小程序跨端方案演进之路

京东 – Taro – 陈嘉健

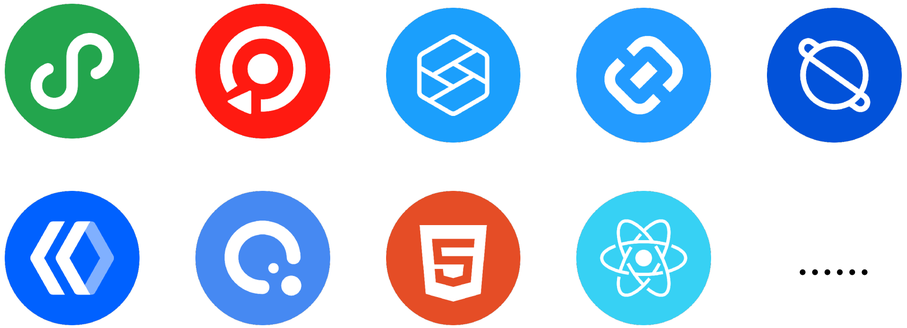
目录

01 背景

What & Why

这是一个百花齐放的时代

各大小程序平台层出不穷　产品发布的渠道越来越多



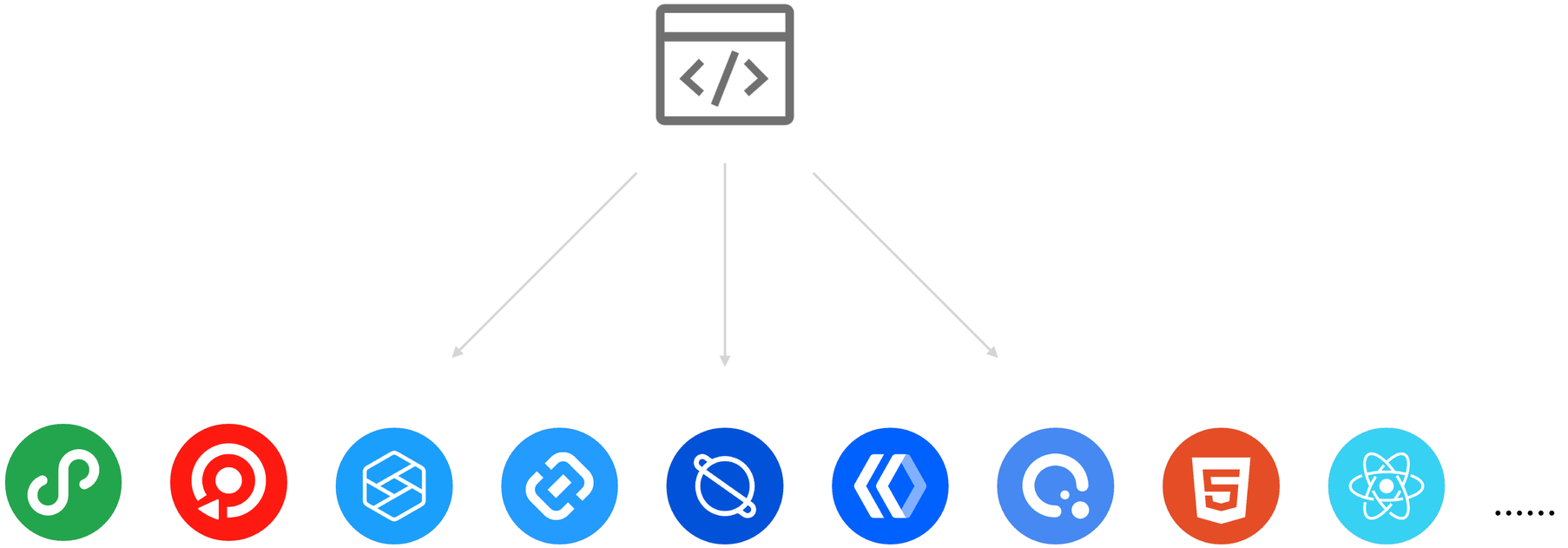
跨平台开发的种种痛点

业务响应不及时

研发成本增加

代码维护困难

一套代码，多平台运行？



开源成果

Taro

跨平台开发解决方案

Github Stars: 31500+

Github Issues: 9000+

Github commits: 21000+

开发者生态

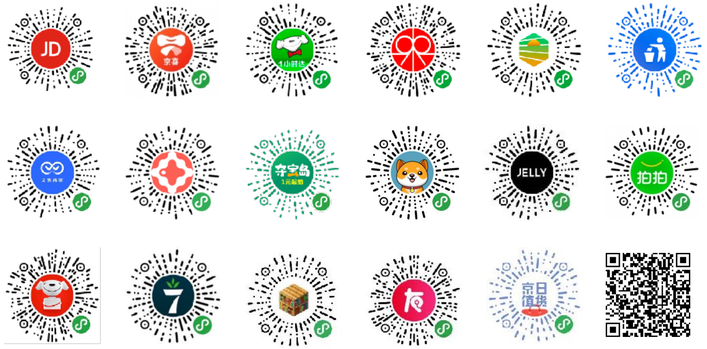
500+ 项目贡献者

20w+ 月度活跃开发者

70+ 开发者交流群

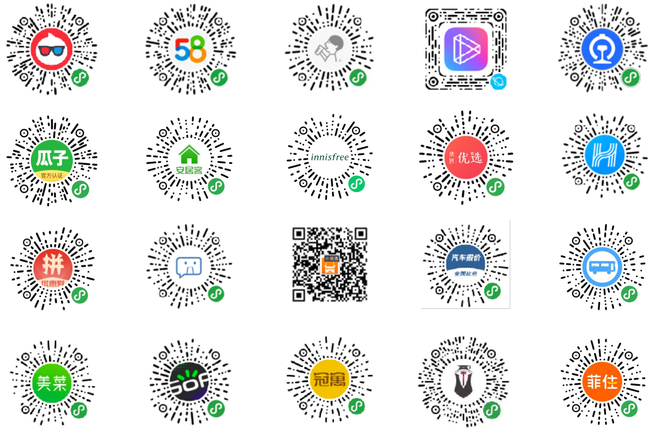
200+ 社区开发物料

内部应用：150+



与 60+ 业界头部企业团队展开合作

外部可公开大型应用：500+



框架特性

特性一：开发跨平台应用

Taro 让开发者使用前端技术栈即可开发出横跨各小程序、快应用、Web、OpenHarmony、React Native 等平台的应用，同时开发者只需要开发与维护同一套代码。

例子

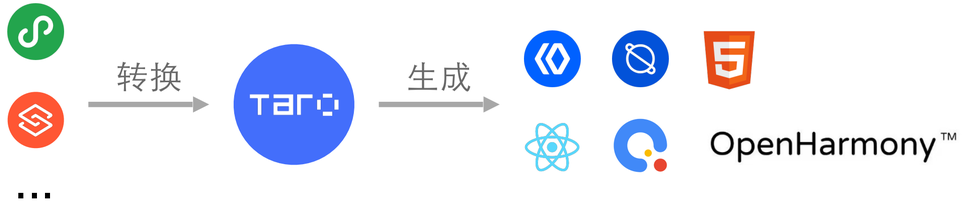
微信小程序

Web

React Native

特性二：让原生小程序跨平台运行

Taro 的反向转换功能让原生小程序可以借助 Taro 适配到各平台。能降低业务跨平台化的改造与维护成本，同时扩充如 OpenHarmony 等新平台的应用生态。



特性三：让 Web 应用跨平台运行

Taro 可以让 Web 端的生态工具、Web 应用直接运行在小程序、快应用以及 OpenHarmony 平台。

02 Taro 1 & 2

统一标准，多平台适配

跨平台开发需要解决的问题

语法差异

组件库差异

API 差异

研发生态复用

各端之间巨大的语法差异

React

```jsx
import React, { Component } from 'react'
import { Text, View } from 'react-native'

export class PageA extends Component {
  componentWillMount () { }

  componentDidMount () { }

  render () {
    return (
      <View>
        <Text>
          页面 A
        </Text>
      </View>
    )
  }
}
```

