(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 109240697 A
(43)申请公布日 2019.01.18

(21)申请号 201710368702.9

(22)申请日 2017.05.22

(71)申请人 腾讯科技(深圳)有限公司
地址 518000 广东省深圳市南山区高新区科技中一路腾讯大厦35层

(72)发明人 张庆吉

(74)专利代理机构 北京派特恩知识产权代理有限公司 11270
代理人 张振伟 张颖玲

(51)Int.Cl.
G06F 8/41(2018.01)
G06F 9/445(2018.01)
G06F 9/48(2006.01)

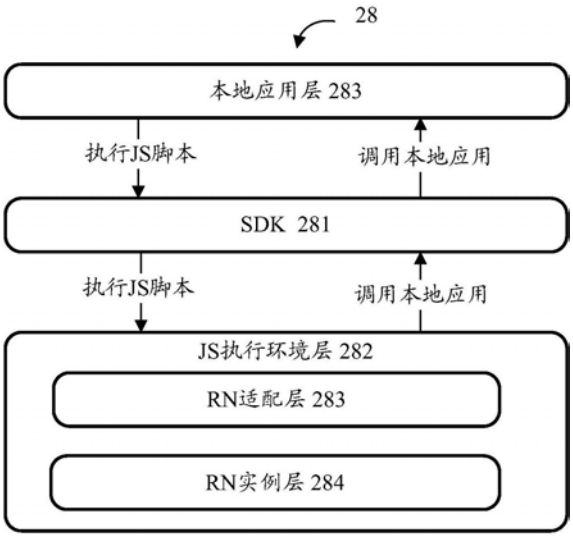
权利要求书3页 说明书10页 附图11页

(54)发明名称

调用处理方法及装置、存储介质

(57)摘要

本发明公开了一种调用处理装置、方法及存储介质;调用处理装置包括:本地应用层,用于将本地应用需要执行的脚本,提交到软件开发套件层,并将软件开发套件层返回的脚本的执行结果返回本地应用;软件开发套件层,用于提供对本地应用的接入,将接入的所述本地应用提交的脚本提交到脚本执行环境层,并将执行环境层返回的脚本的执行结果返回本地应用层;脚本执行环境层,用于通过运行本地交互框架的实例的方式,执行本地应用提交的脚本,并通过向脚本提供回调所述本地应用的方式,将所述脚本的执行结果返回软件开发套件层。实施本发明,能够集约的方式实现脚本与本地应用的通信。



CN 109240697 A

1.一种调用处理装置,其特征在于,包括:

本地应用层,用于将本地应用需要执行的脚本,提交到软件开发套件层,并将所述软件开发套件层返回的脚本的执行结果返回所述本地应用;

软件开发套件层,用于提供对所述本地应用的接入,将接入的所述本地应用提交的脚本提交到所述脚本执行环境层,并将所述执行环境层返回的所述脚本的执行结果返回所述本地应用层;

所述脚本执行环境层,用于通过运行本地交互框架的实例的方式,执行所述本地应用提交的脚本,并通过向所述脚本提供回调所述本地应用的方式,将所述脚本的执行结果返回所述软件开发套件层。

2.如权利要求1所述的调用处理装置,其特征在于,

所述软件开发套件层,具体用于向所述本地应用提供所述脚本执行环境层的接口,通过所述接口与所述本地应用进行数据交互,以及接收所述本地应用注入对象;

所述软件开发套件层,具体用于当所述本地应用接入所述软件开发套件层时,下载所述本地交互框架的代码,所述下载的代码用于创建所述本地交互框架的实例。

3.如权利要求1所述的调用处理装置,其特征在于,

所述脚本执行环境层,包括:

本地交互适配层,用于适配所述软件开发套件层与本地交互实例层进行数据交互;

所述本地交互实例层,用于基于所述本地交互框架的实例,运行所述本地应用提交的脚本。

4.如权利要求3所述的调用处理装置,其特征在于,

所述本地交互适配层,包括:

交互引擎,用于基于所述软件开发套件层下载的所述本地交互框架的代码,创建所述本地交互框架的实例;

脚本模块,用于针对所述本地应用注册到所述本地交互实例层,并执行所述本地应用提交的脚本;

本地模块,用于针对所述本地应用注册到所述本地交互实例层,执行所述本地应用提交的脚本所回调的本地方法,以回调的方式向所述脚本返回所述脚本的执行结果。

5.如权利要求4所述的调用处理装置,其特征在于,

所述本地模块,具体用于将所述本地应用提交的脚本转换为脚本模块标识、脚本方法标识和参数的形式,传递给所述脚本模块;

所述脚本模块,具体用于根据所述本地模块传递的脚本模块标识、脚本方法标识和参数查找对应的脚本并执行。

6.如权利要求4所述的调用处理装置,其特征在于,

所述脚本模块,还用于将所述脚本需要调用的本地方法转换为本地模块标识、本地方法标识和参数的形式,传递给所述本地模块;

所述本地模块,还用于根据本地模块标识、本地方法标识和参数查找对应的本地方法并执行。

7.一种调用处理方法,其特征在于,包括:

软件开发套件层提供对本地应用层的本地应用的接入;

所述本地应用层将接入的所述本地应用需要执行的脚本，提交到所述软件开发套件层；

所述软件开发套件层将所述脚本提交到脚本执行环境层；

所述脚本执行环境层通过运行本地交互框架的实例的方式，执行所述本地应用提交的脚本，并向所述脚本提供回调所述本地应用的方法，将所述脚本的执行结果返回所述软件开发套件层；

所述软件开发套件层向所述本地应用层返回的所述脚本的执行结果，所述本地应用层将所述脚本的执行结果返回所述本地应用。

8.如权利要求7所述的调用处理方法，其特征在于，所述软件开发套件层提供对本地应用层的本地应用的接入，包括：

所述软件开发套件层向所述本地应用提供所述脚本执行环境层的接口，通过所述接口与所述本地应用进行数据交互，以及接收所述本地应用注入的对象；

当所述本地应用接入所述软件开发套件层时，下载所述本地交互框架的代码，所述下载的代码用于创建所述本地交互框架的实例。

9.如权利要求7所述的调用处理方法，其特征在于，所述脚本执行环境层通过运行本地交互框架的实例的方式，包括：

所述脚本执行环境层的本地交互适配层适配所述软件开发套件层、与所述脚本执行环境层的本地交互实例层进行数据交互；

所述本地交互实例层运行所述本地应用提交的所述脚本。

10.如权利要求9所述的调用处理方法，其特征在于，还包括：

所述本地交互适配层的交互引擎，基于所述软件开发套件层下载的所述本地交互框架的代码，创建所述本地交互框架的实例；

所述本地交互适配层的脚本模块针对所述本地应用注册到所述本地交互实例层，执行所述本地应用提交的脚本；

所述本地交互适配层的本地模块针对所述本地应用注册到所述本地交互实例层，执行所述本地应用提交的脚本所回调的本地方法，以回调的方式向所述脚本返回所述脚本的执行结果。

11.如权利要求10所述的调用处理方法，其特征在于，所述执行所述本地应用提交的脚本，包括：

所述本地模块将所述本地应用提交的脚本转换为脚本模块标识、脚本方法标识和参数的形式，传递给所述脚本模块；

所述脚本模块根据所述本地模块传递的脚本模块标识、脚本方法标识和参数查找对应的脚本并执行。

12.如权利要求7所述的调用处理方法，其特征在于，还包括：

所述脚本模块将所述脚本需要调用的本地方法转换为本地模块标识、本地方法标识和参数的形式，传递给所述本地模块；

所述本地模块根据本地模块标识、本地方法标识和参数查找对应的本地方法并执行。

13.一种调用处理装置，包括：

存储器，配置为存储可执行程序；

处理器，配置为通过执行所述存储器中存储的可执行程序时，实现权利要求7至12任一项所述的调用处理方法。

14.一种存储介质，存储有可执行程序，所述可执行程序被处理器执行时，实现权利要求7至12任一项所述的调用处理方法。

调用处理方法及装置、存储介质

技术领域

[0001] 本发明涉及通信技术，尤其涉及一种调用处理方法及装置、存储介质。

背景技术

[0002] JavaScript(简称JS)一种直译式脚本语言，普遍应用在客户端如移动App的开发，JS脚本不需要进行预编译，可以直接在客户端的解释器(称为JavaScript引擎)解释执行，为客户端实现各种功能。

