THE LINUX FOUNDATION 商标

为法律专业人士准备的 WebAssembly (Wasm) 文档

新型技术的许可证合规要求注意事项

2022 年 12 月

作者： ***Tjaldur Software Governance Solutions* 公司** Armijn Hemel 硕士

目录

文件撰写目的

最近有一项名为 WebAssembly 的技术让许多人爱不释手。网上的 WebAssembly 文档主要是针对开发者，侧重于如何使用或开发 WebAssembly。目前缺乏从开源许可证合规要求的角度介绍 WebAssembly 的文档。编写这份文件是为了填补这一空白。

在技术层面上，WebAssembly 并不是独一无二的（过去，大多数技术都以某种方式使用过），但在管理层面上，WebAssembly 是一项可以从根本上改变网速的技术。然而，与过去不同的是，该技术是由所有主要浏览器引擎的开发者合作完成的，是开发 W3C 过程中开发的 WebAssembly。

由于 WebAssembly 技术与现有技术相似，其他开源项目满足许可证合规要求的流程在 WebAssembly 保持不变。由于 WebAssembly 代码主要是通过网络实现交付机制，合规信息可能会根据所在地有所不同。

本文件不是技术文件。本文件不是编程教程，简化和略过了许多技术细节，避免本文件跑题。必要时，会提供包含更多信息的（技术）文件链接。

本文件也不是法律文件，读者不应从文中得出任何法律结论。本文件旨在抛砖引玉，讨论 WebAssembly 开源许可证合规要求的内容，以及通过多种方式推广许可证的潜力。本文件的目标之一是强调潜在的许可证合规要求缺陷，并帮助达成对事实的共识，许可证合规要求缺陷可能会产生潜在法律问题。

WebAssembly 还处于持续发展阶段，所以本文件很可能过段时间就会过时。本文件没有描述某种特定情形，并不意味着因此可以安全使用 WebAssembly 或使用时不存在潜在的许可证合规要求问题。

WebAssembly 是什么?

WebAssembly 是一项新技术，主要用于在 JavaScript 无法满足需求时，创建和部署高性能的 Web 应用程序，尽管 WebAssembly 用途1不局限于此，还有其他用途。从服务器上下载 WebAssembly 程序，通常使用在 Web 浏览器内运行的虚拟机来执行 WebAssembly 程序。为了提高安全性，该虚拟机是沙盒化虚拟机——除非明确允许相关访问，否则沙盒内运行的代码无法访问沙盒外的代码、数据或资源。

与 JavaScript 不同，WebAssembly 程序是以编译程序的形式下载的，可以直接使用浏览器的内置虚拟机运行该程序，不需要解析和解释程序。这与 JavaScript 程序形成鲜明对比，JavaScript 程序是以源代码形式下载的 (通常是最小化的2)，然后由浏览器进行解释。

尽管 WebAssembly 设计为将网络浏览器当做客户端，但也有独立的 WebAssembly 虚拟机，如 WAMR[3]、Wasmer[4] 等等，可以在网络浏览器的之外使用 WebAssembly。

WebAssembly 网站[5] 上对于 WebAssembly 的描述如下:

> “WebAssembly (缩写为 “WASM”) 是一种基于堆栈虚拟机的二进制指令格式。Wasm 设计为可用作编程语言的可移植编译目标，在网络上部署客户端和服务器应用程序”

在 Mozilla 网站上[6] 的 WebAssembly 页面中的描述如下:

> “WebAssembly 是一种可以在现代网络浏览器中运行的新型代码——是一种类似于汇编的低级语言，具有紧凑的二进制格式，能以接近原生的性能运行，为 C/C++、C# 和 Rust 等语言提供了编译目标，在网络上运行这些语言。WebAssembly 设计为可以与 JavaScript 同时运行，两者可以同时起作用。”

