通达信weex开发指引

|  |  |  |  |
| --- | --- | --- | --- |
| Author | 更新内容 | 日期 | 备注 |
| Tdxypf | 完整初稿，基本环境搭建、开发、说明 | 2018.06.11 |  |
| Tdxypf | 支持本地、远程js加载，远程js模式支持开启jsdebug断点调试 | 2018.06.15 |  |
| Tdxypf | 自定义edidt组件支持密码框类型 | 2018.06.20 |  |
| Tdxypf | 增加api调用章节，1、weex消息订阅 | 2018.06.21 |  |
| Tdxypf | 增加weex-ui安装、使用，增加connect-weex.js独立版 | 2018.07.03 |  |
| Tdxypf | 增加webpack热更新编译模式 | 2018.07.04 |  |

# Weex 简介

Weex 致力于使开发者能基于当代先进的 Web 开发技术，使用同一套代码来构建 Android、iOS 和 Web 应用。具体来讲，在集成了 WeexSDK 之后，你可以使用 JavaScript 和现代流行的前端框架来开发移动应用。

Weex 的结构是解耦的，渲染引擎与语法层是分开的，也不依赖任何特定的前端框架，目前主要支持 [Vue.js](https://vuejs.org/) 和 [Rax](https://alibaba.github.io/rax/" \t "_blank) 这两个前端框架。

Weex 的另一个主要目标是跟进当代先进的 Web 开发和原生开发的技术，使生产力和性能共存。在开发 Weex 页面就像开发普通网页一样；在渲染 Weex 页面时和渲染原生页面一样。

一次编写，处处运行

Weex 可以只编写一份代码，开发出三端都可用的页面。

在多个端中使用相同的源代码可以显著提高开发效率，并简化测试，构建和发布流程。在此基础上，Weex 可以将前端的打包、测试流程与手机端监控、发布系统结合起来，提高开发效率。

尽管 Weex 多端都是用的同一份代码，但是仍然支持针对特定的平台开发功能。Weex 提供了 weex.config.env 和 WXEnvironment（它们是相同的）来获得当前的运行时环境。你可以用 WXEnvironment.platform 来确定代码运行在哪个平台上。除了平台以外，WXEnvironment 还包含其他环境信息，如 *osVersion* 和 *deviceModel*，参考 *[Weex variable](http://weex.apache.org/cn/references/weex-variable.html)* 了解更多详细信息。

# 搭建开发环境

## 第一步：安装依赖

Weex 官方提供了 weex-toolkit 的脚手架工具来辅助开发和调试。首先，你需要 Node.js 和 [Weex CLi](https://github.com/weexteam/weex-toolkit" \t "_blank)。

安装 Node.js 方式多种多样，最简单的方式是在 [Node.js 官网](https://nodejs.org/en/) 下载可执行程序直接安装即可。

对于 Mac，可以使用 [Homebrew](http://brew.sh/) 进行安装：

brew install node

安装完成后，可以使用以下命令检测是否安装成功：

$ node -v  
v6.11.3  
$ npm -v  
3.10.10

通常，安装了 Node.js 环境，npm 包管理工具也随之安装了。因此，直接使用 npm 来安装 weex-toolkit。

**注意:**在weex-toolkit1.0.8版本后添加了npm5规范的npm-shrinkwrap.json用于锁定包依赖，故npm版本<5的用户需要通过npm i npm@latest -g更新一下npm的版本，使用前请确认版本是否正确。

安装前建议你的node版本是 >= 8.0，同时建议 npm 版本 >= 5

|  |
| --- |
| $ npm install -g weex-toolkit $ weex -v //查看当前weex版本 |

weex-toolkit也支持直接升级子依赖，如：

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

国内开发者可以考虑使用淘宝的 npm 镜像 —— [cnpm](https://npm.taobao.org/" \t "_blank) 安装 weex-toolkit

