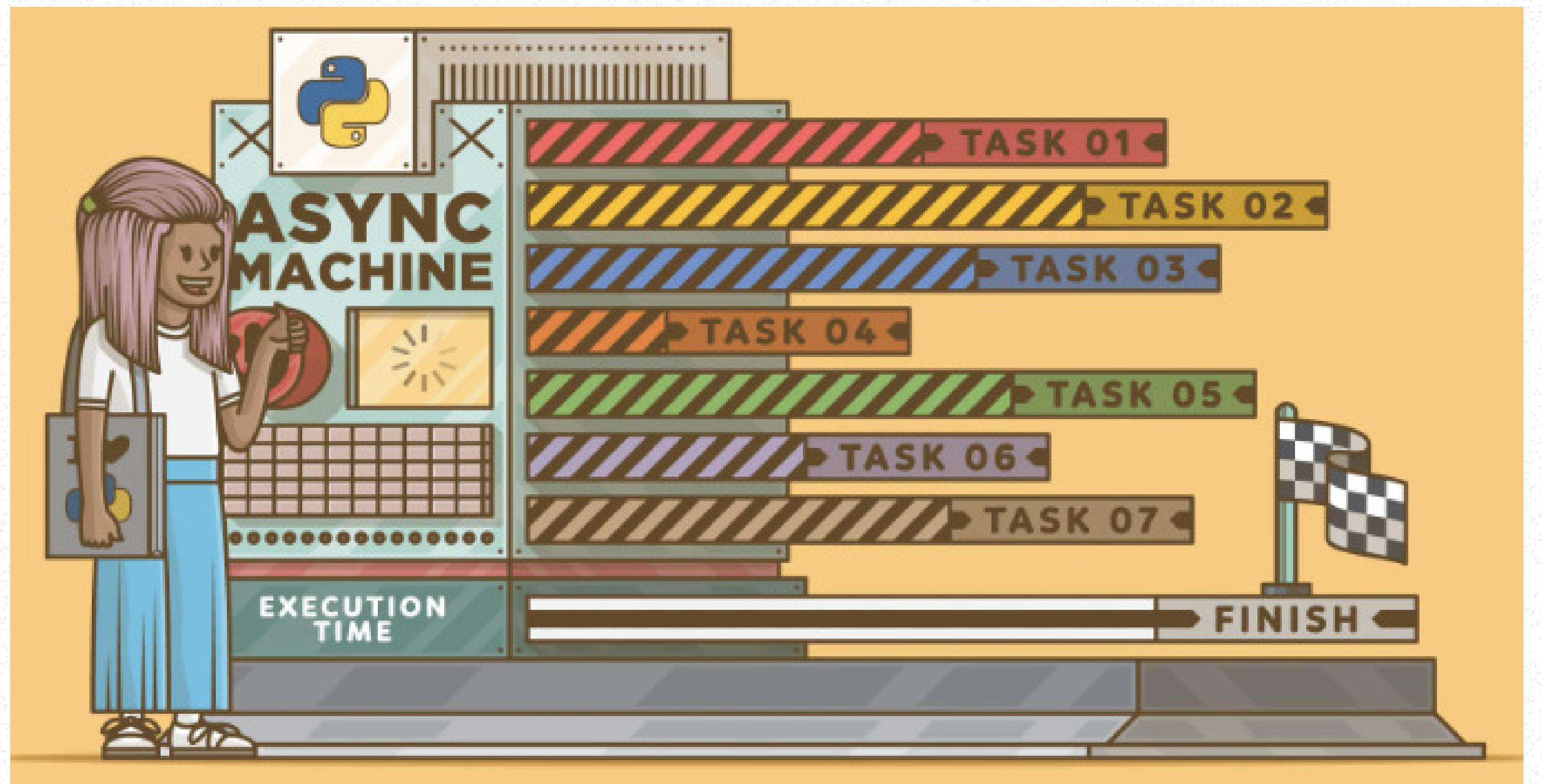
