

# PURITY, SIDE EFFECTS AND IDEMPOTENCE

An introduction to JavaScript concepts, with slight  
nuances of TypeScript and Functional  
Programming..

A hand in a white shirt cuff points to a specific location on a complex, multi-colored subway map. The map is filled with dense lines of various colors (blue, yellow, red, green) representing different transit lines. The background is dark and out of focus.

At the end of the talk, access will be shared:

- ▶ To the recording
- ▶ To the slides

BE WARY, TRAVELERS!!

# PRESENTATION

I'm Pepe, currently a front-end developer, working with React and TypeScript. And almost always tinkering with different technologies.



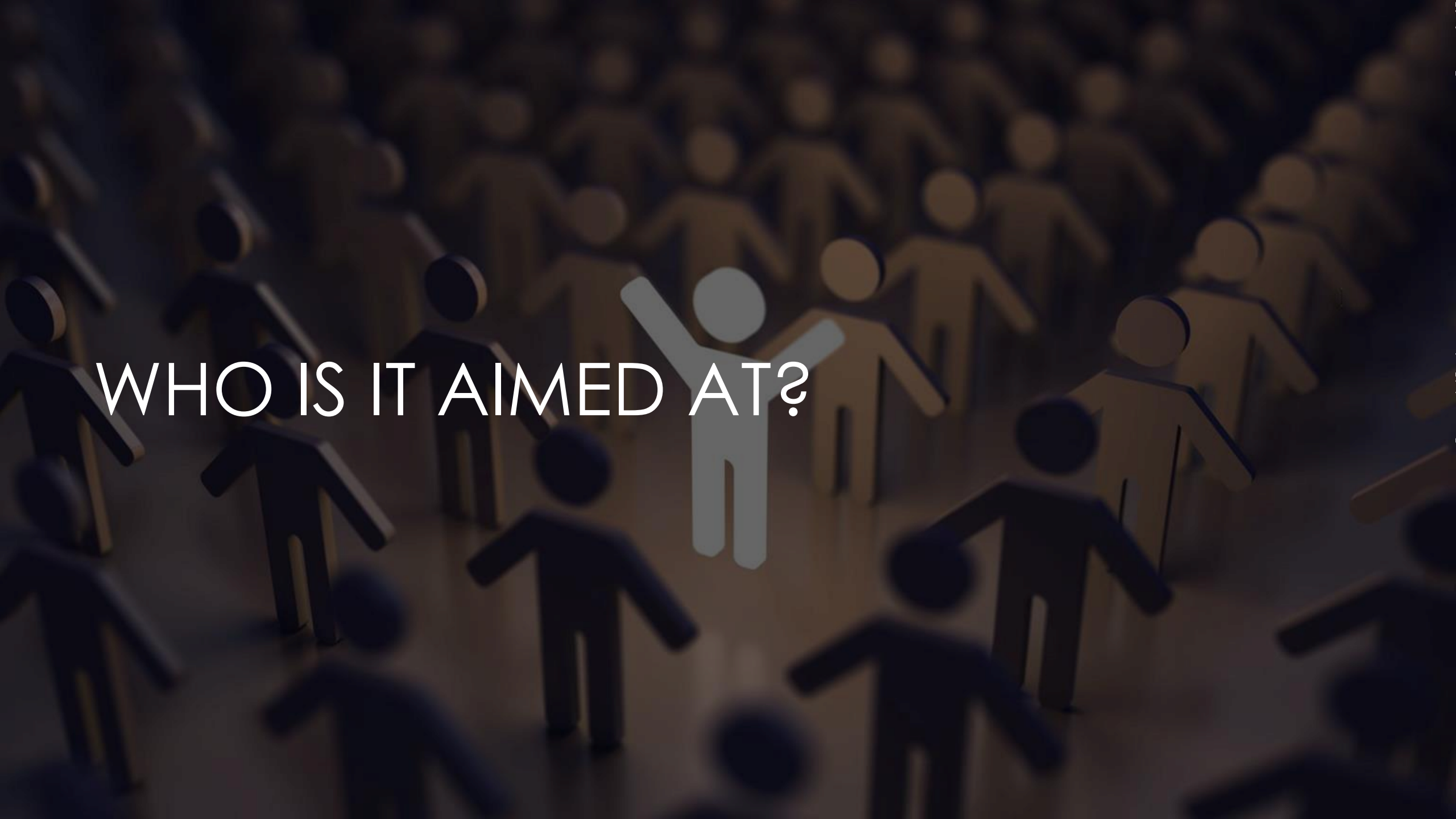
- 
- ▶ Introduction to some JavaScript concepts
  - ▶ Brief introduction to functional programming concepts
    - ▶ Purity of functions
    - ▶ Side effects of a function
  - ▶ Idempotence

# WHAT WILL WE SEE IN THIS TALK?



- ▶ They help us to better understand our work tool, JavaScript, Java, Python.
- ▶ They can guide us to a clearer and higher quality code.
- ▶ Some of these concepts are technical interview questions.
- ▶ They serve as a basis for applying "more complex" techniques.
- ▶ They are applicable to other languages: Java, C#, Python, etc.

