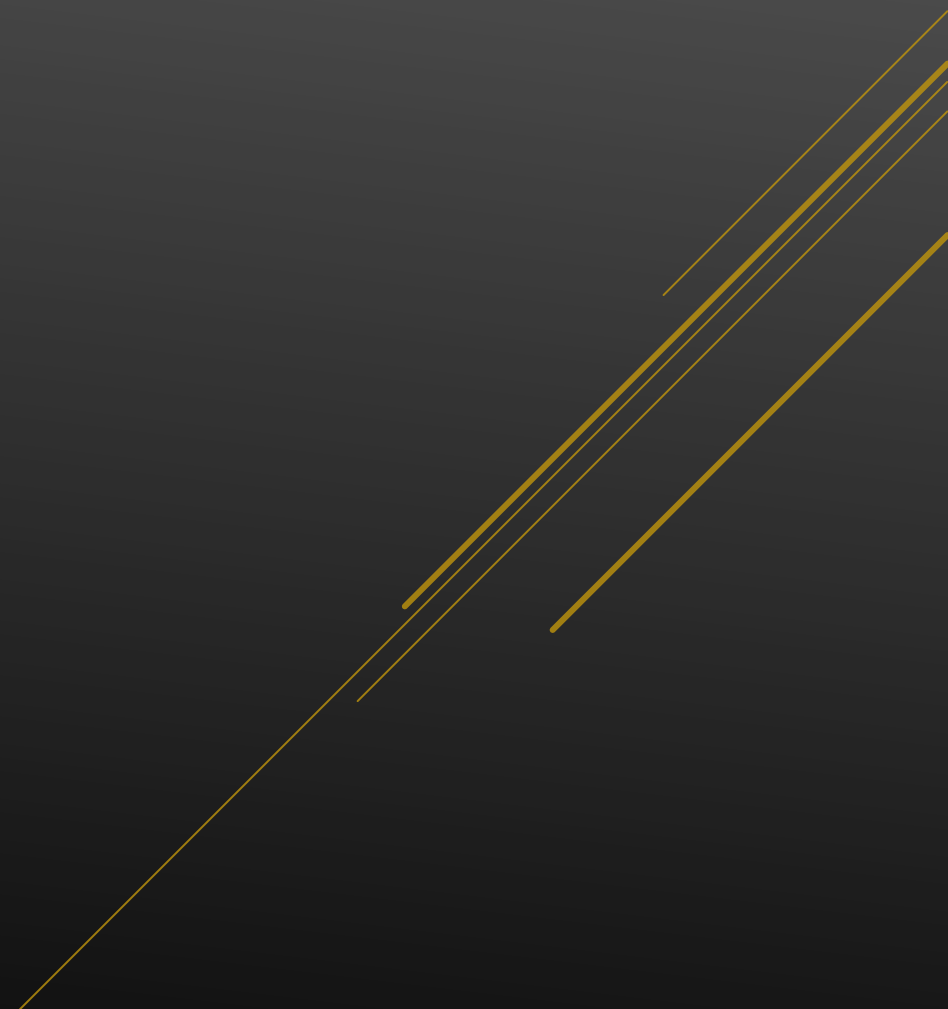


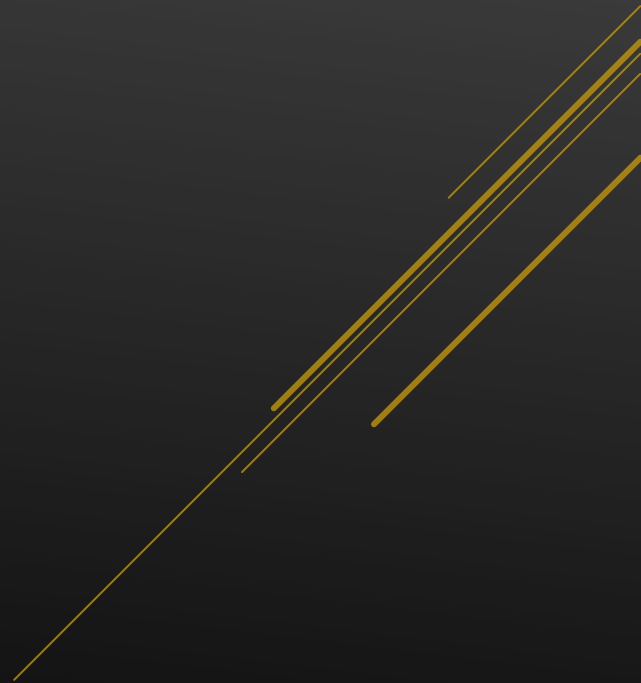
# PUREZA, EFECTOS SECUNDARIOS E IDEMPOTENCIA

Una introducción a conceptos de JavaScript,  
con ligeros matices de TypeScript y  
Programación Funcional.