[0003] 以客户端为浏览器举例，JS脚本可以直接嵌入在HTML页面中来实现自身的功能，当然，也可以作为独立的JS文件存在，实现页面结构和行为的分离，JS脚本能够通过控制超文本标记语言(HTML，Hyper Text Markup Language)页面(下文中简称为页面)内的行为而实现各种的功能，典型地，在HTML页面中JS脚本实现的功能包括：对浏览器事件做出响应；读写HTML元素；将用户提交的表单数据提交到服务器之前进行验证；检测访客的浏览器信息；控制cookie，包括创建和修改cookie。

[0004] 实际应用中，还需要实现JS脚本与本地(Native)应用通信的功能，例如，在页面中点击电话号码呼出拨号程序，点击网页中的笑话通过短信程序自动发送短信，需要由页面中的JS脚本调用拨号/发送短信的本地方法来实现。

[0005] 为了实现JS脚本与本地(Native)应用的通信，相关技术提供创建网络视图(Webview)组件的实例，并通过Webview组件的实例来实现JS方法与本地方法的相互调用的方案，然而，Webview组件的基本功能是加载页面元素并渲染页面，实际应用中存在只需要本地应用与JS脚本通信而不显示HTML页面的情况，当只需要实现JS方法与本地方法的相互调用而不需要显示HTML页面时，使用Webview组件会造成不必要的资源开销。

发明内容

[0006] 本发明实施例提供一种调用处理装置、方法及存储介质，能够集约的方式实现脚本与本地应用的通信。

[0007] 本发明实施例的技术方案是这样实现的：

[0008] 第一方面，本发明实施例提供一种调用处理装置，包括：

[0009] 本地应用层，用于将本地应用需要执行的脚本，提交到软件开发套件层，并将所述软件开发套件层返回的脚本的执行结果返回所述本地应用；

[0010] 软件开发套件层，用于提供对所述本地应用的接入，将接入的所述本地应用提交的脚本提交到所述脚本执行环境层，并将所述执行环境层返回的所述脚本的执行结果返回所述本地应用层；

[0011] 所述脚本执行环境层，用于通过运行本地交互框架的实例的方式，执行所述本地应用提交的脚本，并通过向所述脚本提供回调所述本地应用的方式，将所述脚本的执行结果返回所述软件开发套件层。

[0012] 第二方面，本发明实施例提供一种调用处理方法，包括：

[0013] 软件开发套件层提供对本地应用层的本地应用的接入；

[0014] 所述本地应用层将接入所述软件开发套件的本地应用需要执行的脚本，提交到所述软件开发套件层；

[0015] 所述软件开发套件层将所述脚本提交到脚本执行环境层；

[0016] 所述脚本执行环境层通过运行本地交互框架的实例的方式，执行所述本地应用提交的脚本，并向所述脚本提供回调所述本地应用的方法，将所述脚本的执行结果返回所述软件开发套件层；

[0017] 所述软件开发套件层向所述本地应用层返回的所述脚本的执行结果，所述本地应用层将所述脚本的执行结果返回所述本地应用。

[0018] 第三方面，本发明实施例提供一种调用处理装置，包括：

[0019] 存储器，配置为存储可执行程序；

[0020] 处理器，配置为通过执行所述存储器中存储的可执行程序时，实现本发明实施例提供的调用处理方法。

[0021] 第四方面，本发明实施例提供一种存储介质，存储有可执行程序，所述可执行程序被处理器执行时，实现本发明实施例提供的调用处理方法。

[0022] 本发明实施例具有以下有益效果：

[0023] 1)以软件开发套件作为本地应用与本地交互框架的中介，对于本地应用而言，只需要面向软件开发套件通信，避免了本地交互框架的代码升级等情况下需要调整适配的情况，提升脚本的执行效率；

[0024] 2)基于软件开发套件为本地应用提供接入，软件开发套件作为本地应用与本地交互框架的实例的中介，并基于本地交互框架实现本地应用与脚本的调用，从而，可以基于本地交互框架自有的解释器解释并执行脚本，对于本地应用调用JS脚本的情况而言，不再需要Webview组件即可实现执行JS脚本，避免相关技术中本地应用执行JS脚本时必须创建Webview导致资源开销大的问题。

附图说明

[0025] 图1是本发明实施例提供的JS脚本与本地应用相互调用的一个可选的示意图；

[0026] 图2是本发明实施例提供的调用处理装置的一个可选的硬件结构示意图；

[0027] 图3-1是如图2所示的调用处理装置28的一个可选的结构示意图；

[0028] 图3-2是本发明实施例提供的调用处理装置的一个可选的功能结构示意图；

[0029] 图3-3是本发明实施例提供的RN适配层283支持JS模块2832调用本地模块2833以进行通信的可选的示意图；

[0030] 图3-4是本发明实施例提供的RN适配层283支持本地模块2833调用JS模块2832以进行通信的可选的示意图；

[0031] 图3-5是本发明实施例提供的RN适配层283支持JS模块2832调用Java方法的可选的示意图；

[0032] 图4是本发明实施例提供的调用处理方法的一个可选的流程示意图；

[0033] 图5是本发明实施例提供的本地应用启动时，通过SDK 281向服务器查询RN框架的代码并更新的处理示意图；

[0034] 图6-1至图6-6是本发明实施例提供的页面中的JS脚本请求调用终端操作系统的拨号程序,在得到授权后调出拨号程序的示意图。

具体实施方式

[0035] 以下结合附图及实施例,对本发明进行进一步详细说明。应当理解,此处所描述的具体实施例仅仅用以解释本发明,并不用于限定本发明。另外,以下所提供的实施例是用于实施本发明的部分实施例,而非提供实施本发明的全部实施例,在本领域技术人员不付出创造性劳动的前提下,对以下实施例的技术方案进行重组所得的实施例、以及基于对发明所实施的其他实施例均属于本发明的保护范围。

[0036] 对本发明进行进一步详细说明之前,对本发明实施例中涉及的名词和术语进行说明,本发明实施例中涉及的名词和术语适用于如下的解释。

[0037] 1)Webview组件,基于浏览器引擎显示页面的控件,功能包括:

[0038] 1.1)页面统一资源定位符(URL,Uniform Resource Locator)、页面加载、渲染、页面交互等;可以直接使用HTML文件(网络上或终端本地中)作为页面布局显示页面;

[0039] 1.2)支持本地应用与JS脚本(如嵌入页面中的JS脚本)实现相互调用。

[0040] 2)本地(Native)应用,基于终端的本地操作系统如iOS、安卓(Android)、并使用原生代码编写运行的应用程序,也叫本地App,例如包括操作系统内置的应用如拨号、短信程序,还可包括支持安装的第三方的App,如社交应用、支付应用等。

