WebView

简介

WebView是android中一个非常重要的控件，它的作用是用来展示一个web页面。它使用的内核是`webkit`引擎，4.4版本之后，直接使用Chrome作为内置网页浏览器。

作用

1. 显示和渲染网页；
2. 可与页面JavaScript交互，实现混合开发。

使用介绍

使用WebView之前，不要忘记在清单文件中声明访问网络权限：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

1、加载页面

加载页面一般有以下几种形式：

```
    //方式一：加载一个网页
     webView.loadUrl("http://www.baidu.com");

     //方式二：加载应用资源文件内的网页
     webView.loadUrl("file:///android_asset/test.html");

     //方式三：加载一段代码
     webView.loadData(String data,String mimeType, String encoding);
```

其中，方式一和方式二比较好理解， 着重介绍方式三：

`WebView.loadData()`和`WebView.loadDataWithBaseURL()`是表示加载某一段代码，其中，`WebView.loadDataWithBaseURL()`兼容性更好，适用场景更多；

`webView.loadDataWithBaseURL(String baseUrl, String data, String mimeType, String encoding, String historyUrl))`的五个参数：`baseUrl`表示基础的网页，`data`表示要加载的内容，`mimeType`表示加载网页的类型，`encoding`表示编码格式，`historyUrl`表示可用历史记录，可以为`null`值。

举个例子：

```
String body = "示例：这里有个img标签，地址是相对路径<img
src='/uploads/allimg/130923/1FP02V7-0.png' />";
webView.loadDataWithBaseURL("http://www.jcodecraeer.com", body, "text/html",
"utf-8",null);
```

加载后的网页：

加载有Header的网页：

```
//加载有Header的网页：
HashMap<String,String> header = new HashMap<>();
...
webView.loadUrl(url,header);
```

注意：
如果直接用上面介绍的这三种方式来加载网页，很有可能会弹出系统浏览器进行网页访问，这样使用体验就会很差！
解决办法是在`loadUrl()`之前加上这样一句代码：

```
webView.setWebViewClient(new WebViewClient());
```

2、WebView的生命周期

WebView的生命周期一般跟随Activity：

```
@Override
protected void onResume() {
    super.onResume();
    //恢复webview的状态（不靠谱）
    webView.resumeTimers();
    //激活webView的状态，能正常加载网页
    webView.onResume();
}

@Override
protected void onPause() {
    super.onPause();
    //当页面被失去焦点被切换到后台不可见状态，需要执行onPause
```

```
    //通过onPause动作通知内核暂停所有的动作，比如DOM的解析、plugin的执行、JavaScript执行。
    webView.onPause();

    //当应用程序(存在webview)被切换到后台时，这个方法不仅仅针对当前的webview而是全局的全应用程序的webview
    //它会暂停所有webview的layout，parsing，javascripttimer。降低CPU功耗。（不靠谱）
    webView.pauseTimers();
}

//在关闭了Activity时，如果Webview的音乐或视频，还在播放。就必须销毁Webview
//但是注意：webview调用destory时,webview仍绑定在Activity上
//这是由于自定义webview构建时传入了该Activity的context对象
//因此需要先从父容器中移除webview,然后再销毁webview:
ViewGroup parent = findViewById(R.id.container);
parent.removeView(webView);
webView.destroy();
```

3、WebView的一些常用方法

- 浏览器是否可以前进/后退

```
//WebView是否可以后退
boolean canGoBack = webView.canGoBack();
//WebView后退
webView.goBack();
//WebView是否可以前进
boolean canGoForward = webView.canGoForward();
//WebView前进
webView.goForward();
//以当前的index为起始点前进或者后退到历史记录中指定的steps
//如果steps为负数则为后退，正数则为前进
boolean canGoBackOrForward = webView.canGoBackOrForward(step);
```

注意，点击系统返回键时，是结束当前的Activity，而非调用WebView的`goBack()`方法。

- 重新加载网页和停止加载

```
webView.reload(); //刷新页面（当前页面的所有资源都会重新加载）
webView.stopLoading(); //停止加载
```

