WebAssembly: New Epoch and Tech Revolution

于航(Jason Yu)

PayPal

自我介绍

于航 (JASON YU)

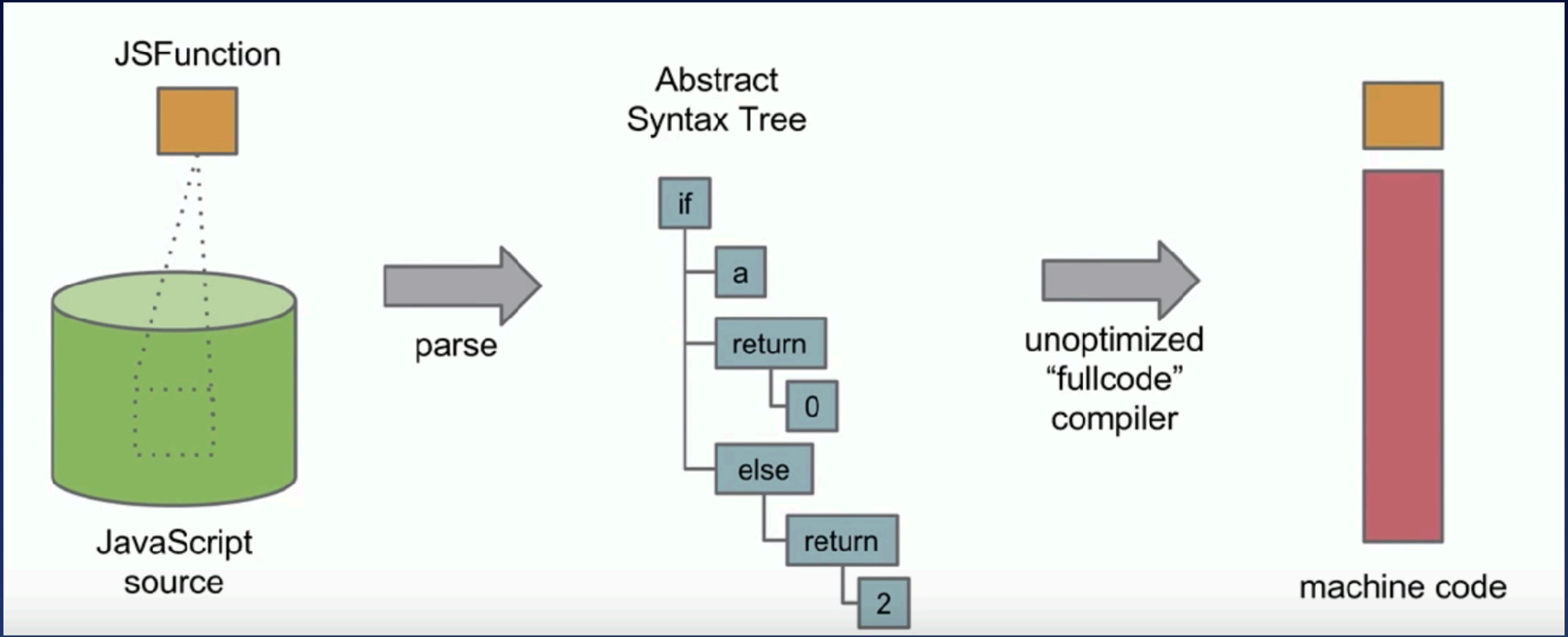
SOFTWARE ENGINEER @ PAYPAL

《深入浅出 WEBASSEMBLY》作者

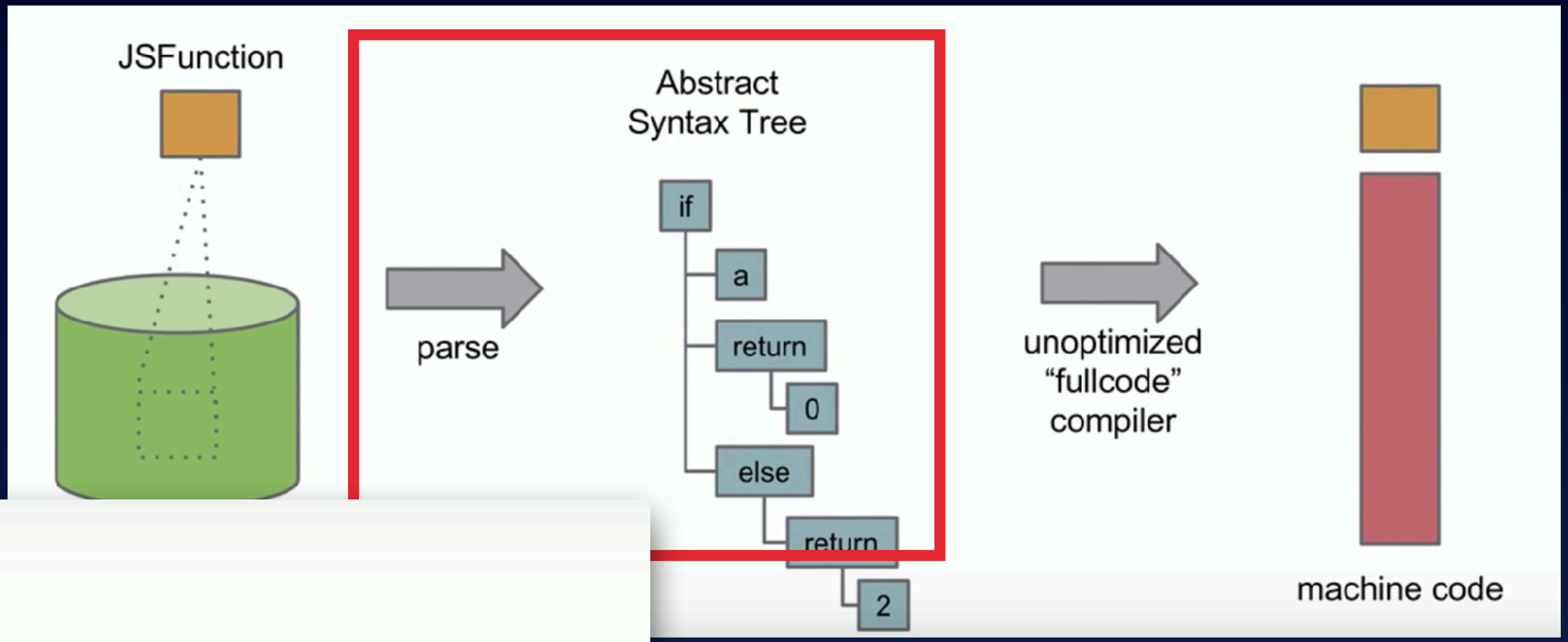
TWVM 作者

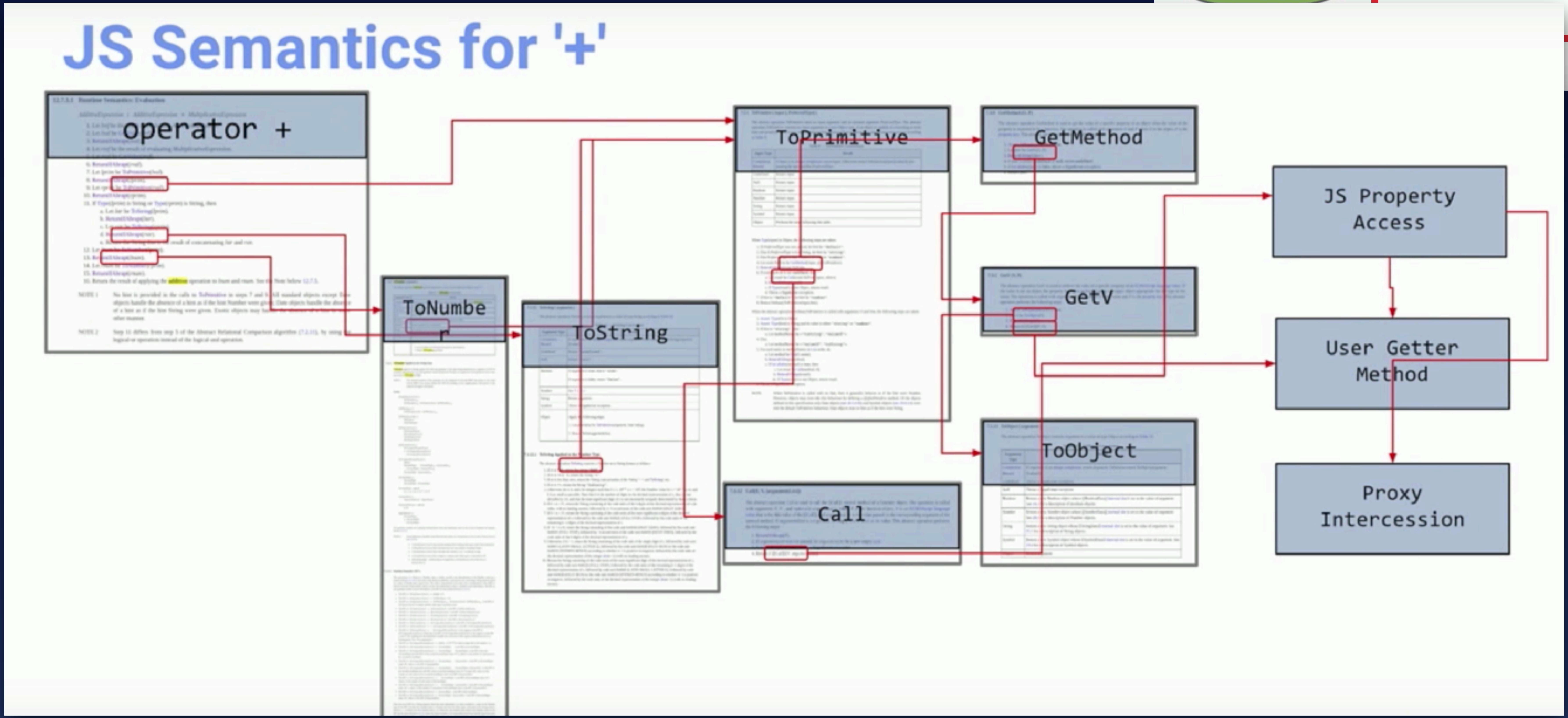
目录

JavaScript 部分执行链路（V8）



ECMA 规定的“+”执行流程



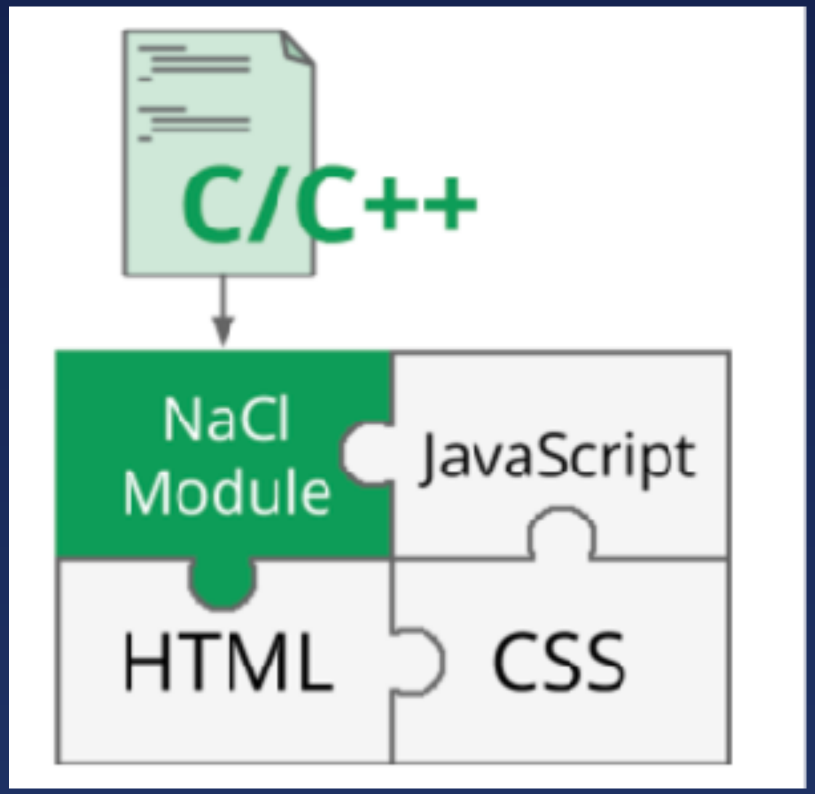


x + y

曾经的尝试 - ASM.js & PNACL

```
function plusOne (x) {
    x = x|0; // x : int
    return (x + 1)|0;
}
```

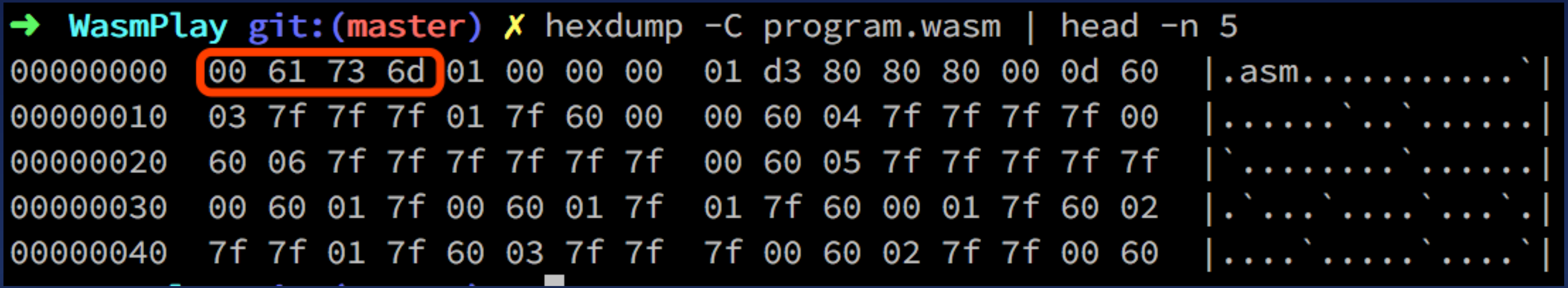
- 是一种 JavaScript 严格子集；
- 通过 Annotation 的方式标注了变量的类型；
- 利于编译器的优化；



- 提供沙盒环境在浏览器中执行的 C/C++ 代码；
- 充分利用 CPU 的特性，如 SIMD、多核心处理等；
- 平台独立，一次编译到处运行；

