(12) 按照专利合作条约所公布的国际申请

(19) 世界知识产权组织
国 际 局

(43) 国际公布日
2018 年 9 月 13 日 (13.09.2018)

WIPO | PCT

(10) 国际公布号
WO 2018/161813 A1

(51) 国际专利分类号:
G06F 17/30 (2006.01)

(21) 国际申请号: PCT/CN2018/077220

(22) 国际申请日: 2018 年 2 月 26 日 (26.02.2018)

(25) 申请语言: 中文

(26) 公布语言: 中文

(30) 优先权:
201710137173.1 2017年3月8日 (08.03.2017) CN

(71) 申请人: 阿里巴巴集团控股有限公司 (ALIBABA GROUP HOLDING LIMITED) [—/CN]; 开曼群岛大开曼资本大厦一座四层 847 号邮箱, Grand Cayman (KY)。

(72) 发明人: 闵洪波 (MIN, Hongbo); 中国浙江省杭州市余杭区文一西路969号3号楼5楼阿里巴巴集团法务部, Zhejiang 311121 (CN)。 朱永盛 (ZHU, Yongsheng); 中国浙江省杭州市余杭区文一西路969号3号楼5楼阿里巴巴集团法务部, Zhejiang 311121 (CN)。 鲁振华 (LU, Zhenhua); 中国浙江省杭州市余杭区文一西路969号3号楼5楼阿里巴巴集团法务部, Zhejiang 311121 (CN)。 林志平 (LIN, Zhiping); 中国浙江省杭州市余杭区文一西路969号3号楼5楼阿里巴巴集团法务部, Zhejiang 311121 (CN)。 蔡艳明 (CAI, Yanming); 中国浙江省杭州市余杭区文一西路969号3号楼5楼阿里巴巴集团法务部, Zhejiang 311121 (CN)。 曾旭 (ZENG, Xu); 中国浙江省杭州市余杭区文一西路969号3号楼5楼阿里巴巴集团法务部, Zhejiang 311121 (CN)。

(74) 代理人: 北京三友知识产权代理有限公司 (BEIJING SANYOU INTELLECTUAL PROPERTY AGENCY LTD.); 中国北京市金融街 35 号国际企业大厦 A 座 16 层, Beijing 100033 (CN)。

(81) 指定国 (除另有指明, 要求每一种可提供的国家保护): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB,

(54) **Title:** RESOURCE LOADING METHOD AND DEVICE

(54) 发明名称: 一种资源加载方法及装置

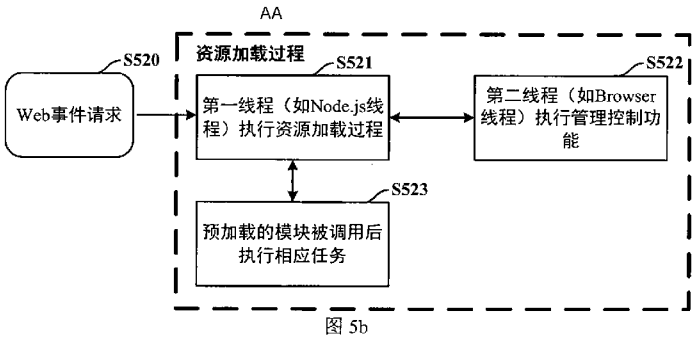

图 5b

S520 WEB EVENT REQUEST
S521 A FIRST THREAD (SUCH AS A NODE.JS THREAD) EXECUTES A RESOURCE LOADING PROCESS
S522 A SECOND THREAD (SUCH AS A BROWSER THREAD) EXECUTES A MANAGEMENT CONTROL FUNCTION
S523 A PRELOADED MODULE EXECUTES A CORRESPONDING TASK AFTER BEING INVOKED
AA RESOURCE LOADING PROCESS

(57) **Abstract:** Disclosed in the present application are a resource loading method and device, applicable to the field of computer technologies. In the present application, a first thread sends a resource loading request to a second thread, the first thread and the second thread being within a single process, the first thread running on a dynamic language application running platform; the first thread receives an instruction returned by the second thread according to the resource loading request; the first thread loads, according to the instruction and on the basis of resources pre-loaded by the process, resources requested by the resource loading request to be loaded, the module pre-loaded by the process comprising a Web engine. The present application can realize the fusion of a Web engine and Node.js.

WO 2018/161813 A1

[见续页]

GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW。

(84) 指定国(除另有指明，要求每一种可提供的地区保护)：ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 欧亚 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 欧洲 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG)。

本国际公布:

— 包括国际检索报告(条约第21条(3))。

(57) 摘要：本申请公开了一种资源加载方法及装置，应用于计算机技术领域。本申请中，第一线程向第二线程发送资源加载请求，其中，所述第一线程和所述第二线程位于同一进程，所述第一线程运行于动态语言应用运行平台；所述第一线程接收所述第二线程根据所述资源加载请求返回的指示；所述第一线程根据所述指示，基于所述进程预加载的资源，对所述资源加载请求所请求加载的资源进行加载，其中，所述进程预加载的模块中包括Web引擎。采用本申请可实现Web引擎与Node.js的融合。

一种资源加载方法及装置

本申请要求2017年03月08日递交的申请号为201710137173.1、发明名称为“一种资源加载方法及装置”的中国专利申请的优先权，其全部内容通过引用结合在本申请中。

技术领域

本申请涉及计算机技术领域，尤其涉及一种资源加载方法及装置。

背景技术

随着移动互联网的快速发展与HTML5（HTML是HyperText Markup Language的英文缩写，即超文本标记语言）技术的逐步成熟，万维网（Web）应用已经成为移动端跨平台应用开发的热门解决方案。

Web应用通过浏览器获取和显示Web资源，以页面形式显示Web资源。浏览器的功能可通过Web引擎实现。Web引擎包含了各种组件，比如用于实现页面渲染的渲染引擎、用于进行管理和控制的浏览器引擎等。

Node.js是JavaScript运行环境（runtime），也称运行平台，提供了多种系统级的应用程序编程接口（Application Programming Interface，API），用于方便地搭建响应速度快、易于扩展的网络应用。

发明内容

本申请实施例提供一种资源加载方法及装置。

本申请实施例提供了一种资源加载方法，包括：

第一线程向第二线程发送资源加载请求；其中，所述第一线程和所述第二线程位于同一进程，所述第一线程运行于动态语言应用运行平台；

所述第一线程接收所述第二线程根据所述资源加载请求返回的指示；

所述第一线程根据所述指示，基于所述进程预加载的资源，对所述资源加载请求所请求加载的资源进行加载，其中，所述进程预加载的资源中包括Web引擎。

本申请实施例提供的一种资源加载装置，包括：第一线程单元和第二线程单元；

所述第一线程单元，用于：

向所述第二线程单元发送资源加载请求；其中，所述第一线程单元和所述第二线程单元属于同一进程单元，所述第一线程单元运行于动态语言应用运行平台；

接收所述第二线程单元根据所述资源加载请求返回的指示；以及，

根据所述指示，基于所述进程单元预加载的资源，对所述资源加载请求所请求加载的资源进行加载；其中，所述进程单元预加载的模块中包括 Web 引擎。

本申请实施例提供的一个或多个计算机可读介质，所述可读介质上存储有指令，所述指令被一个或多个处理器执行时，使得通信设备执行上述方法。

本申请实施例提供的一种通信设备，包括：一个或多个处理器；以及，一个或多个计算机可读介质，所述可读介质上存储有指令，所述指令被所述一个或多个处理器执行时，使得所述装置执行上述方法。

本申请的上述实施例中，应用程序的进程预加载有资源，其中，所述进程预加载的资源中包括 Web 引擎。同一进程中的第一线程向第二线程发送资源加载请求，所述第一线程运行于动态语言应用运行平台；所述第一线程接收所述第二线程根据所述资源加载请求返回的指示；所述第一线程根据所述指示，基于所述进程预加载的模块和/或资源，对所述资源加载请求所请求加载的资源进行加载，从而实现了 Web 引擎与所述动态语言应用运行平台的融合。