# PRESENTACIÓN

Soy Pepe, actualmente desarrollador de front-end, trabajando con React y TypeScript. Y casi siempre trasteando con distintas tecnologías.



# ¿QUÉ VEREMOS EN ESTA CHARLA?

- ▶ Introducción a algunos conceptos de JavaScript
  - ▶ Breve introducción a conceptos de Programación funcional
    - ▶ Pureza de las funciones
    - ▶ Efectos secundarios de una función
  - ▶ Idempotencia
- 
- A series of parallel yellow lines of varying lengths and orientations, located in the bottom right corner of the slide, creating a modern, abstract graphic element.

- 
- ▶ Nos ayudan a entender mejor nuestra herramienta de trabajo, JavaScript, Java, Python
  - ▶ Pueden orientarnos a un código de más claro y de mayor calidad
  - ▶ Algunos de estos conceptos son preguntas de entrevistas técnicas
  - ▶ Sirven de base para aplicar técnicas “más complejas”
  - ▶ Son aplicables a otros lenguajes: Java, C#, Python, etc.

## ¿QUÉ UTILIDAD NOS APORTAN?



¿A QUIÉN ESTÁ  
ORIENTADO?

- ▶ Quienes estén empezando
- ▶ Quienes que quieran adentrarse en el mundo de JavaScript
- ▶ Quienes quieran empezar a adentrarse en la programación funcional
- ▶ Quienes quieran descubrir algo nuevo hoy

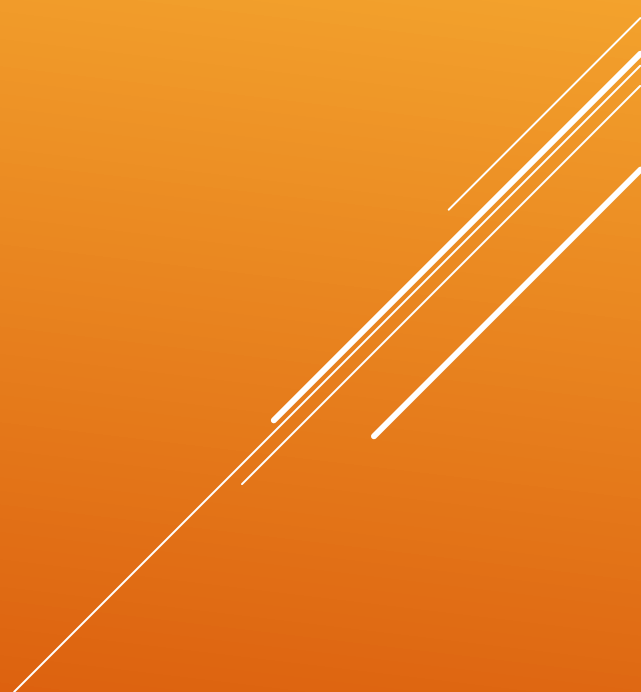
ORIENTADO A...





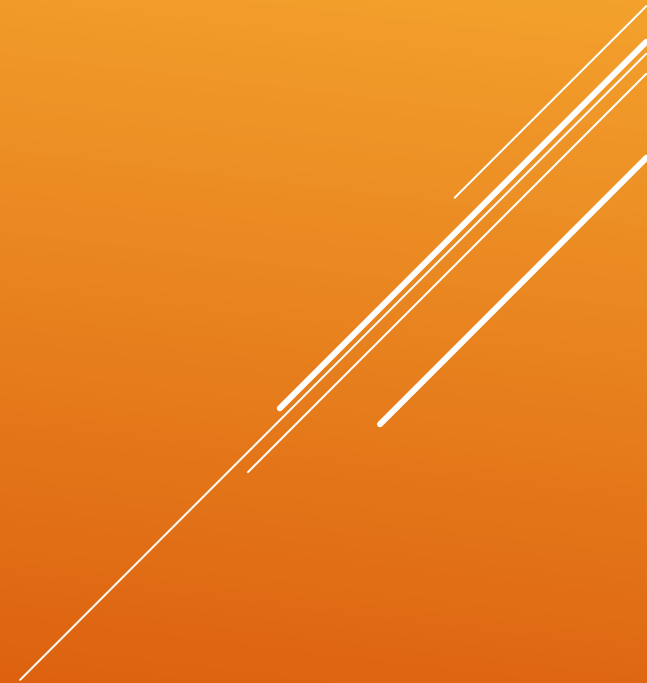
No se requiere JavaScript para esta charla, pero será el lenguaje de elección para los ejemplos y toma de tierra de conceptos.

