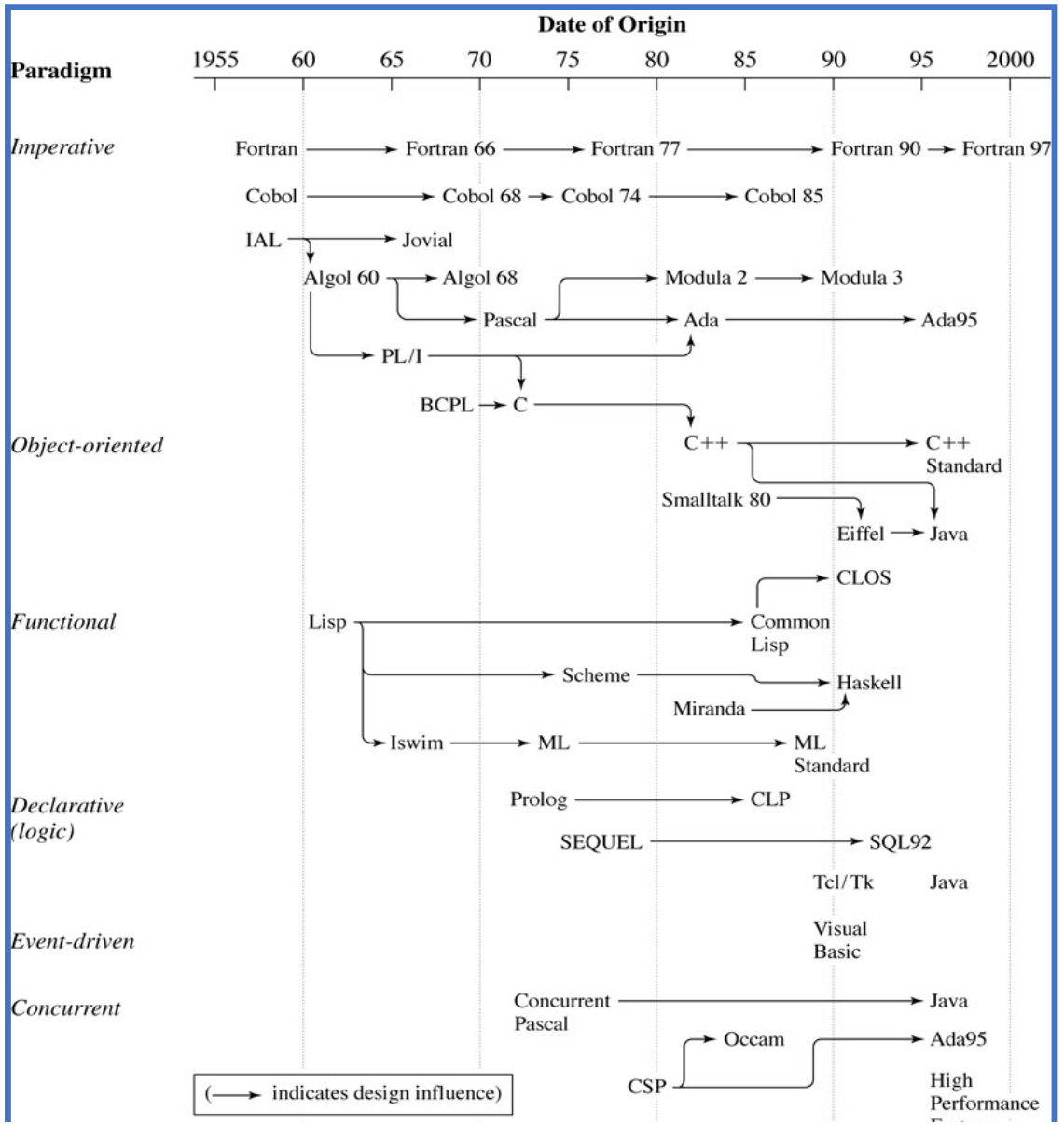
내용

- 프로그래밍언어 역사
- 주요 프로그래밍 언어 (패러다임)
  - 명령형 (객체지향)
  - 함수형
  - 논리형
- 프로그래밍언어 사용 동향
- Haskell (함수형)이 미치는 영향

프로그래밍 언어의 역사

(2000년까지)



