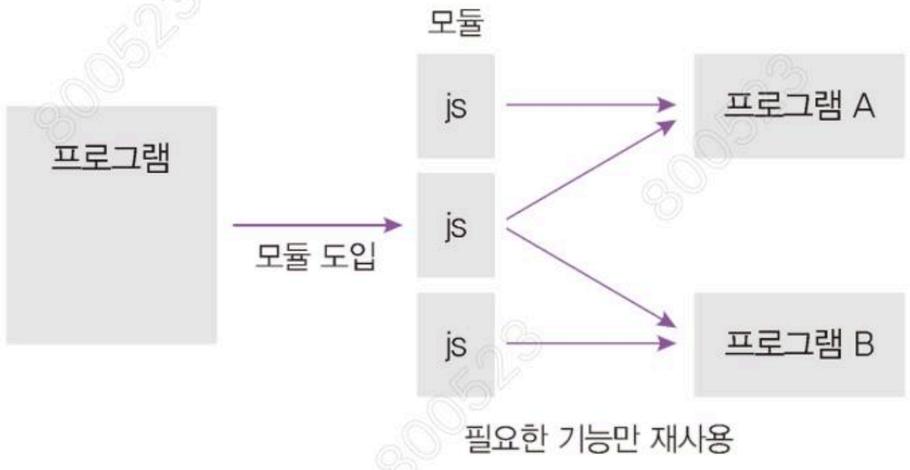
[2주차] Node JS to TS

3장: 노드 기능 알아보기 / 박범식

목차

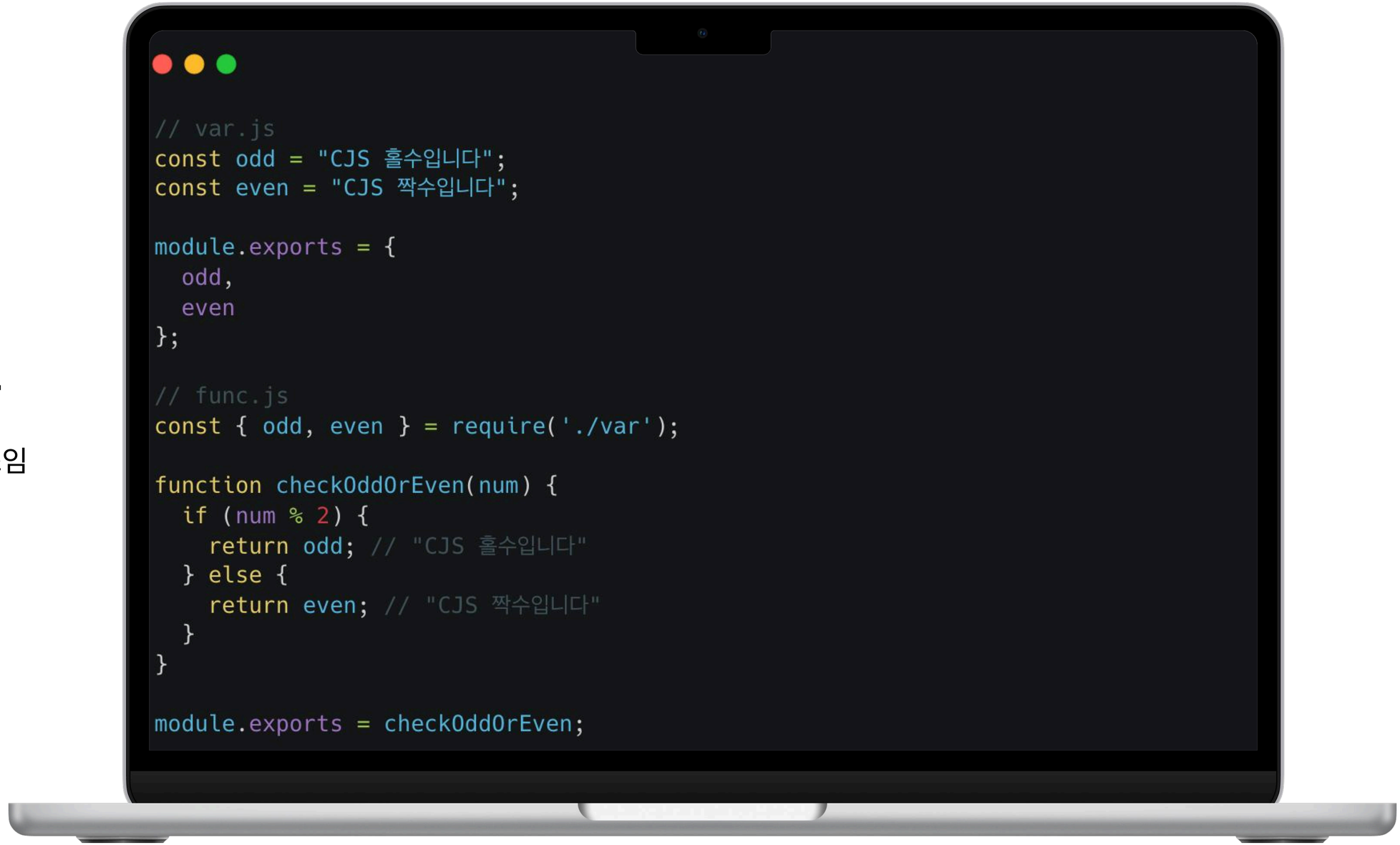
모듈로 만들기



▲ 그림 3-2 모듈과 프로그램

CommonJS 모듈

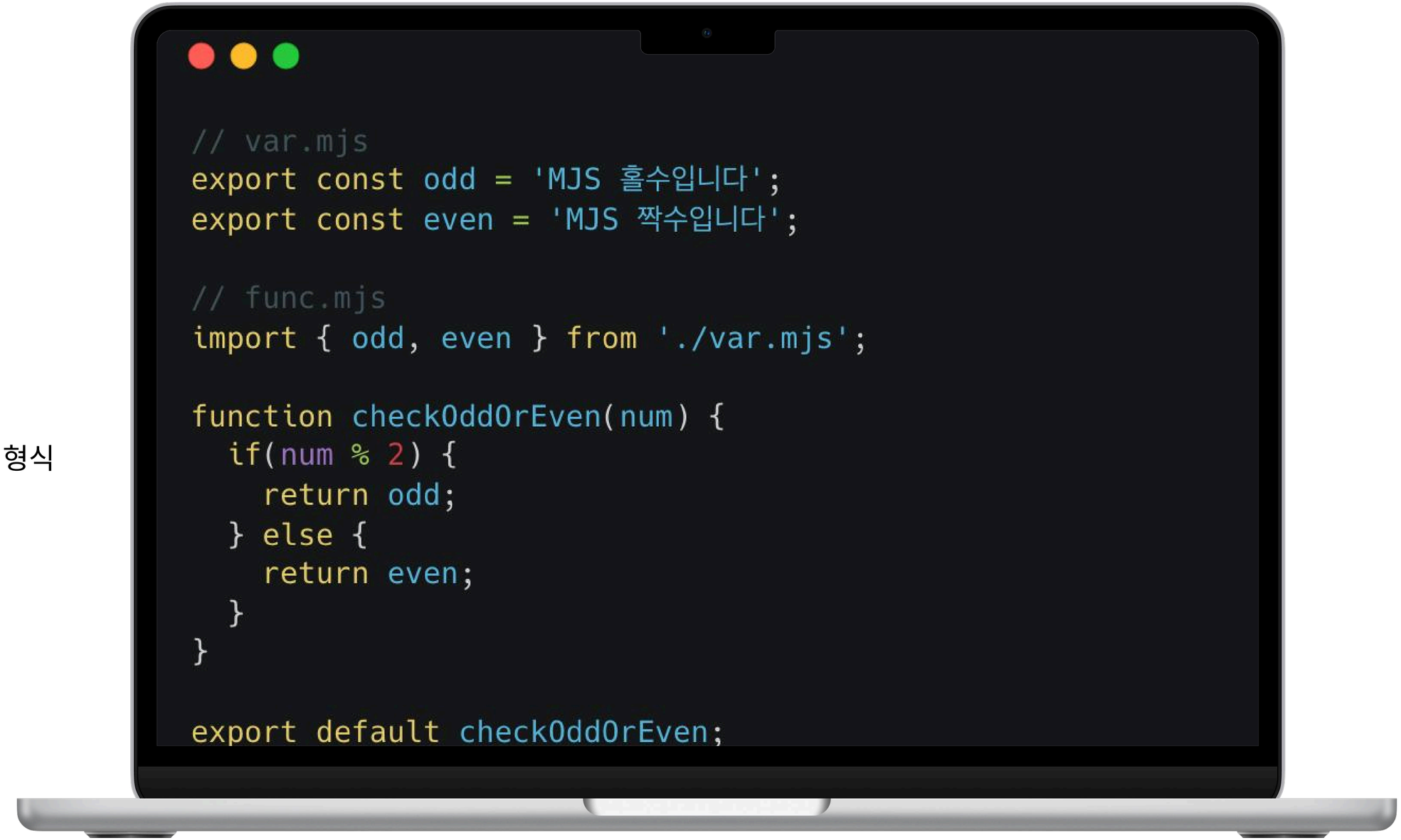
노드 생태계에서 가장 널리 쓰임



```
// var.js
const odd = "CJS 홀수입니다";
const even = "CJS 짝수입니다";

module.exports = {
  odd,
  even
};

// func.js
const { odd, even } = require('./var');

function checkOddOrEven(num) {
  if (num % 2) {
    return odd; // "CJS 홀수입니다"
  } else {
    return even; // "CJS 짝수입니다"
  }
}

module.exports = checkOddOrEven;
```

