# 从 Haskell 到 WebAssembly（1）

**Felis sap...**

函数式编程、编程语言、编程 话题的优秀回答者

已关注



## 项目简介

安利一下我最近在 Tweag I/O 做的项目：一个基于 GHC 的 Haskell to WebAssembly 编译器。项目地址：tweag/asterius

目前项目处于 alpha 阶段，可以编译一些简单的 Haskell 程序并在 Node.js 上运行。支持特性：

- 除 Template Haskell 以外的大多数 GHC 扩展
- ghc-prim/integer-simple/base/array 中不涉及 IO 的部分。IO 靠运行时内建的接口以及 JavaScript FFI
- 运行时暂不支持垃圾回收，可在链接时设定 initial heap size，可在运行期动态申请新内存
- 可以通过 foreign import javascript 语法在 Haskell 程序中运行 JavaScript 片段，支持 first-class 的 JSRef 类型
- 除了需要打开 V8 的实验性 BigInt 扩展以外，只要 JavaScript 引擎实现 WebAssembly MVP 版本的特性即可，无需 anyref/host-bindings/gc 等提案的支持
- 链接器支持 debug mode，可在运行期打印包含内存读写、控制流转移的日志
- 实现了 binaryen 的多层 Haskell 绑定，底层是 raw C bindings 以及自动编译、链接 binaryen 库的功能，中层是一套 Haskell IR 对应 binaryen IR，支持序列化、符号解析与跨模块链接等，高层有一个 monadic EDSL，可以在 Haskell 里用 do-notation 之类的语法糖直接写 WebAssembly 代码

最简单的试用方法是下载 Docker image，每个通过单元测试的 commit 会自动上传一个 Ubuntu-based 的 image。直接 docker pull terrorjack/asterius 可以获得对应 master 分支的最新 image，进一步指示参考 ahc-link --help 即可，可以将 .hs 编译到一个 .wasm 文件和 .js 文

当然，项目状态距离能在浏览器里愉快地跑 Haskell 代码还是有不少差距的，剩下的工作主要是运行时方面的：

- storage manager 增加垃圾回收。虽然无脑默认开个 1G 的堆也已经可以运行不少 demo 了，这样总不是个办法对吧？当然，我要是有 Simon Marlow 十分之一的本事的话这个早就做出来了，原版的 GHC 里面是并行分代 GC，我会先做个非并行非分代的实现然后慢慢迭代的（
- scheduler 支持 Haskell 线程调度，包括运行 unbound thread、中断和恢复 thread 等。这个弄完以后，foreign export javascript 就有戏了，毕竟一个前端语言如果不能从 JavaScript 调用进去的话那没法用啊是不。

上面的坑填上以后还有不少周边的工作可以做，比如加强 Cabal 支持，能编译 Haskell 标准库以外的更多库、做个 WebIDL 到 Haskell 模块的编译器，调 Web API 更容易，etc。在做了，你智猫姐姐骗过人吗（

大致介绍完了，接下来简单说下 WebAssembly 和开发面向 WebAssembly 的编译器的那些事，先讲 non-Haskell specific 的部分，再讲 Haskell specific 的。会分几期慢慢讲。

## WebAssembly 介绍

总之，如果你是个吃瓜群众，你只需要知道，WebAssembly 能让你享受做前端开发以及写 C/C++ 的双份快乐（逃。目前使用第三方语言编译到 WebAssembly 的使用体验最完善的是 C/C++ 和 Rust。Rust 的话官方编译器可以基于 LLVM 的 wasm32 实验性后端生成代码，而 C/C++ 自然是靠我们熟悉的 Emscripten，Emscripten 有 LLVM/binaryen 的模式可以选择。教程网上很多我就不贴了。这里主要写如果你想自己写编译器，顺带蹭一波 WebAssembly 的热点的话，大概需要了解的东西。

关于 WebAssembly 版本：目前主流 JavaScript 都实现了的是 WebAssembly MVP，另外 WebAssembly 官方有不少 proposal 在或快或慢地进行讨论和实现，以下如果没有特别声明，WebAssembly 均一律指代 MVP 版本的 wasm32。

WebAssembly 跑起来是个什么感觉：

1. 首先你要有个 WebAssembly 的 binary file。WebAssembly 官方有一个 S-exp 的语法，那个是给人看的，还有一个 binary format，浏览器认 binary format。一个 binary 代表一个 WebAssembly 的 module。
2. 然后，在 JavaScript 中调用相关接口编译，读入 binary，对其进行 parse 和 validation，生成一个 Module。这个 Module 暂时还不是可运行的。
3. 对编译成功生成的 Module 进行初始化。初始化的时候需要提供 import object，比如 WebAssembly module 中声明需要导入什么 JavaScript 函数的话，相关函数必须在 import object 里存在（不过并不在 JavaScript 侧进行类型检查）。
4. 编译/初始化都是异步的，两个步骤可以合一。初始化完，拿到 Instance 以后，Instance 的

