Knobe 详解

Kbone 框架快速上手

kbone 是由腾讯发布的一款解决微信小程序和 web 端同构的框架。github 地址：https://github.com/Tencent/kbone 。

传统的开发小程序与 web 应用，因为小程序与 web 开发之间有较大区别，代码难以复用。这类同构框架，总结起来就是写一次代码，同时运行在微信小程序端和浏览器端。

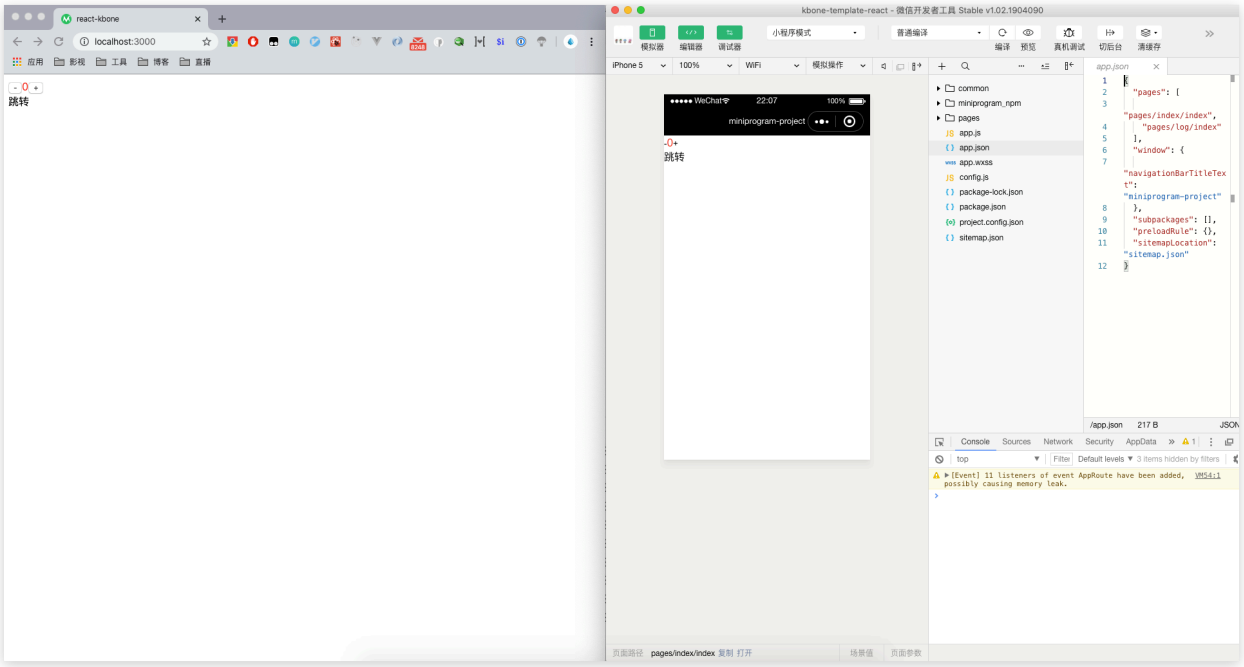
kbone 官方提供了 `kbone-cli` 以及 `vue` , `react` , `preact` , `omi` 的项目样板代码，以及官方提供了 `kbone-ui` 作为 ui 库。

```
# 全局安装 cli 工具
npm i -g kbone-cli
kbone init my-app

# 直接通过 npx 命令运行
npx kbone-cli init my-app
```

