Section 8: Functional Programming in Depth

* Introduction

01:53

* What is Functional Programming
* • in OOP we get Object in modified state ie modify we do in outer environment...here how pure mathematical function behave ..the pure mathematical job is only to perform calculation on the basic of provide algorithms and data ..it should nothing any effect on outer environment it does not have anything to do with object state it just uses the provided data for calculation

perform the calculations that’s all, functional programing avoids changing the state and mutable data

it is Declarative type style of programming that focus on

**"What to solve rather than how to solve"**

it comes from Lambda Calculus

OOP Vs Functional Programming

================================

in Opps Objects method(Object obj);

in FP we can do same with function,

here we can create function within function and we can return function from function

Supplier method(Function fun)

Preview03:47

* Functional Programming Concepts

Preview00:56

Key Concepts

===============

Function as first class citizens

Pure Function

Higher Order Functions

Others Key point

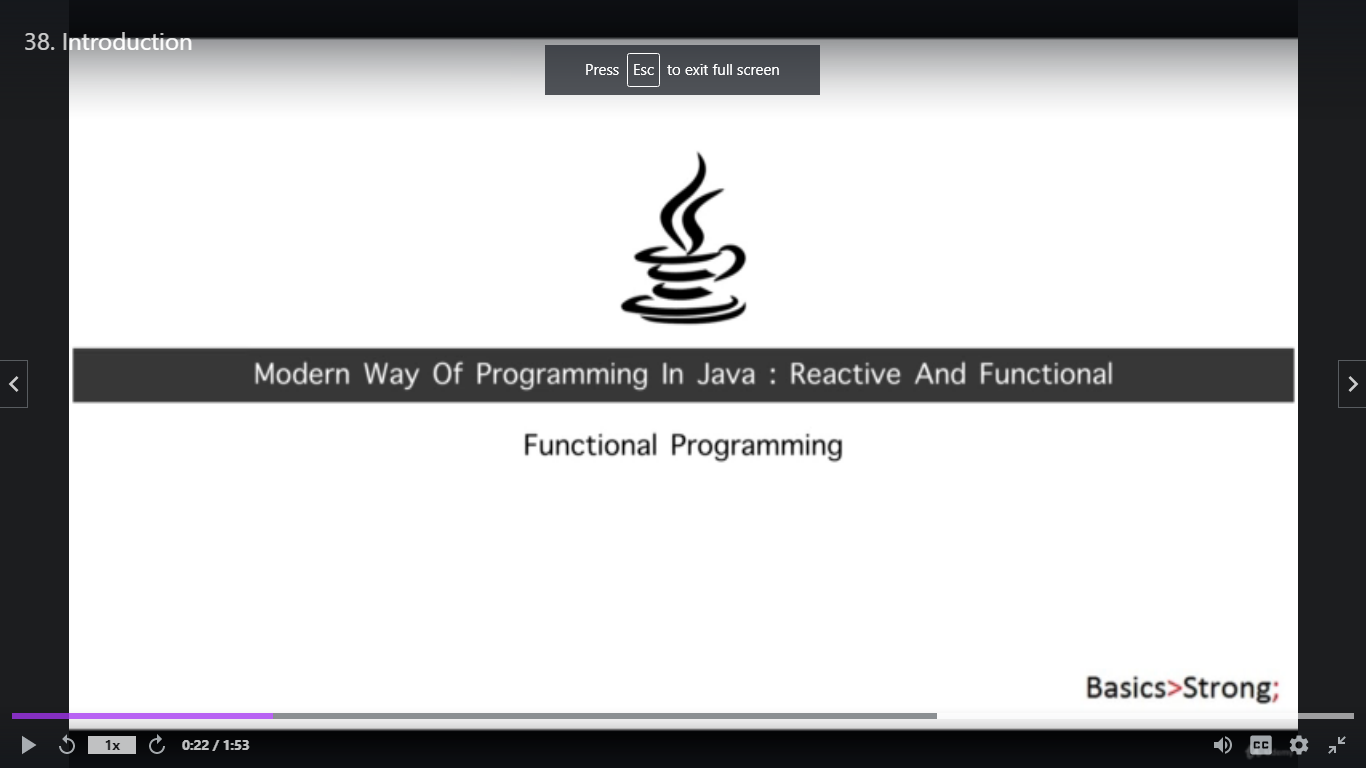
No Side Effects

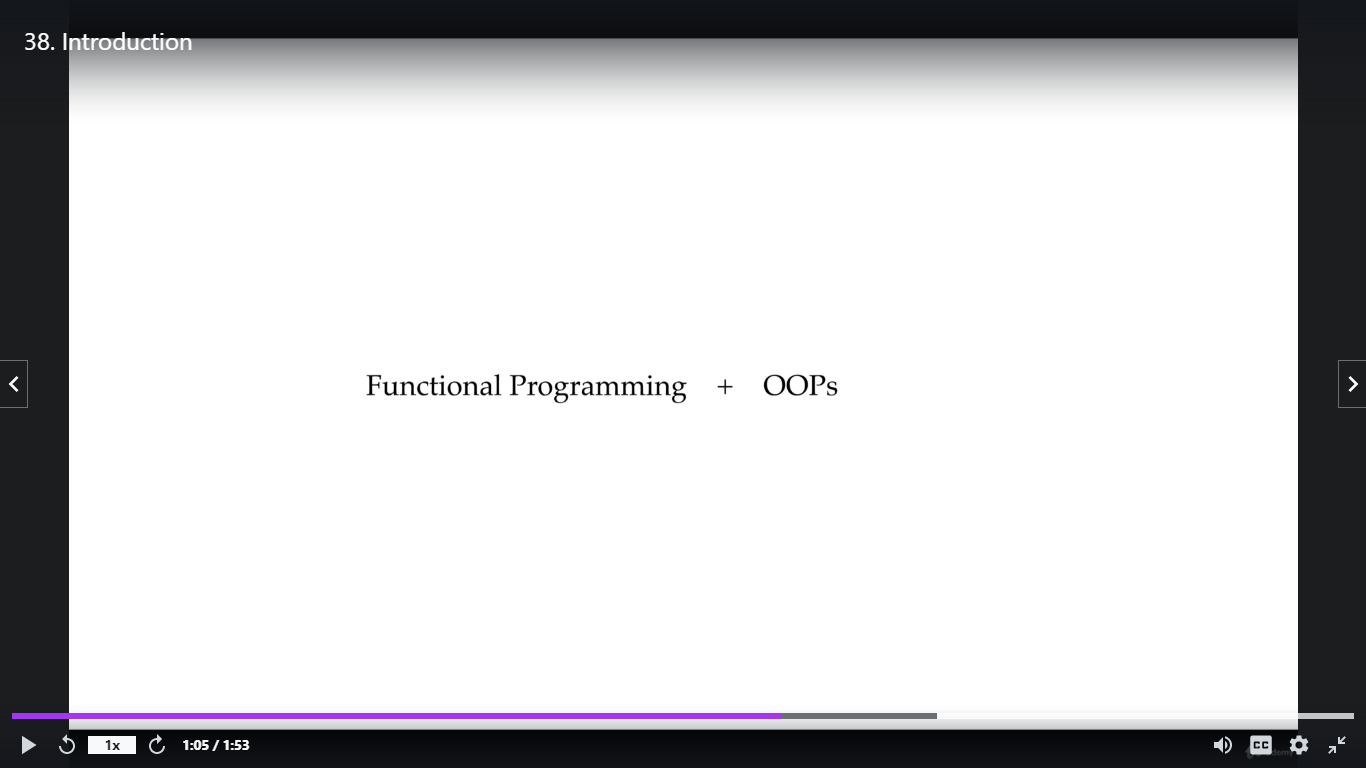
Referential Transparency

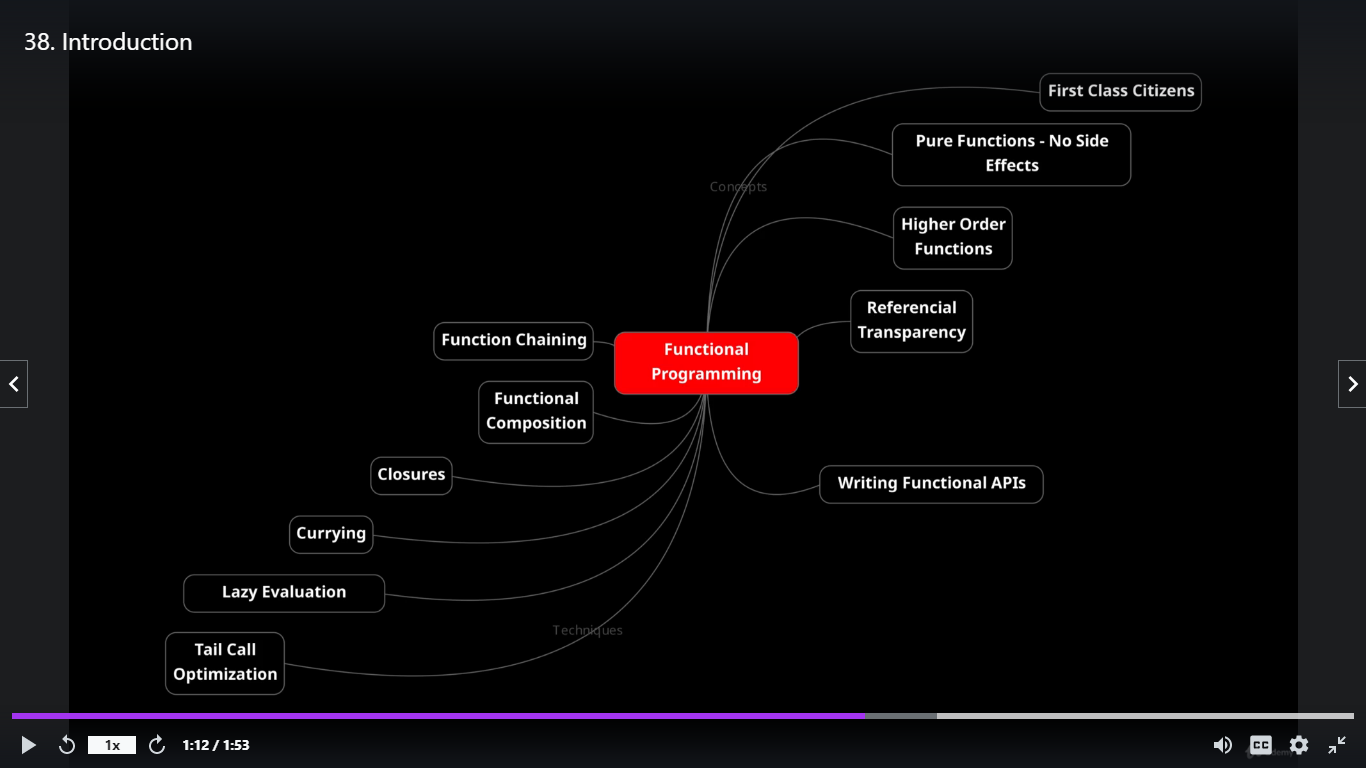
* Functions as First class Citizens
* No Side Effects - Pure Functions
* Higher Order Functions
* Referencial Transparency
* Functional Programming Techniques
* Designing APIs In Functional Way
* Chaining
* Composition
* Closures
* Currying
* Lazy Evaluation
* Tail Call Optimisation aka TCO
* Summary

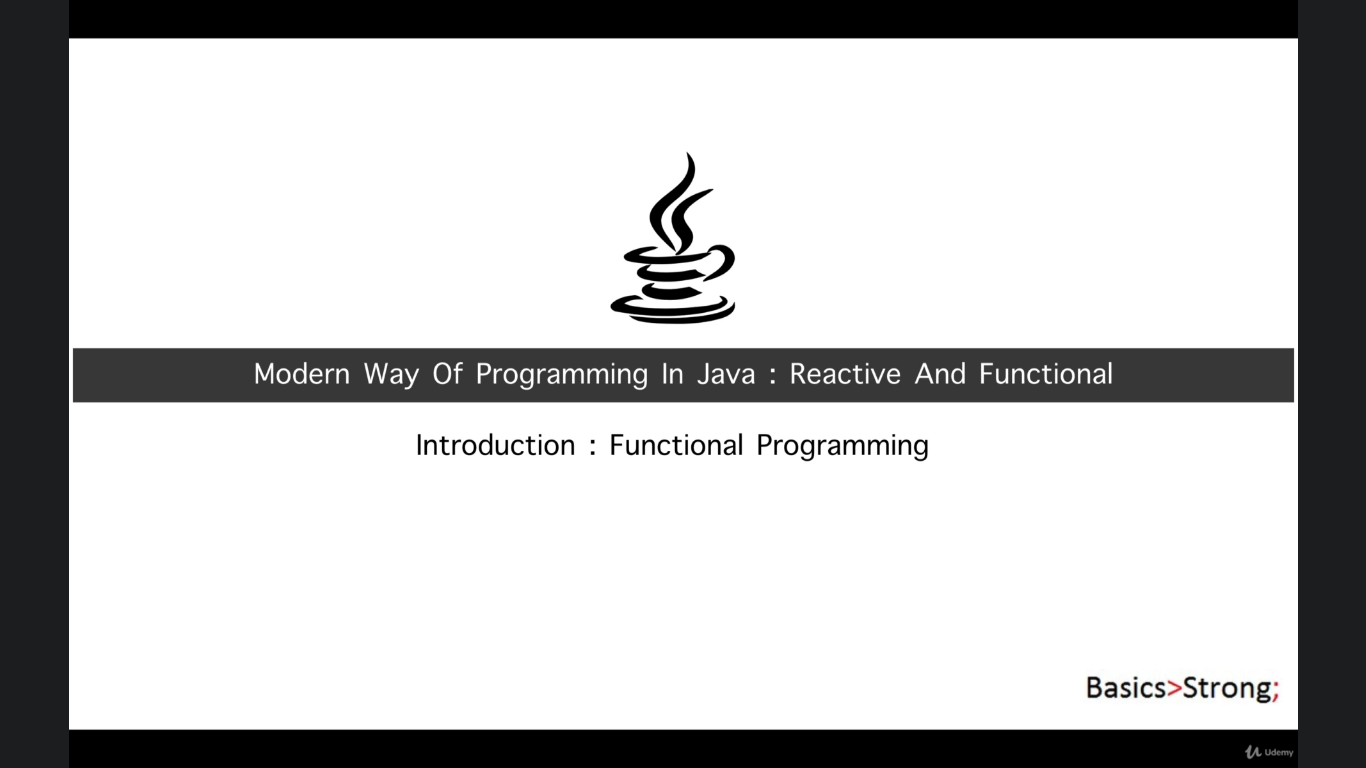
==============================================================================================================================================

