

# GDSC **FRONTEND**

## Weekly Report 2

```
const org = filterByOrg ? study.lead_organization === filterByOrg : true  
const status = filterByStatus ? study.status === filterByStatus : true  
const matchStatus = (status === filterByStatus) ? true : false  
function filterStudies({ studies, filterByOrg, filterByStatus }) {  
  return studies.filter(study => {  
    return org && status && matchStatus  
  })  
}
```

# Study Members



장준  
성



고은  
서



박일  
상

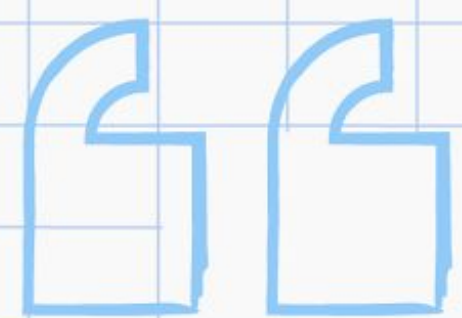


윤예  
원



조하  
연





# 주간 Study

# 이번주에는 무엇을 했을까요?



```
function filterStudies({ studies, filterByOrg = false, filterByStudy = false }) {
  return studies.filter(study => {
    if (filterByOrg) {
      return study.organization !== 'unlabeled organization'
    }
    if (filterByStudy) {
      return study.study !== 'unlabeled study'
    }
    return true
  })
}
```

# Study

모던 JavaScript  
튜토리얼



# Study

## 브라우저: 문서, 이벤트, 인터페이스

파트2에선 브라우저 내 페이지를 다루는 방법에 대해 학습합니다. 요소 추가, 요소의 사이즈와 위치를 조정하는 방법을 비롯하여 동적으로 인터페이스를 생성하는 방법, 인터페이스를 기반으로 사용자와 상호작용 하는 방법 등에 대해 학습할 예정입니다.

### 문서

- |  |                |                      |
|--|----------------|----------------------|
| 1.1 브라우저 환경과 다양한 명세서                     | 1.5 주요 노드 프로퍼티 | 1.9 요소 사이즈와 스크롤      |
| 1.2 DOM 트리                               | 1.6 속성과 프로퍼티   | 1.10 브라우저 창 사이즈와 스크롤 |
| 1.3 DOM 탐색하기                             | 1.7 문서 수정하기    | 1.11 좌표              |
| 1.4 getElement*, querySelector*로 요소 검색하기 | 1.8 스타일과 클래스   |                      |

### 이벤트 기초

- |                 |                |                  |
|-----------------|----------------|------------------|
| 2.1 브라우저 이벤트 소개 | 2.3 이벤트 위임     | 2.5 커스텀 이벤트 디스패치 |
| 2.2 버블링과 캡처링    | 2.4 브라우저 기본 동작 |                  |

### UI 이벤트

- |   |                       |                                 |
|---|-----------------------|---------------------------------|
| 3.1 마우스 이벤트   | 3.3 드래그 앤 드롭과 마우스 이벤트 | 3.5 Keyboard: keydown and keyup |
| 3.2 Moving the mouse: mouseover/out, mouseenter/leave | 3.4 Pointer events    | 3.6 Scrolling                   |

### 폼과 폼 조작

- |                 |  |                     |
|-----------------|--|---------------------|
| 4.1 폼 프로퍼티와 메서드 | 4.3 이벤트: change, input, cut, copy, paste | 4.4 submit 이벤트와 메서드 |
| 4.2 focus와 blur |  |                     |

### 문서와 리소스 로딩

- |  |                       |  |
|--|-----------------------|--|
| 5.1 DOMContentLoaded, load, beforeunload, unload 이벤트 | 5.2 defer, async 스크립트 | 5.3 Resource loading: onload and onerror |
|--|-----------------------|--|

# 모던 JavaScript 튜토리얼 + TO DO 리스트 만들기

## 구현 사항

1

**TO DO**  
추가

2

**TO DO**  
업데이트 및  
UI 즉각 반영

3

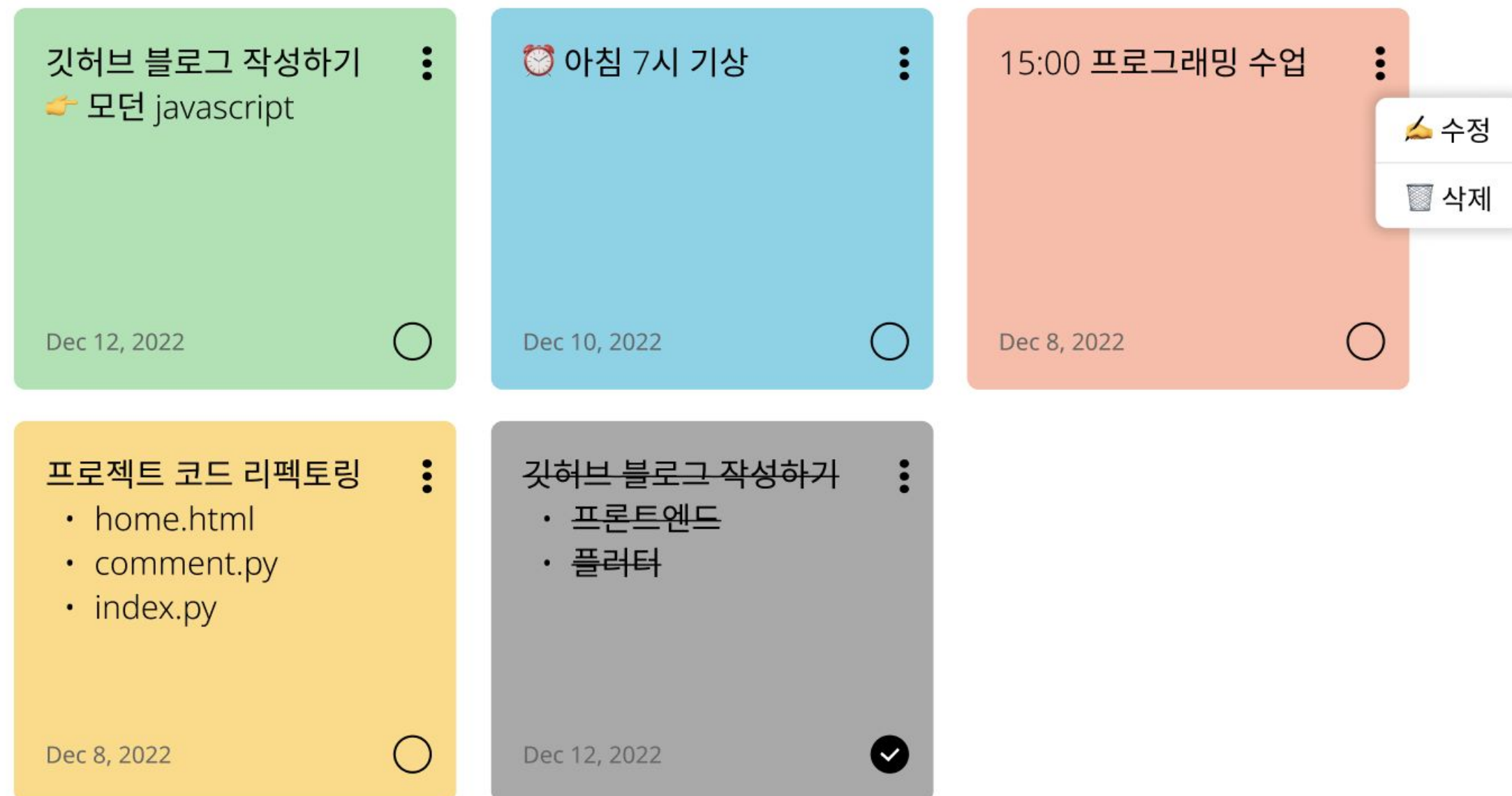
**TO DO**  
삭제

4

**Web  
Storage API**  
활용  
데이터 유지

# 구현 목표

## TODO LIST



Mockup of a TODO list application showing five items:

