좋은 코드는 어떻게 만들 수 있을까?

Simple made easy를 읽고...

김은민

왜 이런 주제를 뽑았나?

- 사람들이 Clojure 언어에 관심을 갖게 하고 싶었다
- 좋은 코드 예제는 많다. 좋은 코드의 본질이 무엇인지 정리하고 싶었다.
- Simple made easy를 정리해보자

좋은 코드는 무엇인가?

- 읽기 쉬운 코드
- 테스트 할 수 있는 코드
- 짧은 코드
- 버그가 없고 정확한 코드
- 빠른 코드
- 메모리를 덜 쓰는 코드
- 확장성이 좋은 코드
- 유지 보수가 가능한 코드

좋은 코드는 무엇인가?

- 읽기 쉬운 코드
- 테스트 할 수 있는 코드
- 짧은 코드
- 버그가 없고 정확한 코드
- 빠른 코드
- 메모리를 덜 쓰는 코드
- 확장성이 좋은 코드
- 유지 보수가 가능한 코드

sourcemaking.com/antipatterns/spaghetti-code

읽기 쉬운 코드란?

```
var result = 0;
for (i = 1; i <= 10; i++) {
    result = result + i;
}
console.log("Result: " + result);
```

- 이해하기 어려우신 분?

읽기 쉬운 코드란?

```
var result = 0;
for (i = 1; i <= 10; i++) {
    result = result + i;
}
console.log("Result: " + result);
```

- var는 변수를 말하는데 변수는 값에 이름을 붙인 것이다. 하지만 그 값은 바뀔 수 있어서 어떤 때는 다른 값일 수 있다. 그리고 변수를 만드는 방법은 var 뒤에 공백을 하나 이상 띄우고 변수명을 써주고 그 다음에는 = 을 붙인 뒤에 값을 써주고 마지막에 ;을 해서 하나의 명령문이 끝난다는 표시를 해준다.

- for는 반복해서 어떤 명령문들을 ...

왜 읽기 쉬운 건가?

- 개발자는 코드를 읽는 법을 알고 있기 때문에 쉽다.
- 모르는 사람은 어렵다.

쉽고 어려운 것은 단순히 알고 모르는 것의 차이인가?

- 코딩을 처음 배우는 사람에게 모든 것을 알려주고 코드를 읽어보라고 한다면?

아는 것만으로는 쉽다고 느끼기 힘들다

```
var result = 0;
for (i = 1; i <= 10; i++) {
    result = result + i;
}
console.log("Result: " + result);
```

- 개발을 처음 배우는 친구에게 변수는 어려웠다
- 순서를 따라가면서 바뀌는 상태 값을 머리 속에 기억하고 있어야 했는데 그게 잘 안된다고 했다

읽기 쉬워야 좋은 코드인데 그럼 어떻게 쉽게 만들지?

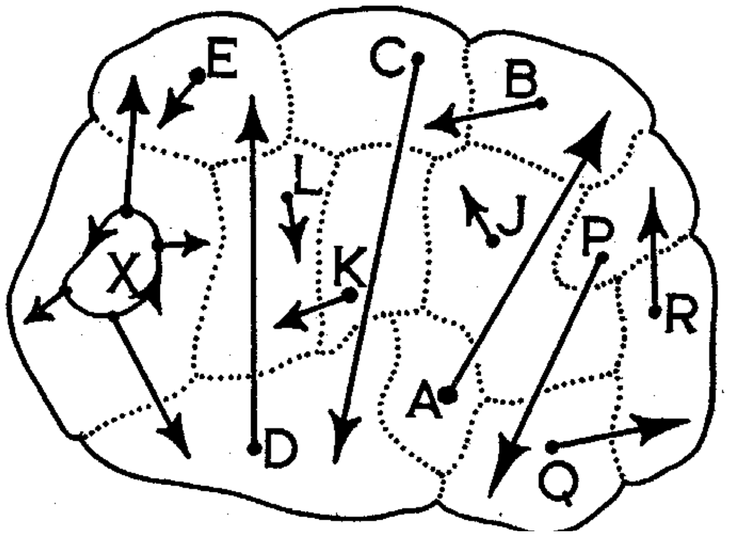
무엇이 어려운 것인가?

```
var result = 0;
for (i = 1; i <= 10; i++) {
    result = result + i;
}
console.log("Result: " + result);
```

- 예제 코드는 "A는 B다"와 같은 지식이 아니다.
- 복잡한 개념들이 서로 엮여 있다. 순서, 값, 시간

어떻게 복잡한 것이 쉽게 되었는가?