[0041] 本地应用是以终端操作系统的本地开发语言开发的应用,以Android为例是以Java开发的应用,以iOS为例,是以面向对象C(OC,Objected C)开发的应用,使用操作系统本地的组件以及布局。

[0042] 3)网络(Web)应用,是使用网页技术如HTML5实现的应用,运行于网络和浏览器上并实现特定功能,如购物、社交等等。

[0043] 4)JS方法,JS脚本中提供的用于实现特定功能的函数,如判断是否电子邮件、判断图片格式,判断是否是整数,获取存储图片的位置,等等。

[0044] 5)本地(Native)方法,本地应用中提供的用于实现特定功能的函数,如拨号、发送短信,等等。

[0045] 6)本地交互(RN,React Native)框架,基于JavaScript,在JS代码中用前端框架(例如React)抽象操作系统原生的用户界面(UI,User Interface)组件,代替DOM元素来渲染,比如以<View>取代 ,以<Image>替代<img>等,具备动态配置能力的开发框架,使用RN框架中的JavascriptCore引擎或V8引擎对JS脚本进行解析,利用桥(Bridge)接的方式映射到对应的本地方法和用户界面(UI,User Interface)控件。

[0046] 参见图1,图1是本发明实施例提供的JS脚本与本地应用相互调用的一个可选的示意图,在终端10中,硬件层11之上运行操作系统层12,应用层13示出了本地应用,操作系统为Android,本地应用加载的页面中嵌入有JS脚本,本地应用在自身的内容空间中创建Webview组件的实例,作为JS脚本的运行环境,Webview组件中的JS解释器解释JS脚本并执行,实现在页面中的各种功能,如加载动态功能、响应用户的各种操作等。

[0047] 以本地应用为浏览器为例,JS脚本实现的功能包括:检测浏览器、根据用户输入自动完成、显示各种功能按钮、在点击页面的按钮后弹出对话框,校验用户在表单填写数据的

语法是否正确并在错误时提示、搜索页面内容和页面截图，在此不能穷举。

[0048] Webview组件还作为JS脚本与本地应用的通信的中介，JS脚本与本地应用的通信，通过于JS脚本与本地应用的相互调用实现，而JS脚本与本地应用的相互调用，表现为JS方法与本地方法的相互调用，下面进行说明。

[0049] 1)本地应用调用JS脚本

[0050] 本地应用需要调用页面中的JS脚本时，首先，本地应用开启Webview组件对于JS的支持，开启Webview组件对于JS的支持的一个示例为：

[0051] contentWebView.getSettings().setJavaScriptEnabled(true)；其中，true表示设置JS支持开启。

[0052] 然后，本地应用使用Webview组件提供的加载方法(记为loadUrl)调用JS方法的名称并传递参数，调用JS脚本并传递参数的一个示例为：

[0053] webView.loadUrl(“javascript:methodName(parameterValues)；其中，methodName表示调用的JS方法的名称，parameterValues表示向JS方法传递的参数。

[0054] 如果调用的JS方法有返回值，而本地应用通过loadUrl方法无法获取到返回值，本地应用定义一个Java方法供JS方法调用，JS方法通过调用Java方法向JS脚本传递返回值。

[0055] 2)JS脚本调用本地应用

[0056] JS脚本需要调用本地应用时，JS脚本中绑定Webview组件提供的用于调用本地应用的接口，然后调用本地应用的方法并传递参数即可，一个示例为：

[0057] window.JSInterfaceName.methodName(parameterValues)；其中，JSInterfaceName表示用于调用本地应用的接口，methodName(parameterValues)表示需要调用的本地应用的方法的名称，methodName(parameterValues)表示需要向调用的本地应用传入的参数。

[0058] 如前所述，在图1中，本地应用是浏览器等需要显示HTML页面的客户端，在这种情况下，浏览器使用了Webview组件的加载页面元素并渲染HTML页面的功能、以及有解释JS脚本并执行的功能。

[0059] 然而，实际应用中JS脚本也可以作为独立于页面的可执行程序，用于供本地应用调用实现非页面相关的功能，例如，呼出终端操作系统的原生应用如拨号应用，由拨号应用是使用终端操作系统自身的UI来实现，因此不需要Webview来实现页面的显示，在这种情况下，本地应用实际上只需要调用JS脚本，然而，基于图1提供的方案，当本地应用仅调用JS脚本、并实现JS脚本与本地应用的相互调用时，本地应用仍然需要创建Webview组件的实例，同时由于Webview组件的资源占用较高且消息处理逻辑复杂，导致本地应用会占用过多的资源。

[0060] 针对上述问题，本发明实施例还提供了一种调用处理装置、方法及存储介质，对于Android中的本地应用，可以在不创建Webview组件的实例的情况下，实现JS脚本和本地应用的通信，从而实现资源的节约化。

[0061] 下面结合附图说明本发明实施例提供的调用处理装置，参见图2，图2是本发明实施例提供的调用处理装置20的一个可选的硬件结构示意图，调用处理装置20可以是智能手机、平板电脑、计算机和个人数字助理等终端设备。所示的调用处理装置20包括：至少一个处理器21、存储器22、至少一个网络接口24和用户接口23。调用处理装置20中的各个组件通

过总线系统25耦合在一起。可理解，总线系统25用于实现这些组件之间的连接通信。总线系统25除包括数据总线之外，还包括电源总线、控制总线和状态信号总线。但是为了清楚说明起见，在图2中将各种总线都标为总线系统25。

[0062] 其中，用户接口23可以包括显示器、键盘、鼠标、轨迹球、点击轮、按键、按钮、触感板或者触摸屏等。

[0063] 可以理解，存储器22可以是易失性存储器或非易失性存储器，也可包括易失性和非易失性存储器两者。其中，非易失性存储器可以是只读存储器(ROM，Read Only Memory)、可编程只读存储器(PROM，Programmable Read-Only Memory)。本发明实施例描述的存储器22旨在包括但不限于这些和任意其它适合类型的存储器。

[0064] 本发明实施例中的存储器22用于存储各种类型的数据以支持调用处理装置20的操作。这些数据的示例包括：用于在调用处理装置20上操作的任何计算机程序，如操作系统26和程序27；联系人数据；电话簿数据；消息；图片；视频等。其中，操作系统26包含各种系统程序，例如框架层、核心库层、驱动层等，用于实现各种基础业务以及处理基于硬件的任务。程序27可以包含各种应用程序，例如媒体播放器(Media Player)、浏览器(Browser)等，用于实现各种应用业务。实现本发明实施例提供的调用处理方法的调用处理装置28，可以作为软件存储在存储器22中，当被处理器21执行时实现本发明实施例提供的调用方法。

[0065] 再参见图3-1，图3-1是如图2所示的调用处理装置28的一个可选的结构示意图，调用处理装置28用以实现本地应用与JS脚本之间的通信，以Android平台举例来说，本地应用可以是第三方的支持执行JS脚本的App，而调用处理装置28涉及软件开发套件(SDK，Software Development Kit)281、JS执行环境层282和本地应用层283，分别进行说明。

[0066] SDK 281，用于下载JS运行环境层282的代码并运行；例如动态下载/升级/安装/启动用于解释并执行JS脚本的解释器；还提供对本地应用的接入，包括向本地应用提供JS执行环境层282的接口，接口提供如数据交互和对象注入的功能；另外，向本地应用提供的JS执行环境层282的接口还可以提供统计分析的功能。

[0067] JS执行环境层282，提供基于RN的JS执行环境，调用处理装置28的通信机制是依赖RN框架自身的通信机制来实现的，因此能够实现较好的稳定性，下文中将对通过RN框架实现本地方法与JS方法的相互调用进行说明；JS执行环境层282包括RN适配层283和RN实例层284，通过RN适配层283适配RN实例层284与SDK 281进行交互，通过RN实例层284能够运行本地交互框架的内部逻辑。

