
쉽고 재미있게 이해하는 Reactive Programming

홍승환 (@harrydrippin)
harrydrippin@jininsa.com



GDG Korea Campus
Campus Summer Party Technical Talk 2nd Session

발표자 소개



홍승환 @harrydrippin

harrydrippin@jininsa.com

github.com/harrydrippin

Career

Jininsa Company

소속 개발자 (2017. 6. 26. ~)

국민대학교 소프트웨어융합대학 소프트웨어학부

2학년 재학 중, 16학번 (2016. 3. 2. ~)

Works

2016년도 창의도전형 R&D 지원 사업 총괄책임자
최우수상 (미래창조과학부장관상)

아라 : 한글 프로그래밍 언어 개발

2015학년도 한국정보올림피아드 공모 부문(고등부) 동상

Reactive?

Reactive Programming

형용사

1. 반응[반작용]을 보이는
2. (화학) 반응을 하는

=

반응형 프로그래밍

프로그램의 흐름

```
if (isGDG()) {  
    awesome();  
} else {  
    notAwesome();  
}
```

우리가 한 프로그래밍은
이런 코드가 한 줄 한 줄
순서대로 실행되는 방식입니다

```
if (isGDG()) {  
    awesome();  
} else {  
    notAwesome();  
}
```

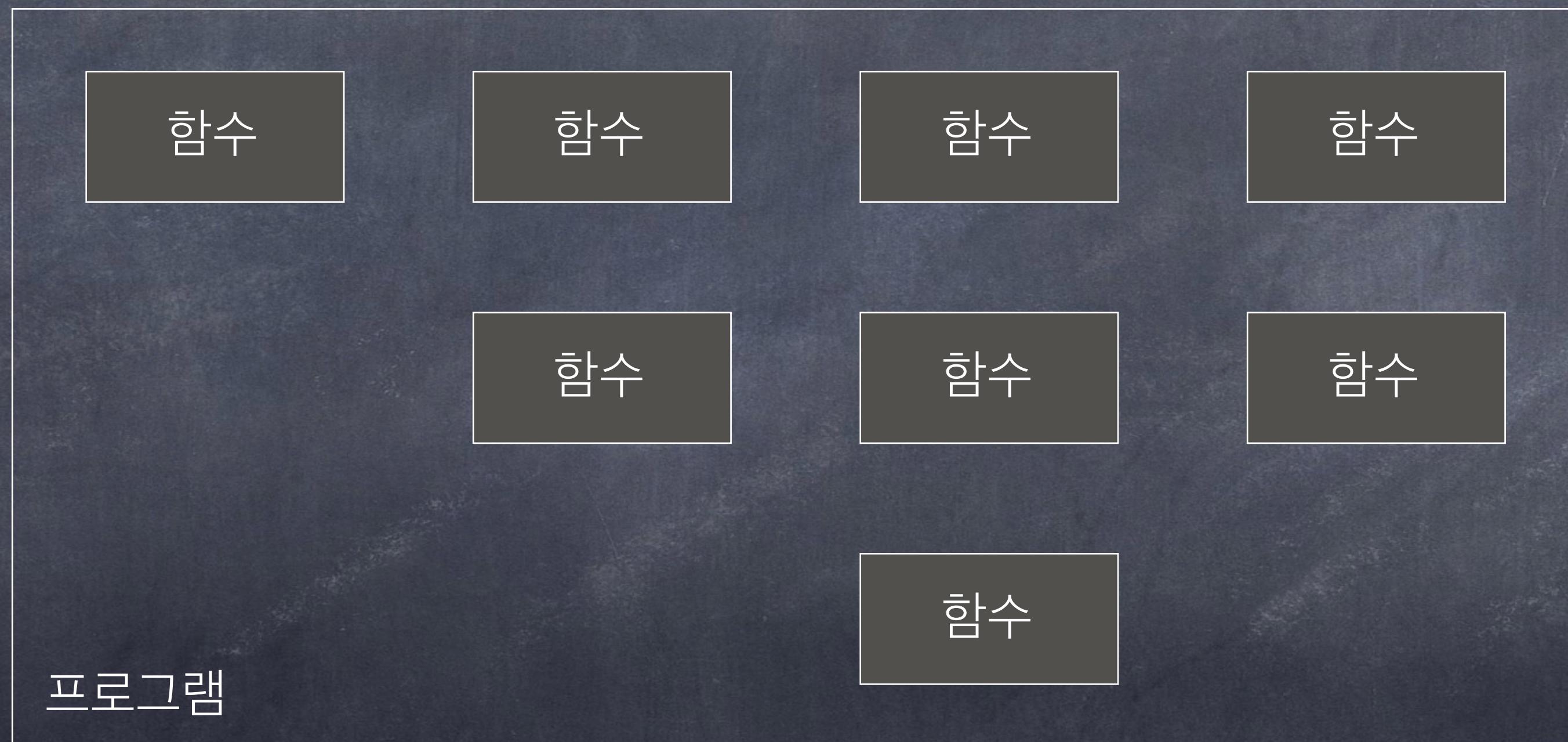
이게 바로
명령형 프로그래밍 Imperative Programming
컨트롤이 흐르는 방식이죠

만약
데이터가 직접
프로그램으로 흐른다면?

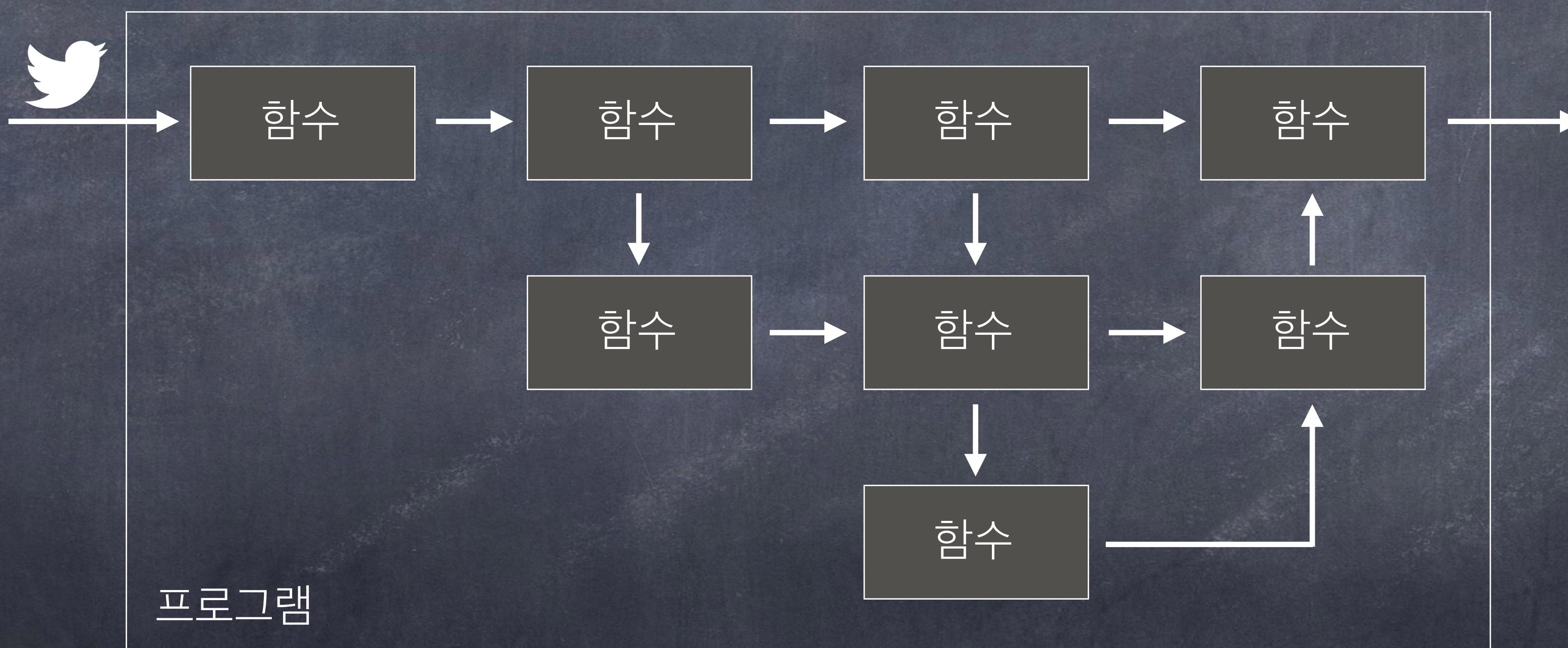


데이터의 흐름

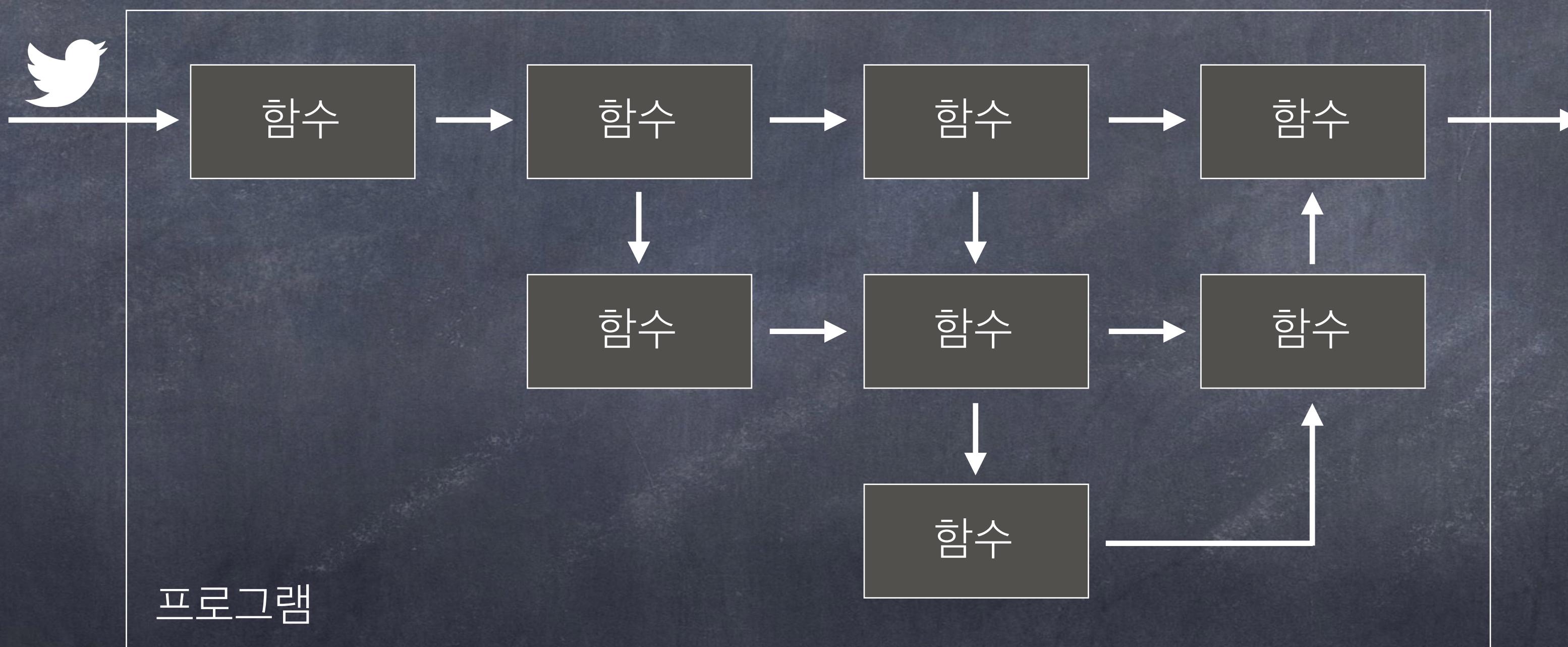
하나의 작업을 하나의 함수가 처리하고
프로그램은 그 함수의 집합인 구조일 때