- Green card: 깃허브 블로그 작성하기 (모던 javascript), Dec 12, 2022
- Blue card: 아침 7시 기상, Dec 10, 2022
- Orange card: 15:00 프로그래밍 수업, Dec 8, 2022 (with edit/delete options)
- Yellow card: 프로젝트 코드 리팩토링 (home.html, comment.py, index.py), Dec 8, 2022
- Grey card: 깃허브 블로그 작성하기 (프론트엔드, 플러터), Dec 12, 2022 (checked)

1. TODO LIST 내용을 웹 스토리지에 저장
2. 포스트잇 느낌의 보드 형식 UI
3. 색상 테마 선택 시, 보드의 색상이 바뀌도록 (에브리타임 시간표처럼)

+ 여유가 있다면 컴포넌트 형식으로 제작



# Coding...

## TODO LIST

🔥 깃허브 블로그 업로드하기

🔥 프로그래밍 퀴즈 제출하기

🕒 아침 8:30 기상

🔥 모던자바스크립트 객체 단위  
노션에 정리



## 로컬 스토리지 데이터 얻기, 저장

```
function getNotes() {  
  return JSON.parse(localStorage.getItem("stickynotes-notes") || "[]");  
} // 로컬 스토리지 특정 key값 데이터 얻기  
  
function saveNotes(notes) {  
  localStorage.setItem("stickynotes-notes", JSON.stringify(notes));  
} // key-value 값을 JSON 문자열로 변환하여 로컬 스토리지에 저장
```

Application

- Manifest
- Service Workers
- Storage

Storage

- Local Storage
  - http://127.0.0.1:5500
- Session Storage
- IndexedDB
- Web SQL
- Cookies

Filter

Key	Value
stickynotes-notes	[{"id":45776,"content":"🔥 깃..."]

▼ [{"id": 45776, "content": "🔥 깃허브 블로그 업로드하기"}, {"id": 0: {id: 45776, "content": "🔥 깃허브 블로그 업로드하기"}, 1: {id: 69994, "content": "🔥 프로그래밍 퀴즈 제출하기"}, 2: {id: 34715, "content": "🕒 아침 8:30 기상"}, 3: {id: 51873, "content": "🔥 모던자바스크립트 객체 단위 노션에 정리"}]



# Coding...

## TODO LIST

📌 깃허브 블로그 업로드하기

📌 프로그래밍 퀴즈 제출하기

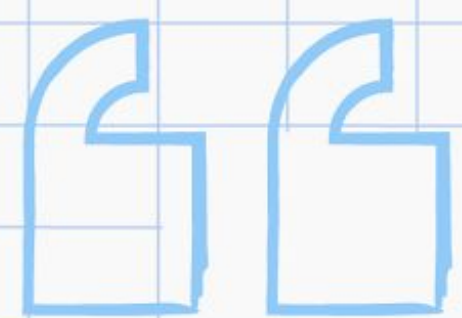
🕒 아침 8:30 기상

📌 모던자바스크립트 객체 단위  
노션에 정리



```
function addNote() {  
  const notes = getNotes();  
  const noteObject = {  
    id: Math.floor(Math.random() * 100000),  
    content: ""  
  };  
  
  const noteElement = createNoteElement(noteObject.id, noteObject.content);  
  notesContainer.insertBefore(noteElement, addNoteButton);  
  
  notes.push(noteObject);  
  saveNotes(notes);  
}  
  
function updateNote(id, newContent) {  
  const notes = getNotes();  
  const targetNote = notes.filter((note) => note.id == id)[0];  
  
  targetNote.content = newContent;  
  saveNotes(notes);  
}  
  
function deleteNote(id, element) {  
  const notes = getNotes().filter((note) => note.id != id);  
  
  saveNotes(notes);  
  notesContainer.removeChild(element);  
}
```





# 주간 Project

# 즐거운 프로젝트 회고록



```
function filterStudies({ studies, filterByOrg = false, filterByStudy = false }) {
  return studies.filter(study => {
    if (filterByOrg) {
      return study.organization !== 'unlabeled organization'
    }
    if (filterByStudy) {
      return study.study !== 'unlabeled study'
    }
    return true
  })
}
```



