

Deno: A Next-generation JavaScript runtime

Yongwook Choi



JSConf Korea 2022

소개

- 최용욱 (Yongwook Choi)
- DX Engineer @ Riiid
- flow@hrmm.xyz
- [@hyp3rflow \(GitHub\)](https://github.com/hyp3rflow)
- [@hrmm_flow \(twitter\)](https://twitter.com/hrmm_flow)



목차

- 새로운 프로젝트 만들어보기
- Deno의 향상된 개발자 경험
- Deno - Node.js 호환성
 - Deno에서 Node.js 코드 실행하기 (Node.js → Deno)
 - Deno 코드를 Node.js에서 실행하기 (Deno → Node.js)
- 어떻게 사용하고 있나요 - pbkit 개발
- 그래서 프로덕션에서 쓸 수 있을까요?

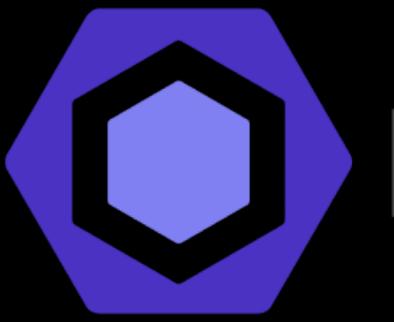
새로운 프로젝트 만들어보기

Node.js + TypeScript

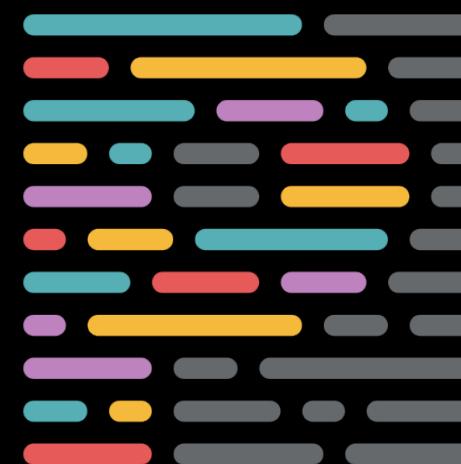
Package manager



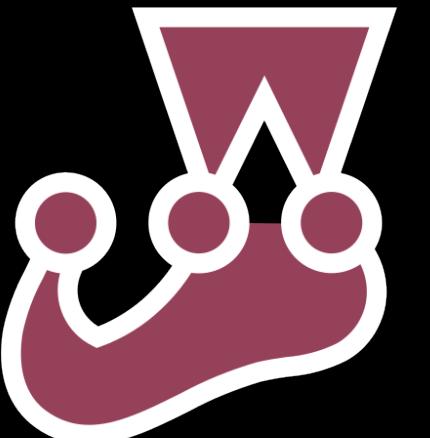
Linter / Formatter



ESLint



Testing Framework



새로운 프로젝트 만들어보기

Node.js + TypeScript

tsc / babel

ts-node

Transpiler → Node.js

sucrase

swc

Package manager

Testing framework

Linter / Formatter

TypeScript





새로운 모던 JavaScript 런타임, Deno

Package manager

Testing framework

Linter / Formatter

TypeScript

URL import

**Deno CLI
Subcommands**

Out of the box



향상된 개발자 경험

- 풍부하고 잘 선별된 표준 라이브러리 사용 가능
- Promise 기반의 비동기 API 제공
- Web Platform API를 준수하고자 노력
- 보안을 위한 권한 관리
- ESMODULE 지원

향상된 개발자 경험

풍부하고 잘 선별된 표준 라이브러리(Standard Library) 사용 가능

- Go에서 제공하는 표준 라이브러리 모듈들을 레버리지하여 작성
- `async`, `bytes`, `io`, `streams`, `http` 등 다양한 모듈들을 제공
 - `async`: `debounce`, `deferred`, `delay` ...
 - `bytes`: `Uint8Array`에 대한 도우미 함수 제공 (`concat`, `equals`, `copy` ...)
 - `io`: `Buffer`, `BufReader`, `BufWriter`
 - `streams`: Streams API에 대한 `Buffer`, `conversion` 제공 (`copy`, `read`, `write` ...)
 - `http`: `HttpServer`, `Errors`, `Status`, `Cookie` ...

