**1. 함수형 프로그래밍, 절차지향 프로그래밍, 객체지향프로그래밍이란 무엇인가**

**절차적 프로그래밍**  
절차적 프로그래밍이란 단순히 순차적인 명령 수행이 아니라 루틴, 서브루틴, 메소드, 함수 등(이를 통틀어 프로시저라고 한다.)을 이용한 프로그래밍 패러다임을 뜻한다. 명령형 프로그래밍의 일종이다.

**객체지향 프로그래밍** - 일급 객체는 클래스  
 - 해당 클래스의 데이터 기반으로 메소드가 동작(멤버 함수 라고도 함)  
 - 상호 작용하는 객체들의 집합

**함수형 프로그래밍** - 일급 객체는 함수  
 - 전달 받은 파라미터 기반으로 함수가 동작(전달 받은 파라미터는 변경을 하지 않고 계산한 결과만 리턴 함)  
 - 상태 값을 지니지 않는 함수 값들의 연속

**\* 일급 객체** - 다른 객체들에 적용 가능한 연산을 모두 지원하는 객체  
 - 변수나 데이터 구조(자료 구조)에 담을 수 있음  
 - 파라미터, 리턴 값으로 사용 가능  
 - 할당에 사용한 이름에 관계 없이 고유한 식별이 가능함

**출처:** <https://gthoya.tistory.com/entry/%EA%B0%9D%EC%B2%B4-%EC%A7%80%ED%96%A5-%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D-vs-%ED%95%A8%EC%88%98%ED%98%95-%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D>

**2. 함수형 프로그래밍이 갖는 장점과 단점은 무엇인가.**

장점

**Multi-Core & Concurrency**Multi-Core CPU를 장착한 컴퓨터가 대세가 되면서 Multi-Threading을 지원하는 소프트웨어가 필요하게 되었다. 무어의 법칙을 따라 발전하던 CPU가 현실적인 한계에 봉착하자, 수직적 전략을 포기하고 여러 개의 칩이 병렬적으로 동작하도록 만드는 수평적 전략을 선택함. Concurrency를 지원하게 되면서 변경 가능한 데이터(Mutable Data)는 개발자에게 어려움을 가져다 주었다.  
함수형 언어의 특징은 데이터가 변경 불가능(Immutable Data)이다. 때문에 Concurrency를 지원하기에 적합하다.

**코드의 간결함**객체지향(OOP)를 오랜 시간 사용해온 프로그래머라면 각종 디자인 패턴에 친숙할 것이지만, 함수형 프로그래밍의 관점에서는 디자인 패턴은 우격다짐이다. 필요한 논리 부분과 추상 부분을 구별해내고 패턴에 맞게 구성하는 것은 부수적이고 장식적인 코드가 많아지게 한다. 함수형 프로그래밍의 중요 개념인 Curry, Partial Application, Monad와 같은 기법이 간결하고 우아한 함수의 구성(Composition)을 가능하게 해준다.

**C#과 Java 같은 기존 언어의 관리모드**기존 언어들이 발전모드에서 관리모드로 접어듦에 따라 프로그래밍 언어와 관련한 새로운 패러다임이 요구되는 현실과 관련이 있다. JVM이나 .NET 플랫폼을 대상으로 여러 언어들이 실험되고 있으며, 실제로 많은 지원이 이루어지고 있다. 때문에 함수형 패러다임은 객체지향 패러다임의 뒤를 이을 새로운 패러다임을 발견하기 위한 전반적인 실험과 암중모색의 와중에 하나의 강력한 대안으로 주목받고 있다.

단점

**Multi-Core & Concurrency**Immutable Data Structure는 Thread-Safe를 쉽게 보장하는 대신 ‘오래된’ 데이터를 사용하게 될 수도 있다. Mutable Data Structure는 항상 최신 데이터를 다룬다는 장점이 있지만, Data Consistency를 보장하기 위해 복잡함을 수반한다. 둘 중 어느 것이 낫다고 볼 수 없다.

**Non-State, No-Side Effect**함수형 프로그래밍은 상태(State)를 배제하여 Side Effect가 없게 동작한다. 하지만 사용자(User)와의 상호작용(Interaction)은 대부분 상태 변화로 모델링된다.

**코드의 간결함으로 인한 학습필요**코드가 간결해질 수 있지만, 간결한 코드를 읽기 위해서 학습이 필요하다.

**생산성이 올라야만 한다.**  
함수형 프로그래밍 스타일로 작성할 수 있는 프로그래머를 채용하는데 드는 비용을 상쇄시킬만큼 생산성이 올라야만 한다.

**해결할 문제들**세상에는 함수형 프로그래밍 스타일에 적합한 문제도 많지만, 적합하지 않은 문제들도 충분히 많다.

**3. 순수함수란 무엇인가.**

순수 함수란 함수형 프로그래밍에 필요한 개념으로 아래 조건을 만족하는 함수를 뜻한다.

**동일한 입력에는 항상 같은 값을 반환해야 한다.**  
**함수의 실행은 프로그램의 실행에 영향을 미치지 않아야 한다. (Side effect 가 없어야 한다)**

예를 들어, 함수 내부에서 인자의 값을 변경하거나 프로그램 상태를 변경하는 것

순수 함수를 호출하면 프로그램의 어떠한 변화도 없고, 입력 값에 대한 결과를 예상할수 있어서 테스트하기가 쉽다.

*// 순수하지 않은 함수, DOM을 변경하는 부수효과를 발생시킴*

function Header(*text*) {

  let h1 = document.createElement('h1');

  h1.innerText = *text*;

  document.body.appendChild(h1);

}

*// 순수한 함수, 부수효과를 발생시키지 않음*

*// DOM을 변경하는 책임은 애플리케이션의 다른 부분이 담당하도록 한다.*

const Header = (*props*) => <h1>{*props*.title}</h1>

출처 : <https://velog.io/@kyusung/%ED%95%A8%EC%88%98%ED%98%95-%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D-%EC%9A%94%EC%95%BD>