在浏览器内的沙盒环境中执行下载的编译代码并不是新概念，支取已出现了几个例子，最明显的是 Java 小程序和 ActiveX。与之前的用例相比，真正让 WebAssembly 脱颖而出的是，该代码是所有主要浏览器引擎的团队都在研究的开放标准，而不是由某个厂家推动的技术，也不是只在一款浏览器中运行的技术，也不是必须使用某个客户端操作系统浏览器才可以良好运行的技术 (例如，ActiveX) 。WebAssembly 并不局限于一种编程语言——可以使用多种语言编写的程序构成 WebAssembly 二进制文件。

尽管最初设计 WebAssembly 的目的是在 JavaScript 无法满足需求的情况下加快运行速度，但预计开发者能够超越 WebAssembly 设计者想象，开发出全新类别的应用或用途。有人正在研究将 WebAssembly 虚拟机嵌入到共享库中，尝试用 WebAssembly 增强每个程序。

WebAssembly 格式

WebAssembly 文档中提到了几种格式。

1. 源代码 (可用几种编程语言中的任何一种写出)
2. 由编译器产生的二进制格式代码，然后加载到虚拟机中并通过虚拟机运行代码
3. 二进制格式的文本表示，类似于汇编语言的指令。

源代码和二进制格式

源代码符合程序员的通常编程习惯。程序员使用 Emscripten 编译器将源代码编译成二进制代码，Emscripten[7] 可以将任何 LLVM 支持的语言编译成 WebAssemblybinary。目前，这些编译器能很好地支持 C/C++、C# 和 Rust 语言。正在开发对其他语言的支持，如 Python[8]。

由网络浏览器内置虚拟机运行的目标代码指令构成该程序的二进制格式程序。由程序员编写的源代码编译器通常会生成这样的二进制格式代码。

文本表示

二进制格式的文本表示通常不是源代码，是沙盒可以执行的二进制指令的文本表示。该文本表示等同于二进制格式，可以使用部分工具可以将二进制指令转换成文本表示，反之亦然。

文本表示是一种类似于汇编程序的语言，指令名为 “S - 表达式”，用于操纵堆栈[9]。例如，本文件下文将举例介绍二进制代码的一小部分文本表示，举例如下:

```
(func (;6;) (type 2) (result i32)
      (local i32 i32 i32)
      i32.const 1078
      local.set 0
      i32.const 0
      local.set 1
      local.get 0
      local.get 1
      call 51
      drop
      i32.const 0
      local.set 2
      local.get 2
      return)
```

类似指令可以操纵虚拟机的状态。

这样的指令结构让人联想到 ARM 处理器或 MIPS 处理器之类的汇编代码。将数据放到堆栈中，从堆栈中弹出，并进行操作。借助一些想象，可以将 WebAssembly 指令看作是一个指令集，而将 WebAssembly 虚拟机看作 CPU[10]。

理论上可以直接使用这些类似汇编器的指令来编写小程序，但程序员更有可能用 Rust、C 或 C++ 等高级语言编写代码 (为了提高效率、加强表现力和可重用性)。然后程序员会使用编译器将高级源代码翻译成二进制代码或文本表示。

尽管在大多数用例中，虚拟机能够运行二进制代码，但大多数 WebAssembly 引擎可以 “及时” 加载和解释文本表示的指令。

WebAssembly 例子

从合规角度而言，有必要了解一下编译过程。具体而言，在编译过程中，包含许可信息的注释会发生什么变化？分发的文件中是否包括源代码文件信息，编译后的二进制格式代码中是否包括必要的要求信息（如有）？

以 Mozilla 开发者网络为例[11]。假设已经安装了必要的组件和工具，如 Emscripten（WebAssembly 编译器）和 wabt（用于转换编译后的二进制代码和文本表示代码的工具），并在 Linux 系统上完成编译。该例子包括一个非常简单的 C 文件，存储在 hello.c 的文件中。

```
#include <stdio.h>
int main() {
        printf("Hello World\n");
}
```

