验证码

接入指引

产品文档

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2024 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

【商标声明】

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

文档目录

接入指引
- 客户端接入
  - Web 客户端接入
  - App 客户端接入
- 服务端接入
  - 接入票据校验（Web 及 App）

版权所有：腾讯云计算（北京）有限责任公司

接入指引
客户端接入
Web 客户端接入

最近更新时间：2023-12-21 11:25:21

前提条件

客户端接入前，需完成新建验证，并在**验证列表**获取所需的 CaptchaAppId 以及 AppSecretKey。步骤如下：

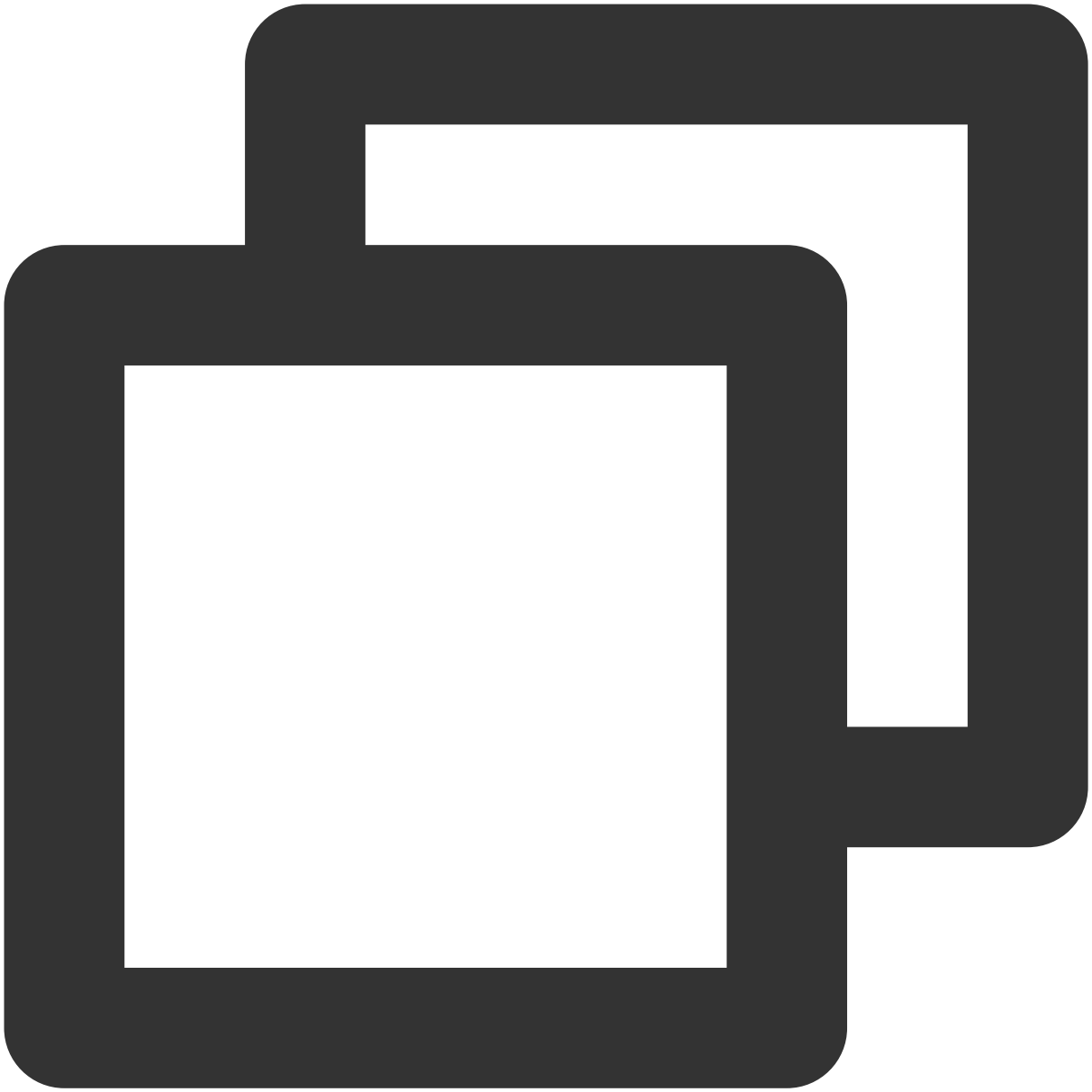
1. 登录 验证码控制台，左侧导航栏选择**图形验证** > **验证管理**，进入验证管理页面。
2. 单击**新建验证**，根据业务场景需求，设置验证名称等参数。
3. 单击**确定**，完成新建验证，即可在验证列表中查看验证码 CaptchaAppId 及 AppSecretKey。

代码示例

以下代码示例，单击**验证**，激活验证码，并弹窗展示验证结果。

注意：

该示例未展示调用票据校验 API 的逻辑。业务客户端完成验证码接入后，业务服务端需二次核查验证码票据结果（未接入票据校验，会导致黑产轻易伪造验证结果，失去验证码人机对抗效果），详情请参见：接入票据校验（Web 及 App）。



```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Web 前端接入示例</title>
    <!-- 验证码程序依赖(必须)。请勿修改以下程序依赖，如通过其他手段规避加载，会导致验证码无法正常
    <script src="https://ca.turing.captcha.qcloud.com/TCaptcha-global.js"></script>
</head>
```

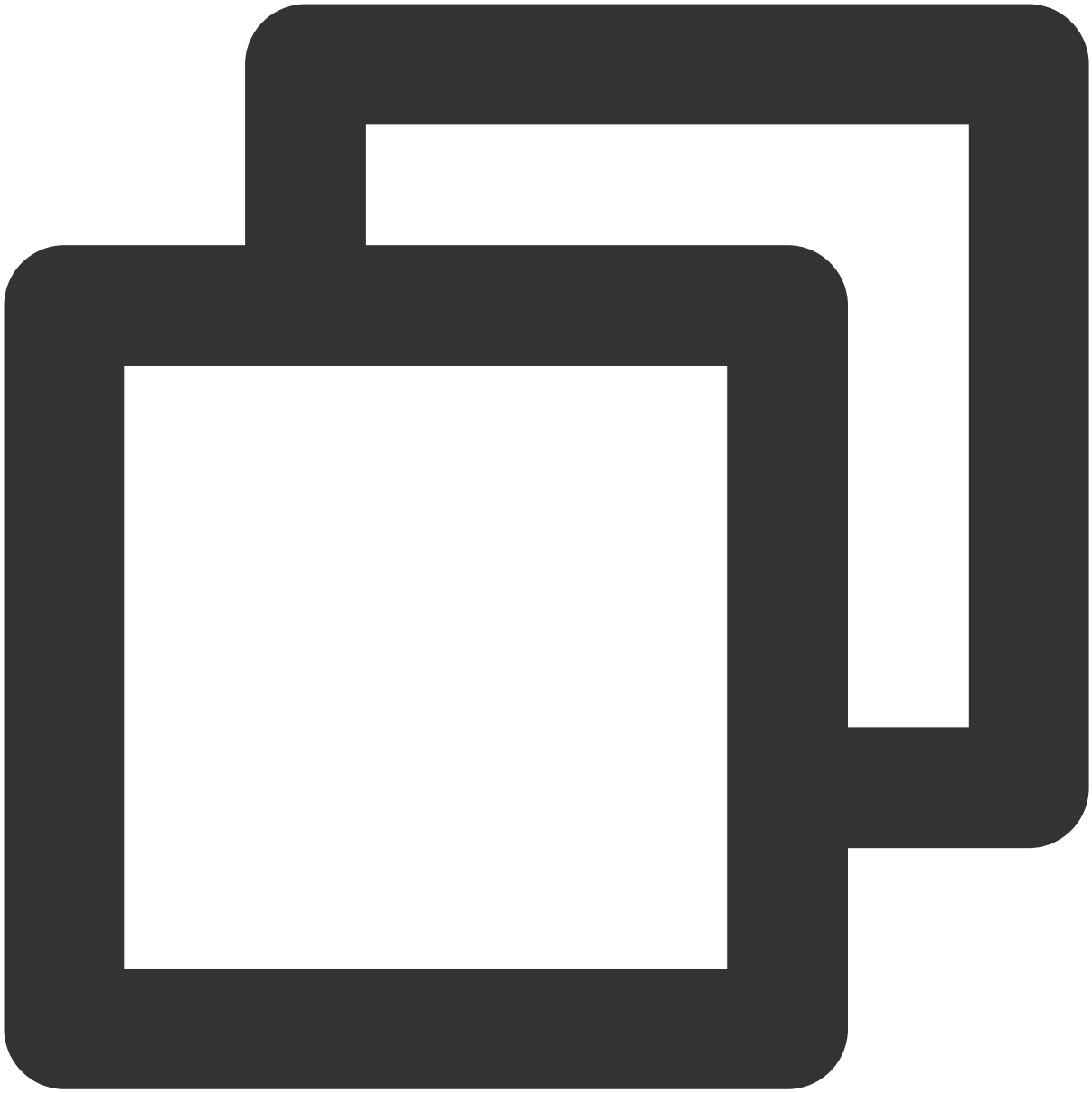
```
<body>
    <button id="CaptchaId" type="button">验证</button>
</body>

<script>

    // 定义回调函数
    function callback(res) {
        // 第一个参数传入回调结果，结果如下：
        // ret         Int        验证结果，0：验证成功。2：用户主动关闭验证码。
        // ticket      String     验证成功的票据，当且仅当 ret = 0 时 ticket 有值。
        // CaptchaAppId        String     验证码应用ID。
        // bizState    Any        自定义透传参数。
        // randstr     String     本次验证的随机串，后续票据校验时需传递该参数。
        console.log('callback:', res);


        // res（用户主动关闭验证码）= {ret: 2, ticket: null}
        // res（验证成功） = {ret: 0, ticket: "String", randstr: "String"}
        // res（请求验证码发生错误，验证码自动返回terror_前缀的容灾票据） = {ret: 0, ticket:
        // 此处代码仅为验证结果的展示示例，真实业务接入，建议基于ticket和errorCode情况做不同的
        if (res.ret === 0) {
            // 复制结果至剪切板
            var str = '【randstr】->【' + res.randstr + '】        【ticket】->【' + res
            var ipt = document.createElement('input');
            ipt.value = str;
            document.body.appendChild(ipt);
            ipt.select();
            document.execCommand("Copy");
            document.body.removeChild(ipt);
            alert('1. 返回结果（randstr、ticket）已复制到剪切板，ctrl+v 查看。\\n2. 打开浏
        }
    }

    // 定义验证码js加载错误处理函数
    function loadErrorCallback() {
      var appid = 'CaptchaAppId';
       // 生成容灾票据或自行做其它处理
      var ticket = 'terror_1001_' + appid + '_' + Math.floor(new Date().getTime() /
      callback({
        ret: 0,
        randstr: '@'+ Math.random().toString(36).substr(2),
        ticket,
        errorCode: 1001,
        errorMessage: 'jsload_error',
      });
```

```
    }

    // 定义验证码触发事件
    window.onload = function(){
        document.getElementById('CaptchaId').onclick = function(){
            try {
                // 生成一个验证码对象
                // CaptchaAppId：登录验证码控制台，从【验证管理】页面进行查看。如果未创建过
                //callback：定义的回调函数
                var captcha = new TencentCaptcha('你的验证码CaptchaAppId', callback
                // 调用方法，显示验证码
                captcha.show();
            } catch (error) {
            // 加载异常，调用验证码js加载错误处理函数
                loadErrorCallback();
            }
        }
    }
</script>

</html>
```

