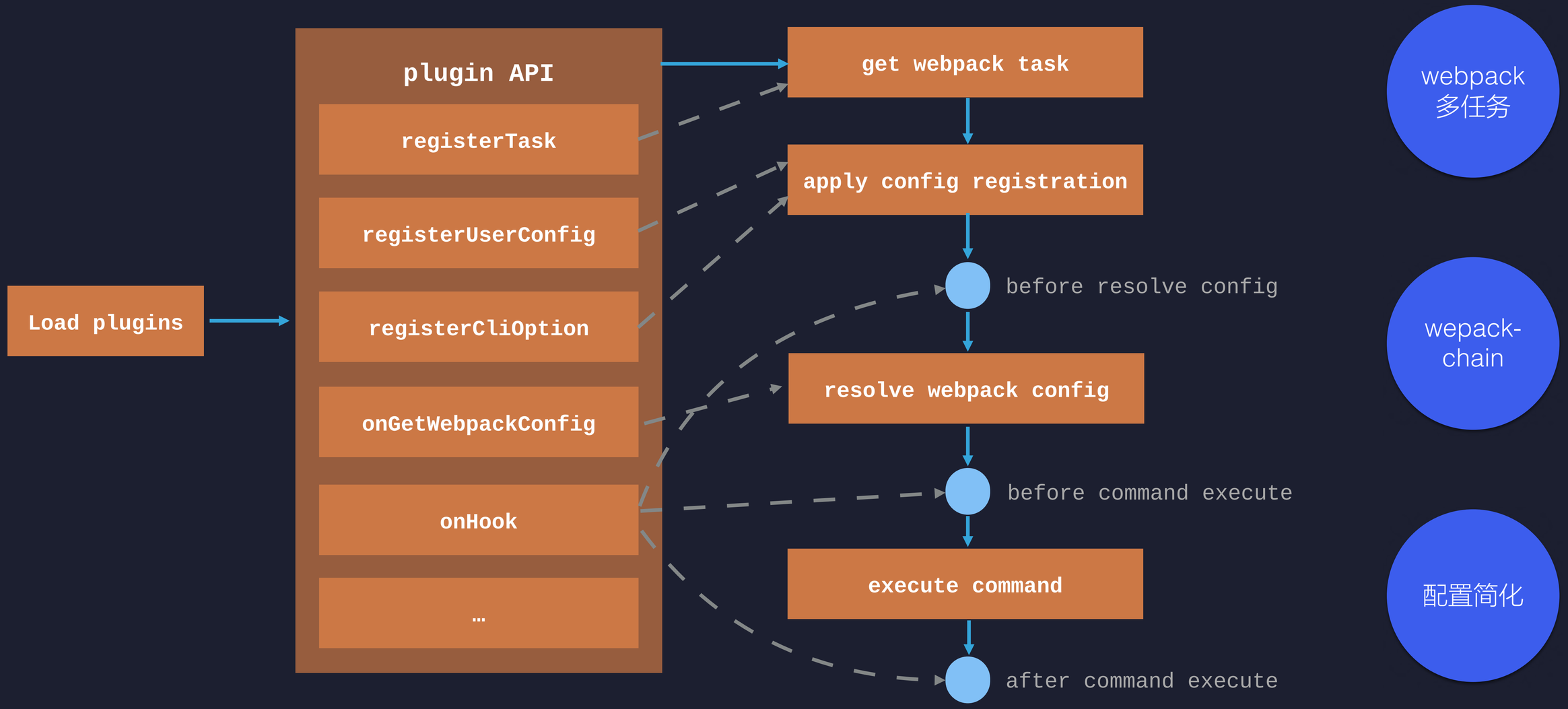
```
import {
  createApp,
  IAppConfig
} from 'ice';

const appConfig: IAppConfig = {
  app: {
    rootId: 'ice-container',
  },
  router: {
    type: 'browser'
  },
  request: {
    baseURL: '/api/'
  }
};

createApp(appConfig);
```

