* [首页](https://km.woa.com/?kmref=km_header)
* [发现](https://km.woa.com/discovery?kmref=km_header)
* [悦读](https://km.woa.com/read?kmref=km_header)
* [乐问](https://km.woa.com/q?kmref=km_header)
* [直播](https://tencent.lexiangla.com/lives?company_from=tencent&kmref=km_header)
* [应用](javascript:void(0))
* [我的K吧](javascript:void(0))
* [[](https://km.woa.com/user/rickyrqzhao)](https://km.woa.com/user/rickyrqzhao)

窗体顶端



窗体底端

[](https://km.woa.com/group/scp)[社交协作产品部](https://km.woa.com/group/50729)

* [首页](https://km.woa.com/group/50729)
* [团队微博](https://km.woa.com/group/50729/twitters)
* [团队讨论](https://km.woa.com/group/50729/topics)
* [团队文章](https://km.woa.com/group/50729/articles)
* [团队文档](https://km.woa.com/group/50729/docs)
* [团队活动](https://km.woa.com/group/50729/events)
* [团队投票](https://km.woa.com/group/50729/surveys)
* [团队日历](https://km.woa.com/group/50729/calendars)
* [团队会议](https://km.woa.com/group/50729/meetings)

目录

插件是什么

插件是如何工作的

addon-sdk

sandbox

client-sdk

基础能力调用

函数服务

为什么需要函数服务

函数服务与service-sdk

invoke调用

总结

浅谈腾讯文档插件2.0架构

* [xkyhuang](https://km.woa.com/user/xkyhuang)

* 2021年09月06日 12:54

* 浏览(286)

* [收藏(14)](javascript:void(0);)

* [评论(1)](javascript:void(0);)

* [分享](javascript:void(0);)

| 导语 背景：入职接近两个月，也完成了自己第一个业务需求：docs-addon-tools，它是一个提供给开发者在本地进行开发调试文档插件能力的工具。该工具的职能是提供第三方开发者在本地开发和测试文档插件的能力。因此在开发该插件的过程中，也逐渐深入地了解插件2.0架构，本篇文章从一个新人的视角，梳理一下腾讯文档插件2.0体系。

**插件是什么**

​ 所谓插件，wiki百科的定义如下：

a **plug-in** (or **plugin**, **add-in**, **addin**, **add-on**, or **addon**) is a [software component](https://en.wikipedia.org/wiki/Software_component) that adds a specific feature to an existing [computer program](https://en.wikipedia.org/wiki/Computer_program). When a program supports plug-ins, it enables customization.[[1]](https://en.wikipedia.org/wiki/Plug-in_%28computing)#cite\_note-1)

