Deno

Deno란?

- Typescript & Javascript Runtine Engine
- Node.js와 동일하게 내부적으로 V8 엔진을 사용한다.
- Node.js 개발진들이 node의 단점을 개발하기 위해서 만든 새로운 엔진

Node.js의 단점(1)

- Promise를 고집하지 않은 것. Node.js의 비동기 호출은 여전히 콜백 API을 기준으로 작성되어 있다.
- 보안문제 취약 - ssh키 등 모든 파일에 대해 접근이 가능하고 승인 없이 모든 권한을 사용할 수 있다.
- GYP 빌드 시스템 - 크롬에서도 사용하는 GN이라는 빌드시스템이 20배정도 빠른데 이를 계속 업그레이드하지 않고 있다.
- GYP는 Native call을 하려면 사용자가 직접 C++ 바인딩을 해야하는데 이를 위한 FFI(Foreign Function Interface)가 없다는 점

Node.js의 단점(2)

- npm이라는 패키지 매니저에 매우 의존적이며 package.json 파일의 모듈 시스템이 단순히 "모듈 시스템"이 아니라 라이센스, 리포지토리 등 필요없는 정보들을 모두 담고있다.
- node_modules라는 존재때문에 모듈을 가져오는 알고리즘이 복잡해졌고 브라우저에서 실제로 가져오는 방식과 맞지도 않아 따로 번들링을 하는 등의 작업을 해야했다.
- Require문법을 쓸 때 확장자를 안써도 되게 한것(왜 단점인지는 잘 모르겠네요. 아마 ts때문인 듯..?)
- index.js -> 브라우저가 index.html을 default로 갖기에 index.js를 지정한게 괜찮아보였으나, 결과적으로 모듈 로딩시 시스템을 더 복잡하게 만들었다.

Deno <-> Node.js와의 차이점

- deno는 npm을 사용하지 않으며 URL또는 파일 경로를 통해 모듈을 로드한다.
- deno는 모듈 결정 알고리즘에 package.json을 사용하지 않는다.
- deno의 모든 비동기적 작업은 promise를 반환한다. 그러므로 NodeJS와는 다른 API를 제공한다.
- deno는 파일, 네트워크, OS환경 접근에 대해 명시적인 권한을 요구한다.
- deno는 핸들링하지 않은 에러 발생시 앱을 종료시킨다.
- ES Modules를 사용하며 require() 구문은 사용하지 않는다. 외부 라이브러리의 경우 URL을 통해 import한다.
    - `import * as log from "https://deno.land/std/log/mod.ts";`

Deno 특징

- 보안이 기본 설정이며 명시적으로 허락이 되지 않는한 파일, 네트워크, OS환경에 접근 할 수 없다.
- Typescript를 기본적으로 지원한다.
- 단일 실행 파일을 제공한다.
- 의존성 검사(deno info), code formatter (deno fmt)와 같은 내장 유틸리티 도구를 제공한다.
- deno와 함계 동작하도록 검수 및 설계된 표준 모듈이 제공된다. (std library 제공!)
- 스크립트 파일은 하나의 js파일로 번들링 되는 기능을 제공한다. (deno bundle)

Deno 세부 특징들

- 블로그 를 참고하시면 됩니다. 매우 자세하게 Deno의 특징, 한계점 등이 나와있습니다.

Deno 설치

```
brew install deno
```

Deno 실습

References

- https://han41858.tistory.com/50
- https://blog.ull.im/engineering/2019/04/14/deno-ryan-dahl-2019-04-04.html
- The Net Ninja 강의