[0068] RN适配层283，用于适配SDK 281与RN实例层284进行数据交互，参见图3-2，图3-2是本发明实施例提供的调用处理装置的一个可选的功能结构示意图，RN适配层283借助于RN适配层283中创建的交互引擎(react Engine)2831、JS模块(JS Module)2832和本地模块(Native Module)2833三个模块来实现，分别对三个模块进行说明。

[0069] 交互引擎2831，用于负责管理RN框架实例的创建，基于SDK 281下载的RN框架的代码，创建RN框架的实例，另外还负责管理RN框架中JS模块2832和本地模块2833之间的逻辑调用；

[0070] JS模块2832，是针对本地应用而注册到RN实例层284的模块，JS模块2382可以根据本地应用调用JS脚本的需求而注册，或者，预先在RN实例层注册，JS模块2832注册到RN实例层284后，本地应用调用的JS脚本能够在JS模块2832得到执行；

[0071] 本地模块2833，是针对本地应用而注册到RN实例层284的模块，本地模块2833可以根据本地应用调用JS脚本的需求而注册到RN实例层284，或者，预先在RN实例层注册，本地模块2833在RN实例层284注册后，JS脚本回调的本地应用的本地方法能够在本地模块2833得到执行。

[0072] 在RN适配层283创建JS模块2832和本地模块2833两个模块之后，本地应用借助这两个模块实现与JS脚本的通信，包括来使用RN框架调用JS代码执行、以及JS脚本调用本地应用的本地方法。

[0073] RN实例层284，用于执行JS代码，具体来说，创建RN实例2841，利用RN实例2841提供的JS运行环境，将本地应用传入的JS脚本在RN实例2841进行执行，并将结果反馈给RN适配层284，由RN适配层284经由SDK 281反馈给本地应用层283的本地应用。通过使用RN实例2841提供的JS运行环境，在此基础上向SDK 281进行了封装，向本地应用提供对JS脚本的调用并返回结果(在需要返回执行结果时)。

[0074] RN实例层284中包含区别与Webview的用于解释执行JS脚本的引擎(JavascriptCore引擎或V8引擎)，基于该引擎在RN实例层284内部提供JS脚本或本地应用的调用机制，通过RN适配层283将此调用机制运作并与SDK 281衔接，进而通过SDK 281提供给本地应用来使用，并且可以做到RN框架代码的动态下载和更新。通常，本地应用需实现与JS脚本的通信时，需要继承的SDK 281的容量通常在几十K字节，较小的体积使得从实现上来讲更加方便快捷。

[0075] 再参见图3-3和图3-4，图3-3是本发明实施例提供的RN适配层283支持JS模块2832调用本地模块2833以进行通信的可选的示意图，图3-4是本发明实施例提供的RN适配层283支持本地模块2833调用JS模块2832以进行通信的可选的示意图，在图3-3和图3-4中，以本地应用为Android应用为例，由于Java语言是Android应用程序的本地开发语言，因此本地模块2833包括的本地方法也称为Java方法(此时本地模块也可以称为Java模块)，JS模块2832包括使用JS实现的各种功能的JS方法。

[0076] 在图3-3和图3-4中，JS模块2832与本地模块2833的相互调用采用桥的机制，示出了Java桥和JS桥，包括用于进行数据传输的动态链接库，例如使用C++开发的动态链接库，对应提供向本地模块2833和JS模块2832的调用；在JS模块2832与本地模块2833分别存有相同的模块配置表，JS模块2832与本地模块2833互相通信时，通过桥里的模块配置表，将所调用方法转为{模块ID,方法ID,参数}的方式传递给发起调用的模块，发起调用的模块通过模块配置表找到对应的方法执行，在需要向发起调用的模块返回结果时，则通过回调(callback)的方式返回，下面分别进行说明。

[0077] 1)JS模块2832调用Java方法

[0078] 在Java桥和JS桥中分别存有相同的模块配置表，模块配置表中包括了终端操作系统的所有的Java模块的ID和Java模块里的Java方法的ID、以及参数；对于JS脚本需要调用的Java方法，JS模块2832通过JS桥中的模块配置表，将需要调用的Java方法转换为{Java模块ID;Java方法ID;参数}这样的形式，并传递给本地模块2833，本地模块2833根据Java桥中的模块配置表查找对应的Java方法执行，执行结果以回调或调用事件的方式返回JS模块2832。

[0079] 举例来说，本地模块2833中可以定义一个方法，记为RCTSQLManager，其中包括回

调方法,记为query:successCallback;JS模块2832可以直接调用RCTSQLManager.query,并通过回调获取所调用的Java方法的执行结果。

[0080] 再以Andorid为例进行说明,图3-5是本发明实施例提供的RN适配层283支持JS模块2832调用Java方法的可选的示意图,在图3-5中,包括以下步骤:

[0081] 步骤101,JS模块2832调用某个本地模块2833开放的可供JS模块2832调用的Java方法。

[0082] 步骤102,本地模块2833将调用分解为模块名称(ModuleName)、方法名称(MethodName)和参数(args)的形式,再传递给消息队列(MessageQueue)处理。

[0083] 在初始化时模块配置表上的每一个模块时,都生成了对应的远程模块(RemoteModule)对象,对象里也生成了跟模块配置表里一一对应的方法,在方法中可以获取对应模块名称和方法名称,并对回调(callback)进行处理,再移交给MessageQueue。

[0084] 步骤103,MessageQueue把JS的callback函数缓存在MessageQueue的一个成员变量里,用CallbackID代表callback,通过保存在MessageQueue的模块配置表把步骤102传进的ModuleName和MethodName对应转为ModuleID和MethodID。

[0085] 步骤104,把步骤103得到的ModuleID、MethodId、CallbackID和args传给Java方法。

[0086] 步骤105,Java方法接收到消息,通过Java桥的模块配置表获取对应的模块和方法。

[0087] 在初始化时Java方法对模块配置表上的每一个Java模块生成了对应的实例并缓存起来,针对Java模块的每一个Java方法也都生成了对应的RCTModuleMethod对象,这里通过ModuleID和MethodID取到对应的Module实例和RCTModuleMethod实例进行调用。

[0088] 步骤106,RemoteModule对象的方法(记为RCTModuleMethod)对JS方法传过来的每一个参数进行处理。

[0089] RCTModuleMethod可以获得要调用的目标方法的每个参数类型,处理JS类型到目标类型的转换,所有JS方法传过来的数字都是基础数据类型,这里会转成对应的整型/长整型/双精度浮点型(int/long/double)等类型,为块(block)类型参数的生成一个block。

[0090] 例如,-(void)select:(int)index response:(RCTResponseSenderBlock)callback这个方法,获得两个参数的类型为int,block,JS方法传过来的两个参数类型是NSNumber,NSString(CallbackID),这时会把NSNumber转为int,NSString(CallbackID)转为一个block,block的内容是把回调的值和CallbackID传回给JS。

[0091] 参数组装完毕后,通过NSInvocation动态调用相应的本地模块2833的Java方法。

[0092] 步骤107,RCTModuleMethod对本地模块2833的Java方法调用完,执行block回调。

[0093] 步骤108,RCTModuleMethod调用RCTModuleMethod生成的block。

[0094] 步骤109,RCTModuleMethod在block中携带CallbackID和block传过来的参数,调用JS模块2832中的MessageQueue回调方法,记为:invokeCallback。