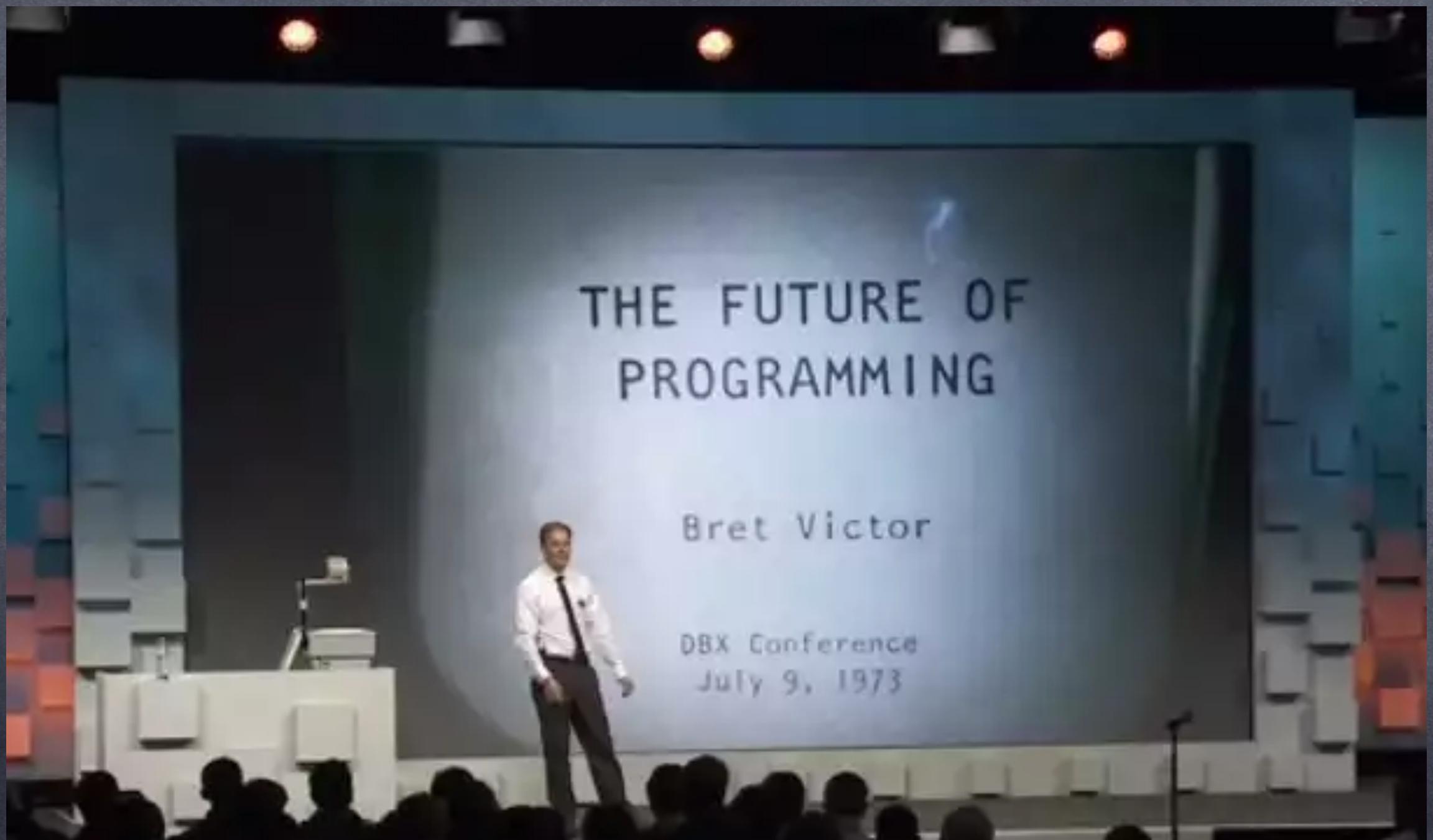
데이터가 함수 사이사이로
흐를 수 있도록 구조를 설계해서



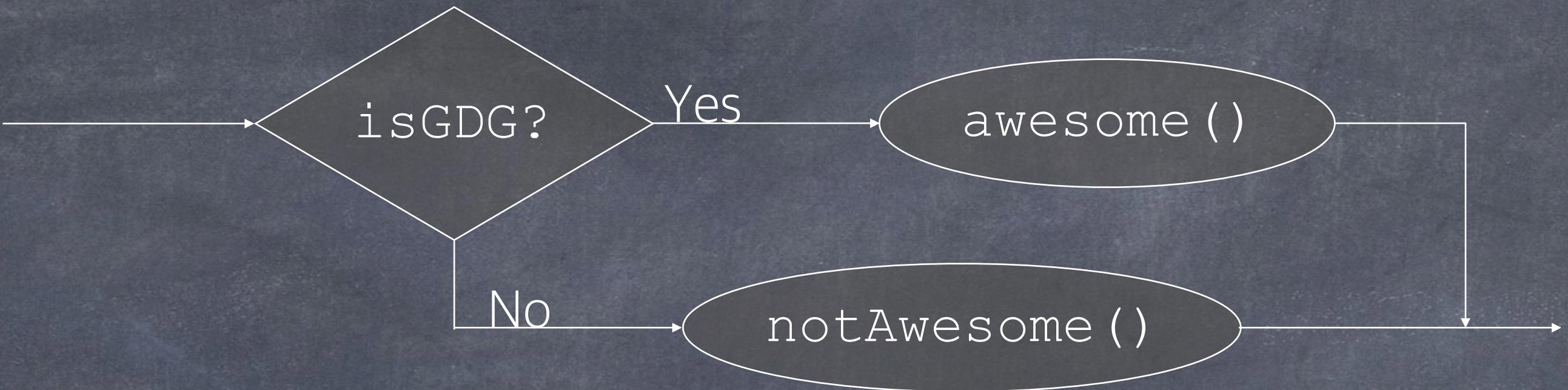
컨트롤 대신
데이터가 직접 흐르게 구성합니다



이거 나온지 40년도 넘은 개념이에요



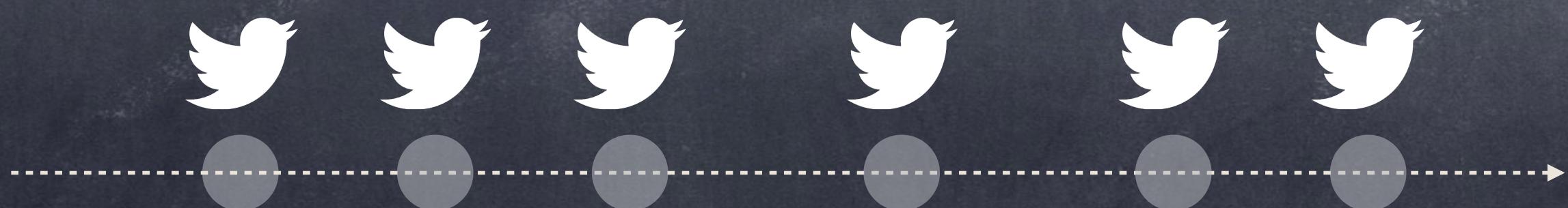
왜 Reactive해야 하는가?

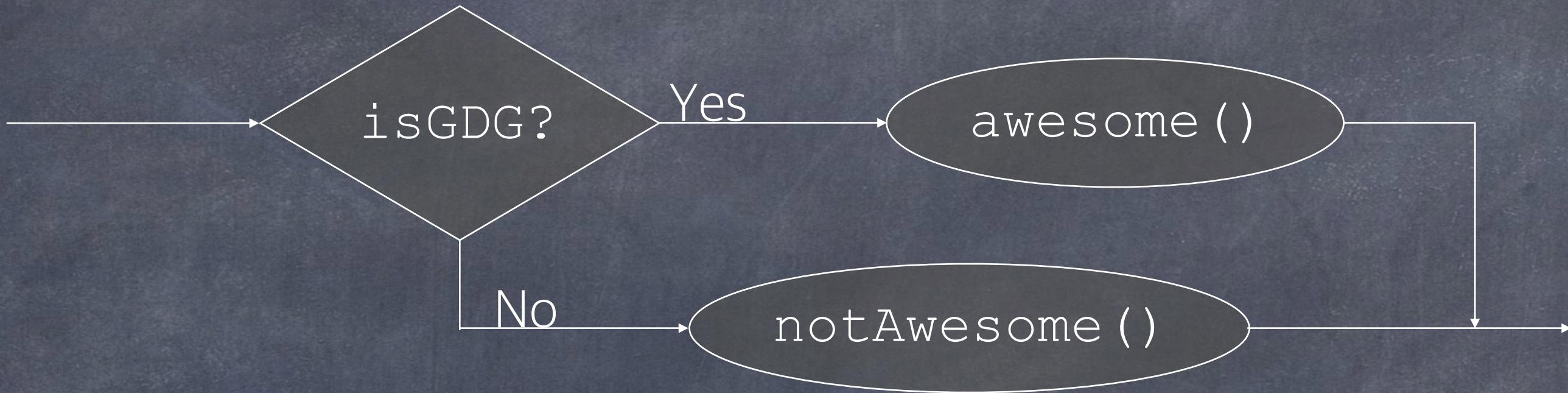


제어 구조의 흐름

VS

데이터의 흐름





제어 구조를 흐르게 하는 방식이
1970년대부터 유행했습니다
우리도 그렇게 배웠죠

하지만
하드웨어의 빠른 발전과
인터넷의 매우 빠른 확산이
이 모든 것을 바꾸기 시작했습니다

1999년 발표자가 2살

인터넷 사용자 280,000,000명
인터넷뱅킹이 생긴 지 5년 밖에 안되었을 시절
Java가 인기를 끌기 시작한 시점

“Write Ones, Run Anywhere”

1999년 발표자가 2살

인터넷 사용자 280,000,000명

인터넷뱅킹이 생긴 지 5년 밖에 안되었을 시절
Java가 인기를 끌기 시작한 시점

“Write Ones, Run Anywhere”

JVM만 깔면 아무데서나 다 돌아가요!

Cloud 같이 엄청 동시에 많이 접속해서 아무데서나 돌릴 수 있어요!

2005년

인터넷 사용자 1,000,000,000명
Facebook 사용자 5,500,000명
Youtube가 2월에 막 생겼음
Twitter랑 Netflix는 있지도 않았을 시절

2014년 0이 몇개야

인터넷 사용자 2,950,000,000명
중국에서만 640,000,000명 접속
미국에서는 280,000,000명 이용
Facebook 사용자 1,300,000,000명
Twitter 사용자 270,000,000명