- 개발자는 연습을 통해 머리가 컴퓨터 처럼 되었다 – W. Ross Ashby의 필수 다양성의 법칙
- 그래서 익숙해졌다



djmarsay.files.wordpress.com/2011/01/ashby-dfb-f14_3_11.gif

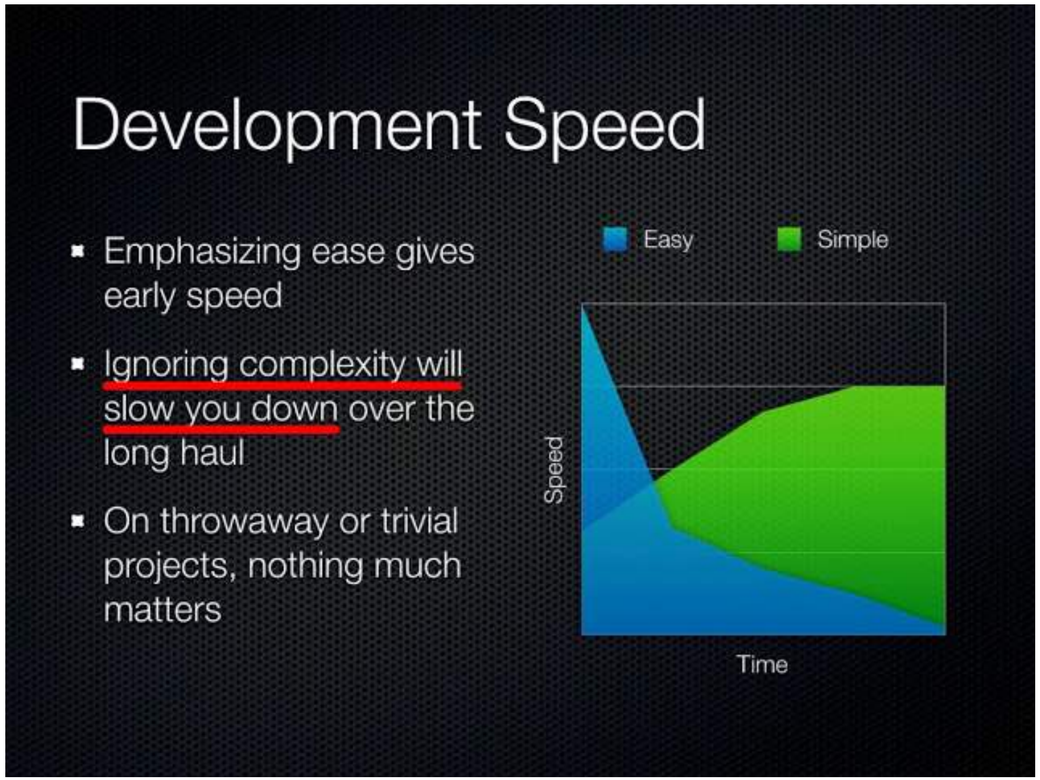
하지만 연습을 통한 쉬움은 상대적이고 주관적이다

- 공 5개로 저글링 할 수 있는 사람은 공 3개로 저글링 하는 것이 쉽다
- 공 3개로 저글링 할 수 있는 사람은 공 5개로 저글링 하는 것이 쉽다
- 공 5개로 저글링 하는 것은 쉬운가 어려운가?
- 누가 코드를 이해하는 연습을 더 했는가?

ko.wikipedia.org/wiki/저글링

심지어 연습을 통한 극복은 끝이 없다

- 소프트웨어는 계속 기능이 추가되면서 복잡해진다.
- 복잡해지면 어떤 사람에게는 처음에 쉬웠지만 나중에는 감당할 수 없다.



