

from

Kernel to runtime

Peek inside a JavaScript runtime

IIT Kanpur

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

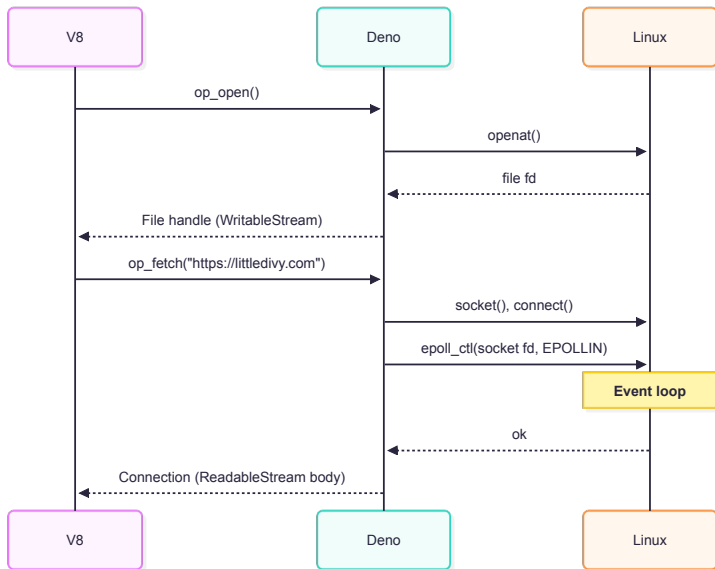
[1] <https://www.trevorlasn.com/blog/benchmarks-for-node-bun-deno>

Scheduling

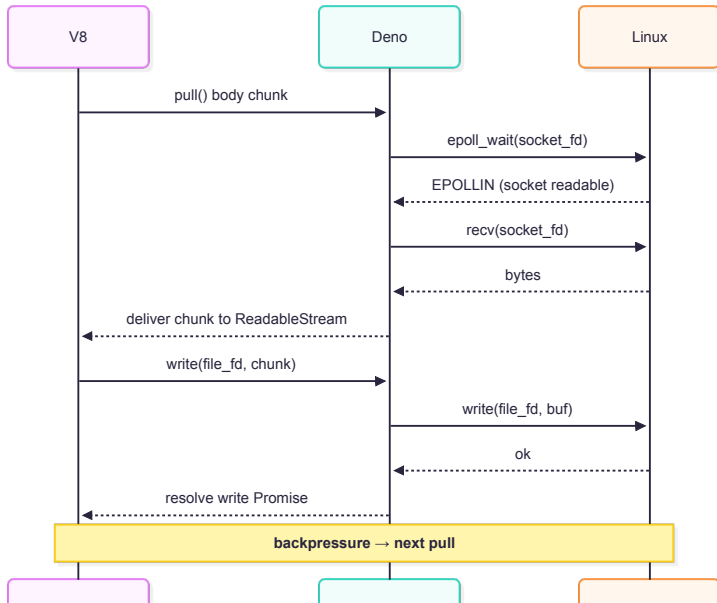
I/O operations are scheduled on the event loop, they *may* be offloaded to a thread pool but the user code is not blocked.

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

Sequential



Concurrent (pipeTo)



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();  
// ...
```

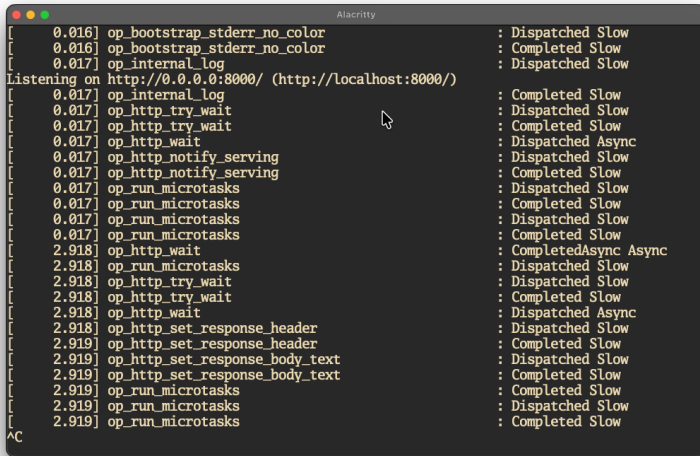
```
struct DatabaseSync { ptr: *mut sqlite3 }
```

```
impl Drop for DatabaseSync {  
    fn drop(&mut self) {  
        unsafe { sqlite3_close(self.ptr) };  
    }  
}
```

```
// ...
```

```
let value: v8::Local<v8::Object> =  
    wrap_cppgc_object(scope, Box::new(DatabaseSync { ptr })))
```

Tracing



```
Alacritty
[ 0.016] op_bootstrap_stderr_no_color : Dispatched Slow
[ 0.016] op_bootstrap_stderr_no_color : Completed Slow
[ 0.017] op_internal_log : Dispatched Slow
Listening on http://0.0.0.0:8000/ (http://localhost:8000/)
[ 0.017] op_internal_log : Completed Slow
[ 0.017] op_http_try_wait : Dispatched Slow
[ 0.017] op_http_try_wait : Completed Slow
[ 0.017] op_http_wait : Dispatched Async
[ 0.017] op_http_notify_serving : Dispatched Slow
[ 0.017] op_http_notify_serving : Completed Slow
[ 0.017] op_run_microtasks : Dispatched Slow
[ 0.017] op_run_microtasks : Completed Slow
[ 0.017] op_run_microtasks : Dispatched Slow
[ 0.017] op_run_microtasks : Completed Slow
[ 2.918] op_http_wait : Completed Async Async
[ 2.918] op_run_microtasks : Dispatched Slow
[ 2.918] op_http_try_wait : Dispatched Slow
[ 2.918] op_http_try_wait : Completed Slow
[ 2.918] op_http_wait : Dispatched Async
[ 2.918] op_http_set_response_header : Dispatched Slow
[ 2.919] op_http_set_response_header : Completed Slow
[ 2.919] op_http_set_response_body_text : Dispatched Slow
[ 2.919] op_http_set_response_body_text : Completed Slow
[ 2.919] op_run_microtasks : Completed Slow
[ 2.919] op_run_microtasks : Dispatched Slow
[ 2.919] op_run_microtasks : Completed Slow
^C
```

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
strace	- -trace-ops

<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],
fdasyncSync: [Function: fdasyncSync],
fdasync: [AsyncFunction: fdasync],
symlink: [AsyncFunction: symlink],
symlinkSync: [Function: symlinkSync],
link: [AsyncFunction: link],
linkSync: [Function: linkSync],
permissions: Permissions O,
Permissions: [Function: Permissions],
PermissionStatus: [Function: PermissionStatus],
pid: 121,
ppid: 1,
noColor: false,
args: [],
mainModule: [Getter],
[Symbol(Deno.internal)]: {
  Console: [Function: Console],
  cssToAmsi: [Function: cssToAmsi],
  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?

me@littledivy.com

[linkedin.com/in/littledivy](https://www.linkedin.com/in/littledivy)