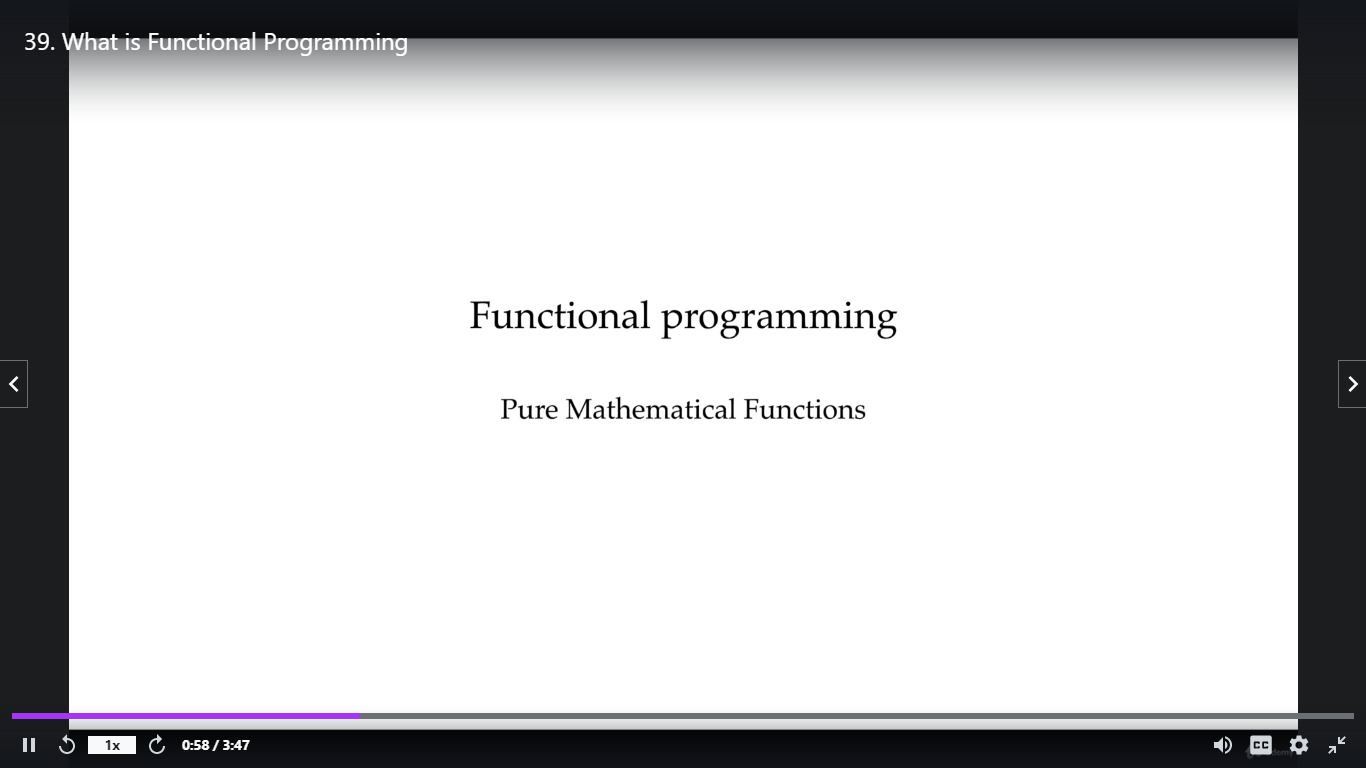
* **What is Functional Programming**



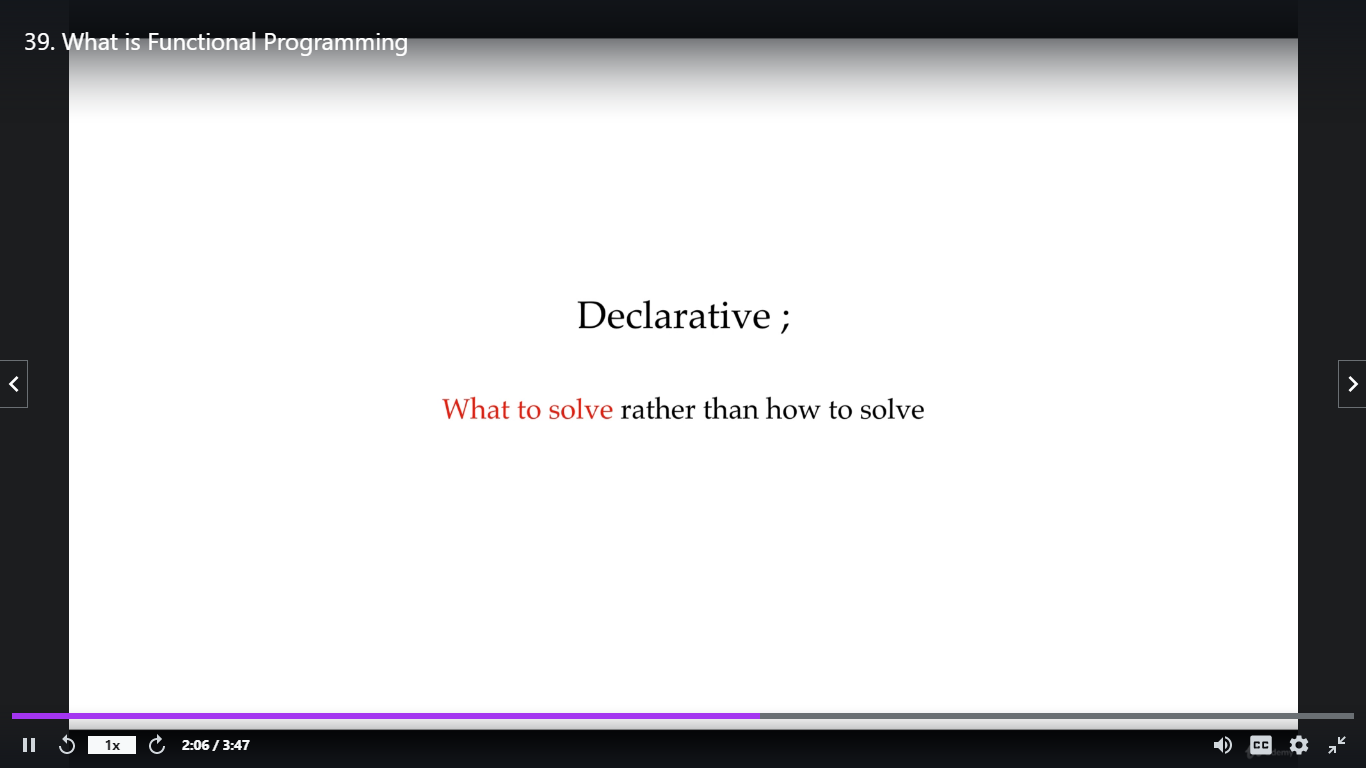


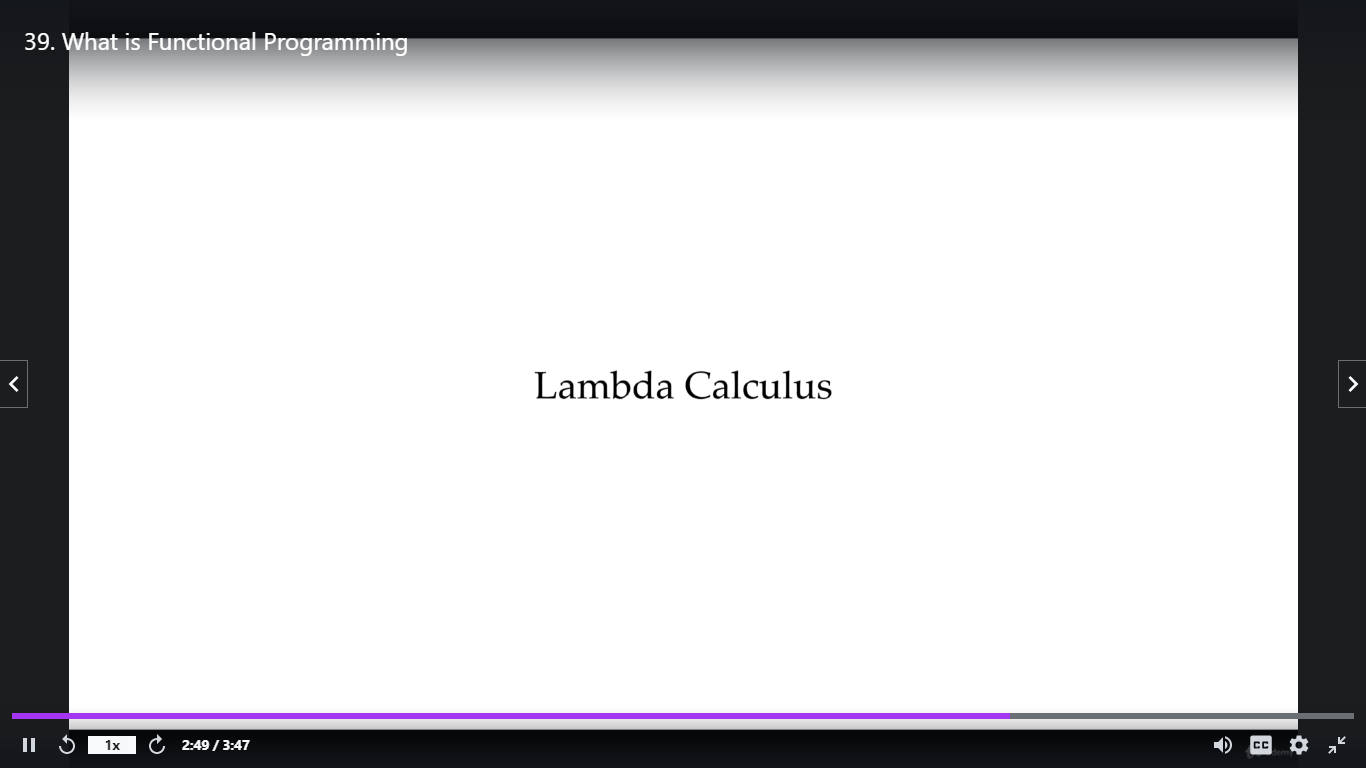


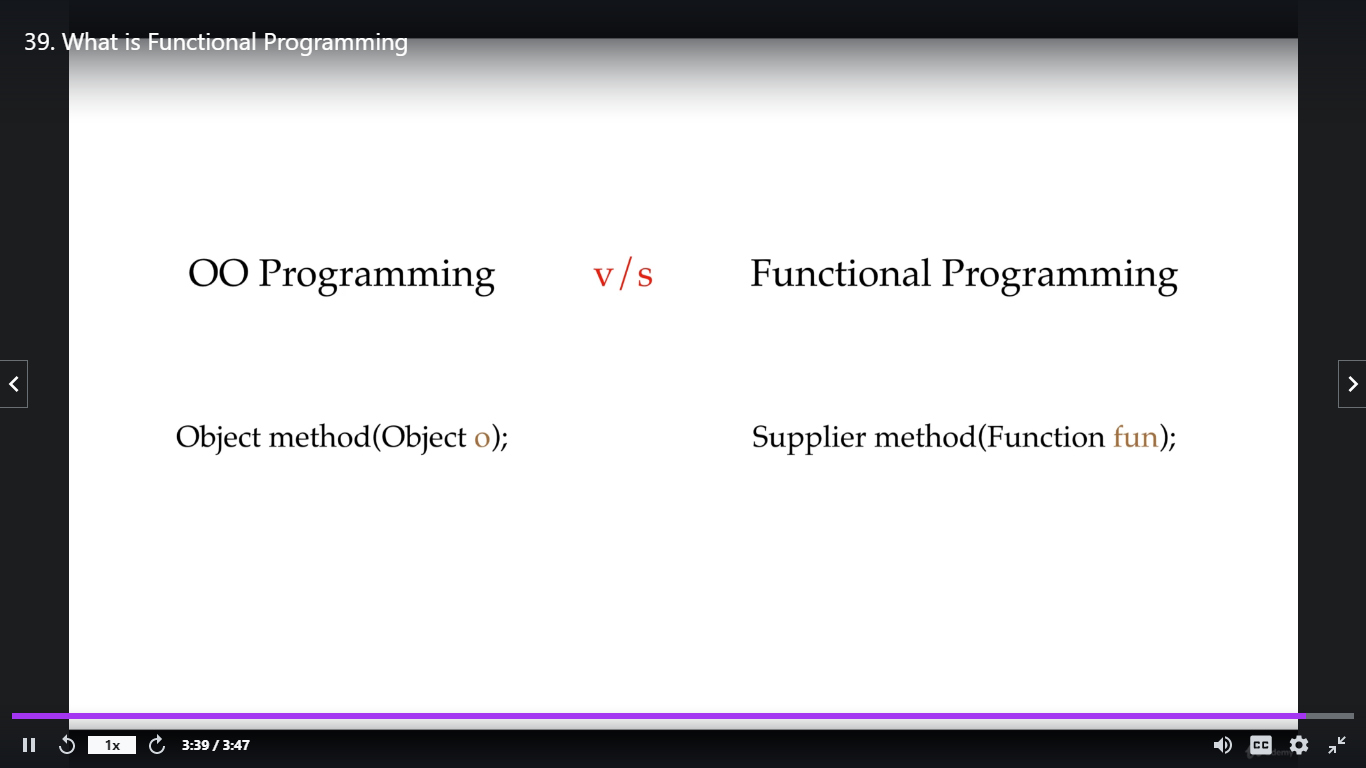


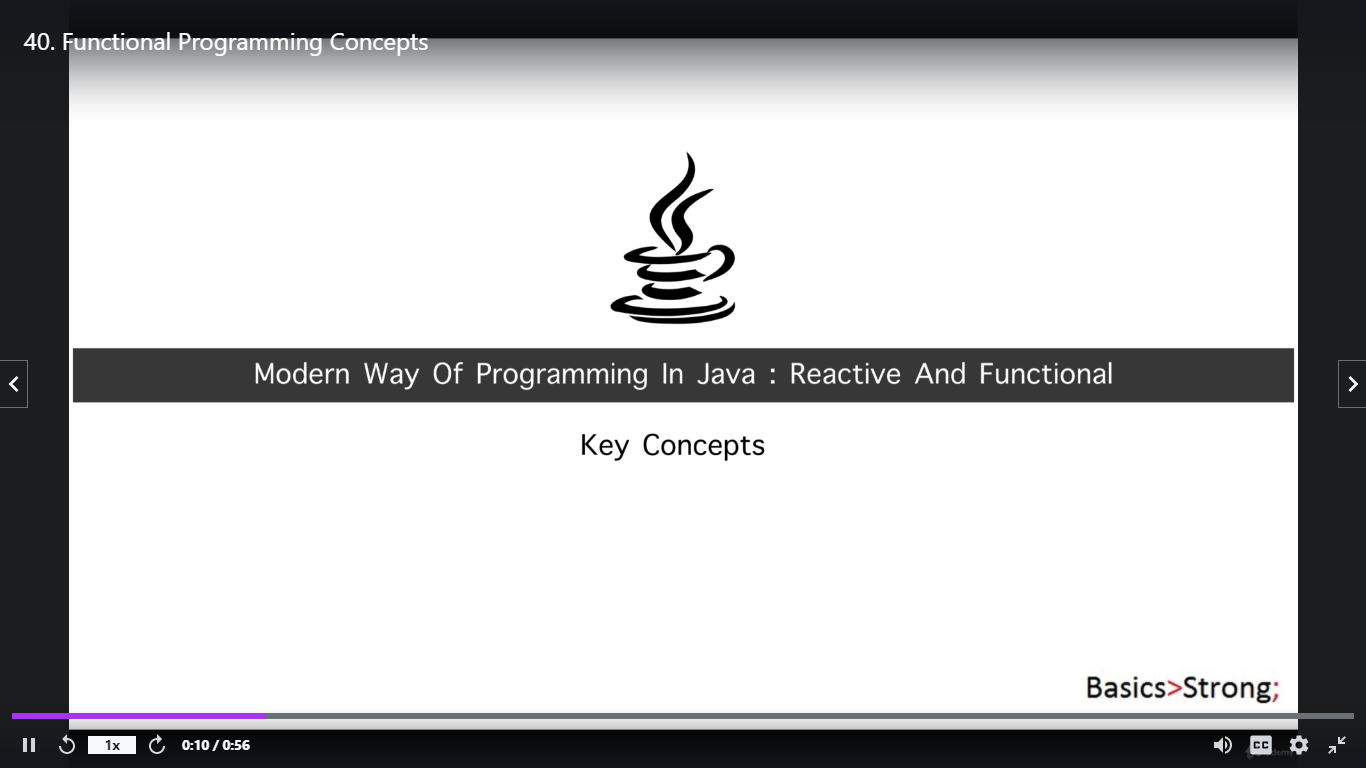


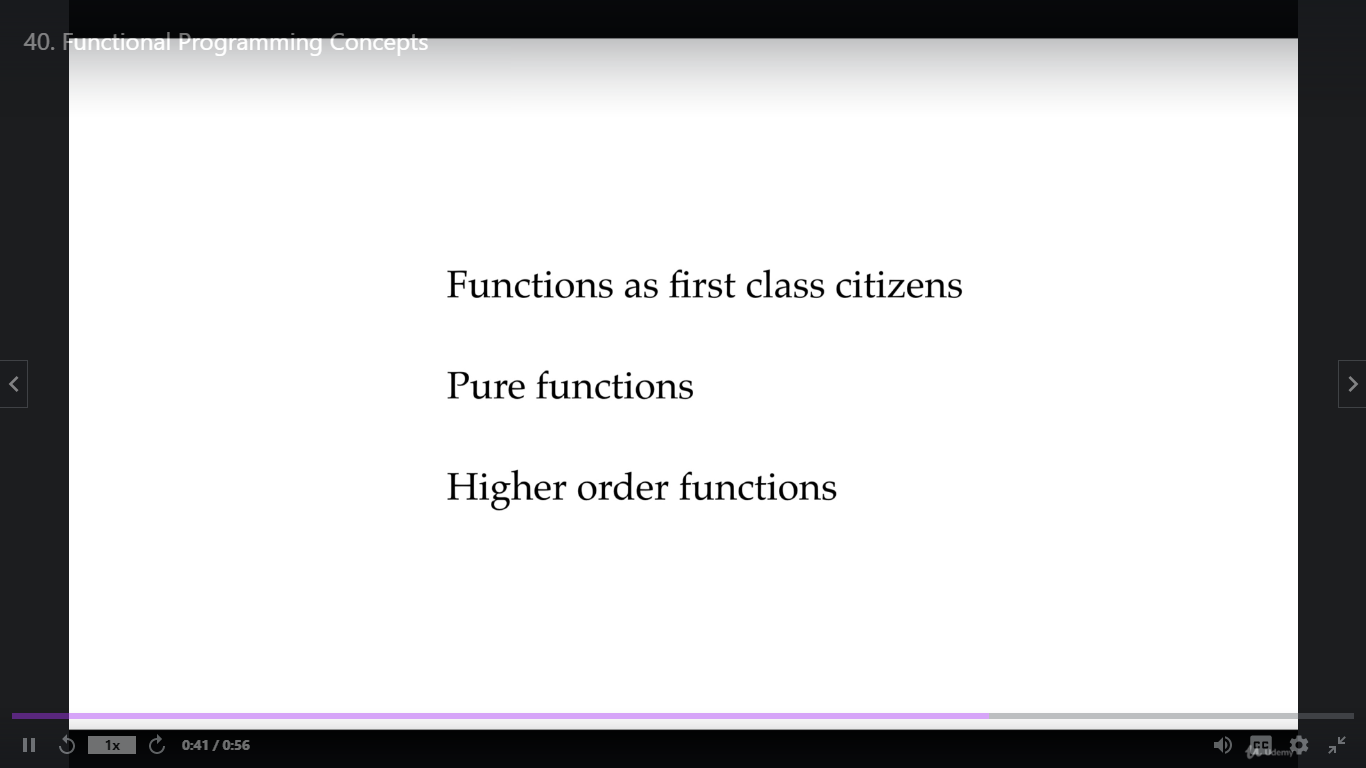
**Functional programming makes use of expression**

