Hello, Functional Programming

Let's discover FP with Typescript

What's functional programming ?

Programming paradigm that treats computation as mathematical function evaluation

What's functional programming ?

Avoids changing-state and mutability

What's functional programming ?

Mindset and coding style philosophy

What's functional programming ?

Sexy misunderstood trend

Benefits → Testability

You'll write pure functions that make code very easy to read and test

Benefits → Concurrency

You'll write stateless code that allow you app to support concurrent operation and scalability

Benefits → Caching

You'll write predictable function output that simplify caching strategies

Benefits → Reasoning

You'll write small function and declaratives API that simplify the way to

Why using Typescript ?

Born from many influences from both OOP and FP paradigms

A multi-paradigm language that allow compromise between productivity and formality

Module objectives

- Referential transparency
- Higher order functions
- Immutable data structures
- Lazy evaluation
- Explicit side effects
- Closures
- Composition
- Recursion
- Declarative
- Currying
- Lambda expressions
- Variadic functions

FP Jargon

Vocabulary that you need to learn to understand the paradigm

Pure functions

Returns a value that is computed using only the arguments passed to it

Avoids mutating its arguments or any other external variables

```
// Impure ❌
function isIndexPage() {
  return window.location.pathname === "/";
}
```

About pure function

```ts
// Pure ✅
function isIndexPage(pathname: string) {
  return pathname === "/";
}
```

About pure function

SIDE effects & Referential transparency

We can expect that the function is not going to interfere (via a state mutation) in of our application.

Higher-order functions

The ability to pass, return and assign a type is considered as being first-class citizen

In FP, functions are treated as first-class citizens, this allows us to create Higher-Order Functions.

Arity

The arity of a function or operation is the number of arguments or operands that the function takes.

FP techniques and patterns

Main functional programming used notions

Composition

Act or mechanism to combine simple functions to build more complicated ones

Result of each function is passed as argument of the next, and the last result is the result of the whole

Composition is the Essence of Programming !

Partial application

Allow us to generate new functions from using an existing function by providing only some of its argument

Function partial application help us to compose functions with different arity.

Pipe

Function or operator used to send the output of one function to another function for further processing

A pipeline consists of a chain, that the output of each function is the input of the next function in the pipe

Point-free style

Coding style in which function definitions do not identify the arguments (or "points") on which they operate.

The definitions merely compose other functions, among which are combinators that manipulate the arguments.

Pattern matching

Match a value (or an object) against some patterns to select a branch of the code.

In FP, pattern matching can be used for matching on standard primitive values such as String.

Laziness

Lazy evaluation, is a strategy which delays the evaluation of an expression until its value is needed.

A synonym can be "call-by-need"

Immutability

An immutable object is an object whose state cannot be modified after it is created.

Lenses

A way for get/set values in a data type.

Lenses are particularly useful when we need to update an immutable object:

Exercises

https://github.com/pu-erhh/fp-exercises

The End