MVVM

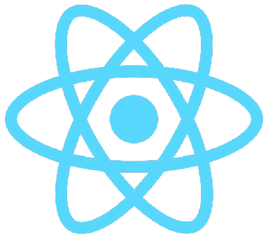
```js
Page({
  data: {
    text: 'This is page A.'
  },

  onLoad(options) { },

  onReady() { },
  // Event handler.
  viewTap() {
    this.setData({
      text: 'Set some data for updating view.'
    }, function () {
      // this is setData callback
    })
  }
})
```

```xml
<view>
  {this.text}
</view>
```

SFC

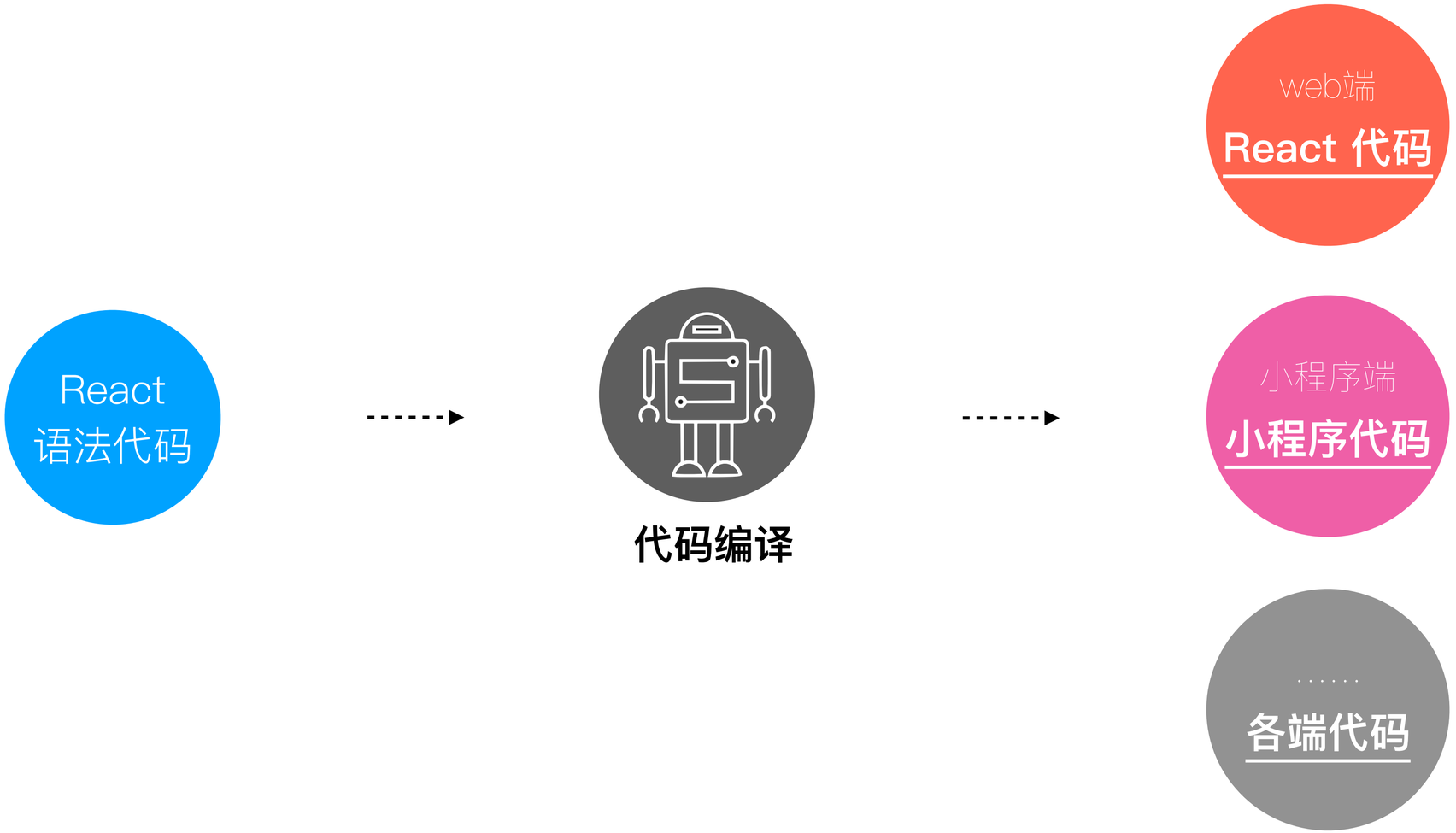
```vue
<template>
  <!-- template里只能有一个根节点 -->
  <div class="a">
    <text>页面 A</text>
  </div>
</template>

<script>
  import router from '@system.router'

  export default {
    private: {
      title: '页面 A'
    },
    onInit () { },

    onReady () { }
  }
</script>
```

统一语法标准



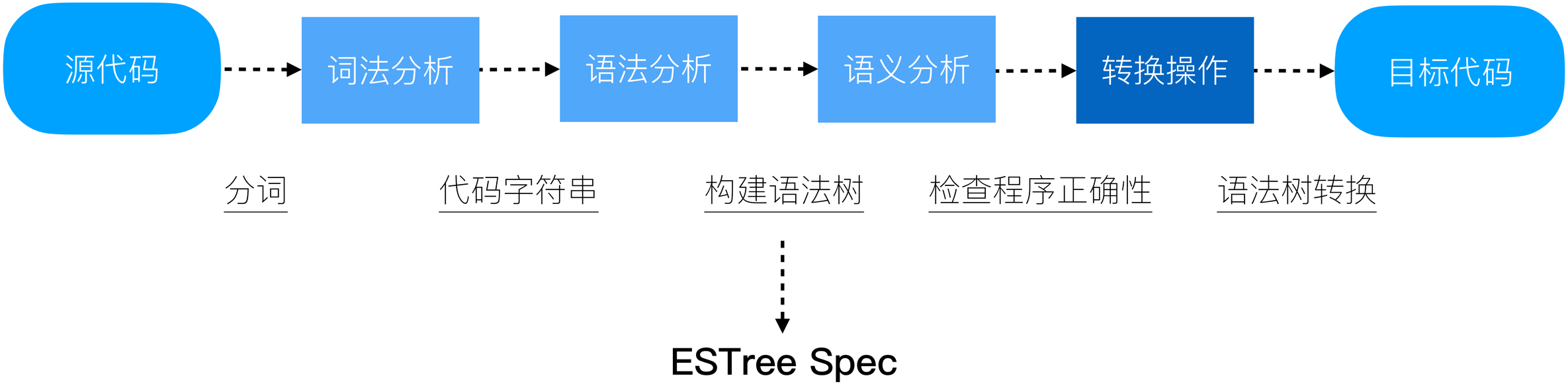
定义一套前端熟悉的语法规范

让 React 语法跨平台 - 思路



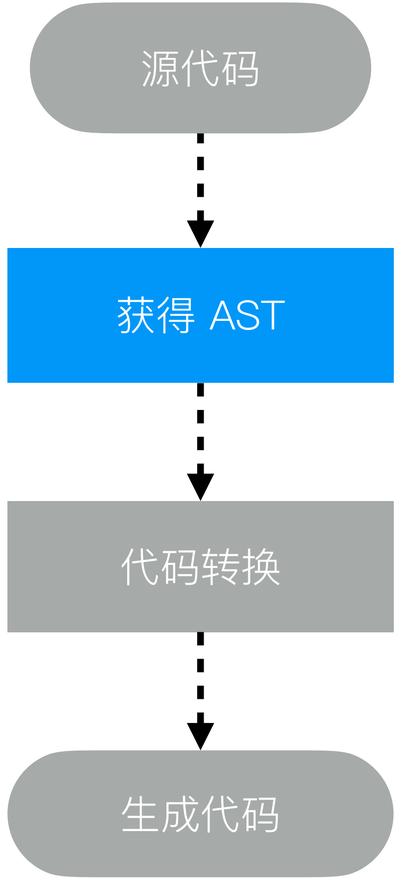
让 React 语法跨平台 - 实现

对 React 代码进行编译转换



让 React 语法跨平台 - 实现

@babel/core	解析代码获取 AST
@babel/types	AST 节点定义，生成节点
@babel/traverse	递归遍历操作 AST
@babel/generator	AST 生成源码

