(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 102866884 B
(45)授权公告日 2016.09.28

(21)申请号 201210317710.8

(22)申请日 2012.08.31

(65)同一申请的已公布的文献号
申请公布号 CN 102866884 A

(43)申请公布日 2013.01.09

(30)优先权数据
13/223296 2011.08.31 US

(73)专利权人 微软技术许可有限责任公司
地址 美国华盛顿州

(72)发明人 H.皮尔森 B.雷克托尔 M.洛夫尔 M.普拉克里亚 S.罗维 T.巴苏 R.A.弗洛达茨科 E.H.奥米亚 J.杜尼茨 A.霍尔塞克 L.W.奥斯特曼 曾炜 N.沃瓦 S.索尔卡 M.阿克西安金

(74)专利代理机构 中国专利代理(香港)有限公司 72001
代理人 李舒 汪扬

(51)Int.Cl.
G06F 9/44(2006.01)

(56)对比文件
US 2009/0199220 A1,2009.08.06,
US 6446137 B1,2002.09.03,
CN 1513145 A,2004.07.14,
CN 101233488 A,2008.07.30,

审查员 章媛

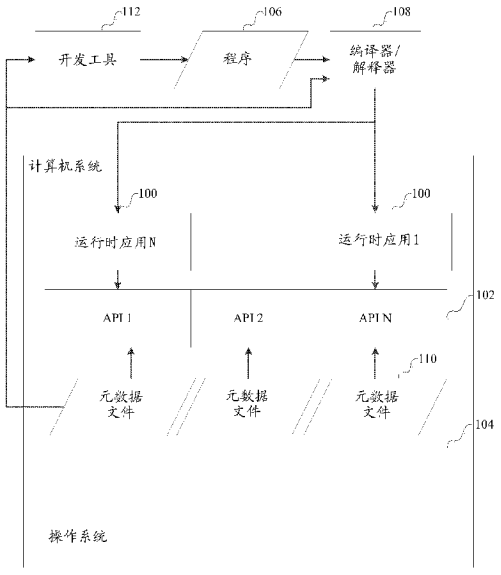
权利要求书1页 说明书8页 附图4页

(54)发明名称

将本机应用编程接口投射到其他编程语言的方法和设备

(57)摘要

有关操作系统应用编程接口的信息以已知的格式被存储在已知的位置。这个信息充分描述由操作系统显露的API，并且被存储在API元数据文件中。语言编译器或解释器使用这个API信息来用目标语言构建本机系统API的自然的和熟悉的表示。语言编译器或解释器可以在编译时和/或运行时间读取所述API信息。元数据被使用来允许应用引用API中的命名元素。构建了投射，所述投射使用元数据来把API中的命名元素映射到目标语言中的命名元素，以及定义包装器，所述包装器在目标表示与本机操作系统表示之间对那些元素的数据编组。



CN 102866884 B

1.一种用于将操作系统的应用编程接口投射到目标编程语言的计算机实施的方法，包括：

将描述操作系统(104)的应用编程接口(102)的元数据(110)接收进存储器中；

接收(300)引用所述操作系统的应用编程接口之一的元素的、在按照目标编程语言的程序中的命名元素的指示；以及

使用该元数据，创建在按照目标编程语言的程序中的投射元素，该投射元素是该操作系统的应用编程接口之一的该元素到(304)该程序中的该命名元素的投射，使得在运行时间，该程序中的该投射元素允许程序访问该操作系统的应用编程接口。

2.权利要求1的计算机实施的方法，其中创建投射元素包括：当程序被编译时，产生创建一个或多个投射元素的代码。

3.权利要求1的计算机实施的方法，其中创建投射元素包括：当程序被解释时，产生创建一个或多个投射元素的代码。

4.权利要求1的计算机实施的方法，其中投射元素按照类型在操作系统表示与应用表示之间对数据编组。

5.权利要求1的计算机实施的方法，其中投射元素包括接口，所述接口包括方法、属性和事件。

6.一种用于实施在操作系统的应用编程接口与按照目标编程语言的应用之间的语言投射的计算机器，包括：

用于将描述该操作系统的应用编程接口的元数据接收进存储器中的装置；以及

用于接收引用所述操作系统的应用编程接口之一的元素的、在按照目标编程语言的程序中的命名元素的指示的装置，以及

用于使用所述元数据，创建在按照目标编程语言的程序中的投射元素的装置，该投射元素是该操作系统的应用编程接口之一的该元素到(304)该程序中的该命名元素的投射，使得在运行时间，该程序中的该投射元素允许程序访问该操作系统的应用编程接口。

7.权利要求6的计算机器，其中创建投射元素包括：在编译时，产生创建一个或多个投射元素的代码。

8.权利要求6的计算机器，其中创建投射元素包括：当程序被解释时，产生创建一个或多个投射元素的代码。

9.权利要求7的计算机器，其中投射元素按照类型在操作系统表示与应用表示之间对数据编组。

10.权利要求7的计算机器，其中投射元素包括接口，所述接口包括方法、属性和事件。

将本机应用编程接口投射到其他编程语言的方法和设备

背景技术

[0001] 操作系统典型地具有几个应用编程接口，这些应用编程接口允许应用访问由操作系统支持的功能性。这样的API典型地由操作系统通过使用某种计算机编程语言中的命名文件或对象而被规定。例如，C编程语言使用可能具有诸如“interface.h”那样的名字的首标文件。同样地，在C#中，称为“P/Invoke”签名的机制被使用来访问操作系统API。编写将利用操作系统API的计算机程序的人典型地在程序中包括对命名的 API文件或对象的引用，或使用由编程语言提供的另一种机制。该程序于是例如包括按照由该API使用的语法的、对由该API定义的函数的调用(call)。