# Project

## TO DO 리스트 만들기

### 구현사항

1

**TO DO**  
추가

2

**TO DO**  
업데이트 및  
**UI** 즉각 반영

3

**TO DO**  
삭제

4

**Web**  
**Storage API**  
활용  
데이터 유지

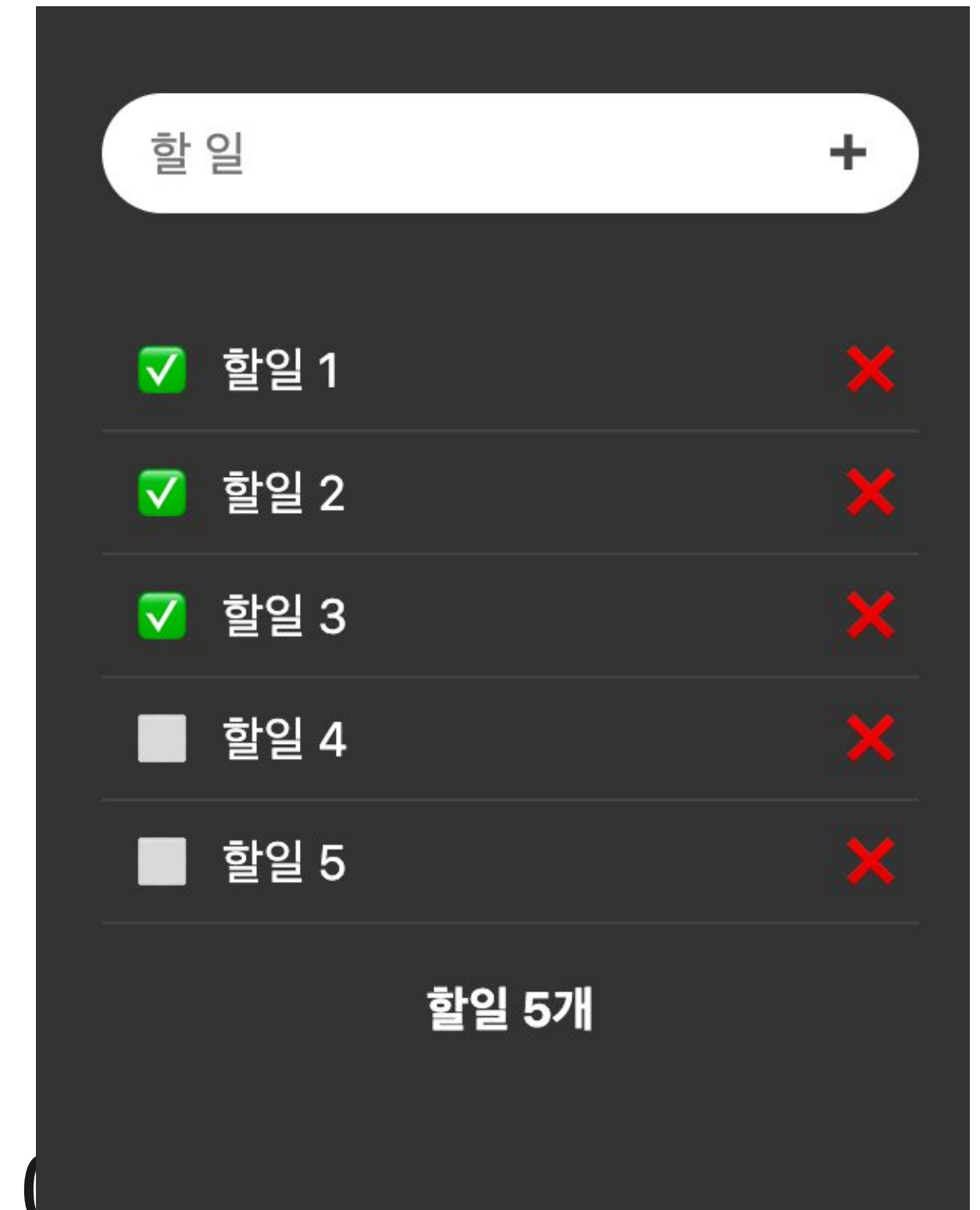
# 고민거리

1. Vanilla JS로 컴포넌트를 구현하는 방법

2. TO DO LIST의 내용을 웹 스토리지에 저장하는 방법

3. 사용자에게 편리한 UI

+ 추가적으로 디자인과 카테고리 기능을 가능한대로 구현할 예정





# Code Review



## 일상

### 이해가 되지 않는 부분

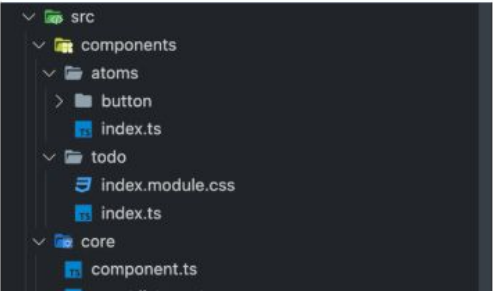
```
class EventListener {
  #target: Target
  // EventInfo[] = [] -> 이부분이 타입스크립트 문법인건지 처음 보는 문법인건지 모르겠어서 이해
  #eventInfoList = []
  //...
}
```

### 좋았던 점

#### Overall

- 각각의 함수에 모두 `@params` 를 통해서 주석을 남겨둬으로써 코드를 읽는 사람에게 부담을 줄이고 함수의 용도에 대해서 확실하게 알 수 있었던 점이 매우 좋습니다.  
→ 저도 앞으로 코드를 작성할 때 주석을 생활화하도록 노력하겠습니다.
- 타입스크립트를 사용하여 작성해주셨는데, 제가 타입스크립트를 잘 모르지만 더더욱 배워야겠다는 다짐을 하게된 계기가 되었습니다.

#### Structure



- 컴포넌트를 구현한 것의 의도에 잘 맞게 버튼을 하나의 작은 컴포넌트로 구현하여 사용한 점
- 로컬스토리지에 저장하는 로직을 따로 `utils` 로 디렉토리를 만들어 분리해 준 점



## 준성

### 좋았던 점

`addEvent` 메서드에 상위 컴포넌트에 이벤트 리스너를 부착한 다음 `isTarget` 으로 이벤트 propagating을 이용해서 실행한 점이 좋습니다.

```
this.component.addEventListener(eventType, (event) => {
  if (!isTarget(event.target)) return false
  callback(event)
})
```

`setEvent` 메서드 속에 `addEvent` 메서드를 사용하여 이벤트를 부착 로직을 분리한 것이 좋습니다!

```
setEvent() {
  const { toggleItem, updateItem, deleteItem } = this.props
  const { id } = this.props.item

  this.addEvent("click", ".toggle-btn", () => {
    toggleItem(id)
  })
  this.addEvent("click", ".modify-btn", () => {
    this.setState({ ...this.state, modify: true })
  })

  this.addEvent("click", ".update-btn", () => {
    updateItem(id, document.querySelector(`#update-todo-${id}`).value)
    this.setState({ ...this.state, modify: false })
  })

  this.addEvent("keypress", `#update-todo-${id}`, ({ key }) => {
    if (key !== "Enter") return false
    updateItem(id, document.querySelector(`#update-todo-${id}`).value)
    this.setState({ ...this.state, modify: false })
  })
}
```



# Project

구현사항 동일

Component + TODO 리스트

Reactive 컴포넌트를 구현하는 방법

Reactive란?

유저와 상호작용 가능한,  
값이 변경되면 바로 바뀌는

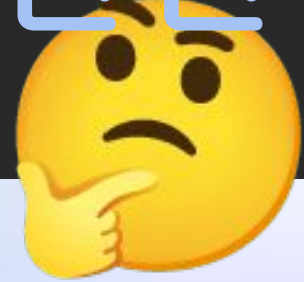


# Reactive Programming?



# Reactive?

잔잔한 수면으로 이해해보기



🌀 잔잔한 수면위

낙엽

= 대상

⚡ 물방울이 떨어짐

= 이벤트 발생

⚙ 파동에 따라

흔들림



# Reactive 🤔

## 엑셀로 구체화해보기

	A	B	C	D	E
1	Total cost	Cost 1	Cost 2		
2	100	50	50		
3					

Functional

Reactive

🌿 대상

= Total cost (A1)

⚡ 이벤트

= Cost1(B2),  
Cost2(C2)의 변화

⚙️ 선언된 동작 실시

= B2 + C2

# Reactive Code

엑셀 예제, 코드로  
구체화하기

	A	B	C
1	Total cost	Cost 1	Cost 2
2	100	50	50
3			

Reactive

```
const [cost1, setCost1] = signal(0)
```

```
const [cost2, setCost2] = signal(0)
```

```
const [TotalCost, setTotalCost] = signal(0)
```

```
track(() => {  
  setTotalCost(cost1() + cost2())  
})
```

```
track(() => {  
  console.log(TotalCost()) 0, 50, 100  
})
```

```
setCost1(50)
```

```
setCost2(50)
```



# Reactive Code

엑셀 예제, 코드로  
구체화하기

🌱 낙엽, 즉

	A	B	C
1	Total cost	Cost 1	Cost 2
2	100	50	50
3			

Reactive

```
const [cost1, setCost1] = signal(0)
```

```
const [cost2, setCost2] = signal(0)
```

```
const [TotalCost, setTotalCost] = signal(0)
```

```
track(() => {  
  setTotalCost(cost1() + cost2())  
})
```

```
track(() => {  
  console.log(TotalCost()) 0, 50, 100  
})
```

```
setCost1(50)
```

```
setCost2(50)
```



# Reactive Code

엑셀 예제, 코드로  
구체화하기

⚙ 선언된

A	B	C	D	E
Total cost	Cost 1	Cost 2		
100	50	50		

*fx* = B2+C2

```
const [cost1, setCost1] = signal(0)
const [cost2, setCost2] = signal(0)

const [TotalCost, setTotalCost] = signal(0)
```

```
track(() => {
  setTotalCost(cost1() + cost2())
})
```

```
track(() => {
  console.log(TotalCost()) 0, 50, 100
})
```

```
setCost1(50)
setCost2(50)
```



# Reactive Code

엑셀 예제, 코드로  
구체화하기

	A	B	C
1	Total cost	Cost 1	Cost 2
2	100	50	50
3			

Reactive

⚡ 이벤트

```
const [cost1, setCost1] = signal(0)
const [cost2, setCost2] = signal(0)

const [TotalCost, setTotalCost] = signal(0)

track(() => {
  setTotalCost(cost1() + cost2())
})

track(() => {
  console.log(TotalCost()) 0, 50, 100
})
```

```
setCost1(50)
setCost2(50)
```



# Reactive Code

엑셀 예제, 코드로  
구체화하기

	A
1	Total cost
2	100
3	

변화된 Total

```
const [cost1, setCost1] = signal(0)
const [cost2, setCost2] = signal(0)

const [TotalCost, setTotalCost] = signal(0)

track(() => {
  setTotalCost(cost1() + cost2())
})

track(() => {
  console.log(TotalCost())
})

setCost1(50)
setCost2(50)
```

