Get started with Deno

@hijiangtao

示例一 Hello world

```
import { serve } from "https://deno.land/std@0.69.0/http/server.ts";

const s = serve({ port: 8000 });

console.log("http://localhost:8000/");

for await (const req of s) {

    req.respond({ body: "Hello World\n" });

}
```

示例二 Chat room

deno run --allow-net --allow-read https://deno.land/std/examples/chat/server.ts

目录

What is Deno

“Deno 是基于 V8 并采用 Rust 构建的，一个简单、现代且安全的 JavaScript 和 TypeScript 运行时环境。”

Deno is a simple, modern and secure runtime for JavaScript and TypeScript that uses V8 and is built in Rust. https://deno.land/

Deno / 基于 V8

V8 是一个由 Google 开发的开源 JavaScript 引擎，其在运行之前将JavaScript 编译成了机器代码，而非字节码或是解释执行它，以此提升性能。更进一步，使用了如Inline Cache 等方法来提高性能。有了这些功能，JavaScript 程序与 V8 引擎的速度媲美二进制编译。在 Deno 中，V8 引擎用于执行 JavaScript 代码。

Deno / 采用 Rust 构建

Rust 是由 Mozilla 主导开发的通用、编译型编程语言。设计准则为“安全、并发、实用”，支持函数式、并发式、过程式以及面向对象的编程风格。Deno 使用 Rust 语言来封装 V8 引擎，通过 libdeno 绑定，我们就可以在 JavaScript 中调用隔离的功能。

Deno / 简单

1. 天生支持 TypeScript - 使用 Deno 运行 TypeScript 代码无需任何手动编译（Deno 会自动为你执行此步骤）

2. 只有一个可执行文件

Deno / 现代

1. 天生支持 TypeScript
2. 只有一个单独的可执行文件
3. 自带实用工具，例如依赖检查器（deno info）和 代码格式化工具（deno fmt）
4. Top Level Await

Deno / 安全

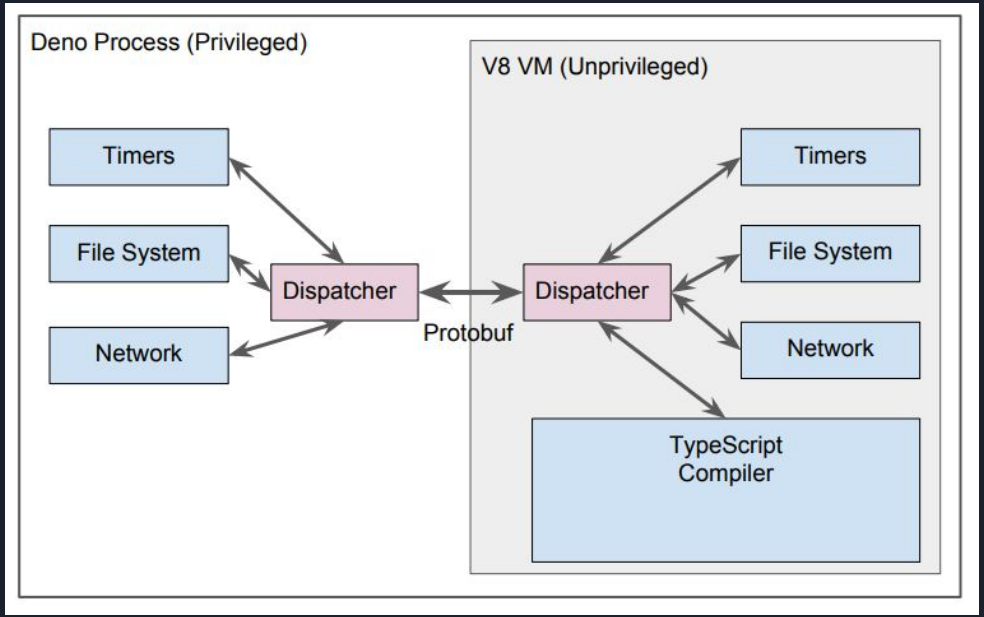
利用 JavaScript 是一个安全沙箱的事实

- 默认情况下，应该在没有任何网络或文件系统写权限的情况下运行脚本
- 用户可以选择通过标志访问：--allow-net --allow-write
- 这允许用户运行不受信任的实用程序

