

# 배열 메서드와 선언형 프로그래밍(리스트 연산)

```
[🐄, 🍟, 🐔, 🌽].map(cook) => [🍔, 🍟, 🍗, 🍔]  
[🍔, 🍟, 🍗, 🍔].filter(isVegetarian) => [🍟, 🍔]  
[🍔, 🍟, 🍗, 🍔].reduce(eat) => 💩
```

우리가 다루는 데이터는 대부분 리스트다.

하나의 값도 원소가 하나인 목록으로 볼 수 있다.

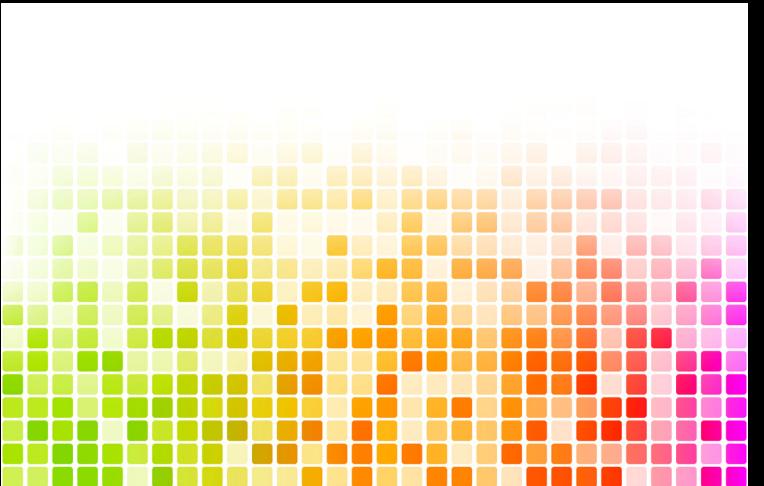
a → [a]

# 배열 메서드와 [ 🐂, 🥔, 🐔, 🌽 ]

## 선언형 프로그래밍(리스트 연산)

진짜!  
JS 자바스크립트

소리는 샘플 리스트  
이미지는 픽셀 리스트  
영상은 프레임 리스트

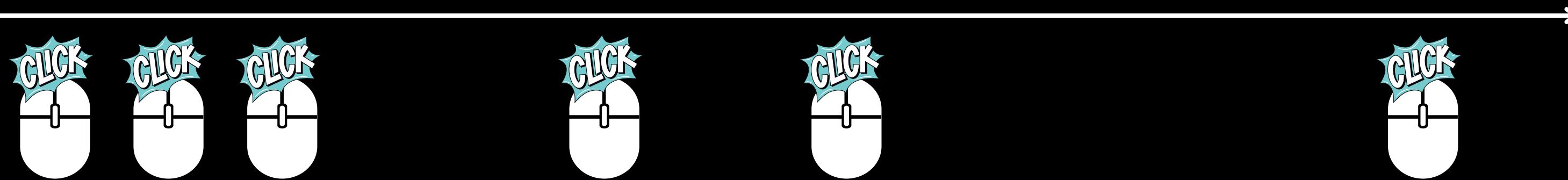


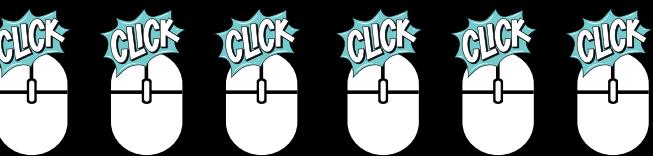
# 배열 메서드와 [ 🐂, 🥔, 🐔, 🌽 ]

