WebAssembly - 技术变革，未来已来

于航(曜彤)

阿里巴巴/本地生活

Geekbang. 极客邦科技 InfoQ

极客邦科技
会议推荐2019

5月

QCon 北京
全球软件开发大会
大会：5月6–8日
培训：5月9–10日

QCon 广州
全球软件开发大会
培训：5月25–26日
大会：5月27–28日

6月

GTLC GLOBAL TECH LEADERSHIP CONFERENCE 上海
技术领导力峰会
时间：6月14–15日

GMTC 北京
全球大前端技术大会
大会：6月20–21日
培训：6月22–23日

7月

ArchSummit 深圳
全球架构师峰会
大会：7月12–13日
培训：7月14–15日

10月

QCon 上海
全球软件开发大会
大会：10月17–19日
培训：10月20–21日

11月

GMTC 深圳
全球大前端技术大会
大会：11月8–9日
培训：11月10–11日

AiCon 北京
全球人工智能与机器学习大会
大会：11月21–22日
培训：11月23–24日

12月

ArchSummit 北京
全球架构师峰会
大会：12月6–7日
培训：12月8–9日

自我介绍

于航 (曜彤)

前端@阿里巴巴 / 本地生活

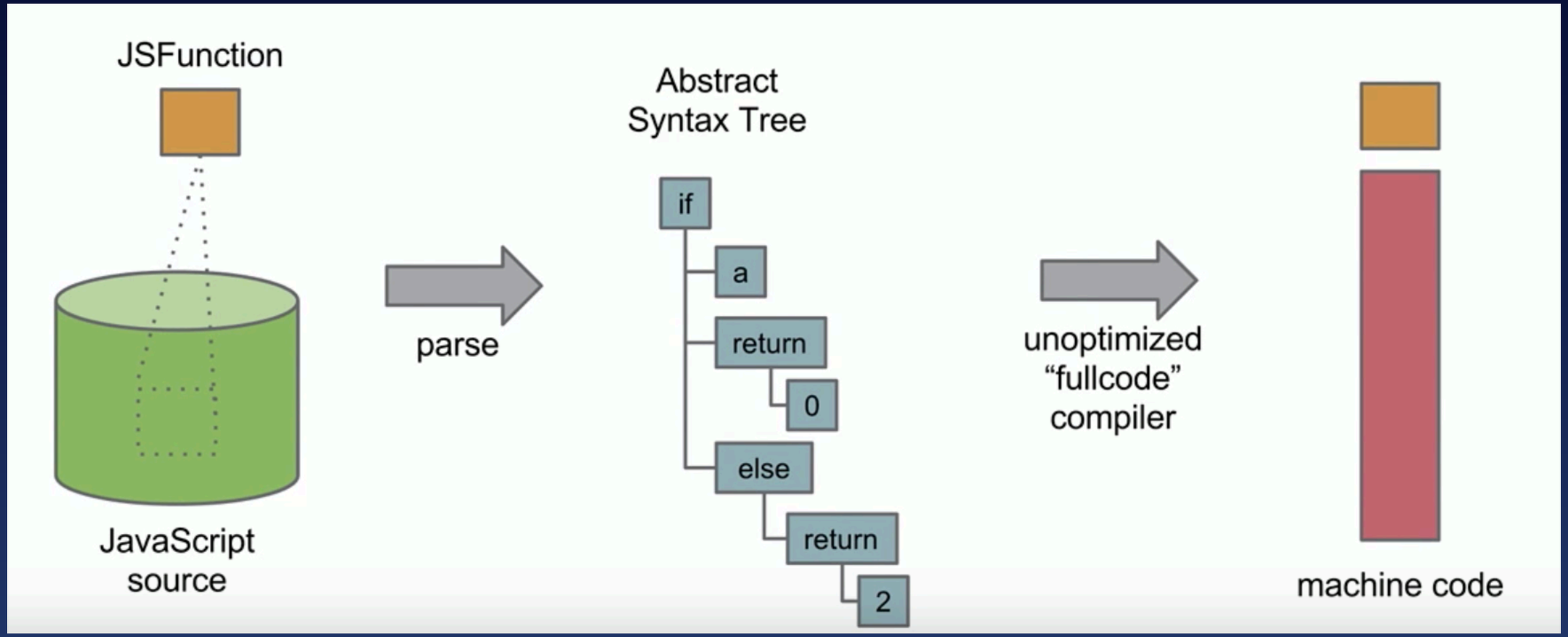
《深入浅出 WEBASSEMBLY》作者

FCC 上海社区负责人

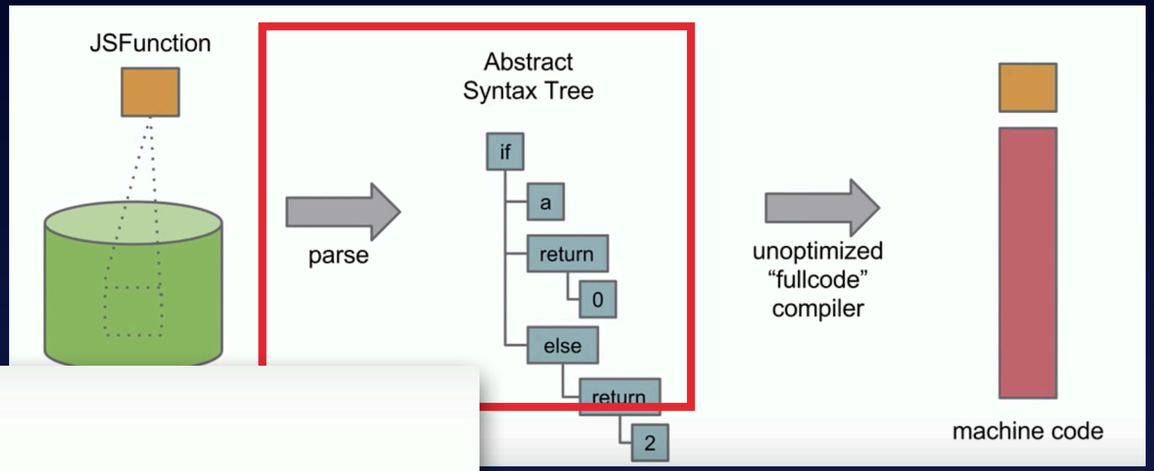
目录

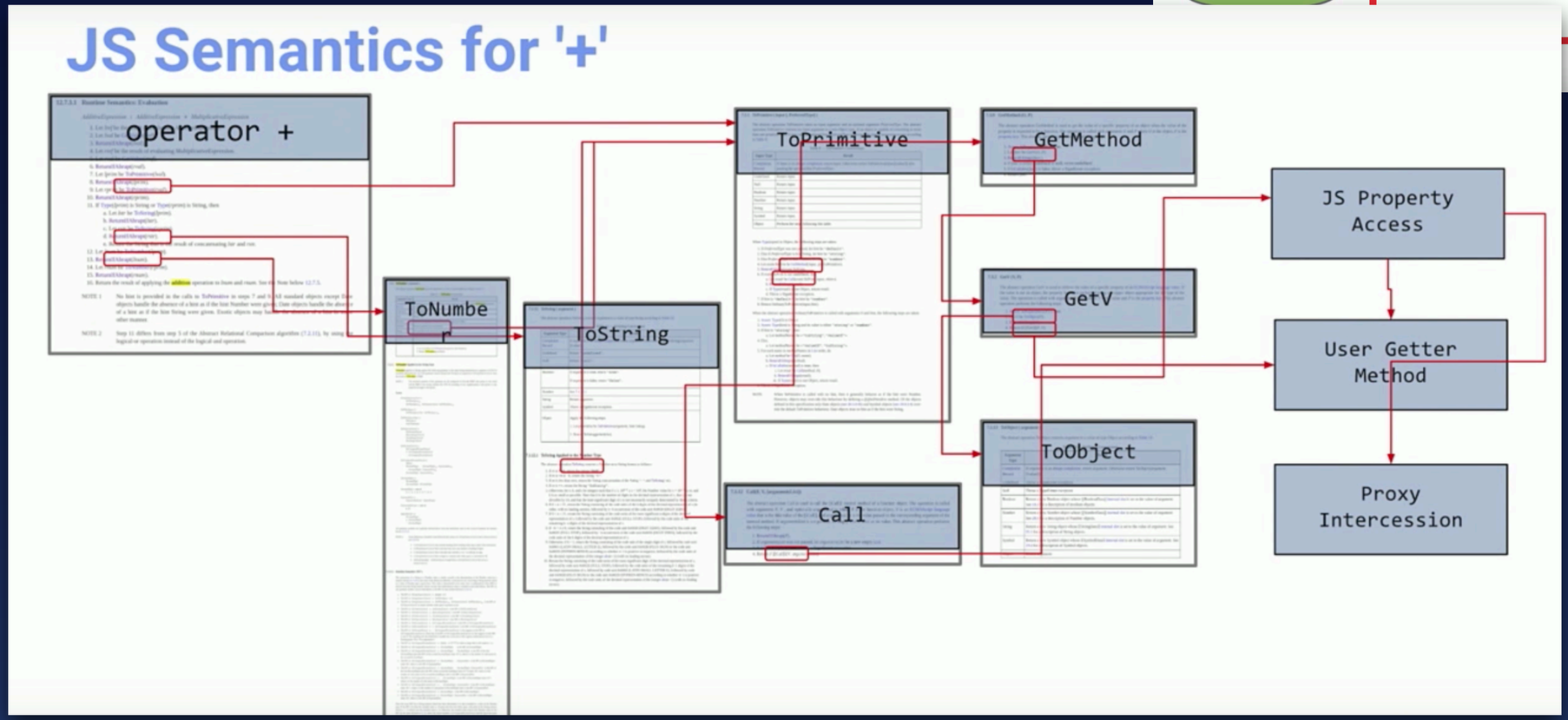
- WebAssembly 简短回顾（背景、原理）；
- 各大公司的 WebAssembly 线上实践；
- Roadmap & Milestone 发展规划；
- 未来可期的 - WASI；

Javascript 部分执行链路（V8）



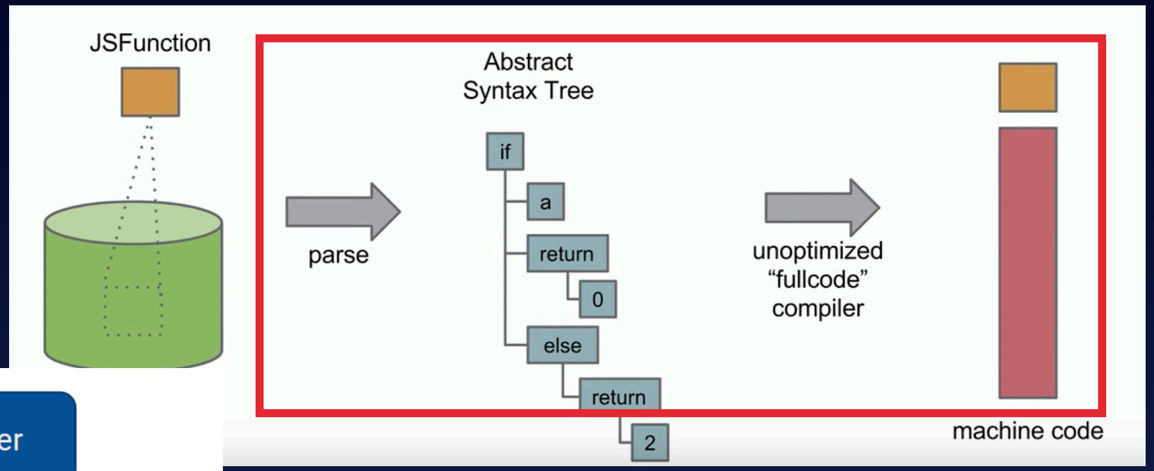
ECMA 规定的“+”执行流程





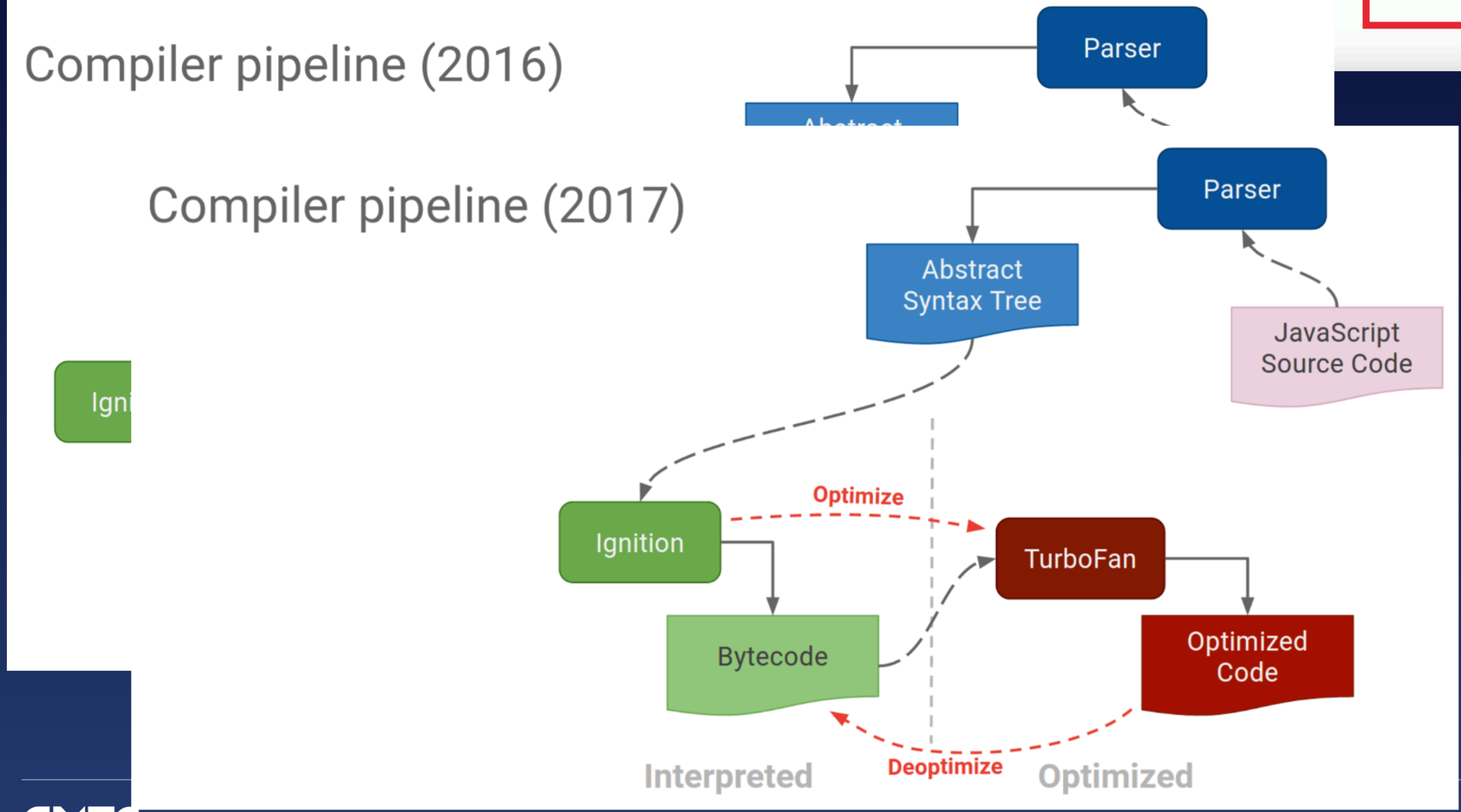
x + y

编译链路优化与问题（V8）



Compiler pipeline (2016)