应用入口 src/app.ts

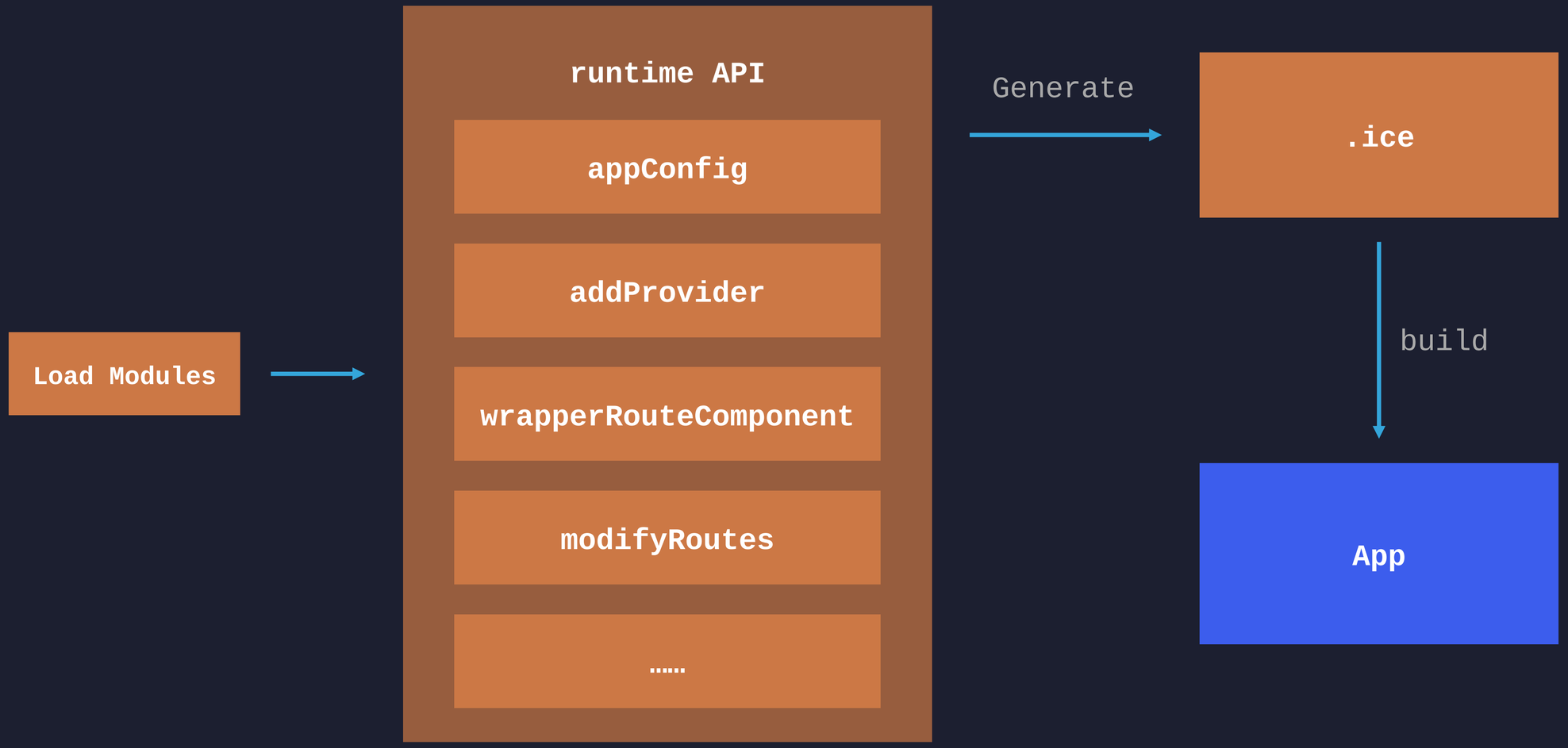
icejs：架构设计


CLI
start、build、test
preset plugins
plugin-app-core
plugin-router
plugin-request
plugin-store
...
ice.js
工程能力
build
webpack task
build config
cli options
custom api
运行时能力
runtime
base api
router
request
store

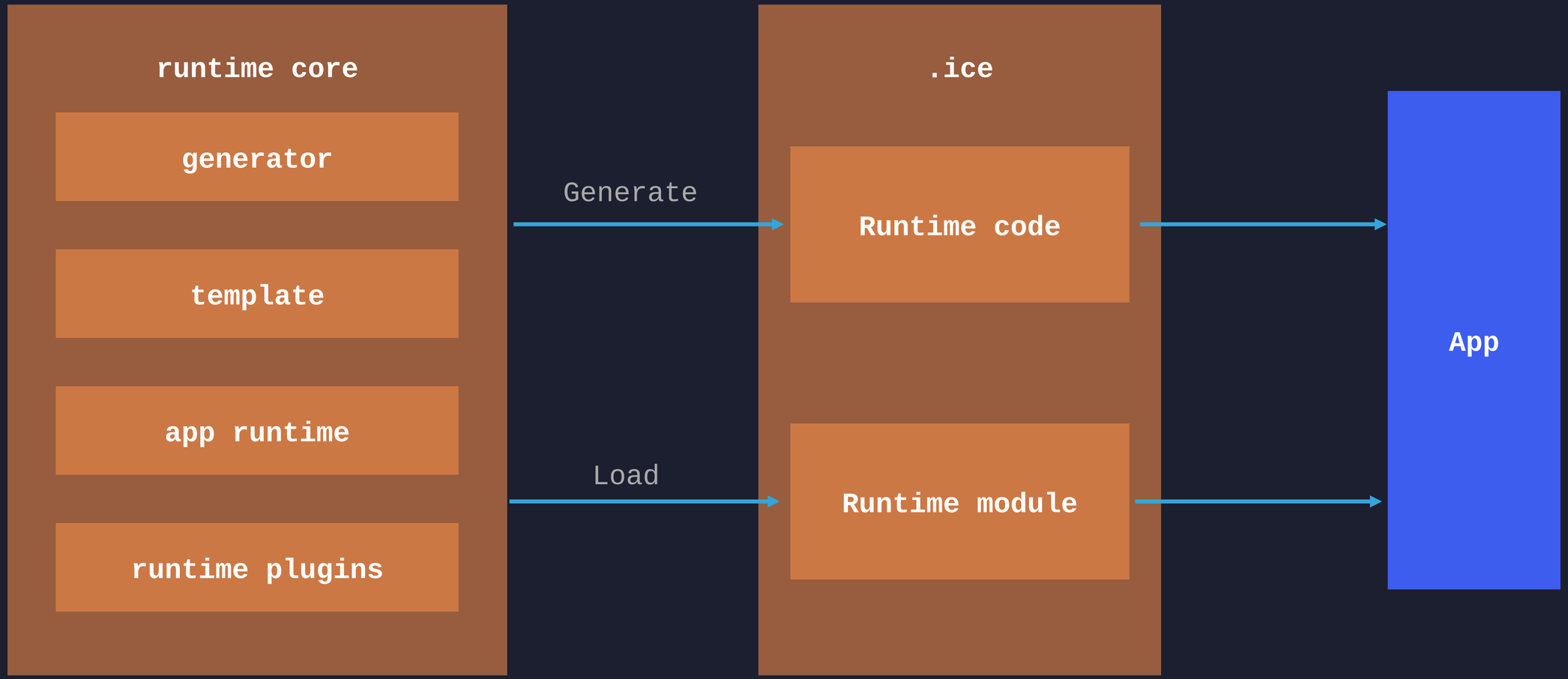
icejs 插件：工程能力扩展



icejs 插件：运行时能力扩展



icejs 插件：运行时能力扩展



icejs 插件：举个例子

工程插件：src/index.ts

```ts
import * as path from 'path';
import * as fse from 'fs-extra';

const PLUFIN_SPM_DIR = 'spm';
const FILE_NAME = 'types.ts';

export default function ({ applyMethod, getValue }) {
  const iceTempPath = getValue('ICE_TEMP');
  const typesSource = path.join(__dirname, '..', 'template', FILE_NAME);
  const typesTarget = path.join(iceTempPath, PLUFIN_SPM_DIR, FILE_NAME);

  applyMethod('addRenderFile', typesSource, typesTarget);

  applyMethod('addIceAppConfigAppTypes', {
    source: `./${PLUFIN_SPM_DIR}/types`,
    specifier: '{ ISpmA }',
    exportName: 'spmA?: ISpmA'
  });
}
```

运行时：src/runtime.ts

```ts
export default ({ appConfig, wrapperRouteComponent }) => {
  const { app } = appConfig;
  const { spmA } = app || {};
  wrapperRouteComponent(wrapperPage.bind(this, spmA));
}

function wrapperPage(spmA, PageComponent) {
  const { pageConfig } = PageComponent;
  const { spm: spmB } = pageConfig || {};

  const SpmWrapperedPage = (props) => {
    useEffect(() => {
      sendPageSPM([spmA, spmB]);
    }, []);

    return <PageComponent {...props} />;
  };
  return SpmWrapperedPage;
}
```

- 利用运行时 API 获取运行时配置，并为页面添加 SPM 发送逻辑
- 利用工程能力生成并导出 ts 类型

icejs 插件：举个例子

```
{
  "plugins": [
    "build-plugin-spm"
  ]
}
```

```
import { runApp, IAppConfig } from 'ice';

const appConfig: IAppConfig = {
  app: {
    spmA: 'a1z8w'
  }
};

runApp(appConfig);
```

```
import * as React from 'react';

const Home = (props) => {
  return <>Home</>
}

Home.pageConfig = {
  spm: '8005215'
}
export default Home;
```