然后 Emscripten 编译器将其编译如下：

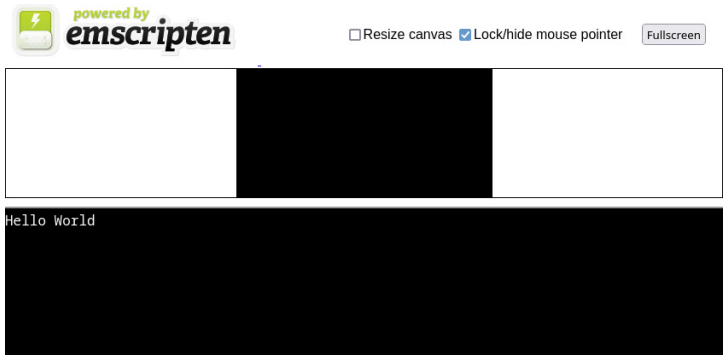
```
$ emcc hello.c -o hello.html
```

会生成以下几个文件：

- hello.html
- hello.js
- hello.wasm

Emscripten 编译器生成的 hello.html 和 hello.js 文件除了使用 WebAssembly 二进制文件的名称，不会使用 C 文件的任何信息。hello.wasm 文件是由 “hello.c” 源代码文件编译的 WebAssembly 二进制文件。

在用户向 Web 服务器请求资源的 Web 会话中，Web 服务器将二进制文件发送至客户端（通常是 Web 浏览器，但也可以是无外设客户端12），并在 Web 浏览器中运行的沙盒环境中运行该文件。网络浏览器必须支持 WebAssembly，才能运行成功。可以使用 JavaScript 代码与运行中的 WebAssembly 二进制文件和 .html 网页交互。



WebAssembly 有一个有趣的特点，在设计指令集时便允许从服务器向客户端串流指令 ——客户端无需等待服务器发送所有带有指令的二进制代码，而是在接收其余的代码时已经开始运行代码13。因此理论上可以创建一个程序，将 WebAssembly 代码当做持续不断的指令流，将其不断地发送至客户端。这段代码可以包含从开放源代码编译的指令，可以为许可证合规要求性事实模式带来意想不到的影响。

使用 “wabt” 包14中的工具可以将二进制文件转换为文本的、类似汇编的表示文件。例如，wasm2wat 工具可以将二进制文件的内容转变成 WebAssembly 指令的文本表示。命令

```
$ wasm2wat hello.asm
```

在简单的 Hello World 举例中，将输出大约 5500 行类似汇编的指令。

遵守 WebAssembly 的开放源代码许可证合规要求

WebAssembly 的开源许可证合规要求与其他情况下许可证合规要求没有明显区别。事实上，两者几乎是完全相同的，将现有的许可证应用于某个新的技术情形，该情形与目前常见的模式略有不同。WebAssembly 确实提出了新的编程模式，对许多律师而言可能是独一无二的。但是，由于这些编程模式是如此的相似，也有很多现成的专业知识，没有必要全盘重来。相反，可以使用来自 OpenChain 项目 15 的最佳实践。

大多数开放源码许可证的条款都规定了只要分发就会激活许可证。分发代码时，需要根据许可证确定需要做什么。有些许可证合规要求提供完整的和相应的源代码 (包括许可证文本) 。例如，GNU 公共许可证合规要求将源代码与二进制文件一同分发，或将源代码的书面报价提供给用户。其他许可证合规要求软件中要包含许可证文本 (有时还要包括一些其他信息) (例如: MIT、2-clause BSD 和 Apache 2) 。上述分类是一种简化，部分许可证会有不同的要求。因此，有必要了解使用的软件许可证以及许可证合规要求。

第一个问题是，是否会出现分发。在标准的 WebAssembly 用例中，会出现分发。将 WebAssembly 二进制代码从 Web 服务器发送到客户端，便是一种典型分发形式。如果您使用任何开放源代码，并且有分发会激活的条款，那么就需要履行许可证义务。即使是看似简单的许可证，也可能需要履行其他与发行无关的许可证义务。