https://www.infoq.com/presentations/Simple-Made-Easy/

그래서 처음부터 복잡하지 않으면 쉽다

- 만약 복잡하면 복잡한 것을 없애 단순하게 만들어야 한다

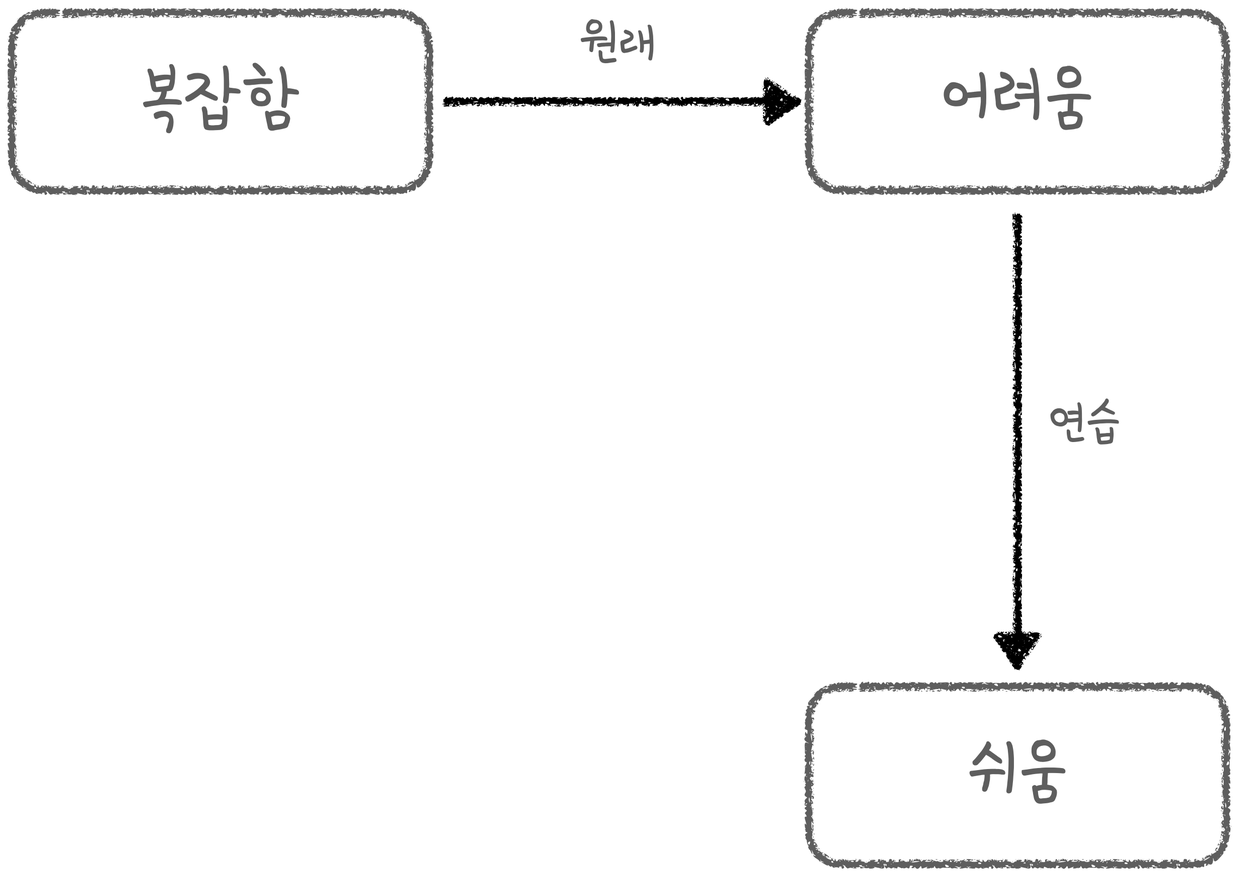
www.bankinghub.eu/banking/research-markets/complexity-kills-european-banking-models-change-complex-world