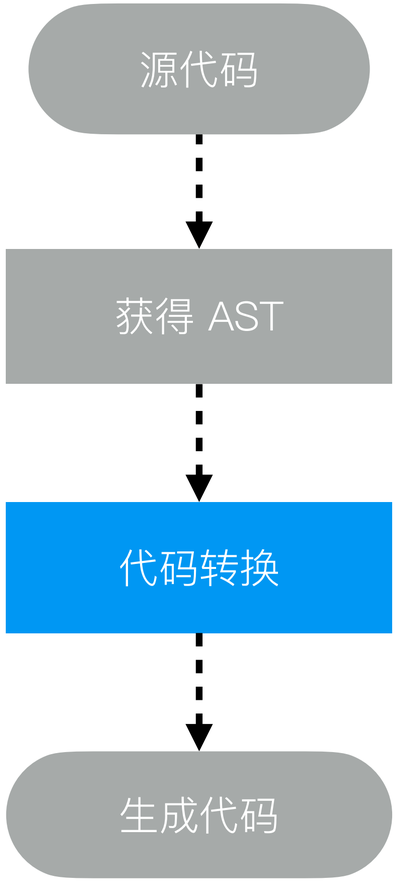


让 React 语法跨平台 - 实现

```
import Taro, { Component } from '@tarojs/taro'

class MyComponent extends Component {
  state = { title: '这是一个标题'}

  render () {
    return (
      <View className='my-component'>
        <View>{this.state.title}</View>
      </View>
    )
  }
}
```

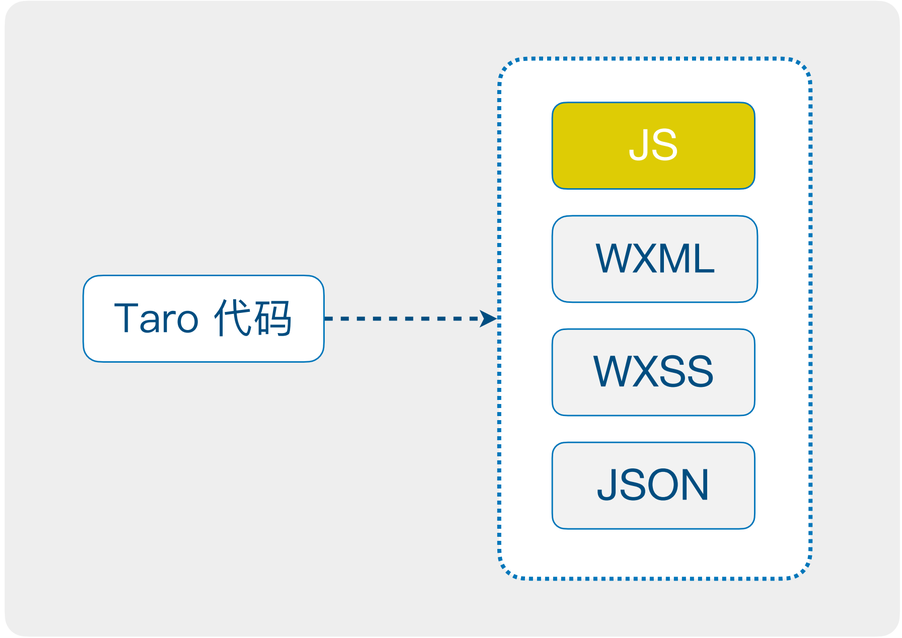
解析转换

```
<view class="my-component">
  <view>{title}</view>
</view>
```

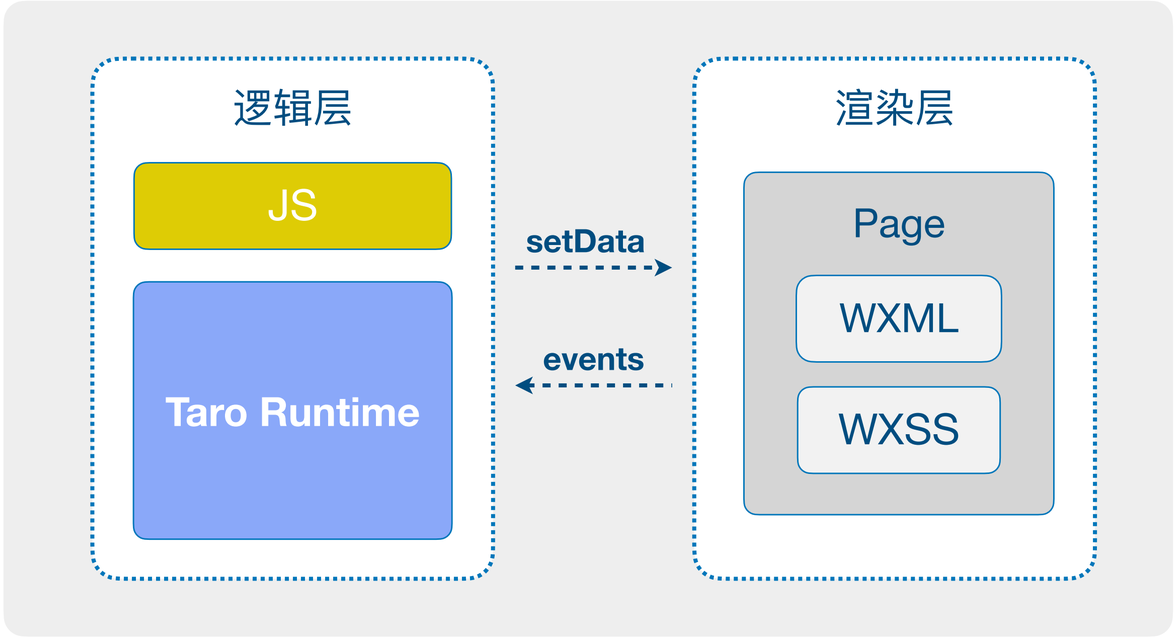


让 React 语法跨平台 - 实现

编译时



运行时



跨平台开发需要解决的问题

语法差异

组件库差异

API 差异

开发生态复用

组件库差异

小程序组件

组件	作用
view	容器组件
text	文子组件
image	图片组件
icon	图标组件
swiper	轮播组件
button	按钮组件
…	…

HTML 标签

标签	作用
div	块级元素标签
span	行内元素标签
a	超链接标签
table	表格标签
ul	无序列表
p	段落标签
…	…

RN 组件

组件	作用
View	容器组件
Text	文本组件
Image	图片组件
TextInput	文本输入组件
ScrollView	可滚动组件
Picker	选择器组件
…	…

API 差异

小程序 API

组件	作用
wx.request	网络请求
wx.uploadFile	上传文件
wx.downloadFile	下载文件
wx.login	微信登录
wx.getStorage	获取本地存储
wx.setStorage	设置本地存储
…	…

Web API

组件	作用
window.fetch	网络请求
window.localStorage	本地存储相关
window.open	打开新页面
window.scrollTo	页面滚动
window.navigator	浏览器相关信息
window.location	文档当前位置信息
…	…