0, 50, 100





# Reactive - 1

```
const [cost1, setCost1] = signal(0)
const [cost2, setCost2] = signal(0)

const [TotalCost, setTotalCost] = signal(0)

track(() => {
  setTotalCost(cost1() + cost2())
})

track(() => {
  console.log(TotalCost()) 0, 50, 100
})

setCost1(50)
setCost2(50)
```

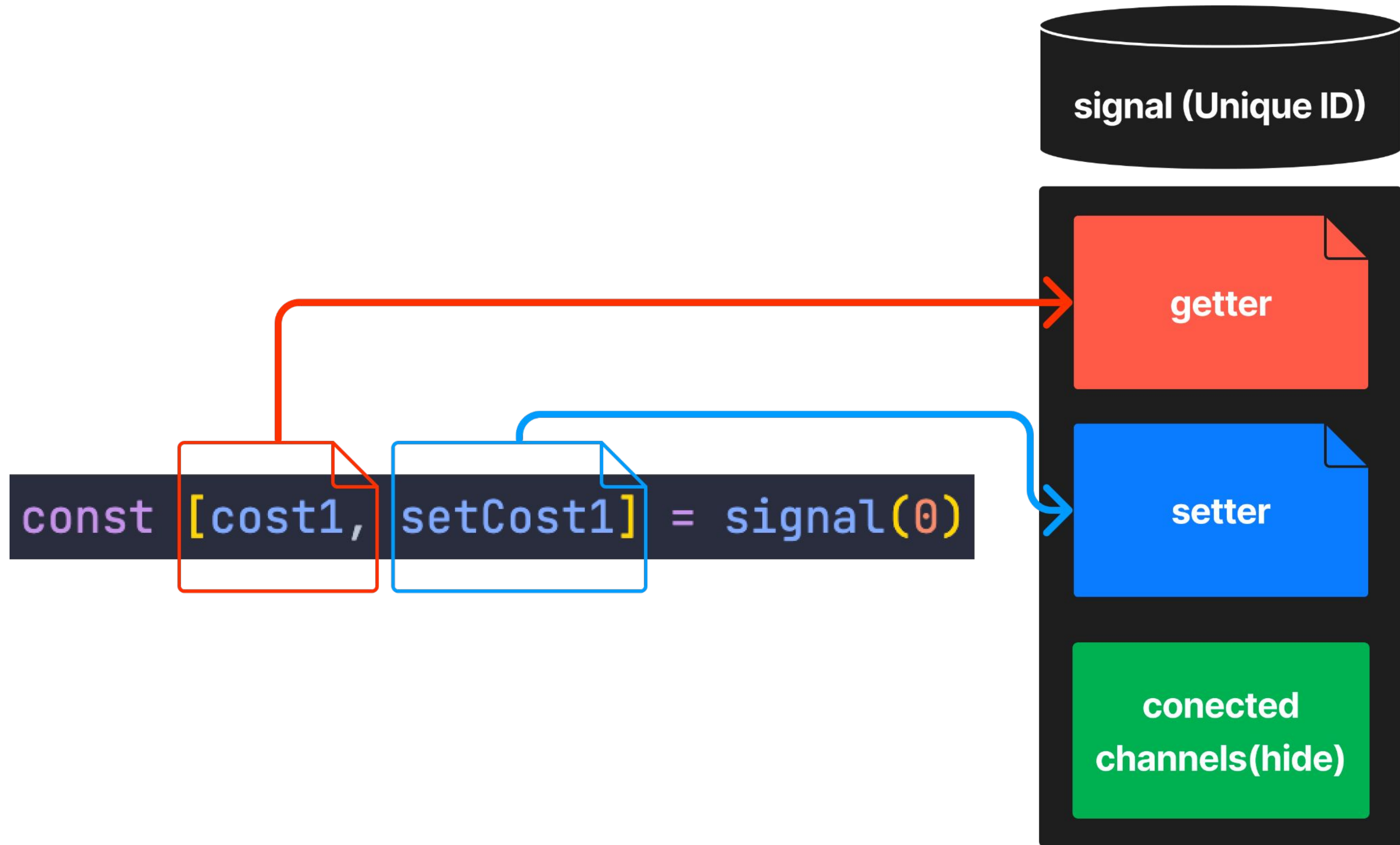


# Reactive

## Reactiveness의 구성요소

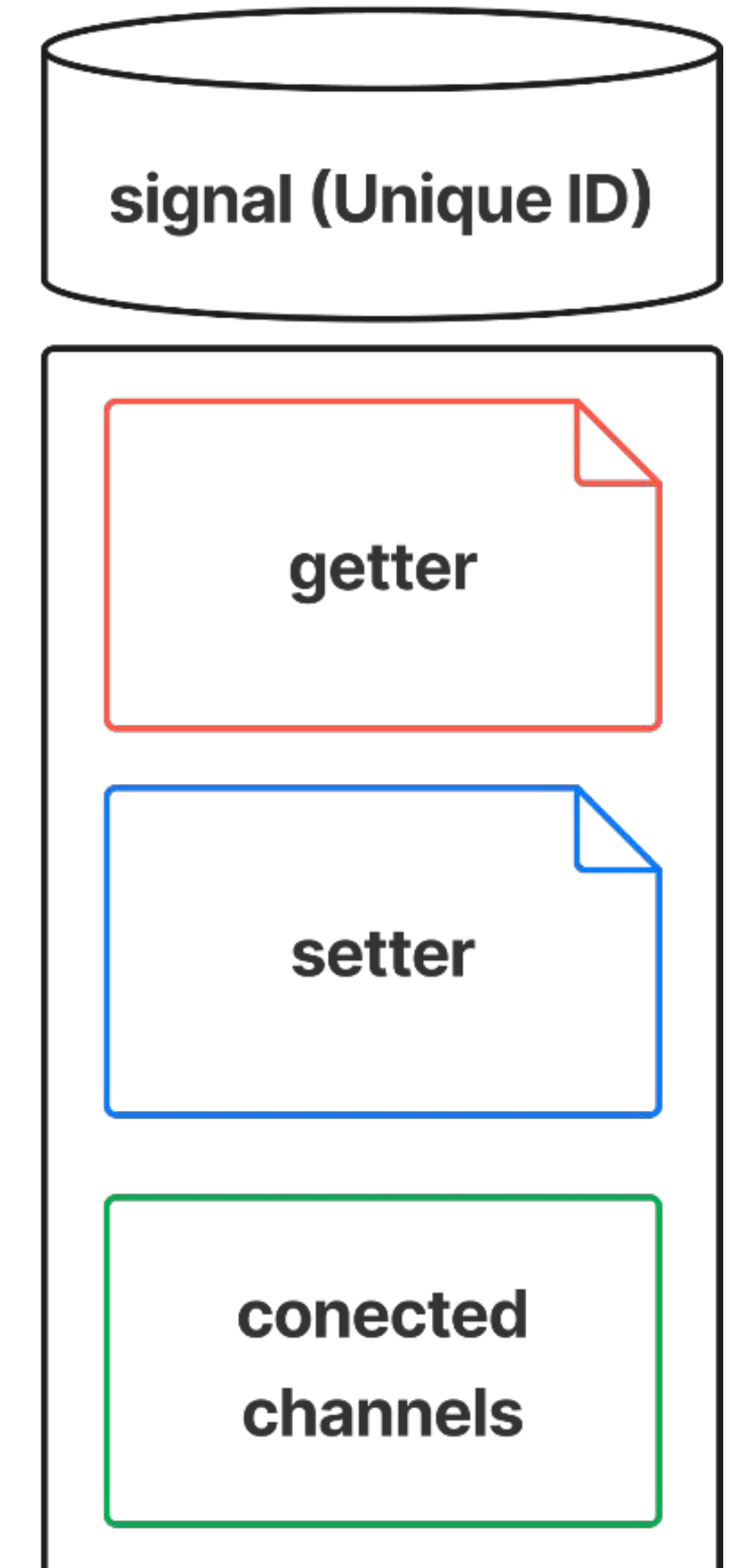
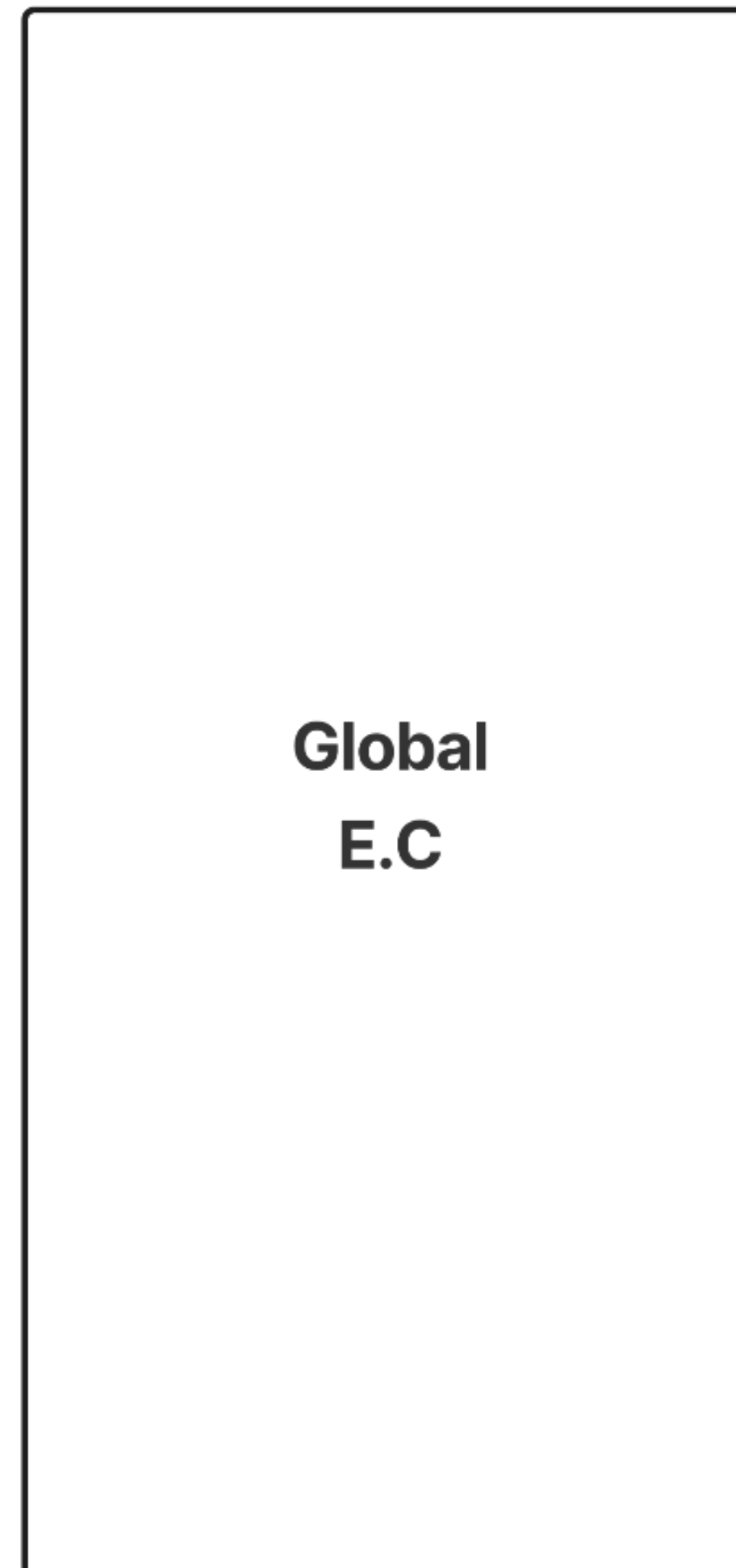
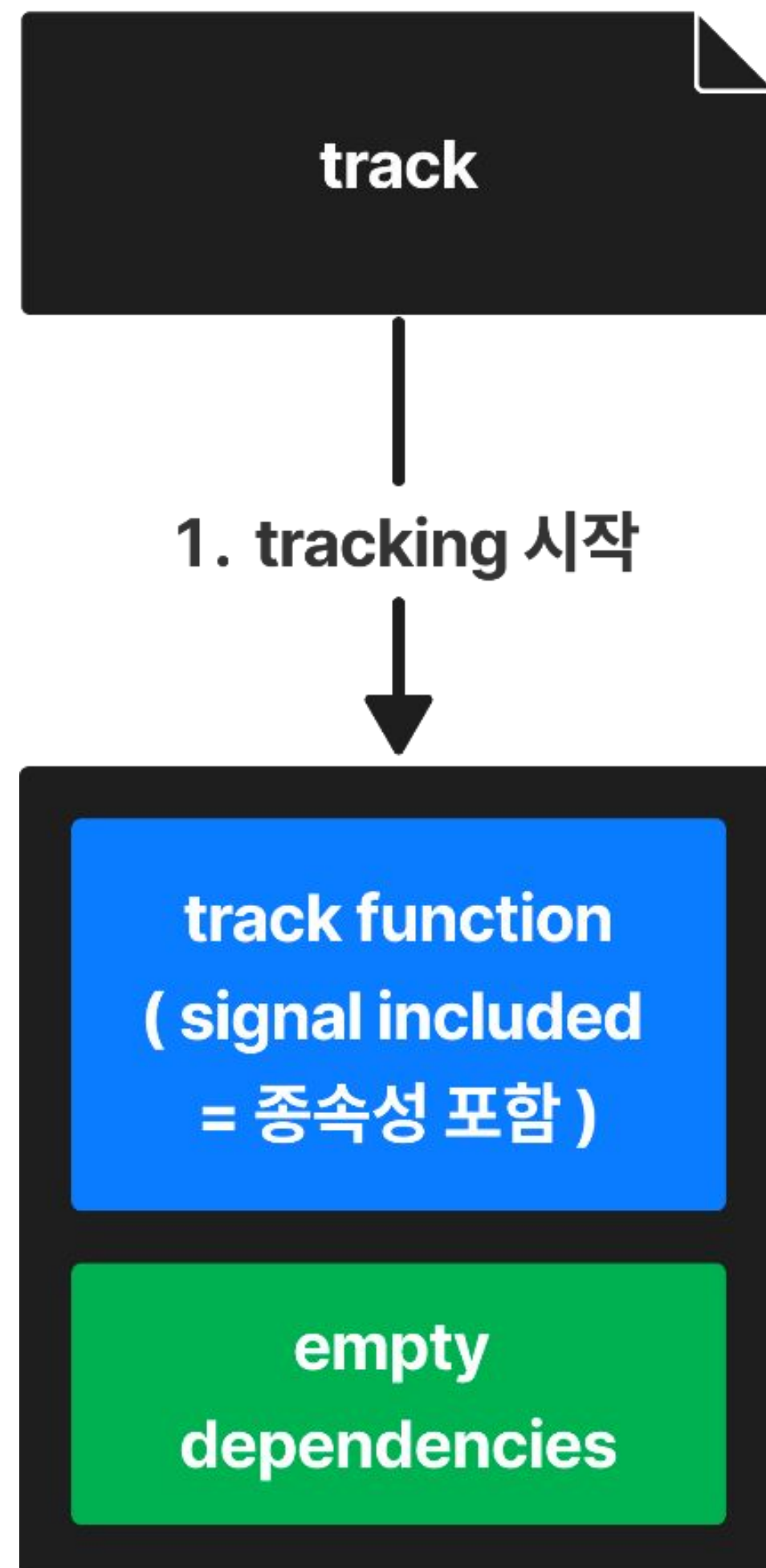