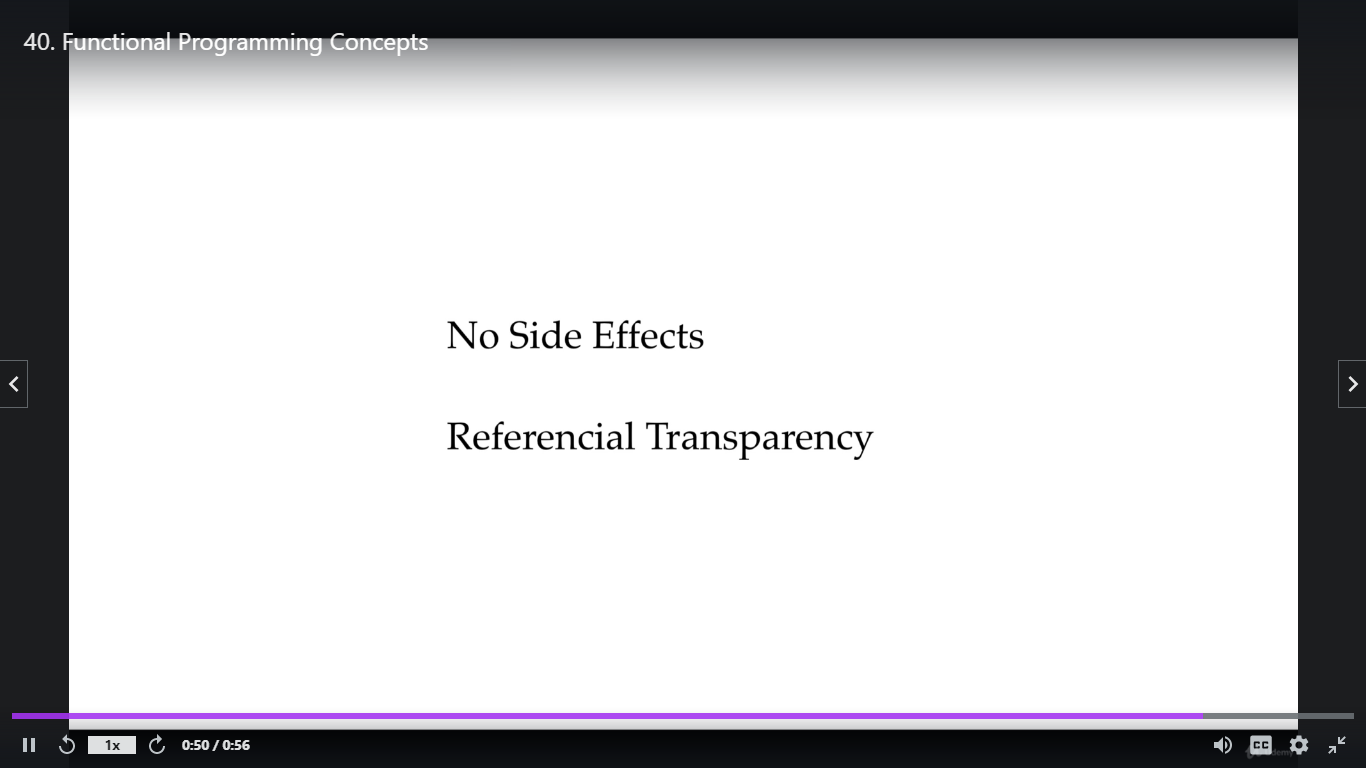


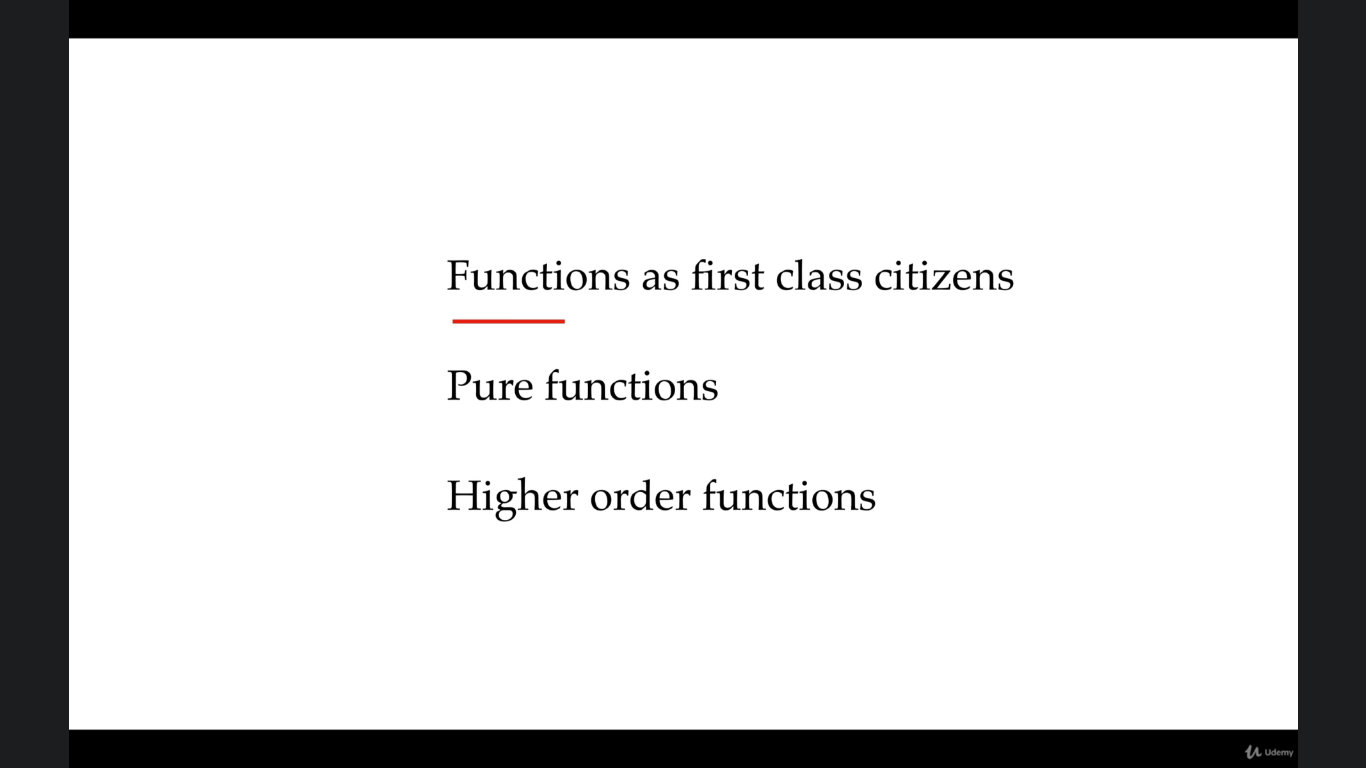


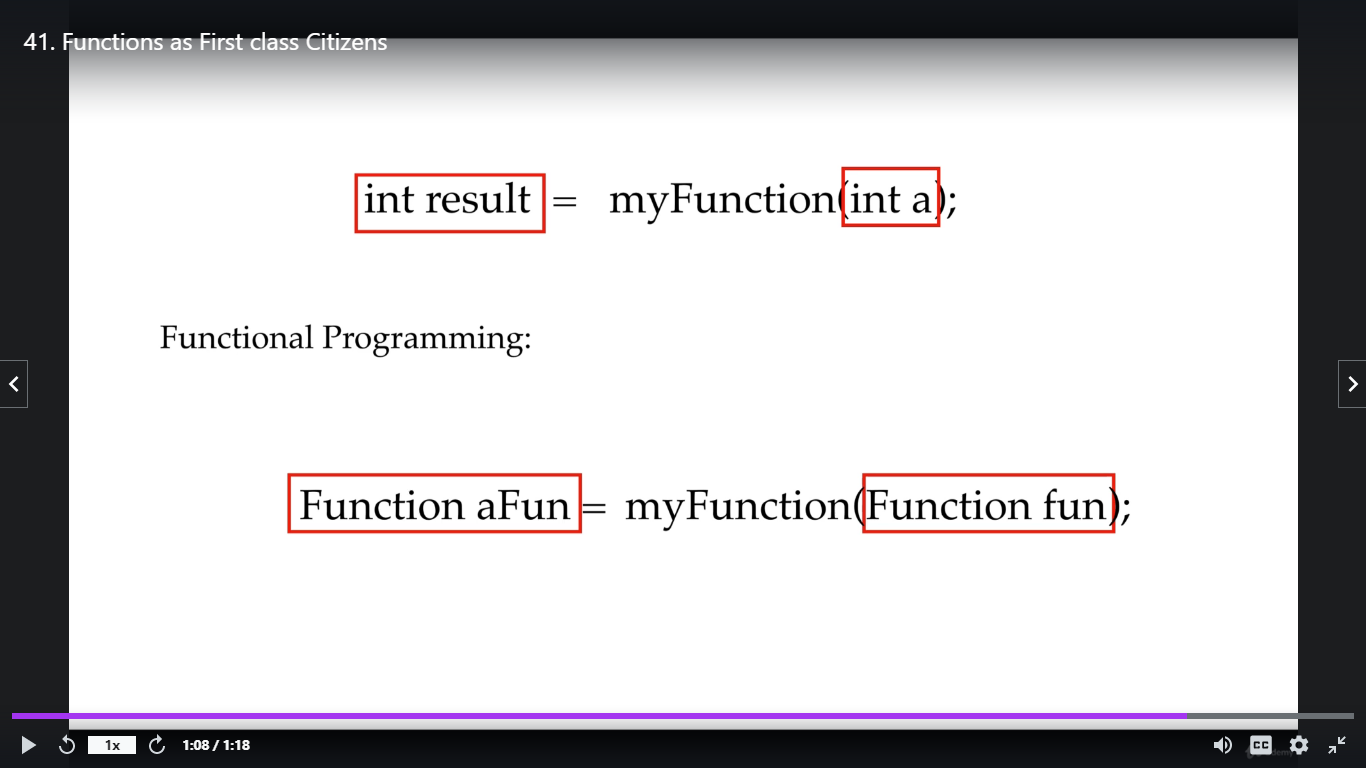




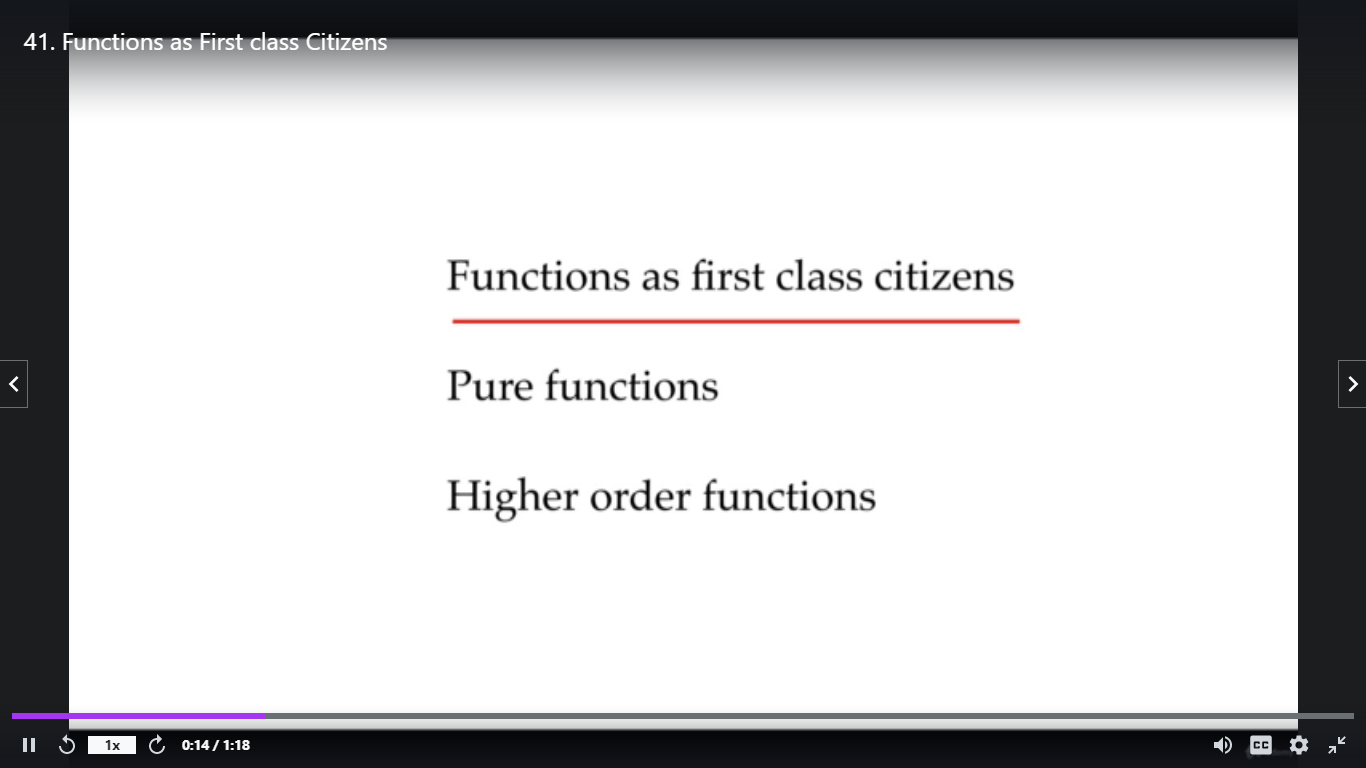


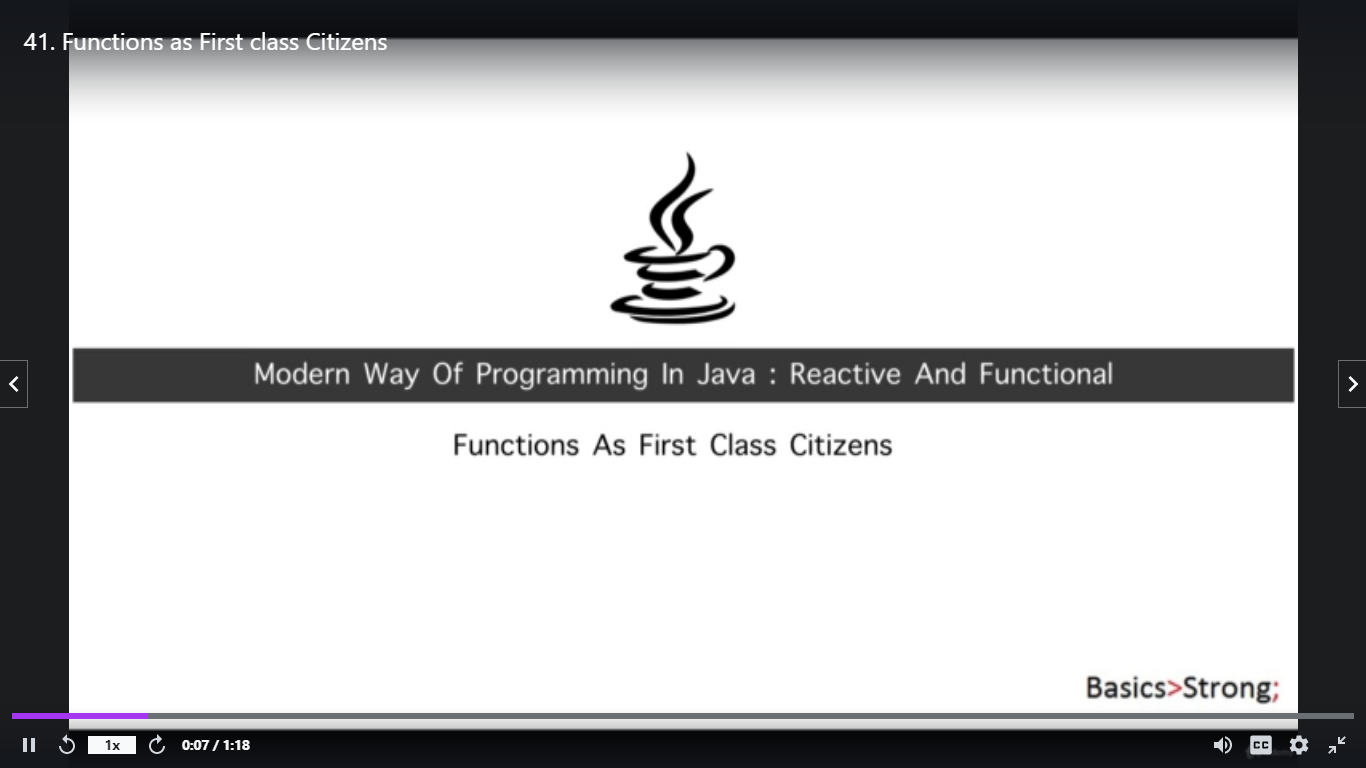


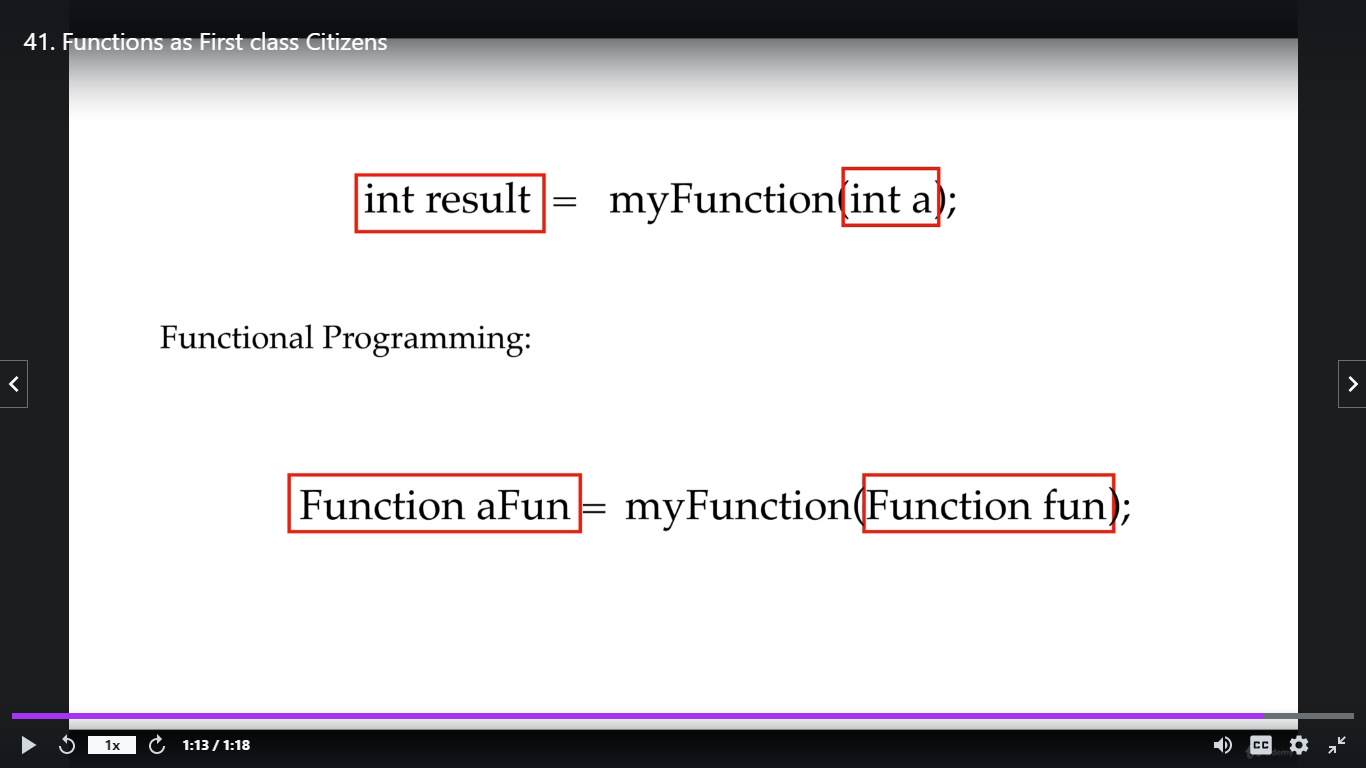




* **Functional Programming Concepts**







Pure Function

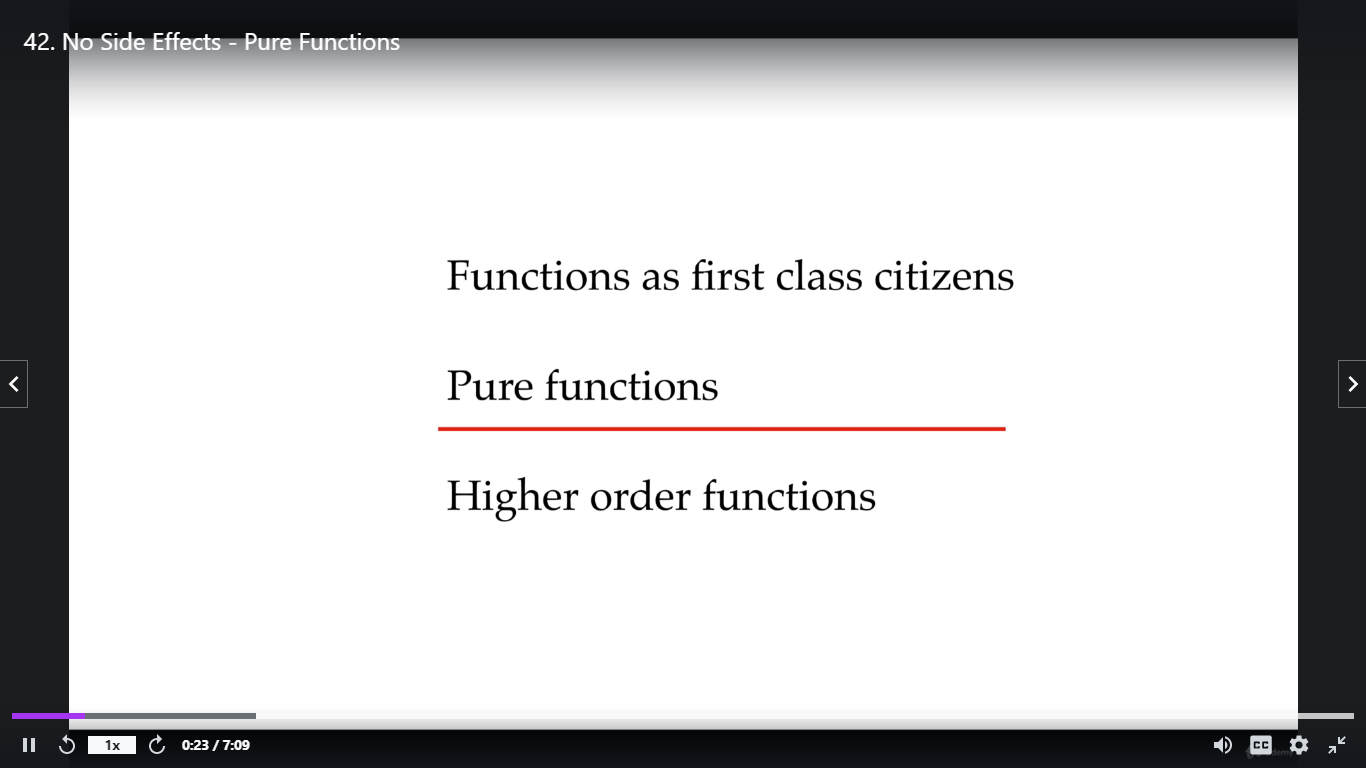
------------