## HOW USEFUL ARE THEY TO US?



WHO IS IT AIMED AT?

- ▶ Those who are just starting out
- ▶ Those who want to get into the world of JavaScript
- ▶ Those who want to get started in functional programming
- ▶ Those who want to discover something new today

AIMED AT...





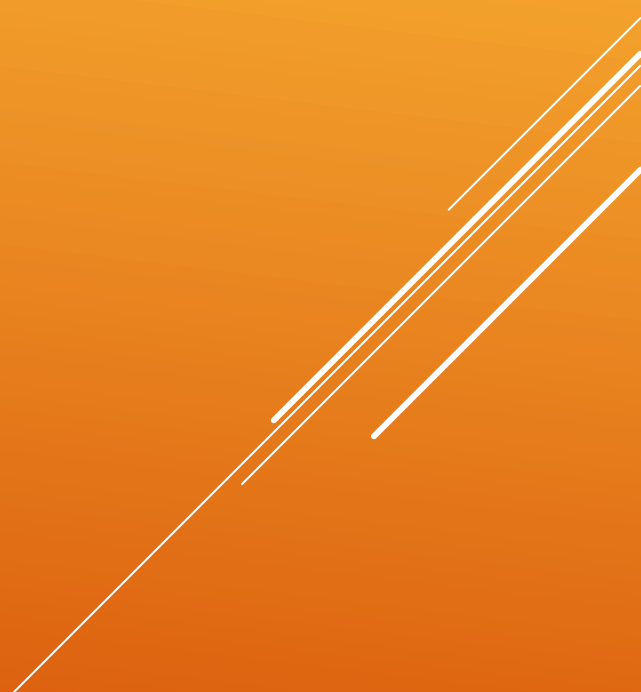
JavaScript is not required for this talk, but will be the language of choice for examples and concept grounding.

NOTICE



Functional programming is a programming paradigm.

# FUNCTIONAL PROGRAMMING



A programming paradigm is nothing more than a programming style, with advantages and disadvantages. OOP/POO is a well-known one.

## WHAT IS A PROGRAMMING PARADIGM?



The "meat", what is functional programming?

- ▶ Declarative programming
- ▶ Divide and conquer
- ▶ More mathematical functions
  - ▶ With deterministic results

# THE FUNCTIONAL PROGRAMMING CHECKLIST



PURITY





The purity of a function is inversely proportional to the amount of side effects it has.

Fewer side effects make a function purer.

# DEFINITION OF PURITY

Pure functions have as few side effects as possible.

Functional programming understands that side effects are sometimes necessary.

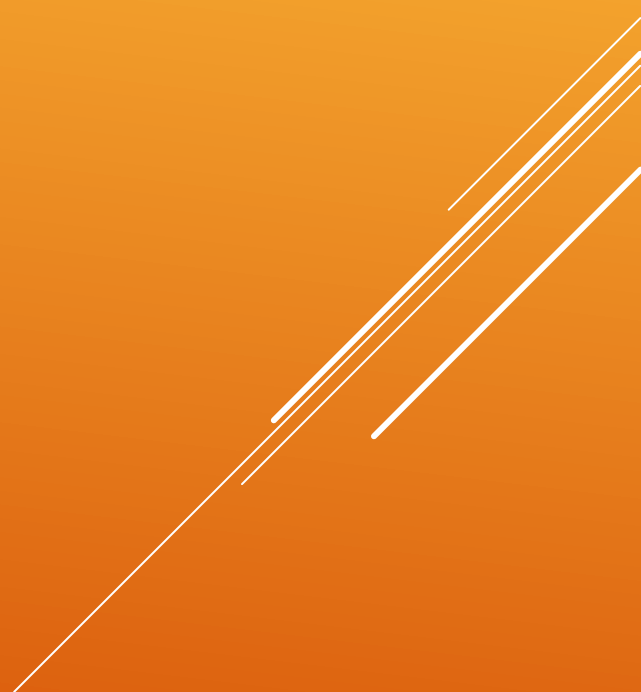
# PURITY AND SIDE EFFECTS

Several thin, parallel white lines of varying lengths and slopes are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.

