

---

# Table of Contents

|             |       |
|-------------|-------|
| 前言          | 1.1   |
| 介绍          | 1.1.1 |
| 需要看的框架文档    | 1.1.2 |
| 安装与调试       | 1.2   |
| 开发环境依赖模块说明  | 1.3   |
| 项目目录说明      | 1.4   |
| 发布运行程序      | 1.5   |
| 构建工具配置说明    | 1.6   |
| App技术框架说明   | 1.7   |
| App配置以及其他方法 | 1.8   |
| 组件说明        | 1.9   |
| App业务逻辑说明   | 1.10  |
| App打包及接口调用  | 1.11  |

# Notebook

Just a brief introduction

这是一个app(android/iOS)项目，但页面视图全部都用的是html5页，没有使用app的原生页面，项目可以直接在PC上运行html5页面。与服务端的交互全部都是走web api接口方式，接口统一配置在common.js的app.api下。

前端h5是基于mui + vue+gulp+html5+css3的前端App多页面项目框架。

app打包技术是用到了eclipse工具打包成APP，本项目使用了原生设备的拍照、定位功能和原生插件（信鸽推送）。对于app的升级是html5资源整个APP更新。这些都是dcloud提供一整套技术解决方案。

本项目需要开发者了解基本的html5、css3、js、vue、gulp的知识。

说明：开发工具用到dcloud的Hbuilder IDE，eclipse IDE，打包工程文件有固定的文件目录一般不需要增改，打包流程[点击这个](#)。

1. 前端UI的部分使用mui框架
2. app打包技术使用eclipse工具
3. 原生App对设备的调用参考dcloud的html5+文档
4. 使用tap点击事件跳转到目标页面，该页面是mui自带的窗口创建打开，非h5原生的a链接跳转
5. 使用.vue文件进行页面功能组件化的开发
6. 使用ajax管理数据加载
7. 使用信鸽实现消息推送
8. 使用gulp实现对模块打包、压缩。
9. pug模板语言、stylus css预处理器

说明：开发中需要了解这些框架语法，至少了解它的一些规则

1. mui框架[文档地址](#)
2. vue框架[文档地址](#)
3. dcloud的html5+文档[文档地址](#)
4. eclipse本地打包app流程[文档地址](#)

## 安装

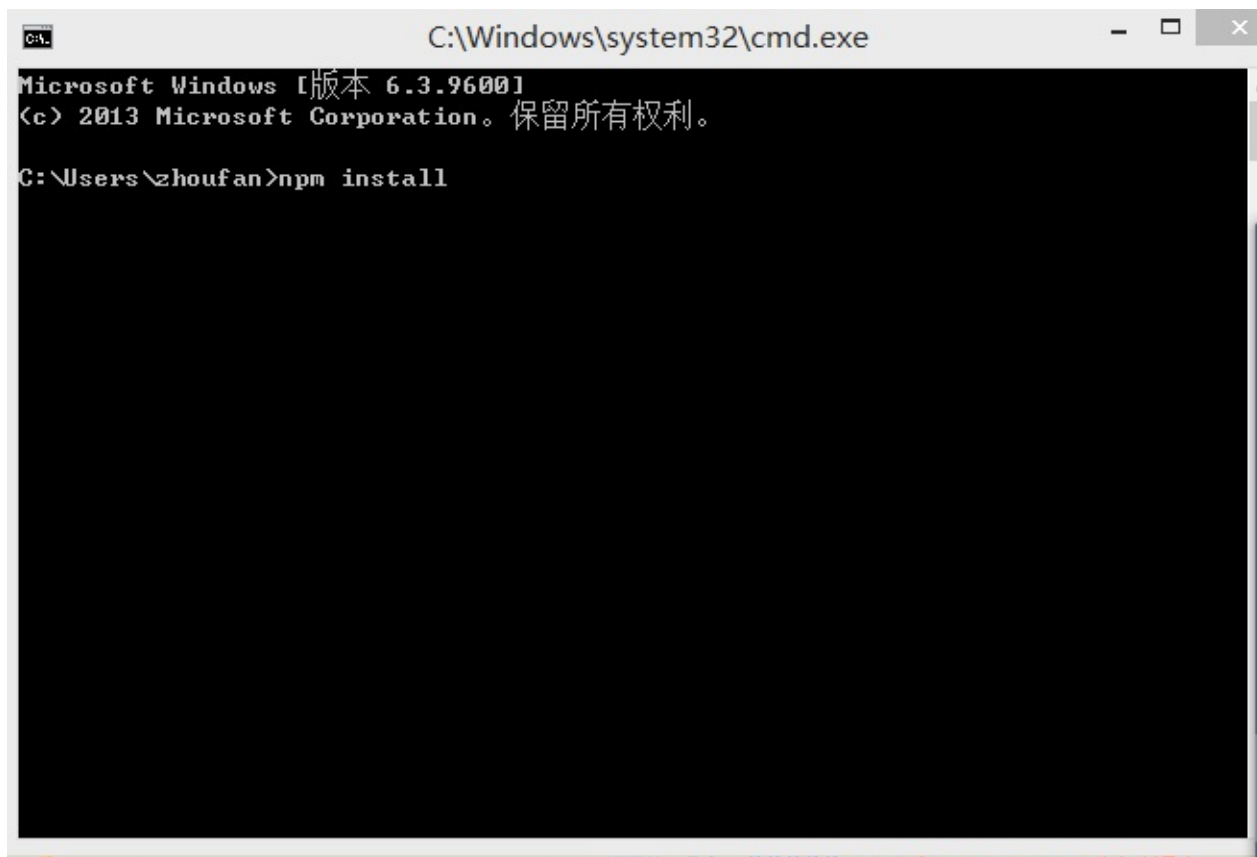
- 下载[HBuilder IDE](#)开发工具，其实HBuilder是dcloud 把eclipse的改造成一个专门应用于app打包、多种语言支持：php、jsp、ruby、python、nodejs等web语言，less、coffee等编译型语言均支持的开发工具
- 下载[node.js](#)，作为前端web的运行环境。我当前的node.js版本是6.10.0 npm版本是5.4.0
- app打包完全是基于manifest.json配置文件，它主要是用来配置app的基本信息（版本号、appid等）、图标(app的应用图标)、sdk配置、模块权限配置、页面引用关系、代码视图，具体参看dcloud提供的[文档](#)。

