From

kernel to runtime

peek inside a javascript runtime

IIT Kanpur (OOSC)

5 Sept 2025

What's Deno?

Deno is an open source JavaScript runtime built on top of V8.

- supports modern web standard APIs
- builtin TypeScript support
- sandbox permission system
- Node.js/npm compatibility

Single thread I/O

Event loop is driven using epoll/IOCP when a file is ready, the kernel notifies epoll_wait()

JavaScript runtimes put I/O operations on the event loop and use Promises/callbacks to notify the user code

One line HTTP server

```
// $ deno --allow-net server.js
Deno.serve(req => new Response("Hello, World!"))
```

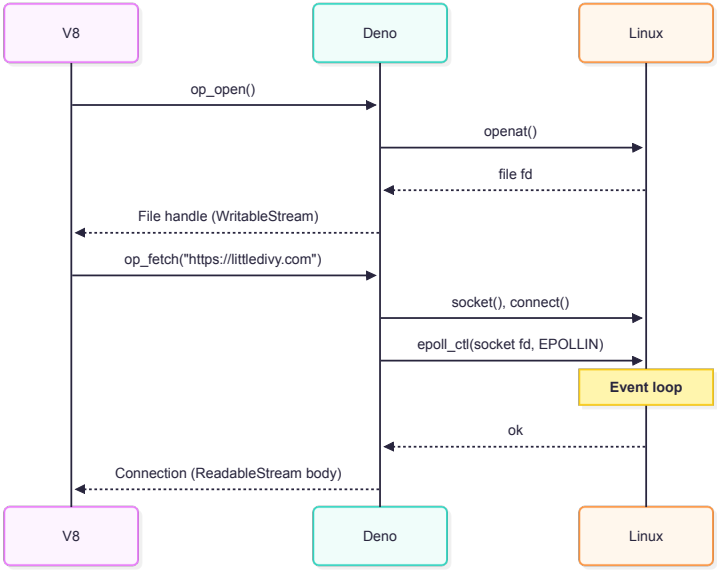
- single threaded
- can handle 130k+ rps[1]
- 1ms p99 latency[1]
- in JavaScript

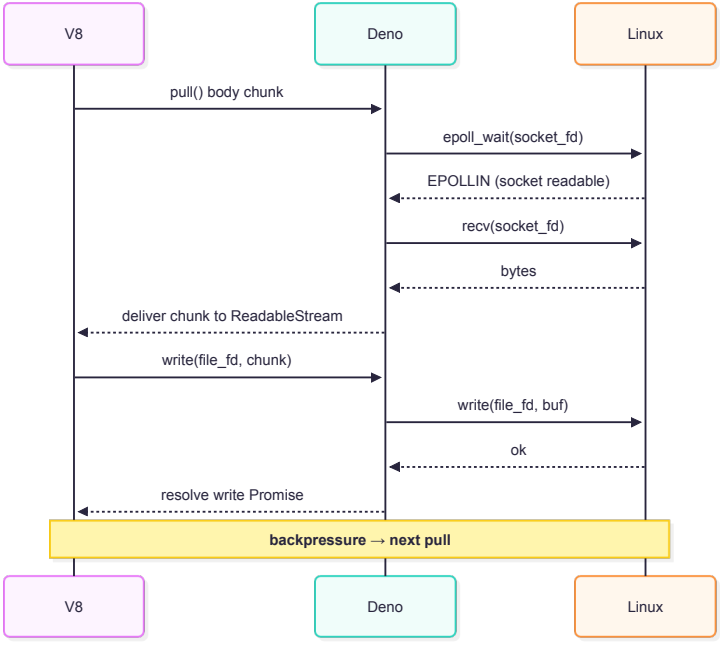
[1] https://www.trevorlasn.com/blog/benchmarks-for-node-bun-deno

Scheduling

I/O operations are scheduled on the event loop, they *may* be offloaded to a thread pool but the user code is not blocked.

```
const file = await Deno.open("index.html");
const req = await fetch("https://littledivy.com");

req.body.pipeTo(file.writable);
```