A pure function has a **deterministic** result.

Its result is expected, i.e., you can determine its result if you know what arguments you are passing to it.

# PURITY AND DETERMINISM



A pure function is one with a deterministic outcome that has as few side effects as possible.

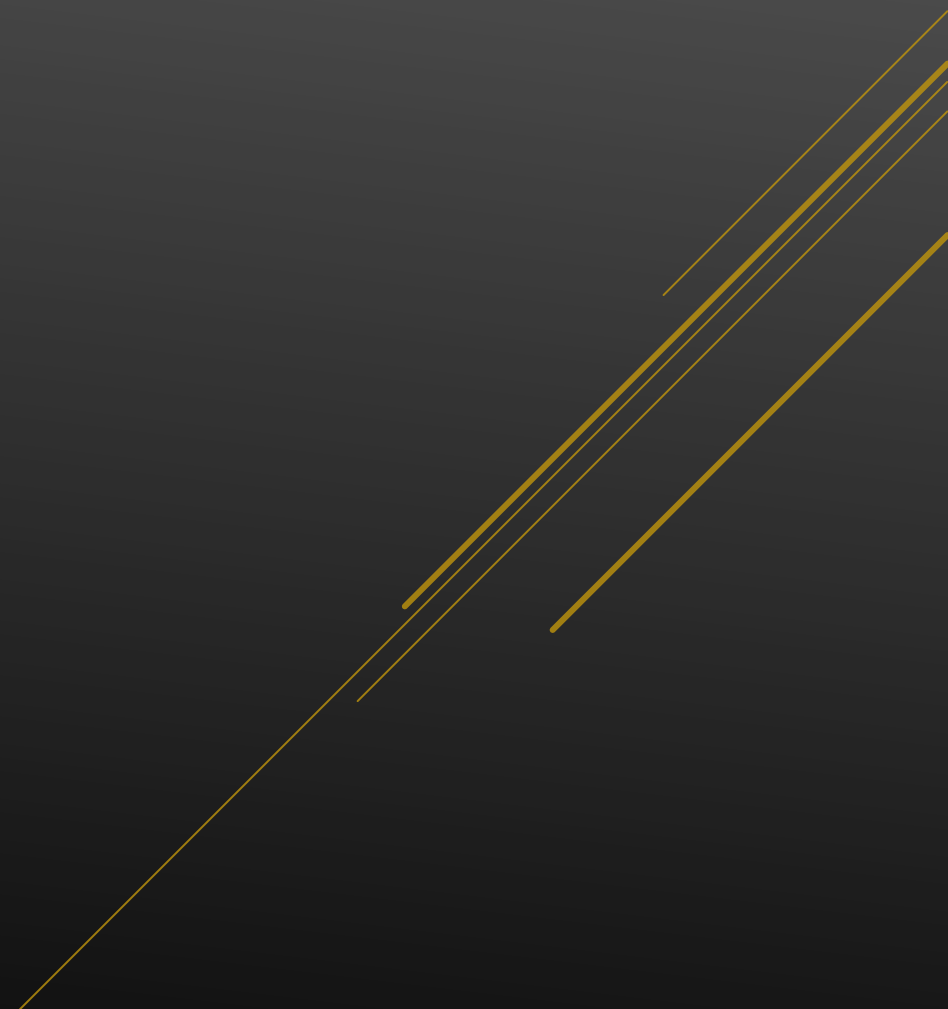
PURITY, UNIFYING CONCEPTS

A series of four parallel white lines of varying lengths, slanted diagonally upwards from left to right, located in the bottom right corner of the slide.



# BUT...

What is this side effect?





# SIDE EFFECTS



The side effects of a function can be understood as actions that go beyond the scope of a function.

## DEFINING SIDE EFFECTS

- ▶ Depends on a state(value, attribute, property, constant) not provided as a parameter.
- ▶ Mutate (modify) a non-local state, outside the body of the function.

If it fulfills one of these properties, the action (instruction) becomes a side effect

## THE CHECKLIST OF A SIDE EFFECT



But then, don't we have to have them? Yes, you have to have side effects, what you have to do is to minimize their use.

A side effect "**damages**" the **trace**, it is an **extra mock** within a test, it makes the result of a function somewhat **more unpredictable**.

## UNDERSTANDING SIDE EFFECTS

With examples, everything is better understood.