- 清除浏览器缓存

```
//清除网页访问留下的缓存
//由于内核缓存是全局的因此这个方法不仅仅针对webview而是针对整个应用程序.
Webview.clearCache(true);

//清除当前webview访问的历史记录
//只会webview访问历史记录里的所有记录除了当前访问记录
Webview.clearHistory();

//这个api仅仅清除自动完成填充的表单数据，并不会清除WebView存储到本地的数据
Webview.clearFormData();
```

获取WebView高度、内容HTML高度和滚动距离

```
  //获取当前可见区域的顶端距整个页面顶端的距离，也就是当前内容滚动的距离。
  webView.getScrollY();
  //获取WebView控件的高度。
  webView.getHeight();webView.getBottom();
  //获取HTML的高度（原始高度，不包括缩放后的高度）
  webView.getContentHeight();
```

WebView下载文件

```
  /**
  * 当下载文件时打开系统自带的浏览器进行下载，当然也可以对捕获到的 url 进行处理在应用内下载。
  **/
  webView.setDownloadListener(new DownloadListener() {
      @Override
      public void onDownloadStart(String url, String userAgent, String
contentDisposition, String mimetype, long contentLength) {
          Uri uri = Uri.parse(url);
          Intent intent = new Intent(Intent.ACTION_VIEW, uri);
          startActivity(intent);
      }
  });
```

4.Webview的常用工具类

4.1 WebSettings：对WebView进行配置和管理。

```
WebSettings webSettings = webView.getSettings();
    //如果访问的页面中要与Javascript交互，则webview必须设置支持Javascript
    webSettings.setJavaScriptEnabled(true);

    //设置自适应屏幕，两者合用
    webSettings.setUseWideViewPort(true); //将图片调整到适合webview的大小
    webSettings.setLoadWithOverviewMode(true); // 缩放至屏幕的大小

    webSettings.setLayoutAlgorithm(WebSettings.LayoutAlgorithm.SINGLE_COLUMN);
//支持内容重新布局

    //缩放操作
    webSettings.setSupportZoom(true); //支持缩放，默认为true。是下面那个的前提。
    webSettings.setBuiltInZoomControls(true); //设置内置的缩放控件。若为false，则该
WebView不可缩放
    webSettings.setDisplayZoomControls(false); //隐藏原生的缩放控件
    webSettings.setTextZoom(2);//设置文本的缩放倍数，默认为 100

    webSettings.setRenderPriority(WebSettings.RenderPriority.HIGH);  //提高渲染的
优先级

    webSettings.setStandardFontFamily("");//设置 WebView 的字体，默认字体为 "sans-
serif"
    webSettings.setDefaultFontSize(20);//设置 WebView 字体的大小，默认大小为 16
    webSettings.setMinimumFontSize(12);//设置 WebView 支持的最小字体大小，默认为 8

    // 5.1以上默认禁止了https和http混用，以下方式是开启
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
        webSettings.setMixedContentMode(WebSettings.MIXED_CONTENT_ALWAYS_ALLOW);
```

```
    }

    //其他操作
    webSettings.setCacheMode(WebSettings.LOAD_CACHE_ELSE_NETWORK); //关闭webview
中缓存
    webSettings.setAllowFileAccess(true); //设置可以访问文件
    webSettings.setJavaScriptCanOpenWindowsAutomatically(true); //支持通过JS打开新
窗口
    webSettings.setLoadsImagesAutomatically(true); //支持自动加载图片
    webSettings.setDefaultTextEncodingName("utf-8");//设置编码格式
    webSettings.setGeolocationEnabled(true);//允许网页执行定位操作
    webSettings.setUserAgentString("Mozilla/5.0 (Windows NT 10.0; WOW64;
rv:50.0) Gecko/20100101 Firefox/50.0");//设置User-Agent


    //不允许访问本地文件（不影响assets和resources资源的加载）
    webSettings.setAllowFileAccess(false);
    webSettings.setAllowFileAccessFromFileURLs(false);
    webSettings.setAllowUniversalAccessFromFileURLs(false);
```