Deno / 安全

不允许将任意原生函数绑定到 V8

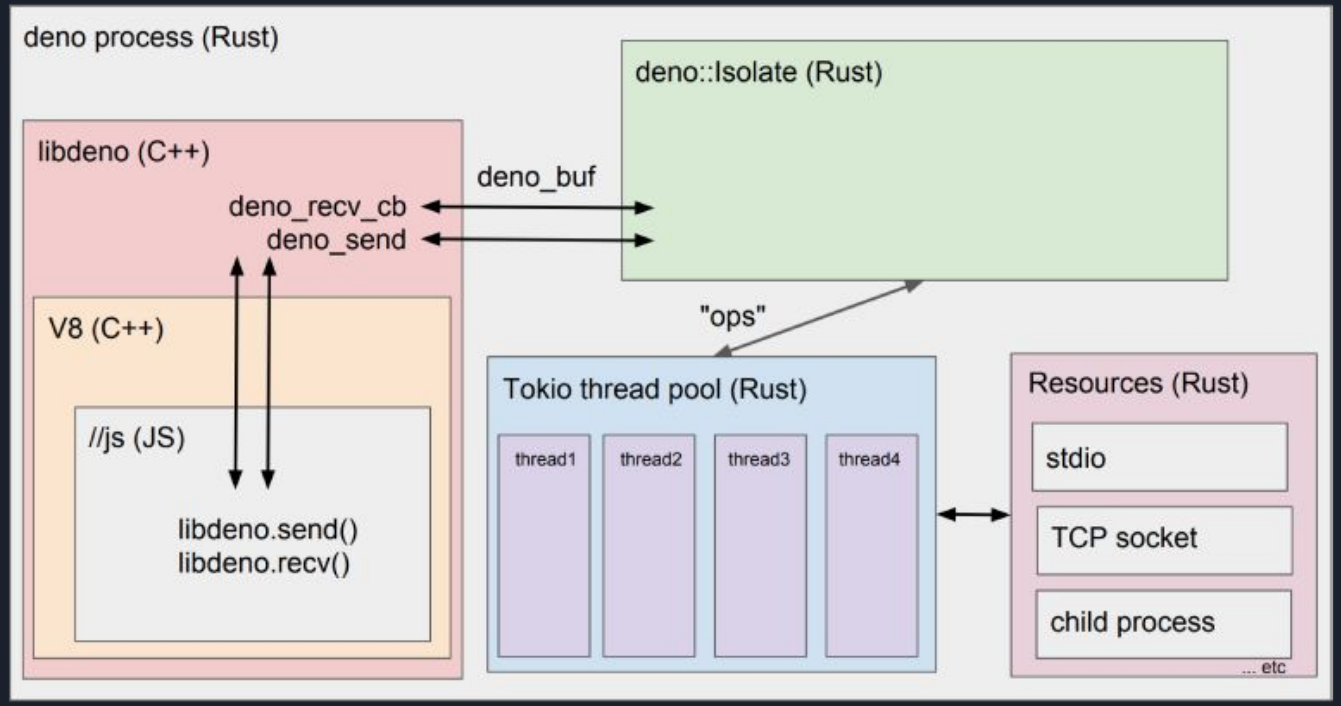
- 所有的系统调用都是通过消息传递完成的（protobuf 序列化）
- 用两个原生函数支持 :send 和 recv
- 简化设计



Deno / JavaScript 和 TypeScript 运行时环境

1. 只有一个单独的可执行文件
2. 有一套经过审核（审计）的标准模块，确保与 Deno 兼容：deno.land/std
3. 去中心化的 Package 管理
4. 内置测试
5. 浏览器兼容的API
6. 执行Wasm二进制文件
7. ES Modules