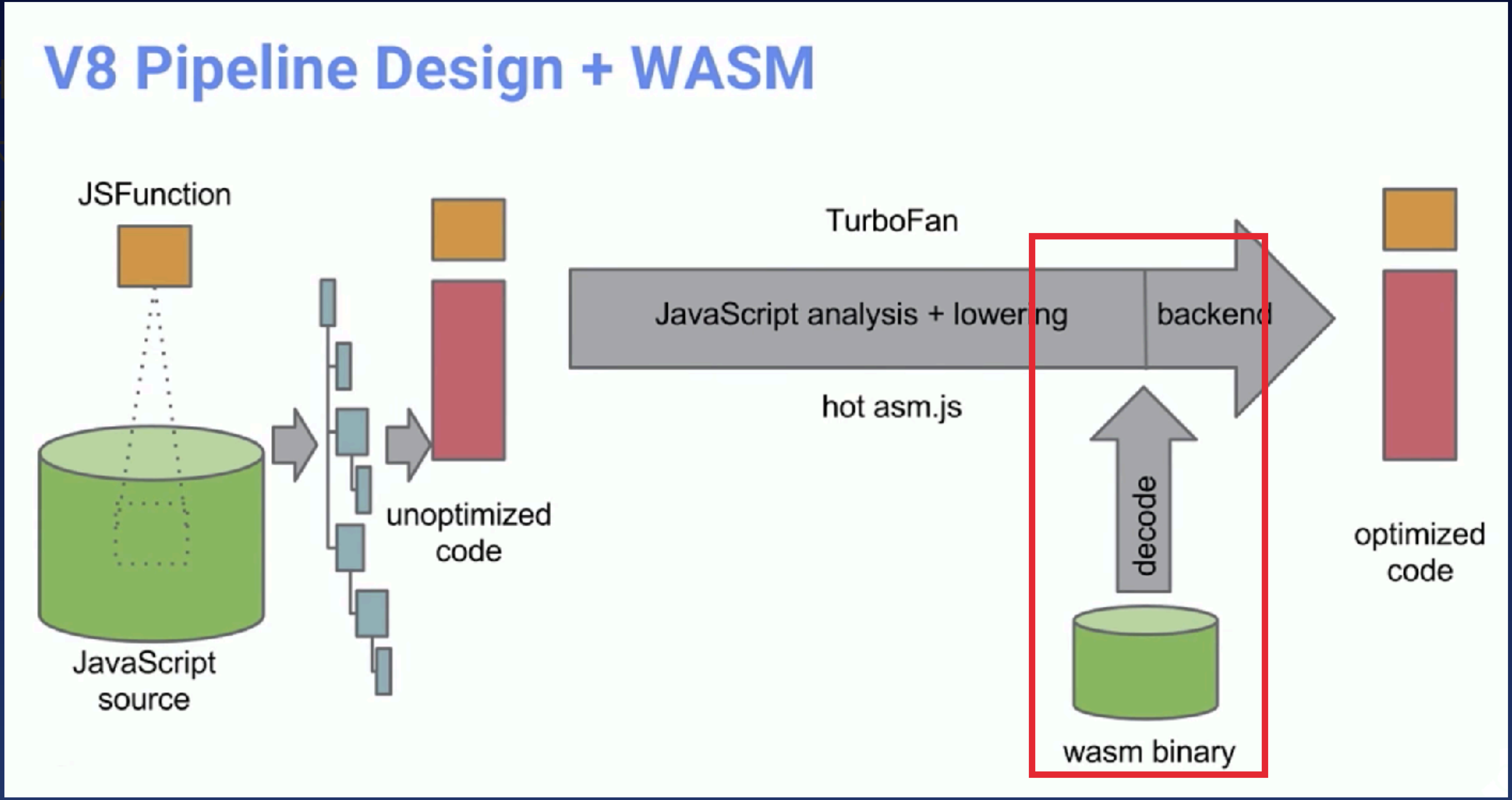
新的方案 - WebAssembly

- 一种新的抽象虚拟机（W3C）标准；
- 四大浏览器已支持该标准 MVP 版本的所有特性；
- 一种以 .wasm 为后缀的二进制格式（0x6d736100）；
- 可以通过标准 Web API 接口在浏览器中加载、解析和运行；



```
➜  WasmPlay git:(master) ✗ hexdump -C program.wasm | head -n 5
00000000  00 61 73 6d 01 00 00 00  01 d3 80 80 80 00 0d 60  |.asm...........`|
00000010  03 7f 7f 7f 01 7f 60 00  00 60 04 7f 7f 7f 7f 00  |......`..`......|
00000020  60 06 7f 7f 7f 7f 7f 7f  00 60 05 7f 7f 7f 7f 7f  |`........`......|
00000030  00 60 01 7f 00 60 01 7f  01 7f 60 00 01 7f 60 02  |.`...`....`...`.|
00000040  7f 7f 01 7f 60 03 7f 7f  7f 00 60 02 7f 7f 00 60  |....`.....`....`|
```