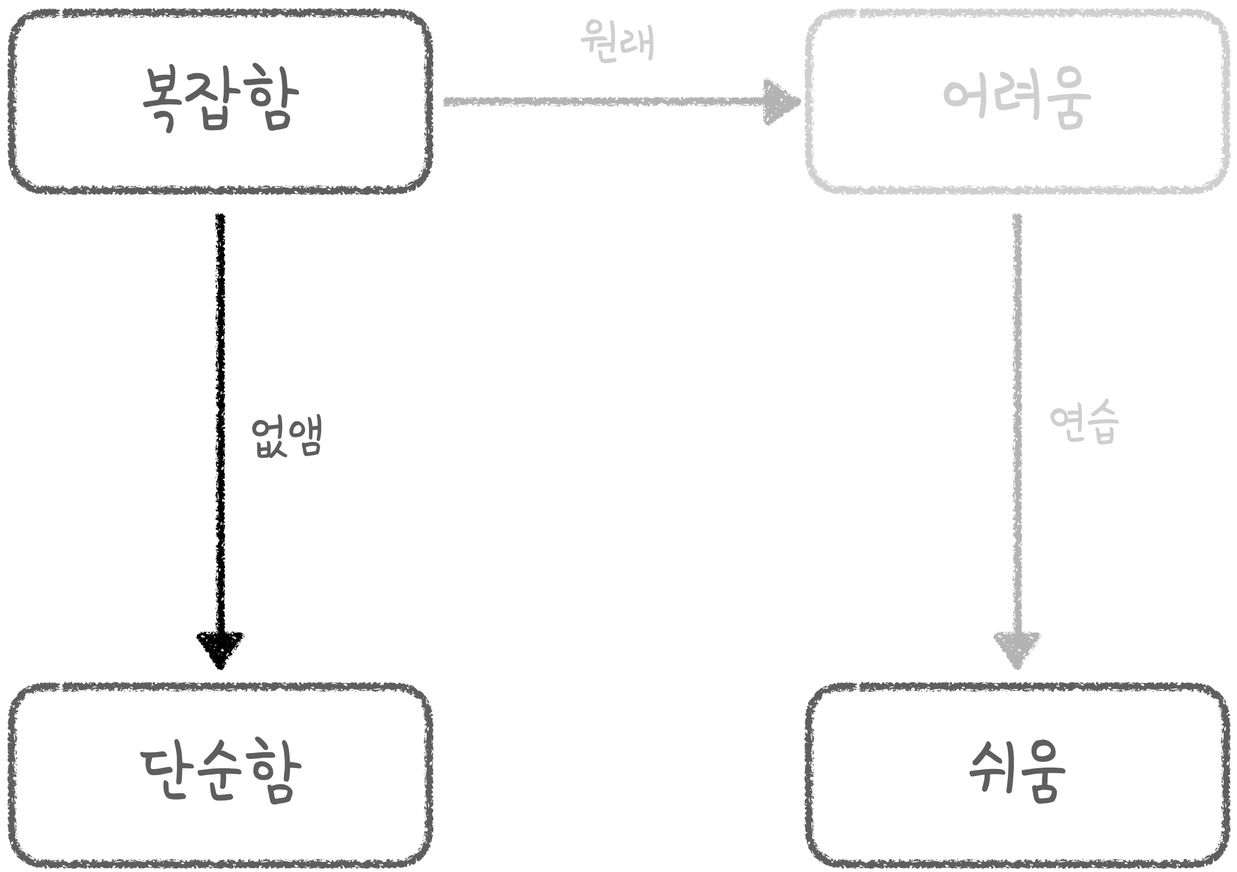
복잡하지 않을 수 있나?