​ 定义中有两个比较重要的点：

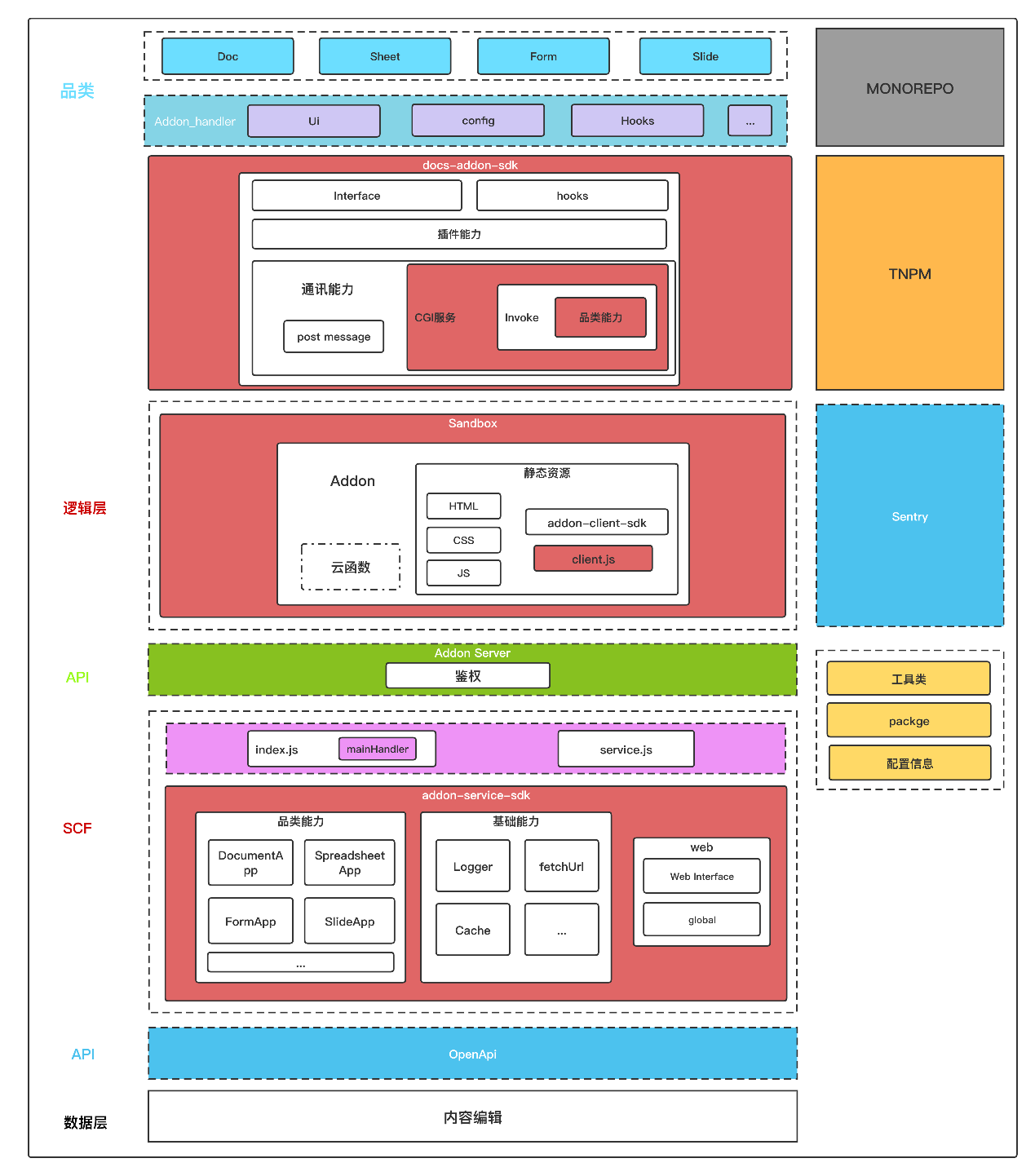
​ 第一个是adds a specific feature，插件就是为了某项功能而诞生的。其实我们知道每一个产品能提供的功能都是有限的，而用户的需求是无穷的，甚至在特定领域中（如NLP，将文字转音频），这并不是文档团队所擅长的，但又确实是某个圈子的用户所需的，如果该特定功能迟迟不满足某些用户，用户一定会去找相应的竞品。

​ 所以基于上述背景，开发团队希望只完成那些“重要”的事，而其余的一些特定能力的提供，使用一组开放API和特定入口，暴露给用户，使用户可以体验其余开发者开发的功能，开发者团队能够参与到文档的生态建设上来，在这个过程中，就满足了特定用户的需求，也即第二个特点：customization。