## npm初始化

\$1. 下载项目在当前项目目录下打开命令行工具输入

```
npm install
```

下载所有依赖，下载速度取决网速和硬件，下载完在执行后面的流程

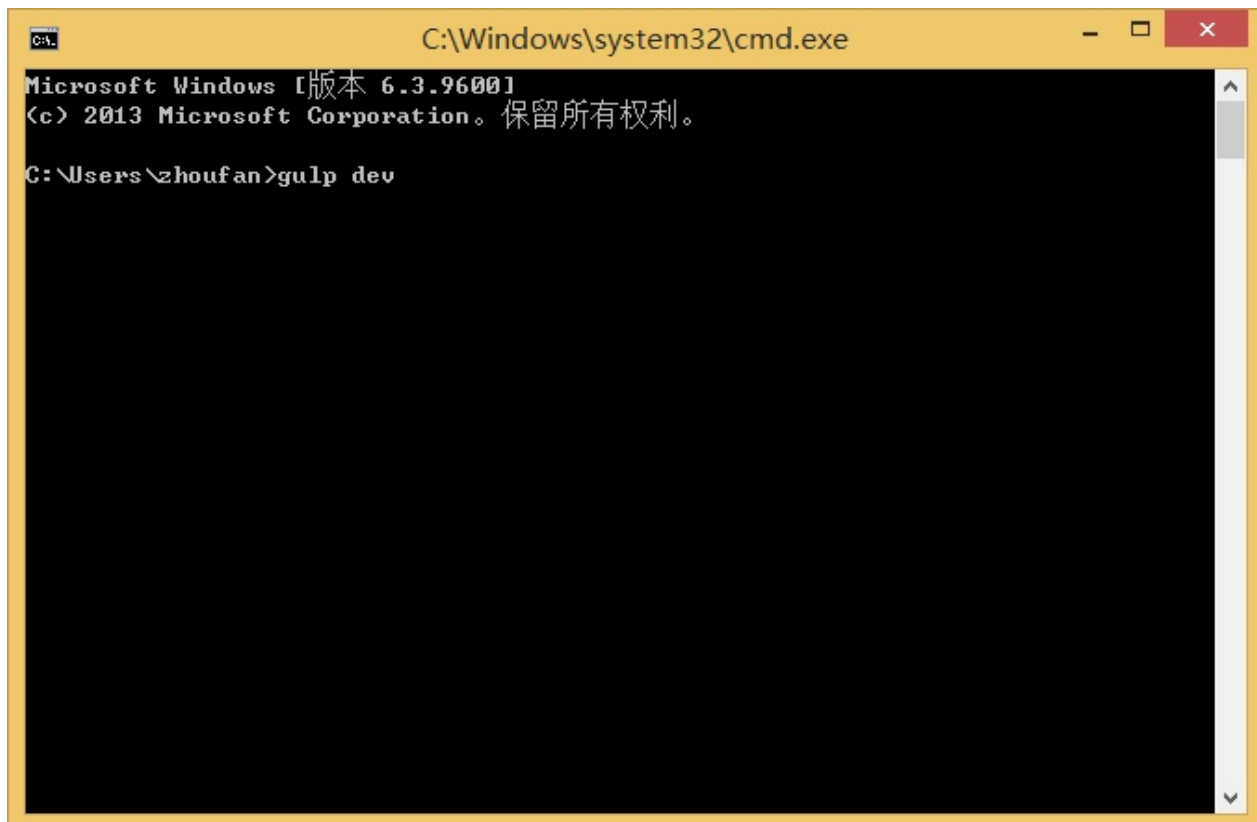


A screenshot of a Windows command prompt window. The title bar reads "C:\Windows\system32\cmd.exe". The window content shows the following text: "Microsoft Windows [版本 6.3.9600]" followed by "(c) 2013 Microsoft Corporation。保留所有权利。". The prompt "C:\Users\zhoufan>" is followed by the command "npm install".

```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation。保留所有权利。

C:\Users\zhoufan>npm install
```

\$2. 执行编译输入  
gulp dev



A screenshot of a Windows command prompt window. The title bar reads "C:\Windows\system32\cmd.exe". The window content shows the following text: "Microsoft Windows [版本 6.3.9600]" followed by "(c) 2013 Microsoft Corporation。保留所有权利。". The prompt "C:\Users\zhoufan>" is followed by the command "gulp dev".

```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation。保留所有权利。

C:\Users\zhoufan>gulp dev
```

**package.json**内容如下

```
{
  "devDependencies": {
    "autoprefixer": "^7.1.1",
    "babel-core": "^6.25.0",
    "babel-loader": "^7.1.1",
    "babel-plugin-syntax-dynamic-import": "^6.18.0",
    "babel-preset-env": "^1.6.0",
    "css-loader": "^0.28.4",
    "gulp": "^3.9.1",
    "gulp-changed": "^2.0.0",
    "gulp-clean-css": "^3.0.3",
    "gulp-htmlmin": "^3.0.0",
    "gulp-load-plugins": "^1.5.0",
    "gulp-postcss": "^6.3.0",
    "gulp-pug": "^3.3.0",
    "gulp-stylus": "^2.6.0",
    "gulp-uglify": "^2.0.1",
    "minimist": "^1.2.0",
    "postcss-loader": "^2.0.6",
    "pug-loader": "^2.3.0",
    "stream-combiner2": "^1.1.1",
    "stylus-loader": "^3.0.1",
    "vinyl-named": "^1.1.0",
    "vue-loader": "^12.2.1",
    "vue-template-compiler": "^2.3.4",
    "webpack": "^2.6.1",
    "webpack-stream": "^3.2.0"
  },
  "browserslist": [
    "Android >= 4.4.3",
    "iOS >= 8"
  ],
  "dependencies": {
    "axios": "^0.16.2",
    "vue": "^2.3.4"
  }
}
```

## 调试

- chrome浏览器pc端调试和hbuilder真机调试，chrome调试比较方便(性能及查看代码显示比较好)，以下只以chrome浏览器讲解

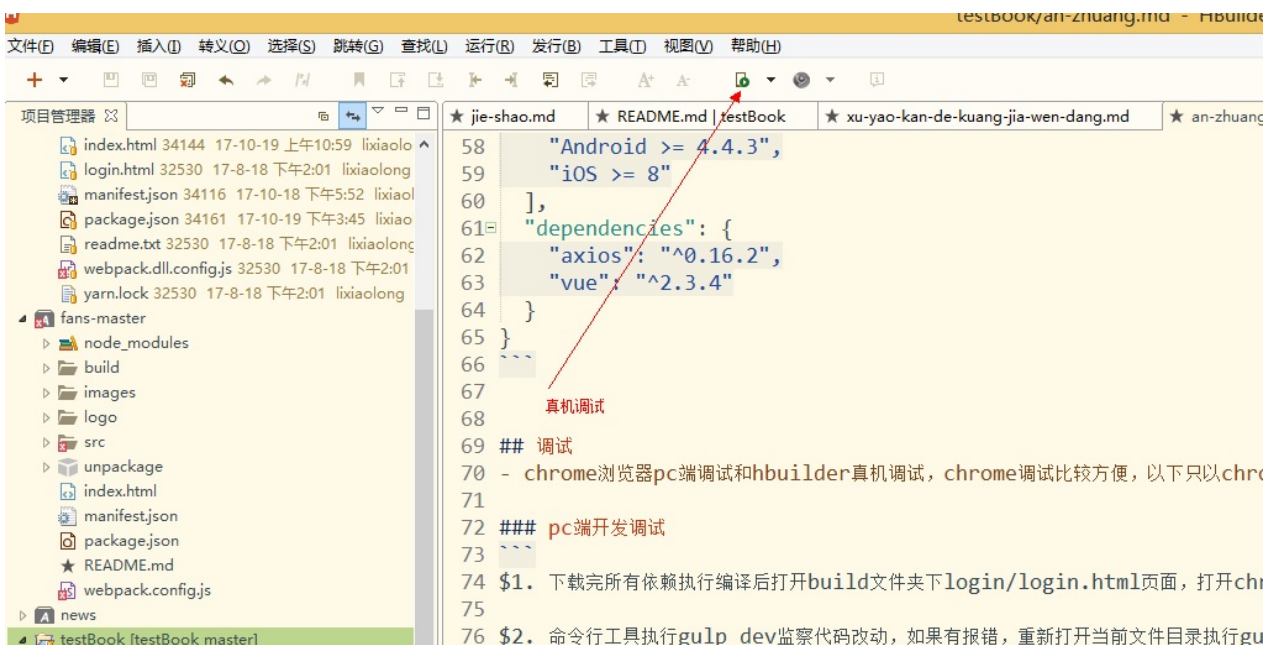
## pc端开发调试

\$1. 下载完所有依赖执行编译后打开build文件夹下login/login.html页面，打开chrome F12按键可以打开开发工具查看html的dom元素和样式以及js代码