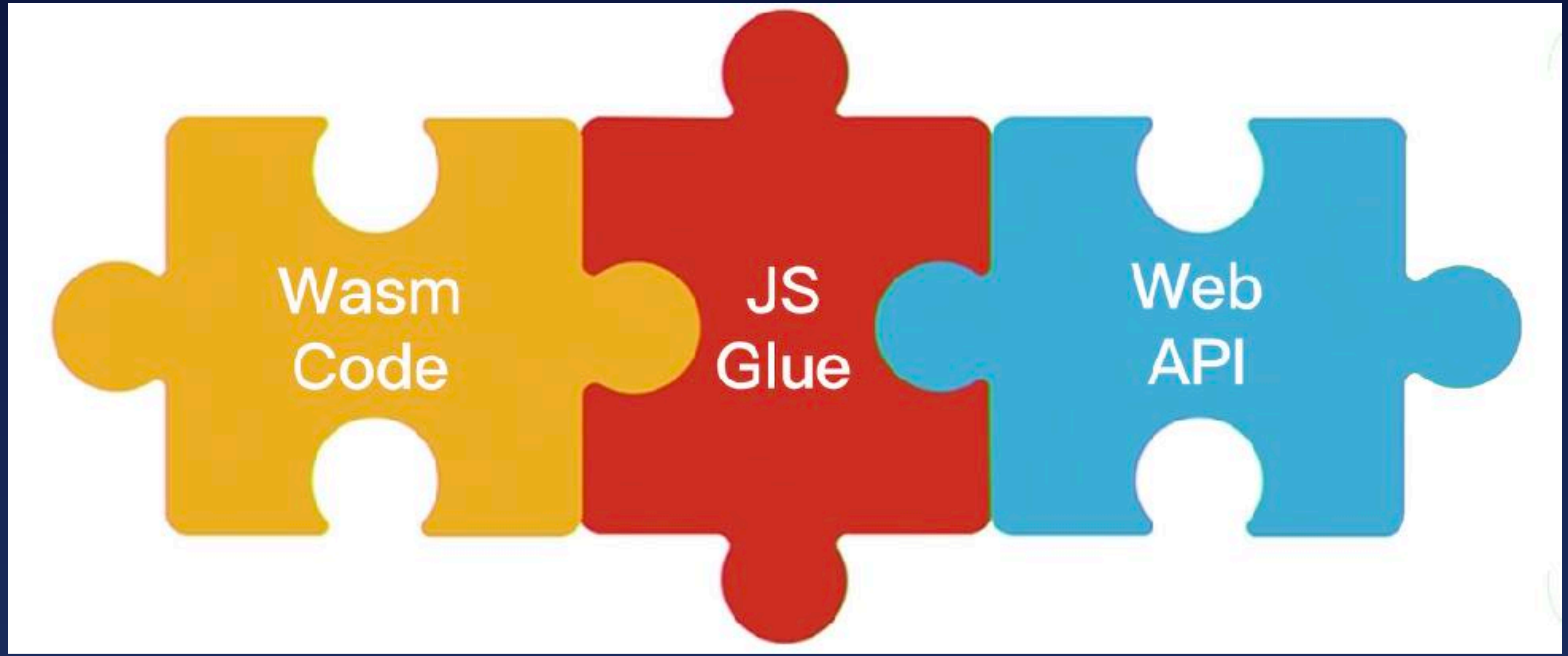
WebAssembly 编译完整链路



使用 Emscripten 构建 Wasm 应用

- *Virtual File System;*
- *Pthread;*
- *Linear Memory;*
- …

C/C++ Source Code



一个简单的例子-C++

C++

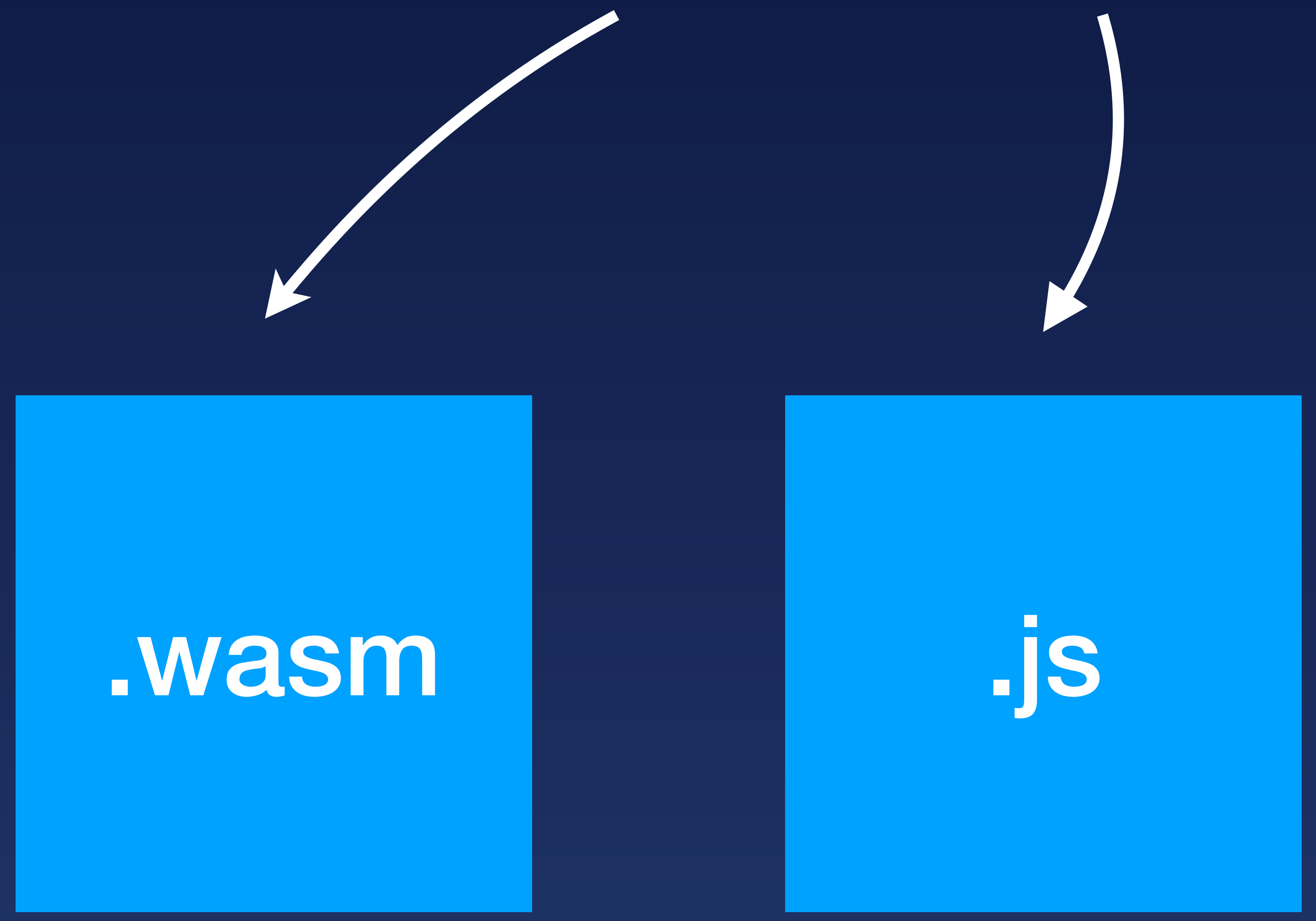
toy.cc

```
#include "emscripten.h"
extern "C" {
    EMSCRIPTEN_KEEPALIVE int add(int x, int y) {
        return x + y;
    }
}
```

一个简单的例子-CLI

CLI

```
emcc toy.cc -s WASM=1 -O3 -o toy.js
```



一个简单的例子-HTML

HTML

toy.html

```
<script>
 fetch('toy.wasm').then(response =>
  response.arrayBuffer()
 ).then(bytes =>
  WebAssembly.instantiate(bytes, {})
 ).then(result => {
  console.log(result.instance.exports['_add'](10, 20));
 });
</script>
```

实例化模块对象

PayPal™

一个简单的例子-WAT

WAT

toy.wasm

类型声明段

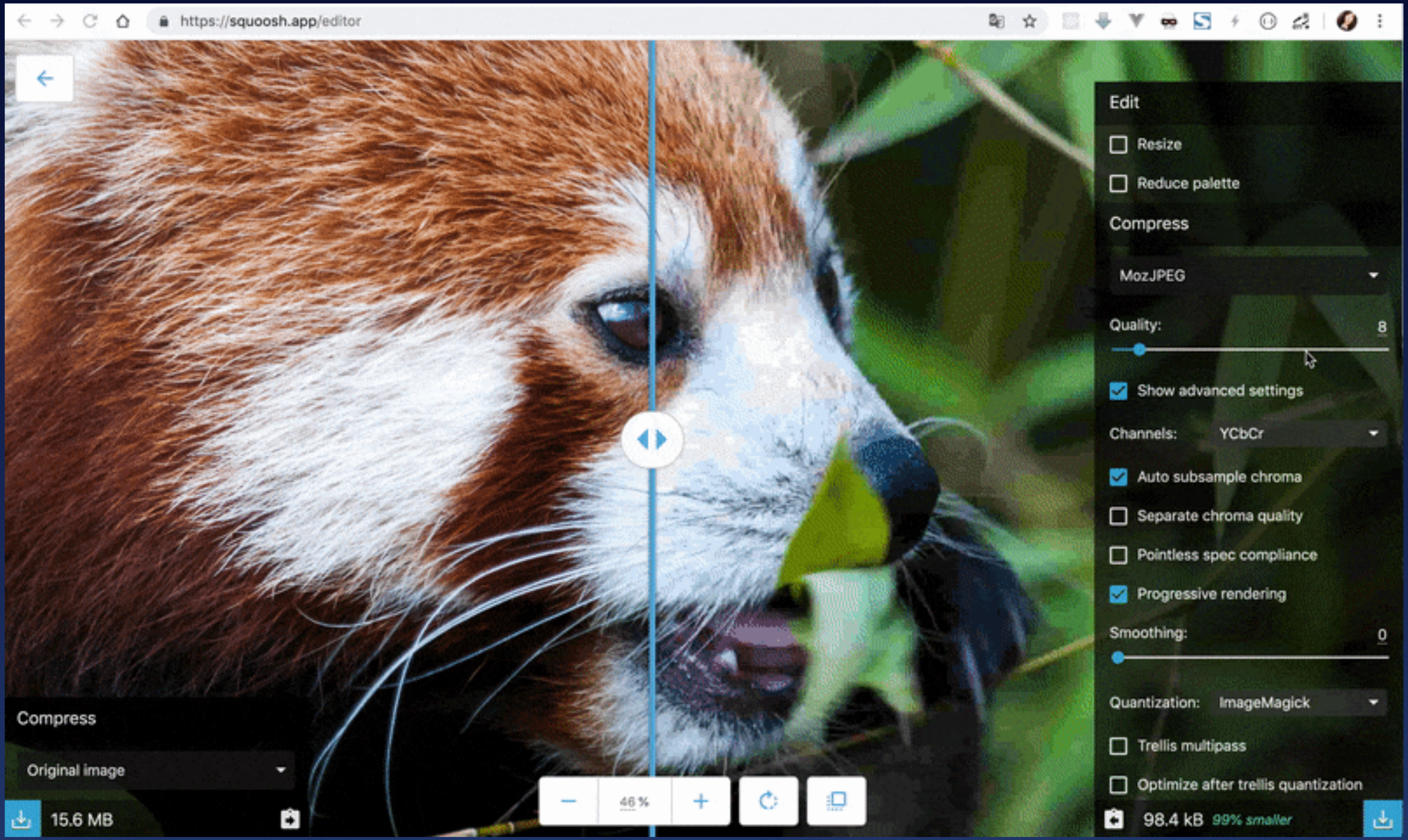
```
(module
(type (;0;) (func (param i32 i32) (result i32)))
(func (;0;) (type 0) (param i32 i32) (result i32)
  get_local 1
  get_local 0
  i32.add)
(export "_add" (func 0)))
```