주요 명령형 프로그래밍 언어

- Fortran (1958년, IBM의 John Backus)
  - 최초의 고급(high-level) 언어
  - 수치 계산 (Formula Translation)
  - 프로그래밍언어의 핵심 요소를 모두 갖춤
    - 산술식, 문장(할당문, 조건문, 반복문, 입출력), 함수 선언 및 호출 등

- Algol (1960, 유럽 공동체)
  - 알고리즘을 표현하기 위한 언어 (Fortran의 수식 계산 전용보다 진보됨)
  - 특정 하드웨어를 기반으로 설계되지 않음
  - 언어 문법을 Context-Free Grammar로 표현 (BNF, Backus Naur Form)
  - Block Structure (즉, 복합문) 및 변수의 영역(scope) 도입
  - 재귀함수
  - Call-by-name, Call-by-value 두 가지 파라메터 전송 방식 도입
  - Stack 기반의 연산법 제시
  - 현대 명령형 언어의 이론적 모델을 제시함

- C 언어 (1970년대, 미국 Bell Lab)
  - Unix 커널을 코딩하기 위한 목적
  - Pointer를 이용한 주소 접근 기능

함수형 프로그래밍

1. MIT 계열

- LISP (1960년, MIT의 John Macathy)
  - List Processing (기존 배열보다 추상화됨)
  - CONS, CAR, CDAR 등의 오퍼레이터
  - 함수형 프로그래밍
  - 인공지능 목적 (수치 계산 보다는 논리적 추론)
- Scheme - Emacs Lisp - Clojure

2. 유럽 계열 (람다 계산법 및 타입이론의 구현)

- ISIM – ML – SML – Ocaml (eager, call-by-value)
- Miranda – Haskell (lazy, call-by-name)

논리 (Logic) 언어

- Prolog (1972, 프랑스 Colmerauer, Roussel)
  - 일차 논리 (First-order Logic) 를 구현
  - 프로그래밍 예:

Database content:

```
friends(dick, ed).
sameperson(meg, wife(ed)).

has(dick, book).
has(dick, mercedes).

likes(dick, mercedes).
likes(dick, wine).
likes(ed, wine).
likes(wife(ed), wine).
```

Question answering:

```
?- friends(dick, ed).
yes

?- likes(X, wine).
X = dick ;
X = ed ;
X = wife(ed)
yes
```

객체지향 언어

- Simula (1960년대, Nygaard and Dahl, Norway)
  - Simulation 프로그래밍 어어
  - Simula67에서 Objects, Classes, Inheritance 등의 개념을 구현

- Smalltalk (1970~80년, Alan Kay, Xerox 팔로 알토 연구소)
  - Human-Computer Symbiosis (HCI)
  - 그래픽스 모니터 기반에서 마우스를 이용한 제어 (HCI)
  - Xerox Star workstation에 적용됨 (최초의 현대적 PC)
    - GUI, Icons, Mouse(Two Buttons), Ethernet, Email, Folders,
    - Apple의 Lisa, MacIntosh를 거쳐 MS의 Windows PC에 영향을 줌

