小程序开发生态：Taro

Taro:一套遵循 React 语法规范的多端统一开发框架

安装**cnpm**并配置淘宝镜像： **npm install -g cnpm --registry=https://registry.npm.taobao.org**

执行脚手架： **cnpm install -g @tarojs/cli** 在**node**控制台

跨端开发： **WePy(**腾讯**)**、 **mpvue(**美团**)**、**Taro(**京东**)**、**uni-app()**框架

1、创建项目： **taro init** 项目名 （在指定的目录下） **taro init Ktaro**

2、编译项目：

微信小程序**:npm run dev:weapp** 打包：**npm run build:weapp**

百度小程序：**npm run dev:swan** 打包： **npm run build:swan**

支付宝小程序：**npm run dev:alipay** 打包： **npm run build:alipay**

字节跳动小程序：**npm run dev:tt** 打包： **npm run build:tt**

H5 模式**:npm run dev:h5** 打包：**npm run build:h5**

React Native：**npm run dev:rn**

使用**ui**组件

3、安装**taro**的**UI**组件**: npm install taro-ui --save || cnpm install taro-ui --save**

npm install -g yarn ------> yarn global add @tarojs/cli

```
E:\WX\Staros>cnpm install taro-ui --save
 Installed 1 packages
 Linked 6 latest versions
 Run 0 scripts
Recently updated (since 2019-02-21): 2 packages (detail see file E:\WX\Staros\no
de_modules\.recently_updates.txt)
 All packages installed (3 packages installed from npm registry, used 1s(netwo
rk 1s), speed 37.39kB/s, json 7(37.61kB), tarball 0B)
```

配置编译源码模块(一定要先配置):

在 taro 项目的 `config/index.js` 中新增如下配置项: esnextModules: ['taro-ui'],

```
h5: {
  publicPath: '/',
  staticDirectory: 'static',
  esnextModules: ['taro-ui'],
  module: {
```

引入全局样式: 在app.js中全局引入一次即可

```
import 'taro-ui/dist/style/index.scss'
```

引用组件: 在相应的页面中引入所需要的UI

```
import { AtButton } from 'taro-ui'        // 解构的方法引入
```

使用UI组件:

```
<AtButton type='primary'>按钮文案</AtButton>
```

4、使用mobx: react数据管理就是redudx和mobx

案例见附件

1、安装依赖:

npm install --save mobx@4.8.0 @tarojs/mobx @tarojs/mobx-h5 @tarojs/mobx-rn

cnpm install --save mobx@4.8.0 @tarojs/mobx @tarojs/mobx-h5 @tarojs/mobx-rn

2、建立数据存储器

2.1、在项目**src**下建立一个**store**的目录

2.2、在**store**下建立**xxx.js**文件

引入observable让数据变成响应式的 **import { observable } from 'mobx'**

创建管理数据 **const todoStore = observable({ ..... })**

创建数据(在管理数据内操作)

```
todos: ['吃饭','睡觉','开课吧学习'],

val:'',
```

处理数据的方法

```
addTodo(item) {

    this.todos.push(item)

},
```

导出数据管理 **export default todoStore**

3、配置**mobx**在**app.js**入口引用数据管理中心

挂载数据 import { Provider } from '@tarojs/mobx'

引入数据管理的**JS** import todoStore from './store/todo'

新建**STORE**数据

```
const store = {

  todoStore
```

}

加载数据

<Provider store={store}>

<Index />

</Provider>

4、使用**mobx**修改获取数据

引入**mobx**服务　　import { observer, inject } from '@tarojs/mobx'

监听数据

@inject('todoStore')

@observer

在类里直接调用方法名使用

this.props.todoStore.方法名

渲染层使用要先解够后才能使用

const {todoStore} = this.props

todoStore.todos.