Next.js Turbopack

Made with GAMMA

Turbopack이란?

차세대 번들러 도구

Turbopack은 **React Server Components와 TypeScript 코드베이스를 위한 고성능 번들러**

- **Rust 언어**로 작성되어 있고, **Next.js에 내장**되어 있습니다.
- **JavaScript와 TypeScript**를 빠르게 빌드·번들링할 수 있도록 최적화되었습니다.

Made with GAMMA

Rust 기반으로의 전환

Babel → SWC

- **Babel**: 자바스크립트/타입스크립트 코드를 다른 버전(JS 표준)으로 변환(트랜스파일)하는 도구.
- **SWC (Speedy Web Compiler)**: Rust로 작성된 트랜스파일러. Babel 대비 **최대 17배 빠른 성능** 제공.

Terser → SWC minifier

- **Terser**: 자바스크립트 코드를 압축/최적화하는 기존 JS 기반 도구.
- **SWC Minifier**: Rust 기반의 압축기. Terser 대비 **최대 6배 빠른 압축 속도**

Webpack → Turbopack

React Server Components와 TypeScript에 최적화

Webpack 대비 **수십 배 빠른 개발 빌드**와 **더 효율적인 프로덕션 빌드**.

Made with GAMMA

왜 Turbopack인가?

Next.js의 성능을 향상시키기 위해 **Turbopack**을 만들었습니다.

1. **Bundling (vs Native ESM)**
2. **Incremental Computation (증분 계산)**
3. **Lazy Bundling**

Made with GAMMA

Bundling vs Native ESM

일부 도구들은 개발 모드에서 **아예 번들링을 생략하고 있습니다.** 대표 예시 **Vite가 있습니다.**

즉, `import`가 있으면 브라우저가 직접 그 파일을 요청합니다.

```
// main.js
import { add } from './utils/math.js';
```

브라우저가 실제로
`GET /src/utils/math.js`
`GET /src/utils/helpers.js`
이런 식으로 **직접 날립니다.**

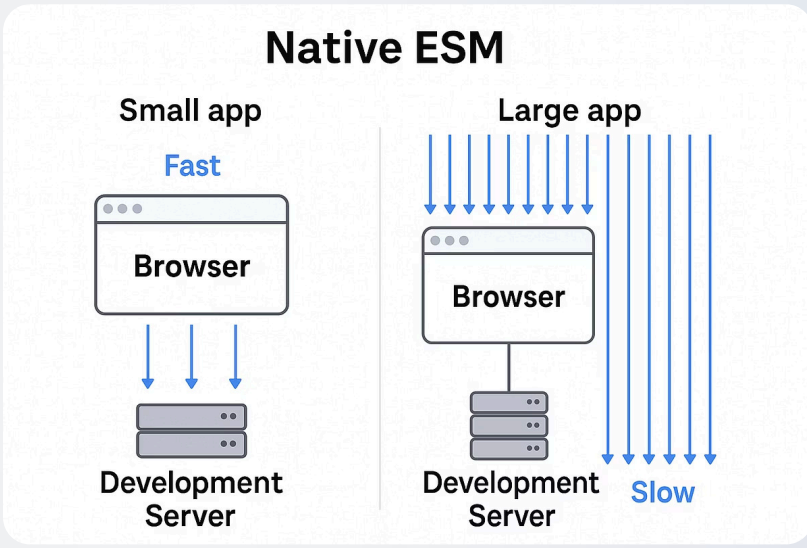
그래서 개발 서버는 코드를 번들링해서 한 번에 주는 대신, 요청 오는 파일만 빠르게 전달하게 됩니다.

수천 개의 JS 모듈이 있는 대규모 앱이라면?

```
// main.js
import { add } from './utils/math.js';
import { add } from './utils/helpers.js';
import { add } from './index.js';
... 수백 ~ 수천 개의 js 파일들
```

브라우저는 실제로 수백 ~ 수천 개의 요청을 직접 날리게 됩니다.