其中，WebView设置缓存，当加载 html 页面时，WebView会在`/data/data/package/`下生成`database`与`cache`两个文件夹
请求的URL记录保存在`WebViewCache.db`，而URL的内容是保存在`WebViewCache`文件夹下:

```
    //缓存模式如下：
    //LOAD_CACHE_ONLY: 不使用网络，只读取本地缓存数据
    //LOAD_DEFAULT: （默认）根据cache-control决定是否从网络上取数据。
    //LOAD_NO_CACHE: 不使用缓存，只从网络获取数据.
    //LOAD_CACHE_ELSE_NETWORK，只要本地有，无论是否过期，或者no-cache，都使用缓存中的数
据。

    //优先使用缓存:
    webSettings.setCacheMode(WebSettings.LOAD_CACHE_ELSE_NETWORK);
    //不使用缓存:
    webSettings.setCacheMode(WebSettings.LOAD_NO_CACHE)
```

```
//开启 DOM storage API 功能 较大存储空间，使用简单
 settings.setDomStorageEnabled(true);
 //设置数据库缓存路径 存储管理复杂数据 方便对数据进行增加、删除、修改、查询 不推荐使用
 settings.setDatabaseEnabled(true);
 final String dbPath = context.getApplicationContext().getDir("db",
Context.MODE_PRIVATE).getPath();
 settings.setDatabasePath(dbPath);
 //开启 Application Caches 功能 方便构建离线APP 不推荐使用
 settings.setAppCacheEnabled(true);
 final String cachePath = context.getApplicationContext().getDir("cache",
Context.MODE_PRIVATE).getPath();
 settings.setAppCachePath(cachePath);
 settings.setAppCacheMaxSize(5 * 1024 * 1024);
```

缓存机制	优势	适用场景
浏览器缓存机制	HTTP协议层支持	静态文件的缓存
Dom Storage	较大存储空间，使用简单	临时、简单数据的缓存， Cookies的扩展
Web SQL Database	存储、管理复杂结构数据	用IndexedDB替代， 不推荐使用
AppCache	方便构建离线App	离线App、静态文件缓存， 不推荐使用
IndexedDB	存储任何类型数据、使用简单， 支持索引	结构、关系复杂的数据存储 Web SQL Database的替代
File System API	支持文件系统的操作	数据适合以文件进行管理的场景， Android系统还不支持

4.2 WebViewClient：处理各种通知和请求事件。

- `shouldOverrideUrlLoading()`：拦截URL请求，重定向（有2个方法，一个是兼容5.0以下，一个是兼容5.0以上，保险起见两个都重写）。
  无论返回`true`还是`false`，只要为WebView设置了`WebViewClient`，系统就不会再将url交给第三方的浏览器去处理了。q其中返回`false`，代表将url交给当前WebView加载，也就是正常的加载状态；`shouldOverrideUrlLoading()`返回`true`，代表开发者已经对url进行了处理，WebView就不会再对这个url进行加载了。
  另外，使用post的方式加载页面，此方法不会被调用。

```
webView.setWebViewClient(new WebViewClient(){

      //重定向URL请求，返回true表示拦截此url，返回false表示不拦截此url。
      @Override
      public boolean shouldOverrideUrlLoading(WebView view, String url) {
          //作用1：重定向url
          if(url.startsWith("weixin://")){
              url = url.replace("weixin://","http://");
              webView.loadUrl(url);
          }

          //作用2：在本页面的webView打开，防止外部浏览器打开此链接
          view.loadUrl(url);
          return true;
      }
  });
```

- `onPageStarted()onPageFinished()`：页面加载时和页面加载完毕时调用。
- `shouldOverrideKeyEvent()`：重写此方法才能处理浏览器中的按键事件。
- `shouldInterceptRequest()`：页面每一次请求资源之前都会调用这个方法（**非UI线程调用**）。
- `onLoadResource()`：页面加载资源时调用，每加载一个资源（比如图片）就调用一次。
- `onReceivedError()`：加载页面的服务器出现错误（比如404）时回调。
- `onReceivedSslError()`：重写此方法可以让webview处理https请求。