## 선언형 프로그래밍(리스트 연산)

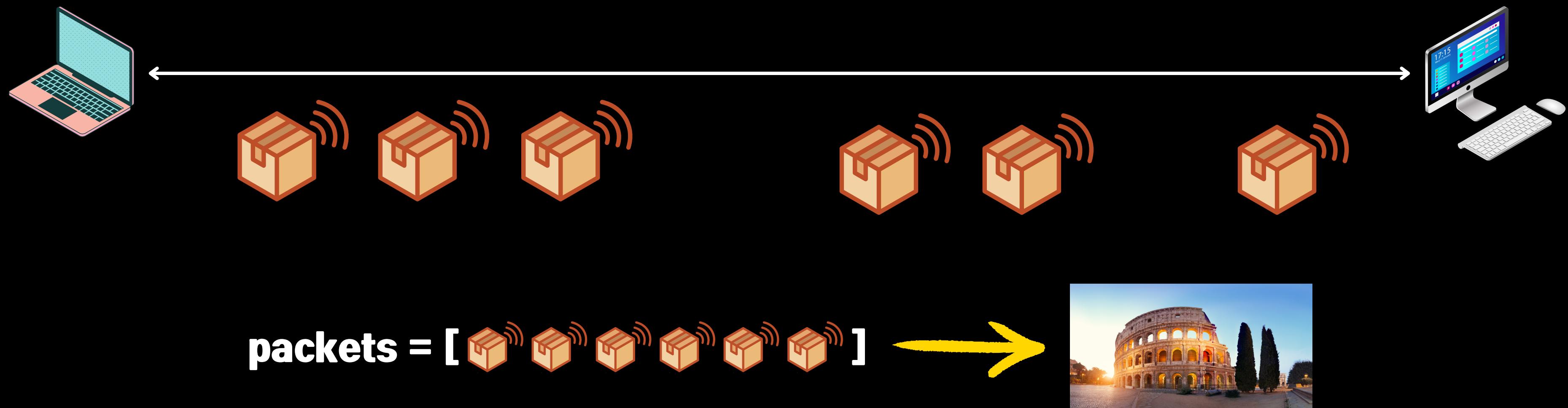
진짜!  
JS 자바스크립트

마우스 클릭 같은 비동기 이벤트도  
긴 시간 축에서 보면 결국 리스트다



clicks = [  ]

## 네트워크 패킷도 리스트다



결국 모든 것은 리스트다



이런 생각에 Observer 패턴을 적용한 것이 Rx다



## 선언형 프로그래밍이란?



Imperative Programming

```
const numbers = Array.of(1, 2, 3, 4, 5);
const result = [];
for (let i = 0; i < numbers.length; i++) {
  result.push(numbers[i] * 2);
}
console.log(result);
```



Declarative Programming

```
const result = Array.of(1, 2, 3, 4, 5)
  .map((n) => n * 2);
console.log(result);
```

명령형

“어떻게” - 알고리즘에 집중

선언형

“무엇을” - 목표에 집중

## 명령형 프로그래밍에서 매우 큰 이미지를 다룬다면?



Imperative Programming

```
const pixels = Array.from({ length: 4096 * 4096 * 3 }).fill(0);
const brightenImage = [];
for (let i = 0; i < pixels.length; i++) {
  brightenImage.push(pixels[i] + 100);
}
console.log(brightenImage);
```

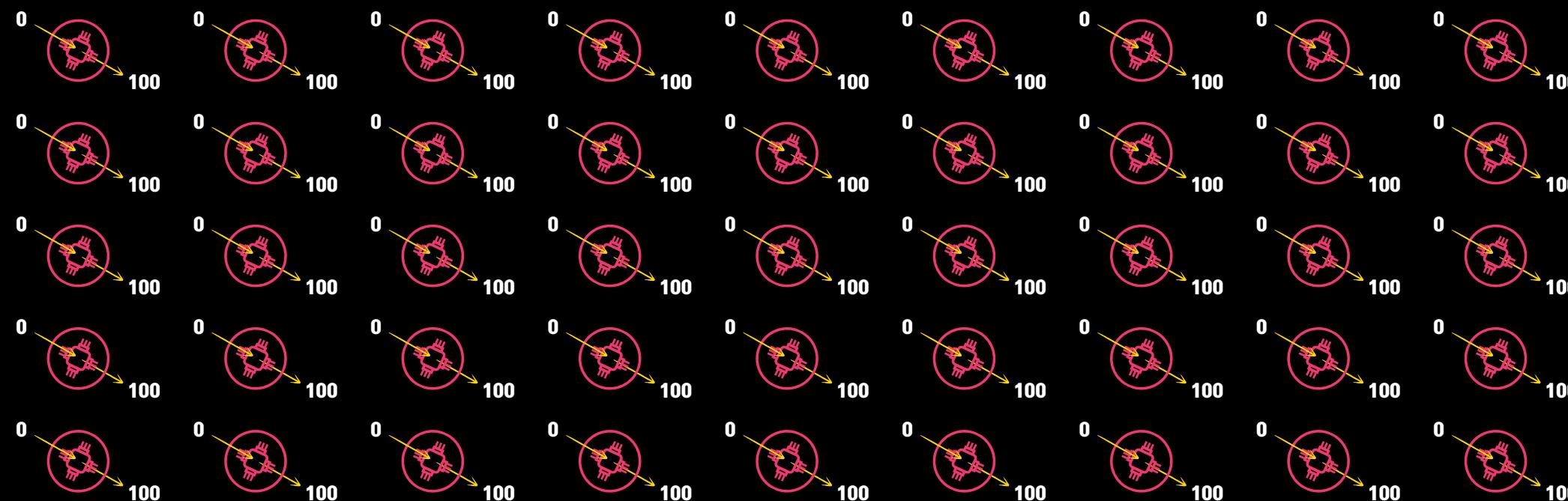
for 루프를 돌면서 순차적으로 픽셀을 접근하여 처리할 수 밖에 없다.

## 선언형 프로그래밍에서 GPU를 사용하는 map을 제공하면 어떨까?



Declarative Programming

```
const pixels = Array.from({ length: 4096 * 4096 * 3 }).fill(0);
const brightenImage = pixels.gpu_map((pixel) => pixel + 100);
console.log(brightenImage);
```



배열 메서드와 [  ,  ,  ,  ]  
선언형 프로그래밍(리스트 연산)

진짜!  
**JS** 자바스크립트

자바스크립트 코드 실습