

한입 크기로 잘라먹는 React.js

강의 소개

강의 소개



JavaScript 기초

Node.js 개념 및 기본 사용법

React.js 입문

JavaScript 응용

React.js 기초

3번의 프로젝트

강의 소개

프로젝트1. 카운터 앱



강의 소개

프로젝트2. 투두리스트

오늘은

Tue Apr 19 2022

새로운 Todo 작성하기

추가

Todo List

검색어를 입력하세요

<input type="checkbox"/> React 공부하기	2022. 4. 19.	
<input type="checkbox"/> 빨래 널기	2022. 4. 19.	
<input type="checkbox"/> 노래 연습하기	2022. 4. 19.	
<input type="checkbox"/> 강아지 산책시키기	2022. 4. 19.	
<input type="checkbox"/> 부모님께 안부전화하기	2022. 4. 19.	

강의 소개

프로젝트3. 감정일기장

< 2022년 5월 >

최신순

시작 일기쓰기

수정하기

2022. 5. 31.
5월의 마지막은 정말 좋은일만 가득해

수정하기

2022. 5. 20.
오늘 날씨가 아주 좋았어 한강에 다녀왔다.

수정하기

2022. 5. 11.
목 오늘은 앙증은 일이 너무 많았어

수정하기

2022. 5. 1.
5월 1일의 첫 일기!

수정하기

강의 소개

The screenshot shows a web application interface. At the top, there are two date挑選器 (date pickers) both set to 2022년 5월. Below them is a green button labeled [데이터 유지] (Keep Data). The main area displays a list of mood entries:

Date	Mood	Description
2022. 5. 31.	😊	5월의 마지막날은 정말 즐거웠던 하루였다.
2022. 5. 20.	😊	오늘 일상이 아주 좋아서 친구에게 대화했습니다.
2022. 5. 10.	😢	우리들은 산책을 많이 하루 걸렸다.
2022. 5. 1.	😐	5월 개막식 첫 일요일.

On the right side of the list, there are two buttons: "현재 State값 저장" (Save current State value) with a right-pointing arrow, and "저장된 State값 꺼내옴" (Get saved State value) with a left-pointing arrow. A large blue cylinder icon is positioned to the right of the list, representing Web Storage.

웹 브라우저 내장 DB
Web Storage

강의 소개

winterlood.com

Git Repository Domains Visit

Production Deployment

The deployment that is available to your visitors.

Open Graph 태그 설정

Build Logs Runtime Logs Instant Rollback

감정 일기장

나만의 작은 감정 일기장

emotion-diary-indol.vercel.app

오늘 2:08

To update your Production Deployment, click here.

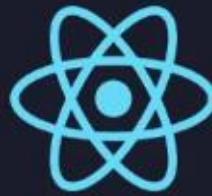
Learn More

This screenshot shows the Vercel dashboard interface. At the top, it displays the domain 'winterlood.com' along with buttons for 'Git Repository', 'Domains', and 'Visit'. Below this, there's a section titled 'Production Deployment' which says 'The deployment that is available to your visitors.' A large button with an upward arrow icon is present. To the right, there's a prominent 'Open Graph 태그 설정' (Open Graph Tag Settings) card. This card features the title '감정 일기장' in large letters, followed by a subtitle '나만의 작은 감정 일기장' and the URL 'emotion-diary-indol.vercel.app'. Above the card, there are tabs for 'Build Logs', 'Runtime Logs', and 'Instant Rollback'. At the bottom of the deployment card, it says '오늘 2:08' (Today 2:08) and 'To update your Production Deployment, click here.' There's also a 'Learn More' button. The overall theme of the dashboard is dark, with light-colored text and buttons.

강의 특징



강의 특징



React.js는 Node.js 기반의 JavaScript 라이브러리



강의 특징

이건 나중에 배워도 되는건가?

이걸 다 언제 공부하지 ..?

이런것까지 다 알아야 되는걸까?



JavaScript

Node.js

React.js

한입 크기로 잘라먹는

안녕! 자바스크립트 

JavaScript는 오늘날 가장 많이 사용 되는 프로그래밍 언어입니다

프로그래밍 언어 사용량 조사 결과

* 2023 Stackoverflow Developer Survey



JavaScript는 무슨 역할을 하는 언어일까?



JavaScript는 무슨 역할을 하는 언어일까?

Web 페이지를 개발하기 위해 필요한 언어들



JavaScript



HTML



CSS



Web 페이지

HTML의 역할

- 요소들의 내용, 배치, 모양을 정하기 위해 사용하는 언어
- 색상이나 디자인 등의 수정은 불가

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"/>
  </head>
  <body>
    <h2>안녕하세요!</h2>
    <button>버튼</button>
  </body>
</html>
```

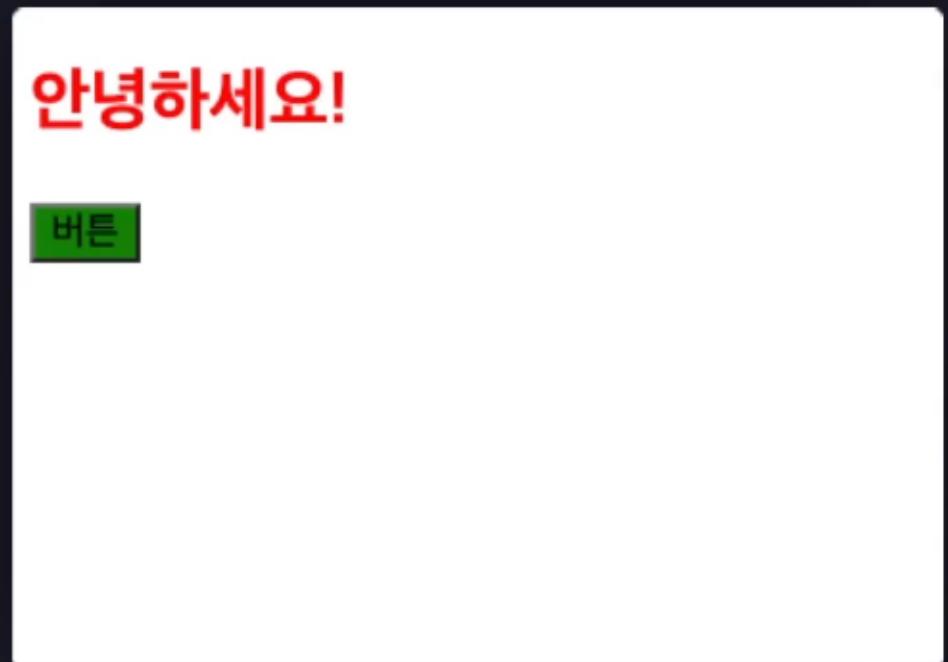
안녕하세요!

버튼

CSS의 역할

- 요소들의 색상, 크기 등의 스타일을 설정할 수 있음

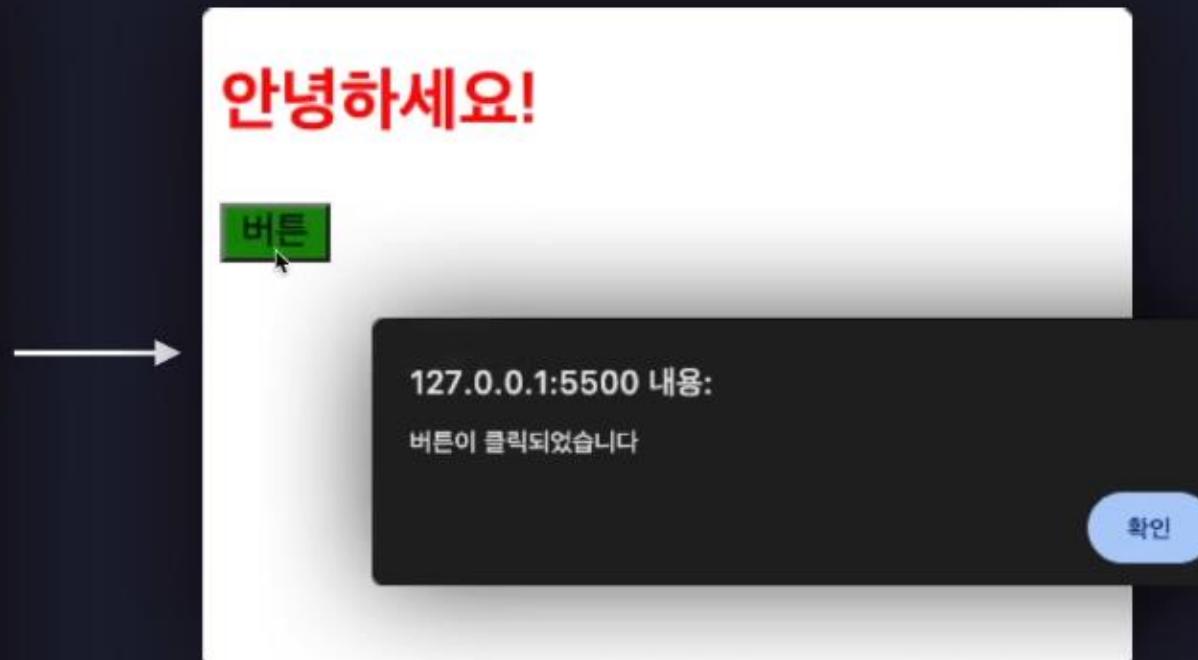
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"/>
    <style>
      h2 { color:red }
      button { background:green }
    </style>
  </head>
  <body>
    <h2>안녕하세요!</h2>
    <button>버튼</button>
  </body>
</html>
```



JavaScript의 역할

- 웹 내부에서 발생하는 다양한 기능을 만들 수 있는 언어
- 웹을 움직이게 하는 “웹의 근육”이라고 표현 할 수 있음

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"/>
    <style>
      h2 { color:red }
      button { background:green }
    </style>
    <script>
      function onClick(){
        alert("버튼이 클릭되었습니다")
      }
    </script>
  </head>
  <body>
    <h2>안녕하세요!</h2>
    <button onclick="onClick()">버튼</button>
  </body>
</html>
```



JavaScript는 어떻게 실행될까?

- JavaScript는 “JavaScript 엔진”에 의해 실행된다



JavaScript 엔진

JavaScript는 어떻게 실행될까?

- 엔진은 게임 구동기, JavaScript는 게임 이라고 이해하면 쉽다



JavaScript 엔진 = 게임 구동기(닌텐도)

JavaScript는 어떻게 실행될까?

- JavaScript 엔진은 브라우저에 기본 탑재되어 있다.
- 따라서 웹 브라우저를 이용하면 간단한 JavaScript 코드를 직접 실행해 볼 수 있다.



Web 브라우저



한입 크기로 잘라먹는

Visual Studio Code 설치하기

Vistual Studio Code란?



Visual Studio Code

- Microsoft에서 개발한 오픈소스 소스코드 에디터
- 줄여서 VSCode라고 부름
- 2024년 현재 가장 많은 개발자들이 사용하는 에디터

Integrated development environment



Visual Studio Code remains the preferred IDE across all developers, increasing its use among those learning to code compared to professional developers (78% vs. 74%).

All Respondents

Professional Developers

Learning to Code

Other Coders

86,544 responses

Visual Studio Code

73.71%

Visual Studio

28.43%

한입 크기로 잘라먹는

JavaScript 실습환경 설정하기

한입 크기로 잘라먹는

변수와 상수



변수, 상수란?

변수, 상수는 값을 저장하는 박스



한입 크기로 잘라먹는

자료형(타입)

자료형이란?

자료형(Type) =

집합

동일한 속성이나 특징을 가진 원소들을 묶은 것

ex. 고양이와 강아지는 “동물”이라는 집합으로 묶임

자료형(Type)은 “집합”이다

String 타입



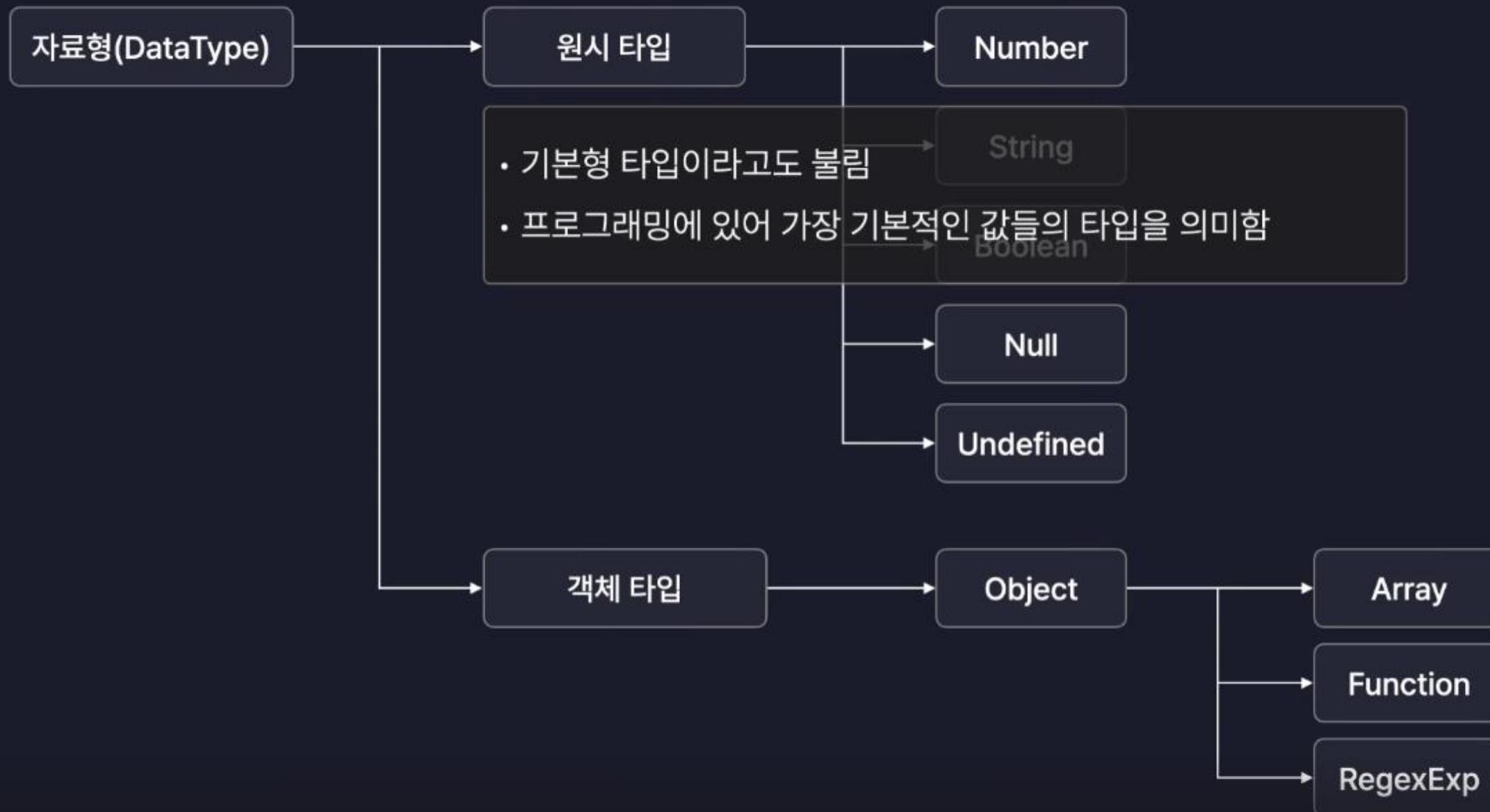
Number 타입



JavaScript의 자료형(DataType)



JavaScript의 자료형(Data Type)



한입 크기로 잘라먹는
형 변환



형 변환이란?

형 변환 (Type Casting)

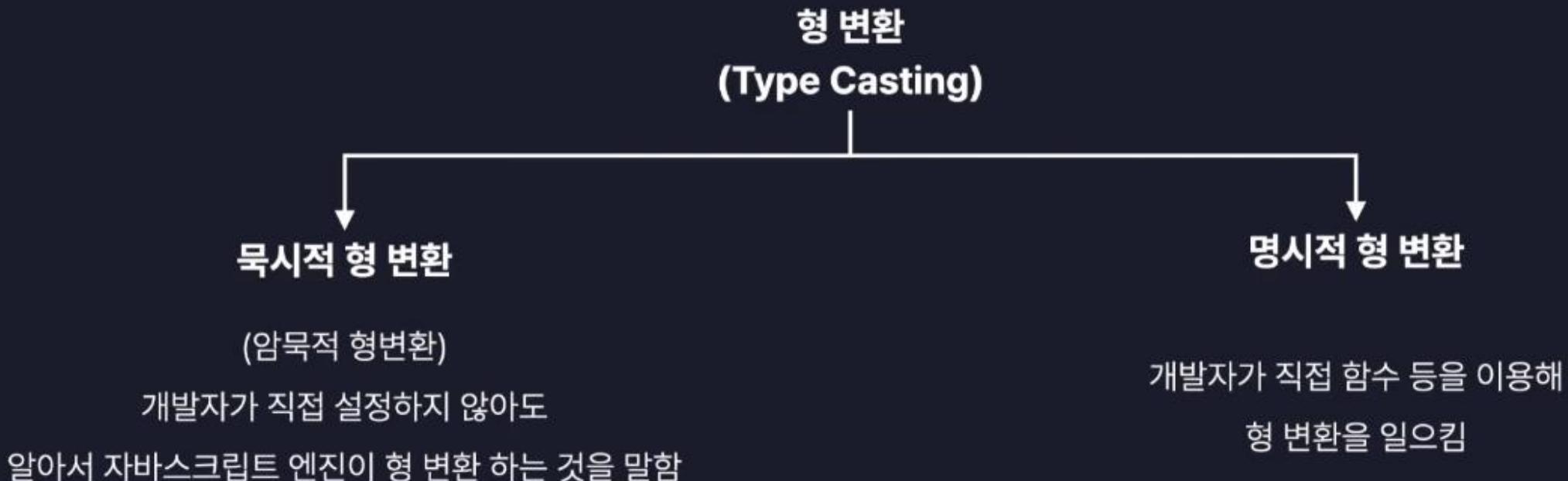
- 어떤 값의 타입을 다른 타입으로 변경하는 것을 말함



형 변환이란?

형 변환 (Type Casting)

- 어떤 값의 타입을 다른 타입으로 변경하는 것을 말함



한입 크기로 잘라먹는

연산자 1



연산자(Operator)란?

- 프로그래밍에서의 다양한 연산을 위한 **기호**, **키워드**를 말함

덧셈 연산자

+

곱셈 연산자

*

뺄셈 연산자

-

나누셈 연산자

/

기타 등등 ...

한입 크기로 잘라먹는

연산자 2



한입 크기로 잘라먹는

조건문 

조건문(Conditional Statement) 이란?

- 특정 조건을 만족했을 때에만 실행되는 코드를 작성하기 위한 문법
- 대표적으로 if, switch 조건문이 존재함



한입 크기로 잘라먹는

반복문 

반복문 (Loop, Iteration)

- 어떠한 동작을 반복해서 수행할 수 있도록 만들어 주는 문법

< 요구 사항 >

1부터 100까지 콘솔에 하나씩 출력하세요

```
console.log(1);  
console.log(2);  
console.log(3);
```

...

```
console.log(98);  
console.log(99);  
console.log(100);
```

한입 크기로 잘라먹는

함수 

중복으로 작성되는 코드들

```
let width1 = 10;  
let height1 = 20;  
let area1 = width1 * height1;
```

```
(...)
```

```
let width2 = 100;  
let height2 = 200;  
let area2 = width2 * height2;
```

```
console.log("면적: ", area1);  
console.log("면적: ", area2);
```



중복으로 작성된
유사한 기능을 하는 코드

중복으로 작성되는 코드들

```
let width1 = 10;  
let height1 = 20;  
let area1 = width1 * height1;  
  
(...)  
  
let width2 = 100;  
let height2 = 200;  
let area2 = width2 * height2;  
  
console.log("면적: ", area1);  
console.log("면적: ", area2);
```

중복으로 작성된
유사한 기능을 하는 코드

- 동일한 기능을 일일이 타이핑 해야 함
- 향후 코드를 수정시 문제 될 수 있음

중복으로 작성되는 코드들

```
getArea(10, 20);  
(...)  
getArea(100, 200);  
console.log("면적: ", area1);  
console.log("면적: ", area2);
```

함수 getArea

```
let width1 = 10;  
let height1 = 20;  
let area1 = width1 * height1;
```

한입 크기로 잘라먹는

함수 표현식과 화살표 함수

한입 크기로 잘라먹는

콜백함수 

콜백 함수(Callback Function)란?

- 자신이 아닌 다른 함수에, 인수로써 전달된 함수를 의미 함

```
function main(value) {  
}  
  
function sub() {  
    console.log("sub");  
}
```

콜백 함수(Callback Function)란?

- 자신이 아닌 다른 함수에, 인수로써 전달된 함수를 의미 함

```
function main(value) {  
    value();  
}  
  
function sub() {  
    console.log("sub");  
}  
  
main(sub);
```

실행 결과

sub

chapter11.js:6

한입 크기로 잘라먹는

스코프



스코프(Scope)란?

- 우리말로 “범위” 를 뜻함
- 변수나 함수에 접근하거나 호출할 수 있는 범위를 말 함

```
function funcA() {  
  let a = 1;  
}  
  
console.log(a);
```

스코프(Scope)란?

- 우리말로 “범위” 를 뜻함
- 변수나 함수에 접근하거나 호출할 수 있는 범위를 말 함

```
function funcA() {  
    let a = 1;  
}  
  
console.log(a);
```

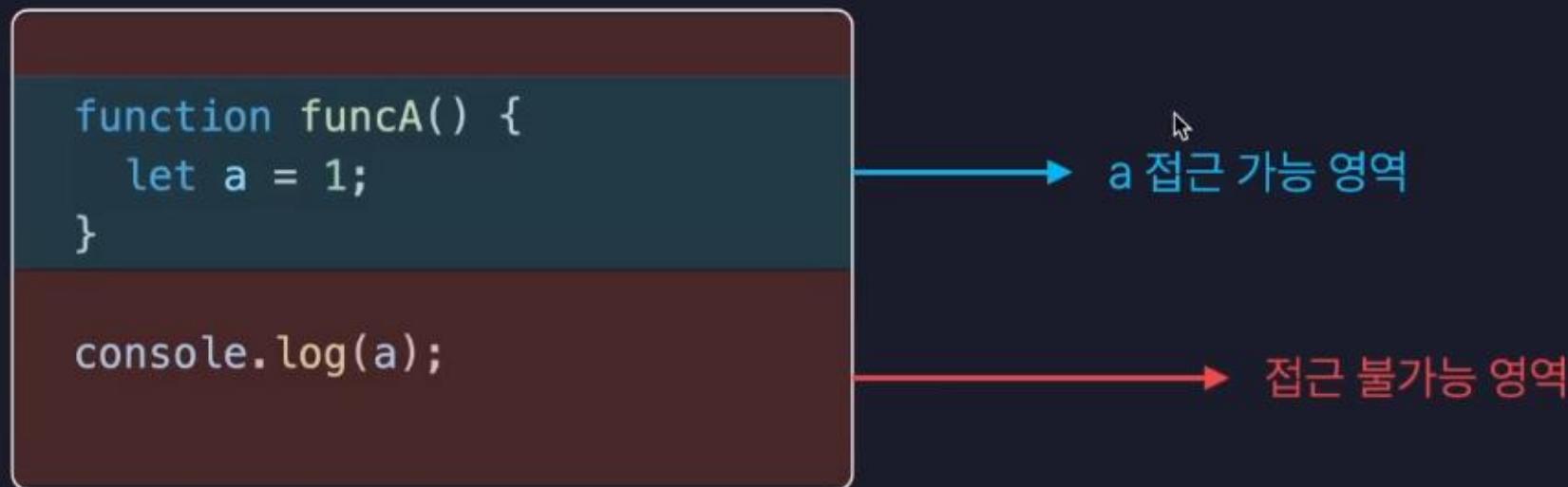
[❗ 오류 발생] a라는 변수는 모릅니다

✖ ▶ Uncaught ReferenceError: chapter12.js:5
a is not defined
at chapter12.js:5:13



스코프(Scope)란?

- 우리말로 “범위” 를 뜻함
- 변수나 함수에 접근하거나 호출할 수 있는 범위를 말 함



한입 크기로 잘라먹는

객체 1



객체(Object)란?

- 원시 타입이 아닌 객체 타입의 자료형
- 여러가지 값을 동시에 저장할 수 있는 자료형을 의미



객체(Object)란?

- 객체를 이용하면 현실세계에 존재하는 어떤 사물이나 개념을 표현하기 용이함

[현실 세상]



[프로그래밍 세상]

```
let person = {  
    name : "차은우",  
    nickName : "siyeonBoy",  
    age : 20,  
    location : "서울시 어딘가"  
}
```

한입 크기로 잘라먹는

객체 2



한입 크기로 잘라먹는

배열



배열(Array)이란?

- 여러개의 값을 **순차적으로** 담을 수 있는 자료 형



한입 크기로 잘라먹는

Truthy & Falsy

JavaScript 에서는 참, 거짓이 아닌 값도 참, 거짓으로 평가한다

```
if (123) {  
    console.log("123 is true");  
} else {  
    console.log("123 is false");  
}  
  
if (undefined) {  
    console.log("undefined is true");  
} else {  
    console.log("undefined is false");  
}
```

123 is true

chapter01.js:2

undefined is false

chapter01.js:10

실행 결과

Truthy & Falsy란?

- 참이나 거짓을 의미하지 않는 값도, 조건문 내에서 참이나 거짓으로 평가하는 특징

```
if (123) { _____  
    console.log("123 is true");  
} else {  
    console.log("123 is false");  
}
```

Truthy 한 값
(참 같은 값)

```
if (undefined) { _____  
    console.log("undefined is true");  
} else {  
    console.log("undefined is false");  
}
```

Falsy 한 값
(거짓 같은 값)

JavaScript의 모든 값은 Truthy 하거나 Falsy 합니다

- 이를 이용하면 조건문을 간결하게 만들 수 있음

Truthy

(참 같은 값)

Falsy

(참 같은 값)



???



???

한입 크기로 잘라먹는

단락 평가



단락 평가(Short-circuit Evaluation)이란?

```
let varA = false;  
  
let varB = true;  
  
// AND 연산자  
console.log(varA && varB);  
  
// OR 연산자  
console.log(varB || varA);
```

한입 크기로 잘라먹는

구조 분해 할당



구조 분해 할당이란?



한입 크기로 잘라먹는

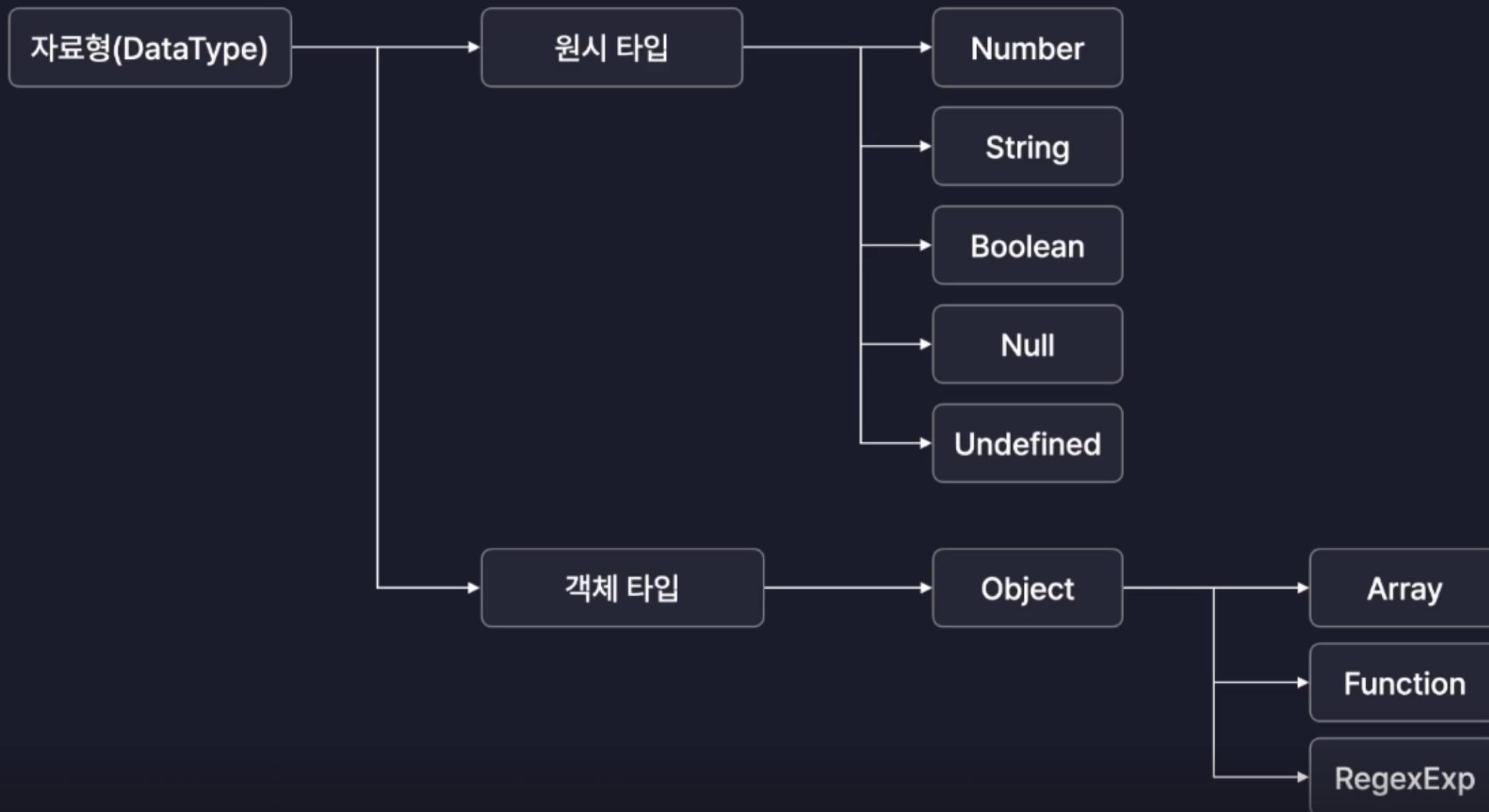
Spread 연산자 & Rest 매개변수

한입 크기로 잘라먹는

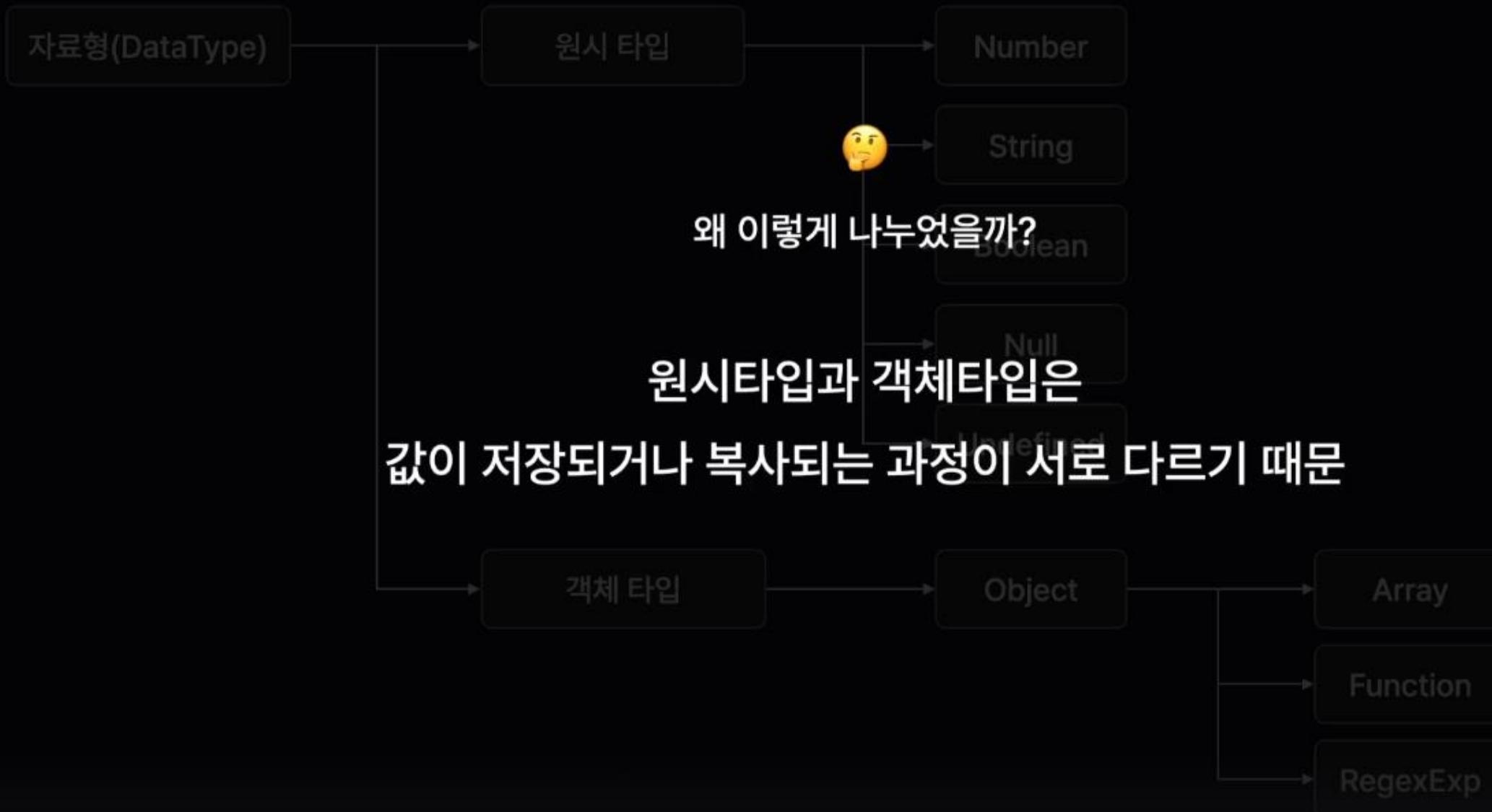
원시타입 VS 객체타입



자바스크립트의 자료형(타입)



자바스크립트의 자료형(타입)



원시타입 VS 객체타입

원시 타입

Number, String, Boolean 등 ...

값 자체로써 변수에 저장되고 복사 된다

객체 타입

Object, Array, Function 등 ...

참조값을 통해 변수에 저장되고 복사된다

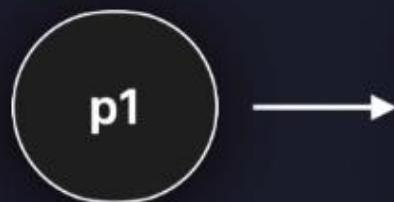
VS

원시 타입 = 불변값

[Code]

```
let p1 = 1;  
let p2 = p1;  
  
p2 = 2;
```

[Name]



[Memory]

1
1
2

설명: 실제 메모리의 값은 수정되지 않음

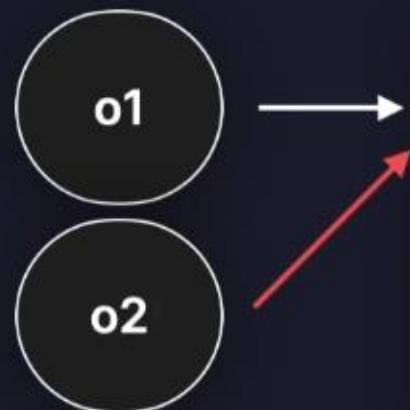
객체타입 = 가변값

[Code]

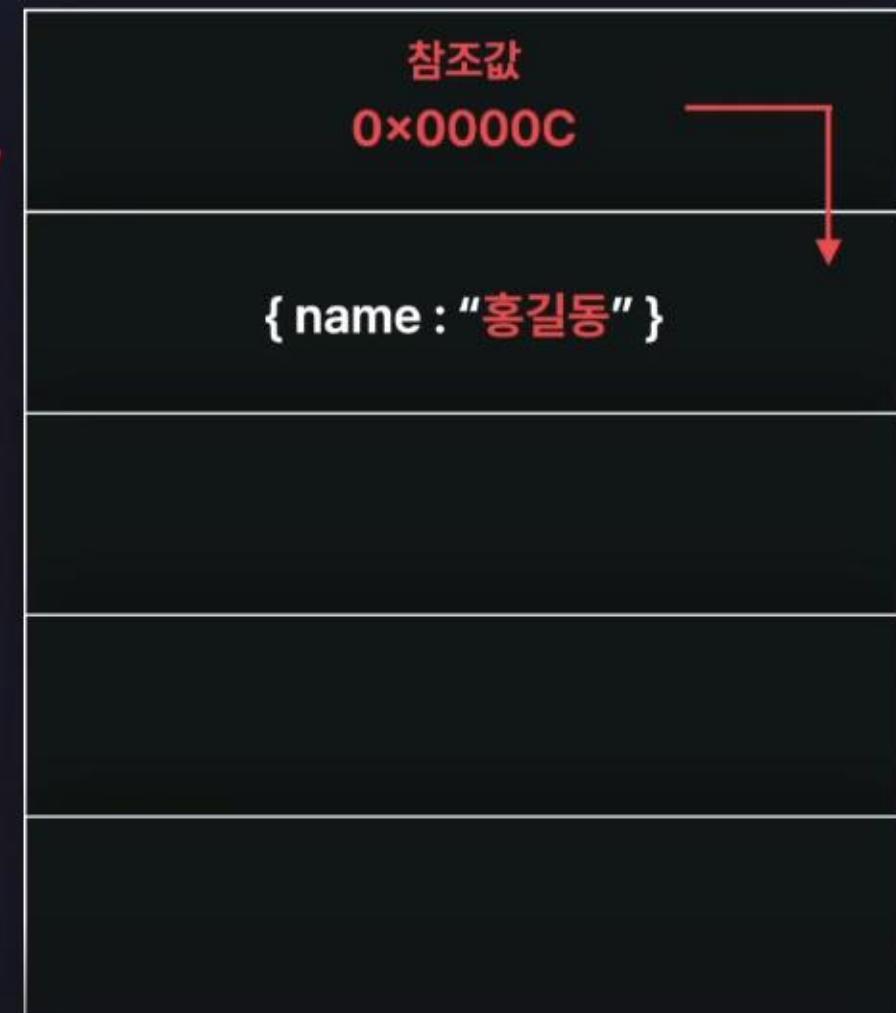
```
let o1 = { name: "김시연" };
let o2 = o1;

o2.name = "홍길동";
```

[Name]



[Memory]



원시타입 VS 객체타입

원시 타입

Number, String, Boolean 등 ...

값 자체로써 변수에 저장되고 복사 된다

불변값이다 (메모리 값 수정 X)

객체 타입

Object, Array, Function 등 ...

참조값을 통해 변수에 저장되고 복사된다

가변값이다 (메모리 값 수정 O)

VS

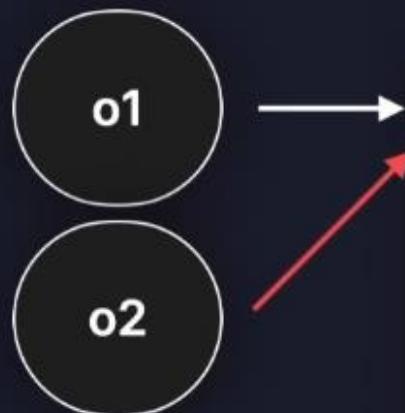
객체타입 = 가변값

[Code]

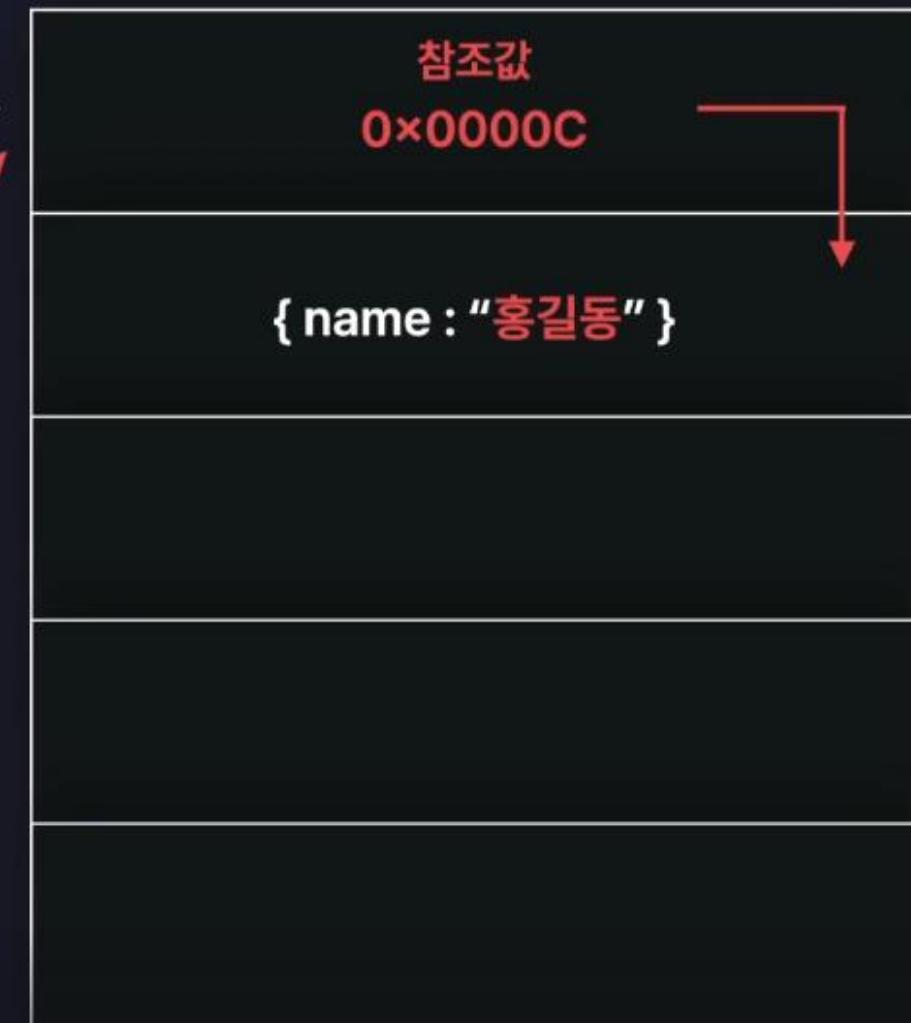
```
let o1 = { name: "김시연" };
let o2 = o1;

o2.name = "홍길동";
```

[Name]



[Memory]



원시타입 VS 객체타입

원시 타입

Number, String, Boolean 등 ...

값 자체로써 변수에 저장되고 복사 된다

불변값이다 (메모리 값 수정 X)

VS

Object, Array, Function 등 ...

참조값을 통해 변수에 저장되고 복사된다

가변값이다 (메모리 값 수정 O)

객체 타입

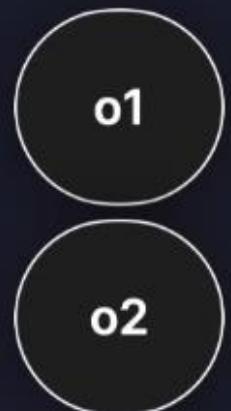
객체 탑재 주의사항 1. 의도치 않게 값이 수정될 수 있다.

[Code]