​ 简而言之，插件就是文档团队提供一个平台，让其他开发者能基于用户创造的内容，发挥想象力，创造出更有丰富的内容来。

​ 而我们团队在做的，就是设计一套对平台而言安全的，能满足第三方开发者使用的，对用户友好的一套插件接入以及使用体系。

​ 目前插件2.0的设计图如下：



**插件是如何工作的**

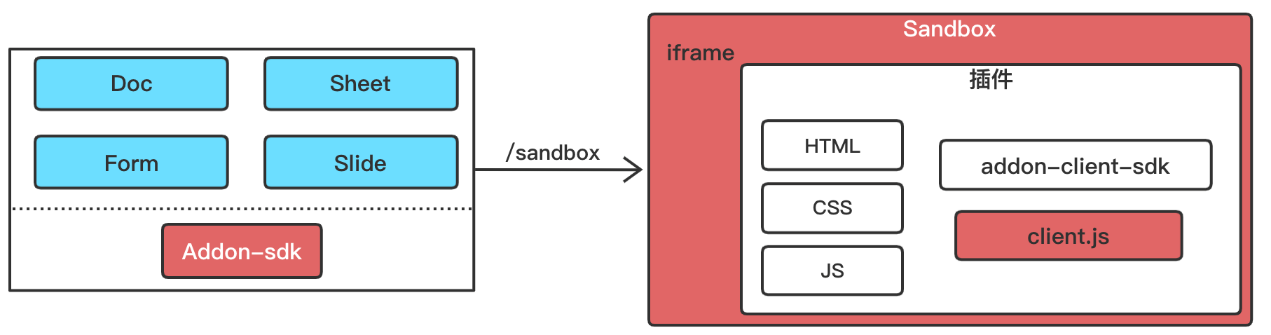
**addon-sdk**

​ 品类即文档的品种，目前开放的文档类型有四种，如上图。与品类侧直接接触的是addon-sdk，addon-sdk通过TNPM发包，由品类侧直接引入，根据addon-sdk的钩子定义，品类侧需要实现钩子，并用这一组钩子初始化sdk，将品类侧的能力传递到sdk中。addon-sdk负责将插件的业务逻辑从品类侧中解耦，帮助文档侧将与插件有关的代码收纳到addon-sdk中。更多有关addon-sdk的能力：[addon-sdk类与构造](https://iwiki.woa.com/pages/viewpage.action?pageId=819266407" \l "id-%E6%8F%92%E4%BB%B6%E5%89%8D%E7%AB%AF%E8%AE%BE%E8%AE%A1%E6%96%B9%E6%A1%88-%E4%B8%89%E3%80%81%E7%B1%BB%E4%B8%8E%E6%8E%A5%E5%8F%A3%E8%AE%BE%E8%AE%A1)

​ 此时，addon-sdk拥有了品类侧提供的能力后，其实就可以监听插件入口的按钮事件，当某个插件入口被点击时，addon-sdk就可以通过文档侧提供的钩子，新建一个iframe标签，然后通过iframe去加载插件了。可是如果直接这么做缺点也非常明显：第一，这种设计方式意味着插件和文档的HTML层级是父子关系，虽然由于跨域插件无法修改文档侧的window对象，但亮着不做隔离还是会存在一些安全问题的可能性，如[postMessage漏洞](https://www.secpulse.com/archives/56637.html)；第二，当插件体系越来越完善，需要扩展数据统计，黑名单拦截等功能时，就又需要将这些功能“塞”到addon-sdk中，addon-sdk就会越来越重了，几乎什么都管。因此，考虑到潜在的安全问题以及能更好的管理各个模块的职责我们开发了sanbox来进行处理上述问题。