\$2. 命令行工具执行gulp dev监察代码改动，如果有报错，重新打开当前文件目录执行gulp dev，修改代码后，刷新页面就可以查看改动

## 移动端开发调试

\$1. 用手机通过usb线连接电脑，打开hbuilder IDE的真机调试，可以查看数据输出和代码修改结果





## 开发环境依赖模块说明

|                                |                                                                                                        |
|--------------------------------|--------------------------------------------------------------------------------------------------------|
| vue                            | //构建用户界面的                                                                                              |
| vue-resource                   | //vue 的http ajax请求插件（本项目没有用它，暂时保留）                                                                     |
| vue-html-loader                | //vue html加载器                                                                                          |
| vue-loader                     | //vue加载器                                                                                               |
| vue-style-loader               | //vue的样式加载器                                                                                            |
| vue-template-compiler          | //vue的模板编译器                                                                                            |
| autoprefixer                   | //css 浏览器兼容性问题处理                                                                                       |
| babel-core                     | //ES6 代码转换器                                                                                            |
| babel-eslint                   | //ES6的代码检查                                                                                             |
| babel-loader                   | //ES6 代码转换器，webpack插件                                                                                  |
| babel-plugin-transform-runtime | //和polyfill类似，替换助手函数                                                                                   |
| babel-preset-es2015            | //ES6 代码编译成现在浏览器支持的ES5                                                                                 |
| babel-preset-stage-2           | //ES6 ES7要使用的语法阶段                                                                                      |
| babel-register                 | //用于改写require命令, 为它加上一个钩子<br>。此后, 每当使用require加载.js、.jsx、.es和.es6后缀名的文件, 就会先用Babel进行转码。                 |
| babel-polyfill                 | //Babel默认只转换新的JavaScript句法（syntax），而不转换新的API，babel-polyfill就是为当前环境提供一个垫片。<br>解决一些浏览器不能识别的语法，比如：Promise |
| css-loader                     | //css 生成                                                                                               |
| file-loader                    | //webpack的文件加载器，主要用于字体<br>将字体文件打包                                                                      |
| html-loader                    | //webpack的html加载器，主要用于html<br>文件的加载                                                                    |
| pug-loader                     | //pug模板语言加载器，主要用于.pug文件<br>的加载                                                                         |
| stylus-loader                  | //css预处理加载器，主要用于.styl文件的<br>加载                                                                         |
| html-webpack-plugin            | //html 文件编译                                                                                            |
| style-loader                   | //webpack的style加载器，主要用于css<br>插入到style标签                                                               |
| url-loader                     | //webpack的url加载器，主要用于图片加<br>载及限制                                                                       |
| webpack                        | //用来构建打包程序                                                                                             |
| webpack-dev-server             | //开发环境下，设置代理服务器                                                                                        |
| webpack-require-http           | //webapck打包环境下的require加载htt<br>p文件的插件                                                                  |



## 项目目录说明

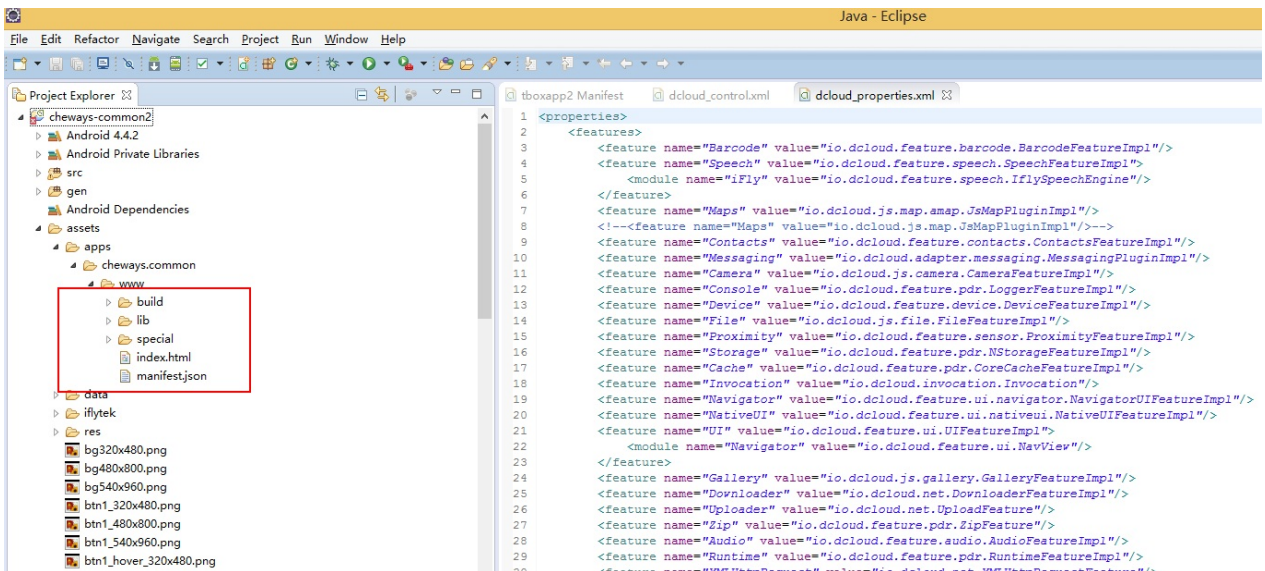
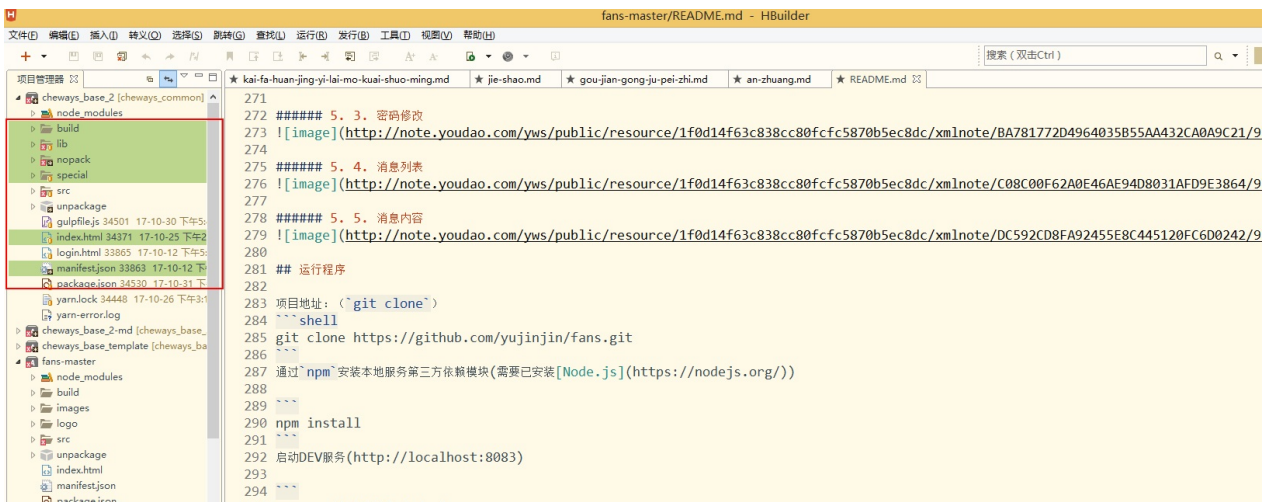
```
|-- node_modules          // 依赖文件目录
|-- build                 // gulp打包后的文件目录
|-- lib                  // 存放js插件资源文件
|   |-- fonts            // 存放各种fonts文件目录
|   |-- mui.js           // mui插件
|   |-- ...              // 其他第三方JS插件
|-- src                  // 源码目录
|   |-- common            // 存放公共属性和方法的目录
|       |-- common.js    // app一些共用方法文件
|       |-- common.css   // 公共样式文件
|   |-- components       // 存放公共组件的目录
|       |-- app-header.vue // 页面头部导航栏公共组件
|       |-- ...          // 其他公共组件
|   |-- pages            // 页面视图文件目录
|       |-- .css         // 对应单页样式文件
|       |-- .html        // 对应页面html文件
|       |-- .js          // 对应页面js文件
|       |-- ...          // 其他静态资源（图片等）
|   |-- imgs             // 存放各种图片文件目录
|   |-- nopack           // 存放公共js库
|   |-- special          // 存放公共参数配置、js库
|       |-- ....        // 其他工具JS文件
|       |-- background.png // 页面背景图片
|       |-- ...          // 其他广告图片和样式文件
|-- unpackaged           // app编译包目录
|-- index.html           // app的首页加载文件
|-- manifest.json        // 打包app的配置文件
|-- package.json         // 配置项目相关信息，通过执
行 npm init 命令创建
```