```
let o1 = { name: "김시연" };
let o2 = o1;

o2.name = "홍길동";
```

[Name]



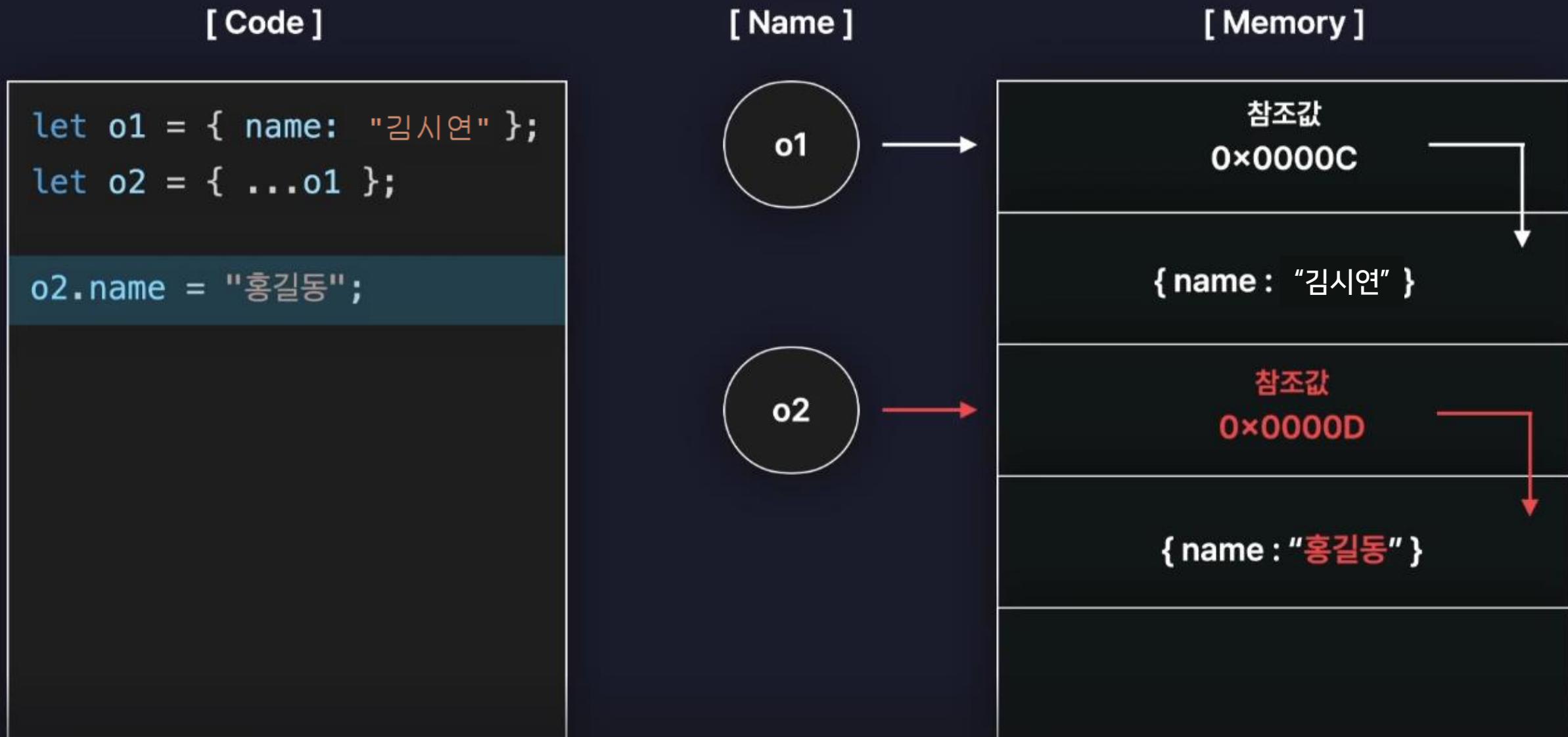
[Memory]



객체 타입 주의사항 1. 의도치 않게 값이 수정될 수 있다.



객체 탑주 의사항 1. 의도치 않게 값이 수정될 수 있다.



객체 타입 주의사항 1. 의도치 않게 값이 수정될 수 있다.

얕은 복사

```
let o1 = { name: "김시연" };
let o2 = o1;
```

객체의 참조값을 복사함

원본 객체가 수정될 수 있어 위험함

깊은 복사

```
let o1 = { name: "김시연" };
let o2 = { ...o1 };
```

새로운 객체를 생성하면서
프로퍼티만 따로 복사 함

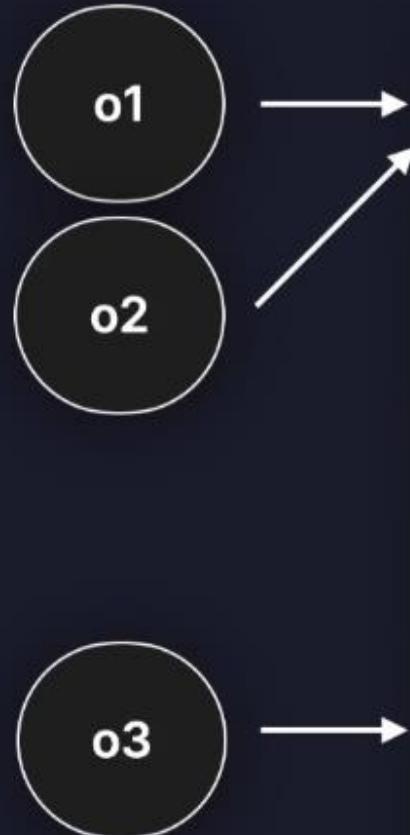
원본 객체가 수정될 일이 없어 안전함

객체 타입 주의사항 2. 객체간의 비교는 기본적으로 참조값을 기준으로 이루어진다

[Code]

```
let o1 = { name: "김시연" };
let o2 = o1;
let o3 = { ...o1 };
```

[Name]



[Memory]



객체 타입 주의사항 2. 객체간의 비교는 기본적으로 참조값을 기준으로 이루어진다

[Code]

```
let o1 = { name: "김시연" };
let o2 = o1;
let o3 = { ...o1 };

console.log(o1 === o2);
console.log(o1 === o3);

console.log(
  JSON.stringify(o1) === JSON.stringify(o3)
);
// 결과 : True
```

JSON.stringify()

자바스크립트 내장 함수

객체를 문자열로 변환하는 기능

객체 타입 주의사항 2. 객체간의 비교는 기본적으로 참조값을 기준으로 이루어진다

얕은 비교

```
o1 === o2
```

참조값을 기준으로 비교

깊은 비교

```
JSON.stringify(o1) === JSON.stringify(o2)
```

객체를 문자열로 변환하여 비교

JSON.stringify 등의 내장 함수를 이용해야 함

객체 타입 주의사항 3. 배열과 함수도 사실 객체이다



객체 타입 주의사항 3. 배열과 함수도 사실 객체이다



한입 크기로 잘라먹는

반복문으로 배열과 객체 순회하기 

순회(Iteration)이란?

- 배열, 객체에 저장된 여러개의 값에 순서대로 하나씩 접근하는 것을 말함

ex) 배열 순회

```
let numbers = [1, 2, 3];
```

ex) 객체 순회

```
let person = {  
    name: "김시연",  
    age: 27,  
    hobby: "테니스",  
};
```

순회(Iteration)이란?

- 배열, 객체에 저장된 여러개의 값에 순서대로 하나씩 접근하는 것을 말함

반복문을 이용한 배열, 객체 순회

```
for (let value of numbers) {  
    console.log(value);  
}  
  
for (let key in Object.keys(person)) {  
    console.log(key);  
}
```

한입 크기로 잘라먹는

배열 메서드 1. 요소 조작

한입 크기로 잘라먹는

배열 메서드 2. 순회와 탐색

한입 크기로 잘라먹는

배열 메서드 3. 변형

한입 크기로 잘라먹는

Date 객체와 날짜

한입 크기로 잘라먹는

동기와 비동기



동기란 무엇일까?

“동기”란 무엇일까?

대학 동기?, 회사 동기?

동기란 무엇일까?

여러개의 작업

Task A

Task B

Task C

동기란 무엇일까?

“동기적으로 처리한다”



동기란 무엇일까?



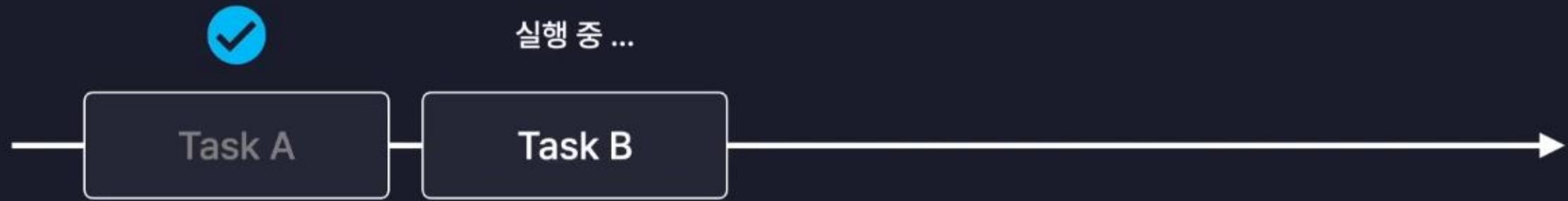
타임라인 

동기란 무엇일까?

실행 중 ...



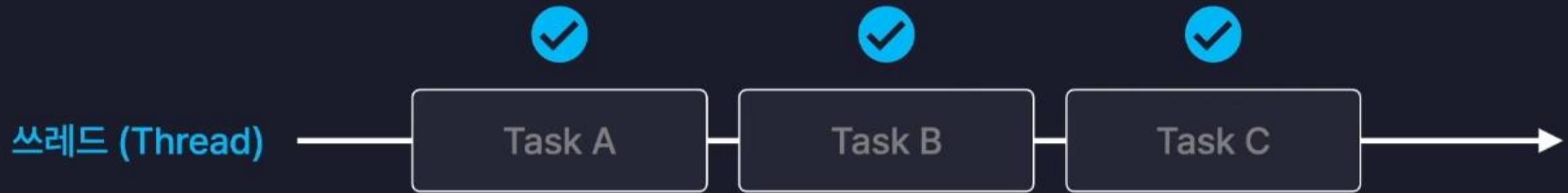
동기란 무엇일까?



동기란 무엇일까?



동기란 무엇일까?



동기란 무엇일까?

“동기”란 무엇일까?

여러개의 작업을 순서대로, 하나씩 처리하는 방식

JavaScript는 “동기”적으로 코드를 실행한다

JavaScript 코드

```
console.log("1");
console.log("2");
console.log("3");
```

실행 결과

```
1
2
3
```

JavaScript는 “동기”적으로 코드를 실행한다

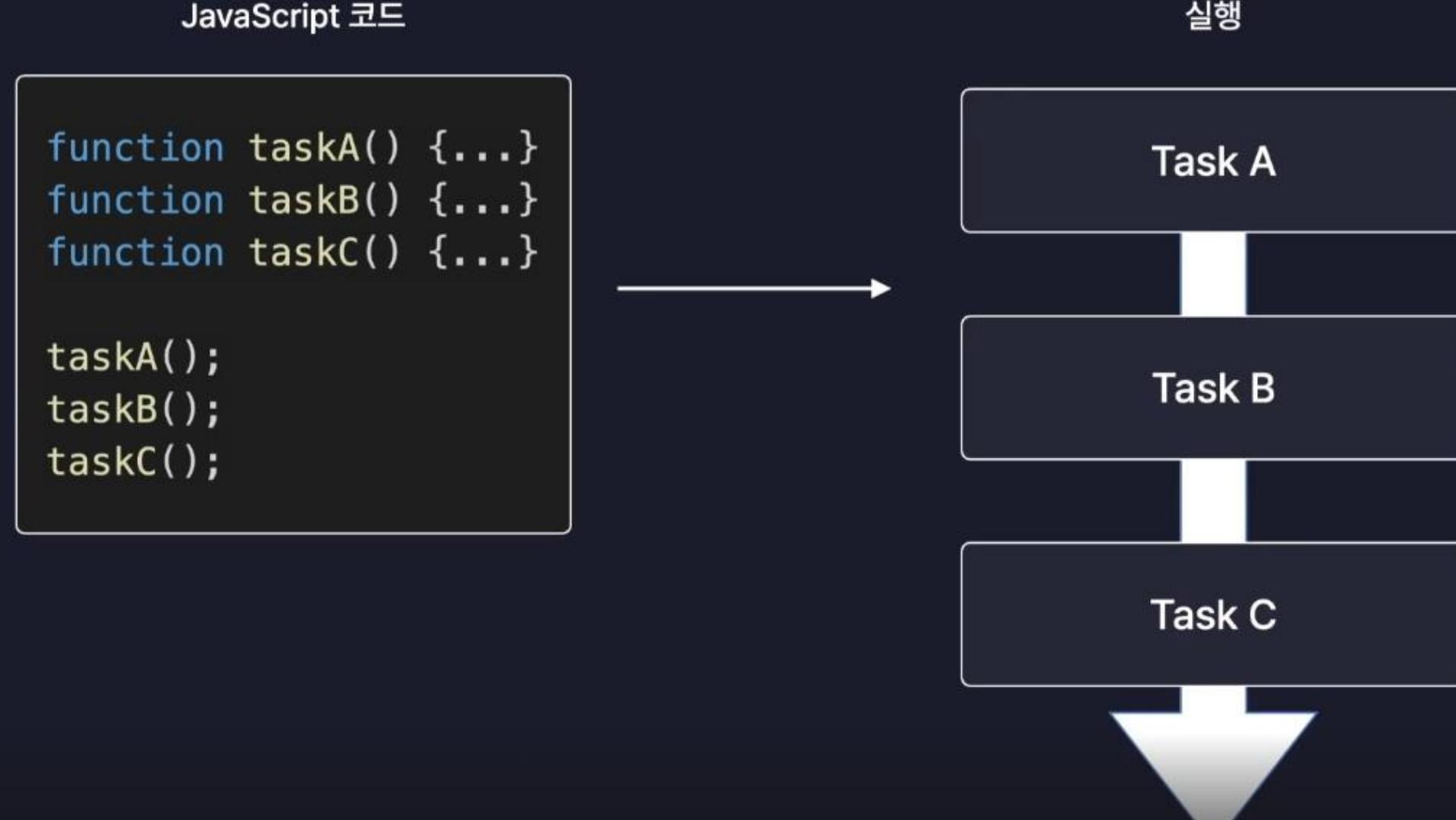
JavaScript 코드

```
function taskA() {  
    console.log("2");  
}  
  
console.log("1");  
taskA();  
console.log("3");
```

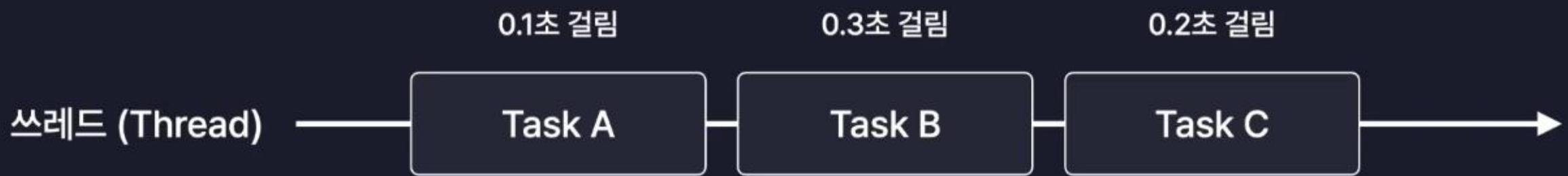
실행 결과