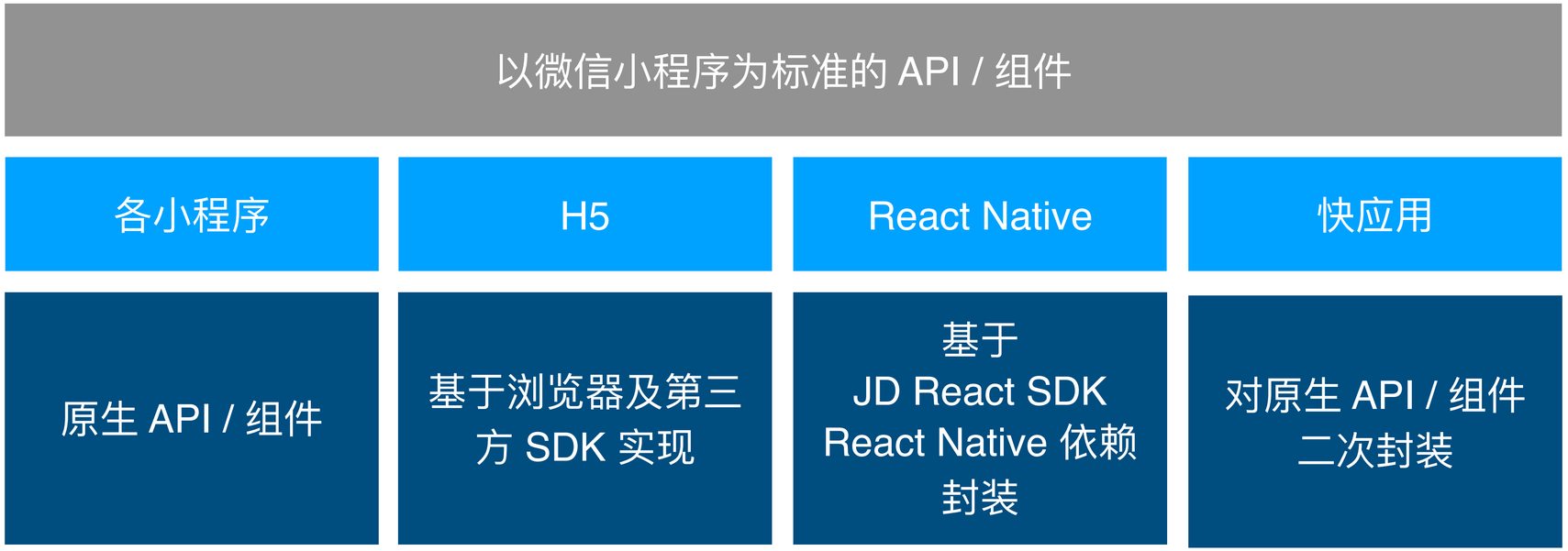
RN API

组件	作用
Alert.alert	启动对话框
AccessibilityInfo	检查读屏应用
Animated	动画相关
Clipboard	剪切板
AsyncStorage	key-value 存储
NetInfo	网络信息
…	…