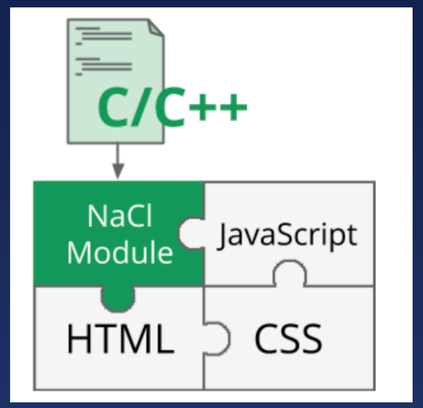
Compiler pipeline (2017)



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；



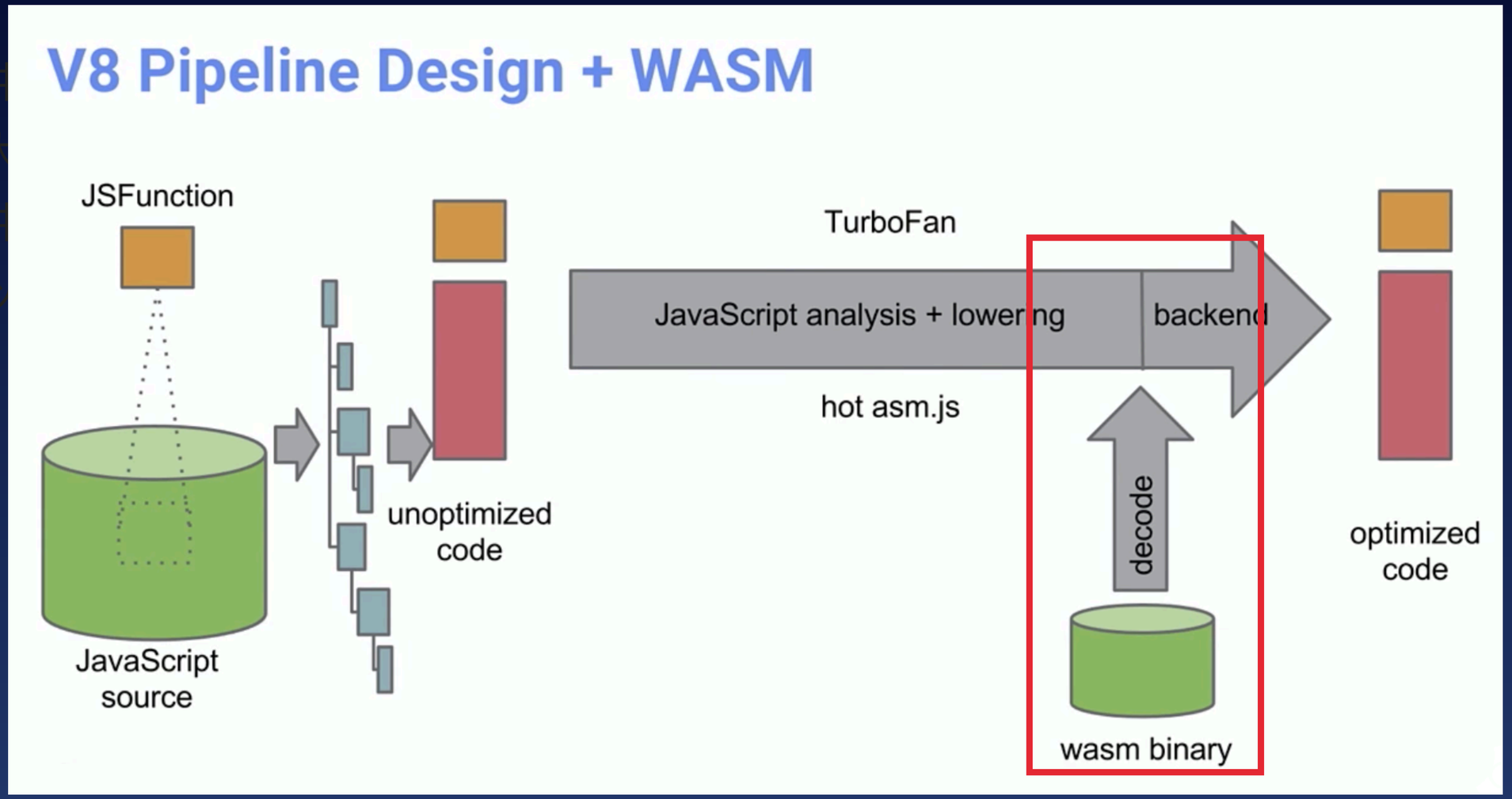
- 提供沙盒环境在浏览器中执行的 C/C++ 代码；
- 充分利用 CPU 的特性，如 SIMD、多核心处理等；
- 平台独立，一次编译到处运行；

