🧩 CJS와 ESM 강의 노트(CommonJS vs ECMAScript Module)

1️⃣ 모듈이란?
모듈(Module) = 코드의 재사용과 유지보수를 쉽게 하기 위해 기능 단위로 분리한 코드 조각

- 예: app.js, math.js, router.js처럼 기능별로 파일을 나눔
- 다른 파일에서 불러오거나(export/import) 사용할 수 있음

Node.js에서는 두 가지 방식의 모듈 시스템을 지원합니다.

구분	CommonJS (CJS)	ECMAScript Module (ESM)
등장 배경	Node.js 초기부터 사용	ES6(2015) 표준 모듈 시스템
파일 확장자	`.js` 또는 `.cjs`	`.mjs` 또는 `package.json`에 `"type": "module"`
모듈 내보내기	`module.exports`	`export`, `export default`
모듈 가져오기	`require()`	`import`
비동기 지원	동기 방식	비동기 방식 (Top-Level Await 가능)

2️⃣ CJS (CommonJS) 방식
Node.js 초기에 만들어진 방식으로, 대부분의 오래된 프로젝트나 라이브러리가 CJS로 작성되어 있습니다.

☑️ 문법

math.cjs

```
function add(a, b) {
  return a + b;
}

module.exports = { add };  // 내보내기
```

app.cjs

```
const math = require('./math.cjs'); // 가져오기

console.log(math.add(2, 3)); // 5
```

⚙️ 실행

```
node app.cjs
```

3️⃣ ESM (ECMAScript Module) 방식 JavaScript 표준(ES6)에서 정식으로 도입된 모듈 시스템으로, 브라우저와 Node.js 모두에서 사용됩니다.

☑️ 문법 math.mjs

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

app.mjs

```
import { add } from './math.mjs';

console.log(add(2, 3)); // 5
```

⚙️ 실행

```
node app.mjs
```

또는 package.json에 "type": "module" 추가 후 .js 파일 확장자로 실행 가능:

```
{
  "type": "module"
}
```

4️⃣ Express에서의 차이점

항목	CJS 방식	ESM 방식
Express 불러오기	`const express = require('express');`	`import express from 'express';`
Router 생성	`const router = express.Router();`	`import { Router } from 'express'; const router = Router();`
서버 실행	`app.listen(3000, ...)` 동일	동일
실행 명령어	`node app.js`	`"type": "module"` 필요 또는 `.mjs` 확장자

💡 CJS 버전 (app.cjs)

```
const express = require('express');
const app = express();
```

```
app.get('/', (req, res) => {
  res.send('Hello from CJS Express!');
});

app.listen(3000, () => console.log('Server running on port 3000'));
```

💡 ESM 버전 (app.mjs)

```
import express from 'express';

const app = express();

app.get('/', (req, res) => {
  res.send('Hello from ESM Express!');
});

app.listen(3000, () => console.log('Server running on port 3000'));
```

6️⃣ 혼용 사용 (CJS + ESM 같이 쓰기)
Node.js는 "파일 단위"로 CJS/ESM을 구분합니다.
즉, 한 파일 안에서는 섞을 수 없지만 서로 다른 파일을 호출하는 건 가능합니다.

예시:

- package.json: "type": "commonjs"
- app.cjs: Express 실행 (CJS)
- routes.mjs: Router 정의 (ESM)

```
// app.cjs
const express = require('express');
const app = express();

(async () => {
  const { default: router } = await import('./routes.mjs');
  app.use('/api', router);
})();

app.listen(3000, () => console.log('http://localhost:3000'));


// routes.mjs
import { Router } from 'express';
const router = Router();
router.get('/', (req, res) => res.send('Hello from mixed mode!'));
export default router;
```

7 장단점 비교

구분	CJS	ESM
장점	Node.js 기본, 설정 간단	브라우저와 호환, 최신 JS 문법 사용 가능
단점	브라우저에서 안 돌아감, Top-Level Await 불가	설정 필요(`type: module`), 일부 라이브러리 호환성 문제
추천 상황	기존 코드 유지보수, 빠른 실습	신규 프로젝트, 프론트엔드·백엔드 통합 환경