- C++ (1979, Bjarne Stroustrup)
  - C + OOP
  - C 언어와 호환성 유지, 비교적 안정된 성능

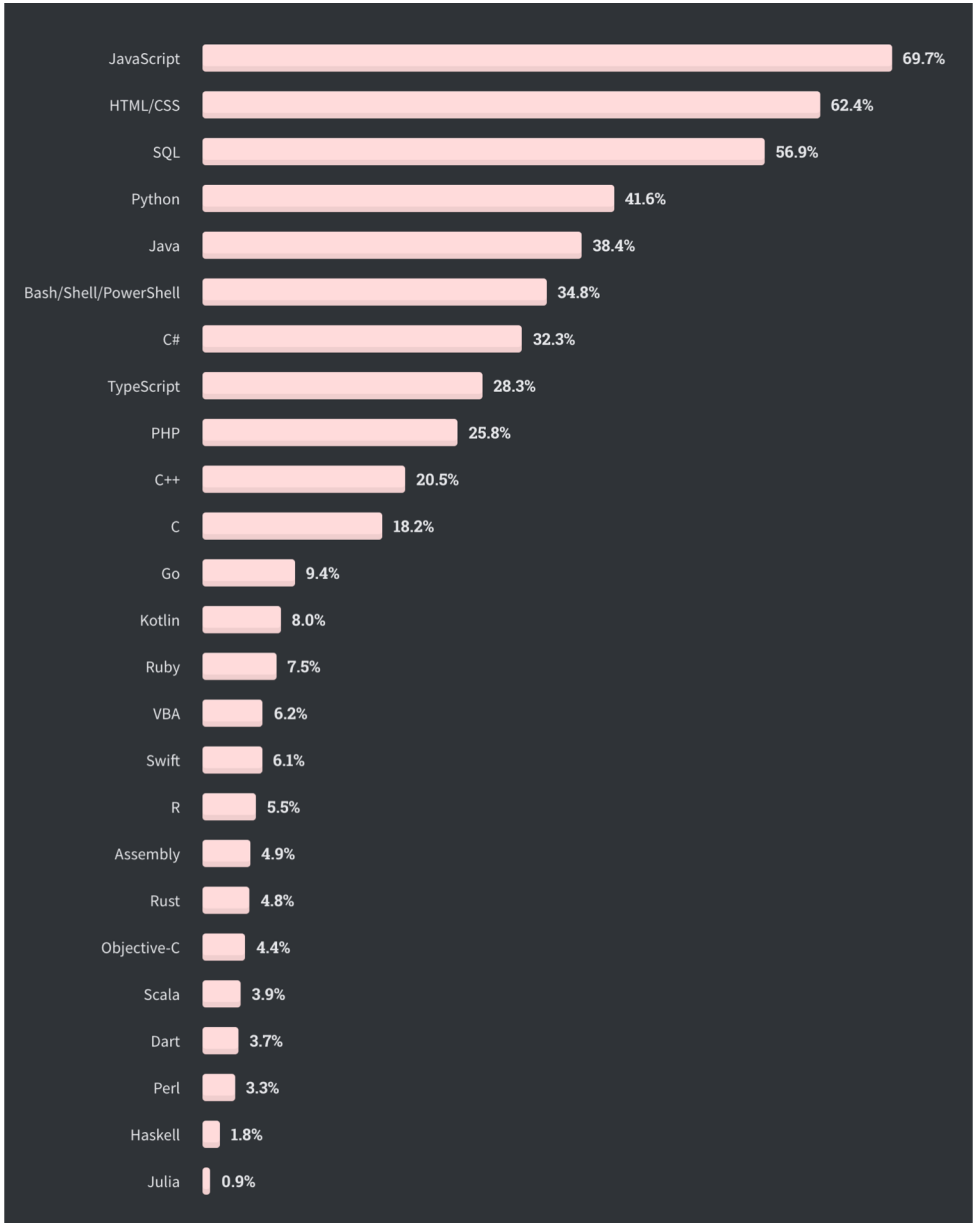
JVM 언어

JVM과 ByteCode 수준에서 호환성을 가짐 (기존 Legacy Code 문제 해결)

- 대표적 언어
  - Java, Closure, JRuby, Kotlin, Scala 등
- Java 언어의 발전
  - JDK 1.0 1996
  - J2SE 5.0 2004 Generics 타입
  - Java SE 8 (LTS) 2014 함수형 프로그래밍 기능 포함
  - Java SE 14 2020.9월 현재 버전
- Scala (2001, 2006, 2011년, Martin Odersky)
  - 객체지향(Java) 스타일과 함수형 프로그래밍의 혼합
  - JVM 기반
- Kotlin (2011년, 러시아)
  - Java 기반
  - 2017년부터 Google의 Android Studio 3.0에서 지원
  - 타입 등에서 함수형 프로그래밍 기능 지원