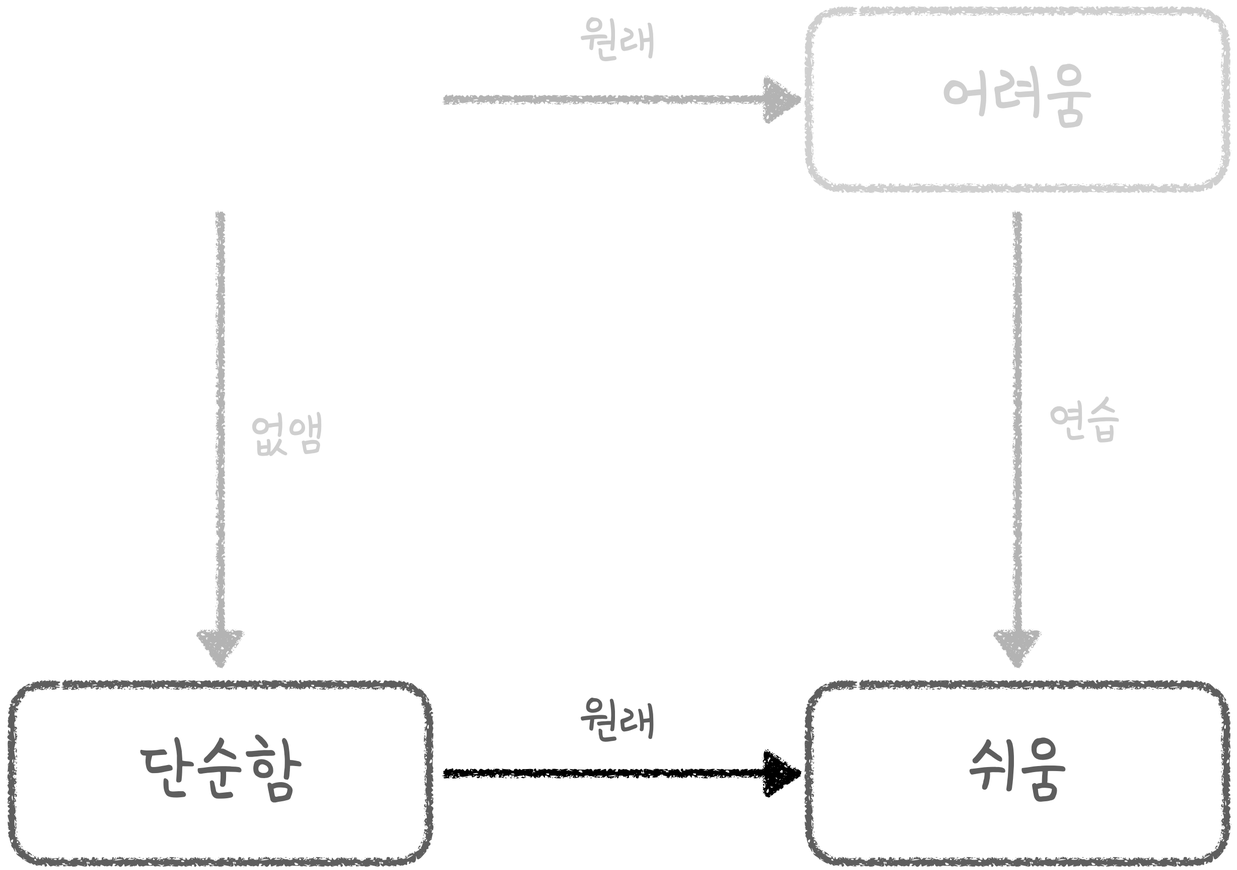
- 맨먼스 미신이라는 책을 쓴 프레드 브룩스가 쓴 No Silver Bullet - Essence and Accident in Software Engineering 이란 논문
  - 본질적 (Essence) 복잡성
  - 부차적 (Accident) 복잡성 (* 맨먼스 미신 번역을 따름)
- 부차적 복잡성은 제거할 수 있다