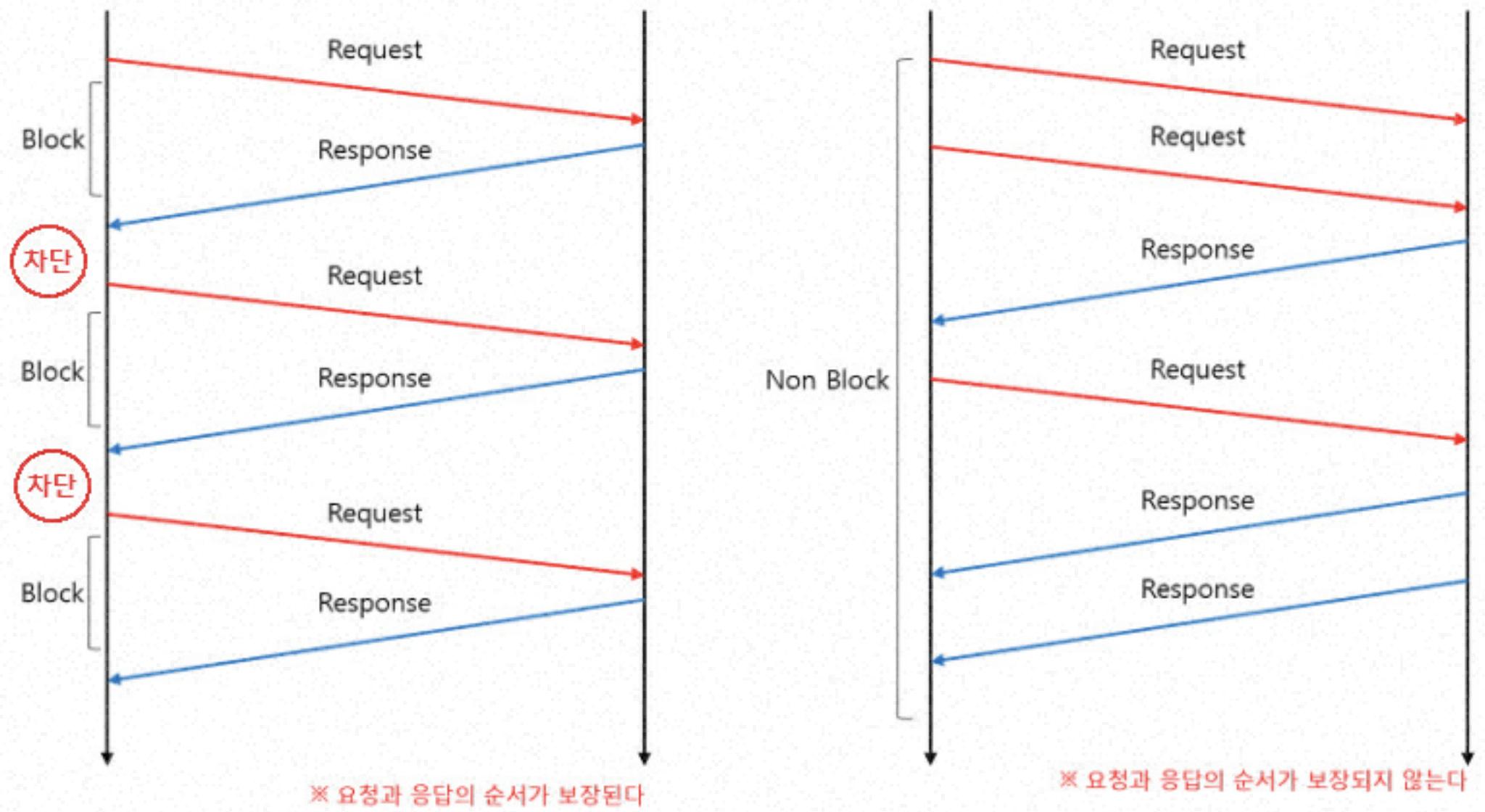