정말 순식간에 IT 업계는
가파른 기울기로 발전했습니다

진짜 얼마 전까지만 해도 우리들은
이런 서비스들을 별 불편없이 이용해왔습니다

10~50+ 서버

3~6h 유지 보수 시간

1~9s 응답 시간

GB 주처리 데이터 단위

그런데 지금은 세상이
순식간에 너무 많이 달라졌습니다

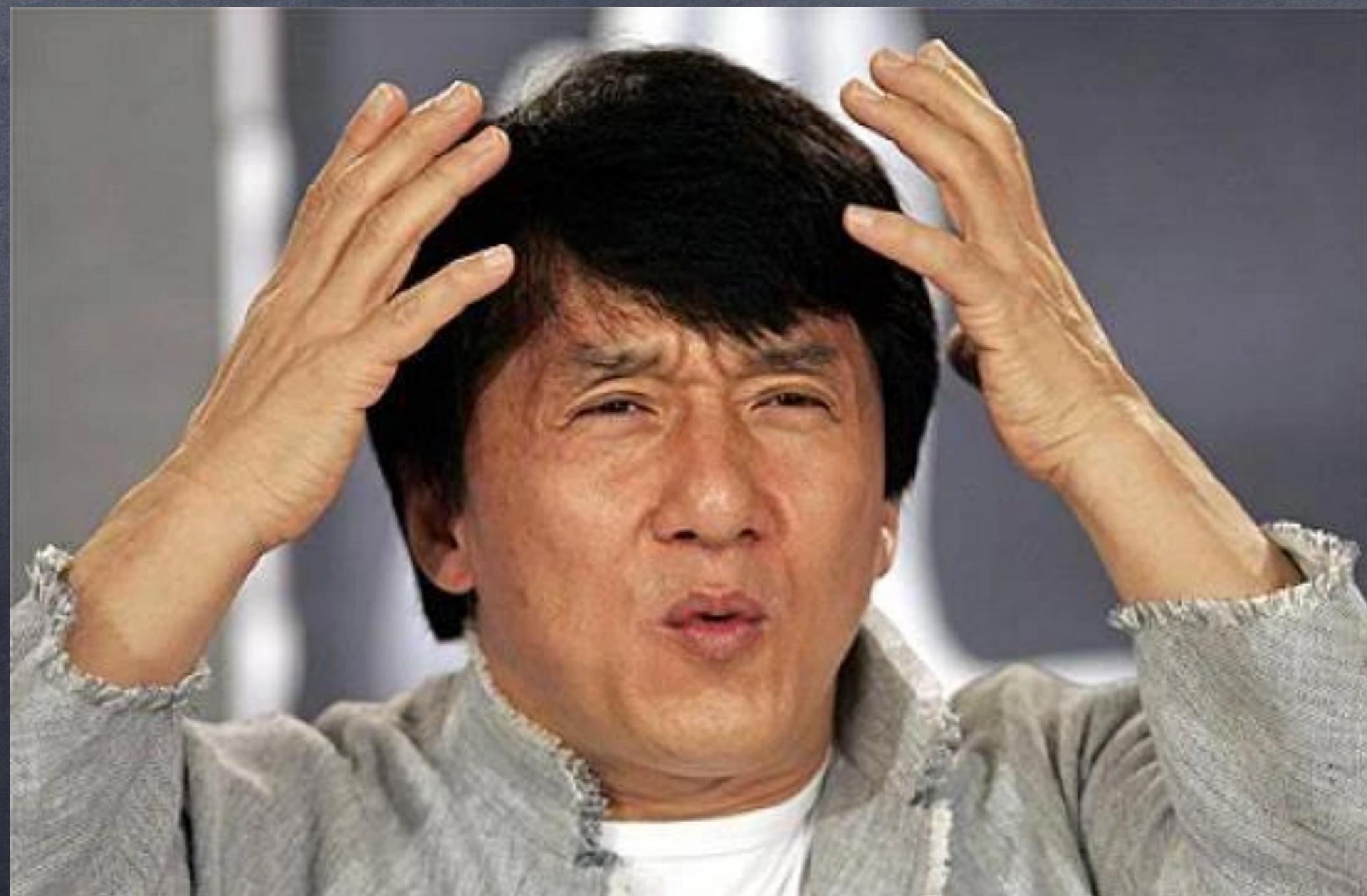
1000+ 클라우드 서버

1/1000 S 응답 시간

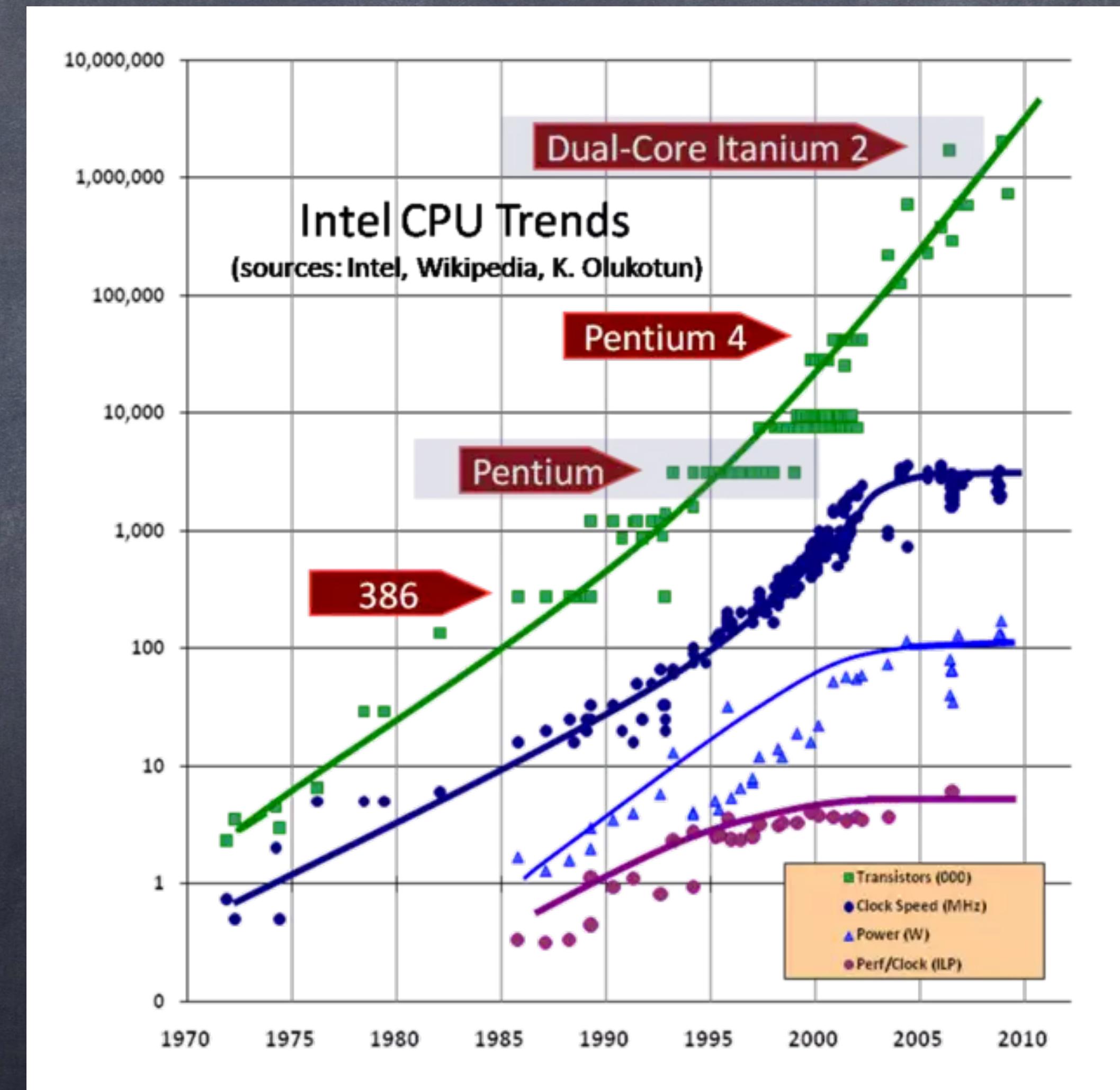
0초 유지 보수 시간

PB 주처리 데이터 단위

“와… 이 정도면 CPU가
못 버티는 거 아니야?”



네, 못 버립니다.



하지만 지금은

프로세서 하나만의 성능이 한계에 부딪혔지만
다들 웬만하면 Multi-threading을 하고
휴대폰에 코어 16개를 박아넣는 시대

동시에 여러가지를 처리하기가 쉬워졌습니다

생각하는 방식을
바꿔야 할 때입니다



Reactive Programming

“

제가 생각하기에 우리들은 두 가지의 Key Point에 관심이 있습니다:

첫째, Stream에 대한 적용이 별도의 버퍼링 없이 존재할 수 있다는 것과 자원 소비율에 따라서 여러 Reactive, Interactive한 모델들 사이를 자동적으로 전환할 수 있다는 것에 대한 확인입니다.

둘째, 우리는 라이브러리, 시스템, 네트워크, 프로세스들 사이에 상호 운용을 허용하는 솔루션을 원합니다.

”

Ben Christensen

Senior Engineer at Netflix, Reactive Streams co-founder, creator of RxJava

Reactive의 세상

명령형 Imperative 이 아닌
선언형 Declarative 프로그래밍

순차적 Procedural 이 아닌
동시적 Concurrent 프로그래밍

선언형 프로그래밍 (Declarative Programming)

선언형 프로그래밍이란,
프로그램이 어떤 방법으로
처리해야 하는지를 나타내기보다
무엇과 같은지를 설명하는 방식입니다

명령형은 구체적인
작동 순서에 집중합니다

```
// Imperative Programming
var array = [1, 2, 3, 4, 5];
var sum = 0;

for (var i = 0; i < array.length; i++) {
  if (array[i] >= 3) sum += array[i];
}

console.log(sum) // => 12
```

하지만 선언형은
알고리즘 그 자체에 집중합니다

```
// Declarative Programming
var array = [1, 2, 3, 4, 5];

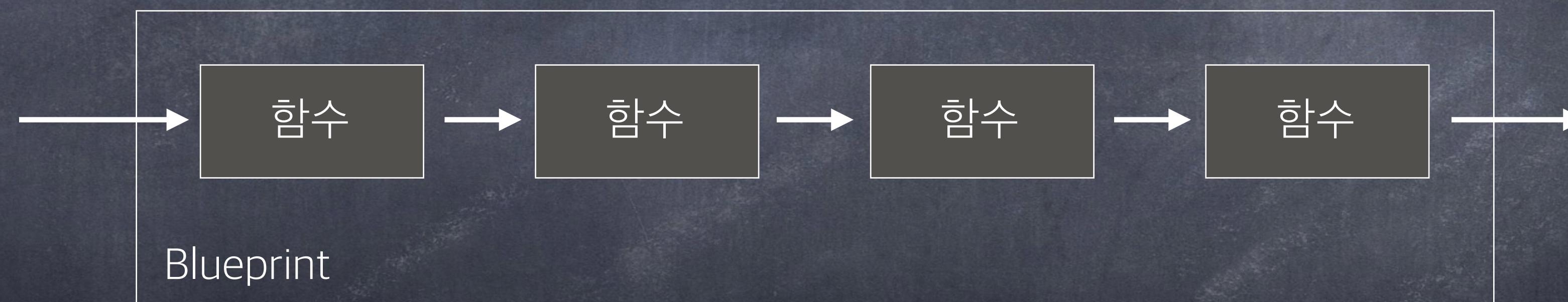
console.log(
  array
    .filter(function(value) {
      return value >= 3;
    }) // => [3, 4, 5]
    .reduce(function(previous, current) {
      return previous + current;
    }); // => 12
);
```

마치 데이터가 걷는 길을 선언하듯 코드를 구성합니다

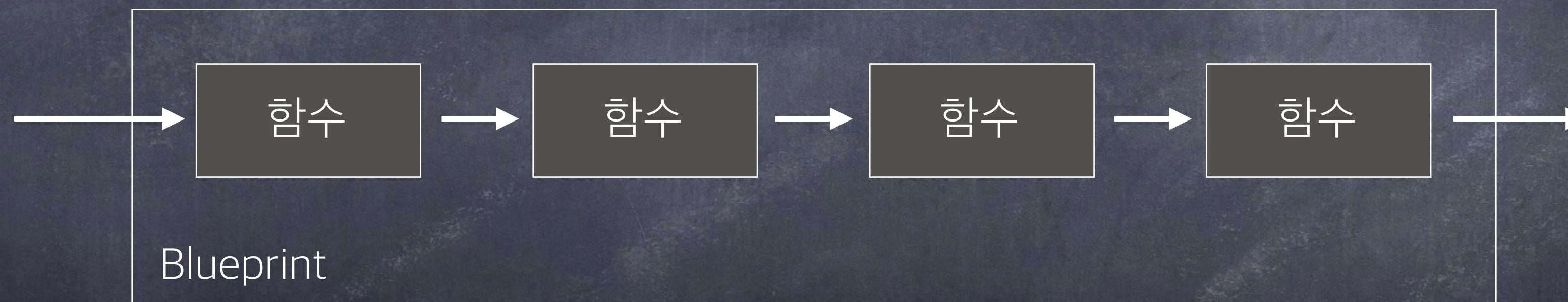
```
// Declarative Programming
var array = [1, 2, 3, 4, 5];

console.log(
  array
    .filter(function(value) {
      return value >= 3;    ① 배열에서 3 이상의 값을 걸러냄
    }) // => [3, 4, 5]
    .reduce(function(previous, current) {
      return previous + current;  ② 모든 값을 합침
    }) ; // => 12
);
```