향상된 개발자 경험

풍부하고 잘 선별된 표준 라이브러리(Standard Library) 사용 가능

- 표준 라이브러리는 Deno API와 다른 버전을 사용할 수 있음



Node.js v14
= API v14



Deno v1.25
!= std v0.153

향상된 개발자 경험

풍부하고 잘 선별된 표준 라이브러리(Standard Library) 사용 가능

- 표준 라이브러리는 Deno API와 다른 버전을 사용할 수 있음

```
import * from "https://deno.land/std@0.145.0/io/mod.ts";
```



Updated!

```
import * from "https://deno.land/std@0.156.0/io/mod.ts";
```

향상된 개발자 경험

Promise 기반의 비동기 API 제공

- Node.js → 비동기 API들이 callback으로 구현
- Promise 형태로 만들기 위해서는?
 - util에 있는 promisify를 이용
 - fs/promises 같은 promise 형식으로 제공되는 API 사용
- Deno → 모든 비동기 API들이 Promise를 반환

Node.js lib/fs.js readFile

```
function readFile(path, options, callback) {
  →callback = maybeCallback(callback || options);
  →options = getOptions(options, { flag: 'r' });
  →const context = new ReadFileContext(callback, options.encoding);
  →context.isUserFd = isFd(path); // File descriptor ownership

  if (options.signal) {
    context.signal = options.signal;
  }
  if (context.isUserFd) {
    process.nextTick(function tick(context) {
      ReflectApply(readFileAfterOpen, { context }, [null, path]);
    }, context);
    return;
  }

  if (checkAborted(options.signal, callback))
    return;

  const flagsNumber = stringToFlags(options.flag, 'options.flag');
  path = getValidatedPath(path);

  const req = new FSReqCallback();
  →req.context = context;
  →req.oncomplete = readFileAfterOpen;
  binding.open(pathModule.toNamespacedPath(path),
               flagsNumber,
               0o666,
               →req);
}
```

Deno runtime/40_read_file.js readFile

```
→async function readFile(path, options) {
  let cancelRid;
  let abortHandler;
  if (options?.signal) {
    options.signal.throwIfAborted();
    cancelRid = ops.op_cancel_handle();
    abortHandler = () => core.tryClose(cancelRid);
    options.signal[abortSignal.add](abortHandler);
  }

  try {
    const read = await core.opAsync(
      "op_readfile_async",
      pathFromURL(path),
      cancelRid,
    );
    return read;
  } finally {
    if (options?.signal) {
      options.signal[abortSignal.remove](abortHandler);
      // always throw the abort error when aborted
      options.signal.throwIfAborted();
    }
  }
}
```

Node.js

lib/internal/fs/promises.js readFile

```
async function readFile(path, options) {
  ··· options = getOptions(options, { flag: 'r' });
  ··· const flag = options.flag || 'r';

  ··· if (path instanceof FileHandle)
  ···   return readFileHandle(path, options);

  ··· checkAborted(options.signal);

  ··· const fd = await open(path, flag, 0o666);
  ··· return handleFdClose(readFileHandle(fd, options), fd.close);
}
```

```
async function readFileHandle(filehandle, options) {
  ··· const signal = options?.signal;

  ··· checkAborted(signal);

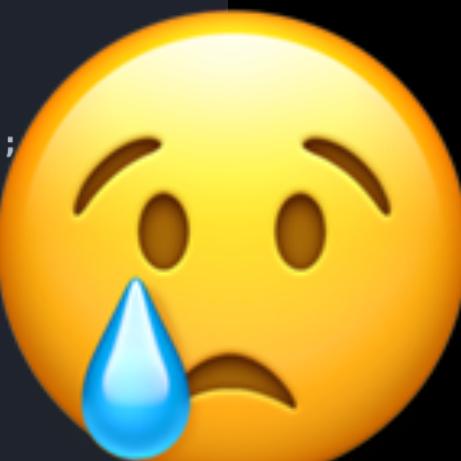
  ··· const statFields = await binding.fstat(filehandle.fd, false, kUsePromises);
  ··· checkAborted(signal);

  ··· let size;
  ··· if ((statFields[1/* mode */] & S_IFMT) === S_IFREG) {
  ···   size = statFields[8/* size */];
  ··· } else {
  ···   size = 0;
  ··· }

  ··· if (size > kIoMaxLength)
  ···   throw new ERR_FS_FILE_TOO_LARGE(size);

  ··· let endOfFile = false;
  ··· let totalRead = 0;
  ··· const noSize = size === 0;
  ··· const buffers = [];
  ··· const fullBuffer = noSize ? undefined : Buffer.allocUnsafeSlow(size);
  ··· do {
  ···   checkAborted(signal);
  ···   let buffer;
  ···   let offset;
  ···   let length;
  ···   if (noSize) {
  ···     buffer = Buffer.allocUnsafeSlow(kReadFileUnknownBufferLength);
  ···     offset = 0;
  ···     length = kReadFileUnknownBufferLength;
  ···   } else {
  ···     buffer = fullBuffer;
  ···     offset = totalRead;
  ···     length = MathMin(size - totalRead, kReadFileBufferLength);
  ···   }

  ···   const bytesRead = (await binding.read(filehandle.fd, buffer, offset,
  ···                         length, -1, kUsePromises)) || 0;
  ···   totalRead += bytesRead;
  ···   endOfFile = bytesRead === 0 || totalRead === size;
  ···   if (noSize && bytesRead > 0) {
  ···     const isBufferFull = bytesRead === kReadFileUnknownBufferLength;
  ···     const chunkBuffer = isBufferFull ? buffer : fullBuffer;
  ···     ArrayPrototypePush(buffers, chunkBuffer);
  ···   }
  ···   if (size > 0) {
  ···     result = totalRead === size ? fileOpPromise : PromisePrototypeThen(
  ···       fileOpPromise,
  ···       (result) => PromisePrototypeThen(closeFunc(), () => result),
  ···       (opError) => PromisePrototypeThen(
  ···         closeFunc(),
  ···         () => PromiseReject(opError),
  ···         (closeError) => PromiseReject(aggregateTwoErrors(closeError, opError))
  ···       )
  ···   );
  ···   }
  ···   return options.encoding ? result : buffers;
  ··· }
}
```