附图说明

本申请的实施例通过示例而非限制的方式示出在所附附图中，类似的附图标记表示类似的元素。

图 1 根据一些实施例示例性地示出了基于 Node.js 的云操作系统架构 100；

图 2 根据一些实施例示例性地示出了预加载有 Web 引擎的 Node.js 的示意图；

图 3 根据一些实施例示例性地示出了 Web 引擎的 API 映射关系示意图；

图 4 根据一些实施例示例性地示出了预加载 Web 引擎后 Node.js 线程和 Browser 线程及相关资源的示意图；

图 5a 根据一些实施例示例性地示出了预加载流程示意图；

图 5b 根据一些实施例示例性地示出了 Web 资源加载流程示意图；

图 6 根据一些实施例示例性地示出了事件循环机制示意图；

图 7 根据一些实施例示例性地示出了资源加载装置的结构示意图；

图 8 根据一些实施例示例性地示出了一种装置示意图。

具体实施方式

虽然本申请的概念易于进行各种修改和替代形式，但是其具体实施例已经通过附图中的示例示出并且将在本文中详细描述。然而，应当理解，没有意图将本申请的概念限制为所公开的特定形式，而是相反，意图是覆盖与本申请以及所附权利要求一致的所有修改、等同物和替代物。

说明书中对“一个实施例”、“实施例”、“说明性实施例”等的引用，指示所描述的实施例可包括特定特征、结构或特性，但是每个实施例可以或可以不必包括特定特征、结构或特性。此外，这样的短语不一定指的是相同的实施例。进一步地，认为在本领域技术人员的知识范围内，当结合实施例描述特定特征、结构或特性时，结合无论是否明确描述的其它实施例影响这样的特征，结构或特性。另外，应当理解，以“A，B和C中的至少一个”的形式包括在列表中的项目可以表示（A）;（B）;（C）;（A和B）;（A和C）;（B和C）;或（A，B和C）。类似地，以“A，B或C中的至少一个”的形式列出的项目可以表示（A）;（B）;（C）;（A和B）;（A和C）;（B和C）或（A，B和C）。

在一些情况下，所公开的实施例可以在硬件、固件、软件或其任何组合中实现。所公开的实施例还可以被实现为由一个或多个暂时性或非暂时性机器可读（例如，计算机可读）存储介质携带或存储的指令，其可以由一个或多个处理器读取和执行。机器可读存储介质可以体现为用于以机器可读形式（例如，易失性或非易失性存储器、介质盘或其他介质）存储或传输信息的任何存储设备，机制或其他物理结构的设备）。

在附图中，一些结构或方法特征可以以特定布置和/或顺序示出。然而，应当理解，可能不需要这样的具体布置和/或排序。相反，在一些实施例中，这些特征可以以与说明性附图中所示不同的方式和/或顺序来布置。另外，在特定图中包括结构或方法特征并不意味着暗示这种特征在所有实施例中都是需要的，并且在一些实施例中可以不包括或可以与其他特征组合。

操作系统用于向用户应用提供操作系统的基础能力，可基于动态语言应用运行平台实现，从而为动态语言应用提供运行环境。其中，Node.js 即为 JavaScript 的运行环境或运行平台。

其中，动态语言是计算机编程语言中的一个语言类别，是一类在运行时可以动态地改变类型、结构的语言，在运行时函数和属性可以被增加、修改和删除。例如 JavaScript、Python、Ruby 等都属于动态语言。动态语言不需要编译即可运行，在运行时需要运行环境的支撑，这个环境叫做运行时环境，它包含动态语言运行所需要的所有要素，例如 Java

虚拟机、JavaScript 引擎等。

以云操作系统（云 OS）中的 Host 系统为例，它可基于 Node.js 实现。Node.js 是 JavaScript 的运行环境，是建立在 Chrome 上的 JavaScript 引擎的 Web 应用程序框架，也就是说，操作系统基于 Node.js 向用户应用提供操作系统的基础能力。Node.js 中包括多种模块，这些模块可通过将用于实现特定功能的代码（比如实现操作系统级服务功能的代码）进行封装得到，比如可封装为组件形式，例如这些模块中可包括实现全球定位系统（Global Positioning System，简称 GPS）定位功能的组件、实现电源管理功能的组件等。这些模块的接口被暴露给应用层，应用层中的应用程序（比如 Web 应用）可通过调用这些模块的接口，通过 JavaScript 引擎解析模块中的代码，执行这些模块提供的方法，从而实现这些模块提供的功能。

对于 Web 应用来说，Web 资源的加载（比如 Web 页面的获取、渲染等操作）由浏览器实现，浏览器则基于 Web 引擎（web engine）实现 Web 资源加载。目前，传统的 HTML5 浏览器，其 Web 引擎中的线程包括两种类型：Browser 线程（浏览器线程）和 Render 线程（渲染线程）。其中，Browser 线程为主线程，可为各个 Web 页面的渲染提供服务和管理能力；Render 线程为子线程，可基于 Web 引擎进行页面渲染以及执行 JavaScript 代码。Browser 线程可创建一个或多个 Render 线程。比如，在加载一个 Web 页面的过程中，Browser 线程创建一个 Render 线程，由该 Render 线程对该 Web 页面进行渲染。如果需要加载多个 Web 页面，则 Browser 线程创建多个 Render 线程，每个 Render 线程对一个 Web 页面进行渲染。

图 1 示例性地示出了一种操作系统架构 100。架构 100 包括应用层 10、应用框架层 20 和操作系统层 30。应用层 10 中包括一个或多个应用，其中可包括浏览器。程序框架层 20 包括动态语言应用运行平台（比如 Node.js）。操作系统层 30 主要用于提供操作系统级别的代码库以及基本的系统服务，比如提供的系统服务可包括设备驱动服务、事件管理服务等。

浏览器中的 Web 引擎处于应用层，不具备使用 Node.js 提供的系统服务的能力。如果需要使 Web 引擎中的渲染（Render）线程能够使用 Node.js 提供的能力，则需要在 Render 线程启动过程中启动 Node.js。考虑到每加载一个 Web 页面就会启动一个 Render 线程，也就需要启动一次 Node.js，因此导致加载 Web 页面的资源开销和时间开销较大。

如何将 Web 引擎与 Node.js 融合，将 Web 引擎作为系统服务的一部分，使基于 HTML 请求的 web 页面访问处理过程能够使用系统提供的各种能力，是目前业界面临的问题。

本申请实施例基于上述架构，使 Web 引擎可以访问到 Node.js 提供的能力，进一步地与上述方式相比还可以减少资源开销和时间开销。下面结合附图对本申请实施例进行详细描述。

本申请实施例中，动态语言应用运行平台启动时可预加载各种资源，其中包括 Web 引擎。

以动态语言应用运行平台为 Node.js 为例，Node.js 启动过程中，Node.js 进程被创建。Node.js 进程预加载各种能力，这里所说的“能力”也可称为资源，可包括前述所描述的用于实现特定功能的模块（比如组件），还可以包括一些配置信息等，比如上下文、事件队列等。Node.js 所预加载的能力中至少包括 Web 引擎（比如 Android 系统中的 Webview 组件（即 web 视图组件）），进一步还可包括 Node.js 提供的能力。其中，Node.js 提供的能力可包括：Node.js 中的模块（比如用于提供特定功能或服务的组件）和/或 JavaScript 运行环境相关信息。Node.js 中的模块可通过对用于实现特定功能的 JavaScript 代码封装得到。每个模块均具有接口（比如 API），可提供给应用层供应用程序调用。JavaScript 运行环境相关信息可包括但不限于 JavaScript 引擎的上下文、事件循环相关配置中的一种或多种。其中，上下文用于描述引擎内部的各种对象、状态和功能。JavaScript 引擎的上下文中可包括由 Node.js 进程所预加载的模块的接口（如 API）。事件循环相关配置具体可包括 Node.js 的事件队列的配置，比如事件队列所在的内存区域位置、队列大小等，以便向该事件队列中存取 Node.js 事件请求和 Web 事件请求。

在一些实施例中，可将预加载的能力（比如 Web 引擎）封装为动态语言应用运行平台中的标准模块，作为该平台的一个标准模块预加载到该平台中。图 2 以该平台为 Node.js 为例，示例性地示出了预加载有 Web 引擎的 Node.js 的示意图。