마치
데이터가 오면 어느 길로 안내할지
Blueprint를 만들어놓는 거죠



마치
~~데이터가 오면 어느 길로 안내할지~~
이벤트가 발생하면 어떻게 반응할지
Blueprint를 만들어놓는 거죠



Reactive Manifesto

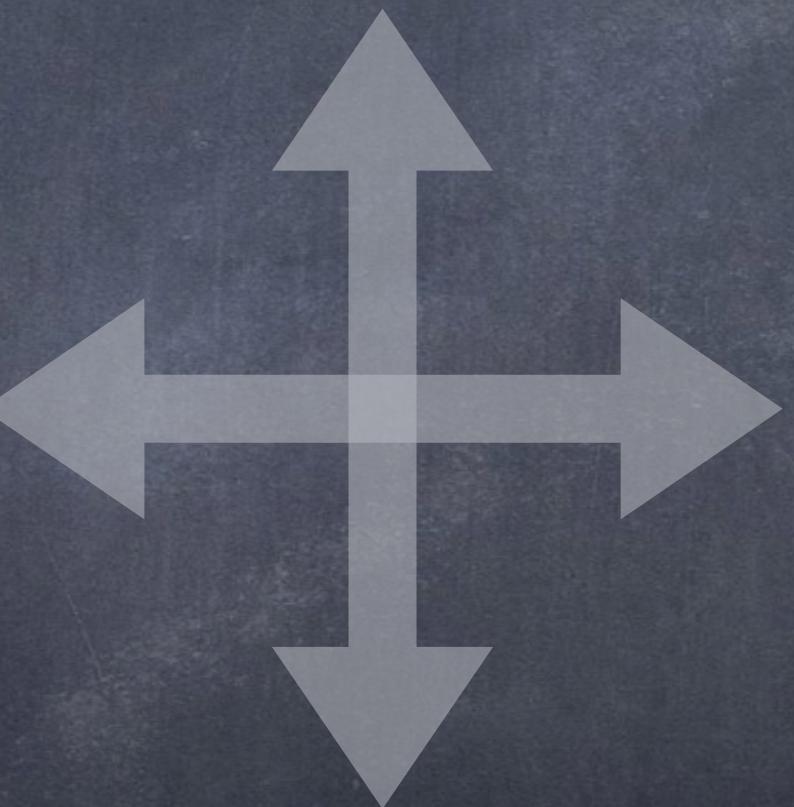
Reactive Manifesto 리액티브 선언문

<http://www.reactivemanifesto.org>

Reactive System의 조건

Responsive 사용자에 대한 반응

Elastic 부하에 대한 반응

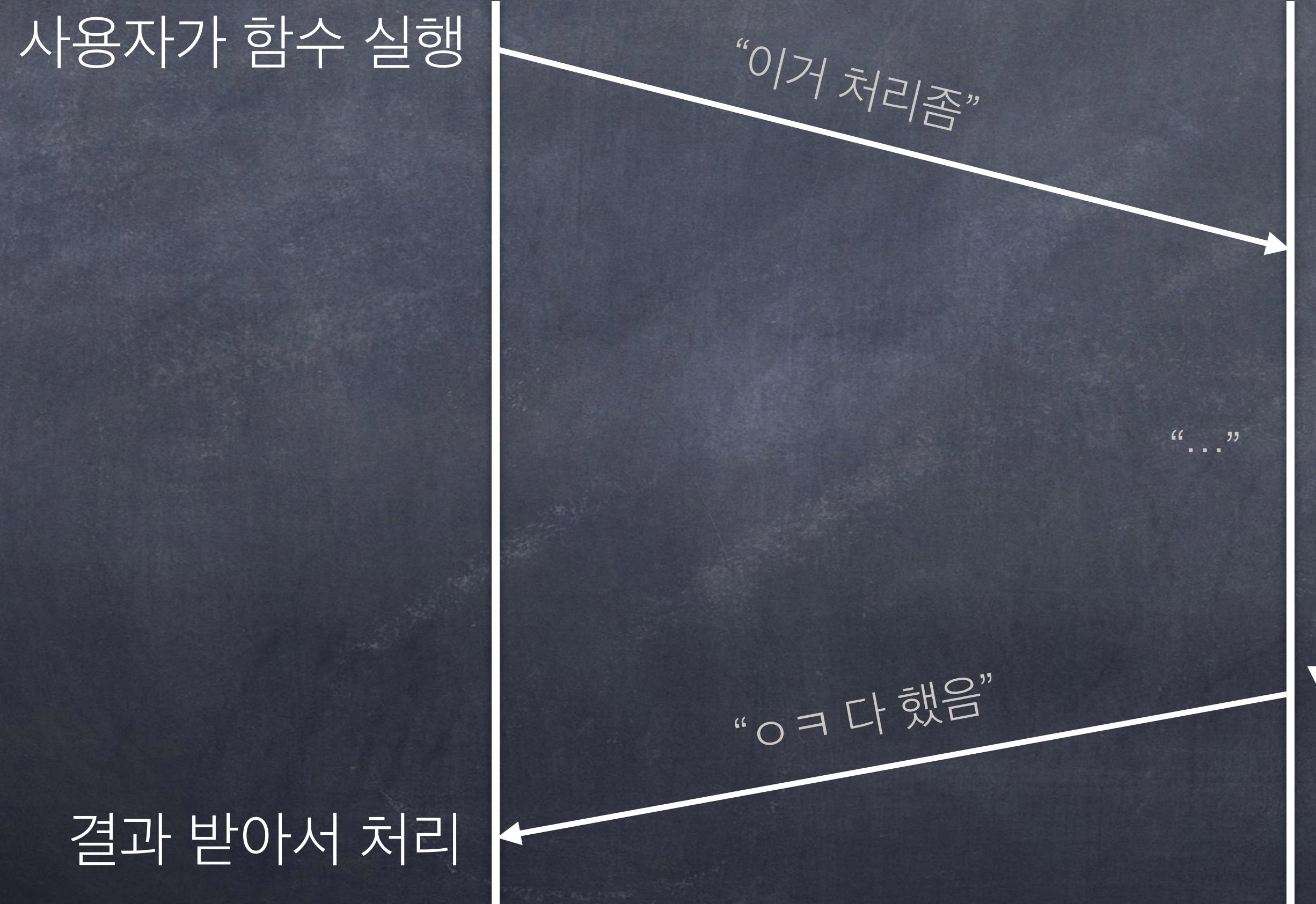


Resilient 장애에 대한 반응

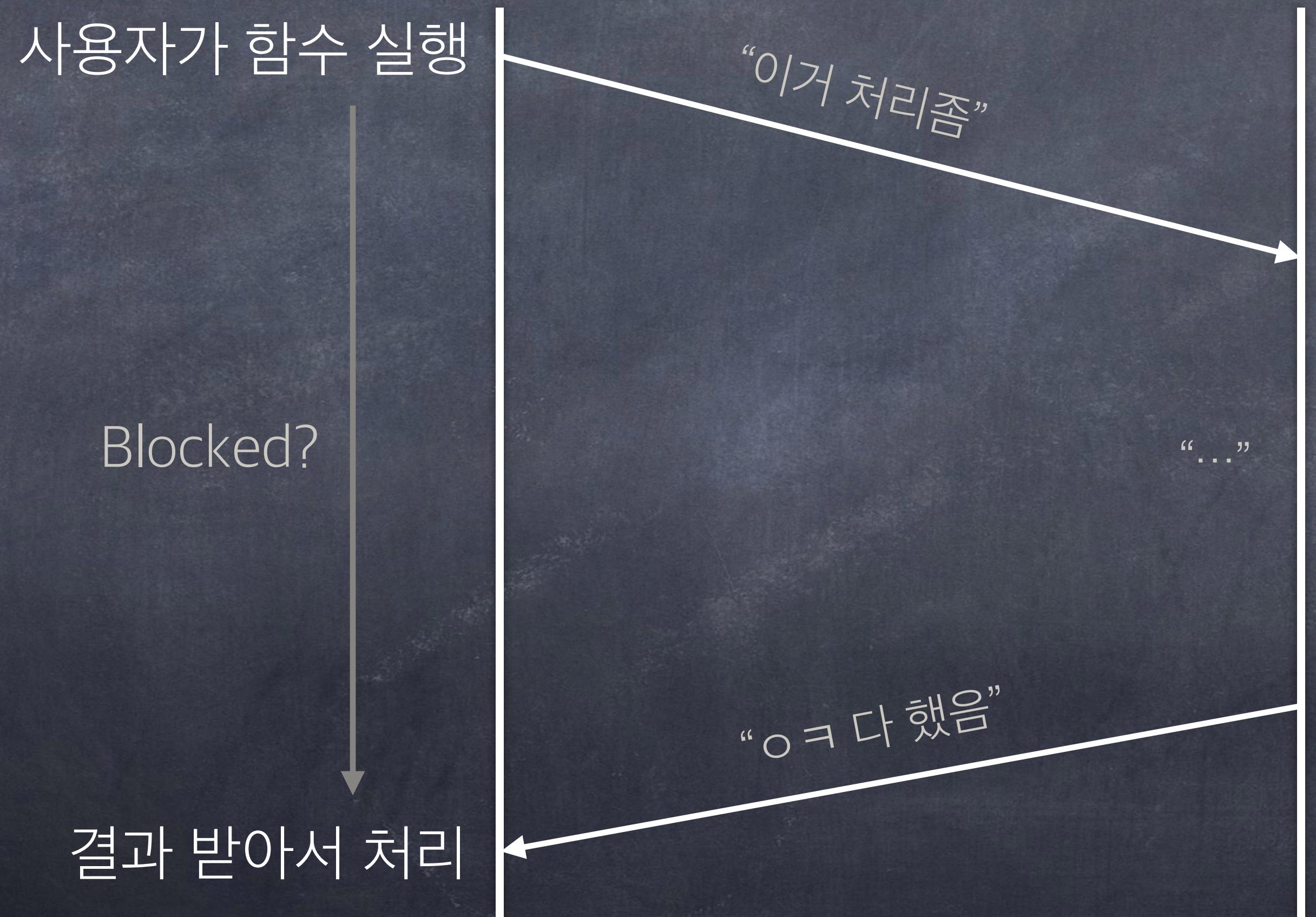
Message Driven 이벤트에 대한 반응

우리가 지금까지 해왔던 것

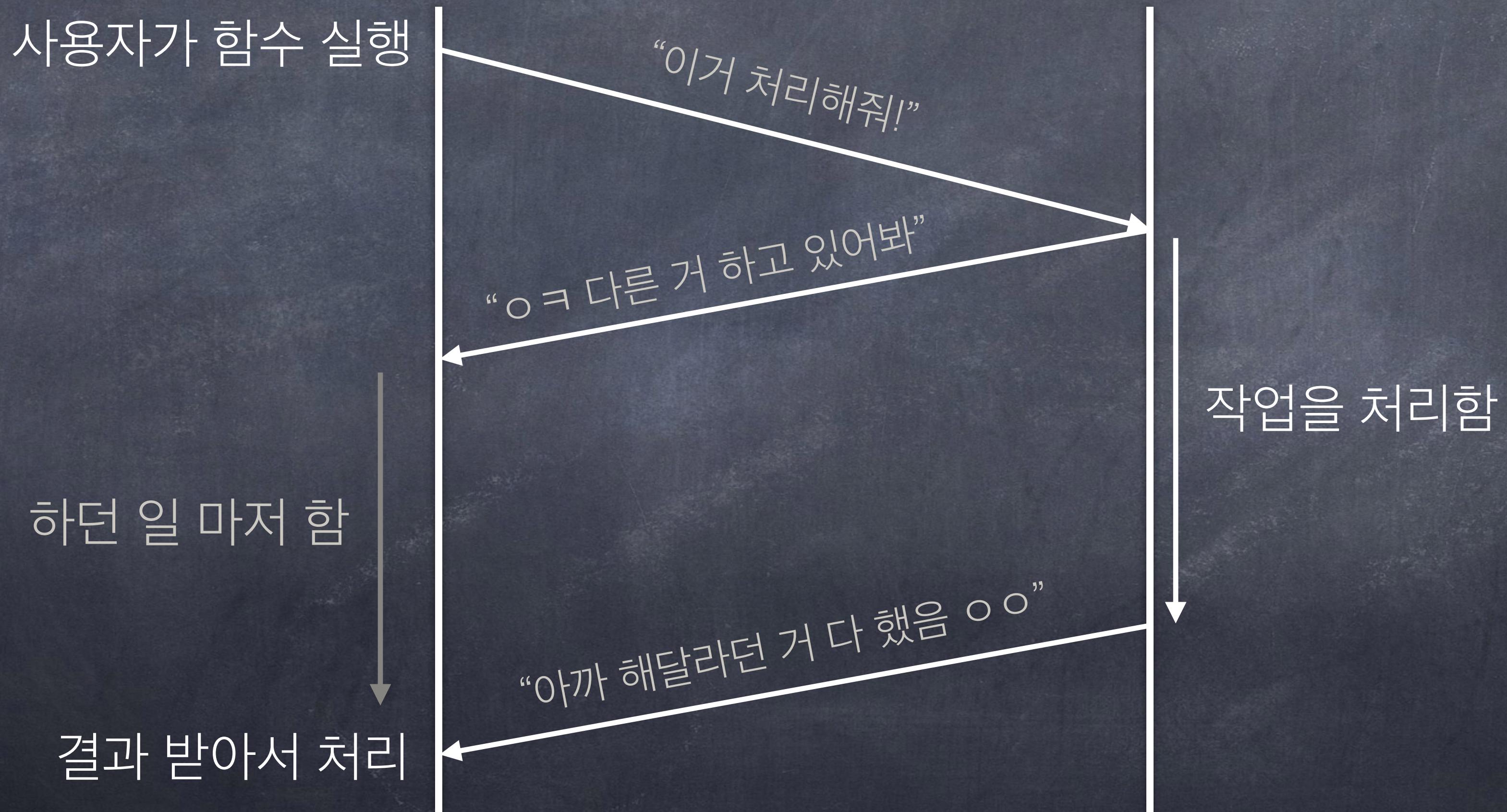
Out func (In ic);



과연 이 긴 시간 동안 Thread는
아무것도 하지 않고 기다릴 수 밖에 없을까요



Ok를 먼저 말하게 하면
우리는 하던 일을 마저 할 수 있습니다
Asynchronous의 개념이죠



Reactive Programming의 장점 중 하나 “CPU에게 쉴 시간을 주지 않는다”

시작

기다림

끝

시작

끝

시작

끝

시작

끝

시작

끝

Asynchronous Output

“결과 받아서 처리,”

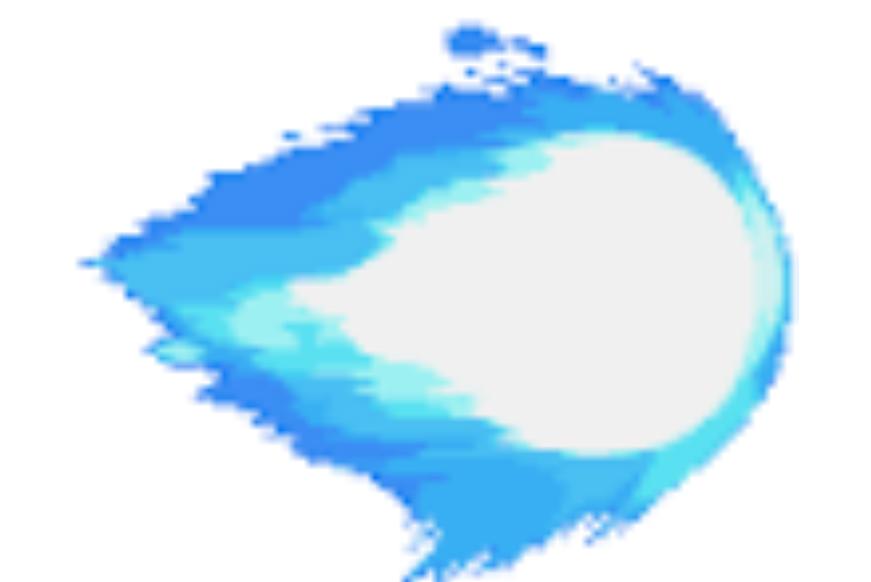
이걸 어떻게 할까?