[0002] 以这种方式定义的API不能由与它们被编写的语言不同的语言直接访问。为了使得用其他语言编写的程序可访问API， API被“包装(wrap)”。这种包装典型地必须按API和按语言来人工地完成，并需要深入理解目标语言和API以及操作系统。因此，许多操作系统API是无法使用的。

发明内容

[0003] 本概要被提供来以简化的形式介绍概念的选择，这些概念在下面的详细说明中被进一步描述。本概要既不打算确认所要求保护的主题的关键特征或必要特征，也不打算被使用来限制所要求保护的主题的范围。

[0004] 当构建操作系统时，关于API的信息被生成并被以已知的格式存储在操作系统内的已知位置。这个信息充分描述由操作系统所显露的(expose)所有的API。这包括(但不限于)各种各样类型的关于API的命名元素的信息，诸如基本类型、枚举类型、结构、委托(delegate)、接口、类(class)、方法、属性和事件。这个信息被存储在API元数据文件中。

[0005] 语言编译器或解释器使用这个API信息来用目标语言构建本机(native)系统API的自然的和熟悉的表示。这个表示随不同的语言而变化(因为所谓的自然的和熟悉的东西是随不同的语言而变化的)。语言编译器或解释器可以在编译时和/或运行时间读取API信息，什么时间对所讨论的语言最适当就在什么时间。例如，类似C++那样的静态编译的语言将仅仅在编译时消费(consume)元数据，而类似Python或JavaScript那样的动态语言将仅仅在运行时间消费元数据。类似.NET或Java那样的环境多半既在编译又在运行时消费元数据。元数据被使用来允许应用引用在API中的命名的元素。构建了投射(projection)，所述投射使用元数据来把API中的命名元素映射为在目标语言中的命名元素，以及定义包装器，所述包装器对在目标表示和本机操作系统表示之间的那些元素的数据进行编组(marshal)。

[0006] 因此，一方面，描述操作系统的应用编程接口的元数据被存储在存储器中。假设有引用了应用编程接口之一的元素的、在按照某个编程语言的程序中的命名元素的指示，该命名元素通过使用元数据被投射到该编程语言中。投射可以在程序的编译或解释期间出现。投射可包括产生创建该编程语言中的一个或多个元素的代码，以及按照类型来对用于所创建的元素的数据编组。接口——包括它们的方法、属性和事件——也可以被如此投射。

投射还可以包括将异常(exception)从操作系统传播到应用。

[0007] 操作系统API到其他语言的这样的投射可以在计算机实施的过程中,在包括一个或多个计算机存储媒体的制造品中,或在计算机器中被体现。

[0008] 在以下的说明中,参考了形成本说明的一部分的附图,且在图上,作为举例说明,显示了本技术的具体的示例性实现。应当理解,可以利用其它实施例以及可以做出结构的改变而不背离本公开内容的范围。

附图说明

[0009] 图1是包括API到其它编程语言的投射的系统的框图。

[0010] 图2是图示开发工具的示例性操作的流程图。

[0011] 图3是图示编译器或解释器的示例性操作的流程图。

[0012] 图4是可以在其中实施这样的系统的示例性计算设备的框图。

具体实施方式

[0013] 下面的段落提供在其中可以实施本机系统API到其它语言的这样的投射的示例性操作环境。

[0014] 参照图1,运行的应用100在运行时间期间访问操作系统104的本机系统API 102。为了使这样的应用具有这样的功能性,典型地通过使用诸如编辑器那样的开发工具112来编写程序106。这样的程序由语言的编译器或解释器108进行编译或解释,以便提供运行时应用100。开发工具112和编译器或解释器108访问充分描述操作系统的API 102的元数据110。开发工具112帮助开发者编写程序,但经由元数据把可用的本机系统API通知开发者,且允许通过使用开发者的编程语言访问那些API。编译器或解释器通过使用元数据来具体化本机系统API到开发者的编程语言的投射。具体地,在API中的命名元素被映射到目标语言中的命名元素,以及在那些元素中的值在由目标语言与操作系统所使用的格式之间进行编组。

[0015] 构建这样的系统是从构建具有用元数据描述的API的操作系统开始的。元数据以与编程语言无关的形式表示API描述的每个命名的元素。这个元数据提供接口的完整的描述。组合的系统元数据可以以ECMA-335 CLI格式被存储在一系列元数据文件中,但具体的格式对于本发明是不重要的。

[0016] 在假设这个上下文后,将结合图2-4更详细地描述这样的系统的示例性实现。

[0017] 在图2上,现在将描述开发工具的操作的例子。当编写计算机程序时,开发者通常使用诸如编辑器那样的某种形式的开发工具。这个编辑器可以完成各种各样的任务,诸如验证语法和提出使开发者输入的字符串填充完整的建议。在这种环境下,描述操作系统应用编程接口的元数据可以以各种各样的方式被使用。具体地,它可被使用来允许开发者发现可用的API。开发工具接收200 字符串输入,诸如可以在API中被用作为标识符(诸如“鼠标”)或名字空间(诸如,操作系统的名称)的关键字。元数据可以被搜索202,以便识别把接收的输入作为标识符或名字空间的一部分的元素。用于该组匹配的元素的标识符可以被收集和返回204给开发工具。开发工具可以把这些元素呈现206给开发者以供选择,并且使用来自元数据的信息,以适合于开发者正在使用的计算机语言的格式呈现这样的元素。