统一组件库、API 标准

以微信小程序的组件库、API 规范作为标准

实现跨平台的组件库、API



跨平台开发需要解决的问题

语法差异

组件库差异

API 差异

研发生态复用（开发 & 构建）

复用 React 开发生态

魔改后支持

react-redux

react-redux-taro

mobx

mobx-taro

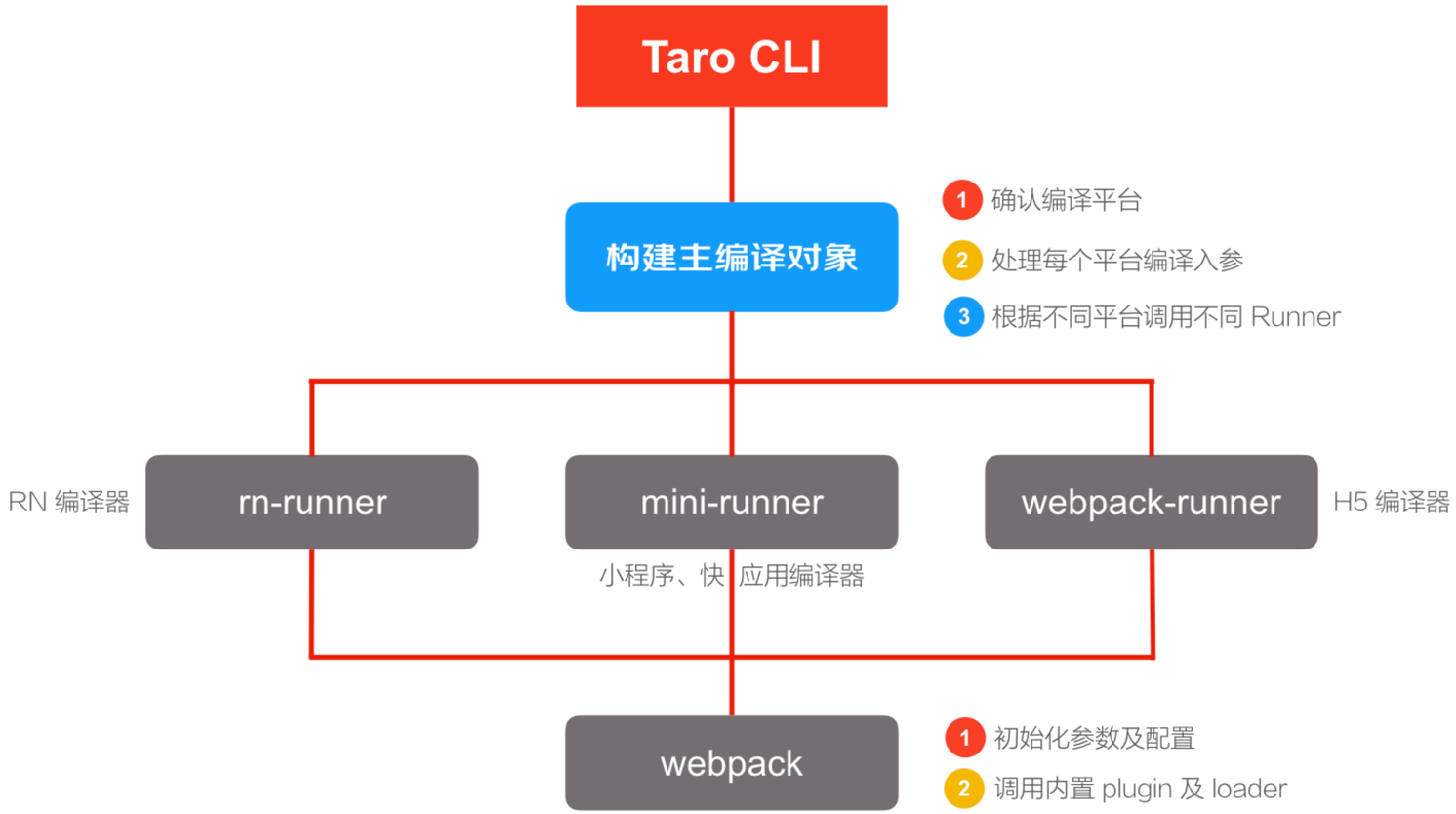
不支持

antd-mobile

react-router

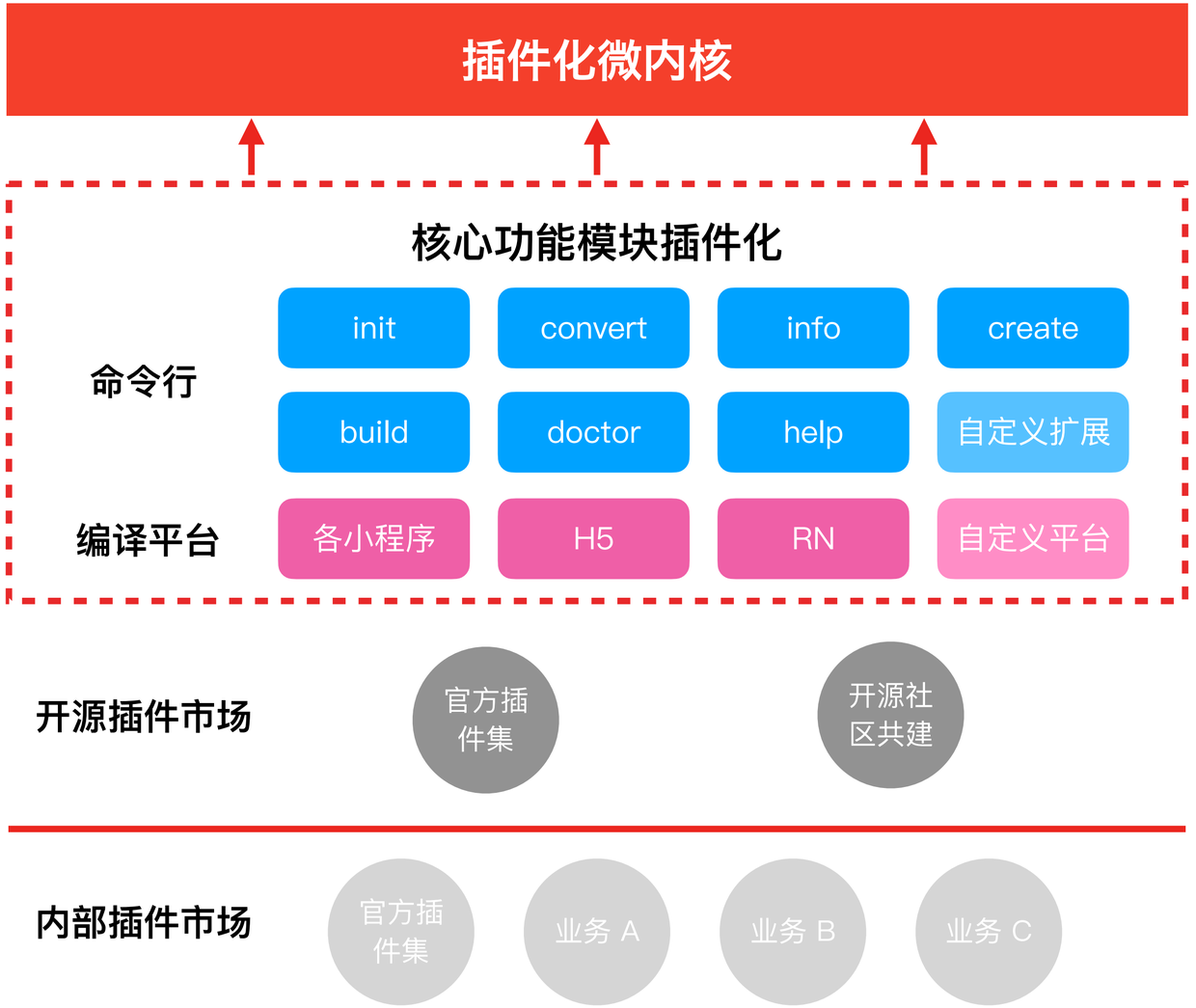
react-query

Taro 2 打造基于 Webpack 的构建系统



各端参数一致，通过 webpackChain 来实现完全自定义 webpack 配置

Taro 2 引入插件化机制



问题与反思

不能 100% 兼容 React 语法

难以复用 React 生态

不支持使用 Vue

03 Taro 3

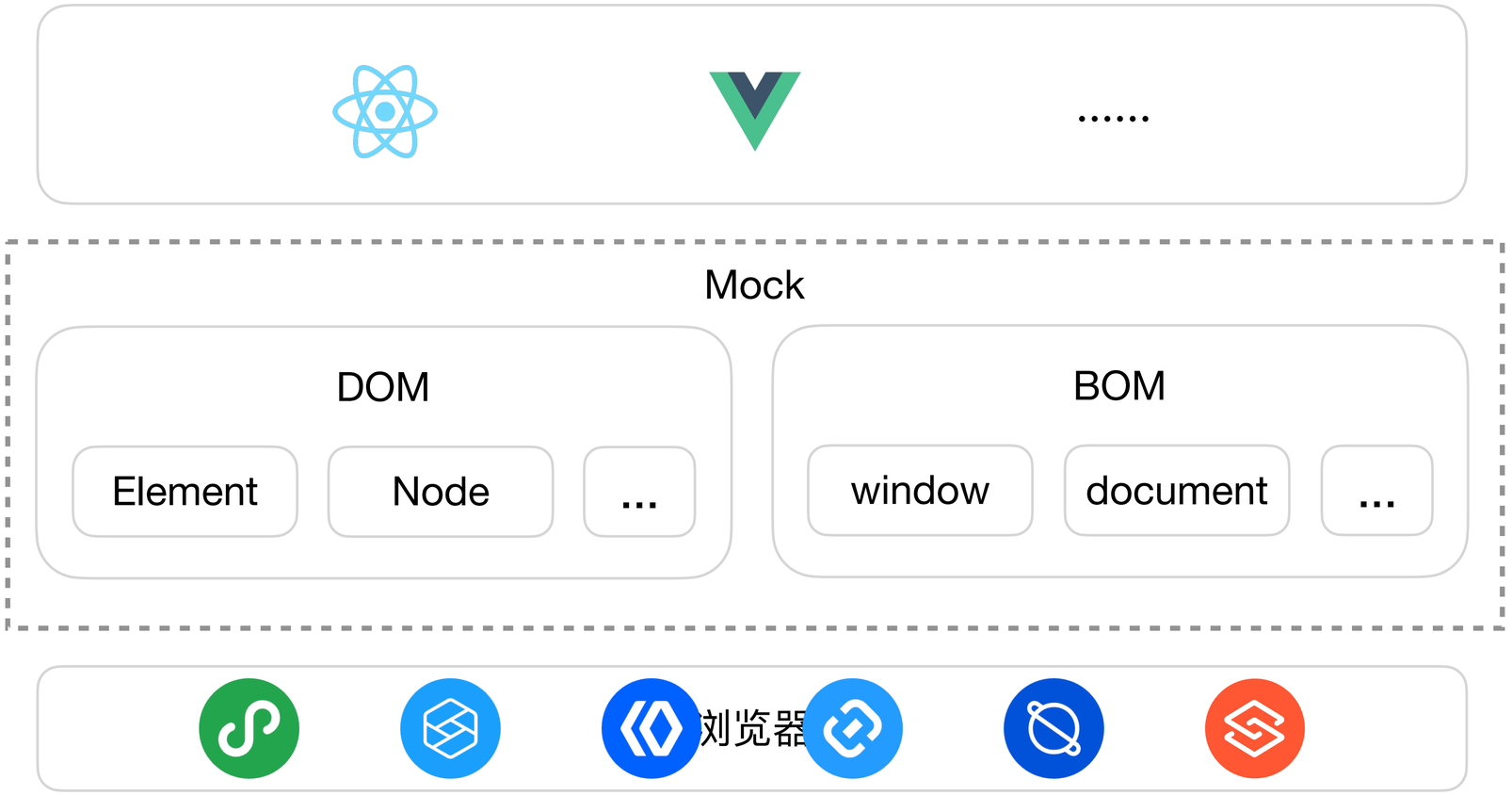
对 Web 开发生态的极致兼容

Taro 3 架构 - 设计思路

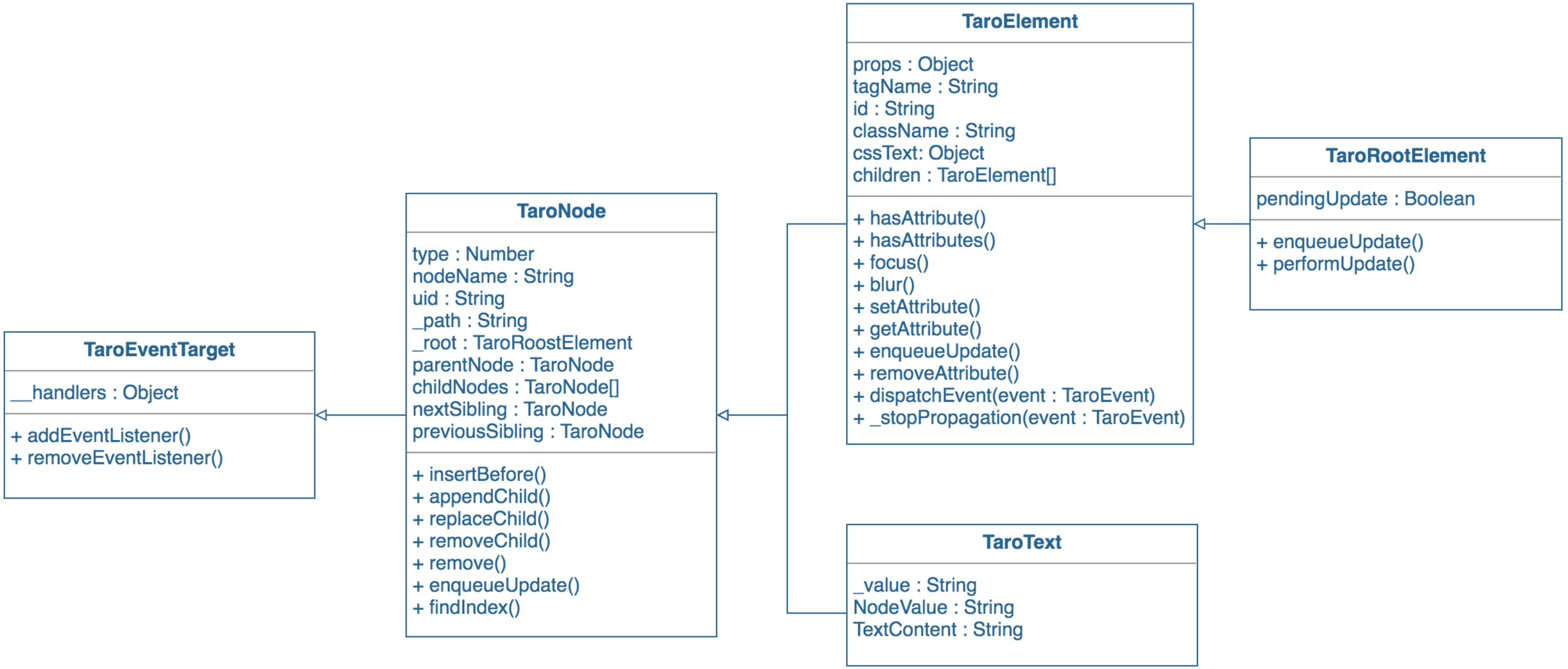
除了编译的方式，有没有其它可以处理语法差异的方法?