# 运行程序

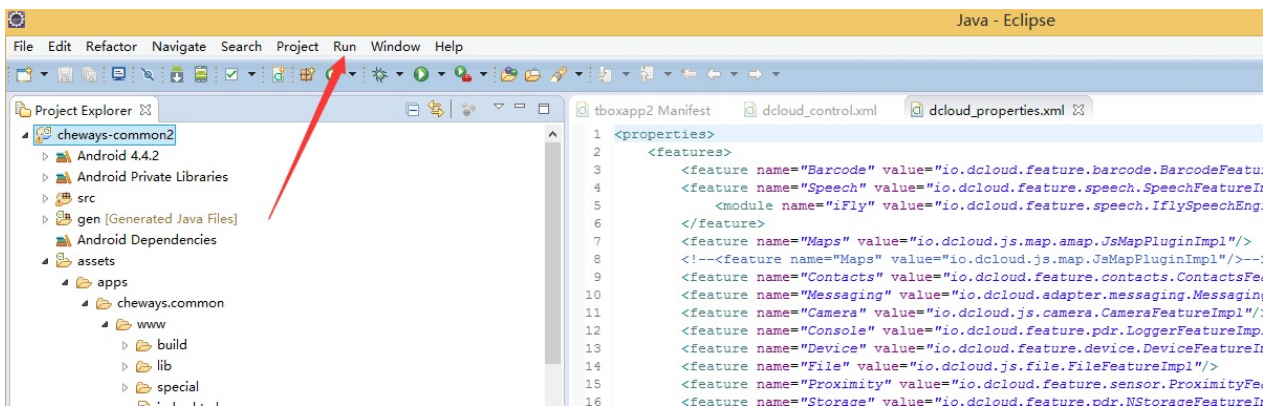
打包发布build代码

gulp build

1.编译打包后将编译后的文件替换eclipse工程文件assets/www/下的文件



2.执行运行下的打包命令：Run as » Android Application



3.可以连接电脑在手机上运行，也可以下载apk包运行查看

## gulpfile.js 配置说明

gulp工具配置，自动执行打包压缩代码等命令

```
var autoprefixer = require('autoprefixer'),
    combiner = require('stream-combiner2'), //整合 streams 来处理
    错误
    gulp = require('gulp'),
    named = require('vinyl-named'),
    path = require('path'),
    plugins = require('gulp-load-plugins')(),
    webpack = require('webpack'),
    webpackStream = require('webpack-stream');

var dest = 'build', proj = '.';

gulp.task('build', function() { //build任务, 执行styles、scriptsRelease、htmls、resources对应样式，js文件，html文件，静态资源图片等打包任务
    gulp.start('styles', 'scriptsRelease', 'htmls', 'resources');
});

gulp.task('dev', ['watch'], function() { //监察任务，代码改动执行资源文件打包
    gulp.start('styles', 'scripts', 'htmls', 'resources');
});

gulp.task('xiaodou', function() { //其他业务任务
    proj = '../cheways_xiaodou';
    dest = '../cheways_xiaodou/build';
    gulp.start('copy', 'styles', 'scripts', 'htmls', 'resources');
});

gulp.task('copy', function(){
    gulp.src('common/**/*.*)
        .pipe(plugins.changed(proj+'/common'))
        .pipe(gulp.dest(proj+'/common'));
    gulp.src('lib/**/*.*)
        .pipe(plugins.changed(proj+'/lib'))
```

```
.pipe(gulp.dest(proj+'/lib'));
gulp.src('index.html')
  .pipe(plugings.changed(proj))
  .pipe(gulp.dest(proj));
});

gulp.task('watch',function(){
  gulp.watch('src/**/*.@(css|styl)', ['styles']);
  gulp.watch('src/**/*.@(html|pug)', ['htmls']);
  gulp.watch('src/**/*.@(png|jpg|ico|svg|gif)', ['resources']);
  gulp.watch(proj+'/special/*.styl', ['styles']);
});

gulp.task('styles',function(){//styles命令处理css样式文件
  gulp.src('src/**/*.css')
    .pipe(plugings.changed(dest))
    .pipe(plugings.postcss([ autoprefixer({cascade: false}) ]
  ))
    .pipe(plugings.cleanCss())
    .pipe(gulp.dest(dest));
  gulp.src('src/**/*.styl')
    .pipe(plugings.changed(dest,{extension:'.css'}))
    .pipe(plugings.stylus({paths: [proj+'/special']}))
    .pipe(plugings.postcss([ autoprefixer({cascade: false}) ]
  ))
    .pipe(gulp.dest(dest));
  gulp.src('common/common.styl')
    .pipe(plugings.changed(proj+'/common',{extension:'.css'}))
  )
    .pipe(plugings.stylus({paths: [proj+'/special']}))
    .pipe(gulp.dest(proj+'/common'));
});

gulp.task('scriptsRelease',function(){//scriptsRelease命令处理js文件压缩
  gulp.src('src/**/*.js')
    .pipe(named(function(file) {
      return file.relative.slice(0, -path.extname(file.path).length)
    })))
```



```
.pipe(webpackStream({
  output: {
    publicPath: '../',
    chunkFilename: '[name].bundle.js',
  },
  resolve: {
    alias: {
      'vue$': 'vue/dist/vue.common.js',
      'cheways-vue$': '../../nopack/js/cheways-vue
.js',
      'special': path.resolve(proj, 'special'),
    }
  },
  module: {
    rules: [
      {test: /\.vue$/, loader: 'vue-loader', optio
ns: {
      postcss: [require('autoprefixer')()]
    }},
      {test: /\.js$/, exclude: /node_modules/, loa
der: "babel-loader" },
    ]
  },
  plugins: [
    new webpack.DllReferencePlugin({
      context: __dirname,
      manifest: require('../lib/vendors-manifest.js
on')
    })
  ],
}, webpack))
.pipe(plugins.uglify())
.pipe(gulp.dest(dest));
});

gulp.task('scripts',function(){
  gulp.src('src/**/*.js')
    .pipe(named(function(file) {
      return file.relative.slice(0, -path.extname(file.pat
h).length)
```

```
    )))
    .pipe(webpackStream({
      output: {
        publicPath: '../',
        chunkFilename: '[name].bundle.js',
      },
      resolve: {
        alias: {
          'vue$': 'vue/dist/vue.common.js',
          'cheways-vue$': '../../nopack/js/cheways-vue
.js',
          'special': path.resolve(proj, 'special'),
        }
      },
      module: {
        rules: [
          {test: /\.vue$/, loader: 'vue-loader', optio
ns: {
          postcss: [require('autoprefixer')()]
        }},
          {test: /\.js$/, exclude: /node_modules/, loa
der: "babel-loader" },
        ]
      },
      plugins: [
        new webpack.DllReferencePlugin({
          context: __dirname,
          manifest: require('./lib/vendors-manifest.js
on')
        })
      ],
      watch: true,
    }, webpack))
    .pipe(gulp.dest(dest));
  });

gulp.task('htmls', function(){//htmls命令处理html文件压缩等
  gulp.src('src/**/*.html')
    .pipe(plugings.changed(dest))
    .pipe(plugings.htmlmin())
```

```
        .pipe(gulp.dest(dest));
    var combined = combiner.obj([
        gulp.src('src/**/*.pug'),
        plugins.changed(dest, {extension: '.html'}),
        plugins.pug({}),
        gulp.dest(dest)
    ]);
    combined.on('error', console.error.bind(console));
});

gulp.task('resources', function() { //resources命令处理资源图片
    gulp.src('src/**/*.@(png|jpg|ico|svg|gif)')
        .pipe(plugins.changed(dest))
        .pipe(gulp.dest(dest));
});

gulp.task('dist', function() {
    var androidProject = 'C:/Users/XL/workspace/cheways-common3/
assets/apps/HelloH5/www/'
    gulp.src('build/**/*.*)
        .pipe(plugins.changed(androidProject+'build'))
        .pipe(gulp.dest(androidProject+'build'))
    gulp.src('lib/**/*.*)
        .pipe(plugins.changed(androidProject+'lib'))
        .pipe(gulp.dest(androidProject+'lib'))
    gulp.src('special/**/*.*)
        .pipe(plugins.changed(androidProject+'special'))
        .pipe(gulp.dest(androidProject+'special'))
    gulp.src('manifest.json')
        .pipe(plugins.changed(androidProject+'www'))
        .pipe(gulp.dest(androidProject+'www'))
})

gulp.task('distas', function() {
    var androidProject = 'C:/Users/XL/AndroidStudioProjects/chew
ays-common3/app/src/main/assets/apps/HelloH5/www'
    gulp.src('build/**/*.*)
        .pipe(plugins.changed(androidProject+'build'))
        .pipe(gulp.dest(androidProject+'build'))
    gulp.src('lib/**/*.*)
```

```
        .pipe(plugins.changed(androidProject+'lib'))
        .pipe(gulp.dest(androidProject+'lib'))
    gulp.src('special/**/*.*')
        .pipe(plugins.changed(androidProject+'special'))
        .pipe(gulp.dest(androidProject+'special'))
    gulp.src('manifest.json')
        .pipe(plugins.changed(androidProject+'www'))
        .pipe(gulp.dest(androidProject+'www'))
    })
```