- 브라우저가 각 모듈을 ESM import로 요청함 → **수천 개의 요청 발생**
- 요청마다 HTTP 연결, 응답 지연 → **dev 서버 속도 급감**



Made with GAMMA

Bundling vs Native ESM

Turbopack은 이런 문제를 해결하기 위해
개발 모드에서도 번들링을 하되, **필요한 부분만 빠르게 번들링**한다고 합니다.

이걸 증분 계산+ Lazy bundling과 결합해서, **네트워크 요청도 줄이고, 빌드도 빠른** 형태로 만들어 변화하고 있습니다.

Made with GAMMA

증분 계산(Incremental Computation) - webpack

전체를 다시 빌드하지 않고, 변경된 부분만 다시 계산하는 것

webpack

```
src/
  ├── index.js
  ├── utils.js
  ├── math/
  │    ├── add.js
  │    └── subtract.js
```

`index.js`가 `utils.js`와 `math/add.js`를 **import**하고 있다고 치면

```
// index.js
import { add } from './math/add.js';
import { log } from './utils.js';
```

`add.js` 파일의 해시가 변경됨을 감지하여 `add.js`와 그것을 의존하는 상위 파일(`index.js`)까지 **재컴파일**

여기서 **add.js 파일만 수정**하면 Webpack은 이렇게 동작합니다

- add.js 파일의 해시가 변경됨을 감지
- add.js와 그것을 의존하는 상위 파일(index.js)까지 **재컴파일**

Made with GAMMA

증분 계산(Incremental Computation)

Turbopack

```
// add.js
export function add(a, b) {
  return a + b;
}

export function multiply(a, b) {
  return a * b;
}
```

여기서 multiply()만 수정했는데, 실제 페이지에서 add()만 사용 중이라면?

변경된 함수(multiply)만 다시 계산하고 add 관련 번들은 **건드리지 않습니다.**

즉, **파일 내 특정 함수만 바뀌어도 전체 파일이 아니라 그 함수 경로만 다시 계산**하는 구조

Made with GAMMA

Lazy Bundling

개발 단계부터 요청된 라우트/모듈만 즉시 컴파일하고 나머지는 요청 시점에 **on-demand**

첫 화면만 본다면 **그 화면에 필요한 모듈 경로만** 우선 컴파일하고, 사용자가 다른 라우트로 이동할 때 그제야 관련 모듈을 **그때** 빌드합니다.

콜드 스타트/ fast 리프레시가 훨씬 빠르며, 큰 규모의 앱의 경우 체감이 클거라고 예상합니다.

Made with GAMMA

Next.js에서 Turbopack 사용법

개발 환경 설정

```
//package.json
{
  "scripts": {
    "dev": "next dev --turbopack",
    "build": "next build --turbopack",
    "start": "next start"
  }
}
```

현재 지원 상태

현재 dev build 환경에서 안정적으로 사용할 수 있으며, 점진적으로 production build 지원이 확대되고 있습니다

Made with GAMMA

dev build 적용 후 비교

npm run dev

webpack vs **turbopack**

```
> next dev

   ▲ Next.js 15.5.5
   - Local:        http://localhost:3000
   - Network:      http://192.168.202.35:3000

 ✓ Starting...
 ✓ Ready in 1128ms
 ○ Compiling / ...
 ✓ Compiled / in 3.8s (1028 modules)
(node:14744) [DEP0169] DeprecationWarning: `url.parse()
```

Add to Chat ⌘L

```
> next dev --turbopack

   ▲ Next.js 15.5.5 (Turbopack)
   - Local:        http://localhost:3000
   - Network:      http://192.168.202.35:3000

 ✓ Starting...
 ✓ Compiled middleware in 72ms
 ✓ Ready in 885ms
 ○ Compiling / ...
 ✓ Compiled / in 3.1s
```

npm run dev

webpack vs **turbopack**

```
   ▲ Next.js 15.5.5
   - Local:        http://localhost:3000
   - Network:      http://192.168.202.35:3000

 ✓ Starting...
 ✓ Ready in 939ms
 ○ Compiling / ...
 ✓ Compiled / in 3.8s (1028 modules)
 GET / 200 in 4148ms
^C
kimbeomsu@gimbeomsuui-MacBookPro front %
kimbeomsu@gimbeomsuui-MacBookPro front % npm run dev

> sizz-front@0.1.0 dev
> NODE_NO_WARNINGS=1 next dev

   ▲ Next.js 15.5.5
   - Local:        http://localhost:3000
   - Network:      http://192.168.202.35:3000

 ✓ Starting...
 ✓ Ready in 1085ms
 ✓ Compiled / in 457ms (1005 modules)
kimbeomsu@gimbeomsuui-MacBookPro front %
```