接入说明

步骤1：动态引入验证码 JS

Web 页面需动态引入验证码 JS，在业务需要验证时，唤起验证码进行验证。



```
<!-- 动态引入验证码JS示例 -->
<script src="https://ca.turing.captcha.qcloud.com/TCaptcha-global.js"></script>
```

注意：

必须动态引入验证码 JS。如通过其他手段规避动态加载，会导致验证码无法正常更新，对抗能力无法保证，甚至引起误拦截。

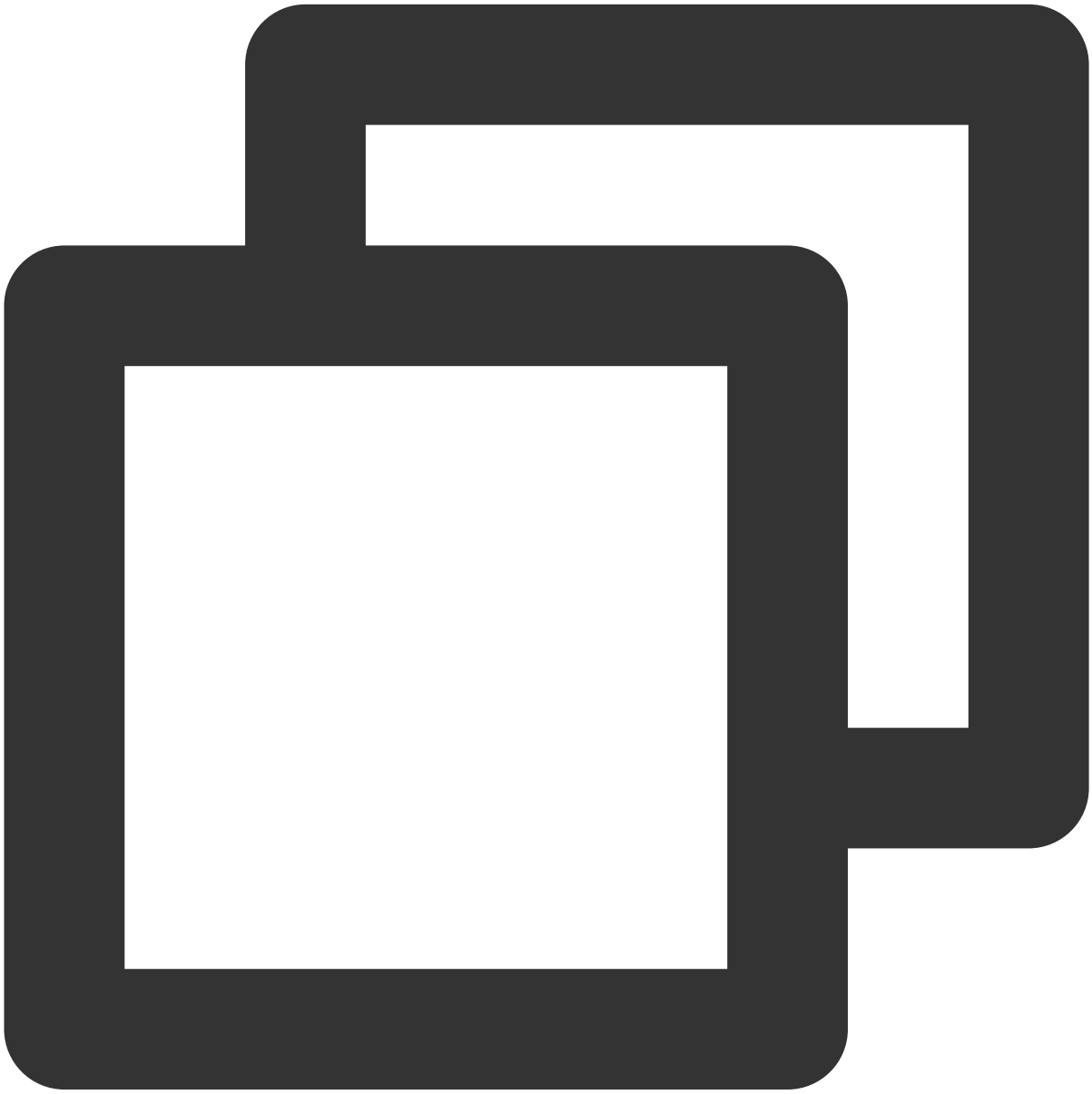
步骤2：创建验证码对象

引入验证码 JS 后，验证码会在全局注册一个 `TencentCaptcha` 类，业务方可以使用这个类自行初始化验证码，并对验证码进行显示或者隐藏。

注意：

触发验证码的元素不要使用 `id="TencentCaptcha"` ，TencentCaptcha 属于系统默认 id，不可使用。

构造函数



```
new TencentCaptcha(CaptchaAppId, callback, options);
```

参数说明

参数名	值类型	说明
CaptchaAppId	String	验证码 CaptchaAppId：登录 验证码控制台，在验证管理页面进行查看。如果未创建过验证，请先新建验证。 注意：不可使用客户端类型为小程序的 CaptchaAppId，会导致数据统计错误。
callback	Function	验证码回调函数，详情请参见 callback 回调函数。
options	Object	验证码外观配置参数, 详情请参见 options 外观配置参数。

callback 回调函数

验证结束后，会调用业务传入的回调函数，并在第一个参数中传入回调结果。回调结果字段说明如下：

字段名	值类型	说明
ret	Int	验证结果，0：验证成功。2：用户主动关闭验证码。
ticket	String	验证成功的票据，当且仅当 ret = 0 时 ticket 有值。
CaptchaAppId	String	验证码应用 ID。
bizState	Any	自定义透传参数。
randstr	String	本次验证的随机串，后续票据校验时需传递该参数。
errorCode	Number	错误 code ，详情请参见 回调函数 errorCode 说明。
errorMessage	String	错误信息。

回调函数 errorCode 说明

errorCode	说明
1001	TCaptcha-global.js加载错误
1002	调用 show 方法超时
1003	中间 js 加载超时
1004	中间 js 加载错误
1005	中间 js 运行错误
1006	拉取验证码配置错误/超时

1007	iframe 加载超时
1008	iframe 加载错误
1009	jquery 加载错误
1010	滑块 js 加载错误
1011	滑块 js 运行错误
1012	刷新连续错误3次
1013	验证网络连续错误3次

options 外观配置参数

options 参数用于对验证码进行定制外观设置，默认可以设置为空。

注意：

验证码弹窗内部不支持调整样式大小，如果需要调整，可在弹窗最外层用 `class=tcaptcha-transform` 的元素设置 `transform:scale();` 。验证码更新可能会改变元素的 id，class 等属性，请勿依赖其他验证码元素属性值覆盖样式。

如果手机原生端有设置左右滑动手势，需在调用验证码 show 方法前禁用，验证完成后再打开，防止与验证码滑动事件冲突。

配置名	值类型	说明
bizState	Any	自定义透传参数，业务可用该字段传递少量数据，该字段的内容会被带入 callback 回调的对象中。
enableDarkMode	Boolean/String	开启自适应深夜模式或强制深夜模式。 开启自适应深夜模式: {"enableDarkMode": true} 强制深夜模式: {"enableDarkMode": 'force'}
ready	Function	验证码加载完成的回调，回调参数为验证码实际的宽高： {"sdkView": { "width": number, "height": number }} 该参数仅为查看验证码宽高使用，**请勿使用此参数直接设定宽高**。
needFeedBack	String	自定义帮助链接: {"needFeedBack": 'url地址' }
loading	Boolean	是否在验证码加载过程中显示loading框。不指定该参数时，默认显示 loading 框。 显示loading框: {"loading": true} 不显示loading框: {"loading": false}
userLanguage	String	指定验证码提示文案的语言，优先级高于控制台配置。 支持传入值同 navigator.language 用户首选语言，大小写不敏感。 详情参见 userLanguage 配置参数。

userLanguage 配置参数

参数名	说明
zh-cn	简体中文
zh-hk	繁体中文（中国香港）
zh-tw	繁体中文（中国台湾）
en	英文
ar	阿拉伯语
my	缅甸语
fr	法语
de	德语
he	希伯来语
hi	印地语
id	印尼语
it	意大利语
ja	日语
ko	朝鲜语
lo	老挝语
ms	马来语
pl	波兰语
pt	葡萄牙语
ru	俄语
es	西班牙语
th	泰语
tr	土耳其语
vi	越南语