Deno 架构图



Deno 的异步运行时 / Tokio

Tokio 是 Rust 编程语言的异步运行时，提供异步事件驱动平台，构建快速，可靠和轻量级网络应用。利用 Rust 的所有权和并发模型确保线程安全。Tokio 构建于 Rust 之上，提供极快的性能，使其成为高性能服务器应用程序的理想选择。在 Deno 中 Tokio 用于并行执行所有的异步 IO 任务。

Deno and Node

Node 是什么？

V8 上一个 JavaScript 运行时 / 运行于服务端的 JavaScript

示例一 Hello world

```
const http = require('http');

http.createServer(function (request, response) {
    // 发送 HTTP 头部
    // HTTP 状态值: 200 : OK
    // 内容类型: text/plain
    response.writeHead(200, {'Content-Type': 'text/plain'});
    // 发送响应数据 "Hello World"
    response.end('Hello World\n');
}).listen(8888);

// 终端打印如下信息
console.log('Server running at http://127.0.0.1:8888/');
```

Node 历史速览

2009年3月，Ryan Dahl 在博客宣布准备基于 V8 创建一个轻量级 Web 服务器并提供一套库

2009年5月，Ryan Dahl 在 GitHub 上发布了最初版本

2010年底，Ryan Dahl 加入 Joyent 公司全职负责 Node 的发展

2011年7月，Node 在微软支持下发布了 Windows 版本

2011年11月，Node 超越 Ruby on Rails，成为 GitHub 上关注度最高的项目

2012年1月底，Ryan Dahl 在对 Node 架构设计满意的情况下，将掌门人身份转交给 Isaac Z Schlueter（NPM 作者），自己转向一些项目研究

2018年初，Ryan Dahl 开始重新使用Node

2018年6月，JSConf EU 上 Ryan Dahl 表示 Node 存在太多设计错误了

JSConf EU 2018

Design Mistakes in Node

没有坚持 Promise

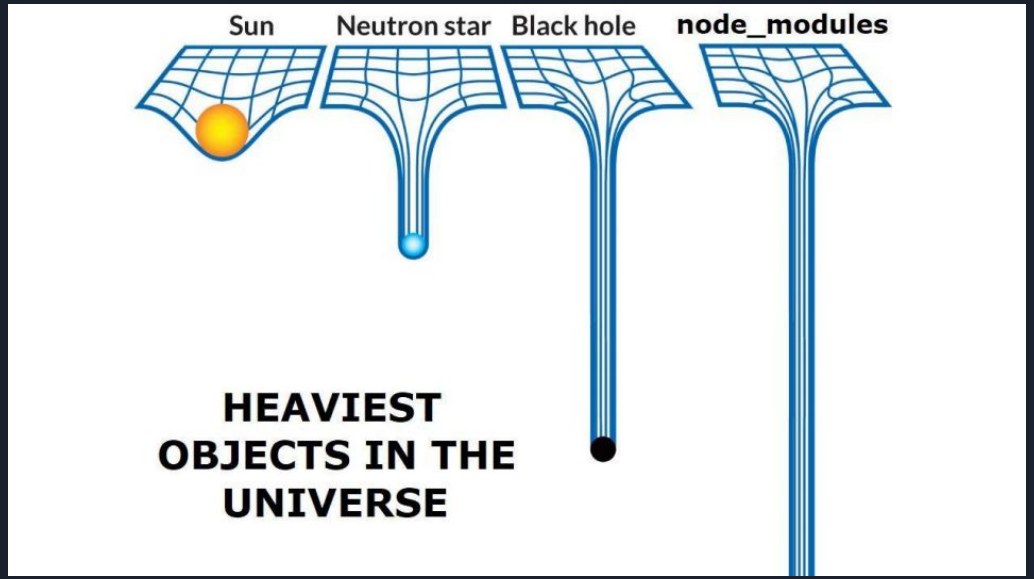
1. 2009.6 Node 中添加了 Promise
2. 2010.2 Promise 被删除
3. Promise 是 async/await 的必要抽象
4. 标准化与 async/await 受其影响