例如，以 MIT 许可证 16 为例，该许可证是网站开发者中广泛使用的许可证 (很多 JavaScript 代码都是使用该许可证获得许可的) 。这也是较短的开源许可证之一; 但是，许可证包含分发二进制软件时的许可证合规要求 (许可证条款之一) 。您可以在 SPDX 网站上找到以下 MIT 许可证模板 (在此文中强调) :

> MIT 许可证
>
> 版权 (c) <年> <版权持有者>
>
> 特此允许任何获得本软件和相关文档文件 (“本软件”) 副本的人无限制地处理本软件，包括但不限于使用、复制、修改、合并、出版、分发、分许可和/或销售本软件副本的权利，并接受本软件的人可以进行此类操作，但须遵守以下条件。
>
> **必须将上述版权声明和本许可声明包括在该软件的所有副本或实质部分中。**
>
> “按原样” 提供本软件，不含任何明示或暗示的保证，包括但不限于对适销性、特定用途的适用性和非侵权性的保证。任何情况下，无论是由本软件、与本软件相关、使用本软件的或其他本软件交易引起的合同、侵权行为或其他诉讼，作者或版权持有人都不对任何索赔、损害或其他责任负责。

许可证规定，每次以任何形式 (源代码或二进制) 分发软件的副本时，也必须与软件一起分发许可证文本和版权声明。WebAssembly 代码库在 MIT 许可证下获得许可时，要与二进制 .wasm 文件或文本表示文件一同发送 MIT 许可文本和版权声明。

当为了实现高度优化，大多数的 .wasm 文件都是通过后台发送给网络浏览器，有意没有向用户展示该过程，您如何做到一同发送这些文件呢? 如果二进制代码包含许可证信息，而用户可以很容易地获取该许可证信息，说明您满足了许可证的条款。下一个问题是: 编译代码时，二进制文件中是否包含许可证信息?

作为测试，已将之前的 “hello.c” 文件复制到 “hello_license.c” ，并作为注释，已在 “hello_license.c” 的顶部添加上述 MIT 许可证文本。带许可证文本的文件要比原始文件大得多。正常的源代码文件只有 65 字节，而带有许可证信息的源代码文件则有 1217 字节。:

```
$ du -b *.c
65   hello.c
1217   hello_license.c
```

为了验证编译器对第二个文件顶部的许可证文本的处理，将两个 C 文件编译成 wasm 文件。

```
$ emcc hello.c -o hello.html
$ emcc hello_license.c -o hello_license.html
```

并比较文件大小和进行 MD5 校验。

```
$ du -b *.wasm
12394   hello_license.wasm
12394   hello.wasm
$ md5sum *.wasm
74118fea55ff6490d8fdd9abb201c3b1  hello_license.wasm
74118fea55ff6490d8fdd9abb201c3b1  hello.wasm
```

如上所述，因为两个编译命令的输出相同，编译器丢弃了注释（此例中是许可证文本）。这意味着 wasm 二进制文件不包含任何许可信息，而只得到二进制文件的用户无法提取或访问许可文本；因此，用户必须通过其它途径获得许可信息。

有几种方法可以获得许可信息。

通过浏览器传递许可证信息

如果用户可以与加载在页面上的 WebAssembly 代码互动，可以通过几种潜在选项满足许可证条款，选项如下:

- 在用户正在浏览的网页上显示许可证信息
- 在网页的可读 HTML 源代码中显示许可证信息（这样，用户查看源代码时就可以看到许可证信息）。
- 将许可证信息作为注释补充到与 WebAssembly 二进制文件一起发送的 JavaScript 代码中。
- 将许可证信息存储在一个单独的文件或 URL 中，然后链接到网页源代码或 JavaScript 文件中。
- 加入一个行业标准的许可证标识符，如 SPDX® ISO/IEC 5962:2021，用户可以根据标识符来识别许可证文本（注意，可能还需要加入版权声明）。