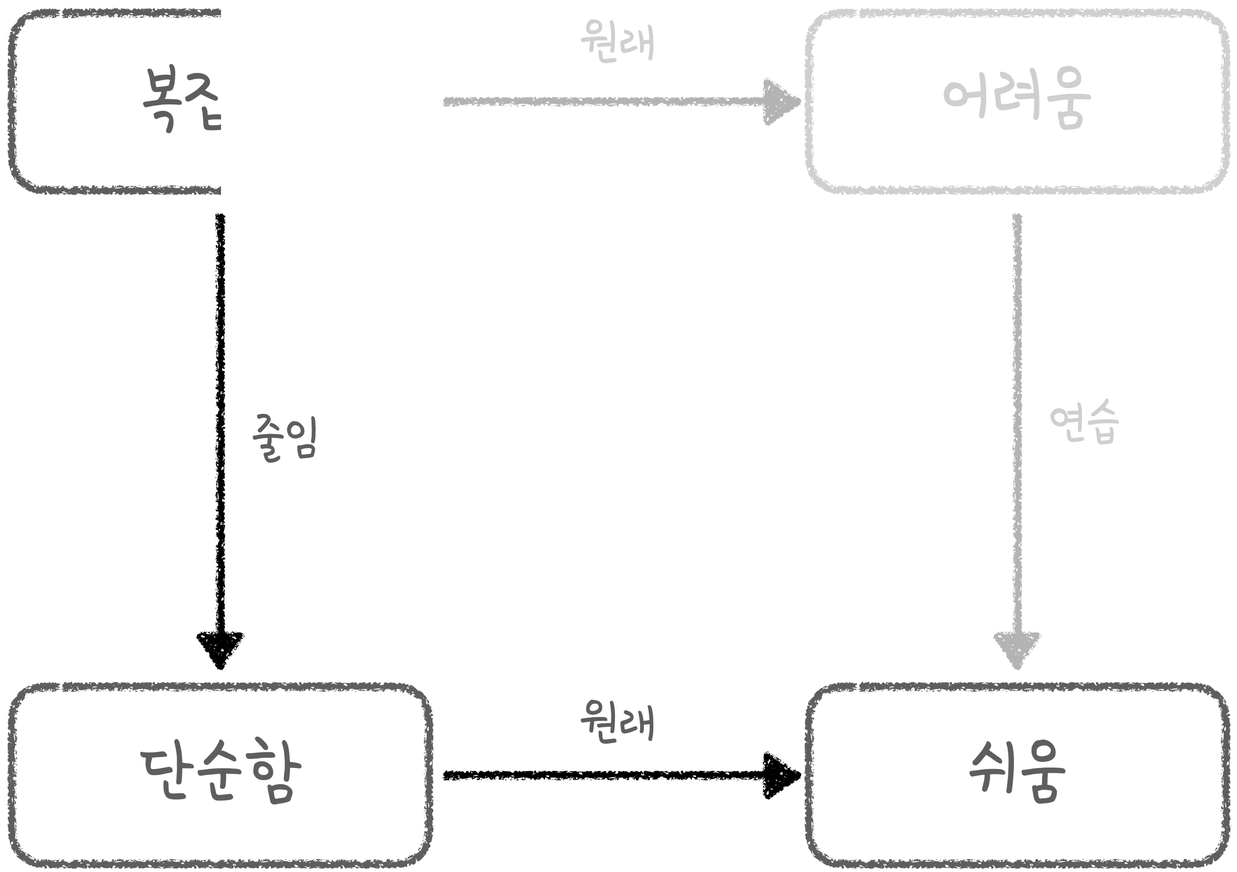
en.wikipedia.org/wiki/Fred_Brooks

더 가기 전에 지금까지 한 내용을 그림으로 정리


복잡함
원래
어려움
연습
쉬움


복잡함
원래
어려움
없앰
연습
단순함
쉬움


원래
어려움
없앰
연습
단순함
원래
쉬움


복잡
원래
어려움
줄임
연습
단순함
원래
쉬움

복잡한 것을 없애려면 복잡하다는 말이 무엇인지 알아야한다

- Complex
  - 어원 com-plex 는 "함께"와 "엮이다"를 합친 말이다
- Simple
  - 어원 sim-ple 은 "하나"와 "엮이다"를 합친 말이다.
- 결국 복잡하다는 것은 하나 이상의 개념이 엮여 서로 땔 수 없는 것을 말한다.
- 단순하다는 것은 하나의 개념을 말한다.
  - 한가지 일을 해라