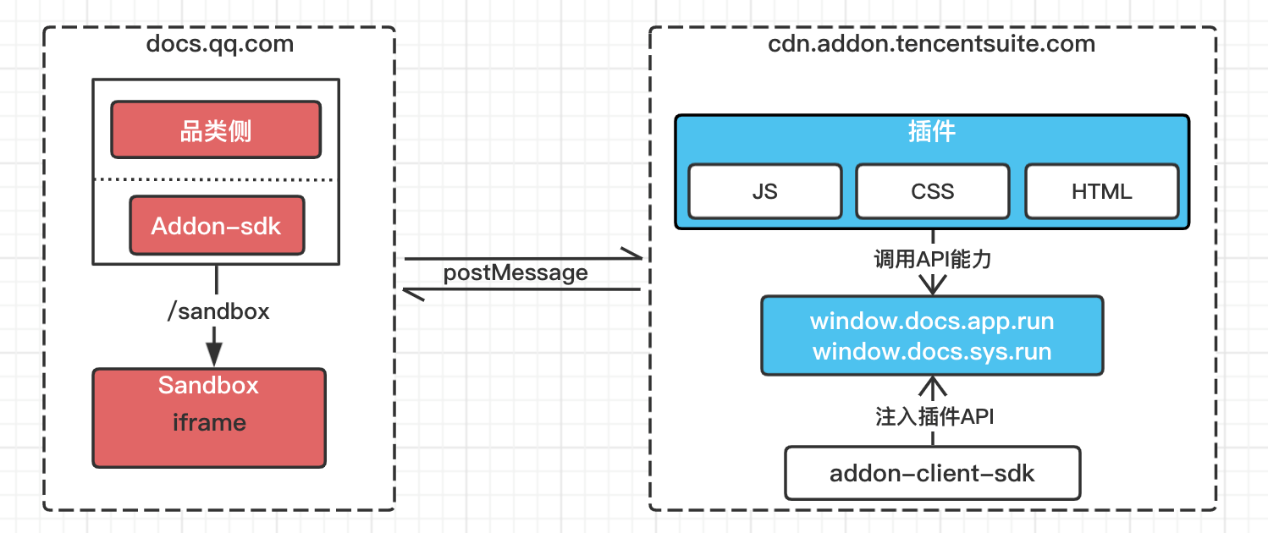
**sandbox**

​ sanbox是部署在docs.qq.com的前端页面，是与addon-sdk直接通讯的组件。sandbox主要负责通过<iframe>标签唤起插件。当品类侧监听到打开插件的事件被触发时，会通过<iframe>标签加载sanbox，同时sandbox加载完成后，会解析location.href的query中的插件url和addonId参数，使用<iframe>标签去初始化插件。因此，在HTML元素的角度上来说，插件是以<iframe>的方式运行在sandbox之上，sandbox也以<iframe>运行在品类侧的dialog或sidebar中，可参考下图。因为安全问题等种种问题，插件被部署在固定的域名上：cdn.addon.tencnetsuite.com，但同时也引入了通信无法简单的通过DOM调用或者HTTP完成。为了解决不同域的通信问题，我们选择了postMessage来解决插件与sandbox的通信。



**client-sdk**

​ 如果需要插件开发者自己处理与sandbox的通信，会苦不堪言，同时我们也需要在sanbox维护很多安全校验的逻辑，更何况，postMessage本身就不是一种完美的通信方式。我们希望当未来我们希望更换另一种通信策略时，开发者是无感的，插件是能向上兼容的。因此，我们开发了client-sdk，client-sdk的最大用武之地就是能够使插件开发者无感知地调用开放能力。我们将client-sdk部署在与插件同域的CDN上，开发者开发的插件需要通过<script>标签去引入client-sdk。在加载HTTP资源时，会先加载并执行client-sdk。client-sdk在window.docs属性下生成一系列API的声明，插件就可以直接使用这一系列API，实现无感知的调用插件开放能力。通讯的细节则是被是实现在client-sdk中，大大减少了开发者的心智负担。

​ 由于client-sdk、sandbox、addon-sdk是能够进行postMessage进行通信的，addon-sdk则可以将品类提供的基础能力提供到插件侧去调用。client-sdk在window的某个属性下注入开放能力（因为插件与client-sdk是同属一个window环境），插件能通过window.xxx去直接调用这些能力，并且通过为方法的返回值设置两个属性：成功时回调和失败时回调，插件开发者就可以往这里去注册自己的业务的回调实现了。当然，client-sdk在这里注册的方法并不是直接调用文档的基础能力，因为安全等问题，肯定不会让插件侧直接调用到品类的基础能力。因此实现其实仅仅是传递一个消息，通过postMessage从client-sdk传到sandbox，sandbox传到addon-sdk，addon-sdk完成调用后，会以同样的方式将消息传递回来，然后client-sdk判断调用成功还是失败，进而执行插件注册的回调函数。使用这种机制提供文档基础能力给插件，能在sandbox 或者 client-sdk实现各自的安全校验逻辑，且各个sdk互相的依赖也只是这一套postMessage通信机制，能更好的做到去耦合。 

**基础能力调用**

基础能力被称为系统方法，会被注入到window.docs.sys.run中。

​ 为了更好的理解melo的基础侧能力是如何被插件调用的，这里举一个onClose方法被调用的例子。当插件想要执行onClose方法时，会通过window.docs.sys.run.onClose调用 onClose API，client-sdk监听到调用后，将函数名和参数通过postMessage的方式通知到sanbox，sanbox再通过postMessage的方式传递到addon-sdk，addon-sdk会根据传过来的函数名和参数，执行品类侧注入的钩子函数onClose，插件被关闭；同样地，如果通过window.docs.sys.run.onClose调用的方法是异步，会在addon-sdk处理异步请求，请求返回后再将返回结果传到插件。

**函数服务**

**为什么需要函数服务**

​ 直到这里，我们已经解决了插件时如何调用品类基础能力了，可我们设想的插件体系会逐渐开放更多更丰富的能力给第三方开发者使用。然而我们却不可以将所有的能力都实现在addon-sdk。原因有三：

​ 第一：我们希望暴露更多的插件能力，并提供插件运行时的能力；在插件1.0体系中，我们通过提供前端沙盒去执行开发者的Js代码，但当能力增多时，沙盒就会变重，就会影响文档的速度及稳定性。