在一些实施例中，Web 引擎由 c/c++语言编写，而 Node.js 支持 JavaScript 语言，因此需要将 Web 引擎的符合 c/c++标准的 API 映射为 Node.js 中符合 JavaScript 标准的 API，并将映射得到的符合 JavaScript 标准的 API 提供给应用层，Node.js 中保存该 API 的映射关系。图 3 示例性地示出了 Web 引擎的 API 映射关系示意图。其中，Web 引擎中可包括多种组件，比如 Window system 组件（窗口系统组件）、UI Elements 组件（用户界面部件组件）、Layout 组件（布局组件）、Event Handling 组件（事件处理组件）等，这些符合 c/c++标准的组件的 API 被一一映射为对应的符合 JavaScript 标准的 API。

在一些实施例中，进一步地，预加载的能力还可包括但不限于以下中的一种或多种组合：

➢ 操作系统提供的能力

操作系统提供的能力可包括：操作系统层提供的系统服务等。操作系统层中可包括用于提供系统服务的模块，这些模块可由用于实现特定能力的代码封装得到。每个模块均具有接口（比如 API）。本申请实施例中，可通过将操作系统层中的模块的 API 添加到动态语言引擎（比如 JavaScript 引擎）的上下文中，实现对操作系统能力的预加载。

➢ 用户自定义的能力

用户自定义的能力，即用户自定义的用于实现特定功能的代码，可被封装为模块，每个模块均具有接口（如 API）。本申请实施例中，可通过将用户自定义的模块的 API 添加到动态语言引擎（比如 JavaScript 引擎）的上下文中，实现对用户自定义能力的预加载。

在一些实施例中，Node.js 中的 JavaScript 上下文中的模块接口（比如 API）被添加到 Web 引擎的 JavaScript 上下文中，这样，对于应用程序的开发者来说，可利用 Web 引擎的 JavaScript 上下文中包含的 API 进行应用程序开发，当该应用程序运行时，Node.js 线程可通过这些接口调用相应的模块实现相应的功能，即访问这些接口对应的能力。由于 Node.js 的上行文中的模块接口可包括 Node.js 中的模块接口，还可包括操作系统提供的模块接口以及用户自定义的模块的接口，从而使得 Node.js 线程能够访问的能力更加丰富。Node.js 的 JavaScript 上下文中包含有 Web 引擎的 JavaScript 上下文中的模块接口也可被添加到 Node.js 的 JavaScript 上下文中。

Node.js 进程中的主线程称为 Node.js 线程，它可以共享 Node.js 进程的运行环境以及 Node.js 进程预加载的能力。也就是说，Node.js 线程可以访问 Node.js 进程所预加载的各种能力。由于 Node.js 进程预加载了 Web 引擎，因此，Node.js 线程可以基于 Web 引擎实现 Web 引擎所提供的功能，比如页面渲染功能。

图 4 示例性地示出了预加载 Web 引擎后 Node.js 线程和浏览器线程（Browser 线程）及相关资源的示意图。如图所示，应用进程中包括 Node.js 线程和 Brower 线程，Node.js 线程和 Browser 线程之间可以通信。Node.js 线程基于 Node.js 的事件循环机制进行事件的处理。Node.js 线程可基于 Web 引擎实现 Web 资源加载的操作。作为一个例子，Node.js 线程可通过调用 Web 引擎提供的页面加载方法，实现页面加载。在页面加载方法的执行过程中，Node.js 线程可基于 Web 引擎中的 JavaScript 上下文中的 API，调用相应的模块（组件）。JavaScript 引擎对被调用的模块进行解析，获得操作系统层提供给 JavaScript 引擎的接口，并基于该接口调用操作系统层提供的能力（比如图中的设备能力、网络服

务、电源管理等），以实现页面加载操作。对于图 4 中所示的应用进程来说，其中的 Node.js 线程为主线程，Browser 线程为子线程，即 Browser 线程可由 Node.js 线程创建得到，它们之间可基于线程间通信机制进行通信。该应用进程与其他进程（比如图中位于操作系统层的网络服务模块对应的进程）之间基于进程间通信（Inter-Process Communication，IPC）机制进行通信。

比如，Node.js 线程可调用图 4 中的模块 1（网络模块）以执行页面获取方法，在该方法的执行过程中，可基于操作系统层提供的接口调用操作系统层提供的网络服务，以获取所请求的页面。再比如，Node.js 线程可调用图 4 中的模块 2（能力获取模块）以执行设备能力（比如显示屏幕大小等设备信息）的获取方法以及根据该设备能力确定显示模式，在该方法的执行过程中，可基于操作系统层提供的接口调用操作系统层提供的设备能力，以获取设备能力信息。

图 5a 示例性地示出了本申请实施例提供的预加载流程。如图所示，在 S510 中，进程启动（比如接收到在浏览器窗口中打开一个页面的事件后启动一个 Node.js 进程）。在 S511~S514 中，该进程执行初始化过程，在该过程中，该进程预加载网络模块、系统模块、渲染引擎等模块，具体预加载过程可参见前述实施例的描述。该进程可以是 Node.js 进程。

进一步地，在 S511~S514 中，第一线程被创建。第一线程通过调用 Web 引擎提供的新建对象的方法创建 Web 组件对象，并启动该 Web 组件对象，创建第二线程。此后第一线程即可基于该 Web 组件对象，通过调用该对象提供的资源加载方法，在浏览器窗口中加载页面。其中，第一线程可以是 Node.js 线程，第二线程可以是 Browser 线程。

图 5b 示例性地示出了本申请实施例提供的资源加载流程。如图所示，在 S520~S523 中，当第一线程获得 Web 事件请求后（比如用户通过浏览器发送 HTML 请求，以请求打开一个 web 页面，则用户的上述操作触发生成 web 事件请求），执行资源加载过程，以加载所请求的 web 页面，该资源加载过程可包括资源的获取以及页面的渲染等。在该资源加载过程中，第一线程向第二线程发送资源加载请求，接收第二线程根据该资源加载请求返回的指示，根据该指示，基于第一线程所在的进程预加载的资源，对资源加载请求所请求加载的资源进行加载。该过程中，第二线程主要用于实现管理和控制功能可指示第一线程调用资源加载过程所需的系统模块，以实现该资源加载过程；资源获取以及页面渲染等操作由第一线程执行。第一线程可通过调用预加载的模块，比如网络模块、系统模块等，执行相应的任务。需要说明的是，第一线程每次获得 Web 事件请求后，可

根据预加载的模块对该事件请求进行响应。

上述过程中，第二线程向第一线程发送的指示可包括有关资源加载的控制信息（比如控制指令），或者资源加载操作需要使用的资源的指示信息（比如需要调用的 API），或者资源加载所需的系统信息。作为一个例子，第二线程可以向第一线程发送的指示中可包括以下控制信息中的一种，以控制第一线程的网页加载操作：开始加载网页的控制信息、停止加载网页的控制信息，加载上一页或下一页的信息。作为另一个例子，第二线程向第一线程发送的指示中可携带系统模块提供的 API，使得第一线程可以利用该 API 调用相关系统模块的函数，以实现诸如加减音量、动态获取设备电池情况、实时获取地理位置或设备震动等功能。再例如，第二线程向第一线程发送的指示也可以包括系统信息（即第第二线程将系统信息传递给第一线程），该系统信息可供第一线程使用，比如该系统信息可包括设备的媒体接入控制（Media Access Control，MAC）地址、设备标识，用户账户信息等。

作为一个例子，web 页面加载过程可包括：Node.js 线程基于网络模块获取 Web 事件请求所请求的 HTML 文档；解析获得到的 HTML 文档，将 HTML 文档中的标签转换为文档对象模型（Document Object Model，DOM）树中的 DOM 节点，得到 DOM 树结构；解析层叠样式表（Cascading Style Sheets，简称 CSS）文件，解析得到的信息与 HTML 文档中可见的指令（如等）被用来构建渲染树（Render 树），这棵树主要由一些包含颜色和宽高等属性组成的矩形组成，它们将依次按顺序显示在屏幕上；Node.js 线程根据 Render 树执行布局过程，确定每个节点在屏幕上对应的坐标及其覆盖和回流情况等，遍历该 Render 树， 使用 UI 后端层绘制每个节点。

