

WEEX 总结

浅尝 hybrid 开发



Author：周荣锋、崔庆超

康之元信息技术有限公司

目录

WEEX 使用同一套代码来构建 ANDROID、IOS 和 WEB 应用1

使用 Weex 需掌握	1
使用 Weex 场景建议	1
搭建开发环境.....	2
创建 weex 项目	4
开发 weex 项目	5
weex 样式	8
weex 环境变量	9
weex 跨域问题	9
weex POST 请求问题	11
weex vue-router	11

WEEX 三端开发.....15

ios 开发.....	15
Android 开发	16
源码依赖(IDE Android Studio)	18
js 文件打包及加载:.....	19
Web 开发.....	19

参考 填坑帮助	19
----------------------	-----------

WEEX

使用同一套代码来构建 ANDROID、IOS 和 WEB 应用

使用 WEEX 需掌握

Weex 目标是实现三端互通，使用一套代码，解决现阶段开发过程中需要支持三个平台的基本问题，由于牵扯三端，因此在开发中需要掌握以下语言

Android

IOS

H5/CSS/JavaScript

Vue.js

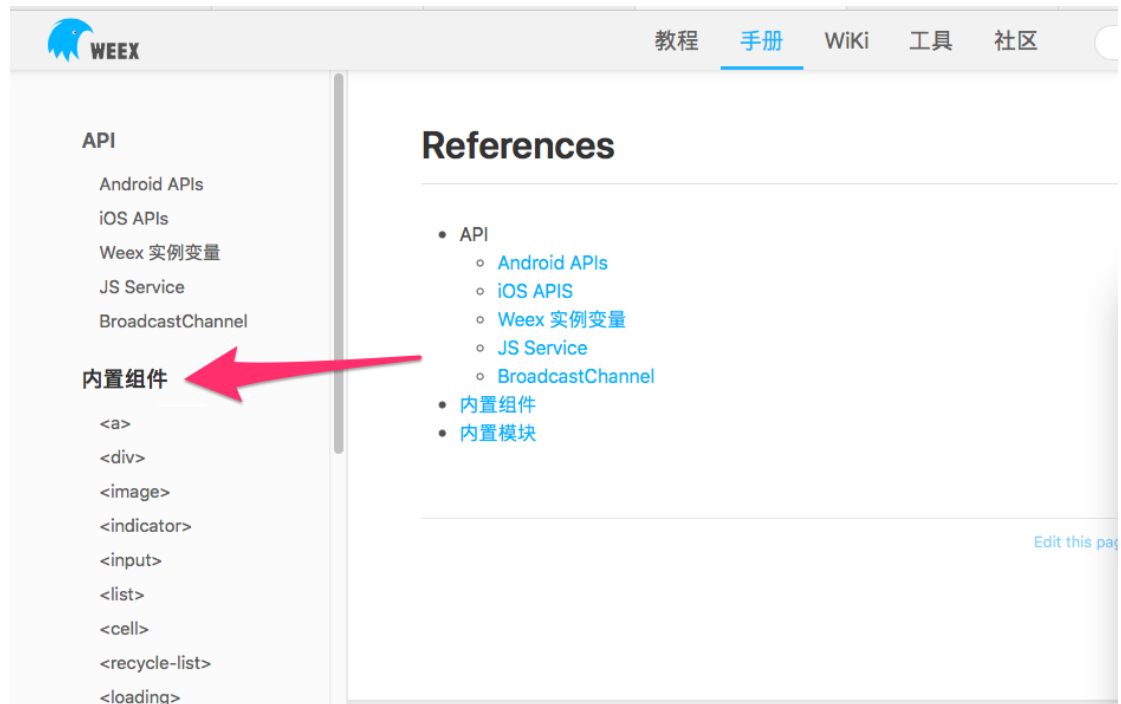
...(其它鬼)