# AVISO



La programación funcional es un paradigma de programación.

# PROGRAMACIÓN FUNCIONAL





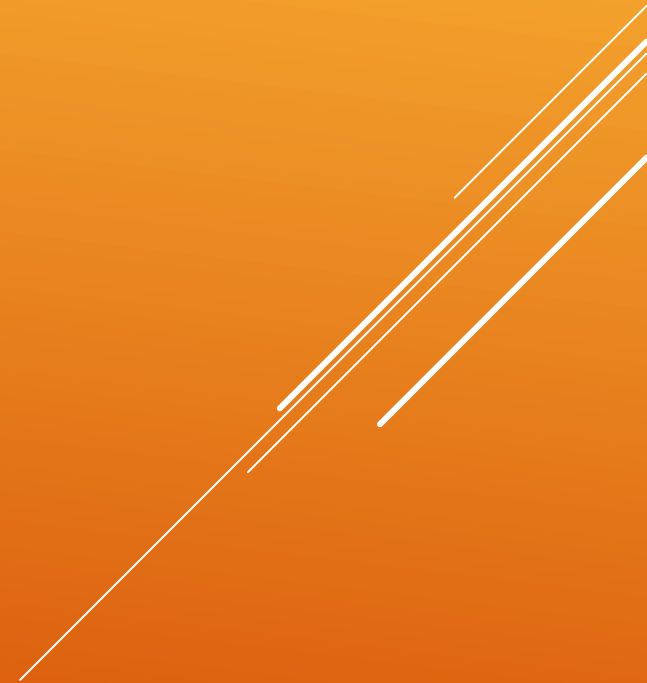
Un paradigma de programación no es más que un estilo de programación, con ventajas e inconvenientes. OOP/POO es uno muy conocido.

¿QUÉ ES UN PARADIGMA DE PROGRAMACIÓN?

La “chicha”, ¿qué es la programación funcional?

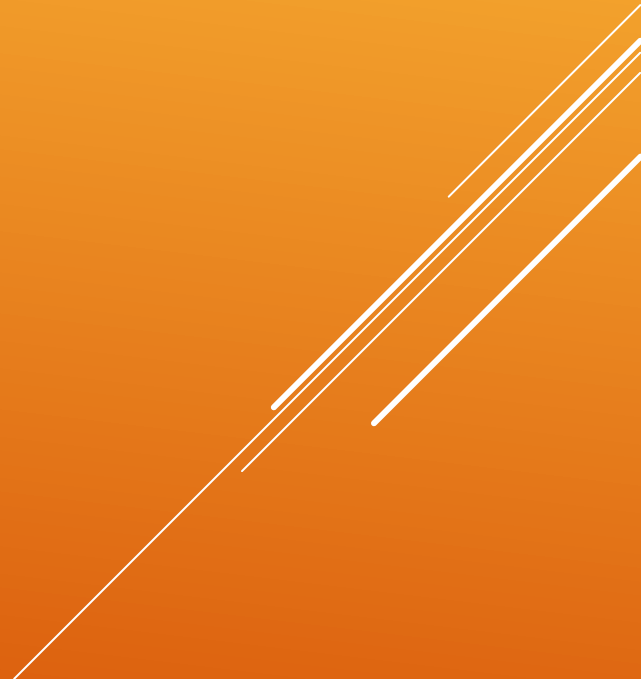
- ▶ Programación declarativa
- ▶ Divide y vencerás
- ▶ Funciones más matemáticas
  - ▶ Con resultados determinísticos