通过文本表示交付代码

另一种可能性是生成并交付二进制代码的文本表示，并在该文件的顶部注释（以两个分号开头的行）中添加相关许可证信息，例如（为方便阅读将例子截断）。

```
;; MIT 许可证
;;
;; 版权 (c) <年> <版权持有者>
;;
;; 特此允许任何人免费获得许可
[...]
```

或使用 SPDX 许可证 ID[17]:

```
;; 版权 (c) <年> <版权持有者>
;; SPDX - 许可证 - 标识符: MIT
[...]
```

应注意，分发 WebAssembly 程序的默认模式通常不是文本格式。使用文本格式而不是二进制格式可能会影响性能，因为必须先将文本格式转换回二进制格式，或者解释文本，开发者不希望看到这种情况发生。

上述方法要求首先从原始源代码中提取许可证文本和其他信息。当然还可以与 WebAssembly 二进制文件一同提供包含所有法律文本的完整和对应源代码。这样，无需提取许可证文

本, 对于某些许可证来说, 相比先提取许可证文本和版权声明并单独提供许可证和版权信息, 这可能是更好的方法。例如, 对于在 2.0 版 GNU 通用公共许可证获得许可的程序, 如果是通过在网络上提供拷贝的方式发布的这些程序, 通常而言, 根据 GPL-2.0 (第 3 节) 要求, 提供对源代码的 “同等访问” , 足以履行主要许可证义务。

概括而言, 有以下五种方式:

1. 在 (互动的) 网站上显示许可证信息。
2. 在网页源代码或 JavaScript 文件中加入许可证信息, 或者添加一个单独的文件, 并将文件链接到在网页源代码或 JavaScript 文件中。
3. 使用文字表述, 并添加相应的许可信息。
4. 提供完整的和对应的源代码, 并与二进制代码一同提供源代码。
5. 以上任意方式, 但根据具体的许可证文本使用行业标准的许可证标识符。

为合规可能采取的步骤

让我们来探讨一下分发 WebAssembly 二进制文件时满足许可证合规要求的几个解决方案。

在第一种情况下, 选择的合规解决方案是, 将任何声明和其他法律文本 (例如, 书面报价) 存储在一个单独的文件中, 将该文件与 WebAssembly 二进制文件一同保存, 并按下文所述提供这些文件:

在第二种情况下, 提供相应的源代码, 并提供所有的许可证文本。这两种情形并不相互排斥: 您可以提供源代码存档和带有所有许可证信息的单独文件, 以及 (如适用) 书面报价。

这两种情况的共同点是: 首先, 您需要审查许可证义务, 看是否应该提供任何内容, 如果是, 应该提供哪些内容 (只提交许可证文本或可能提交更多信息, 如版权声明、作者声明、完整和相应的源代码, 或书面报价) 。

场景 1: 包含许可声明和其他法律信息的单独文件

在第一种情况下, 提供一个包含所有许可声明和其他法律信息的单独文件。为此, 请遵循以下步骤:

1. 从源代码中提取许可声明、版权声明和作者声明 (手动提取或使用 FOSSology[18]、ScanCode[19] 或其他工具提取) 。
2. 创建一个文件或文件集合 (例如, SPDX 格式[20] 的 SBOM 文件或单独的文本文件) , 要包含必要的信息 (许可证文本, 必要时加上书面报价) , 并将相关文件与 WebAssembly 二进制文件一同存储。
3. 确保用户可以找到带有法律信息的文件:
    a. 在 HTML 或 JavaScript 源代码中可接入文件链接。
    b. 在网页上显示带有许可证信息的文件链接。

在 “Hello World” 中, 这意味着从 “hello.c” 中提取许可信息, 并通过 SBOM 文件或某个单独的文件提供许可证信息, 通过网络服务器提供该文件, 并在网站上链接或在 HTML 代码中指向该文件。.