使用 WEEX 场景建议

Weex 目标虽是三端使用，由于平台差异性以及 Weex 技术的相对新兴性，某些功能如使用 Weex 开发，几乎与三端各自开发所花精力一致。

如何快速定位哪个功能是否适合使用 Weex 开发？

方法很简单，通读 Weex 手册，如手册中内置组件不能很好实现项目需求或者没有对应组件实现需求，则需要衡量使用 Weex 的必要性。我们在尝试使用 Weex 开发的过程中，就遇到了图表、富文本这类功能，经过一些天的头痛后，最终放弃使用 Weex 实现图表功能，富文本功能也是在移动端自定义相关组件来完成



搭建开发环境

你的电脑需要安装依赖 Node.js 和 weex-toolkit

node.js

最快捷方法，直接进入官网 <https://nodejs.org/en/> 请下载推荐版，否则可能出现 node 太新，与 weex 开放不兼容问题

Download for macOS (x64)

8.12.0 LTS

Recommended For Most Users

10.11.0 Current

Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

Or have a look at the [Long Term Support \(LTS\) schedule](#).

Sign up for [Node.js Everywhere](#), the official Node.js Monthly Newsletter.

在终端，通过 `node -v` 命令查看是否安装成功，此时，`npm` 包管理工具也随之安装

```
Mac-16g:~ guanliyuan$ node -v
v8.12.0
Mac-16g:~ guanliyuan$ npm -v
6.4.1
Mac-16g:~ guanliyuan$
```

更多安装方式可参考 [Node.js 官方信息](#)

接下来，我们使用 `npm` 命令来安装 `weex-toolkit`

`npm` 是一个 JavaScript 包管理工具，它可以让开发者轻松共享和重用代码。
`Weex` 很多依赖来自社区，同样，`Weex` 也将很多工具发布到社区方便开发者使用。

安装前，确定 `npm` 版本大于 5，否则可以通过 `npm i npm@latest -g` 更新一下 `npm` 的版本

`npm install -g weex-toolkit`

安装完后，通过 `weex -v` 查看当前 `weex` 版本

`weex-toolkit` 升级子依赖方法(可忽略此步，如按步骤来一切正常)

`weex update weex-devtool@latest` //@后标注版本后，`latest` 表示最新

使用淘宝镜像安装 `weex-toolkit`

```
$ npm install -g cnpm --registry=https://registry.npm.taobao.org
$ cnpm install -g weex-toolkit
```

提示：

如果提示权限错误（*permission error*），使用 **sudo** 关键字进行安装

```
$ sudo cnpm install -g weex-toolkit
```

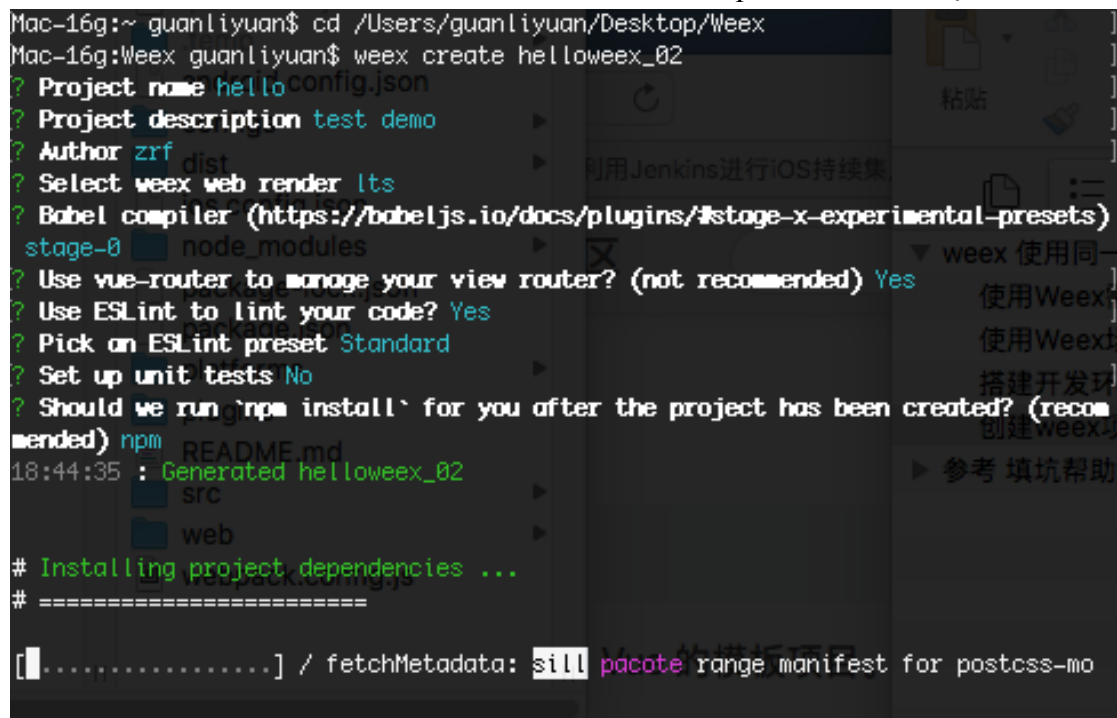
基本依赖已经安装完毕，接下来，开始一步步深入...

