Denoの紹介

Press Space for next page →

自己紹介

- 📝 飯野陽平（wheatandcat）
- 🏢 フリーランスエンジニア（シェアフル株式会社CTO）
- 💻 Blog: https://www.wheatandcat.me/
- 📚 Booth: https://wheatandcat.booth.pm/
- 🛠️ 今までに作ったもの
  - memoir
  - ペペロミア
  - Atomic Design Check List

Denoとは？

- V8 JavaScriptエンジンをベースに実装されたJavaScript/TypeScriptランタイム
- **DenoはNode.jsの反省から生まれた** を修正すべく、Node.jsの開発者を中心に開発が行われている

Denoの特徴

- 主な特徴は以下が挙げられる
  - TypeScriptを標準でサポート
  - パーミッションシステム
  - 生産性
  - Web標準への準拠
  - ES Modulesベースのモジュールシステム
  - 最新のJavaScript仕様に準拠

Denoの特徴

- 主な特徴は以下が挙げられる
  - TypeScriptを標準でサポート
  - パーミッションシステム
  - 生産性
  - Web標準への準拠
  - ES Modulesベースのモジュールシステム
  - 最新のJavaScript仕様に準拠

Denoの特徴

- 主な特徴は以下が挙げられる
  - TypeScriptを標準でサポート
  - パーミッションシステム
  - 生産性
  - Web標準への準拠
  - ES Modulesベースのモジュールシステム
  - 最新のJavaScript仕様に準拠

サンプルを作ってみる

以下のチュートリアルをベースに、どんな感じで実装するのか試してみる。

- Deno チュートリアル

インストール & 実行

Macなら以下のコマンドを実行

```
$ brew install deno
```

インストール & 実行

Macなら以下のコマンドを実行

```
$ brew install deno
```

DenoはURLから直接ファイルを実行できるので、以下のコマンドで実行可能

```
$ deno run https://deno.land/std@0.135.0/examples/welcome.ts


Welcome to Deno!
```

インストール & 実行

Macなら以下のコマンドを実行

```
$ brew install deno
```

DenoはURLから直接ファイルを実行できるので、以下のコマンドで実行可能

```
$ deno run https://deno.land/std@0.135.0/examples/welcome.ts


Welcome to Deno!
```

実行ファイルは、こちら

インストール & 実行

Macなら以下のコマンドを実行

```
$ brew install deno
```

DenoはURLから直接ファイルを実行できるので、以下のコマンドで実行可能

```
$ deno run https://deno.land/std@0.135.0/examples/welcome.ts


Welcome to Deno!
```

実行ファイルは、こちら

スクリプト実行方法

Denoでは以下のようなスクリプトの実行をサポートしている。

```
$ deno run main.ts
$ deno run https://mydomain.com/main.ts
$ cat main.ts | deno run -
```

パーミッション

Denoはファイル実行時の安全性を確保するため、デフォルトでは以下を禁止している

- ファイルの読み込み/書き込み
- ネットワークアクセス
- 環境変数の取得
- 外部コマンド実行
- プラグインの読み込み

```
$ deno run --allow-net net_client.ts
$ deno run --allow-read net_client.ts
$ deno run --allow-net net_client.ts
$ deno run --allow-net=example.com net_client.ts
$ deno run --allow-env net_client.ts
```

コマンドラインの紹介①

Denoでは標準で多くのコマンドをサポートしているので紹介

■ lint & fmt

```
$ deno lint ./
$ deno fmt ./
```

■ test

```
$ deno test
$ deno test --coverage=cov_profile
```

コマンドラインの紹介②

■ Watch mode

修正ファイルをすれば自動で再起動

```
$ deno run --watch main.ts
$ deno test --watch
$ deno fmt --watch
```

■ bundle

依存する複数のファイルを1つのjsファイルにして出力する。

```
$ deno bundle https://deno.land/std@0.83.0/examples/colors.ts colors.bundle.js
```

コマンドラインの紹介③

■ ドキュメント出力

JSDocを元にドキュメント出力

```
$ deno doc
```

■ break pointを使用

```
$ deno run --inspect-brk --allow-read --allow-net https://deno.land/std@0.135.0/http/file_server.ts
```

上記を実行してChromeで以下のURLにアクセスする。

```
chrome://inspect
```

コマンドラインの紹介の補足

紹介したCLTツールの詳細は以下で確認できる

- Deno Contributing

Lifecycle Event ①

Denoはブラウザ互換のライフサイクルイベントをサポートしている。

■ main.ts

```
const handler = (e: Event): void ⇒ {
  console.log(`got ${e.type} event in event handler (main)`);
};

window.addEventListener("load", handler);
window.addEventListener("unload", handler);

window.onload = (e: Event): void ⇒ {
  console.log(`got ${e.type} event in onload function (main)`);
};
window.onunload = (e: Event): void ⇒ {
  console.log(`got ${e.type} event in onunload function (main)`);
};

console.log("log from main script");
```

Lifecycle Event ②

コマンドを実行した結果は以下の通り。

```
$ deno run main.ts
log from main script
got load event in onload function (main)
got load event in event handler (main)
got unload event in onunload function (main)
got unload event in event handler (main)
```

試しにAPIを作ってみる

```
import { Application, Router, RouterContext } from "https://deno.land/x/oak@v6.5.0/mod.ts";

const app = new Application();
const router = new Router();

app.addEventListener("listen", ({ hostname, port, secure }) => {
  console.log(
    `Listening on: ${secure ? "https://" : "http://"}${hostname ??
      "localhost"}:${port}`,
  );
});
app.addEventListener("error", (evt) => {
  console.log(evt.error);
});
router.get('/', (ctx: RouterContext) => {
  ctx.response.body = "Hello World!";
})
app.use(router.routes());
app.use(router.allowedMethods());

await app.listen({ port: 8001 });
```

試しにReactを使ってみる

```
import { listenAndServe } from "https://deno.land/std@0.99.0/http/mod.ts";
import React from "https://jspm.dev/react@17.0.2";
import ReactDOMServer from "https://jspm.dev/react-dom@17.0.2/server";
function App() {
  return <div>Hello SSR</div>;
}
listenAndServe({ port: 8080 }, (req) => {
  req.respond({
    status: 200,
    headers: new Headers({
      "Content-Type": "text/html",
    }),
    body: ReactDOMServer.renderToString(
      <html>
        <head></head>
        <body>
          <div id="app">
            <App />
          </div>
        </body>
      </html>
    ),
  });
});
```

Denoでのサードパーティの扱い

Denoでは以下のサイトで公開されているパッケージに関して問題なく使用可能

- https://deno.land/x

既存のnpmは以下のレジストリを経由してインストール可能

- https://esm.sh/
- https://jspm.org/

ただし、上記が以下の使用してNode.js→Denoに変換しているので、**ポリフィルが未サポートのものを使用しているとエラーになるので注意**

- https://deno.land/std@0.106.0/node#supported-modules

まとめ

- Denoの標準でサポートしている内容が多く、基礎的な部分では標準だけでやっていけそう。
- 高い柔軟性とセキュリティの両方が考慮した設計になっている。
- バックエンドとフロンドを、両方共いい感じに実装できる感はある。
- 既存のnpmの資産がどこまで流用できるのかで、実際に使用していくかが決まりそう。この辺は、まだ調査不足なのでもう少し実装してみて確認してみる予定。

ご清聴ありがとうございました