[0018] 现在参照图3，现在将描述语言编译器或解释器的操作的例子。在处理计算机程序时，无论是编译还是解释它，均检测300计算机程序的元素的序列。确定302计算机程序的元素是否是对操作系统的API的引用。例如，这可以通过在元数据中查找该元素而被确定。通过使用元数据，创建304投射元素，其允许在计算机程序与操作系统之间交换数据。具体地，编译器或解释器实施操作系统API中的命名元素到目标语言中的命名元素的投射。元数据允许创建对象，和在由程序使用的数据格式与由操作系统使用的格式之间对值编组。在运行时间，这个元素允许程序访问306操作系统的API，和在应用与操作系统之间对数据编组。

[0019] 在目前描述了这样的系统的总的操作后，现在将描述特定的例子。具体地，现在将描述在描述API的元数据与该API的元素的计算机语言特定的表示之间的示例性投射的更多细节。

[0020] 以下的描述仅仅是一个可能的实现，而并不被看作为是对本发明的限制。具体地，应当理解，下面仅仅是可以被实施的语言投射的例子，以及用于这种语言的其它实现是可能的，且到其他语言的其他投射是可能的。

[0021] 在本例中，JavaScript是本机系统API通过使用元数据将被投射到的编程语言。在下面的例子中，给出如何把某些种类的元素投射到JavaScript编程语言的说明。

[0022] 当脚本(script)试图创建由操作系统API定义的特定对象的实例时，投射器对象负责访问元数据以创建在不同类型之间转换数据、可能牵涉到创建代理对象的编组存根(marshal stub)，以及负责调度方法调用、管理事件和管理回调函数。

[0023] 对于是诸如整数、字符串等等那样的基本类型的命名元素，下面是把那些API元素投射到JavaScript的方式。

[0024] 操作系统具有几种带符号和不带符号整数类型，诸如一字节的不带符号整数(UInt8)、四字节的不带符号整数(UInt32)、四字节的带符号整数(Int8)、以及八字节的带符号和不带符号整数(UInt64和Int64)。这些类型的命名元素被投射为JavaScript数值(Number value)。当JavaScript数被编组到操作系统值中时，则所述值的类型被强制成JavaScript数，随后是由ES5 ToInt32规范所定义的过程。对于UInt8，结果施加了模2^8。对于Int32，施加模2^16。对于UInt32，施加模2^32。

[0025] 被编组到JavaScript值中的64比特整数，如果是带符号的，若落入[-2^53, 2^53]范围中，或如果是不带符号的，若落入 [0, 2^53]范围中，则被表示为标准的数值。如果它落在这个范围以外，则它被表示为具有定制的后备存储(backing store)的数值，该定制的后备存储保存全部64比特整数数据。对于这些定制的数值进行的数学运算使得该值被强制成处在[-2^53, 2^53]范围中的标准数字表示，或者在不带符号的情况下是处在[0, 2^53]范围中的标准数字表示。如果该值处在这个范围以外，则将引发类型错误(Type Error)。被编组成64比特整数的JavaScript值在它本身是被投射的值的情况下被直接分派；否则，传送对于该值施加EC5 “ToInteger”转换的结果。

[0026] 操作系统API的、是字符(用16比特统一代码(Unicode)表示)、字符串或全球唯一标识符(GUID)的命名元素可以被表示为JavaScript字符串，以及被投射成JavaScript 中的命名的字符串。

[0027] 被编组到JavaScript中的字符被转换成包含用统一代码值表示的单个字符的JavaScript字符串值。被编组成字符的JavaScript值的类型经由ES5 ToString操作被强制

成JavaScript字符串，且第一个字符被保留。然后该单个字符作为Char16值被传递。

[0028] 被编组到JavaScript 中的字符串被转换成JavaScript字符串。被编组成字符串的JavaScript值的类型被强制成JavaScript字符串。

[0029] 被编组成JavaScript值的GUID被转换成字符串格式。被编组成GUID的JavaScript值的类型被强制成字符串，然后被解析成由操作系统使用的格式。

[0030] 操作系统可以具有带有如下的命名元素的API，即该命名元素是表示时间点的日期时间结构体(struct)或是表示时间量的时间跨度结构体。日期时间结构可被投射到JavaScript 中作为日期实例，其具有匹配于日期时间结构体数据(它具有与日期实例不同的范围和精度)的后备存储。时间跨度结构体被转换成毫秒，并作为JavaScript数被返回。同样地，JavaScript数可以从毫秒被转换到100-纳秒单位，以作为时间跨度结构体进行传递。

[0031] 应当指出，通过解释元数据，在某些情形下，投射也可以把来自本机环境的类型重新映射成在语言投射中的现有类型。例如，当在本机API和语言投射中的类型具有可兼容的数据布局时(这对于基本数据类型可以容易地出现)，这是可能的和想要的。通过元数据映射，投射可以简单地把所有的操作，诸如在语言类型上的方法或属性，直接重新定向到本机类型。这使得那些类型的使用对于语言开发者而言是更自然和熟悉的。例如，日期时间结构重新映射可以以这样的方式被实施。在本机API中，日期时间结构被显露为在元数据中的Windows.Foundation.DataTime，而不用任何附加的操作。在C#投射中，这种类型可以通过丰富的支持，被重新定向到在C#中的System.DataTimeOffset。