동기 / 비동기 블로킹 / 논 블로킹



여민수

□ 동기와 비동기 vs 블로킹과 논블로킹



동기 / 비동기

요청한 작업에 대해 작업을 순차적으로 수행할지 아닌지

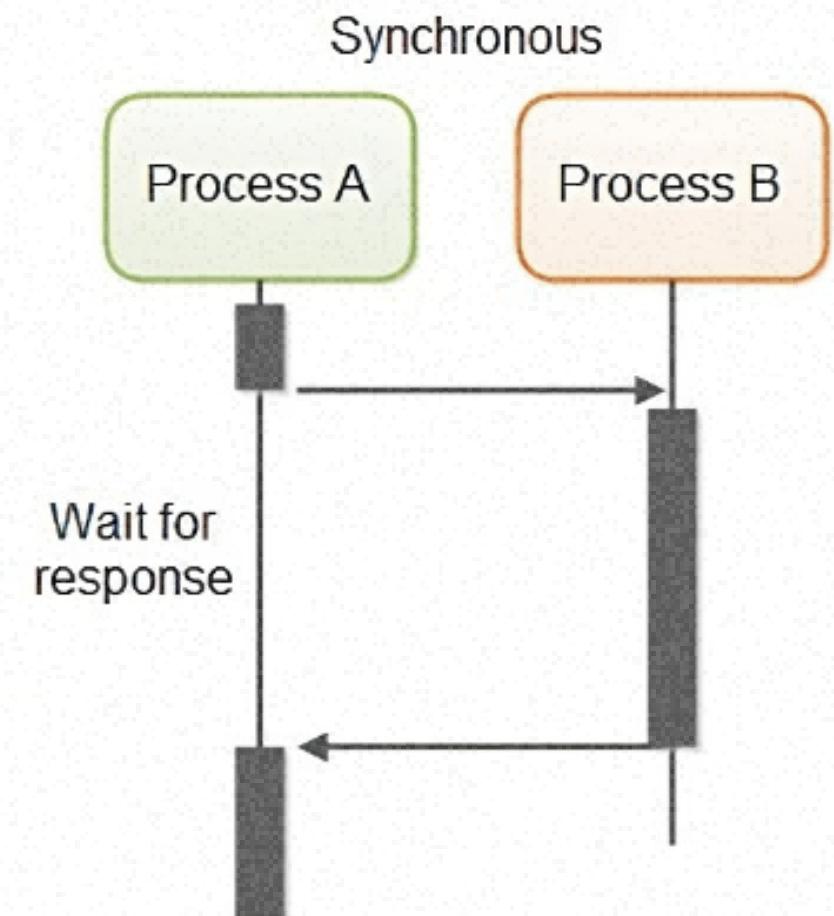
블로킹 / 논블로킹

현재 작업이 차단되느냐 아니냐

□ 동기(Synchronous) / 비동기(Asynchronous)

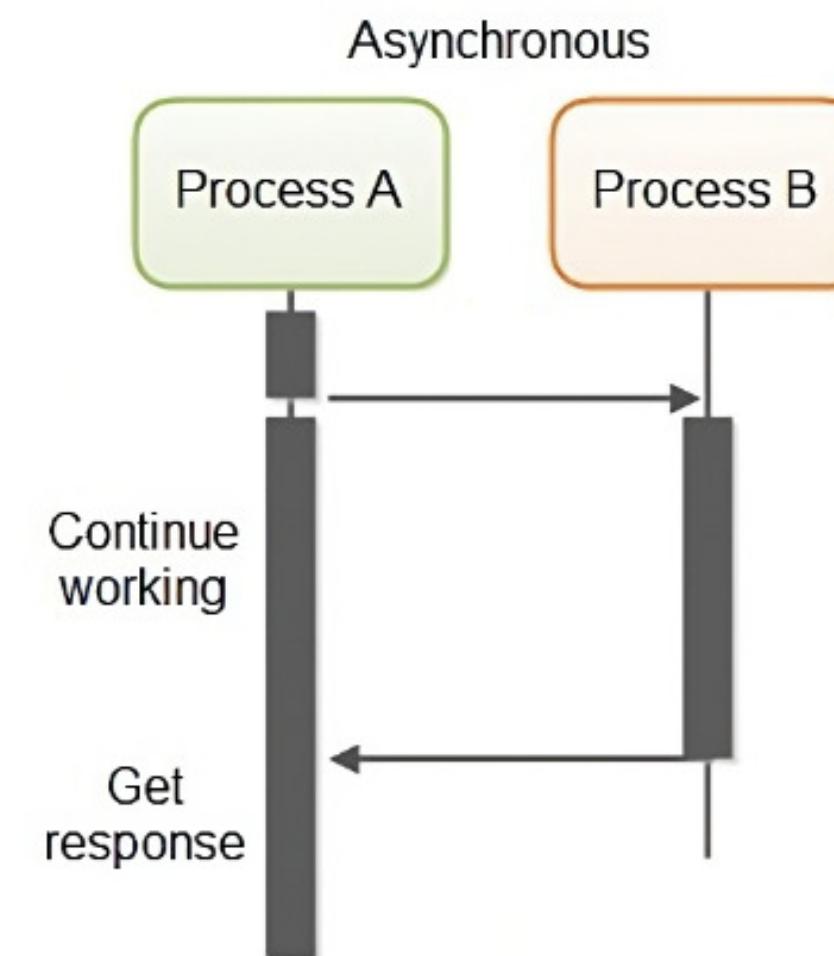
동기(Synchronous)