通过 cli 创建的项目，官方会默认给我们一个项目模板，通过执行不同命令，编译打包为不同环境中使用的代码：

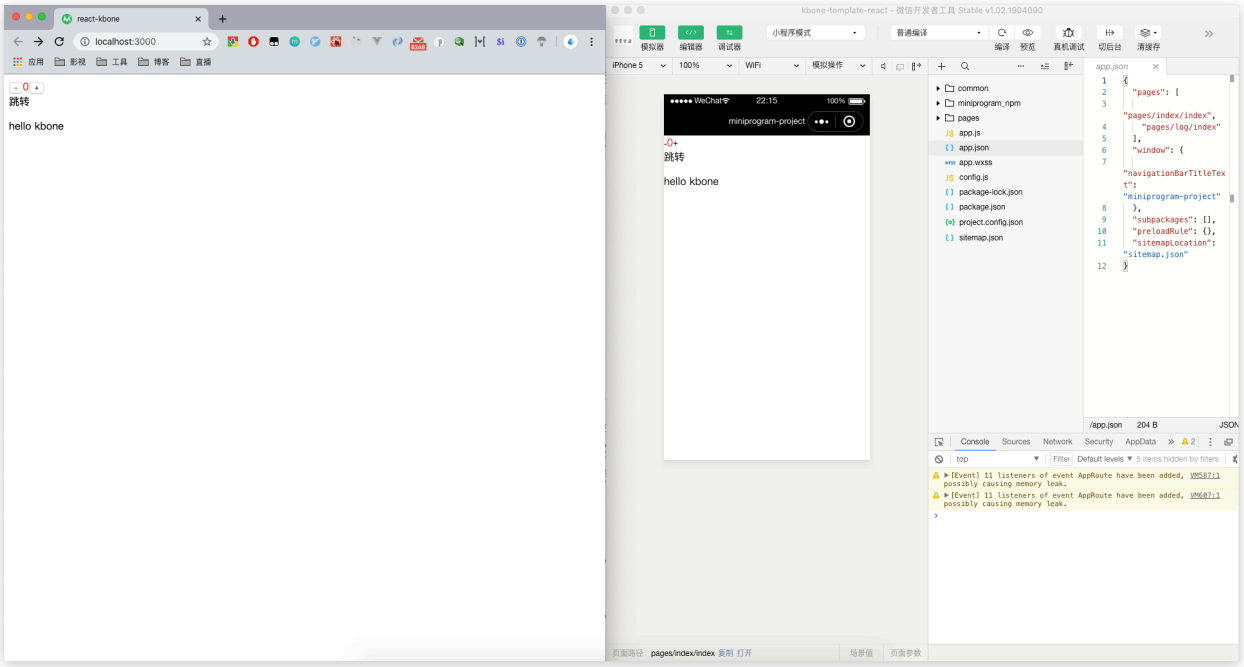
```
// 开发小程序端
npm run mp

// 开发 Web 端
npm run web
```

web 端的编译打包结果会自动启动在本地 3000 端口，小程序端的内容则需要我们手动将结果在开发者工具中打开。



我们只需要更新 src 文件夹中的源码，等待打包结果的更新，就能在两端看到更新的结果。在这个例子中我们更新一下 `Counter` 组件中的 DOM 结构，增加了一个 p 标签，可以看到两端同步更新了相同的内容。

```
function Counter() {
  const [count, setCount] = useState(0)
  return (
    <div>
      <button onClick={() => setCount(count - 1)}>-</button>
      <span>{count}</span>
      <button onClick={() => setCount(count + 1)}>+</button>
      <div onClick={clickHandle}>跳转</div>
      <p>hello kbone</p>
    </div>
  )
}
```



详解 Kbone 在实际使用中的用法

我们可以在 cli 创建项目时选择具体需要初始化的项目模板，我们可以选择 react、vue、preact、omi 等，方便不同技术栈的同学使用。

（具体代码详见视频讲解）

我们可以发现，如果我们需要使用小程序原生跳转的话，需要我们使用多个 webpack entry 进行打包，跳转时使用小程序相关 API 进行页面跳转，否则页面中还是使用客户端路由进行跳转。

剖析 Kbone 与社区其他框架差异

与 taro/remax 等同构框架的区别比较大，前两者都是一个在编译期编译完成所有页面内容，但 kbone 是在运行时渲染小程序内容。

taro 是一个多端开发框架，它能够一次编码使小程序内容最终跑在多个平台上，支持非常多的平台。

Remax 是一个可以让完整的 React 应用运行在小程序当中。Remax 通过 react-reconciler 实现了一个小程序渲染器。也就是类似于 react-dom 做的事情，把 react 应用中的 vDom 最终渲染到小程序里面。