[0032] 作为另一个例子，如果在API中的命名元素是方法，则它把值HRESULT作为它的返回类型，其被转换成JavaScript中的异常事件。返回的HRESULT由JavaScript引擎检验是否成功。如果HRESULT指示失败，则JavaScript引擎在JavaScript侧抛出(throw)异常事件。因此，对于启用(invoke)操作系统API的方法的JavaScript，HRESULT失败被表面化(surface)为JavaScript异常事件。对于消费JavaScript方法的API 方法(诸如，对于以下描述的回调或委托)，JavaScript方法调用被包装在异常事件块中(或者是如由宿有API的JavaScript提供的等同物)，这样使得所捕获的异常事件可作为HRESULTS被传播。然而，操作系统也允许HRESULT处在方法和属性的入或出参数位置中。在这些情形下，HRESULTS被编组为不带符号32比特值(如以上描述的)。

[0033] 对于在API中是枚举类型的命名元素(其是一组命名的常数)，这些在JavaScript中被表示为对于每个命名值包含一只读字段的对象。

[0034] 对于在API中是“结构体”的命名元素(命名的数据字段的集合)，这些在JavaScript中被表示为JavaScript对象。被编组到JavaScript中的结构体被转换成对象。在结构体中的每个命名的字段变为在JavaScript对象中的命名的属性。在结构体中的每个命名的字段的值按字段的底层(underlying)类型被编组。是对象类型的JavaScript值可以被编组到结构体中。JavaScript对象或它的原型对于结构体的每个命名的字段包含一命名字段，以及每个命名字段的值按照底层类型被编组。在JavaScript对象中的、不具有操作系统API结构体中的等同物的额外字段被忽略。如果任何结构体值的编组失败，则返回编组错误。

[0035] 在本例中，操作系统不具有用于数组的类型，而代之以允许对方法的自变量是一

个配对，即：不带符号整数长度（它表示在数组中元素的数量，而不是字节），后随指向数组的第一元素的指针。

[0036] 当把数组编组到JavaScript中时，具有以下特征的对象被创建。该对象具有针对在0与数组的长度减1之间的每个整数值的属性——它们是可枚举的、可写入的和不可配置的，以及最初被设置为向量的长度的“长度”属性，它是不可写入的、不可枚举的和不可配置的。它的原型是数组原型对象。它的对索引（index）属性的[[Put]]操作设置在底层的本机数组上规定的索引。它的对索引属性的[[GetOwnProperty]]操作索引到底层的本机数组中。它不具有作为“数组”的[[class]]。当编组JavaScript对象时，如果对象具有[[class]]“数组”，则数组被复制到本机数组中，并且传递对于这个数组的引用。如果对象是投射的数组，则传递底层的本机数组。

[0037] API还可以具有是委托或回调函数的命名元素，它是对单个可启用（invokable）的方法的引用。这些可被投射到JavaScript中作为可调用的对象。投射器将回调委托包装在定制的编组的对象中。

[0038] 被编组到JavaScript中的委托被包装在JavaScript函数对象中。当函数对象被启用（invoke）时，自变量被编组到如由委托所规定的等同的参数类型中，然后包装的委托对象被启用。如果任何自变量无法被编组，则委托调用失败。如果比起参数中的委托有更少的JavaScript自变量，则委托调用失败。超出参数中的委托以外的额外JavaScript自变量被忽略。在委托被启用后，外出的参数被编组成JavaScript类型。如果任何外出的参数无法被编组，则委托调用失败。然后，外出的参数被编组到JavaScript值，并且被返回。

[0039] 如果本机JavaScript函数对象正被编组到委托中，则那个可调用的对象被包装在对应委托类型的委托中。当委托被启用时，进入的参数被编组到JavaScript类型中，然后，JavaScript函数对象被启用。如果任何的自变量无法被编组，则委托调用失败。在委托被启用后，返回的值根据以下的规则被映射成该委托的外出的参数。首先，如果没有外出的参数，则来自JavaScript函数对象的返回值被忽略。第二，如果委托规定了单个外出的参数，则来自JavaScript函数对象的返回值被编组到该类型中。如果该编组失败，则委托调用失败。第三，如果所述委托规定多个外出参数，则来自JavaScript函数对象返回值的返回值是具有对于每个外出参数的命名属性的对象。每个命名属性被编组到对应的外出参数的类型中。如果返回值不是对象，则委托调用失败。如果返回的对象不包含对于每个外出参数的命名属性，则委托调用失败。如果返回的对象包含不对应于外出参数的额外的命名属性，则它们被忽略。如果任何的命名属性无法被编组到对应的外出参数类型，则委托调用失败。

[0040] 以下的例子图示用于被称为IString Collection的委托的元数据，和创建这个委托的实例的伪-JavaScript 代码。

[0041] 示例性元数据：

[0042] [uuid(…)]

[0043] delegate HRESULT Comparer([in] HSTRING s1,[in] HSTRING s2,

[0044] [out,retval] Boolean* value)

[0045] interface IStringCollection

[0046] {

[0047] HRESULT Sort([in] Comparer compare);

[0048] }

[0049] 伪-JavaScript

[0050] //创建实现IStringCollection的类的实例

[0051] strCol=getIStringCollection()

[0052] strCol.sort(function(s1,s2) {return s1>s2; });

