MASSIMILIANO MANTIONE
GIANLUCA CARUCCI

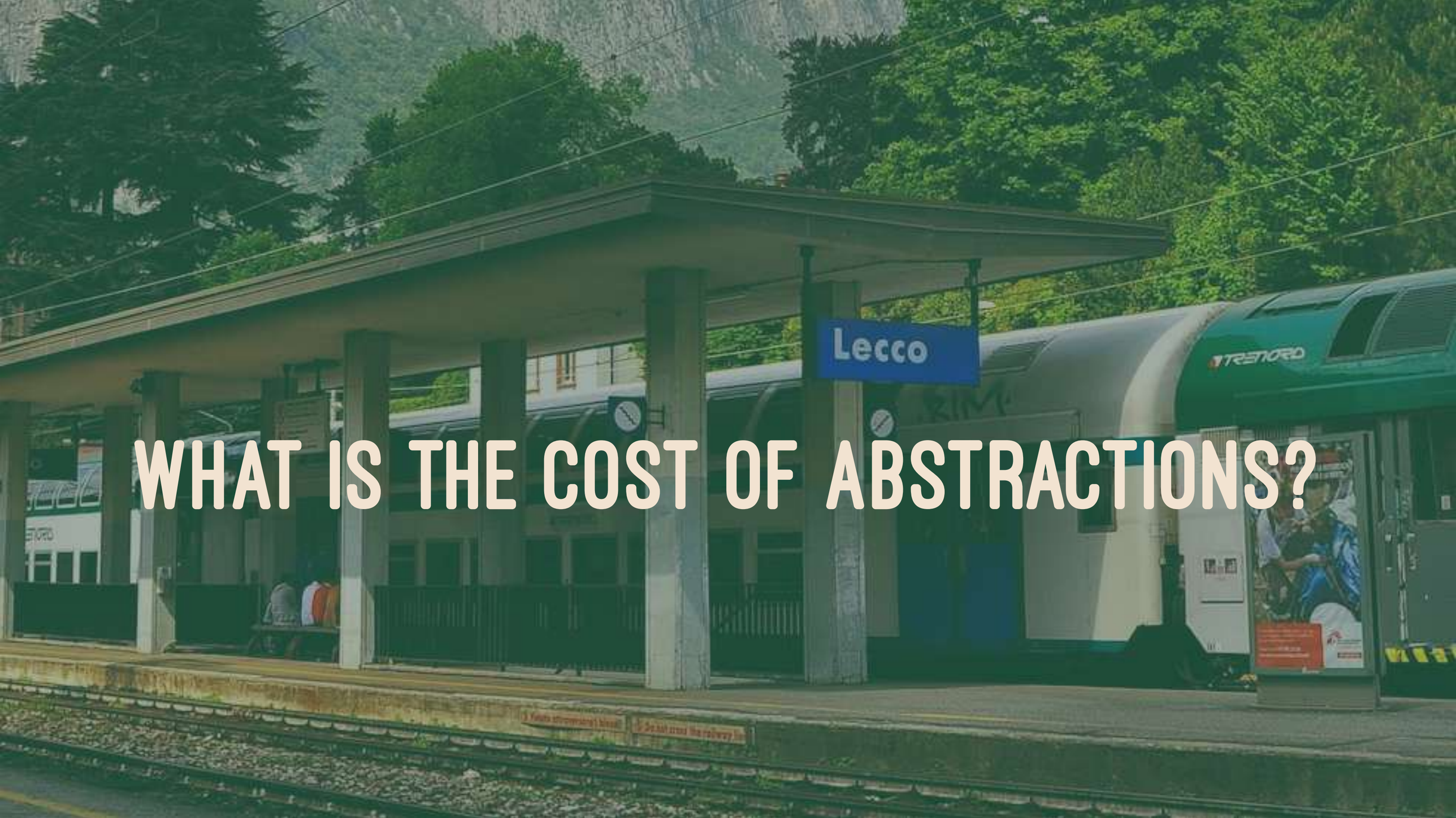
the cost of abstractions

TSconf

25 OCTOBER 2019

@M_a_s_s_i
www.linkedin.com/in/massimilianomantione
fbk.com/caruccigianluca
gianlucacarucci5
www.linkedin.com/in/rucka


9 3/4
HOGWARTS EXPRESS
ONCE UPON A TIME...
9 3/4

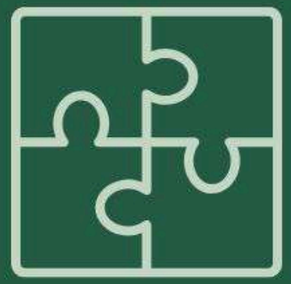


WHAT IS THE COST OF ABSTRACTIONS?

ehm...

POINT OF VIEW

performance



maintainability

Gianluca Carucci

Software Engineer

"Computer programs are the most complex things that humans make."

- Douglas Crockford,
JavaScript: The Good Parts

Massimiliano Mantione

Software Engineer

"bla bla blabla bla blabla bla blabla bla blabla bla blabla bla blabla bla blabla bla blabla bla blabla bla blabla bla bla"

- bla bla

Google

mono

unity

SIEMENS

HYPER FAIR VIRTUAL REALITY


WHAT DOES COST MEAN?

IT'S NOT ALL ABOUT PERFORMANCE

CONSIDER TYPESCRIPT

AS ABSTRACTION

OVER JAVASCRIPT

COMPILE FROM TYPESCRIPT

```
const processor: AsyncProcessor = async (
  orderId: string): Promise<PlacedOrderResult> => {
  const order = await orderService(orderId)
  if (order == null) {
    return {
      success: false
    }
  }
  const validationResult = await validationService(order)
  if (!validationResult.valid) {
    return placedOrderFailed
  }
  return await placeOrderService(order)
}
```

TO JAVASCRIPT

```
const processor                    = async (
  orderId         )                                  => {
  const order = await orderService(orderId)
  if (order == null) {
    return {
      success: false
    }
  }
  const validationResult = await validationService(order)
  if (!validationResult.valid) {
    return placedOrderFailed
  }
  return await placeOrderService(order)
}
```

IT'S NOT ALL ABOUT PERFORMANCE

- NO PERFORMACE PENALTY

IT'S NOT ALL ABOUT PERFORMANCE

- NO PERFORMACE PENALTY
- **YOU WILL PAY A** BUILD TIME **COST**

IT'S NOT ALL ABOUT PERFORMANCE

- NO PERFORMACE PENALTY
- YOU WILL PAY A BUILD TIME COST
- COGNITIVE OVERHEAD

[COGNITIVE OVERHEAD]

COGNITIVE: SOMETHING WE SHOULD KNOW
OVERHEAD: AN EXTRA EFFORT

[COGNITIVE OVERHEAD]

COGNITIVE: SOMETHING WE SHOULD KNOW
OVERHEAD: AN EXTRA EFFORT

THE EXTRA LEARNING EFFORT THE WHOLE TEAM MUST SPEND!

PERFORMACE MATTERS

LET'S MEASURE

"BUY A BOOK" USE CASE

- CREATE AN ORDER OF BOOKS

"BUY A BOOK" USE CASE

- CREATE AN ORDER OF BOOKS
- VALIDATE THE ORDER

"BUY A BOOK" USE CASE

- CREATE AN ORDER OF BOOKS
- VALIDATE THE ORDER
- PLACE THE ORDER

BENCHMARK

> "BUY A BOOK" USE CASE

BENCHMARK

- "BUY A BOOK" USE CASE
- 500K DIFFERENT ORDERS

BENCHMARK

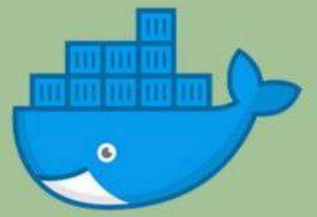
- "BUY A BOOK" USE CASE
- 500K DIFFERENT ORDERS
- 5% FAILURE RATE

BENCHMARK

- "BUY A BOOK" USE CASE
- 500K DIFFERENT ORDERS
- 5% FAILURE RATE
- MEASURE MEAN ORDER TIME

slides

code



environment

http://tsconf2019.carucci.org/


SEIKO
SEIKO
READY
STEADY
GO!