函数声明段

导出段

PayPal™

WASM 实际应用 - 在线图像处理 Squoosh



Edit
- [ ] Resize
- [ ] Reduce palette

Compress

MozJPEG

Quality: 8

- [x] Show advanced settings

Channels: YCbCr

- [x] Auto subsample chroma
- [ ] Separate chroma quality
- [ ] Pointless spec compliance
- [x] Progressive rendering

Smoothing: 0

Quantization: ImageMagick

- [ ] Trellis multipass
- [ ] Optimize after trellis quantization

libimagequant (C)

MozJPEG (C++)

webp (C)

Wasm 语言编译器 / 解释器

Lua

Microsoft .NET

OCaml

C++

AS

GOLANG

PayPal™

其他应用领域

- 视频/直播编解码；
- 在线图像/视频处理应用；
- 基于边缘计算的机器/深度学习：MXNet.js；
- 高性能 Web 游戏：Unity、Unreal、Ammo.js 等游戏库和引擎；
- 区块链 Ethereum 核心；
- 前端框架：sharpen、asm-dom、yew；
- IOT：wasmachine；

Wasm MVP 两个重点：

高性能计算

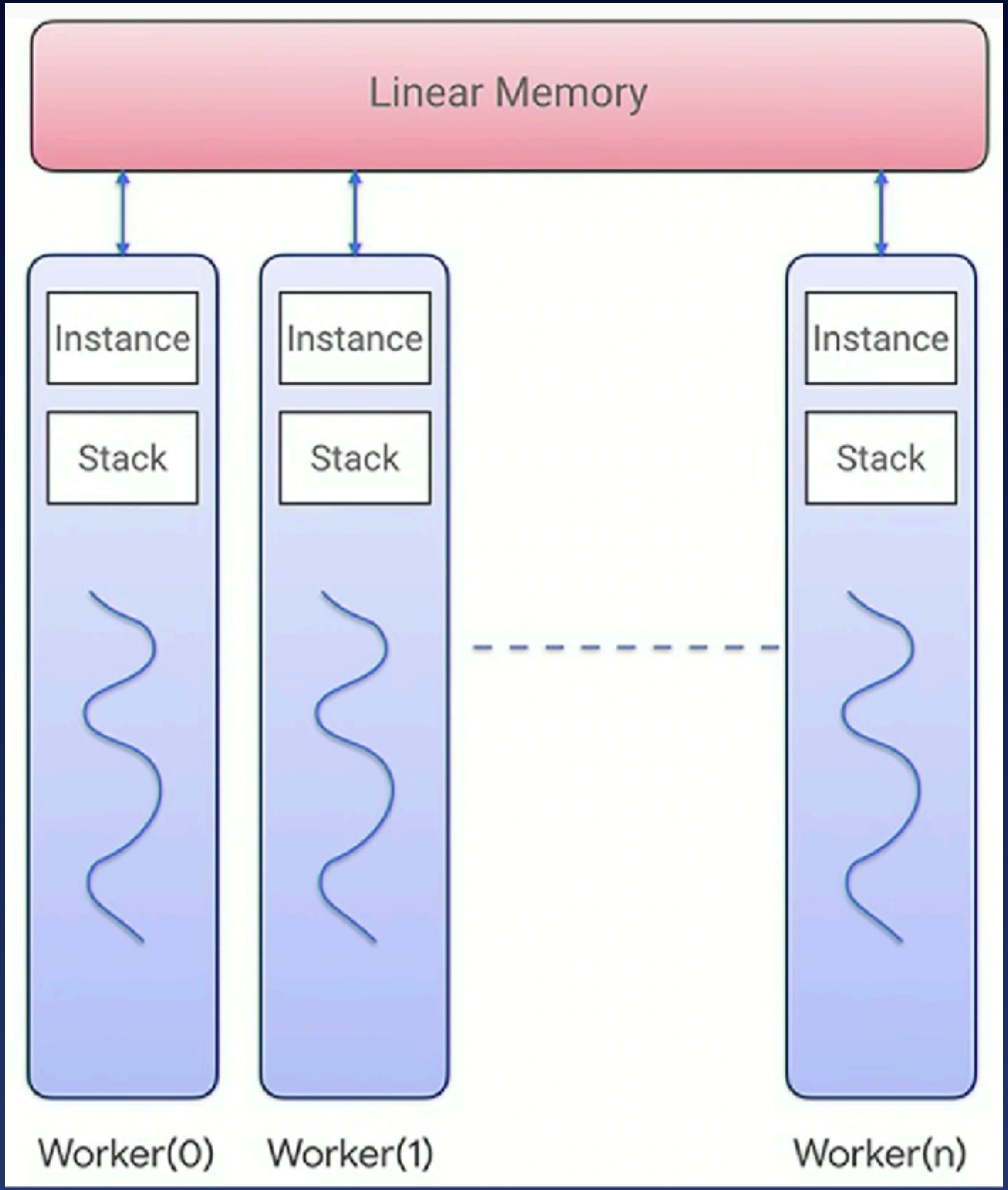
代码库复用

PayPal™

WebAssembly Post-MVP

WebAssembly Thread (Chrome74)

- i32.atomic.load8_u 等原子操作；
- i32.atomic.wait 可用于实现互斥锁；
- 可用于移植 Pthreads 多线程；
- SharedArrayBuffer 共享内存；

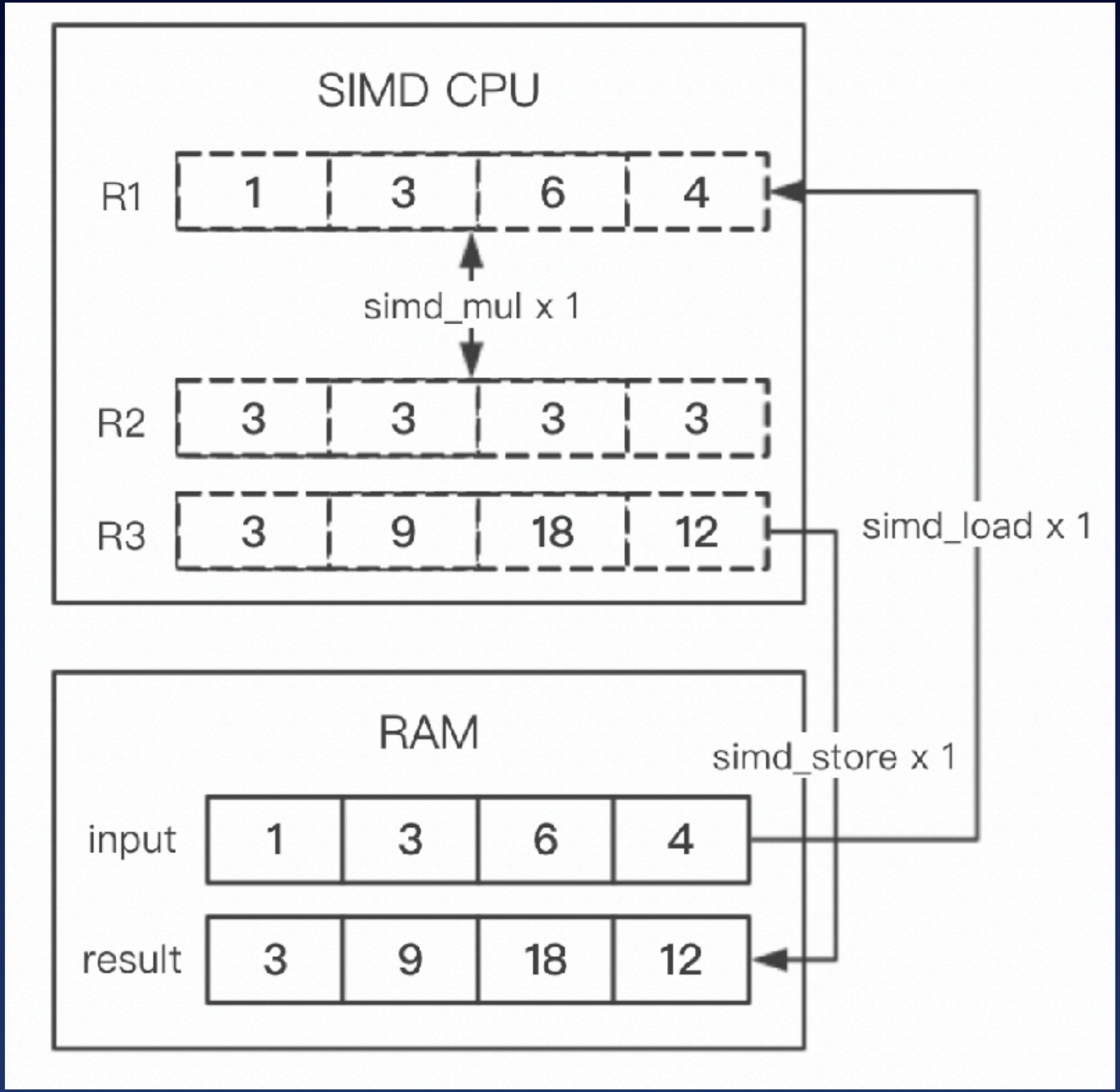


PayPal™

WebAssembly Post-MVP

WebAssembly 128-bit SIMD

- 固定 128 位（bit）；
- i8x16.add(a: v128, b: v128) -> v128；



WebAssembly Post-MVP

Reference Types &

WebIDL Binding &

Wasm Interface Types &

GC

- 新的 “anyref / funcref” 类型，用于引用宿主值；
- 更好地与宿主（比如浏览器）进行交互；
- 为 Wasm 与平台之间提供统一的类型标准；
- …