```
1  
2  
3
```

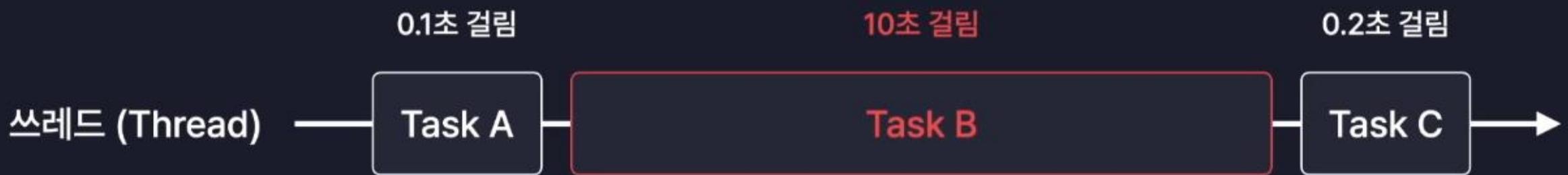
동기 방식에는 치명적인 단점이 존재한다



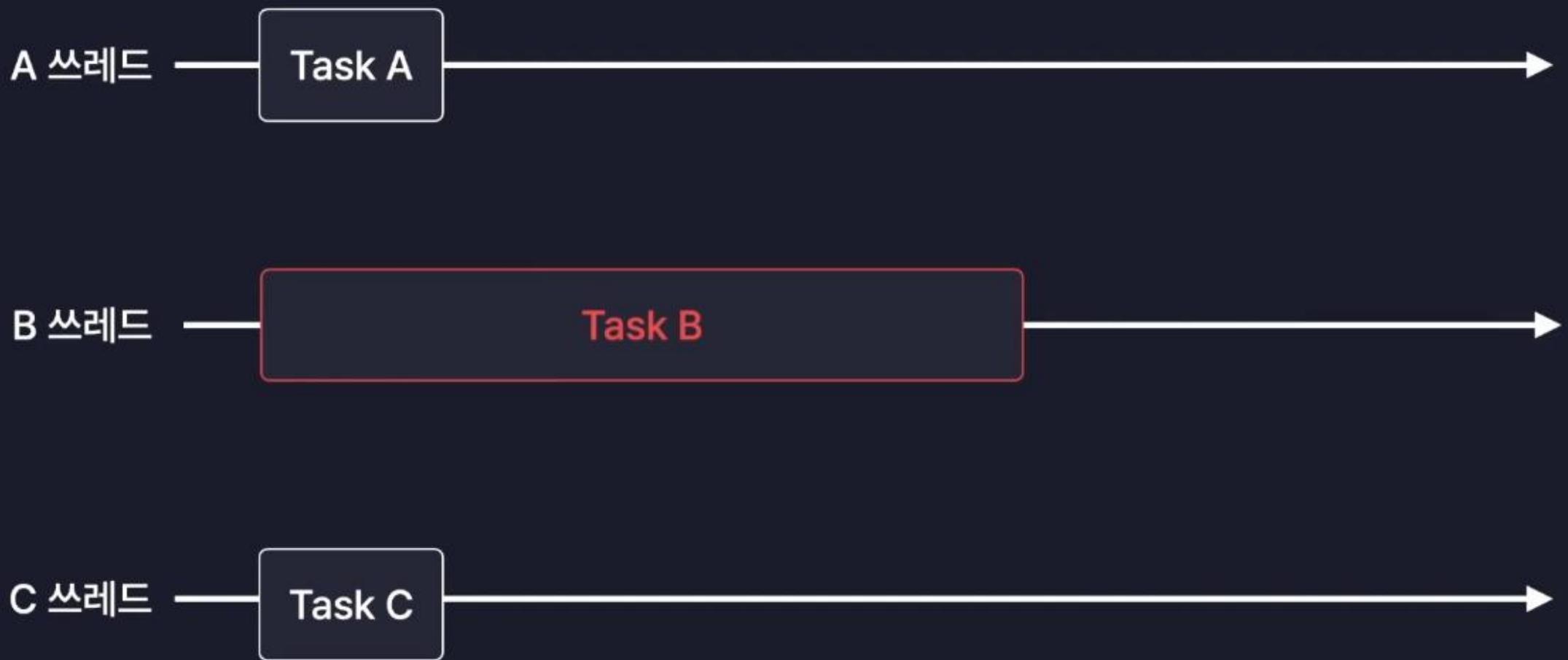
동기 방식에는 치명적인 단점이 존재한다



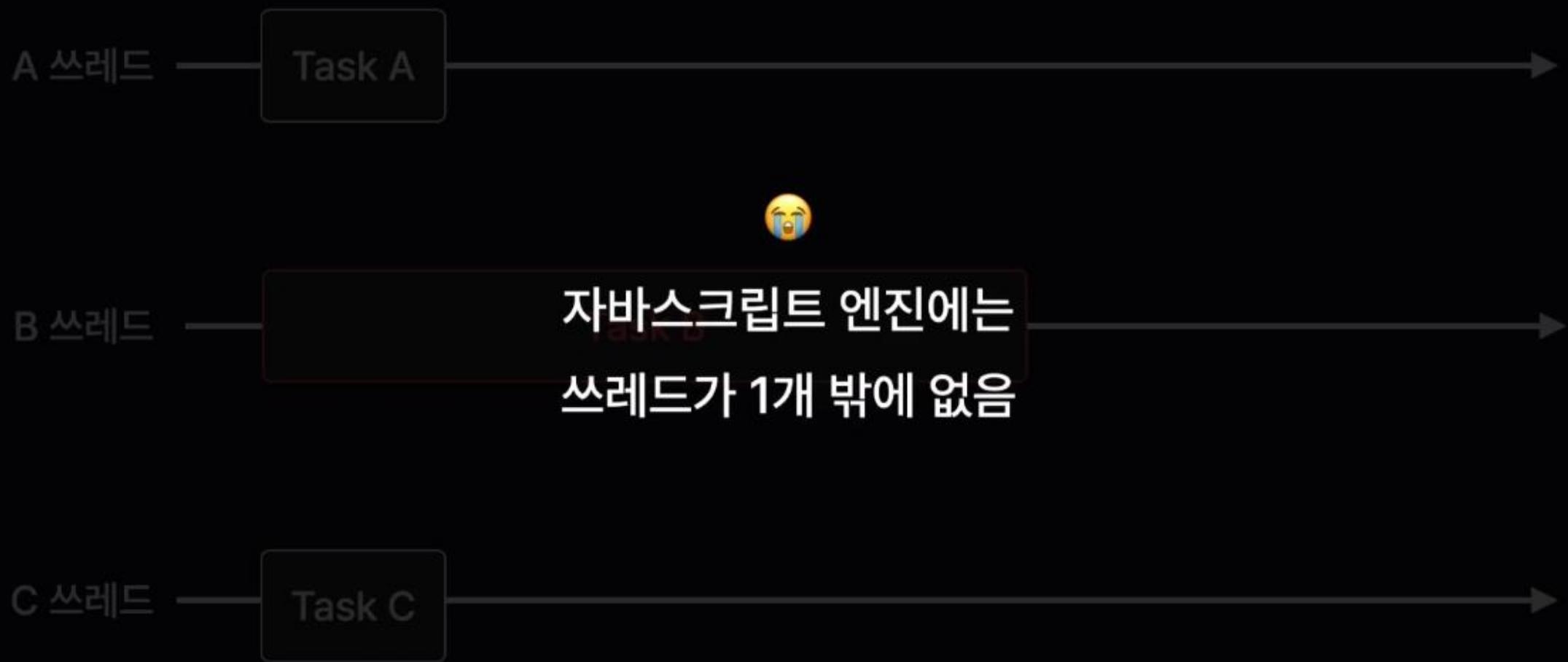
동기 방식에는 치명적인 단점이 존재한다



동기 방식에는 치명적인 단점이 존재한다



동기 방식에는 치명적인 단점이 존재한다



비동기란 무엇일까?

“비동기”란 무엇일까?

동기적이지 않다는 뜻

작업을 순서대로 처리하지 않음

비동기란 무엇일까?

여러개의 작업

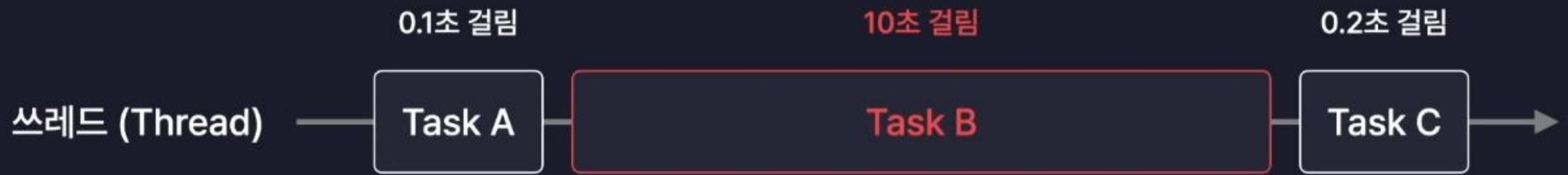
... 진행 중

Task A

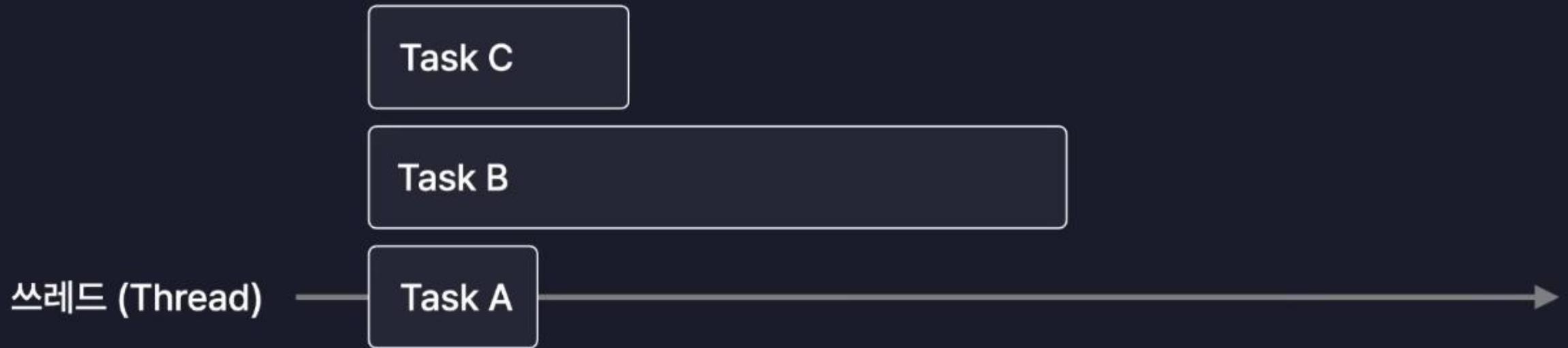
Task B

Task C

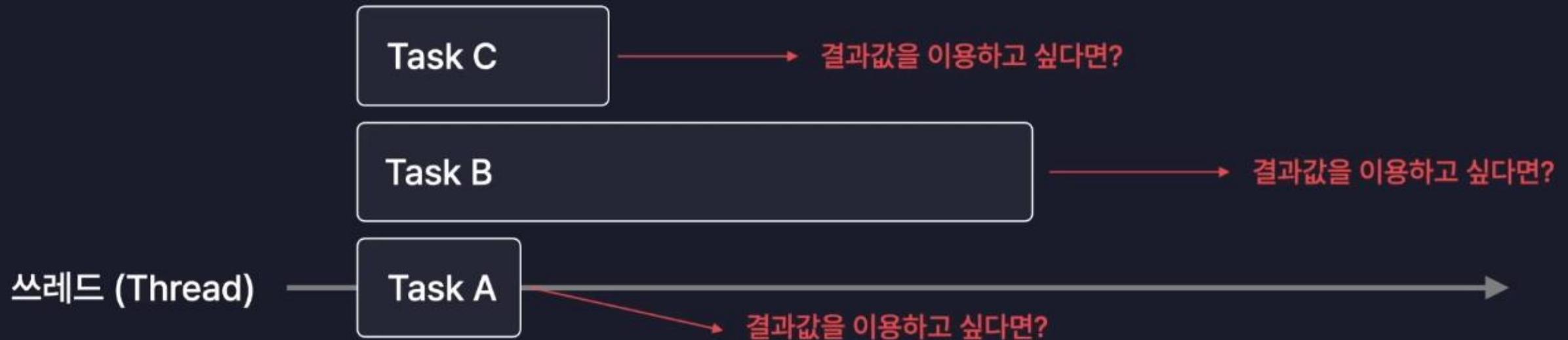
비동기란 무엇일까?



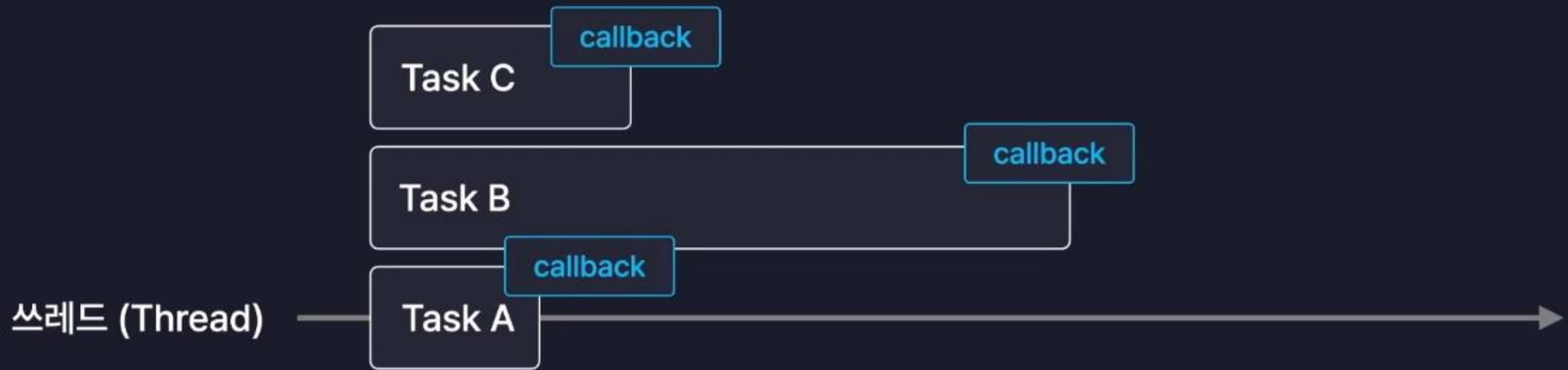
비동기란 무엇일까?



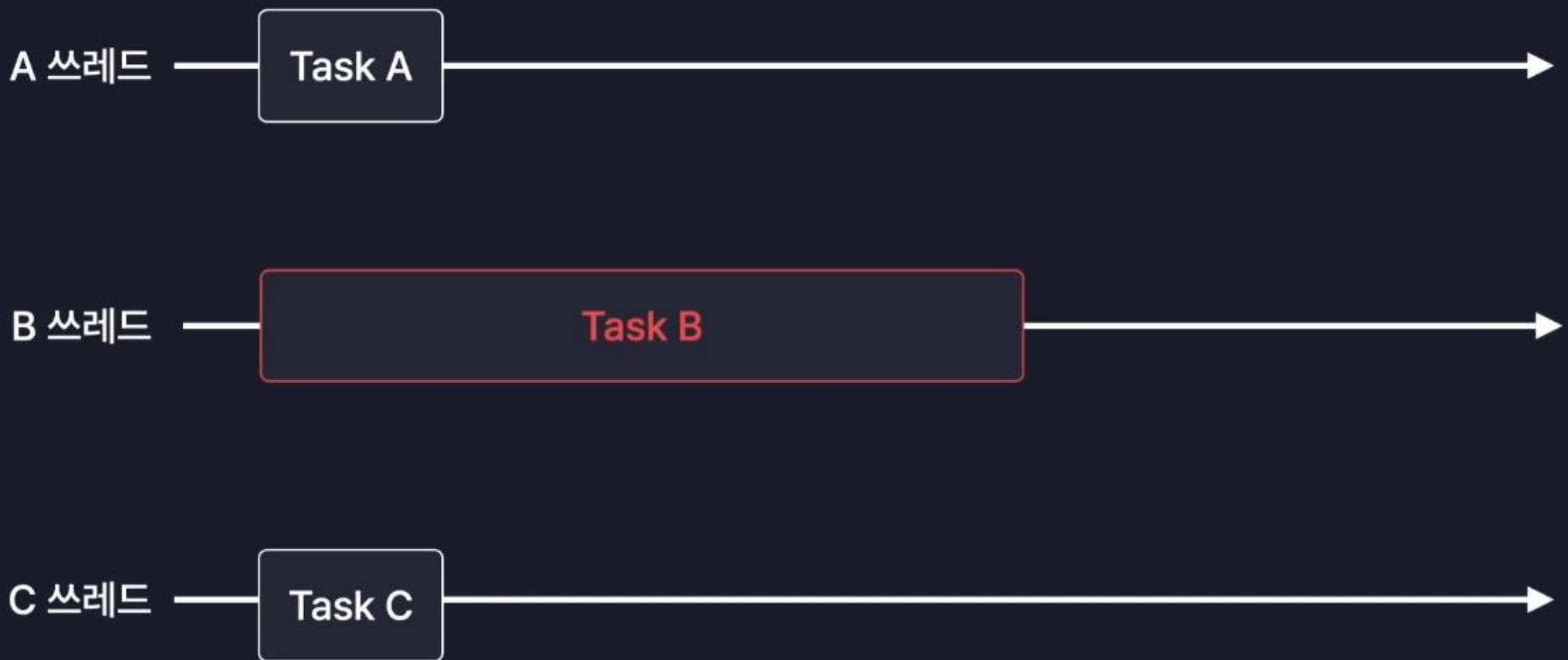
비동기란 무엇일까?



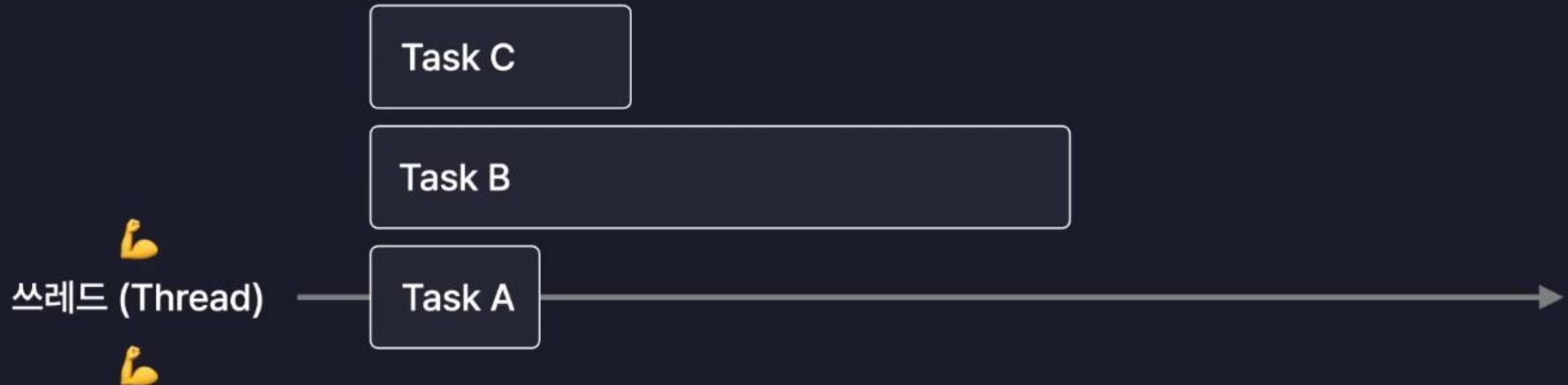
비동기란 무엇일까?



의문



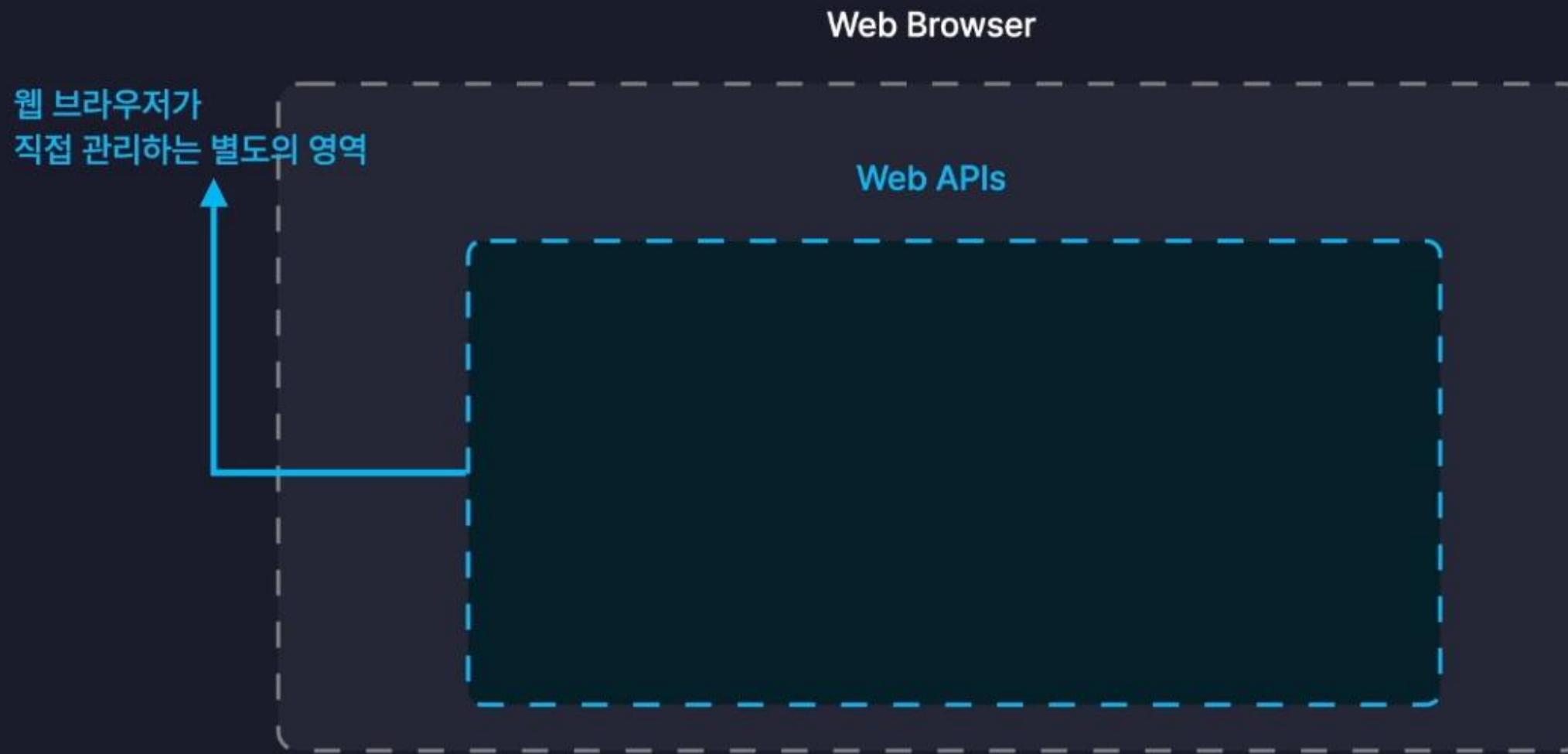
🤔 어떻게 동시에 작업을 처리하는거지?



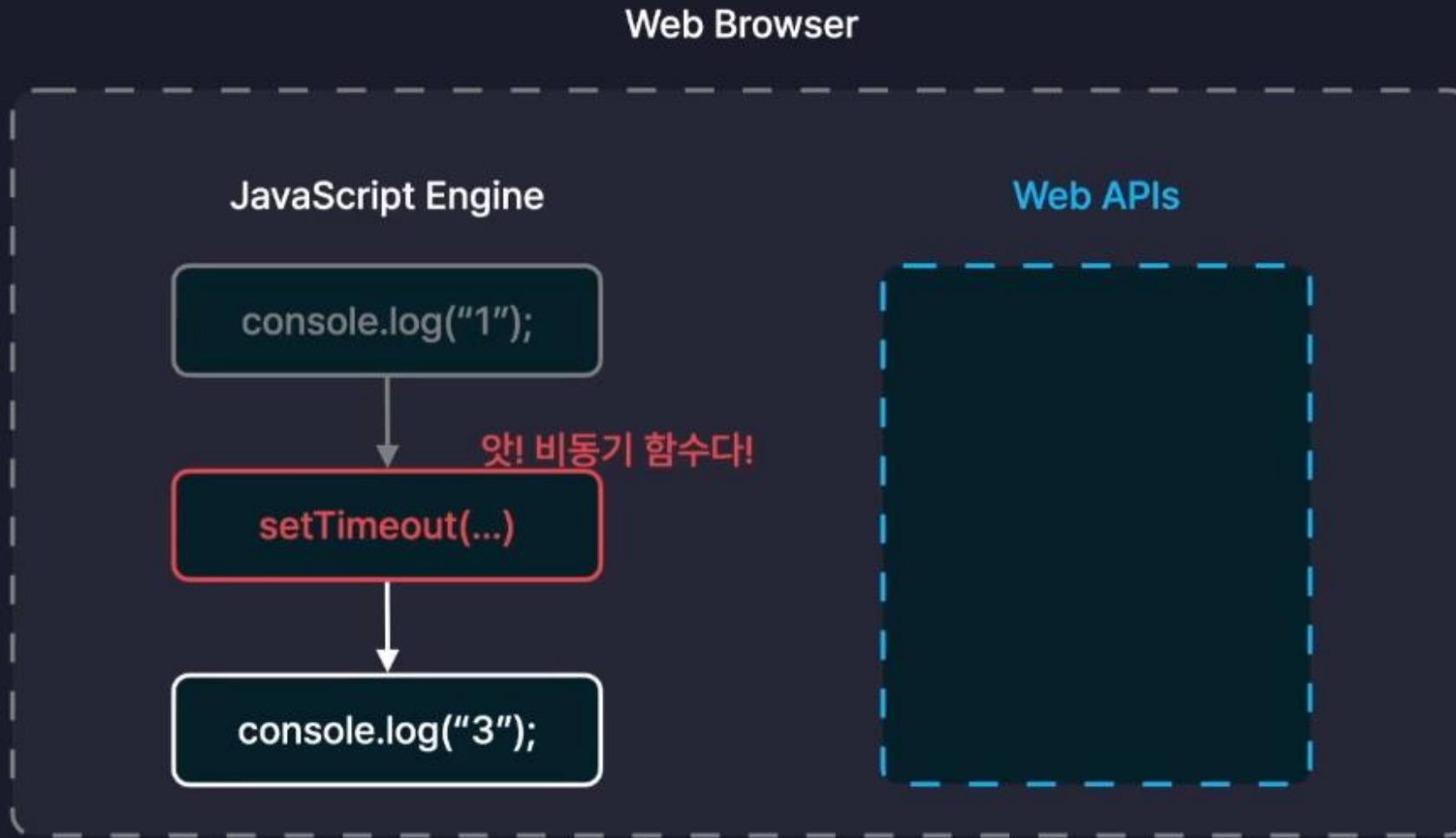
🤔 어떻게 동시에 작업을 처리하는거지?



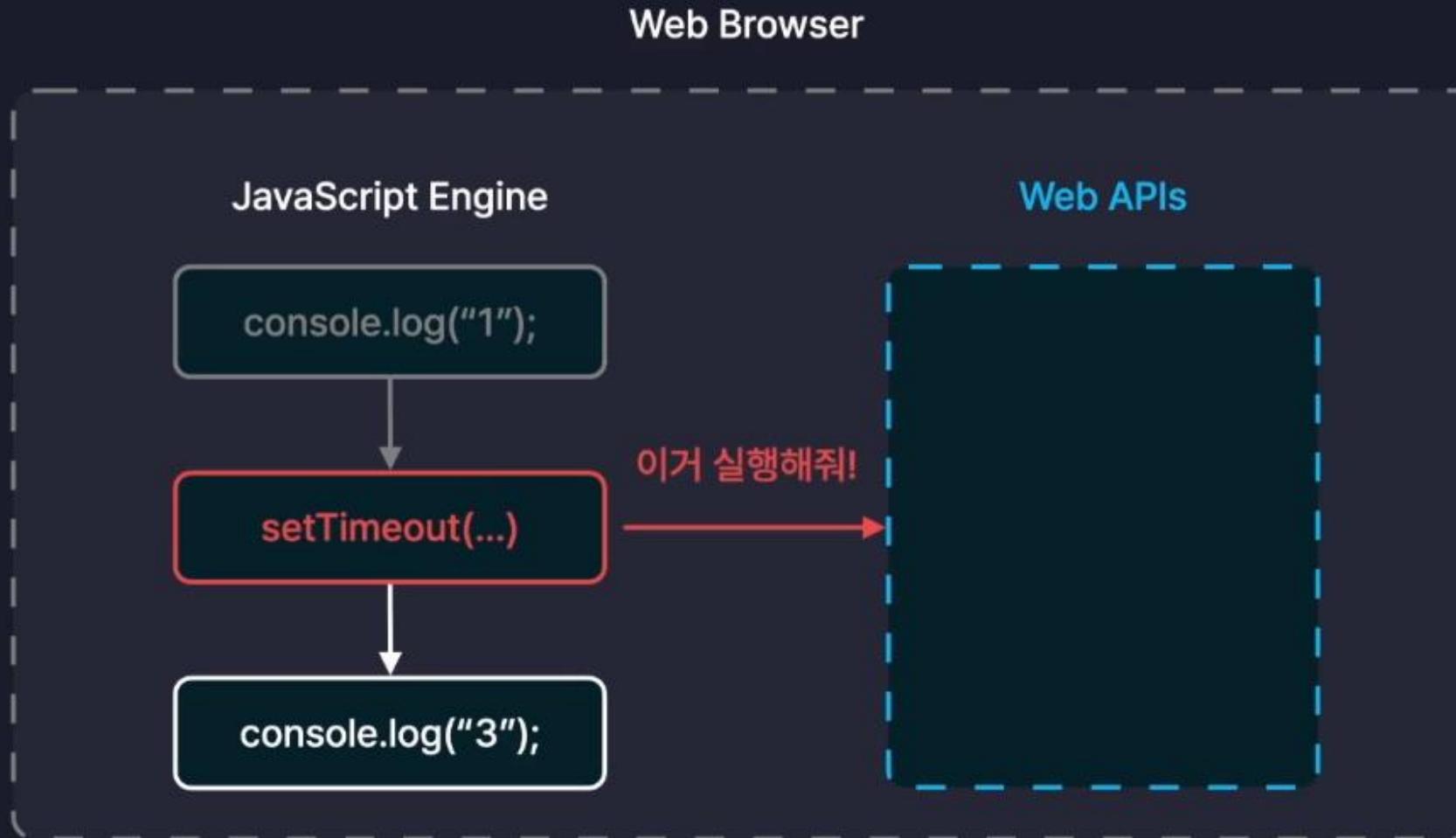
💡 어떻게 동시에 작업을 처리하는거지?



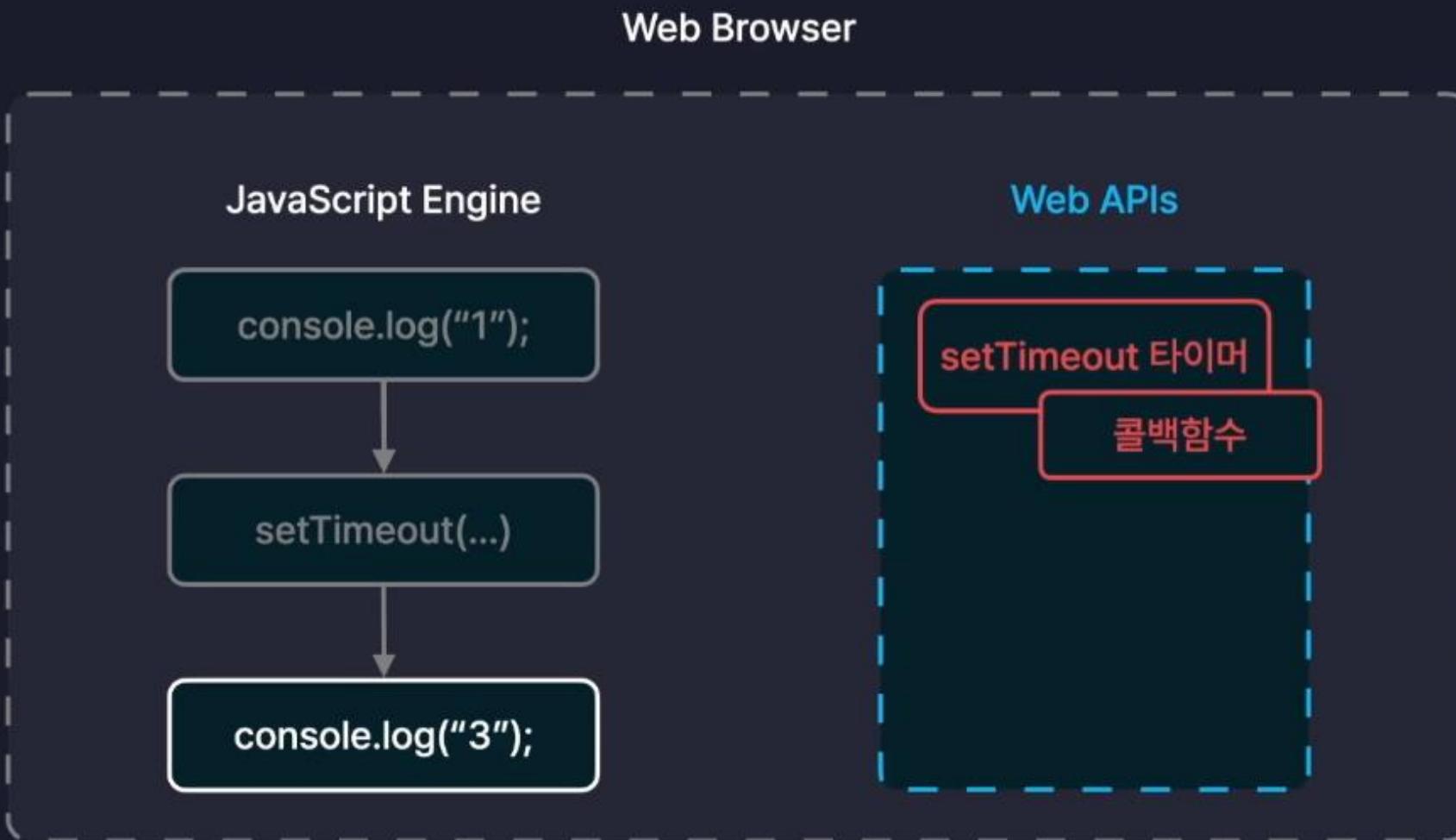
JavaScript의 비동기 처리



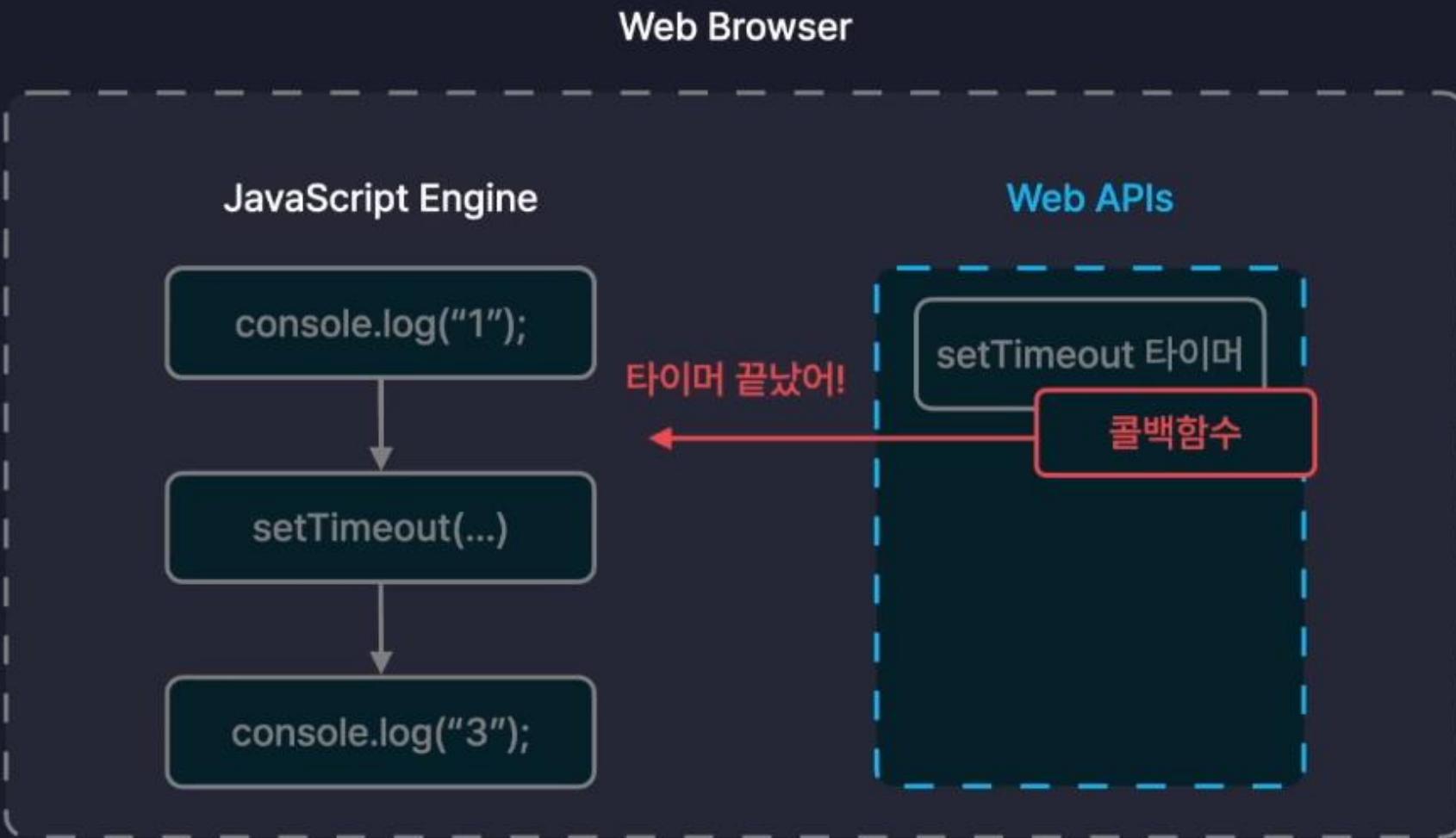
JavaScript의 비동기 처리



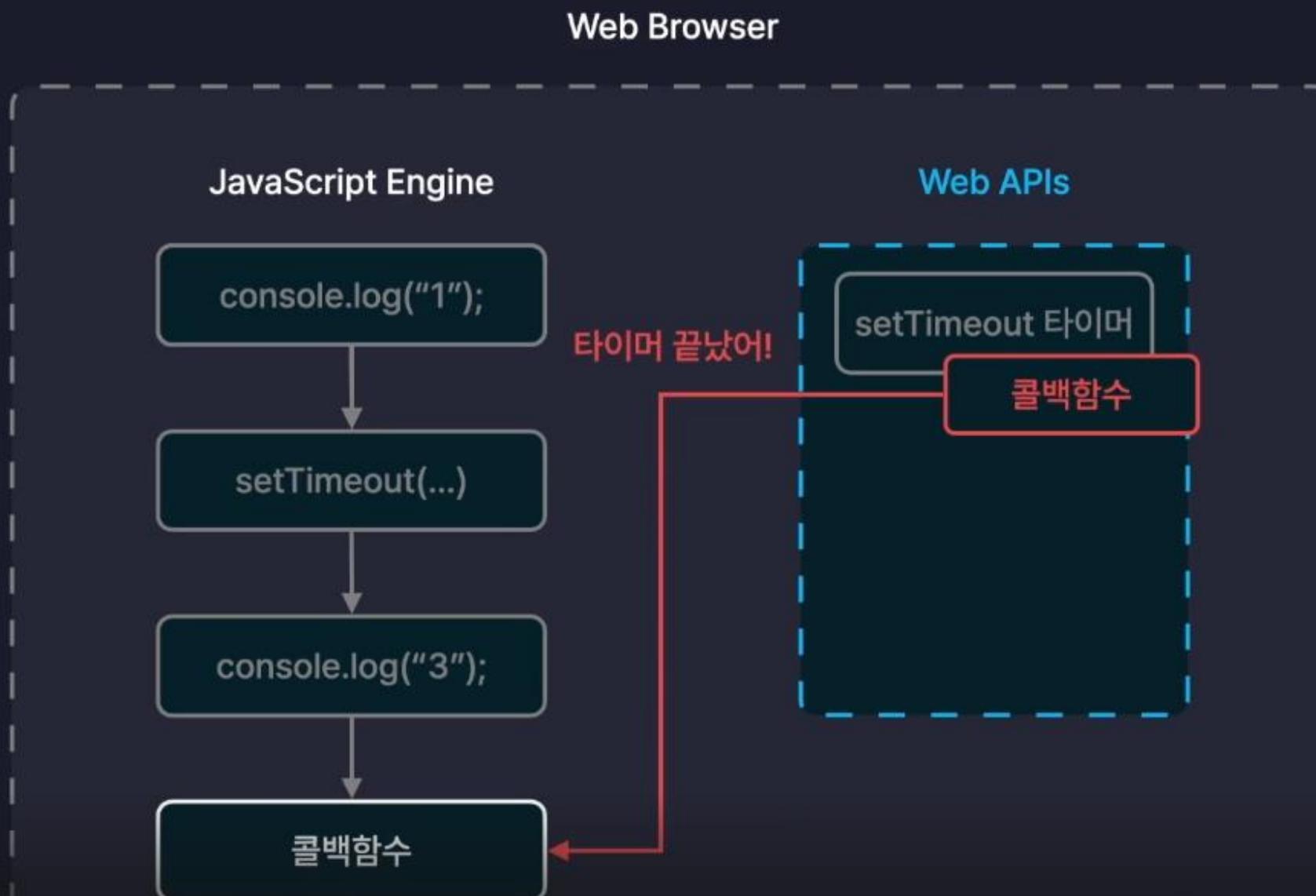
JavaScript의 비동기 처리



JavaScript의 비동기 처리



JavaScript의 비동기 처리



한입 크기로 잘라먹는

비동기 작업 처리하기 1. 콜백 함수

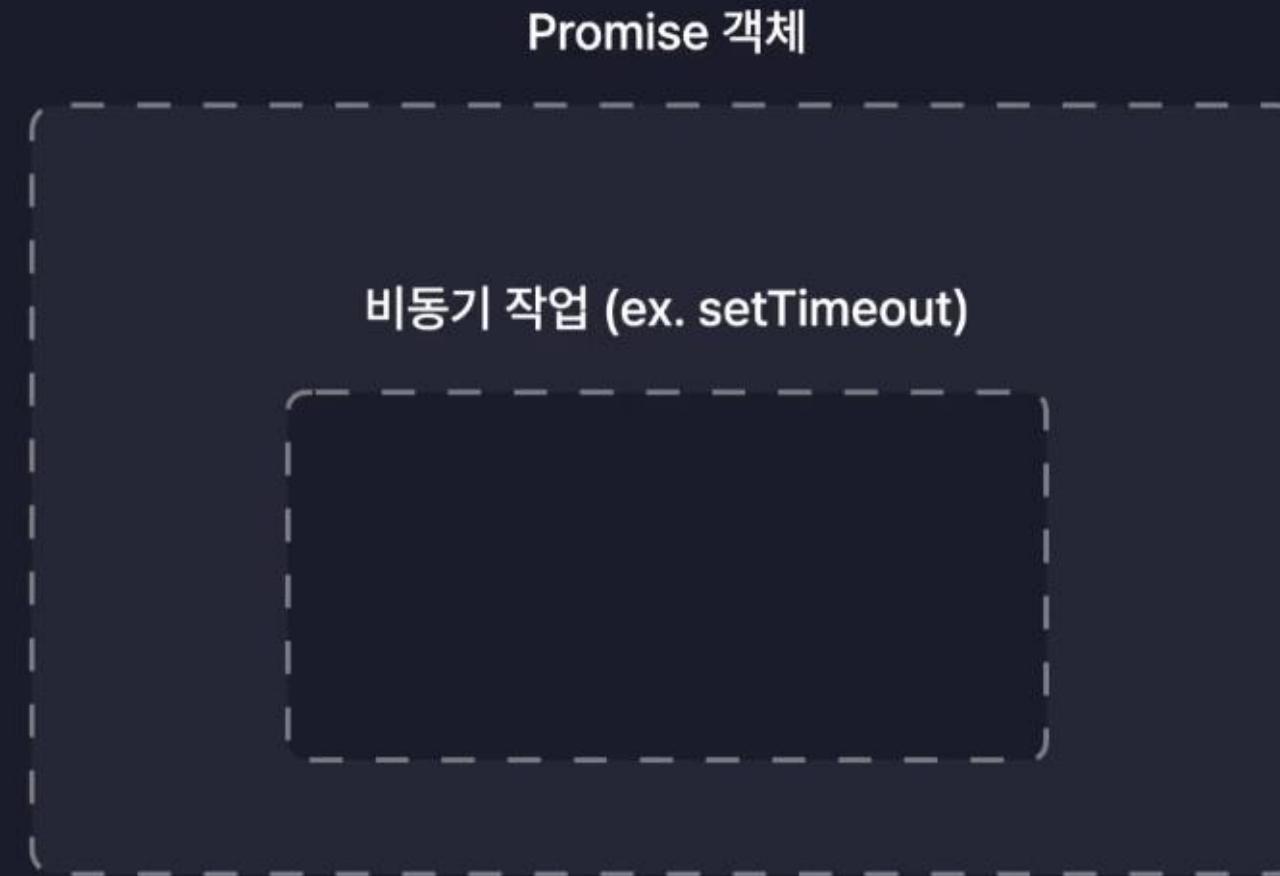
한입 크기로 잘라먹는

비동기 작업 처리하기 2. Promise

Promise 란?

비동기 작업을
효율적으로 처리할 수 있도록 도와주는
자바스크립트의 내장 객체

Promise는 비동기 작업을 감싸는 객체이다



Promise는 비동기 작업을 감싸는 객체이다

Promise 객체

비동기 작업 (ex. setTimeout)

Promise의 효능

- 비동기 작업 실행
- 비동기 작업 상태 관리
- 비동기 작업 결과 저장
- 비동기 작업 병렬 실행
- 비동기 작업 다시 실행
- 기타 등등 ...

Promise는 비동기 작업을 감싸는 객체이다

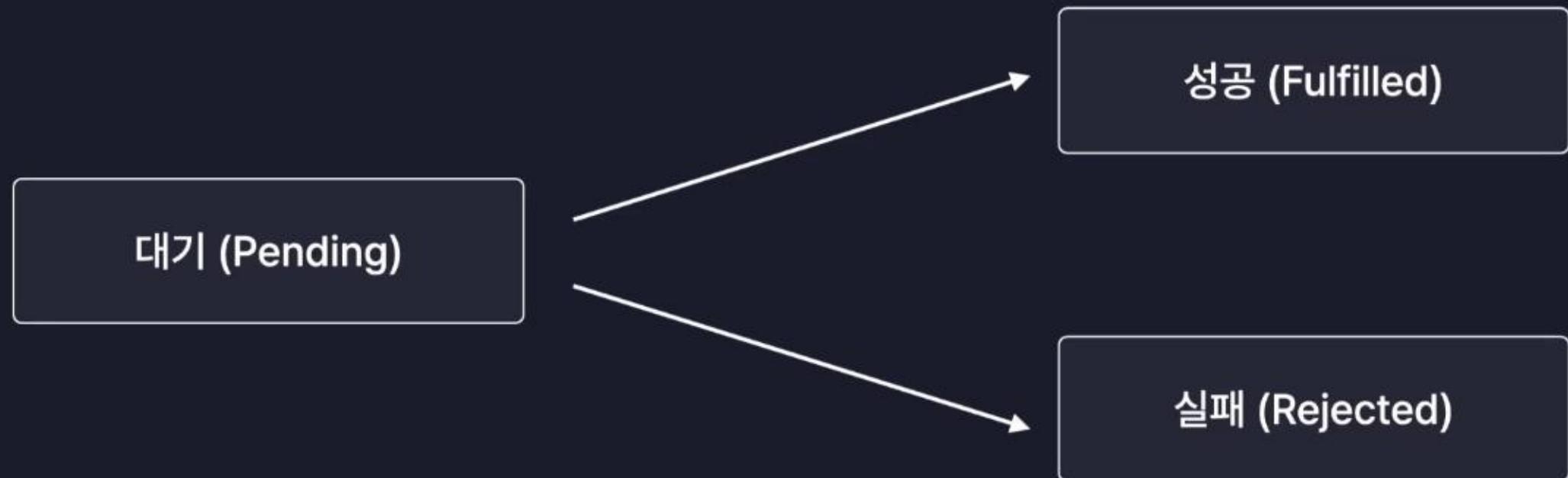
Promise 객체

비동기 작업 (ex. setTimeout)

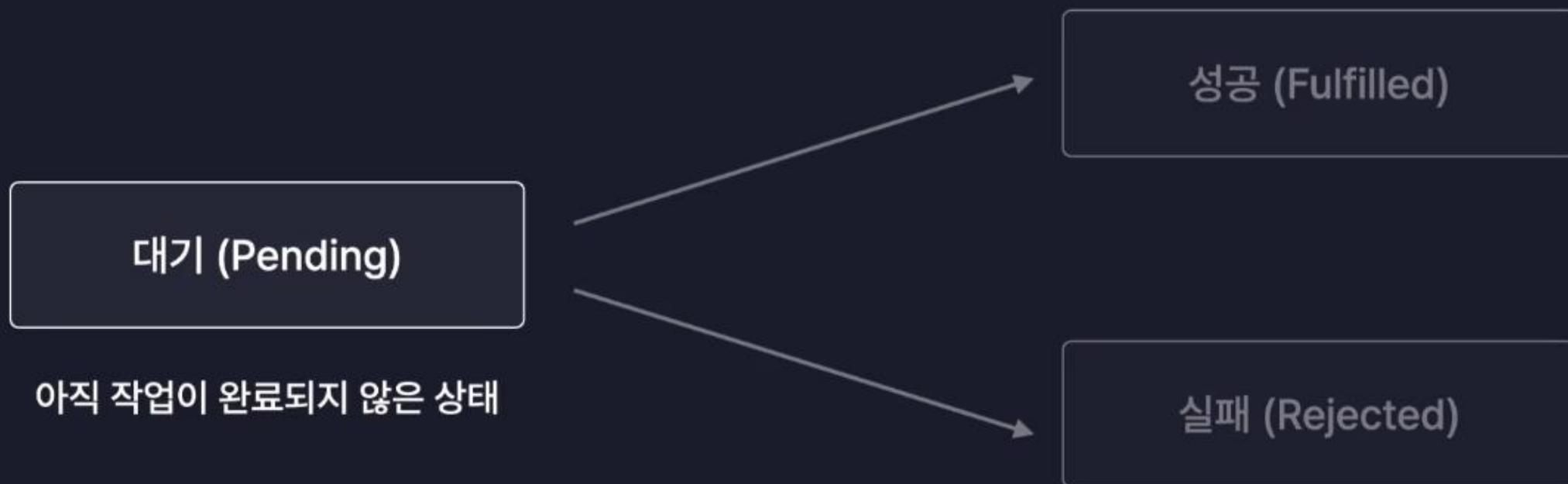
Promise의 효능

- 비동기 작업 실행
- 비동기 작업 상태 관리
- 비동기 작업 결과 저장
- 비동기 작업 병렬 실행
- 비동기 작업 다시 실행
- 기타 등등 ...

Promise의 3가지 상태



Promise의 3가지 상태



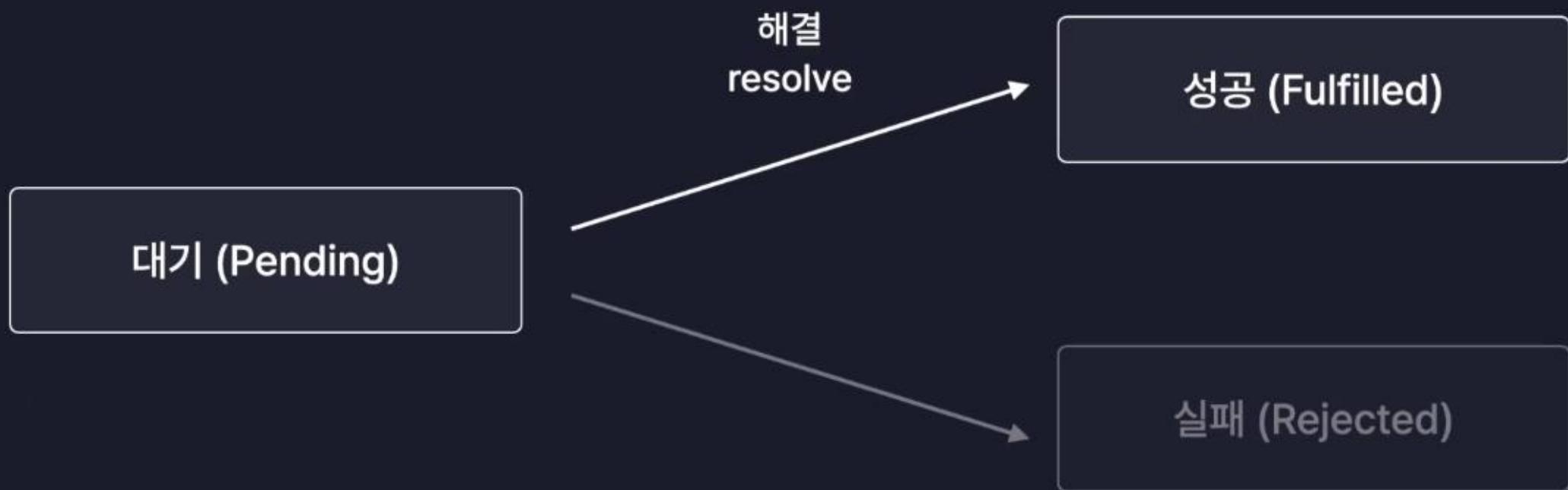
Promise의 3가지 상태



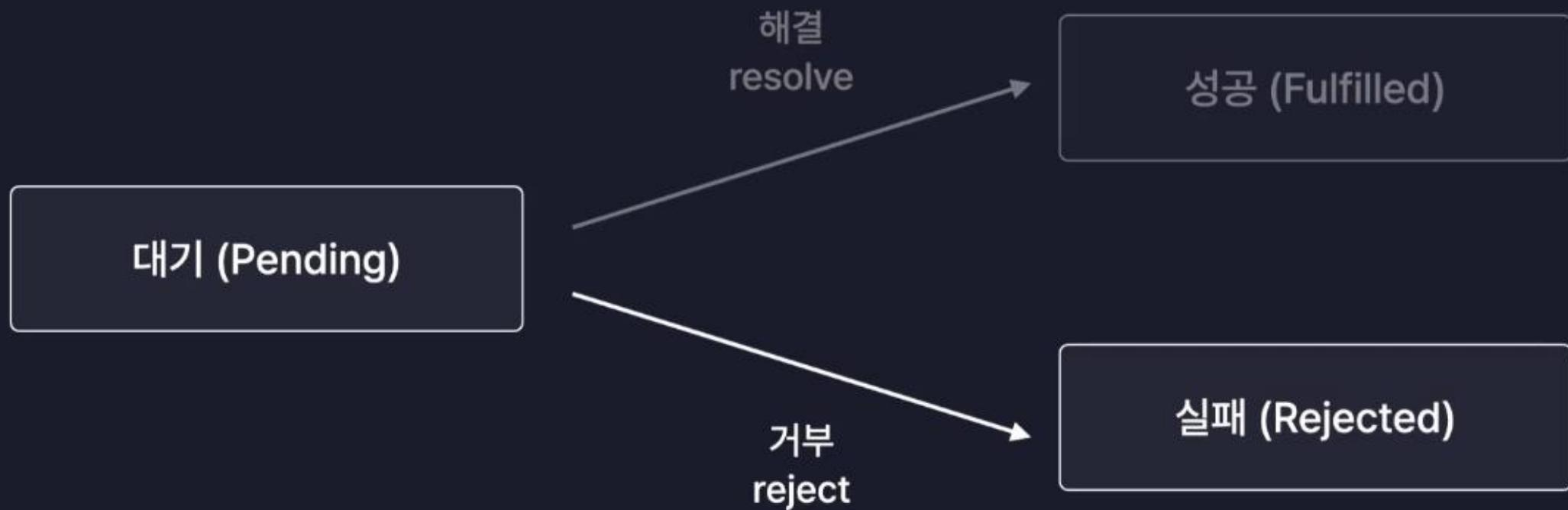
Promise의 3가지 상태



Promise의 3가지 상태



Promise의 3가지 상태



Promise의 3가지 상태



Promise의 3가지 상태

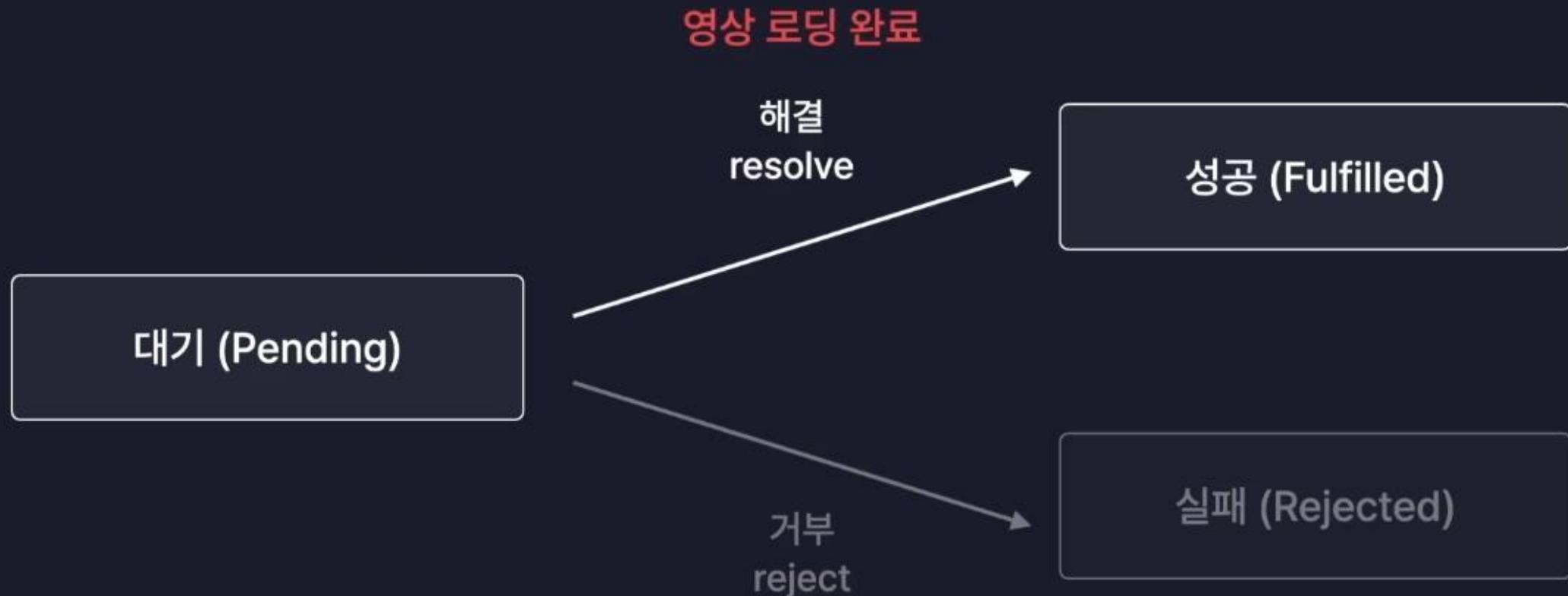


유튜브 영상 로딩 중

대기 (Pending)



Promise의 3가지 상태



Promise의 3가지 상태

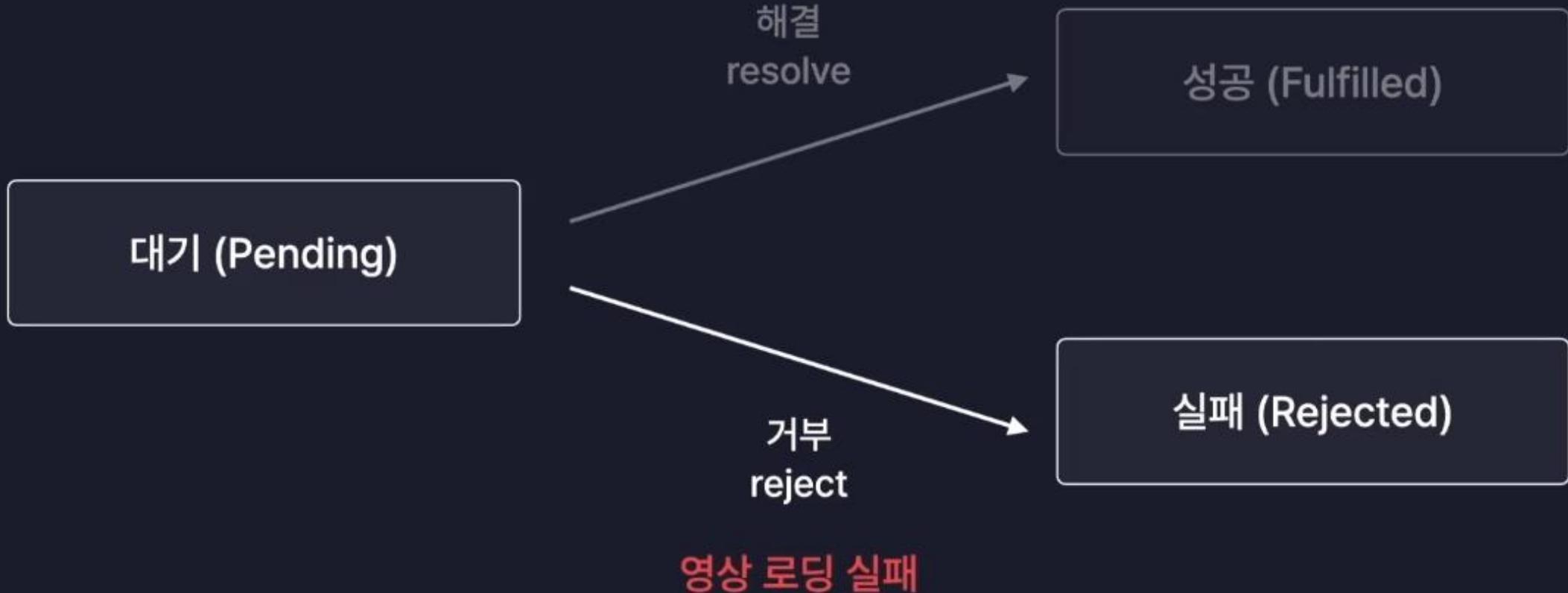


시청 가능