ASYNC TYPESCRIPT

```
const order = await orderService(orderId)
if (order == null) {
  return {
    success: false
  }
}
const validationResult = await validationService(order)
if (!validationResult.valid) {
  return placedOrderFailed
}
return await placeOrderService(order)
```

ASYNC

...

ASYNC

ASYNC TYPESCRIPT 6.772µS

NEXT STEP

ADD AN ABSTRACTION LAYER (FP-TS)

FUNCTIONAL TYPESCRIPT

```
return pipe(
  orderService(orderId),
  chain(validationService),
  chain(placeOrderService)
)
```

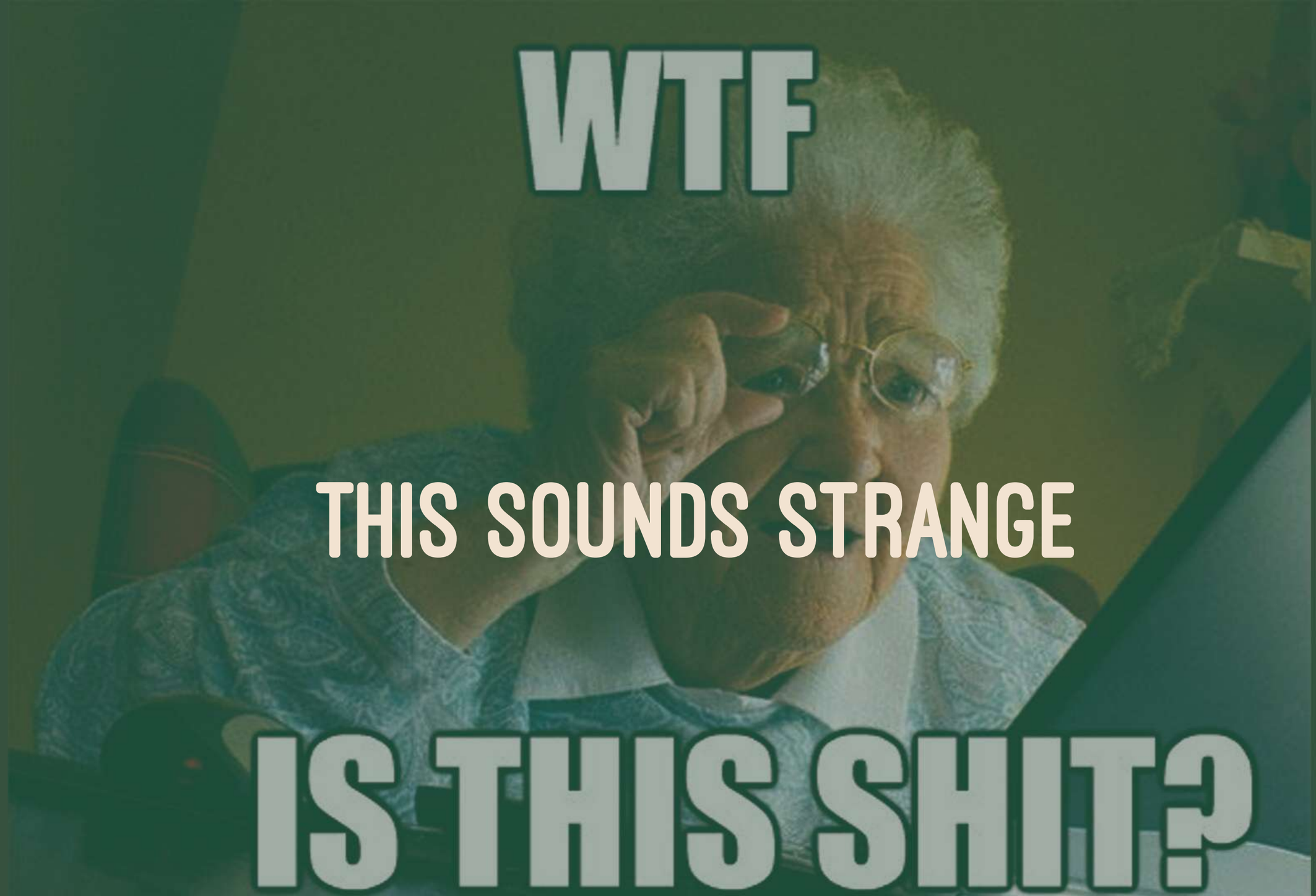
FROM ASYNC TO FP

ASYNC TYPESCRIPT 6.772µS

FROM ASYNC TO FP

ASYNC TYPESCRIPT 6.772µs

FUNCTIONAL TYPESCRIPT 5.952µs


WTF
THIS SOUNDS STRANGE
IS THIS SHIT?

FROM ASYNC TO FP

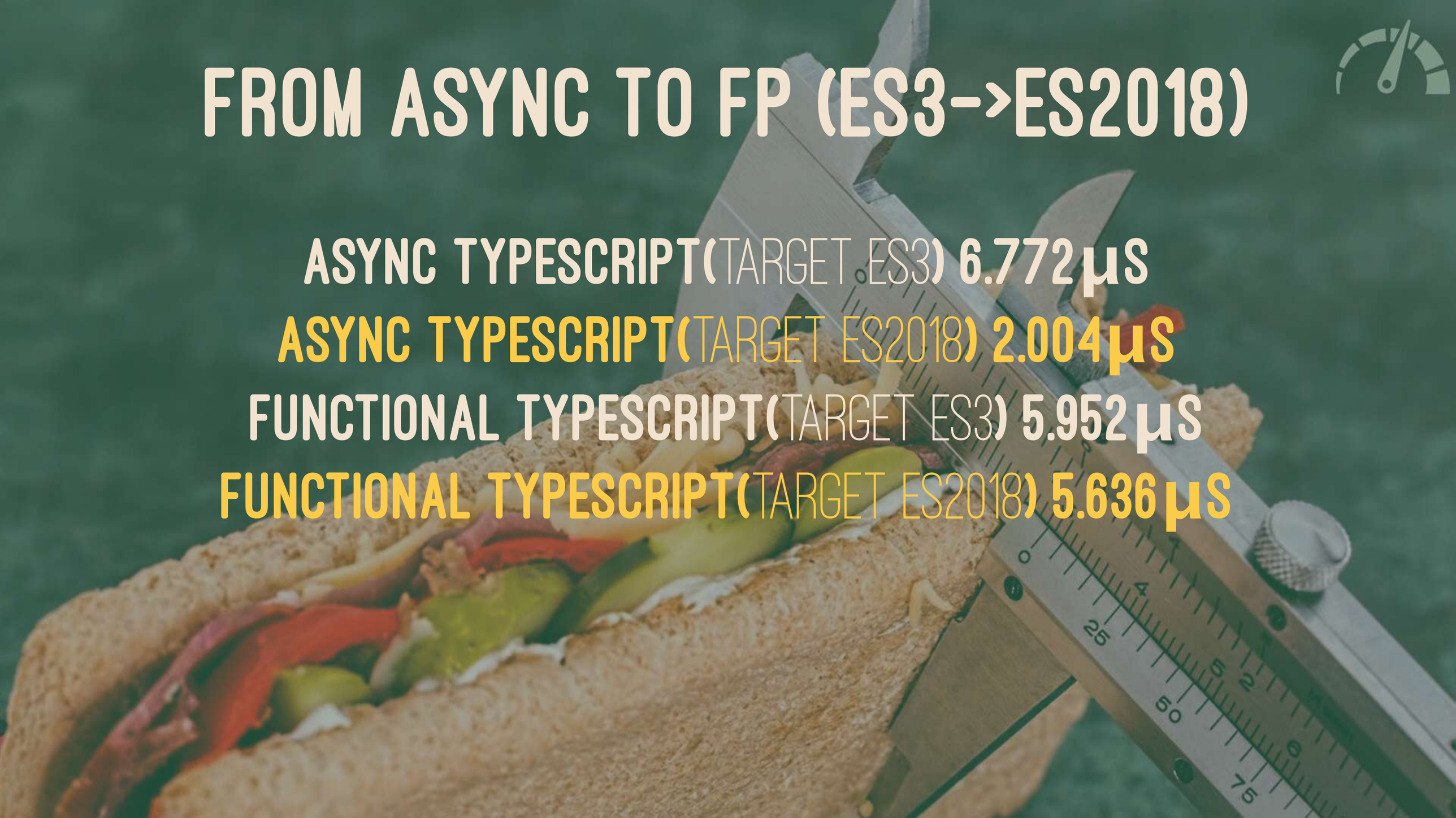
ASYNC TYPESCRIPT 6.772µs

FUNCTIONAL TYPESCRIPT 5.952µs

FROM ASYNC TO FP

ASYNC TYPESCRIPT(TARGET ES3) 6.772µs

FUNCTIONAL TYPESCRIPT(TARGET ES3) 5.952µs



FROM ASYNC TO FP (ES3->ES2018)