- `doUpdateVisitedHistory()`：更新历史记录。
- `onFormResubmission()`：应用程序重新请求网页数据。
- `onReceivedHttpAuthRequest()`：获取返回信息授权请求。
- `onScaleChanged()`：WebView发生缩放改变时调用。
- `onUnhandledKeyEvent()`：Key事件未被加载时调用。

补充，关键方法调用流程：

情况一：`loadUrl()`无重定向时

```
onPageStarted->onPageFinished
```

情况二：`loadUrl()`网页A重定向到B时

```
onPageStarted->shouldOverrideUrlLoading->onPageStarted->onPageFinished-
>onPageFinished
```

情况三：在已加载的页面中点击链接，加载页面A（无重定向）

```
shouldOverrideUrlLoading->onPageStarted->onPageFinished
```

情况四：在已加载的页面中点击链接，加载页面A（页面A重定向至页面B）

```
shouldOverrideUrlLoading->onPageStarted->shouldOverrideUrlLoading-
>onPageStarted->onPageFinished->onPageFinished
```

情况五：执行`goBack/goForward/reload`方法

```
onPageStarted->onPageFinished
```

情况六：发生资源加载

```
shouldInterceptRequest->onLoadResource
```

4.3 WebChromeClient：辅助 WebView 处理 Javascript 的对话框，网站图标，网站标题等等。

`WebChromeClient`类常用方法：

- `onProgressChanged()`：获得网页的加载进度并显示。
- `onReceivedTitle()`：获得网页的标题时回调。
- `onReceivedIcon()`：获得网页的图标时回调。
- `onCreateWindow()`：打开新窗口时回调。
- `onCloseWindow()`：关闭窗口时回调。
- `onJsAlert()`：网页弹出提示框时触发此方法。

```
 @Override
  public boolean onJsAlert(WebView view, String url, String message, JsResult
result) {
      Toast.makeText(MainActivity.this,"Im alert",Toast.LENGTH_SHORT).show();

      //部分机型只会弹出一次提示框，调用此方法即可解决此问题。
      result.cancel();
      //返回true表示不弹出系统的提示框，返回false表示弹出
      return true;
  }
```

- `onJsConfirm()`：网页弹出确认框时触发此方法。

```
 @Override
  public boolean onJsConfirm(WebView view, String url, String message, JsResult
result) {

      //confirm表示确认，cancel表示取消。
      result.confirm();
      //返回true表示不弹出系统的提示框，返回false表示弹出
      return true;
  }
```

- `onJsPrompt()`：网页弹出输入框时触发此方法。

```
@Override
  public boolean onJsPrompt(WebView view, String url, String message, String
defaultValue, JsPromptResult result) {
      //confirm表示确认（并返回值），cancel表示取消。
      result.confirm("8848");
      //返回true表示不弹出系统的提示框，返回false表示弹出
      return true;
  }
```

4.4 Cookie 相关

WebView可以在需要的时候自动同步cookie，只需要获得`CookieManager`的对象将cookie设置进去就可以了。

```
从服务器的返回头中取出 cookie 根据Http请求的客户端不同，获取 cookie 的方式也不同，请自行获取。
```

4.4.1 设置cookie，若两次设置相同，则覆盖上一次的

```
/**
 * 将cookie设置到 WebView
 * @param url 要加载的 url
 * @param cookie 要同步的 cookie
 */
public static void syncCookie(String url,String cookie) {
    if (Build.VERSION.SDK_INT < Build.VERSION_CODES.LOLLIPOP) {
        CookieSyncManager.createInstance(context);
    }
    CookieManager cookieManager = CookieManager.getInstance();
    cookieManager.setCookie(url, cookie);//如果没有特殊需求，这里只需要将session id
以"key=value"形式作为cookie即可
}
```