성공 (Fulfilled)

실패 (Rejected)

Promise의 3가지 상태



Promise의 3가지 상태



Connect to the internet

You're offline. Check your connection.

Retry

reject

성공 (Fulfilled)

시청 불가능 한 상태

실패 (Rejected)

한입 크기로 잘라먹는

비동기 작업 처리하기 3. `async/await`

한입 크기로 잘라먹는

Node.js를 소개합니다

갑자기 왜 Node.js를 배워야 하나요?



갑자기 웬 Node.js?

갑자기 왜 Node.js를 배워야 하나요?

Node.js 기반



React.js

Next.js

Vue.js

Svelte

...

Node.js란?



Node.js

JavaScript 실행 환경(Run Time)

= 구동기

Node.js란?



Node.js란?



Node.js란?

* <https://nodejs.org>



Node.js®는 Chrome V8 JavaScript 엔진으로 빌드된 JavaScript 런타임입니다.

Security releases now available

다운로드 - macOS

18.16.1 LTS

20.3.1 현재 버전

Node.js란?

* <https://nodejs.org>



Node.js®는 Chrome V8 JavaScript 엔진으로 빌드된 JavaScript 런타임입니다.

Security releases now available

다운로드 - macOS

18.16.1 LTS

20.3.1 현재 버전

Node.js는 왜 만든걸까?



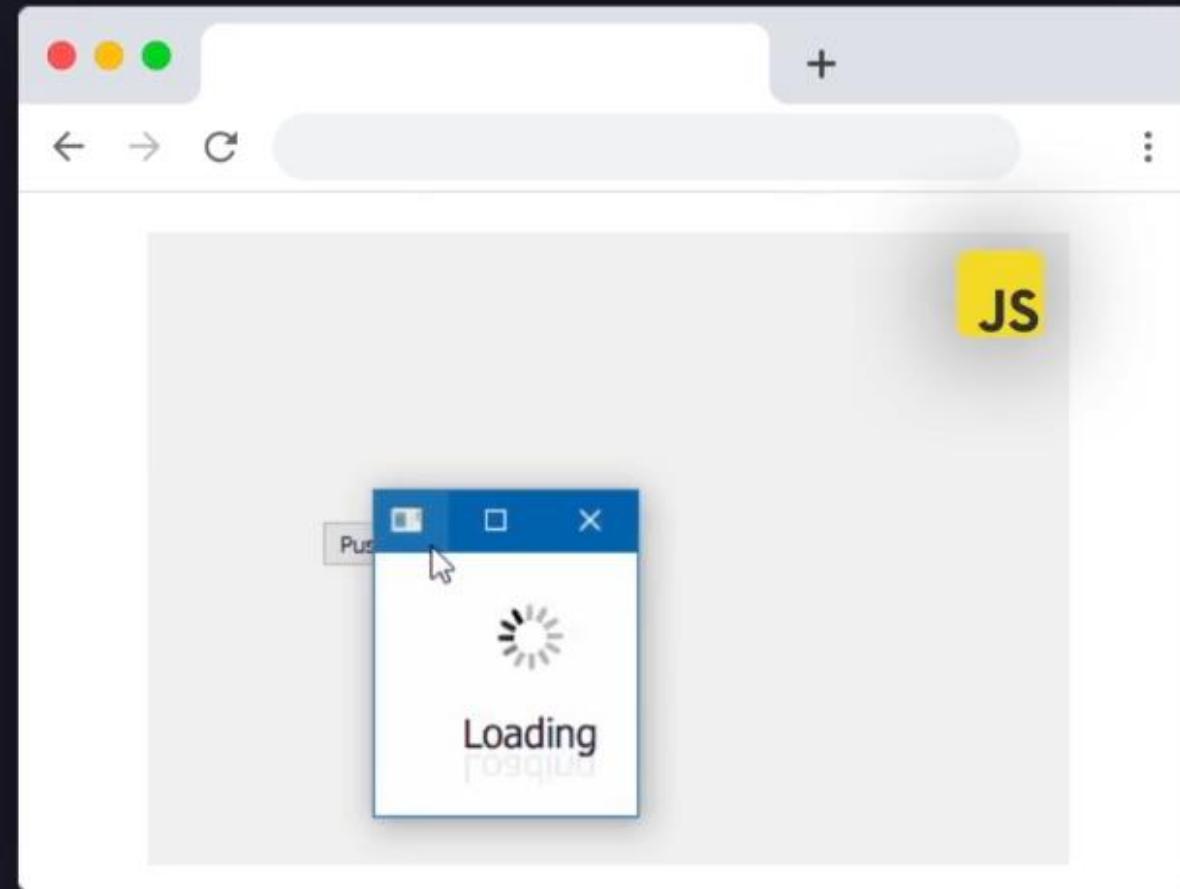
이거 왜 만들었을까?

JavaScript 히스토리



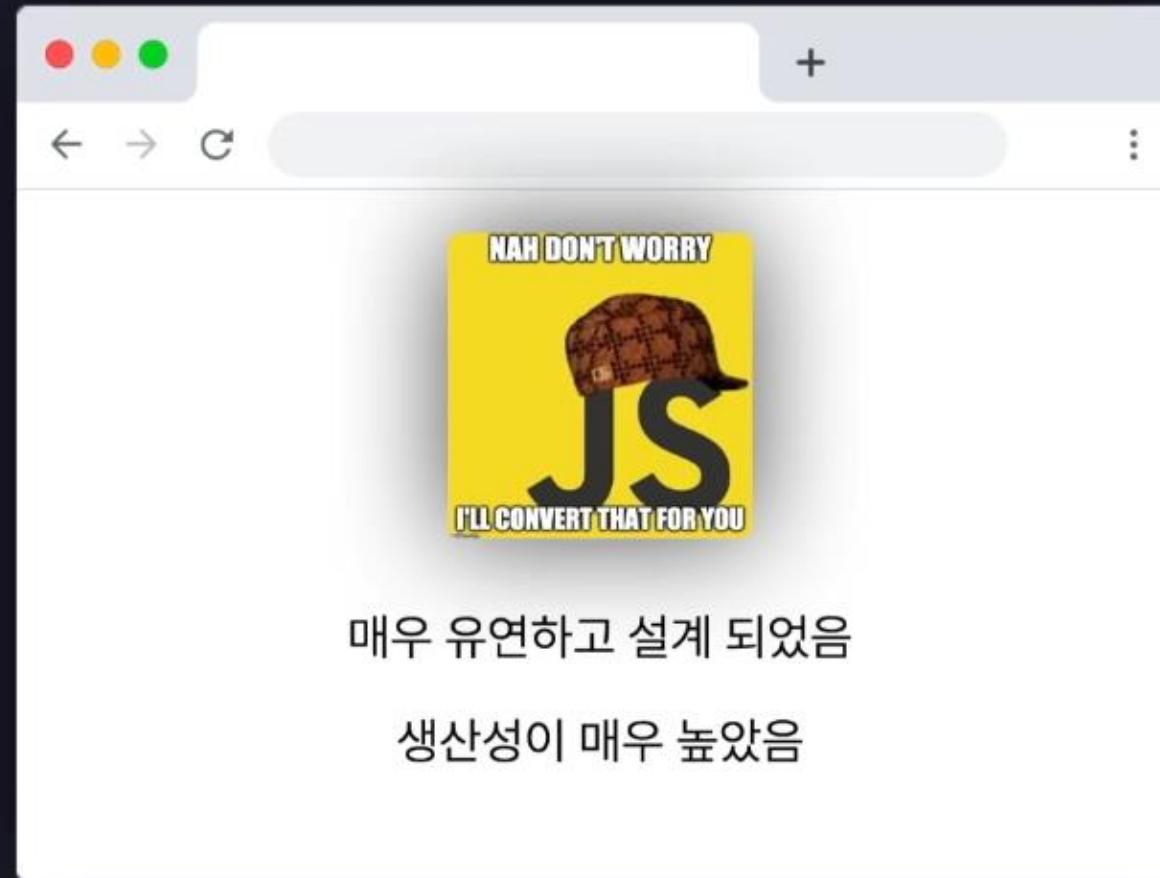
JavaScript 히스토리

웹 브라우저



JavaScript 히스토리

웹 브라우저

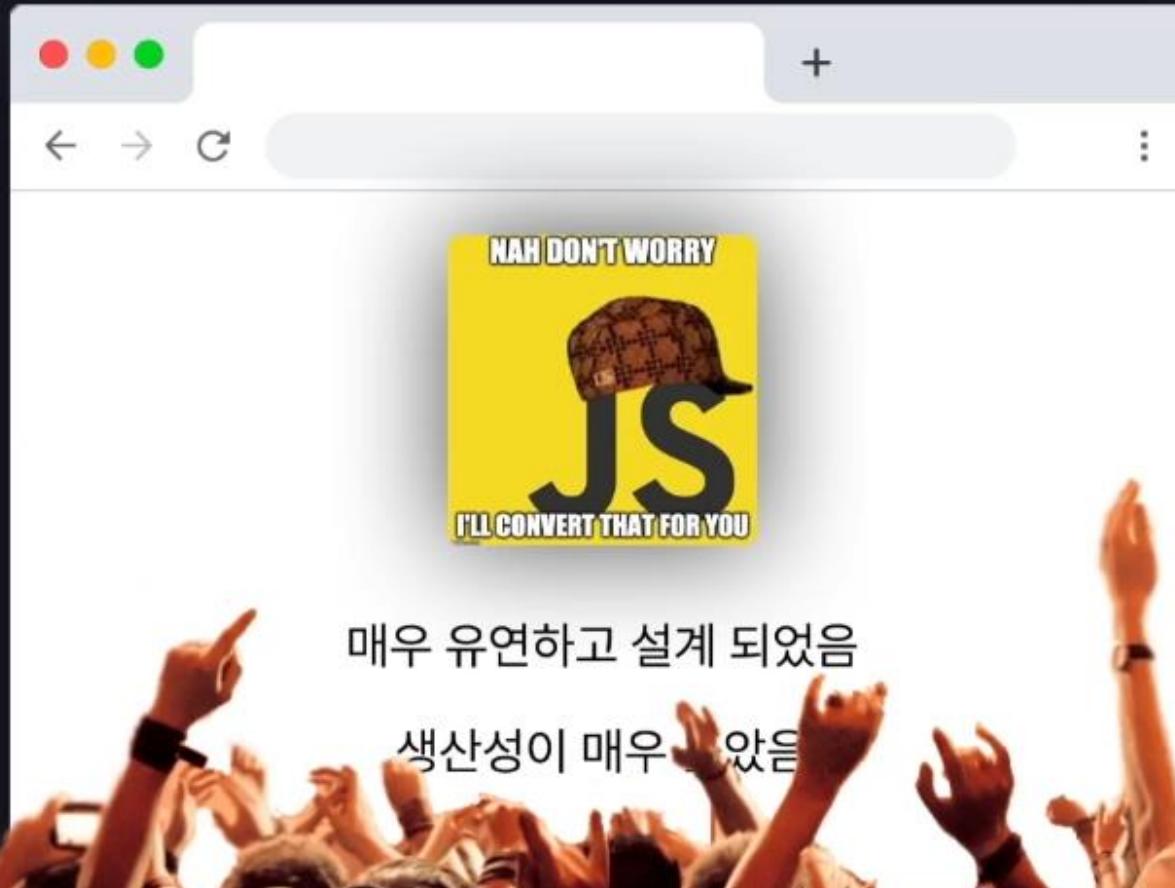


매우 유연하고 설계 되었음

생산성이 매우 높았음

JavaScript 히스토리

웹 브라우저 밖에서도
쓰게 해주세요!



매우 유연하고 설계 되었음

생산성이 매우 높았음

자바스크립트로
웹 서버도 만들고 싶어요!!

JavaScript 히스토리



Node.js

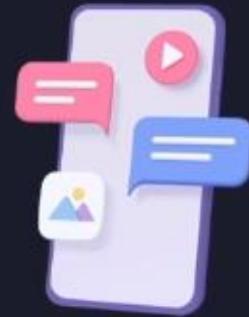


JavaScript 히스토리



웹 서버

넷플릭스,
에어비엔비,
등등 ...



모바일 앱

페이스북,
인스타그램,
등등 ...



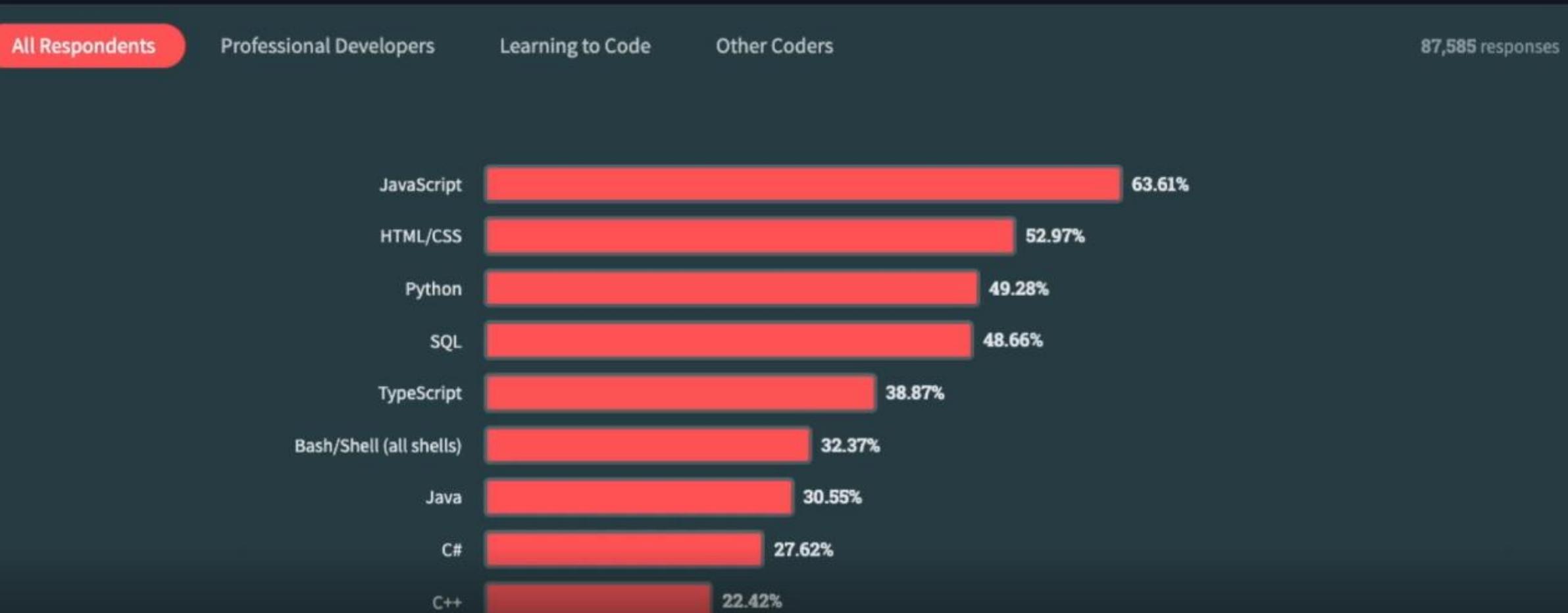
데스크톱 앱

슬랙,
디스코드,
등등 ...

JavaScript 히스토리

프로그래밍 언어 사용량 조사 결과

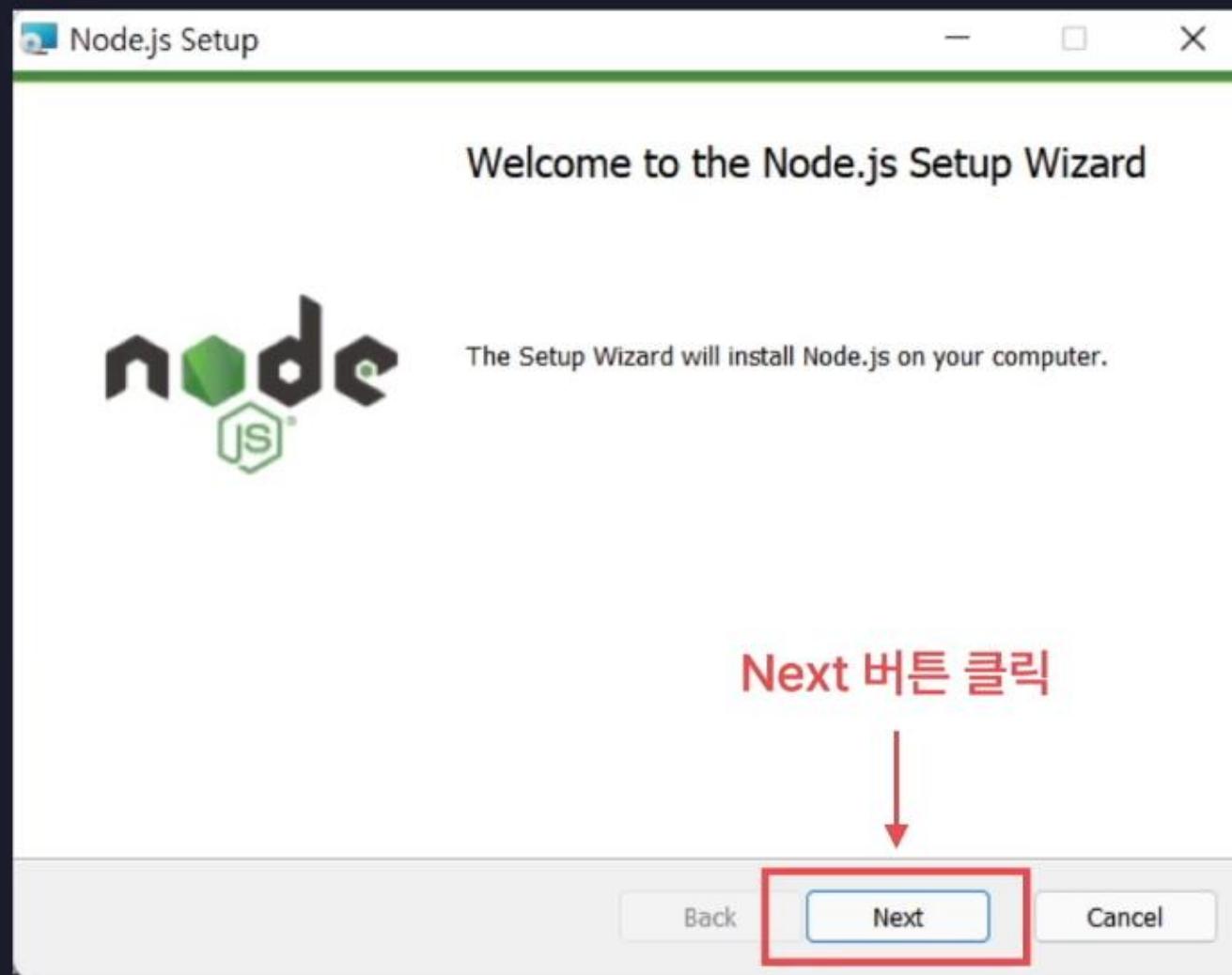
* 2023 Stackoverflow Developer Survey



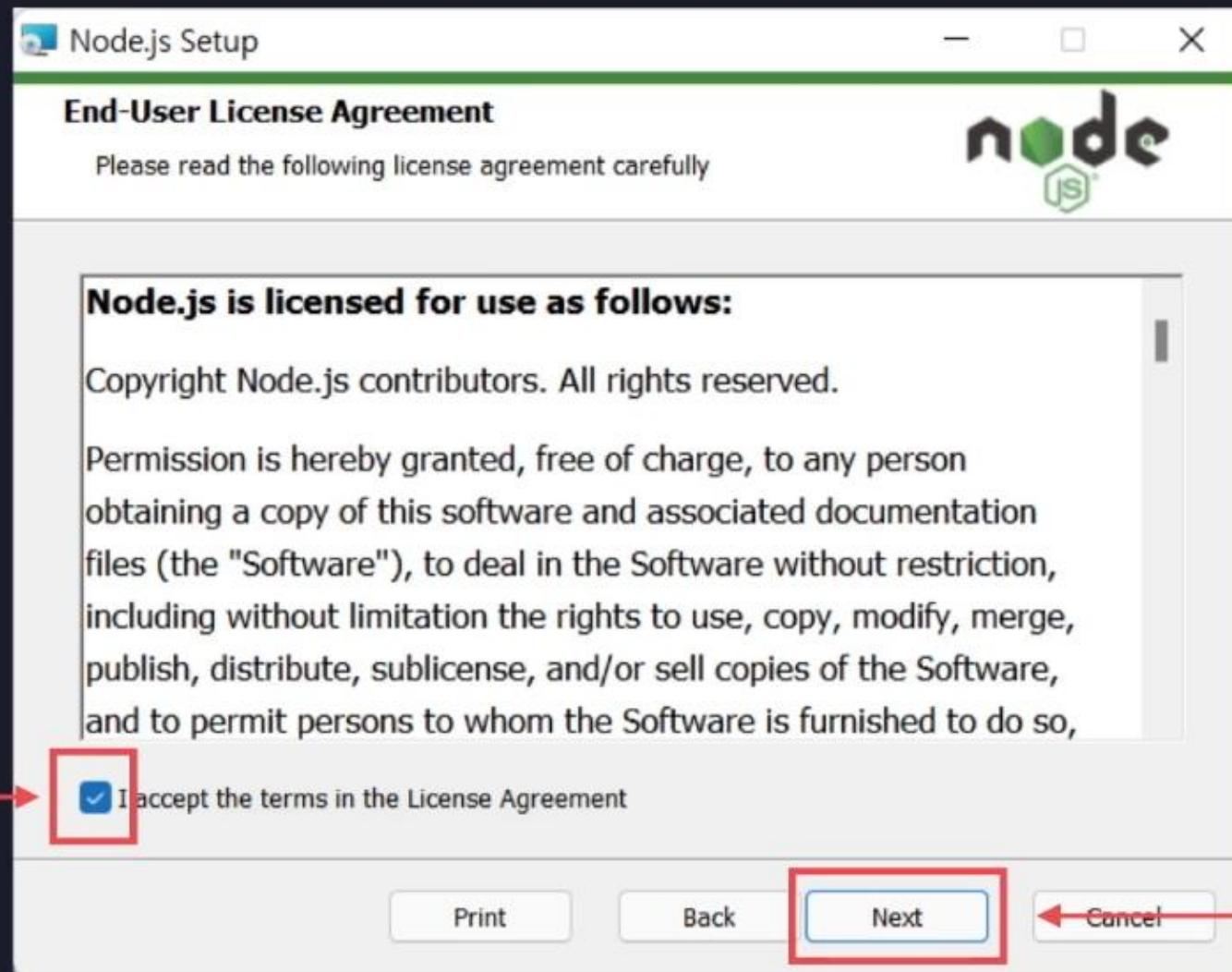
한입 크기로 잘라먹는

Node.js 설치하기

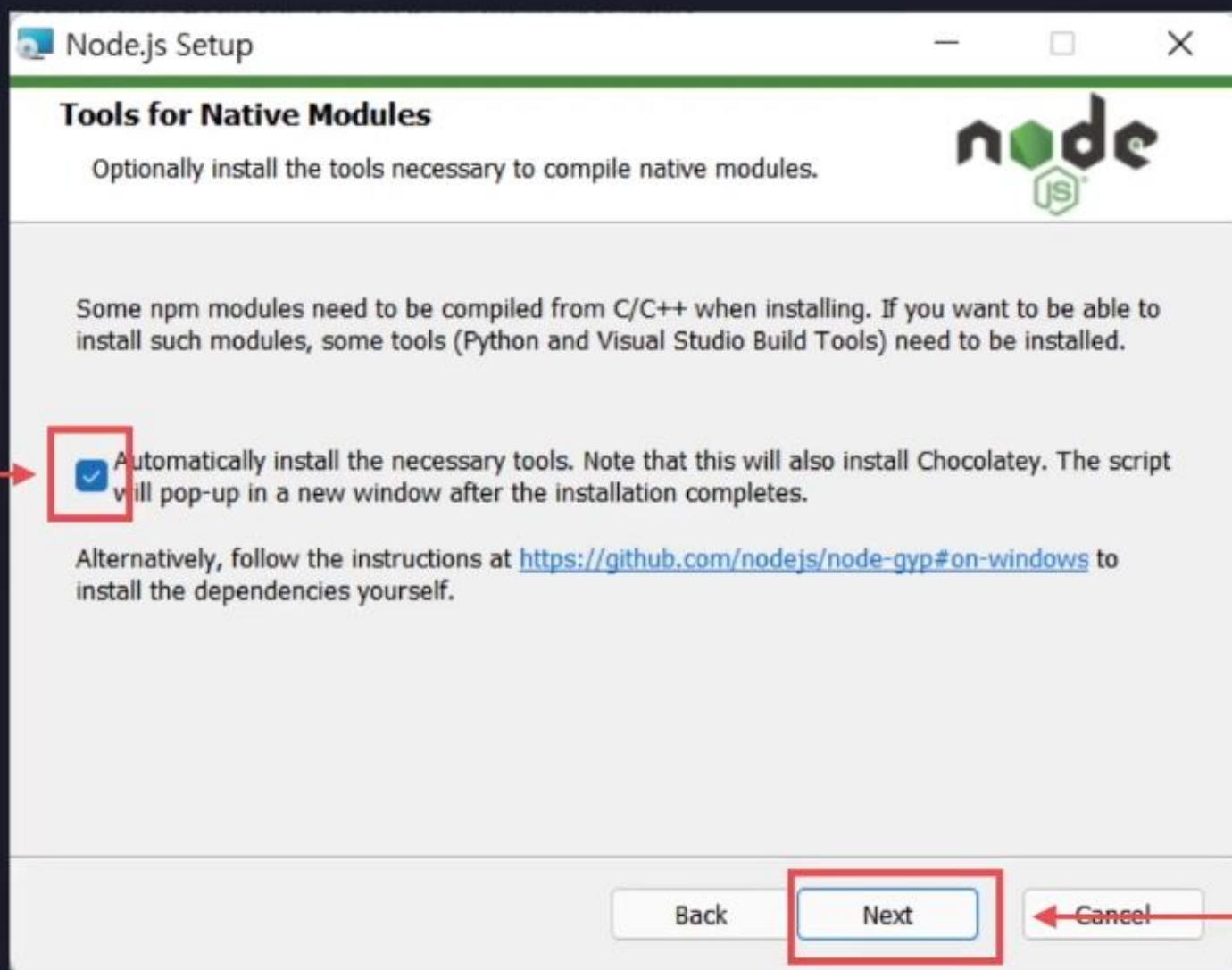
Node.js 설치하기 (Windows)



Node.js 설치하기 (Windows)

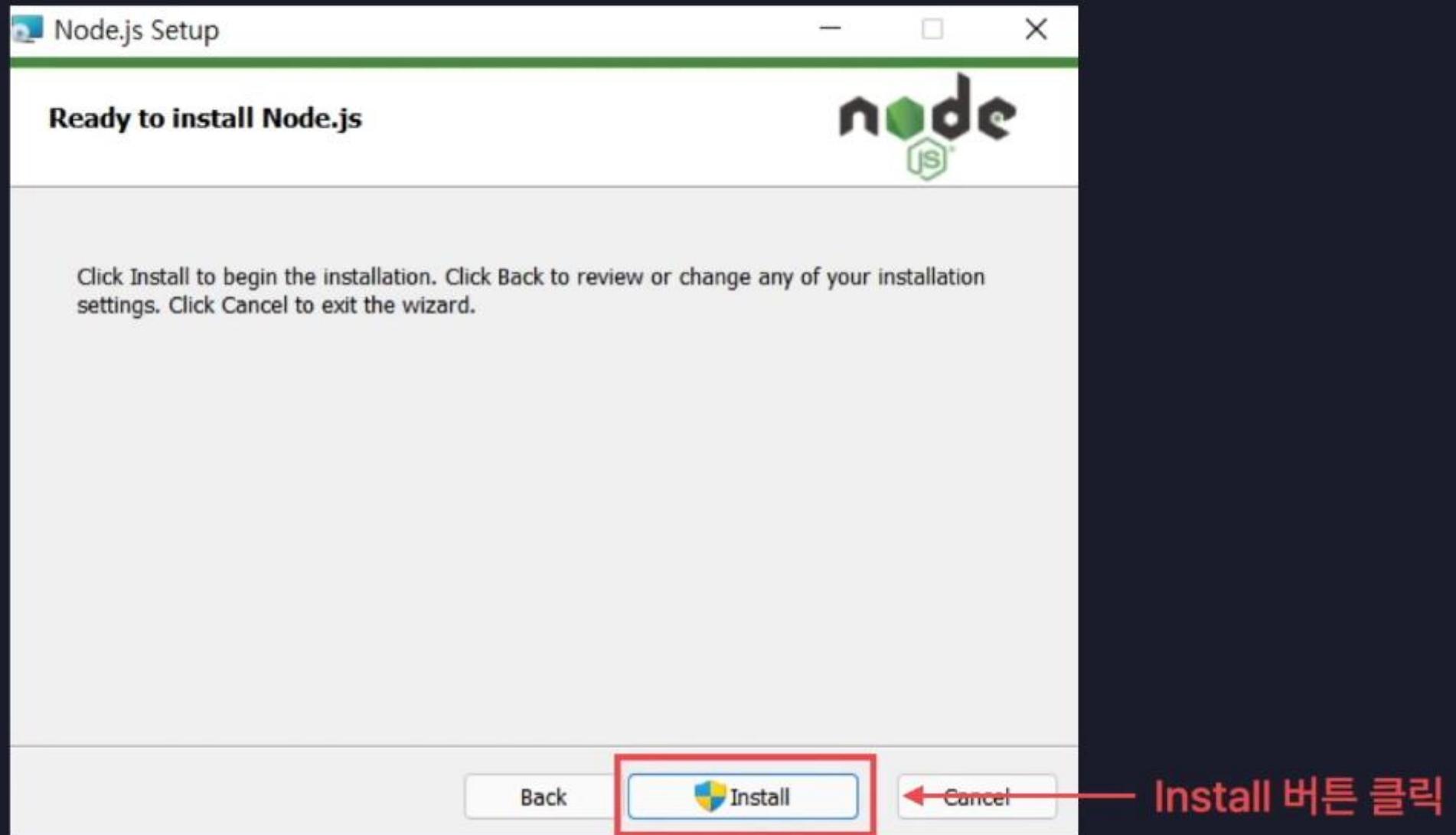


Node.js 설치하기 (Windows)

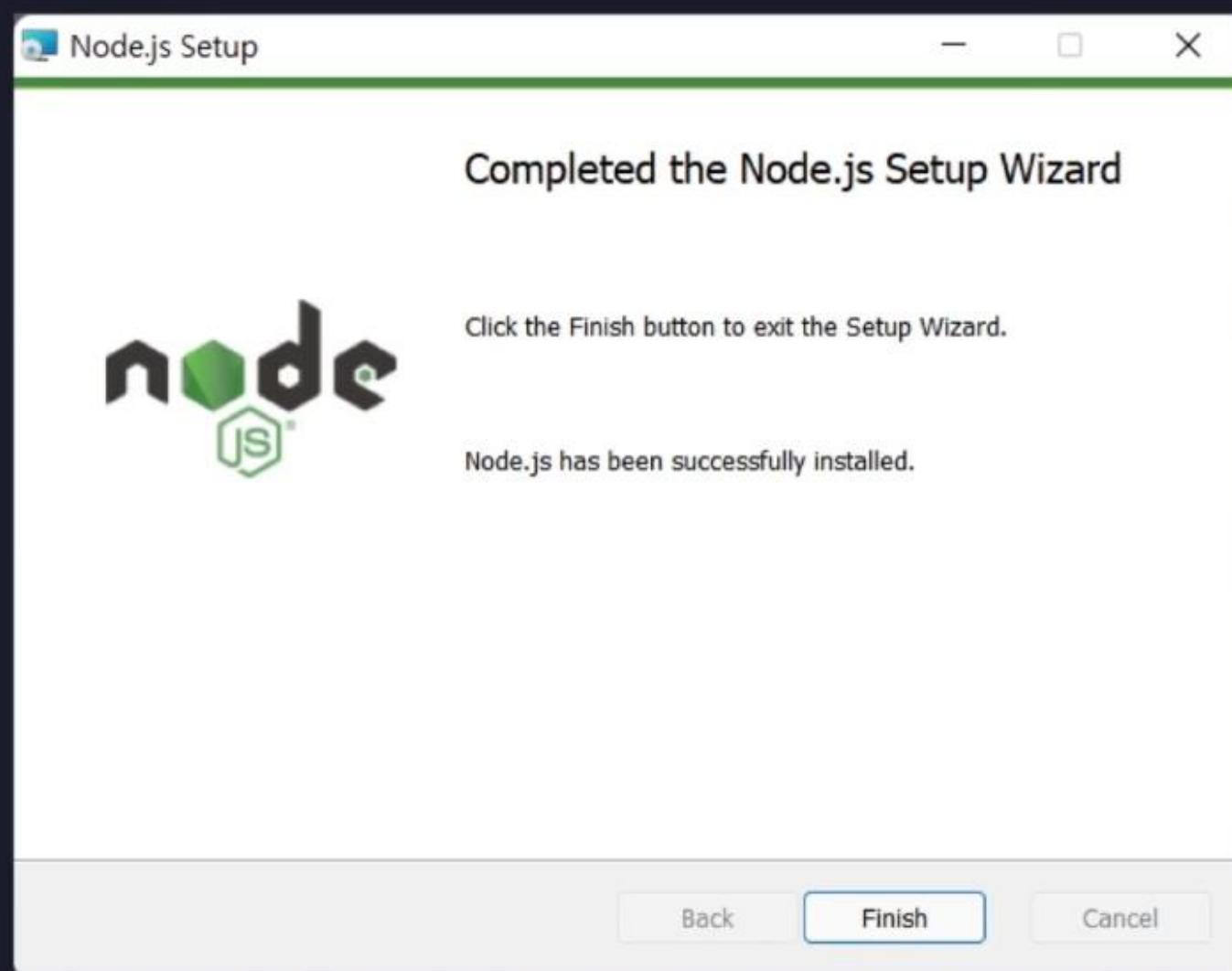


Next 버튼 클릭

Node.js 설치하기 (Windows)



Node.js 설치하기 | (Windows)



한입 크기로 잘라먹는

Node.js 사용하기

프로젝트(Project)

프로젝트

특정 목적을 갖는 프로그램의 단위



프로젝트(Project)

패키지

Node.js에서 사용하는 프로그램의 단위



한입 크기로 잘라먹는

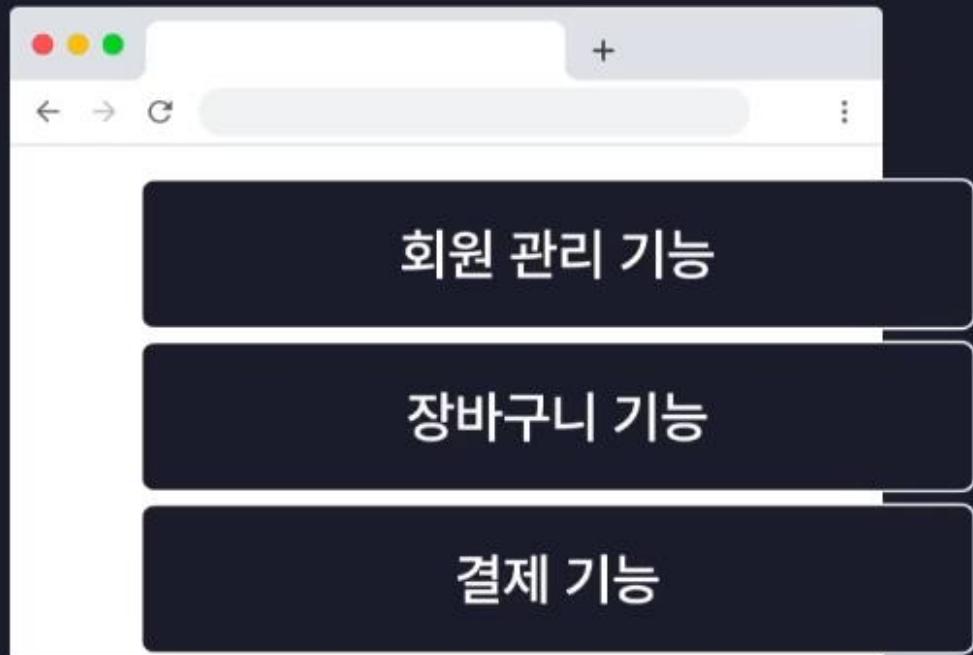
Node.js 모듈 시스템 이해하기

모듈 시스템(Module System)이란?

모듈 시스템

모듈을 다루는 시스템

모듈(Module)이란?



모듈(Module)이란?