配置工程插件

↓

开启运行时能力

↓

设置 SPM 内容

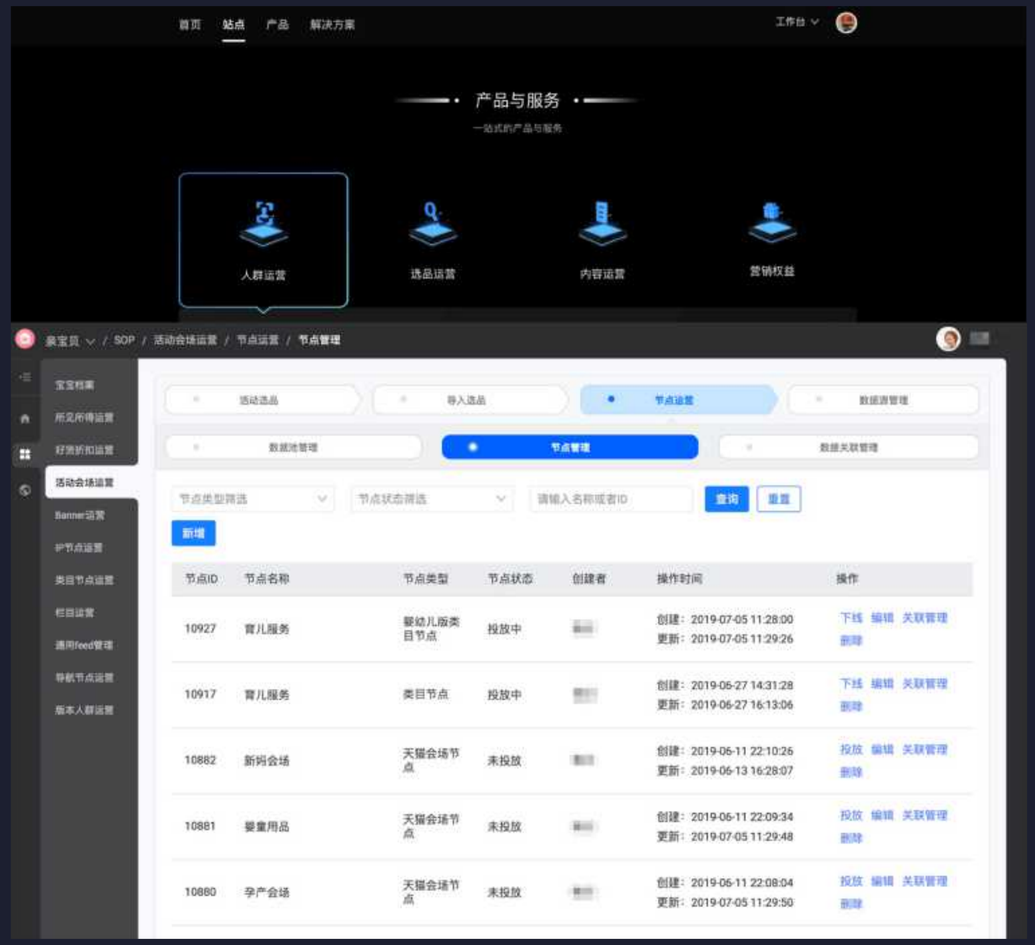
通过 icejs 插件让业务能力开箱即用

icejs 插件生态


扩展插件
plugin-faas
plugin-spm
plugin-icestark
plugin-fusion
plugin-antd
plugin-debug
plugin-esbuild
…
内置插件
plugin-ssr
plugin-store
plugin-logger
plugin-service
plugin-router
plugin-request
plugin-mpa
plugin-config
plugin-react-app
plugin-app-core
ice.js