​ 第二：插件API的执行是需要上下文的，可能会需要用户当前选择的元素、fileId、cursor等，而能够直接获得这些上下文的是addon-sdk，因此也不可能将这些能力放在sandbox中，会增加通讯成本；更不可能放在不同域的clien-sdk中处理。

​ 第三：插件的全部静态资源包括HTML文件都是部署在cdn上的，插件想要访问自己的后台资源，就需要后台配置跨域，可该域名却又是全部插件共用的，就无法保证插件后台之间的安全性。

所以为了解决这两个问题，我们提供了开发云函数的能力。

**函数服务与service-sdk**

云函数（FaaS）是一段运行在云端的、轻量的、无关联的、并且可重用的代码。

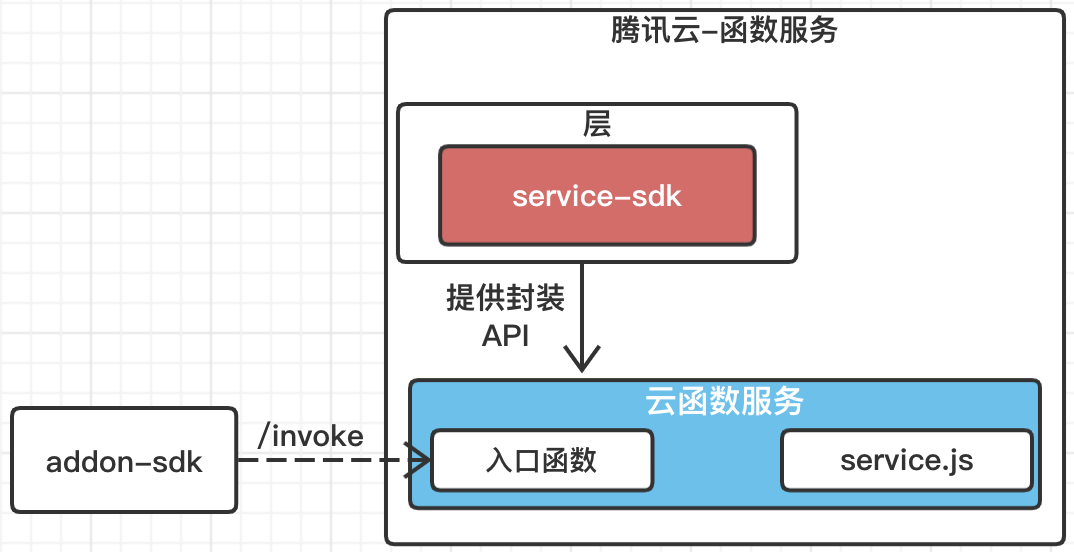
​ 借助腾讯云的能力，我们为每个插件部署了独立的函数服务，函数的运行时环境是Node，service-sdk为运行时环境中的global对象注入插件API，开发者就可以通过调用这些能力，开发自己的云函数。在发布插件时，我们将这些云函数整合到一个service.js文件中，并将该文件上传到腾讯云。因此，开发者开发的“插件”实际上包括两个部分，其一是部署在CDN上的静态资源，其二就是service.js。

​ 在腾讯云的函数服务中，我们能够指定一个index.js文件中的函数作为HTTP请求的handler，并且能够为函数服务添加层。

如果您的云函数（Faas）拥有较多的依赖库或公共代码文件，您可以使用 SCF 中的层进行管理。使用层管理，您可以将依赖放在层中而不是部署包中，可确保部署包保持较小的体积

​ 因此在函数服务中，我们将service-sdk作为每一个函数服务的层，并在函数配置中指定入口文件index.js和执行函数mainHandler中，通过配置层来为函数服务提供依赖。在入口函数中，通过require引入service-sdk（层）的能力，service-sdk便会初始化并将API挂载在global对象中。更多service-sdk的细节，指路：[service-sdk架构图](https://iwiki.woa.com/pages/viewpage.action?pageId=819266407" \l "id-%E6%8F%92%E4%BB%B6%E5%89%8D%E7%AB%AF%E8%AE%BE%E8%AE%A1%E6%96%B9%E6%A1%88-%E5%90%8E%E7%AB%AFaddon-service-sdk)。

​ 更多函数服务的特性，指路：[基本概念](https://cloud.tencent.com/document/product/583/9210)、[nodejs开发](https://cloud.tencent.com/document/product/583/11060)、[层管理](https://cloud.tencent.com/document/product/583/45760)