[0053] 接口不是作为对象直接投射到JavaScript中。然而，接口可以是参数，并且返回操作系统API方法的类型。

[0054] 为了提供在目标编程语言中自然的投射，在以上的例子中，成员被投射到JavaScript中，使它们的名字更改为camelCase，遵循JavaScript中的强的规范，以便对于成员使用camelCase名字。类型－－类似于JavaScript构造器函数－－通常是PascalCased，并且以该形式被投射。同样地，用于addEventListener模式的枚举属性、结构体字段、事件名称在camelCase中都有名字。

[0055] 被编组到JavaScript中的、根据静态类型信息不知要作为运行时间类的表示的接口实例经历以下步骤。首先，进行该接口的调用，以得到用于该接口的运行时间类名称。如果成功的话，则对象被投射为运行时间类实例对象(下面描述的)。否则，对象就好像它是未命名的运行时间类的实例那样被投射，该实例正好实现已知要实现的接口和它的间接(transitively)需要的接口。

[0056] 检查被编组到接口类型的JavaScript值，以便弄明白它是投射的运行时间类实例对象还是投射的接口实例对象。如果它是运行时间类对象，以及该对象代理的原始值实现了接口类型，则该接口的对象实现被传递到操作系统。否则，引发类型错误异常事件。

[0057] 在操作系统API中的对象可以是运行时间类的实例。运行时间类实现一组一个或多个接口(在下面定义)。某个运行的对象的被实现接口的列表可以根据返回该运行的对象的方法的元数据或通过访问运行时间类名称并查找元数据而被确定。由于JavaScript是一种基于原型的动态语言，所以它没有类的构造。类构造在JavaScript中作为对象被投射。

[0058] 因此，操作系统API对象在JavaScript中作为对象被投射。在类的所有被实现接口上定义的方法、属性和事件的联合代表了类型成员，所述类型成员经由被投射的JavaScript对象的原型被显露为在被投射的JavaScript对象上可用的命名的属性。JavaScript语言投射的消费者可以直接访问该类的任何成员而不用关心该成员实际上是在哪个接口上被定义的。

[0059] JavaScript对象是动态的，这意味着，可以在任何时间在对象上添加或去除新的属性。被投射的对象可以支持添加新的属性和方法，只要预定义的接口成员不被废弃(override)或删除即可。在JavaScript方面，被投射的对象是可扩展的，但命名的类型成员的集合是不可配置的。被投射的对象具有原型，实例成员被定义在来自运行时间类实现的接口的成员集合上。

[0060] 如上所指出的，这样的接口具有方法、参数和事件。

[0061] 在语言投射层上的方法被实现为vtable，每种方法一个槽(slot)。有关接口的元数据提供方法名称以及参数的名称、类型和方向(入/出)。方法在JavaScript中被投射为在被投射的运行时间类或接口对象上的可调用的属性。这些属性是{Writable(可写)=false(假), Enumerable(可枚举)=true(真), Configurable(可配置)=false(假)}。当被调用

时，自变量按照它们对应的参数类型被编组，所述方法被用这些值调用。返回值被编组到JavaScript值，JavaScript对象返回作为该值。

[0062] 在语言投射层上的属性被实现为得到(get)和/或设置(set)方法。访问该属性值便调用所述得到方法，而更新该属性值则启用所述设置方法。属性可被读出或写入(即，得到和设置方法可用)，或只被读出(即，只有得到方法可用)。属性在JavaScript中被投射为属性。属性值的编组取决于底层的属性类型而如上所述地工作。

[0063] 在语言投射层上的事件被实现为添加和去除事件收听者方法。添加方法取得委托实例，并返回描述收听者的数据，而去除方法取得描述收听者的数据，并且不返回任何东西。

[0064] 在JavaScript中，有至少一个事件投射到其上的任何被投射的运行时间类或接口对象，得到被添加到它的原型的两个附加的属性，addEventListener和removeEventListener。这些属性是{Writable:false, Enumerable:true, Configurable:false}，且它们被分派给可调用的对象。

[0065] addEventListener函数取得代表要收听的事件的名称的字符串自变量、分派作为收听者的回调函数和被忽略的可选的第三参数。这个函数调用底层的add_Event方法，传递被编组的回调函数作为委托，以及把最终得到的令牌存储在由回调函数对象的参考身份锁上(key)的映像(map)中。

[0066] removeEventListener函数取得代表要从中去除收听者的事件的名称的字符串自变量、应当被去除的回调函数和被忽略的可选的第三参数。这个函数通过在所存储令牌的映像中的参考身份而查找回调函数，以及如果找到令牌，则调用底层的remove_Event方法，传递检索到的令牌。

[0067] 当事件被激发(fire)时，将启用被作为回调来传递的任何JavaScript函数对象。被传递到该函数对象的自变量将是被提供给EventHandler委托的自变量的经编组的值。

[0068] 如前文所示，通过具有用于语言的API投射层，由被存储在操作系统中的元数据所规定的操作系统API的命名元素可被使用来自动创建对象，并且在操作系统API格式与应用编程语言格式之间对数据编组。应当理解，前文仅仅是在示例性操作系统与示例性编程语言之间的投射的例子，而本发明不限于这个例子。

[0069] 在目前描述了示例性实现后，现在将描述这样的系统被设计来在其中操作的计算环境。以下的说明打算提供其中可以实施这种系统的适用的计算环境的概略的、一般的描述。所述系统可以通过许多通用或专用的计算硬件配置被实施。可以适用的熟知的计算设备的例子包括，但不限于，个人计算机、服务器计算机、手持或膝上型设备(例如，媒体播放器、笔记本电脑、蜂窝电话、个人数据助理、录音机)、多处理器系统、基于微处理器的系统、机顶盒、游戏控制台、可编程消费电子装置、网络PC、小型计算机、大型计算机、包括任何的上述系统或设备的分布式计算环境等等。