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  |....`.....`....`|
```

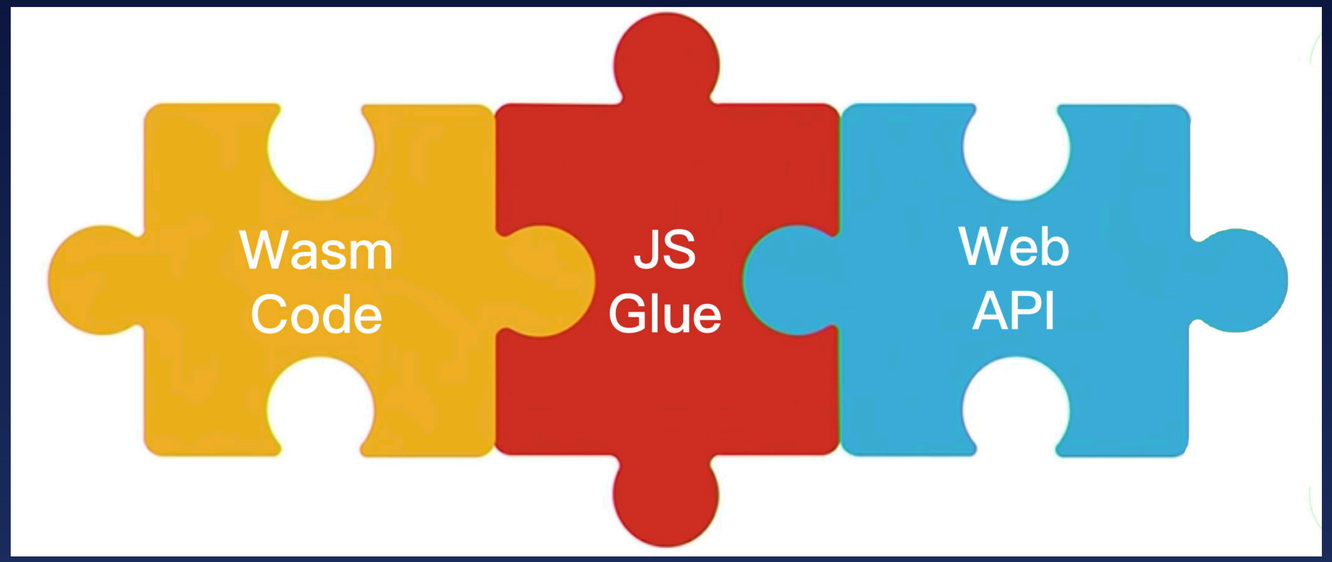
WebAssembly 编译完整链路



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++

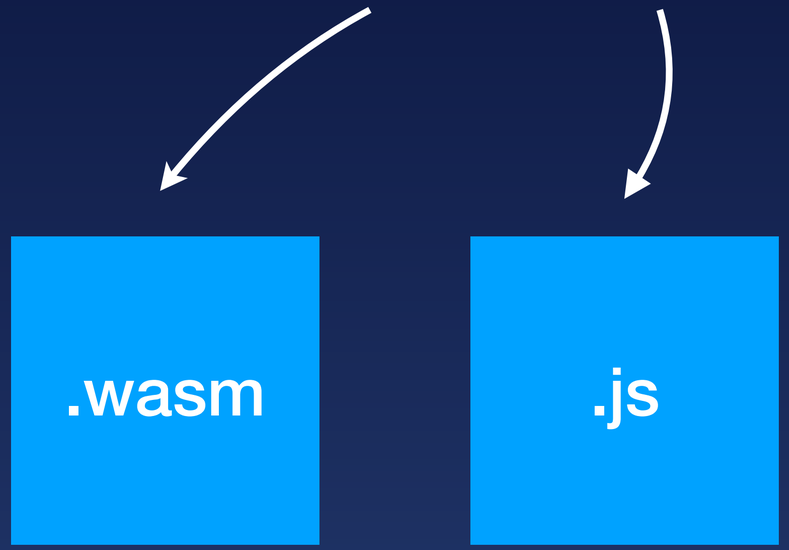
C++

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

一个简单的例子-WAT

WAT

toy.wasm

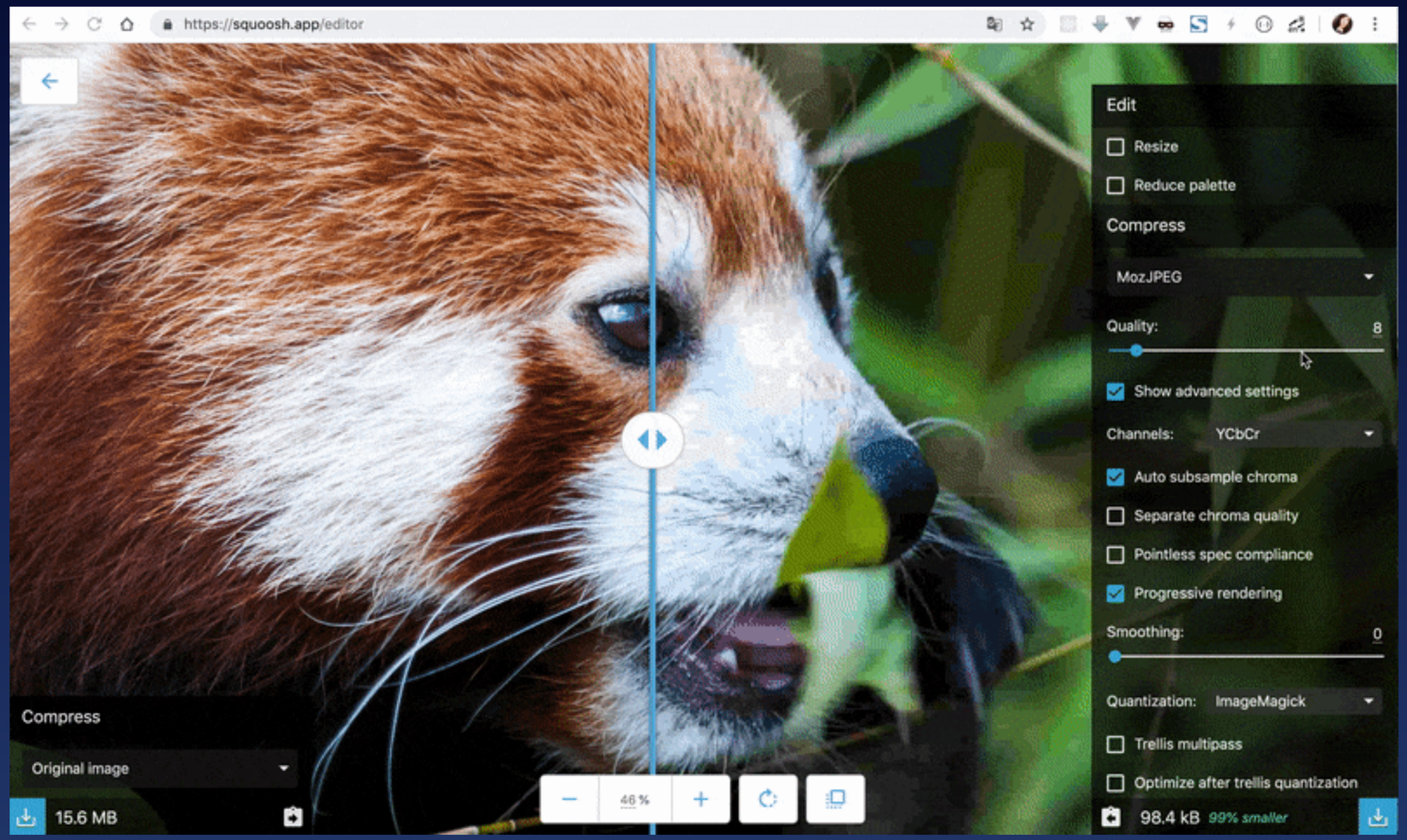
```
(module
  (type (;0;) (func (param i32 i32) (result i32)))
  (func (;0;) (type 0) (param i32 i32) (result i32)
    get_local 1
    get_local 0