요청한 작업에 대해 완료 여부를 따져 순차대로 처리하는 것을 의미



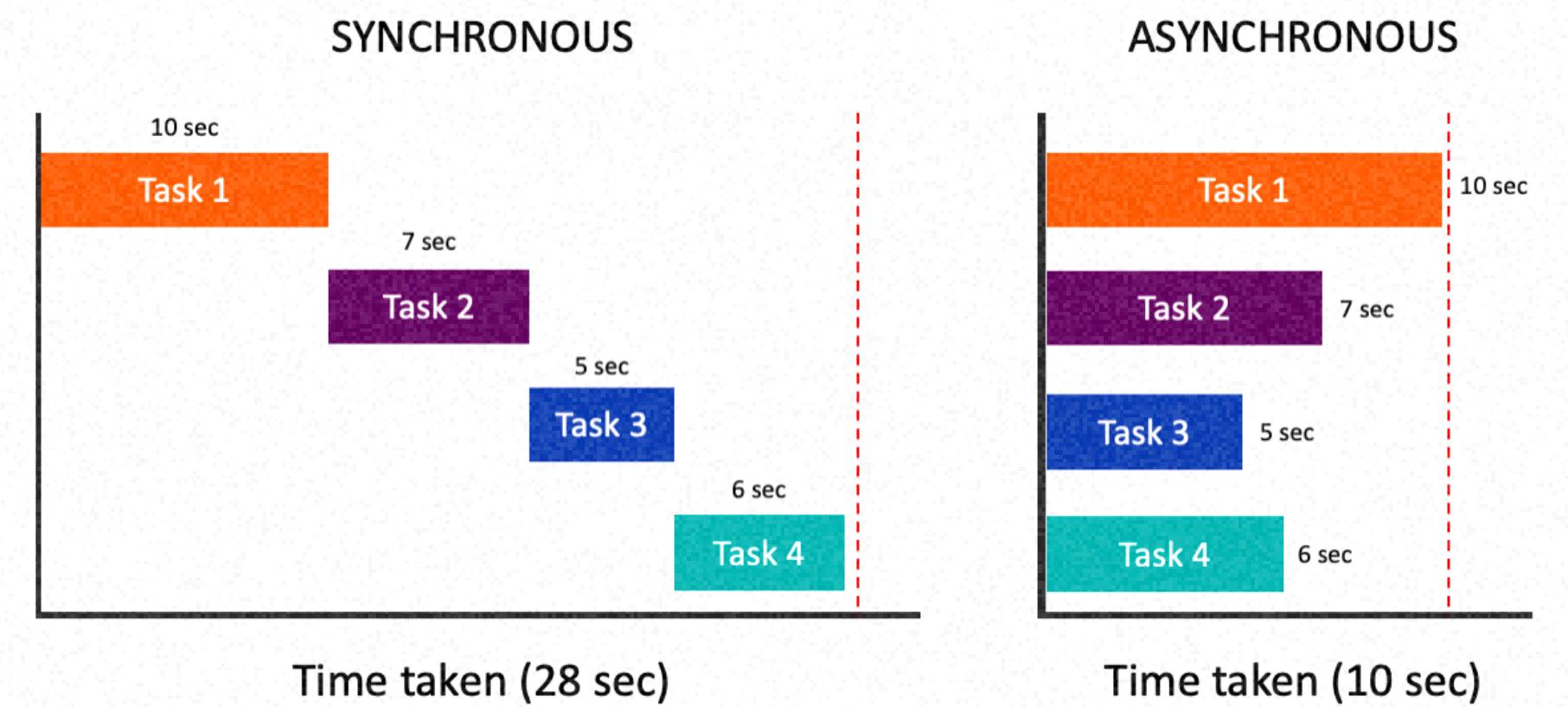
비동기(Asynchronous)

요청한 작업에 대해 완료 여부를 따지지 않음
→ 자신의 다음 작업을 그대로 수행



□ 비동기의 성능 이점

비동기 작업은 동기보다 성능이 우수
→ I/O작업과 같은 느린 작업이 발생할 때 기다리지 않고
동시에 처리가 가능하기 때문



□ 동시 처리가 가능하려면?