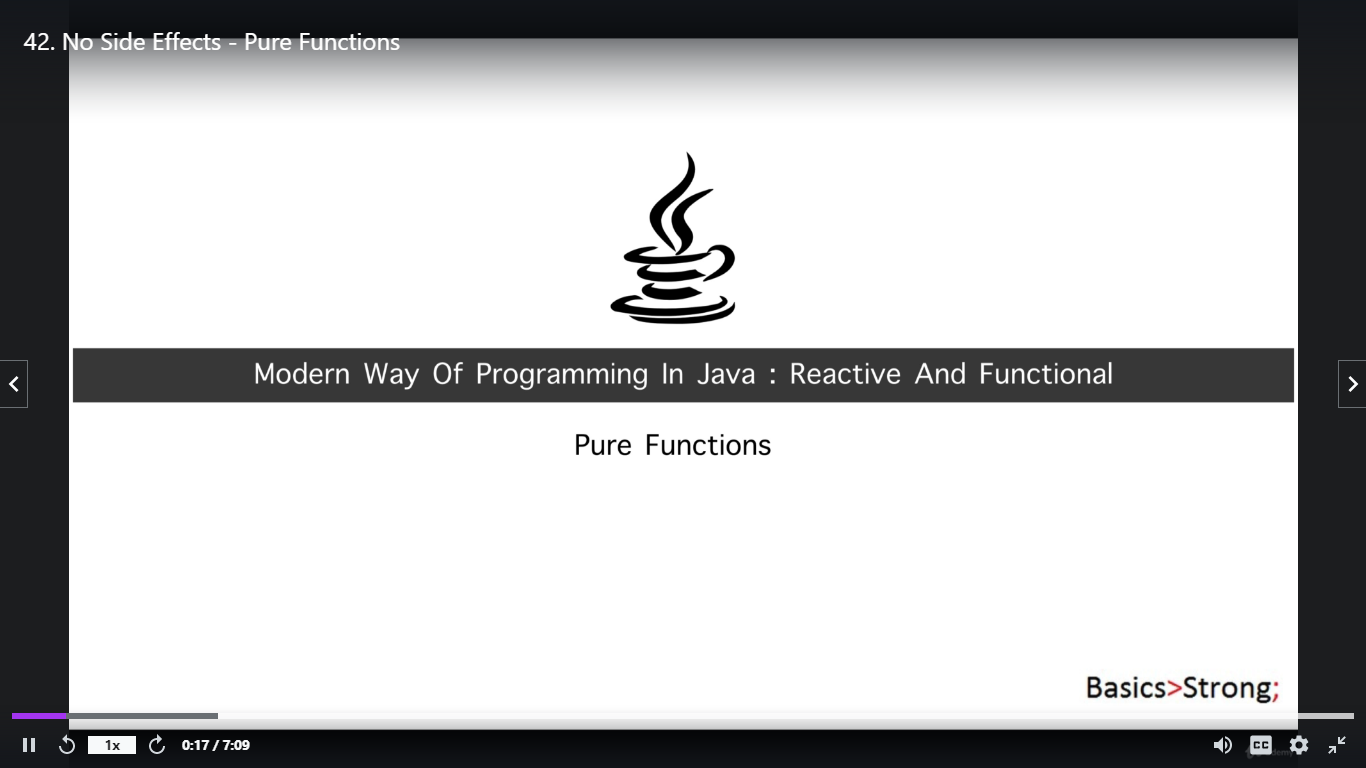
Pure Function always returns same result with same input, it never changes the argument that passed to it

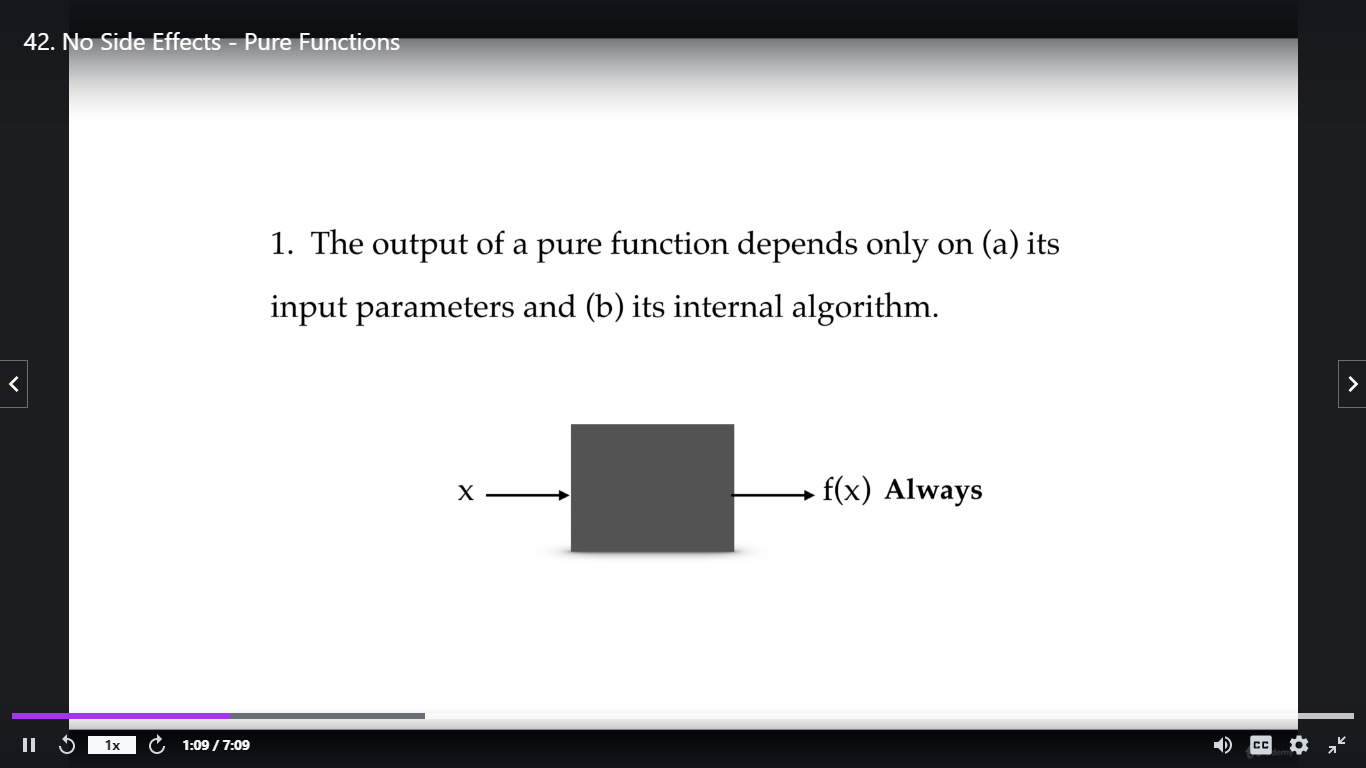
Pure function never change/update/modify the input value rather it does operation/calculation on it

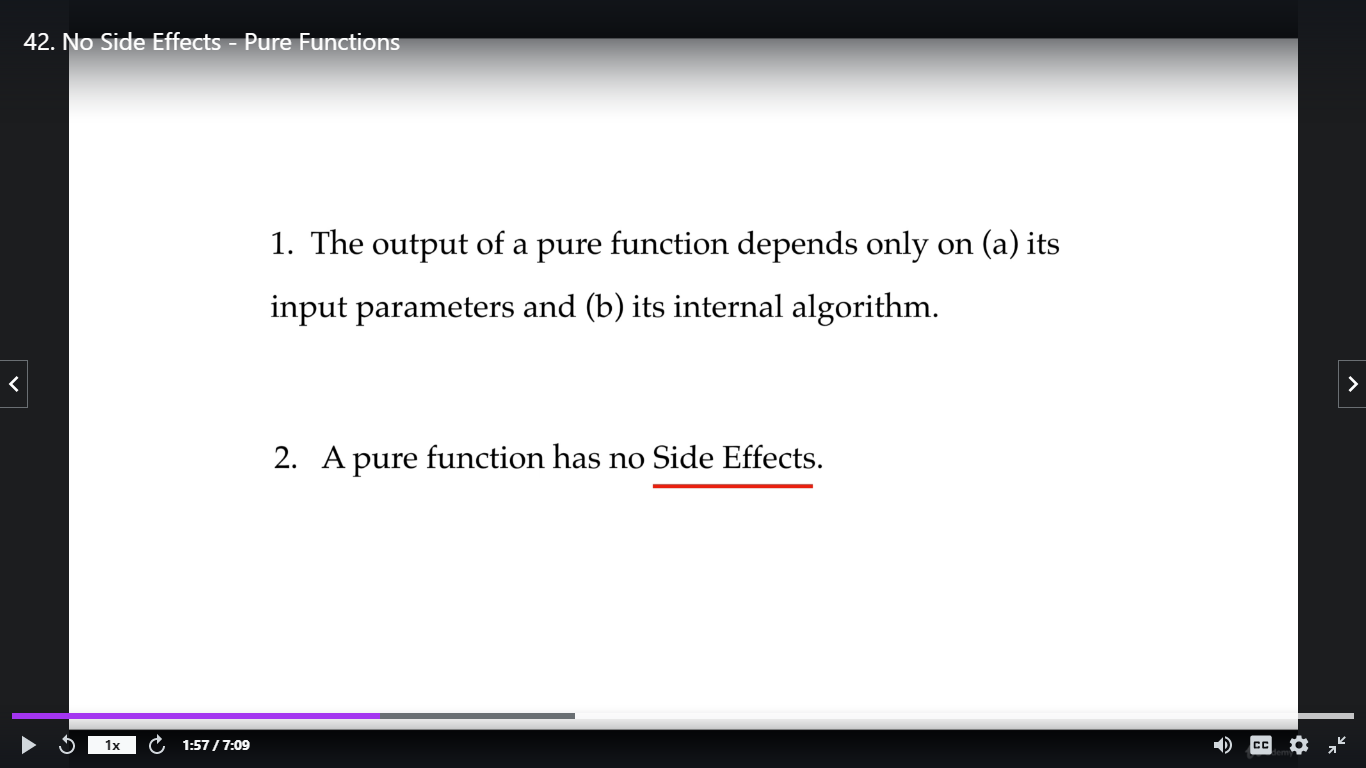
Pure function can use fearless in multithreaded environment

