Deno

A secure runtime for JavaScript and TypeScript

Juho Vepsäläinen

ViennaJS online 27.01.2021

@survivejs

Schedule

`18:00–18:15` - Greetings, casual chat

`18:15–19:00` - Introduction to Deno. We'll code through simple samples.

`19:00–20:00` - Split into groups to code smaller projects with Deno?

`20:00–20:30` - Check projects and resolve potential issues as well as we can

`20:30–21:00` - Casual chat and wrap up

Brief evolution of server-side JavaScript

- SpiderMonkey (Brendan Eich, Mozilla)
- Rhino (Java)
- Node.js (Ryan Dahl) – V8 (Google)
- Deno (Ryan Dahl)

Installation

Hello world

```
deno run https://deno.land/std/examples/welcome.ts
```

But why?

- When Node.js was designed in 2009, mistakes were made and new concepts that are now standard had to be invented
- As Node.js has a large userbase, it’s difficult and slow to evolve the system
- Deno addresses these issues and gives a fresh start
- Deno is not a fork – it’s a new effort

Minimal server

```
import { serve } from "https://deno.land/std@0.84.0/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" });
}
```

1. Run the server using `deno run`
2. Run the server with the right permissions
3. Check the browser (localhost:8000)

Pros

- Supports TypeScript out of the box
- Comes with a strong stdlib that gets better day by day
- Integrates well with upcoming technologies such as Rust and WebAssembly
- Embraces standard web standards (workers and more)
- Forward looking, fast evolving

Cons

- Poor support for npm packages
- Controversial packaging model (no npm, distributed instead)
- Third-party and platform support is lacking (Netlify etc.)
- Slightly slower (but more consistent) than Node
- tsc is slow (needs to be rewritten in in Rust)

Philosophy

- Secure by default (explicit permissions)
- No npm, no `node_modules`. Load resources through urls like browsers
- Code works both in frontend and backend where possible
- ES Modules by default instead of CommonJS
- Unlike Node, includes formatter, linter, bundler, testing. Think Deno as a toolkit.

Secure by default

```
deno run --allow-read mod.ts
```

Code works in frontend and backend where possible

```
const response = await fetch("https://api.github.com/orgs/denoland");

console.log(response);
```

1. Set up a file with code you would expect to work in frontend
2. Run it through Deno
3. See what happens with objects like `window` or `location`

ES Modules instead of CommonJS

```
import {
  add,
  multiply,
} from "https://x.nest.land/ramda@0.27.0/source/index.js";
import { subtract } from "./local-arithmetic.ts";

export { add, multiply, subtract };
```

1. Set up a couple of modules to test the system
2. Run it through Deno
3. See what happens when you break modules. Test `import()`.

deno.land

Deno is a toolkit

- `deno install` – Install and distribute executable code
- `deno fmt` – Format Deno code (JS and TS)
- `deno repl` – REPL in Deno
- `deno bundle` – Bundle input as a single executable file
- `deno compile --unstable` – Compile the script as a self-contained executable
- `deno doc` – Generate documentation based on JSDoc
- `deno info` – Inspect the dependency graph
- `deno lint --unstable` – Lint Deno code (JS and TS)
- Includes a unit testing approach

Install and distribute code

```
deno install
```

Format Deno code (JS and TS)

```
deno fmt
```

REPL in Deno

```
deno repl
```

Bundle input as a single executable file

```
deno bundle
```

Compile the script as a self-contained executable

```
deno compile --unstable
```

Generate documentation based on JSDoc

```
deno doc
```

Inspect the dependency graph

```
deno info
```

Lint Deno code (JS and TS)

```
deno lint --unstable
```

Unit testing approach

```
import { assert } from "https://deno.land/std@0.84.0/testing/asserts.ts";

Deno.test("Hello Test", () ⇒ {
  assert("Hello");
});
```

1. Set up a file with the test
2. Run it through Deno
3. Break the test and run again

Project topics


ssg

Aleph.js

starts-with/deno

gustwind

Bring your own topic

Plan

1. Choose an interesting topic or bring your own
2. It’s ok to work with friends if you want
3. We’ll use the main channel for supporting you and answering any questions you might have. Feel free to mute us while working!
4. Let’s check out the work we did around 20:00. If you want show your project, this is the time.

Kiitos