不要哭、

创建 WEEX 项目

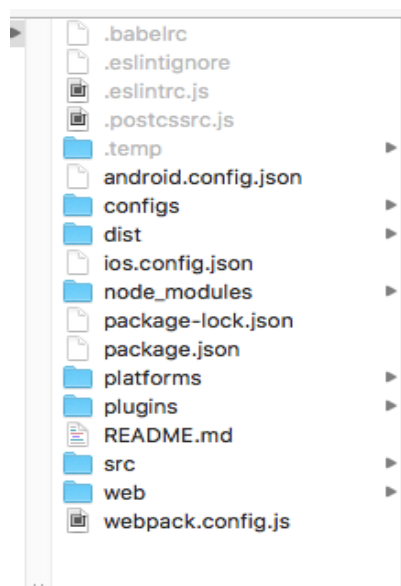
先 **cd** 到目标文件夹下

\$ **weex create helloweex** ,创建过程中，会要求填写一堆信息（如果你创建没出现，不要得意，可能你这个环境安装就有问题，webpack 版本过低？）

A terminal window showing the execution of the 'weex create helloweex_02' command. The terminal displays a series of prompts for project configuration. The user provides the following inputs: Project name: helloweex, Project description: test demo, Author: zrf, Select weex web render: its, Babel compiler: stage-0, Use vue-router: Yes, Use ESLint: Yes, ESLint preset: Standard, Set up unit tests: No, and Should we run npm install: Yes. The terminal then shows the generated file structure: README.md, src, and web. It also displays the command to install project dependencies: 'npm install'. The terminal output is as follows:

```
Mac-16g:~ guanliyuan$ cd /Users/guanliyuan/Desktop/Weex
Mac-16g:Weex guanliyuan$ weex create helloweex_02
? Project name helloweex
? Project description test demo
? Author zrf
? Select weex web render its
? Babel compiler (https://babeljs.io/docs/plugins/#stage-x-experimental-presets) stage-0
? Use vue-router to manage your view router? (not recommended) Yes
? Use ESLint to lint your code? Yes
? Pick an ESLint preset Standard
? Set up unit tests No
? Should we run `npm install` for you after the project has been created? (recommended) npm
18:44:35 : Generated helloweex_02
  README.md
  src
  web
# Installing project dependencies ...
# =====
[.....] / fetchMetadata: sill pacote range manifest for postcss-mo
```