# LA CHECKLIST DE LA PROGRAMACIÓN FUNCIONAL





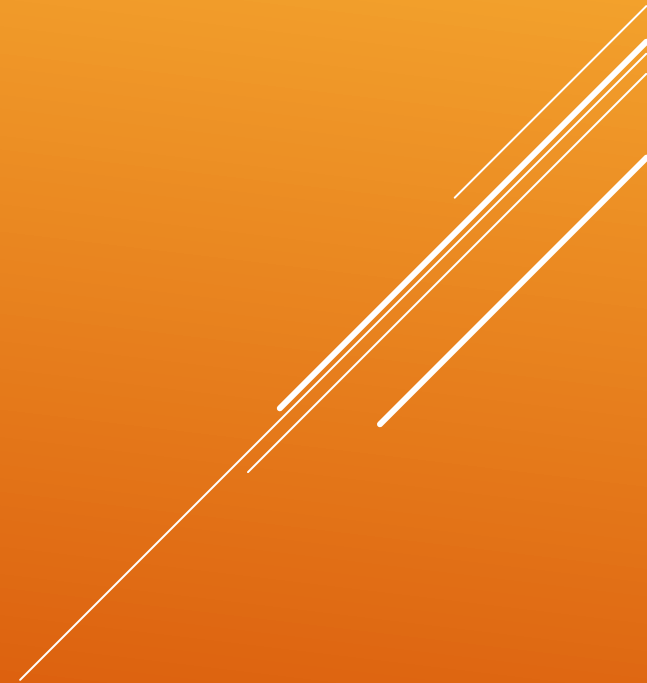
PUREZA



La pureza (pure, pureness) de una función es inversamente proporcional a la cantidad de efectos secundarios que tiene.

Menos efectos secundarios hacen de una función que sea más pura

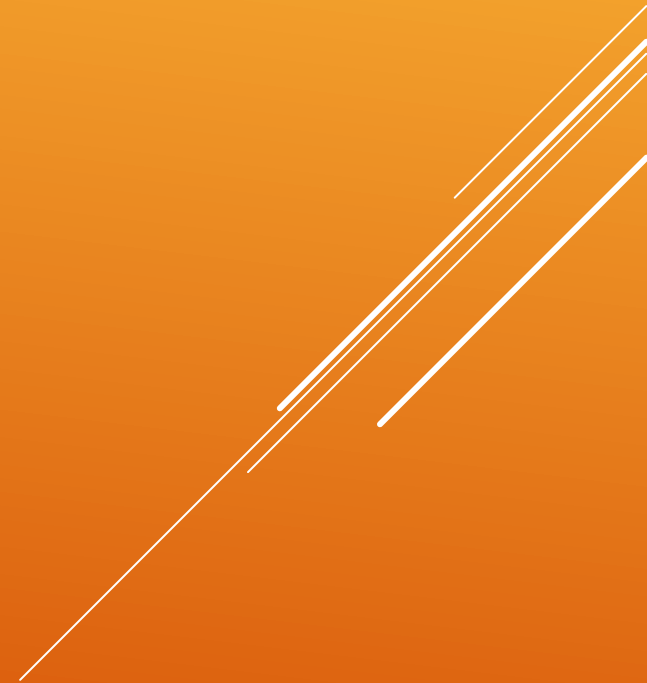
## DEFINICIÓN DE PUREZA



Las funciones puras tienen la menor cantidad posible de efectos secundarios.

La programación funcional entiende que los efectos secundarios a veces son necesarios

# PUREZA Y EFECTOS SECUNDARIOS



Una función pura tiene un resultado **determinístico**.

Su resultado es esperable, es decir, puedes determinar su resultado si sabes qué argumentos le estás pasando