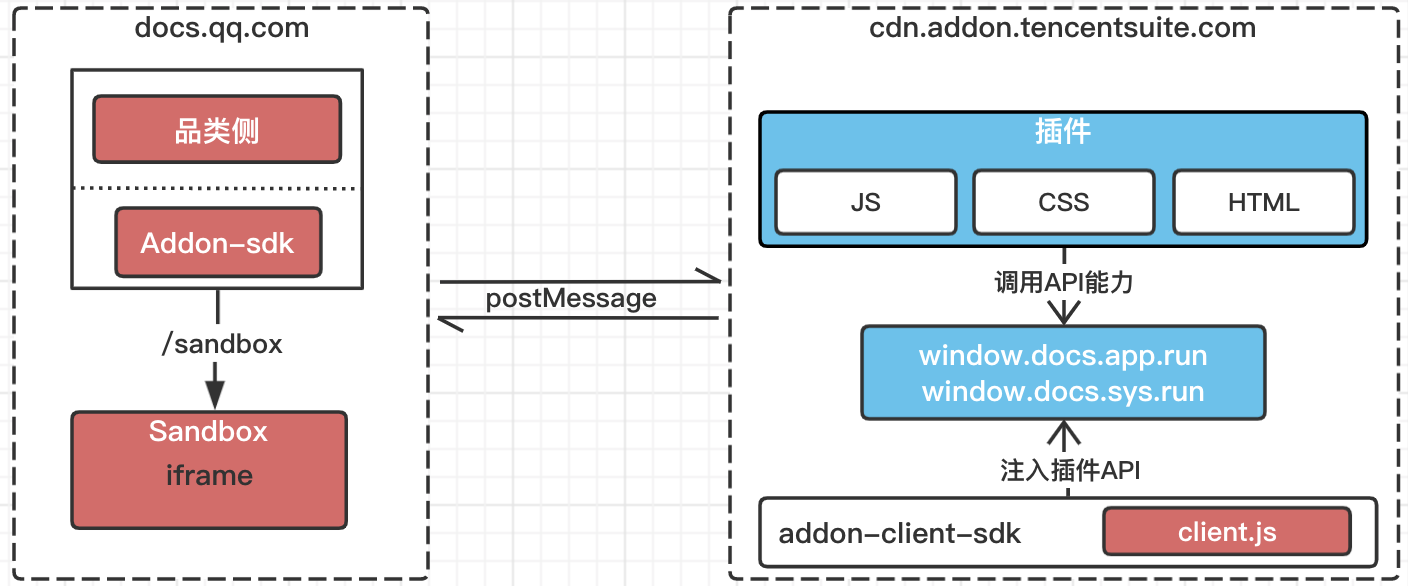


**invoke调用**

​ 我们回过头来捋一下，因为上文提到的三个限制，我们提供了云函数服务。在函数服务中，通过引入service-sdk来初始化云函数环境；当执行事件被触发时（执行事件可以理解为请求），会调用用户开发的service.js中相应的函数。可是现在还需要解决一个问题，插件的前端又是如何感知云函数的存在的呢，插件前端是通过一种什么机制去调用云函数？函数结果又是怎么返回到插件前端呢？我们将这一系列问题整理，可以用以下几个问题引导着理解云函数调用的全过程：

1. 插件前端该如何得知拥有的云函数能力，并且如何调用云函数能力？
2. 插件前端调用云函数后，如何通知云函数被执行？
3. 云函数调用与上文提到的onClose例子有何区别呢？

​ 首先插件前端如何得知拥有的云函数能力呢，起关键作用的是部署在CDN上的client-sdk和client.js(如下图)，client.js里只有一个方法getServiceFunctions，该函数返回值是关于service.js函数的声明列表。在client-sdk执行前，client.js会将getServiceFunctions挂在window下，client-sdk执行时就能先执行该函数，并将相应的函数声明挂载在window.docs.app.run上。