此时在目标文件夹下生成了一个 **helloweex** 文件夹



开发 WEEX 项目

先、我们站在三端的高度看世界，可能会迷茫，到底应该在哪里开发？

我们就很开心的遇到了这样的情况，环境什么都配置好了，硬是不知道需要在哪里开，然后就对着 Weex 项目混乱地研究了几天。

请看上页项目结构截屏，对于初级菜鸟的我们，只需要关注 src 文件夹、platforms 文件夹可能根本不需要（如果你已经有了对应的 SDK 代码），dist 文件夹不需要我们去特别关注，因为它通过 src 而来的，三端中使用的就是 dist 文件夹下相关 js 文件。我们是使用 sublime 来进行 weex 项目编写

如果需要关注 platforms 文件夹状态

默认情况下 `weex create` 命令并不初始化 iOS 和 Android 项目，你可以通过执行 `weex platform add` 来添加特定平台的项目。

```
weex platform add ios    //生成 ios 项目
weex platform add android
```

由于网络环境的不同，安装过程可能需要一些时间，请耐心等待。如果安装失败，请确保自己的网络环境畅通。

不建议使用

```
weex run ios
weex run android
weex run web
```


这几个命令在模拟器上启动，因为体验下来真的很差，你如果是这几端的开发人员，相信有你已经习惯的方式去启动的

现在是不是大概清楚点？对、编写代码的时候，只要在 `src` 文件夹下去编写。在 `package.json` 中，已经配置好了几个常用的 `npm script`，分别是：

- `build`: 源码打包，生成 JS Bundle
- `dev`: webpack watch 模式，方便开发
- `serve`: 开启 HotReload 服务器，代码改动的将会实时同步到网页中

我们先通过 `npm install` 安装项目依赖。之后运行项目根目录下的 `npm run dev & npm run serve` 开启 watch 模式和静态服务器。也可以尝试使用 `npm start` 启动

当浏览器自动打开 weex h5 页面后，每次修改了代码，只要点击保存或快捷键，界面会自动更新，我们可以马上看到更新后结果

代码如下所示：

```
<template>
  <div class="wrapper" @click="update">
    <image :src="logoUrl" class="logo"></image>
    <text class="title">Hello {{target}}</text>
  </div>
</template>

<style>
.wrapper { align-items: center; margin-top: 120px; }
.title { padding-top: 40px; padding-bottom: 40px; font-size: 48px; }
.logo { width: 360px; height: 156px; }
</style>

<script>
export default {
  data: {
    logoUrl: 'http://img1.vued.vanthink.cn/vued08aa73a9ab65dcbd360ec54659ada97c.png'
    target: 'World'
  },
  methods: {
```

```
update: function (e) {  
    this.target = 'Weex'  
    console.log('target:', this.target)  
}  
}  
}  
</script>
```

按页面生成 JS

```
weex compile src dist
```

读取本地资源问题

Weex SDK 提供 `local` scheme 来访问打包在应用程序中的资源，此 scheme 无法在 H5 环境下使用。目前，开发者可以在 `image` 组件和字体文件中使用本地资源。

- 在 iOS 中，Weex 会在 `bundle resources` 中查找。例如，`image` 组件的 `src` 属性为 `local:///app_icon'`，Weex 会加载 `bundle resource` 中名为 `app_icon` 的图像资源，而字体文件也以相同的方式工作。
- 在 Android 中，`image` 组件将从 `drawable` 资源文件夹加载，如 `res/drawable-xxx`。但加载字体文件是不同的，Android 框架无法从 `res` 加载字体文件，因此 SDK 将从 `assets` 文件夹加载它。

跟着官网这句话去尝试，在 iOS 端，你会发现并没有显示出本地图片（起码我验证是这么一个结果），最终发现可能是个平台的差异性，在 iOS 上需要加上图片后缀，如 `app_icon.png`。但是 iOS 内部图片都是由 `@2x` 或者 `@3x` 后缀，需要实现本地读取图片，估计还要耗费一些精力封装兼容方法。。。