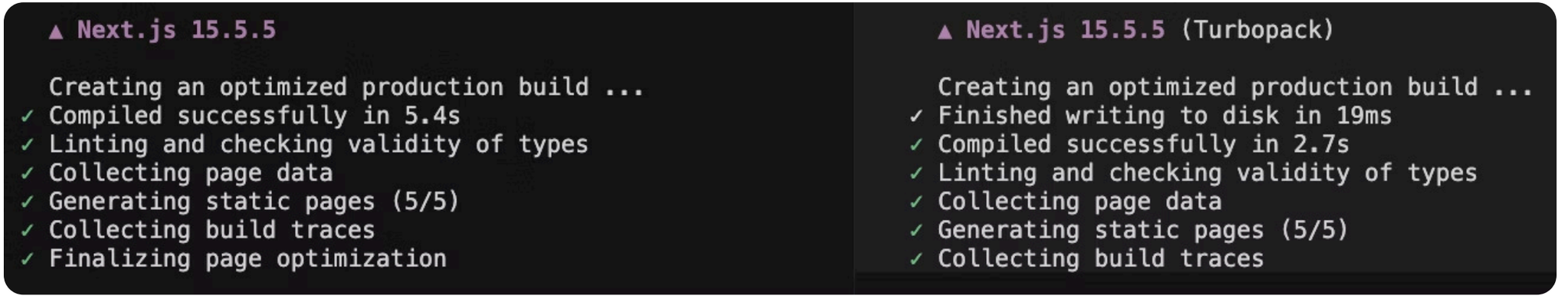
```
   ▲ Next.js 15.5.5 (Turbopack)
   - Local:        http://localhost:3000
   - Network:      http://192.168.202.35:3000

 ✓ Starting...
 ✓ Compiled middleware in 55ms
 ✓ Ready in 613ms
 ○ Compiling / ...
 ✓ Compiled / in 3s
 GET / 200 in 3512ms
^C
kimbeomsu@gimbeomsuui-MacBookPro front % npm run dev

> sizz-front@0.1.0 dev
> NODE_NO_WARNINGS=1 next dev --turbopack

   ▲ Next.js 15.5.5 (Turbopack)
   - Local:        http://localhost:3000
   - Network:      http://192.168.202.35:3000

 ✓ Starting...
 ✓ Compiled middleware in 57ms
 ✓ Ready in 713ms
 ○ Compiling / ...
 ✓ Compiled / in 2.9s
```