场景 2: 与二进制文件一同提供完整的和相应的源代码

在第二种场景下, 与二进制文件一同提供完整的和相应的源代码。一个好处是不需要从源代码中提取许可声明、版权声明和作者声明, 因为源代码已包含相关声明。其他方面也大致相同:

1. 使用必要的源代码创建一个存档 (取决于源代码的许可证, 可能包括构建脚本) , 并把与 WebAssembly 二进制文件一同保存该存档。
2. 确保用户可以找到带有源代码的文件。
    a. 在 HTML 或 JavaScript 源代码中加入一个指向源代码存档的链接。
    b. 在网页上显示源代码存档的链接。

对于 “Hello World” 程序, 将源代码文件上传到网络服务器并在网站上提供文件链接, 或在 HTML 代码中指向该文件。

动态链接和派生作品

对于一些许可证 (尤其是通用公共许可证) , 在每次合规活动中反复出现的主题是理解什么是衍生作品和动态链接, 所以值得研究的是, 是否也可以为 WebAssembly 开展合规活动, 如果可以, 如何开展。当然, 只有研究实际代码, 才能确定某样东西是另一个程序的衍生品, 并且不能给出概括声明。

WebAssembly 程序不一定是独立的程序。WebAssembly 在设计时考虑的是 WebAssembly 程序可以与其他软件结合, 主要是与 JavaScript 结合。

JavaScript 封装

WebAssembly 与 JavaScript 相互配合 WebAssembly 可以加快那些对性能要求高的运行, 而 JavaScript 无法满足运行速度要求。JavaScript 可以操作网页上的数据, 将困难的工作传递给 WebAssembly 代码。该机制并非 JavaScript 和 WebAssembly 独有的。在其他编程语言中, 也有可以调用其他语言编写的代码。例如, 可以用 C 语言编写的模块扩展 Python 程序 (这取决于所使用的 Python 解释器, 而且 C 语言可能不会在每一款 Python 解释器中都可以运行或者运行流畅) , Java 带有名为 “Java Native Interface” (JNI) 的机制, 等等。

Mozilla 网站对 WebAssembly 中的 JavaScript 封装描述如下[21]:

> “更重要的是, 您甚至不需要知道如何创建 WebAssembly 代码, 也能用好 WebAssembly 代码。可以将 WebAssembly 模块导入到 Web (或 Node. js) 应用程序中, 通过 JavaScript 显示 WebAssembly 的功能以供使用。JavaScript 框架可以使用 WebAssembly 来获取巨大的性能优势和新功能, 同时还能让 Web 开发者轻松使用功能。”

JavaScript 代码可以使用封装机制调用 WebAssembly 函数[2223]。对于从 JavaScript 传递给 WebAssembly 的数据, 封装机制首先将需要将 JavaScript 中使用的数据类型转换为 WebAssembly 支持的数据类型。然后, 封装机制在虚拟机中使用转换后的参数调用 WebAssembly 函数, 获得结果, 将结果转换回 JavaScript 的数据类型, 并将这些数据发送给 JavaScript 函数, 进一步处理结果。

可以用两种不同的标准方式创建这些封装。第一种方式是定义一个带有函数调用的表格 , 获取并加载 WebAssembly 二进制代码, 将 WebAssembly 代码中的函数分配给表中的空位。第二种方式是首先获取 WebAssembly 代码, 加载代码, 然后访问 WebAssembly 模块25 的功能, 由虚拟机将这些功能导出。从 WebAssembly 将 WebAssembly 函数导出至 JavaScript, 叫做 “导出” 。