[0070] 图4图示适用的计算系统环境的例子。该计算系统环境仅仅是适用的计算环境的一个例子，且不打算用来对这样的计算环境的使用或功能性的范围提出任何限制。该计算环境不应当被解释为具有涉及在示例性操作环境中图示的任一部件或部件组合的依赖性或要求。

[0071] 参照图4，示例性计算环境包括计算机器，诸如计算机器400。在它的最基本的配置

中，计算机器400典型地包括至少一个处理单元402和存储器404。该计算设备可包括多个处理单元和/或附加的共同处理单元，诸如图形处理单元420。取决于计算设备的确切配置和类型，存储器404可以是易失性(诸如RAM)、非易失性(诸如ROM、闪存等等)或这二者的某种组合。这种最基本的配置在图4上用虚线406图示。另外，计算机器400还可以具有附加的特征/功能性。例如，计算机器400还可以包括附加的存储装置(可拆卸的和/或非可拆卸的)，包括，但不限于，磁盘或光盘或磁带。这样的附加的存储装置在图4上用可拆卸的存储装置408和非可拆卸的存储装置410图示。计算机存储媒体包括以任何方法或技术实施的、用于存储诸如计算机程序指令、数据结构、程序模块或其它数据那样的信息的易失性和非易失性的、可拆卸和非可拆卸的媒体。存储器404、可拆卸的存储装置408和非可拆卸的存储装置410都是计算机存储媒体的例子。计算机存储媒体包括，但不限于，RAM、ROM、EEPROM、闪存或其它存储器技术、CD-ROM、数字多功能盘(DVD)或其它光学存储装置、盒式磁带、磁带、磁盘存储装置或其他磁存储设备、或可被使用来存储想要的信息并可以由计算机器400访问的任何其它介质。任何这样的计算机存储媒体可以是计算机器400的一部分。

[0072] 计算机器400还可以包含通信连接412，其允许所述设备与其它设备通信。通信连接412是通信媒体的例子。通信媒体典型地以诸如载波或其它输送机制那样的调制的数据信号承载计算机程序指令、数据结构、程序模块或其它数据，并且通信媒体包括任何信息传递媒体。术语“调制的数据信号”是指这样的信号，即：该信号使它的特性中的一个或多个以这样的方式被设置或改变使得将信息编码在信号中，由此改变信号的接收设备的配置或状态。作为例子，而不是限制，通信媒体包括有线媒体，诸如有线网或直接连线的连接，以及包括无线媒体，诸如声学、RF、红外和其它无线媒体。

[0073] 计算机器400可以具有各种输入设备414，诸如显示器、键盘、鼠标、笔、照相机、触摸输入设备等等。还可以包括输出设备416，诸如扬声器、打印机等等。所有的这些设备在本领域都是熟知的，在这里不需要详细讨论。

[0074] 本系统可以在软件的一般上下文中被实施，所述软件包括由计算机器处理的计算机可执行的指令和/或计算机解释的指令，诸如是程序模块。通常，程序模块包括例行程序、程序、对象、构件、数据结构等等，它们在被处理单元处理时指令处理单元执行特定的任务或实施特定的抽象数据类型。本系统可以在分布式计算环境下被实践，其中任务是由通过通信网链接的远程处理设备执行的。在分布式计算环境中，程序模块可以位于包括存储器存储设备的本地和远端计算机存储媒体中。

[0075] 在所附权利要求的前序中的术语 “制造品”、“过程”、“机器”、和“事物的组成”打算用来把权利要求限制到这样的主题，即所述主题被认为属于由35 U.S.C.§101中的这些术语定义的可专利主题的范围内。

[0076] 这里描述的任何的或所有的前述替换实施例可以以想要的任何组合被使用来形成另外的混合实施例。应当理解，在所附权利要求中定义的主题不必限于上述的特定实现。上述的特定实现仅仅作为例子被公开。