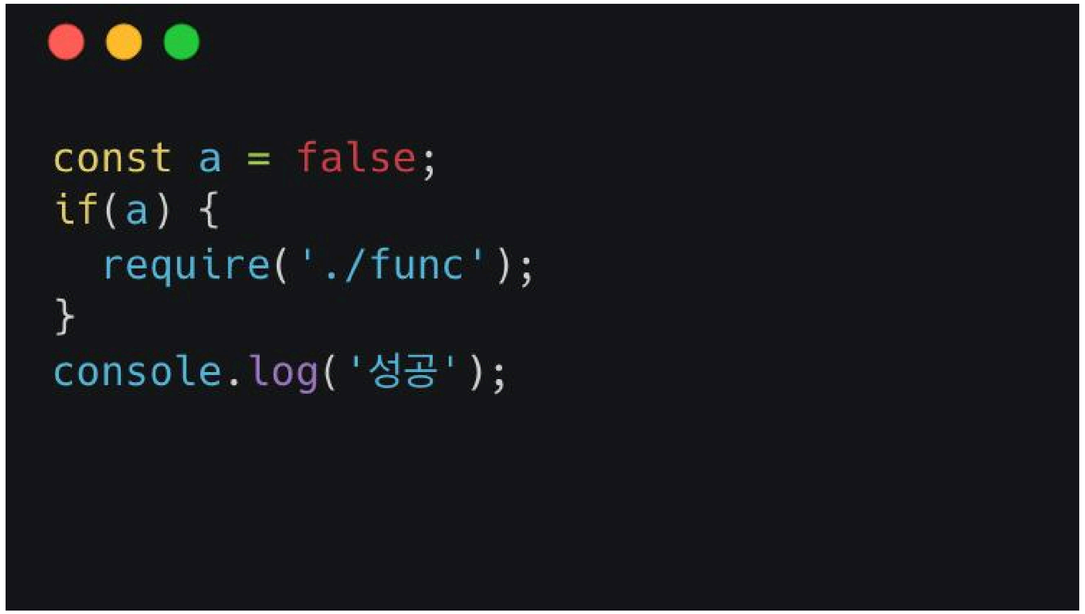
ES 모듈

공식적인 자바스크립트 모듈 형식



```mjs
// var.mjs
export const odd = 'MJS 홀수입니다';
export const even = 'MJS 짝수입니다';

// func.mjs
import { odd, even } from './var.mjs';

function checkOddOrEven(num) {
  if(num % 2) {
    return odd;
  } else {
    return even;
  }
}

export default checkOddOrEven;
```

Dynamic Import



```
const a = false;
if(a) {
  require('./func');
}
console.log('성공');
```

CommonJS

조건문에서 이렇게 require 을 이용하여 모듈을 임포트하려고 하는 것을 다이나믹 임포트라고 한다.

```
const a = true;
if(a) {
  import './func.mjs'; // 실패

  const m1 = await import('./func.mjs');
  console.log(m1); // 성공
}
```

ES Module

ES 모듈에서는 if 문 안에 import하는 것이 불가능하다. 이럴때는 import 함수를 사용해서 모듈을 동적으로 불러와야한다.

노드 내장 객체

global 객체

- 브라우저에서 window와 같은 전역 객체 (모든 파일에서 접근할 수 있다)
  - window.open 메서드를 그냥 open으로 호출할 수 있는 것처럼 global도 생략할 수 있다
  - CommonJS 모듈시스템에서 require도 global.require 에서 global이 생략된 것이다.
  - console 객체도 원래는 global.console 이다
- global 객체 내부에는 매우 많은 속성이 들어있음

```
> globalThis === global
true
> global.console
Object [console] {
  log: [Function: log],
  info: [Function: info],
  debug: [Function: debug],
  warn: [Function: warn],
```

