WebView_Kit实践

通过YtWebView.creatWebView(this, 1, SysConfig.LOGGING_ENABLED)获取对象实例，在parentView中addView(mYtWebView)

使用基础设置中的YtWebViewJsInterface,
在YtWebView实例化时已经实现了js的注入

在Application中初始化js的name，
YtWebViewJsInterface.initJsName("MallJsBridge");
若不进行初始化则默认为 MallJsBridge

继承YtWebViewJsInterface或者自定义实现YtJsInterface接口

mYtWebView.addJavascriptInterface(WebViewJsInterfaceTest.class, WebViewJsInterfaceTest.JS_BRIDGE);
若name重复，则会进行覆盖

实现YtWebView.YtWebviewLoadListener接口，改接口整合了WebViewClient 和 WebChromeClient 内方法的接口，统一进行设置

在YtWebview中已经添加了H5选择图片的支持设置，若有改需求，则需要在Activity载体的onActivityResult中实现
mYtWebView.onActivityResult(requestCode, resultCode, data);

实现YtJsCallbackHandler接口，其中handleJsFunctionSelf方法用于js方法分发时进行前置判断，若存在非全局js接口，则可在重写该方法时进行逻辑处理，返回true则不会进行后续全局js的分发

自定义方法类HandlerTest，在该类中定义js静态方法；
若不想单独管理js方法的具体实现，也可以将方法直接写在下一步的HandlerRegistry中

自定义HandlerRegistry实现YtJsHandlerRegistrar接口，重写registry()，在此处进行js方法注册。
YTJavaScriptFactory.registry("Fu'n'c'Name", HandlerTest.class);

在Application中完成js注册初始化
new HandlerRegistry().registry();