    i32.add)
  (export "_add" (func 0)))
```

类型声明段

函数声明段

导出段

WASM 实际应用（一） - 在线图像处理 Squoosh

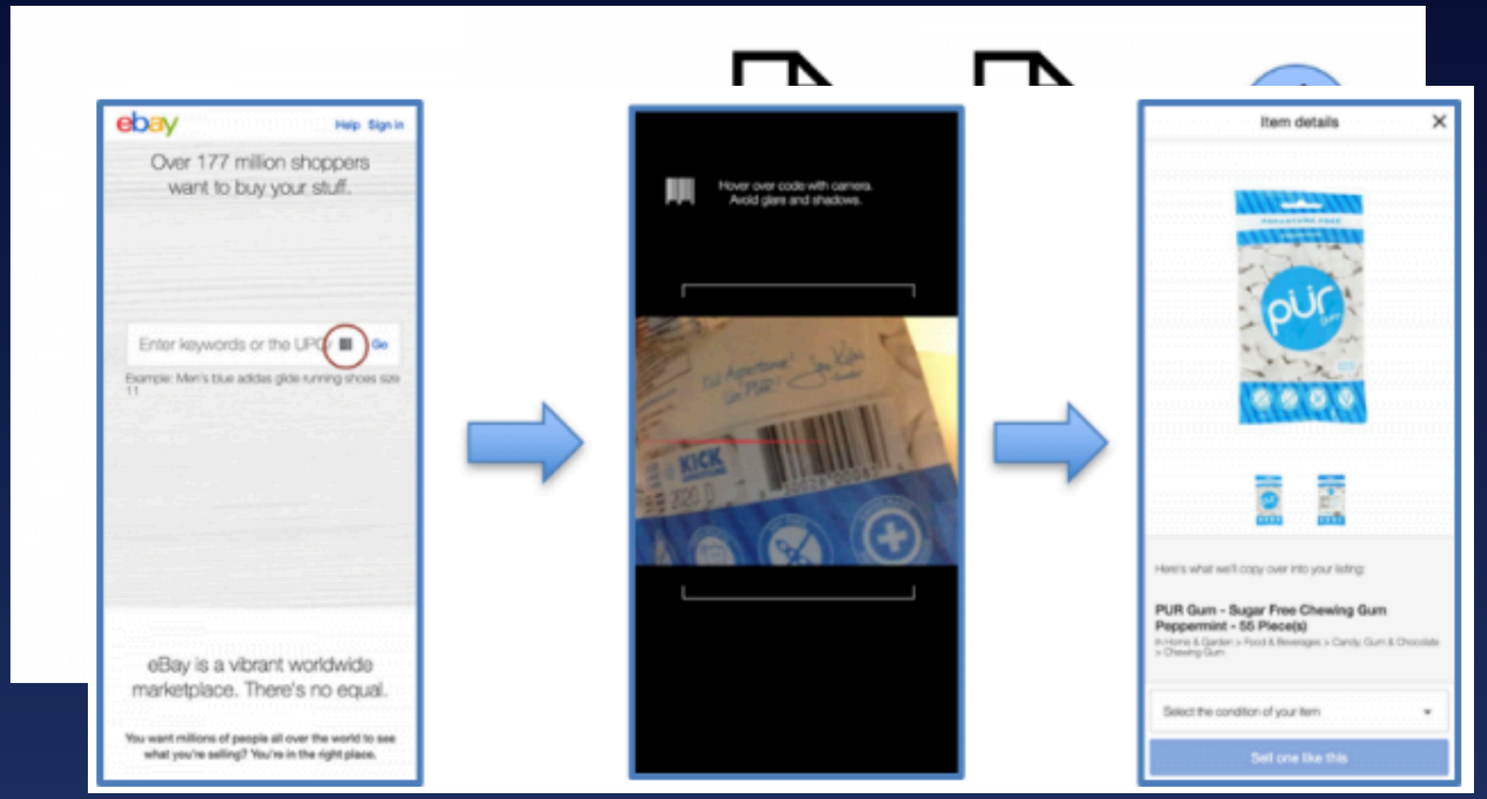


libimagequant (C)

MozJPEG (C++)

webp (C)

WASM 实际应用（二） - Ebay Barcode scanner



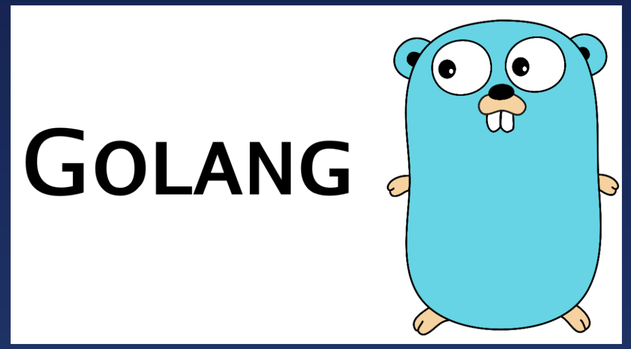
ZBar (C)

Custome Lib (C/C++)

BarcodeReader (JavaScript)

***三个 Worker 对应三种方案，依次竞争；**

Wasm 语言编译器 / 解释器



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：

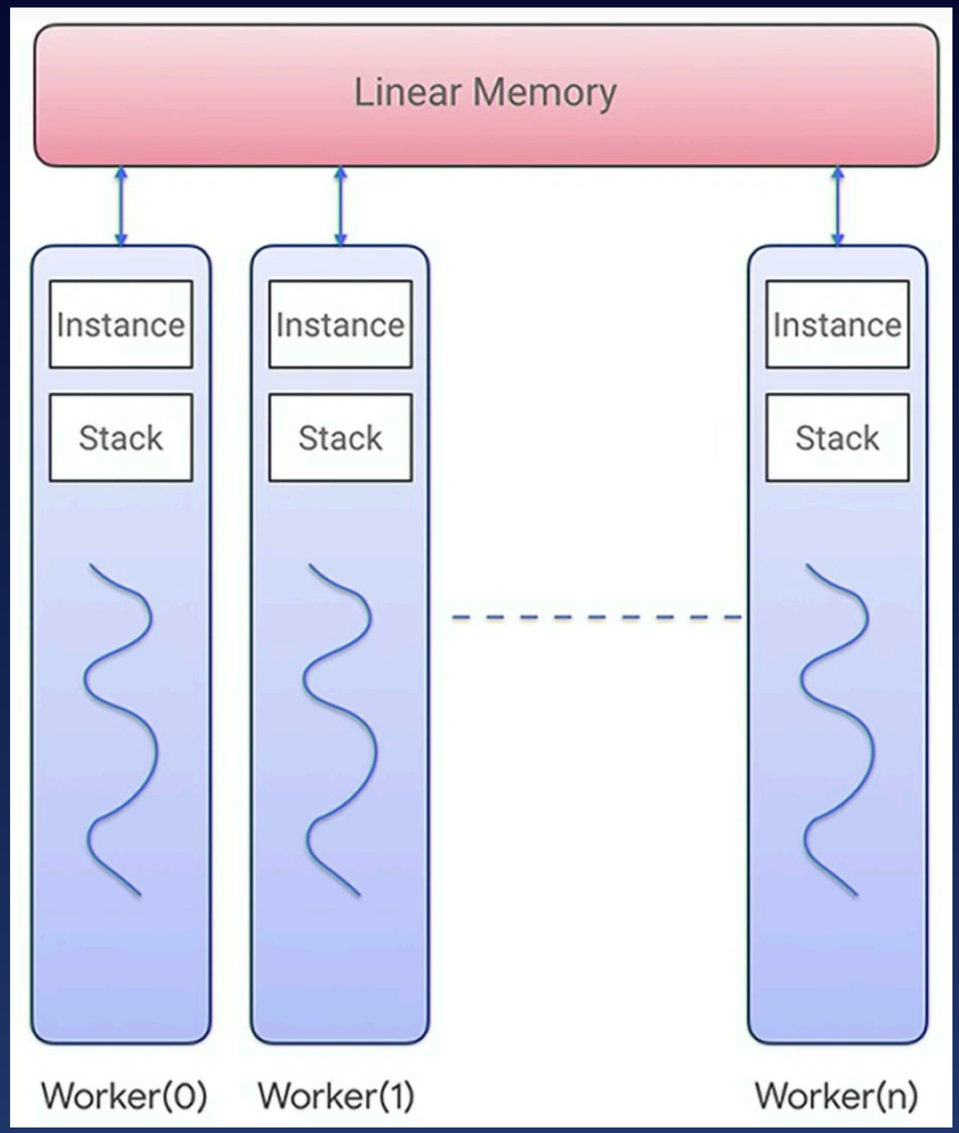
高性能计算

代码库复用

WebAssembly Post-MVP

WebAssembly Thread (Chrome74)