Deno stdlib

fs/move.ts

```
/** Moves a file or directory */
export async function move(
  ··src: string,
  ··dest: string,
  ·{ overwrite = false }: MoveOptions = {},
) {
  ·const srcStat = await Deno.stat(src);

  ·if (srcStat.isDirectory && isSubdir(src, dest)) {
    ·throw new Error(
      ···`Cannot move '${src}' to a subdirectory of itself, '${dest}'.`,
    );
  }

  ·if (overwrite) {
    ·try {
      ···await Deno.remove(dest, { recursive: true });
    } catch (error) {
      ···if (!(error instanceof Deno.errors.NotFound)) {
        ·····throw error;
      }
    }
  } else {
    ·try {
      ···await Deno.lstat(dest);
    } catch {
      ·····// Do nothing...
    }
  }

  ···await Deno.rename(src, dest);

  ···return;
}
```

향상된 개발자 경험

Web Platform API를 준수하고자 노력

- Deno에서는 비슷한 Web API가 있는 경우 바퀴의 재발명을 지양
- Fetch, Console, Streams, Web Storage, Web Worker 등 Web API 구현
 - 브라우저에서 사용하던 API를 Deno에서도 사용 가능

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet	WebView Android	Deno
localStorage	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	4	12	3.5	10.5	4	18	4	11	3.2	1.0	37	1.16

*...

향상된 개발자 경험

보안을 위한 권한 관리, ESM 지원

- 파일 시스템, 네트워크, 환경 변수 접근 등이 필요할 때 추가적인 권한 필요
 - Node.js → 권한 설정이 불가능하여 악의적인 코드가 마음대로 접근 가능
- CommonJS를 지원하지 않고 오직 ES Modules만을 지원
 - 오래되고 다양한 모듈 방식들은 코드/툴링의 난이도를 복잡하게 만듦

Node.js 호환성

Node.js 코드를 Deno에서 실행하기

Deno 코드를 Node.js에서 실행하기

Node.js 호환성

1. std/node: Deno에서 Node.js API 사용하기

- 표준 라이브러리의 std/node 모듈을 사용 가능
- 모든 기능이 지원되지는 않으나,
자주 사용하는 라이브러리 대응을 목표로 호환을 넓혀가고 있음
- <https://deno.land/std/node>