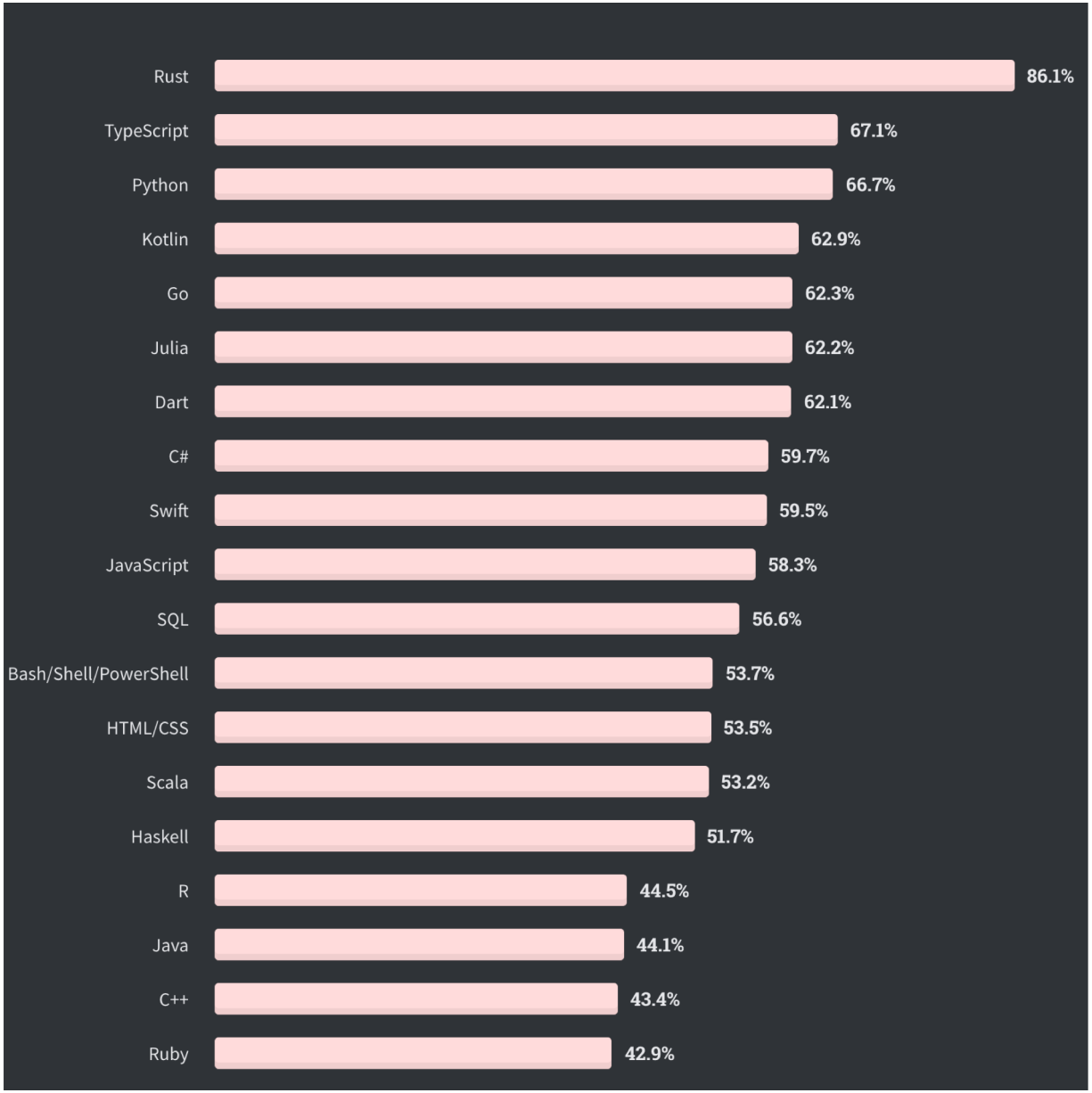
최신동향
- Stackoverflow 2020
- 사용 언어

https://insights.stackoverflow.com/survey/2020#technology-most-loved-dreaded-and-wanted-languages-loved



최신동향
- Stackoverflow 2020
- loved 언어

https://insights.stackoverflow.com/survey/2020#technology-most-loved-dreaded-and-wanted-languages-loved



동향 분석 (응용 분야의 영향)