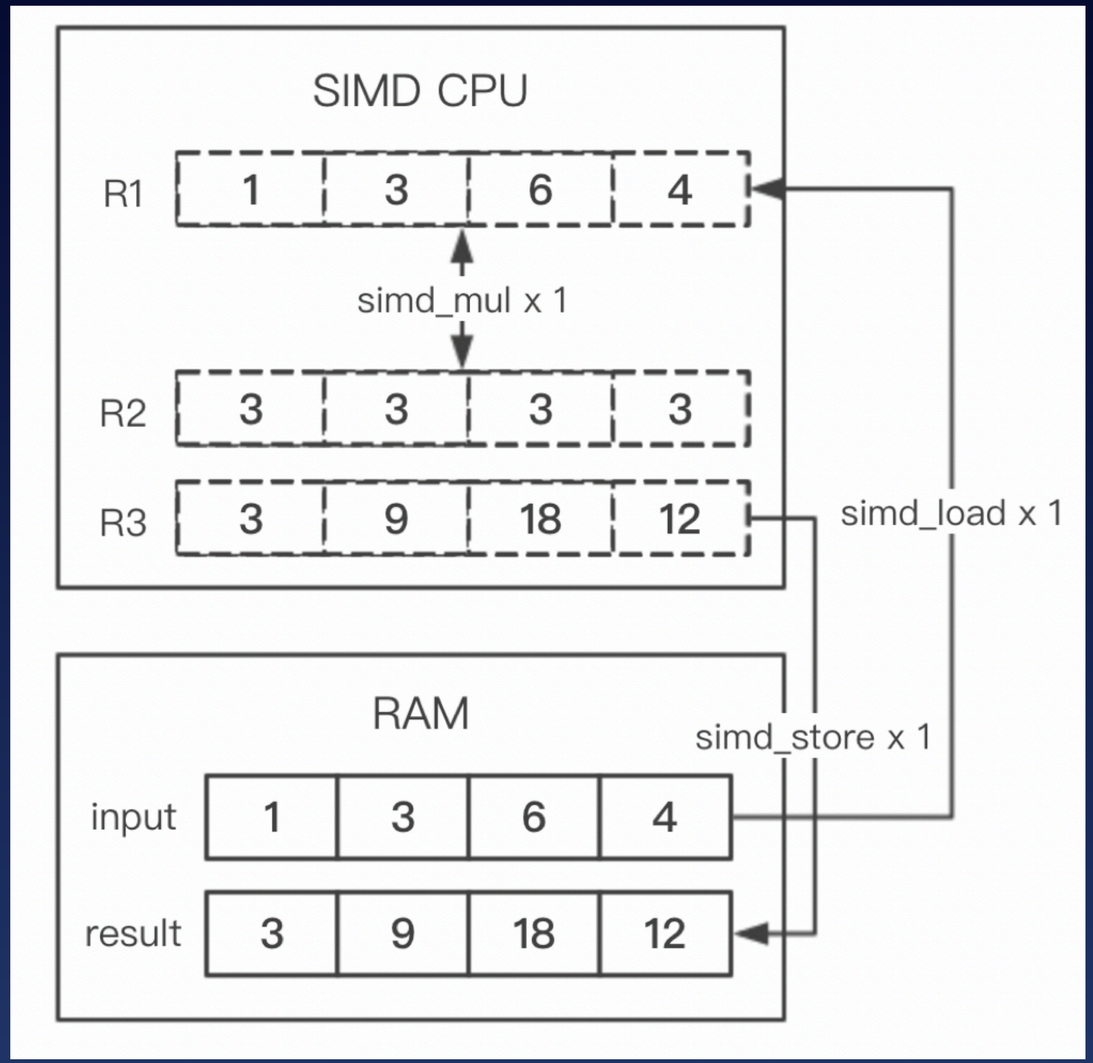
- i32.atomic.load8_u 等原子操作；
- i32.atomic.wait 可用于实现互斥锁；
- 可用于移植 Pthreads 多线程；
- SharedArrayBuffer 共享内存；



WebAssembly Post-MVP

WebAssembly 128-bit SIMD

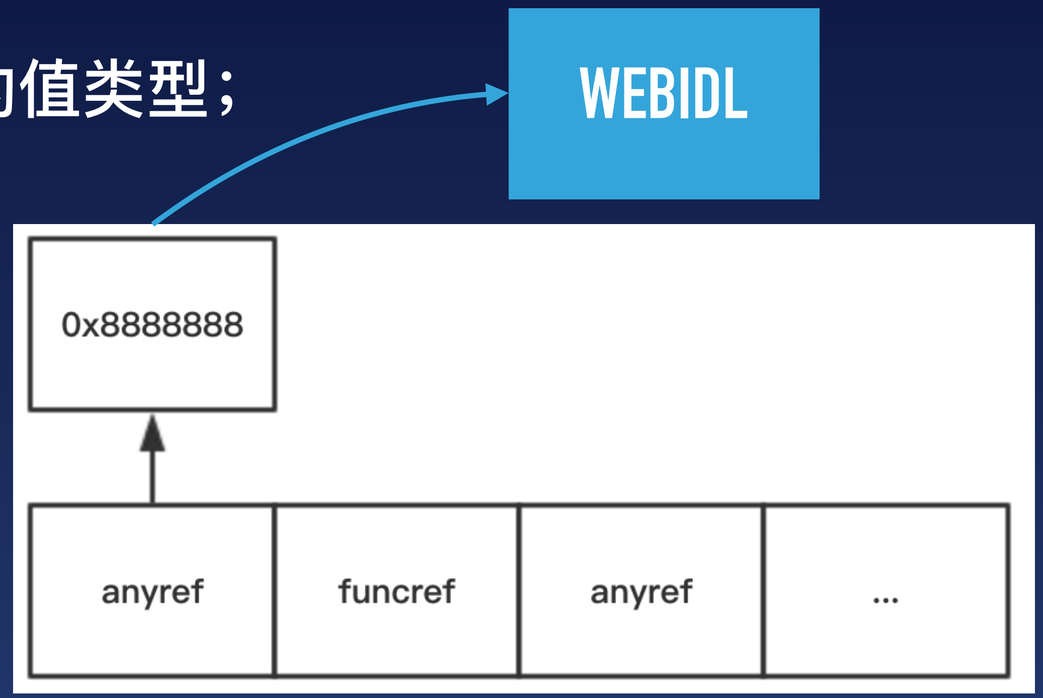
- 固定 128 位（bit）；
- i8x16.add(a: v128, b: v128) -> v128；



WebAssembly Post-MVP

Reference Types

- 新的 “anyref / funcref” 类型，用于引用宿主的值类型；
- 更好地与宿主（比如浏览器）进行交互；
- 所引用的值是不透明且抽象的；



WebAssembly Post-MVP

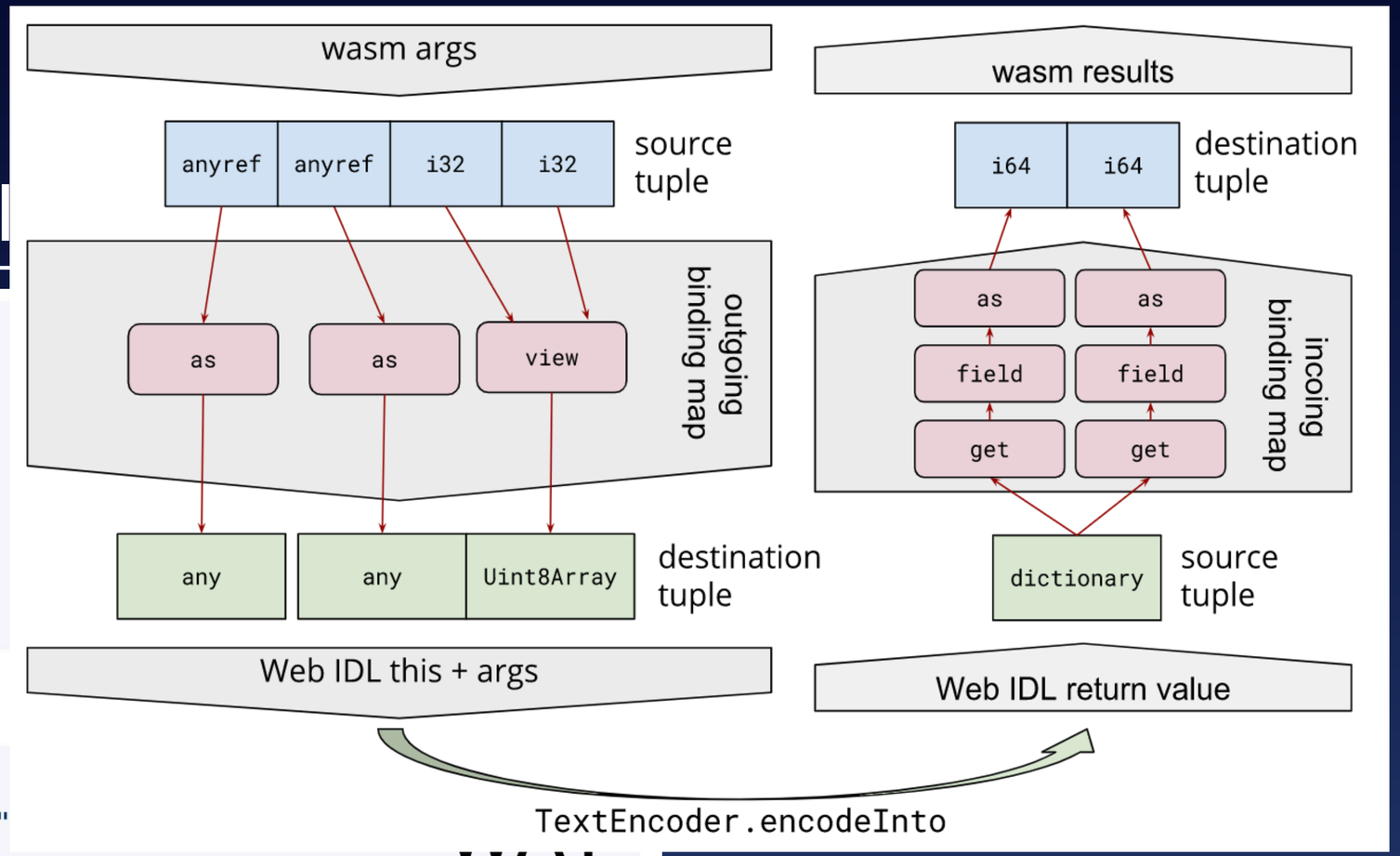
WebIDL Binding Proposal