注意：

1. 同步 cookie 要在 WebView 加载 url 之前，否则 WebView 无法获得相应的 cookie，也就无法通过验证。
   cookie应该被及时更新，否则很可能导致WebView拿的是旧的session id和服务器进行通信。
2. cookie应该被及时更新，否则很可能导致WebView拿的是旧的session id和服务器进行通信。
3. CookieManager会将这个Cookie存入该应用程序
   `data/data/package_name/app_WebView/Cookies.db`

4.4.2 获取cookie

```
/**
 * 获取指定 url 的cookie
 */
public static String syncCookie(String url) {
    CookieManager cookieManager = CookieManager.getInstance();
    return cookieManager.getCookie(url);
}
```

4.4.3 清除cookie

```
// 这个两个在 API level 21 被抛弃
CookieManager.getInstance().removeSessionCookie();
CookieManager.getInstance().removeAllCookie();

// 推荐使用这两个， level 21 新加的
CookieManager.getInstance().removeSessionCookies();// 移除所有过期 cookie
CookieManager.getInstance().removeAllCookies(); // 移除所有的 cookie

//设置清除cookie后的回调方法
private void removeCookie(Context context) {
    CookieManager.getInstance().removeAllCookies(new ValueCallback<Boolean>() {
        @Override
        public void onReceiveValue(Boolean value) {
            // 清除结果
        }
    });
}
```

5、其他使用场景介绍

5.1 长按保存图片或者拨打电话

一般浏览器都有长按保存图片或者拨打电话的功能，实现这个功能和`WebView.HitTestResult`这个类有关，这个类会将你触摸网页的地方的类型和其他信息反馈给你。

`WebView.HitTestResult`的常用方法：

- `HitTestResult.getType()`：获取所选中目标的类型，可以是图片，超链接，邮件，电话等等。
- `HitTestResult.getExtra()`：获取额外的信息。

类型和意义：

```
WebView.HitTestResult.UNKNOWN_TYPE //未知类型

WebView.HitTestResult.PHONE_TYPE //电话类型

WebView.HitTestResult.EMAIL_TYPE //电子邮件类型

WebView.HitTestResult.GEO_TYPE //地图类型

WebView.HitTestResult.SRC_ANCHOR_TYPE //超链接类型

WebView.HitTestResult.SRC_IMAGE_ANCHOR_TYPE //带有链接的图片类型

WebView.HitTestResult.IMAGE_TYPE //单纯的图片类型

WebView.HitTestResult.EDIT_TEXT_TYPE //选中的文字类型
```

步骤：
1、给WebView设置长按监听事件；
2、获取WebView长按时的`WebView.HitTestResult`的事件类型，如果是图片，则做处理。

```
webView.setOnLongClickListener(new View.OnLongClickListener() {
    @Override
    public boolean onLongClick(View view) {
        WebView.HitTestResult result = ((WebView) view).getHitTestResult();
        if(result != null){
            switch (result.getType()){
                case WebView.HitTestResult.SRC_IMAGE_ANCHOR_TYPE:
                    String imgUrl = result.getExtra();
                    ...
                    return true;
                ...
            }
        }
        return false;
    }
});
```

Android与JavaScript交互

Android与JavaScript的交互依赖于WebView，具体 参考： https://www.jianshu.com/p/5cc2eae14e07 ；

WebView中常见问题

1、Android5.0 WebView中Http和Https混合问题

1.1 在Android 5.0上 Webview 默认不允许加载 Http 与 Https 混合内容，解决办法：

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {

webView.getSettings().setMixedContentMode(WebSettings.MIXED_CONTENT_ALWAYS_ALLOW
);
}
```

参数说明：

- `MIXED_CONTENT_ALWAYS_ALLOW`：允许从任何来源加载内容，即使起源是不安全的；
- `MIXED_CONTENT_NEVER_ALLOW`：不允许Https加载Http的内容，即不允许从安全的起源去加载一个不安全的资源；
- `MIXED_CONTENT_COMPATIBILITY_MODE`：当涉及到混合式内容时，WebView 会尝试去兼容最新Web浏览器的风格。

1.2 设置WebView接受所有网站的证书

