



# panicThreshold

panicThreshold 옵션은 **React** 컴파일러가 컴파일 중 오류를 처리하는 방식을 제어합니다.

```
{
  panicThreshold: 'none' // Recommended
}
```

- 레퍼런스

- [panicThreshold](#)

- 사용법

- [프로덕션 설정 \(권장\)](#)
- [개발 중 디버깅](#)

## 레퍼런스

### panicThreshold

컴파일 오류가 빌드를 실패시켜야 하는지 아니면 최적화를 건너뛰어야 하는지를 결정합니다.

### 타입

```
'none' | 'critical_errors' | 'all_errors'
```

### 기본값

'none'

## 옵션

- **'none'** (기본값, 권장): 컴파일할 수 없는 컴포넌트를 건너뛰고 빌드를 계속 진행합니다.
- **'critical\_errors'**: 치명적인 컴파일러 오류에서만 빌드를 실패시킵니다.
- **'all\_errors'**: 모든 컴파일러 진단에서 빌드를 실패시킵니다.

## 주의 사항

- 프로덕션 빌드에서는 항상 **'none'** 을 사용해야 합니다.
- 빌드 실패는 애플리케이션이 빌드되지 않도록 합니다.
- 컴파일러는 **'none'** 을 사용하면 문제가 있는 코드를 자동으로 감지하고 건너뛵니다.
- 더 높은 임계값은 개발 중 디버깅에만 유용합니다.

## 사용법

### 프로덕션 설정 (권장)

프로덕션 빌드에서는 항상 **'none'** 을 사용하세요. 이것이 기본값입니다.

```
{  
  panicThreshold: 'none'  
}
```

이렇게 하면 다음을 보장합니다.

- 컴파일러 문제로 인해 빌드가 실패하지 않습니다.
- 최적화할 수 없는 컴포넌트도 정상적으로 실행됩니다.
- 최대한 많은 컴포넌트가 최적화됩니다.
- 안정적인 프로덕션 배포가 가능합니다.

### 개발 중 디버깅

문제를 찾기 위해 일시적으로 더 엄격한 임계값을 사용합니다.

```
const isDevelopment = process.env.NODE_ENV === 'development';

{
  panicThreshold: isDevelopment ? 'critical_errors' : 'none',
  logger: {
    logEvent(filename, event) {
      if (isDevelopment && event.kind === 'CompileError') {
        // ...
      }
    }
  }
}
```

< [이전](#)  
[logger](#)

[다음](#) >  
[target](#)

## Meta Open Source

Copyright © Meta Platforms, Inc

uwu?

## React 학습하기

[빠르게 시작하기](#)

[설치하기](#)

[UI 표현하기](#)

[상호작용성 더하기](#)

[State 관리하기](#)

[탈출구](#)

## API 참고서

[React APIs](#)

[React DOM APIs](#)

## 커뮤니티

[행동 강령](#)

[팀 소개](#)

## 더 보기

[블로그](#)

[React Native](#)

문서 기여자

개인 정보 보호

감사의 말

약관