Let's say that we have retrieved information about a user from the DB, but we don't want to return all the information, for that, we have a function that filters the body of the answer.

## EXAMPLES

A side effect would be to retrieve the user information in the same function that does the filtering.

Or store the retrieved user information in a global variable and access it to filter the fields to return.

In TypeScript, ***formatCurrentUser(): ResponseUser.***

In Java, ***ResponseUser formatCurrentUser().***

# IT IS A SIDE EFFECT WHEN...

Following the example from before, it would not be a side effect if our function receives the user's information and returns the response.

In TypeScript, ***formatUser(user: User): ResponseUser***

In Java, ***ResponseUser formatUser(User user user)***

IT WOULD NOT BE A SIDE EFFECT IF...



In case of being in the frontend, this same case can occur.

We have retrieved the user information from the appropriate endpoint. Assuming that our app is not multilingual.

We would not retrieve the user directly from a store, we would have a function that would be in charge of formatting the user information, ***formatUser(user: User): string***

## CONTEXTUALIZING IN JAVASCRIPT

$y = g(x)$

Secant Lines

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$
$$f(x) = \lim_{h \rightarrow 0} \frac{(x+h)^2 - x^2}{h}$$
$$= \lim_{h \rightarrow 0} \frac{x^2 + 2xh + h^2 - x^2}{h}$$
$$= \lim_{h \rightarrow 0} \frac{2xh + h^2}{h}$$
$$= \lim_{h \rightarrow 0} h(2x + h)$$

$g(x+h) - g(x)$

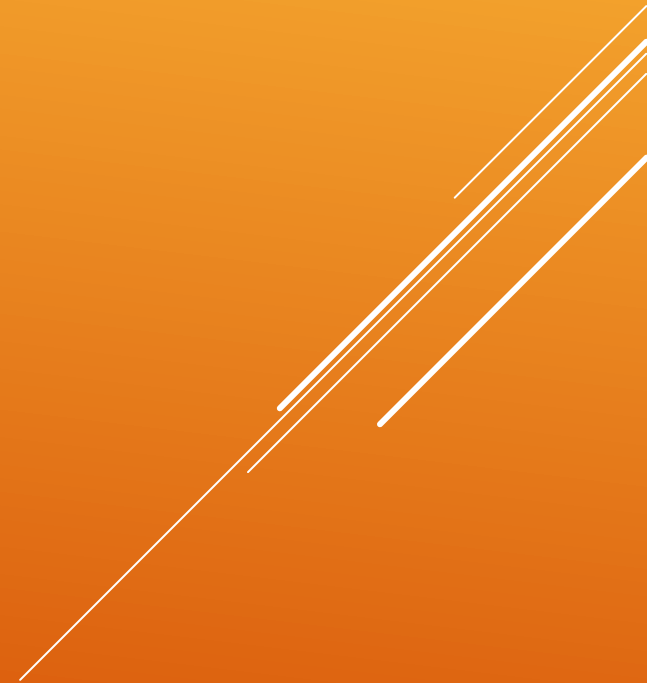
$-x+h$

# IDEMPOTENCE

Idempotence is the property of a mathematical function to be completely pure.

That is, it has no side effects, and its result will always be the same, given that we pass it the same arguments.