Taro 3 架构 - 设计思路

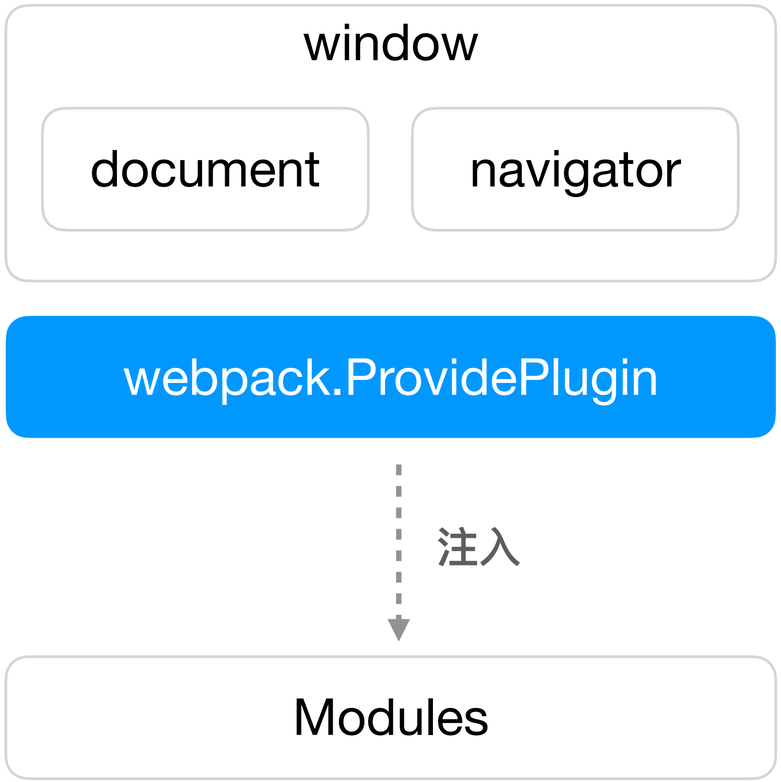
模拟实现浏览器环境，让 React、Vue、Web 生态库直接运行



Taro 3 架构 - 模拟 DOM

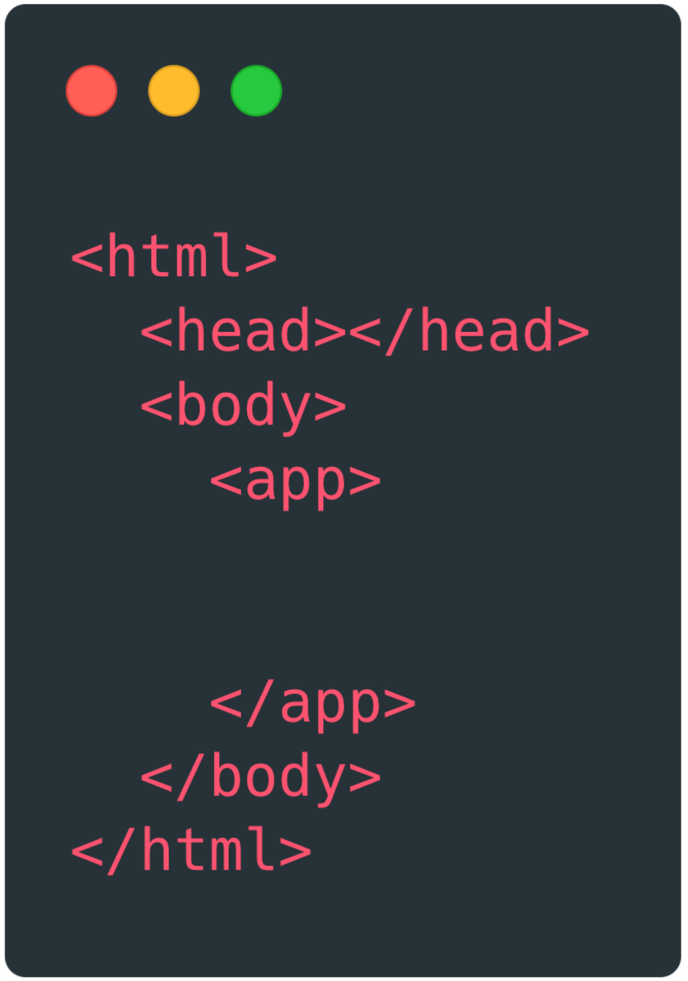


Taro 3 架构 - 模拟 BOM

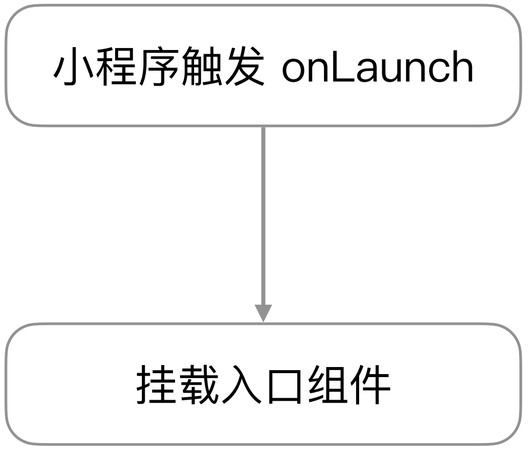



```
plugin.providerPlugin = getProviderPlugin({
  window: ['@tarojs/runtime', 'window'],
  document: ['@tarojs/runtime', 'document'],
  navigator: ['@tarojs/runtime', 'navigator'],
  requestAnimationFrame: ['@tarojs/runtime', 'requestAnimationFrame'],
  cancelAnimationFrame: ['@tarojs/runtime', 'cancelAnimationFrame']
})
```

Taro 3 架构 - 初始化 DOM 树

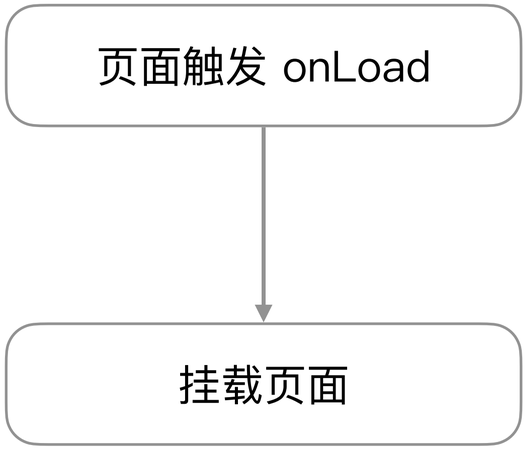


Taro 3 架构 - 挂载 App 入口组件




```
<html>
  <head></head>
  <body>
    <app>
      <App />
    </app>
  </body>
</html>
```