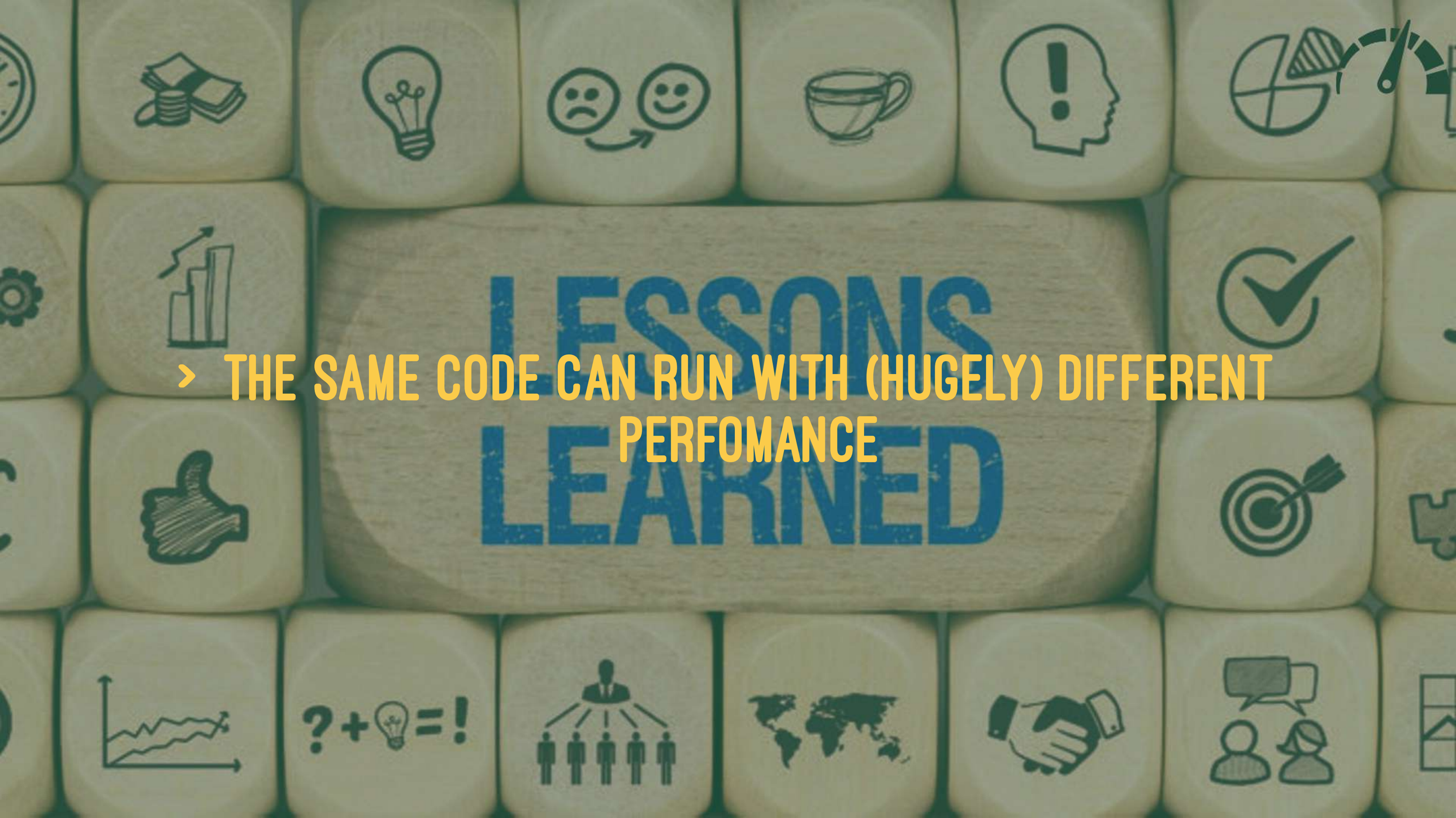
ASYNC TYPESCRIPT(TARGET ES3) 6.772µs

ASYNC TYPESCRIPT(TARGET ES2018) 2.004µs

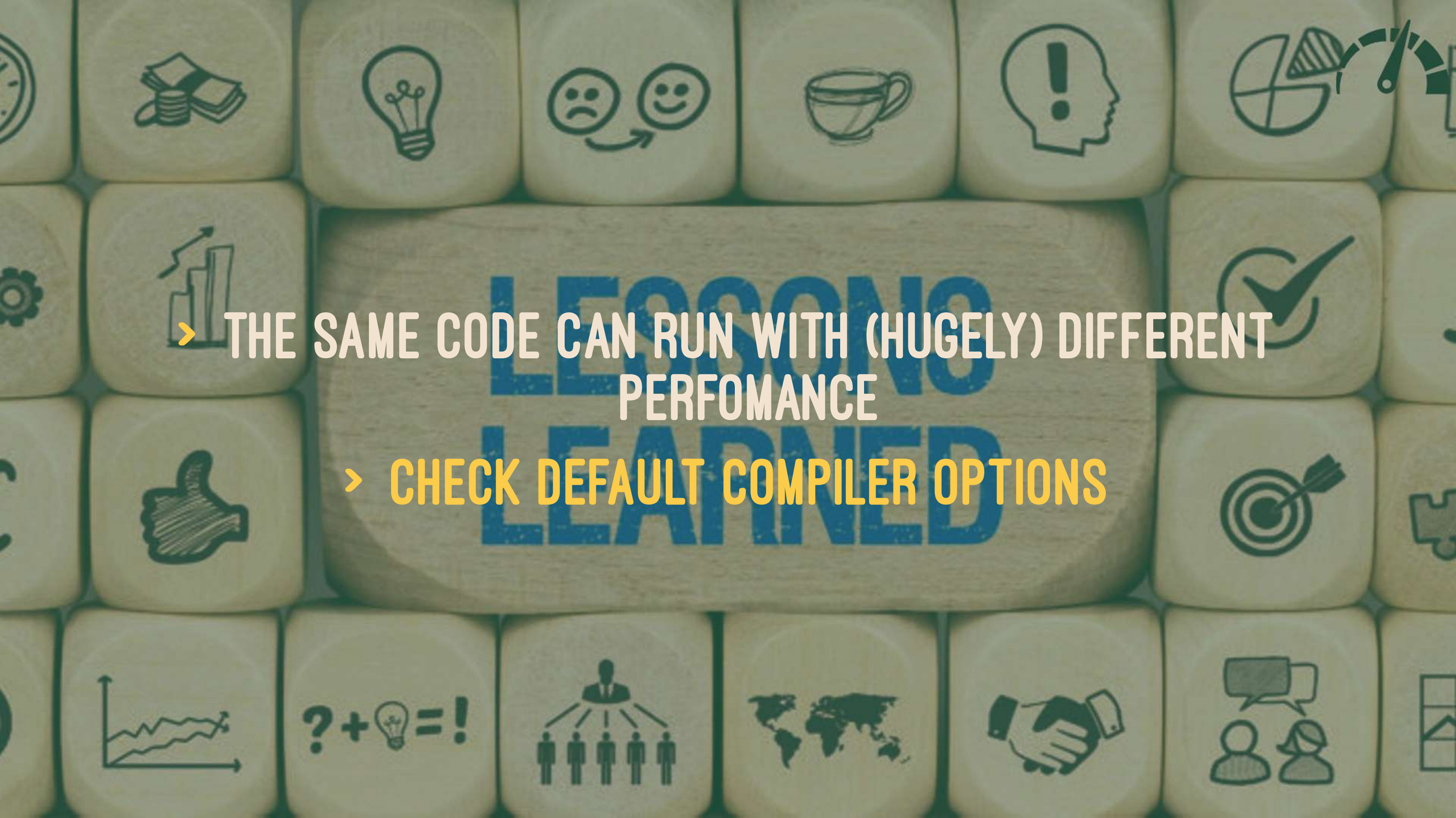
FUNCTIONAL TYPESCRIPT(TARGET ES3) 5.952µs