​ 前端执行云函数时，前半段方式与window.docs.sys.run类似，client-sdk监听到调用之后，通过postMessage发送给sanbox，sandbox再通过postMessage发送给addon-sdk，addon-sdk会发现这个方法并非品类侧注册的方法后，会向后台发送，后台完成身份信息校验和插件权限校验后（特定的插件只拥有特定的API能力），会向腾讯云函数服务发送请求。/invoke请求中会带上文档的上下文，调用函数名以及函数参数等，当/invoke请求到达入口函数mainHandler时，mainHandler首先会将service-sdk的能力挂载在global对象上，然后引入service.js，如果调用函数名为service.js中的函数，则会将该函数执行，然后将结果返回到addon-sdk，完成/invoke过程，addon-sdk拿到异步调用的结果后，通过postMessage通知sandbox，sandbox通知client-sdk执行相应的回调函数。

​ 从上一段可以看出来，其实整个链路从插件到addon-sdk是没有区别的，对于window.docs.sys,run域下的函数，如open、close、insertElement、updateElement等基础能力，都属于品类侧实现的钩子函数，是文档的基础能力，直接由品类侧提供；而对于一些获取登陆态，配置插件子菜单或插件Toolbar等能力，是插件独有的逻辑，这些能力都被抽离到service-sdk，service-sdk封装成API，提供给service.js调用。所以插件体系实际上是由两部分去提供的：第一部分是基础能力docs.sys.run，主要由品类侧提供的函数实现，在addon-sdk中执行；另一类是比较“插件向”的API（docs.app.run），如addonSubItem, getAccessToken等方法。将插件的开放能力分成两部分去提供，能有效避免插件体系和品类侧核心代码过耦合，还能最大程度的降低插件体系的发展壮大的过程中对品类侧的带来负面的影响。

**总结**

​ 梳理完文档插件2.0架构，我们来总结一下上文介绍到的sdk的作用：

**addon-sdk**：

1. 提供插件函数的类型定义
2. 获取插件列表以及发起invoke请求
3. postMessage的起点或终端，处理插件API调用的地方
4. 处理开发者在本地调试时的代理

**sandbox：**

1. 唤起插件
2. 处理消息中转和校验

**client-sdk：**

1. 将基础能力和用户特定注入到window.docs.{app|sys}属性中，供插件调用
2. 通过postMessage传递调用信息
3. 处理window.docs.{app|sys}的回调

**service-sdk：**

1. 将插件API封装，提供运行时环境
2. 提供函数入口，处理/invoke调用

​ 本文是从新人视角出发，比较直白地整理了一下插件体系的组成部分和一些重要的调用流程。因此本文难免会有思考过于片面或者错误的地方，大家可以在评论区指出，可以一起思考学习～

最后更新于 2021-09-07 23:37

本文申报

如果觉得我的文章对您有用，请随意赞赏

赏

[3人已赞赏](javascript:void();)

[[](https://km.woa.com/user/totang)](https://km.woa.com/user/totang)[[](https://km.woa.com/user/gamyhuang)](https://km.woa.com/user/gamyhuang)[](https://km.woa.com/user/mylochen)

仅供内部学习与交流，未经公司授权切勿外传

标签：[前端工程...](https://km.woa.com/group/scp/articles?tag_id=100516" \o "前端工程化" \t "_blank)(1) [腾讯文档](https://km.woa.com/group/scp/articles?tag_id=152048)(2) [插件架构](https://km.woa.com/group/scp/articles?tag_id=129657)(1)



本文专属二维码，扫一扫还能分享朋友圈

想要微信公众号推广本文章？[点击获取链接](javascript:void(0);)

我顶 (15)

收藏 (14)

* [转载](javascript:void(0);)

* [收录](javascript:void(0);)
* [评论](javascript:void(0);)(1)
* [反馈](javascript:void(0);)

分享到

大家评论

[](https://km.woa.com/user/laozhang)

[laozhang](https://km.woa.com/user/laozhang)

2021-09-06 14:31:22

q啊

 顶 [回复](javascript:void(0);)

[](https://km.woa.com/user/rickyrqzhao)



[切换到更多功能](javascript:void(0);)

关于作者

[](https://km.woa.com/user/xkyhuang)

[xkyhuang(黄一帆)](https://km.woa.com/user/xkyhuang)

PCG\社交协作产品部\平台开发一组员工

作者文章

* [浅谈腾讯文档插件2.0架构](https://km.woa.com/posts/show/520743?kmref=author_post)

Copyright©1998-2021 Tencent Inc. All Rights Reserved

腾讯公司研发管理部 版权所有

[广告申请](https://km.woa.com/chartlets/km/) [反馈问题](javascript:void(0);)

[971/998/269 ms]