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

## 第二步：初始化

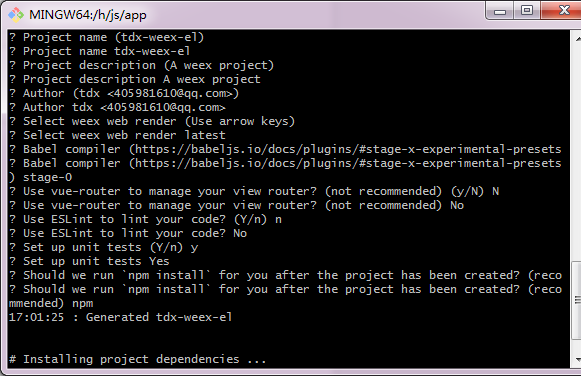
然后初始化 Weex 项目：

|  |
| --- |
| $ weex create tdx-weex-project |

执行完命令后，在 tdx-weex-project 目录中就创建了一个使用 Weex 和 Vue 的模板项目。

下面是具体初始化过程的示例图：输入如下命令进入初始化过程

$ weex create tdx-weex-el



确认工程名称、描述、作者...信息。

因weex暂时使用不到vue-rounter，故在过程中选择NO,

Eslint 语法检查插件，可选可以不选

## 第三步：开发

之后我们进入项目所在路径，weex-toolkit 已经为我们生成了标准项目结构。

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。之后运行根目录下的 npm run dev & npm run serve开启 watch 模式和静态服务器。

然后我们打开浏览器，进入 http://localhost:8080/index.html 即可看到 weex h5 页面。

初始化时已经为我们创建了基本的示例，我们可以在 src/index.vue 中查看。

## 第四步：引入实际开发工程

#### Js底层依赖

connect-weex.js

请从tfs代码服务器webapp tlibs-es6目录下载并引入业务开发中。

#### 交互示例

已经包含在基础工程，webcall 、服务、行情、交易calltql均已打通。示例代码

Webcall示例：

tdxCt.tdxgetAccList(function(data) {

let res = typeof data === 'object' ? data : JSON.parse(data);

let resobj = typeof res[0] === 'object' ? res[0] : JSON.parse(res[0])

});

交易示例：

let value = e.tdxEdit;

let entry = '2110';

let stream = [{

"F110": \_this.userInfo.WTFS || '7',

"F402": value

}];

let pramObj = {};

tdxCt.JYCallTql(entry, stream, pramObj, function(data) {

let res = tdxCt.FormatResult(data);

})

服务、行情示例：

let entry = 'CWServ.mzx\_yw';

let stream = {

"CallName": "mzx\_yw",

"Params": ["103", "", "1", "20"]

};

let pramObj = {};

tdxCt.HQCallTql(entry, stream, pramObj, function(data) {

console.log(data)

})

# 加载weex js方法

## UIconfig 配置

<Item ID="Stock.kfsjj.jjrg" Image="" Title="基金认购" Type="Weex" RunTag="tdx-trade-weex/jjrg.weex.js" RunParam="PageID:Mobile.Stock.kfsjj.jjrg;initData:fuc=jjrg,qsid=39;" Tool="Back,,TradeInfo,,WeexScan" PreIf="#IFPtJyLogin#"/>

WeexScan 为启用调试模式扫一扫入口

## openurl打开

tdxCt.WEBCallTql("tdxOpenUrl", {

"OpenName": "weex",

"OpenType": "native",

"OpenUrl": "tdx-weex/index.js",

"OpenParam": {

// "UrlType": "Absolute"

"LoadByWeex":true

},

function(data) {

}

})

注意：链接支持本地、远程链接js,调试模式请使用远程链接js

# 调试工具devtool：

## 介绍

伴随着weex的正式开源，对一款针对weex框架的简单易用的调试工具的呼声也日趋强烈。weex devtools就是为weex前端和native开发工程师服务的一款调试工具，可同时检查weex里DOM属性和Javascript 代码断点调试，支持IOS和Android两个平台。

Chrome devtools对于前端开发者来说最熟悉不过，有广泛的用户基础.weex devtools实现了[Chrome Debugging Protocol](https://developer.chrome.com/devtools/docs/debugger-protocol" \t "_blank)，其使用体验和普通的web开发一致，对于前端开发者是零学习成本，其主要功能分为两大部分——Debugger和Inspector,第一个版本已经随weex0.6.1 发布, 手淘也已接入。

