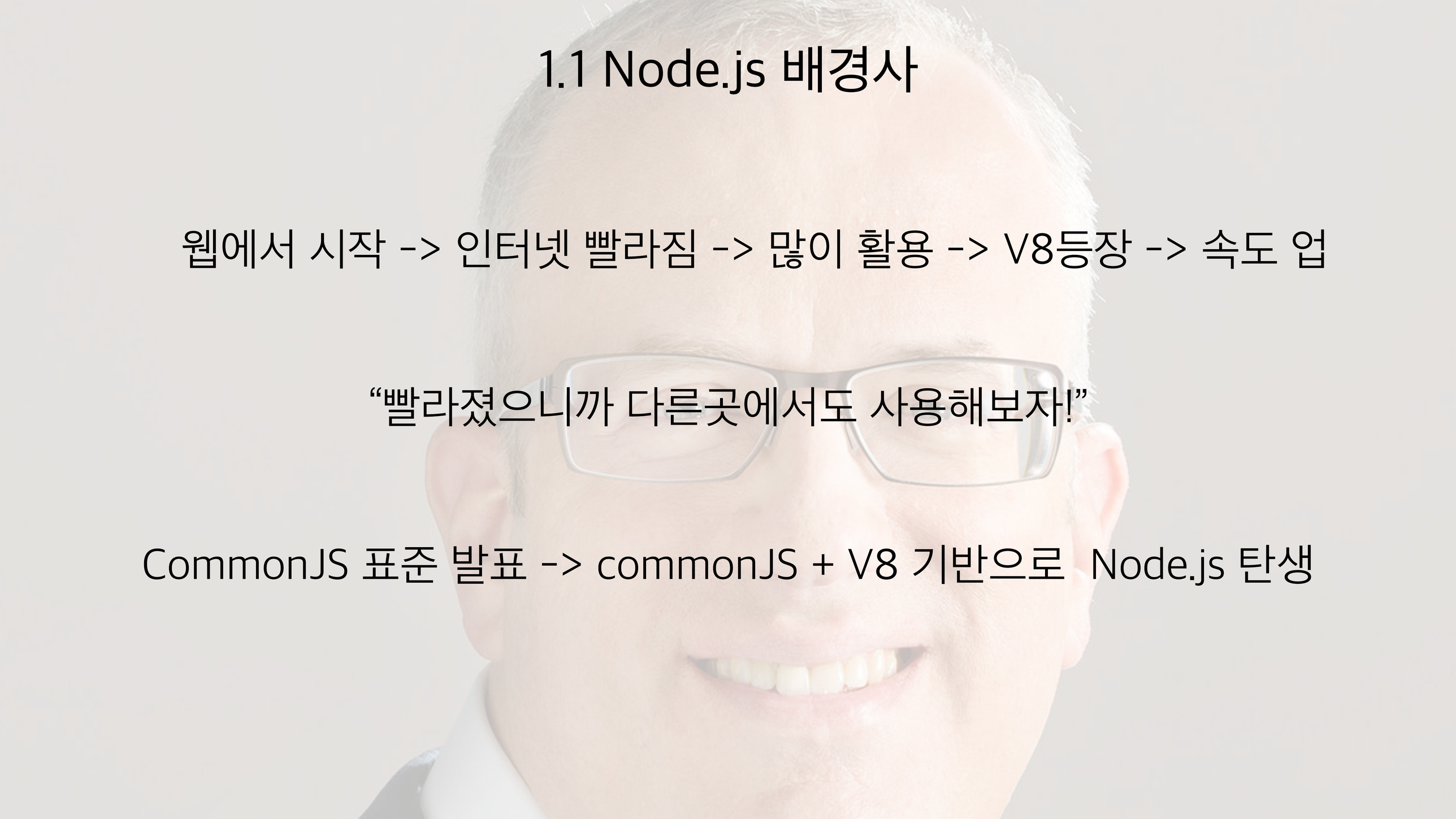
1. Node.js 개요

1.1 Node.js 배경사

웹에서 시작 -> 인터넷 빨라짐 -> 많이 활용 -> V8등장 -> 속도 업

“빨라졌으니까 다른곳에서도 사용해보자!”

CommonJS 표준 발표 -> commonJS + V8 기반으로 Node.js 탄생



CommonJS

CommonJS(http://www.commonjs.org/) 는 JavaScript를 브라우저에서뿐만 아니라, 서버사이드 애플리케이션이나 데스크톱 애플리케이션에서도 사용하려고 조직한 자발적 워킹 그룹이다. CommonJS의 'Common'은 JavaScript를 브라우저에서만 사용하는 언어가 아닌 일반적인 범용 언어로 사용할 수 있도록 하겠다는 의지를 나타내고 있는 것이라고 이해할 수 있다.

