# deno

Ryan Dahl

Brooklyn JS

January 2019

https://tinyclouds.org/deno_bkjs.pdf

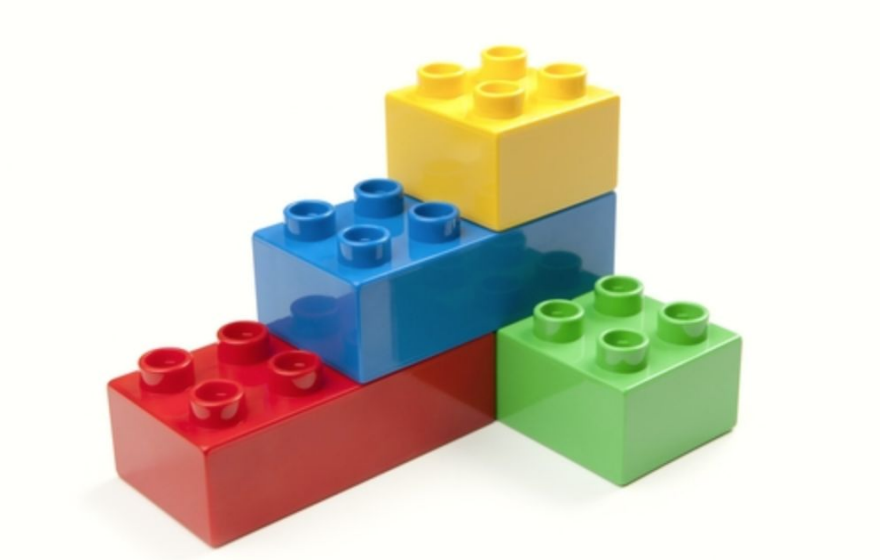
# Disclaimer

This talk is aimed at **experienced enthusiasts**

If this isn't you: **don't panic**

Ignore this talk - use Node. Node isn't going anywhere!

I want a **fun** and **productive** system for scripting.



# Deno is for the next decade of JavaScript

- deno is distributed as a **single executable file** (and always will be).
- **No additional software** is needed to run any deno program.
- deno uses only "**ES modules**" not require().
- deno can execute **TypeScript** out of the box.
- deno makes **security guarantees**, so you can run code fearlessly.
- deno is **embeddable** as a Rust crate
- deno is **browser compatible**

# Putting the link back in linking

All imports are either relative or via URLs.

```
import { red } from "https://deno.land/x/std@v0.2.6/colors/mod.ts";
console.log(red("Hello world!"));
```

We consider it of the utmost importance to keep the module resolution algorithm dumb.

- It works exactly as URLs do in browser JS.
- Filename extensions are required.
- `index.js` no longer has any special meaning.
- There are no "bare imports" like `import * as react from "react"`

# Security CLI flags

`--allow-net` — access the network

`--allow-write` — write to disk

`--allow-env` — read from env vars

`--allow-run` — run subprocesses

Or `-A` to allow all.

(`--allow-read` is still under construction)

# Deno is (kind of) like an OS

| Linux | Deno |
| --- | --- |
| Processes | Web Workers |
| Syscalls | Ops |
| File descriptors (fd) | Resource ids (rid) |
| Scheduler | Tokio |
| Userland: libc++ / glib / boost | deno_std |
| `/proc/$$/stat` | `deno.metrics()` |
| man pages | `deno --types` |

# Resources are like file descriptors

```
> deno.resources()

{ 0: "stdin", 1: "stdout", 2: "stderr", 3: "repl" }

> deno.close(3)
```

# Userspace is deno_std (Standard Modules)

Deno core is kept as minimal as possible.

However it's useful to have high quality standard modules.

https://github.com/denoland/deno_std

- All code is reviewed by me (at least superficially)
- Doesn't have external deps
- Is tagged with each Deno release: **https://deno.land/x/std@v0.2.6/**

# Metrics tell you about the runtime

> deno.metrics()

{ opsDispatched: 3, opsCompleted: 3, bytesSentControl: 180, bytesSentData: 0, bytesReceived: 276 }

> deno.metrics()

{ opsDispatched: 5, opsCompleted: 5, bytesSentControl: 288, bytesSentData: 0, bytesReceived: 436 }

Internal Design