WebAssembly 代码也可以访问 JavaScript 函数。从 JavaScript 将这些函数导入到 WebAssembly, 叫做 “导入” 。

导入和导出的函数可以在 WebAssembly 和 JavaScript 代码之间建立一个耦合。这种耦合的紧密程度取决于实际情况。可能存在最小的耦合 (例如, 导出函数的通用 WebAssembly 模块) , 但在 WebAssembly 代码和 JavaScript 代码之间也有可能有更紧密的耦合。为了确定是否有耦合达到构成衍生作品的程度或 “离不开” 另一个程序的程度, 有必要了解代码如何与其他代码互动, 有哪些导入和导出方法, 以及如何使用这些方法。

动态链接

可以动态链接 WebAssembly 中的代码。在不同的编译器中，似乎并没有一个标准的机制来支持动态链接，而是取决于每个编译器的支持情况。 Emscripten 编译器26支持共享模块 (有一些限制)，可以实现动态链接。同样的方法在其他 WASM 编译器中可能行不通。 Emscripten 支持在加载时 (与主模块一起加载) 或在运行时 (运行程序后调用侧模块) 进行动态链接。WebAssembly 的该领域的标准化似乎还处于发展早期，需要进一步研究以充分了解其影响。

获取 WebAssembly 原生代码

已提出调用虚拟机内部或外部特定功能的建议，包括调用 WebAssembly 系统接口 (WASI)[27]。WASI 的目的是提供标准化的和平台中立的 API，以调用虚拟机实现或暴露的功能。这些建议目前仍在开发中，可能不会在每个虚拟机都可以实现。WASI 接口让 WebAssembly 应用程序可以调用主机功能。例如，您可以让应用程序将文件写入并保存到本地主机系统[28]。

WebAssembly 的反编译

有些情况下, 唯一可用的代码是二进制代码或其文本表示, 而在需要源代码的地方没有源代码 (例如, 检查时, 检查源代码通常比检查 WebAssembly 二进制代码或文本表示更容易) 。获得类似源代码的方法之一是反编译 WebAssembly 的二进制代码。有一些工具可以反编译, 例如, wabt 包中的 wasm - decompile 工具29。您可以按以下方式调用该工具:

```
$ wasm-decompile hello.wasm
```

输出结果是类似于源代码文件, 尽管结果不一定接近于原始源代码。hello.c 原文件只有 6 行, 而反编译后的代码有 1800 多行。

这很常见。WebAssembly 虚拟机的用途相当简单, 只有一小套指令。结果是, 使用高级语言编写复杂的程序时, 必须将代码翻译成虚拟机的简单指令, 但因为 WebAssembly 没有同样的表达能力, 可能导致产生大量的 WebAssembly 指令。从低级别的指令转为高级别的语言, 要做的工作就增加了, 而且也可能无法重现原来写出的代码。

可以用各种语言编写 WebAssembly 程序, 如 C、C++、Rust 等, 这些都可以编译成相同的 WebAssembly 指令 (也可以合并成一个程序) 。除非保留额外的信息, 说明最初是用什么语言写的程序, 否则很难重新创造出与原始程序相似的结果。

很可能在未来, 会开发出更智能或更先进的反编译器。在此之前, 明智的做法是不要依赖 WebAssembly 程序的反编译输出来反映原始源代码的结构或具体内容。

外发许可

到目前为止，重点是进入许可：如何处理来自第三方的代码，这些代码包含重复使用和/或重新分发开放源代码。您应该考虑如何为自己的贡献获取许可。根据选择的许可证，您也可以采取一些措施，为代码的下游接受者提供便利，这些接受者也可能想重新分发您的代码。

选择一个许可证

第一步是选择一个许可证。您可能要考虑的事情包括：

1. 您使用的其他组件是如何获取许可证的？有必要为您的代码选择一个与其他组件的许可证兼容的许可证。建议进行许可证兼容性检查。
2. 使用某一许可证下许可代码时，有什么义务？

如果下游接受者想重新发布代码，选择的许可证对您的代码下游接受者产生影响。最好了解同一生态系统中的其他人是如何处理代码许可的。毕竟，这是您的代码最有可能结合的代码。在许多生态系统中，往往会倾向于使用特定许可证。