Remax 同时 React 进行了强绑定，不支持除 React 之外的 Web 开发框架。Kbone 和 Taro 都支持 React 和类 React 以及 Vue。

Kbone

Kbone 内部实现了轻量级的 DOM 和 BOM API，把 DOM 更改的绑定到小程序的视图更改。也就是说，Kbone 并不太关心开发者使用什么框架，只要框架使用的 DOM API 被 Kbone 实现的 DOM API 覆盖到，框架就能通过 Kbone 在小程序运行。

另外 Kbone 更为专注于微信小程序开发和 H5 开发，而本节对比的其它三个小程序框架均支持多种平台的小程序开发。

深度剖析 Kbone 的实现原理

接下来我们讲解一下 kbone 的实现原理：

kbone 是一个在小程序运行时增加 DOM 相关接口从而实现的渲染。

首先 kbone 通过 lerna 来管理多个仓库的，lerna 是一个多仓库管理的工具，我们可以先简单介绍一下。我们多个项目同时发布或着互相引用时，就可以使用 lerna 来对多个仓库进行发布。

在 kbone 中，通过 lerna 将相关工具分为了 kbone-cli，miniprogram-element，miniprogram-render，mp-webpack-plugin，reduce-loader，vue-cli-plugin-kbone 和 vue-imporve-loader

kbone-cli: 命令行工具。主要在于初始化一些参数的配置相关的内容。

miniprogram-element: （重要）用于在运行时将 html 内容转化为小程序的内置组件，以及一些基础模版部分。

miniprogram-render：（重要）在小程序中模拟的一些 DOM/BOM 环境的内容，例如 window/location/history/navigator ... 的浏览器环境下的对象方法，document.getElementById 等方法，事件模型等内容。

mp-webpack-plugin：（重要）主要处理一些小程序基本骨架，reset css 等内容。

reduce-loader: 一个 webpack loader 来处理一些无用的代码。

vue-cli-plugin-kbone：vue-cli 的插件，这样就能在 vue-cli 初始化的项目中来直接使用 kbone

vue-improve-loader：用于精简 vue 项目中 DOM 树结构的 webpack loader。

cli 全称是 command line tool，简单介绍介绍一下 chalk 和 commander 的用法。

首先我们需要明白，所有的小程序可以渲染的组件，都已经提前通过 template 的形式组织到了一起，只需要我们使用正确的数据进行渲染即可渲染出来相应的组件。

miniprogram-element 中，component 代表所有微信小程序里新增的组件，通过配置将所有组件的属性和方法提前列举出来，从而最终在页面中渲染一个大的模版，我们通过改变数据就能创建出不同的 DOM 元素出来。

```
<template name="tree">
    <block wx:for="{{childNodes}}">
        <text wx:if="{{item.type == 'text'}}">{{item.value}}</text>
        <view wx:if="{{item.type == 'view'}}">{{item.value}}</view>
        <button wx:if="{{item.type == 'button'}}" size="{{item.size}}">{{item.value}}</button>
    </block>
</template>
```

`document.createElement => this.setData({childNodes: [{type: 'xx', value: 'xx'}]})`

miniprogram-render 里面有两点比较重要：

- DOM 元素的继承

DOM 元素类似于浏览器中的继承关系，一层一层实现 Element/Node 内部的方法，使得我们在访问内部元素节点或者调用方法时不会报错。

- 事件模型

  通过遍历 DOM 元素列表，模拟运行时的捕获和冒泡阶段，在对应阶段执行已绑定的事件回掉，达到与浏览器中完全相同的执行时机和执行结果。

在事件当中，eventTarget 作为最顶层的对象，会分别通过触发事件的类型，来决定最终执行的策略，例如之前我们集合到一起的 handles 对象和最终的一些通用方法比如 bindTap。

eventTarget 的 $$process 和 $$trigger 方法作为比较重要的两个方法，$$process 用作事件捕获和冒泡的流程，$$trigger 作用是触发当前 DOM 元素上注册的事件。