在一些实施例中，在 S521 中，Node.js 线程可使用 Node.js 中预加载的资源执行该方法。比如，Node.js 线程可基于 Web 引擎的 JavaScript 上下文中模块的接口，通过调用对应模块提供的方法，加载所请求的资源。由于 Web 引擎的 JavaScript 上下文中的接口，可包括 Node.js 中原有的模块的接口，还可包括操作系统层中的模块的接口以及用户自定义的模块的接口，从而将各种能力融合在 Web 引擎中。

在另一些实施例中，在 S521 中，Node.js 线程可基于在 Node.js 启动过程中初始化的 Node.js 中的 JavaScript 引擎，对所请求加载的 Web 资源进行加载，而无需再次初始化该 JavaScript 引擎。具体地，Node.js 线程可根据 Web 引擎提供的用于资源加载的模块的接口，调用相应的模块；通过 Node.js 进程预加载的 JavaScript 引擎对该模块进行解析，得到该模块所调用的操作系统中的模块接口；根据解析得到的模块接口调用操作系统中的

相应模块。

Node.js 和 Web 引擎均采用事件循环（Event loop）机制。现有技术中，Web 引擎的事件循环与 Node.js 的事件循环彼此独立。本申请实施例中，Web 引擎被预加载到动态语言应用运行平台（如 Node.js）中，因此 Web 引擎的事件与该平台的事件可基于事件队列进行事件循环。在没有事件发生的情况下，事件循环处于等待状态并阻塞，当有事件发生时（比如 Web 引擎有事件发生时），会将事件循环唤醒。比如，当 Web 引擎有事件发生时，可通过异步方式唤醒事件循环，该 Web 引擎的事件被存储于事件队列。该事件队列通常采用先进先出的机制。

图 6 以 Node.js 为例，示例性地示出了本申请实施例的事件循环机制。如图所示，Node.js 事件请求和 Web 事件请求按照发生的先后顺序被存储在事件队列中。在本申请实施例中，将 Web 事件请求作为一个整体参与事件循环。比如可以将 Web 引擎事件存储为一个子队列。Node.js 线程从事件队列中读取事件进行响应时，如果当前读取到的是该子队列，根据调度策略的规定，将该子队列中的全部事件请求取出，Web 引擎事件关联的处理程序（Handler）会处理该子队列中的所有事件请求，在全部响应完成后，返回到事件循环，从事件队列继续读取其他的事件请求进行响应，实现了将 Web 引擎中的事件处理融入到了 Node 的事件循环中。在另外的例子中，调度策略也可以规定：如果当前读取到的是 Web 引擎对应的子队列，则根据预设的处理能力（比如规定一次取出 K 个事件请求进行响应，K 的取值可设置），将该子队列中的相应数量的事件请求取出，Web 事件请求的处理程序（Handler）会处理所取出的 Web 事件请求，在响应这些事件请求后，返回到事件循环。

通过以上描述可以看出，本申请的上述实施例中，Node.js 进程预加载有资源，其中，所述 Node.js 进程预加载的资源中包括 Web 引擎。同一 Node.js 进程中的 Node.js 线程向浏览器线程发送资源加载请求，其中，所述 Node.js 线程为主线程，所述浏览器线程为子线程；所述 Node.js 线程接收所述浏览器线程根据所述资源加载请求返回的指示；所述 Node.js 线程根据所述指示，基于所述 Node.js 进程预加载的资源，对所述资源加载请求所请求加载的资源进行加载，从而实现了 Web 引擎与 Node.js 的融合。

由于 Web 引擎的通用性，因此具有较好的跨平台能力。本申请实施例可在保证 Web 引擎跨平台应用的前提下，将其融入 Node.js，使其能够访问 Node.js 提供的能力，为提供功能更加灵活和丰富的应用提供了可能性。

本申请的上述实施例可应用于移动终端，比如手机、智能穿戴设备、车载设备、个

人数字助理（Personal Digital Assistant，PDA）等。以应用于基于云操作系统的手机为例，采用本申请实施例，可由 Node.js 的主线程基于 Web 引擎实现页面渲染，从而使得页面渲染操作可使用 Node.js 主线程所能访问的能力，实现了 Node.js 与 Web 引擎的融合。

基于相同的技术构思，本申请实施例还提供了一种资源加载装置，可实现前述实施例描述的资源加载流程。

图 7 示例性地示出了本申请实施例提供的资源加载装置的结构示意图。该装置可包括：第一线程单元 701 和第二线程单元 702，第一线程单元 701 和所述第二线程单元 702 属于同一进程单元，所述第一线程单元运行于动态语言应用运行平台。进一步地，还装置还可包括事件调度单元 703。

第一线程单元 701 用于：向所述第二线程单元发送资源加载请求，接收第二线程单元 702 根据所述资源加载请求返回的指示；以及，根据所述指示，基于所述进程单元预加载的资源，对所述资源加载请求所请求加载的资源进行加载；其中，所述进程单元预加载的模块中包括 Web 引擎。

可选地，所述进程对应的事件队列中包括所述 Web 引擎对应的子队列，所述子队列中包括 Web 事件请求。事件调度单元 703 用于：从所述事件队列中获取待处理的事件请求，若获取到所述子队列，则按照时间先后顺序获取所述子队列中的 Web 事件请求，并将获取到的 Web 事件请求发送给所述第一线程单元。第一线程单元 701 具体用于：接收到 Web 事件请求后向所述第二线程单元 702 发送资源加载请求。

可选地，所述进程对应的事件队列中还包括动态语言应用运行平台事件请求。事件调度单元 703 还用于：所述子队列中的 Web 事件请求处理完成后，返回到所述事件队列，并获取待处理的动态语言应用运行平台事件请求；或者，所述子队列中的设定数量的 Web 事件请求处理完成后，返回到所述事件队列，并获取待处理的动态语言应用运行平台事件请求。

可选地，所述进程单元具体用于：启动时进行资源的预加载，所预加载的资源中包括所述 Web 引擎，以及包括动态语言应用运行平台所提供的模块、操作系统提供的模块、自定义的模块中的一种或多种组合；其中，所述模块通过对实现特定功能的代码进行封装得到。

可选地，所述进程单元具体用于：启动时进行资源的预加载，所预加载的资源中包括动态语言上下文中包括 Web 引擎提供的模块的接口，以及所述进程预加载的模块的接口，所述进程预加载的模块包括：动态语言应用运行平台所提供的模块、操作系统提供

的模块、自定义的模块中的一种或多种组合。

可选地，第一线程单元 701 具体用于：根据所述 Web 引擎提供的接口，调用相应的模块，被调用的模块用于加载所请求加载的资源；通过所述进程预加载的动态语言引擎对该模块进行解析，得到该模块所调用的操作系统中的模块接口；根据解析得到的模块接口调用操作系统中的相应模块。

可选地，所述第一线程单元具体用于：根据 web 页面访问请求，向第二线程发送资源加载请求，所述资源加载请求用于请求加载所请求的 web 页面的资源。

可选地，所述第一线程单元为所在进程中的主线程单元，所述第二线程单元为所在进程中的子线程单元。

可选地，所述动态语言应用运行平台为 Node.js，所述第一线程单元为 Node.js 线程单元，所述第二线程单元为浏览器线程单元。

基于相同的技术构思，本申请实施例还提供了一种装置 800，该装置 800 可实现前述实施例描述的流程。

图 8 示例性地示出了根据各种实施例的示例装置 800。装置 800 可包括一个或多个处理器 802，系统控制逻辑 801 耦合于至少一个处理器 802，非易失性存储器（non-volatile memory，NMV）/存储器 804 耦合于系统控制逻辑 801，网络接口 806 耦合于系统控制逻辑 801。

处理器 802 可包括一个或多个单核处理器或多核处理器。处理器 802 可包括任何一般用途处理器或专用处理器（如图像处理器、应用处理器基带处理器等）的组合。

一个实施例中的系统控制逻辑 801，可包括任何适当的接口控制器，以提供到处理器 802 中的至少一个的任何合适的接口，和/或提供到与系统控制逻辑 801 通信的任何合适的设备或组件的任何合适的接口。