[0095] 步骤110,MessageQueue通过CallbackID找到相应的JS回调方法。

[0096] 步骤111,MessageQueue调用JS回调方法,并把Java方法带过来的参数一起传回JS模块2832,完成回调。

[0097] 整个流程可以概括为:JS模块2832的调用转为ModuleID/MethodID形式->

callback转CallbackID->本地模块2833根据ID获得方法->处理参数->调用Java方法->回调CallbackID->JS模块2832通过CallbackID拿到callback执行。

[0098] 2)本地模块2833调用JS方法

[0099] 在Java桥和JS桥中分别存有相同一份模块配置表,模块配置表中出包括了终端操作系统的所有的JS模块的ID和JS模块里的JS方法的ID、以及参数;对于本地模块2833需要调用的JS方法,本地模块2833通过Java桥中的模块配置表,将需要调用的JS方法转换为{JS模块ID;JS方法ID;参数}这样的形式,传递给JS模块2832,JS模块2832根据JS桥中的模块配置表查找对应的JS方法执行,执行结果以回调或调用事件的方式返回本地模块2833。

[0100] 再参见图2,实现本发明实施例提供的调用处理方法的调用处理装置28,可以作为软件存储在存储器22中,当被处理器21执行时实现本发明实施例提供的调用方法,就调用方法来说,参见图4,图4是本发明实施例提供的调用处理方法的一个可选的流程示意图,结合图3-1至图3-5示出的代码处理装置,对调用处理的步骤进行说明。

[0101] 步骤201,SDK 281提供对本地应用层283的本地应用的接入。

[0102] 在本发明可选实施例中,SDK 281向本地应用提供JS执行环境层282的接口,通过接口与本地应用进行数据交互,以及供本地应用注入对象。

[0103] 步骤202,SDK 281当本地应用接入SDK 281时,下载RN框架的代码。

[0104] 下载的RN框架的代码用于供JS执行环境层2832的交互引擎2831创建RN框架的实例。

[0105] 如前所述,本地应用通过SDK 282接入JS执行环境层282,相当于在本地应用集成了RN框架,那么,对于本地应用来说,通过RN框架能够容易地实现需要调用的JS代码的更新,下面进行说明。

[0106] RN框架的代码中会将一系列资源打包,打包的文件记为JS bundle文件,包括一系列的可供本地应用调用的JS脚本,当RN实例层284中的RN框架的实例被创建后,加载JS bundle文件,由RN实例层284的解释器(JavascriptCore或V8引擎)解释本地应用调用的JS脚本并执行。

[0107] 参见图5,图5是本发明实施例提供的本地应用启动时,通过SDK 281向服务器查询RN框架的代码并更新的处理示意图,包括以下几种情况:

[0108] 1)更新

[0109] 如果服务器维护的RN框架代码的版本有更新,则下载JS Bundle文件。

[0110] 2)不更新

[0111] 如果服务器维护的RN框架代码的版本没有更新,判断终端本地是否还有缓存的JSBundle文件,包括以下两种情况:

[0112] 2.1)存在:本地存在JS Bundle文件,即有过更新操作。那么本地应用直接加载在缓存目录下的JS Bundle文件;2.2)不存在:终端本地不存在JS Bundle文件,表示之前从未有过更新操作,那么本地应用使用初始化时打包的JS Bundle文件。

[0113] 步骤203,本地应用层283将本地应用需要执行的脚本提交到SDK 281。

[0114] SDK 281向本地应用提供JS执行环境层282的接口,支持本地应用通过JS执行环境层282的接口与RN框架实例2841进行数据交互,以及支持本地应用注入对象。

[0115] 步骤204,SDK 281将接入的本地应用提交的脚本提交到JS执行环境层282。

[0116] 步骤205,JS执行环境层282通过运行RN框架的实例的方式,执行本地应用提交的脚本。

[0117] 步骤206,JS执行环境层282向脚本提供回调本地应用的方法,将脚本的执行结果返回SDK 281。

[0118] 在本发明可选实施例中,JS执行环境层282的RN适配层283适配SDK 281与RN实例层284进行数据交互;举例来说:RN适配层283的交互引擎2831基于SDK 281下载的本地交互框架的代码,在RN实例层284创建RN框架的实例,通过RN框架的实例,JS模块2832与本地模块2833实现通信,表现为模块的相互调用。

[0119] 就JS模块2832调用本地模块2833来说,JS模块2832针对本地应用注册到RN实例层284,执行本地应用提交的脚本,本地模块2833针对本地应用进行注册到RN实例层284,执行本地应用提交的脚本所回调的本地方法,以回调的方式向脚本返回本地方法的执行结果,一个示例如图3-3和图3-4所示,不再重复说明。

[0120] 另外,通过RN实例,还实现本地模块2833对JS模块的调用,对于本地模块2833需要调用的JS方法,转换为{JS模块ID;JS方法ID;参数}这样的形式,传递给JS模块2832,JS模块2832根据JS桥中的模块配置表查找对应的JS方法执行,执行结果以回调或调用事件的方式返回本地模块2833,一个示例如图3-4所示,这里不再重复说明。

[0121] 步骤207,SDK 281向JS执行环境层282返回的脚本的执行结果。

[0122] 步骤208,本地应用层283将SDK 281返回的脚本的执行结果返回本地应用。

[0123] 再结合一个图6-1至图6-6示出的一个实际的应用场景进行说明,基于本发明实施例提供的代码调用方案,本地应用中集成有前述的SDK时,由SDK下载RN框架的代码,创建RN框架的实例,并加载JS Bundle文件;以本地应用为社交客户端为例,用户在聊天窗口中收到其他用户发送的电话号码,用户期望拨打该电话号码,在点击电话号码时,JS Bundle文件中的拨号.JS被执行,拨号.JS在社交客户端中请求拨号的授权,在得到授权后,通过RN框架的实例实现调用手机操作系统的拨号程序,将号码作为参数传入拨号程序,拨号程序使用操作系统原生的UI组件显示拨号界面,由于RN实例使用自身的引擎替代Webview解释并执行JS方法,因此不需要通过创建Webview组件的实例的方式来解析执行JS,相较于Webview组件方式而言能够明显节约资源开销。

[0124] 本发明实施例还提供一种存储介质,存储有可执行程序,当可执行程序被执行时实现本发明实施例提供的代码调用方法,如图4示出的代码调用方法;就存储介质而言,可以包括快闪存储器(Flash Memory)、磁表面存储器、光盘、或只读光盘(CD-ROM,Compact Disc Read-Only Memory)等类型;当可执行程序被执行时实现代码调用方法,例如:软件开发套件层提供对本地应用层的本地应用的接入;本地应用层将接入的本地应用需要执行的脚本,提交到软件开发套件层;软件开发套件层将脚本提交到脚本执行环境层;脚本执行环境层通过运行本地交互框架的实例的方式,执行本地应用提交的脚本,并向脚本提供回调本地应用的方法,将脚本的执行结果返回软件开发套件层;软件开发套件层向本地应用层返回的脚本的执行结果,本地应用层将脚本的执行结果返回本地应用。上述的代码调用方法可以根据针对图4的说明而理解,这里不再重复说明。

[0125] 综上所述,本发明实施例具有以下有益效果:

[0126] 1)以软件开发套件作为本地应用以RN框架的中介,对于本地应用而言,只需要面

向软件开发套件通信，避免了RN框架升级等情况下需要调整适配的情况，提升JS脚本的执行效率；

[0127] 2)基于软件开发套件为本地应用提供接入，软件开发套件作为本地应用与本地交互框架的实例的中介，并基于本地交互框架实现本地应用与脚本的调用，从而，可以基于本地交互框架自有的解释器解释并执行脚本，对于本地应用调用JS脚本的情况而言，不再需要Webview组件即可实现执行JS脚本，避免相关技术中本地应用执行JS脚本时必须创建Webview导致资源开销大的问题。