微前端

业务背景



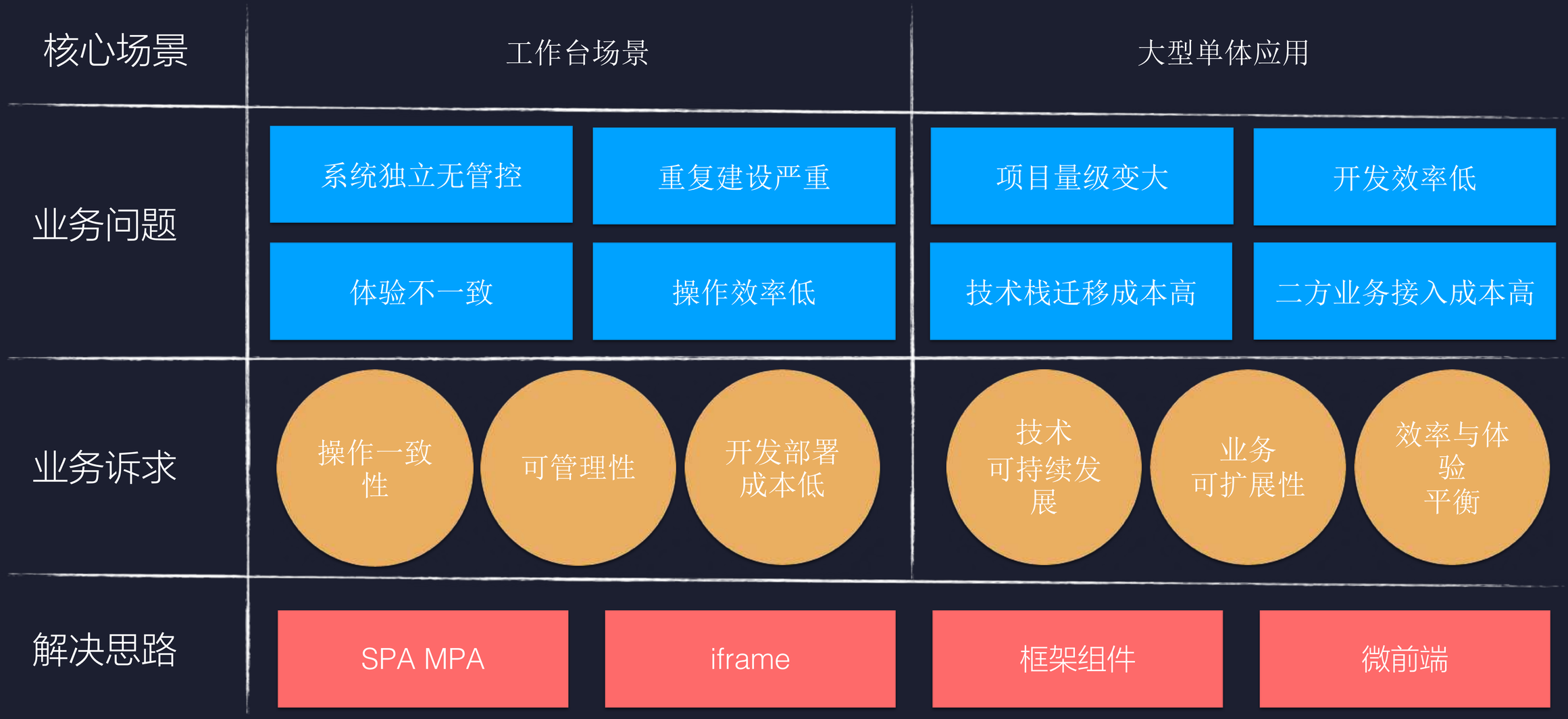
面向运营小二的系统很多且都是独立的

体验不一致

操作效率低

重复建设多

什么场景需要微前端？


核心场景
工作台场景
大型单体应用
业务问题
系统独立无管控
重复建设严重
项目量级变大
开发效率低
体验不一致
操作效率低
技术栈迁移成本高
二方业务接入成本高
业务诉求
操作一致性
可管理性
开发部署成本低
技术可持续发展
业务可扩展性
效率与体验平衡
解决思路
SPA MPA
iframe
框架组件
微前端

技术选型对比

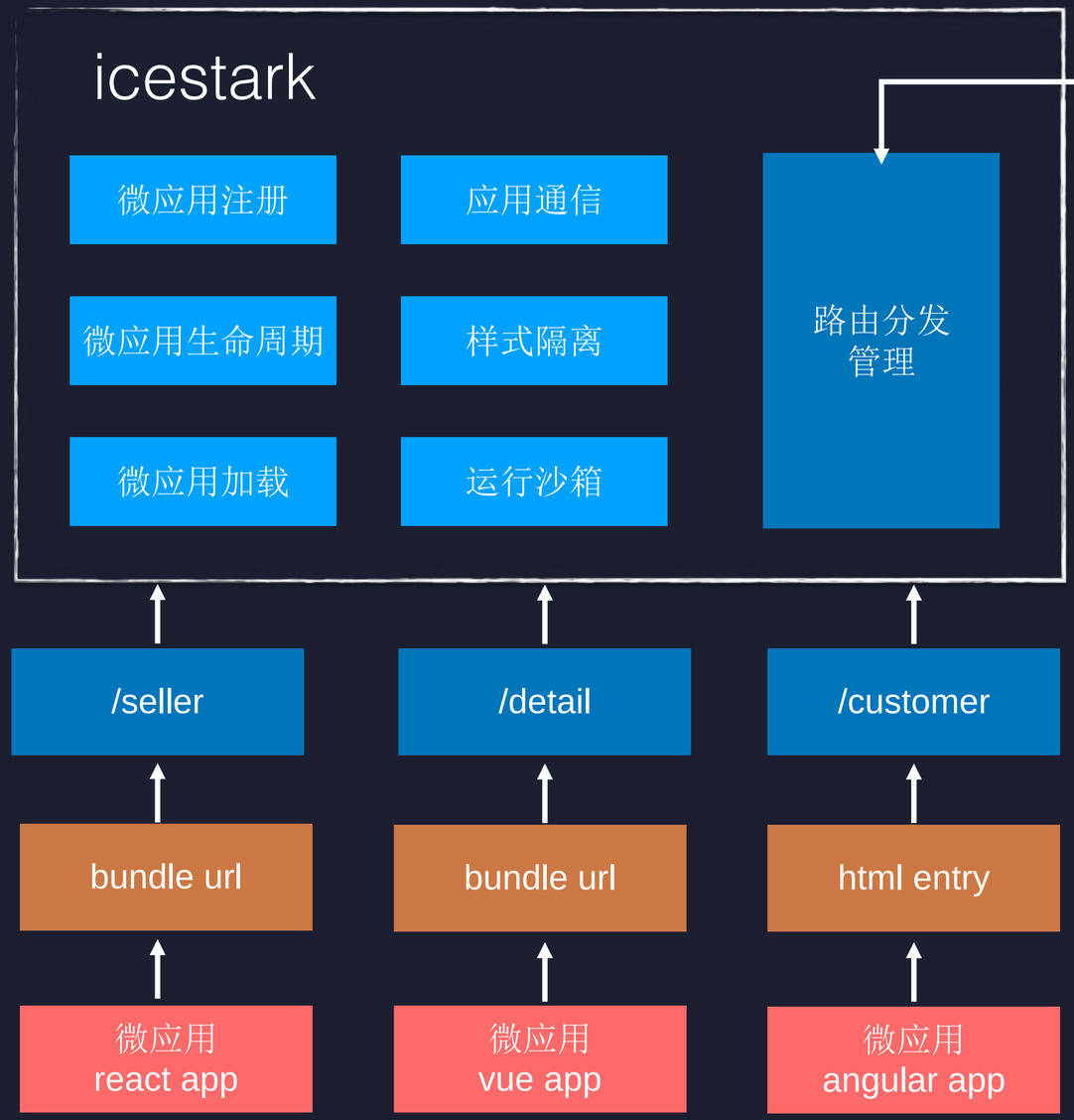
	SPA/MPA	iframe	框架组件	微前端
一个系统	√	√	x	√
用户体验	√	x	x	√
技术复杂度	√	x	x	x
系统健壮性	x	√	√	√
开发体验	x	√	√	√
二方接入	-	√	√	√
三方接入	x	√	√	x
微应用部署成本	-	x	x	√