FUNCTIONAL TYPESCRIPT(TARGET ES2018) 5.636µs


LESSONS
LEARNED



- THE SAME CODE CAN RUN WITH (HUGELY) DIFFERENT PERFOMANCE



- THE SAME CODE CAN RUN WITH (HUGELY) DIFFERENT PERFOMANCE
- CHECK DEFAULT COMPILER OPTIONS


NEXT STEP
ENFORCE BUSINESS RULES AT COMPILE TIME

CHECKED FUNCTIONAL TYPESCRIPT

```
type NotValid = Left<Error>
type Valid<A> = Right<A>
type Validated<A> = Either<Error, A>

function validationService (o: Order): Validated<Order>  {
  const r = validateOrder(order)
  if (r.valid) {
    return valid<Order>(order)
  } else {
    return notvalid(`${r.error}`)
  }
}
```

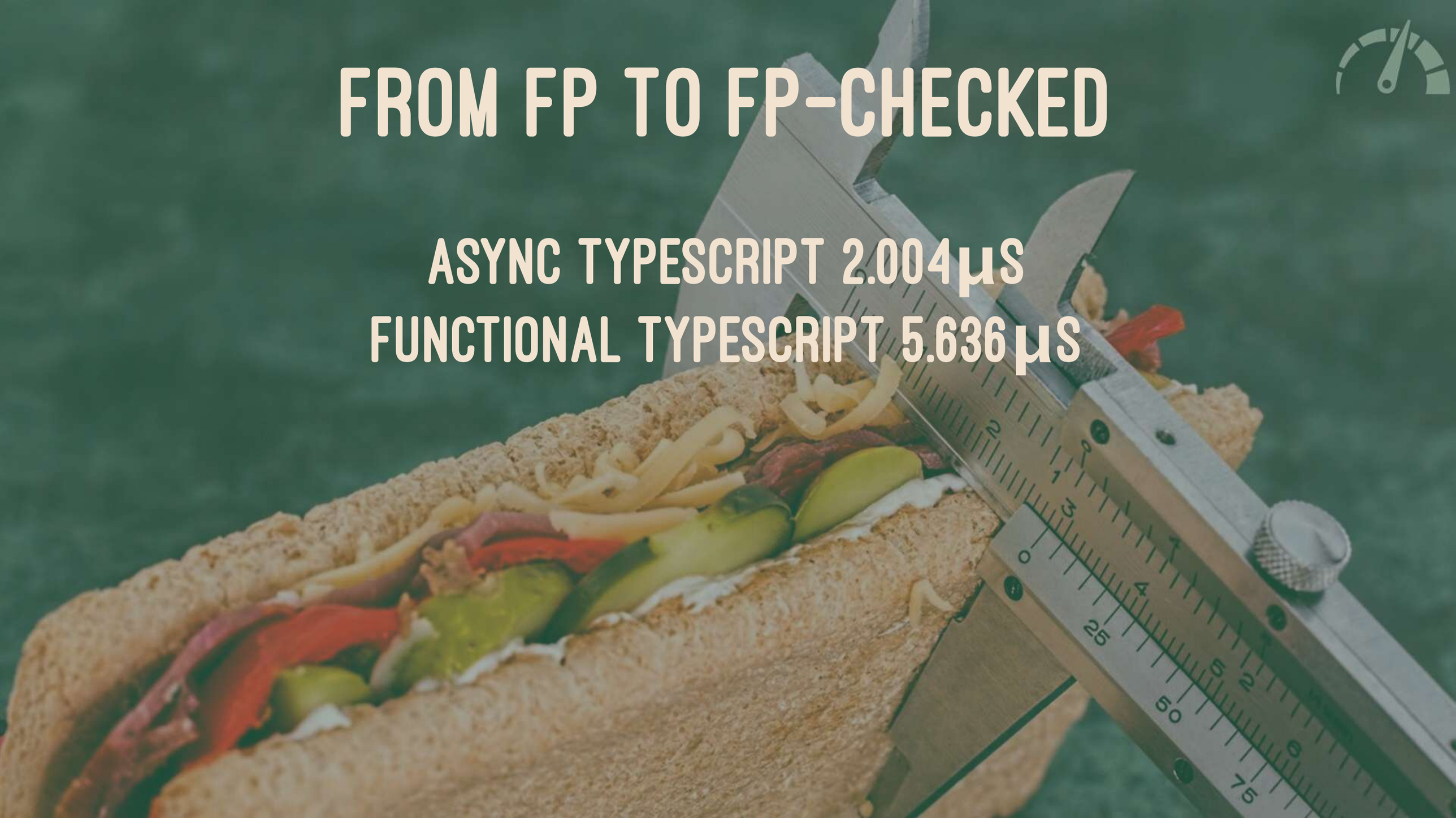
CHECKED FUNCTIONAL TYPESCRIPT

```
function calculateAmountService (order: Valid<Order>) {
  return pipe(
    order.right.items.map(item =>
      pipe(
        bookService(item.bookId),
        map(b => b.price * item.quantity)
      )
    ),
    array.sequence(taskEither),
    map(amounts => {
      return amounts.reduce((a, b) => a + b, 0)
    })
  )
}
```

CHECKED FUNCTIONAL TYPESCRIPT

```
return pipe(
  orderId,
  orderService,
  map(validationService),
  chain(mapTask(placeOrderService))
)
```