Made with GAMMA

production build 적용 후 비교

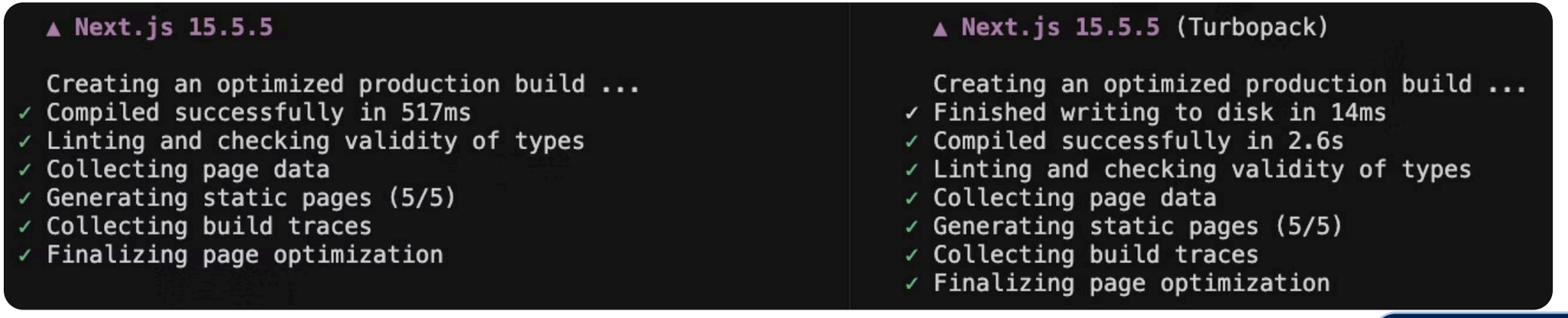
npm run build

1번째



npm run build

2번째



Made with GAMMA

Next.js 16v (beta)

npm run dev

1번째

```
 ▲ Next.js 16.0.0-canary.4 (Turbopack)
 - Local:        http://localhost:3000
 - Network:      http://192.168.202.35:3000
 - Experiments (use with caution):
   ✓ turbopackFileSystemCacheForBuild
   ✓ turbopackFileSystemCacheForDev

✓ Starting...
✓ Ready in 455ms
○ Compiling / ...
✓ Compiled / in 2.5s
GET / 200 in 2649ms
```

2번째

```
 ▲ Next.js 16.0.0-canary.4 (Turbopack)
 - Local:        http://localhost:3000
 - Network:      http://192.168.202.35:3000
 - Experiments (use with caution):
   ✓ turbopackFileSystemCacheForBuild
   ✓ turbopackFileSystemCacheForDev

✓ Starting...
✓ Ready in 269ms
✓ Compiled / in 53ms
GET / 200 in 208ms
```

npm run build

1번째

```
 ▲ Next.js 16.0.0-canary.4 (Turbopack)
 - Experiments (use with caution):
   ✓ turbopackFileSystemCacheForBuild
   ✓ turbopackFileSystemCacheForDev

 Creating an optimized production build ...
✓ Compiled successfully in 2.6s
✓ Finished TypeScript in 747ms
✓ Collecting page data in 147ms
✓ Generating static pages (4/4) in 169ms
✓ Finalizing page optimization in 6ms
```

2번째

```
 ▲ Next.js 16.0.0-canary.4 (Turbopack)
 - Experiments (use with caution):
   ✓ turbopackFileSystemCacheForBuild
   ✓ turbopackFileSystemCacheForDev

 Creating an optimized production build ...
✓ Compiled successfully in 76ms
✓ Finished TypeScript in 623ms
✓ Collecting page data in 143ms
✓ Generating static pages (4/4) in 165ms
✓ Finalizing page optimization in 5ms
```

Made with GAMMA

체감하는 이점

저지연 Fast Refresh(HMR)

증분 계산(Incremental Computation) + 지연 번들링(Lazy Bundling)

지속 캐시(파일시스템 캐싱, 베타)

큰 앱의 재시작·재컴파일 시간을 더 줄이는 **FileSystem 캐시**가 Next16(beta)에 들어왔습니다.

콜드 스타트/첫 컴파일 단축

dev 서버 띄우고 첫 라우트 진입까지의 대기 시간이 줄어듭니다.

빌드(프로덕션)

Next.js 15.5부터 next build --turbopack 베타가 열렸고, 멀티코어 환경에서 **2–5배 단축 사례**가 보고됐다. **소형 앱은 체감이 제한적**일 수 있다.

증분·지연 번들링으로 “필요한 것만” 컴파일

Turbopack은 요청된 경로에 필요한 모듈만 **lazy)** 빌드합니다. 네이티브 ESM 개발 흐름에서 발생하는 네트워크 워터폴과 과도한 컴파일 낭비를 줄여 대형 앱일수록 효과가 큽니다.

Made with GAMMA

버전 히스토리

1. **v16.0(beta)**
   Turbopack becomes the default bundler for Next.js
2. **v15.5.0**
   Turbopack support for `build` beta
3. **v15.3.0**
   Experimental support for `build`
4. **v15.0.0**
   Turbopack for `dev` stable

Made with GAMMA

출처 및 참고 자료

- Introducing Turbopack
- API Reference: Turbopack | Next.js

Made with GAMMA