<input checked="" type="checkbox"/> assert	<input checked="" type="checkbox"/> path posix
<input checked="" type="checkbox"/> assert/strict <i>partly</i>	<input checked="" type="checkbox"/> path win32
<input checked="" type="checkbox"/> async_hooks <i>partly</i>	<input checked="" type="checkbox"/> perf_hooks
<input checked="" type="checkbox"/> buffer	<input checked="" type="checkbox"/> process <i>partly</i>
<input checked="" type="checkbox"/> child_process <i>partly</i>	<input checked="" type="checkbox"/> punycode
<input checked="" type="checkbox"/> cluster <i>partly</i>	<input checked="" type="checkbox"/> querystring
<input checked="" type="checkbox"/> console <i>partly</i>	<input checked="" type="checkbox"/> readline
<input checked="" type="checkbox"/> constants <i>partly</i>	<input checked="" type="checkbox"/> repl <i>partly</i>
<input checked="" type="checkbox"/> crypto <i>partly</i>	<input checked="" type="checkbox"/> stream
<input checked="" type="checkbox"/> dgram <i>partly</i>	<input checked="" type="checkbox"/> stream promises
<input type="checkbox"/> diagnostics_channel	<input checked="" type="checkbox"/> stream web <i>partly</i>
<input checked="" type="checkbox"/> dns <i>partly</i>	<input checked="" type="checkbox"/> string_decoder
<input checked="" type="checkbox"/> events	<input checked="" type="checkbox"/> sys
<input checked="" type="checkbox"/> fs <i>partly</i>	<input checked="" type="checkbox"/> timers
<input checked="" type="checkbox"/> fs promises <i>partly</i>	<input checked="" type="checkbox"/> timers promises
<input checked="" type="checkbox"/> http <i>partly</i>	<input type="checkbox"/> tls
<input type="checkbox"/> http2	<input type="checkbox"/> trace_events
<input checked="" type="checkbox"/> https <i>partly</i>	<input checked="" type="checkbox"/> tty <i>partly</i>
<input checked="" type="checkbox"/> inspector <i>partly</i>	<input checked="" type="checkbox"/> url
<input checked="" type="checkbox"/> module	<input checked="" type="checkbox"/> util <i>partly</i>
<input checked="" type="checkbox"/> net	<input checked="" type="checkbox"/> util types <i>partly</i>
<input checked="" type="checkbox"/> os <i>partly</i>	<input type="checkbox"/> v8
<input checked="" type="checkbox"/> path	<input checked="" type="checkbox"/> vm <i>partly</i>
	<input checked="" type="checkbox"/> wasi
	<input type="checkbox"/> webcrypto
	<input checked="" type="checkbox"/> worker_threads
	<input type="checkbox"/> zlib

Node.js 호환성

2. Node compatibility mode: Node.js 코드를 Deno로 실행시키기

- Node compatibility mode를 이용해 Deno 위에서 실행 가능
 - CommonJS 코드 실행 가능 (`__dirname`, `__filename`도 처리)
 - Node.js global(Buffer, process, ..) 사용 가능
 - Node.js API를 std/node polyfill로 변경
 - deno run --compat 플래그로 사용 가능

```
● ● ●  
const payload = "filename: " + __filename + "\n";  
process.stdout.write(new TextEncoder().encode(payload));
```



```
Object.defineProperty(globalThis, "process", {  
    value: processModule,  
    enumerable: false,  
    writable: true,  
    configurable: true,  
});  
  
(function (require, __filename, ...) {  
    (function (require, __filename, ...) {  
        const payload = "filename: " + __filename + "\n";  
        process.stdout.write(new TextEncoder().encode(payload));  
    }).call(this, require, __filename, ...);  
}) // -> call this function with computed values
```

Node.js 호환성

3. Deno-friendly CDN: NPM 모듈을 Deno에서 import하기

- Deno 친화적인 CDN에서 Deno와 호환되는 NPM 라이브러리를 제공
 - esm.sh / skypack
 - CJS → ESM 변환
 - TypeScript를 위한 typing 제공 (X-Typescript-Types header)
 - Compat mode처럼 std/node polyfill도 주입
- import from "<https://esm.sh/react>"

Node.js 호환성

4. NPM Specifier: 더 편하게 NPM 모듈 사용하기

- 이전의 Node Compatibility mode가 삭제되고 새롭게 추가된 호환성 방법
- `import * from "npm:express@4"`
- deno run으로도 npm에 있는 Node.js 코드 실행 가능

```
~/ .deno (0.426s)
deno run --unstable -A npm:cowsay hello

<----->
-----
 \  ^__^
  (oo)\_____
  (__)\       )\/\
    ||----w |
    ||     ||
```

