

## 一、原生语言：

---

苹果手机：ios系统，由swift和c++/object-c语言编写，后缀名为 ipa（ios开发的安装包的后缀名）  
swift语言是一种开源的语言（半开半闭）

安卓手机：android系统，由java（android）语言编写，后缀名为apk 开源的

wp手机：windowphone系统，由c#语言编写，后缀名分为两类（wp7 wp8的是xap wp8.1以后用8.1的sdk开发的是appx）一般说到混合开发不考虑这个版本

序号	系统	语言	后缀名
1	ios	oc/c++/swift	ipa
2	android	android	apk
3	wp	c#	appx

## 二、混合开发：

---

原生语言开发+js的开发

缘由：

js无法调动系统的原生功能（拍照、短信、打电话、通讯录），但是原生（android、ios、wp）可以  
原生有很大的适配问题（特别是android），js可以很好的解决这个问题

术语：

hybridapp ---- js+ android/ios写的

webapp m站 touch端项目 手机网站 ---- 纯js写的

nativeapp -- 纯android/ios编写的程序

## 三、android + js的开发模式

---

准备工具：

java jdk

链接：[https://pan.baidu.com/s/1\\_LsDWTLTa7Hnj3PtRFzkGg](https://pan.baidu.com/s/1_LsDWTLTa7Hnj3PtRFzkGg) 提取码：b2ke