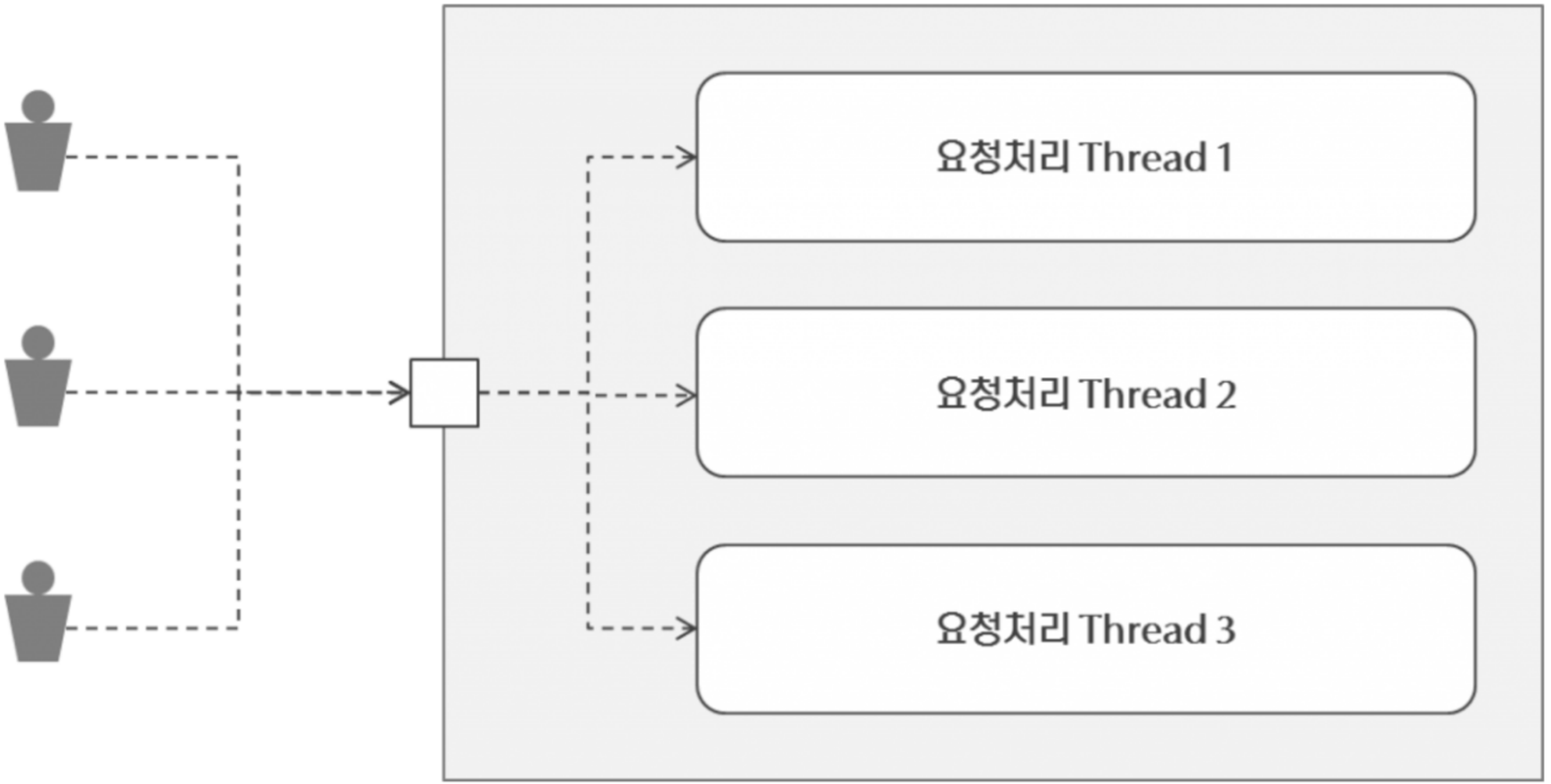
이 그룹은 JavaScript를 범용적으로 사용하기 위해 필요한 '명세(Specification)'를 만드는 일을 한다. 이 그룹에서 현재까지 정의한 명세로는 Module 명세 1.0, 1.1, 1.1.1 등이 있다. Node.js 모듈도 Module 명세 1.0을 따르고 있다.