```
import React, { useEffect, useReducer, useRef } from "react";
import "./App.css";
import { BrowserRouter, Route, Routes } from "react-router-dom";

import Home from "./pages/Home";
import New from "./pages/New";
import Edit from "./pages/Edit";
import Diary from "./pages/Diary";

const reducer = (state, action) => {
  let newState = [];
  switch (action.type) {
    case "INIT": {
      return action.data;
    }
    case "CREATE": {
      newState = [action.data, ...state];
      break;
    }
    case "REMOVE": {
      newState = state.filter(it => it.id !== action.targetId);
      break;
    }
    case "EDIT": {
      newState = state.map((it) =>
        it.id === action.data.id ? { ...action.data } : it
      );
      break;
    }
    default:
      return state;
  }

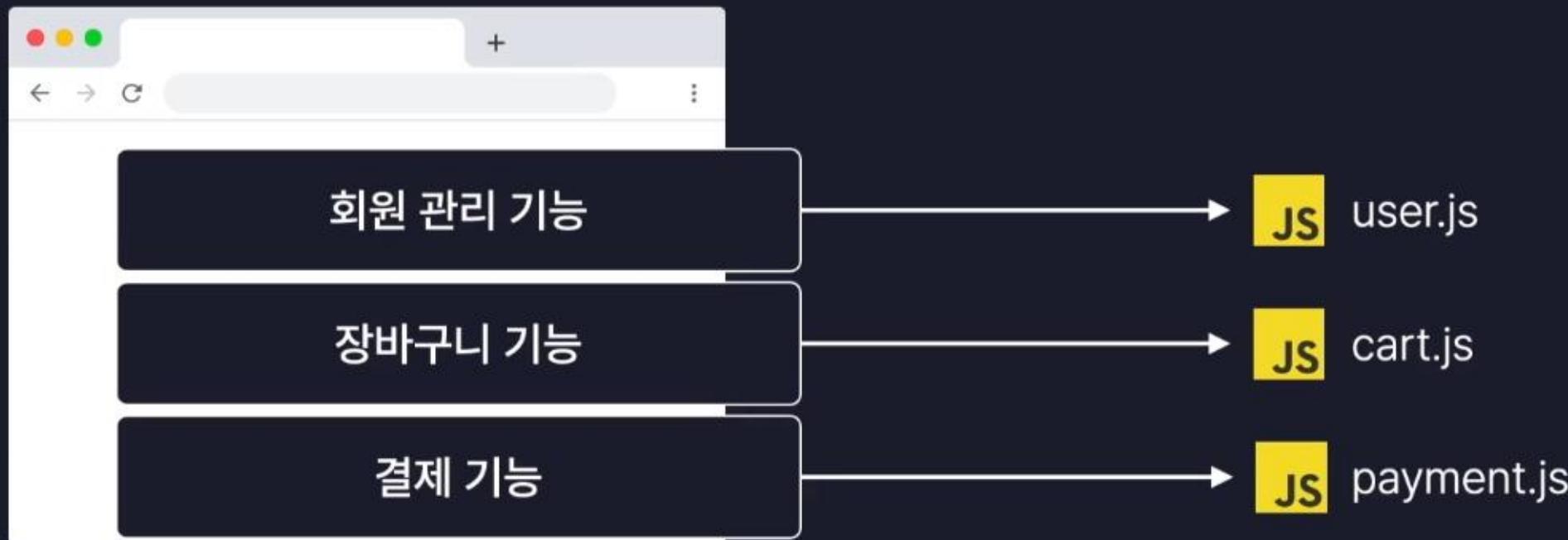
  localStorage.setItem("diary", JSON.stringify(newState));
  return newState;
};

export const DiaryStateContext = React.createContext();
export const DiaryDispatchContext = React.createContext();

function App() {
  const [data, dispatch] = useReducer(reducer, []);
  useEffect(() => {
    const localData = localStorage.getItem("diary");
    if (localData) {
      const diaryList = JSON.parse(localData).sort(
        (a, b) => parseInt(b.id) - parseInt(a.id)
      );
      if (diaryList.length >= 1) {
        dispatch({ type: "CREATE", data: diaryList[0] });
      }
    }
  }, [dispatch]);
  return (
    <DiaryStateContext.Provider value={{ data, dispatch }}>
      <DiaryDispatchContext.Provider value={dispatch}>
        <BrowserRouter>
          <Routes>
            <Route path="/" element={<Home />} />
            <Route path="/new" element={<New />} />
            <Route path="/edit/:id" element={<Edit />} />
            <Route path="/diary" element={<Diary />} />
          </Routes>
        </BrowserRouter>
      </DiaryDispatchContext.Provider>
    </DiaryStateContext.Provider>
  );
}

export default App;
```

모듈(Module)이란?



모듈(Module)이란?



모듈 시스템(Module System)이란?

모듈 시스템

모듈을 생성하고, 불러오고, 사용하는 등의
모듈을 다루는 다양한 기능을 제공하는 시스템

 user 모듈

 cart 모듈

 payment 모듈

모듈 시스템(Module System)이란?

JavaScript의 모듈 시스템

Common JS (CJS)

ES Module (ESM)

AMD

UMD

...

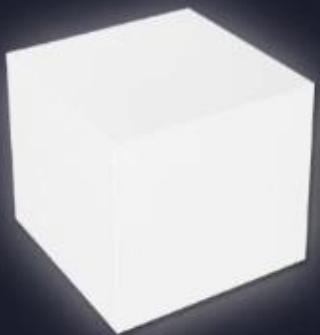
한입 크기로 잘라먹는

Node.js 라이브러리 사용하기

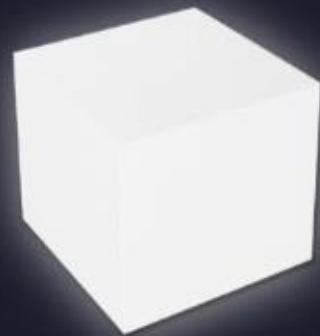
라이브러리란?

프로그램을 개발할 때 필요한 다양한 기능들을 미리 만들어 모듈화 해 놓은 것

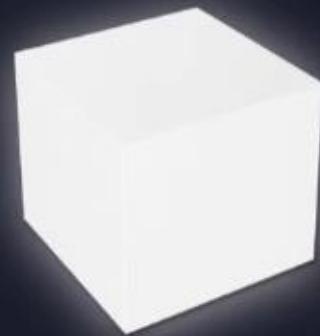
날짜 라이브러리



수학 라이브러리



그래픽 라이브러리



라이브러리란?

프로그램을 개발할 때 필요한 다양한 기능들을 미리 만들어 모듈화 해 놓은 것

dayjs TS

1.11.8 • Public • Published 24 days ago

[Readme](#) [Code](#) Beta [0 Dependencies](#) [12,120 Dependents](#) [121 Versions](#)

English | 简体中文 | 日本語 | Português Brasileiro | 한국어 | Español (España) | Русский | Türkçe |
한국어

DAY.JS

Fast 2kB alternative to Moment.js with the same modern API

gzip size 2.95 kB npm v1.11.8 build no longer available coverage 100% license MIT

Android	Firefox	Chrome	IE	iPhone	Edge	Safari
4.4 <small>△</small> ✓	4 <small>△</small> 7 ✓	26 <small>△</small> 7 ✓	9 <small>△</small> 7 ✓	9.3 <small>△</small> 10.11 ✓	13 <small>△</small> 10 ✓	8 <small>△</small> 10.10 ✓
6.0 <small>△</small> ✓	60 <small>△</small> 7 ✓	66 <small>△</small> 7 ✓	11 <small>△</small> 10 ✓	11 <small>△</small> 10.12 ✓	17 <small>△</small> 10 ✓	11 <small>△</small> u ✓

Day.js is a minimalist JavaScript library that parses, validates, manipulates, and displays dates and times for modern browsers with a largely Moment.js-compatible API. If you use Moment.js, you already know how to use Day.js.

Install
`> npm i dayjs`

Repository [github.com/iamkun/dayjs](#)

Homepage [day.js.org](#)

Weekly Downloads 16,484,889 

Version 1.11.8 License MIT

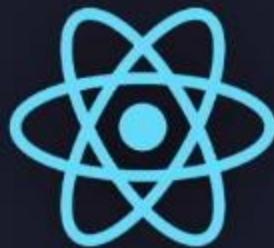
Unpacked Size 653 kB Total Files 447

Issues 609 Pull Requests 141

한입 크기로 잘라먹는

React.js를 소개합니다

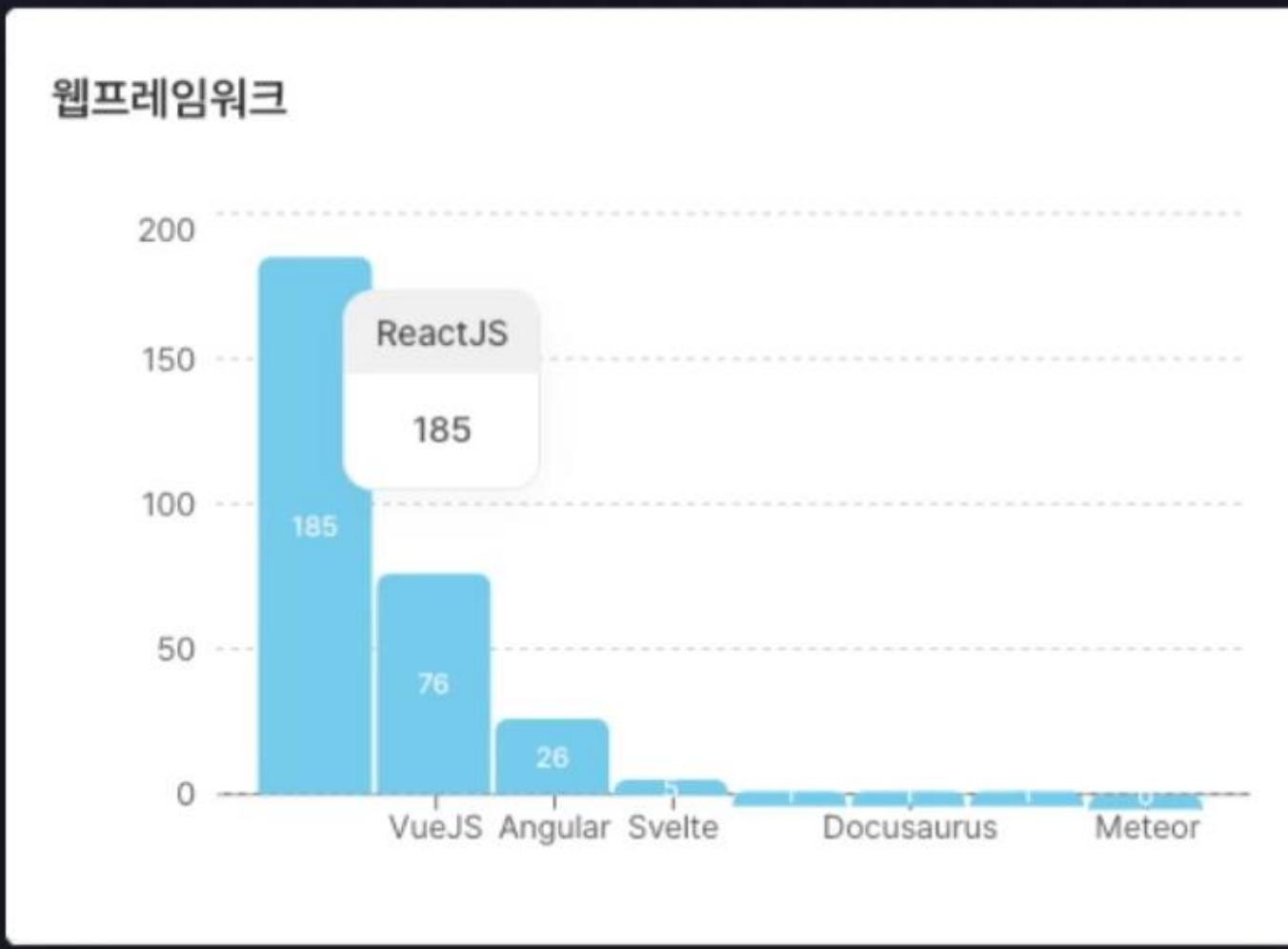
React.js란?



React.js로 만들어진 서비스들

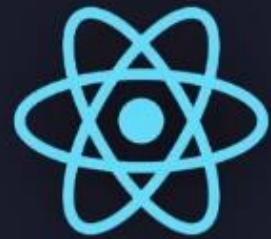


React.js는 얼마나 인기있을까?



* 출처: 코드너리(codenary.co.kr)

React의 기술적인 특징

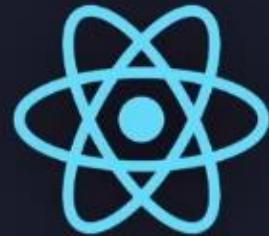


1. 컴포넌트를 기반으로 UI를 표현한다

2. 화면 업데이트 구현이 쉽다

3. 화면 업데이트가 빠르게 처리된다

React의 기술적인 특징 1. 컴포넌트를 기반으로 UI를 표현한다



1. 컴포넌트를 기반으로 UI를 표현한다

2. 화면 업데이트 구현이 쉽다

3. 화면 업데이트가 빠르게 처리된다

React의 기술적인 특징 1. 컴포넌트를 기반으로 UI를 표현한다



React의 기술적인 특징 1. 컴포넌트를 기반으로 UI를 표현한다

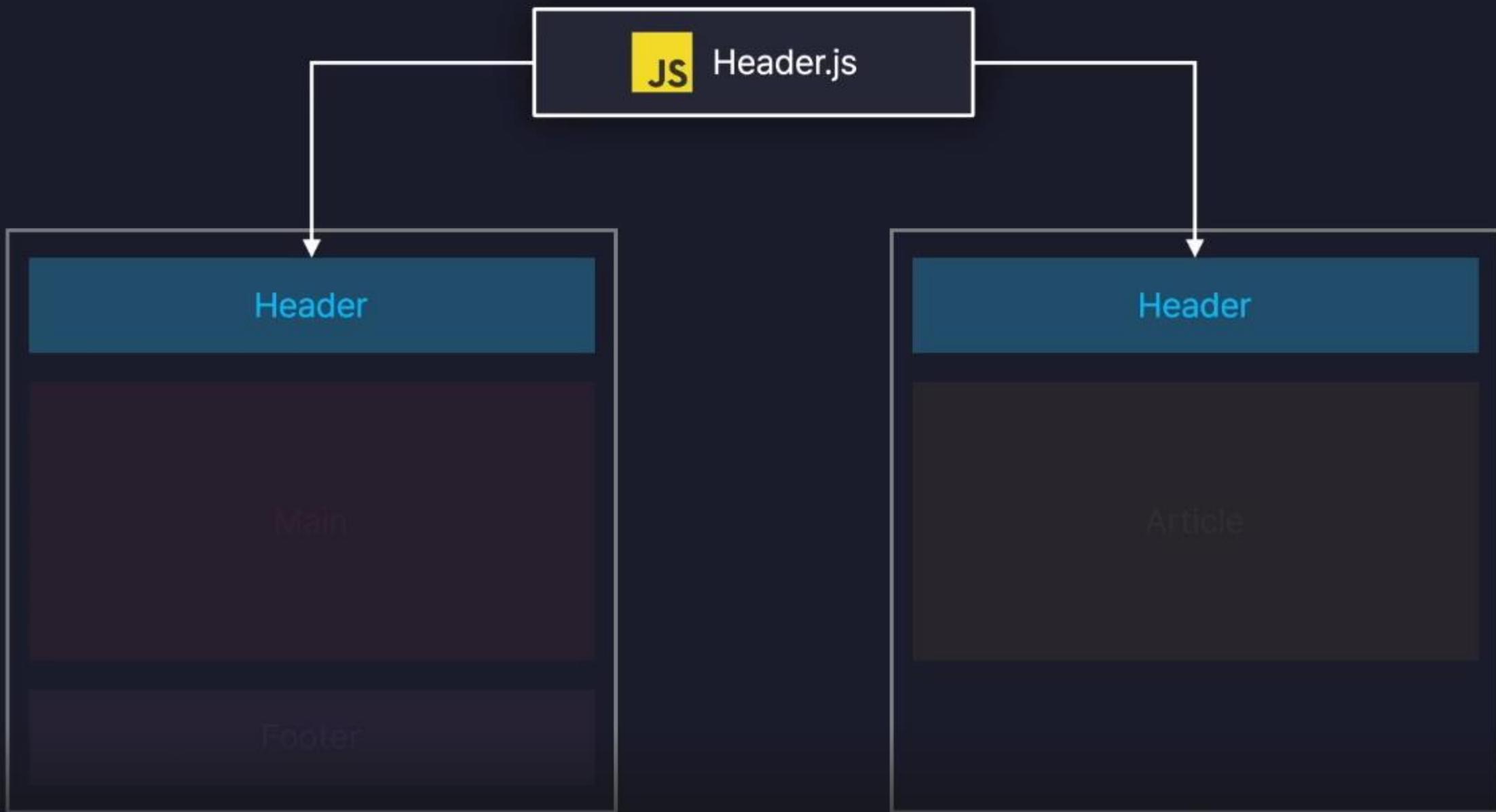
페이지 A



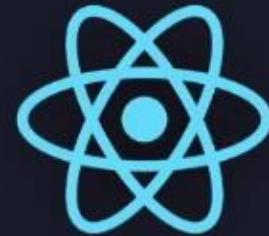
페이지 B



React의 기술적인 특징 1. 컴포넌트를 기반으로 UI를 표현한다



React의 기술적인 특징 2. 화면 업데이트 구현이 쉽다



1. 컴포넌트를 기반으로 UI를 표현한다

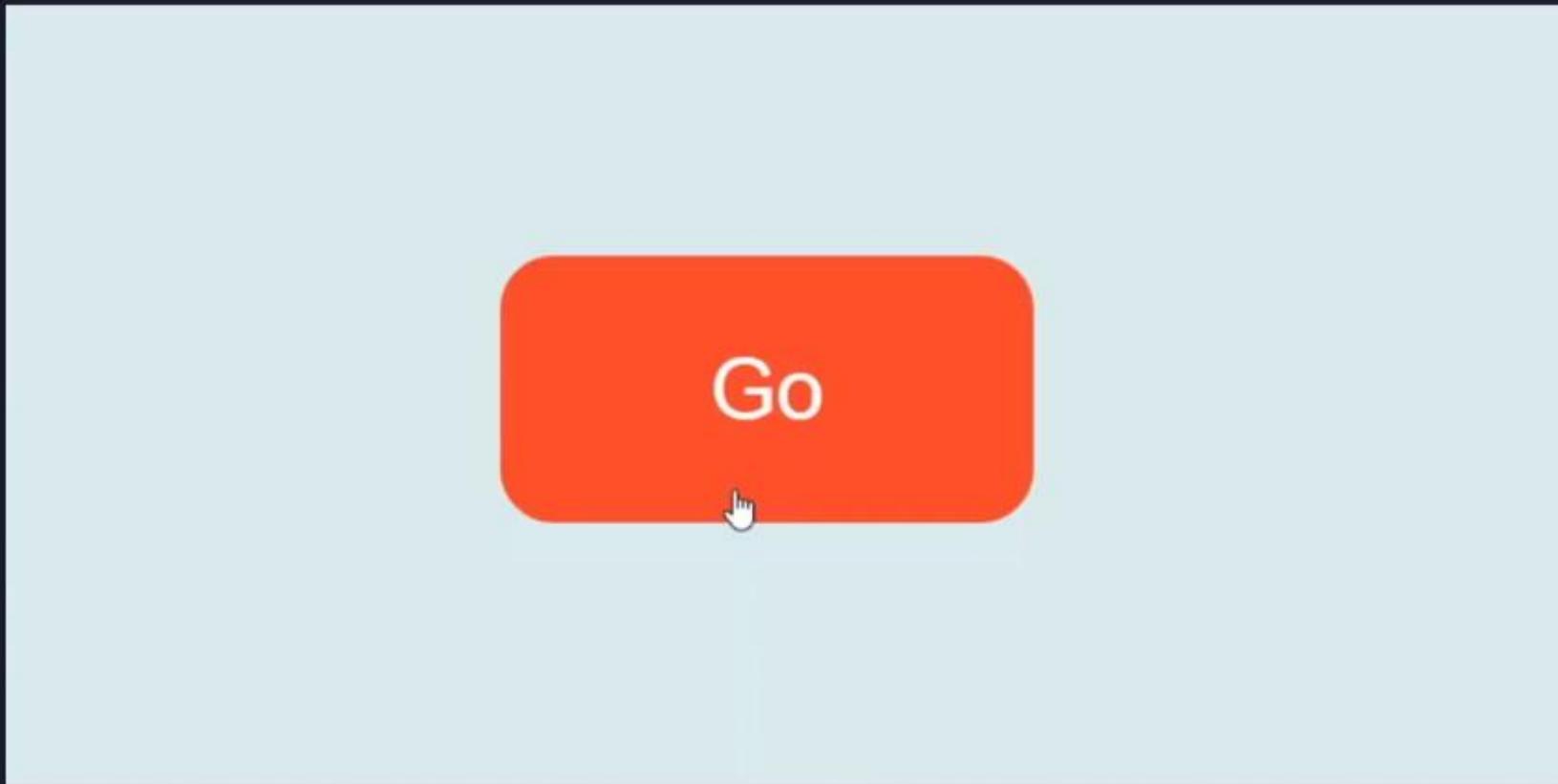
2. 화면 업데이트 구현이 쉽다

3. 화면 업데이트가 빠르게 처리된다

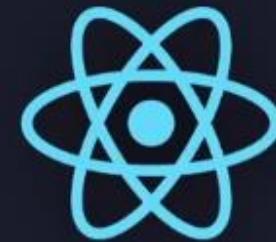
React의 기술적인 특징 2. 화면 업데이트 구현이 쉽다

업데이트란?

사용자의 행동(클릭, 드래그)에 따라 웹 페이지가 스스로 모습을 바꿔 상호작용 하는 것



React의 기술적인 특징 2. 화면 업데이트 구현이 쉽다



React.js에서는 화면 업데이트를 구현하기 쉽다

선언형 프로그래밍

React의 기술적인 특징 2. 화면 업데이트 구현이 쉽다

선언형 프로그래밍

과정은 생략하고 목적만 간결히 명시하는 방법

마치 우리가 식당에서 주문하는 것과 유사함

“토마토 파스타 하나 주세요”



React의 기술적인 특징 2. 화면 업데이트 구현이 쉽다

명령형 프로그래밍

목적을 이루기 위한 모든 일련의 과정을 설명하는 방식

마치 진상(?) 손님처럼 주문 하는 방식

"주방으로 가셔서 면 100g을 꺼내세요, 그리고 뜨거운 물에 9분간 삶으세요

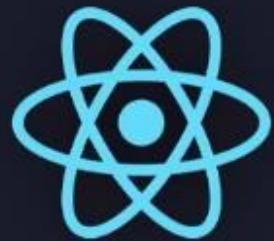
그런 다음 후라이팬을 꺼내 불을 올리고 토마토 소스와 함께 볶으세요

다 되었다면 접시에 담아서 나에게 가져와 주세요"



React의 기술적인 특징 2. 화면 업데이트 구현이 쉽다

선언형 프로그래밍



목적만 깔끔하게 명시

코드가 간결함

명령형 프로그래밍



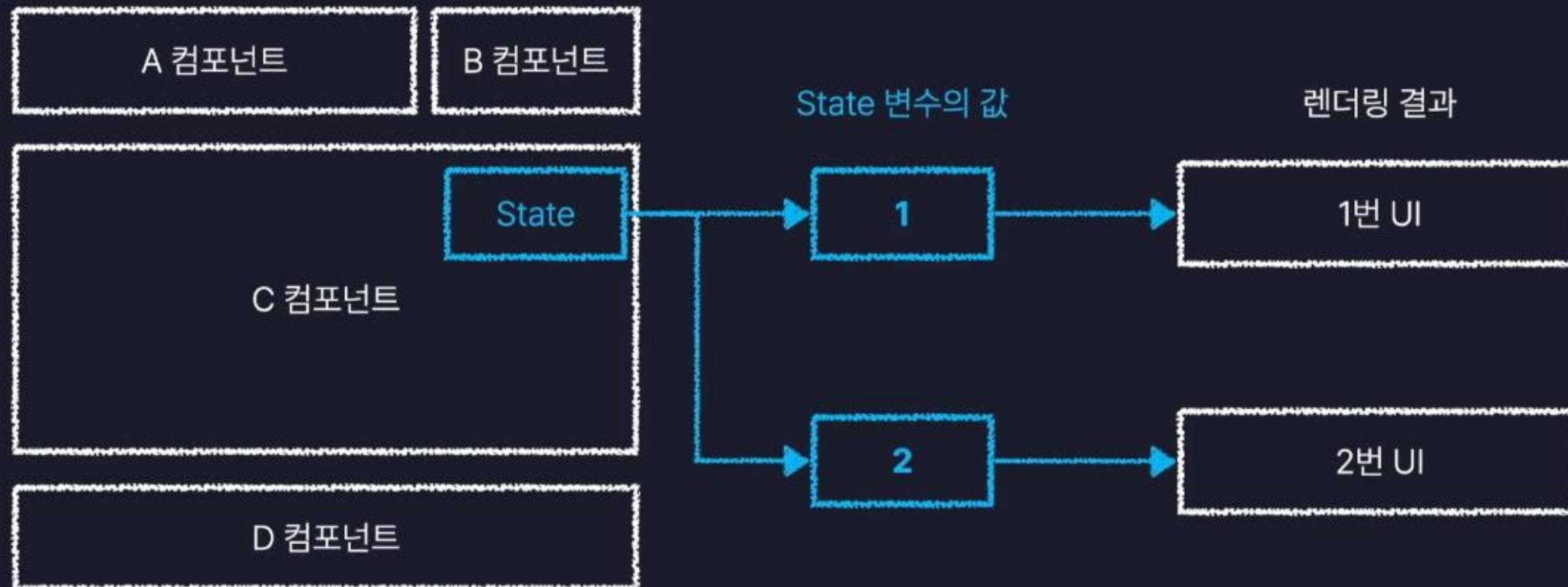
모든 과정을 하나 하나 다 설명

코드가 길고 복잡함

React의 기술적인 특징 2. 화면 업데이트 구현이 쉽다

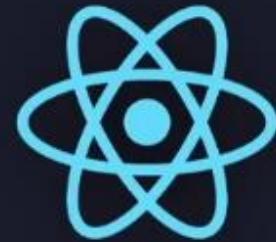


React의 기술적인 특징 2. 화면 업데이트 구현이 쉽다



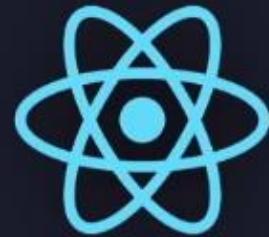
React의 기술적인 특징 2. 화면 업데이트 구현이 쉽다

선언형 프로그래밍



업데이트를 위한 복잡한 동작을 직접 정의할 필요 없이
특정 변수의 값을 바꾸는 것 만으로도
화면을 업데이트 시킬 수 있다.

React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다



화면 업데이트를 쉽게 구현할 수 있으면서

동시에 빠르게도 처리해 줌

🤔 어떻게 이럴 수 있을까?

React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다



[선수 지식 필요]

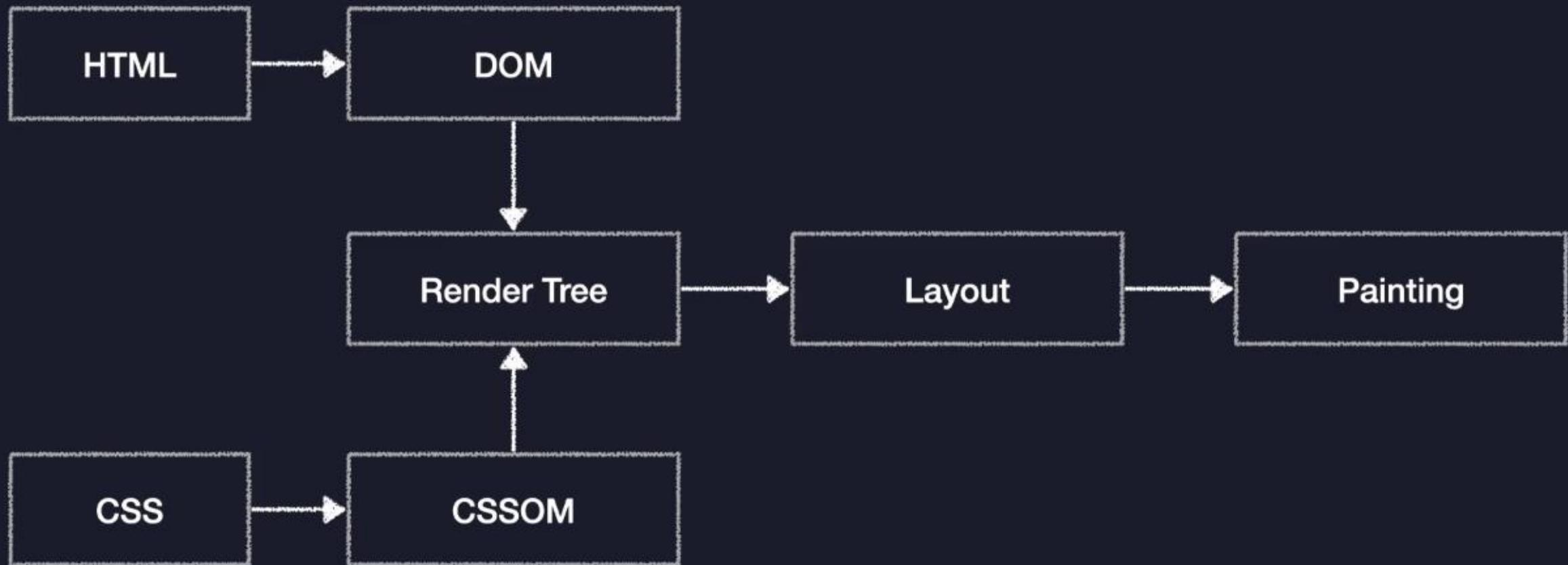
브라우저는 어떻게 동작하는가?

HTML, CSS로 만든 페이지를 어떻게 렌더링 하는가?

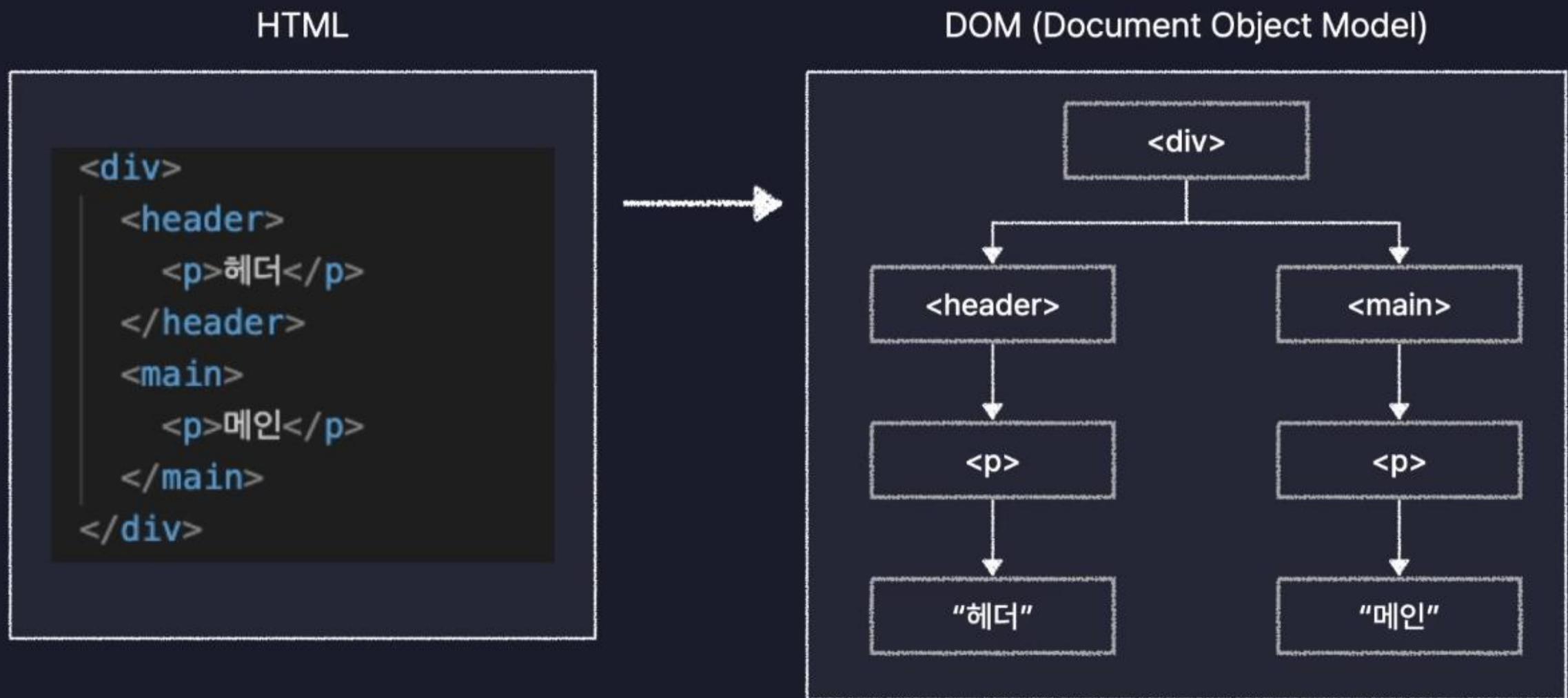
화면 업데이트는 어떻게 처리 되는가?

React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다

브라우저의 렌더링 과정 (Critical Rendering Path)

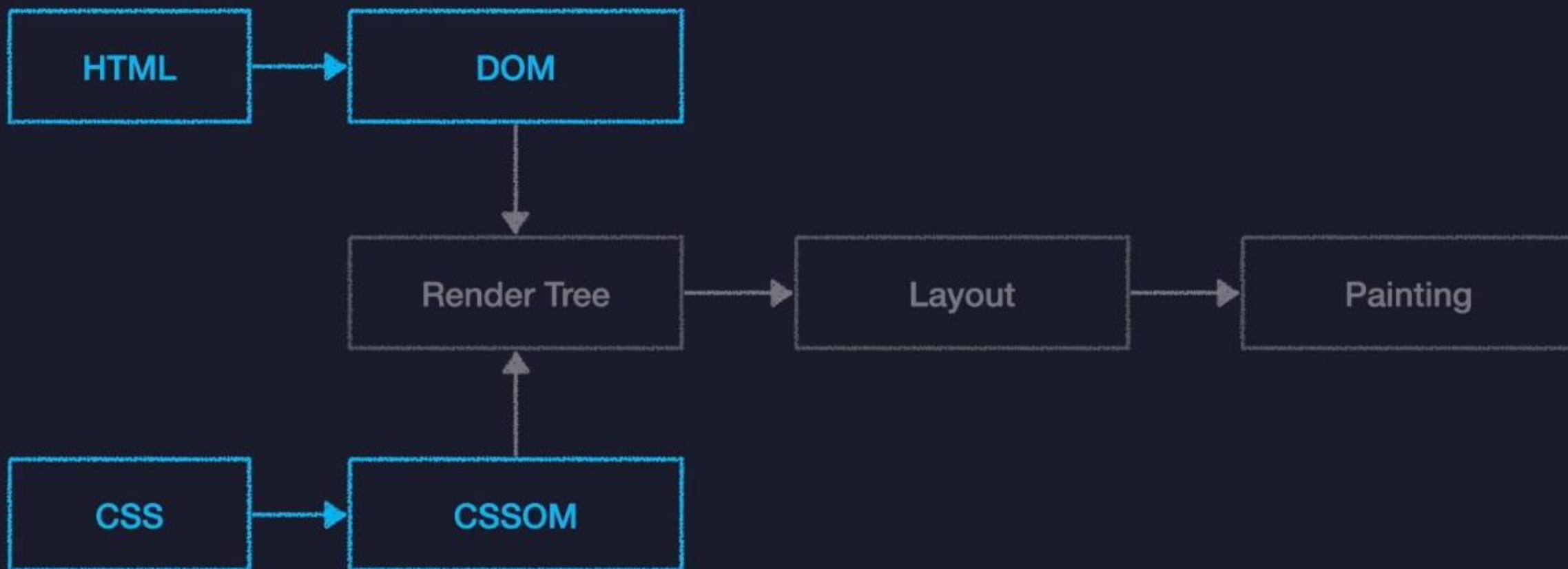


React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다



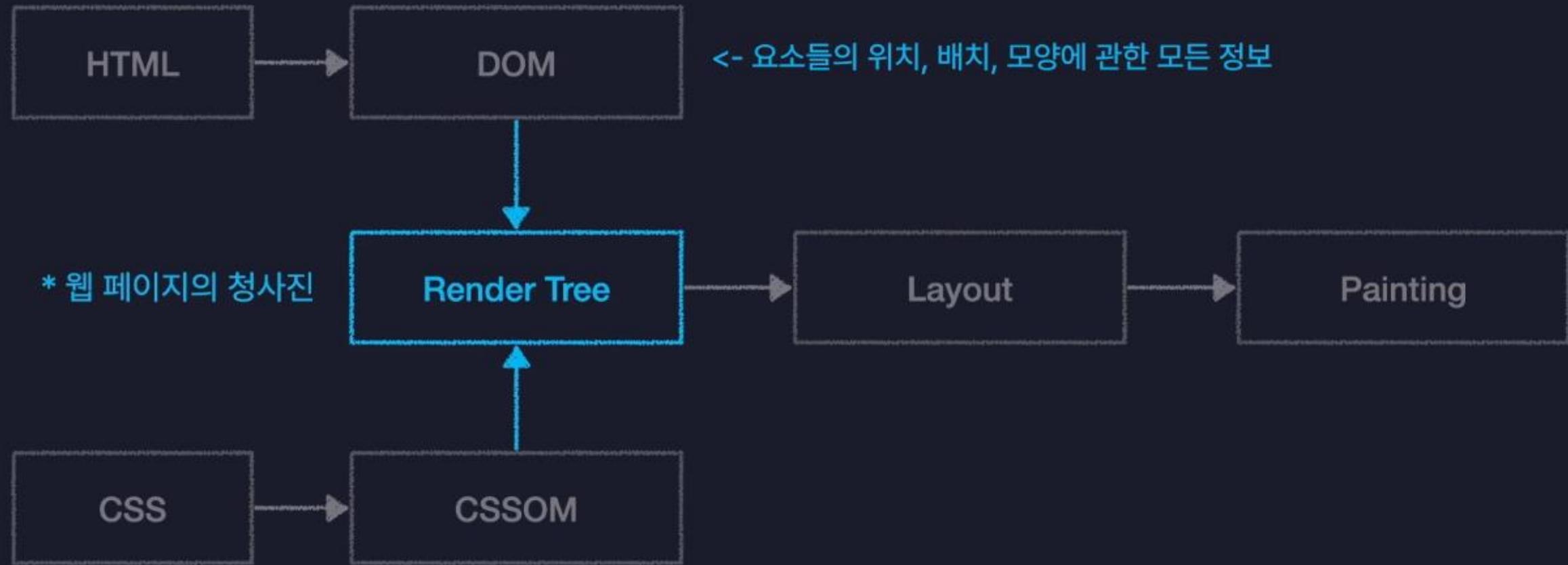
React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다

브라우저의 렌더링 과정 (Critical Rendering Path)



React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다

브라우저의 렌더링 과정 (Critical Rendering Path)



React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다

브라우저의 렌더링 과정 (Critical Rendering Path)



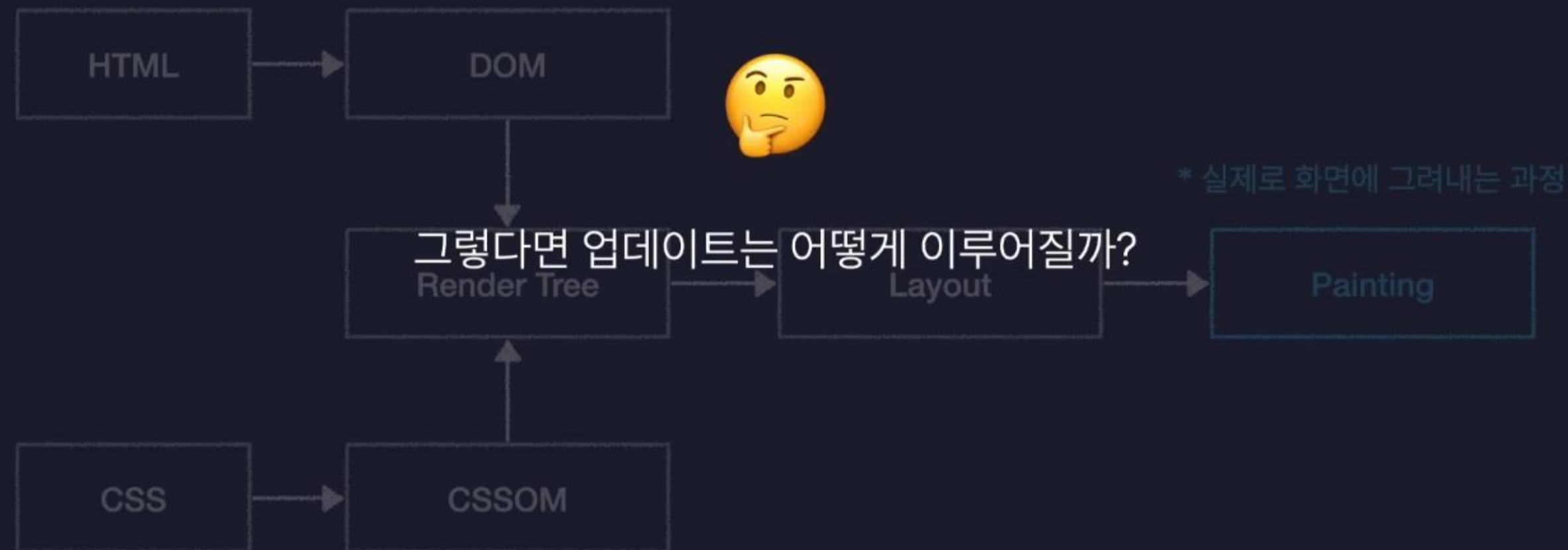
React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다

브라우저의 렌더링 과정 (Critical Rendering Path)



React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다

브라우저의 렌더링 과정 (Critical Rendering Path)



React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다

JavaScript가 DOM을 수정하면 업데이트가 발생한다



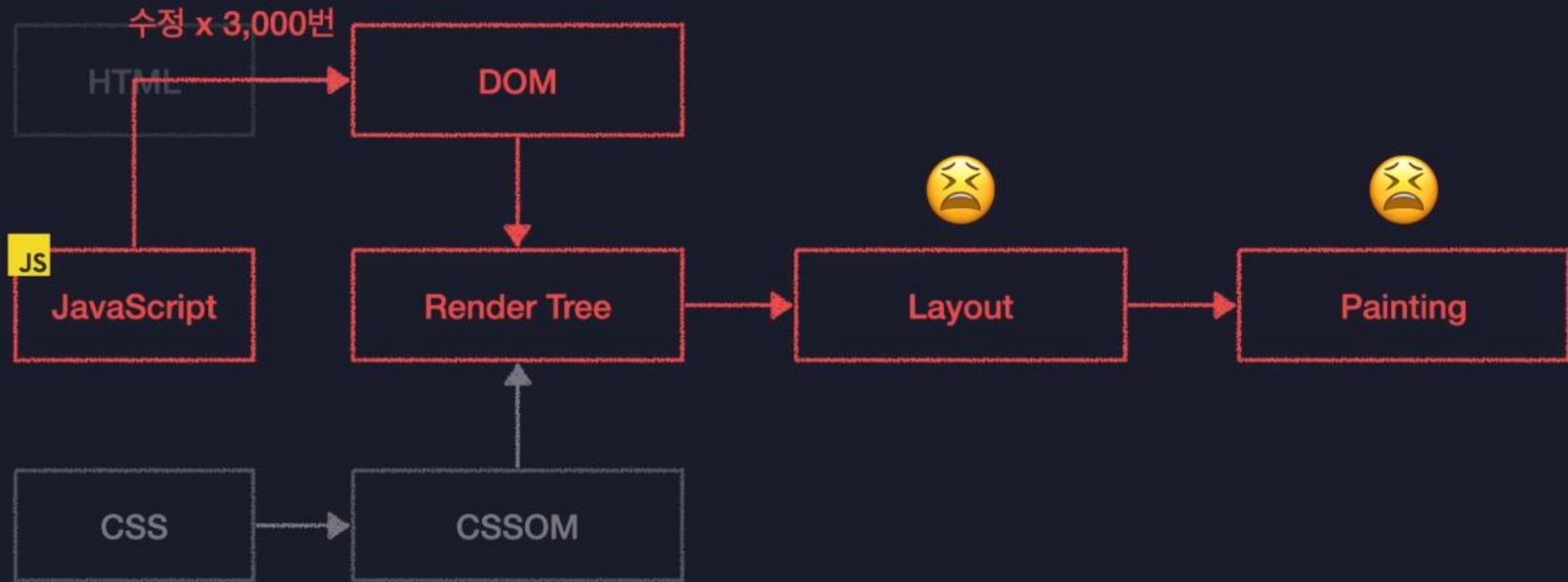
React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다

주의! Layout, Painting은 매우 오래걸리는 과정임



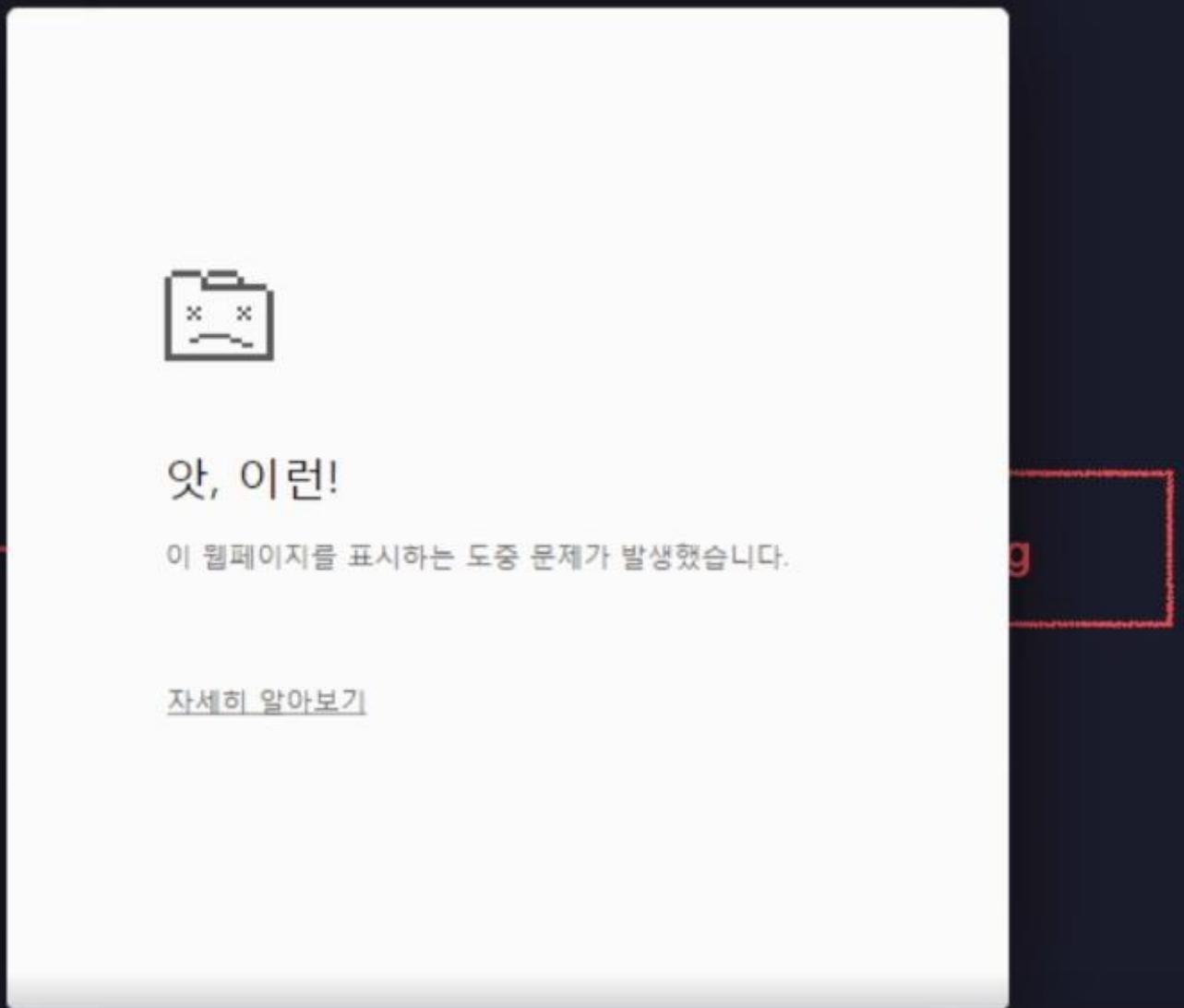
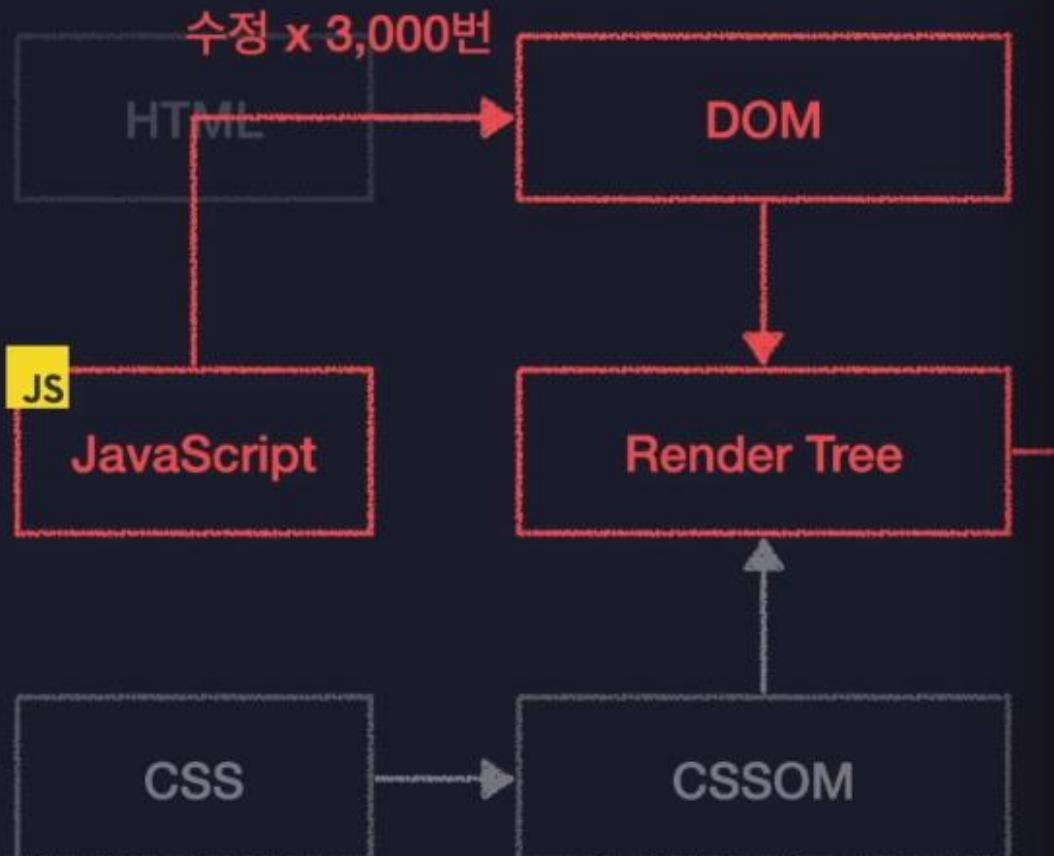
React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다

JavaScript가 DOM을 수정하면 업데이트가 발생한다



React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다

JavaScript가 DOM을 수정하면 업데이트가 발생한다



React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다

index.html