一个实施例中的系统控制逻辑 801，可包括一个或多个内存控制器，以提供到系统内存 803 的接口。系统内存 803 用来加载以及存储数据和/或指令。例如，对应装置 800，在一个实施例中，系统内存 803 可包括任何合适的易失性存储器。

NVM/存储器 804 可包括一个或多个有形的非暂时的计算机可读介质，用于存储数据和/或指令。例如，NVM/存储器 804 可包括任何合适的非易失性存储装置，如一个或多个硬盘（hard disk device，HDD），一个或多个光盘（compact disk，CD），和/或一个或多个数字通用盘（digital versatile disk，DVD）。

NVM/存储器 804 可包括存储资源，该存储资源物理上是该系统所安装的或者可以被

访问的设备的一部分，但不一定是设备的一部分。例如，NVM/存储器 804 可经由网络接口 806 被网络访问。

系统内存 803 以及 NVM/存储器 804 可分别包括临时的或持久的指令 810 的副本。指令 810 可包括当由处理器 802 中的至少一个执行时导致装置 800 实现图 5a、图 5b 描述的方法之一或组合的指令。各实施例中，指令 810 或硬件、固件，和/或软件组件可另外地/可替换地被置于系统控制逻辑 801，网络接口 806 和/或处理器 802。

网络接口 806 可包括一个接收器来为装置 800 提供无线接口来与一个或多个网络和/或任何合适的设备进行通信。网络接口 806 可包括任何合适的硬件和/或固件。网络接口 806 可包括多个天线来提供多输入多输出无线接口。在一个实施例中，网络接口 806 可包括一个网络适配器、一个无线网络适配器、一个电话调制解调器，和/或无线调制解调器。

在一个实施例中，处理器 802 中的至少一个可以与用于系统控制逻辑的一个或多个控制器的逻辑一起封装。在一个实施例中，处理器中的至少一个可以与用于系统控制逻辑的一个或多个控制器的逻辑一起封装以形成系统级封装。在一个实施例中，处理器中的至少一个可以与用于系统控制逻辑的一个或多个控制器的逻辑集成在相同的管芯上。在一个实施例中，处理器中的至少一个可以与用于系统控制逻辑的一个或多个控制器的逻辑集成在相同的管芯上以形成系统芯片。

装置 800 可进一步包括输入/输出装置 805。输入/输出装置 805 可包括用户接口旨在使用户与装置 800 进行交互，可包括外围组件接口，其被设计为使得外围组件能够与系统交互，和/或，可包括传感器，旨在确定环境条件和/或有关装置 800 的位置信息。

权 利 要 求 书

1.一种资源加载方法，其特征在于，包括：

第一线程向第二线程发送资源加载请求；其中，所述第一线程和所述第二线程位于同一进程，所述第一线程运行于动态语言应用运行平台；

所述第一线程接收所述第二线程根据所述资源加载请求返回的指示；

所述第一线程根据所述指示，基于所述进程预加载的资源，对所述资源加载请求所请求加载的资源进行加载，其中，所述进程预加载的资源中包括 Web 引擎。

2.如权利要求 1 所述的方法，其特征在于，所述进程对应的事件队列中包括所述 Web 引擎对应的子队列，所述子队列中包括 Web 事件请求；

所述第一线程向所述第二线程发送资源加载请求，包括：

所述第一线程从所述事件队列中获取待处理的事件请求；

若获取到所述子队列，则按照时间先后顺序获取所述子队列中的 Web 事件请求，并在获取到 Web 事件请求后向所述第二线程发送资源加载请求。

3.如权利要求 2 所述的方法，其特征在于，所述进程对应的事件队列中还包括动态语言应用运行平台事件请求；

所述方法还包括：

所述子队列中的 Web 事件请求处理完成后，返回到所述事件队列，并获取待处理的动态语言应用运行平台事件请求；或者，

所述子队列中的设定数量的 Web 事件请求处理完成后，返回到所述事件队列，并获取待处理的动态语言应用运行平台事件请求。

4.如权利要求 1 所述的方法，其特征在于，还包括：

所述进程启动时进行资源的预加载，所预加载的资源中包括所述 Web 引擎，以及包括动态语言应用运行平台所提供的模块、操作系统提供的模块、自定义的模块中的一种或多种组合；其中，所述模块通过对实现特定功能的代码进行封装得到。

5.如权利要求 1 所述的方法，其特征在于，所述对所述资源加载请求所请求加载的资源进行加载，包括：

根据所述 Web 引擎提供的接口调用相应的模块，被调用的模块用于加载所请求加载的资源；

通过所述进程预加载的动态语言引擎对该模块进行解析，得到该模块所调用的操作系统中的模块接口；

根据解析得到的模块接口调用操作系统中的相应模块。

6.如权利要求 1 至 5 任一项所述的方法，其特征在于，所述第一线程向第二线程发送资源加载请求，包括：

所述第一线程根据 web 页面访问请求，向第二线程发送资源加载请求，所述资源加载请求用于请求加载所请求的 web 页面的资源。

7.如权利要求 1 至 5 中任一项所述的方法，其特征在于，所述第一线程为所在进程中的主线程，所述第二线程为所在进程中的子线程。

8.如权利要求 1 至 5 中任一项所述的方法，其特征在于，所述动态语言应用运行平台为 Node.js，所述第一线程为 Node.js 线程，所述第二线程为浏览器线程。

9.一种资源加载装置，其特征在于，包括：第一线程单元和第二线程单元；

所述第一线程单元，用于：

向所述第二线程单元发送资源加载请求；其中，所述第一线程单元和所述第二线程单元属于同一进程单元，所述第一线程单元运行于动态语言应用运行平台；

接收所述第二线程单元根据所述资源加载请求返回的指示；以及，

根据所述指示，基于所述进程单元预加载的资源，对所述资源加载请求所请求加载的资源进行加载；其中，所述进程单元预加载的模块中包括 Web 引擎。

10.如权利要求 9 所述的装置，其特征在于，所述进程对应的事件队列中包括所述 Web 引擎对应的子队列，所述子队列中包括 Web 事件请求；

还包括：事件调度单元，所述事件调度单元用于：

从所述事件队列中获取待处理的事件请求；

若获取到所述子队列，则按照时间先后顺序获取所述子队列中的 Web 事件请求，并将获取到的 Web 事件请求发送给所述第一线程单元；

所述第一线程单元具体用于：接收到 Web 事件请求后向所述第二线程单元发送资源加载请求。

11.如权利要求 10 所述的装置，其特征在于，所述进程对应的事件队列中还包括动态语言应用运行平台事件请求；

所述事件调度单元还用于：

所述子队列中的 Web 事件请求处理完成后，返回到所述事件队列，并获取待处理的动态语言应用运行平台事件请求；或者，

所述子队列中的设定数量的 Web 事件请求处理完成后，返回到所述事件队列，并

获取待处理的动态语言应用运行平台事件请求。

12.如权利要求 9 所述的装置，其特征在于，所述进程单元具体用于：

启动时进行资源的预加载，所预加载的资源中包括所述 Web 引擎，以及包括动态语言应用运行平台所提供的模块、操作系统提供的模块、自定义的模块中的一种或多种组合；其中，所述模块通过对实现特定功能的代码进行封装得到。

13.如权利要求 9 所述的装置，其特征在于，所述第一线程单元具体用于：

根据所述 Web 引擎提供的接口调用相应的模块，被调用的模块用于加载所请求加载的资源；

通过所述进程预加载的动态语言引擎对该模块进行解析，得到该模块所调用的操作系统中的模块接口；

根据解析得到的模块接口调用操作系统中的相应模块。

14.如权利要求 9 至 13 任一项所述的装置，其特征在于，所述第一线程单元具体用于：根据 web 页面访问请求，向第二线程发送资源加载请求，所述资源加载请求用于请求加载所请求的 web 页面的资源。

