**原文地址**

**https://zhuanlan.zhihu.com/p/34191831**

**微信小程序也要强行热更代码，鹅厂不服你来肛我呀**

[brambles](https://www.zhihu.com/people/brambles)[brambles](https://www.zhihu.com/people/brambles)

2 天前

前言

微信为了限制小程序的热更越过审核，所以限制了 eval / new Function 等方式动态执行代码。其实说实话，限制热更本来能理解，但是你这审核速度如此龟速，让版本迭代和bug修复都异常艰难。尤其是今年春节前夕的时候，我们小程序一审就是5天！但是隔壁某大厂的告诉我们他们小程序审核才用了3个小时，所以也不难猜测我们这些创业小厂永远都是被疯狂插队的。

好，鹅厂既然你不仁，那就特么我就别怪我不义了。不能用 eval / new Function 就不能动态执行代码了？不存在的！这种限制聊胜于无，不过一张纸而已，有本事就关了 ajax 呀。

首先用 JavaScript 写一个 JavaScript 解释器

热更的第一步，就是用 JavaScript 写一个 JavaScript 解释器。对于很多小伙伴来说，写一个JavaScript 解释器这听起来天方夜谭，但事实上却是非常简单，因为使用 JavaScript 实现 JavaScript 解释器，所以语义几乎可以完全复用。

我现在列一下具体实现步骤：

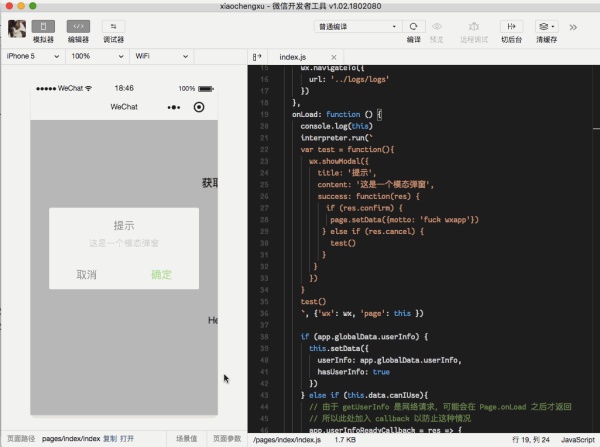
1. 解析 JavaScript 代码字符串，得到 JavaScript 代码的抽象语法树（AST）。解析这个步骤不需要自己做，有非常多优秀的现成的库，比如 [acornjs/acorn](http://link.zhihu.com/?target=https%3A//github.com/acornjs/acorn) 。结果会解析成标准的 ESTree。
2. 因为 JavaScript 的语法树是有标准格式的 [estree/estree](http://link.zhihu.com/?target=https%3A//github.com/estree/estree) ，所以只需要对照这个标准格式进行实现语法树的求值，只要保证 JavaScript 一样的语义，就和 eval / new Function 等效果差不多了。

需要注意的细节：

1. 作用域。比如 var 声明的变量是函数作用域，const / let 是词法作用域，这些要注意区分。
2. || 和 && 运算符。这两个运算符有短路的效果，所以不能和其他运算符一样先求两边的值。
3. 函数闭包。就像上述所说的，用 JavaScript 实现 JavaScript 解释器几乎可以完全复用 JavaScript 语义，在这里就体现的淋漓精致了。解释器函数闭包就可以用语言本身的闭包来进行实现，非常轻松愉快。
4. 注入标准库。我们做这个解释器的目的不是为了做沙盒，所以没有对环境进行隔离，但是即便如此，我们还是需要把 JavaScript 的标准库注入进去。标准库可以在 [JavaScript 标准库](http://link.zhihu.com/?target=https%3A//developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects)找到。
5. new 操作符。这个问题我研究了很久，最终找到了还算是有效的方案，在下面示例代码里面有。
6. 打断控制流的操作。break / return / continue 这类会打断控制流的操作，处理起来需要小心。

示例代码：[bramblex/jsjs](http://link.zhihu.com/?target=https%3A//github.com/bramblex/jsjs)

效果如下：



效果图

然后设计完整热更方案

page / component 这种东西我暂时还没研究过能怎么更新，毕竟我对小程序也不是非常熟悉。我们就直接讨论如何热更程序代码吧。

首先既然我们要热更，我们就必须思考几个问题：

1. 从哪里更新代码？毫无疑问，我们需要一个用来更新的服务器。
2. 更新下来的代码放在那里？目前来看，更新下来的放在小程序里面的 Storage 里面就很好。
3. 什么时候知道代码需要更新了？要解决这个问题，那需要我们要在远端服务器和本地记录代码版本，对比版本来确定是否需要更新。
4. 代码什么时候时候执行？我建议是把热更的代码当成一个单独的模块，在 App 创建之前就可以执行了

接下来，解决问题，设计具体热更方案：

1. 代码从服务器更新，有两个 api 。/update/version 返回 v0.0.1 这样的版本号，/update/code 返回代码字符串。
2. 本地的 Storage 有 version 和 code 两个 key 进行存储对应的本地代码和版本。
3. 当打开小程序的时候，获取比较本地版本和远程版本，如果版本一致怎不管，如果版本不一致则更新本地代码。
4. 执行热更的代码，热更部分代码可以用 module.exports 或者你喜欢的变量形势导出暴露的接口，保存在一个模块里面。
5. 然后各个 page 调用热更部分的模块代码，嵌进业务代码里面。

具体方案的示例代码我就不给了，因为我也没有……毕竟，之后如果需要有什么项目需要热更小程序的时候，再写这部分代码，反正已经设计好了。

最后

这个方案还有非常多能够改进的地方，比如用实现 IR 中间码来进行传输和使用，提升性能和内存占用等等。不过这是后话了，之后有时间再做改善吧。