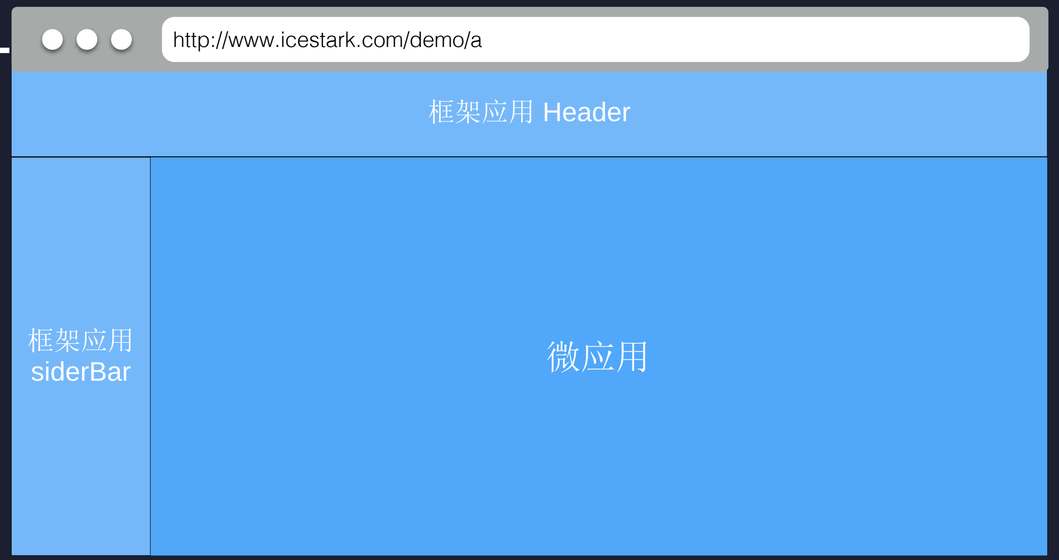
在大型系统的业务场景下，微前端能带来体验和效率的平衡

设计原则


单一体系技术栈统一
不用学习新概念新流程
技术栈无关
开发体验一致
中心化路由
独立开发部署
路由管理
应用分发
独立开发
独立部署

微前端 icestark 方案介绍





- **主应用：**负责 Layout 以及微应用配置与注册渲染，可以使用不同的框架
- **微应用：**按业务功能纬度拆分应用，通常是一个 SPA 应用，可以使用不同的框架

https://github.com/ice-lab/icestark

icestark 快速使用

应用接入

配置基准路由

注册微应用配置

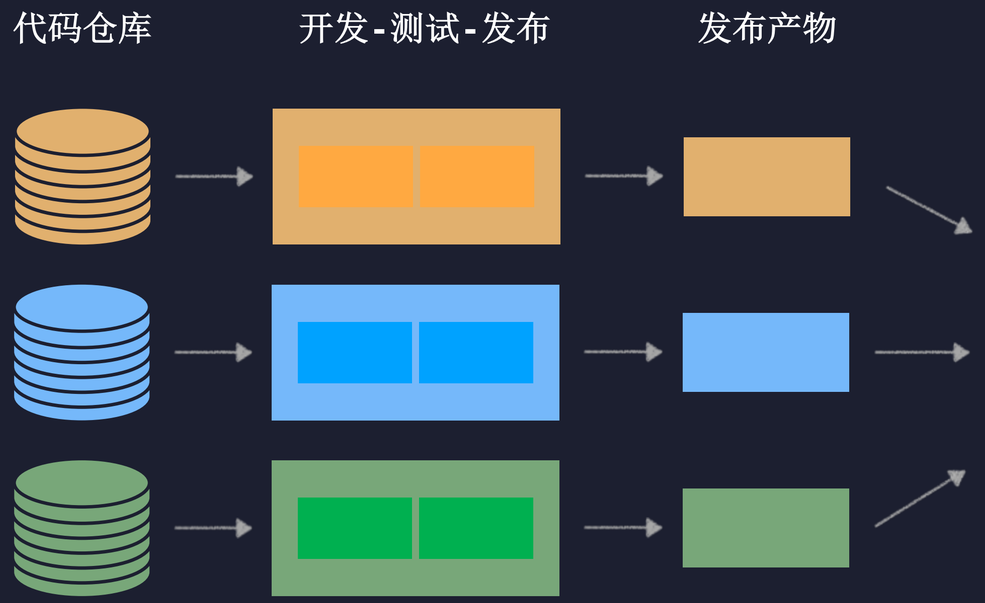
```
import ReactDOM f
import router fro
+ import { isInIc

+ if (isInIcestar
+   registerAppEnt
+     ReactDOM.ren
+   });
+   registerAppLea
+     ReactDOM.unm
+   });
+ } else {
ReactDOM.render(r
+ }
```

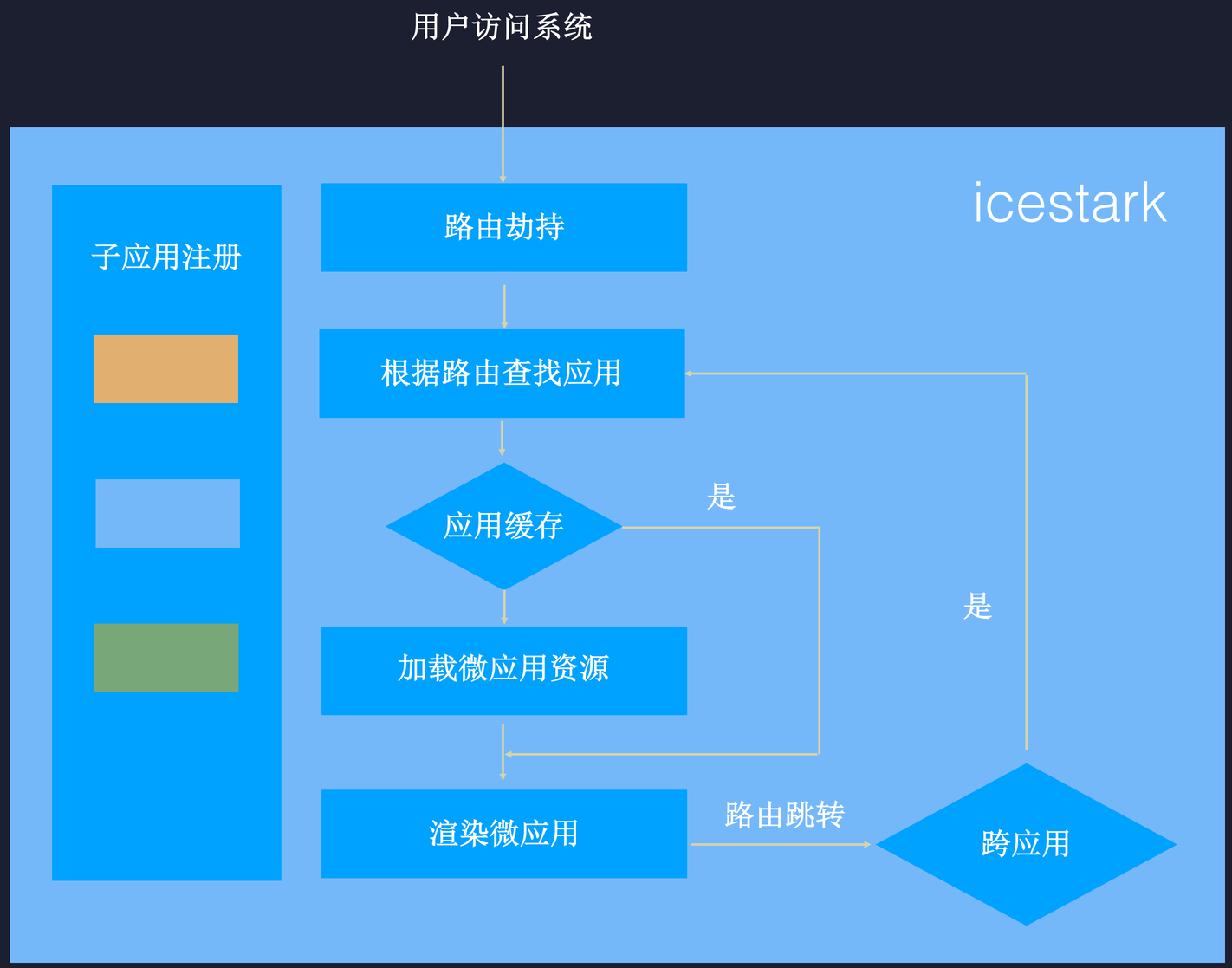
```
import React from 'react';
import { BrowserRouter as Router, Switch, Rou
+ import { getBasename } from '@ice/stark-app

export default () => {
  return (
-     <Router>
+     <Router basename={getBasename()}>
        <Switch>
          // ...
        </Switch>
      </Router>
  );
};
```

```
<AppRouter>
  <AppRoute
    path={['/', '/message', '/about']}
    exact
    title="通用页面"
    url={['//unpkg.com/icestark-child-common/build/js/index.js']}
  />
  <AppRoute
    path="/seller"
    title="商家平台"
    url={[
      '//unpkg.com/icestark-child-seller/build/js/index.js',
      '//unpkg.com/icestark-child-seller/build/css/index.css',
    ]}
  />
</AppRouter>
```