**\*\*jdk配置文档 \*\***

<https://jingyan.baidu.com/article/6dad5075d1dc40a123e36ea3.html>

**\*\*android-studio \*\***

intellij编辑器插件扩展而来（intellij在13-14年之前特别火爆，直到现在还被很多人青睐）

链接：<https://pan.baidu.com/s/1bFXhgpC2XNJqLdZsmyBLtg> 提取码：rzv6

**\*\*安装教程 \*\***

<https://www.runoob.com/android/android-studio-install.html>

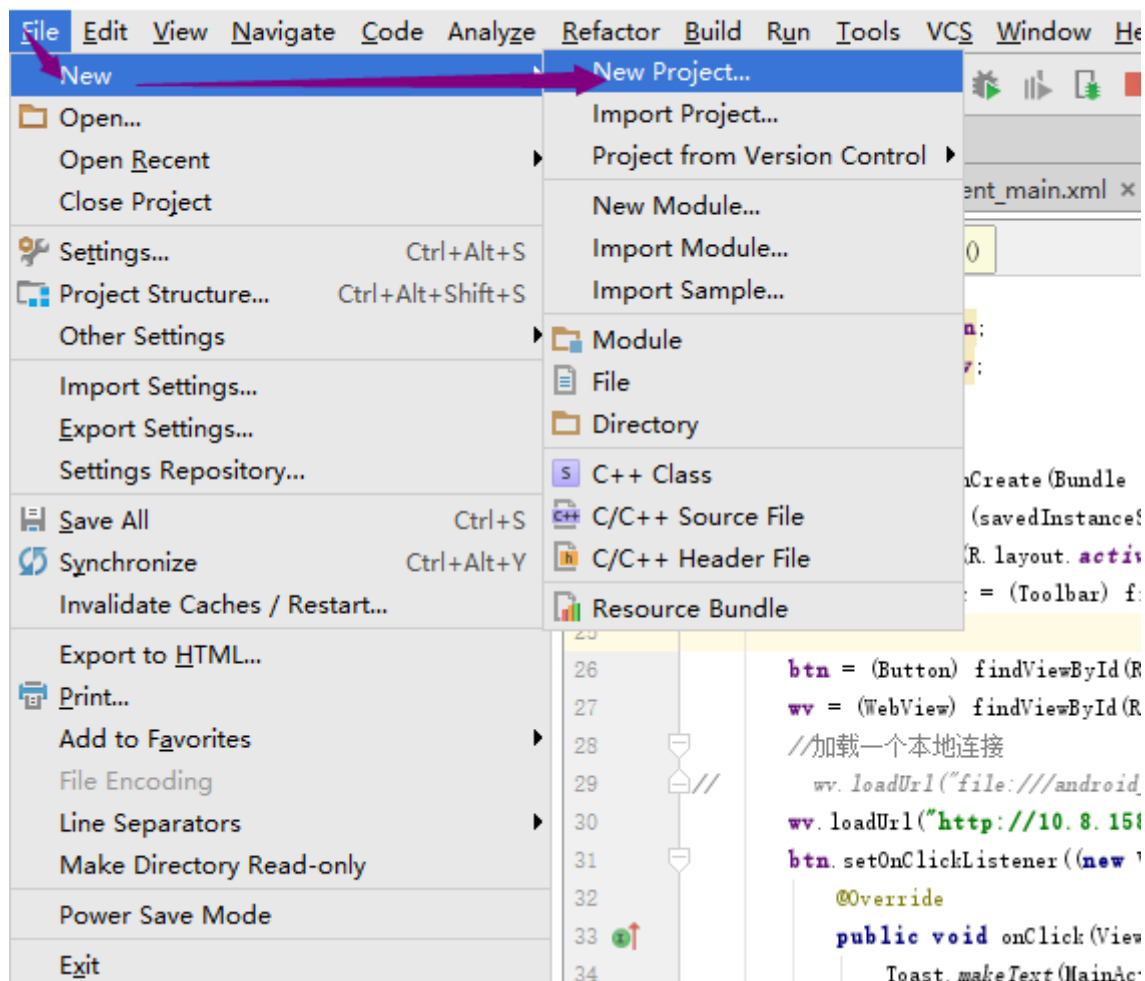