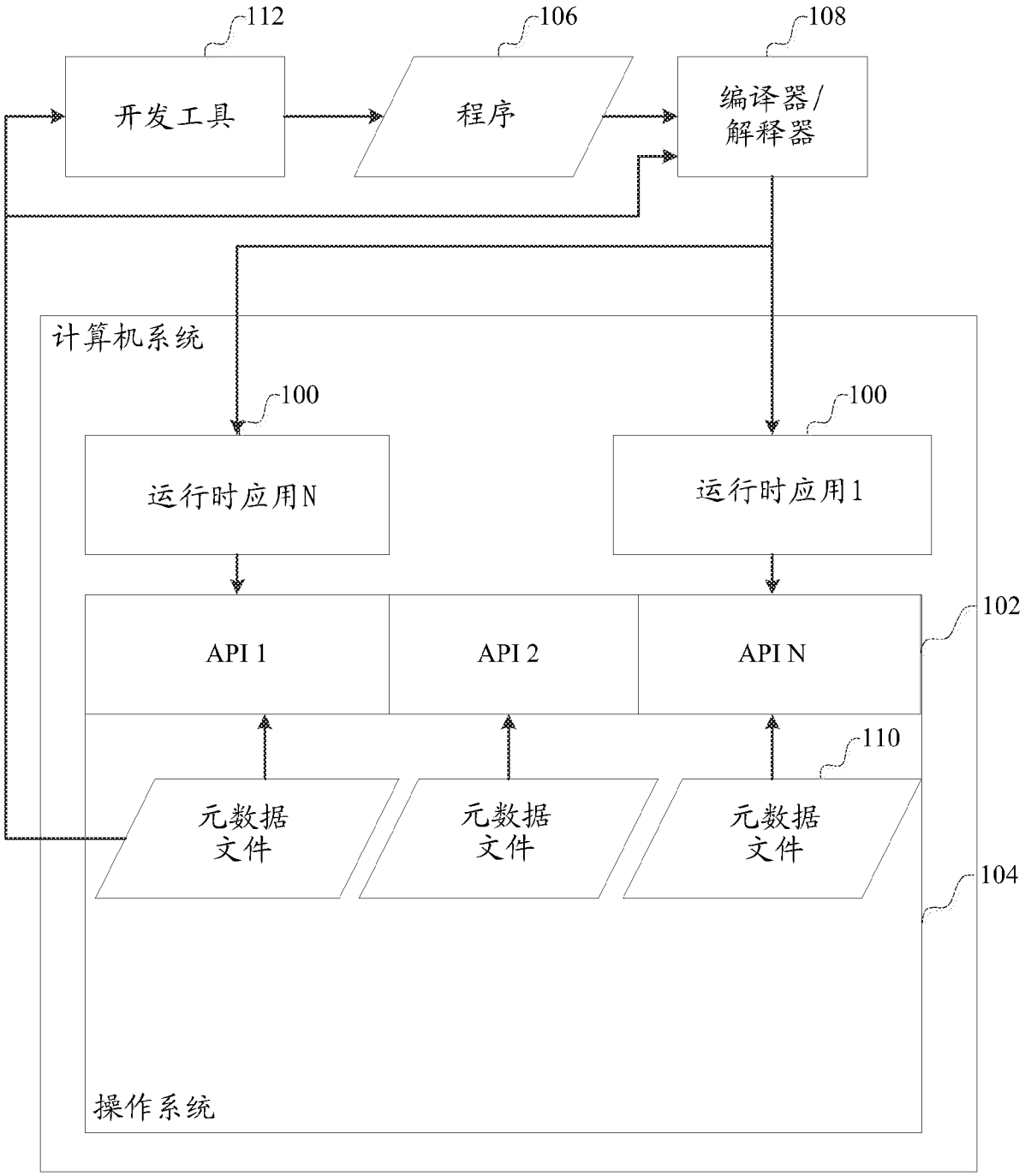

112
106
108
开发工具
程序
编译器/
解释器
计算机系统
100
100
运行时应用N
运行时应用1
API 1
API 2
API N
102
110
元数据
文件
元数据
文件
元数据
文件
104
操作系统