版权所有：腾讯云计算（北京）有限责任公司

步骤3：调用验证码实例方法

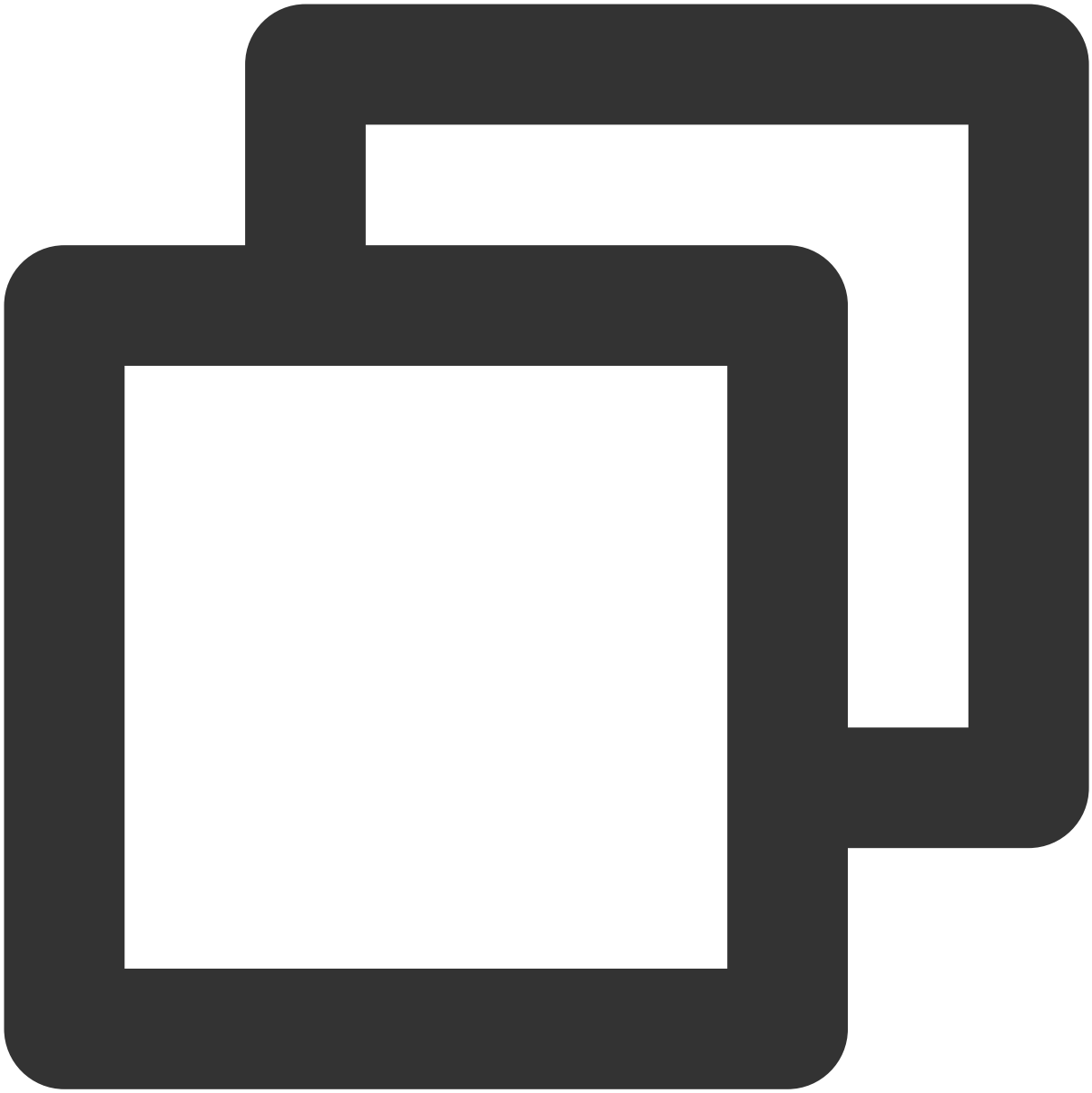
TencentCaptcha 的实例提供一些操作验证码的常用方法：

方法名	说明	传入参数	返回内容
show	显示验证码，可以反复调用。	无	无
destroy	隐藏验证码，可以反复调用。	无	无
getTicket	获取验证成功后的 ticket。	无	`Object: {"CaptchaAppId":"","ticket":""}`

步骤4：容灾处理

为保障验证码 Captacha 服务端异常时不阻塞客户网站正常业务流程，建议参考如下方式接入验证码。

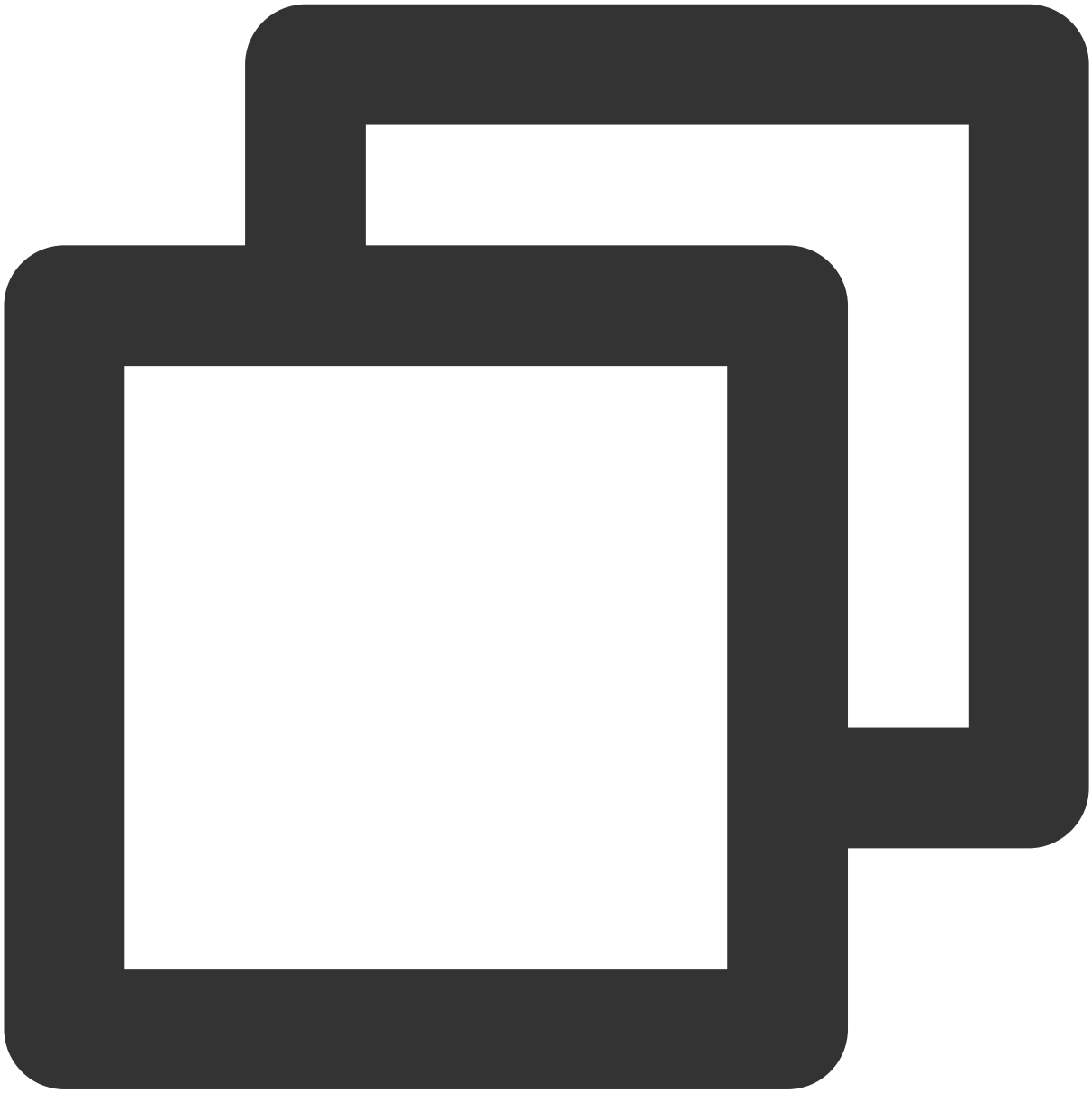
1. 定义 JS 加载错误处理函数。



```
// 错误处理函数作用：在JS加载或初始化错误时，保障事件流程正常
// 函数定义需在script加载前
function loadErrorCallback() {
 var appid = ''
 // 生成容灾票据或自行做其它处理
 var ticket = 'terror_1001_' + appid + Math.floor(new Date().getTime() / 1000);
 callback({
     ret: 0,
     randstr: '@'+ Math.random().toString(36).substr(2),
     ticket,
     errorCode: 1001,
```

```
      errorMessage: 'jsload_error',
  });
}
```