# WHAT IS IDEMPOTENCE

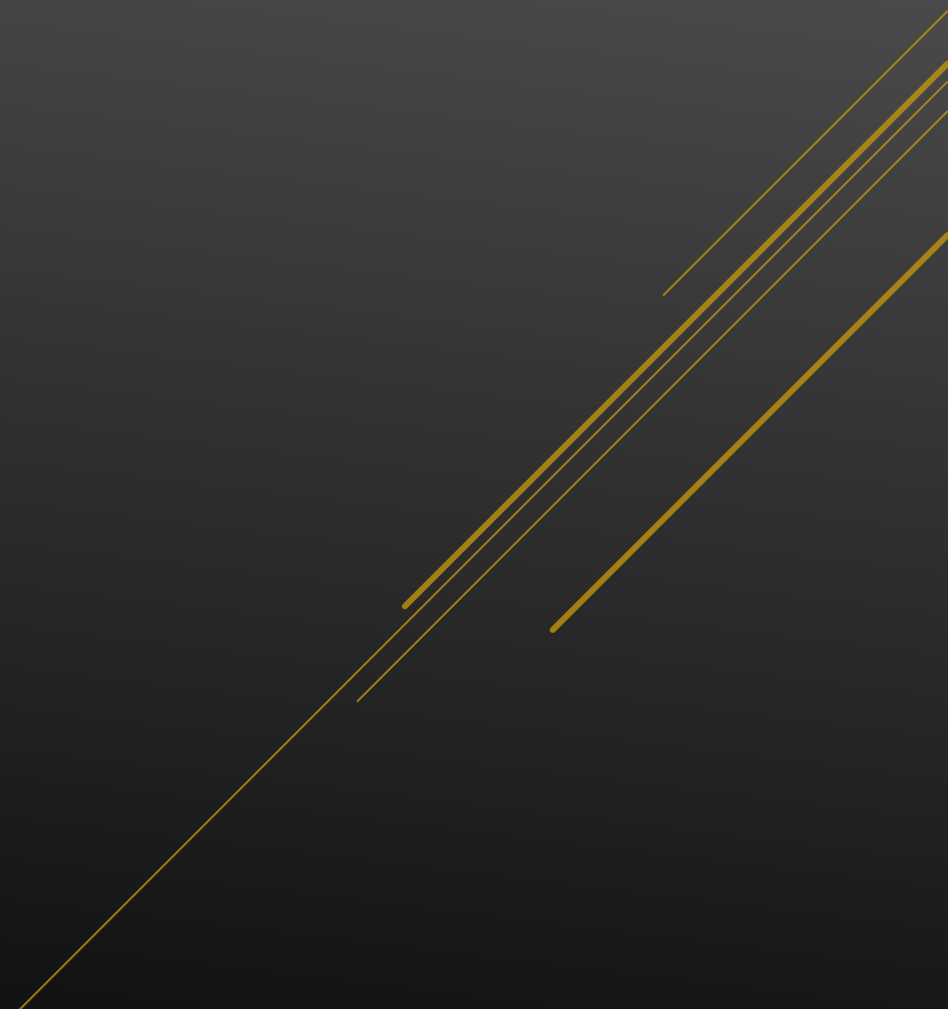


- ▶ Purity of functions (traceability)
- ▶ Possibility to save computations
  - ▶ The same result with the same arguments could be cached...
- ▶ Saying a not so well known word for something that is commonly known

## BENEFITS OF IDEMPOTENCE

# “TRADE-OFF” OF IDEMPOTENCE.

Not everything can be, nor should be,  
idempotent.



- ▶ **Sum**

- ▶  $1 + 1$  sometimes outputs 7, but  $2 + 2$  will always outputs 4

- ▶ **Factorial**

- ▶ The factorial of a number will always be the same,  $\text{factorial}(3) == 6$

# EXAMPLES OF IDEMPOTENCE



- ▶ **Math.random**

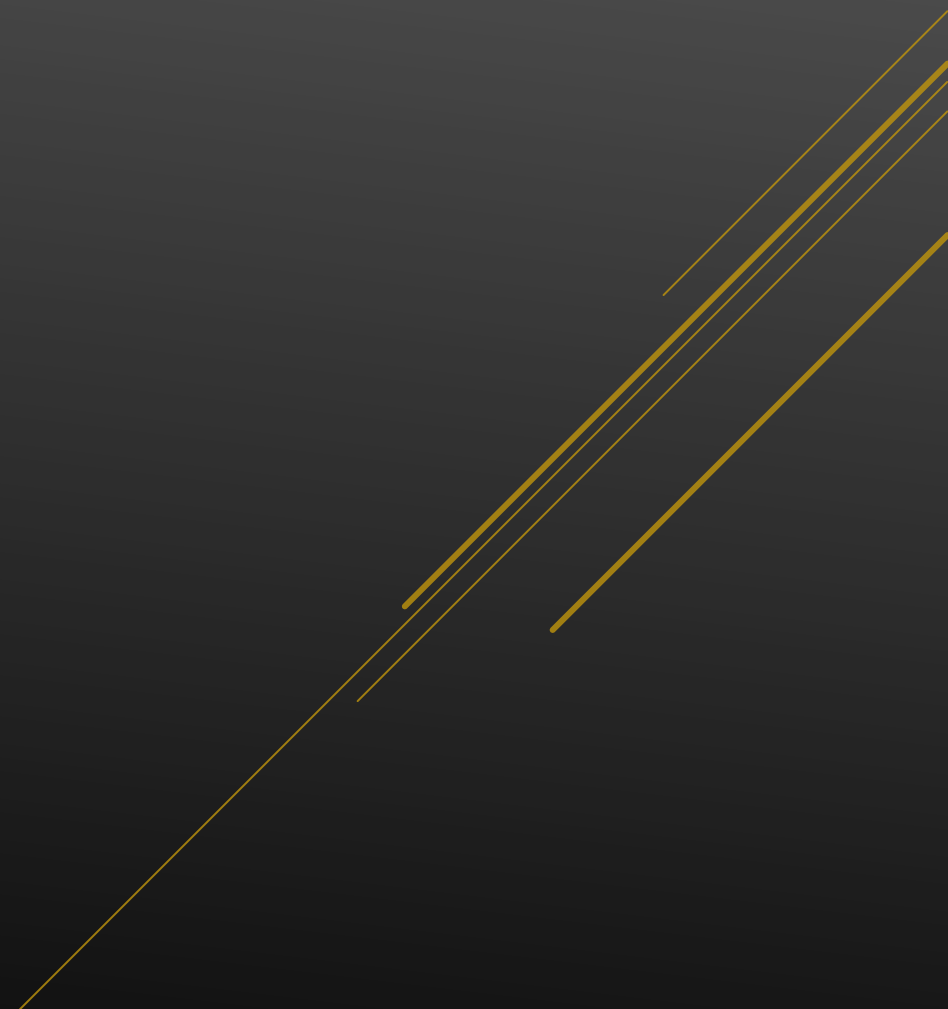
- ▶ It will almost always return a different number