동시처리 = 두 개 이상의 작업이 동시에 실행되는 것을 의미

동시처리는 **멀티 스레드**, **멀티 프로세싱**과 같은 방식으로 구현 가능

⇒ **비동기 방식을 통해 대규모 트래픽에서도 안정적으로 동작하는 웹 애플리케이션을 구현할 수 있다!**

자바스크립트의 경우 비동기 작업 요청 시 Web API에 작업 인가 → 이벤트 루프에 의해 비동기 작업이 순차적으로 처리 단, 자바스크립트는 **싱글 스레드** 언어! (이벤트 루프에 의해 비동기 프로그래밍을 지원)

□ 그래서 둘이 뭐가 다른데?



비동기: 출력 순서 ⇒ 코드의 실행 순서와 관련된 개념

논블로킹: 여러 개의 업무를 한 번에 처리 ⇒ 병렬 실행과 관련된 개념

□ 비동기와 Non-blocking



시각에 따라 **비동기 코드**도 맞고, **논블로킹 코드**도 맞음

논블로킹 코드 시점: `setTimeout` 함수가 자신의 타이머 작업을 수행하기 위해
메인 함수를 블로킹하지 않고 백그라운드에서 실행 → 논블로킹 코드

비동기 코드 시점: 위 코드는 위에서부터 아래로 순차적으로 실행되지 않음
즉, 출력 순서와 정의된 코드 라인 순서가 맞지 않음 → 비동기 코드

□ 비동기와 Non-blocking



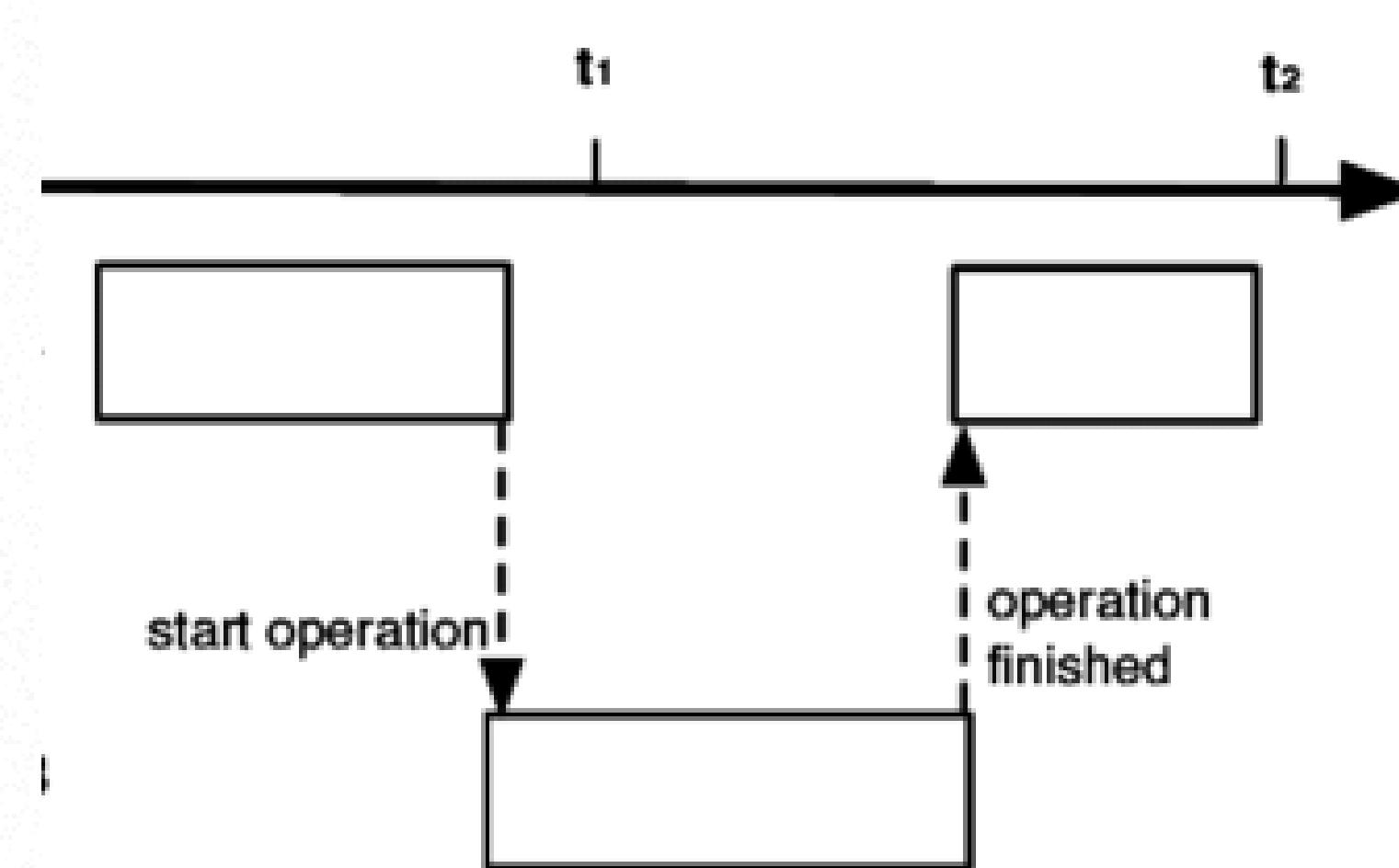
많은 개발자들이 비동기와 논블로킹의 차이를 혼동하고 있음

The screenshot shows a terminal window with three colored dots (red, yellow, green) at the top left. At the top right, it says "JAVASCRIPT". The code inside the terminal is:

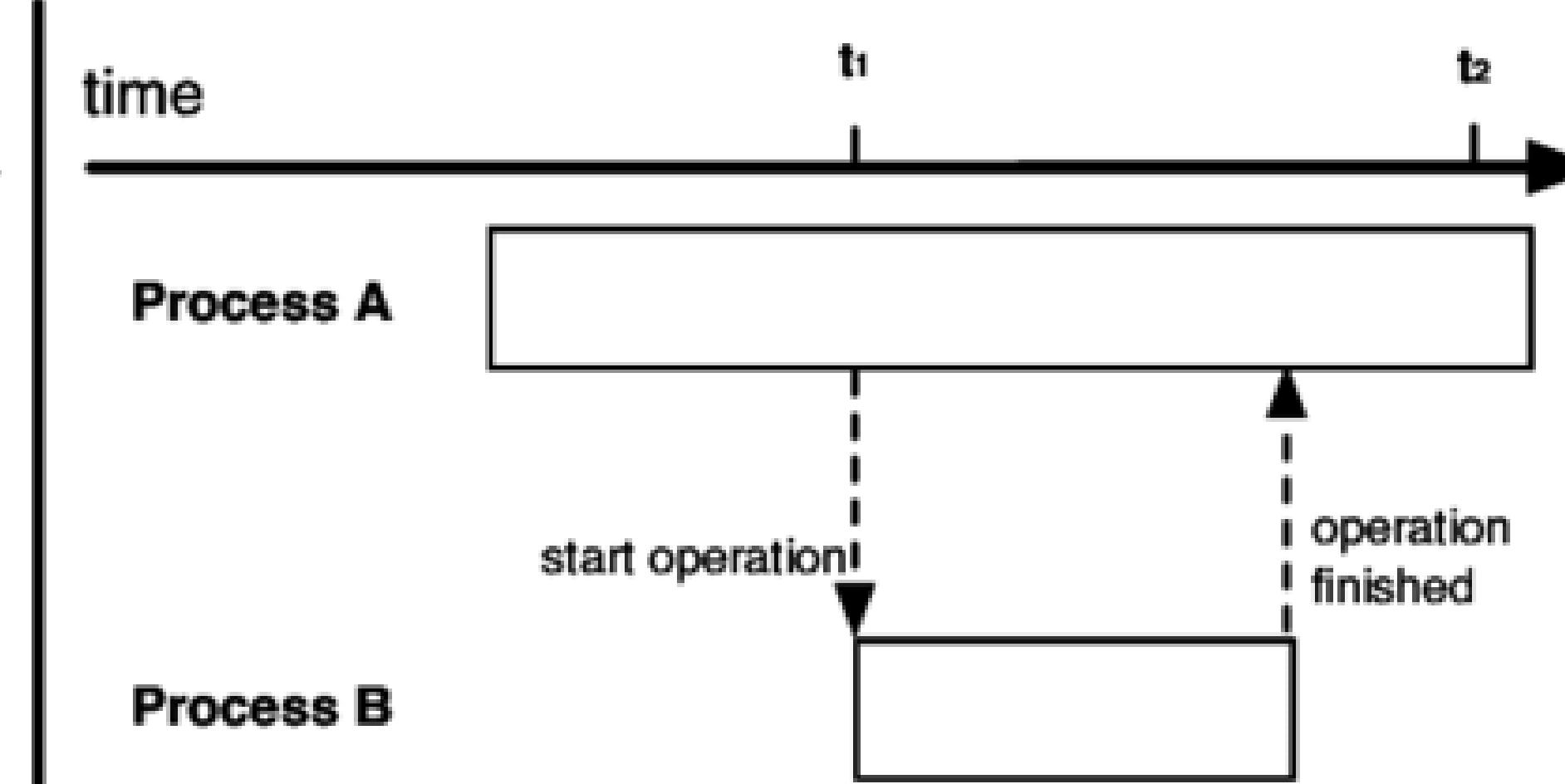
```
1 console.log("시작");
2
3 setTimeout(() => {
4     console.log("1초 후에 실행됩니다!");
5 }, 1000);
6
7 console.log("끝");
```