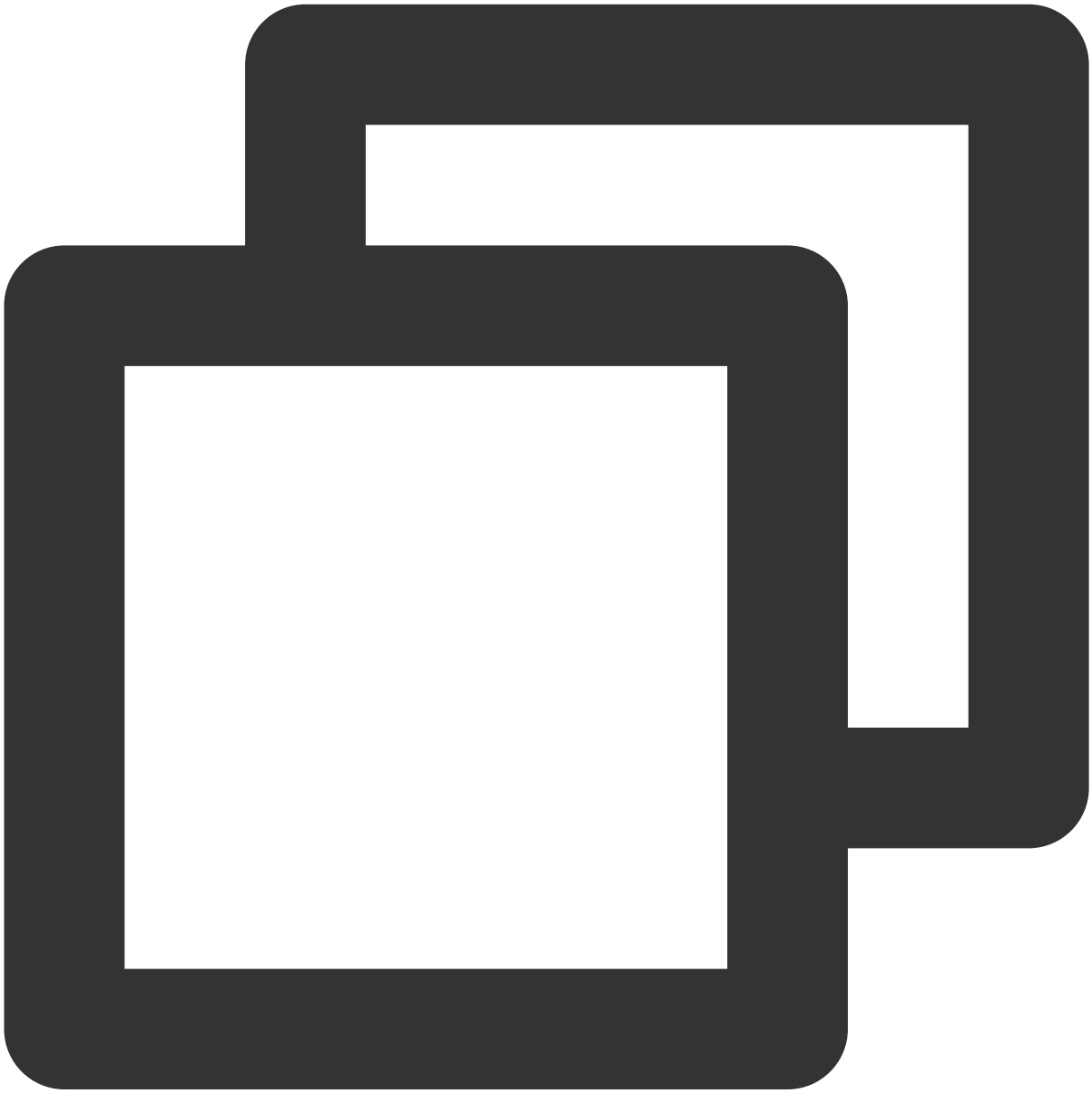
2. 调用验证码实例 catch 到错误时，调用 JS 加载错误处理函数。



```
try {
  // 生成一个验证码对象
  var captcha = new TencentCaptcha('你的验证码CaptchaAppId', callback, {});
  // 调用方法，显示验证码
  captcha.show();
} catch (error) {
```

```
  // 加载异常，调用验证码js加载错误处理函数
  loadErrorCallback();
}
```

3. 定义验证码 callback 回调函数时，根据 ticket 和 errorCode （而非 ret）的情况做处理。errorCode 的定义参考步骤2中"回调函数 errorCode 说明"。



```
function callback(res) {
  // res（用户主动关闭验证码）= {ret: 2, ticket: null}
  // res（验证成功） = {ret: 0, ticket: "String", randstr: "String"}
  // res（请求错误，返回terror_前缀的容灾票据） = {ret: 0, ticket: "String", randstr: "St
```

```
  if (res.ticket){
      //根据errorCode情况做特殊处理
      if(res.errorCode === xxxxx){
          //自定义容灾逻辑（例如跳过这次验证）
      }
    }
 }
```

完整容灾方案，详情请见业务容灾方案。

注意：

业务客户端完成验证码接入后，服务端需二次核查验证码票据结果（未接入票据校验，会导致黑产轻易伪造验证结果，失去验证码人机对抗效果），详情请参见：接入票据校验(Web及App)。

常见问题

详情参见 接入相关问题。

App 客户端接入

最近更新时间：2023-12-21 11:25:28

前提条件

客户端接入前，需完成新建验证，并在**验证列表**获取所需的 CaptchaAppId 以及 AppSecretKey。步骤如下：

1. 登录 验证码控制台，左侧导航栏选择**图形验证 > 验证管理**，进入验证管理页面。
2. 单击**新建验证**，根据业务场景需求，设置验证名称等参数。
3. 单击**确定**，完成新建验证，即可在验证列表中查看验证码 CaptchaAppId 及 AppSecretKey。

接入步骤

注意：

App 客户端（Android/iOS）当前仅支持通过 Webview 引入 H5页面进行接入。

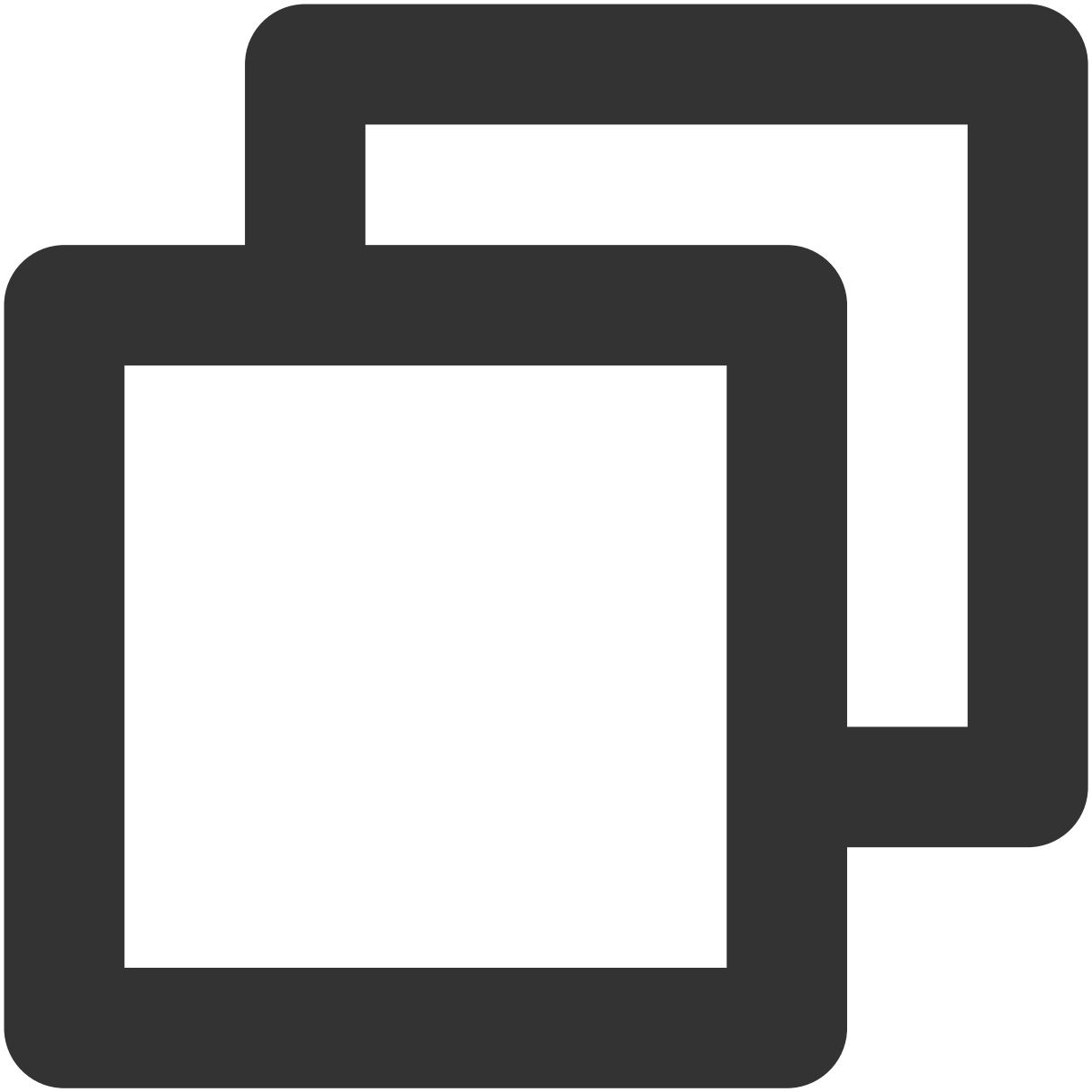
Android 接入

Android 接入主要流程

1. 在 Android 端利用 WebView 引入 H5页面。H5 页面接入验证码，详情请参见 Web 客户端接入。
2. 在 H5 页面中，通过调用验证码 JS，渲染验证页面，并将 JS 返回的参数值传到 Android App 业务端。
3. Android App 业务端把相关参数（票据 ticket、随机数等）传入业务侧后端服务进行票据验证，详情请参见 接入票据校验(Web及App)。