WEEX 样式

Weex 对于长度值目前只支持像素值，不支持相对单位（em、rem 等）

```
<template>
```

```
</template>
```

结构中只能包括一个块元素

每个样式只支持单一写法，不支持组合写法，如：`border: 1 solid #ff0000;`

你可以通过 `background-image` 属性创建线性渐变。

```
background-image: linear-gradient(to top,#a80077,#66ff00);
```

目前暂不支持 `radial-gradient`（径向渐变）。

Weex 目前只支持两种颜色的渐变，渐变方向如下：

- `to right`
从左向右渐变
- `to left`
从右向左渐变
- `to bottom`
从上到下渐变
- `to top`
从下到上渐变
- `to bottom right`
从左上角到右下角
- `to top left`
从右下角到左上角

表示百分比值，如“50%”，“66.7%”等。

它是 CSS 标准的一部分，但 Weex 暂不支持。Weex 默认屏幕宽度为 750px，根据输入的像素，自动 scale 到对应的大小。

假设您需要显示固定为 88 px 的导航栏，该导航栏的高度将是：

```
var height = 88 * 750 / env.deviceWidth
```

Weex 在 0.13 版本 SDK 里实现了事件冒泡机制。注意事件冒泡默认是不开启的，你需要在模板根元素上加上属性 `bubble=true` 才能开启冒泡。

```
{
  handleClick (event) {
    // 阻止继续冒泡。
    event.stopPropagation()
  }
}
```

WEEX 环境变量

Weex 提供了 `weex.config.env` 和全局的 `WXEnvironment` 变量（它们是等价的）来获取当前执行环境的信息。

```
weex.config.env === WXEnvironment
```

Weex 环境变量中的字段:

字段名	类型	描述
<code>platform</code>	String	Current running platform, could be “Android”, “iOS” or “Web”.
<code>weexVersion</code>	String	The version of Weex SDK.
<code>appName</code>	String	Mobile app name or browser name.
<code>appVersion</code>	String	The version of current app.
<code>osName</code>	String	The OS name, could be “Android” or “iOS”.
<code>osVersion</code>	String	The version of current OS.
<code>deviceModel</code>	String	Mobile phone device model. (native only)
<code>deviceWidth</code>	Number	Screen resolution width.
<code>deviceHeight</code>	Number	Screen resolution height.

WEEX 跨域问题

使用浏览器调试时，如果界面包含网络请求，会遇到

```
▲ [WDS] Disconnected!
▶ GET http://192.168.31.168:8089/#/AddStudio
▶ OPTIONS https://jkbtest.sybercare.com/services/RestServices/yundihealth/users/joinInWorkroom
▲ 已拦截跨源请求：同源策略禁止读取位于 https://jkbtest.sybercare.com/services/RestServices/yundihealth/users/joinInWorkroom 的远程资源。（原因：CORS 请求未能成功）。[详细了解]
```

接受不到网络请求结果，使用下面方法能解决此问题

打开 weex 项目，并且找到 configs 文件夹下，打开 config.js, 找到 'proxyTable' , 在里面添加如下代码

```
proxyTable: {  
  '/api': { //使用"/api"来代替"https://jkbtest.sybercare.com/"  
    target: 'https://jkbtest.sybercare.com/', //源地址  
    changeOrigin: true, //改变源  
    pathRewrite: {  
      '^/api': '' //路径重写  
    }  
  }  
},
```

target：你需要请求的服务器地址，替换成对应地址即可.增加完后，如果无效，建议重启服务试试

跨域处理前网络请求写法

```
stream.fetch({ //https://jkbtest.sybercare.com/  
  method: 'POST',  
  type: 'json',  
  url: 'https://jkbtest.sybercare.com/' + 'services/RestServices/yundihealth/users/  
    joinInWorkroom',  
  body: JSON.stringify({  
    comcode : '000100010001',  
    phone : '18200000001',  
  })),  
  headers: {'content-type': 'application/json'}  
// url: 'http://10.242.69.181:8089/yanxuan/' + api  
, res => {  
  console.log('====结果'+JSON.stringify(res))  
  if (res.ok) { //成功  
    succ(res.data)  
  } else {  
    fail(res)  
  }  
}, progress =>{  
  console.log('====进度'+JSON.stringify(progress))  
})
```