改造迁移成本低 研发流程零切换

icestark 工作原理

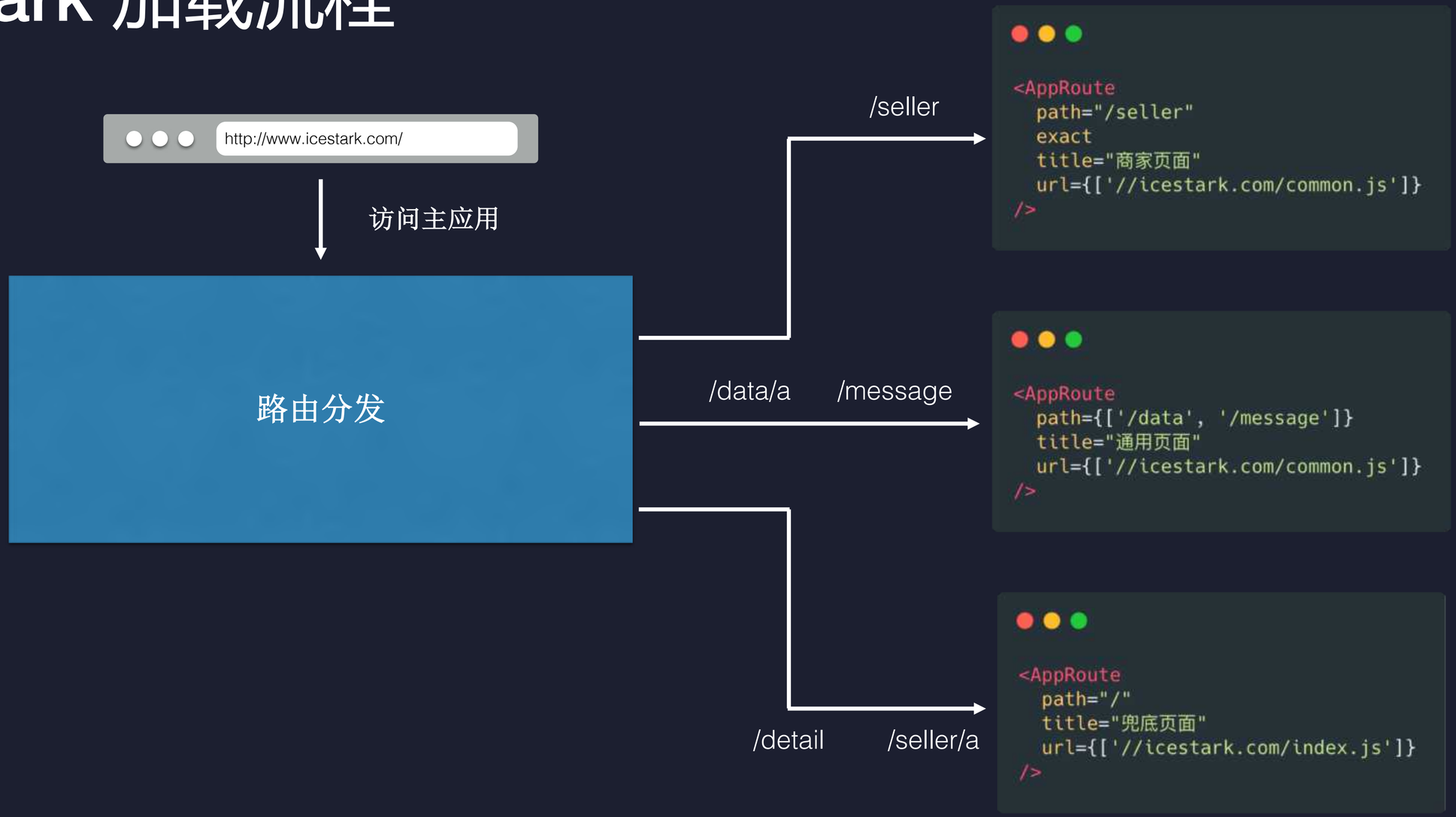




微应用独立开发，独立部署

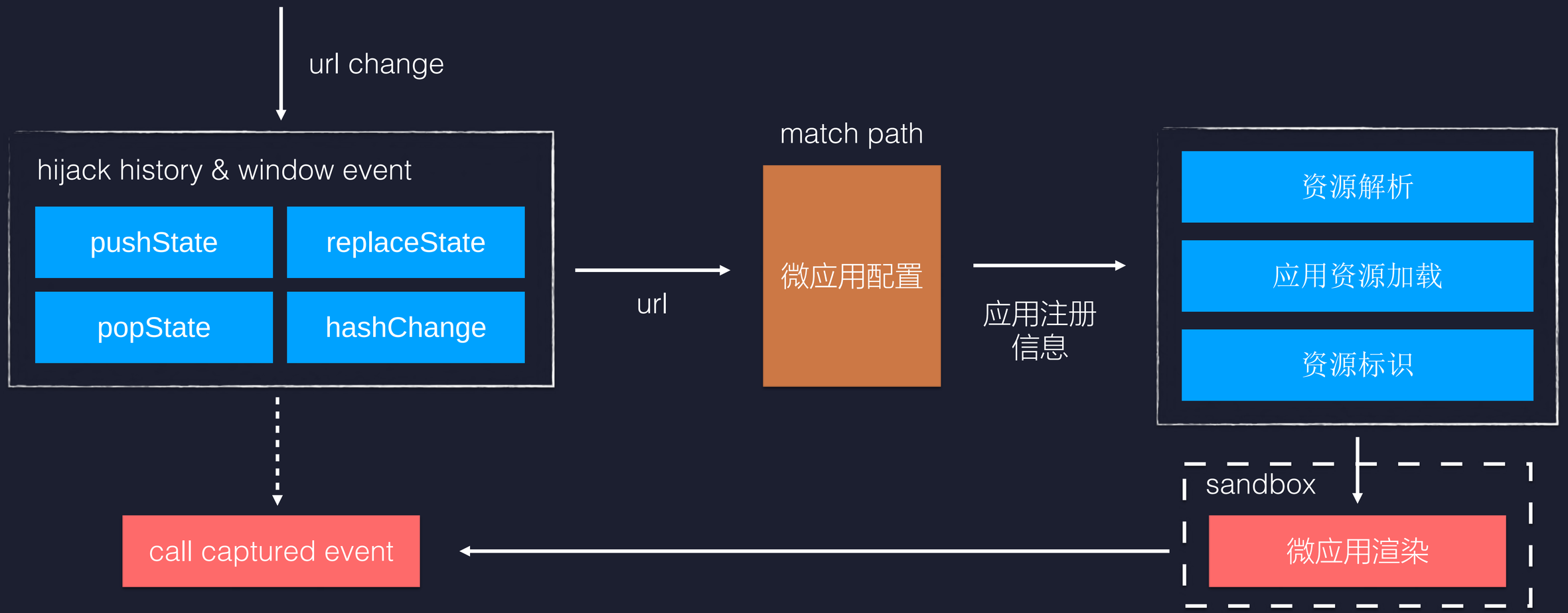
框架应用集成微应用达到系统一致性体验

icestark 加载流程



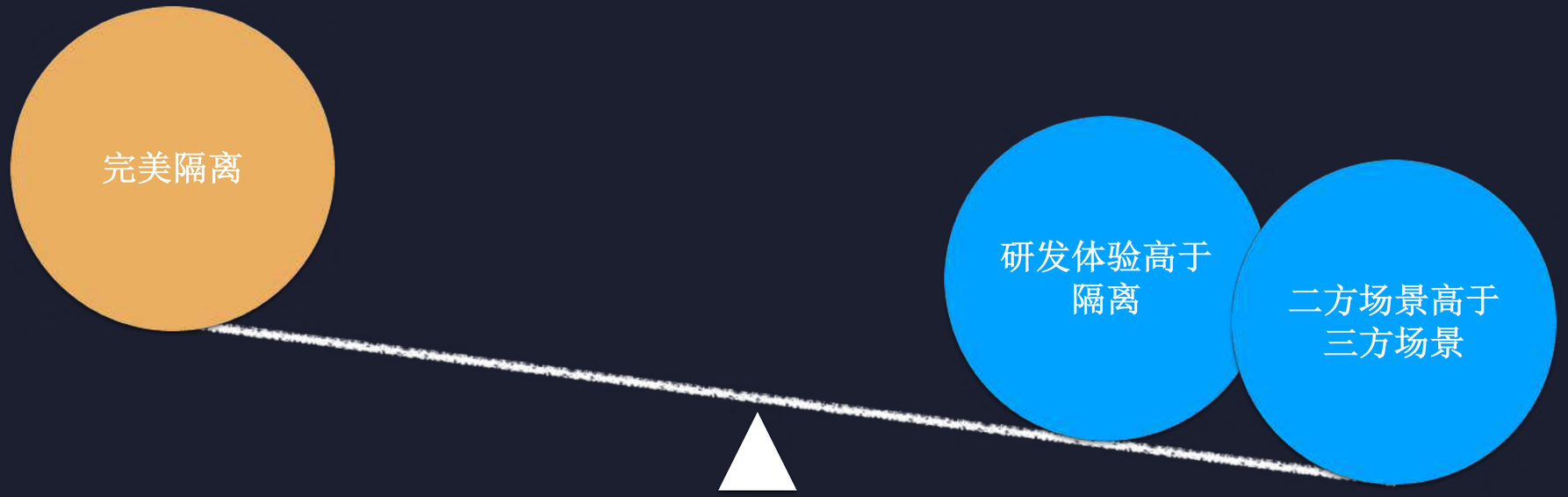
基准路由设置，快捷配置微应用

icestark 路由监听



通过路由劫持监听 url 变化，匹配资源进行加载
标识加载资源，在下一个应用加载前移除

icestark 隔离



icestark 样式隔离

业务代码样式隔离

CSS Modules

```
// CSS 样式
.test {
  color: #fff;
}

// CSS Modules 样式
.Homr--test-3XuHNoA {
  color: #fff;
}
```

基础组件样式隔离

基础组件 namespace

```
// 微应用基础样式
.next-btn {
  color: #fff;
}

// 框架应用 prefix 修改
.next-icestark-btn {
  color: #fff;
}
```