- ▶ **Time.now** or **Date.now**

- ▶ Time is constantly changing

EXAMPLES THAT ARE **NOT** IDEMPOTENT

LET'S RECAP...



## ► **Functional Programming**

- A divide and conquer programming paradigm

## ► **Purity**

- A function with the least amount of side effects and a deterministic result.

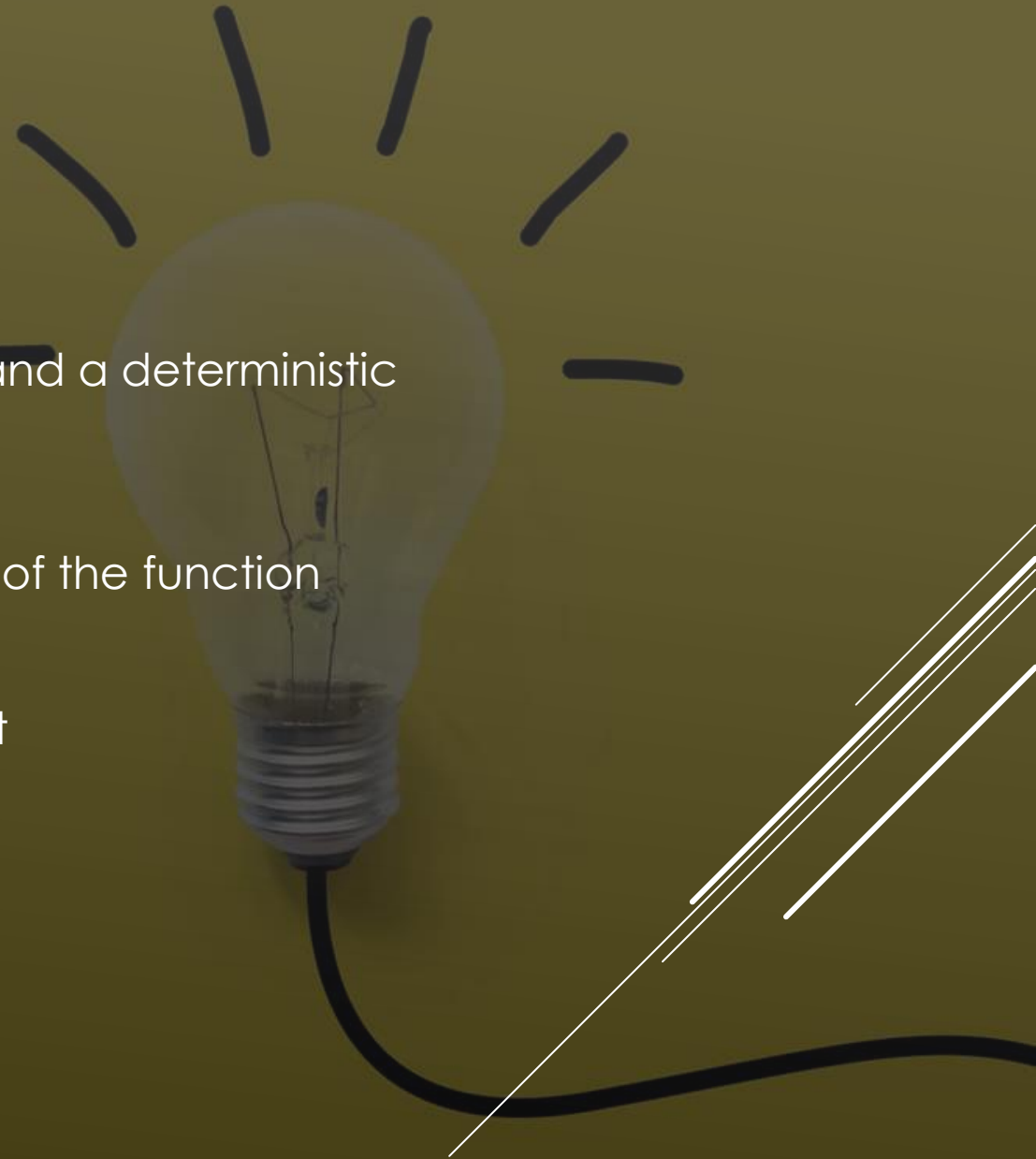
## ► **Side effects**


- Mutations and dependencies outside the scope of the function

## ► **Idempotence**

- A pure function, same parameters == same result

WE HAVE SEEN...



- 