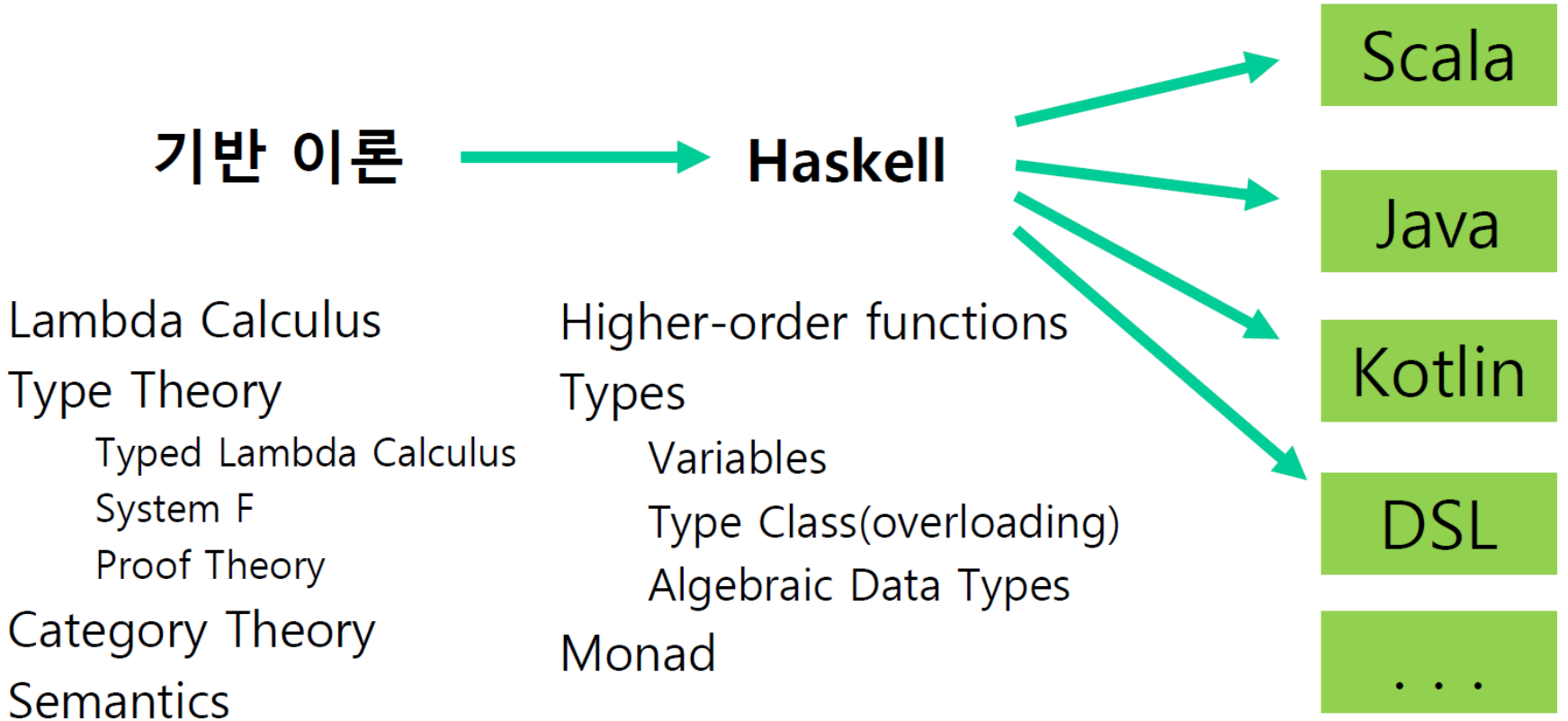
- 웹 프로그래밍의 강세
  - JavaScript, HTML, TypeScript, Ruby
- 머신 러닝의 강세
  - Python(Tensorflow), Julia, R
- 함수형 프로그래밍 기능 포함
  - Rust, Kotlin, Swift, JavaScript, Scala, Haskell
- 타입 기능 강화
  - TypeScript : JavaScript와 함께 사용
  - Rust: C++를 대체할 가능성
  - 에러 발생 가능성을 타입 checking으로 제거

- 병렬성(동시성)
  - Go
- 일반 범용 코딩
  - Java, C#, SQL,
  - C 언어의 사용이 저하됨
- Smartphone App
  - Kotlin
  - Swift