**android手机或者android模拟器**

<http://droid4x.haimawan.com/>

**数据线**

## 四、代码分析

---



#### Configure your new project

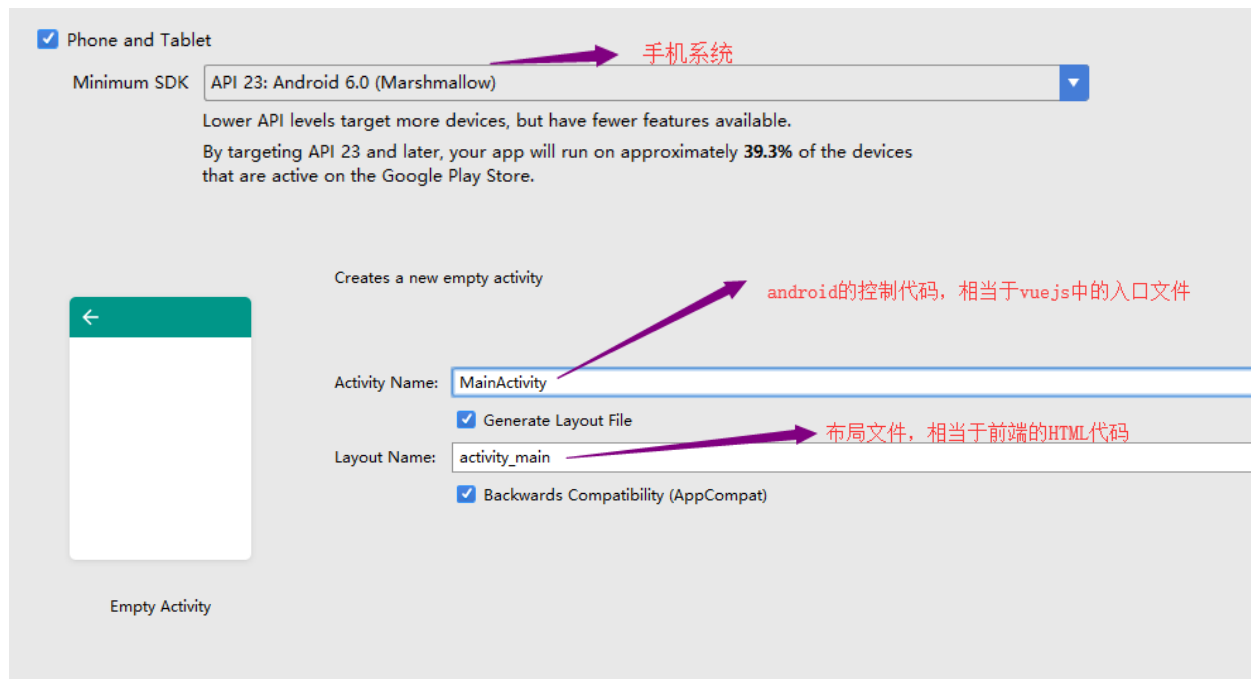
Application name:  应用名称

Company domain:  公司标识

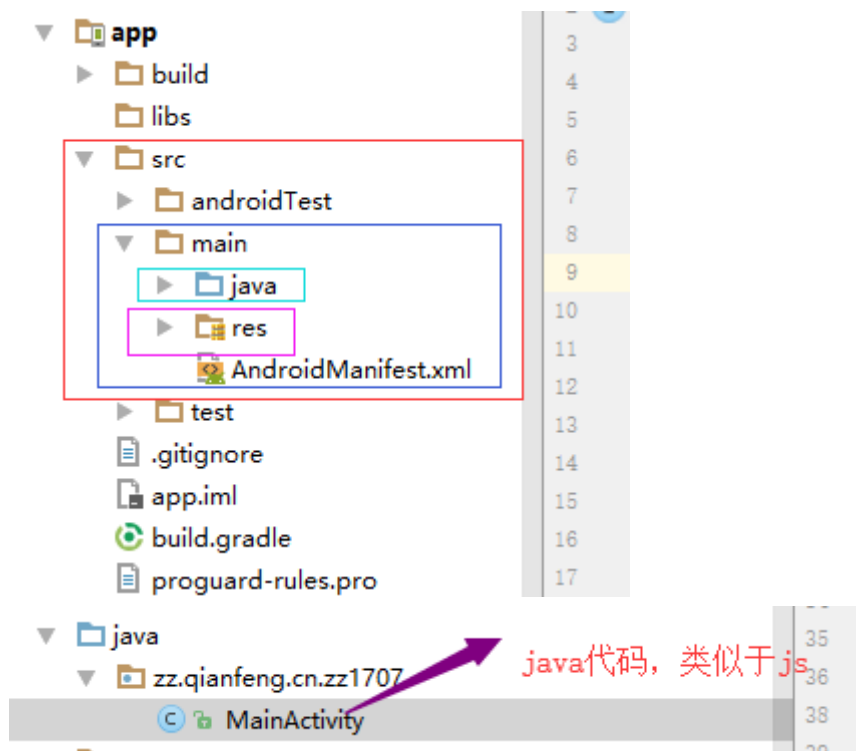
Package name:  应用包名，应用标识  
手机打开软件，提示你更新的依据；  
你点了外卖，外卖推送消息；  
数据的统计