WebAssembly Post-MVP

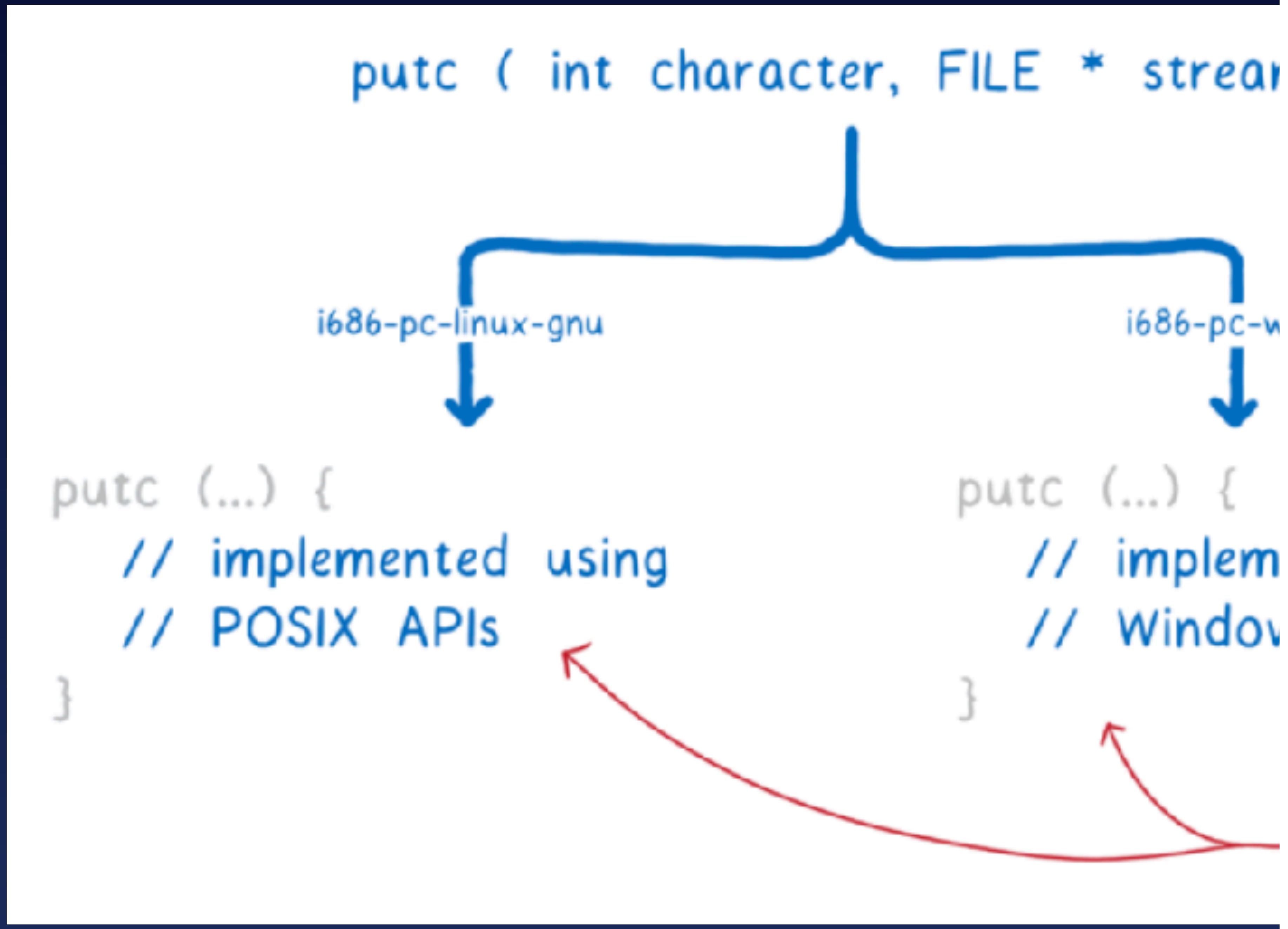
- Tail Call Optimization;
- Custom Annotation Syntax in the Text Format;
- Garbage collection;
- Exception handling;
- JavaScript BigInt to WebAssembly i64 integration;
- …

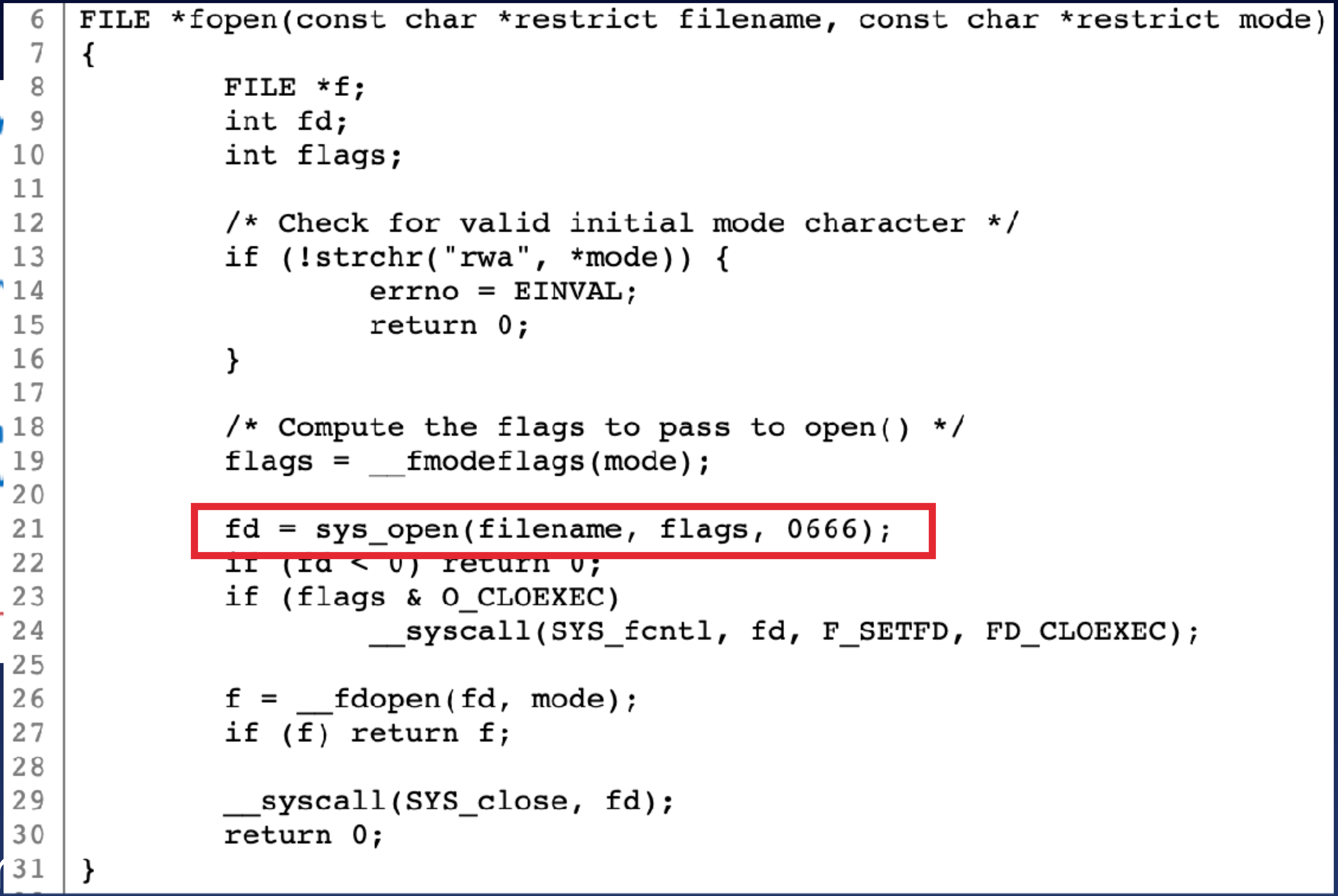
Wasm 虚拟机 - WASI

WASI

WebAssembly System Interface

传统接口抽象（musl）



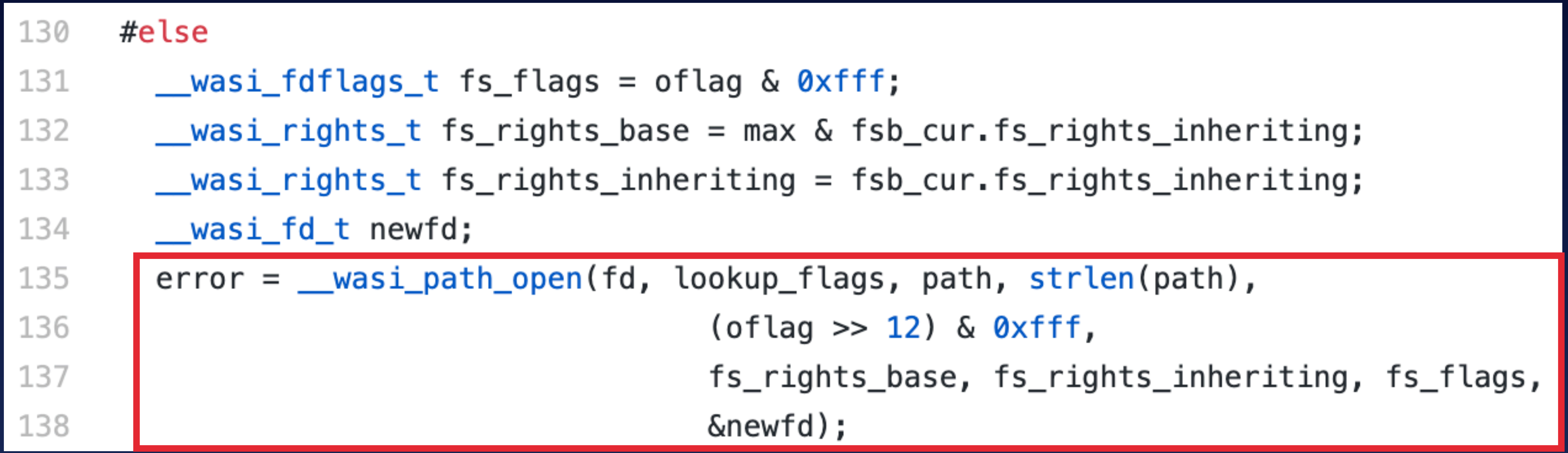


```
FILE *fopen(const char *restrict filename, const char *restrict mode)
{
        FILE *f;
        int fd;
        int flags;

        /* Check for valid initial mode character */
        if (!strchr("rwa", *mode)) {
                errno = EINVAL;
                return 0;
        }

        /* Compute the flags to pass to open() */
        flags = __fmodeflags(mode);

        fd = sys_open(filename, flags, 0666);
        if (fd < 0) return 0;
        if (flags & O_CLOEXEC)
                __syscall(SYS_fcntl, fd, F_SETFD, FD_CLOEXEC);

        f = __fdopen(fd, mode);
        if (f) return f;

        __syscall(SYS_close, fd);
        return 0;
}
```