```
webView.setWebViewClient(new WebViewClient() {
            @Override
            public void onReceivedSslError(WebView view, SslErrorHandler
handler, SslError error) {
                // handler.cancel();// Android默认的处理方式
                handler.proceed();// 接受所有网站的证书
                // handleMessage(Message msg);// 进行其他处理
            }
});
```

注意：一定要移除`super.onReceivedSslError(view, handler, error)`方法，否则不生效。

2、避免WebView引起的内存泄漏

2.1 WebView所在的Activity新启一个进程

```
<activity
    android:name=".ui.activity.Html5Activity"
    android:process=":lyl.boon.process.web">
    <intent-filter>
        <action android:name="com.lyl.boon.ui.activity.htmlactivity"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</activity>
```

结束的时候直接`System.exit(0)`退出当前进程。

2.2 动态创建WebView

在代码中创建WebView，而不是在XML中创建：

```
LinearLayout.LayoutParams params = new
LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.MATCH_PARENT);
        mWebView = new WebView(getApplicationContext());
        mWebView.setLayoutParams(params);
        mLayout.addView(mWebView);
```

2.3 Activity销毁的时候先移除WebView

```
@Override
protected void onDestroy() {
    if (mWebView != null) {
        ((ViewGroup) mWebView.getParent()).removeView(mWebView);
        mWebView.destroy();
        mWebView = null;
    }
    super.onDestroy();
}
```

3、混淆导致JavaScript不可用

在`proguard-rules.pro`文件中配置：

```
-keepattributes *Annotation*
-keepattributes *JavascriptInterface*
-keep public class org.mq.study.webview.DemoJavaScriptInterface{
    public <methods>;
}
```

如果是内部类：

```
-keepattributes *Annotation*
-keepattributes *JavascriptInterface*
-keep public class
org.mq.study.webview.webview.DemoJavaScriptInterface$InnerClass{
    public <methods>;
}
```

4、监听WebView网页加载完毕

`WebViewClient`的`onPageFinished()`方法不能保证一定是在网页加载完毕后调用，也有可能是一些其他情况下也会调用，因此，最好用`WebChromeClient`的`onProgressChanged()`方法来监听：

```
webView.setWebChromeClient(new WebChromeClient() {

    @Override
    public void onProgressChanged(WebView view, int position) {
        progressBar.setProgress(position);
        if (position == 100) {
            progressBar.setVisibility(View.GONE);
        }
        super.onProgressChanged(view, position);
    }
});
```

5、 HTML5 video 在 WebView 全屏显示

当网页全屏播放视频时会调用`WebChromeClient.onShowCustomView()`方法，所以可以通过将 video 播放的视图全屏达到目的。

```
@Override
public void onShowCustomView(View view, CustomViewCallback callback) {
    if (view instanceof FrameLayout && fullScreenView != null) {
        // A video wants to be shown
        this.videoViewContainer = (FrameLayout) view;
        this.videoViewCallback = callback;
        fullScreenView.addView(videoViewContainer, new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.MATCH_PARENT));
        fullScreenView.setVisibility(View.VISIBLE);
        isVideoFullscreen = true;
    }
}

@Override
public void onHideCustomView() {
    if (isVideoFullscreen && fullScreenView != null) {
        // Hide the video view, remove it, and show the non-video view
        fullScreenView.setVisibility(View.INVISIBLE);
        fullScreenView.removeView(videoViewContainer);