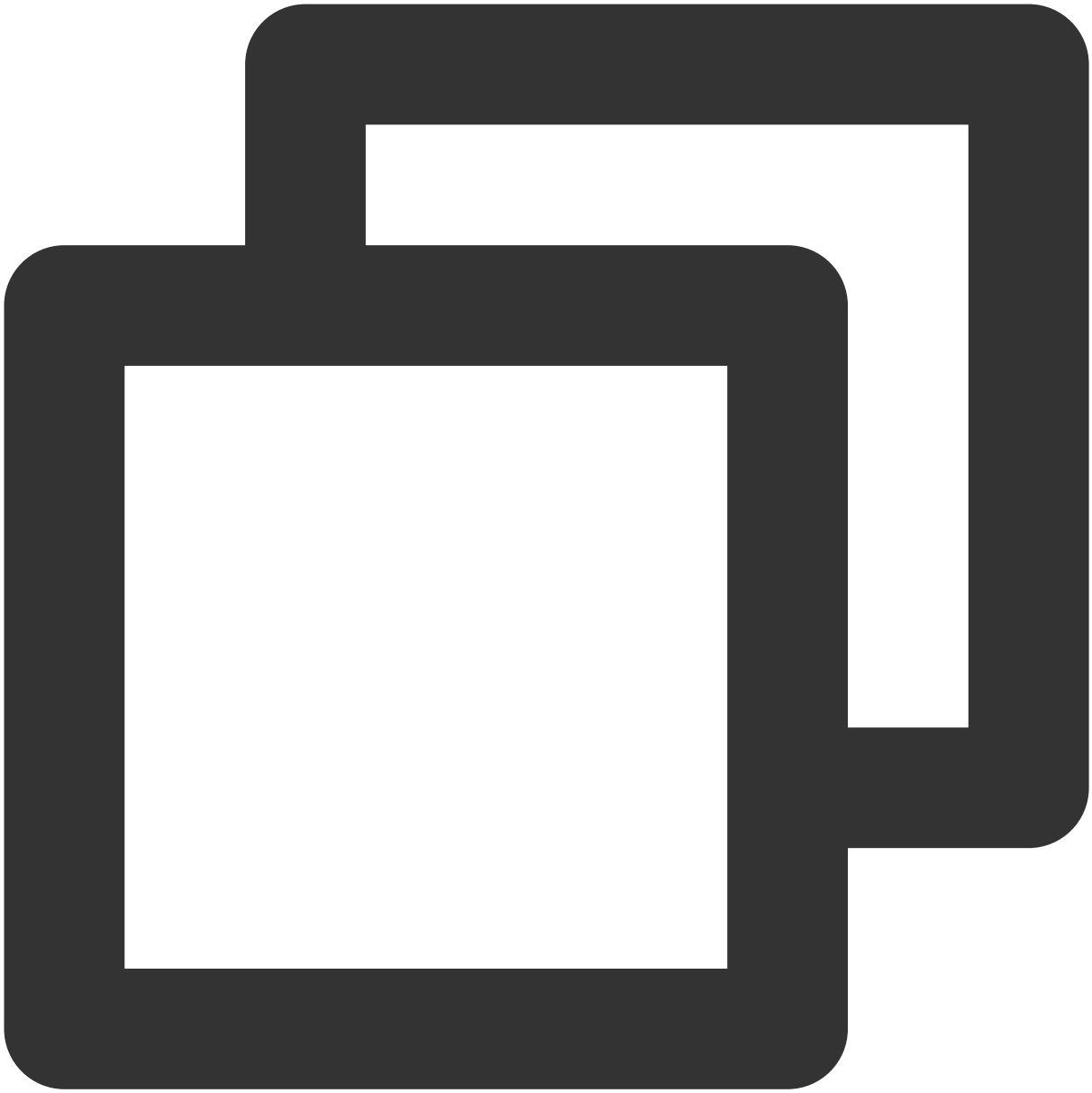
Android 接入详细步骤

1. 在项目的工程中，新建一个 Activity 并导入 WebView 组件所需的包。



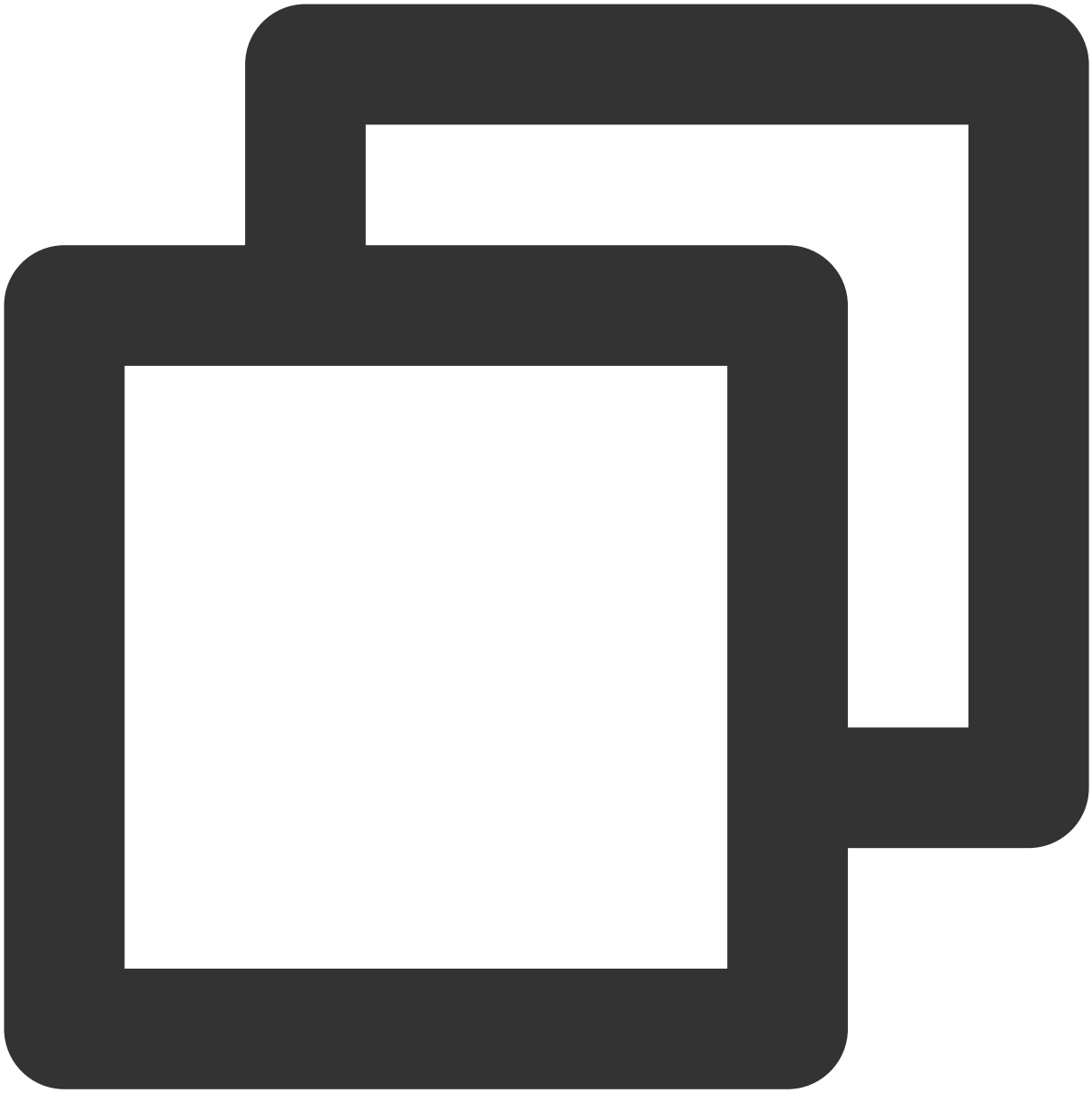
```
import android.webkit.WebView;
import android.webkit.WebSettings;
import android.webkit.WebViewClient;
import android.webkit.WebChromeClient;
```

2. 添加相关权限，如开启网络访问权限以及允许 App 进行非 HTTPS 请求等。



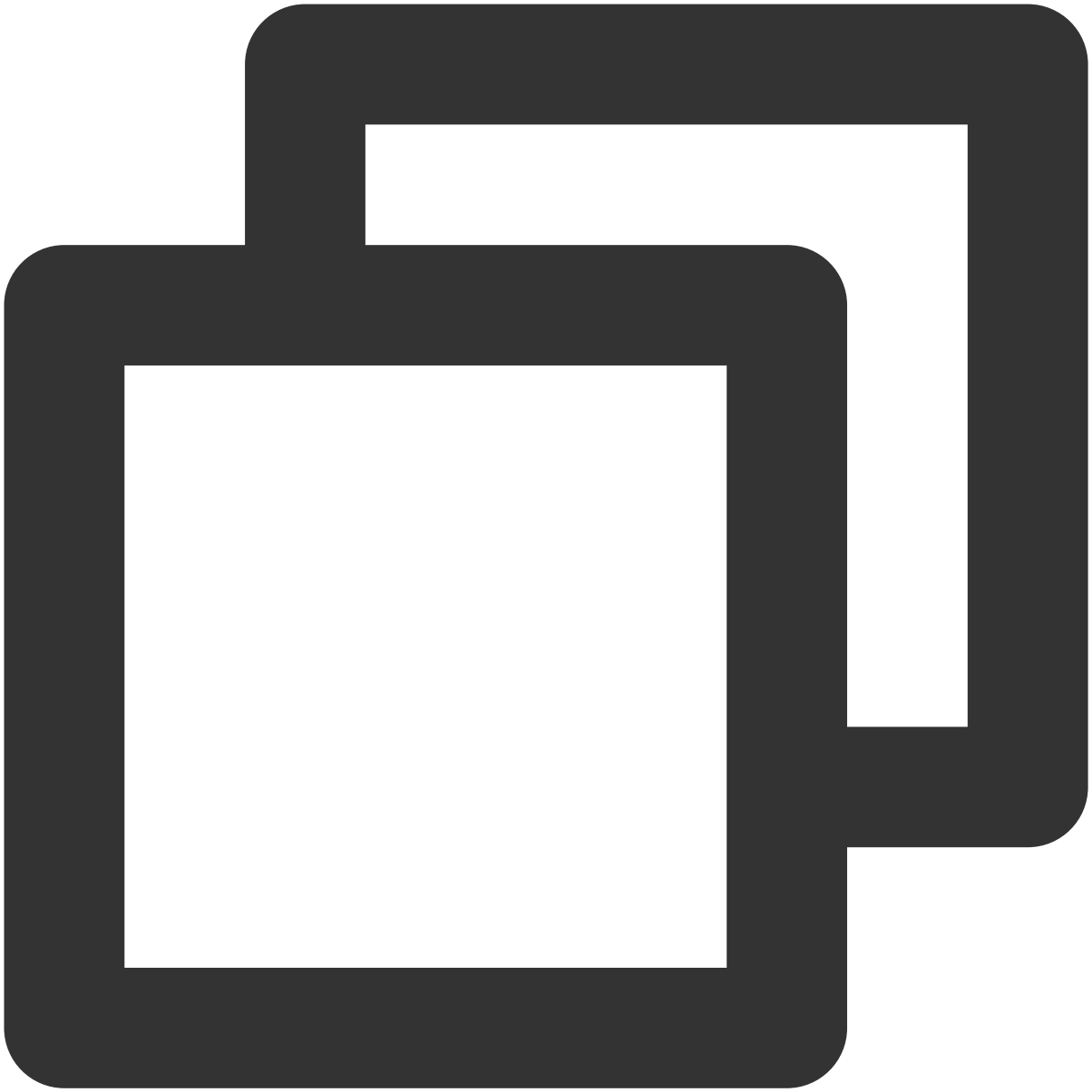
```
<uses-permission android:name="android.permission.INTERNET"/>
<application android:usesCleartextTraffic="true">...</application>
```