```
11 <script>
12   function onClick() {
13     const $ul = document.getElementById("ul");
14     for (let i = 0; i < 3000; i++) {
15       $ul.innerHTML += `<li>${i}</li>`;
16     }
17   }
18 </script>
19 <body>
20   <button onclick="onClick()">리스트 추가하기</button>
21   <ul id="ul"></ul>
22 </body>
```

렌더링 결과

리스트 추가하기!

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다

index.html

```
11 <script>
12   function onClick() {
13     const $ul = document.getElementById("ul");
14     for (let i = 0; i < 3000; i++) {
15       $ul.innerHTML += `<li>${i}</li>`;
16     }
17   }
18 </script>
19 <body>
20   <button onclick="onClick()">리스트 추가하기</button>
21   <ul id="ul"></ul>
22 </body>
```

😡 Super Bad Practice

- 3,000번 DOM을 수정
- 성능 측정 결과 : **4,500ms**

React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다

```
11 <script>
12     function onClick() {
13         const $ul = document.getElementById("ul");
14         let list = "";
15
16         for (let i = 0; i < 3000; i++) {
17             list += `<li>${i}</li>`;
18         }
19
20         $ul.innerHTML = list;
21     }
22 </script>
23 <body>
24     <button onclick="onClick()">리스트 추가하기</button>
25     <ul id="ul"></ul>
26 </body>
```

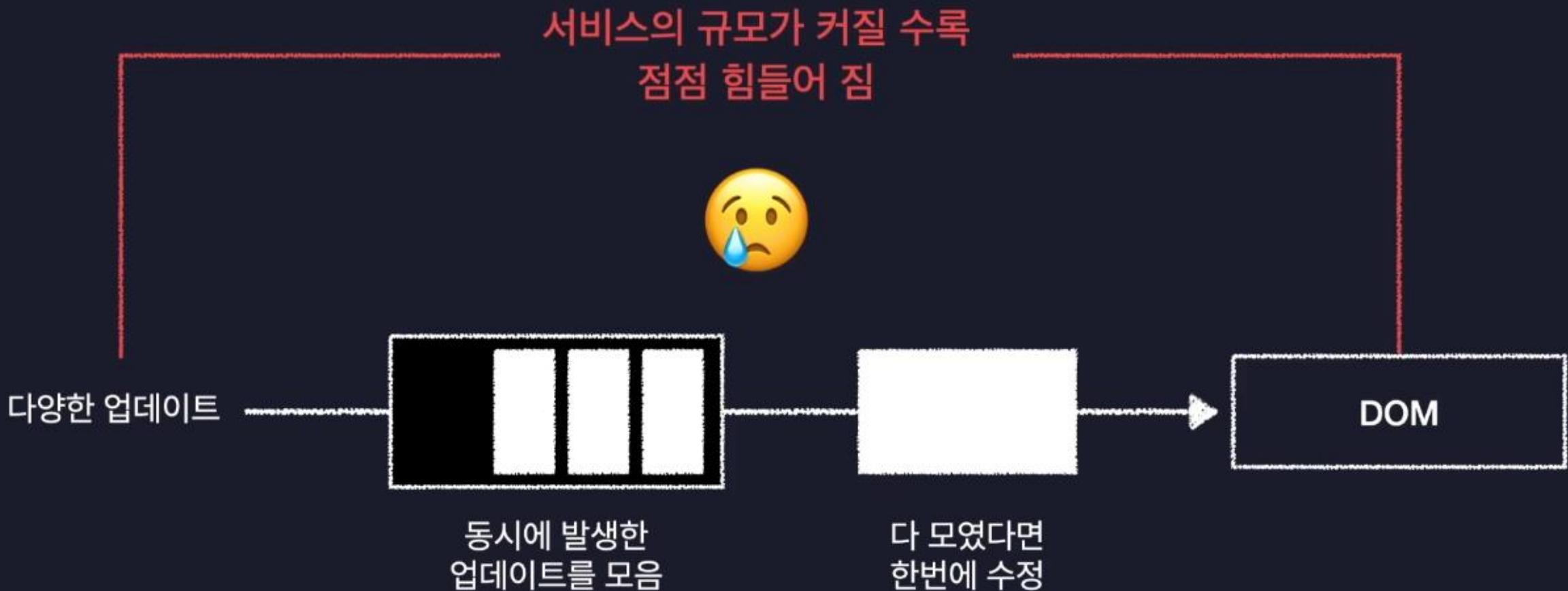
Good Practice

- DOM은 딱 한번만 수정함
- 성능 측정 결과 : **250ms**
(약 22배 개선 됨)

React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다

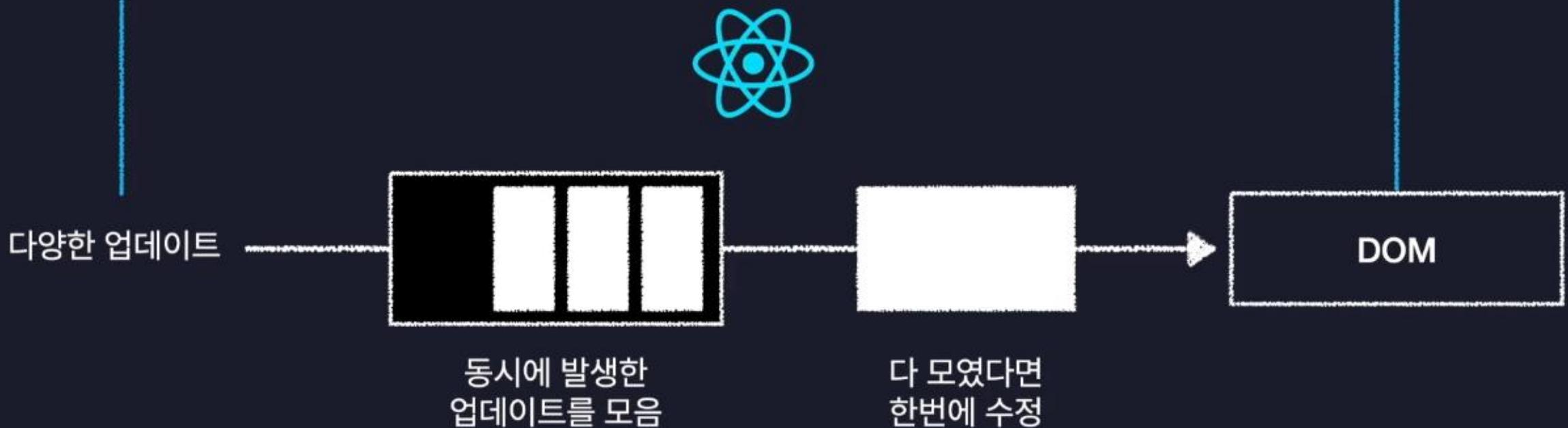


React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다



React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다

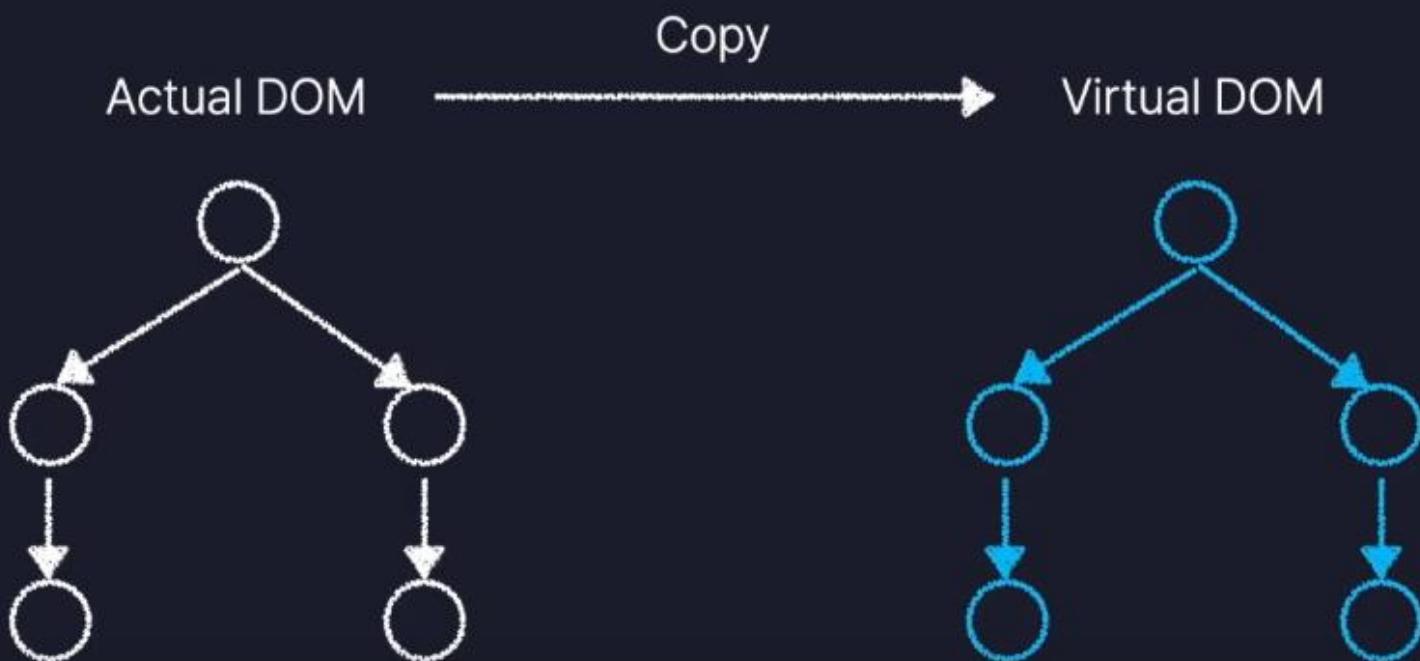
React는 이 과정을 자동으로 해줌
(Feat. Virtual DOM)



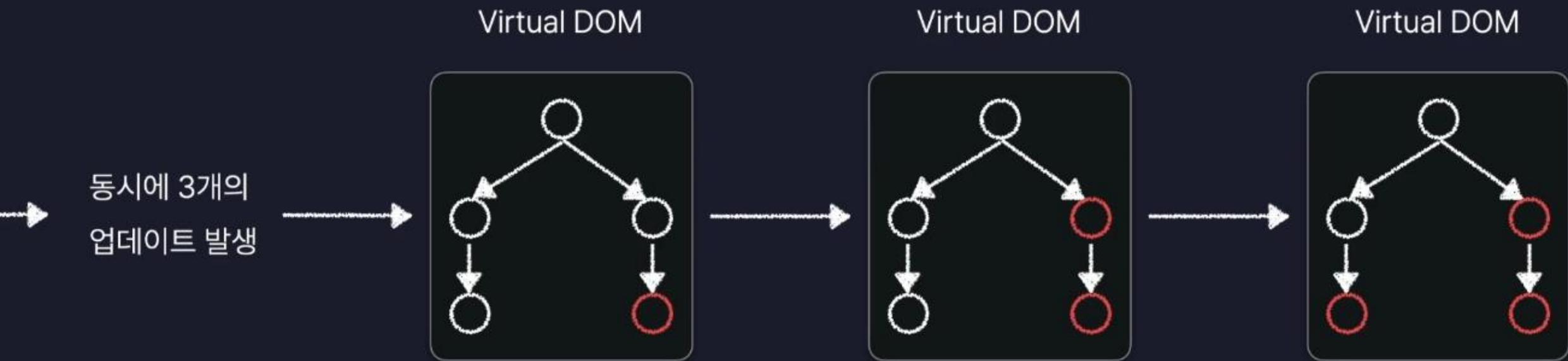
React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다

Virtual DOM 이란?

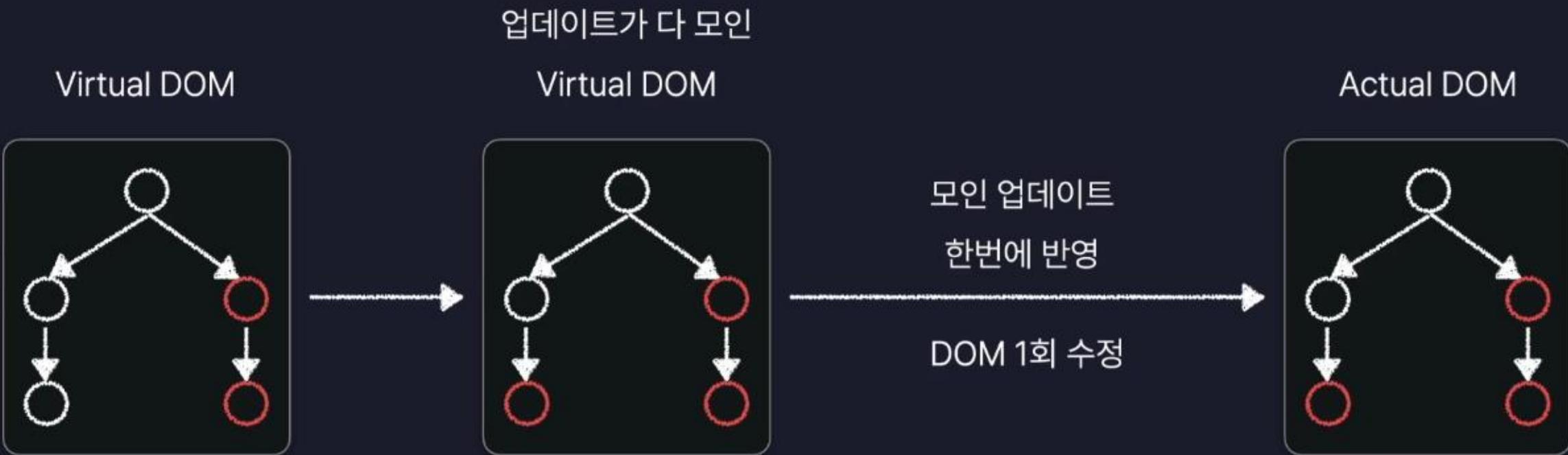
- DOM을 자바스크립트 객체로 흉내낸 것으로 일종의 복제판이라고 생각하면 된다
- React는 업데이트가 발생하면 실제 DOM을 수정하기 전에 이 가상의 복제판 DOM에 먼저 반영해본다.
 - 연습 스윙 같은 느낌이라고 생각하면 된다.



React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다



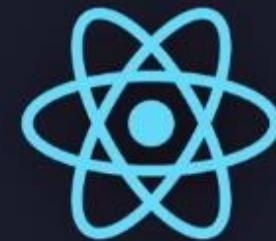
React의 기술적인 특징 3. 화면 업데이트가 빠르게 처리된다



한일 크기로 잘라먹는

첫번째 React App 생성하기

React Application?



React App, React Application



React App 생성하기

Nanananananananana.. Pat Man!

Pro Teams Pricing Documentation Sign Up Sign In

npm Search packages Search Sign Up Sign In

react DT
18.2.0 • Public • Published a year ago

Readme Code Beta 1 Dependency 0 Dependents 1,381 Versions

react

React is a JavaScript library for creating user interfaces.

The `react` package contains only the functionality necessary to define React components. It is typically used together with a React renderer like `react-dom` for the web, or `react-native` for the native environments.

Note: by default, React will be in development mode. The development version includes extra warnings about common mistakes, whereas the production version includes extra performance optimizations and strips all error messages. Don't forget to use the `production build` when deploying your application.

Usage

Install

```
> npm i react
```

Repository github.com/facebook/react

Homepage reactjs.org/

Weekly Downloads 19,051,936

Version 18.2.0 License MIT

React App 생성하기

The screenshot shows the npm package page for 'react'. At the top, there's a navigation bar with links for 'Pro', 'Teams', 'Pricing', and 'Documentation'. Below that is the npm logo and a search bar with the placeholder 'Search packages'. To the right of the search bar are 'Sign Up' and 'Sign In' buttons.

The main content area shows the package details for 'react' version 18.2.0. It includes a 'Readme' tab (which is selected), a 'Code' tab, and a 'Beta' tab. Other tabs include '0 Dependents' and '1,381 Versions'. The 'Readme' tab contains the following text:

1. Node.js 패키지 생성
2. React 라이브러리 설치
3. 기타 도구 설치 및 설정
(입문자에게는 권장하기 어려움)

The 'Readme' also contains a note about development mode and production build, a 'Usage' section, and links to the repository (github.com/facebook/react), homepage (reactjs.org), and weekly downloads (19,051,936).

React App 생성하기



Vite

차세대 프론트엔드 개발 툴

기본 설정이 적용된 React App 생성 가능

React App 생성하기



- If your app doesn't have an existing setup for compiling JavaScript modules, set it up with **Vite**. The **Vite** community maintains **many integrations with backend frameworks**, including Rails, Django, and Laravel. If your backend framework is not listed, **follow this guide** to manually integrate **Vite** builds with your backend.

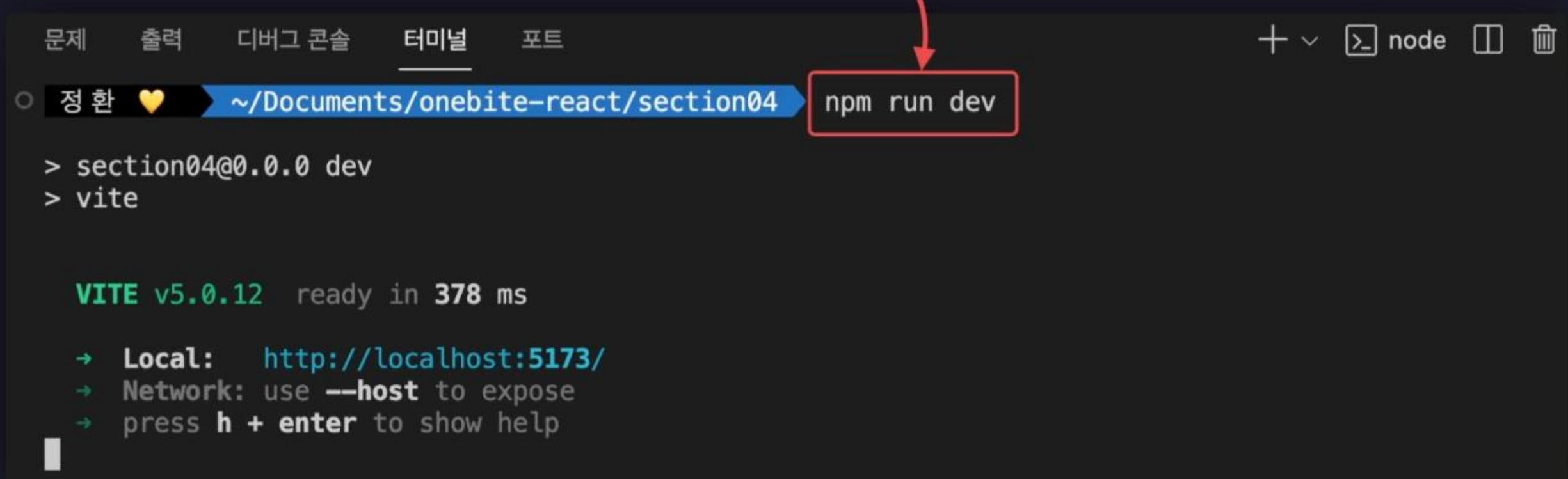
React 공식 문서에서도 권장하고 있음

한입 크기로 잘라먹는

React App 구동 원리 살펴보기

React App은 어떻게 구동되는걸까?

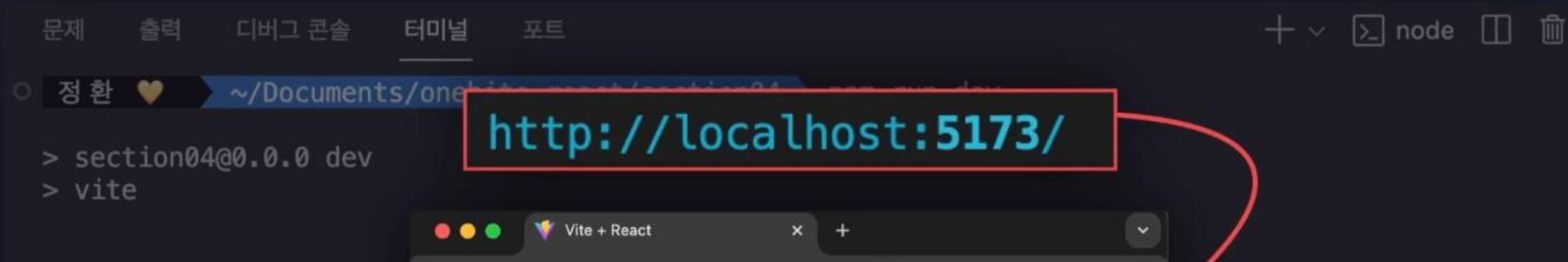
2. React App 가동



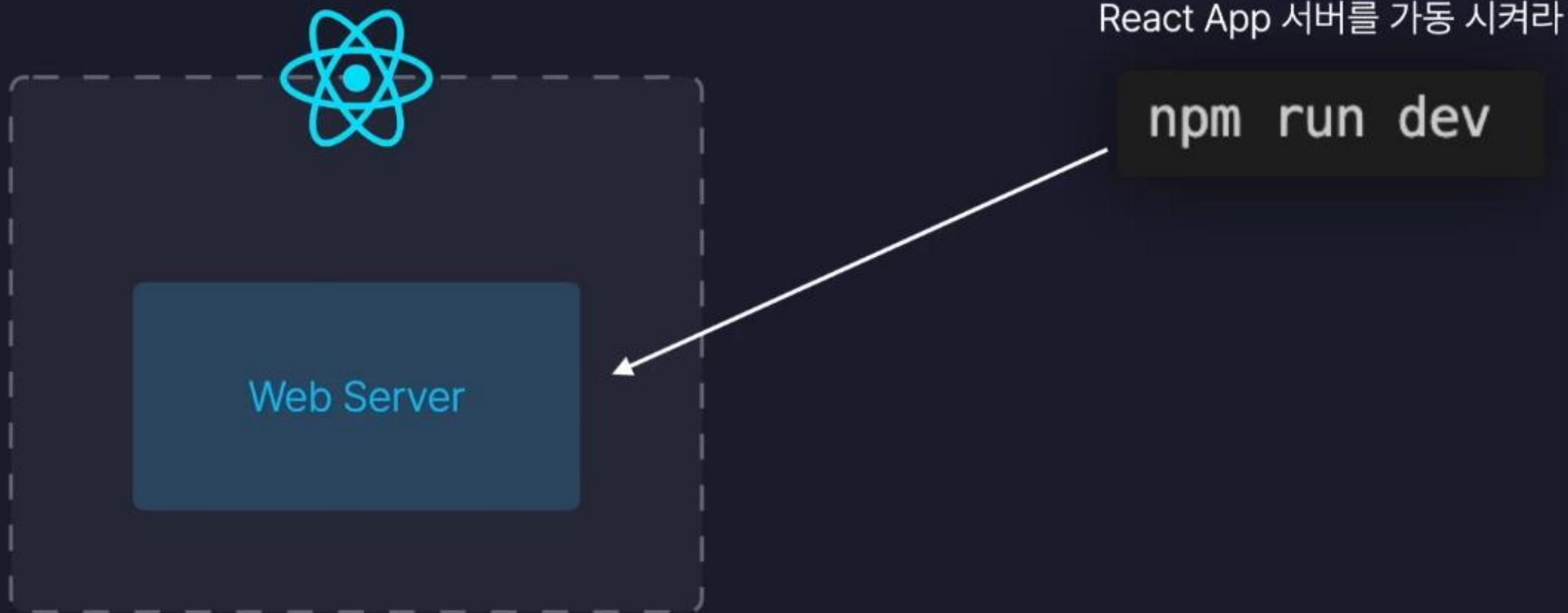
```
문제    출력    디버그 콘솔    터미널    포트    +    node    □    ✖
○ 정환 ❤️ ➤ ~/Documents/onebite-react/section04 ➤ npm run dev
> section04@0.0.0 dev
> vite