문제: 위 코드는 비동기 코드일까, 논블로킹 코드일까?

□ Blocking / Non-blocking



Blocking operations



Non-blocking operations

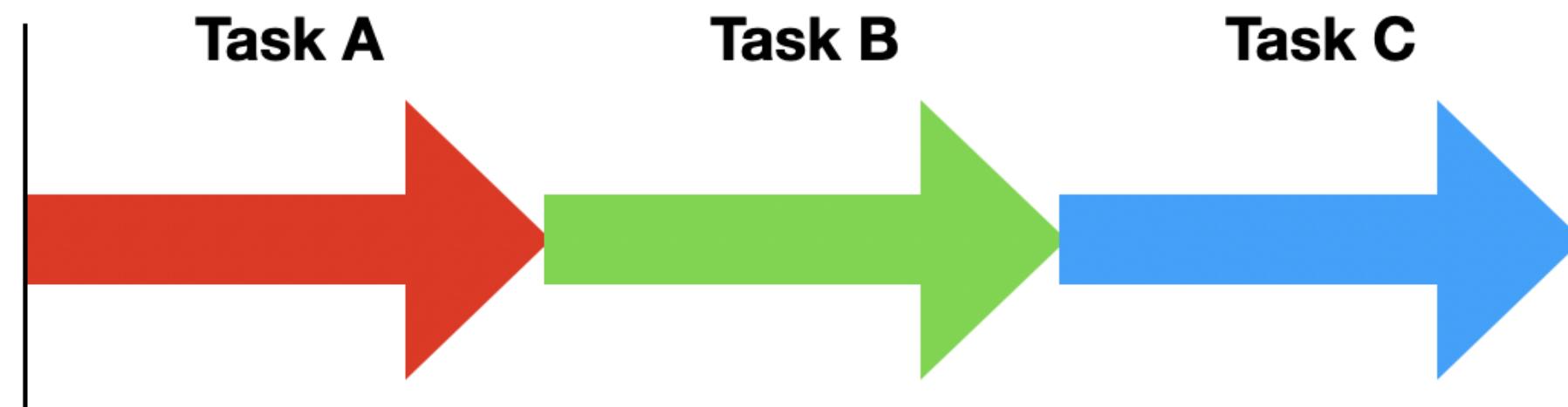
**Blocking과 Non-blocking은 전체적인
작업의 흐름 자체를 막나 안막나의 차이!**