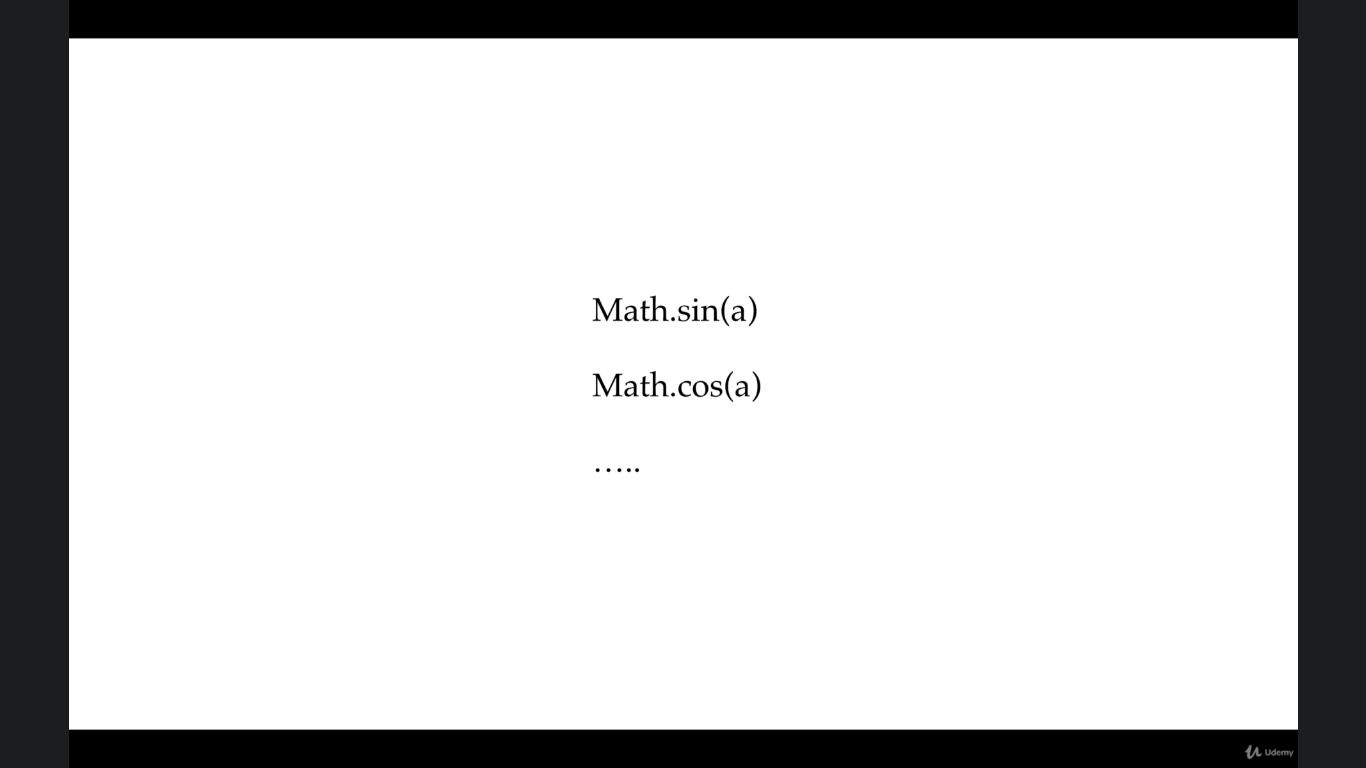
Pure Function

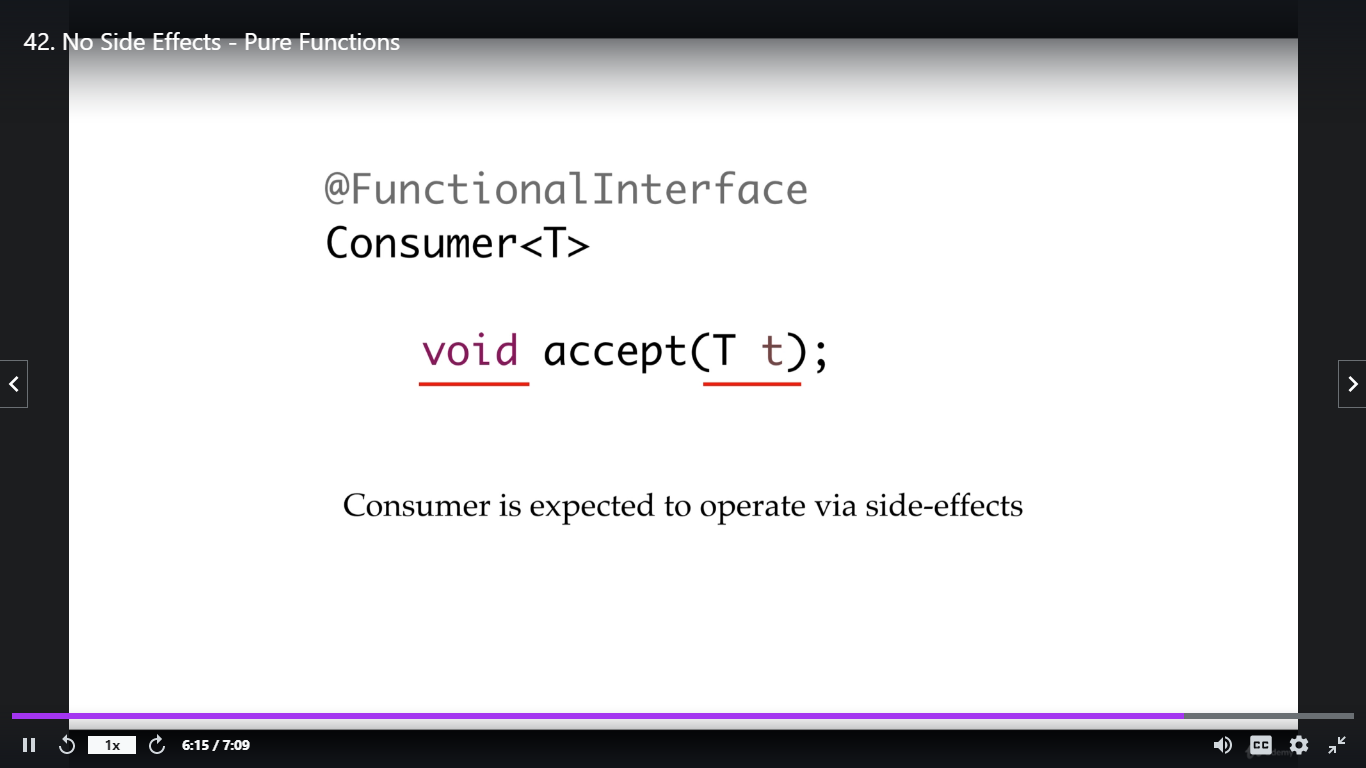


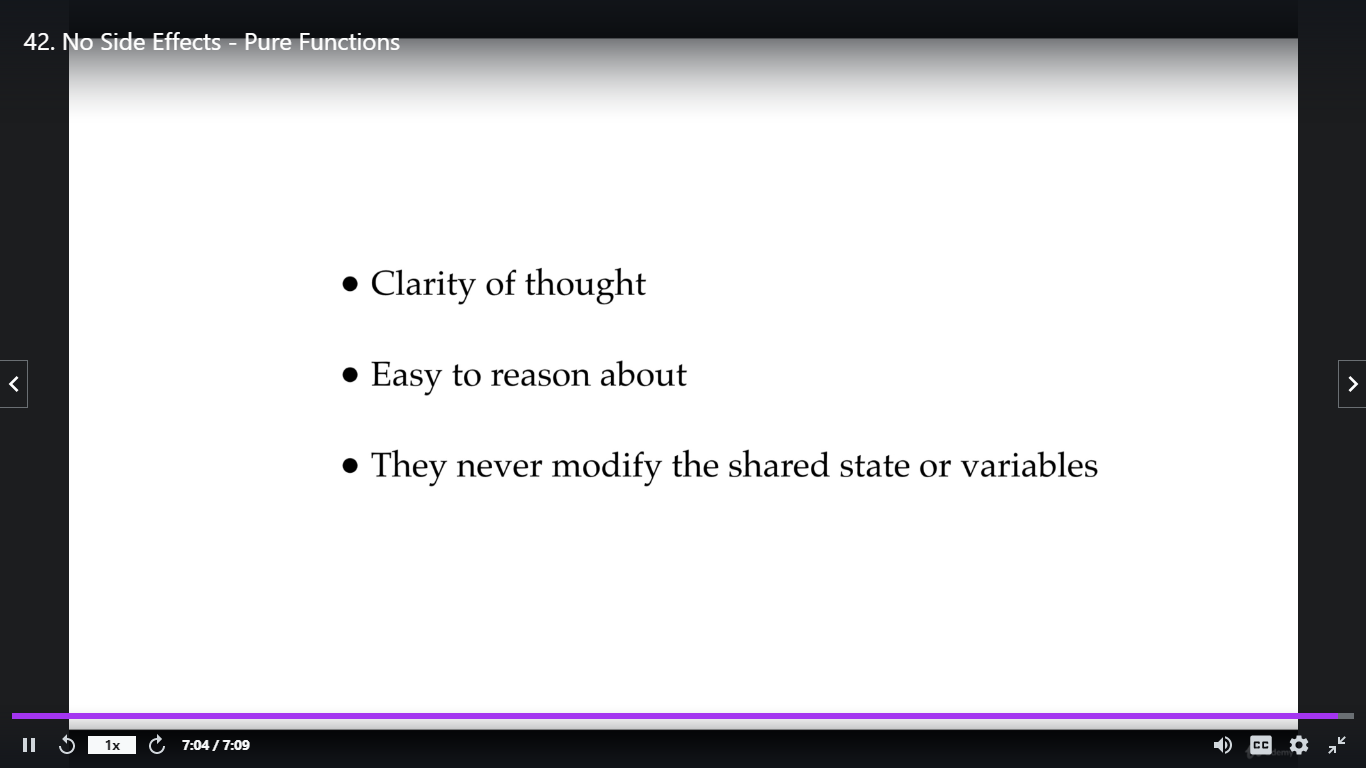






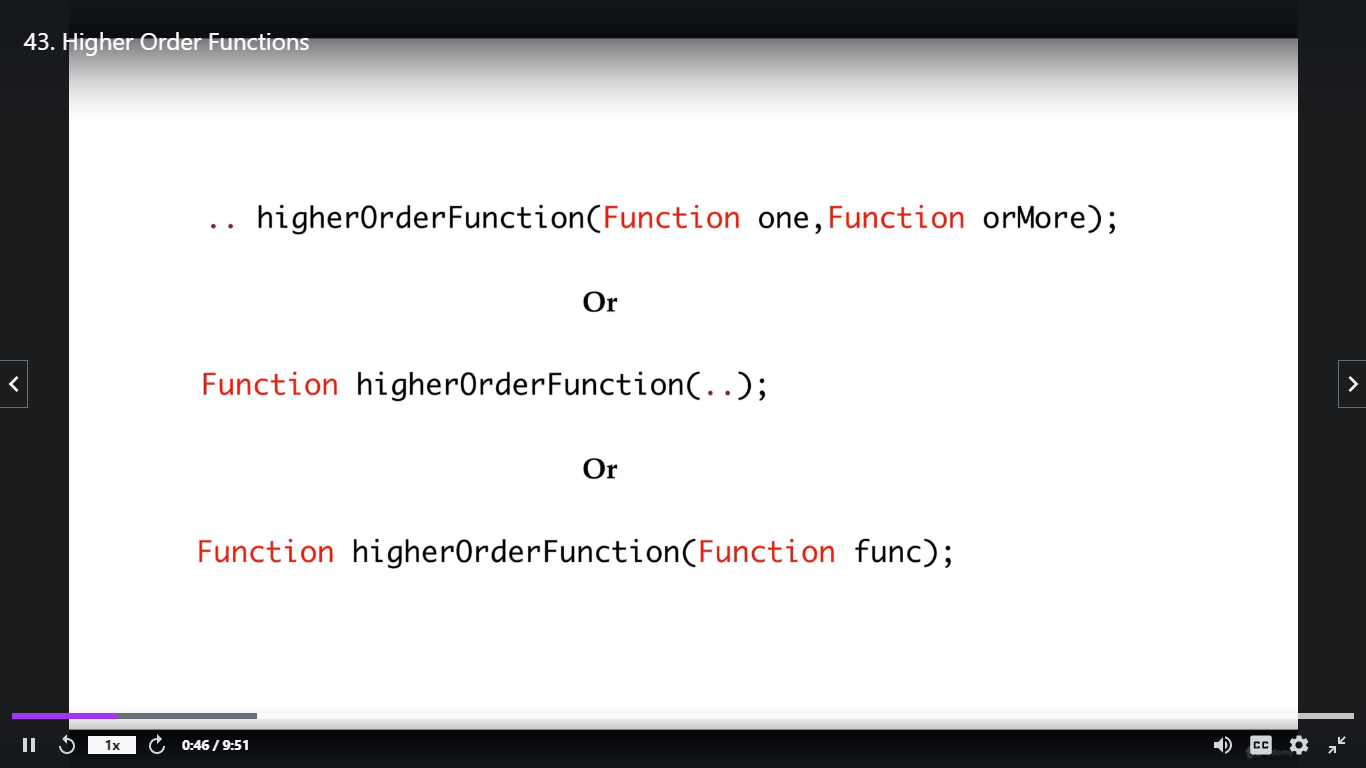


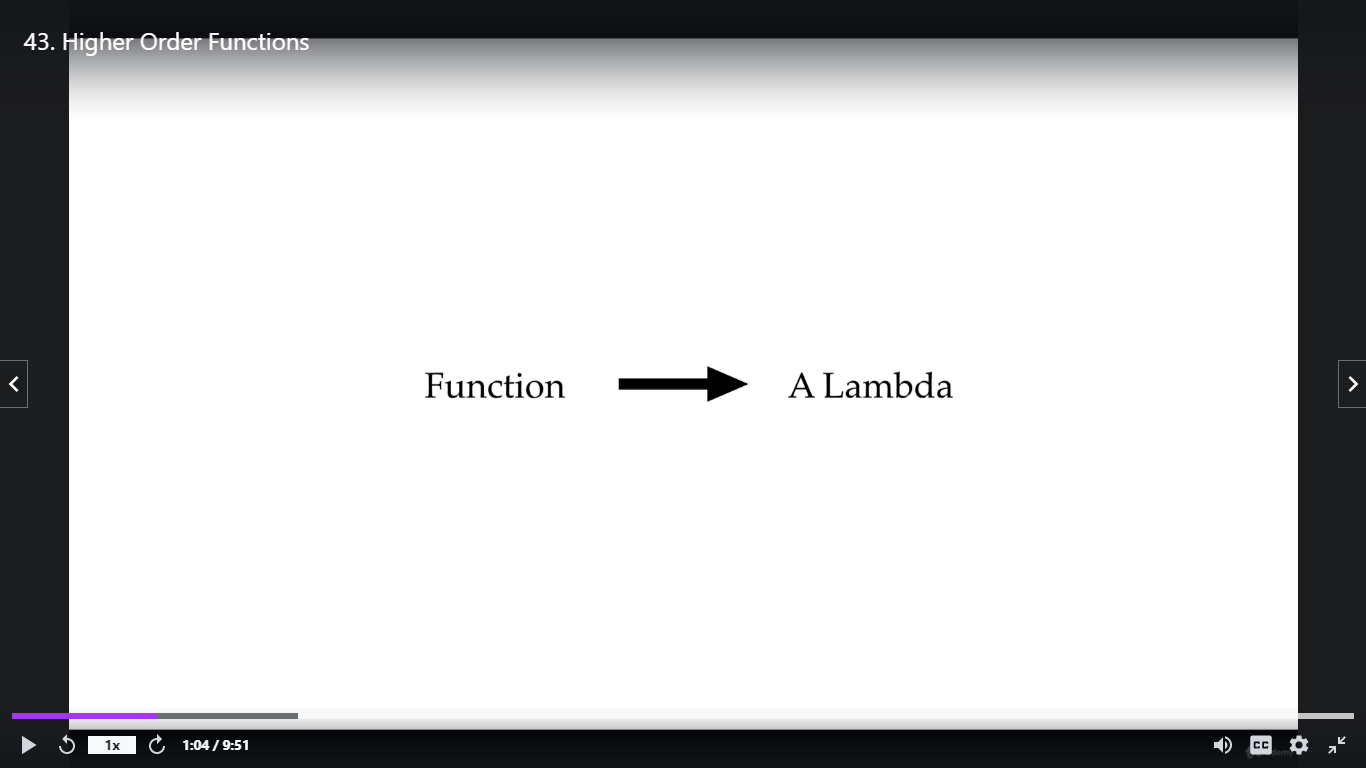




Higher Order Functions

It is a function that takes function as an argument or return function output or does both

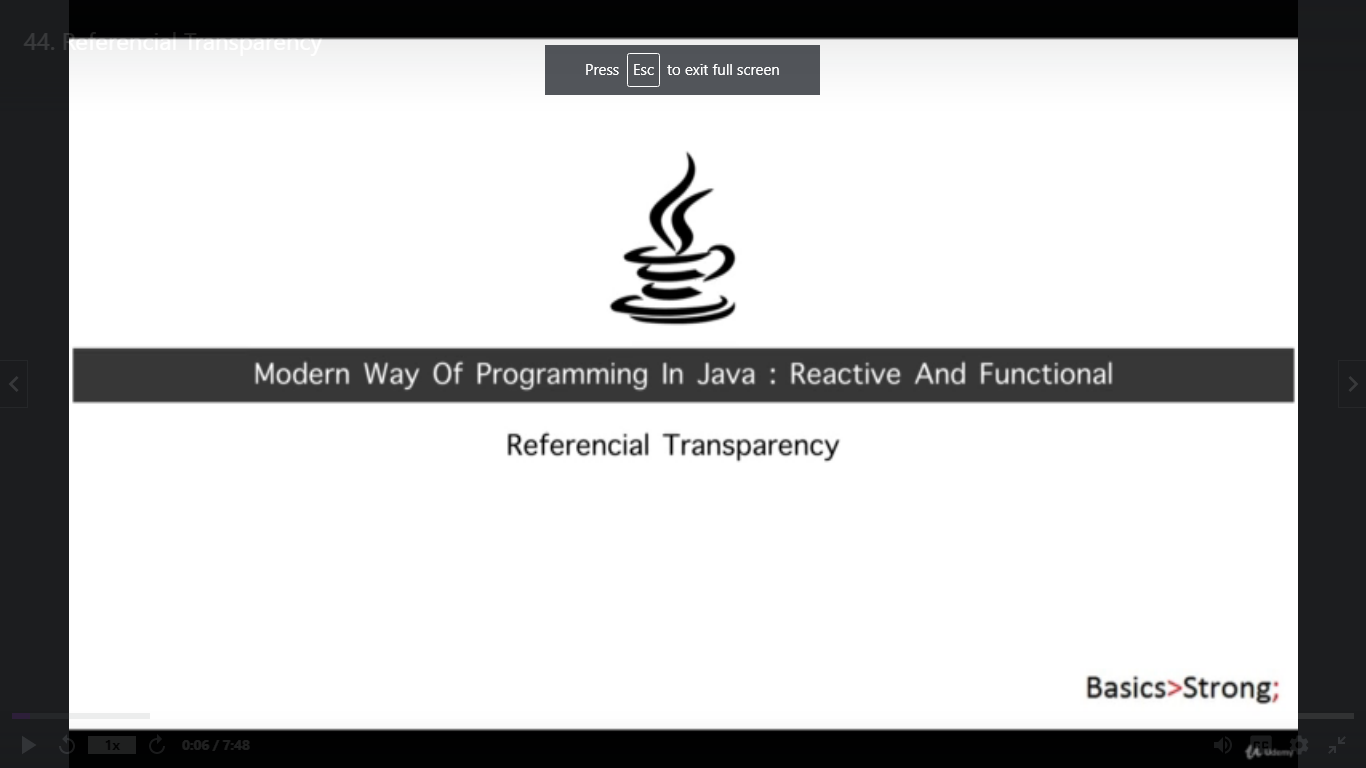




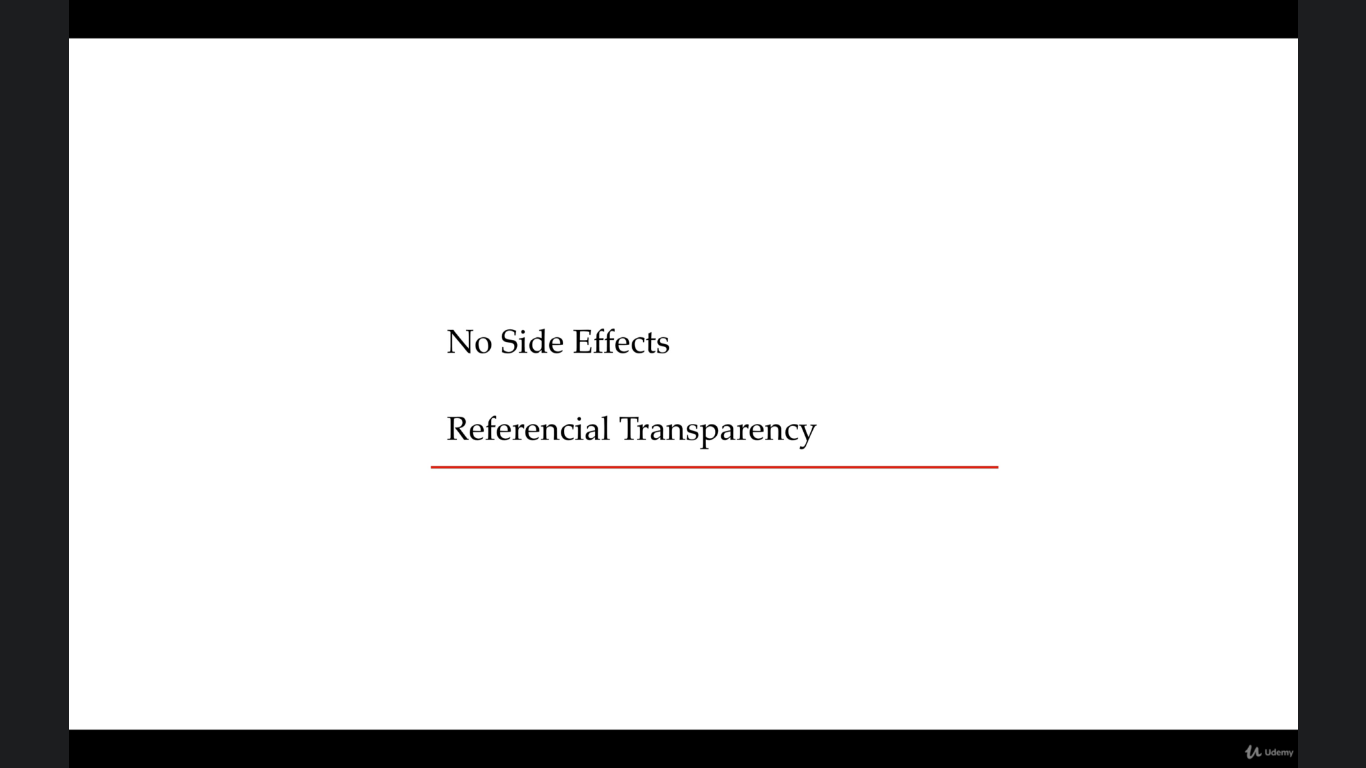
Higher order function that takes Lambda as argument and return Lambada or does both