```
dictionary TextEncoderEncodeIntoResult {
  unsigned long long read;
  unsigned long long written;
};

[Constructor]
interface TextEncoder {
  TextEncoderEncodeIntoResult encodeInto(USVString source,
};
```

TextEncoder.encodeInto

```
(@webidl type $TextEncoderEncodeIntoResult
   (dict (field "read" unsigned long long) (field "written"
(@webidl type $EncodeIntoFuncWebIDL
   (func (method any)
      (param USVString Uint8Array)
      (result (dict $TextEncoderEncodeIntoResult))))
```



WebAssembly Post-MVP

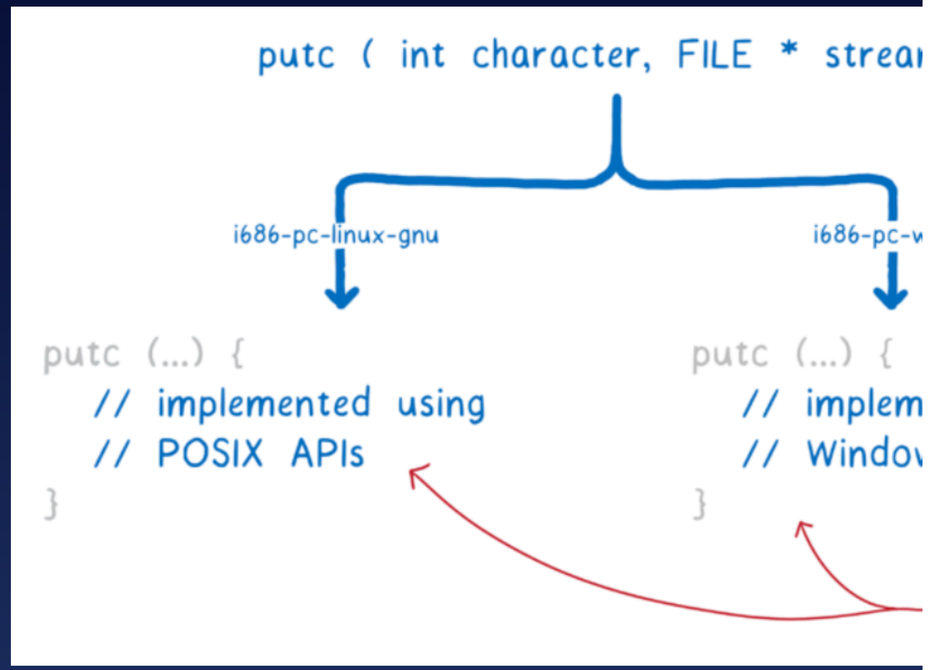
- Tail Call Optimization；
- Custom Annotation Syntax in the Text Format；
- Garbage collection；
- Exception handling；
- JavaScript BigInt to WebAssembly i64 integration；
- …

Wasm 虚拟机 - WASI

WASI

WebAssembly System Interface

传统接口抽象（musl）