图 1

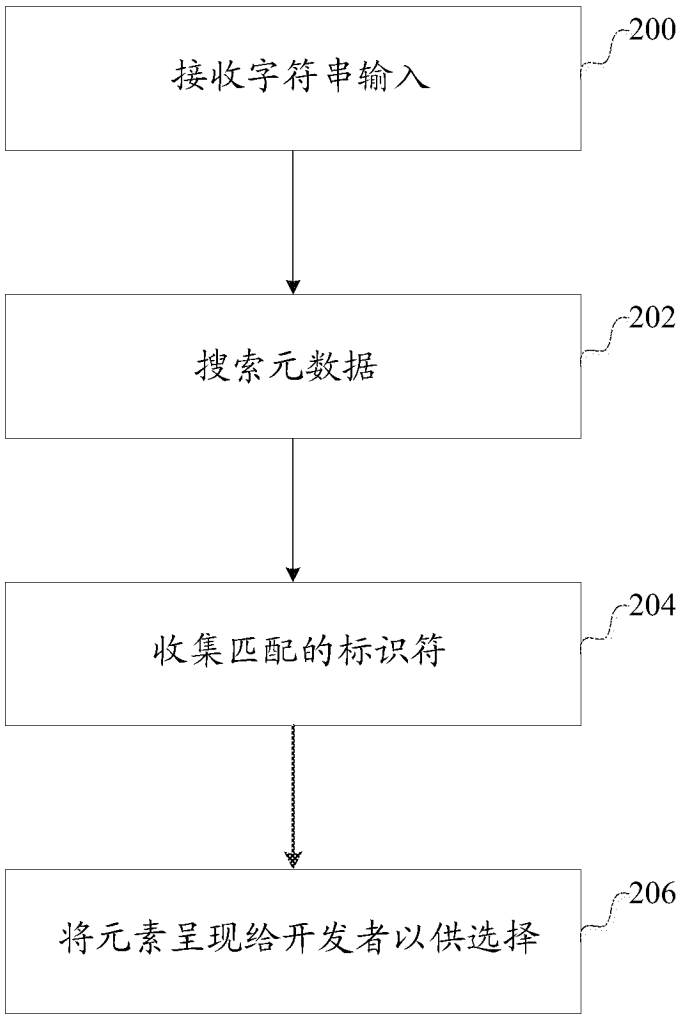


图 2

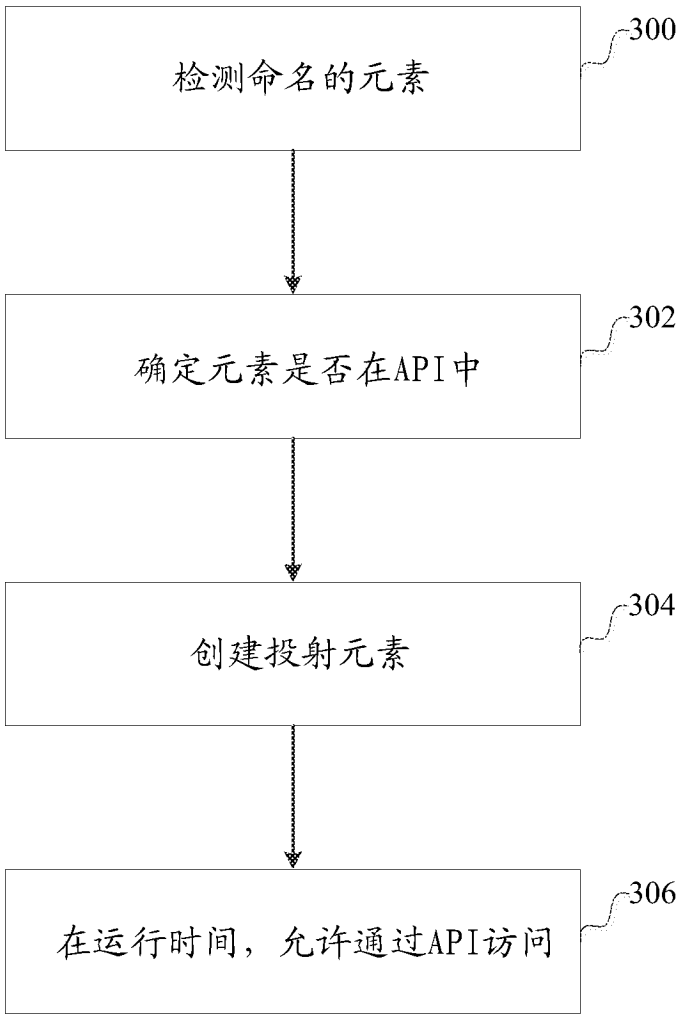

检测命名的元素
300
确定元素是否在API中
302
创建投射元素
304
在运行时间，允许通过API访问
306

图 3

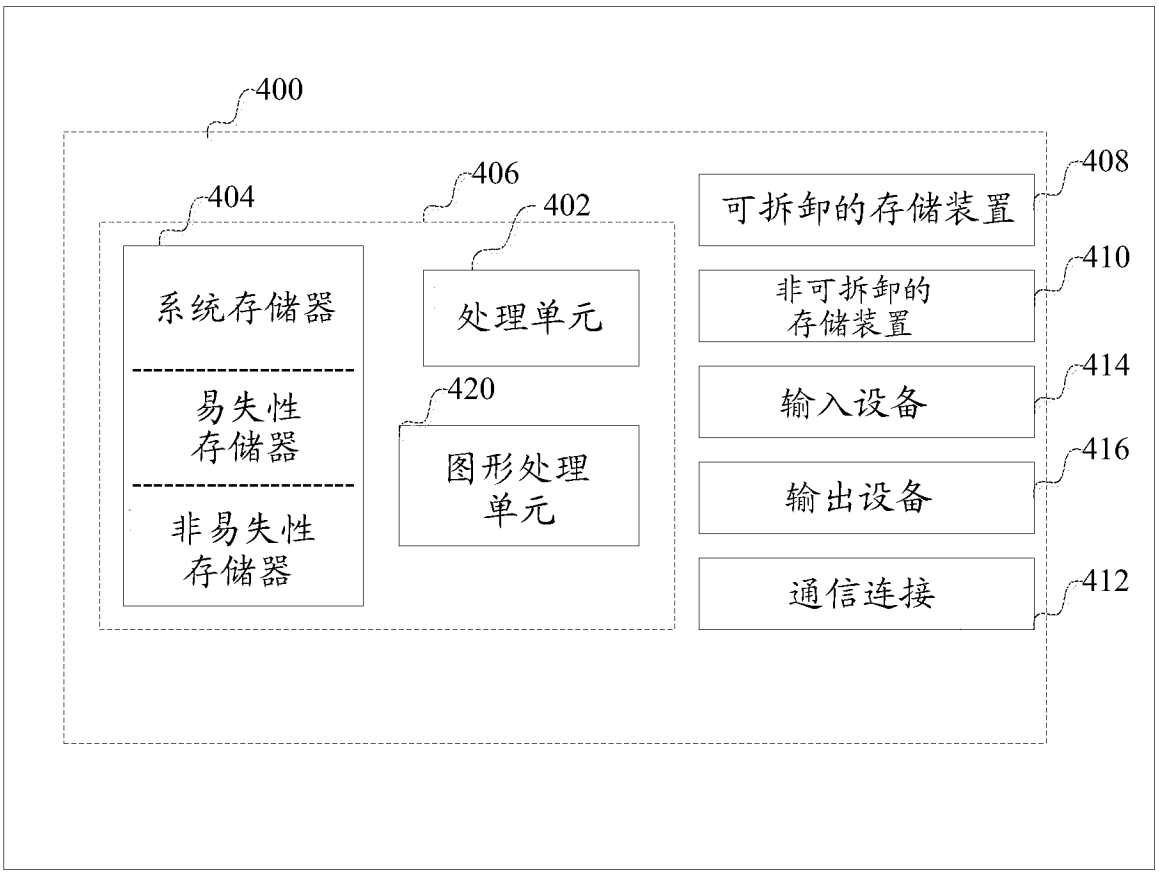

400
404
406
402
408
410
414
420
416
412
系统存储器
易失性
存储器
非易失性
存储器
处理单元
图形处理
单元
可拆卸的存储装置
非可拆卸的
存储装置
输入设备
输出设备
通信连接

图 4