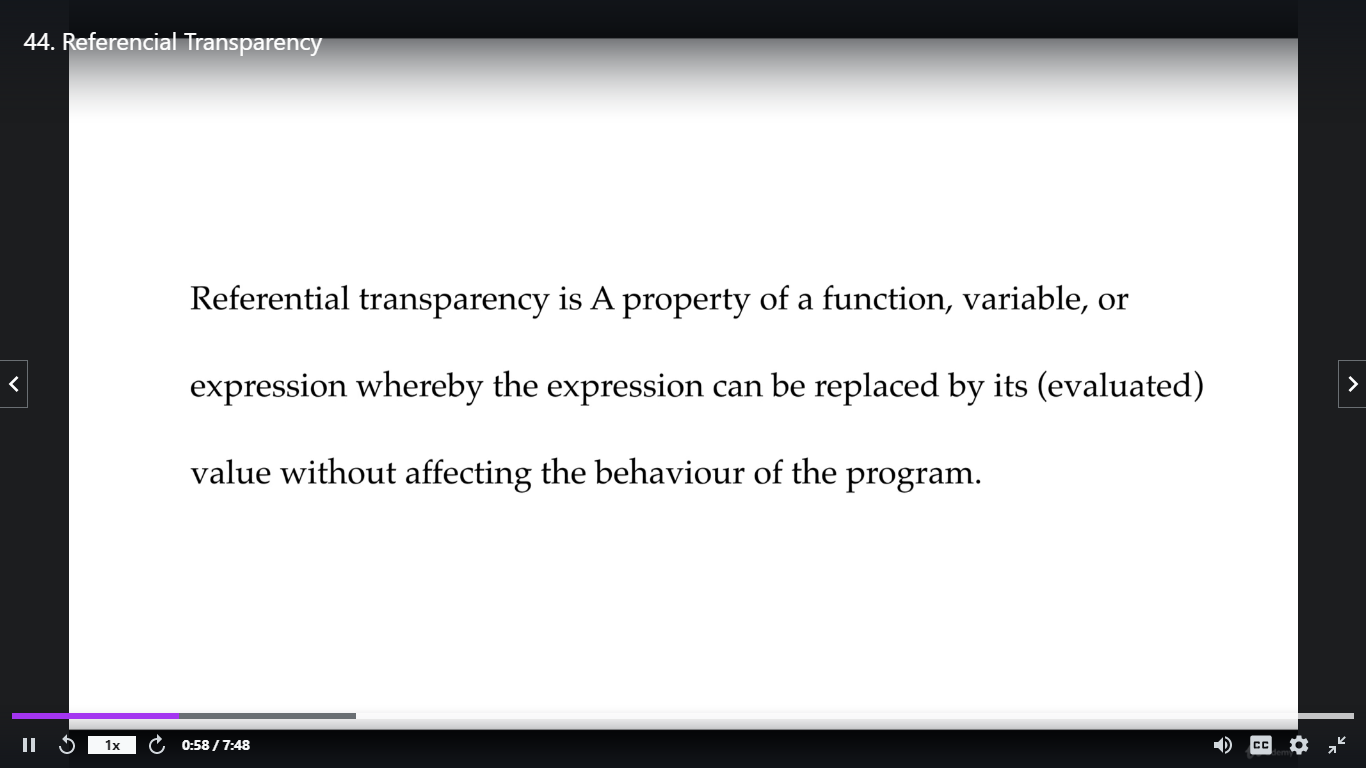
* Referential Transparency

============================



\*Pure function has no any side effect





Ex-

Tiger is bigger than kite ⬄ (fill in the balnk) is bigger than kite

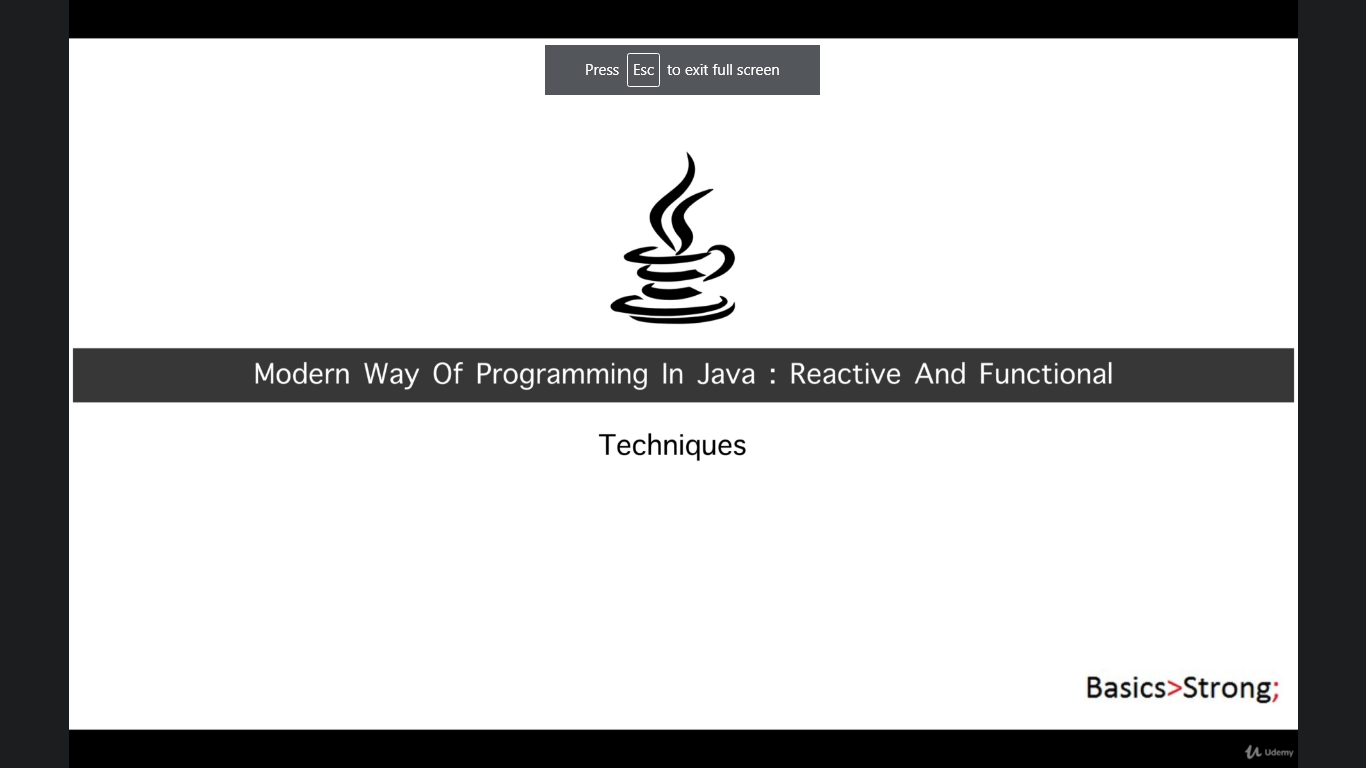
India capital is bigger than Kite/Delhi is bigger than kite

Above both statements are transparent in blank

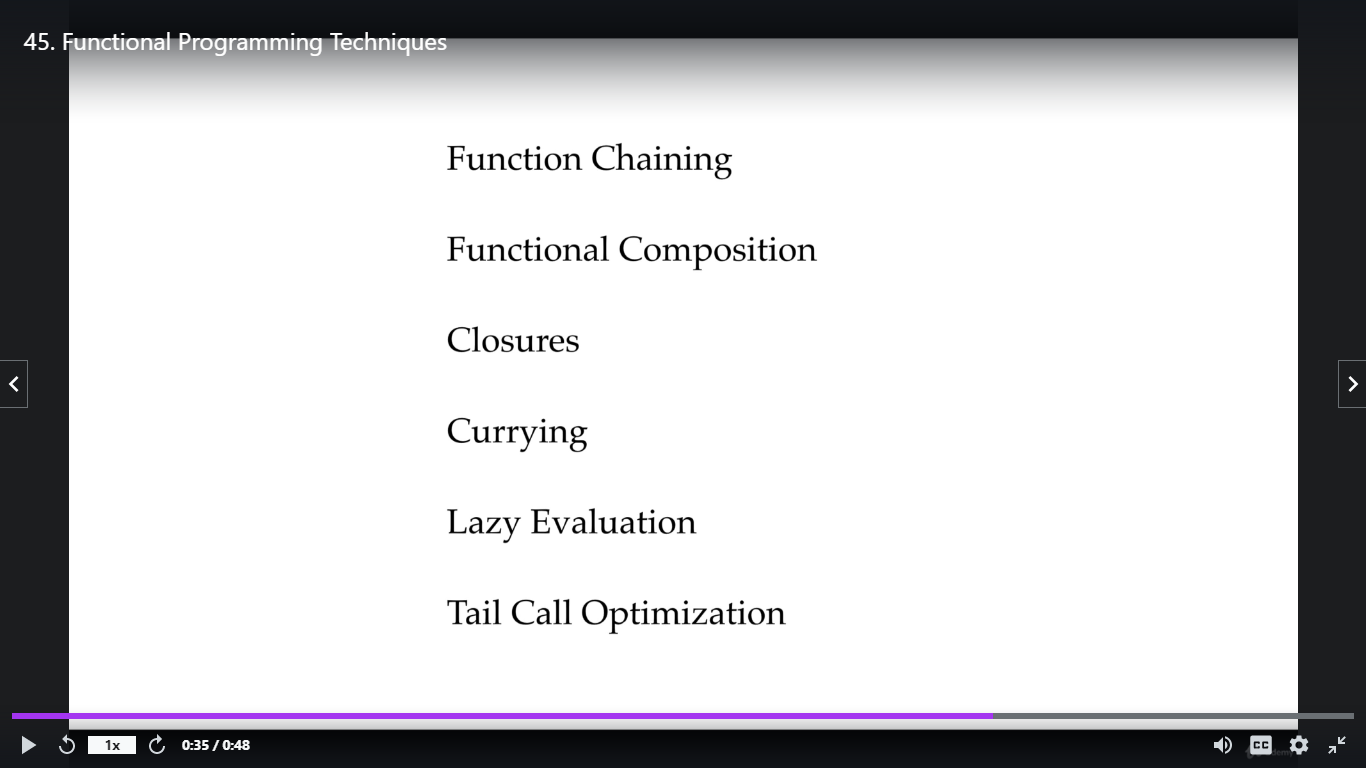
Ex- x= 7+(5\*2) ⬄ 7+(10), ⬄ 7+(5+5) all are same

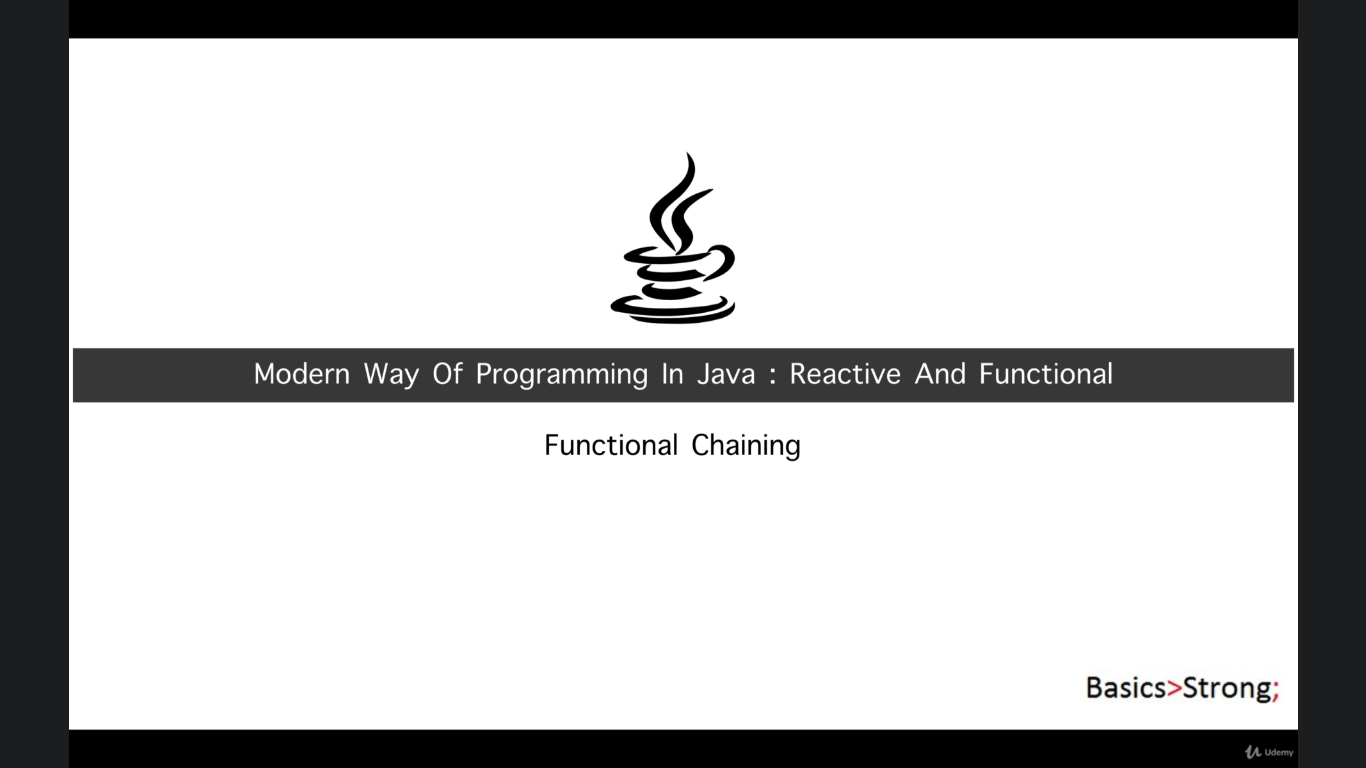
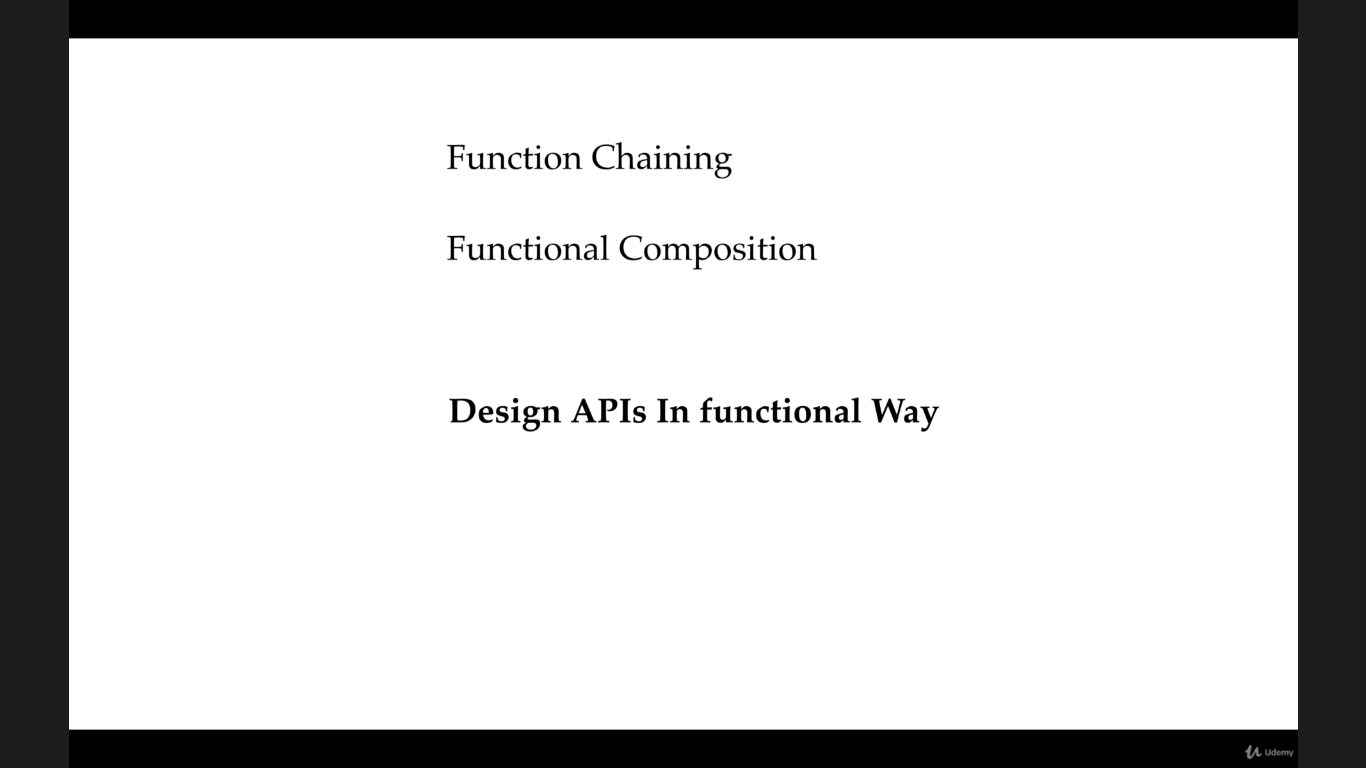
Pure function always achieves referential transparent but referential transparent may or may not be pure function