System Calls

File / Socket / Memory / Pr

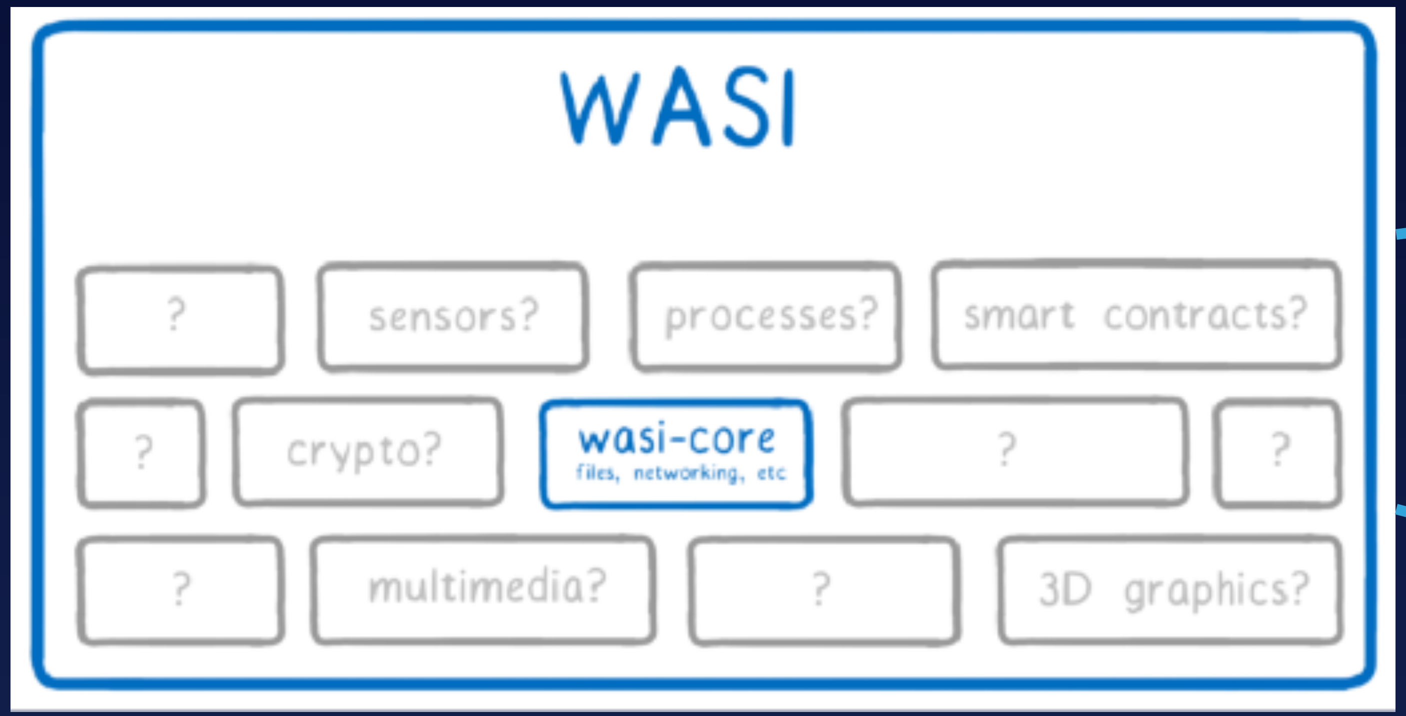
WASI - 接口抽象（wasi-libc）

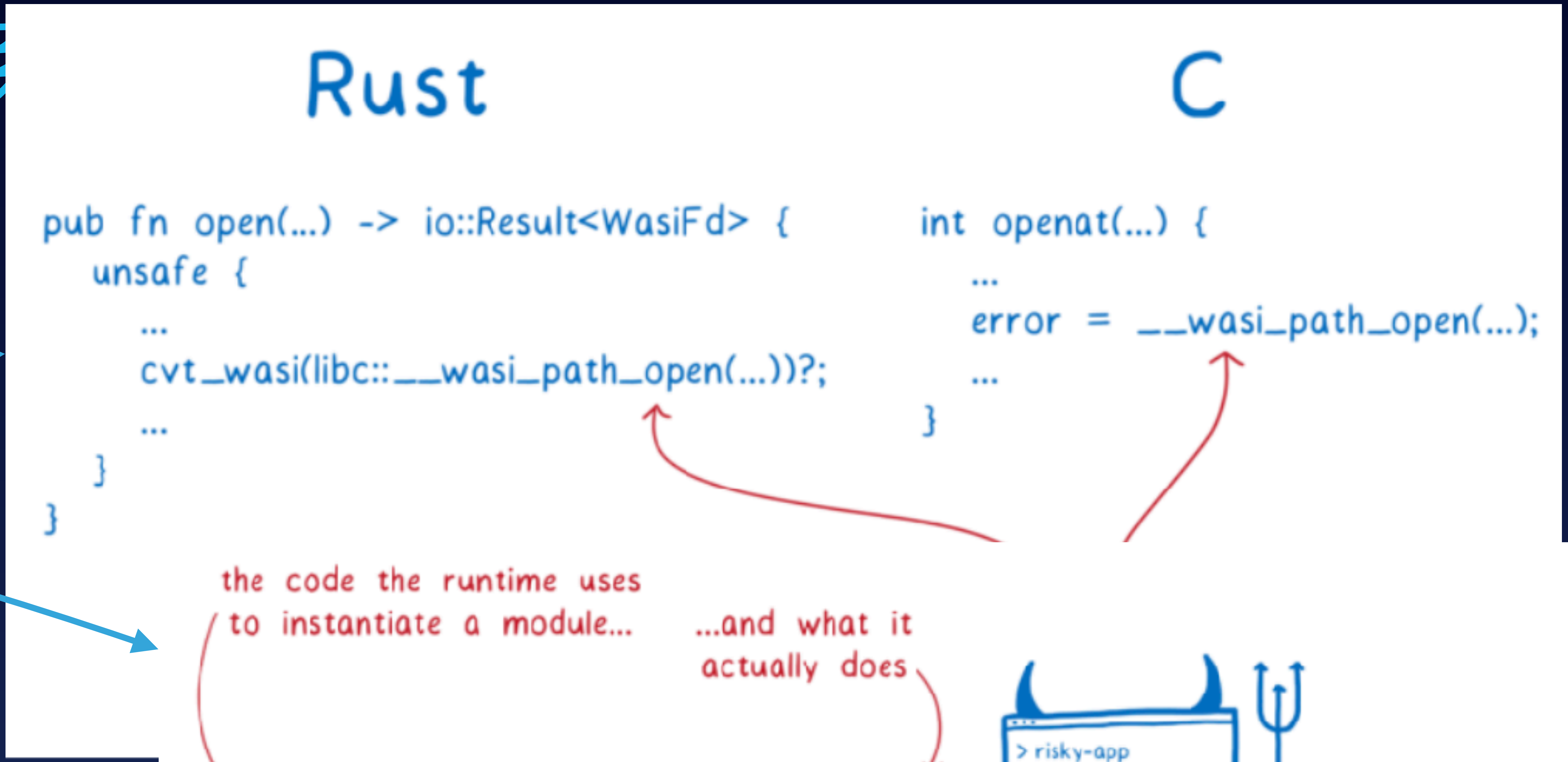


```c
#else
  __wasi_fdflags_t fs_flags = oflag & 0xfff;
  __wasi_rights_t fs_rights_base = max & fsb_cur.fs_rights_inheriting;
  __wasi_rights_t fs_rights_inheriting = fsb_cur.fs_rights_inheriting;
  __wasi_fd_t newfd;
  error = __wasi_path_open(fd, lookup_flags, path, strlen(path),
                               (oflag >> 12) & 0xfff,
                               fs_rights_base, fs_rights_inheriting, fs_flags,
                               &newfd);
```

__wasi_* ➡ WASI 标准

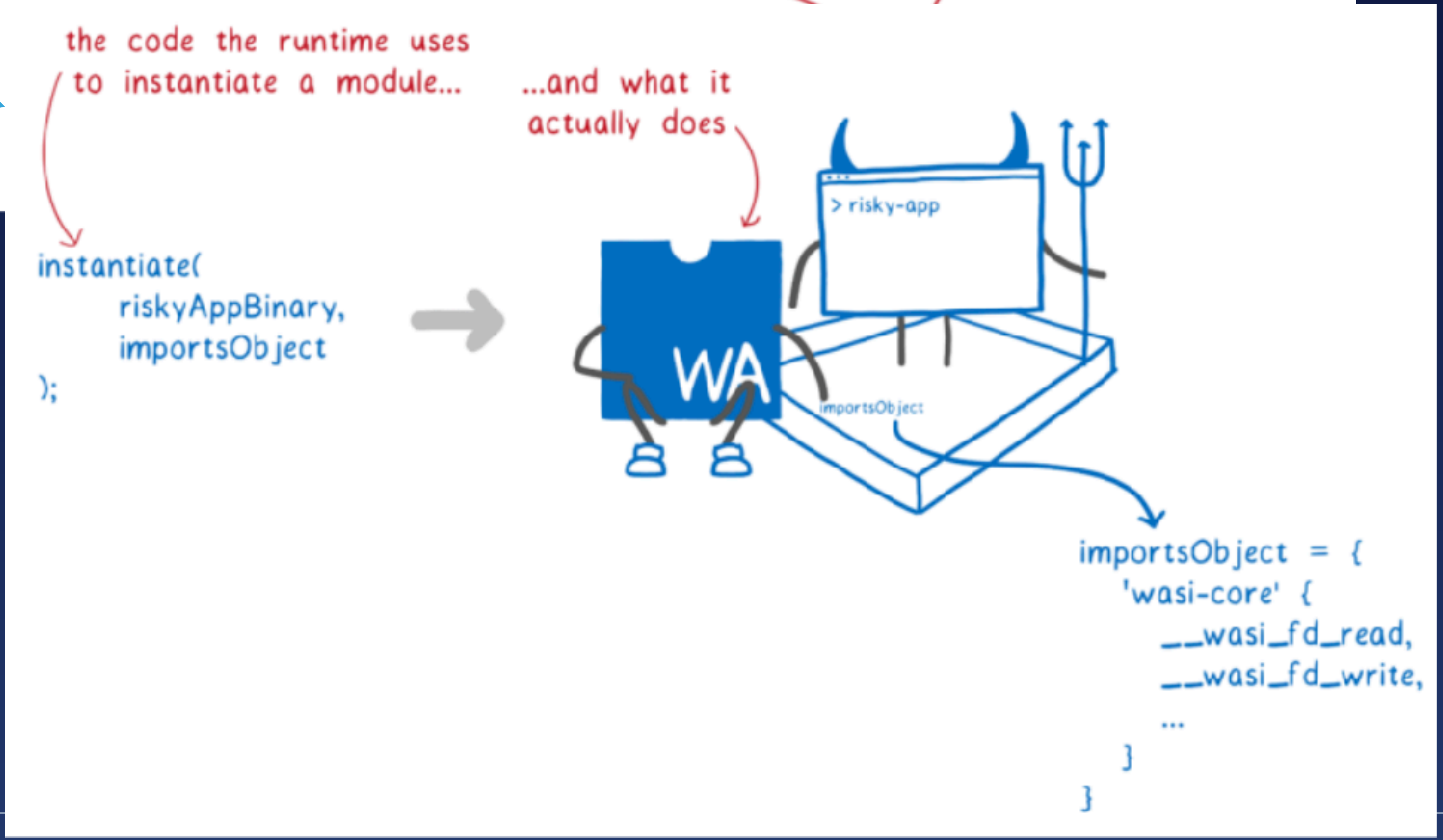
WASI - 接口调用关





```
pub fn open(...) -> io::Result<WasiFd> {
  unsafe {
    ...
    cvt_wasi(libc::__wasi_path_open(...))?;
    ...
  }
}
```