# Reactive - signal

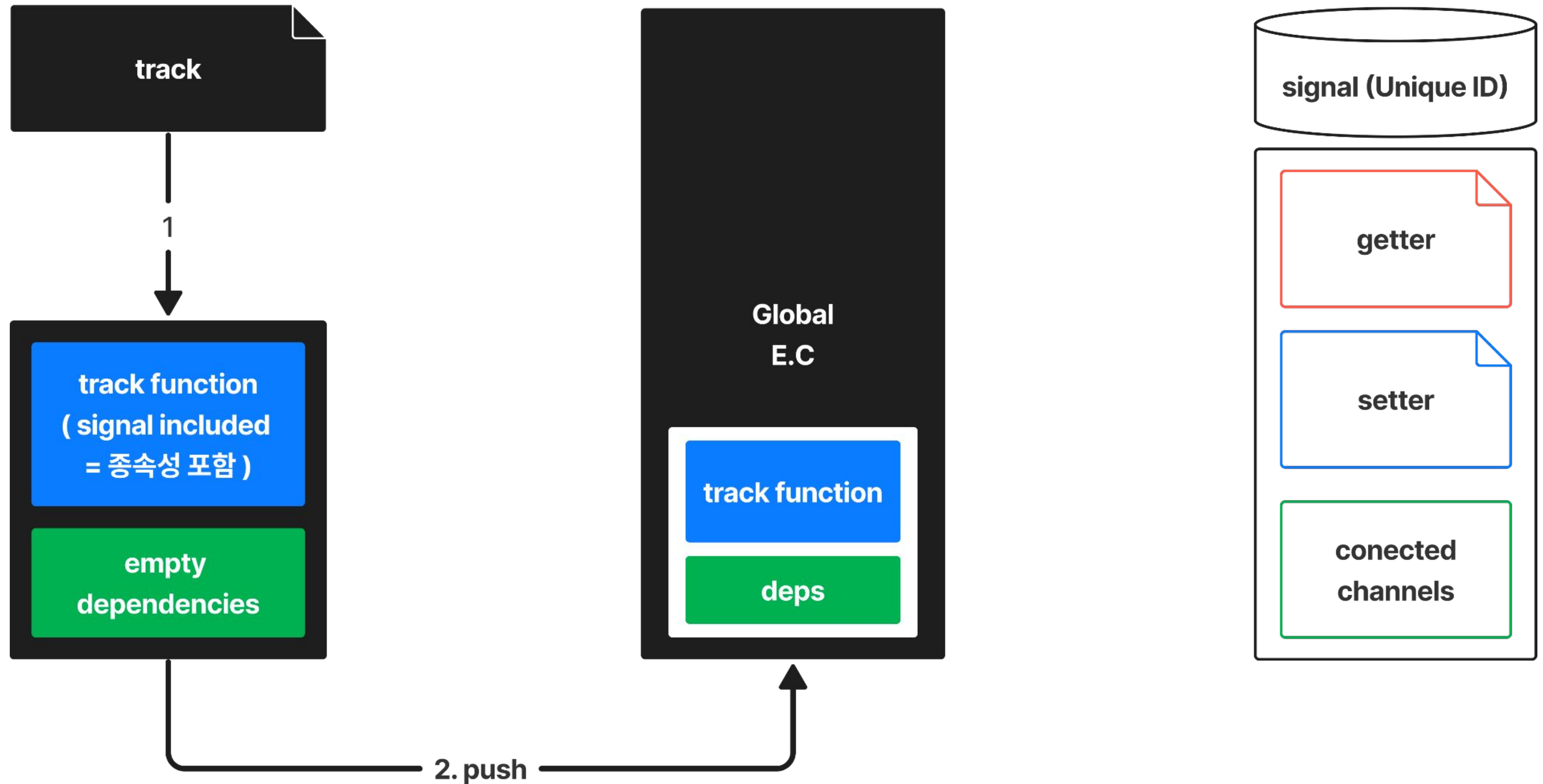




# Reactive - 1

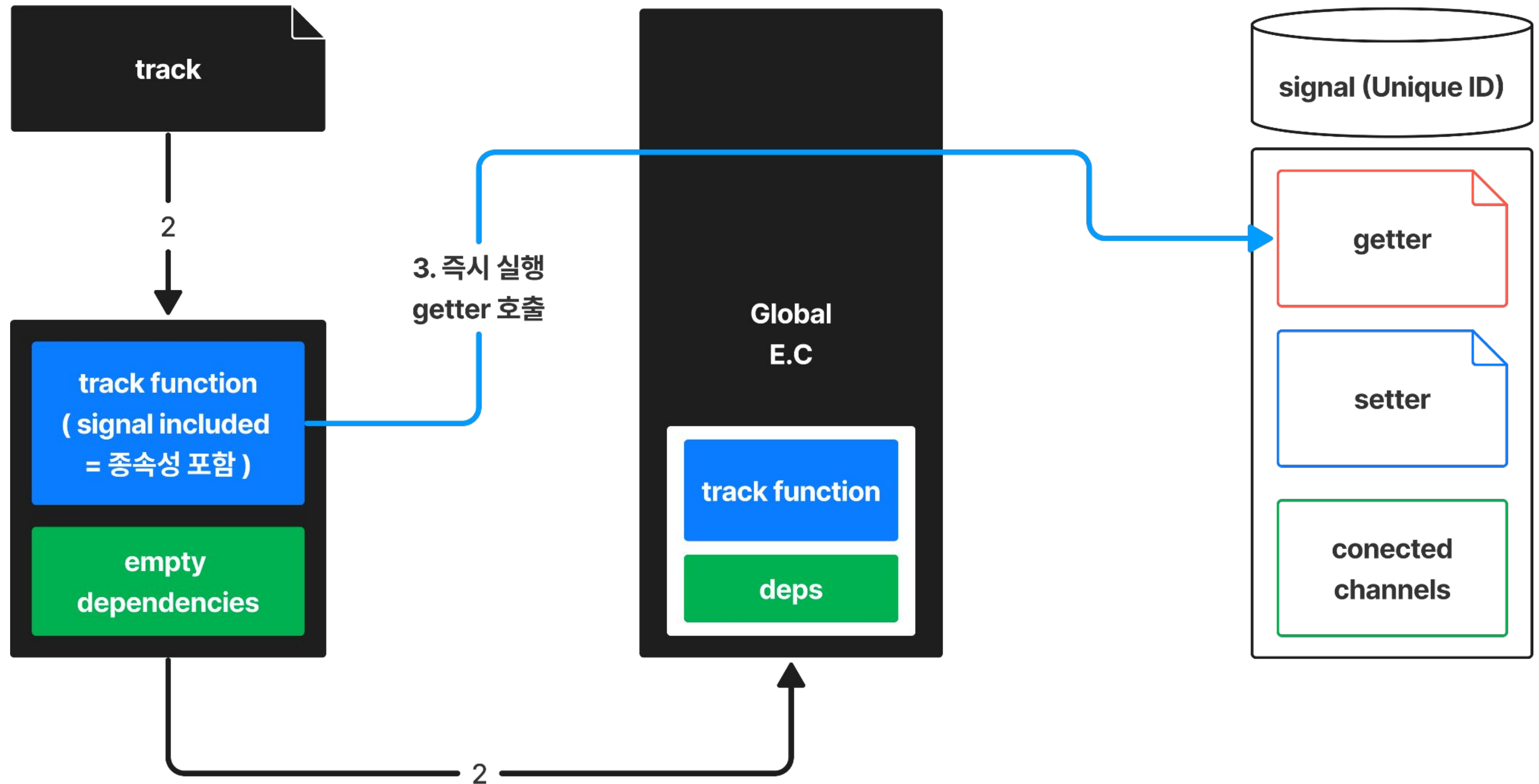


# Reactive - 2

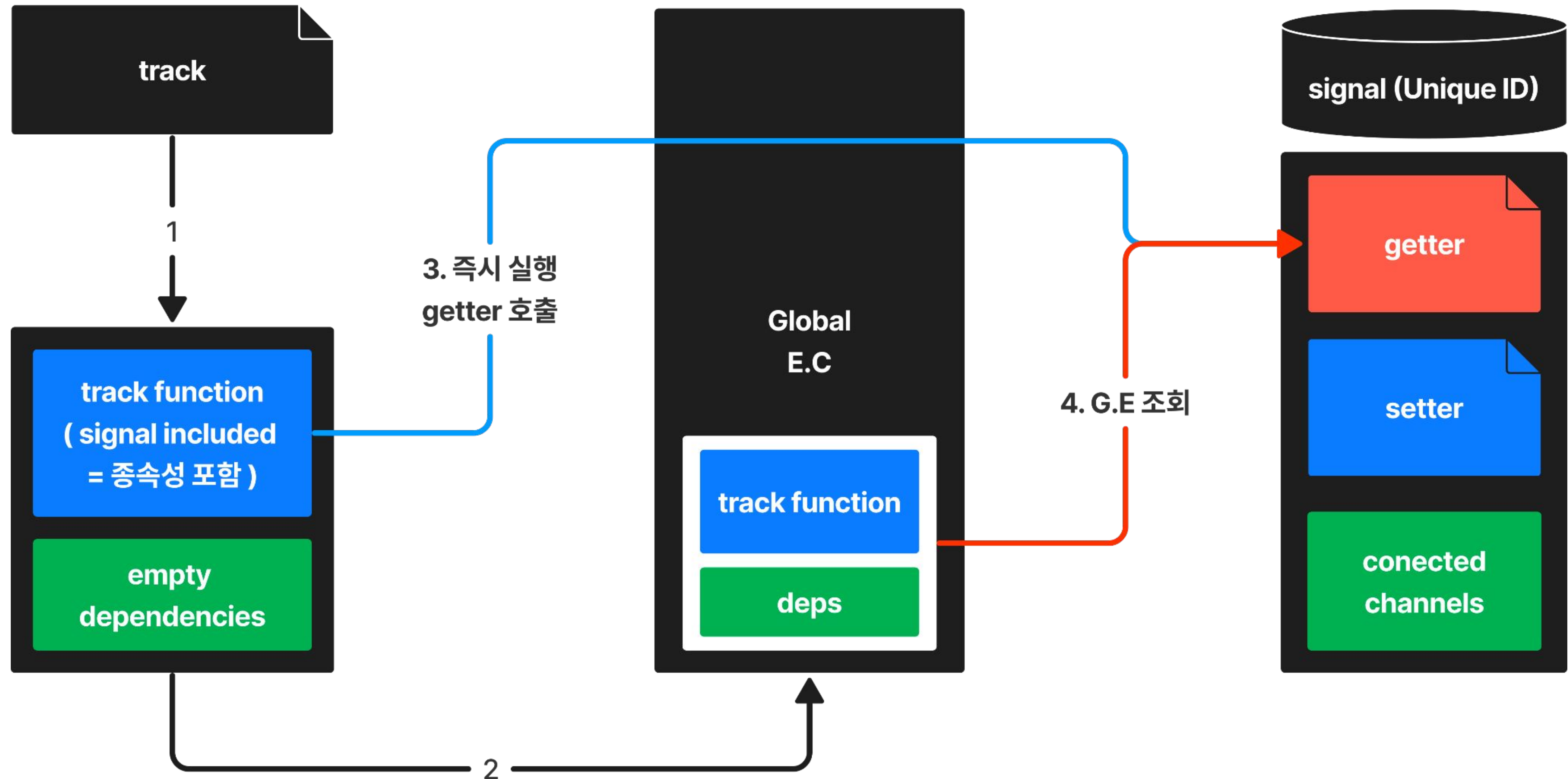




# Reactive - 3

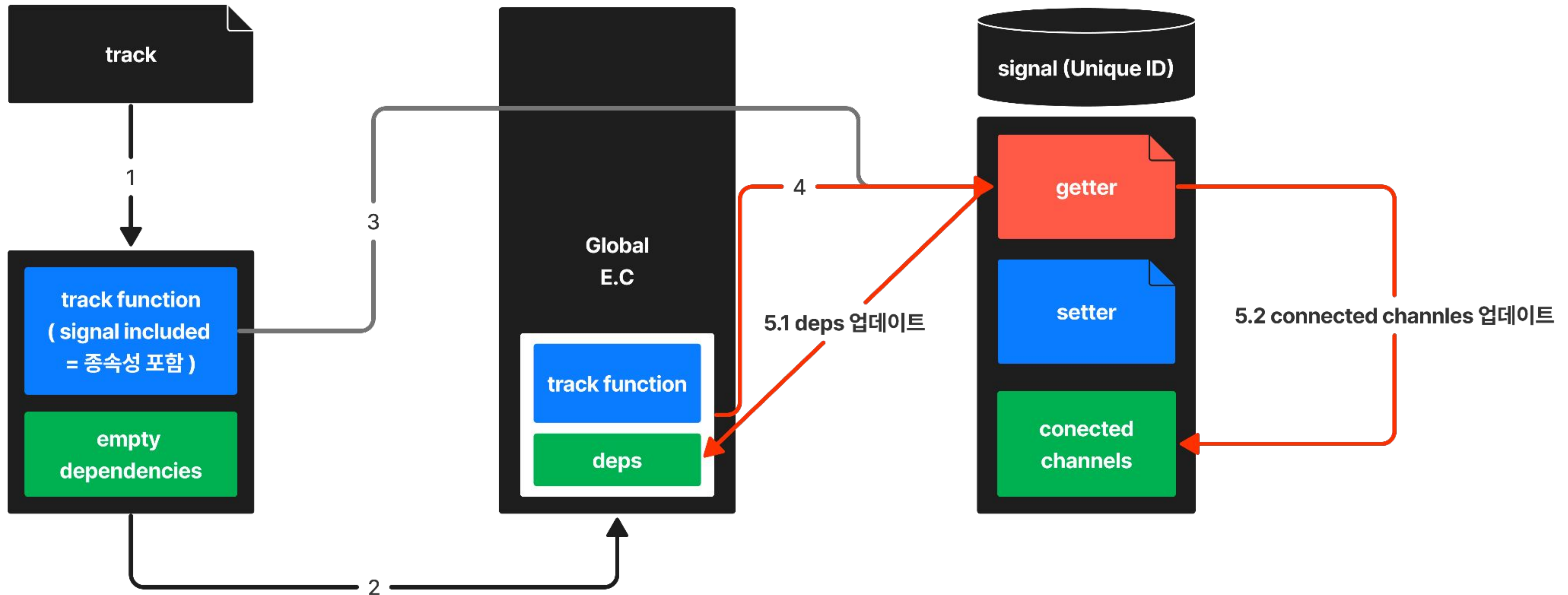


# Reactive - 4

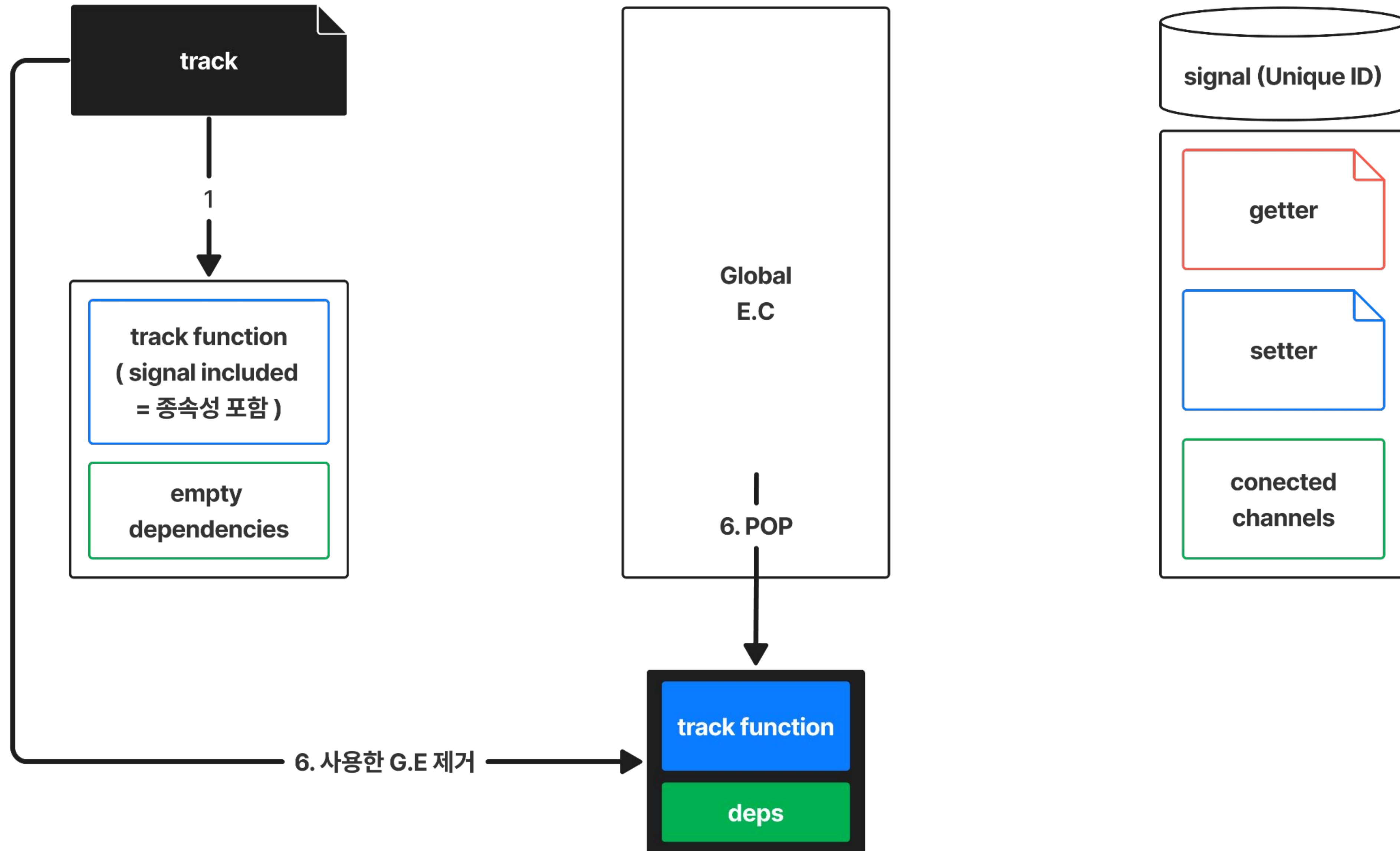




# Reactive - 5.1 & 5.2

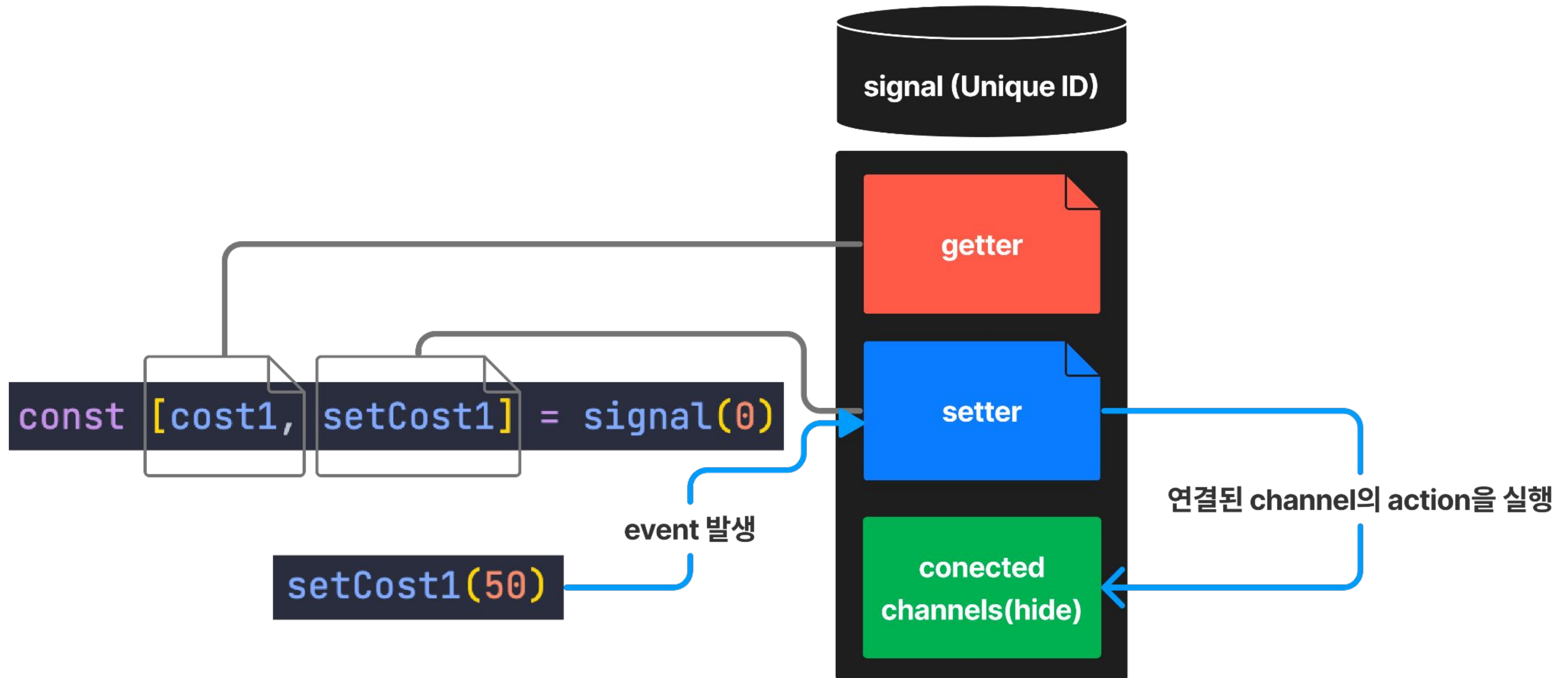


# Reactive - 6





# Reactive - 값 업데이트



# Reactive - 그래서 나온게 100



```
const [cost1, setCost1] = signal(0)
const [cost2, setCost2] = signal(0)

const [TotalCost, setTotalCost] = signal(0)

track(() => {
  setTotalCost(cost1() + cost2())
})

track(() => {
  console.log(TotalCost()) 0, 50, 100
})

setCost1(50)
setCost2(50)
```





개발자는 낭만이  
있다

참고자료



# 구현 결과

Total Done

All 📁

Do 📖

Done 🎉

A hand-drawn sketch of a globe, drawn in blue ink on a grid background. The globe is represented by a circle with several horizontal lines (latitude) and one vertical line (longitude). A small vertical line extends from the bottom of the circle, ending in a short horizontal bar, suggesting a stand or base.

“

A blue rectangular area filled with a grid of small white dots. At the top of the image, there is a white, irregular, torn-paper-like shape that partially obscures the top edge of the blue area. The dots are arranged in a regular grid pattern within the blue area.



# Project 마치고

다음주 학습 내용

Snake game 만들기 / 일상, 준성

React 학습 / 예원

최종 프로젝트 1개 진행 / 하연