코드에서 복잡한 것

- 상태 – 상태는 상태를 쓰는 모든 것이 함께 엮여 있다
- 객체 – 상태와 값이 엮여 있다
- 메소드 – 함수와 상태가 엮어 있고 어떤 언어는 네임스페이스까지 엮여 있다
- 문법 – 의미와 순서가 엮여 있다
- 변수 – 시간과 값이 엮여 있다
- 절차적 반복문 – 무엇과 어떻게가 엮여 있다
- 상속 – 타입과 타입이 엮여 있다
- 배열 – 항목과 항목의 순서가 엮여 있다
- Call Chain – 호출과 호출이 엮여 있다
- 절차적 프로그래밍 – 순서(절차)가 엮여 있다

복잡한 코드를 단순하게 만들 수 있는 방법

- 상태 – 값으로 바꾼다
- 객체 – 값으로 바꾼다
- 메소드 – 함수와 네임스페이스로 바꾼다
- 문법 – 데이터로 바꾼다
- 변수 – 가능하면 사용하지 않는다
- 절차적 반복문 – 컬렉션 함수로 바꾼다
- 상속 – 다형성으로 바꾼다
- 배열 – 연관 배열로 바꾼다
- Call Chain – 큐를 이용한다
- 절차적 프로그래밍 – 선언적 프로그램으로 바꾼다

복잡성을 제거하려면

- 다른 것과 엮이지 않게 해야한다
- 다른 곳으로 옮기기 쉽다
  - 유연 / 확장 가능

언어가 단순하면 어떨까?

- 상태 – 없고 값만 있음
- 객체 – 없고 값만 있음
- 메소드 – 없고 함수만 있음
- 문법 – 없음?
- 절차적 반복문 – 없음
- 상속 – 없고 다형성만 있음

I wanted to make ‘doing the right thing’ not a matter of convention and discipline, but the default.

Rich Hickey

그것은 바로  Clojure ㅋㅋㅋㅋㅋㅋㅋㅋㅋ

- 상태 – 없고 값만 있음
- 객체 – 없고 값만 있음
- 메소드 – 없고 함수만 있음
- 문법 – 없음?
- 절차적 반복문 – 없음
- 상속 – 없고 다형성만 있음

그런데 문법이 없다고?

- Lisp 언어는 문법이 없다
- Lisp은 언어에서 쓰는 자료 형식으로 코드를 표현한다 (Homoiconicity)
  - (+ 1 (+ 2 3))