Design Mistakes in Node

require("module") 不符合 Web 标准且不明确

Design Mistakes in Node

无底洞的node_modules



Design Mistakes in Node

没有利用好V8 沙箱的优势

GYP 指导用户编写 C++ 来绑定到 V8

冗余的 package.json

冗余的 index.js

……

Deno vs Node

- 两者都是基于V8 引擎开发的;
- 两者都非常适合在服务器端上编写 JavaScript 应用。

Deno vs Node

- Node.js 用 C++ 和 JavaScript 编写。Deno 用 Rust 和 TypeScript 编写。
- Node.js 有软件包管理器，NPM。Deno 不会有，而会允许你从 URL 导入任何 ES 模块。
- Node.js 使用 CommonJS 模块语法导入软件包。Deno 使用 ES 标准模块导入。
- Deno 在其所有 API 和标准库中都使用现代 ECMAScript 功能，而 Node.js 使用基于回调的标准库，并且没有计划对其进行升级。

Deno vs Node

- Deno 通过权限控制提供了一个安全的沙箱环境，程序只能访问由用户设置为可执行标志的文件。Node.js 程序可以直接访问用户足以访问的任何内容。

- Deno 长期以来一直在探索将程序编译成单个可执行文件的可能性，从而使得该可执行文件可以在没有外部依赖项（例如 Go）的情况下运行，但这并不是一件容易的事，如果做得到，将会成为更有话语权的游戏规则改变者。

Debate on Deno

“在努力做好（Deno）这件事和推出一款可用的产品之间很难取得平衡。因为还没有在开发工具支持特性上取得实质进展，官方团队再次推迟发布日期。官方团队进行了详细的讨论，得出结论是 2 个月的时间足够了。巧合的是，这是自 Deno 问世以来的两周年纪念日。因此，我们将 2020 年 5 月 13 日定为 1.0 版本发布日期。我们鼓励贡献者在 4 月 20 日之前对 API 进行重大修改——在那之后，我们将对 API 进行完善和 bug 修复。当然，在 1.0 之后，API 将继续发展和改进，但是我们将为一些接口提供显式的稳定性保证。”

2020年5月13日，Deno 1.0 如期而至 https://deno.land/v1

Since v1.0

V1.4 2020.09.13

V1.3 2020.08.13

V1.2 2020.07.13

V1.1 2020.06.13

作为普通开发者，有什么我们需要了解的？

Shell (Mac, Linux)

```
$ curl -fsSL https://deno.land/x/install/install.sh | sh
```

PowerShell (Windows)

```
$ iwr https://deno.land/x/install/install.ps1 -useb | iex
```

Homebrew (Mac)

```
$ brew install deno
```

安全的 Deno

Deno 是默认安全的，这体现在默认没有环境、网络访问权限、文件读写权限、运行子进程的能力。所以如果直接运行一个依赖权限的文件会报错：

```
deno run file-needing-to-run-a-subprocess.ts
# error: Uncaught PermissionDenied: access to run a subprocess,
run again with the --allow-run flag
```

可以通过参数方式允许权限的执行，有 --allow-read、--allow-write、--allow-net 等：

```
deno --allow-read=/etc
```

Deno 标准库

Deno Module	Description	npm Equivalents
colors	Adds color to the terminal	chalk, kleur, and colors
datetime	Helps working with the JavaScript `Date` object	
encoding	Adds support for external data scructures like base32, binary, csv, toml and yaml	
flags	Helps working with command line arguments	minimist
fs	Helps with manipulation of the file system	
http	Allows serving local files over HTTP	http-server
log	Used for creating logs	winston
testing	For unit testing assertion and benchmarking	chai
uuid	UUID generation	uuid
ws	Helps with creating WebSocket client/server	ws