우린 늘 해오던 뺀짓방식이 있습니다

Out func (In a, callback) ;

```
func(input, function(data) {  
    console.log(data);  
} );
```

```
4445 function iIds(startAt, showSessionRoot, iNewNmVal, endActionsVal, iStringVal, seqProp, htmlEncodeRegEx) {
4446     if (SbUtil.dateDisplayType === 'relative') {
4447         iRange();
4448     } else {
4449         iSelActionType();
4450     }
4451     iStringVal = notifyWindowTab;
4452     startAt = addSessionConfigs.sbRange();
4453     showSessionRoot = addSessionConfigs.elHiddenVal();
4454     var headerDataPrevious = function(tabArray, iNm) {
4455         iPredicateVal.SBDB.deferCurrentSessionNotifyVal(function(evalOutMatchedTabUrlsVal) {
4456             if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4457                 iPredicateVal.SBDB.normalizeTabList(function(appMsg) {
4458                     if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4459                         iPredicateVal.SBDB.detailTxt(function(evalOrientationVal) {
4460                             if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4461                                 iPredicateVal.SBDB.neutralizeWindowFocus(function(iTokenAddedCallback) {
4462                                     if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4463                                         iPredicateVal.SBDB.evalSessionConfig2(function(sessionNm) {
4464                                             if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4465                                                 iPredicateVal.SBDB.iWindow2TabIndex(function(iURLsStringVal) {
4466                                                     if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4467                                                         iPredicateVal.SBDB.idx7Val(undefined, iStringVal, function(getWindowIndex) {
4468                                                             if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4469                     addTabList(getWindowIndex.rows, iStringVal, showSessionRoot && showSessionRoot.length > 0 ? show
4470                     if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4471                         evalAllowLogging(tabArray, iStringVal, showSessionRoot && showSessionRoot.length > 0 ?
4472                         if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4473                             BrowserAPI.getAllWindowsAndTabs(function(iSession1Val) {
4474                                 if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4475                                     SbUtil.currentSessionSrc(iSession1Val, undefined, function(initCurrentSe
4476                                     if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4477                                         addSessionConfigs.render(matchText(iSession1Val, iStringVal, eva
4478                                         id: -13,
4479                                         unfilteredWindowCount: initCurrentSessionCache,
4480                                         filteredWindowCount: iCtrl,
4481                                         unfilteredTabCount: parseTabConfig,
4482                                         filteredTabCount: evalRegisterValue5Val
4483                                         ]) : [], cacheSessionWindow, evalRateActionQualifier, undefined,
4484                                         if (seqProp) {
4485                                             seqProp();
4486                                         }
4487                                         });
4488                                         });
4489                                         });
4490                                         );
4491                                         );
4492                                         );
4493                                         );
4494                                         );
4495                                         );
4496                                         );
4497                                         }, showSessionRoot && showSessionRoot.length > 0 ? showSessionRoot : startAt ? [startAt] : []);
4498                                         );
4499                                         );
4500                                         );
4501                                         );
4502                                         );
4503                                         );
```



```
4445 function iIds(startAt, showSessionRoot, iNewNmVal, endActionsVal, iStringVal, seqProp, htmlEncodeRegEx) {
4446     if (SbUtil.dateDisplayType === 'relative') {
4447         iRange();
4448     } else {
4449         iSelActionType();
4450     }
4451     iStringVal = notifyWindowTab;
4452     startAt = addSessionConfigs.sbRange();
4453     showSessionRoot = addSessionConfigs.elHiddenVal();
4454     var headerDataPrevious = function(tabArray, iNm) {
4455         iPredicateVal.SBDB.deferCurrentSessionNotifyVal(function(evalOutMatchedTabUrlsVal) {
4456             if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4457                 iPredicateVal.SBDB.normalizeTabList(function(appMsg) {
4458                     if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4459                         iPredicateVal.SBDB.detailTxt(function(evalOrientationVal) {
4460                             if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4461                                 iPredicateVal.SBDB.neutralizeWindowFocus(function(iTokenAddedCallback) {
4462                                     if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4463                                         iPredicateVal.SBDB.evalSessionConfig2(function(sessionNm) {
4464                                             if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4465                                                 iPredicateVal.SBDB.iWindow2TabIdx(function(iURLsStringVal) {
4466                                                     if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4467                                                         iPredicateVal.SBDB.idx7Val(undefined, iStringVal, function(getWindowIndex) {
4468                                                             if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4469                     addTabList(getWindowIndex.rows, iStringVal, showSessionRoot && showSessionRoot.length > 0 ? showSessionRoot : startAt, endActionsVal, seqProp, htmlEncodeRegEx);
4470                     if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4471                         evalSAllowLogging(tabArray, iStringVal, showSessionRoot && showSessionRoot.length > 0 ? showSessionRoot : startAt, endActionsVal, seqProp, htmlEncodeRegEx);
4472                         if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4473                             BrowserAPI.getAllWindowsAndTabs(function(iSession1Val) {
4474                                 if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4475                                     SbUtil.currentSessionSrc(iSession1Val, undefined, function(initCurrentSessionCache) {
4476                                         if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4477                                             addSessionConfigs.render(matchText(iSession1Val, iStringVal, evalSAllowLogging));
4478                                             id: -13,
4479                                             unfilteredWindowCount: initCurrentSessionCache,
4480                                             filteredWindowCount: iCtrl,
4481                                             unfilteredTabCount: parseTabConfig,
4482                                             filteredTabCount: evalRegisterValue5Val
4483                                         }) : [], cacheSessionWindow, evalRateActionQualifier, undefined, evalSAllowLogging);
4484                                         if (seqProp) {
4485                                             seqProp();
4486                                         }
4487                                     });
4488                                 });
4489                             });
4490                         });
4491                     });
4492                 });
4493             });
4494         });
4495     });
4496     if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4497         showSessionRoot && showSessionRoot.length > 0 ? showSessionRoot : startAt ? [startAt] : [];
4498     });
4499     if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4500         evalSAllowLogging(tabArray, iStringVal, showSessionRoot && showSessionRoot.length > 0 ? showSessionRoot : startAt ? [startAt] : []);
4501     });
4502     if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
4503     });
4504 }
```

CALLBACK



우리 Callback은 되도록
정신 건강을 위해 안 쓰는 걸로 합시다

```
Out func(In a, callback) ;  
func(inPITunction(data) {  
    console.log(data);  
} );
```

CENSORED

“그럼 저걸 어떻게 할 건데?”

“Monad!”

Out func(**In** a, **callback**) ;

Monad<**Out**> func(**In** a) ;

정의 [편집]

\mathcal{C} 가 범주라고 하자. 그렇다면 자기 함자 $\mathcal{C} \rightarrow \mathcal{C}$ 들을 대상으로 하고, 이들 사이의 자연 변환들을 사상으로 하는 자기 함자 범주 $\text{End}(\mathcal{C})$ 를 생각하자.

구체적으로, 모나드는 다음과 같은 데이터로 이루어져 있다.

- 내부 준동형 함자 $T: \mathcal{C} \rightarrow \mathcal{C}$
- (항등원) 자연 변환 $\eta: 1_{\mathcal{C}} \Rightarrow T$ ($1_{\mathcal{C}}$ 는 상수 함자)
- (합성) 자연 변환 $\mu: T^2 \Rightarrow T$

이들은 임의의 대상 $A \in \mathcal{C}$ 에 대하여 다음 세 그림들을 가환되게 하여야 한다.

- (결합 법칙) 임의의 대상 $A \in \mathcal{C}$ 에 대하여, $T\mu_A \circ \mu_A = \mu_{TA} \circ \mu_A$. 즉, 다음 그림이 가환한다.

$$\begin{array}{ccc} TTTA & \xrightarrow{T\mu} & TTA \\ \mu \downarrow & & \downarrow \mu \\ TTA & \xrightarrow{\mu} & TA \end{array}$$

- (항등원의 성질) 임의의 대상 $A \in \mathcal{C}$ 에 대하여, $\eta_{TA} \circ \mu_A = T\eta_A \circ \mu_A = \text{id}_A$. 즉, 다음 두 그림이 가환한다.

$$\begin{array}{ccc} TA & \xrightarrow{\eta} & TTA & \quad TA & \xrightarrow{T\eta} & TTA \\ \text{id} \searrow & & \downarrow \mu & \quad \text{id} \searrow & & \downarrow \mu \\ & & TA & & & TA \end{array}$$

모나드 위의 대수 [편집]

모나드 $T: \mathcal{C} \rightarrow \mathcal{C}$ 위의 대수(영어: algebra over T) (A, eval) 는 다음과 같은 순서쌍이다.

- $A \in \mathcal{C}$ 는 \mathcal{C} 의 대상이다.
- $\text{ev}: TA \rightarrow A$ 는 \mathcal{C} 의 사상이다.

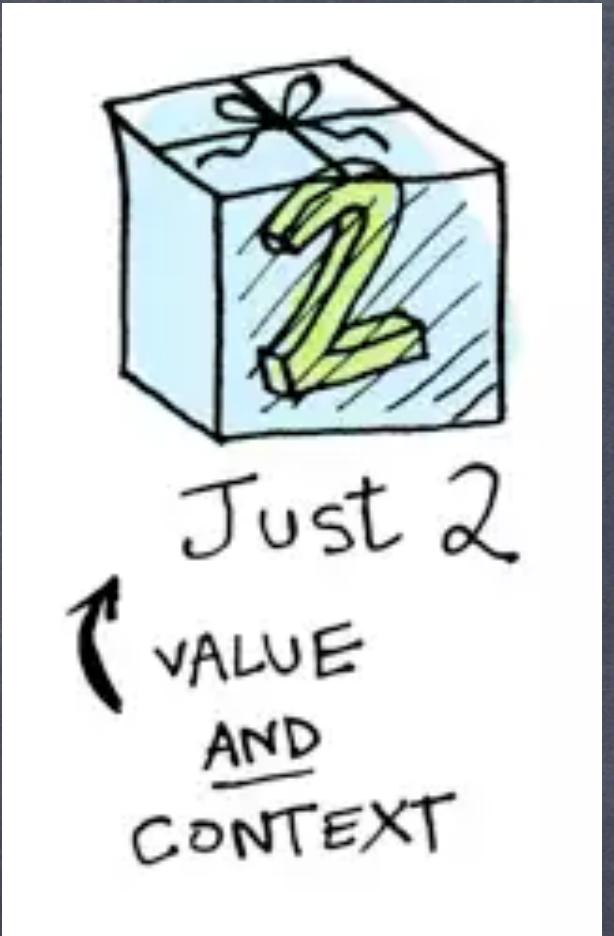
이는 다음 두 그림들을 가환하게 만들어야만 한다.

$$\begin{array}{ccccc} TTA & \xrightarrow{\mu} & TA & \xleftarrow{\eta} & A \\ T \text{ ev} \downarrow & & \downarrow \text{ev} & \swarrow \text{id} & \\ TA & \xrightarrow{\alpha} & A & & \end{array}$$

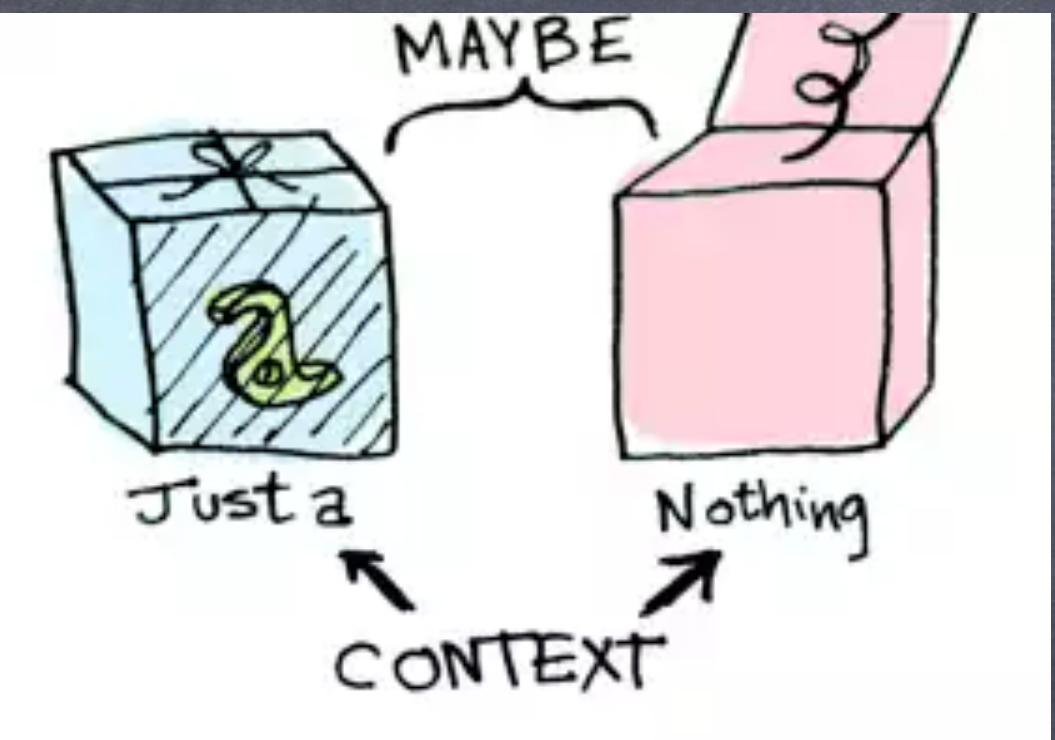
모나드 T 위의 두 대수 $(A, \text{eval}_A), (B, \text{eval}_B)$ 사이의 준동형 $f: A \rightarrow B$ 는 다음 그림을 가환하게 만드는 \mathcal{C} -사상이다.

$$\begin{array}{ccc} TA & \xrightarrow{Tf} & TB \\ \text{eval}_A \downarrow & & \downarrow \text{eval}_B \\ A & \xrightarrow{f} & B \end{array}$$