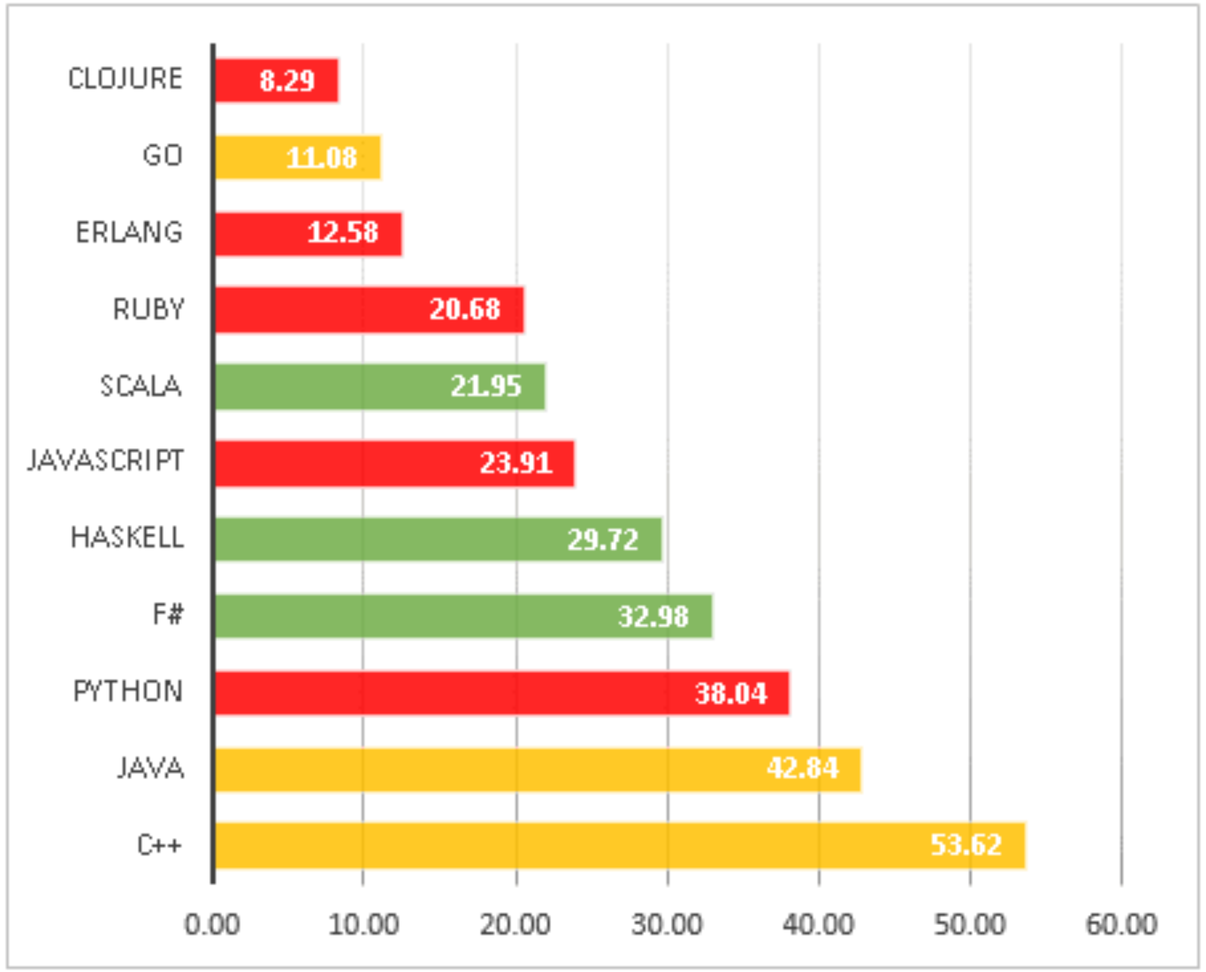
클로저를 배우려면

- 초보자를 위한 클로저 - 김은민
  - eunmin.gitbooks.io/clojure-for-beginners/content/
- 클로저 시작하기 - 캐린 마이어 저 / 박상규, 김만명, 김영태 역 /. 인사이트
  - www.yes24.com/Product/Goods/25537884
- 클로저 프로그래밍의 즐거움 - 마이클 포거스, 크리스 하우저 공저 / 김선호 역 / 비제이퍼블릭
  - ww.yes24.com/Product/Goods/24555451?scode=032&OzSrank=6
- Facebook Clojure korea
- clojure-korea.slack.com
- 김은민에게 연락해주세요!

단순한 코드는 쉽고 좋은 코드를 만든다

- 읽기 쉬운 코드
- 테스트 할 수 있는 코드
- 짧은 코드
- 버그가 없고 정확한 코드
- 빠른 코드
- 메모리를 덜 쓰는 코드
- 확장성이 좋은 코드
- 유지 보수가 가능한 코드

정적 타입 언어는 버그를 줄여주는가?



itnext.io/why-simple-is-so-complex-362bc835b763

Fools ignore complexity. Pragmatists suffer it. Some can avoid it. Geniuses remove it.

Alen Perlis

결론

- 쉽다는 말과 단순하다는 말은 구분해서 사용하세요
- 어떤 것과 엮이지 않는 단순한 코드를 만드세요
- 단순한 코드가 쉽고 좋은 코드 입니다
- 쉬운 코드가 항상 좋은 코드라고 착각하지 마세요
- Clojure를 배워보세요

감사합니다

결론

- 쉽다는 말과 단순하다는 말은 구분해서 사용하세요
- 어떤 것과 엮이지 않는 단순한 코드를 만드세요
- 단순한 코드가 쉽고 좋은 코드 입니다
- 쉬운 코드가 항상 좋은 코드라고 착각하지 마세요
- Clojure를 배워보세요