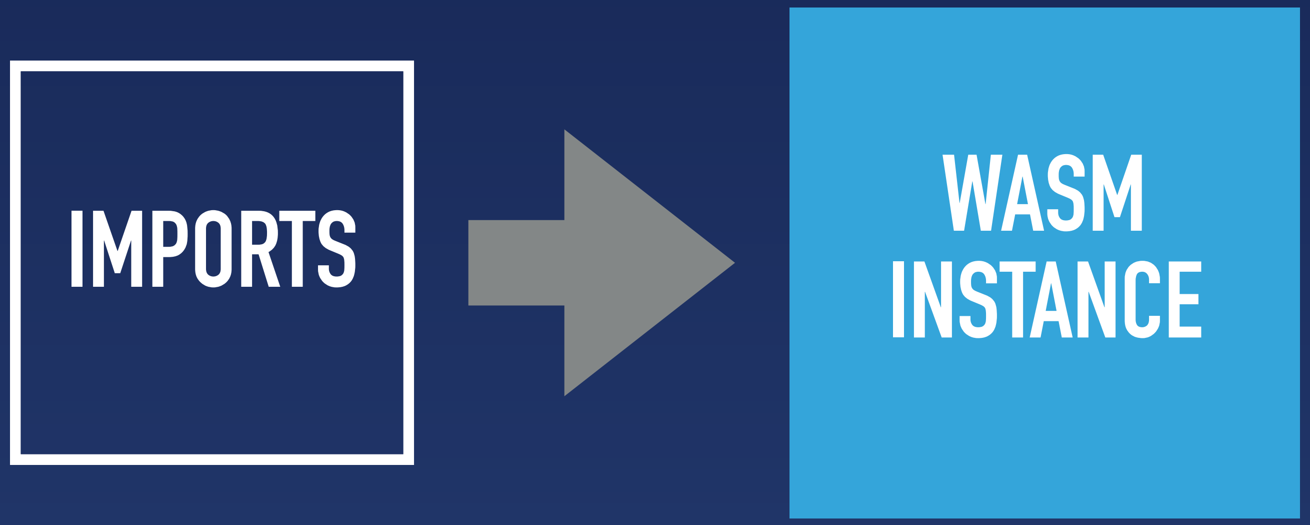
```
int openat(...) {
  ...
  error = __wasi_path_open(...);
  ...
}
```



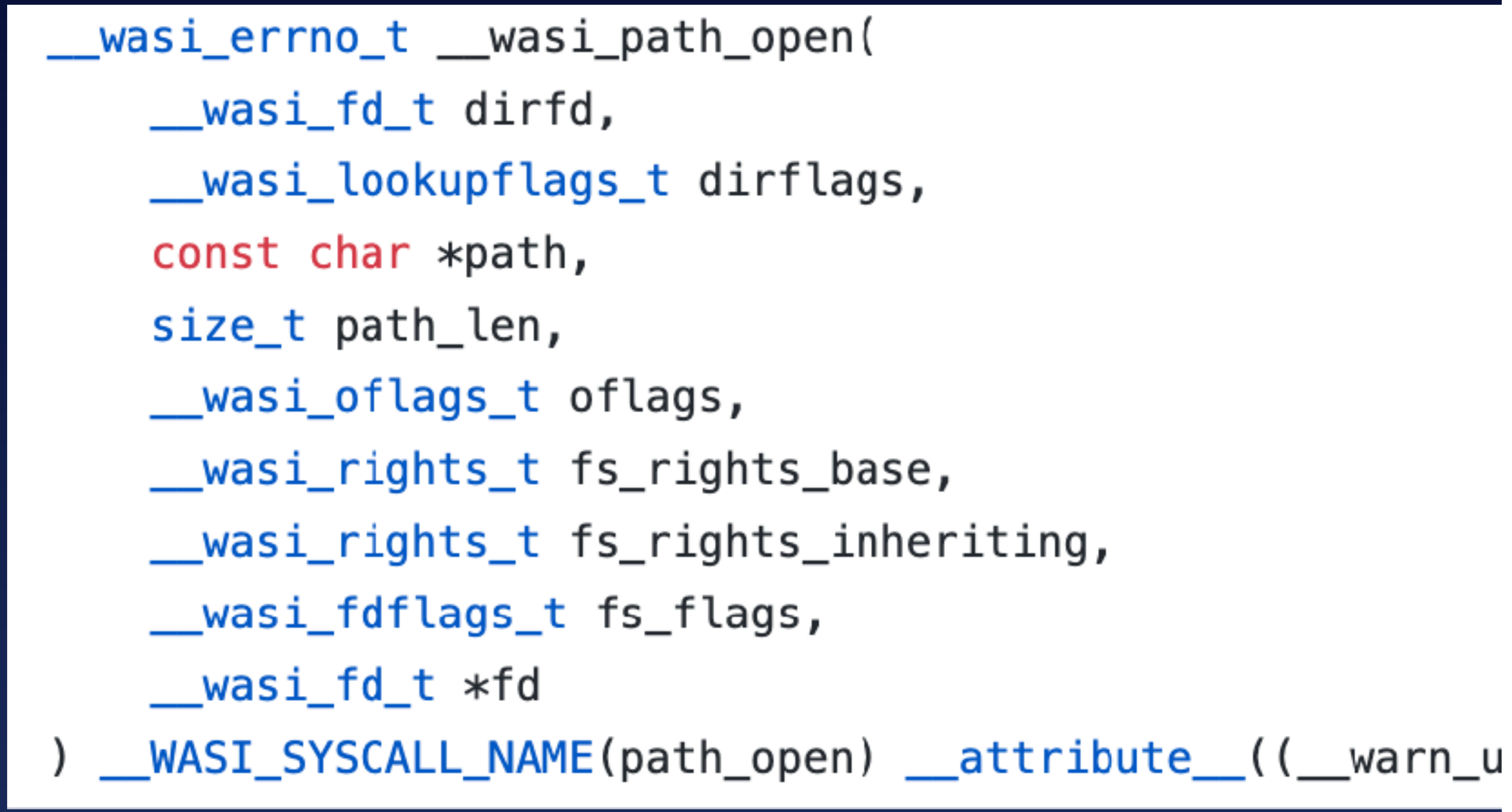
```
instantiate(
    riskyAppBinary,
    importsObject
);
```

```
importsObject = {
  'wasi-core' {
    __wasi_fd_read,
    __wasi_fd_write,
    ...
  }
}
```