跨域处理后写法，主要观察 url

```
console.log('开始请求')  
stream.fetch({ //https://jkbtest.sybercare.com/  
  method: 'POST',  
  type: 'json',  
  url: '/api' + 'services/RestServices/yundihealth/users/joinInWorkroom',  
  body: JSON.stringify({  
    comcode : '000100010001',  
    phone : '18200000001',  
  })),  
  headers: {'content-type': 'application/json'}  
// url: 'http://10.242.69.181:8089/yanxuan/' + api  
, res => {  
  console.log('====结果'+JSON.stringify(res))  
  if (res.ok) { //成功  
    succ(res.data)  
  } else {  
    fail(res)  
  }  
}, progress =>{  
  console.log('====进度'+JSON.stringify(progress))  
})
```

WEEX POST 请求问题

如上页截屏为 post 请求正确写法，在上面写法中，需要注意两点

传参，需要将 object 对象转换为 string 对象

```
body:JSON.stringify({  
  comcode : '000100010001',  
  phone : '18200000001',  
}),
```

header, post 需要配置 content-type 为'application/json'

如未按上述方法处理，将出现请求 415 错误

HTTP 请求 415 错误 - 不支持的媒体类型(Unsupported media type)

WEEX VUE-ROUTER

Weex 版本

```
Last login: Mon Oct 15 09:21:00 on console  
Mac-16g:~ guanliyuan$ weex -v  
v1.3.11  
- weexpack : v1.2.1  
- weex-builder : v0.4.1  
- weex-previewer : v1.5.1  
Mac-16g:~ guanliyuan$
```

Src 路径结构

```
▼ src  
  ▼ components  
    <> HelloWorld.vue  
  ▼ index  
    /* entry.js  
    <> index.vue  
    /* router.js
```

entry.js 内容

某些全局方法与全局类可以定义在此处, 如: import mixins from './mixins',在此 js 文件中可以处理公共方法: 网络请求、时间格式等. 在其他 Vue 页面中可以直接调用

```
/* weex initialized here, please do not move this line */
const router = require('./router') //导入路由文件
const App = require('@index.vue') //导入根组件
/* eslint-disable no-new */
new Vue(Vue.util.extend({el: '#root', router}, App)) // '#root'根组件id 等同 App.el = '#root'
router.push('/') //指定一个路由入口, 初始化显示的页面内容
```

index.vue 简介

index.vue 为项目主入口, 不可单独当一个页面处理, 类似 IOS 开发中的 AppDelegate。如下网易严选 weex 项目 index.vue 截屏。Router-view 用于控制多页面问题, tab-bar 为主架构结构, 因此放在 Index.vue 中

```

<template>
  <div class="app-wrapper">
    <router-view class="r-box"></router-view>
    <tab-bar @tabTo="onTabTo"></tab-bar>
  </div>
</template>
<style>
  body{
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
    color:#333;
  }
</style>
<style scoped>
  .app-wrapper{
    background-color: #f4f4f4;
  }
  .r-box{
    position: absolute;
    top:0;
    left: 0;
    right: 0;
    bottom: 0;
  }
</style>

<script>
  var modal = weex.requireModule('modal');
  import util from './assets/util';
  import tabBar from './assets/components/tabBar.vue';
  export default {
    data () {
      return {
      }
    },
    components: {
      'tab-bar': tabBar
    },
    created () {
      util.initIconFont();
    },
    methods: {
      onTabTo(_result){
        let _key = _result.data.key || '';
        this.$router && this.$router.push('/'+_key)
      }
    }
  }
</script>