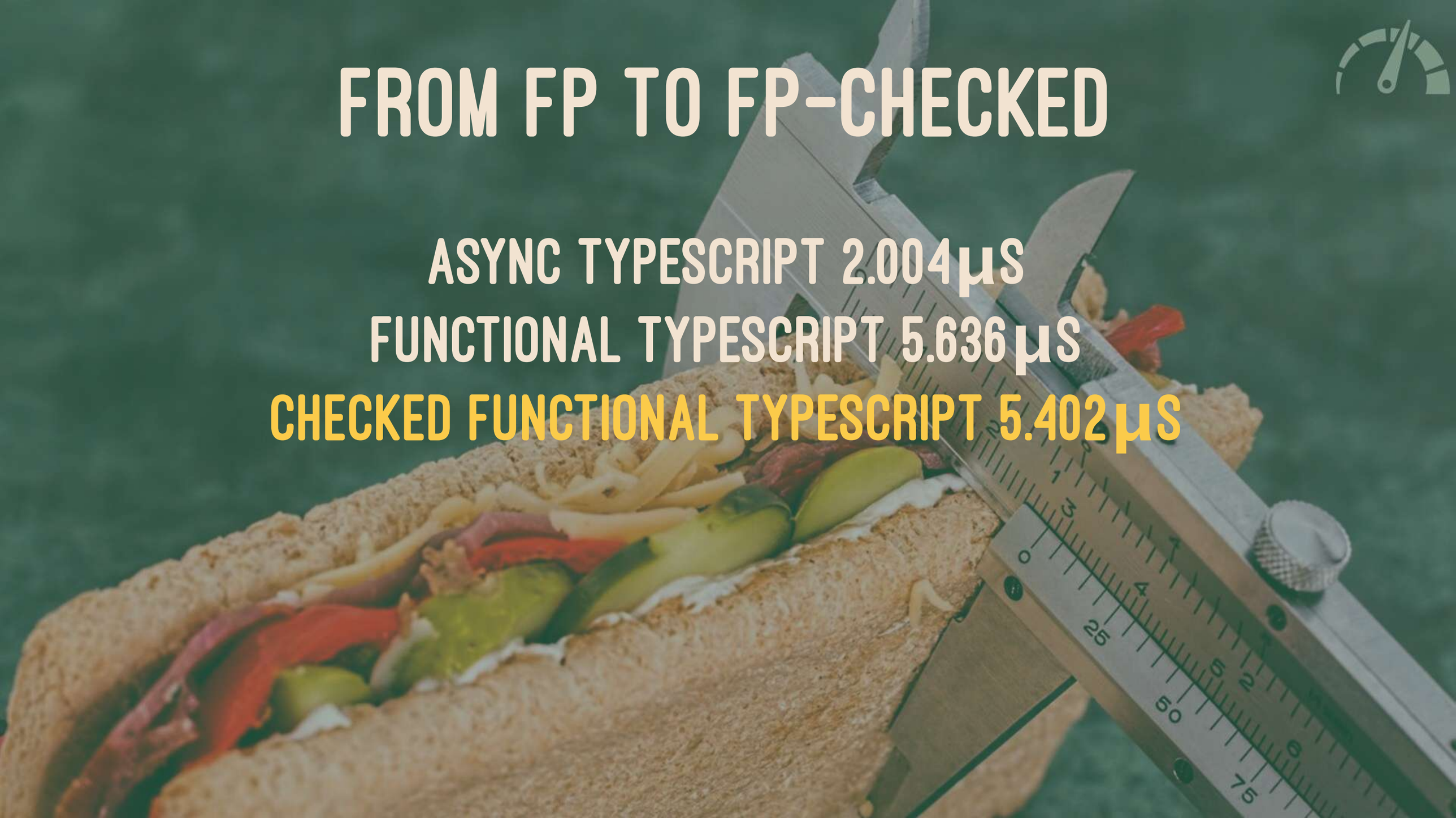
FROM FP TO FP-CHECKED



ASYNC TYPESCRIPT 2.004µs

FUNCTIONAL TYPESCRIPT 5.636µs

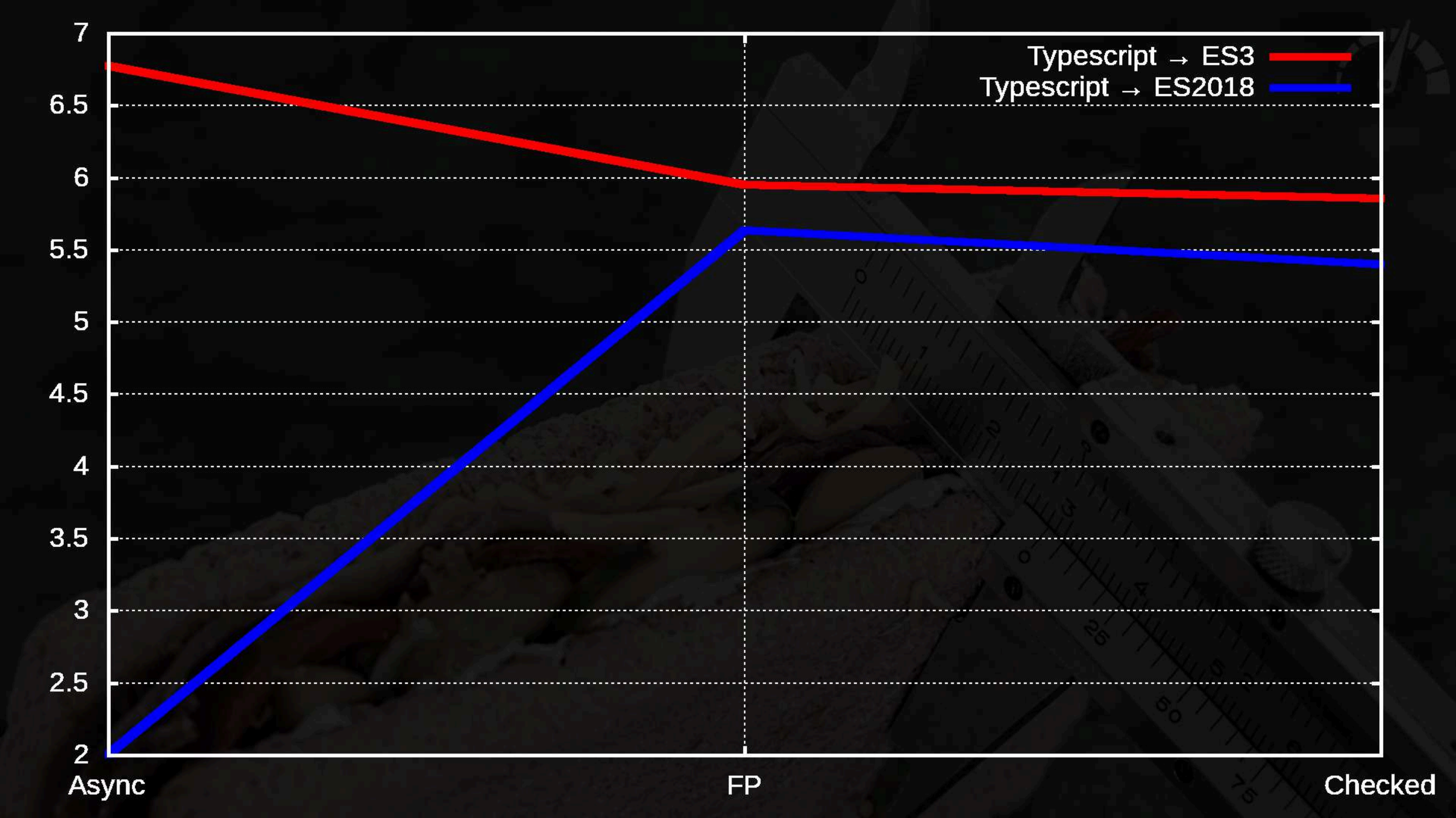
FROM FP TO FP-CHECKED



ASYNC TYPESCRIPT 2.004µs

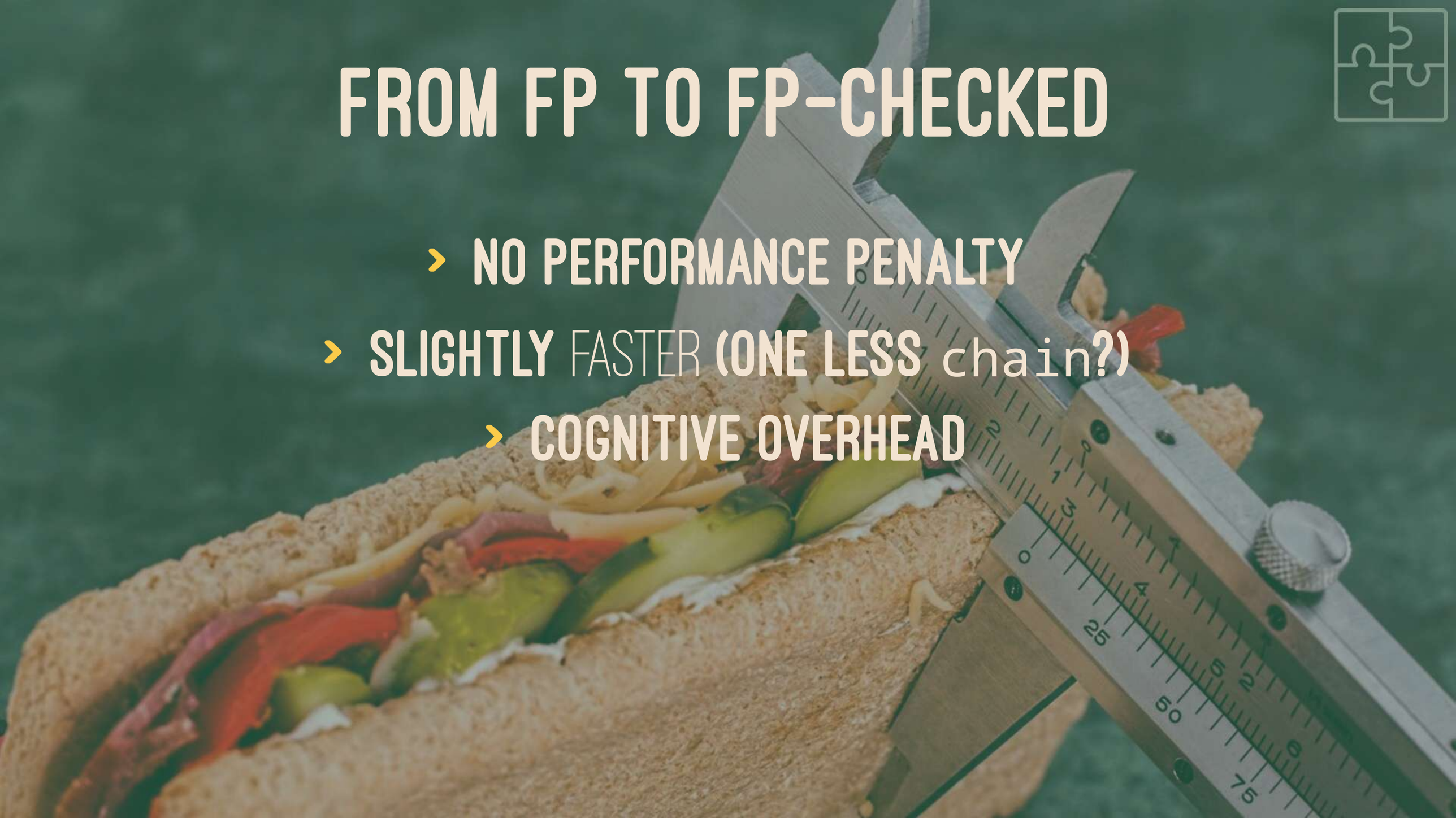
FUNCTIONAL TYPESCRIPT 5.636µs

CHECKED FUNCTIONAL TYPESCRIPT 5.402µs

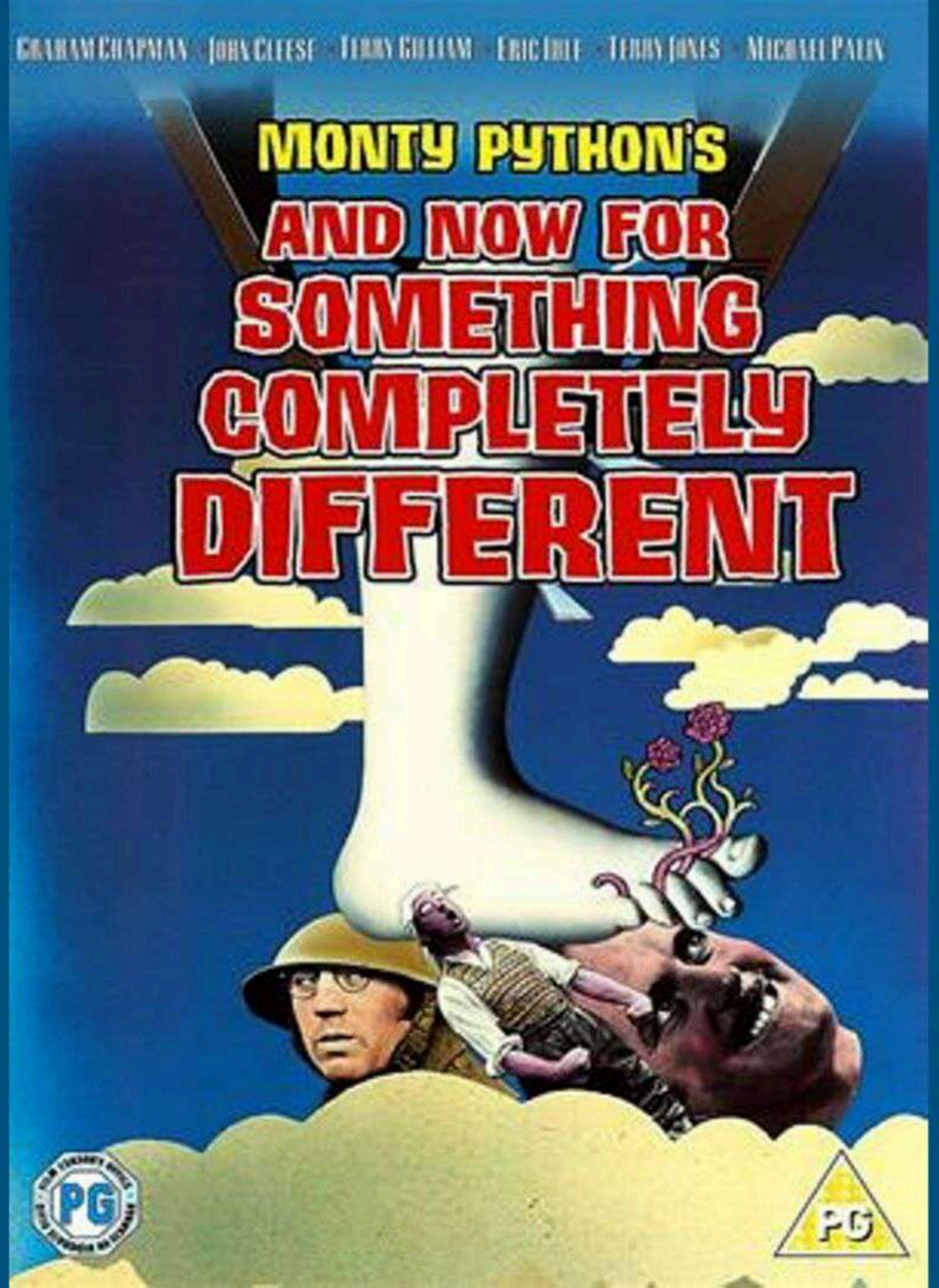

7
6.5
6
5.5
5
4.5
4
3.5
3
2.5
2
Typescript → ES3
Typescript → ES2018
Async
FP
Checked

FROM FP TO FP-CHECKED

- NO PERFORMANCE PENALTY
- SLIGHTLY FASTER (ONE LESS `chain`?)
- COGNITIVE OVERHEAD




CAN WE HAVE
THE BEST OF
BOTH WORLDS?


GRAHAM CHAPMAN JOHN CLEESE TERRY GILLIAM ERIC IDLE TERRY JONES MICHAEL PALIN
MONTY PYTHON'S
AND NOW FOR
SOMETHING
COMPLETELY
DIFFERENT
PG
PG


NEXT STEP
A DIFFERENT LANGUAGE

The Rust programming language

Jump to: Rust and the future of systems programming Unlocking the power of parallel

Safe systems programming with Rust Growing the Rust community Putting Rust into p

Rust is a new open-source systems programming language created by Mozilla and a community of volunteers, designed to help developers create fast, secure applications which take full advantage of the powerful features of modern multi-core processors. It prevents segmentation faults and guarantees thread safety, all with an easy-to-learn syntax.