# App技术框架说明

## 1. 首页加载页面（**index.html**）

app的入口加载文件，也是app的初始化。主要做app UI的初始化、VUE的初始化、app的常用JS加载、业务自动登录等

```

window.vm = new Vue({//vue实例
  el: '#app',      //vue 挂载对象
  mounted: function() {//生命周期回调函数，当前为dom加载完全后执行属性和方法
    var vm = this;
    mui.init({//mui 初始化
      preloadPages: [pageConfig['browser_main']] //预加载内
置浏览器
    });
    mui.plusReady(function() {//mui plus对象初始化
      plus.runtime.getProperty(plus.runtime.appid, function(
n(wgtinfo) {
        localStorage.version = wgtinfo.version;
      });
      //无网络提示
      document.addEventListener('netchange', function() {
        var nt = plus.networkinfo.getCurrentType();
        switch(nt) {
          case plus.networkinfo.CONNECTION_ETHERNET:
          case plus.networkinfo.CONNECTION_WIFI:
          case plus.networkinfo.CONNECTION_CELL2G:
          case plus.networkinfo.CONNECTION_CELL3G:
          case plus.networkinfo.CONNECTION_CELL4G:
            break;
          default:
            //防止多个出现警告
            if (!localStorage.alerting) {
              localStorage.alerting = 'true';
              mui.alert(vm.$t("index.noInter[0]"),
vm.$t("index.noInter[1]"), vm.$t("index.noInter[2]"), function(
) {
                localStorage.removeItem('alertin
g');
              });
            }
            break;
          }
        }, false);
      });
    });
  });
}

```

## 2. 单页面（src目录下例：login.html）

```

window.vm = new Vue({//vue实例
  el: '#app2',//vue 挂载对象
  data: {//数据变量
    lang: navigator.language.slice(0, 2),
    showLogin: false,
    userName: '',
    password: '',
    type: 3, //3表示普通用户，2表示管理员
  },
  mounted: function() {//生命周期回调函数，当前为dom加载完全后执行属性和方法
    var _self = this;
    window.addEventListener('pagebeforeshow', this.pageBeforeShow, false);

    mui.init({});
    mui.later(function() {
      _self.showLogin = true;
    }, 2000);
    mui.plusReady(function() {
      //手动关闭启动页
      plus.navigator.closeSplashscreen();
      plus.navigator.setStatusBarBackground('#fff');
      plus.navigator.setStatusBarStyle('dark');
    });
    //一秒内连续点击两次，退出应用，仅安卓有效；
    mui.back = this._back;
    window.addEventListener('fleetLogin', function(event) {
      _self.fleetLogin(event.detail);
    });
    window.addEventListener('lastAutoLogin', function(event)
    {
      var ll = JSON.parse(localStorage.lastLogin)
      _self.autoLogin({mobile: ll.name, password: ll.pwd})
    }
    ;

    });
    window.addEventListener('autoLogin', function(event) {
      _self.autoLogin(event.detail);
    });
  }
});

```

```
    });  
  },  
  computed: { //计算属性  
    canLogin: function() {  
      return this.userName.trim().length > 0 && (this.password.trim().length >= 6 || (this.type==2 && this.password.trim().length >= 1));  
    },  
  },  
  methods: { //业务方法  
    autoLogin: function(d) {  
      this.userName = d.mobile;  
      this.password = d.password;  
      this.login();  
    },  
    pageBeforeShow: function() {  
      this.userName = JSON.parse(localStorage.lastLogin||'{}').name||'';  
      localStorage.removeItem('userData');  
    },  
    login: function() {  
      var _self = this,  
          userName = _self.userName.trim(),  
          password = _self.password.trim(),  
          cid = "";  
  
      if(!userName || !password) {  
        mui.toast(this.$t("login.notEmpty"));  
        return;  
      }  
      //若是手机号，传上次的设备号给后端（用于后端确定token）  
      if (app.Reg.IS_MOBILE.test(userName)) {  
        cid = localStorage["bind:" + userName]||'';  
      }  
      mui('#pw')[0].blur();  
      var loginData = {  
        name: _self.userName,  
        pwd: _self.password,  
        type: _self.type,  
        cid:cid,
```



```

        packageName: localStorage.appId,
    };
    app.ajax({
        url: app.api['login2'],
        data: loginData,
        showWaiting: this.$t("login.waiting"),
        closeWaiting: false,
        success: function(data, message) {
            app.closeWaiting();
            if (data.prompt == 1) { //设备号已激活但手机号
未注册过
                mui.confirm(message, localStorage.appName, [_self.$t("login.goRegister[0]"), _self.$t("login.goRegister[1]")], function(e) {
                    if (e.index == 0) {
                        mui.openWindow({url: "../register/register.html?mobile="+data.clink||''});
                    } else {
                        continueLogin();
                    }
                })
            } else {
                continueLogin();
            }

            function continueLogin() {
                _self.logins = data;
                localStorage.lastLogin = JSON.stringify(loginData);
                _self.password = '';

                if(!data.device) { //管理员
                    app.closeWaiting();
                    localStorage.adminToken = data['access_token'];
                    vm.openWindow(pageConfig['device-list']);
                    return;
                }
            }
        }
    });