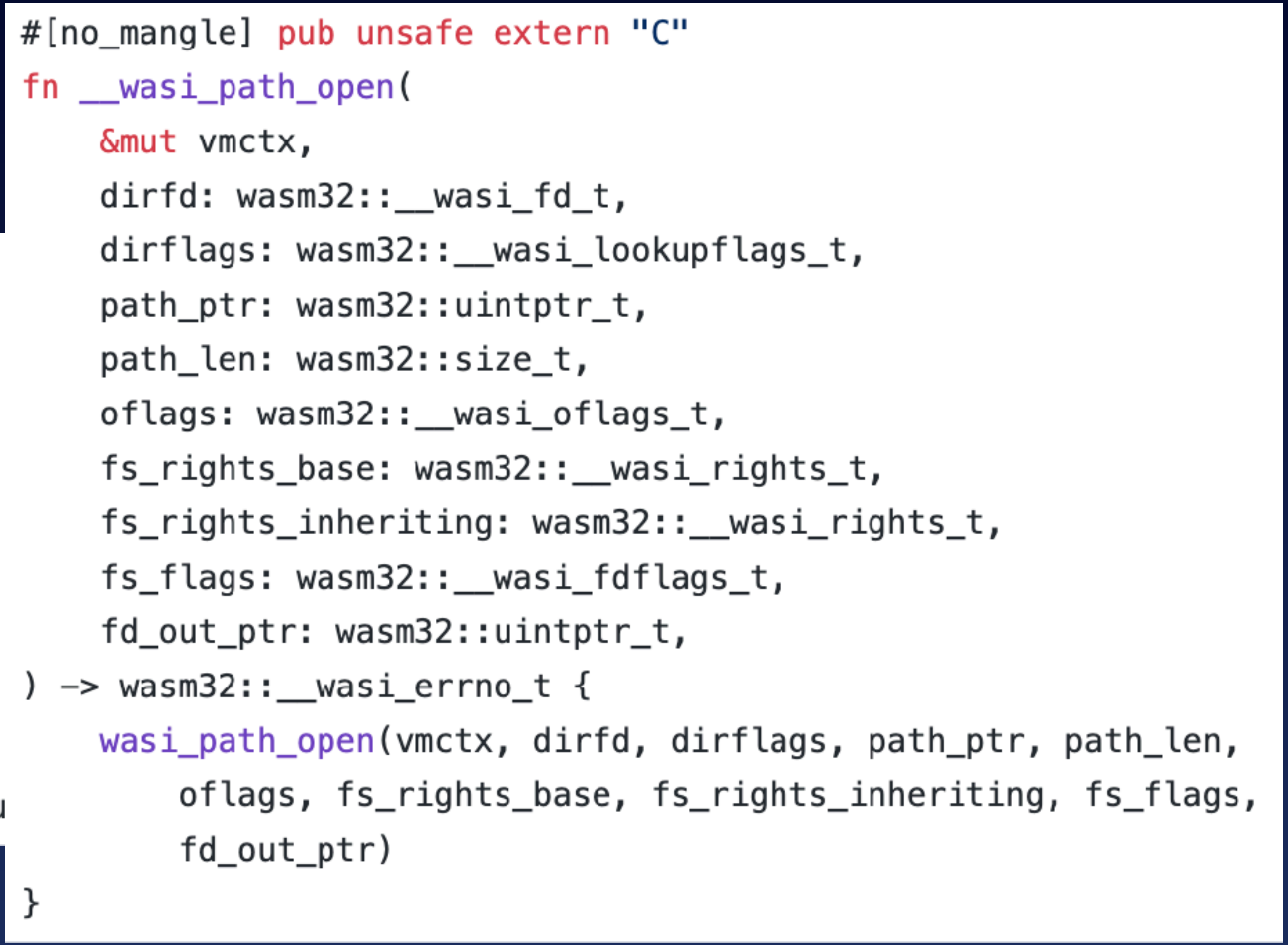
__wasi_path_open



WASI - 宿主实现细节



```c
__wasi_errno_t __wasi_path_open(
    __wasi_fd_t dirfd,
    __wasi_lookupflags_t dirflags,
    const char *path,
    size_t path_len,
    __wasi_oflags_t oflags,
    __wasi_rights_t fs_rights_base,
    __wasi_rights_t fs_rights_inheriting,
    __wasi_fdflags_t fs_flags,
    __wasi_fd_t *fd
) __WASI_SYSCALL_NAME(path_open) __attribute__((__warn_u
```



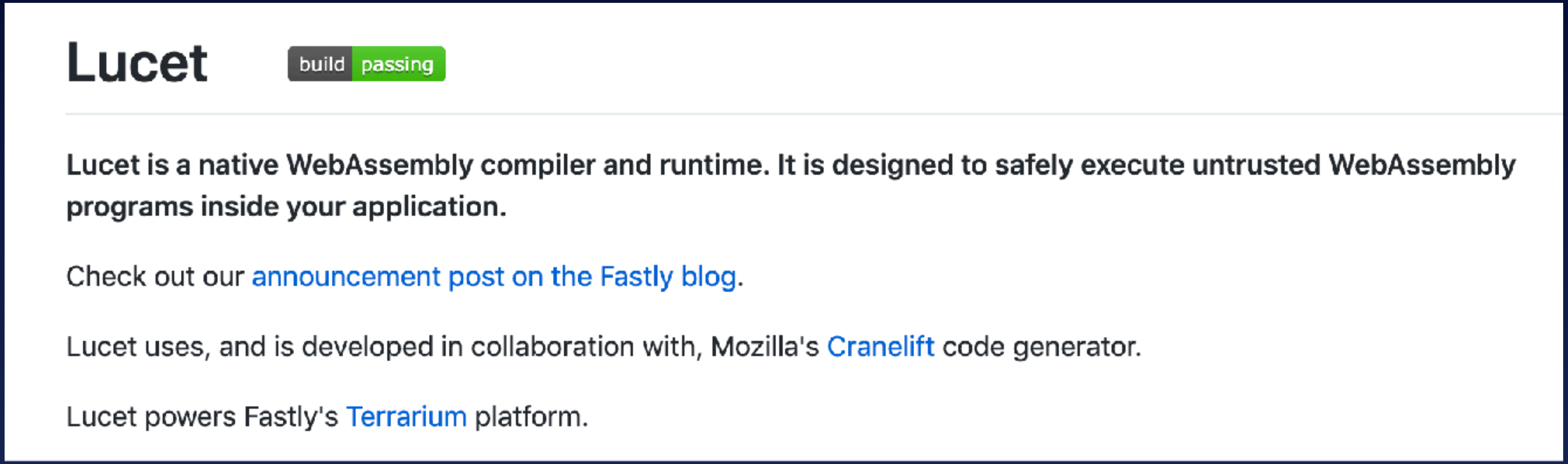
```rust
#[no_mangle] pub unsafe extern "C"
fn __wasi_path_open(
    &mut vmctx,
    dirfd: wasm32::__wasi_fd_t,
    dirflags: wasm32::__wasi_lookupflags_t,
    path_ptr: wasm32::uintptr_t,
    path_len: wasm32::size_t,
    oflags: wasm32::__wasi_oflags_t,
    fs_rights_base: wasm32::__wasi_rights_t,
    fs_rights_inheriting: wasm32::__wasi_rights_t,
    fs_flags: wasm32::__wasi_fdflags_t,
    fd_out_ptr: wasm32::uintptr_t,
) -> wasm32::__wasi_errno_t {
    wasi_path_open(vmctx, dirfd, dirflags, path_ptr, path_len,
        oflags, fs_rights_base, fs_rights_inheriting, fs_flags,
        fd_out_ptr)
}
```

wasi-libc 函数定义

Lucet 宿主函数实现（Rust）

PayPal™

WASI - Lucet 宿主实现



Lucet build passing

Lucet is a native WebAssembly compiler and runtime. It is designed to safely execute untrusted WebAssembly programs inside your application.

Check out our announcement post on the Fastly blog.

Lucet uses, and is developed in collaboration with, Mozilla's Cranelift code generator.

Lucet powers Fastly's Terrarium platform.

一个 WebAssembly Compiler

&& (WASI)

一个 WebAssembly Runtime

PayPal™

WASI - Lucet 一个例子 - C/C++

宿主依赖的

文件操作

```
#include <stdio.h>
int main(int argc, char** argv) {
    FILE * file;
    if ((file = fopen("lucent-wasi", "w+"))) {
        fputs("Hello CAP!\n", file);
    }
    return 0;
}
```

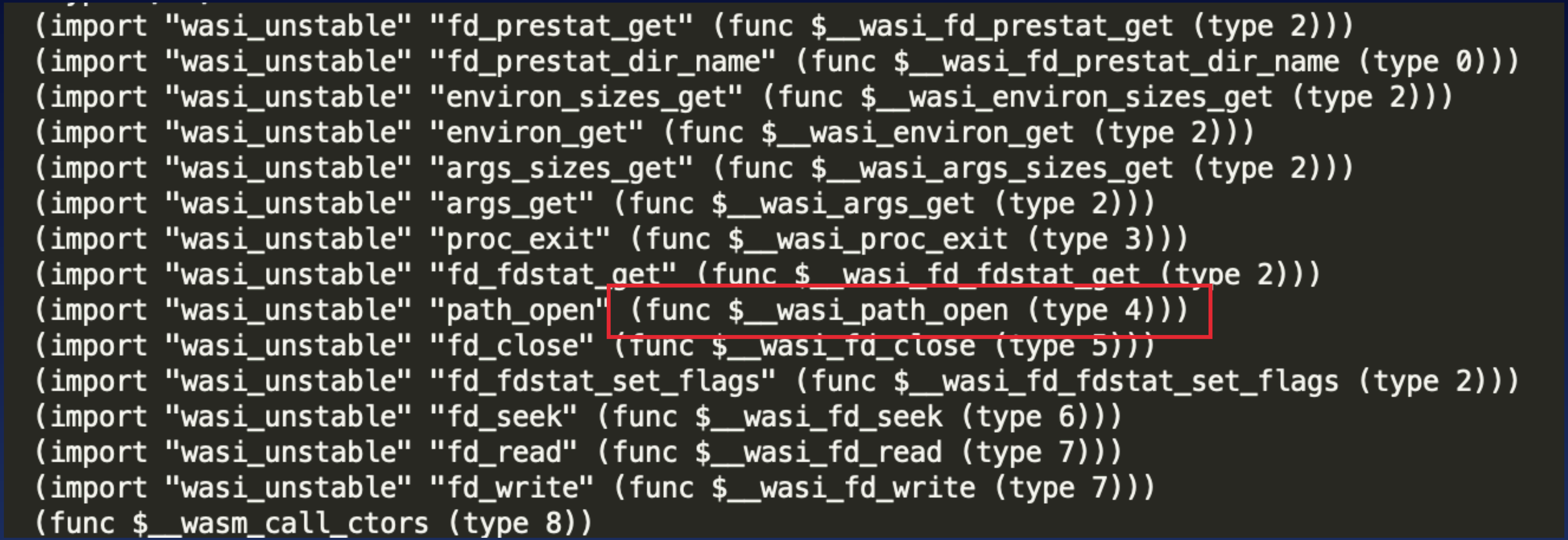
PayPal™

WASI - Lucet 一个例子 - 编译和运行

```
wasm32-unknown-wasi-clang hello.c -o hello.wasm
lucetc-wasi hello.wasm -o hello.so
lucet-wasi hello.so --dir .:.
```

指定目录映射关系

WASI - Lucet 一个例子 - WAT 细节



```
(import "wasi_unstable" "fd_prestat_get" (func $__wasi_fd_prestat_get (type 2)))
(import "wasi_unstable" "fd_prestat_dir_name" (func $__wasi_fd_prestat_dir_name (type 0)))
(import "wasi_unstable" "environ_sizes_get" (func $__wasi_environ_sizes_get (type 2)))
(import "wasi_unstable" "environ_get" (func $__wasi_environ_get (type 2)))
(import "wasi_unstable" "args_sizes_get" (func $__wasi_args_sizes_get (type 2)))
(import "wasi_unstable" "args_get" (func $__wasi_args_get (type 2)))
(import "wasi_unstable" "proc_exit" (func $__wasi_proc_exit (type 3)))
(import "wasi_unstable" "fd_fdstat_get" (func $__wasi_fd_fdstat_get (type 2)))
(import "wasi_unstable" "path_open" (func $__wasi_path_open (type 4)))
(import "wasi_unstable" "fd_close" (func $__wasi_fd_close (type 5)))
(import "wasi_unstable" "fd_fdstat_set_flags" (func $__wasi_fd_fdstat_set_flags (type 2)))
(import "wasi_unstable" "fd_seek" (func $__wasi_fd_seek (type 6)))
(import "wasi_unstable" "fd_read" (func $__wasi_fd_read (type 7)))
(import "wasi_unstable" "fd_write" (func $__wasi_fd_write (type 7)))
(func $__wasm_call_ctors (type 8))
```

Wasm 总结建议

找到性能瓶颈，选择性使用

做好降级方案，保证可用性

Wasm 观望建议

持续加码， 未来可期

Q&A