VITE v5.0.12 ready in 378 ms
→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
```

React App은 어떻게 구동되는걸까?



React Application?



React Application?

React Application?

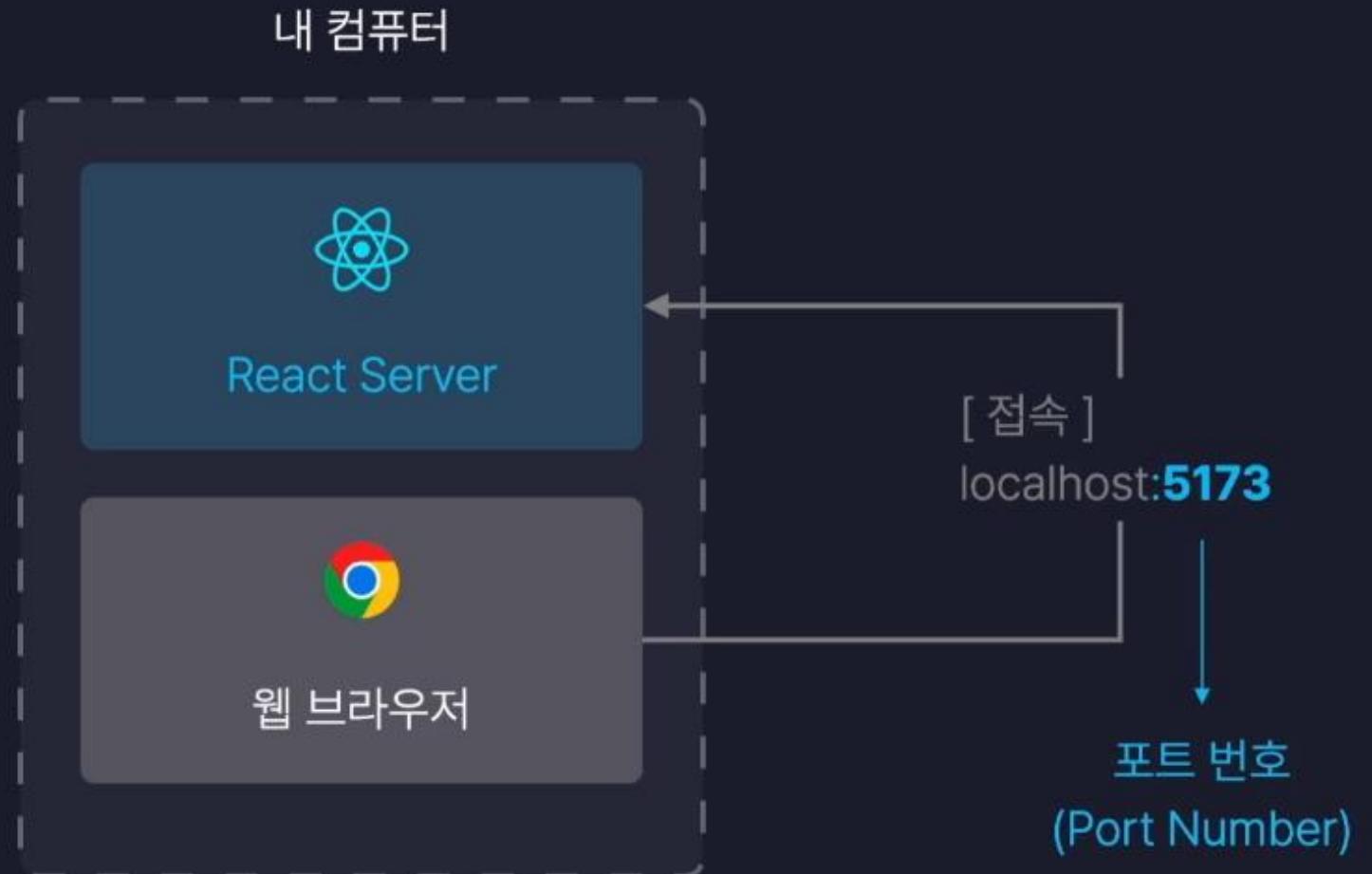
[http://localhost :5173](http://localhost:5173)

우리 컴퓨터를 의미 함

React Application?



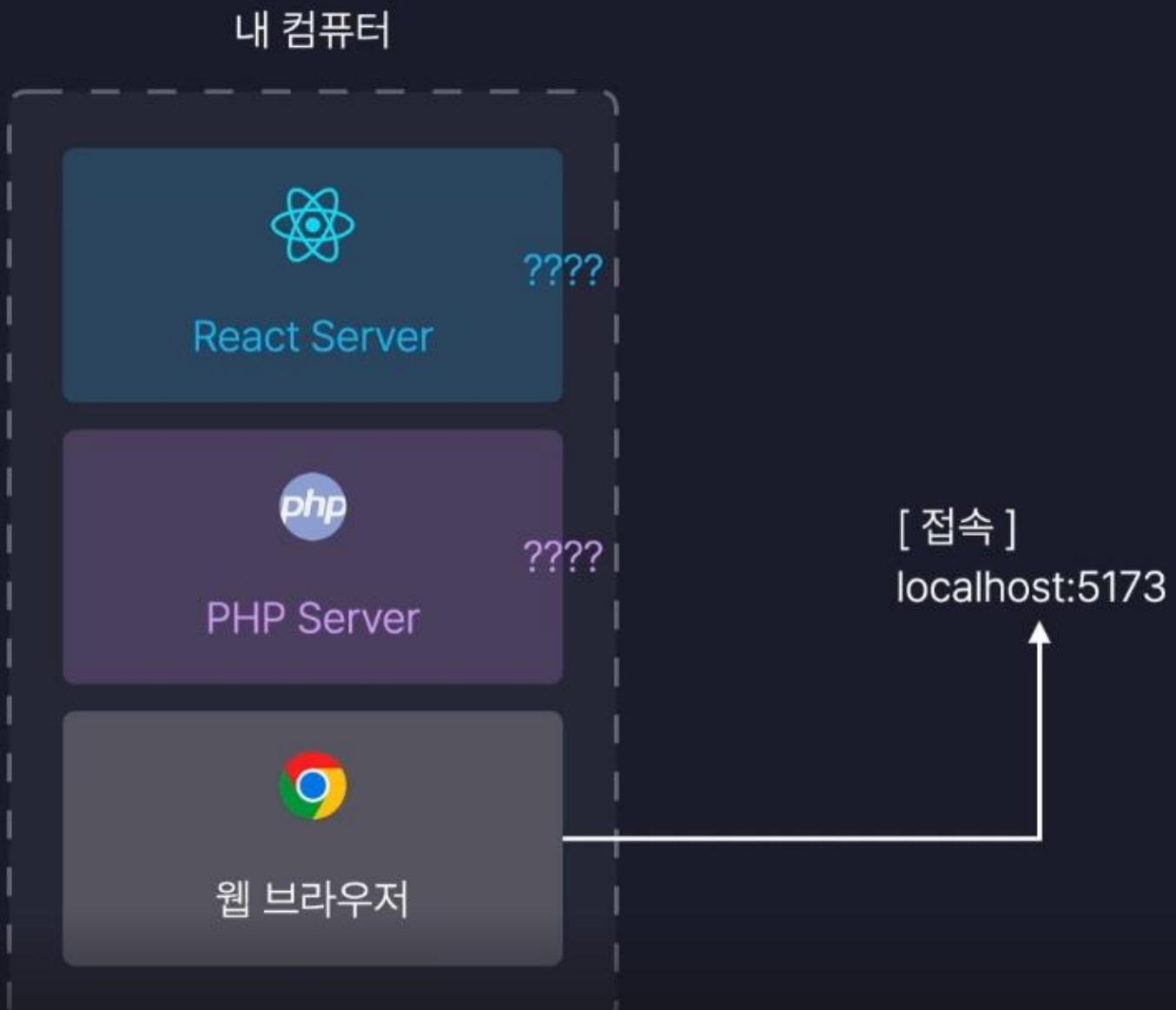
React Application?



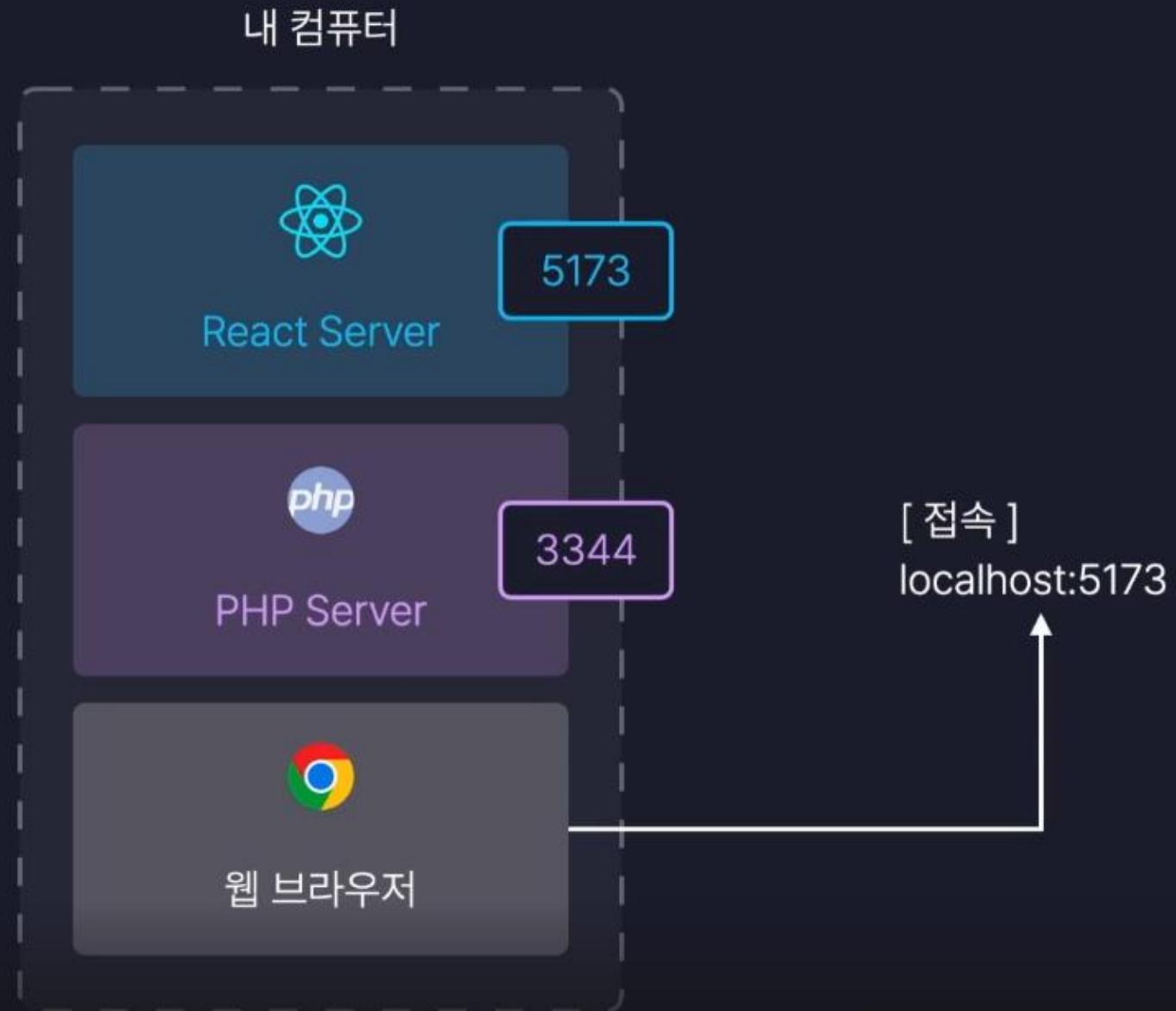
React Application?



React Application?



React Application?



React Application?

서울특별시
xx동 거주



영희

인천광역시
xx동 거주



철수

React Application?

서울특별시
xx동 거주



영희

우리 집에 갈래?

인천광역시
xx동 거주

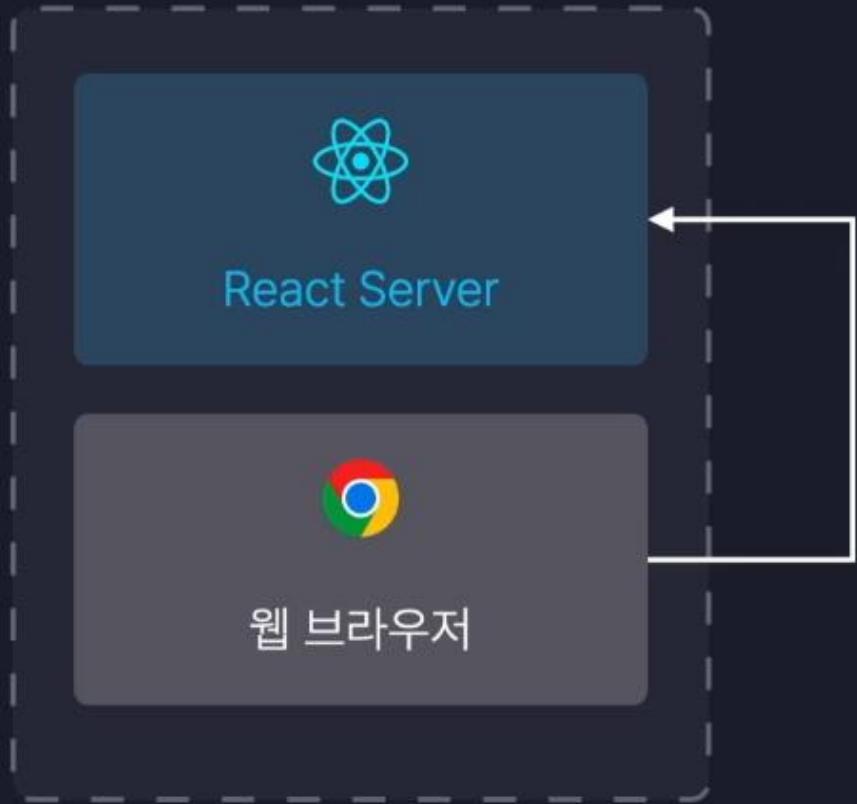


철수

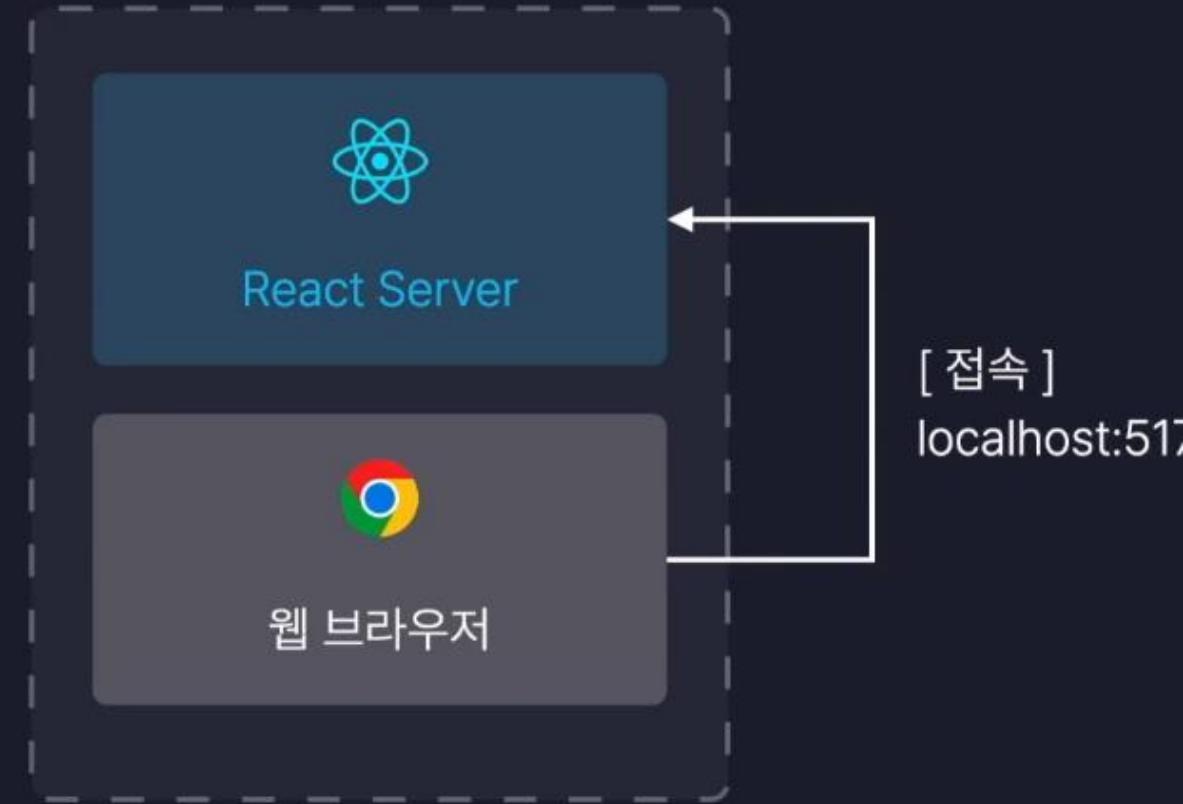
우리 집에 갈래?

React Application?

A의 컴퓨터



B의 컴퓨터



한입 크기로 잘라먹는

실습 준비하기

한입 크기로 잘라먹는

React 컴포넌트(Component)

한입 크기로 잘라먹는

JSX - UI 표현하기

JSX란?

React Component

```
function Footer() {  
  return (  
    <footer>  
      <h1>footer</h1>  
    </footer>  
  );  
}
```

JSX란?

React Component

```
function Footer() {  
  return (  
    <footer>  
      <h1>footer</h1>  
    </footer>  
  );  
}
```

JavaScript 에서는
문법적인 오류로 판단함

JSX란?

React Component

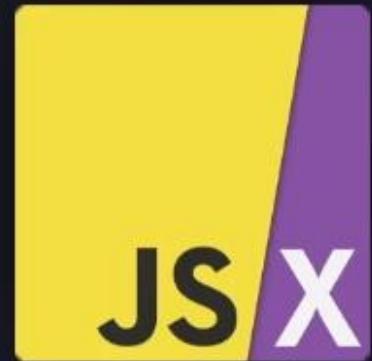
```
function Footer() {  
  return (  
    <footer>  
      <h1>footer</h1>  
    </footer>  
  );  
}
```



React.js 에서는
JSX 문법을 사용하므로
적법하다고 판단함

* JSX: 확장된 자바스크립트 문법

JSX란?



JSX

(JavaScript Extensions)

확장된 자바스크립트의 문법을 말함

JSX = JavaScript Extensions

React Component

```
function Footer() {  
  return (  
    <footer>  
      <h1>footer</h1>  
    </footer>  
  );  
}
```

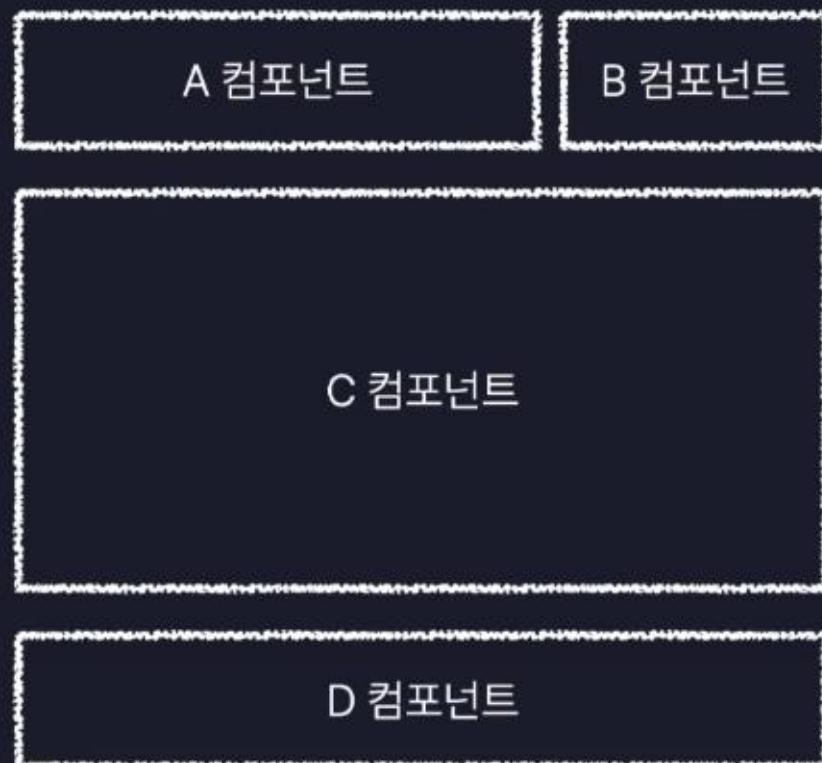
← JavaScript

← HTML

한입 크기로 잘라먹는

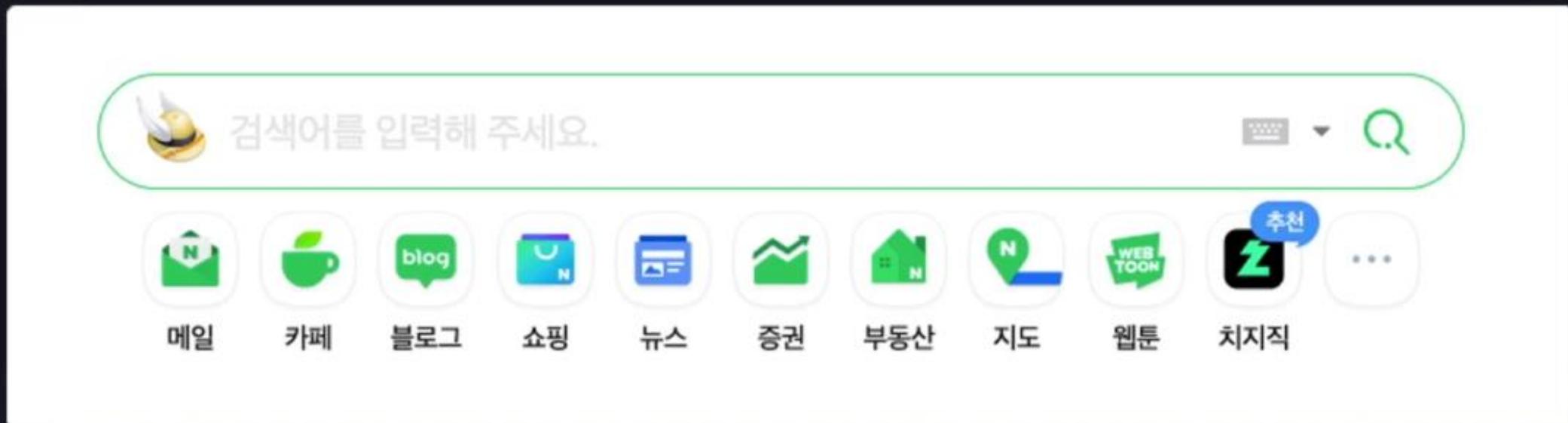
Props - 컴포넌트에 값 전달하기

Props란?



Props란?

NAVER 메인을
React.js로 구현한다면?

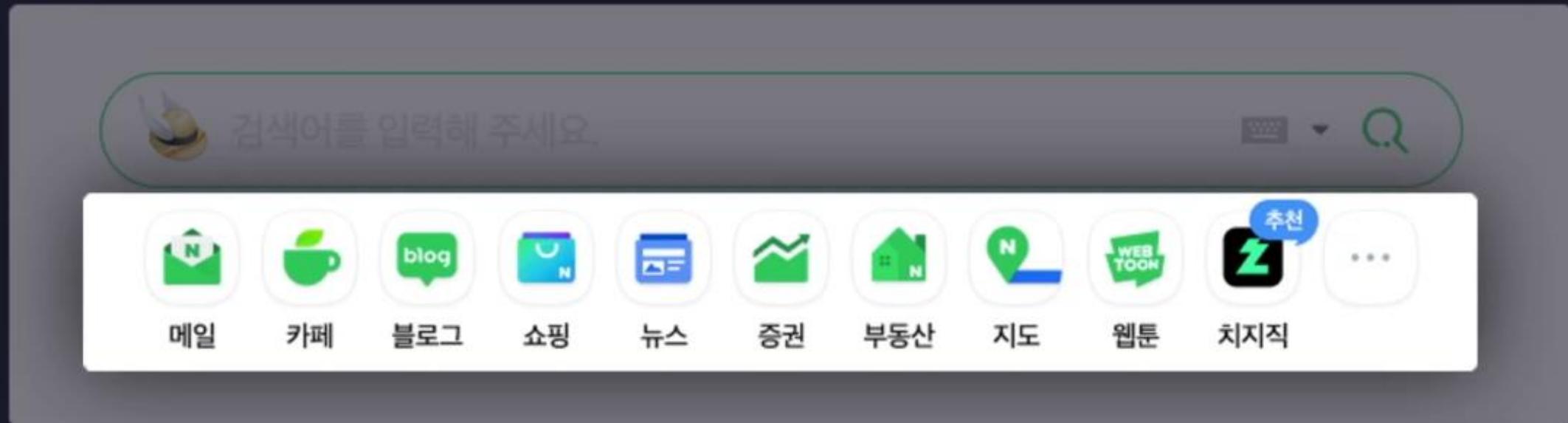


Props란?

SearchBar 컴포넌트

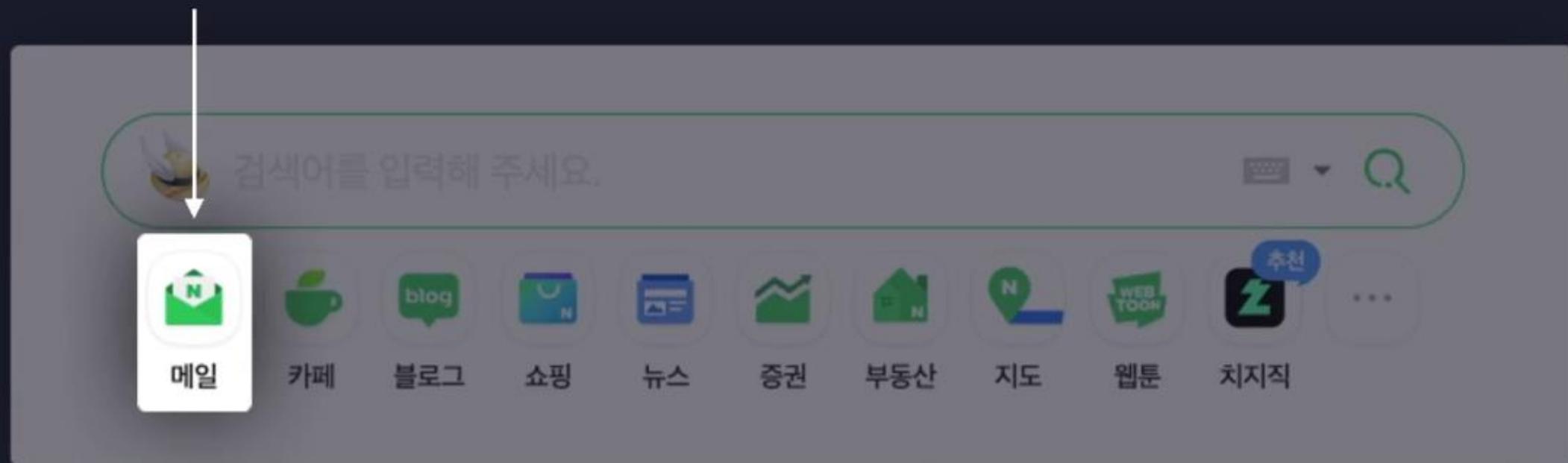


Props란?



Props란?

Button 컴포넌트



Props란?

Button 컴포넌트



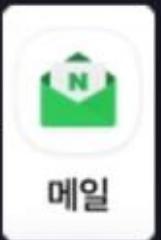
Props란?

```
function App() {  
  return (  
    <>  
    <Button/>  
    <Button/>  
    <Button/>  
    ...  
    </>  
  );  
}
```

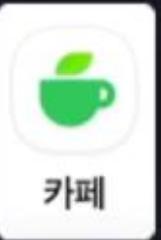


Props란?

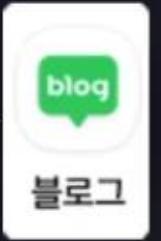
```
function App() {  
  return (  
    <>  
    <Button text={"메일"} img={"mail.png"} />  
    <Button text={"카페"} img={"cafe.png"} />  
    <Button text={"블로그"} img={"blog.png"} />  
    ...  
  </>  
);  
}
```



메일



카페



블로그

Props란?

```
function App() {  
  return (  
    <>  
      <Button text={"메일"} img={"mail.png"} />  
      <Button text={"카페"} img={"cafe.png"} />  
      <Button text={"블로그"} img={"blog.png"} />  
      ...  
    </>  
  );  
}
```

값을 전달할 수 있음



Props란?

```
function App() {  
  return (  
    <>  
      <Button text={"메일"} img={"mail.png"} />  
      <Button text={"카페"} img={"cafe.png"} />  
      <Button text={"블로그"} img={"blog.png"} />  
      ...  
    </>  
  );  
}
```

값을 전달할 수 있음



Props Properties 줄임 말

한입 크기로 잘라먹는

이벤트 핸들링

이벤트 핸들링이란?

Event Handling

웹 내부에서 발생하는 사용자의 행동

ex) 버튼 클릭, 메세지 입력, 스크롤 등등

이벤트 핸들링이란?

Event Handling

다루다, 취급하다, 처리하다

이벤트 핸들링이란?

Event Handling

이벤트가 발생했을 때 그것을 처리하는 것

ex) 버튼 클릭시 경고창 노출

한일 크기로 잘라먹는

State - 상태 관리하기

State란?

전구



State란?

켜진 상태의 전구



State란?

전구



State란?

꺼진 상태의 전구



State란?



State

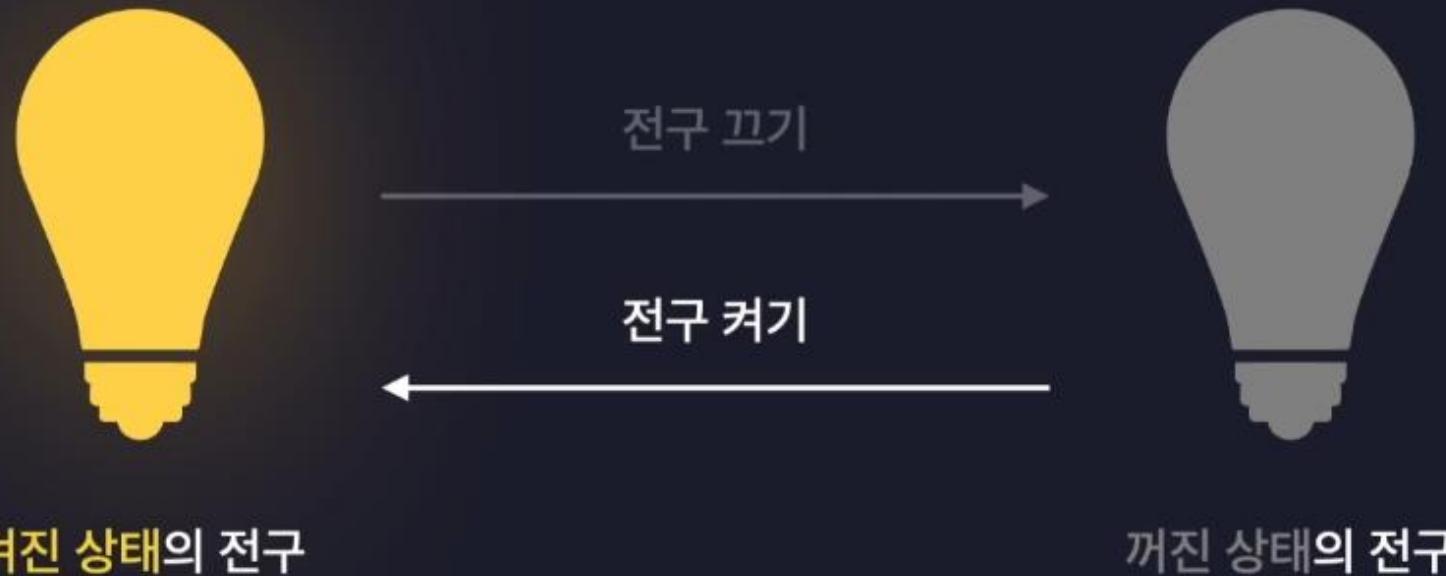
상태



켜진 상태의 전구

꺼진 상태의 전구

State는 변화할 수 있다



State란?

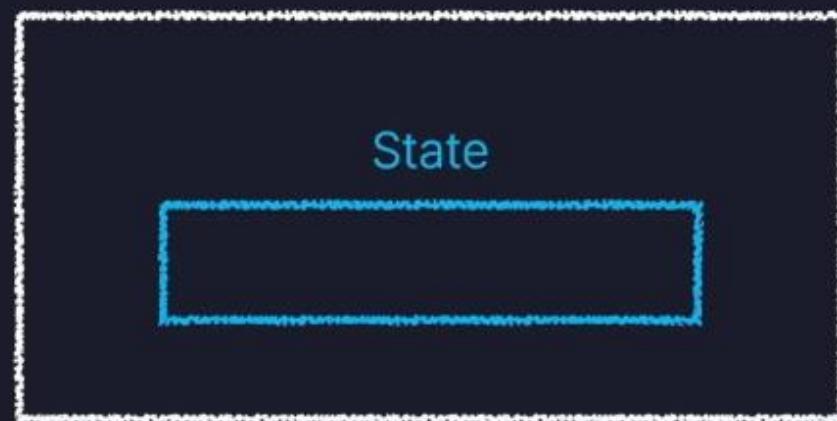
State

현재 가지고 있는 형태나 모양을 정의

변화할 수 있는 동적인 값

State란?

React 컴포넌트

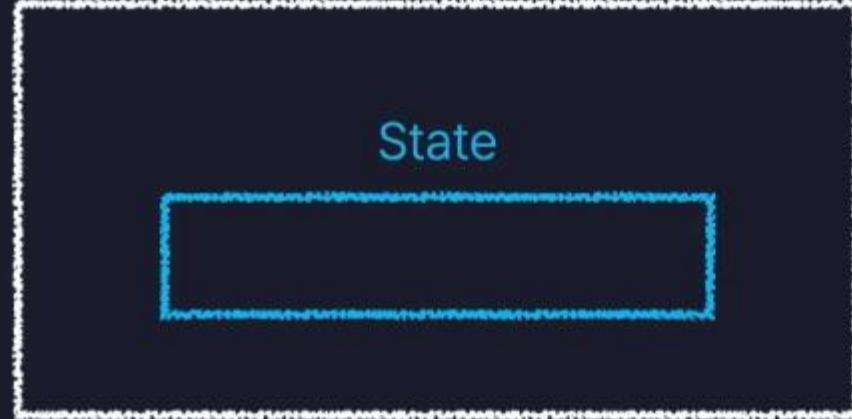


State란?

우리가 지금까지
만든 컴포넌트



앞으로 만들어볼
컴포넌트



State란?

State를 갖는

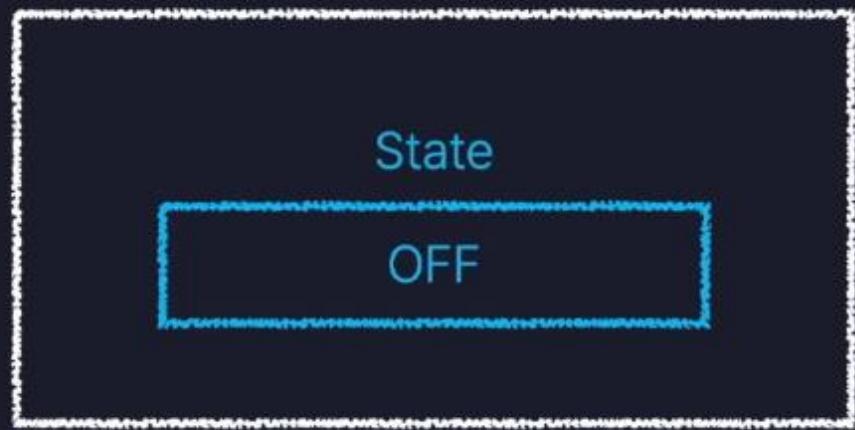
React 컴포넌트



State의 값에 따라
렌더링 되는 UI가 결정된다

State란?

State를 갖는
React 컴포넌트



렌더링 결과

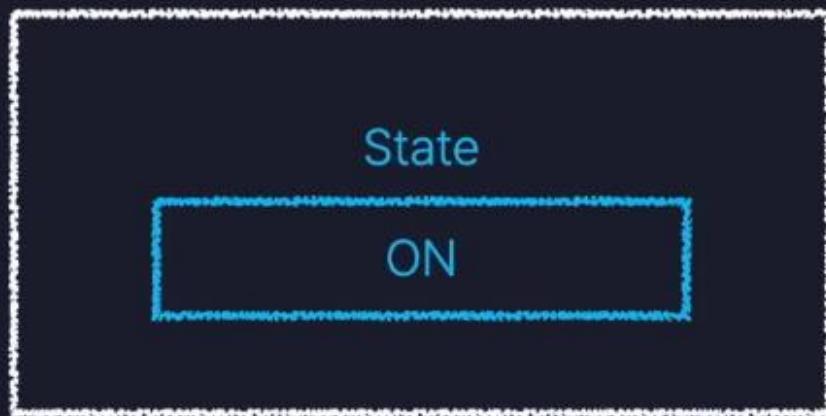


State란?



State란?

State를 갖는
React 컴포넌트



렌더링 결과



State란?

State를 갖는

React 컴포넌트

렌더링 결과

리 렌더(Re-Render)

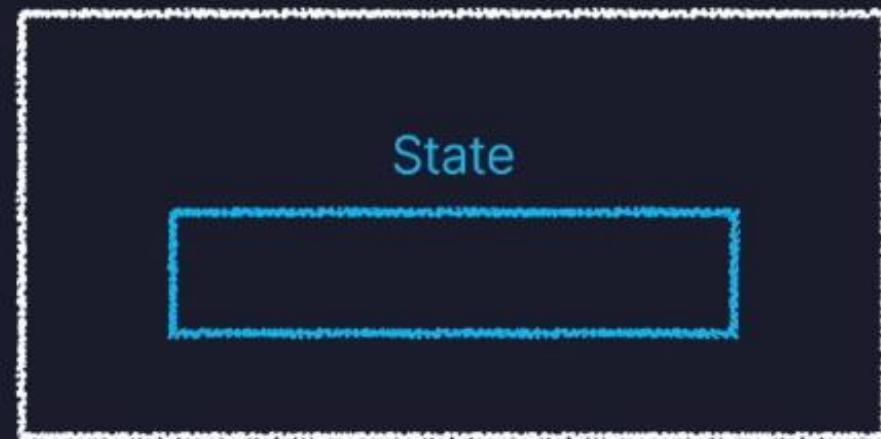
State

리 렌더링 (Re-Rendering)



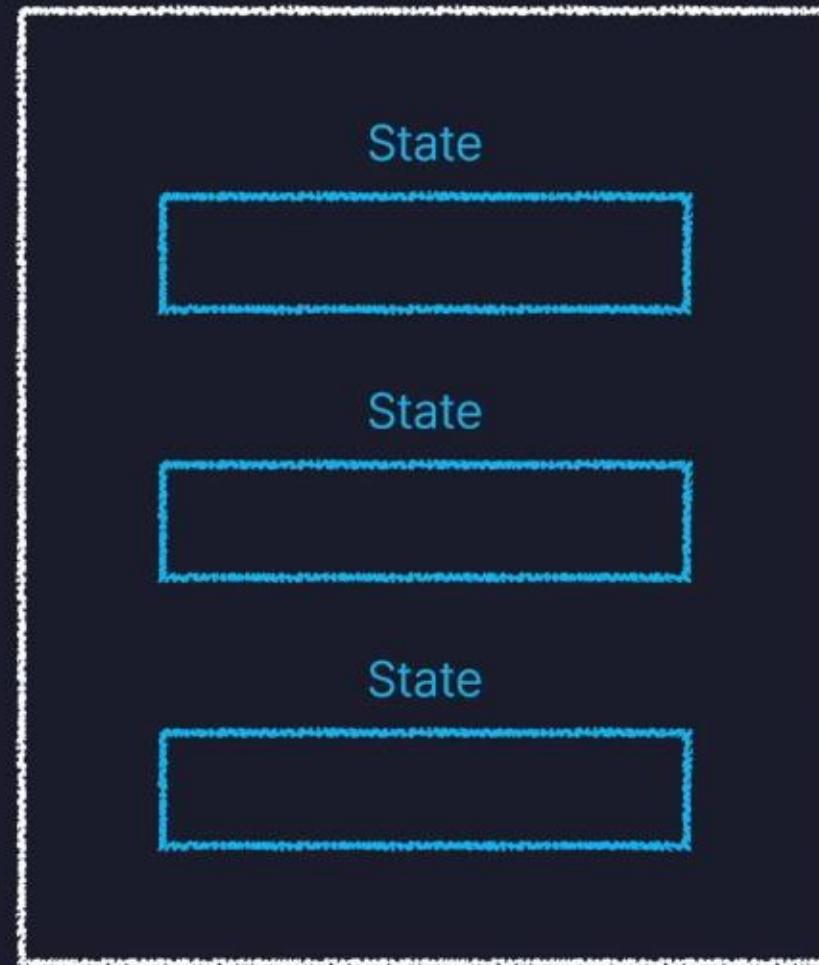
State란?

React 컴포넌트



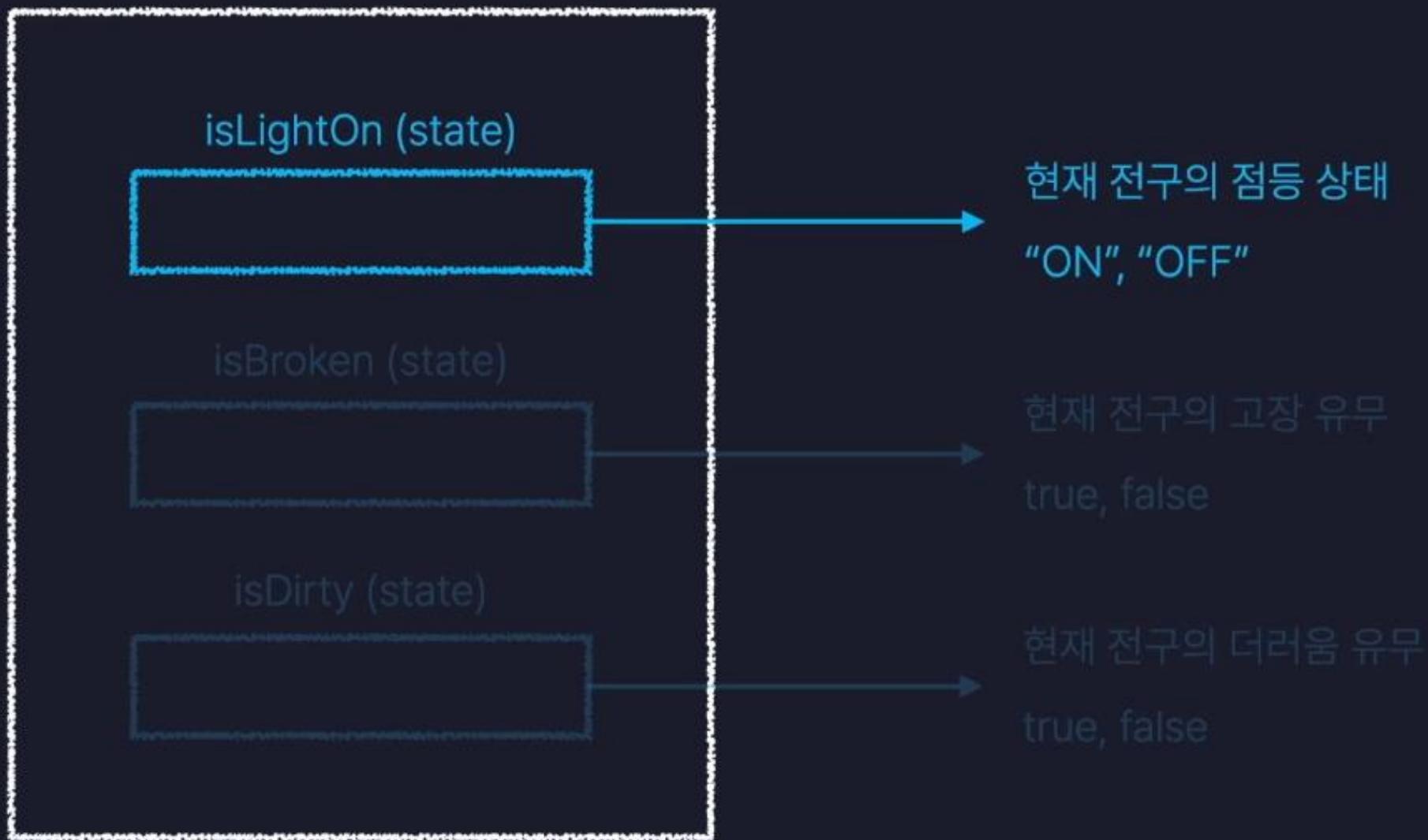
State란?

React 컴포넌트



State란?

React 컴포넌트



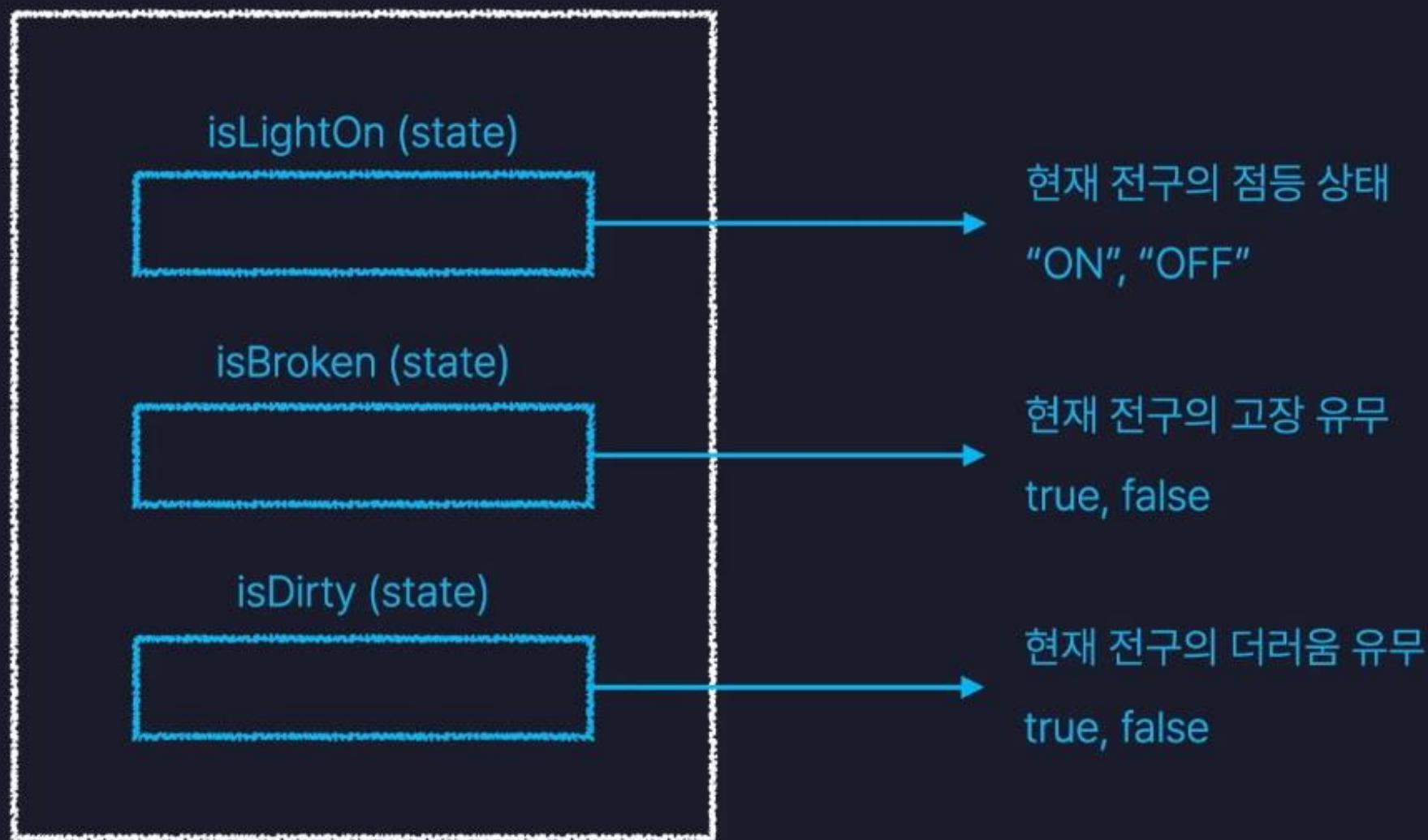
State란?

React 컴포넌트



State란?

React 컴포넌트



한입 크기로 잘라먹는

State를 Props로 전달하기

한입 크기로 잘라먹는

State로 사용자 입력 관리하기 1

한입 크기로 잘라먹는

State로 사용자 입력 관리하기 2

한입 크기로 잘라먹는

useRef - 컴포넌트의 변수 생성하기

useRef란?

useRef

새로운 Reference 객체를 생성하는 기능

```
const refObject = useRef()
```

useRef란?

useRef

새로운 Reference 객체를 생성하는 기능

```
const refObject = useRef()
```



컴포넌트 내부의 변수

useRef란?

useRef

Reference 객체를 생성

컴포넌트 내부의 변수로 활용 가능

어떤 경우에도 리렌더링을 유발하지 않음

useState

State를 생성

컴포넌트 내부의 변수로 활용 가능

값이 변경되면 컴포넌트 리렌더링

useRef란?