[0128] 以上所述，仅为本发明的具体实施方式，但本发明的保护范围并不局限于此，任何熟悉本技术领域的技术人员在本发明揭露的技术范围内，可轻易想到变化或替换，都应涵盖在本发明的保护范围之内。因此，本发明的保护范围应以所述权利要求的保护范围为准。

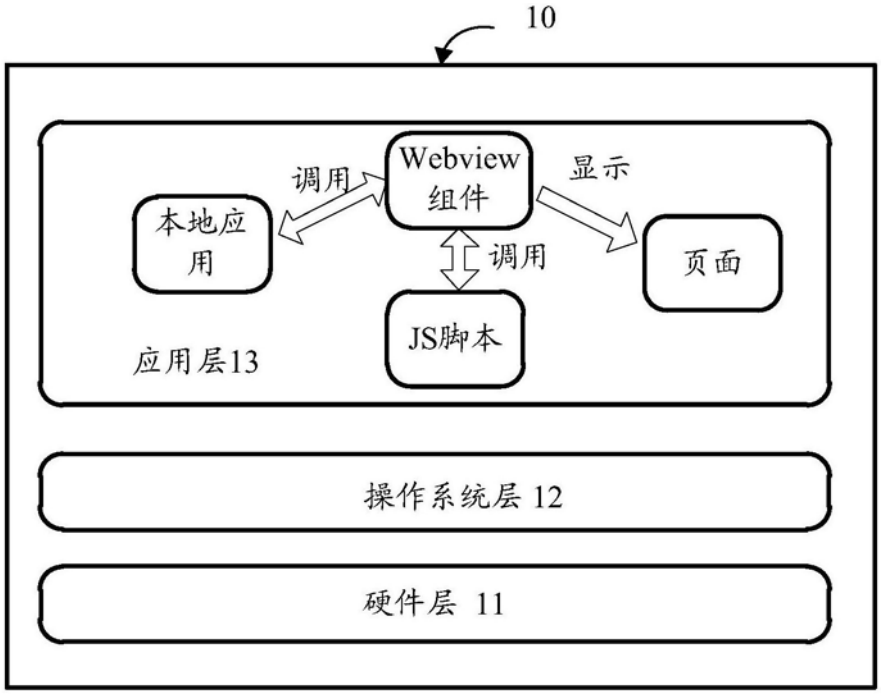

10
Webview
组件
调用
显示
本地应
用
调用
页面
JS脚本
应用层13
操作系统层 12
硬件层 11

图1

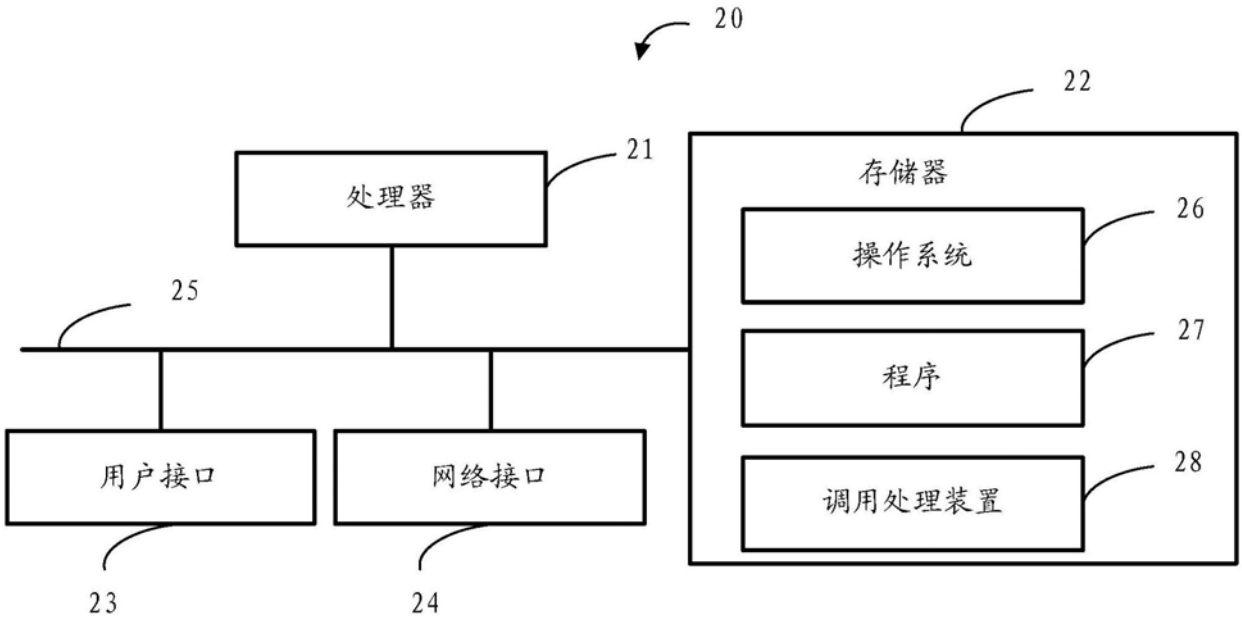

20
22
21
处理器
存储器
26
操作系统
25
27
程序
用户接口
网络接口
28
调用处理装置
23
24

图2

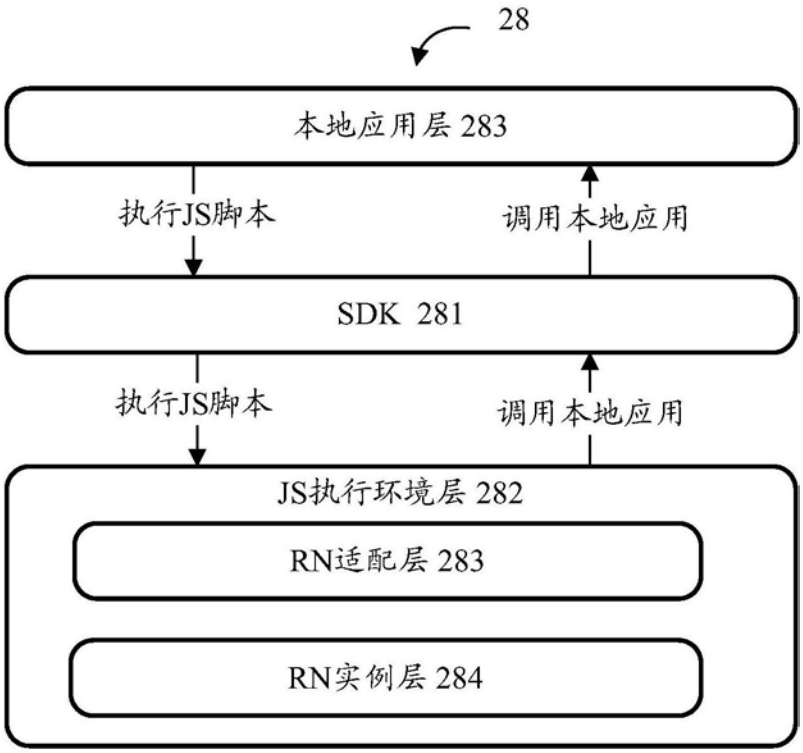

28
本地应用层 283
执行JS脚本
调用本地应用
SDK 281
执行JS脚本
调用本地应用
JS执行环境层 282
RN适配层 283
RN实例层 284