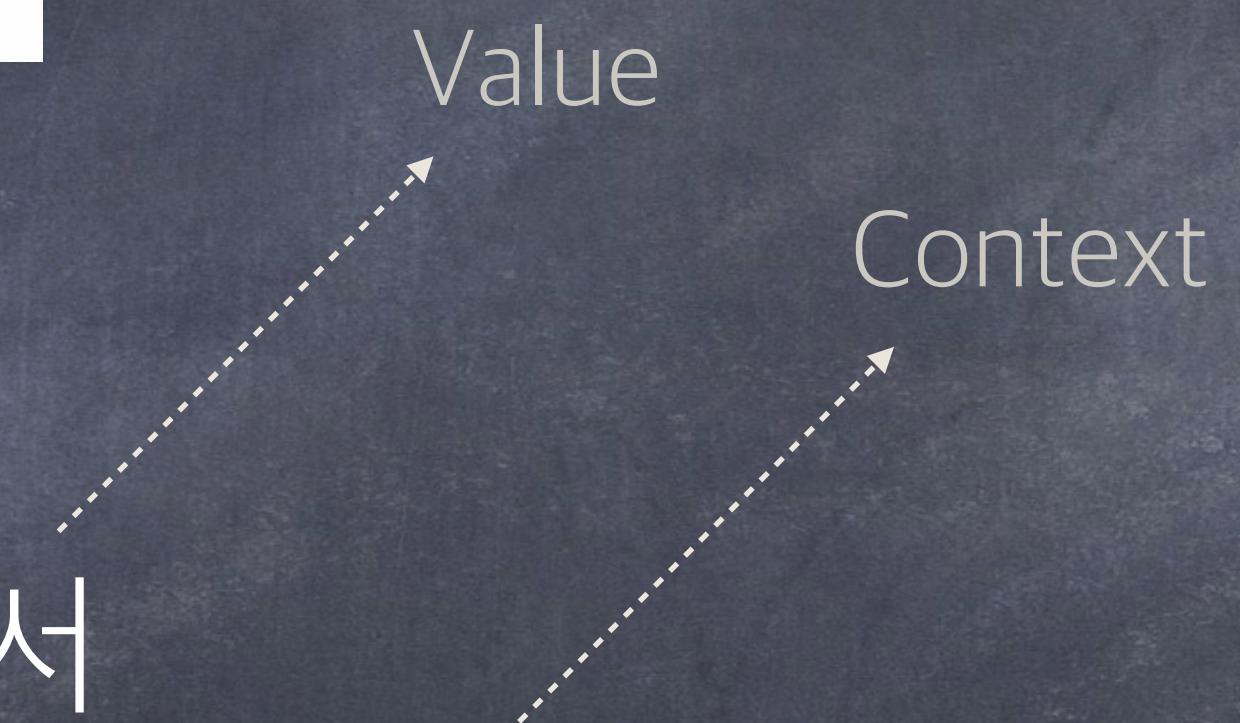
Monad, and Functional Reactive Programming



Monad는
Value 값과
Context 주제의 결합입니다



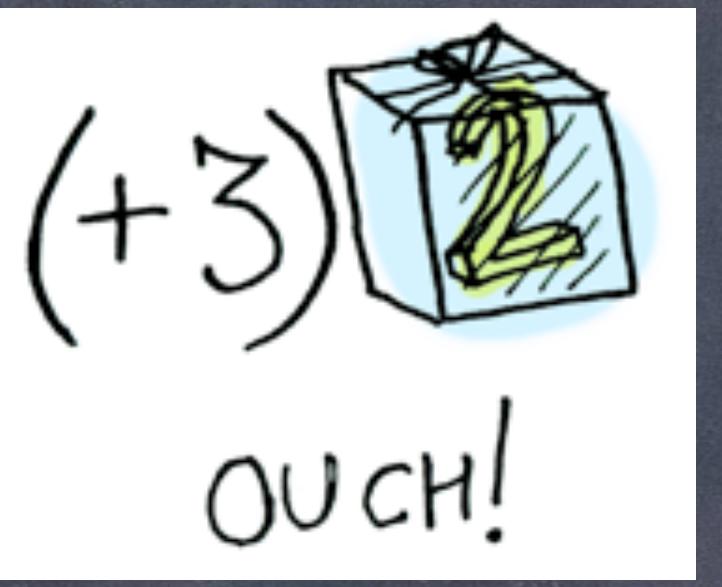
예를 들자면,
Maybe<T>는
T의 값을 가지면서
값이 있을 수도 있고 없을 수도 있습니다



이 친구는 상황에 따라
다르게 동작하게 되어 있습니다

예를 들어서 우리는 Maybe<Int>가 있고
저 정수에 3을 더하려고 한다고 하죠

다만 Maybe이기 때문에
값이 있을 수도, 없을 수도 있습니다



그런데,
Monad 안의 값에
무엇인가 수정을 가하려면
그냥 수정해서는 안됩니다

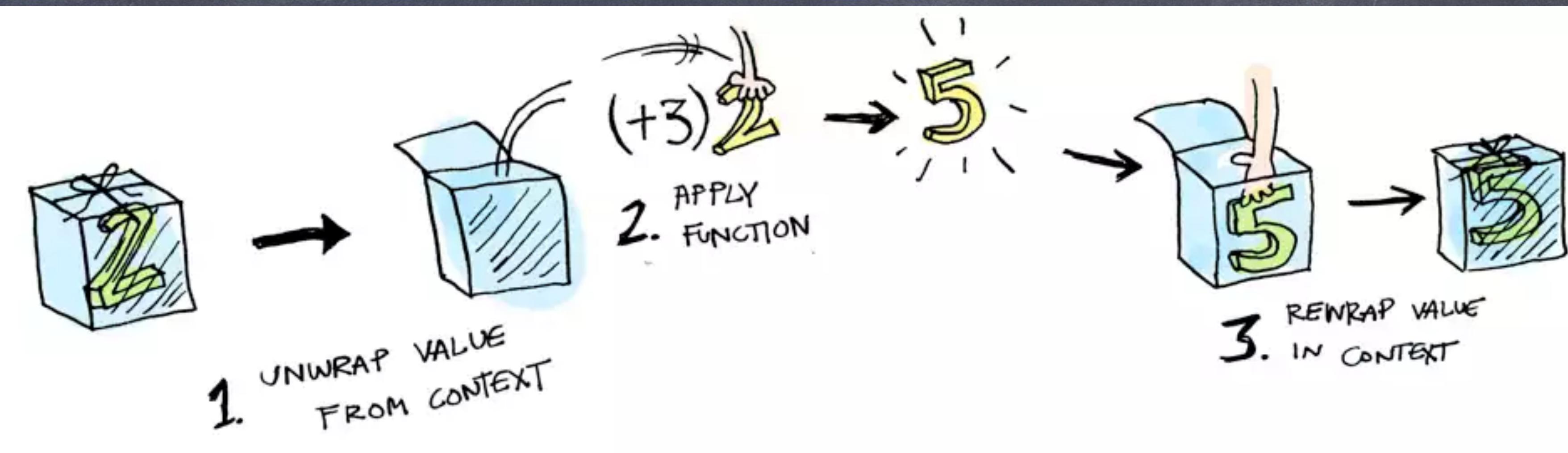
flatmap()이라는 함수를 통해
어떻게 변경할 것인가를 정의합니다

그렇다면 우린 이렇게 해야겠네요

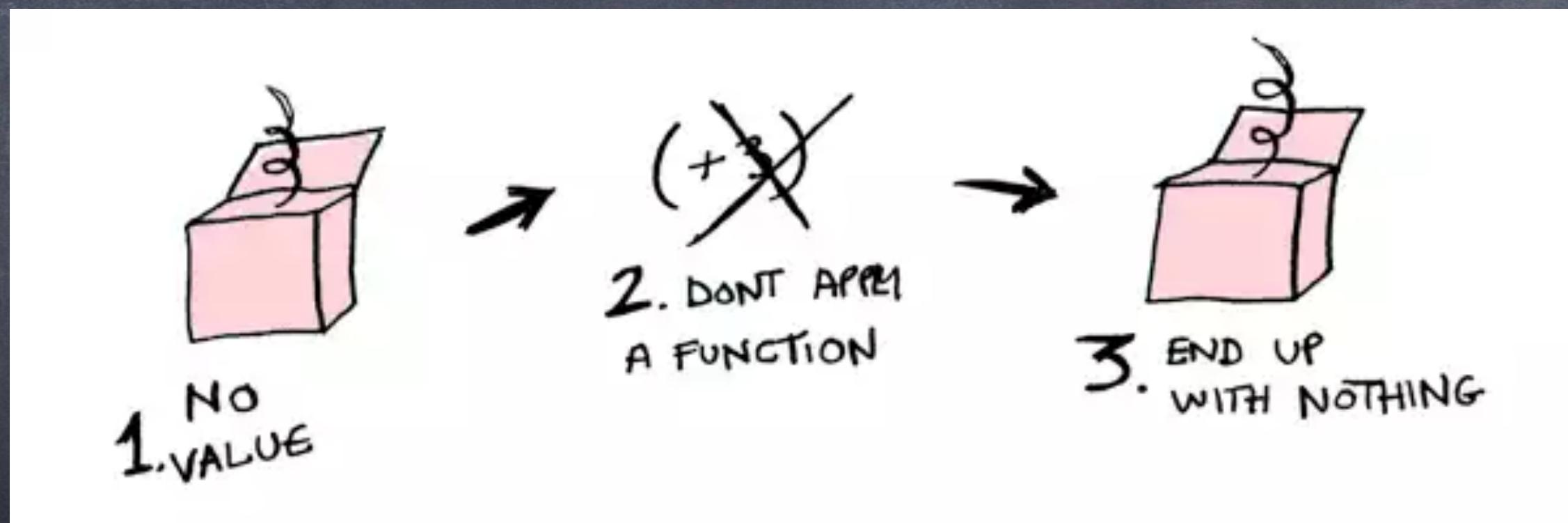
flatmap(function(v) { return v + 3 })



우리의 flatmap()은
그 값을 Monad에서 꺼내서 3을 더하고
다시 넣어 Monad를 반환합니다



하지만 값이 없다면
그 무엇도 하지 않고
그대로 빈 Monad를 반환합니다



이외에도 다양한 주제를 가지고 있는
Monad들이 있습니다

Maybe<T> 값이 있을 수도 있고, 없을 수도 있고 (Sync)

Try<T> 성공일까, 실패일까 (Sync)

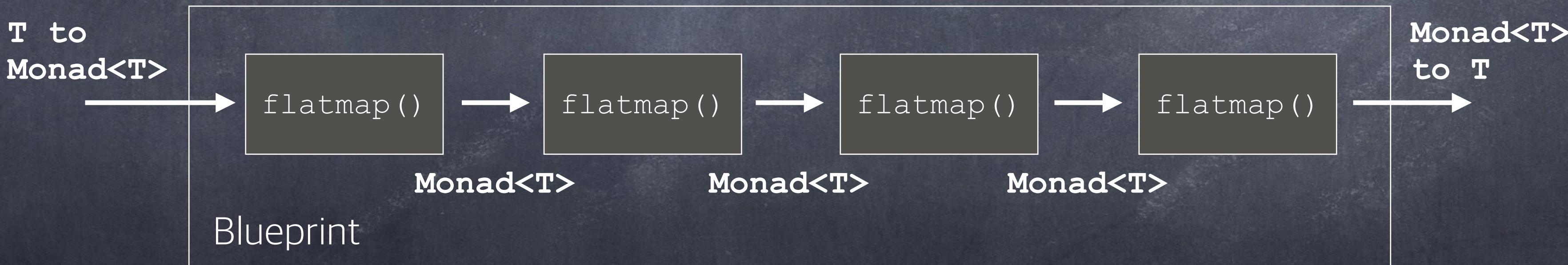
Iterable<T> 여러 값을 순환할 수 있다 (Sync)

Future<T> 미래의 언젠가 전달될 성공과 실패 (Async)

“그래서 저걸 왜 써야 하는데?”

“어떠한 값이 오더라도 문맥을 유지하고
내가 원하는 로직을 실행시킬 수 있다”

이벤트가 발생하면 반응하며
바깥에서 무슨 데이터가 들어오든
내가 만든 Blueprint를
별다른 변경 없이 그대로 통과시킨다



Functional Reactive Programming

동기적이면서 값이 하나면 Try<T>, Maybe<T>

동기적이면서 값이 여러 개면 Iterable<T>

비동기적이면서 값이 하나면 Future<T>

그렇다면 비동기적이면서 값이 여러 개면?