# PUREZA Y EL DETERMINISMO



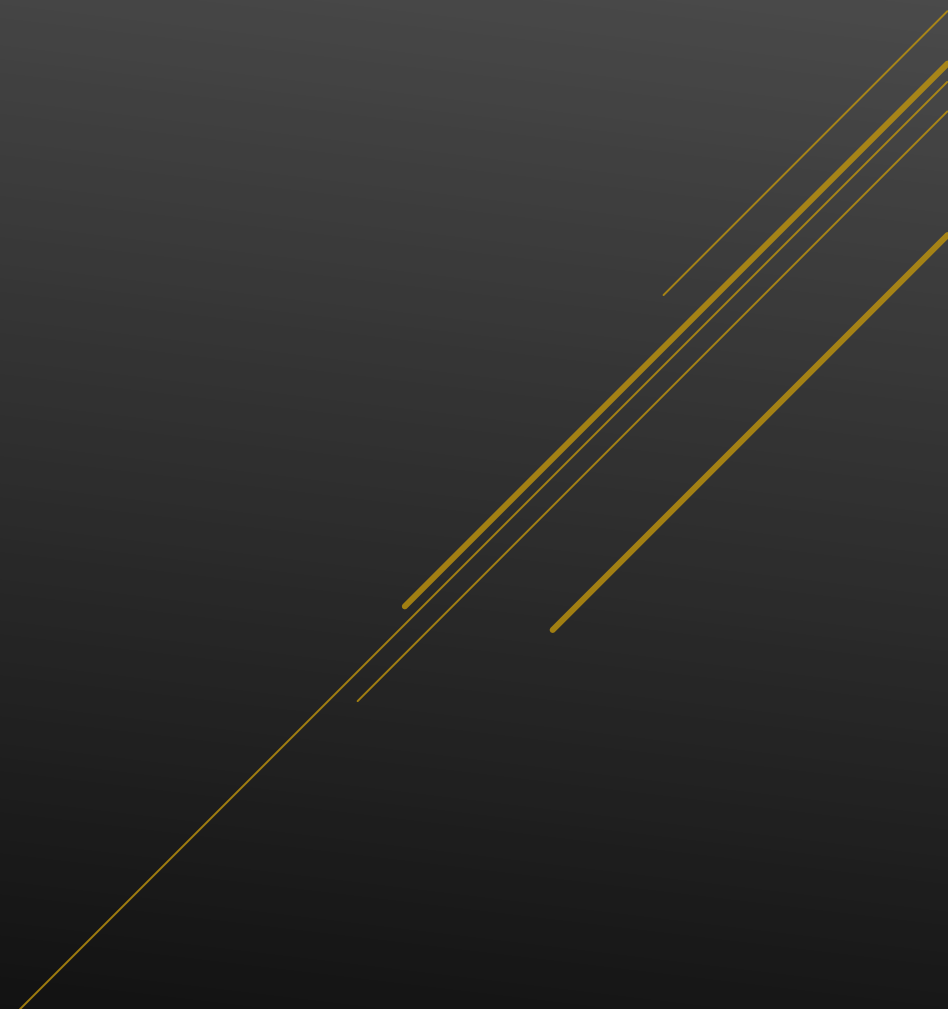
Una función pura es aquella con un resultado determinístico que tiene la menor cantidad posible de efectos secundarios.

PUREZA, UNIFICANDO CONCEPTOS

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

# PERO...

¿Qué es eso de un efecto secundario?





# EFFECTOS SECUNDARIOS

Los efectos secundarios (side effects) de una función se pueden entender como acciones que van más allá del alcance (scope) de una función.

## DEFINIENDO LOS EFECTOS SECUNDARIOS



- ▶ Depende de un estado (valor, atributo, propiedad, constante) no proporcionado como parámetro.
- ▶ Muta (modifica) un estado no local, fuera del cuerpo de la función.

Si cumple una de estas propiedades, la acción (instrucción) pasa a ser un efecto secundario

## LA CHECKLIST DE UN EFECTO SECUNDARIO



Pero entonces, ¿no hay que tenerlos? Sí que hay que tener efectos secundarios, lo que hay que hacer es minimizar su uso.

Un efecto secundario **daña la traza**, es un **mock extra** dentro de un test, hace que el resultado de una función sea algo **más impredecible**.

## ENTENDIENDO LOS EFECTOS SECUNDARIOS



Con ejemplos, todo se entiende mejor.

Pongamos que hemos recuperado información acerca de un usuario de la BDD, pero no queremos devolver toda la información, para ello, tenemos una función que filtra el cuerpo de la respuesta.

# EJEMPLOS

Un efecto secundario sería recuperar la información del usuario en la misma función que hace el filtro.

O guardar en una variable global la información del usuario que se ha recuperado y acceder a esta para filtrar los campos a devolver.

En TypeScript, ***formatCurrentUser(): ResponseUser***

En Java, ***ResponseUser formatCurrentUser()***

ES UN EFECTO SECUNDARIO  
CUANDO...

Siguiendo el ejemplo de antes, no sería un efecto secundario si nuestra función recibe la información del usuario y devuelve la respuesta.

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

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

NO SERÍA UN EFECTO SECUNDARIO  
SI...

En caso de estar en el frontend, este mismo caso puede darse.

Hemos recuperado la información del usuario del endpoint adecuado. Suponiendo que nuestra app no es multidioma.