ex) 파일을 읽을 때

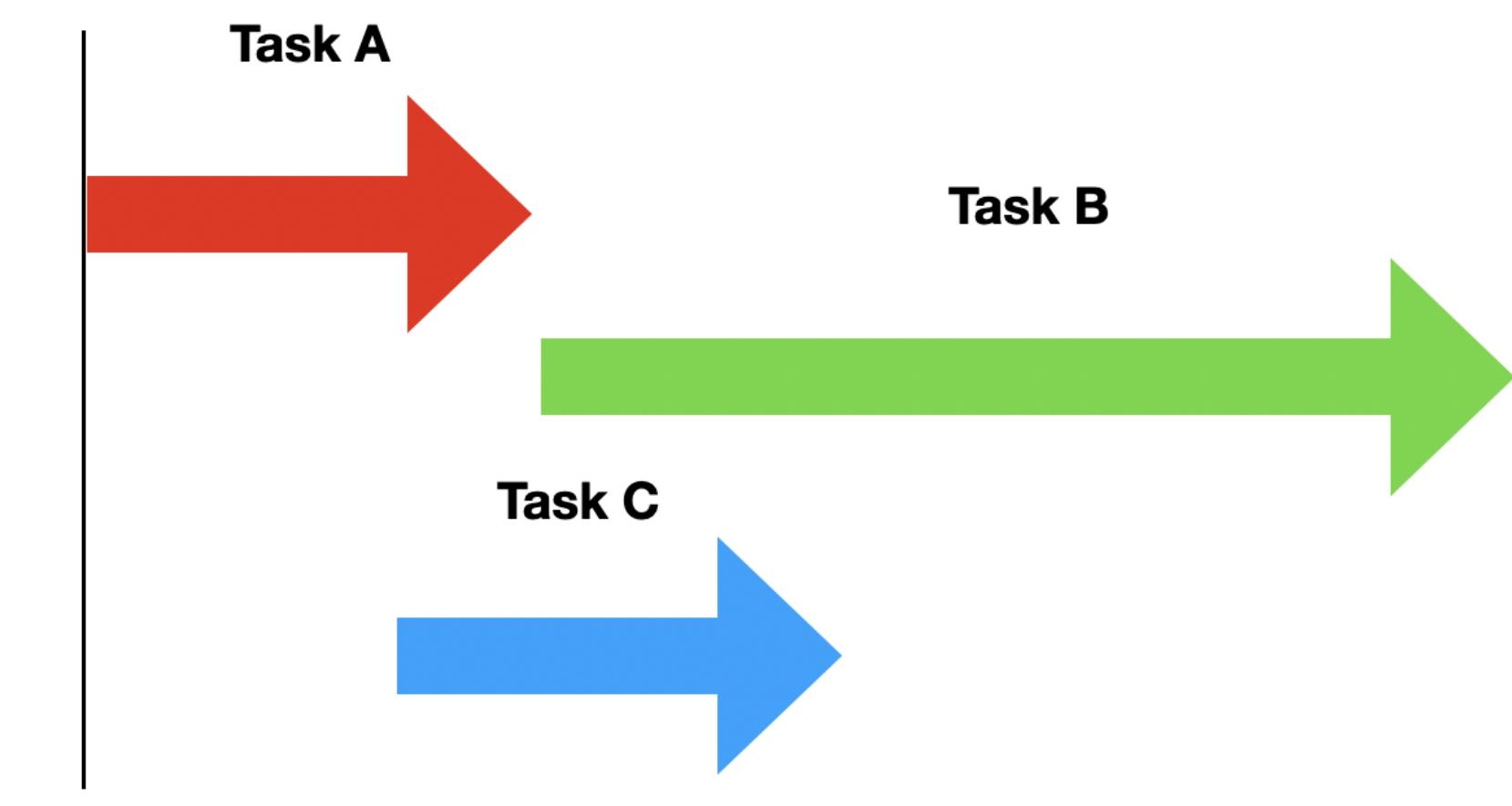
Blocking: 파일을 다 읽을 때 까지 대기

Non-blocking: 파일을 다 읽지 않아도 다른 작업 가능

□ 동기와 비동기의 차이점?



동기(Synchronous)



비동기(Asynchronous)

동기와 비동기는 작업 순서 처리 차이!

- 동기 작업은 요청한 작업에 대해 순서가 지켜지는 것을 의미
- 비동기 작업은 순서가 지켜지지 않을 수 있음을 의미