15.如权利要求 9 至 13 中任一项所述的装置，其特征在于，所述第一线程单元为所在进程中的主线程单元，所述第二线程单元为所在进程中的子线程单元。

16.如权利要求 9 至 13 中任一项所述的装置，其特征在于，所述动态语言应用运行平台为 Node.js，所述第一线程单元为 Node.js 线程单元，所述第二线程单元为浏览器线程单元。

17.一个或多个计算机可读介质，所述可读介质上存储有指令，所述指令被一个或多个处理器执行时，使得通信设备执行如权利要求 1 至 8 中任一项所述的方法。

18.一种通信设备，其特征在于，包括：

一个或多个处理器；以及

一个或多个计算机可读介质，所述可读介质上存储有指令，所述指令被所述一个或多个处理器执行时，使得所述设备执行如权利要求 1 至 8 中任一项所述的方法。

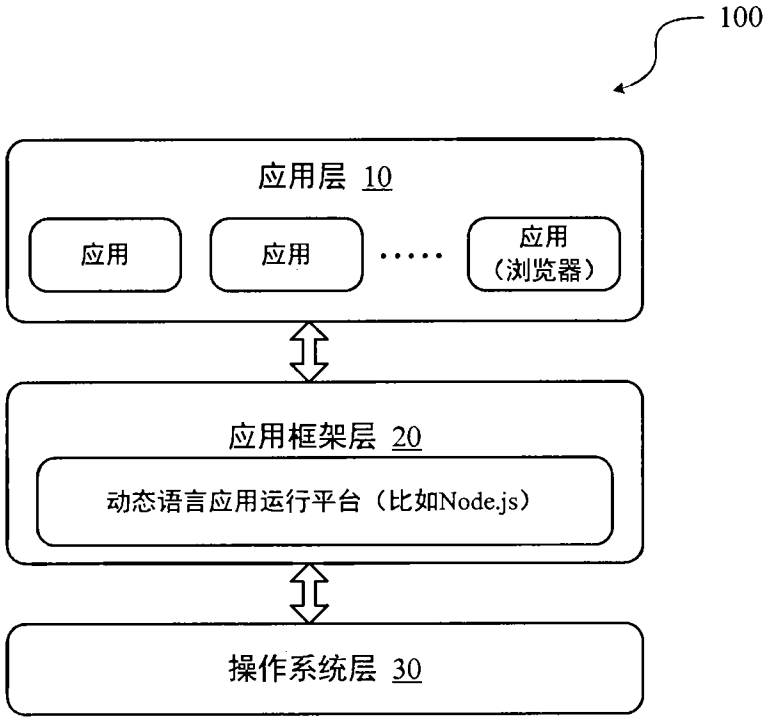
100
应用层 10
应用
应用
.....
应用
（浏览器）
应用框架层 20
动态语言应用运行平台（比如Node.js）
操作系统层 30

图 1

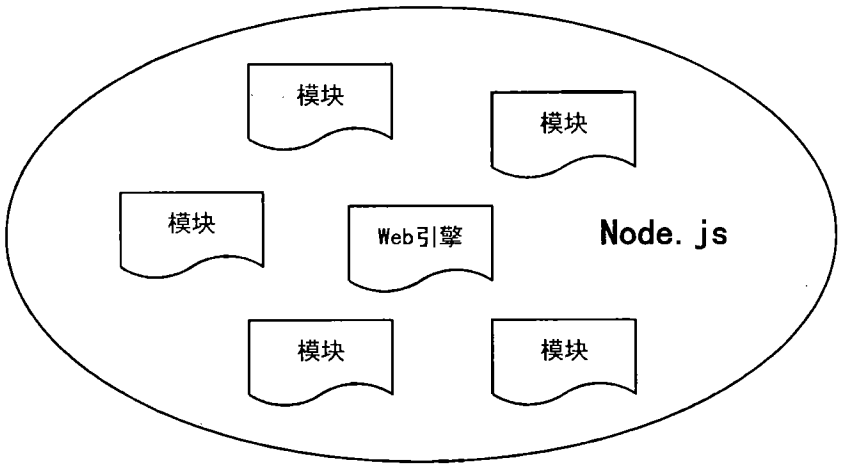
模块
模块
模块
Web引擎
Node. js
模块
模块

图 2

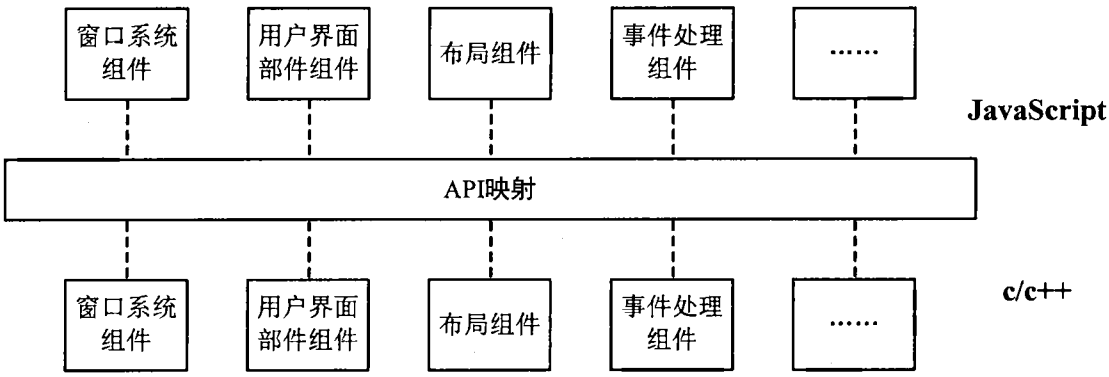
窗口系统组件
用户界面部件组件
布局组件
事件处理组件
.....
JavaScript
API映射
窗口系统组件
用户界面部件组件
布局组件
事件处理组件
.....
c/c++

图 3

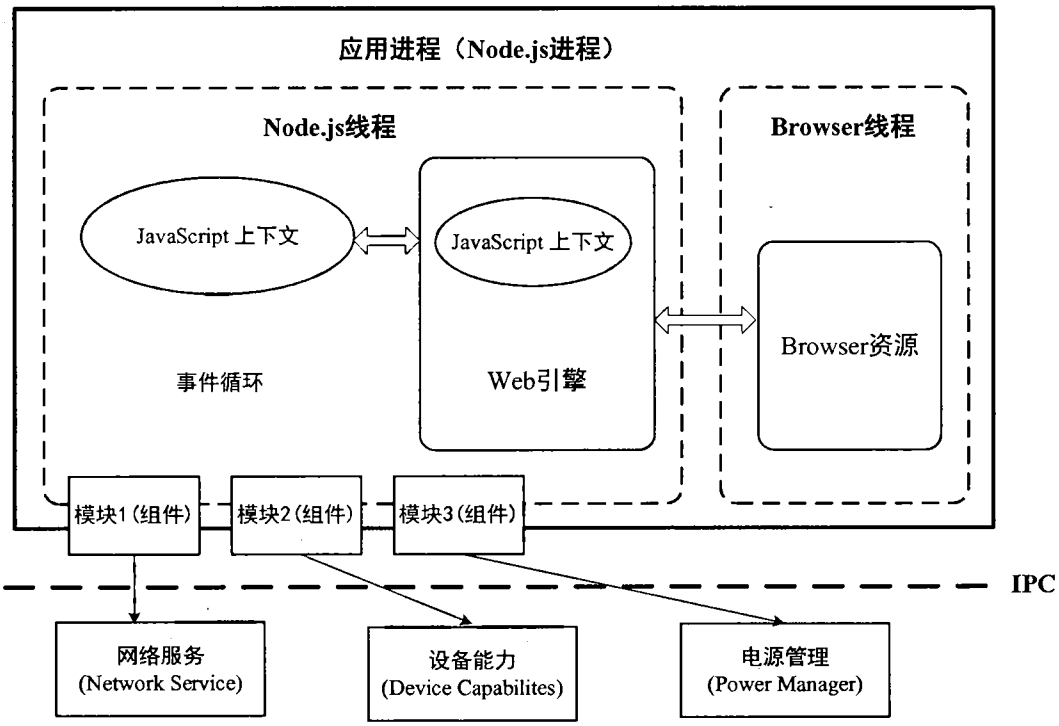
应用进程（Node.js进程）
Node.js线程
Browser线程
JavaScript 上下文
JavaScript 上下文
Browser资源
事件循环
Web引擎
模块1(组件)
模块2(组件)
模块3(组件)
IPC
网络服务
(Network Service)
设备能力
(Device Capabilites)
电源管理
(Power Manager)