        // Call back (only in API level <19, because in API level 19+ with
chromium webview it crashes)
        if (videoViewCallback != null &&
!videoViewCallback.getClass().getName().contains(".chromium.")) {
            videoViewCallback.onCustomViewHidden();
        }

        isVideoFullscreen = false;
        videoViewContainer = null;
        videoViewCallback = null;
    }
}
```

注意：很多的手机版本在网页视频播放时是不会调用这个方法的，所以这个方法局限性很大

6、`loadData()`加载中文数据出现乱码

使用`loadDataWithBaseURL()`方法代替`loadData()`加载数据，不会出现乱码问题。 为`loadData()`的mimeType参数传入`“text/html;charset=UTF-8”`，也可以解决乱码问题。

7、外部JavaScript攻击

由于应用中的WebView没有对`file:///`类型的url做限制，可能导致外部攻击者利用Javascript代码读取本地隐私数据。

7.1 如果WebView不需要使用file协议，直接禁用所有与file协议相关的功能即可（不影响对assets和resources资源的加载）。

```
webSettings.setAllowFileAccess(false);
webSettings.setAllowFileAccessFromFileURLs(false);
webSettings.setAllowUniversalAccessFromFileURLs(false);
```

7.2 如果WebView需要使用file协议，则应该禁用file协议的Javascript功能。具体方法为：在调用loadUrl方法前，以及在shouldOverrideUrlLoading方法中判断url的scheme是否为file。如果是file协议，就禁用Javascript，否则启用Javascript。

```
//调用loadUrl前
if("file".equals(Uri.parse(url).getScheme())){//判断是否为file协议
    webView.getSettings().setJavaScriptEnabled(false);
}else{
    webView.getSettings().setJavaScriptEnabled(true);
}
webView.loadUrl(url);

//WebViewClient中做的操作
@Override
public boolean shouldOverrideUrlLoading(WebView view, WebResourceRequest
request) {
    if("file".equals(request.getUrl().getScheme())){//判断是否为file协议
        view.getSettings().setJavaScriptEnabled(false);
    }else{
        view.getSettings().setJavaScriptEnabled(true);
    }
    return false;
}
```

8、WebView闪烁

WebView关闭硬件加速功能即可。

```
webView.setLayerType(View.LAYER_TYPE_SOFTWARE,null);
```

9、缩放引起的应用崩溃

`setDisplayZoomControls(true)`方法会允许显示系统缩放按钮，这个缩放按钮会在每次出现后的几秒内逐渐消失。但是在部分系统版本中，如果在缩放按钮消失前退出了Activity，就可能引起应用崩溃。

解决办法：调用`setDisplayZoomControls(false)`方法不显示系统缩放按钮，反正使用手势捏合动作就可以实现网页的缩放功能了。如果确实需要使用缩放按钮，就需要在Activity的`onDestroy()`方法中隐藏WebView。

10、webView控件`padding`不起作用

在一个布局文件中有一个WebView，想使用padding属性让左右向内留出一些空白，但是padding属性不起左右，内容照样贴边显示，反而移动了右边滚动条的位置。android的bug，用一个外围的layout包含webview，可以有所改进，但不能完全解决。其实正确的做法是在webView的加载的css中增加padding,没必要为了padding而更改xml布局文件。

一个很不错的WebViewActivity分享（来源于网络）

```
import android.graphics.Bitmap;
```

```
import android.os.Bundle;
import android.os.Message;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.KeyEvent;
import android.webkit.GeolocationPermissions;
import android.webkit.WebChromeClient;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;

import com.lyl.test.R;

public class Html5Activity extends AppCompatActivity {

    private String mUrl;

    private LinearLayout mLayout;
    private WebView mWebView;

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

        Bundle bundle = getIntent().getBundleExtra("bundle");
        mUrl = bundle.getString("url");

        Log.d("Url:", mUrl);

        mLayout = (LinearLayout) findViewById(R.id.web_layout);


        LinearLayout.LayoutParams params = new
LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.MATCH_PARENT);
        mWebView = new WebView(getApplicationContext());
        mWebView.setLayoutParams(params);
        mLayout.addView(mWebView);

        WebSettings mWebSettings = mWebView.getSettings();
        mWebSettings.setSupportZoom(true);
        mWebSettings.setLoadWithOverviewMode(true);
        mWebSettings.setUseWideViewPort(true);
        mWebSettings.setDefaultTextEncodingName("utf-8");
        mWebSettings.setLoadsImagesAutomatically(true);