包管理与三方库

Deno 用 deps.ts 来集中管理依赖，并且会自动处理依赖缓存，添加 --load 便可强制刷新缓存

```
// deps.ts
export { assert } from
"https://deno.land/std@v0.39.0/testing/asserts.ts";
export { green, bold } from
"https://deno.land/std@v0.39.0/fmt/colors.ts";

// 使用
// import { assert } from
"https://deno.land/std@v0.39.0/testing/asserts.ts";
import { assert } from "./deps.ts";

// 锁定依赖
deno --lock=lock.json
```

ESModule

Deno 使用官方 ESModule 规范，但引用路径必须加上后缀

Deno 不需要声明依赖，代码引用路径便是依赖声明

但是

Deno 会是另一个「包袱重重」的 Node 么？

去中心化管理

无底洞的node_modules 被替换为 url 引入，岂不是以后可以更轻而易举的跑路了？

缺失的版本管理

标准库与浏览器兼容

对最新语言规范以及 TypeScript 的支持

原生支持 TypeScript

TypeScript 编译成 JavaScript 再继续执行

如何评价 deno 把一些内部模块从 ts 改回 js ？

编译速度、高性能代码、类型约束不一致

比如 lib.*.d.ts 和 deno api 实现所导出的 d.ts 有一定 gap，进而影响 web api 的开发范式，进而为了与 whatwg 对齐而引入的多余代码（例如原文中的define name），进而影响性能

https://docs.google.com/document/d/1_WvwHl7BXUPmoiSeD8G83JmS8ypsTPqed4Btkqkn_-4/preview#

关于 Deno 的更多

Deno Land https://deno.land/

更多的更多

Deno Manual https://deno.land/manual

Deno GitHub https://github.com/denoland/deno

Deno doc https://doc.deno.land/https/github.com/denoland/deno/releases/latest/download/lib.deno.d.ts

Deno, a new way to JavaScript. Ryan Dahl. JS Fest 2019 Spring

https://www.youtube.com/watch?v=z6JRlx5NC9E

10 Things I Regret About Node.js - Ryan Dahl - JSConf EU

https://www.youtube.com/watch?v=M3BM9TB-8yA

Design Mistakes in Node https://tinyclouds.org/jsconf2018.pdf

Rust language https://www.rust-lang.org/

V8 JavaScript Engine https://v8.dev/

V8 GitHub https://github.com/v8/v8

Tokio GitHub https://github.com/tokio-rs/tokio

Deno 1.0: What you need to know https://blog.logrocket.com/deno-1-0-what-you-need-to-know/

Deno 运行时入门教程：Node.js 的替代品 https://www.ruanyifeng.com/blog/2020/01/deno-intro.html

Deno 入门手册 https://zhuanlan.zhihu.com/p/142002526

和更多的更多的更多

《Design Mistakes in Node》Node 之父 Ryan Dahl 演讲 PPT 中文版 (2018 JS Conf Berlin)
https://zhuanlan.zhihu.com/p/37637923

JavaScript Engines: The Good Parts™ - Mathias Bynens & Benedikt Meurer - JSConf EU 2018
https://www.youtube.com/embed/5nmpokoRaZI

Deno 中文社区 https://deno.js.cn/

JavaScript 引擎基础：原型优化 https://zhuanlan.zhihu.com/p/42630183

JavaScript 引擎基础：Shapes 和 Inline Caches https://zhuanlan.zhihu.com/p/38202123

从 CLI 指令通读 Deno v1.x 全特性 https://juejin.im/post/6857058738046861320

Deno Architecture https://deno.land/manual/contributing/architecture

Ryan Dahl - An interesting case with Deno https://www.youtube.com/watch?v=1b7FoBwxc7E

Deno 1.0: What you need to know https://blog.logrocket.com/deno-1-0-what-you-need-to-know/

Getting Started With Deno 1.0 https://www.htmlhints.com/article/getting-started-with-deno-10/91

了不起的 Deno 入门与实战 https://zhuanlan.zhihu.com/p/141832695

精读《Deno 1.0 你需要了解的》https://github.com/dt-fe/weekly/issues/248

What’s Deno?

A secure runtime for JavaScript and TypeScript.

Q&A

Thanks!

@hijiangtao