3. 在 Activity 的布局文件中，添加 WebView 组件。



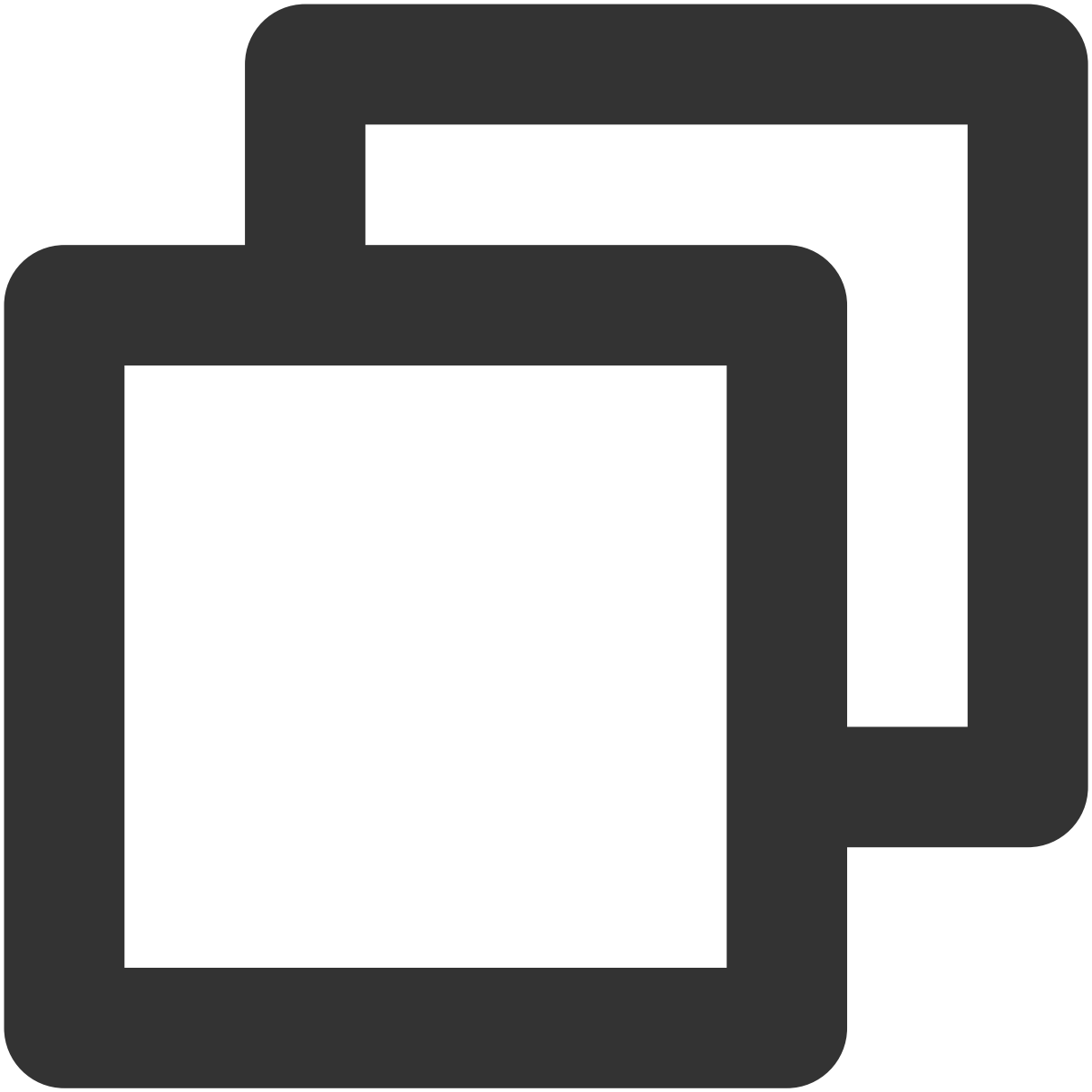
```
<WebView
android:id="@+id/webview"
android:layout_height="match_parent"
android:layout_width="match_parent"
/>
```

4. 在项目的工程中，添加自定义 JavascriptInterface 文件，并定义一个方法用来获取相关数据。



```
import android.webkit.JavascriptInterface;
public class JsBridge {
@JavascriptInterface
public void getData(String data) {
System.out.println(data);
}
}
```

5. 在 Activity 文件中，加载相关 H5 业务页面。



```
public class MainActivity extends AppCompatActivity {
private WebView webview;
private WebSettings webSettings;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
initView();
}
```

```
private void initView() {
webview = (WebView) findViewById(R.id.webview);
webSettings = webview.getSettings();
webSettings.setUseWideViewPort(true);
webSettings.setLoadWithOverviewMode(true);
// 禁用缓存
webSettings.setCacheMode(WebSettings.LOAD_NO_CACHE);
webview.setWebViewClient(new WebViewClient(){
@Override
public boolean shouldOverrideUrlLoading(WebView view, String url) {
view.loadUrl(url);
return true;
}
});
// 开启js支持
webSettings.setJavaScriptEnabled(true);
webview.addJavascriptInterface(new JsBridge(), "jsBridge");
// 也可以加载本地html(webView.loadUrl("file:///android_asset/xxx.html"))
webview.loadUrl("https://x.x.x/x/");
}
}
```

6. 在 H5 业务页面中接入验证码，详情请参见 Web 客户端接入 文档，并使用 JSBridge 传回验证数据给具体业务端。

注意：

业务客户端完成验证码接入后，服务端需二次核查验证码票据结果（未接入票据校验，会导致黑产轻易伪造验证结果，失去验证码人机对抗效果），详情参见 接入票据校验 （Web 及 App）。

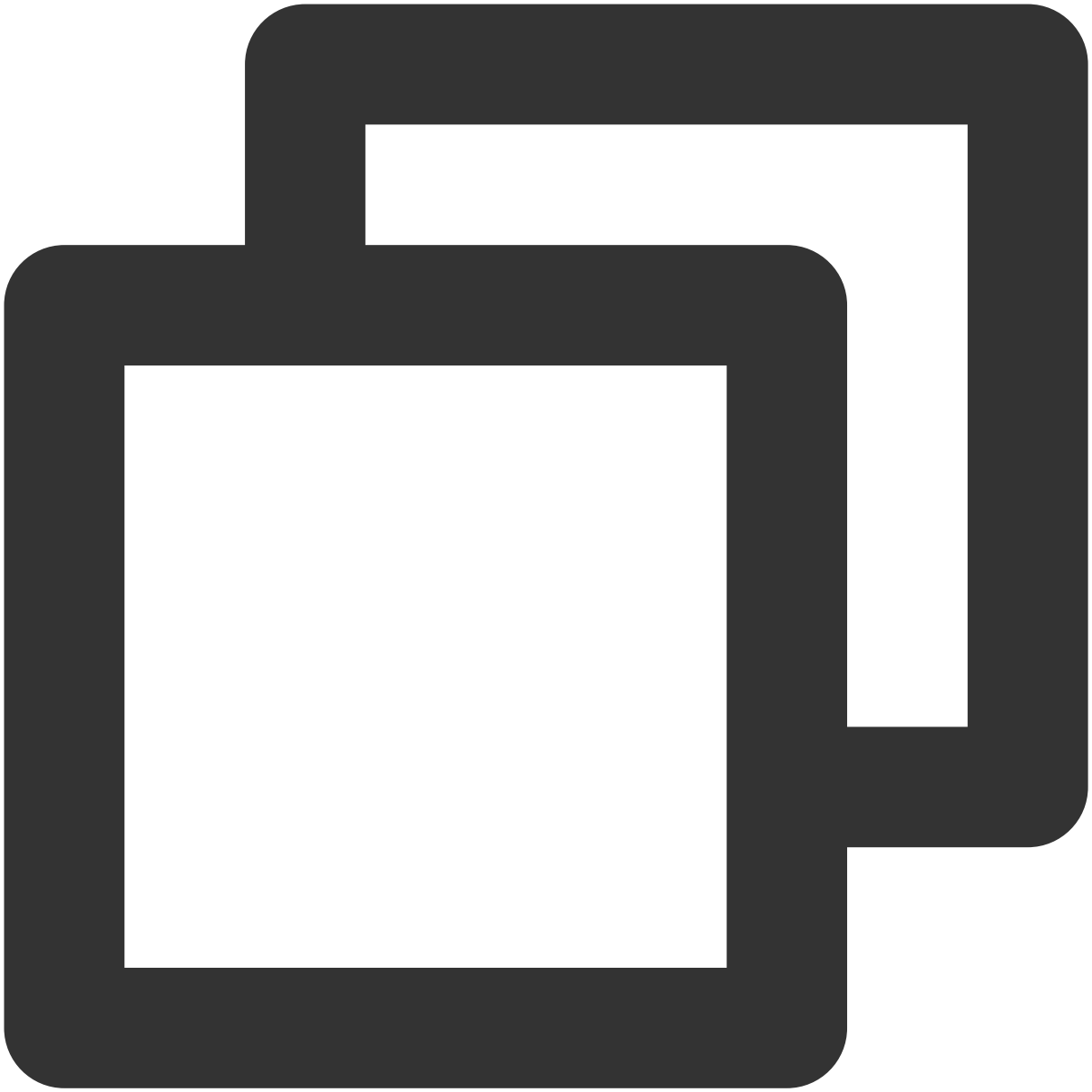
iOS 接入

iOS 接入主要流程

1. 在 iOS 中打开 WebView，通过 JSBridge 触发 HTML 页面，同时注入方法，供 HTML 调用传入验证结果。
2. 在 HTML 页面中接入验证码，详细请参见 Web 客户端接入，通过调用验证码 JS，渲染验证页面，并调用 iOS 注入的方法传入验证结果。
3. 通过 JSBridge 将验证结果返回到 iOS，并把相关参数（票据 ticket、随机数等）传入业务侧后端服务进行票据验证，详情请参见 接入票据校验(Web及App)。