Functional Programming Technique

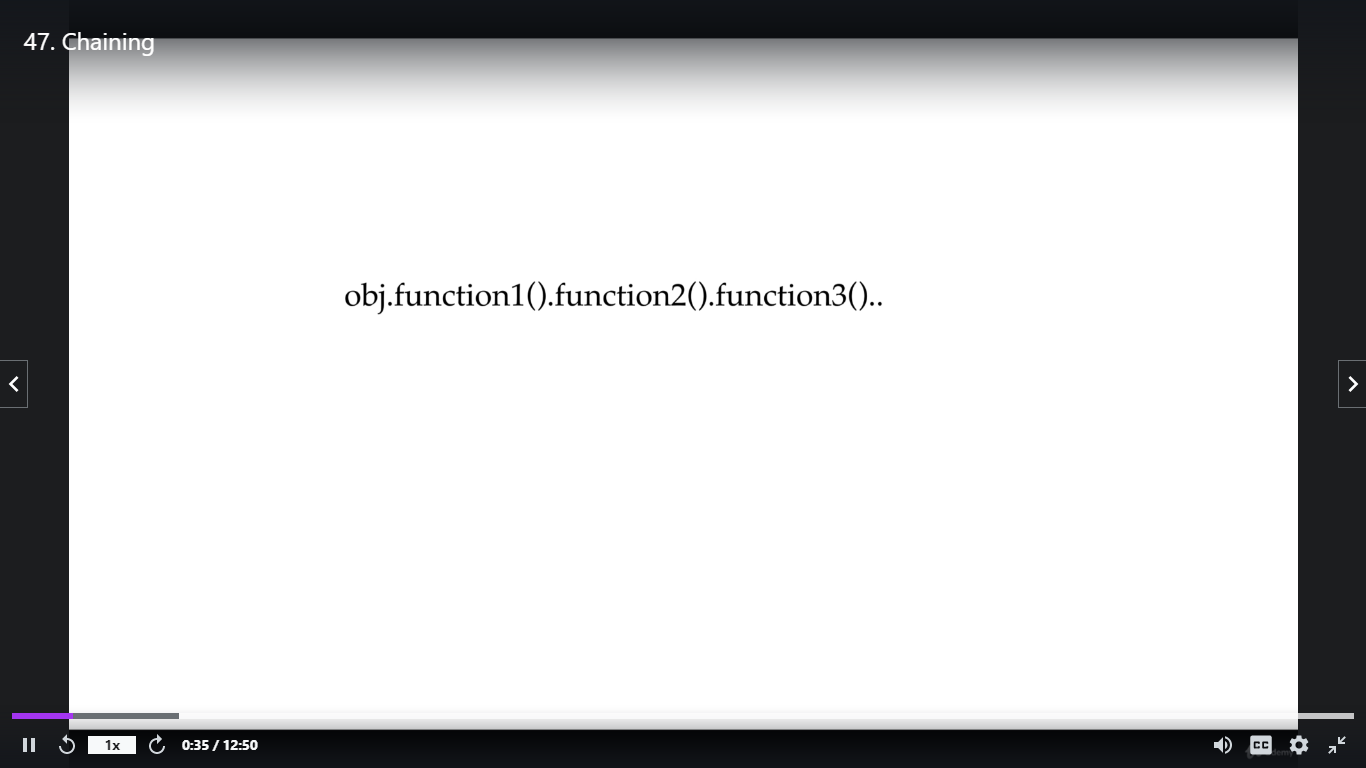


These are below, to write better functional code we need to understand these property

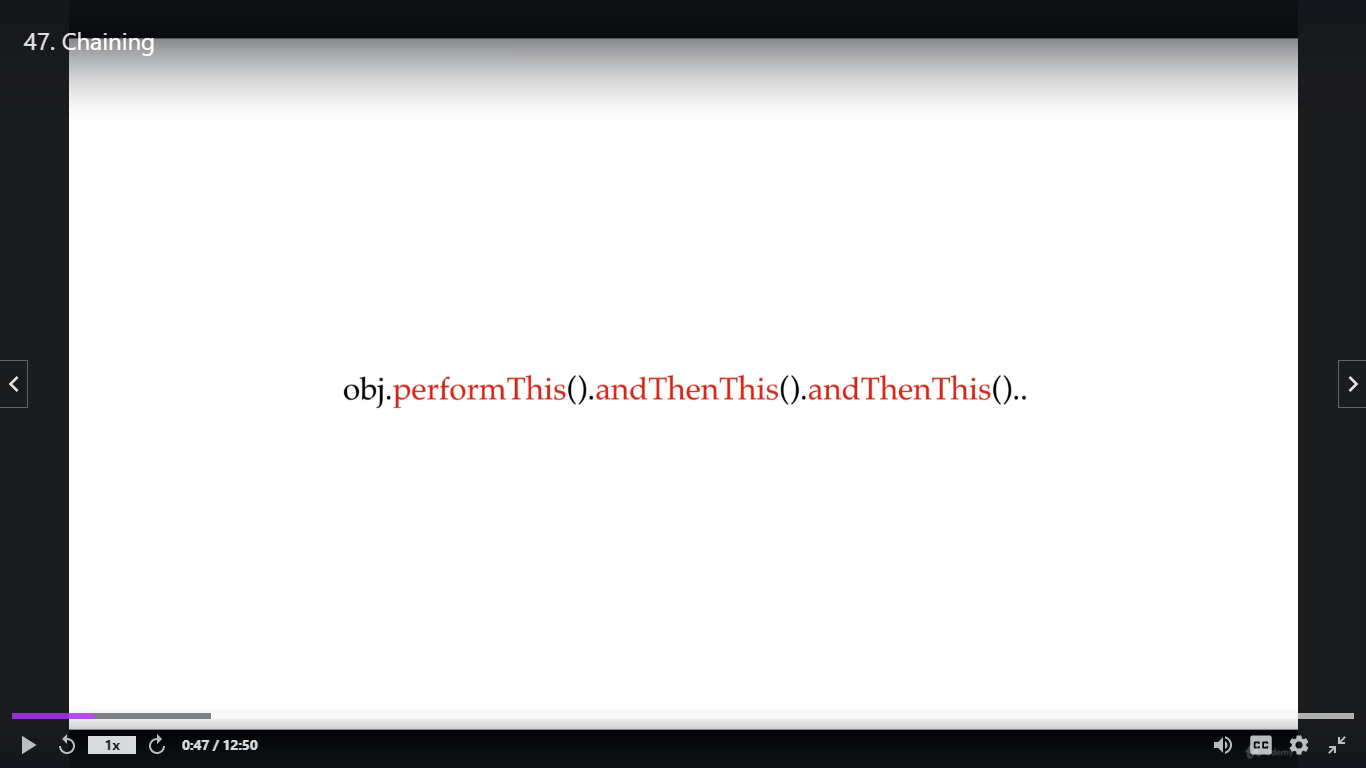


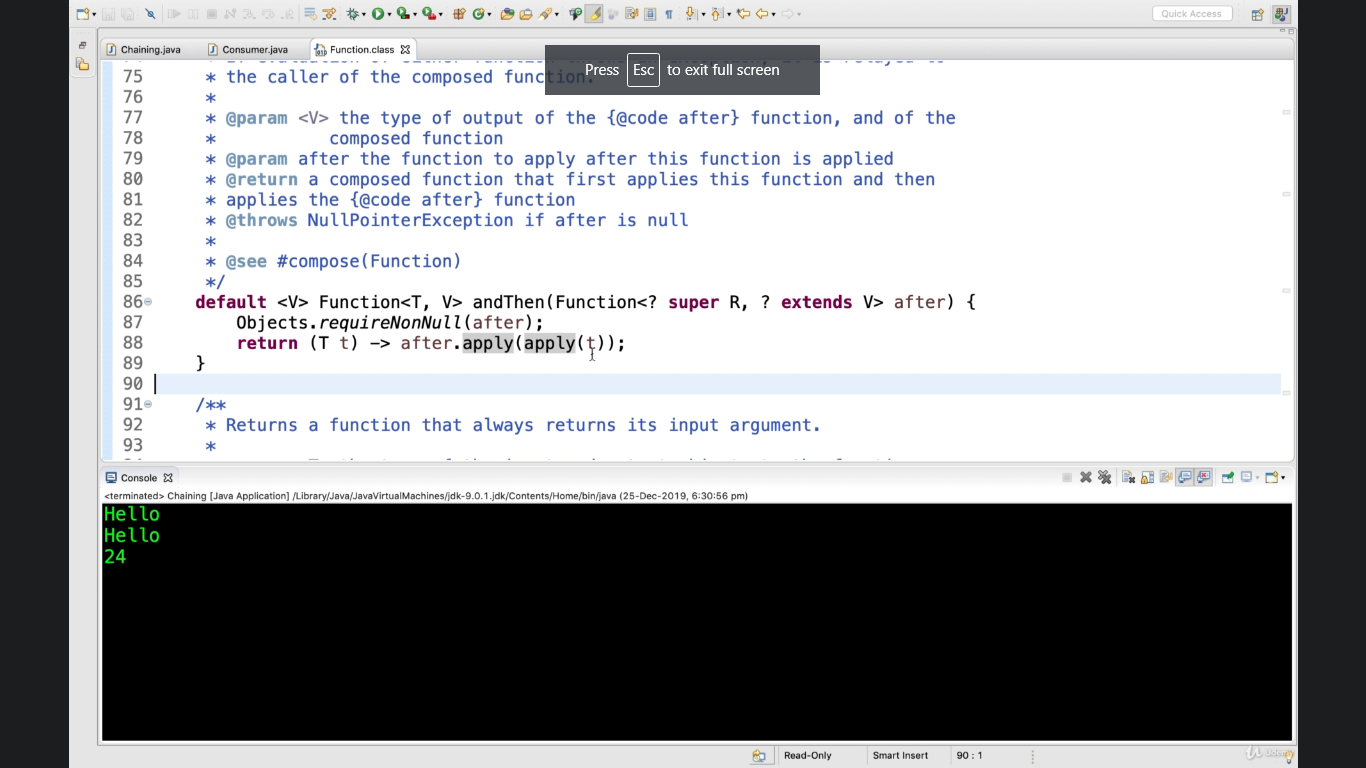


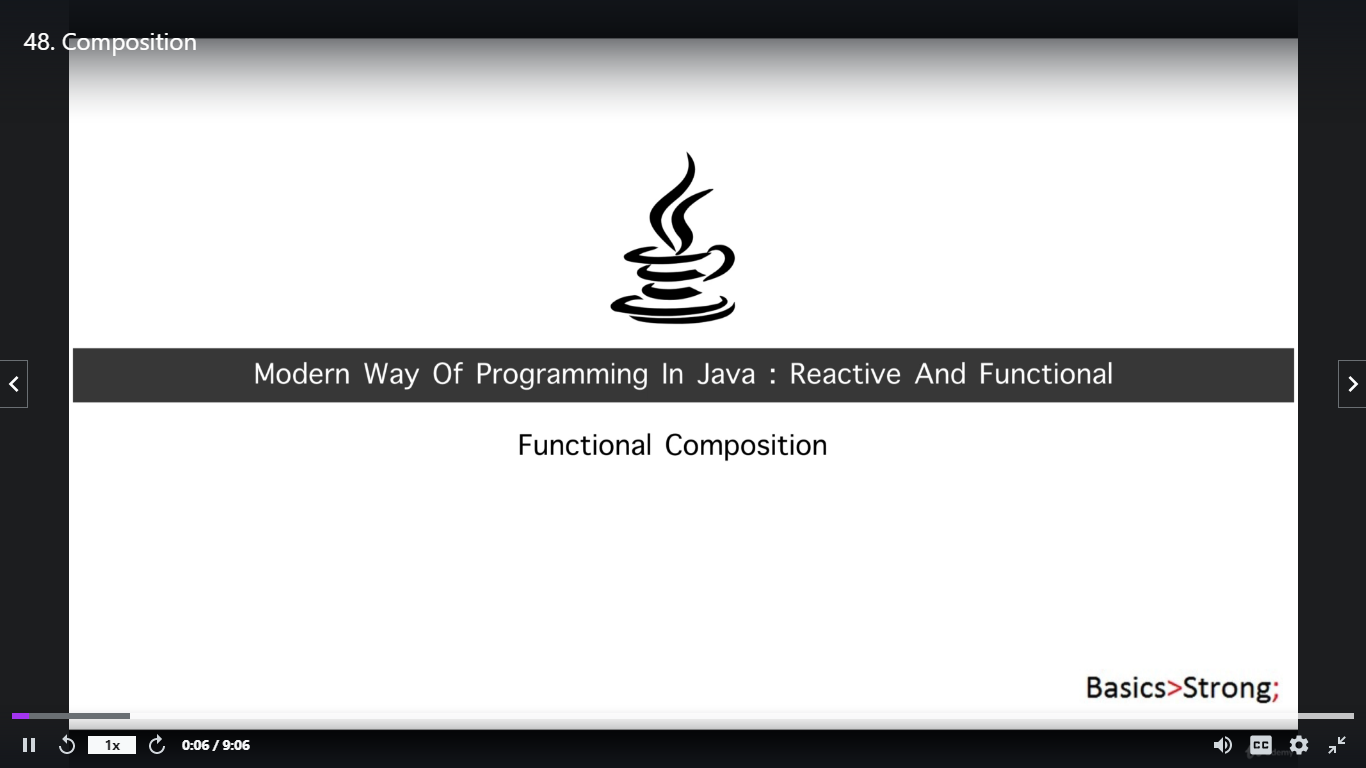
FC where we chain the function, each method returns object align the cost together in a single statement without requiring variables to store the intermediate results



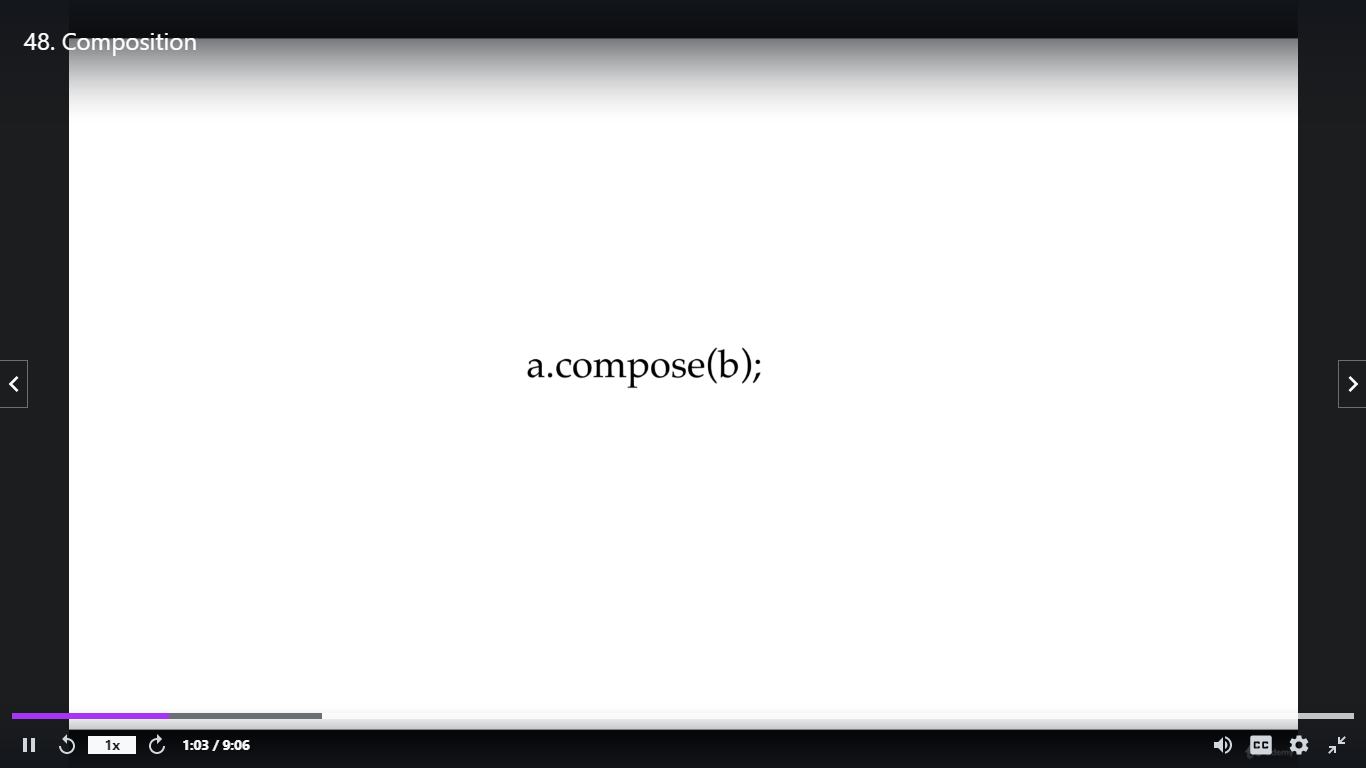
It is a technique to simplify the code where multiple functions are apply in a row one after another one by one in sequence