图 4

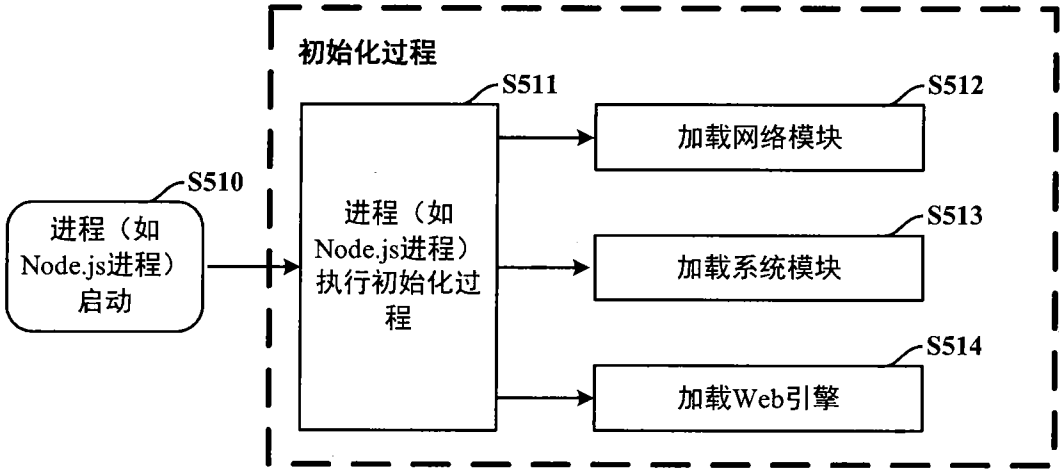
初始化过程
S510
进程（如Node.js进程）启动
S511
进程（如Node.js进程）执行初始化过程
S512
加载网络模块
S513
加载系统模块
S514
加载Web引擎

图 5a

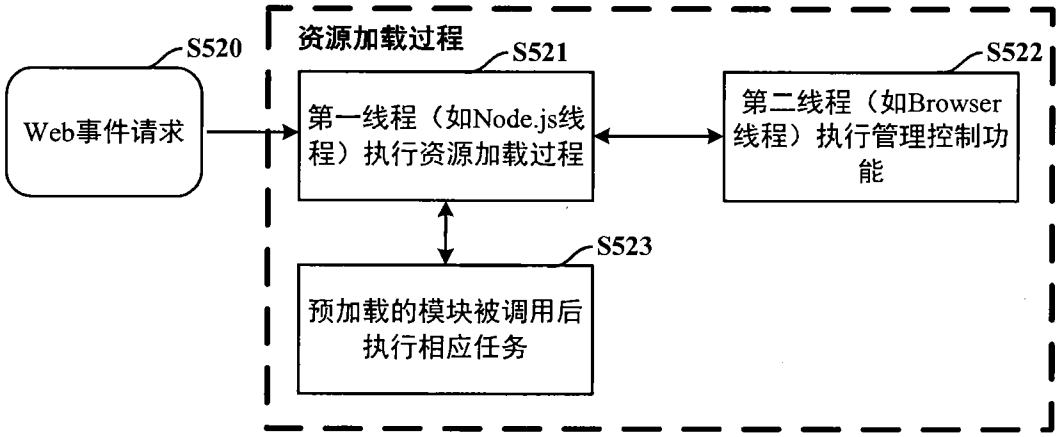
资源加载过程
S520
Web事件请求
S521
第一线程（如Node.js线程）执行资源加载过程
S522
第二线程（如Browser线程）执行管理控制功能
S523
预加载的模块被调用后执行相应任务

图 5b

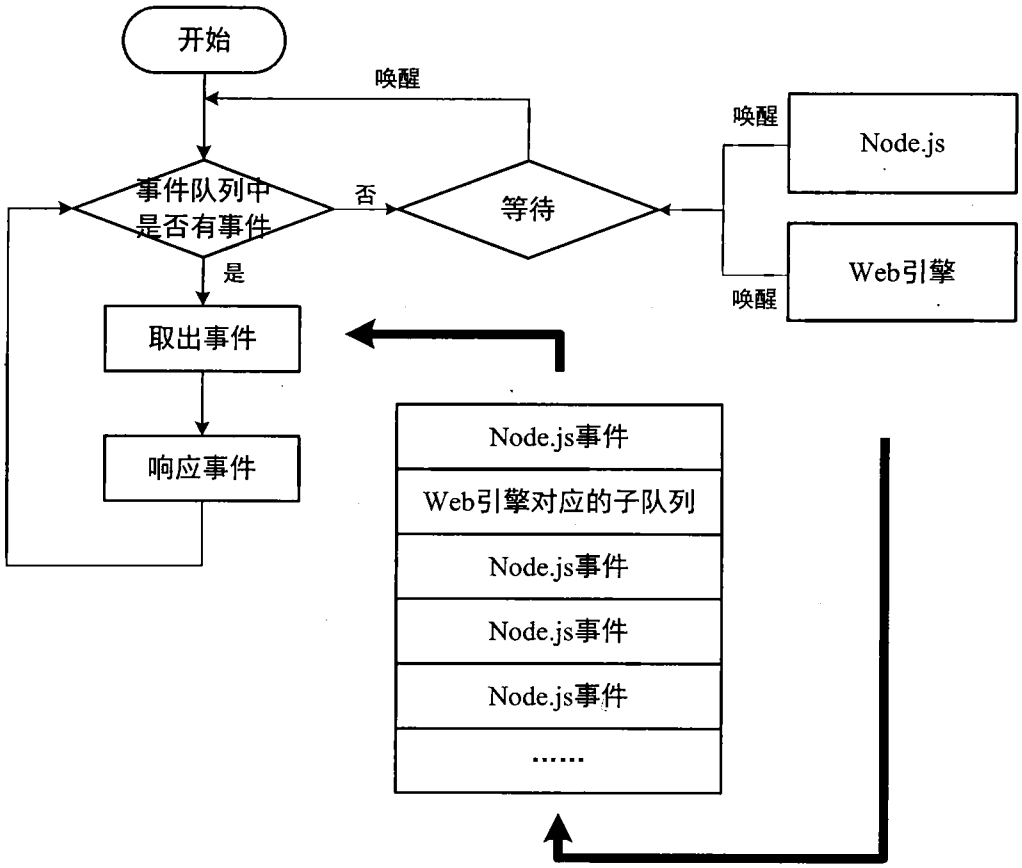
开始
唤醒
事件队列中
是否有事件
否
等待
唤醒
Node.js
唤醒
Web引擎
是
取出事件
响应事件
Node.js事件
Web引擎对应的子队列
Node.js事件
Node.js事件
Node.js事件
.....