```

```
        if (data.contents == '4') {
            localStorage.activateToken = data['access_token'];

            app.closeWaiting();
            mui.alert(_self.$t("login.goActivate[0]"), '', _self.$t("login.goActivate[1]"), function(e) {
                if (e.index == 0)
                    vm.openWindow({url: '../activate/activate.html', styles: {top: 0}});
            });
        } else {
            _self.userLogin(data);
        }
    },
    error: function() {
        app.closeWaiting();
    }
});

},
fleetLogin: function(item) {
    localStorage.setItem('token', localStorage.adminToken);
    localStorage.setItem('user', new User(item.cid, "admin", item.clink));
    app.ajax({
        url: app.api.decideGetPush,
        data: {cid: item.cid},
        success: function() {}
    });
    mui.openWindow(pageConfig['main']);
},
findPassword: function() {
    vm.openWindow(pageConfig['find-password']);
},
switchToggle: function(event) {
    this.type = event.detail.isActive ? 2 : 3;
},
userLogin: function(data) {
```

```
//刷新手机号绑定的最新登录设备，用于手机号登录
localStorage.setItem("bind:" + data.ctel, data.device[0]);

localStorage.token = data['access_token'];
var u = new User(data.device[0], 'normal', data.click);

localStorage.user = JSON.stringify(u);
mui.openWindow(pageConfig['main']);
},
_back: function() {
    var self = this;
    if(!self.firstTime) {
        self.firstTime = new Date().getTime();
        plus.nativeUI.toast(this.$t("login.doubleExit"));
    };

    setTimeout(function() {
        self.firstTime = null;
    }, 1000);
} else {
    if(new Date().getTime() - self.firstTime < 1000)
    {
        var main = plus.android.runtimeMainActivity();

        main.moveTaskToBack(false);
    }
}
});
```

### 3. app打包配置（manifest.json）

app打包配置的应用名称，appid，横竖屏等

注：manifest里大部分配置在真机运行时是不生效的,生效的部分以蓝色字体表示,其他部分需要通过App打包才可看到效果

基本信息

应用名称:

车卫士

appid:

HelloH5

云端获取

版本号:

1.2.1

页面入口:

index.html

选择...

应用描述:

车载定位监控系统

☐

应用是否全屏显示

应用全屏配置说明

☒

debug模式 (android是否支持日志输出及chrome调试)

应用资源是否解压

☒不解压直接运行

☐解压资源后运行

根据重力感应自动横竖屏



☒ portrait



☐ upside down



☒ landscape left



☐ landscape right

## app配置以及其他方法（src/common.js）

### 一、app公用ajax方法

根据业务逻辑定义了公共ajax方法，token注入、请求体处理，以及返回消息处理

```

    * ajax方法
    * @param {Object} setting
    */
    app.ajax = function(setting) {
        var defaultSetting = {
            dataType: setting.dataType || 'json', //服务器返回json
            type: (setting.url.lastIndexOf('.json') > -1 ? 'get'
            : (setting.type || 'post')),
            timeout: setting.timeout || 15000, //超时时间设置为15秒
            async: true,
            showWaiting: setting.showWaiting || false,
            success: function(resp, status, xhr) {
                app.log(/\n/([\\w\\.]+)$/.exec(setting.url)[1] + '
                返回数据:' + JSON.stringify(resp));
                switch(resp.flag) {
                    case '2000':
                        mui.isFunction(setting.success) && setti
                        ng.success(resp.result, resp.message);
                        break;
                    case '777': //token自动续期
                        app.ajax({
                            url: app.api['login'],
                            data: JSON.parse(localStorage.lastLo
                            gin),
                            success: function(data) {
                                if(data.contents == '3') { //未激
                                活
                                    localStorage.activateToken =
                                    data['access_token'];
                                    app.closeWaiting();
                                }
                            }
                        });
                    }
                }
            }
        };
        app.ajax(defaultSetting);
    };

```

```
        mui.confirm(lang.activate[0]
, '', [lang.activate[1], lang.activate[2]], function(e) {
            if(e.index == 0) {
                userLogin(data);
            } else if(e.index == 1)
        {
            vm.openWindow({ url:
'../activate/activate.html', styles: { top: 0 } });
        }
    });
} else if(data.contents == '4')
{
    localStorage.activateToken =
data['access_token'];

    app.closeWaiting();
    mui.alert(lang.activate[0],
'', lang.activate[2], function(e) {
        if(e.index == 0)
            vm.openWindow({ url:
'../activate/activate.html', styles: { top: 0 } });
        });
    } else {
        userLogin(data);
    }

    function userLogin(data) {
        if(localStorage.adminToken !=
null)

            localStorage.adminToken
= data['access_token'];

        else localStorage.token = da
ta['access_token'];

        app.ajax(setting);
    }
},
error: function() {
    mui.toast(lang.loginFailed)
    if (window.plus) mui.fire(plus.w
ebview.getWebviewById('browser_main'), 'logout');
}
```

```
        });
        break;
    case '778':
        alert(resp.message);
        if (window.plus)
            mui.fire(plus.webview.getWebviewById
('browser_main'), 'logout');
        else mui.plusReady(function() {
            mui.fire(plus.webview.getWebviewById
('browser_main'), 'logout');
        })
        break;
    case '401':
    case '200':
    case '204':
    case '400':
    case '404':
    case '429':
    case '500':
    default:
        if (resp.message.indexOf("Token无效")>-1)
{
            app.ajax({
                url: app.api.getDevices,
                data: {
                    psize: 100,
                },
                success: function(d) {
                    var devices = d.items.map(fu
nction(item) {
                        return item.cid
                    });
                    if (devices.indexOf(app.getU
ser().uid)==-1) {
                        mui.toast(lang.lastCid+a
pp.getUser().uid+lang.removed+devices[0]);
                        var u = JSON.parse(local
Storage.user);
                        u.uid = devices[0];
                        localStorage.user = JSON
```

```
.stringify(u);
                                }
                                }
                            })
                        } else if(resp.message != '访问过于频繁')
                            mui.toast(resp.message);
                        mui.isFunction(setting.error) && setting
.error(resp.result);
                            break;
                        }
                    },
                    complete: function(xhr, status) {
                        if(setting.showWaiting && setting.closeWaiting !
= false) {
                            app.closeWaiting();
                        }
                        var resp = xhr.response;
                        if(status === 'success' && defaultSetting.dataTy
pe.toLowerCase() === 'json') {
                            resp = JSON.parse(resp);
                        }
                        if(typeof setting.complete == 'function') settin
g.complete(resp);
                    },
                    error: function(xhr, type, errorThrown) {
                        var errorMsg = {
                            'timeout': lang.timeout,
                            'error': lang.error,
                            'parsererror': lang.parsererror
                        };
                        mui.toast((errorMsg[type] || lang.ajaxError));
                        if(typeof setting.error == 'function') setting.e
rror(xhr, type, errorThrown);
                    }
                };
                setting.beforeSend && (defaultSetting.beforeSend = setti
ng.beforeSend);
                defaultSetting.data = setting.data || {};

                if(setting.url && skipAPI.indexOf(setting.url) == -1 &&
```



```
!defaultSetting.data.cid) {
    var user = app.getUser();
    user && (defaultSetting.data.cid = user.uid);
}
for (var i in defaultSetting.data) {
    if (defaultSetting.data[i]==null) delete defaultSetting.data[i];
}
defaultSetting.headers = {
    flag: app.config.flag,
    lang: navigator.language.slice(0, 2)
};
if(localStorage.adminToken != null) {
    defaultSetting.headers.token = localStorage.adminToken;
} else if(localStorage.token != null) {
    defaultSetting.headers.token = localStorage.token;
}
if(setting.headers) mui.extend(true, defaultSetting.headers, setting.headers);
if(setting.showWaiting) app.showWaiting(typeof setting.showWaiting === 'string' ? setting.showWaiting : undefined);
app.log('token:' + defaultSetting.headers.token);
app.log('正在请求api:' + setting.url);
app.log('发送数据:' + JSON.stringify(defaultSetting.data));
mui.ajax(setting.url, defaultSetting);
};
```