的玩意或者要让你传个 callback 啥的。

以上是跑起来的感觉，那么写起来的感觉呢？

- 类型系统：静态强类型，没有任何的隐式转换，基本类型就只有 i32/i64/f32/f64 这几种。函数类型是从若干个基本类型的参数值生成 0 个或 1 个基本类型的结果值。
- 线性内存：有一个 32 位寻址的内存空间可以读写基本类型，支持 unaligned access。内存大小可以按 64KB 的页为基本单位，声明初始/最大大小、查询当前大小和申请扩充。函数代码等数据并不放在线性内存里。哦对了，0 也是合法的内存地址。
- 全局变量：除线性内存以外，另一个跨函数可以共享的全局状态就是全局变量，可以声明全局变量的初始化值，以及是否可变。
- 栈虚拟机：WebAssembly 没有“表达式”概念，而是假设运行时实现了一个栈，多数指令的作用就是从栈顶弹出若干个基本类型的值，做做计算，把结果写到栈顶。
- 函数抽象：呵呵，没有尾调用。把想要支持被间接调用的函数放到一个表中，编号就是其“函数指针”，可以用 call_indirect 指令进行间接调用，间接调用在运行时进行类型检查。另外，函数可以声明局部变量。
- 异常处理：可以调用 trap 立即中止 WebAssembly 代码，在 JavaScript 侧抛出异常。WebAssembly 自身不能处理异常。
- 导入导出：支持导入导出的对象有：函数、全局变量、线性内存。函数必须满足 WebAssembly 的函数类型要求。不用说，JavaScript 的引用类型是不支持的。另外 i64 也不行（不过有 workaround）
- 控制流：没有任意跳转指令。有 block/loop 指令、label 和作用域的概念，所有的跳转指令只能跳转到外层的 label，有点类似于 lambda calculus 里面的参数绑定。

更具体的细节，可以去慢慢抠 WebAssembly spec，以及 MDN 上有关 WebAssembly 的 API 描述。如果有不清楚的地方，WebAssembly 官方有个 OCaml 写的参考解释器可以看看源码。（参考解释器不带 JavaScript 引擎，所以如果想要观测副作用，得通过修改全局变量/线性内存来实现）

## 生成 WebAssembly 模块

以上介绍了单个 WebAssembly 模块大概长什么样，以及怎么跑起来。首先你需要有个 binary 模块，这个东西怎么生成，难道要对照 spec 自己拼二进制串？这里列几种不同的思路：

1. 通过 S-exp 语法的源代码，解析以后重新生成 binary。比如 WebAssembly 官方提供的 wabt 工具，可以在 S-exp 和 binary 之间进行转换，并且支持 validate 和 interpret 操作。基于 wabt 的在线版工具 WebAssembly Studio 可以直接在网页上编辑和运行 S-exp 格式的 WebAssembly 模块。
2. 通过 LLVM 的 wasm32 后端生成 WebAssembly module。前面可以对接 clang（不带 C 标准库支持），后面可以对接 lld，lld 实现了实验性的 WebAssembly 链接功能。如果不关心单个

3. 通过 binaryen 库生成 WebAssembly module，这个主要就是面向编译器开发者的。本项目底层也用到了 binaryen，所以接下来着重介绍 binaryen。

binaryen 本身是一个 WebAssembly 官方支持的 C++ 库，提供了 C/JavaScript 接口可供第三方编译器开发者使用。binaryen 库主要支持的功能包括：

- 解析/生成/验证单个 WebAssembly 模块，支持 S-exp 文本格式的解析和输出。与 V8/SpiderMonkey 等引擎相比，binaryen 支持的 WebAssembly 特性集相对保守，目前除了 MVP 以外，额外支持的特性只有 thread/atomics 相关指令。
- 解释执行 WebAssembly 模块。这个解释器和官方的 spec 解释器/wabt 解释器支持的功能大致类似。
- 支持两套不同的 IR：基于表达式的递归 IR，以及贴近 WebAssembly 底层的栈 IR。C API 中目前只支持表达式 IR，主要出于历史原因：WebAssembly 草案最初是表达式 IR，binaryen 也这么实现，后来转向栈 IR 以后，表达式 IR 的接口仍然保留了下来。对于编译器开发者而言，使用表达式 IR 可以稍微比栈 IR 省一点点工作（其实就是后序遍历一下 AST 的事）
- WebAssembly 中涉及函数名、函数类型名、函数表索引等需要索引的地方，一律是一个 i32，而在 binaryen 中可以用字符串命名，这点微小的工作对编译器开发者还是很贴心的，不要求自行维护符号表。
- 最重要的是，binaryen 中提供了 Relooper 算法的实现。很多编译器涉及的中间表示是带任意跳转的控制流图，控制流图的节点之间不一定存在拓扑序，也就无法非常简单地用一系列嵌套的 block/loop 来表示。Relooper 算法接受控制流图输入，输出基于 block/loop 实现的控制流。