console 객체

- **console 객체는 보통 디버깅을 위해서 사용**
  - **에러 발생 시 에러 내용을 콘솔에 표시하기 위해서도 사용**
  - **코드 실행 시간을 알아보려 할 때 사용**

```
console.log("일반 로그 메시지");
console.info("정보 메시지");
console.warn("경고 메시지");
console.error("에러 메시지");

const user = { name: "Siku", age: 24 };
console.table(user); // 객체를 표 형식으로 출력

console.time("timer");
for (let i = 0; i < 1000000; i++) {}
console.timeEnd("timer"); // 실행 시간 측정
```

timer 객체

- 타이머 기능을 제공하는 함수들이 포함되어있음

```
// 1. 일정 시간 후 한 번 실행
const timeout = setTimeout(() => {
  console.log("⏰ 1초 후 실행됨");
}, 1000);

// 2. 일정 간격으로 반복 실행
const interval = setInterval(() => {
  console.log("🔁 0.5초마다 실행");
}, 500);

// 3. 즉시(현재 이벤트 루프 이후) 실행
setImmediate(() => {
  console.log("⚡ 즉시 실행 (이벤트 루프 끝나면)");
});

// 4. 일정 시간 후에 반복 종료
setTimeout(() => {
  clearInterval(interval);
  console.log("🛑 반복 종료");
}, 2500);
```

process 객체

- 현재 실행되고 있는 노드 프로세스에 대한 정보를 담고있음

```
console.log("📂 실행 파일 경로:", process.argv[0]);      // node 실행 경로
console.log("📁 현재 스크립트:", process.argv[1]);        // 실행 중인 파일 경로
console.log("🌍 현재 작업 디렉토리:", process.cwd());
console.log("💻 플랫폼:", process.platform);
console.log("🧠 메모리 사용량:", process.memoryUsage().rss, "bytes");

// 환경 변수 접근
console.log("🔑 NODE_ENV:", process.env.NODE_ENV);

// 종료 이벤트
process.on("exit", (code) => {
  console.log(`프로세스가 종료됩니다. (코드: ${code})`);
});

// 프로세스 종료
setTimeout(() => {
  console.log("🛑 프로세스 종료 호출");
  process.exit(0);
}, 1000);
```

노드 내장 모듈

필요한 모듈에 대해서는 직접 찾아보는게 효율적

os 모듈

- 노드는 os 모듈에 정보가 담겨 있어 정보를 가져올 수 있다.

```
// os_module_examples.js
const os = require('os');

// 1️⃣ 운영체제 정보 확인: os.type(), os.platform(), os.release()
console.log('OS Type:', os.type());           // 예: 'Linux', 'Darwin', 'Windows_NT'
console.log('Platform:', os.platform());      // 예: 'linux', 'win32', 'darwin'
console.log('Release:', os.release());        // 예: '5.15.0-1051-azure'

// 2️⃣ CPU 정보 확인: os.cpus()
const cpus = os.cpus();
console.log('CPU Model:', cpus[0].model);
console.log('Core Count:', cpus.length);

// 3️⃣ 메모리 사용량 확인: os.totalmem(), os.freemem()
console.log('Total Memory (MB):', (os.totalmem() / 1024 / 1024).toFixed(2));
console.log('Free Memory (MB):', (os.freemem() / 1024 / 1024).toFixed(2));

// 4️⃣ 네트워크 인터페이스 확인: os.networkInterfaces()
console.log(os.networkInterfaces());

// 5️⃣ 현재 사용자 정보 확인: os.userInfo()
const user = os.userInfo();
console.log('Username:', user.username);
console.log('Home Directory:', user.homedir);
```

path 모듈

- 폴더와 파일의 경로를 쉽게 조작하도록 도와주는 모듈이다.

```
// path_module_examples.js
const path = require('path');

// 1️⃣ path.join() – 경로를 OS에 맞게 안전하게 합치기
const fullPath = path.join('user', 'local', 'bin', 'node');
console.log('join →', fullPath);
// Windows: 'user\\local\\bin\\node'
// Linux/Mac: 'user/local/bin/node'

// 2️⃣ path.resolve() – 절대 경로 생성
const absPath = path.resolve('src', 'app.js');
console.log('resolve →', absPath);
// 현재 작업 디렉터리 기준 절대 경로 (예: /home/user/project/src/app.js)

// 3️⃣ path.basename(), path.extname(), path.dirname() – 파일 경로 분석
const filePath = '/home/user/project/index.html';
console.log('basename →', path.basename(filePath));  // 'index.html'
console.log('extname  →', path.extname(filePath));   // '.html'
console.log('dirname  →', path.dirname(filePath));   // '/home/user/project'
```

QnA

자유롭게 질문해주세요.