Taro 3 架构 - 挂载页面

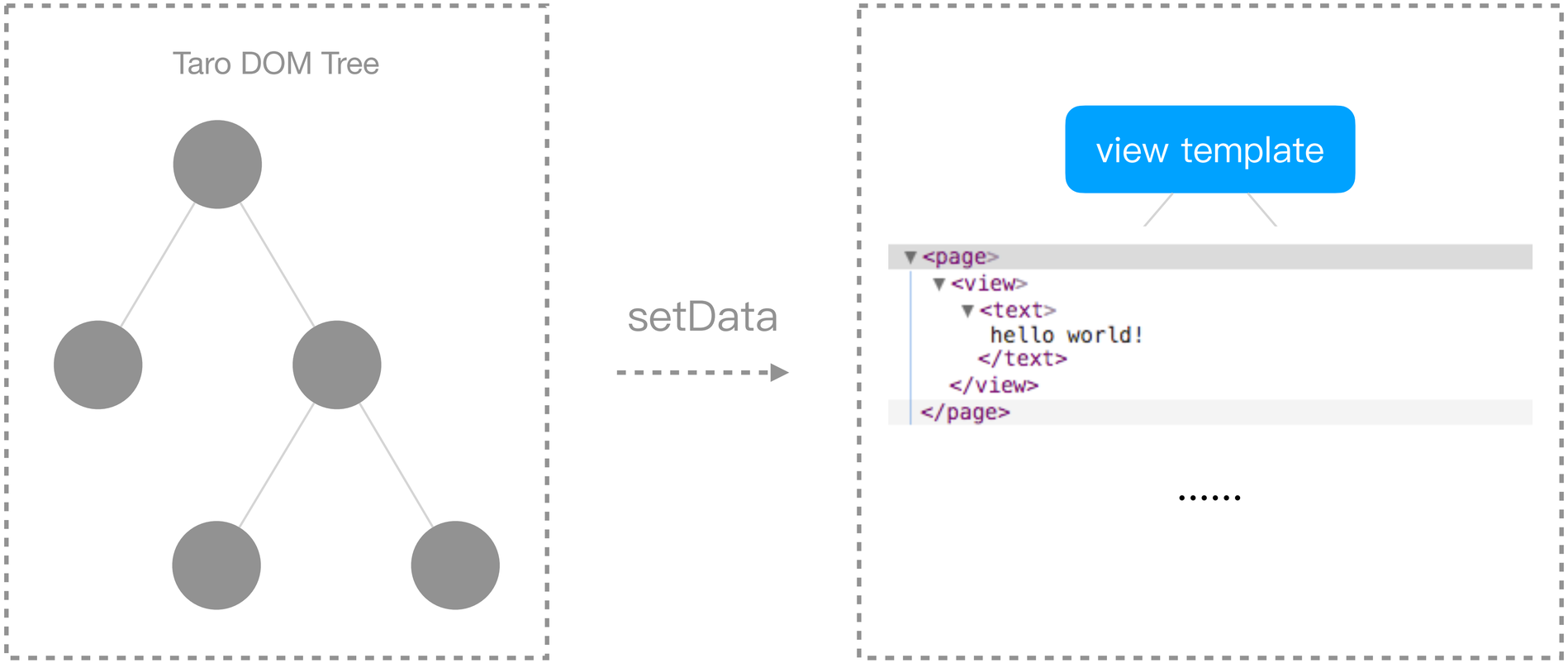



```
<html>
  <head></head>
  <body>
    <app>
      <App>
        <Page />
      </App>
    </app>
  </body>
</html>
```

Taro 3 架构 - 渲染方案

逻辑层

视图层



Taro 3 架构 - 渲染流程

App Service（逻辑层）

React/Vue

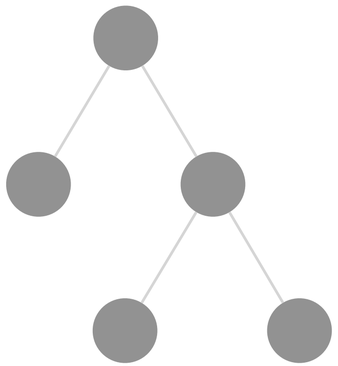
1. 调用

Taro 运行时

- BOM
- DOM
- API

2. 渲染

Taro 虚拟 DOM

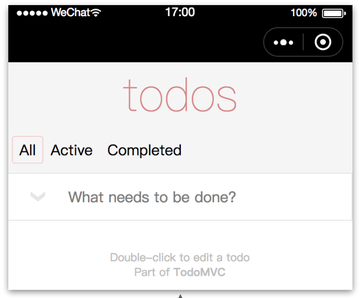


DOM 树

DOM 树的数据结构

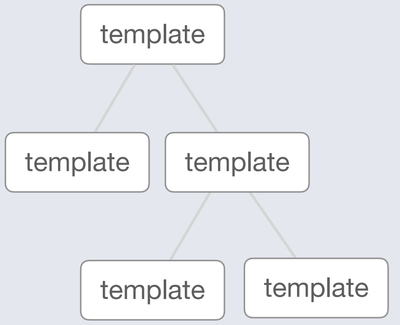
3. setData

View（视图层）



4. 渲染

XML

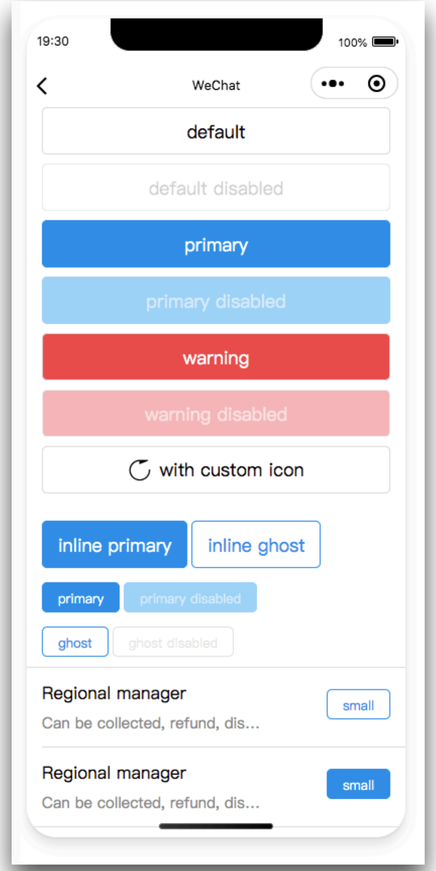


Taro 3 的优势

Taro 1 & 2	不能 100% 兼容 React 语法	不支持使用 Vue	难以复用 React 生态
Taro 3	运行真正的 React	支持使用 Vue	支持复用大部分 Web 生态