☐ Include C++ support

Project location:  手机软件中的唯一标识，如果两个软件的包名相同，  
后者会将前者覆盖掉



代码展示为java文件夹下为控制文件，相当于js文件，res文件相当于资源文件以及布局文件



app

build

libs

src

androidTest

main

java

zz.qianfeng.cn.zz1707

MainActivity

res

drawable

layout

activity\_main.xml

content\_main.xml

menu

mipmap-hdpi

mipmap-mdpi

mipmap-xhdpi

mipmap-xxhdpi

mipmap-xxxhdpi

```
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
android:theme="@style/AppTheme.AppBarOverlay"
<android.support.v7.widget.Toolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="?attr/colorPrimary"
    app:popupTheme="@style/AppTheme.PopupOverlay" />
</android.support.design.widget.AppBarLayout>
<include layout="@layout/content_main" />
<android.support.design.widget.FloatingActionButton
    android:id="@+id/fab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom|end"
    android:layout_margin="16dp"
    app:srcCompat="@android:drawable/ic_dialog_email" />
</android.support.design.widget.CoordinatorLayout>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="android.support.design.widget.AppBarLayout$ScrollingViewBehavior"
    tools:context="zz.qianfeng.cn.zz1707.MainActivity"
    android:orientation="vertical"
    tools:showIn="@layout/activity_main">
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
/>
<Button
    android:id="@+id/btn"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="button test" />
<WebView
    android:id="@+id/wv"
    android:layout_width="match_parent"
    android:layout_height="match_parent"></WebView>
</LinearLayout>
```

线性布局：从上到下布局

容器有多大，就有多大

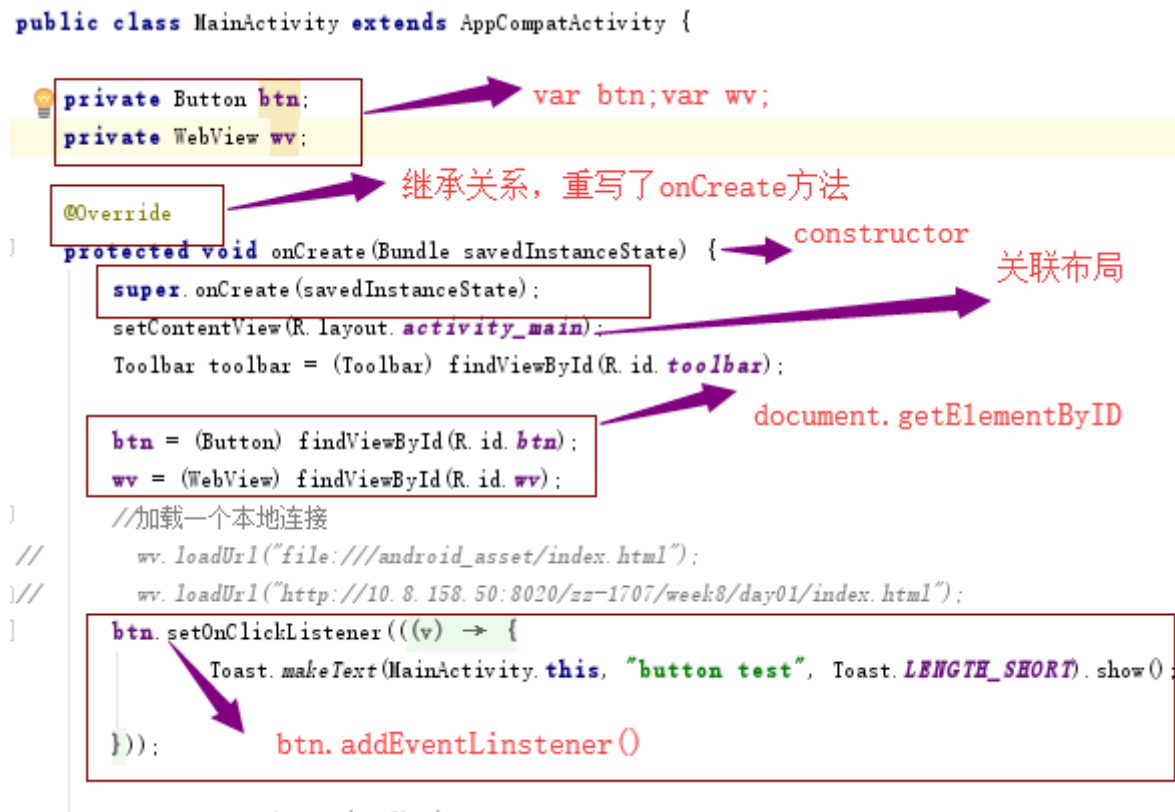
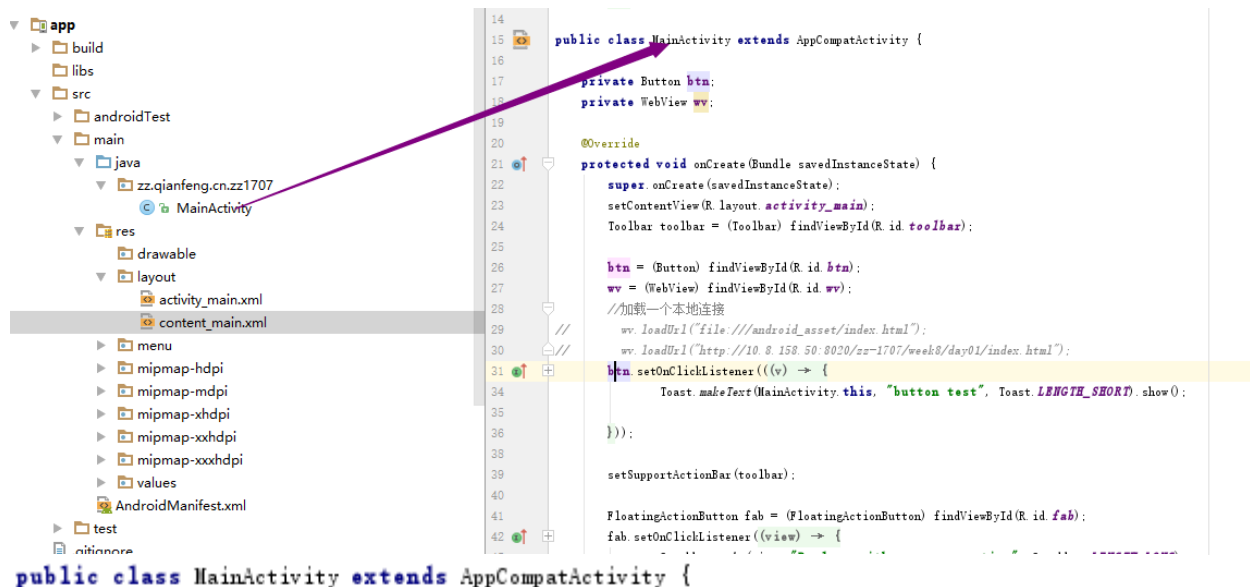
控制线性布局的方向