- An aerial photograph of a winding asphalt road through rolling green hills. A small car is visible on the road, navigating a sharp curve. The landscape is lush and green, with some darker patches of vegetation. The overall tone is slightly muted, giving it a professional or artistic feel.
- ▶ Write clearer and higher quality code
  - ▶ Understand the ecosystem better
  - ▶ Conduct good technical interviews
  - ▶ New ways to face the same old problems

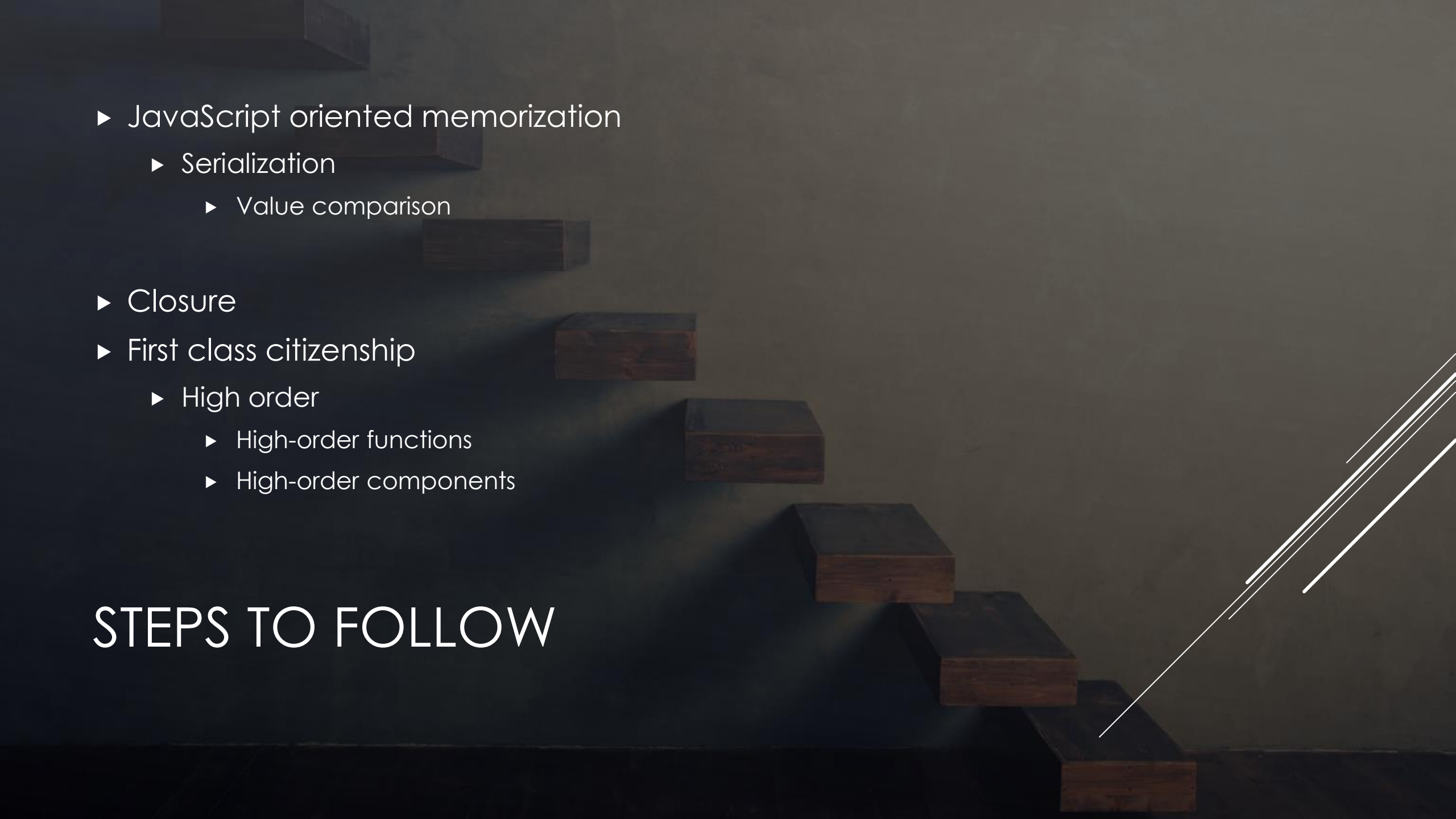
WHICH ALLOWS US TO...