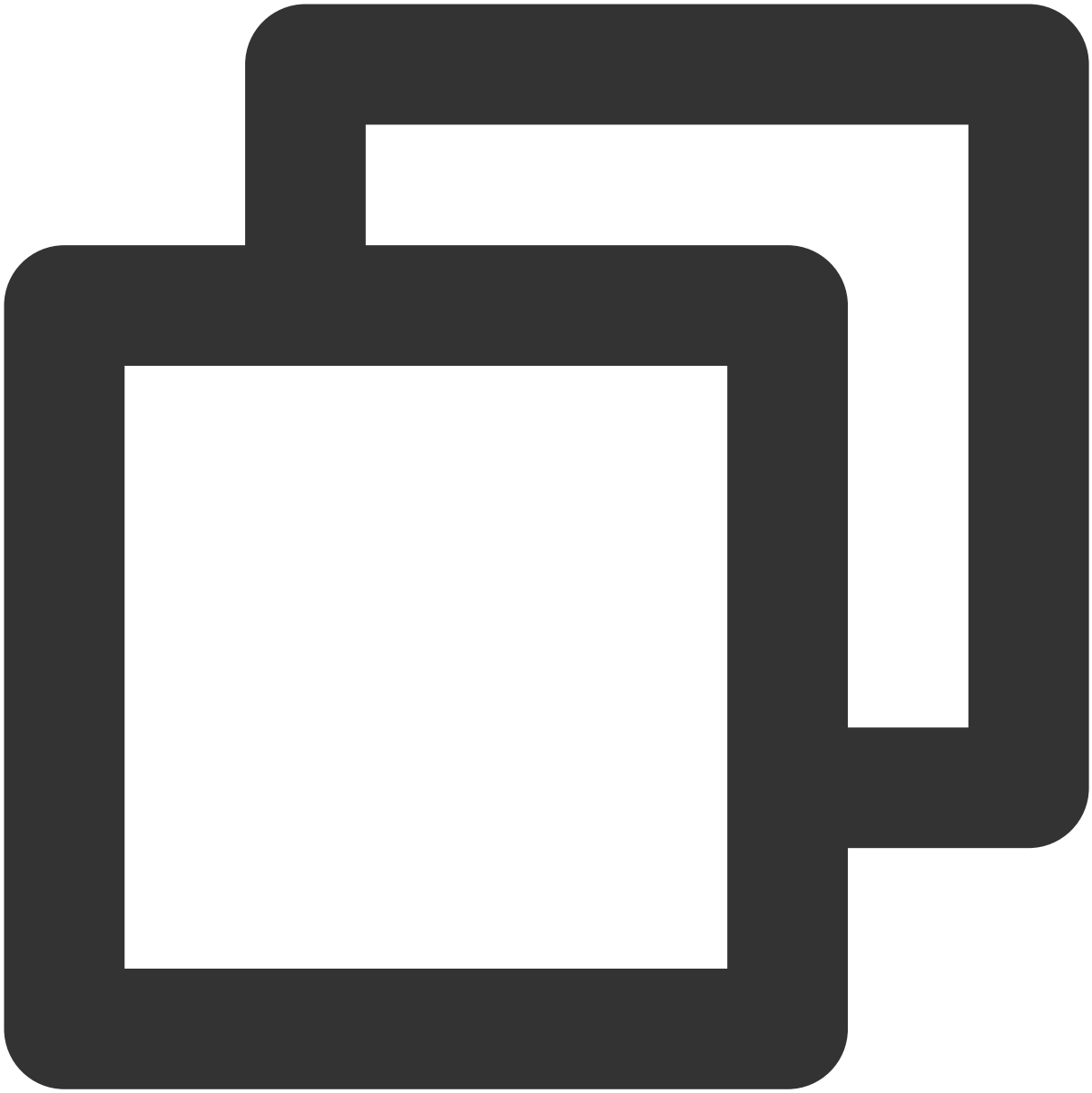
iOS 接入的详细操作步骤

1. 在控制器或 view 中导入 WebKit 库。



```
#import <WebKit/WebKit.h>
```

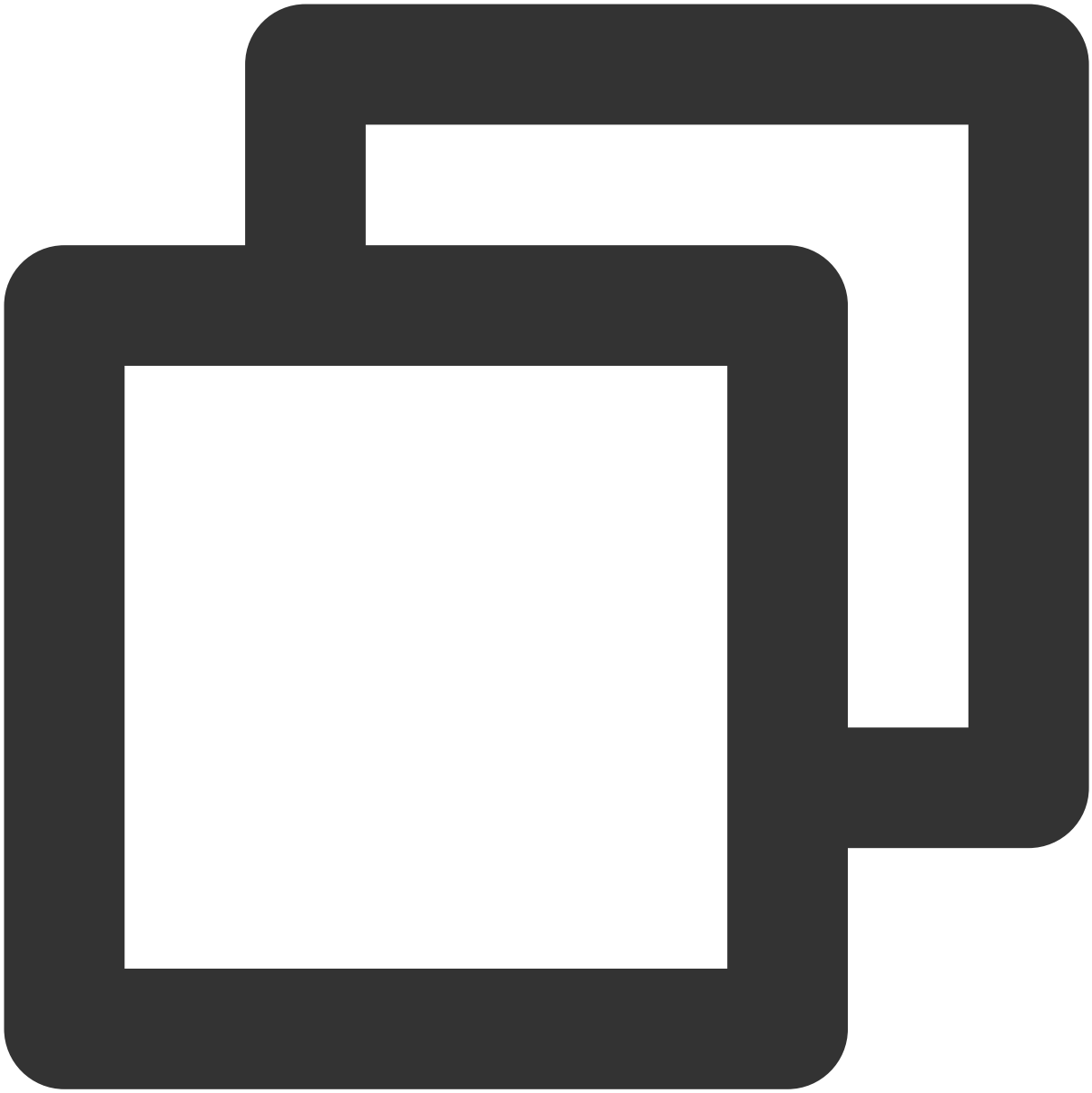
2. 创建 WebView 并渲染。



```
-(WKWebView *)webView{
if(_webView == nil){
 //创建网页配置对象
 WKWebViewConfiguration *config = [[WKWebViewConfiguration alloc] init];
 // 创建设置对象
 WKPreferences *preference = [[WKPreferences alloc]init];
 //设置是否支持 javaScript 默认是支持的
 preference.javaScriptEnabled = YES;
 // 在 iOS 上默认为 NO，表示是否允许不经过用户交互由 javaScript 自动打开窗口
 preference.javaScriptCanOpenWindowsAutomatically = YES;
 config.preferences = preference;
```

```
 //这个类主要用来做 native 与 JavaScript 的交互管理
 WKUserContentController * wkUController = [[WKUserContentController alloc] init];
 //注册一个name为jsToOcNoPrams的js方法 设置处理接收JS方法的对象
 [wkUController addScriptMessageHandler:self  name:@"jsToOcNoPrams"];
 [wkUController addScriptMessageHandler:self  name:@"jsToOcWithPrams"];
 config.userContentController = wkUController;
 _webView = [[WKWebView alloc] initWithFrame:CGRectMake(0, 0, SCREEN_WIDTH, SCREEN_
 // UI 代理
 _webView.UIDelegate = self;
 // 导航代理
 _webView.navigationDelegate = self;
 //此处即需要渲染的网页
 NSString *path = [[NSBundle mainBundle] pathForResource:@"JStoOC.html" ofType:nil]
 NSString *htmlString = [[NSString alloc]initWithContentsOfFile:path encoding:NSUTF
 [_webView loadHTMLString:htmlString baseURL:[NSURL fileURLWithPath:[[NSBundle main
}
return _webView;
}
[self.view addSubview:self.webView];
```