In addition, Rust offers zero-~~cost~~ *overhead* abstractions, move semantics, guaranteed memory safety, threads with no data races, trait-based generics, pattern matching, type inference, and efficient C bindings, with a minimum runtime size.

To learn more about Rust, you can:

- Watch the videos below for a closer look at the power and benefits Rust provides.
- Read the book *The Rust Programming Language* online.
- Download the Rust compiler, check out examples, and learn everything you could

WHAT DOES ZERO COST MEAN?

- ZERO COST **FOR THE ABSTRACTIONS YOU** DO NOT USE
  - **WHAT YOU** DO **USE. CANNOT BE DONE** ANY BETTER

WHAT DOES ZERO COST MEAN?

- ZERO COST **FOR THE ABSTRACTIONS YOU** DO NOT USE
  - **WHAT YOU** DO **USE. CANNOT BE DONE** ANY BETTER
    - **THIS MEANS "ZERO** RUNTIME OVERHEAD"

WHAT DOES ZERO COST MEAN?

- ZERO COST FOR THE ABSTRACTIONS YOU DO NOT USE
- WHAT YOU DO USE, CANNOT BE DONE ANY BETTER
- THIS MEANS "ZERO RUNTIME OVERHEAD"
- YOU WILL PAY A BUILD TIME COST
- PLUS COGNITIVE OVERHEAD...