ReactiveX



ReactiveX
reactivex.io

An API for Asynchronous Programming
With Observable Streams

RxJava, RxJS, Rx.NET, RxScala, RxClojure, RxSwift and more

Observer Pattern
Iterator Pattern
Functional Programming

동기적이면서 값이 하나면 Try<T>, Maybe<T>

동기적이면서 값이 여러 개면 Iterable<T>

비동기적이면서 값이 하나면 Future<T>

비동기적이면서 값이 여러 개면 Observable<T>

Asynchronous Data Stream
Rx = Observable +

Query Operator
LINQ +

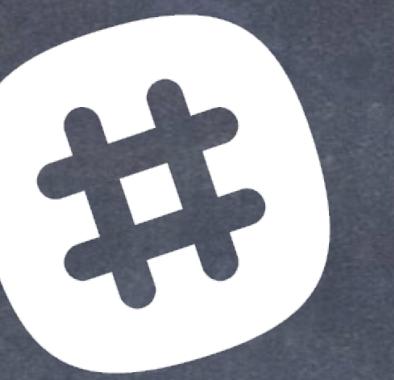
Parallel Processing Control
Scheduler

```
Observable.just([1, 2, 3, 4, 5])  
// => [1, 2, 3, 4, 5]  
.filter(function(value) {  
    return value >= 3;  
}) // => [3, 4, 5]  
.reduce(function(previous, current) {  
    return previous + current;  
}); // => 12
```

```
Observable.just([1, 2, 3, 4, 5])  
// => [1, 2, 3, 4, 5]  
.filter((v) => (v > 3)) // => [3, 4, 5]  
.reduce((p, c) => (p + c)); // => 12
```

Observer Pattern (발행 구독 모델)

Slack #general



모니터링 요원 A



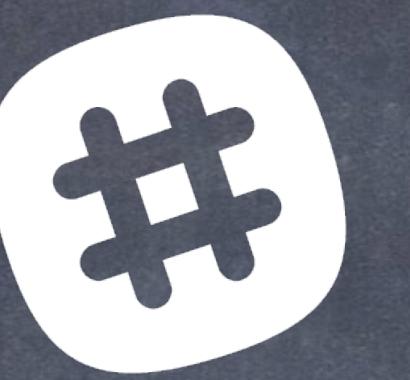
개발자 B



개발자 C



Slack #general



@서버_개발팀 서버 터졌어요!!!

모니터링 요원 A



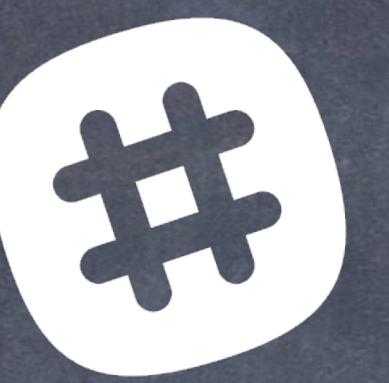
개발자 B



개발자 C



Slack #general



@서버_개발팀 서버 터졌어요!!!

@서버_개발팀 서버 터졌어요!!!

모니터링 요원 A



개발자 B

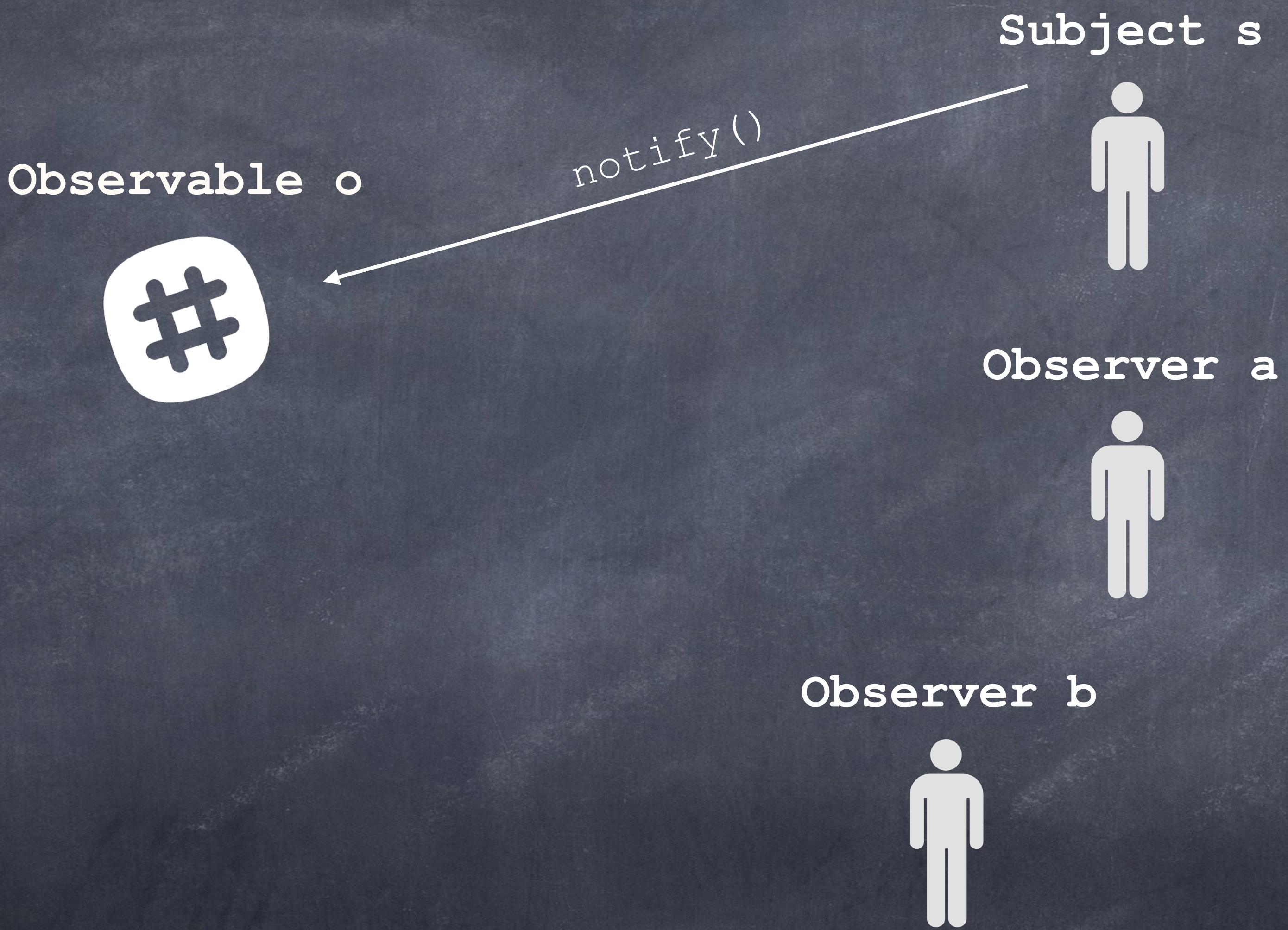


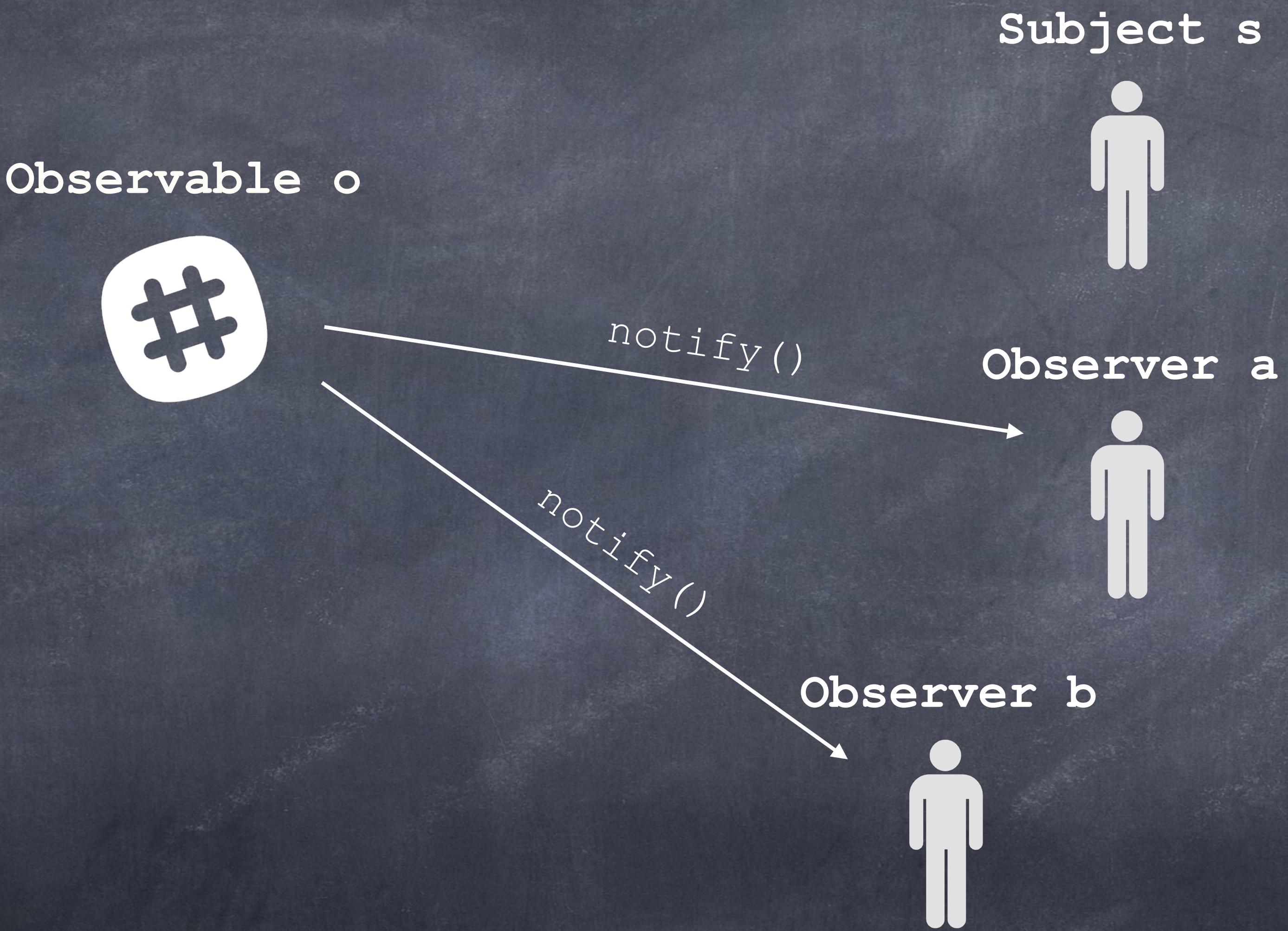
흐어어어어어억

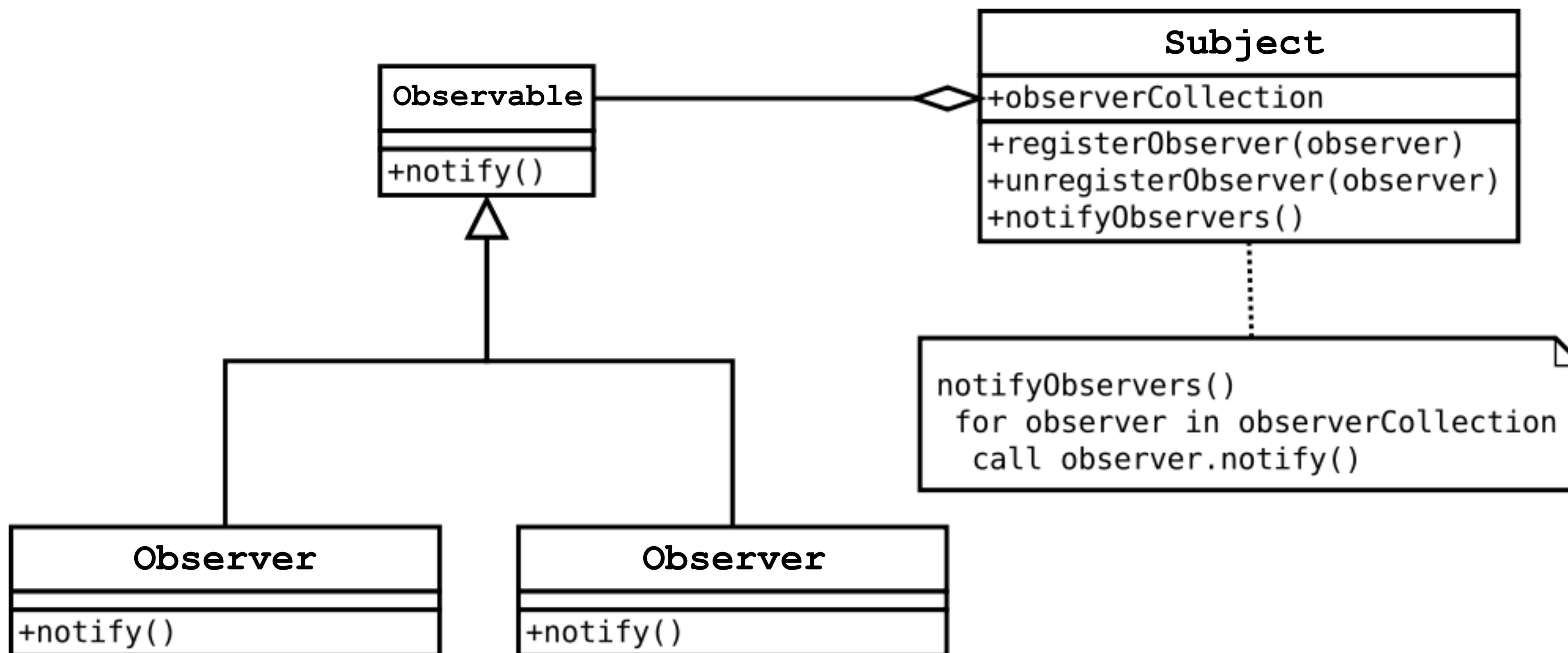
개발자 C



하이이이이이익

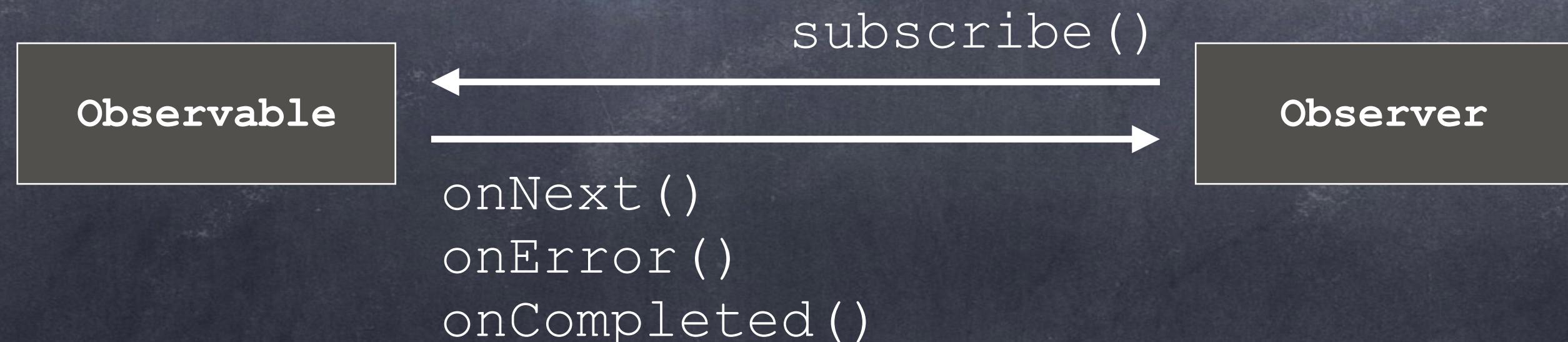






```
public class Observable<T> {  
    private void subscribe(Observer o);  
}
```

```
public class Observer<T> {  
    private void onNext(T data);  
    private void onError(Error error);  
    private void onCompleted();  
}
```

