小程序框架对比

微信小程序 VS 支付宝小程序 VS 百度小程序 VS ...

各家小程序之间有很多相似的地方，例如相似的开发环境，相似的运行时渲染内容。

微信文档：https://developers.weixin.qq.com/miniprogram/dev/api/

百度文档：https://smartprogram.baidu.com/docs/develop/api/apilist/

支付宝文档：https://opendocs.alipay.com/mini/api

相似内容：

- 均有原生组件和 webview 中渲染的组件
- 大部分配置和项目结构（app.json/app.js）都相同，甚至之前出现了支付宝小程序文档直接拷贝微信小程序文档的情况。
- 他们都支持第三方平台，来对用户授权的小程序进行操作。

不同的内容：

- 部分 API 内容互不相同（wx/swan/my）
- 不同平台侧重的能力不同，支付宝侧重商家侧，百度小程序更侧重流量相关获取，微信小程序更侧重基础能力。
- 只有微信小程序支持个人申请使用，百度/支付宝仅支持企业使用。

百度的 API 文档中，出现了很多 AI 相关的小程序 API 来调用

小程序之间的差异具体体现在 api 的差异当中，目前各家的小程序在运行时的差异已经非常小了，底层功能都已经非常稳定。书写小程序代码的过程中只需要注意对应 API 的使用即可。

wepy VS mpVue VS remax VS taro

他们本质都是让小程序开发更加友好，或是在小程序同构中展示对应的能力。

wepy

wepy 是一个专注于小程序开发的开发框架，通过它我们能更好的组织我们的小程序代码。

特性：

- 类 Vue 开发风格
- 支持自定义组件开发
- 支持引入 NPM 包
- 支持 Promise
- 支持 ES2015+ 特性，如 Async Functions
- 支持多种编译器，Less/Sass/Stylus/PostCSS、Babel/Typescript、Pug
- 支持多种插件处理，文件压缩，图片压缩，内容替换等
- 支持 Sourcemap，ESLint 等

- 小程序细节优化，如请求列队，事件优化等

代码：

```
<style lang="less">
    @color: #4D926F;
    .num {
        color: @color;
    }
</style>
<template>
  <div class="container">
    <div class="num" @tap="num++">
      {{num}}
    </div>
    <div>{{text}}</div>
    <input v-model="text"></input>
  </div>
</template>
<config>
{
  usingComponents: {
    customCompoent: '@/components/customComponent',
    vendorComponent: 'module:vendorComponent'
  }
}
</config>
```

```
<script>
  import wepy from '@wepy/core';

  wepy.page({
    data: {
      num: 0,
      text: 'Hello World',
    },
  });
</script>
```

总结：

- 使用内部的约束来开发小程序，最终静态编译为小程序目录结构
- vue 风格，只支持微信小程序，后期如果想换框架则迁移困难

源码：

通过 bin/wepy-build.js 执行编译，通过文件名称判断不同的编译方式，不同编译方式注册的回调函数 core/plugins/parser/ 注册。

mpvue

mpvue 是一个通过 vue 进行开发，同时支持多端的框架。目前支持 微信、百度、头条和支付宝小程序。

他与 wepy 虽然同是 vue 进行开发，但是之间还是有些区别的，主要体现在它在运行时还依赖了部分 vue.runtime.js 的代码。

特点

- `mpvue` 保留了 vue.runtime 核心方法，无缝继承了 `Vue.js` 的基础能力
- `mpvue-template-compiler` 提供了将 vue 的模板语法转换到小程序的 wxml 语法的能力
- 修改了 vue 的建构配置，使之构建出符合小程序项目结构的代码格式： json/wxml/wxss/js 文件

```
<template>
    <!-- 支持 -->
    <div class="container"
:class="computedClassStr"></div>
    <div class="container" :class="{active:
isActive}"></div>

    <!-- 不支持 -->
    <div class="container"
:class="computedClassObject"></div>
</template>
```

```
<script>
    export default {
        data () {
            return {
                isActive: true
            }
        },
        computed: {
            computedClassStr () {
                return this.isActive ? 'active'
: ''
            },
            computedClassObject () {
                return { active: this.isActive
}
            }
        }
    }
</script>
```