A RUST IMPLEMENTATION

- FAITHFUL TO THE TYPESCRIPT ONE

A RUST IMPLEMENTATION

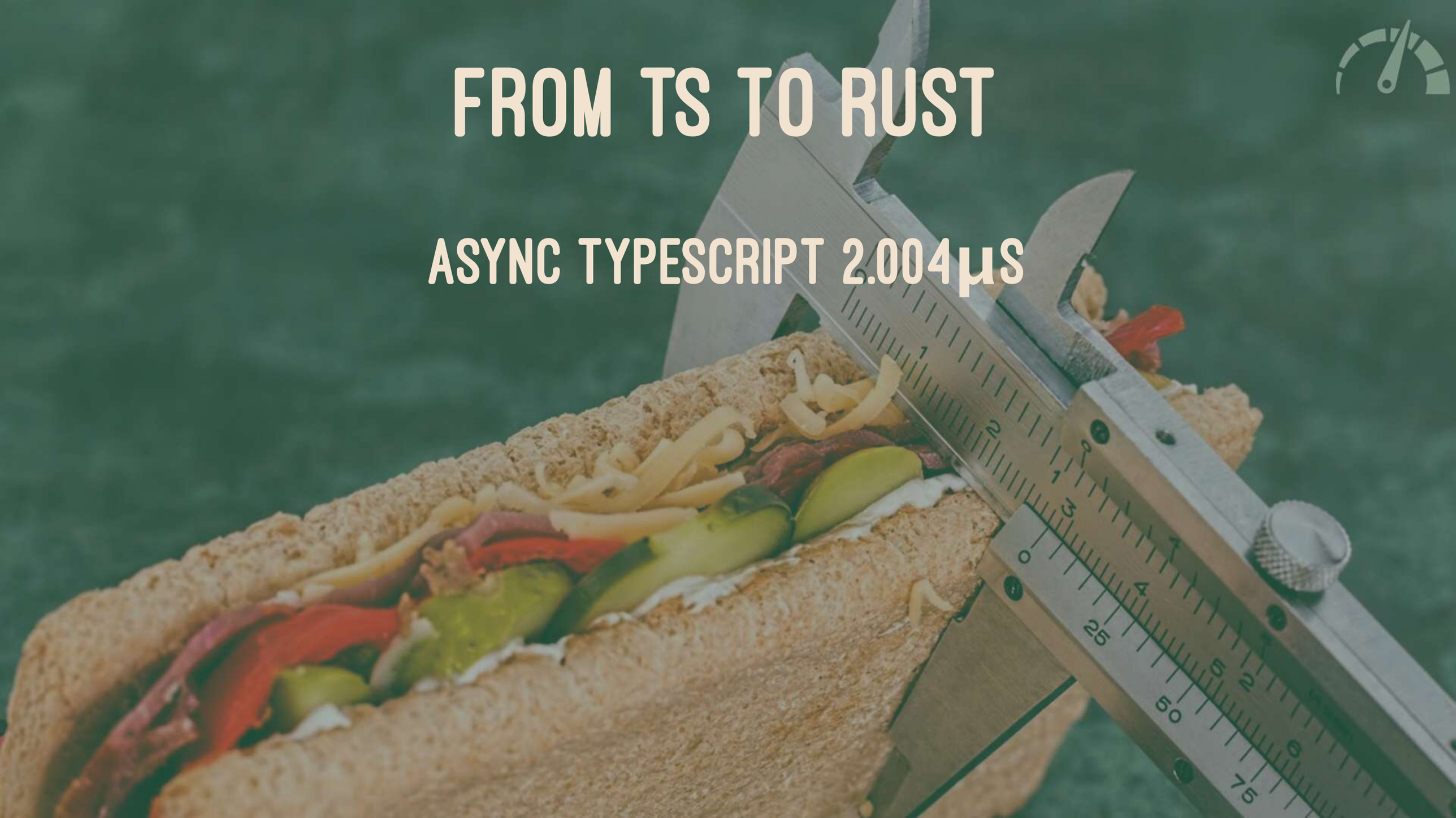
- FAITHFUL TO THE TYPESCRIPT ONE
  - LINE BY LINE ADAPTATION

A RUST IMPLEMENTATION

- FAITHFUL TO THE TYPESCRIPT ONE
  - LINE BY LINE ADAPTATION
    - LET'S BENCHMARK!

FROM TS TO RUST

ASYNC TYPESCRIPT 2.004µs

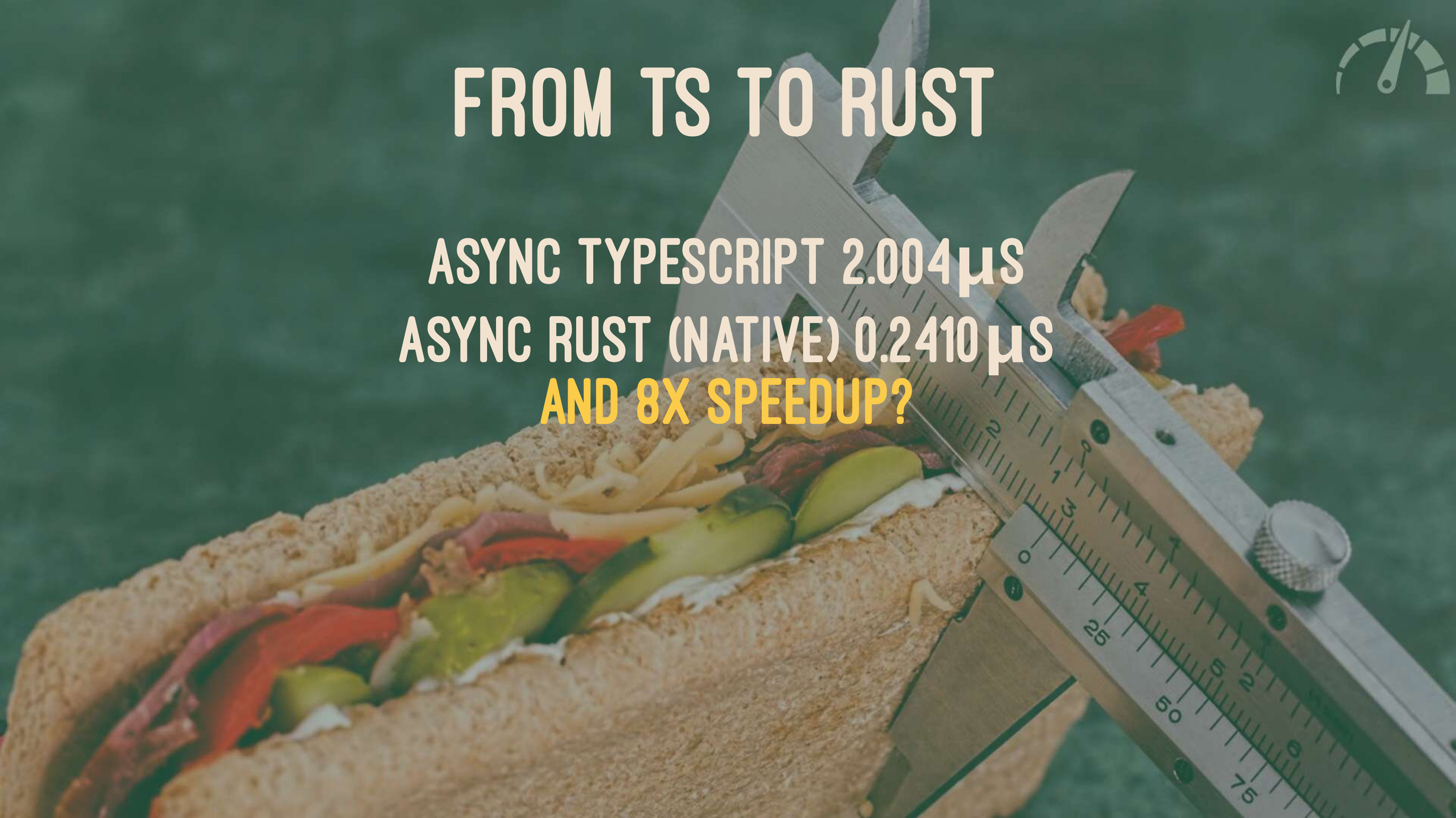



FROM TS TO RUST
ASYNC TYPESCRIPT 2.004µs
WHAT DO YOU EXPECT?

FROM TS TO RUST

ASYNC TYPESCRIPT 2.004µs

ASYNC RUST (NATIVE) 0.2410µs



FROM TS TO RUST

ASYNC TYPESCRIPT 2.004µs

ASYNC RUST (NATIVE) 0.2410µs

AND 8X SPEEDUP?

WAT?

DOES THIS MAKE SENSE?

WAT?

DOES THIS MAKE SENSE?
IS IT A FAIR COMPARISON?

WAT?

DOES THIS MAKE SENSE?
IS IT A FAIR COMPARISON?
LET'S INVESTIGATE

WAT?

DOES THIS MAKE SENSE?
IS IT A FAIR COMPARISON?
LET'S INVESTIGATE
WE START FROM SCRATCH

WAT?