        //调用JS方法.安卓版本大于17,加上注解 @JavascriptInterface
        mWebSettings.setJavaScriptEnabled(true);

        saveData(mWebSettings);

        newWin(mWebSettings);

        mWebView.setWebChromeClient(webChromeClient);
        mWebView.setWebViewClient(webViewClient);
        mWebView.loadUrl(mUrl);
    }
```

```
    @Override
    public void onPause() {
        super.onPause();
        webView.onPause();
        webView.pauseTimers(); //小心这个！！！暂停整个 WebView 所有布局、解析、JS。
    }

    @Override
    public void onResume() {
        super.onResume();
        webView.onResume();
        webView.resumeTimers();
    }

    /**
     * 多窗口的问题
     */
    private void newWin(WebSettings mWebSettings) {
        //html中的_bank标签就是新建窗口打开，有时会打不开，需要加以下
        //然后 复写 WebChromeClient的onCreateWindow方法
        mWebSettings.setSupportMultipleWindows(false);
        mWebSettings.setJavaScriptCanOpenWindowsAutomatically(true);
    }

    /**
     * HTML5数据存储
     */
    private void saveData(WebSettings mWebSettings) {
        //有时候网页需要自己保存一些关键数据,Android WebView 需要自己设置
        mWebSettings.setDomStorageEnabled(true);
        mWebSettings.setDatabaseEnabled(true);
        mWebSettings.setAppCacheEnabled(true);
        String appCachePath =
getApplicationContext().getCacheDir().getAbsolutePath();
        mWebSettings.setAppCachePath(appCachePath);
    }

    WebViewClient webViewClient = new WebViewClient(){

        /**
         * 多页面在同一个WebView中打开，就是不新建activity或者调用系统浏览器打开
         */
        @Override
        public boolean shouldOverrideUrlLoading(WebView view, String url) {
            view.loadUrl(url);
            return true;
        }

    };

    WebChromeClient webChromeClient = new WebChromeClient() {

        //==========HTML5定位
==============================================================
        //需要先加入权限
        //<uses-permission android:name="android.permission.INTERNET"/>
```

```
        //<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>
        //<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
        @Override
        public void onReceivedIcon(WebView view, Bitmap icon) {
            super.onReceivedIcon(view, icon);
        }

        @Override
        public void onGeolocationPermissionsHidePrompt() {
            super.onGeolocationPermissionsHidePrompt();
        }

        @Override
        public void onGeolocationPermissionsShowPrompt(final String origin,
final GeolocationPermissions.Callback callback) {
            callback.invoke(origin, true, false);//注意个函数，第二个参数就是是否同意
定位权限，第三个是是否希望内核记住
            super.onGeolocationPermissionsShowPrompt(origin, callback);
        }
        //=========HTML5定位
==========================================================

        //=========多窗口的问题
==========================================================
        @Override
        public boolean onCreateWindow(WebView view, boolean isDialog, boolean
isUserGesture, Message resultMsg) {
            WebView.WebViewTransport transport = (WebView.WebViewTransport)
resultMsg.obj;
            transport.setWebView(view);
            resultMsg.sendToTarget();
            return true;
        }
        //=========多窗口的问题
==========================================================
    };

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_BACK && mWebView.canGoBack()) {
            mWebView.goBack();
            return true;
        }

        return super.onKeyDown(keyCode, event);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();

        if (mWebView != null) {
            mWebView.clearHistory();
            ((ViewGroup) mWebView.getParent()).removeView(mWebView);
            mWebView.loadUrl("about:blank");
            mWebView.stopLoading();
```

```
            mWebView.setWebChromeClient(null);
            mWebView.setWebViewClient(null);
            mWebView.destroy();
            mWebView = null;
        }
    }

}
```

参考资料

- 史上最全WebView使用，附送Html5Activity一份
- Android：这是一份全面 & 详细的Webview使用攻略
- 带你解决 WebView 里的常见问题
- Android WebView全面讲解