源码

mpvue 的源码中，vue 的作者也写了很多，主要也因为这个项目也用到了很多 vue 运行时的东西，所以不同于 wepy 的编译型， mpvue 除了基本的静态编译，运行时还是加入了一些 vue.runtime.js 中的元素的，在项目迁移时会稍稍容易些。

例如在 platforms/mp/compiler/wx 中，加入了编译和解析微信小程序部分的方法。

在旁边的 runtime 中，实现了一系列的 diff/events/生命周期等等功能。

remax

remax 是最近由阿里推出的一款小程序制作框架，能同时运行在微信小程序/支付宝小程序/字节跳动小程序当中。

特性：

- 使用 react 的风格和方式来进行小程序的开发
- 使用 Remax 把代码转换到多个小程序平台。
- 完整的 TypeScript 支持。

使用

```
export default class IndexPage extends
React.Component {
  // 页面组件的 didMount 触发时机是在 onLoad 的时候
  componentDidMount() {
    console.log('IndexPage load');
  }

  onShow() {
```

```
    console.log('IndexPage show');
  }

  render() {
    return <View>Hello world!</View>;
  }
}
```

源码

相比于其他框架，remax 使用静态编译的部分就比较少，他主要应对的就是运行时的工作。

例如使用其他静态编译类型的框架，对于以下情况就会有心无力：

```
  render() {
    const child = <View>Hello remax!</View>;
    return <View>Hello world! {child} </View>;
  }

  render() {
    return <View>Hello world! <View>Hello remax!</View></View>;
  }
```

因为这里的内部 child 是通过 js 来控制的，静态编译大部分情况下不会分析到具体组件和 UI 与变量之间的联系，大部分框架只是使用一个简单的 xml 库来结构分析，所以这种变量在运行时决定 UI 就无法实现。

remax 不会有这样的问题，他在运行时实现了一套类似 react-dom ，唯一有区别的是把 react-dom 里面与 document，window 有关的内容都替换成了小程序里已有的内容，这样实现了一套 react-reconciler。

```
import ReactReconciler from 'react-reconciler';
let reconciler = ReactReconciler({
  createInstance(type) {
    return document.createElement(type);
    // 替换为小程序内部方法即可实现实例的创建
  },
  appendChild(parent, child) {
    parent.appendChild(child);
  },
  removeChild(parent, child) {
    parent.removeChild(child);
  },
  insertBefore(parent, child, before) {
    parent.insertBefore(child, before);
  }
})
```

taro

taro 目前支持微信/百度/支付宝/百度小程序和 h5，算是目前支持的平台最多，也在持续维护的一个框架平台。它也是一套遵循 React 语法规范的 **多端开发** 解决方案。通过 **Taro** 的编译工具，将源代码分别编译出可以在不同端（微信/百度/支付宝/字节跳动/QQ/京东小程序、快应用、H5、React-Native 等）运行的代码。

特性

- 支持使用 npm/yarn 安装管理第三方依赖
- 支持使用 ES7/ES8 甚至更新的 ES 规范，一切都可自行配置
- 支持使用 CSS 预编译器，例如 Sass 等
- 支持使用 Redux 进行状态管理
- 支持使用 MobX 进行状态管理
- 小程序 API 优化，异步 API Promise 化等等

使用

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

```
import { View, Button } from
"@tarojs/components";

export default class Index extends Component {
  constructor() {
    super(...arguments);
    this.state = {
      title: "首页",
      list: [1, 2, 3]
    };
  }

  add = e => {
    // dosth
  };

  render() {
    return (
      <View className="index">
        <View className="title">
{this.state.title}</View>
        <View className="content">
          {this.state.list.map(item => {
            return <View className="item">
{item}</View>;
          })}
```

```
            <Button className="add" onClick=
{this.add}>
              添加
            </Button>
          </View>
        </View>
    );
  }
}
```

源码

他是一个静态编译的框架，所以也就是说我们最终的代码都是已经编译为能直接在平台中运行的了，所以主要的工作对于 taro 来说是在编译阶段进行。

对于微信小程序的转译，我们只需要关注 taro-tarnsformer-wx 即可，在 index.ts 中，通过 ts 和 babel 将编译的结果进行分析，将分析到的 AST 结构进行处理，通过不同的 AST 结果执行不同的操作。

执行完成之后，通过 parseJSXChildren 等解析方法，将 jsx 转化为 wxml。