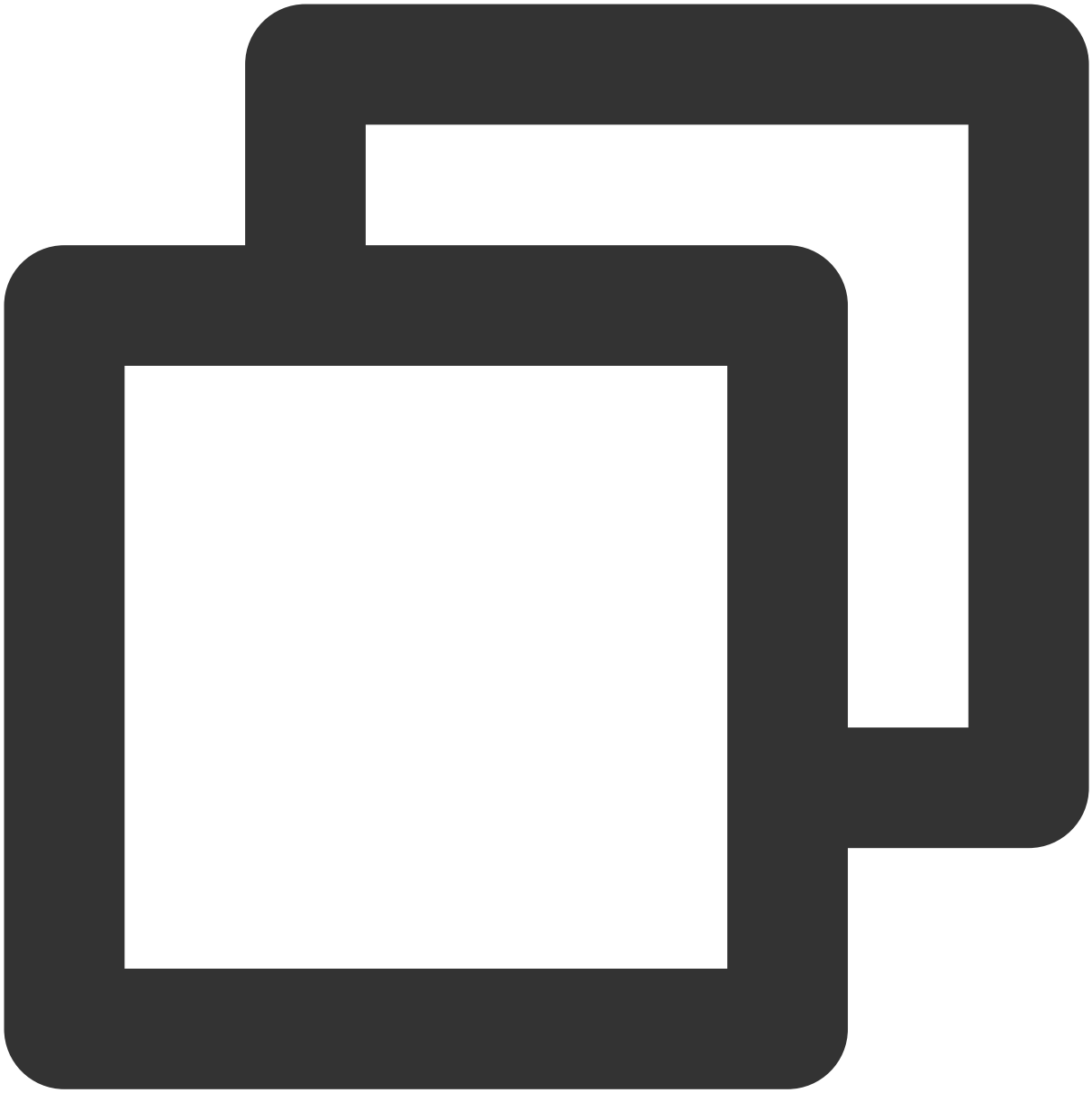
3. 代理方法，处理一些响应事件。



```
// 页面开始加载时调用
-(void)webView:(WKWebView *)webView didStartProvisionalNavigation:(WKNavigation *)n
}
// 页面加载失败时调用
-(void)webView:(WKWebView *)webView didFailProvisionalNavigation:(null_unspecified
[self.progressView setProgress:0.0f animated:NO];
}
// 当内容开始返回时调用
-(void)webView:(WKWebView *)webView didCommitNavigation:(WKNavigation *)navigation
}
// 页面加载完成之后调用
```

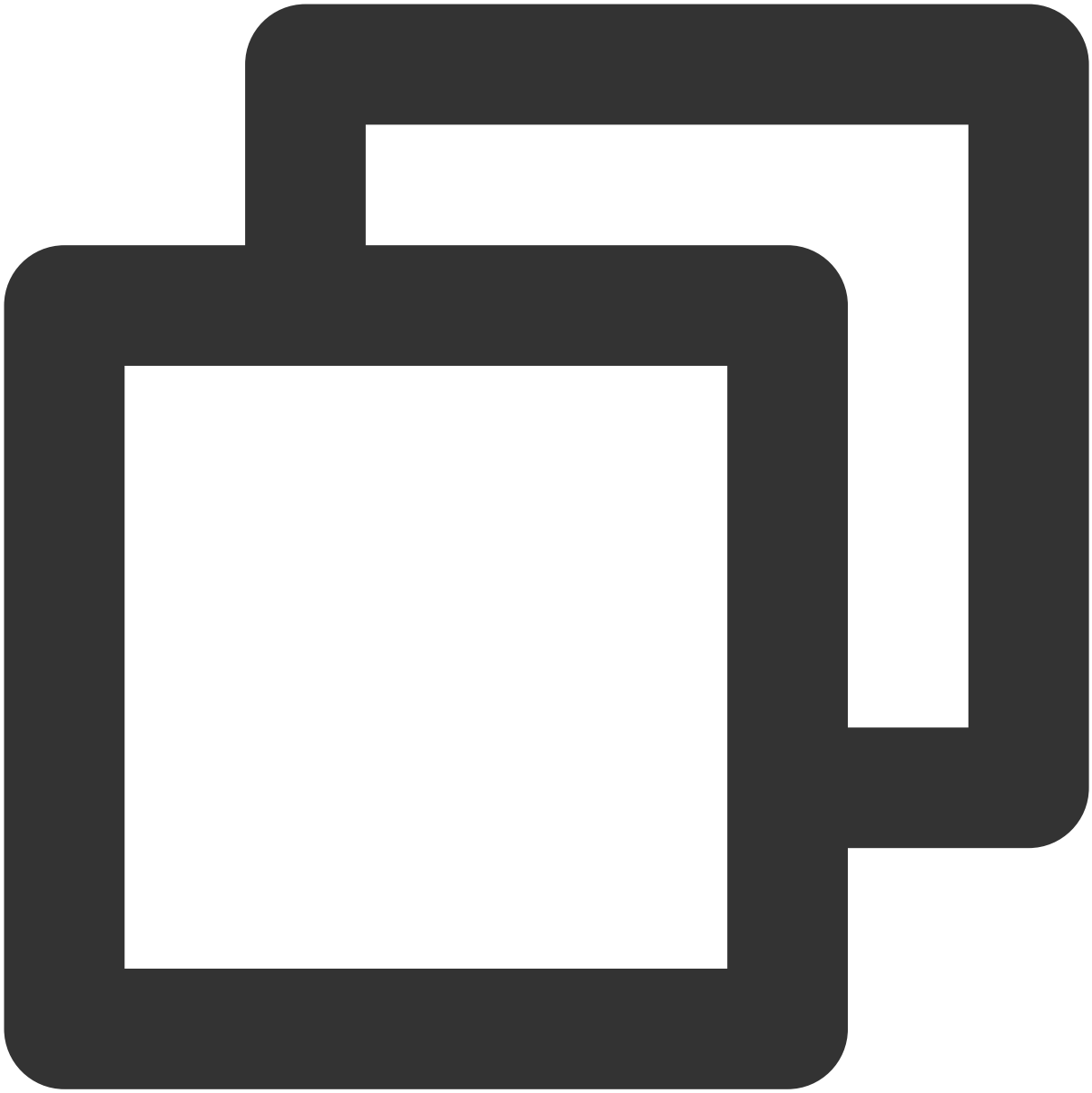
```
-(void)webView:(WKWebView *)webView didFinishNavigation:(WKNavigation *)navigation
[self getCookie];
}
//提交发生错误时调用
-(void)webView:(WKWebView *)webView didFailNavigation:(WKNavigation *)navigation wi
[self.progressView setProgress:0.0f animated:NO];
}
// 接收到服务器跳转请求即服务重定向时之后调用
-(void)webView:(WKWebView *)webView didReceiveServerRedirectForProvisionalNavigatio
}
```

4. JS 将参数传给 OC。



```
<p style="text-align:center"> <button id="btn2" type = "button" onclick = "jsToOcFu
function jsToOcFunction()
{
window.webkit.messageHandlers.jsToOcWithPrams.postMessage({"params":"res.randstr"})
}
```

5. 将渲染好的 WebView 展示在视图上，调用验证码服务，将数据传给客户端。



```
-(void)userContentController:(WKUserContentController *)userContentController didRe
//此处message.body即传给客户端的json数据
//用message.body获得JS传出的参数体
NSDictionary * parameter = message.body;
//JS调用OC
if([message.name isEqualToString:@"jsToOcWithPrams"]){
//在此处客户端得到js透传数据 并对数据进行后续操作
parameter[@"params"]
}
}
```

注意：

业务客户端完成验证码接入后，服务端需二次核查验证码票据结果（未接入票据校验，会导致黑产轻易伪造验证结果，失去验证码人机对抗效果），详情请参见 接入票据校验（Web 及 App）。

常见问题

详情参见 接入相关问题。

服务端接入
接入票据校验（Web 及 App）

最近更新时间：2024-03-01 11:20:47

服务端需调用票据校验 API，对客户端验证结果进行二次校验。

注意：

未接入票据校验，会导致黑产轻易伪造验证结果，失去验证码人机对抗效果。

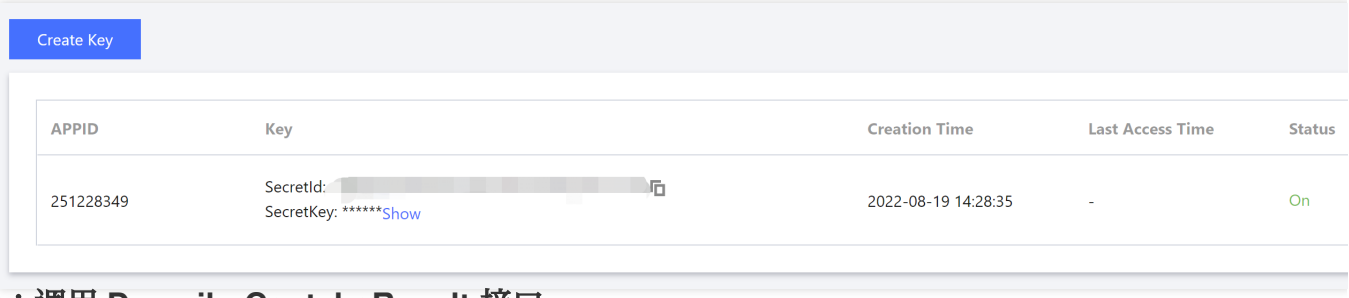
接入步骤

步骤1：安装对应语言 SDK

腾讯云提供多语言 SDK，您可以基于业务需求，安装对应语言的 SDK，详情请参见：SDK中心。

步骤2：获取云 API 密钥

1. 登录 访问管理控制台，左侧导航栏选择**访问秘钥** > **API 密钥管理**。
2. 在 API 密钥管理页面，如已创建密钥，在该页面查看即可；如未创建密钥，单击**新建密钥**，即可生成所需密钥。



步骤3：调用 DescribeCaptchaResult 接口

建议使用 API Explorer，在线生成代码。API 详情参见：核查验证码票据结果（Web 及 APP）。

常见问题

详情请参见 接入相关问题。