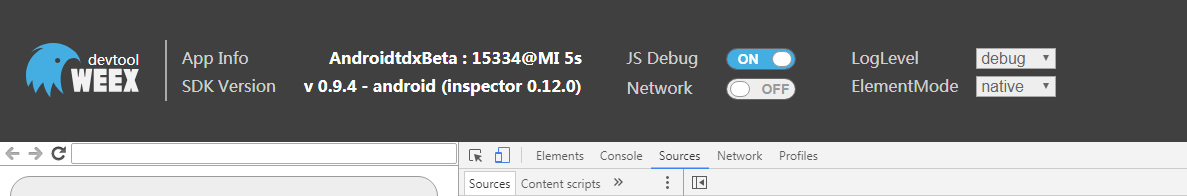
以下是Devtools的使用介绍:

## 开启调试

执行下面命令：

$weex debug

单纯启动一个调试服务器,并同时唤起chrome浏览器打开调试主页. 这个调试主页上会有一个二维码,使用 通达信weexdebug扫这个二维码可以开启Playground调试. 开启调试后,进入相关weex页面即可进入调试模式。



## webpack热更新模式

工程根目录configs目录config.js,修改const ip=本机ip,及端口dev.port

Npm run start 启动服务

App配置对应的链接`http://ip:port/dist/`+js路径，即可热更新编译调试。

例如：http://192.168.0.235:8084/dist/pages/putong/stock/zhcx.weex.js

# 自定义组件

自定义edit标签，基础工程已有示例,具体使用方案参考工程实例；

< tdxEditTextV2 tdxHint=”请输入” tdxPasswordType=’yes’ @ tdxEditViewV2\_TextChanged=’editOnchange’ ></ tdxEditTextV2>

说明：

1 tdxHint

设置默认提示的文字

2 tdxPasswordType

yes表示设置为密码模式 不设置或者设置为其它表示非密码模式

3 tdxEditViewV2\_TextChanged 输入回调事件

tdxEdit:当前输入的内容（密码模式下为加密后的内容）

tdxEditLen:当前输入内容的长度

tdxEditPasswordKey:密码模式下 会传递的一个Key对应输入的密码。发送请求的时候如果要填入输入的密码，则输入密码对应的Key，底层会自动将Key转换为对应的原始密码

# API调用

weex页面订阅消息，Webcall示例

tdxCt.WEBCallTql(tdxRegNotificationWeex ,{

NotificationName:"abc" //订阅消息类型

OperCallBack:"testcallback" // 订阅回调函数

InstanceId:"" //当前页面实例ID

})

交易示例：

let value = e.tdxEdit;

let entry = '2110';

let stream = [{

"F110": \_this.userInfo.WTFS || '7',

"F402": value

}];

let pramObj = {};

tdxCt.JYCallTql(entry, stream, pramObj, function(data) {

let res = tdxCt.FormatResult(data);

})

服务、行情示例：

let entry = 'CWServ.mzx\_yw';

let stream = {

"CallName": "mzx\_yw",

"Params": ["103", "", "1", "20"]

};

let pramObj = {};

tdxCt.HQCallTql(entry, stream, pramObj, function(data) {

console.log(data)

})

# Weex-UI的安装与使用

进入到项目录，执行下面命令，安装

npm i weex-ui –S

使用示例：

<template>

<wxc-button text="确定"

@wxcButtonClicked="wxcButtonClicked"></wxc-button>

</template>

<script>

import { WxcButton } from 'weex-ui' //import 引入

export default {

components: { WxcButton }, //注册组件

methods: {

wxcButtonClicked (e) {

console.log(e)

}

}

}

</script>

# 相关官方资源

教程：

<http://weex.apache.org/cn/guide/set-up-env.html>

在线编辑：

<http://dotwe.org/vue/>

组件、元素、样式示例：

<https://hanks10100.github.io/weex-vue-examples/>

weex-ui:

<https://alibaba.github.io/weex-ui/#/cn/>

vue 教程：

<https://cn.vuejs.org/v2/guide/>

npmjs:

<https://www.npmjs.com/>

在线app案例：



# 参考信息

weex默认使用750px \* 1334px作为适配尺寸, 实际渲染时由于浮点数的误差可能会存在几px的误差, 出现细线等样式问题, 可以通过加减几个px来调试。