No recuperaríamos el usuario directamente de un store, tendríamos una función que se encargaría de formatear la información del usuario, ***formatUser(user: User): string***

# CONTEXTUALIZANDO EN JAVASCRIPT

$y = g(x)$

Secant Lines

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

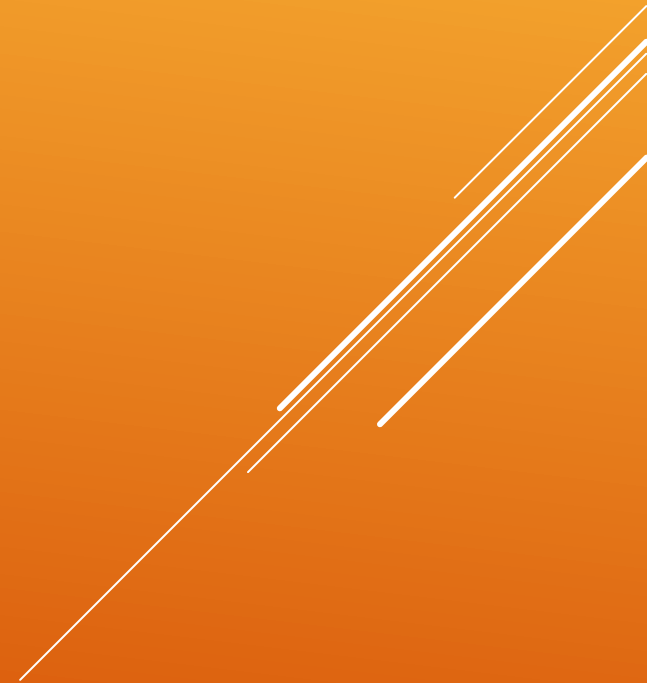
$g(x+h) - g(x)$

# IDEMPOTENCIA

La idempotencia (idempotency, idempotent) es la propiedad de una función matemática de ser completamente pura.

Es decir, no tiene efectos secundarios y su resultado será siempre el mismo, dado que le pasemos los mismos argumentos.