Node.js 호환성

Future: Implement Node API #13633

- 성능 상 혹은 이식성의 이점을 위해 Node.js Addon을 사용
- 저수준의 Addon 코드에서 JavaScript 값을 조작할 수 있는 Node API 제공
- NPM에 있는 sqlite나 이미지 처리 라이브러리 sharp

Node.js 호환성

1. Deno에서 작성된 코드를 Node.js에서 쓰고 싶은 경우

- dnt (Deno - Node transform) - <https://github.com/denoland/dnt>
- Deno 코드를 Node.js 코드로 변환해주는 Deno 라이브러리
- dnt를 통해 NPM 패키지를 만든 후 Node.js에서 사용 가능

```
// ex. scripts/build_npm.ts
import { build, emptyDir } from "https://deno.land/x/dnt/mod.ts";

await emptyDir("./npm"); ←

await build({
  entryPoints: ["./mod.ts"], ←
  outDir: "./npm",
  shims: {
    // see JS docs for overview and more options
    deno: true,
  },
  package: { ←
    // package.json properties
    name: "your-package",
    version: Deno.args[0],
    description: "Your package.",
    license: "MIT",
    repository: {
      type: "git",
      url: "git+https://github.com/username/repo.git",
    },
    bugs: {
      url: "https://github.com/username/repo/issues",
    },
  },
});
}

// post build steps
Deno.writeFileSync("LICENSE", "npm/LICENSE");
Deno.writeFileSync("README.md", "npm/README.md");
```

Node.js 호환성

1. Deno에서 작성된 코드를 Node.js에서 쓰고 싶은 경우

- 모듈 식별자를 재작성
- Deno namespace의 shim 주입
- skypack/esm.sh → package.json
- NPM package 지정 가능

Node.js 호환성

2. Deno에서 작성된 코드를 Node.js에서 쓰고 싶은 경우

- Deno API에 대한 의존성이 없다면 dnt를 사용하지 않아도 됨
- esbuild + esbuild_deno_loader
 - Deno의 import 의존성을 해결할 수 있도록 해주는 플러그인

Node.js 호환성

2. Deno에서 작성된 코드를 Node.js에서 쓰고 싶은 경우

- github.com/esbuild/deno-esbuild
- [github.com/lucacasonato/esbuild deno loader](https://github.com/lucacasonato/esbuild_deno_loader)
- 간단하게 esbuild를 이용해 번들링하는 Deno 코드 완성!

```
import * as esbuild from "https://deno.land/x/esbuild@v0.14.51/mod.js";
import { denoPlugin } from "https://deno.land/x/esbuild_deno_loader@0.5.2/mod.ts";

await esbuild.build({
  ··plugins: [denoPlugin()],
  ··entryPoints: ["https://deno.land/std@0.150.0/hash/sha1.ts"],
  ··outfile: "./dist/sha1.esm.js",
  ··bundle: true,
  ··format: "esm",
});
esbuild.stop();
```

어떻게 사용하고 있나요

pbkit 라이브러리 개발

pbkit Core

pbkit Codegen

**Protobuf
Language
Server**

어떻게 사용하고 있나요

pbkit 라이브러리 개발

NPM / Deno 의존성 없음

브라우저에서 동작해야 함

pbkit Core

import 문의 확장자만 제거

어떻게 사용하고 있나요

pbkit 라이브러리 개발

Deno API 의존성 없음

번들링 될 필요가 있음

pbkit Codegen

`esbuild`

+ `esbuild_deno_loader`

어떻게 사용하고 있나요

pbkit 라이브러리 개발

Protobuf
Language
Server

NPM 의존성: `vscode-jsonrpc`

Node.js (VSCode)에서 실행

`dnt (deno-node transform)` 사용

어떻게 사용하고 있나요

pbkit 라이브러리 개발

pbkit Core

확장자 제거

pbkit Codegen

esbuild

**Protobuf
Language
Server**

dnt

그래서 프로덕션에서 쓸 수 있을까요?

- 개발 툴링 같이 NPM 의존성이 적은 분야라면 GOOD
- 의존하는 NPM 모듈이 호환되는지 여부 확인 ~~기여해주세요~~
- Node.js 호환성을 제공해야 하는 입장에서는 큰 문제 없음

감사합니다

