构造函数使用 return 但没有返回值，或者返回的是一个原始值，则这个返回值会被忽略，仍然以新创建的对象作为调用表达式的值。

8.2.4 间接调用

JavaScript 函数是对象，与其他 JavaScript 对象一样，JavaScript 函数也有方法。其中有两个方法——call() 和 apply()，可以用来间接调用函数。这两个方法允许我们指定调用时的 this 值，这意味着可以将任意函数作为任意对象的方法来调用，即使这个函数实际上并不是该对象的方法。这两个方法都支持指定调用参数。其中，call() 方法使用自己的参数列表作为函数的参数，而 apply() 方法则期待数组值作为参数。8.7.4 节会详细介绍 call() 和 apply() 方法。

8.2.5 隐式函数调用

有一些 JavaScript 语言特性看起来不像函数调用，但实际上会导致某些函数被调用。要特别关注会被隐式调用的函数，因为这些函数涉及的 bug、副效应和性能问题都比常规函数更难排查。因为只看代码可能无法知晓什么时候会调用这些函数。

以下是可能导致隐式函数调用的一些语言特性。

- 如果对象有获取方法或设置方法，则查询或设置其属性值可能会调用这些方法。更多信息可以参见 6.10.6 节。
- 当对象在字符串上下文中使用时（比如当拼接对象与字符串时），会调用对象的 toString() 方法。类似地，当对象用于数值上下文时，则会调用它的 valueOf() 方法。更多细节可以参见 3.9.3 节。
- 在遍历可迭代对象的元素时，也会涉及一系列方法调用。第 12 章从函数调用层面解释了迭代器的原理，演示了如何写这些方法来定义自己的可迭代类型。
- 标签模板字面量是一种伪装的函数调用。14.5 节演示了如何编写与模板字面量配合使用的函数。
- 代理对象（参见 14.7 节）的行为完全由函数控制。这些对象上的几乎任何操作都会导致一个函数被调用。

8.3 函数实参与形参

JavaScript 函数定义不会指定函数形参的类型，函数调用也不对传入的实参进行任何类型检查。事实上，JavaScript 函数调用连传入实参的个数都不检查。接下来几节介绍函数调用时传入的实参少于或多于声明的形参时会发生什么。同时，这几节也演示了如何