## QUÉ ES LA IDEMPOTENCIA



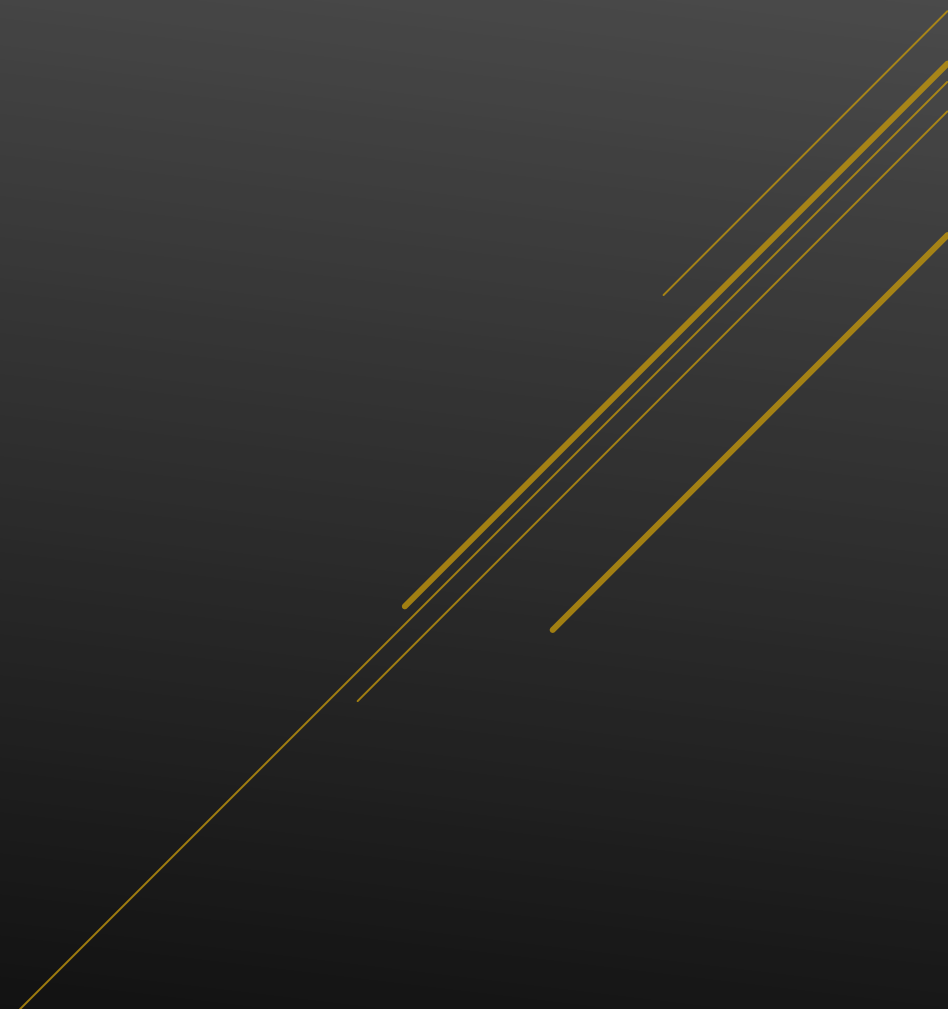


- ▶ Pureza de las funciones (trazabilidad)
- ▶ Posibilidad de ahorrarse computaciones
  - ▶ Un mismo resultado con los mismos argumentos podría cachearse...
- ▶ Decir una palabra no tan conocida para algo que es comúnmente conocido

## BENEFICIOS DE LA IDEMPOTENCIA

# “PEGA” DE LA IDEMPOTENCIA

No todo puede, ni ha de ser, idempotente.



- ▶ **Suma**

- ▶  $1 + 1$  a veces da 7, pero  $2 + 2$  siempre dará 4

- ▶ **Factorial**

- ▶ El factorial de un número será siempre el mismo

# EJEMPLOS DE IDEMPOTENCIA

- ▶ **Math.random**

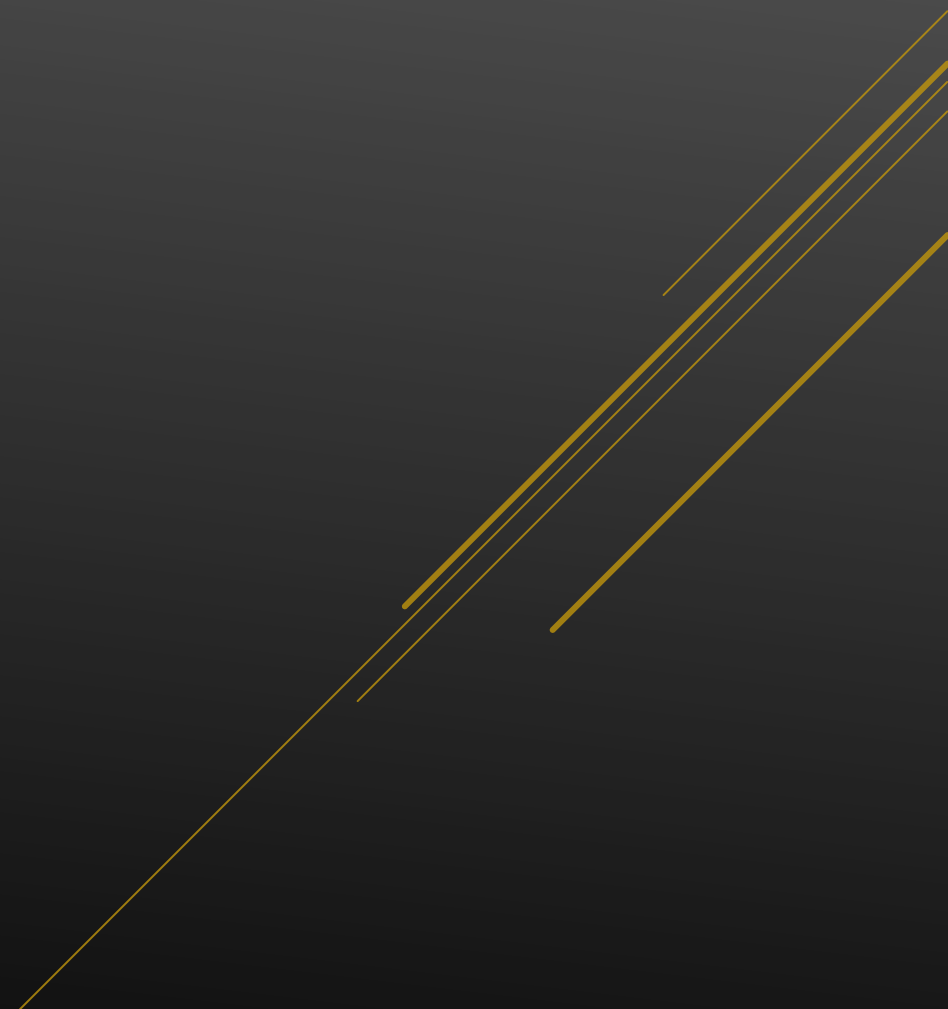
- ▶ Casi siempre devolverá un número diferente

- ▶ **Time.now**

- ▶ El tiempo está en constante cambio

EJEMPLOS QUE **NO** SON  
IDEMPOTENTES

RECAPITULEMOS...