拥抱 Web 生态 - H5 同构

Antd Mobile



Vant UI

WeUI

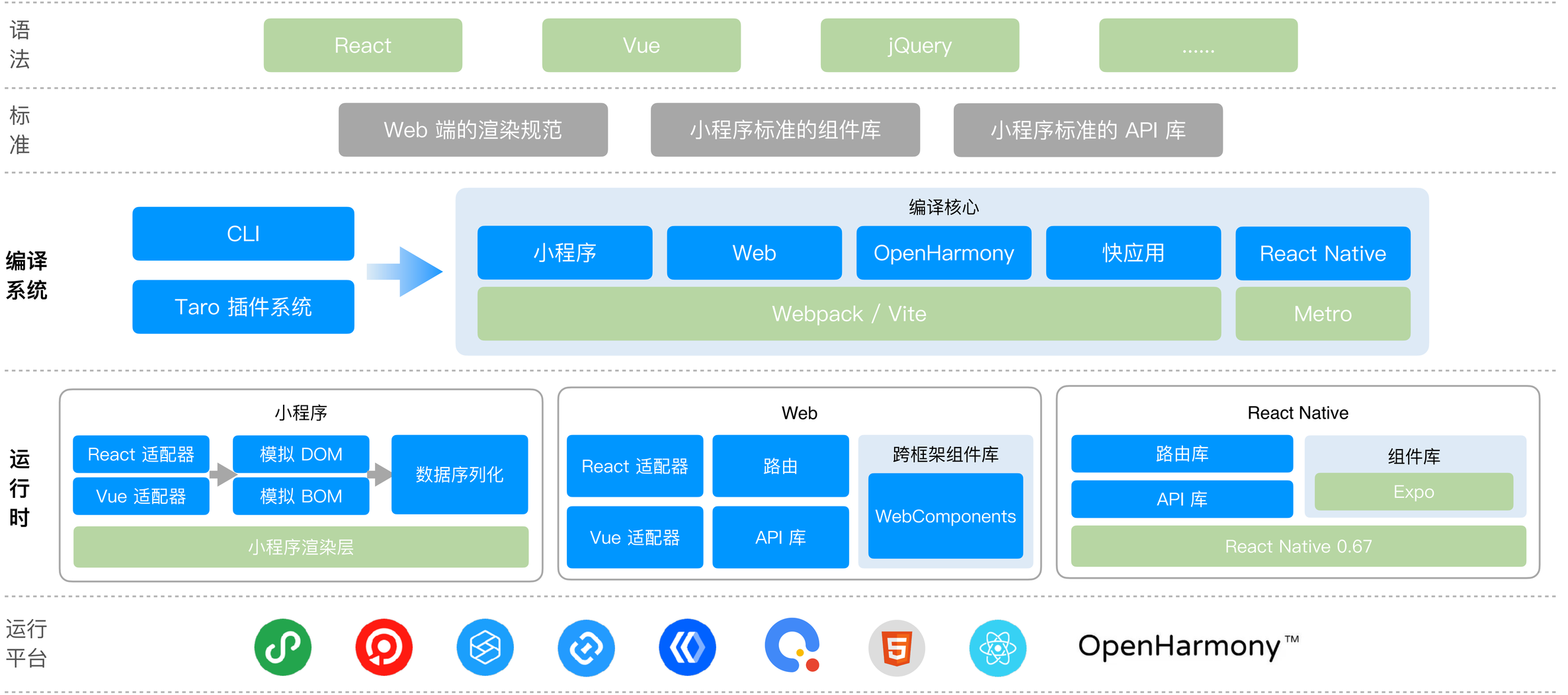
开放式架构 - 支持拓展到任意平台

Taro 适配鸿蒙方案 - 鸿蒙插件

Taro 适配鸿蒙方案 - 思路

差异	 语法差异	 组件库差异	 API 差异
规范	React & Vue	微信小程序规范	微信小程序规范
方案思路	在鸿蒙模拟实现 Web 环境	使用鸿蒙组件去实现	使用鸿蒙 API 去实现

Taro 架构图


语法
React
Vue
jQuery
.....
标准
Web 端的渲染规范
小程序标准的组件库
小程序标准的 API 库
编译系统
CLI
Taro 插件系统
编译核心
小程序
Web
OpenHarmony
快应用
React Native
Webpack / Vite
Metro
运行时
小程序
React 适配器
Vue 适配器
模拟 DOM
模拟 BOM
数据序列化
小程序渲染层
Web
React 适配器
Vue 适配器
路由
API 库
跨框架组件库
WebComponents
React Native
路由库
API 库
组件库
Expo
React Native 0.67
运行平台
OpenHarmony™

04 展望未来

Future

W3C 小程序标准有助于降低跨平台适配的成本

MiniApp Manifest

W3C Working Draft 01 July 2022

▶ More details about this document

MiniApp Packaging

W3C Working Draft 29 July 2022

▶ More details about this document

MiniApp Lifecycle

W3C Working Draft 18 August 2022

▶ More details about this document

Taro 开发规划

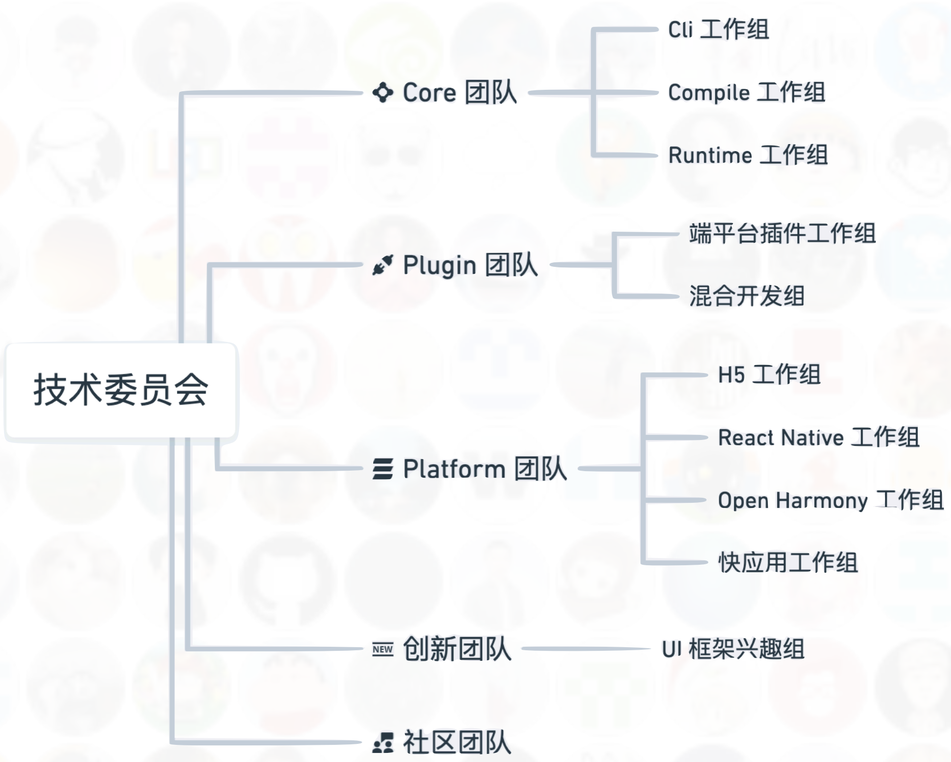
Vite

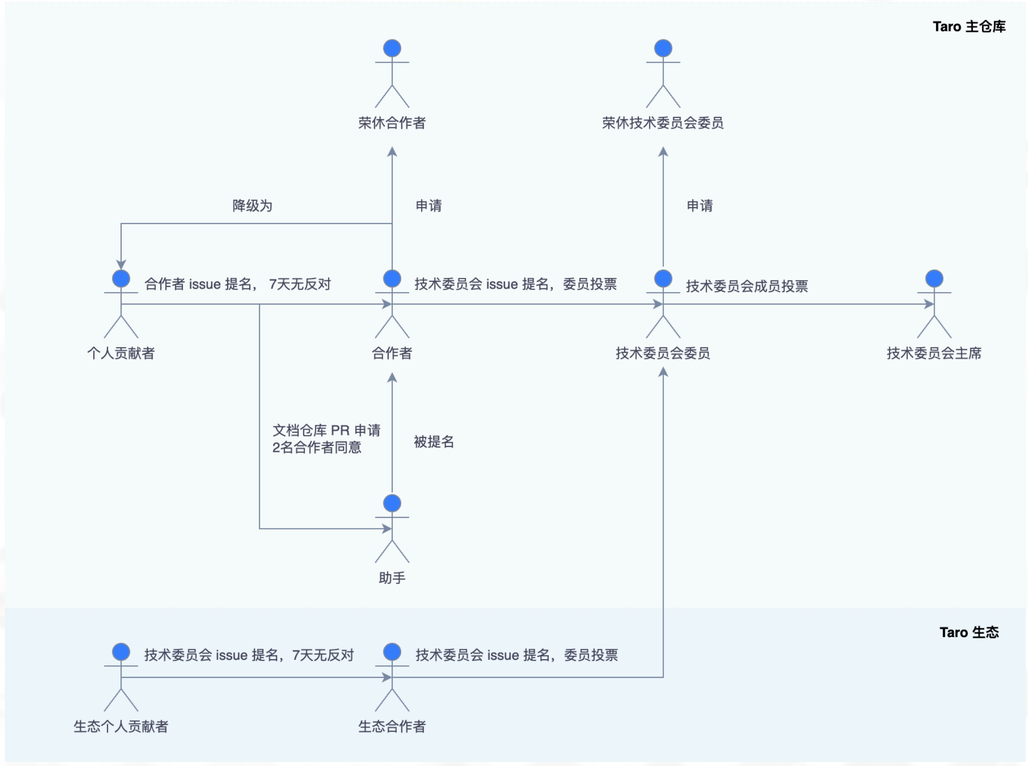
Flutter

欢迎参与 Taro 共建

社区现状及构成

社区目前由 Taro 技术委员会推动，负责决策项目整体走向并推动特性研发与落地等等，通过双周例会向关注 Taro 的贡献者与开发者们同步当前各项工作和任务进程。





THANKS

FOR YOUR WATCHING