shadow DOM

处理逃逸容器节点

事件机制兼容

……

icestark 脚本隔离

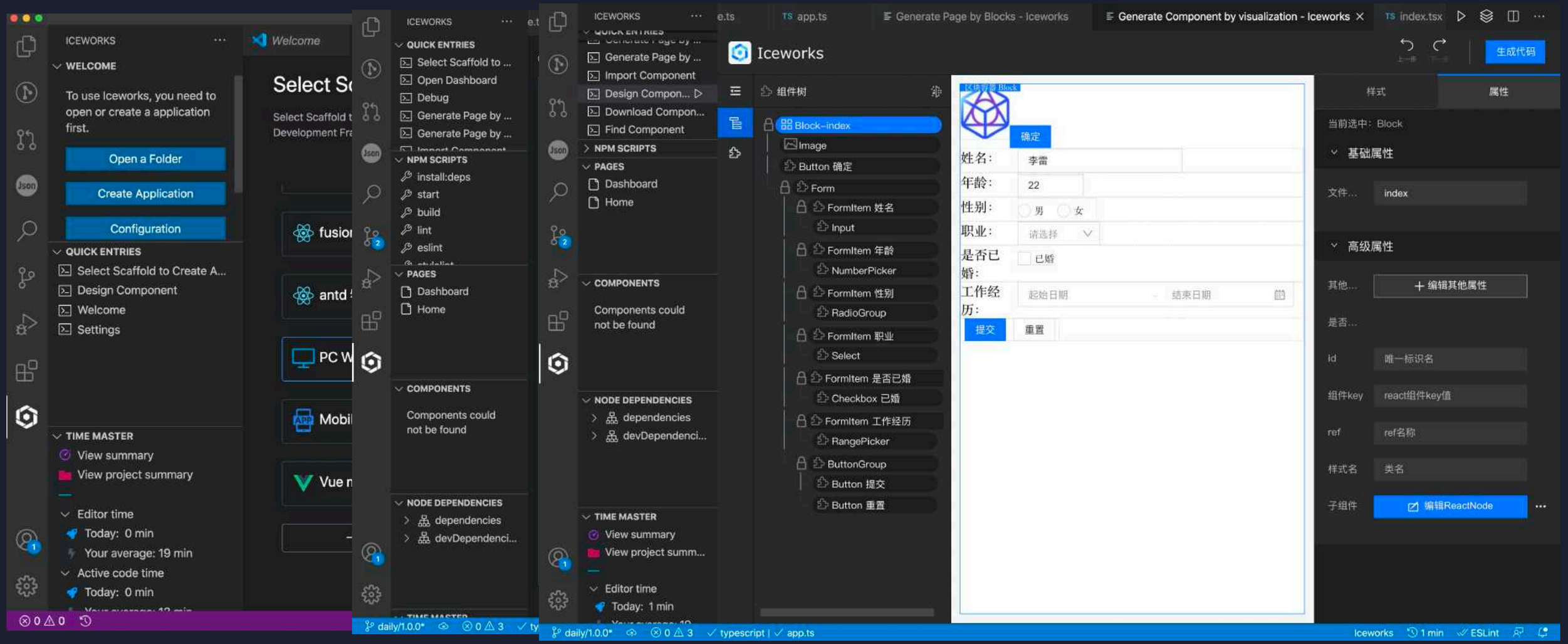
icestark 脚本隔离

三方接入方案

iframe

```
<AppRouter>
  <AppRoute
    path="/seller"
    url={[
      '//unpkg.com/icestark-child-seller/build/js/index.js',
      '//unpkg.com/icestark-child-seller/build/css/index.css',
    ]}
  />
  <AppRoute
    title="/project"
    entry="https://example.com/index.html"
  />
  <AppRoute
    path="/user"
    render={() => {
      return <iframe src="https://baidu.com" />
    }}
  />
</AppRouter>
```

研发工具 Iceworks



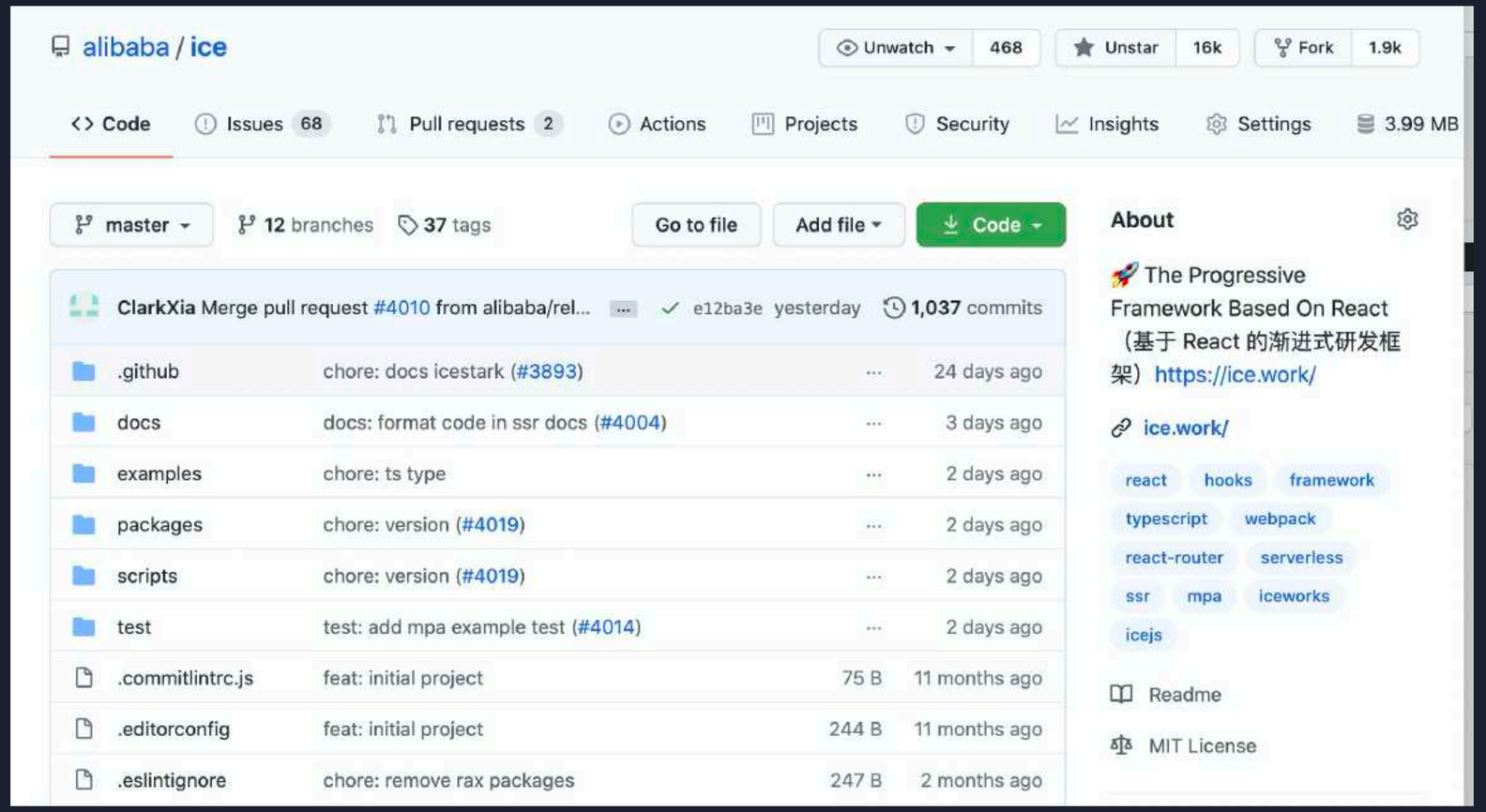
https://ice.work/

总结

- **体系分层**：框架 -> 工具 -> 平台 -> 业务
- **框架**： icejs 在通过插件能力提供工程和运行时能力定制，提供各种业务场景下的最佳解决方案
- **微前端**：在工作台以及大型单体应用下，基于 icestark 可以快速搭建起一套微前端的解决方案

欢迎关注开源社区

加入钉钉交流群



欢迎 star：https://github.com/alibaba/ice

THANKS

QCon+ 案例研习社