图3-1

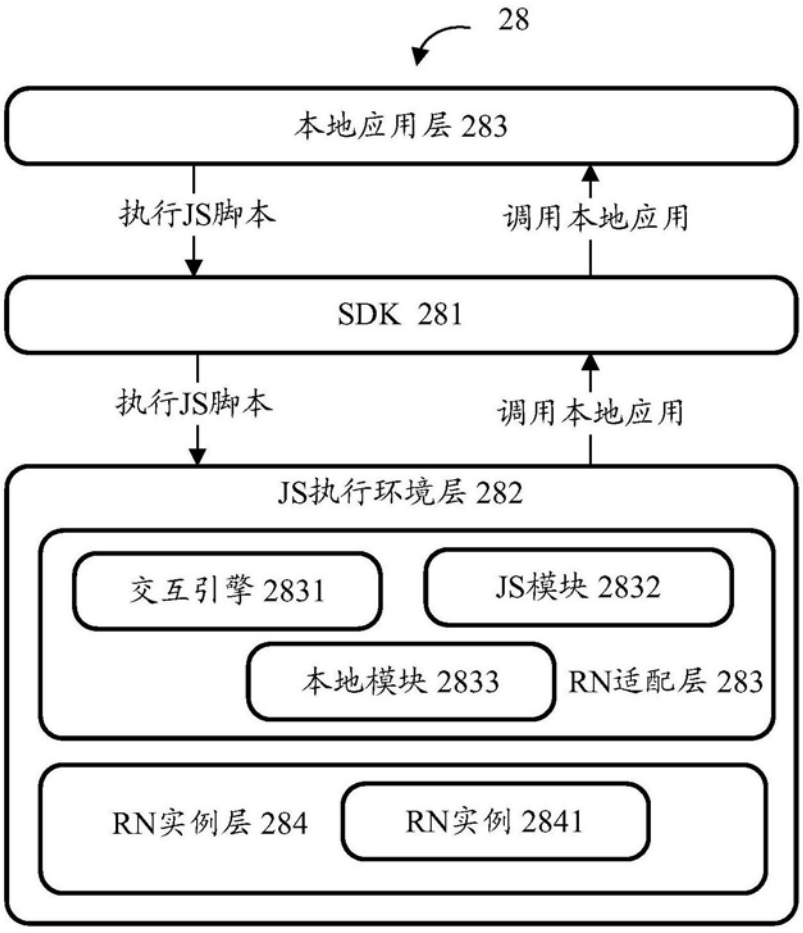

28
本地应用层 283
执行JS脚本
调用本地应用
SDK 281
执行JS脚本
调用本地应用
JS执行环境层 282
交互引擎 2831
JS模块 2832
本地模块 2833
RN适配层 283
RN实例层 284
RN实例 2841

图3-2

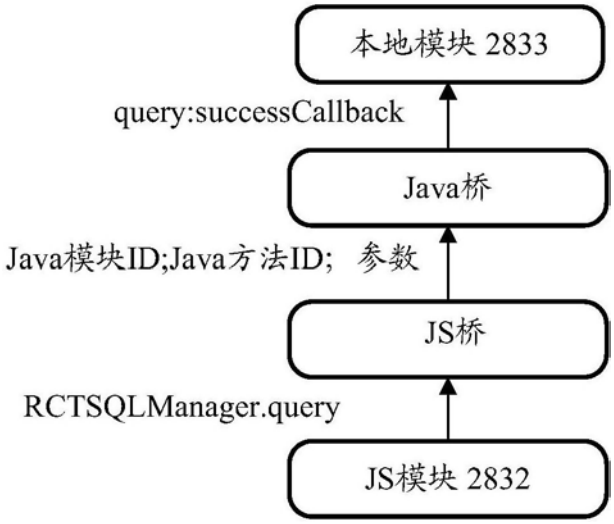

本地模块 2833
query:successCallback
Java桥
Java模块ID;Java方法ID; 参数
JS桥
RCTSQLManager.query
JS模块 2832

图3-3

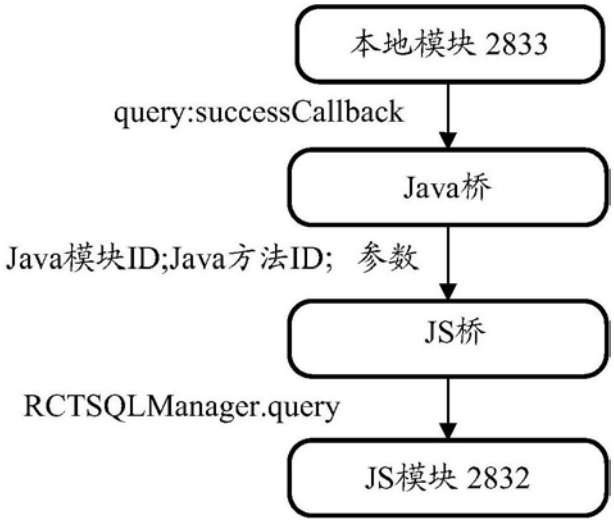

本地模块 2833
query:successCallback
Java桥
Java模块ID;Java方法ID; 参数
JS桥
RCTSQLManager.query
JS模块 2832