```
6   FILE *fopen(const char *restrict filename, const char *restrict mode)
7   {
8           FILE *f;
9           int fd;
10          int flags;
11
12          /* Check for valid initial mode character */
13          if (!strchr("rwa", *mode)) {
14                  errno = EINVAL;
15                  return 0;
16          }
17
18          /* Compute the flags to pass to open() */
19          flags = __fmodeflags(mode);
20
21          fd = sys_open(filename, flags, 0666);
22          if (fd < 0) return 0;
23          if (flags & O_CLOEXEC)
24                  __syscall(SYS_fcntl, fd, F_SETFD, FD_CLOEXEC);
25
26          f = __fdopen(fd, mode);
27          if (f) return f;
28
29          __syscall(SYS_close, fd);
30          return 0;
31  }
```

System Calls

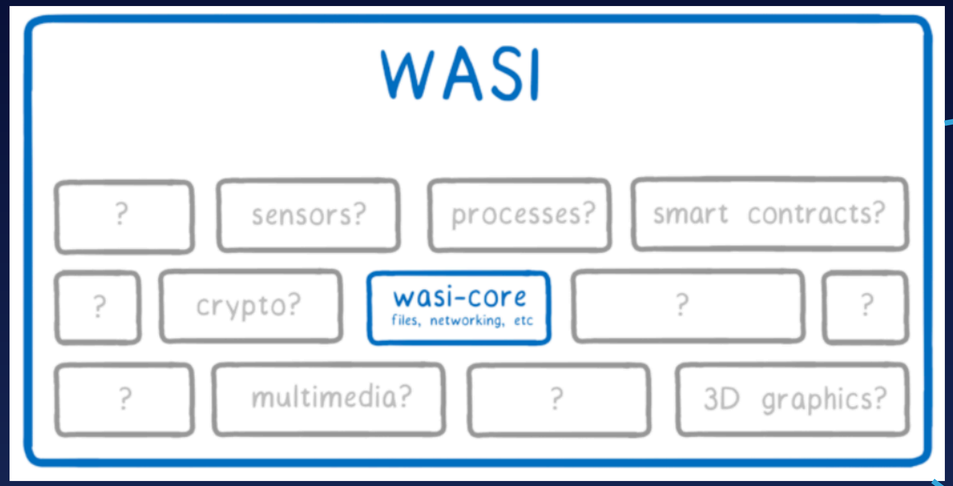
File / Socket / Memory / Pr

WASI - 接口抽象（wasi-libc）

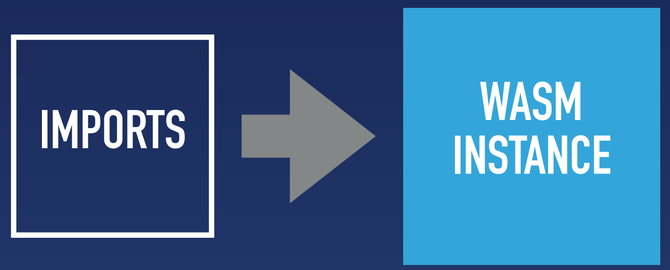
```
130  #else
131    __wasi_fdflags_t fs_flags = oflag & 0xfff;
132    __wasi_rights_t fs_rights_base = max & fsb_cur.fs_rights_inheriting;
133    __wasi_rights_t fs_rights_inheriting = fsb_cur.fs_rights_inheriting;
134    __wasi_fd_t newfd;
135    error = __wasi_path_open(fd, lookup_flags, path, strlen(path),
136                             (oflag >> 12) & 0xfff,
137                             fs_rights_base, fs_rights_inheriting, fs_flags,
138                             &newfd);
```

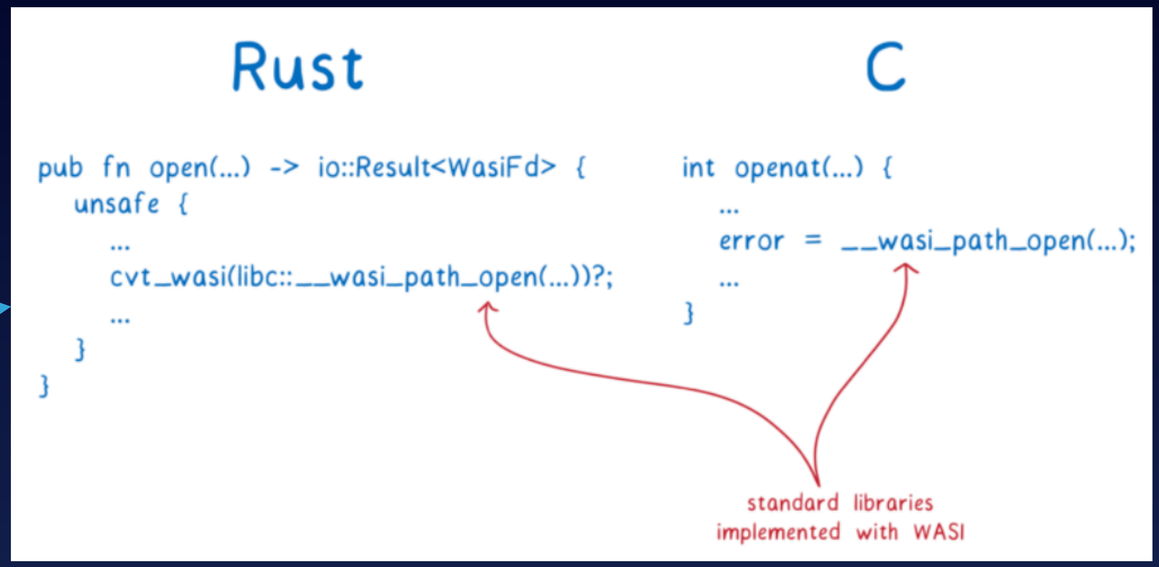
__wasi_* → WASI 标准

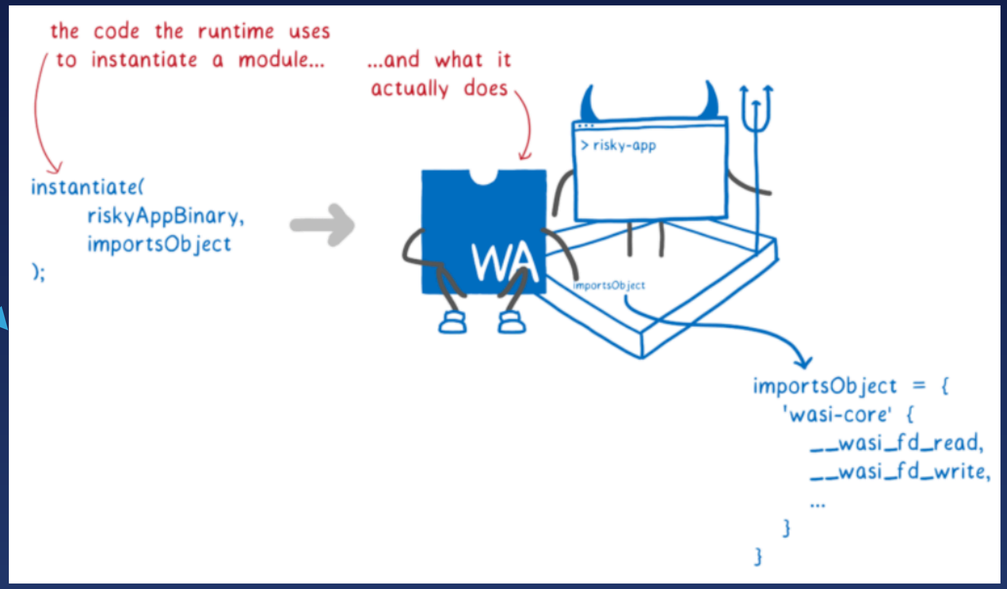
WASI - 接口调用关系



__wasi_path_open







WASI - 宿主实现细节

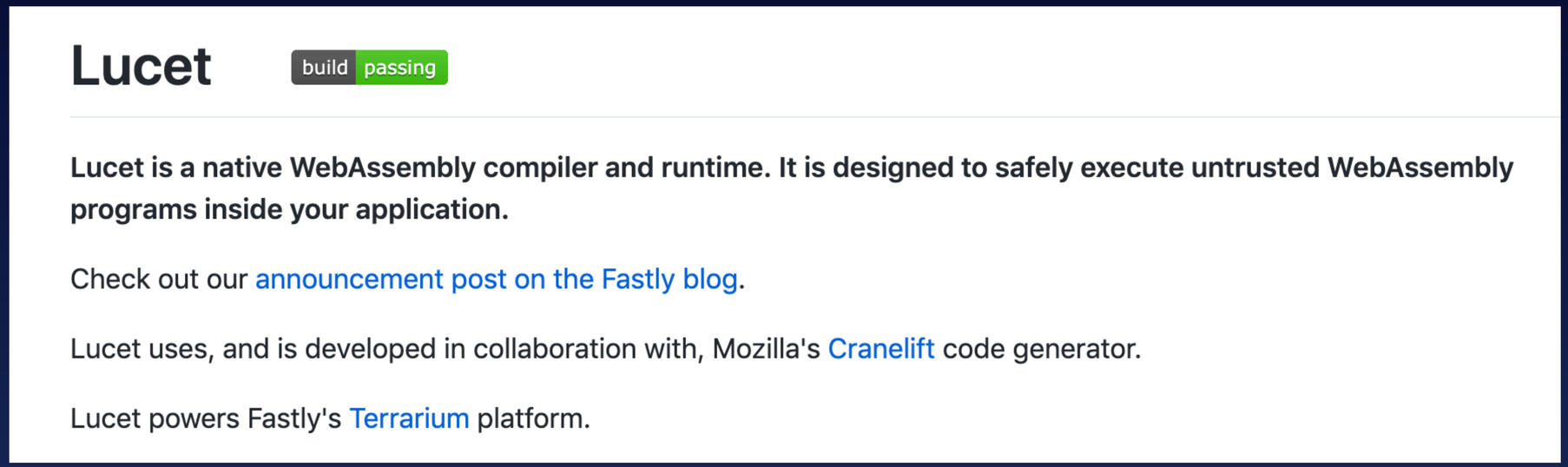
```c
__wasi_errno_t __wasi_path_open(
    __wasi_fd_t dirfd,
    __wasi_lookupflags_t dirflags,
    const char *path,
    size_t path_len,
    __wasi_oflags_t oflags,
    __wasi_rights_t fs_rights_base,
    __wasi_rights_t fs_rights_inheriting,
    __wasi_fdflags_t fs_flags,
    __wasi_fd_t *fd
) __WASI_SYSCALL_NAME(path_open)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

wasm-libc 函数定义

```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)
}
```

Lucet 宿主函数实现（Rust）

WASI - Lucet 宿主实现



一个 WebAssembly Compiler

&& (WASI)

一个 WebAssembly Runtime

WASI - Lucet 一个例子 - C/C++

宿主依赖的

文件操作

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

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

Geekbang 极客邦科技 | InfoQ 极客传媒

InfoQ官网 全新改版上线

促进软件开发领域知识与创新的传播

关注InfoQ网站
第一时间浏览原创IT新闻资讯

免费下载迷你书
阅读一线开发者的技术干货

THANKS

GMTC
全球大前端技术大会