span/p

内容有多少，宽高就为多少

button#btn

浏览器，由android、ios、wp原生语言进行提供的组件



如需需要给页面添加一个网页

```

ww.loadUrl("file:///android_asset/index.html");    ///加载本地的页面    ----- 代码在手机, 更新需要下载替换
ww.loadUrl("https://h5.m.taobao.com/#index");    /// 加载远程的页面    ----- 更新无需下载

```

默认的webviw浏览器是纯洁的, 不支持js

```

/**
 * webView目前还不具备真正浏览器的功能
 */
webView.getSettings().setJavaScriptEnabled(true); //相当于google浏览器中的内容设置中的js的设置

```

//默认情况下，点击webview中的链接，会使用Android系统自带的浏览器打开这个链接。

//如果希望点击链接继续在我们自己的Browser中响应，必须覆盖 webview的WebViewClient对象：

//新建两个对象MWebViewClient 和 MWebChromeClient，他们分别继承自WebViewClient和WebChromeClient /\* Android WebView 做为承载网页的载体控件，他在网页显示的过程中会产生一些事件，并回调给我们的应用程序，以便我们在网页加载过程中做应用程序想处理的事情。比如说客户端需要显示网页加载的进度、网页加载发生错误等等事件。

WebView提供两个事件回调类给应用层，分别为WebViewClient,WebChromeClient开发者可以继承这两个类，接手相应事件处理。

WebViewClient 主要提供网页加载各个阶段的通知，比如网页开始加载onPageStarted，网页结束加载onPageFinished等； WebChromeClient主要提供网页加载过程中提供的的数据内容，比如返回网页的title,favicon等。\*/

```
//处理页面加载各个阶段
MWebViewClient mWebViewClient = new MWebViewClient(webView, getApplicationContext());
webView.setWebViewClient(mWebViewClient);
//提供网页加载过程中提供的的数据内容
MWebChromeClient mWebChromeClient = new MWebChromeClient( getApplicationContext());
webView.setWebChromeClient(mWebChromeClient);
```

```
MWebViewClient.java
public class MWebViewClient extends WebViewClient {
    private WebView webView;
    private Context context;

    public MWebViewClient(WebView webView) {
        super();
        this.webView = webView;
    }

    public MWebViewClient(WebView webView, Context context) {
        super();
        this.webView = webView;
        this.context = context;
    }

    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        webView.loadUrl(url);
        return true;
    }

    @Override
    public void onPageStarted(WebView view, String url, Bitmap favicon) {
        super.onPageStarted(view, url, favicon);
    }
}
```

```

@Override
public void onPageFinished(WebView view, String url) {
    super.onPageFinished(view, url);
}

@Override
public void onLoadResource(WebView view, String url) {
    super.onLoadResource(view, url);
}

@Override
public void onReceivedSslError(WebView view,
    SslErrorHandler handler, SslError error) {
    super.onReceivedSslError(view, handler, error);
}
}

```

MWebChromeClient.java

```

public class MWebChromeClient extends WebChromeClient {
    private Context context;

    public MWebChromeClient(Context context) {
        super();
        this.context = context;
    }

    @Override
    public boolean onJsAlert(WebView view, String url, String message,
        JsResult result) {
        return super.onJsAlert(view, url, message, result);
    }

    @Override
    public void onReceivedTitle(WebView view, String title) {
        super.onReceivedTitle(view, title);
    }

    @Override
    public boolean onJsConfirm(WebView view, String url, String message,
        JsResult result) {
        return super.onJsConfirm(view, url, message, result);
    }

    @Override
    public boolean onJsPrompt(WebView view, String url, String message,
        String defaultValue, JsPromptResult result) {
        return super.onJsPrompt(view, url, message, defaultValue, result);
    }
}

```

假设你现在需要调用系统的拍照功能，android工程师会为你提供一个对象和他对应的方法  
WebViewTakePhoto为对象，takePhoto为拍照的方法，



```
webView.addJavascriptInterface(new MJavascriptInterface1(getApplicationContext()),  
"WebViewTakePhoto");
```

通过这样的形式将地址传递给前端，testParams方法由前端定义

```
webView.loadUrl("javascript:testParams(http://img4.imgtn.bdimg.com/it/u=103061881,2842093305&fm=26&gp=0.jpg)");
```

```
MJavascriptInterface1  
class MJavascriptInterface1{  
    private Context context;  
  
    public MJavascriptInterface1(Context context) {  
        // TODO Auto-generated constructor stub  
        this.context = context;  
    }  
    @JavascriptInterface  
    public void takePhoto(){  
        Toast.makeText(context, "拍照", Toast.LENGTH_SHORT).show();  
        //android调用拍照功能  
  
        //  
        webView.loadUrl("javascript:testParams('http://img4.imgtn.bdimg.com/it/u=103061881,2842093305&fm=26&gp=0.jpg')");  
    }  
}
```

前端调用拍照方法，在自己的写的触发函数中

```
window.WebViewTakePhoto.takePhoto()
```

说：webview --- android、ios的组件 ---- js运行的环境

进行配置

```
--- 支持js  
  
--- 链接本组件内跳转
```

根据需求先写页面，假设你的一个按钮需要调用系统的功能，在你的按钮事件中调用由android工程师提供的相对应的对象和其方法即可，同时前端也会定义一些方法，但是前端自己不调用，android会根据webview的loadUrl方法进行调用，并且传参（根据需求看）

百度网盘链接地址

## 四、IOS + js的混合开发

---

参考百度网盘观看

链接: <https://pan.baidu.com/s/19fZvDn2NWjoLHTa2L09MRQ> 提取码: ogvx