```


Router-view 多页面配置

所有的页面在 router.js 中进行配置,初次显示页面可以在 router 中使用 redirect 来重定向 entry.js 中最后一句 router.push('/')

```
router.js
1  /**
2   * Created by zwill on 2017/8/29.
3   */
4  import Router from 'vue-router'
5  import ViewHome from './assets/views/home.vue'
6  import ViewTopic from './assets/views/topic.vue'
7  import ViewClass from './assets/views/class.vue'
8  import ViewShop from './assets/views/shop.vue'
9  import ViewMy from './assets/views/my.vue'
10
11  Vue.use(Router)
12
13
14  export default new Router({
15    // mode: 'abstract',
16    routes: [
17      { path: '/', redirect: '/home' },
18      { path: '/home', component: ViewHome },
19      { path: '/topic', component: ViewTopic },
20      { path: '/class', component: ViewClass },
21      { path: '/shop', component: ViewShop },
22      { path: '/my', component: ViewMy }
23    ]
24  })
```

也可以直接修改 entry.js 中 router.push 方法

如：router.push('/studio') //指定一个路由入口，初始化显示的页面内容

```
router.js
1  /* global Vue */
2  import Router from 'vue-router'
3  // import HelloWorld from '@components/HelloWorld'
4  import Studio from '@page/Studio'
5  import Report from '@page/Report'
6
7  Vue.use(Router)
8
9  module.exports = new Router({
10    routes: [
11      {
12        path: '/studio',
13        name: 'Studio',
14        component: Studio
15      },
16      {
17        path: '/report',
18        name: 'Report',
19        component: Report
20      }
21    ]
22  })
```

WEEX

三端开发

IOS 开发

依赖第三方框架，请确保工程中包括 SDWebImage/WeexSdk 框架

必不可少的 WXImgLoaderDefaultImpl 文件

除非你能立字据，说你整个项目都不会有网络图片下载，否则在工程里少不了它的影子

Weex 与 IOS 设备尺寸差异性

由于 Weex 长宽是一个固定值，为了适配不同机型，会产生一个属性用于记录像素缩放比

```
/**
 * scale factor from css unit to device pixel.
 */
@property (nonatomic, assign, readonly) CGFloat pixelScaleFactor;
```

如下两种情形传输高度情况

IOS 计算的长宽传输到 Weex 展示

```
WXPixelType pix = _label.frame.size.height;
if ([WXInteractiveDataManager shareManager].fPixScale > 0.001) {
```

```

        pix = pix / [WXInteractiveDataManager shareManager].fPixScale /
2; //将坐标转化为 weex 坐标 由于 weex 中富文本标签做了乘以 2 的 y 处理，此次除以 2，使真实结果
不变
    }

```

fPixScale 为 pixelScaleFactor 化身

Weex 长宽传输到 IOS 展示使用 WXConvert

```

pix = [WXConvert WXPixelType:weexHeight scaleFactor:[WXInteractiveDataManager
shareManager].fPixScale];

```

ANDROID 开发

Android 集成有两种方式

1. 源码依赖：能够快速使用 WEEX 最新功能，可以根据自己项目的特性进行相关改进。
2. SDK 依赖：WEEX 会在 jcenter 定期发布稳定版本。[jcenter](#)
注:国内可能需要翻墙

前期准备

- 已经安装了 [JDK](#) version>=1.7 并配置了环境变量
- 已经安装 [Android SDK](#) 并配置环境变量。
- Android SDK version 23 (compileSdkVersion in [build.gradle](#))
- SDK build tools version 23.0.1 (buildToolsVersion in [build.gradle](#))
- Android Support Repository >= 17 (for Android Support Library)

快速接入

如果你是尝鲜或者对稳定性要求比较高可以使用依赖 SDK 的方式。

步骤如下：

1. 创建 Android 工程，没有什么要特别说明的，按照你的习惯来。
2. 修改 build.gradle 加入如下基础依赖

```

compile 'com.android.support:recyclerview-v7:23.1.1'
compile 'com.android.support:support-v4:23.1.1'