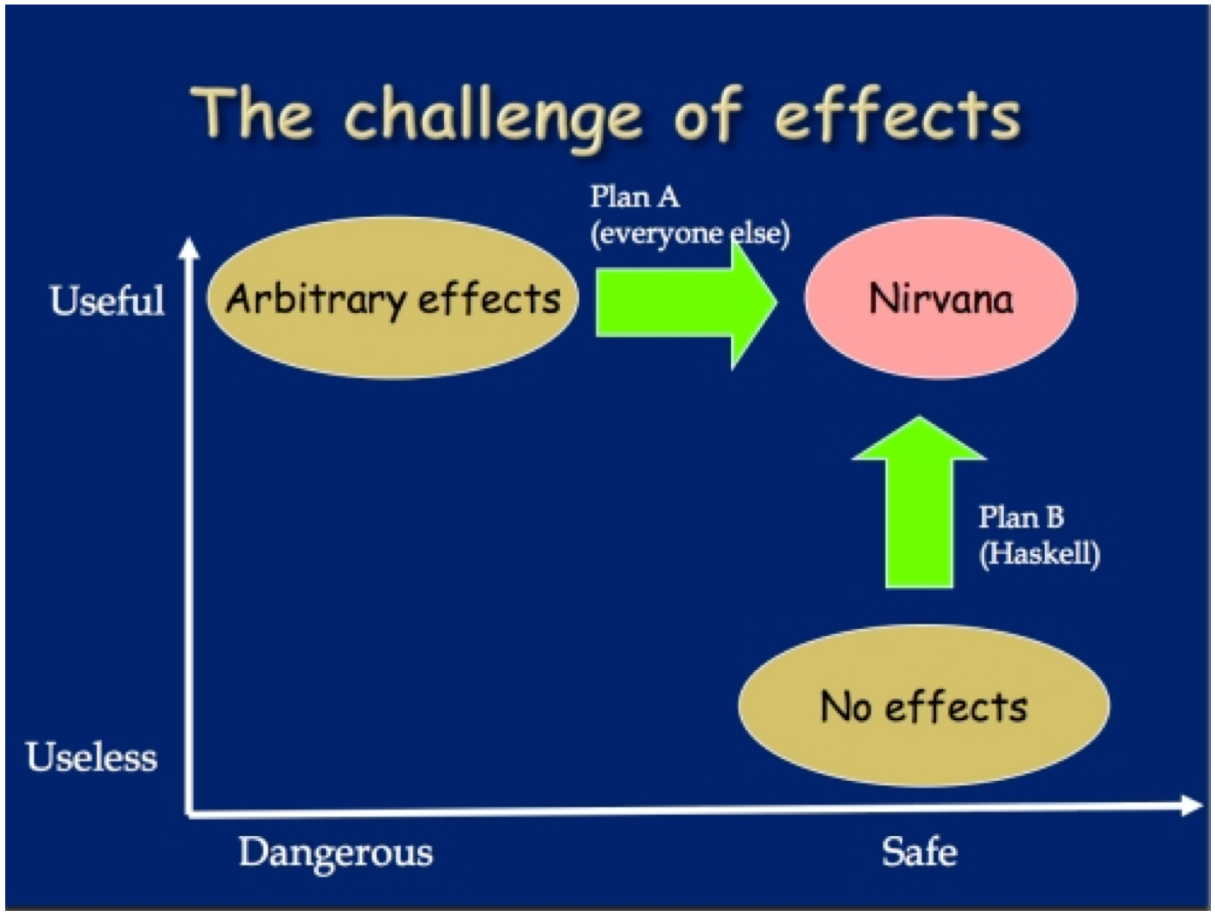
발전 방향

- 편리성 (Python 기준)
  - 코드의 간결성
  - IDE 등의 REPL (Read, Evaluation, Print, Loop) 기능, 대화형(interactive)
  - Python의 Idle, 자바의 jshell, Haskell의 Ghci 등
- 안전성
  - Static Type에 의한 안전성 “컴파일러가 체크해서 통과된 코드는 안전성이 높다“
  - 어떤 타입(?)
  - Haskell, Ocaml, Coq, Agda, Scala, Java 등
- Virtual Machine에 의한 호환성
  - JVM, .Net, 등
- 강력한 응용 분야 지원
  - 웹 시스템 (JavaScript, TypeScript, HTML, React…)
  - 인공지능 (Python, R)
  - 안드로이드 App (Kotlin)
- 함수형 프로그래밍 계열(Haskell, F#, Clojure, Scala, Erlang, Kotlin, Java8 …)
- 가독성, 간결성

Haskell (함수형 프로그래밍)이 미치는 영향



Peyton Jones가 생각하는 궁극의 PL (2008년)
- 순수성(no effect)과 비순수성(effect)의 조화 : 두 가지 접근법



- Plan A
  - 기존 언어의 effect 사용을 제한
  - 전역 변수 및 할당문 사용 제한
  - 순수 함수 기능 추가 (고차 함수, 타입, 람다 ….)
  - Scala, Java 8, Kotlin …
- Plan B
  - 순수하기만 한 언어의 비현실성
  - 순수성을 유지하면서, effect의 기능을 추가함
  - Haskell의 Monad