## ► Programación funcional

- Un paradigma de programación

## ► Pureza

- Una función con la menor cantidad de efectos secundarios y un resultado determinístico

## ► Efectos secundarios

- Mutaciones fuera del alcance de la función


## ► Idempotencia

- Una función pura, *mismos parámetros == mismo resultado*

# HEMOS VISTO...






- 
- 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 serene and natural.
- ▶ Escribir código más claro y de mayor calidad
  - ▶ Entender mejor el ecosistema
  - ▶ Hacer buenas entrevistas técnicas
  - ▶ Nuevos caminos ante los problemas de siempre

LO CUÁL NOS PERMITE...

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

# Y AHORA... ¿POR DÓNDE PODRÍA CONTINUAR?

En siguientes episodios...

- 
- ▶ Memoización orientada a JavaScript
    - ▶ Serialización
      - ▶ Comparación de valores
  - ▶ Closure
  - ▶ Ciudadanía de primera clase
    - ▶ Alto orden
      - ▶ Funciones de alto orden
      - ▶ Componentes de alto orden

## PASOS A SEGUIR



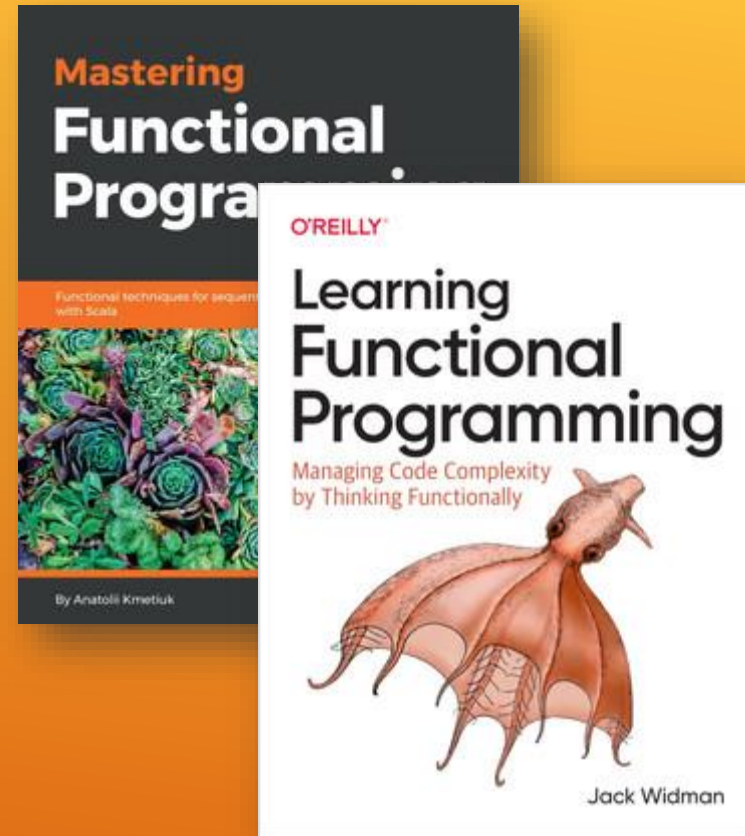
Learning Functional Programming  
by Jack Widman, O'Reilly

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>

# LIBROS





Anjana Vakil: Aprendiendo Programación Funcional con JavaScript - JSUnconf 2016



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

CHARLAS



Functional Programming for Pragmatists • Richard Feldman • GOTO 2021

Puedes leer el artículo al respecto en

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

¡¡CUIDADO!! Contiene spoilers de posibles siguientes charlas, pero el orden es el mismo.

# ARTÍCULO

Al terminar la charla se compartirá el acceso:

- A la grabación
- A las diapositivas

¡Que tengáis un buen día!

GRACIAS POR TU ATENCIÓN