V8
Deno
Linux
op_open()
openat()
file fd
File handle (WritableStream)
op_fetch("https://littledivy.com")
socket(), connect()
epoll_ctl(socket fd, EPOLLIN)
Event loop
ok
Connection (ReadableStream body)
V8
Deno
Linux


V8
Deno
Linux
pull() body chunk
epoll_wait(socket_fd)
EPOLLIN (socket readable)
recv(socket_fd)
bytes
deliver chunk to ReadableStream
write(file_fd, chunk)
write(file_fd, buf)
ok
resolve write Promise
backpressure → next pull
V8
Deno
Linux

Permission system

Virtual permission system that restricts access to OS resources.

```
deno run server.ts # blocked
deno run --allow-net server.ts # OK
```

Memory management

JavaScript objects are garbage collected.

How does it cleanup native files, sockets and other resources?

Resources

Resources are like fds: integer handles for open files, sockets, etc.

```
console.log(Deno.resources());
// { 0: "stdin", 1: "stdout", 2: "stderr" }
Deno.close(0);
```

This allows users to manually close native resources.

Garbage collectable resources

GC'able resources are attach to a JavaScript object. The native resource is freed when the object is collected.

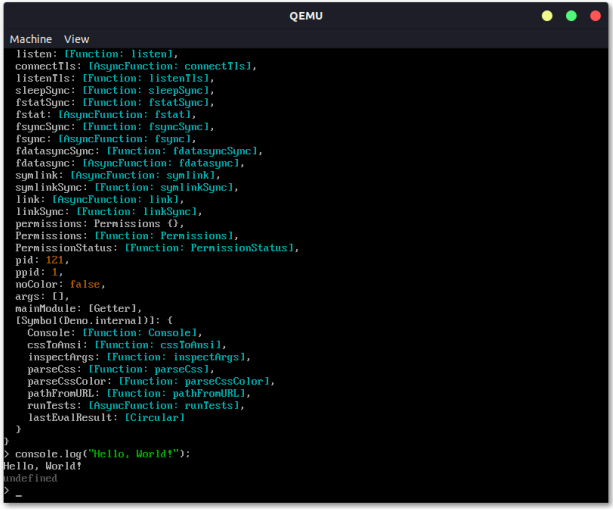
```
import { DatabaseSync } from "node:sqlite";

const db = new DatabaseSync();
// ...
```

Bonus: Deno OS

minimal Linux kernel build with a Deno userspace.

Linux	**Deno**
Processes	Web Workers
File descriptors (fd)	Resource ids (rid)
Syscalls	Ops
Scheduler	Tokio
man pages	`deno types`

https://github.com/littledivy/deno-os


QEMU
Machine View
listen: [Function: listen],
connectTls: [AsyncFunction: connectTls],
listenTls: [Function: listenTls],
sleepSync: [Function: sleepSync],
fstatSync: [Function: fstatSync],
fstat: [AsyncFunction: fstat],
fsyncSync: [Function: fsyncSync],
fsync: [AsyncFunction: fsync],
fdatasyncSync: [Function: fdatasyncSync],
fdatasync: [AsyncFunction: fdatasync],
symlink: [AsyncFunction: symlink],
symlinkSync: [Function: symlinkSync],
link: [AsyncFunction: link],
linkSync: [Function: linkSync],
permissions: Permissions {},
Permissions: [Function: Permissions],
PermissionStatus: [Function: PermissionStatus],
pid: 121,
ppid: 1,
noColor: false,
args: [],
mainModule: [Getter],
[Symbol(Deno.internal)]: {
Console: [Function: Console],
cssToAnsi: [Function: cssToAnsi],
inspectArgs: [Function: inspectArgs],
parseCss: [Function: parseCss],
parseCssColor: [Function: parseCssColor],
pathFromURL: [Function: pathFromURL],
runTests: [AsyncFunction: runTests],
lastEvalResult: [Circular]
}
}
> console.log("Hello, World!");
Hello, World!
undefined
> _

Get involved

Github: https://github.com/denoland/deno

Discord: https://discord.gg/deno

open issues, ideas or contribute code

Thanks!

Questions?

Source:

me@littledivy.com

discord.gg/deno