Three white diagonal lines of varying lengths, positioned in the bottom right corner of the slide, pointing towards the bottom right.

# AND NOW... WHERE COULD I CONTINUE?

In the following episodes...



- 
- ▶ JavaScript oriented memorization
    - ▶ Serialization
      - ▶ Value comparison
  - ▶ Closure
  - ▶ First class citizenship
    - ▶ High order
      - ▶ High-order functions
      - ▶ High-order components

# STEPS TO FOLLOW

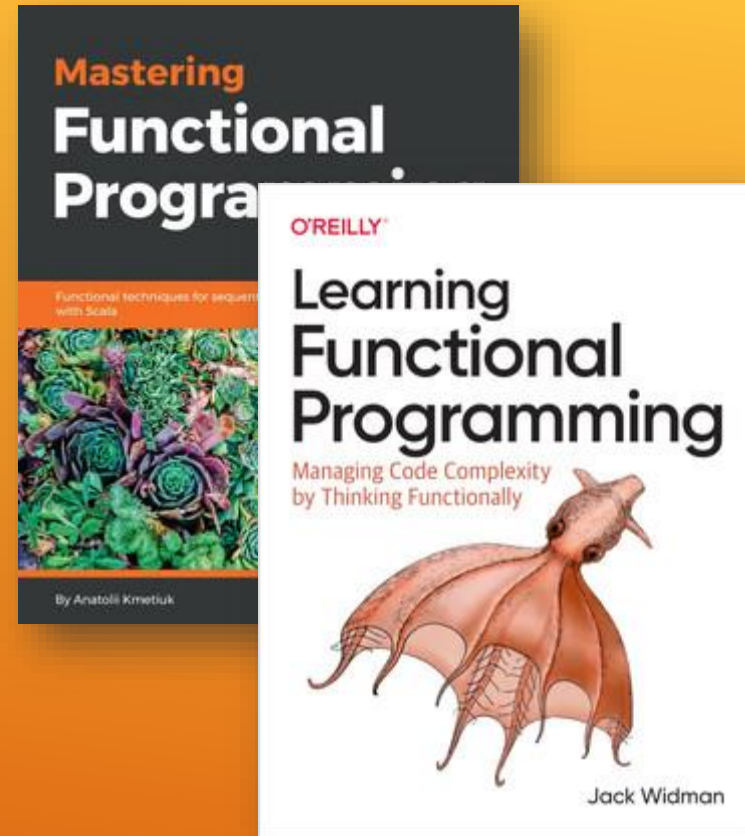


Learning Functional Programming  
by Jack Widman, O'Reilly

O'Reilly:  
<https://www.oreilly.com/library/view/learning-functional-programming/9781098111748/>  
Amazon: <https://amzn.eu/d/0CVqCi6>

Mastering Functional Programming  
by Anatolii Kmetiuk, Packt Publishing

Packt:  
<https://www.packtpub.com/product/mastering-functional-programming/9781788620796>  
Amazon: <https://a.co/d/dBo8L3m>



# BOOKS



Anjana Vakil: Learning Functional Programming with JavaScript - JSUnconf 2016



Why Isn't Functional Programming the Norm?  
– Richard Feldman

## TALKS



Functional Programming for Pragmatists • Richard Feldman  
• GOTO 2021

You can read the article at

➤ <https://medium.com/@jofaval/a60130f073ef>

BEWARE!!! Contains spoilers for possible next talks, but the order is the same.

# ARTICLE

Have a nice day!

THANK YOU FOR YOUR ATTENTION

Several thin, white, parallel diagonal lines are positioned in the bottom right corner of the slide, extending from the right edge towards the center.