```

```
compile 'com.android.support:appcompat-v7:23.1.1'
compile 'com.alibaba:fastjson:1.1.46.android'
compile 'com.taobao.android:weex_sdk:0.18.0'
```

3. 注:版本可以高不可以低。

代码实现

注:附录中有完整代码地址

- 实现图片下载接口, 初始化时设置。

```
/**
 * Created by lixinke on 16/6/1.
 */
public class ImageAdapter implements IWXImgLoaderAdapter {

    @Override
    public void setImage(String url, ImageView view, WXImageQuality quality, WXImageStr:
        //实现你自己的图片下载, 否则图片无法显示。
    }
}
```

- 初始化

```
/**
 * 注意要在 Manifest 中设置 android:name=".WXApplication"
 * 要实现 ImageAdapter 否则图片不能下载
 * gradle 中一定要添加一些依赖, 否则初始化会失败。
 * compile 'com.android.support:recyclerview-v7:23.1.1'
 * compile 'com.android.support:support-v4:23.1.1'
 * compile 'com.android.support:appcompat-v7:23.1.1'
 * compile 'com.alibaba:fastjson:1.1.45'
 */
public class WXApplication extends Application {

    @Override
    public void onCreate() {
        super.onCreate();
        InitConfig config=new InitConfig.Builder().setImgAdapter(new ImageAdapter()).build
        WXSDKEngine.initialize(this,config);
    }
}
```

```

    try {
        WXSDKEngine.registerModule("poneInfo", PhoneInfoModule.class); //注册 Module
        WXSDKEngine.registerModule("netInfo", WeexNetModule.class);
        WXSDKEngine.registerComponent("rich", RichText.class, false); //注册 Component
    } catch (WXException e) {
        e.printStackTrace();
    }
}
}
}

```

- 开始渲染

```

public class MainActivity extends AppCompatActivity implements IWXRenderListener {

    WXSDKInstance mWXSDKInstance;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mWXSDKInstance = new WXSDKInstance(this);
        mWXSDKInstance.registerRenderListener(this);
        /**
         * WXSAMPLE 可以替换成自定义的字符串，针对埋点有效。
         * template 是 we transform 后的 js 文件。
         * option 可以为空，或者通过 option 传入 js 需要的参数。例如 bundle js 的地址等。
         * jsonInitData 可以为空。
         * width 为 -1 默认全屏，可以自己定制。
         * height 为 -1 默认全屏，可以自己定制。
         */
        mWXSDKInstance.render("WXSample", WXFileUtils.loadFileContent("hello.js", this), null);
    }
}

```

源码依赖(IDE Android Studio)

这种方式不利于 weex sdk 的更新维护 不推荐使用。

JS 文件打包及加载:

将编写好的 index.vue 文件编译成 index.is:

```
weex compile index.vue dist
```

然后将 js 文件拷贝的 android/ios 项目 android 的放在 assets 目录下，使用
WXFileUtils.loadAsset("index.js" , this)获取文件路径

WEB 开发

Weex build web/npm run build:prod:web/npm run pack:web 生成 release 文件夹，
里面 html 文件为 web 端使用

参考

填坑帮助

weex 官网：<http://weex.apache.org/cn/guide/index.html>

weex IDE: <https://www.jianshu.com/p/f2fca2719f23>

weex 例子: <https://weex.apache.org/cn/examples.html>

weex web 打包启示: <https://www.cnblogs.com/crazycode2/p/7989932.html>

weex UI 组件: <https://alibaba.github.io/weex-ui/#/?id=weex-ui>

weex vue-router: <https://www.jianshu.com/p/2a2ae8a4118f>

vue mixin: https://blog.csdn.net/china_bomber_20/article/details/79034365
请求 415: <http://www.cnblogs.com/cocoajin/p/3986204.html>