## 二、信鸽方法配置

```
//信鸽推送
mui.plusReady(function() {
    var _BARCODE = 'XGPush', B = window.plus.bridge;
    window.plus.XGPush = {
        unregister: function() {
            return B.execSync(_BARCODE, "unregister", []);
        },
        getToken: function() {
            return B.execSync(_BARCODE, "getToken", []);
        },
        //arg123是内容，分三行显示，extra是额外信息用于点击后处理
        createMessage: function(arg1, arg2, arg3, title, extra)
    {
        return B.execSync(_BARCODE, "createMessage", [arg1,
arg2, arg3, title, extra]);
    },
    };
    if (mui.os.android) {
        window.plus.XGPush.register = function(sc, ec) {
            return B.exec(_BARCODE, "register", [B.callbackId(sc
, ec)]);
        }
    } else { //ios下register为关键词
        window.plus.XGPush.register = function(sc, ec) {
            return B.exec(_BARCODE, "registers", [B.callbackId(s
c, ec)]);
        }
    }
})
```

### 三、特定对象获取定义

```
Object.defineProperty(window, 'lsUserInfo', {
  get: function() {
    return localStorage.userInfo ? JSON.parse(localStorage.userInfo) : {
      "info": {
        "cowner": "",
        "usex": 0,
        "ubirthday": "",
        "epphoto": './userimg.png',
        "bgphoto": './back.png',
        "clink": "",
        "cendtime": '',
      },
      "devList": {
        "list": []
      }
    };
  }
});
```

#### 四、定义全局变量

```
window.User = function (uid, role, clink) {
  this.uid = uid;
  this.role = role || 'normal';
  this.clink = clink || '';
}
```

#### 五、定义全局函数方法

```
/**
 * 获取User
 */
app.getUser = function() {
  return JSON.parse(localStorage.user || '{}');
};
```

## 六、公共样式（**src/common.css**）

stylus预处理器设置的全局变量处理

```
//调小英文字体
html:lang(en) {
  font-size: 110%;
}
```

## 七、接口统一配置

配置app的所有接口

```
if(app.config.isDebug) {
  //app.config.apiDomain = 'http://che.ijiashequ.cn:8010/loveApi';
  //app.config.apiDomain = 'http://192.168.33.81/loveApi';
  //API吴长
  //app.config.apiDomain ='http://192.168.33.152:8088/api'
  ;//API刘本地
  //app.config.apiDomain ='http://192.168.33.93:8080/api';
  //API郑本地
  app.config.apiDomain = 'http://192.168.33.158:8007/loveApiIn'; //API裴本地
  //app.config.apiDomain = 'http://192.168.33.114/api';//
  谢API本地
  //app.config.apiDomain = 'http://192.168.33.92:8080/api'
  ; //张磊测试
}
//常用正则表达式
app.Reg = {};
app.Reg.MODULE_NAME = /([\w-]+\)\.html/; //匹配模块名
app.Reg.HTTP_URL = /^((https|http)?:\//)/; //校验http url
app.Reg.IS_MOBILE = /^[3|4|5|7|8]\d{9}$/; //手机号码
app.Reg.EMAIL = /^[a-zA-Z0-9_-]+@[a-zA-Z0-9_-]+(\.[a-zA-Z0-9_-]+)+$/; //邮箱
//所有api接口 http://192.168.33.8:7070/svnpro/weixin/car_guard
d/doc/车卫士通用版业务接口.doc
app.api = {
  'getcaruserinfo': '/carguard/user/getCarUserInfo.do', //
```

```
?name=030650611(车辆用户信息)
    'updatecaruserinfo': '/carguard/user/updateCarUserInfo.do', //?id=16775&num=15001914584&name=德忠
    'getcarusermap': '/carguard/user/getCarUserMapInfo.do', //?name=030619991(首页地理位置+幻灯片宣传信息)
    'getRealTimePosition': '/cheway/point/getRealTimePosition.do', //?cid=030650611(获取实时位置信息)
    'getcardaydistance': '/carguard/user/getCarTance.do', //?cid=030619991(今日历程<单位:千米>)
    'getlastwarninfo': '/carguard/warn/searchLastWarn.do', //?name=030619991(告警记录列表)
    'getwarninfo': '/carguard/warn/searchWarn.do', //?name=030619991&lastId=11111<告警最大内码>(告警记录列表)
    'setwarninfo': '/carguard/warn/setCarWarnByHaved.do', //?id=4907999(设置已读的告警记录)
    'getAlertsRecord': '/cheway/alert/getAlertsRecord.do', //获取一天的告警记录列表
    'saveSuperCare': '/carguard/other/saveSuperCare.do', //?cid=030619991&val=0<0 撤防 1 设防>(设置超级设防和撤防状态)
    'getSuperCare': '/carguard/other/getSuperCare.do', //?cid=030619991(获取超级设防和撤防状态)
    'saveNormalCare': '/carguard/other/saveNormalCare.do', //?cid=030619991&type=0<0:设防,1:撤防>(设置设防和撤防状态)
    'getNormalCare': '/carguard/other/getNormalCare.do', //?cid=030619991(获取设防和撤防状态)
}
```

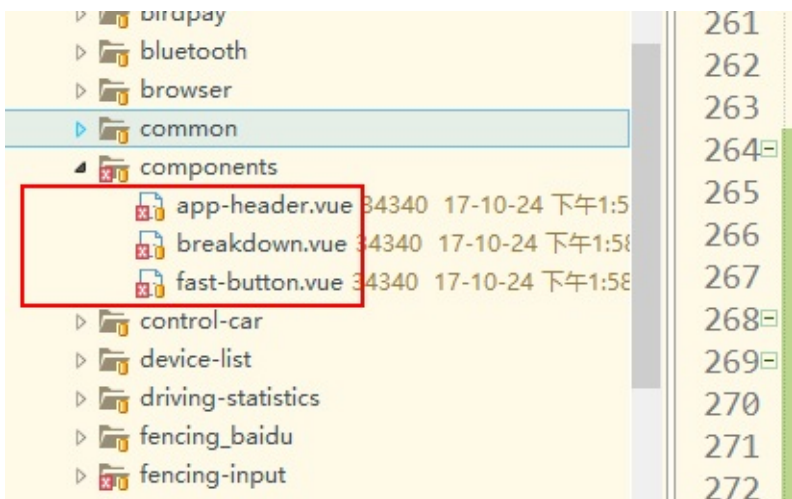
## 视图组件

app等待框和关闭等待框，页面调用app.showWaiting('...'),关闭app.closeWaiting()

```
/** * 显示等待框* @param {String} message*/
app.showWaiting = function(message) {
    if(window.plus) {
        return plus.nativeUI.showWaiting(message || lang.wai
ting);
    }
};
/**关闭等待框*/
app.closeWaiting = function(waitingObj) {
    if(window.plus)(waitingObj && waitingObj.close()) || plu
s.nativeUI.closeWaiting();
};
```