图 6

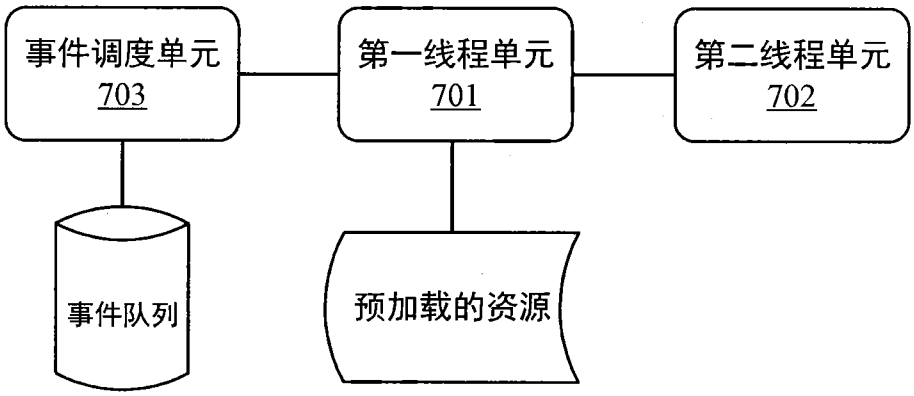
事件调度单元
703
第一线程单元
701
第二线程单元
702
事件队列
预加载的资源

图 7

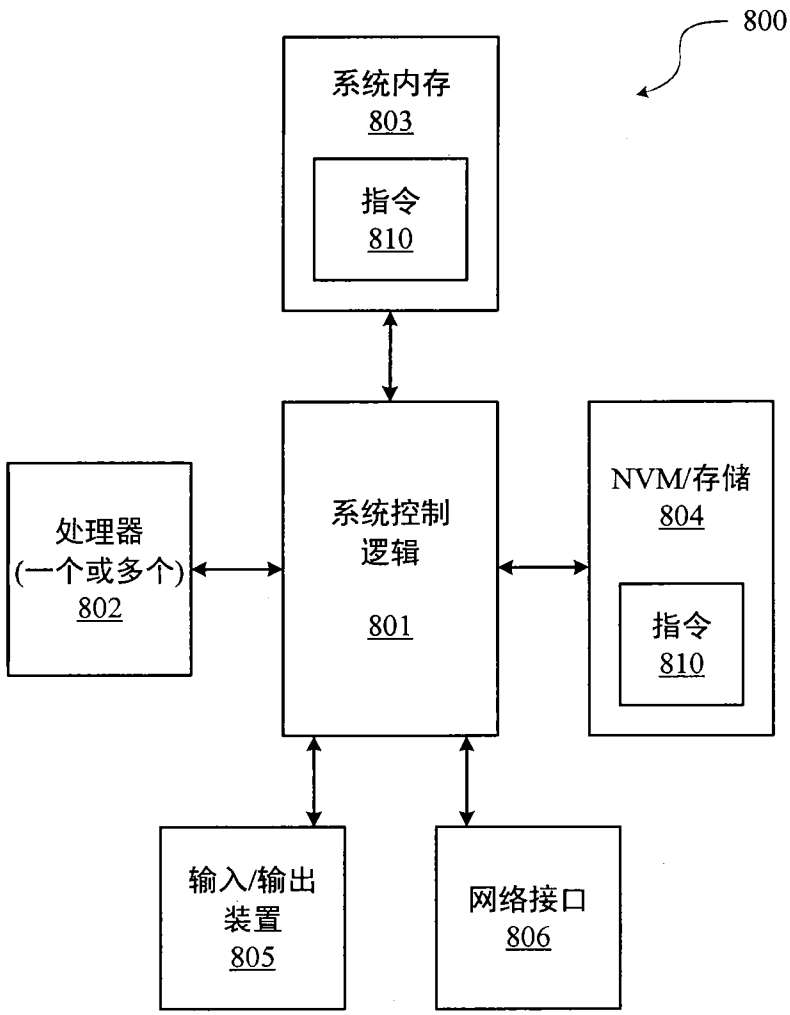
800
系统内存
803
指令
810
处理器
(一个或多个)
802
系统控制
逻辑
801
NVM/存储
804
指令
810
输入/输出
装置
805
网络接口
806

图 8

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2018/077220

A. CLASSIFICATION OF SUBJECT MATTER

G06F 17/30 (2006.01) i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 17/30

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CPRSABS; CNTXT; CNKI; DWPI: WEB, 主线程, 子线程, 资源, 预加载, master thread, sub thread, resources, pre load

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	CN 105701246 A (QINGDAO HISENSE ELECTRIC CO., LTD.), 22 June 2016 (22.06.2016), description, paragraphs 20-31 and 40-42	1-18
Y	CN 105893446 A (ADOBE SYSTEMS INC.), 24 August 2016 (24.08.2016), description, paragraphs 51-52	1-18
A	CN 105912689 A (ZHENGZHOU XIZHI INFORMATION TECHNOLOGY CO., LTD.), 31 August 2016 (31.08.2016), entire document	1-18
A	CN 105607895 A (ALIBABA GROUP HOLDING LIMITED), 25 May 2016 (25.05.2016), entire document	1-18

☐ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
11 May 2018	30 May 2018
Name and mailing address of the ISA State Intellectual Property Office of the P. R. China No. 6, Xitucheng Road, Jimenqiao Haidian District, Beijing 100088, China Facsimile No. (86-10) 62019451	Authorized officer YAN, Shiying Telephone No. 62411767

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2018/077220

Patent Documents referred in the Report	Publication Date	Patent Family	Publication Date
CN 105701246 A	22 June 2016	None	
CN 105893446 A	24 August 2016	DE 102016001467 A1	18 August 2016
		GB 201602643 D0	30 March 2016
		AU 2015271945 A1	01 September 2016
		GB 2536354 A	14 September 2016
		US 2016239468 A1	18 August 2016
CN 105912689 A	31 August 2016	None	
CN 105607895 A	25 May 2016	HK 1224031 A0	11 August 2017

国际检索报告

国际申请号
PCT/CN2018/077220

A. 主题的分类

G06F 17/30(2006.01)i

按照国际专利分类(IPC)或者同时按照国家分类和IPC两种分类

B. 检索领域

检索的最低限度文献(标明分类系统和分类号)

G06F17/30

包含在检索领域中的除最低限度文献以外的检索文献

在国际检索时查阅的电子数据库(数据库的名称，和使用的检索词（如使用））

CPRSABS;CNTXT;CNKI;DWPI:WEB,主线程，子线程，资源，预加载，master thread ,sub thread,resourecs,pre load

C. 相关文件

类 型*	引用文件，必要时，指明相关段落	相关的权利要求
Y	CN 105701246 A (青岛海信电器股份有限公司) 2016年 6月 22日 (2016-06-22) 说明书第20-31，40-42段	1-18
Y	CN 105893446 A (奥多比公司) 2016年 8月 24日 (2016-08-24) 说明书第51-52段	1-18
A	CN 105912689 A (郑州悉知信息科技股份有限公司) 2016年 8月 31日 (2016-08-31) 全文	1-18
A	CN 105607895 A (阿里巴巴集团控股有限公司) 2016年 5月 25日 (2016-05-25) 全文	1-18

☐ 其余文件在C栏的续页中列出。

☑ 见同族专利附件。

* 引用文件的具体类型：

“A” 认为不特别相关的表示了现有技术一般状态的文件

“E” 在国际申请日的当天或之后公布的在先申请或专利

“L” 可能对优先权要求构成怀疑的文件，或为确定另一篇引用文件的公布日而引用的或者因其他特殊理由而引用的文件（如具体说明的）

“O” 涉及口头公开、使用、展览或其他方式公开的文件

“P” 公布日先于国际申请日但迟于所要求的优先权日的文件

“T” 在申请日或优先权日之后公布，与申请不相抵触，但为了理解发明之理论或原理的在后文件

“X” 特别相关的文件，单独考虑该文件，认定要求保护的发明不是新颖的或不具有创造性

“Y” 特别相关的文件，当该文件与另一篇或者多篇该类文件结合并且这种结合对于本领域技术人员为显而易见时，要求保护的发明不具有创造性

“&” 同族专利的文件

国际检索实际完成的日期	国际检索报告邮寄日期
2018年 5月 11日	2018年 5月 30日

ISA/CN的名称和邮寄地址	受权官员
中华人民共和国国家知识产权局(ISA/CN) 中国北京市海淀区蓟门桥西土城路6号 100088	颜世莹
传真号 (86-10)62019451	电话号码 62411767

国际检索报告
关于同族专利的信息

国际申请号
PCT/CN2018/077220

检索报告引用的专利文件			公布日（年/月/日）	同族专利			公布日（年/月/日）
CN	105701246	A	2016年 6月 22日	无			
CN	105893446	A	2016年 8月 24日	DE	102016001467	A1	2016年 8月 18日
				GB	201602643	D0	2016年 3月 30日
				AU	2015271945	A1	2016年 9月 1日
				GB	2536354	A	2016年 9月 14日
				US	2016239468	A1	2016年 8月 18日
CN	105912689	A	2016年 8月 31日	无			
CN	105607895	A	2016年 5月 25日	HK	1224031	A0	2017年 8月 11日