所以，如果你是个编译器开发者，想要体验一波 WebAssembly 的代码生成的话，大致需要做什么事呢？

- 翻一下 MDN 上 WebAssembly 的文档，大致知道怎么在 JavaScript 中编译和运行一个 binary 模块。Node.js 中的接口大致相同，不过不支持流式编译。
- 可以通过拼接文本生成 S-exp 源代码，然后调 wabt 生成 binary，或者调用 binaryen 的接口。
- 拿到 binary 之后，再生成一个很小的 JavaScript wrapper 脚本，这个脚本负责读入 binary 然后执行。
- 有什么不理解的地方，多做做实验，然后回头翻一下 WebAssembly spec。不过，一开始一行代码都不写就去翻 spec，我是不推荐的，很容易自以为看懂了结果根本不是那么回事。一点小小的人生经验（逃

预定下一期介绍 asterius 编译器的代码生成器。

发布于 2018-08-05

1 人已赞赏

函数式编程 WebAssembly Haskell



## 文章被以下专栏收录

**不动点高校现充部**
一切与编程语言理论、函数式编程相关的杂谈。
已关注

**雾雨魔法店**
http://zhuanlan.zhihu.com/marisa/20419321
已关注

## 推荐阅读

**当WebAssembly遇到BuckleScript**

今天一大早 发现Google V8的core developer来BuckleScript 提issue:Int64 compilation duplicates effects · Issue #1154 · bloomberg/bucklescript 很好奇Google再用BuckleScript干什么...

张宏波


**WebAssembly 系列（五）为什么 WebAssembly 更快?**

胡子大哈 发表于前端大哈


**WebAssembly 系列（一）生动形象地介绍 WebAssembly**

胡子大哈 发表于前端大哈

**Wel Wel**

胡子

## 22 条评论

切换为时间排序

写下你的评论...

刘闽晟 2 个月前

所以你们没有照搬 GHC 的 RTS?

neo lin 2 个月前

啊，智猫姐姐超棒的，求抱!

1

Felis sapiens (作者) 回复 刘闽晟 2 个月前

RTS 是手写的，先只支持一部分 GHC RTS 接口再慢慢扩充

赞 查看对话

Oling Cat 2 个月前

标题序号不从 0 开始差评!

赞

dram 2 个月前

抱抱~

赞

朱天琦 2 个月前

刚好在玩WebAssembly，期待Haskell的引入

赞

jayceelee 2 个月前

居然是个小姐姐，居然是个小姐姐，收下我的膝盖

赞

Inflation 2 个月前

黑暗gc22，我康到了!

赞

baozii 2 个月前

赞赞赞!

赞

baozii 2 个月前

“加强 Cabal 支持，能编译 Haskell 标准库以外的更多库、做个 WebIDL 到 Haskell 模块的编译器，调 Web API 更容易，etc。在做了，你智猫姐姐骗过人吗” 先mark了

2

除 Template Haskell 以外的大多数 GHC 扩展

为什么？不能先expand再编译吗？

赞

Felis sapiens (作者) 回复 圆角骑士魔理沙 2 个月前

TH expand 的环境应该与运行环境一样，也就是说要用 JS 运行时来跑。比如我用 runIO 在 TH splice 里面跑一个 IO，这个 IO 是 wasm 环境里才有的，这种就不行了。当然，对特别简单的情况（不需要 runIO），TH 只在 GHC 进程内部跑，那么是可以支持的，但是支持不完善

赞 查看对话

圆角骑士魔理沙 回复 Felis sapiens (作者) 2 个月前

可否弄个WASM monad，IO可以转过去，不能(safe)转去IO，其实就是个newtype，这样表示wasm环境的东西别想了

赞 查看对话

Felis sapiens (作者) 回复 圆角骑士魔理沙 2 个月前

那这样以来所有的 wasm 环境下的 IO 都用这个来表示，所有的标准库全部报销没法用了。。（

赞 查看对话

圆角骑士魔理沙 2 个月前

能用啊。。。能转过去，有MonadCatch啥啥啥

赞

圆角骑士魔理沙 回复 Felis sapiens (作者) 2 个月前

或者，接着用IO，但是提供一个没实现的typeclass wasm，main :: wasm => IO ()

赞 查看对话

Felis sapiens (作者) 回复 圆角骑士魔理沙 2 个月前

我可能理解错了你的意思。你是不是说，有个 Wasm :: Type -> Type，然后 IO 可以 coerce 进去，然后我 TH splice 里面只能跑 IO，普通代码能跑 Wasm?

赞 查看对话

Felis sapiens (作者) 回复 圆角骑士魔理沙 2 个月前

那假如我的 TH splice 里调用着调用着发个 filesystem/network 的请求的话。。那普通运行和 GHC 进程内运行就根本不一致了啊。。说白了这不是类型能解决的问题，而是运行时

赞 查看对话

不过偷懒的话可以假装不一致也没关系，readme 里面直接说支持 Template Haskell，后面打个星号，加一句 terms and conditions apply 不就行了么（逃。不过我是只诚实的猫娘呢，没办法

1 查看对话

圆角骑士魔理沙 回复 Felis sapiens (作者) 2 个月前

不能不一致吗。。。。

赞 查看对话