图3-4

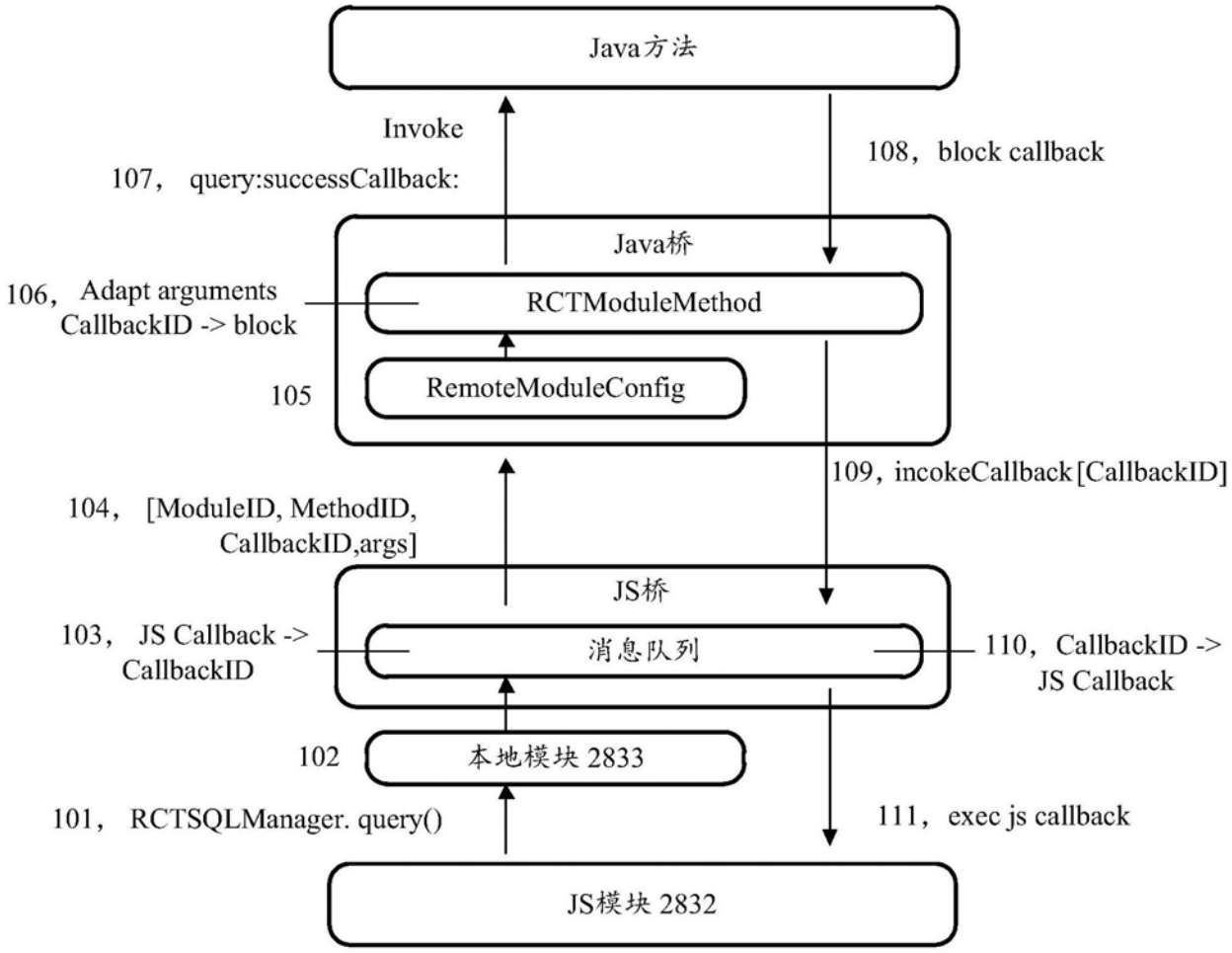

Java方法
Invoke
107， query:successCallback:
108， block callback
Java桥
106， Adapt arguments
CallbackID -> block
RCTModuleMethod
105
RemoteModuleConfig
109，incokeCallback[CallbackID]
104， [ModuleID, MethodID,
CallbackID,args]
JS桥
103， JS Callback ->
CallbackID
消息队列
110， CallbackID ->
JS Callback
102
本地模块 2833
101， RCTSQLManager. query()
111， exec js callback
JS模块 2832

图3-5

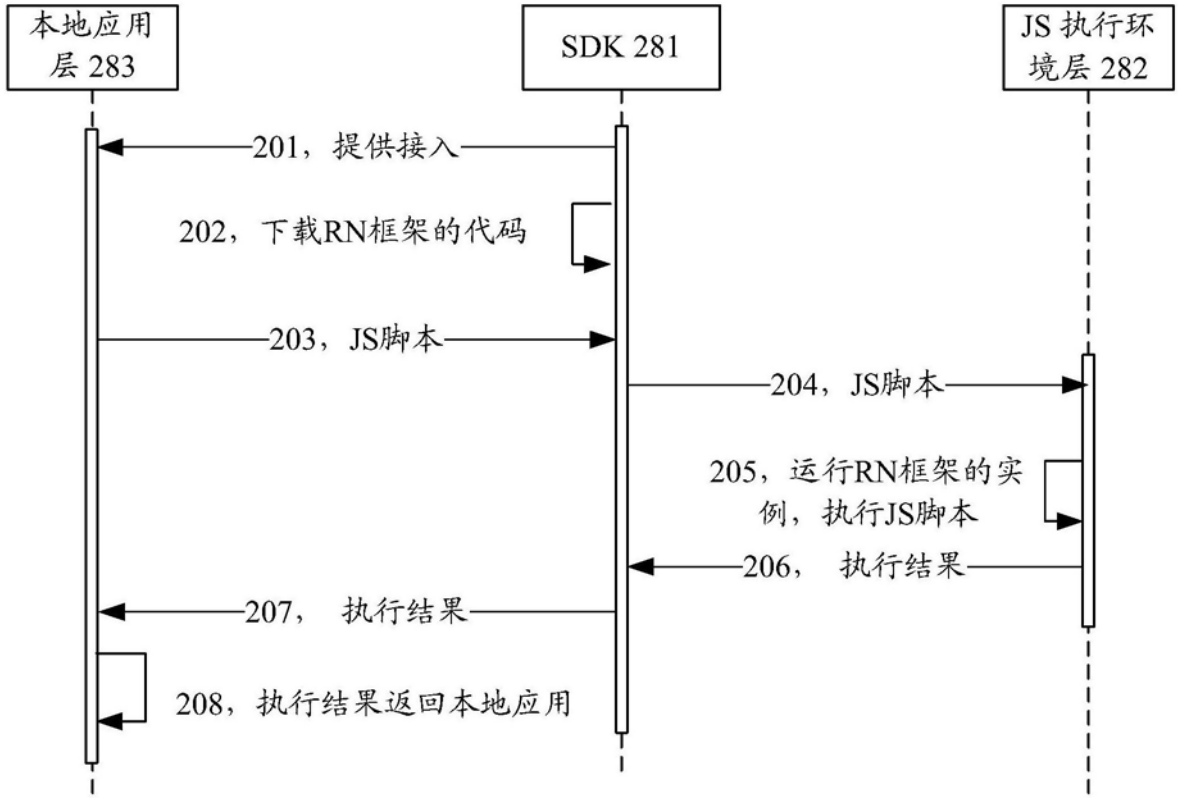

本地应用层 283
SDK 281
JS 执行环境层 282
201，提供接入
202，下载RN框架的代码
203，JS脚本
204，JS脚本
205，运行RN框架的实例，执行JS脚本
206， 执行结果
207， 执行结果
208，执行结果返回本地应用

图4

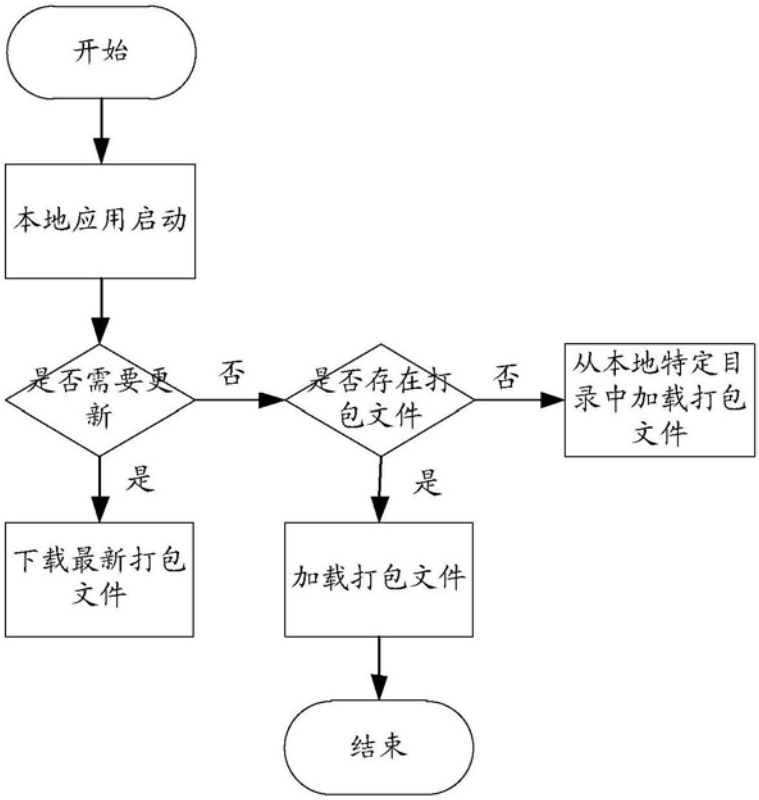

开始
本地应用启动
是否需要更新
否
是否存在打包文件
否
从本地特定目录中加载打包文件
是
是
下载最新打包文件
加载打包文件
结束

图5

图6-1


11:52
0.54
ReactNativeAppDemo
加载最新BUNDLE
进入RN界面
向RN发消息
发送成功

图6-2

图6-3


11:48
7.19
React Native界面
ReactNativeAppDemo申请拨打电话权限
拒绝
允许

图6-4

图6-5

图6-6