## VUE JS组件

下面是src/components目录JS组件文件的截图，主要是存放app的头部，按钮等组件



## app自己的业务

目前app自己所特有的业务逻辑有自动登录、登出、设备号类型判断、站点本地存储等一些业务处理。具体代码如下：

```
//设备手机信息本地存储，切入后台和前台事件处理，网络监听变化
mui.plusReady(function() {
    plus.runtime.getProperty(plus.runtime.appid, function(wgtinfo) {
        var info = {
            model: plus.device.model, //手机型号
            vendor: plus.device.vendor, //手机厂家
            osName: plus.os.name, //操作系统名称
            osVersion: plus.os.version, //操作系统版本
            osLanguage: plus.os.language, //操作系统语言
            appid: plus.runtime.appid, //程序包名
            appName: wgtinfo.name, //程序名称
            appVersion: wgtinfo.version, //程序版本号
            appDescription: wgtinfo.description, //程序简介

            uuid: plus.device.uuid
            //clientid: plus.push.getClientInfo().clientId, //推送服务令牌（设备唯一标识）
        };
        localStorage.version = wgtinfo.version;
        localStorage.setItem('info', JSON.stringify(info));

        if (cnt.cnt == 0 || localStorage.lastVersion == localStorage.version) {
            vm.go();
        } else {
            vm.getIntro();
        }
    });
    //无网络提示
    document.addEventListener('netchange', function() {
        var nt = plus.networkinfo.getCurrentType();
        switch(nt) {
            case plus.networkinfo.CONNECTION_ETHERNET:
```

```
        case plus.networkinfo.CONNECTION_WIFI:
        case plus.networkinfo.CONNECTION_CELL2G:
        case plus.networkinfo.CONNECTION_CELL3G:
        case plus.networkinfo.CONNECTION_CELL4G:
            break;
        default:
            //防止多个出现警告
            if (!localStorage.alerting) {
                localStorage.alerting = 'true';
                mui.alert('当前无网络连接，您将无法收到推
送告警', '警告', '我知道了', function() {
                    localStorage.removeItem('alertin
g');

                });
            }
            break;
    }
}, false);
app.log('启动耗费时间:' + plus.runtime.launchLoadedTi
me);

//切入后台事件
document.addEventListener('pause', function() {
    //发送数据start
    if(localStorage.appId == 'com.birdsh.cheways.dys
') {

        logeye.emit({
            events: 'dys_index_pause'
        })
    }
    if(localStorage.appId == 'com.birdsh.cheways.lf'
) {

        logeye.emit({
            events: 'lf_index_pause'
        })
    }
    //发送数据end
}, false)

//切入前台事件
document.addEventListener('resume', function() {
    //发送数据start
```



```
        if(localStorage.appId == 'com.birdsh.cheways.dys
    ') {
        logeye.emit({
            events: 'dys_index_resume'
        })
    }
    if(localStorage.appId == 'com.birdsh.cheways.lf'
) {
        logeye.emit({
            events: 'lf_index_resume'
        })
    }
    //发送数据end
    }, false)
    //设置只支持竖屏显示
    plus.screen.lockOrientation('portrait-primary');
});

//app自动登录
methods: {
    getIntro() {
        for (var i = 1; i <= cnt.cnt; i++) {
            this.introImgs.push({src: '../../special/introdu
ce/'+i+'.png'})
        }
        localStorage.lastVersion = localStorage.version;
    },
    go() {
        if(localStorage.adminToken != null) {
            mui.openWindow(pageConfig['device-list']);
        } else if(localStorage.token != null) {
            mui.openWindow(pageConfig['main']);
        } else
            mui.openWindow(pageConfig['login']);
    }
}

//默认配置
var defaultConfig = {
```

```
hasCare: false, //是否有保养记录
hasCarsPush: false, //是否有车队推送我的消息
hasMyCard: false, //是否有我的卡证
hasMyInsu: false, //是否有我的保险
hasMyMotorcade: false, //是否有我的车队
hasRenew: false, //弹出框是否有续费
hasRenewAndDeal: false, //我的页面是否有去续费和交易记录
hasShare: false, //是否有分享
hasWeiXinPay: false, //是否有微信支付
hideMyCar: false, //是否隐藏我的爱车
hideVoltage: false, //是否隐藏首页和位置里的电压
hideInsuInPay: false, //支付和交易记录界面是否隐藏保险文字
isMotor: false, //首页是否只显示摩托车图标
showServiceEnd: false, //我的页面是否显示服务期止
www: 'http://m.wapreach.com/GetItem?channelid=1661&itemid=49
20&cpid=372', //实时监控跳转链接
}
```

//app退出登录

```
window.addEventListener('logout', function() {
    var loginURL = '../template/login.html?hideFlip=1';
    mui.openWindow({
        url:loginURL,
        id: 'login',
        show: {
            aniShow: 'slide-in-left'
        },
    });
    function close(id) {
        var ww = plus.webview.getWebviewById(id);
        if (ww) ww.close('none');
    }
    close('main');
    close('device-list');
    close('setting');
    close('user-info');
    localStorage.removeItem('encrypt');
    localStorage.removeItem('adminToken');
    localStorage.removeItem('activateToken');
    localStorage.removeItem('token');
```

```
localStorage.removeItem('user');  
localStorage.removeItem('lastCheckPay');  
plus.XGPush.unregister();  
});
```

## app打包及原生JS接口调用

- app打包技术是用eclipse工具通过manifest.json配置打包成android和iOS,具体教程参看dcloud提供的[文档](#)。
- 原生设备的接口教程参看dcloud提供的[文档](#)。
- app的更新，dcloud提供三种解决方案。目前采用第一种解决方案，第二种方案作为备用。
  - 整包(apk/ipa)升级
  - App资源在线升级更新（生成移动App资源升级包直接下载更新）
  - App资源在线增量升级更新（增量升级包是针对某个历史版本到新版本的增量，所以对于升级服务器来讲需要保留所有历史版本，并且分别生成每个历史版本到新版本的增量升级包。）其更新的JS的代码如下

```
// src/mine/update.js
//资源在线升级更新
updateWgt(){
    plus.downloader.createDownload("http://demo.dcloud.net.cn/test/update/H5EF3C469.wgt", {filename:"_doc/update/"}, function(d,status){
        plus.nativeUI.showWaiting("下载wgt文件...");
        if ( status == 200 ) {
            app.log.debug("下载wgt成功："+ d.filename);
            plus.nativeUI.showWaiting("安装wgt文件...");
            plus.runtime.install(d.filename, {} ,function(){
                plus.nativeUI.closeWaiting();
                app.log.debug("安装wgt文件成功！");
                plus.nativeUI.alert("应用资源更新完成！",function(){
                    plus.runtime.restart();
                });
            },function(e){
                plus.nativeUI.closeWaiting();
                app.log.debug("安装wgt文件失败["+e.code+"]："+e.message);
                plus.nativeUI.alert("安装wgt文件失败["+e.code+

```

```

    "]" : "+e.message);
        });
    } else {
        app.log.debug("下载wgt失败!");
        plus.nativeUI.alert("下载wgt失败!");
    }
    plus.nativeUI.closeWaiting();
}).start();
},
//整包更新
updateApk(){
    if(app.Config.device.isAndroid){
        plus.downloader.createDownload("", {filename:"_doc/u
pdate/"}, function(d,status){
            plus.nativeUI.showWaiting("下载app文件...");
            if ( status == 200 ) {
                app.log.debug("下载app成功："+ d.filename);
                plus.nativeUI.showWaiting("安装app文件...");
                plus.runtime.install(d.filename, {} ,functio
n(){
                    plus.nativeUI.closeWaiting();
                    app.log.debug("安装app文件成功!");
                    plus.nativeUI.alert("应用资源更新完成!",fu
nction(){
                        plus.runtime.restart();
                    });
                },function(e){
                    plus.nativeUI.closeWaiting();
                    app.log.debug("安装app文件失败["+e.code+"
: "+e.message);
                        plus.nativeUI.alert("安装app文件失败["+e.c
ode+"] : "+e.message);
                });
            } else {
                app.log.debug("下载wgt失败!");
                plus.nativeUI.alert("下载wgt失败!");
            }
            plus.nativeUI.closeWaiting();
        }).start();
    } else if(app.Config.device.isIOS){

```

```
        //iOS平台的ipa无法安装，此时需要跳转到appstore，提示用户自  
        动点击升级更新，跳转到appstore的方法为打开应用的appstore地址  
        var url='itms-apps://itunes.apple.com/cn/app/hello-h  
        5+/id682211190?l=zh&mt=8';// HelloH5应用在appstore的地址  
        plus.runtime.openURL(url);  
    }  
}
```