选择一个标准的许可证标题

强烈建议使用标准的许可证标题，而不是发明您自己的。许可证扫描器和法律专业人士可以识别标准许可证标题。编写新版本会造成不必要的复杂化。SPDX 许可证网站[30]列出了许多许可证的常见标准标题模板，供您使用。

以易于获取的版本提供许可证信息

如果您为自己的代码挑选了一个带有许可证义务的许可证，比如披露许可证文本或完整的相应源代码，如果您还提供有软件材料清单 (SBOM)，或者至少提供创建 SBOM 文件的必要信息，会对您的下游接受者确实有帮助。要创建 SBOM 文件，请参见 SPDX[31]。还有其他传达项目许可信息的轻量级机制，比如 About[32] 或 REUSE[33] (后者是基于 SPDX 元数据字段的机制)。

总结

WebAssembly 是一项相对较新的技术，但这并不意味着该可以像其他技术一样不遵守发布软件的规则；许可证条款仍适用。毫无疑问，会使用开放源码软件，这意味着在分发 WebAssembly 软件时需要遵守开放源码许可证合规要求。幸运的是，您可以运用社区中已有的大量专业知识和经验。

这份文件探讨了 WebAssembly 开源要求的一些潜在问题领域，如 WebAssembly 生态系统中使用哪些格式，允许添加或不允许添加许可证文本和 允许添加或不允许添加许可证文本和其他许可证披露文件的位置，与其他代码的互动 (JavaScript 封装，动态链接) 和外发许可证。现在，WebAssembly 生态系统会提出最适合该系统的许可证合规要求最佳实践。

关于作者

Armijn Hemel 硕士，是一位来自荷兰的开源许可证合规高级工程师。在过去的 15 年里，他帮助数百家公司完成开源合规工作，包括在法庭上与版权巨头斗争，以及在上市前审计软件。他还积极编写开源工具，以帮助管理开源供应链。.

鸣谢

特别鸣谢 Luis Villa (Tidelift)、Steve Winslow (Boston Technology Law)、和 Mike Dolan (The Linux Foundation) 对草稿的意见和改进。也感谢法律界的成员，他们支持各种形式的合规技术和开源创新，包括增进成员对 WebAssembly 的理解，确保在世界各地的组织和管辖区有一个繁荣的开源生态系统。.

免责声明

本报告未经修改。Linux 基金会及其作者、贡献者和赞助者明确拒绝任何保证 (明示、暗示或其他) , 包括与本报告有关的适销性、非侵权性、特定用途的适用性或所有权的暗示保证。在任何情况下, Linux 基金会及其作者、贡献者和赞助者都不对任何其他方的利润损失或任何形式的间接、特殊、偶然或后果性的损害负责, 无论这些损害来自与本报告有关的任何类型的诉讼, 无论是否基于违约、侵权行为 (包括疏忽) 或其他原因, 也无论是否已告知相关人士这种损害的可能性。对本报告创作的赞助并不构成任何赞助商对其结论的认可。

THE LINUX FOUNDATION 商标

Linux Foundation Research 成立于 2021 年，探索不断增长的大规模开源合作，提供对新兴技术趋势、最佳实践和开源项目的全球影响的洞察力。通过使用项目数据库和网络，以及对定量和定性方法的最佳实践的承诺，Linux 基金会研究部正在创建开源洞察力的首选代码库，使世界各地的组织受益。

twitter.com/linuxfoundation

facebook.com/TheLinuxFoundation

linkedin.com/company/the-linux-foundation

youtube.com/user/TheLinuxFoundation

github.com/LF-Engineering

2022 年 12 月

如需引用该作品，请按以下方式引用：Armijn Hemel，WebAssembly (WASM) 的法律专业人士。为法律专业人士准备的 WebAssembly (Wasm) 介绍，Linux 基金会，2022 年 12 月。