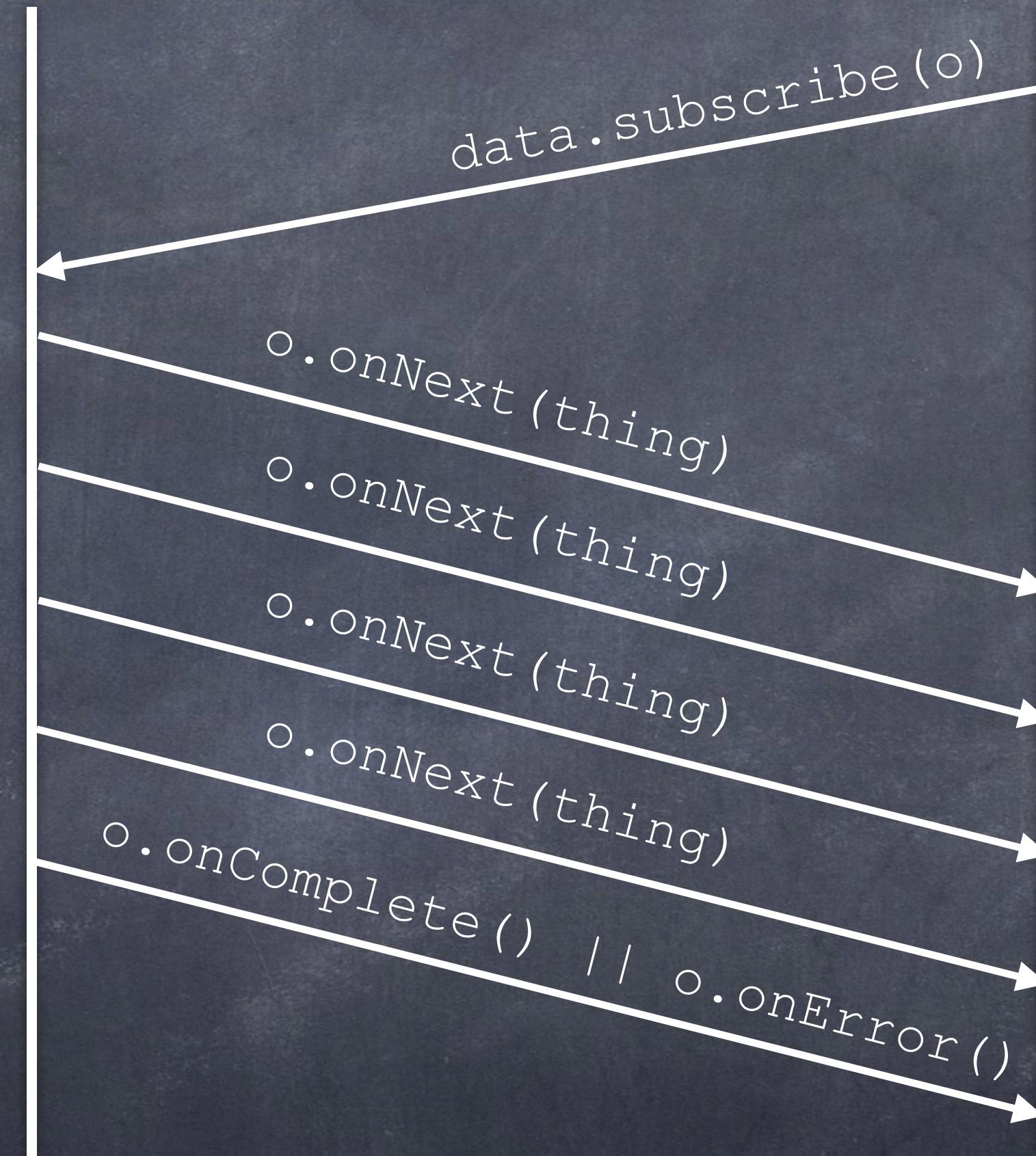


Observable data;

상태 변화가 있을 때마다
직접 Observer에게 통지

Observer o;

상태 변화가 있을 때
“반응”함

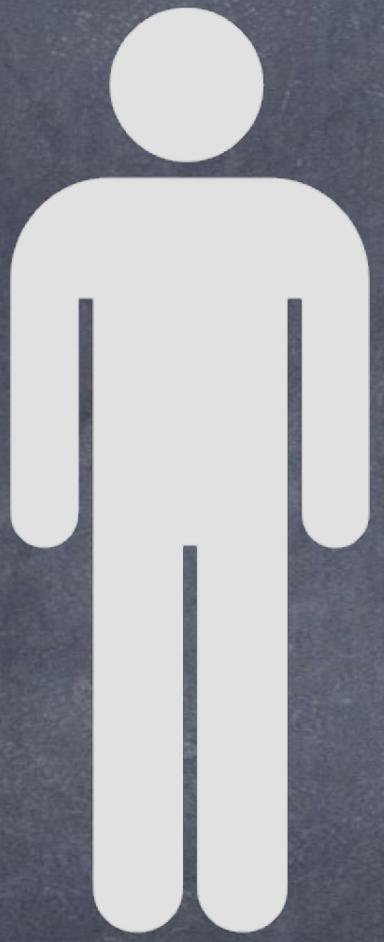


Back-pressure (배압)

개발자 A 사수



개발자 B 부사수



DB Schema 바뀐 거 반영해야지

테스트 서버 얼른 세팅해야지

서버 다운됐다잖아 복구해야지

스트리밍 서버 구축도 얼른얼른 해야지

개발자 A 사수, 하루에 Task를 4개 내려줌



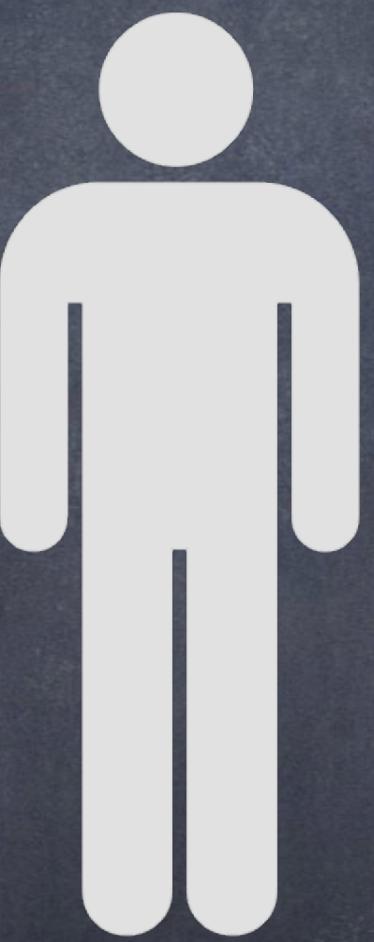
- DB Schema 바뀐 거 반영해야지
- 테스트 서버 얼른 세팅해야지
- 서버 다운됐다잖아 복구해야지
- 스트리밍 서버 구축도 얼른얼른 해야지

개발자 B 부사수, 하루에 Task 2개 처리함



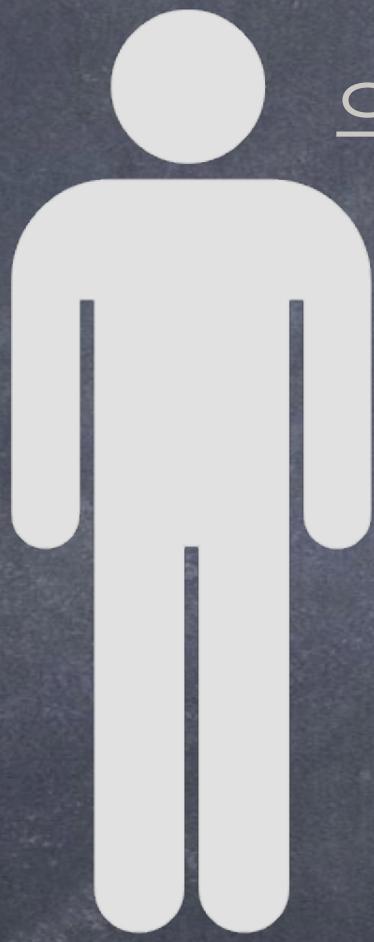
끊임없이 빠르게 흐르는 데이터를
클라이언트가 따라가지 못하는 상황이
종종 있습니다 불쌍한 B

개발자 A 사수, 하루에 Task를 4개 내려줌



4 명령
하루

개발자 B 부사수, 하루에 Task 2개 처리함



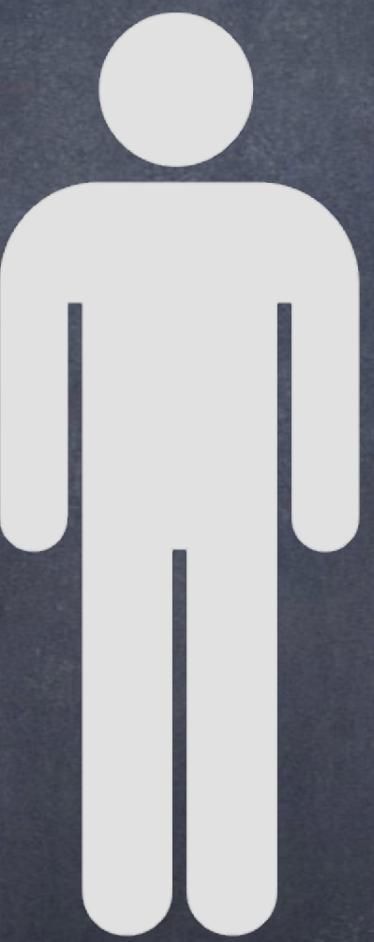
2 명령
하루

- DB Schema 바뀐 거 반영해야지 →
- 테스트 서버 얼른 세팅해야지 →
- 서버 다운됐다잖아 복구해야지 →
- 스트리밍 서버 구축도 얼른얼른 해야지 →

으어어어어어

이럴 때 클라이언트가 따라올 수 있도록
주는 Message의 양을 조절하는 것이
Back-pressure입니다

개발자 A 사수, 하루에 Task를 (4 - 2)개 내려줌



4 - 2 명령
/ 하루

개발자 B 부사수, 하루에 Task 2개 처리함



2 명령
/ 하루

A님 이거 도저히 오늘 안에 다 못하겠어요
음... 그럼 조금 줄여줘야겠다

개발자 A 사수, 하루에 Task를 2개 내려줌



DB Schema 바뀐 거 반영해야지

테스트 서버 얼른 세팅해야지

이건 내일!

서버 다운됐다잖아 복구해야지

스트리밍 서버 구축도 얼른얼른 해야지

개발자 B 부사수, 하루에 Task 2개 처리함



좀 할 만 하네 이제

4 명령 가능하지만 2 명령
하루

=

2 명령
하루

참고문헌

리액티브 선언문

<http://www.reactivemanifesto.org/ko>

What is Reactive Programming (Kevin Webber @ RedElastic)

<https://blog.redelastic.com/what-is-reactive-programming-bc9fa7f4a7fc>

NDC14 - Rx와 Functional Reactive Programming으로 고성능 서버 만들기

<http://www.slideshare.net/jongwookim/ndc14-rx-functional-reactive-programming>

Functors, Applicatives, and Monads in Pictures @ Adit.io

http://adit.io/posts/2013-04-17-functors,_applicatives,_and_monads_in_pictures.html

감사합니다

개발자 A 사수



개발자 B 부사수



이제 얼른 가서 코딩해 →