```
const refObject = useRef()
```

특정 DOM 요소에 접근

```
<div>
  <textarea
    ref={refObject}
    name="bio"
    value={input.bio}
    onChange={onChange}
  />
</div>
```

useRef란?

```
const refObject = useRef()
```

```
<div>
  <textarea
    ref={refObject}
    name="bio"
    value={input.bio}
    onChange={onChange}
  />
</div>
```

[요소 조작]

Focus!



한입 크기로 잘라먹는

React Hooks

React Hooks란?



React Hooks

클래스 컴포넌트의 기능을

함수 컴포넌트에서도 이용할 수 있도록

React Hooks란?

2017년도 이전의 React.js



Class 컴포넌트

모든 기능을 이용할 수 있음
ex) State, Ref, etc...



Function 컴포넌트

UI 렌더링만 할 수 있음

React Hooks란?

2017년도 이전의 React.js



Class 컴포넌트

모든 기능을 이용할 수 있음
ex) State, Ref, etc...

문법이 복잡함



Function 컴포넌트

UI 렌더링만 할 수 있음

React Hooks란?



Class 컴포넌트

모든 기능을 이용할 수 있음
ex) State, Ref, etc...

React Hooks



Function 컴포넌트

UI 렌더링만 할 수 있음

React Hooks란?



React Hooks란?

React Hooks



사실 이 친구들은 React Hooks 였어요

useState

*State 기능을 낚아채오는 Hook

useRef

*Reference 기능을 낚아채오는 Hook

React Hooks란?

React Hooks



이름 앞에 동일한 접두사 `use` 가 붙음

`useState`

`useRef`

React Hooks란?

React Hooks

useState

useRef

useEffect

useReducer

...

(대략 20개정도)

React Hooks란?

함수 컴포넌트 내부에서만 호출 가능

```
const Register = () => {
  const [input, setInput] = useState({
    name: '',
    birth: '',
    country: '',
    bio: '',
  });
}
```

React Hooks란?

조건문, 반복문 내부에서는 호출 불가

```
if (true) {  
  | const [state, setState] = useState();  
}  
|
```

React Hooks란?

나만의 Hook도 제작 가능 (Custom Hook)

```
function useInput() {  
    // input의 값을 관리하는 Custom Hook  
    // ...  
}
```

한입 크기로 잘라먹는

카운터 앱 - 프로젝트 소개 및 준비

프로젝트 소개. Counter App

Simple Counter

현재 카운트 :

0

-1 -10 -100 +100 +10 +1

프로젝트 소개. Counter App

Simple Counter

현재 카운트 :

0

-1 -10 -100 +100 +10 +1

→ <Viewer/>

프로젝트 소개. Counter App

Simple Counter

현재 카운트 :

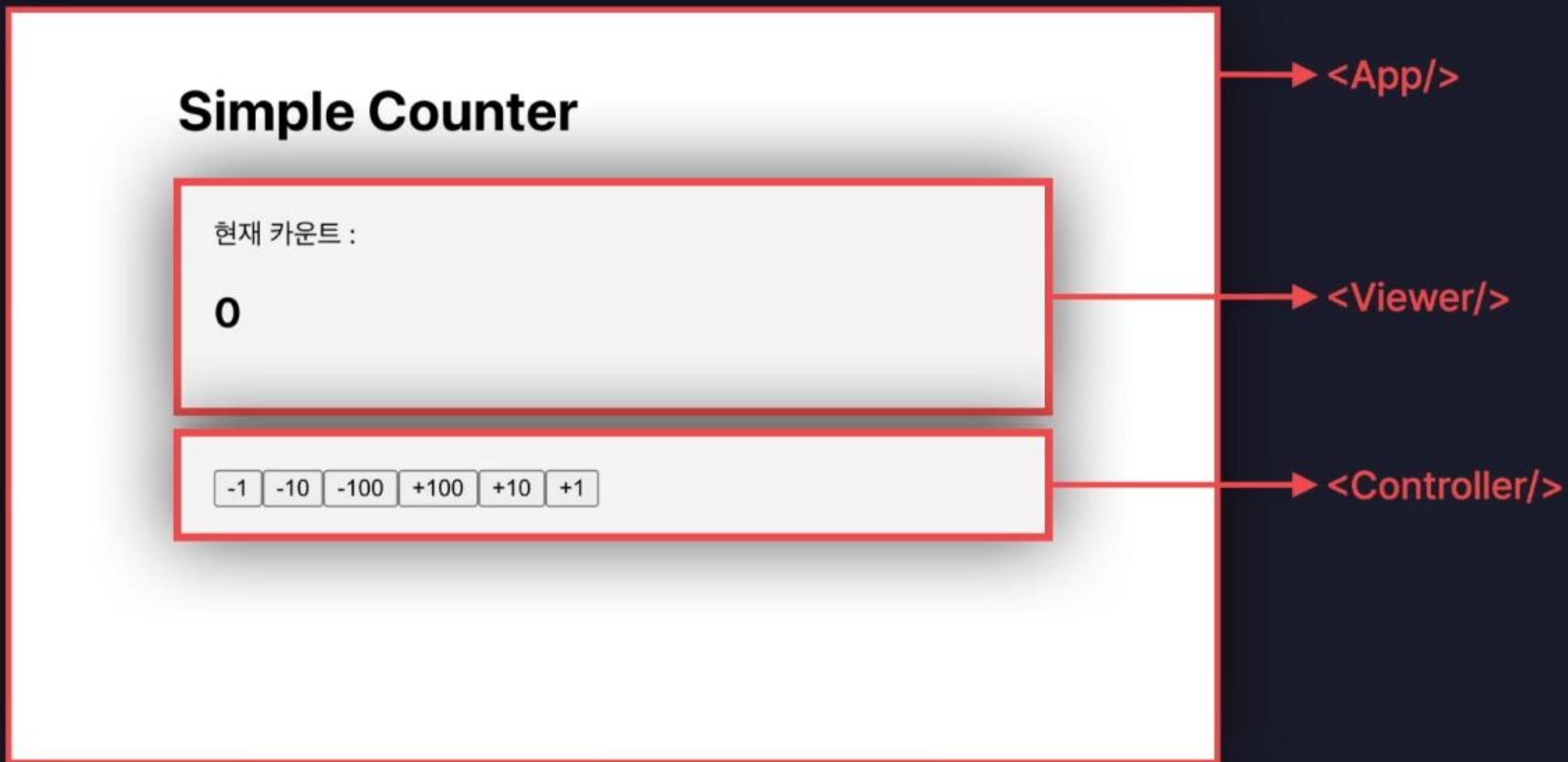
0

-1 -10 -100 +100 +10 +1

→ <Viewer/>

→ <Controller/>

프로젝트 소개. Counter App



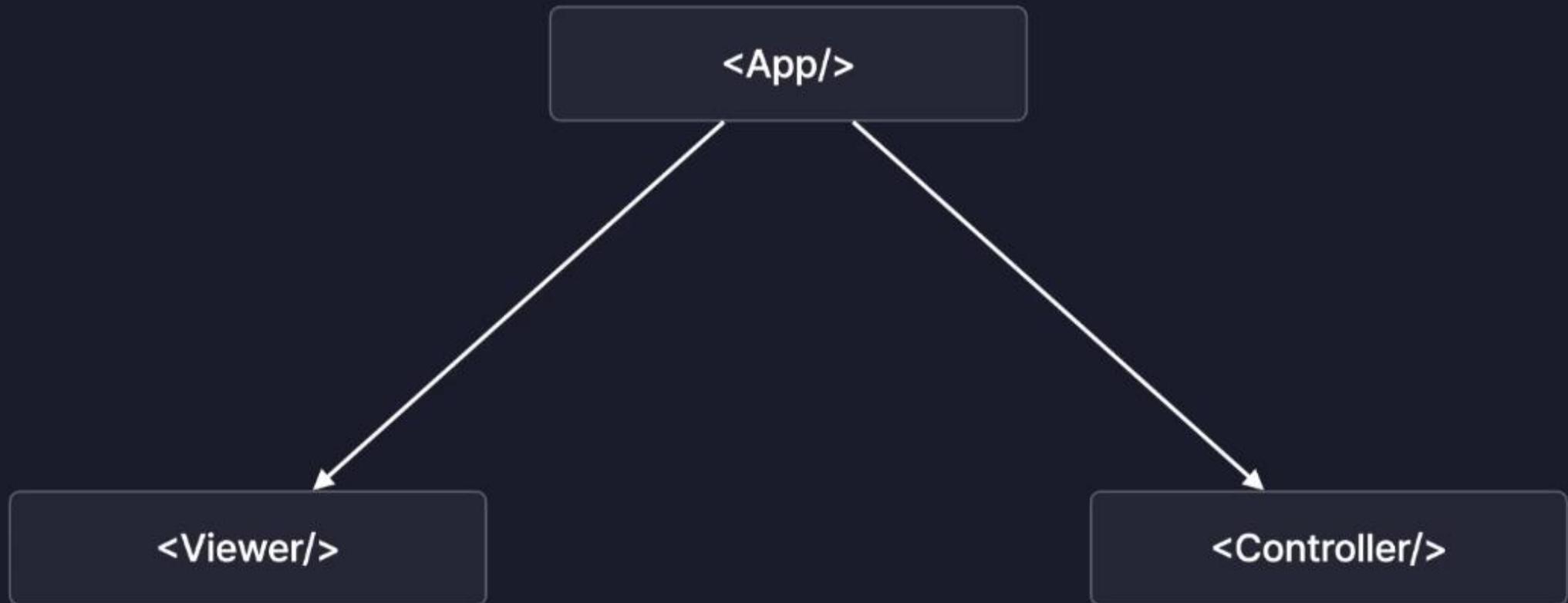
한입 크기로 잘라먹는

카운터 앱 - UI 구현하기

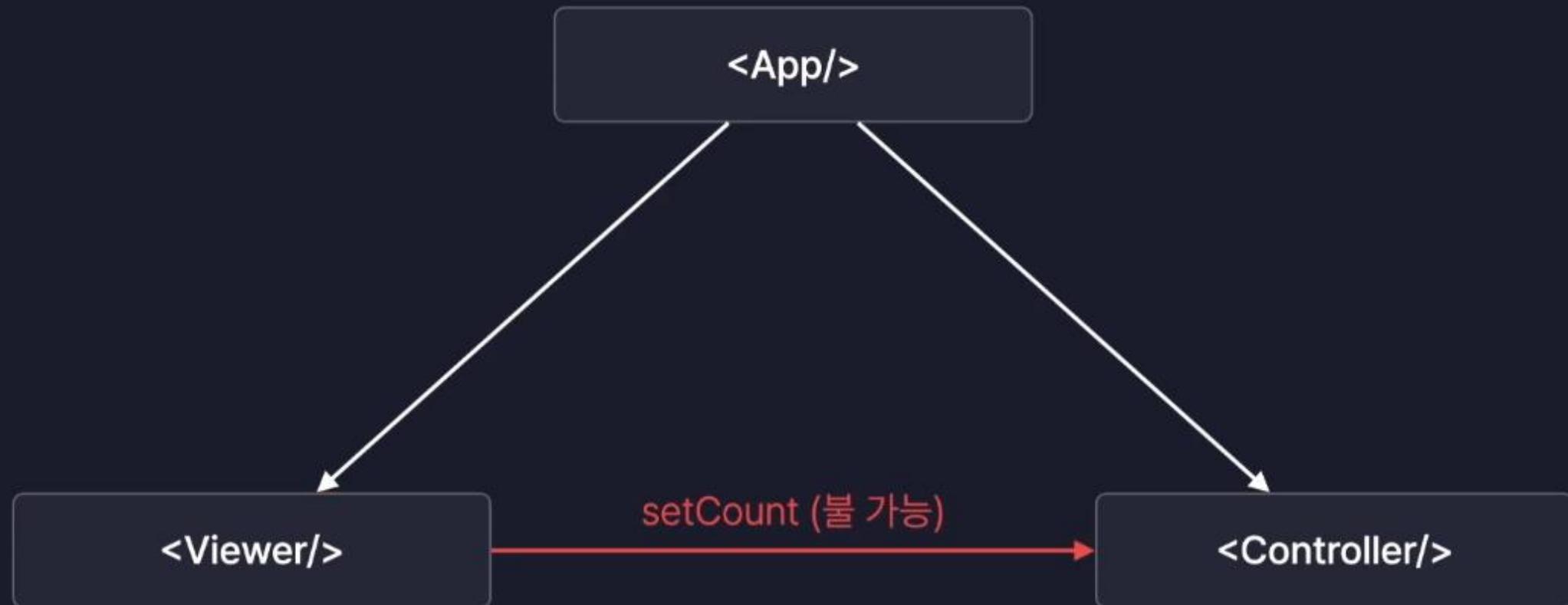
한입 크기로 잘라먹는

카운터 앱 - 기능 구현하기

Counter App의 컴포넌트 계층 구조

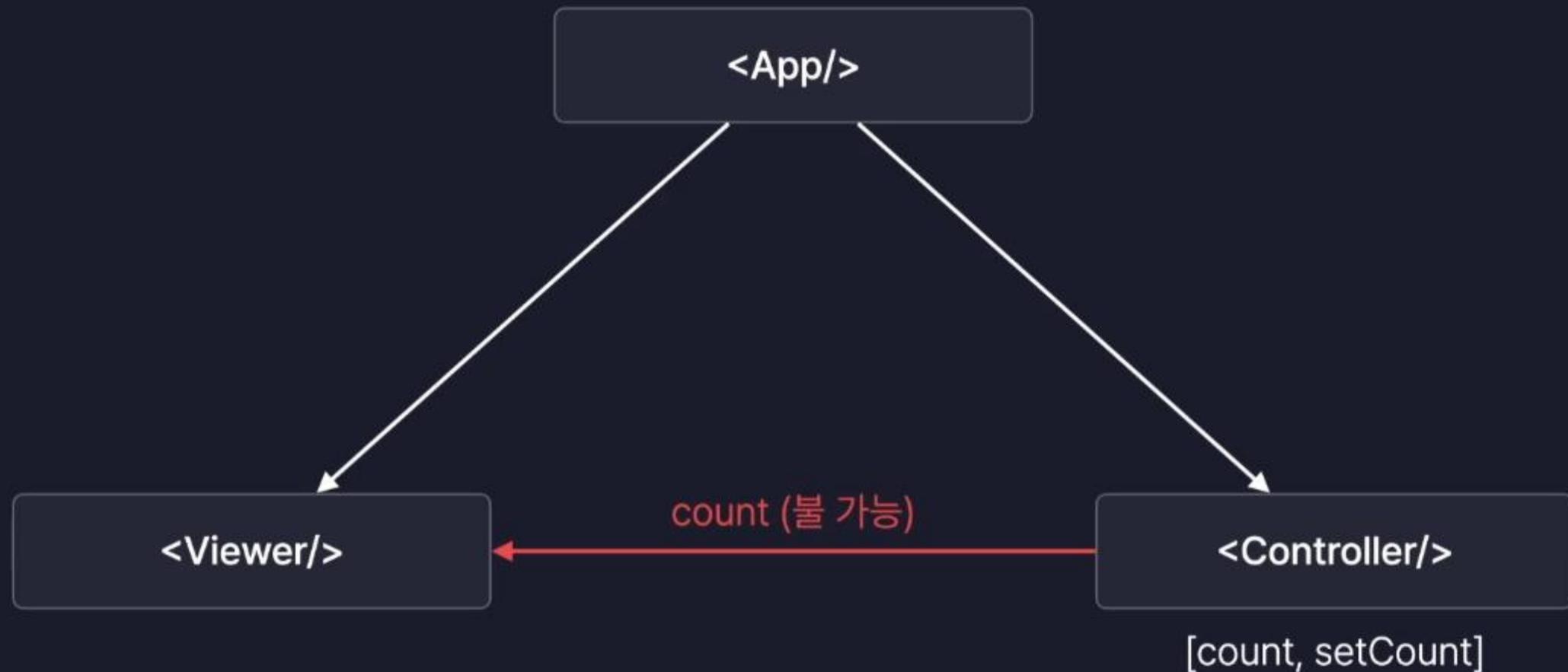


Counter App의 컴포넌트 계층 구조



[count, setCount]

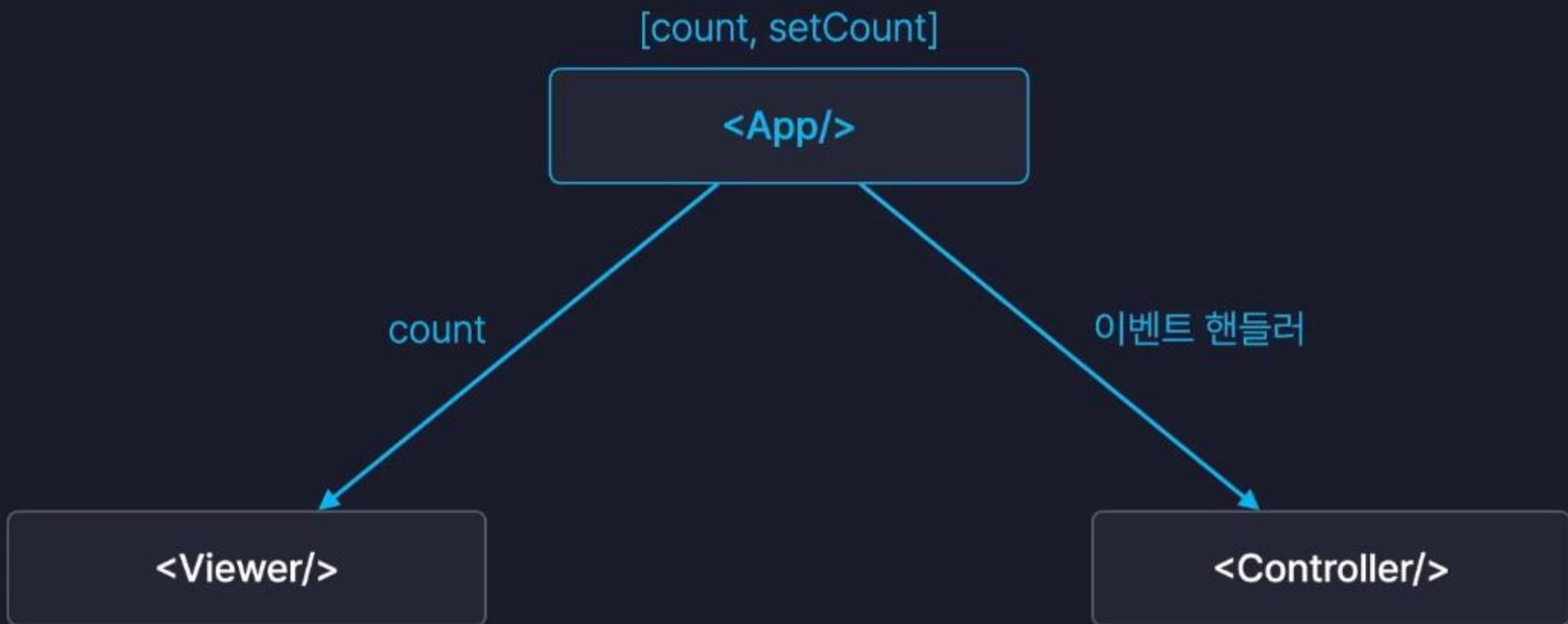
Counter App의 컴포넌트 계층 구조



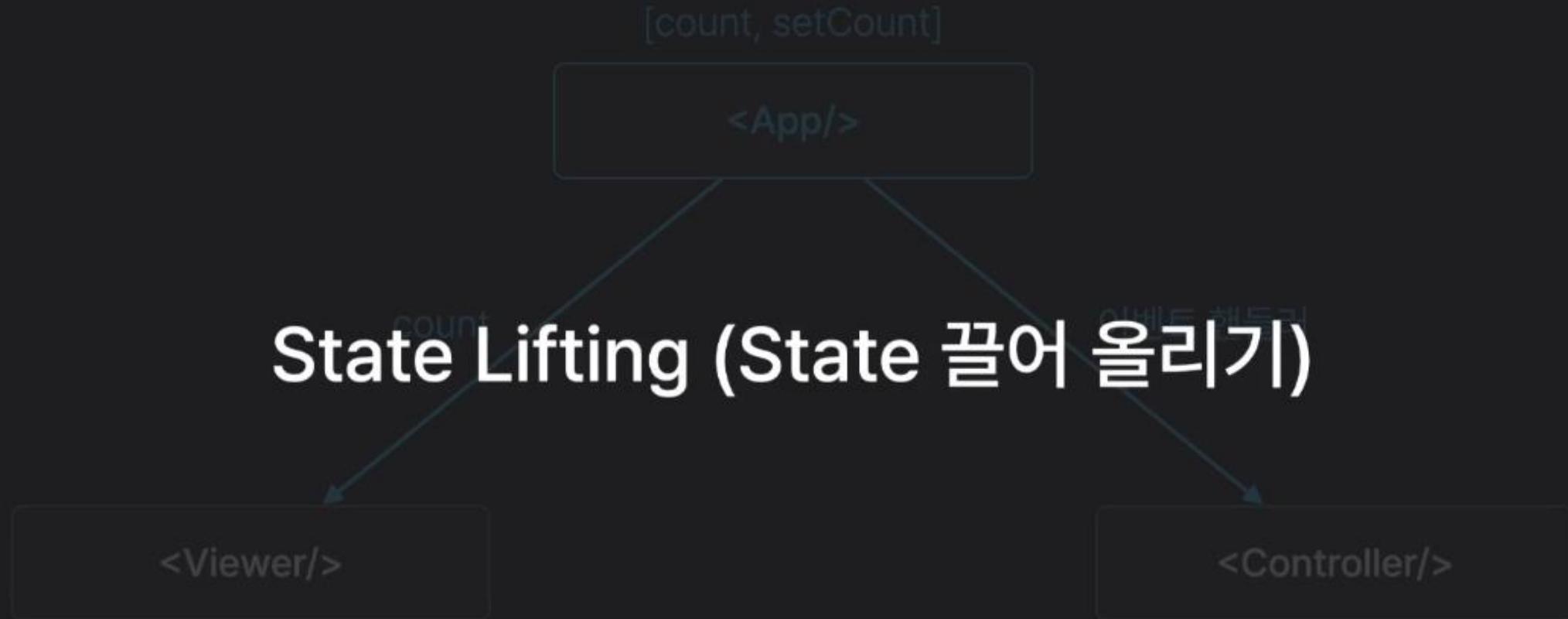
Counter App의 컴포넌트 계층 구조



Counter App의 컴포넌트 계층 구조



State Lifting (State 끌어 올리기)



한입 크기로 잘라먹는

라이프사이클(LifeCycle)

라이프사이클(LifeCycle) = 생애 주기



*출처 : 국민재난안전포털

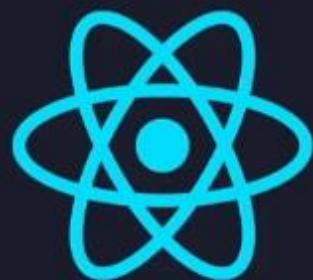
리액트 컴포넌트의 라이프사이클



리액트 컴포넌트의 라이프사이클



리액트 컴포넌트의 라이프사이클



Like. 탄생

- 컴포넌트가 탄생하는 순간
- 화면에 처음 렌더링 되는 순간

"A 컴포넌트가 Mount 되었다!"

⇒ A 컴포넌트가 화면에 처음으로 렌더링 되었다.



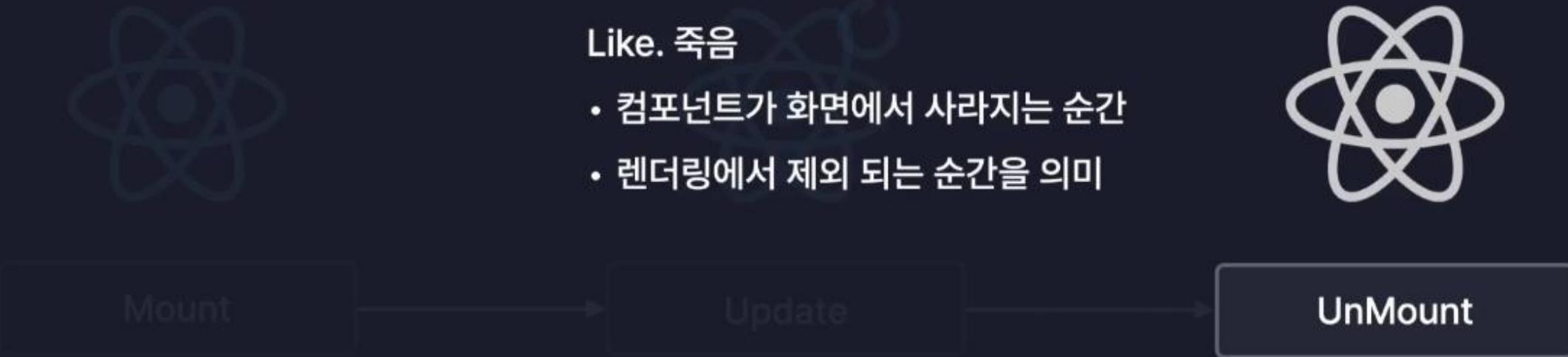
리액트 컴포넌트의 라이프사이클



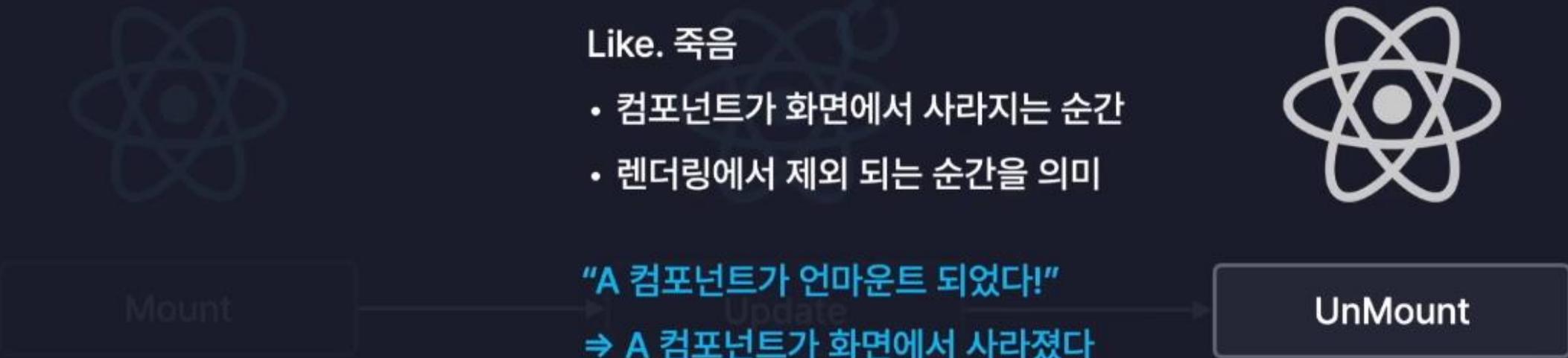
리액트 컴포넌트의 라이프사이클



리액트 컴포넌트의 라이프사이클



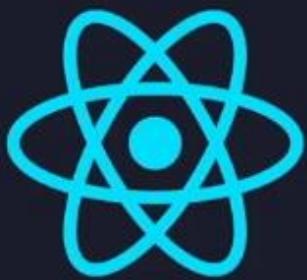
리액트 컴포넌트의 라이프사이클



리액트 컴포넌트의 라이프사이클



리액트 컴포넌트의 라이프사이클



Mount

Update

UnMount

서버에서 데이터를 불러오는 작업!

Like. 변화

Like. 죽음

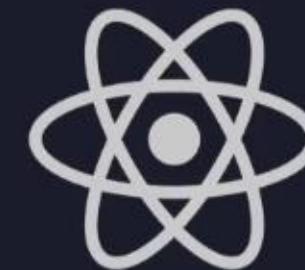
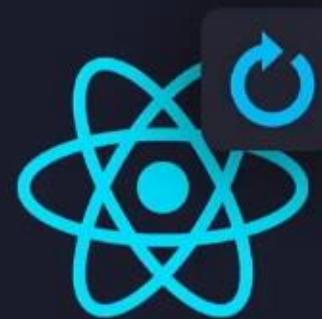
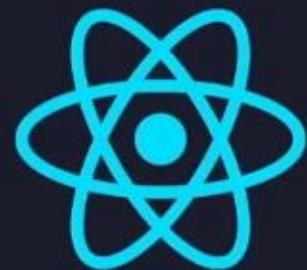
리액트 컴포넌트의 라이프사이클



리액트 컴포넌트의 라이프사이클



리액트 컴포넌트의 라이프사이클



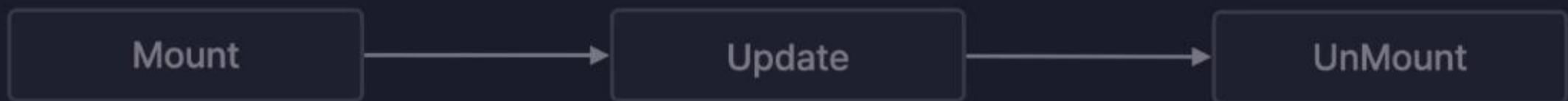
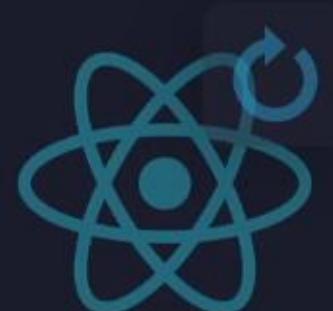
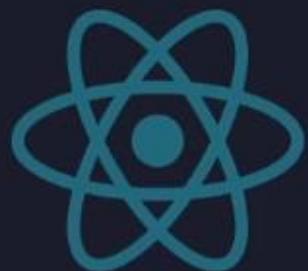
서버에서 데이터를 불러오는 작업

어떤 값이 변경되었는지 콘솔에 출력

컴포넌트가 사용하던 메모리 정리

리액트 컴포넌트의 라이프사이클

useEffect



서버에서 데이터를 불러오는 작업

어떤 값이 변경되었는지 콘솔에 출력

컴포넌트가 사용하던 메모리 정리

한입 크기로 잘라먹는

useEffect 사용하기

useEffect란?

useEffect

리액트 컴포넌트의 사이드 이펙트를 제어하는

새로운 React Hook

사이드 이펙트(SideEffect)란?

사이드 이펙트

우리말로 “**부작용**” 이라는 뜻

사이드 이펙트(SideEffect)란?

사이드 이펙트

우리말로 “부작용”이라는 뜻

리액트에서는 “부수적인 효과”, “파생되는 효과” 정도로 해석 가능

사이드 이펙트(SideEffect)란?

ex)

과식을 하면 살이 찐다

사이드 이펙트(SideEffect)란?

ex)

과식을 하면 살이 찐다

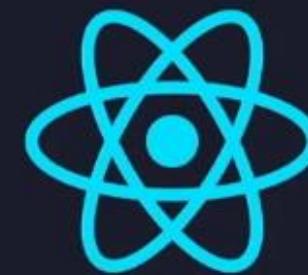


파생되는 효과

사이드 이펙트(Side Effect)

React 컴포넌트의 사이드 이펙트

React 컴포넌트의 사이드 이펙트



컴포넌트의 동작에 따라 파생되는 여러 효과

React 컴포넌트의 사이드 이펙트



사이드 이펙트를 제어하는 useEffect



사이드 이펙트를 제어하는 useEffect

useEffect

컴포넌트 내부의 값 변경

콘솔에 변경된 값 출력

컴포넌트 마운트

콘솔에 “Mount” 라고 출력

컴포넌트 업데이트(리렌더)

콘솔에 “Update” 라고 출력

컴포넌트 언마운트

콘솔에 “Unmount” 라고 출력

한입 크기로 잘라먹는

useEffect로 라이프사이클 제어하기

한입 크기로 잘라먹는

React 개발자 도구 사용하기



chrome 웹 스토어

디스커버

확장 프로그램

테마



React Developer Tools

추천 4.0 ★ (평점 1.5천개)

[확장 프로그램](#)[개발자 도구](#)

4,000,000 사용자

The screenshot shows the React Developer Tools extension running in a browser. The main window displays a 'todos' application with a header, a list of todos, and a footer with three links. In the bottom right corner of the browser window, the React DevTools interface is visible. It has tabs for Elements, Sources, Console, Components, Profiler, Performance, and Network. The Components tab is active, showing a tree structure of the React component hierarchy. The 'TodoTextInput' component is selected in the tree, highlighted with a blue background. To the right of the tree, detailed information about this component is shown: its props (including 'newTodo' with a checked checkbox, 'onSave' function, and placeholder text 'What needs to be done?'), its hooks (a single 'State' hook with the value 'Try React DevTools'), and the components it is rendered by ('Header' and 'App').

한일 크기로 잘라먹는

TodoList - 프로젝트 소개 및 준비

오늘은 

Fri Mar 15 2024

새로운 Todo...

추가

Todo List 

검색어를 입력해주세요

-
- | | | | |
|--------------------------|------------|--------------|--------------------|
| <input type="checkbox"/> | React 공부하기 | 2024. 3. 15. | 삭제 |
|--------------------------|------------|--------------|--------------------|

-
- | | | | |
|--------------------------|------|--------------|--------------------|
| <input type="checkbox"/> | 빨래하기 | 2024. 3. 15. | 삭제 |
|--------------------------|------|--------------|--------------------|

-
- | | | | |
|--------------------------|---------|--------------|--------------------|
| <input type="checkbox"/> | 노래 연습하기 | 2024. 3. 15. | 삭제 |
|--------------------------|---------|--------------|--------------------|

한입 크기로 잘라먹는

ToDoList - UI 구현하기

한입 크기로 잘라먹는

ToDoList - 기능 구현 준비하기

한입 크기로 잘라먹는

TodoList - Create(투두 추가하기)

한일 크기로 잘라먹는

TodoList - Read (투두 리스트 렌더링)

한입 크기로 잘라먹는

TodoList - Update(투두 수정하기)

한일 크기로 잘라먹는

TodoList - Delete(투두 삭제하기)