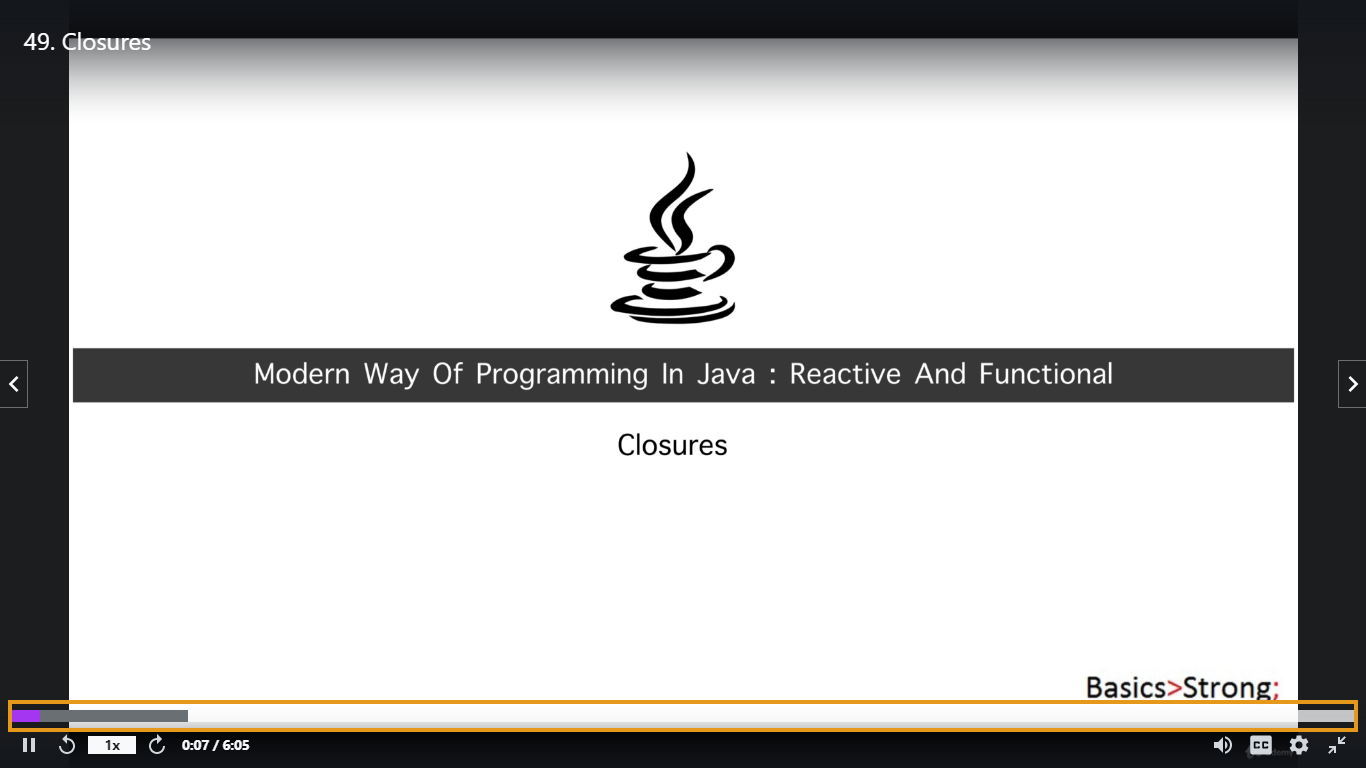


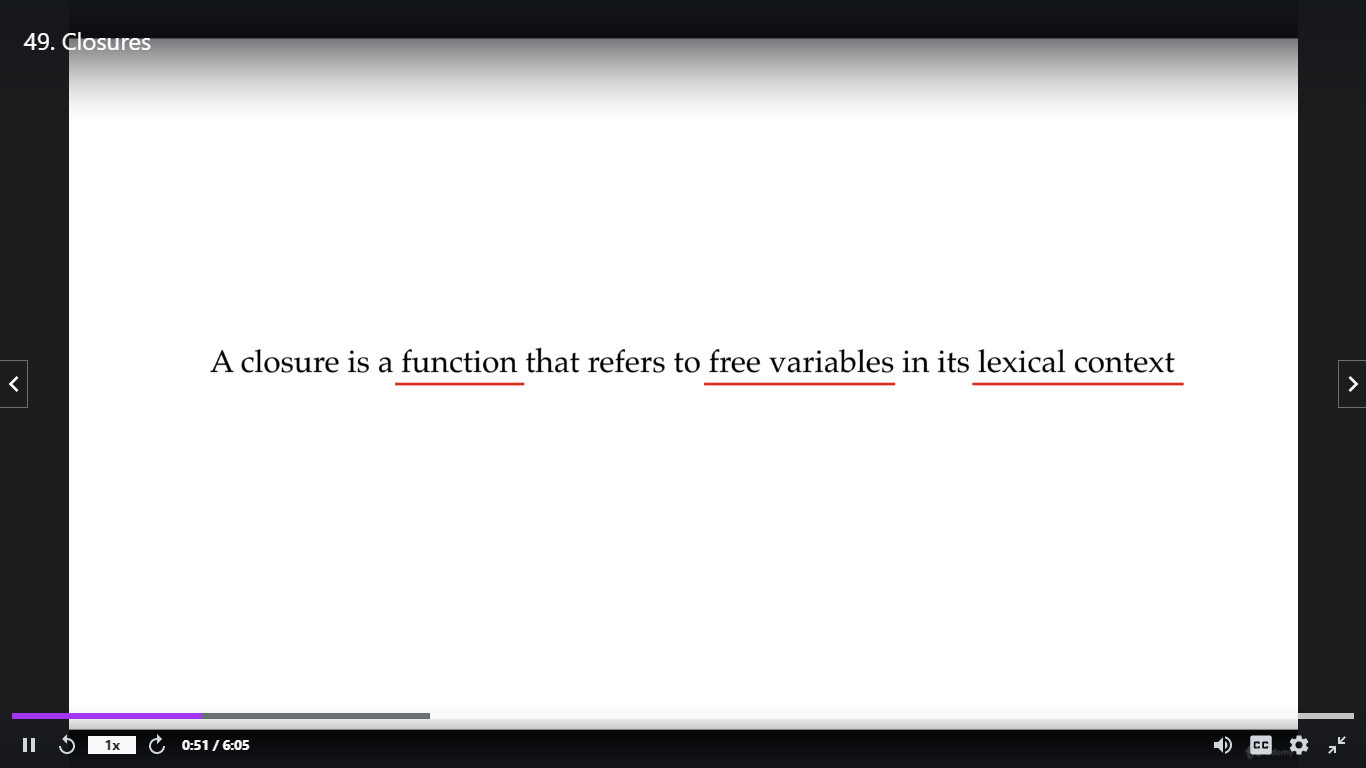


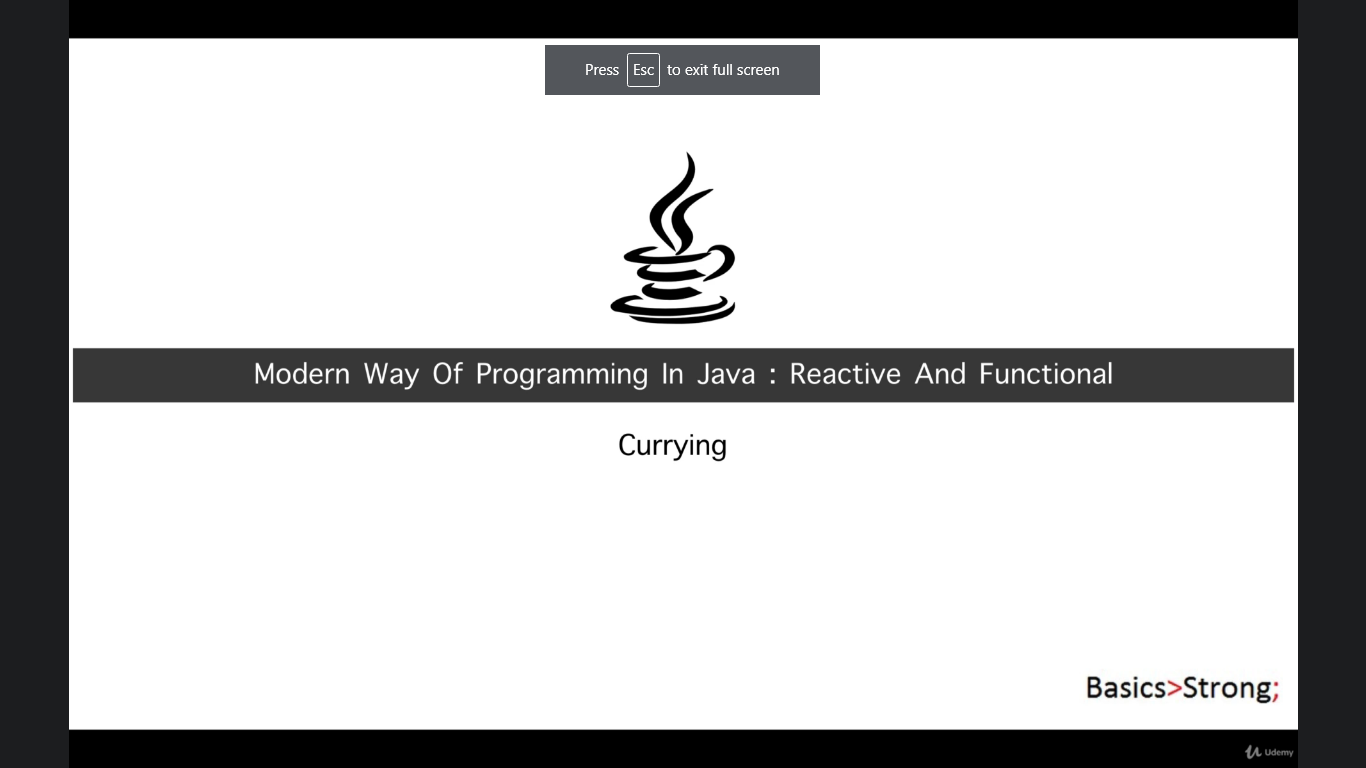
Composing function is different from Chaining function, composition follow reverse direction to the chaining



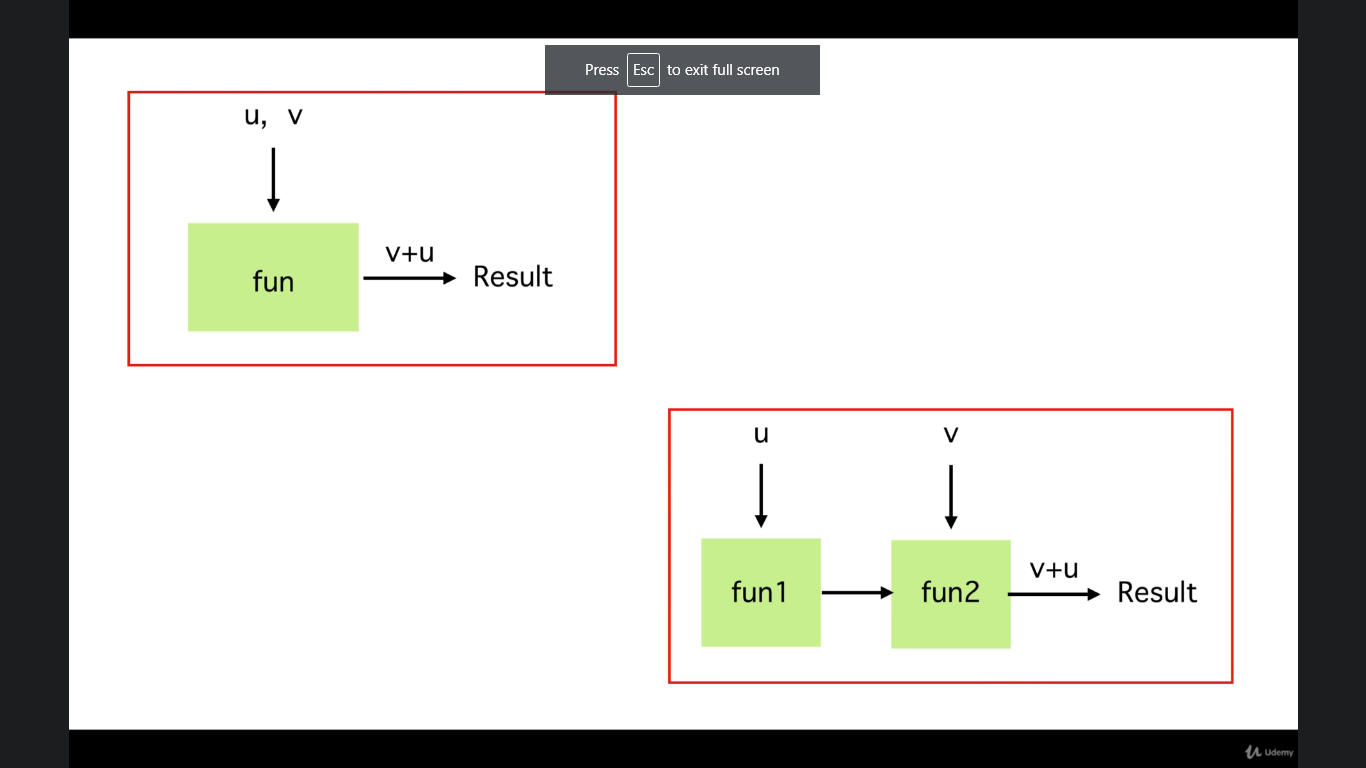
Here Function (b) get executed first then the function a. get executed on the result of a.

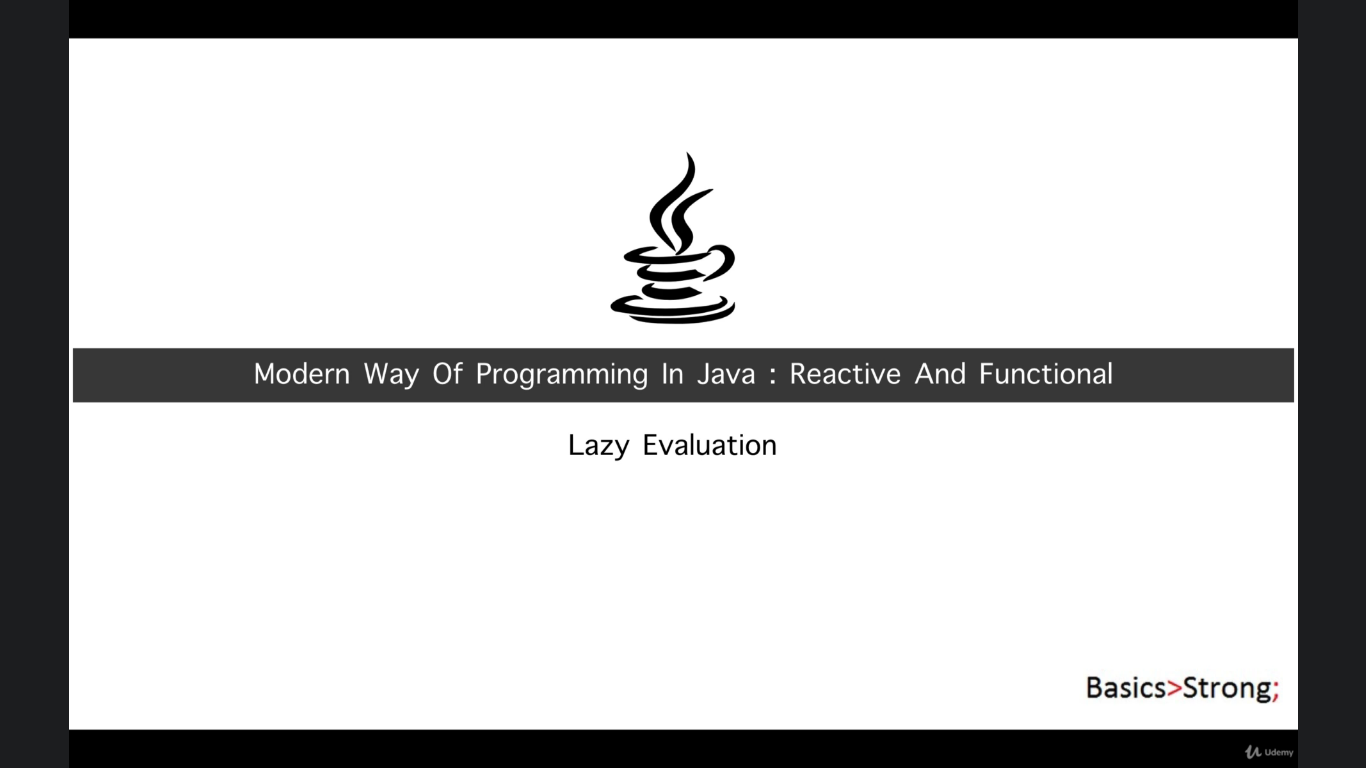


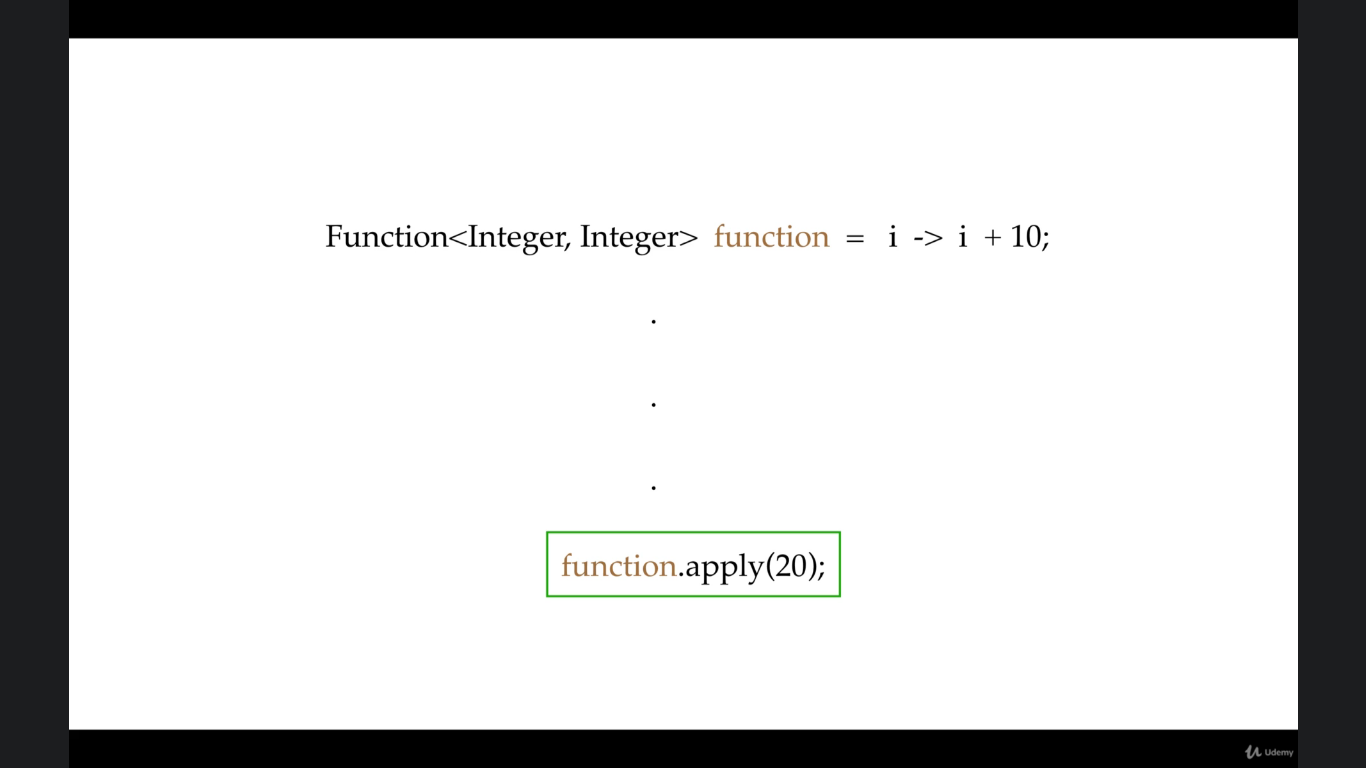




Currying Example, it is a technique basically restructure the multiple parameter function into multiple function having single parameter each





z

Whenever we write lambda it not get executed, it is just the implementation. It gets executed whenever we call it; So We can say lambda is lazy. Yes lazy evaluator

