Introduction to Deno

A secure runtime for JavaScript and TypeScript

What is it?

From the website: Deno is a simple, modern and secure runtime for JavaScript and TypeScript that uses V8 Engine (same as Node) and is built in Rust.

1. Secure by default. No file, network, or environment access, unless explicitly enabled.
2. **Supports TypeScript out of the box.**
3. Ships only a single executable file.
4. Has **built-in utilities** like a dependency inspector (called “deno info”) and a code formatter (called “deno fmt”).
5. Has a set of reviewed (audited) **standard modules** that are guaranteed to work with Deno: deno.land/std

Why do we care?

1. Sounds like Node.js? Same person (Ryan Dahl) who created Node.js created Deno!
2. Both Javascript and Typescript can be used outside of the browser. No need to transpile, Deno works with Typescript.
3. Deno has a standard library.
4. Deno can use WebAssembly.
5. No package.jsons needed! (most controversial change but more on that later)
6. Enforces styling/auto-linting of Typescript at runtime!
7. Just made it to V1.0+ this year so it's (mostly) ready for production.

Getting Started - Install and Example

```
shift3@everyday:~$ curl -fsSL https://deno.land/x/install/install.sh | sh
```

```
shift3@everyday:~$ deno run https://deno.land/std/examples/welcome.ts
Download https://deno.land/std/examples/welcome.ts
Warning Implicitly using master branch https://deno.land/std/examples/welcome.ts
Check https://deno.land/std/examples/welcome.ts
Welcome to Deno 🦕
```

```
shift3@everyday:~$ deno upgrade
Checking for latest version
downloading https://github.com/den
4-unknown-linux-gnu.zip
Version has been found
Deno is upgrading to version 1.2.2
```

Getting Started - Part 2 (Simple web server)

```ts
import { serve } from "https://deno.land/std@0.61.0/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" });
}
~
```

```
shift3@everyday:~/Documents/Github/deno-developer-talk/examples/intro$ deno run --allow-net intro_world_server.ts
http://localhost:8000/
```

Hello World

Getting Started - Part 3 (More complicated)

Other example programs (all based on https://deno.land/manual):

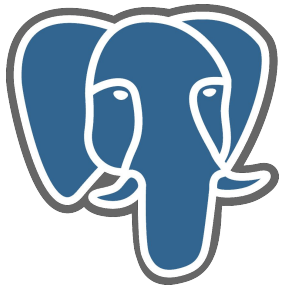
- Making an HTTP request
- Reading Files
- TCP server
- An implementation of the unix "cat" program
- File server
- TCP echo server
- Run subprocess
- Inspecting and revoking permissions
- Handle OS Signals
- File system events

Example: REST API

How to build a RESTFUL API with deno?

- Deno (duh)
- OAK: A middleware framework for Deno's net server
- PostgreSQL: The World's Most Advanced Open Source Relational Database





WebAssembly - A quick look

- Deno/Node.js/Browsers execute JavaScript on C/C++ based runtimes.
- Allows developers to use compiled languages and libraries on the web (and everything else)!
- Most modern day browsers already support WebAssembly as a standard.

Example:
LISP interpreter in the browser (Javascript bindings into C -> LISP)!
https://github.com/michaelachrisco/ToyLisp

Warning: This is a talk in itself!

Deno + WebAssembly

- Deno supports WebAssembly out of the box.
- C/C++/Rust/Go and many more work with Deno bindings.
- Most modern day browsers already support WebAssembly as a standard.

Practical Example:
Creating web-based GUIs for desktop applications with Rust and Deno.
Heavily based on work here:
https://github.com/webview/webview_deno

Warning: This is a talk in itself!

Conclusions: Pros of Deno

Deno has some great ideas:

- Supports both Javascript and Typescript out of the box.
- Has easy to use bindings for WebAssembly.
- Works very well on server-side Typescript.
- Core libraries seem to be the most stable part of the system.
- I found year old projects still working on the Deno core API.

Conclusions: Not-so Pros of Deno

3rd party Deno Libraries need work:

- When doing research for this presentation, there were an abundant amount of libraries doing the same things (REST API, ORMS, etc…) with varying degrees of longevity.
- Issues with installing 3rd parties that do not support the most recent version of Deno.

Conclusions: Opinions

- You will either love or hate the import process in Deno. It’s very similar to how Go deals with importing its libraries.
- When dealing with import errors, found myself forking and adding fixes to own github repositories.

Youtube Resources

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

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

- Deno in 100 seconds: https://www.youtube.com/watch?v=F0G9lZ7gecE
- (6 hours) Deno Course - Better than Node.js? - https://www.youtube.com/watch?v=TQUy8ENesGY

References and Resources

- Logo from https://github.com/denolib/high-res-deno-logo/blob/master/deno_hr_circle.svg
- Main website: https://deno.land/
- Documentation and example programs: https://deno.land/manual
- FreeCodeCamp more in depth 6 hour talk: https://www.youtube.com/watch?v=TQUy8ENesGY
- Deno crash course with Rest API: https://www.youtube.com/watch?v=NHHhiqwcfRM&t=5s
- Oak REST API: https://github.com/oakserver/oak
- Other Oak REST API: https://medium.com/javascript-in-plain-english/building-crud-apis-using-deno-and-oak-9f71ec106b0e

Questions?

CC Dino pics!

https://search.creativecommons.org/search?q=dinosaur