대규모 네트워크 어플리케이션 개발

이벤트 기반 비동기 방식으로 네트워크 입출력

1.2 이벤트 기반 비동기 방식

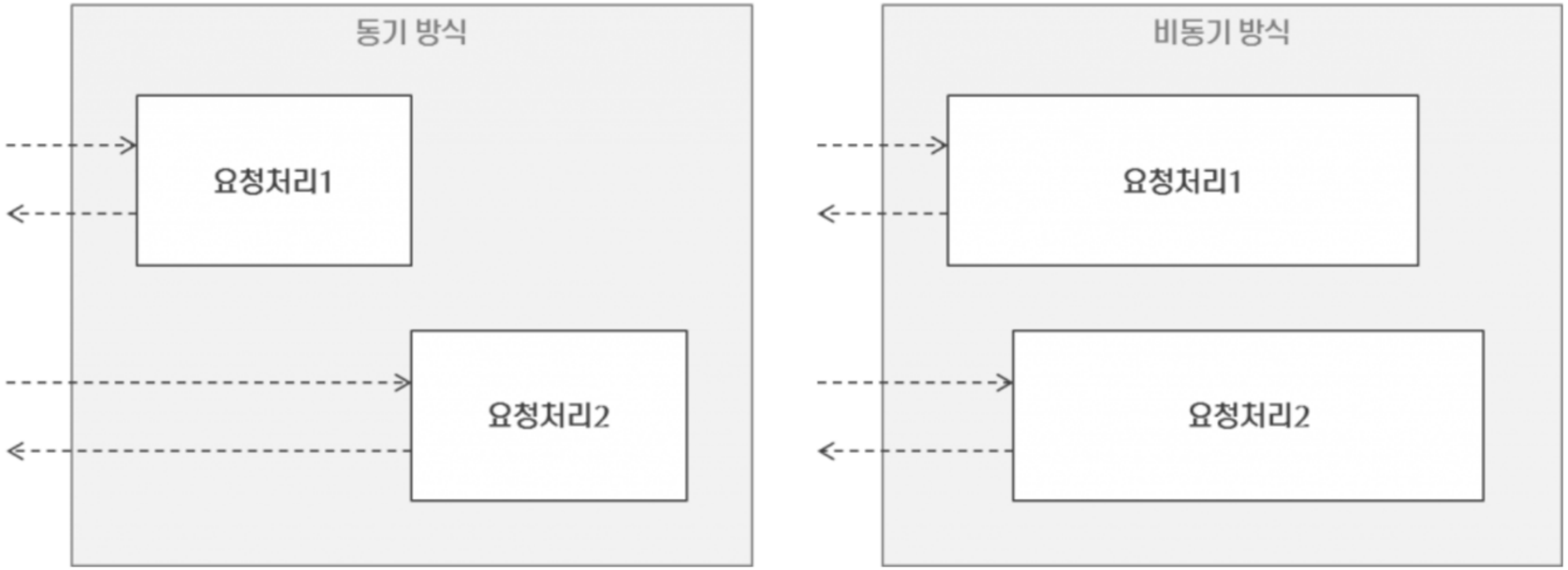
기존 네트워크 어플리케이션 - thread 기반 동기방식



Multi-Thread 방식

Multi-Thread 기반의 서버는 일반적으로 클라이언트의 요청마다 Thread를 발생시킵니다. 이 말은 동시 접속자 수가 많을 수록 Thread가 많이 발생한다는 의미이며 그만큼 메모리 자원도 많이 소모한다는 의미입니다. 그러나 서버의 자원은 제한되어 있으므로 일정 수 이상의 Thread는 발생시킬 수 없습니다.

Node.js - 이벤트 기반 비동기 처리방식



동기 방식과 비동기 방식

동기방식의 처리는 하나의 요청이 처리되는 동안 다른 요청이 처리되지 못하며 요청이 완료되어야만 다음 처리가 가능한 방식입니다. 동기 방식은 IO 처리를 Blocking 하는데 지금까지는 이 문제를 Thread로 처리했습니다.

이 문제를 비동기 방식으로 처리할 수도 있습니다. 비동기 방식은 하나의 요청 처리가 완료되기 전에 제어권을 다음 요청으로 넘깁니다. 따라서 IO 처리인 경우 Blocking 되지 않으며 다음 요청을 처리할 수 있는 것입니다.

장점

javascript 에서 오는 생산성의 향상

싱글 스레드 ->
대용량 네트워크 어플리케이션에 적합

단점

싱글스레드 -> 작업 시간이 길어지면 성능 저하

중첩된 콜백 -> 유지보수의 어려움

->인한 가독성 하락

Node.js 활용

commonJS http://d2.naver.com/helloworld/12864

nodejs 비동기 www.nextree.co.kr/p7292/

장단점 http://bcho.tistory.com/876