DOES THIS MAKE SENSE?
IS IT A FAIR COMPARISON?
LET'S INVESTIGATE
WE START FROM SCRATCH
WITH A SYNCHRONOUS TYPESCRIPT VERSION

START FROM SCRATCH

FROM SYNCHRONOUS **TYPESCRIPT VERSION**

START FROM SCRATCH

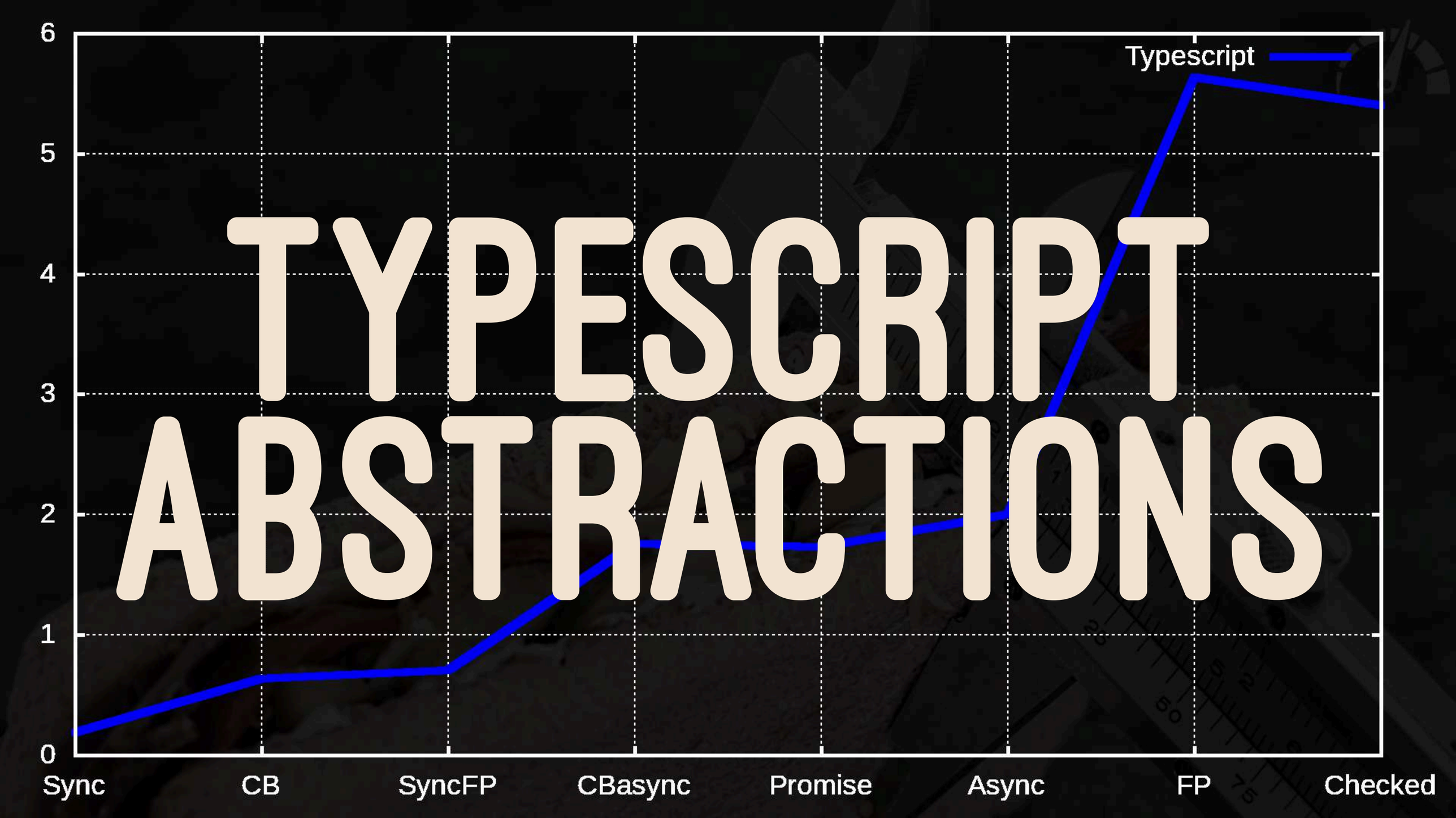
FROM SYNCHRONOUS TYPESCRIPT VERSION
THEN WE ADD MINIMAL ABSTRACTIONS

START FROM SCRATCH

FROM SYNCHRONOUS TYPESCRIPT VERSION
THEN WE ADD MINIMAL ABSTRACTIONS
ONE BY ONE

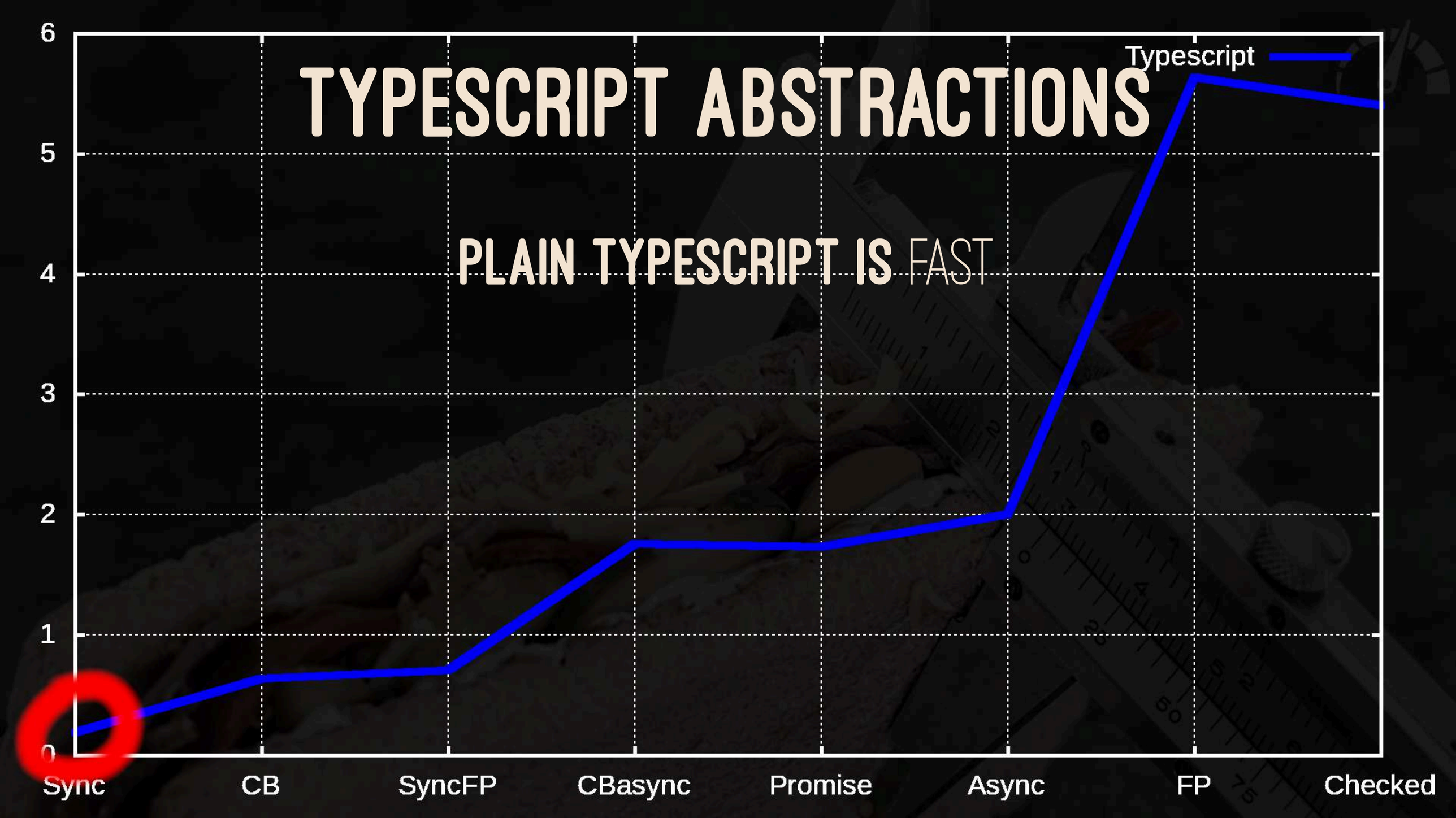
START FROM SCRATCH

FROM SYNCHRONOUS TYPESCRIPT VERSION
THEN WE ADD MINIMAL ABSTRACTIONS
ONE BY ONE
AND BENCHMARK EACH STEP

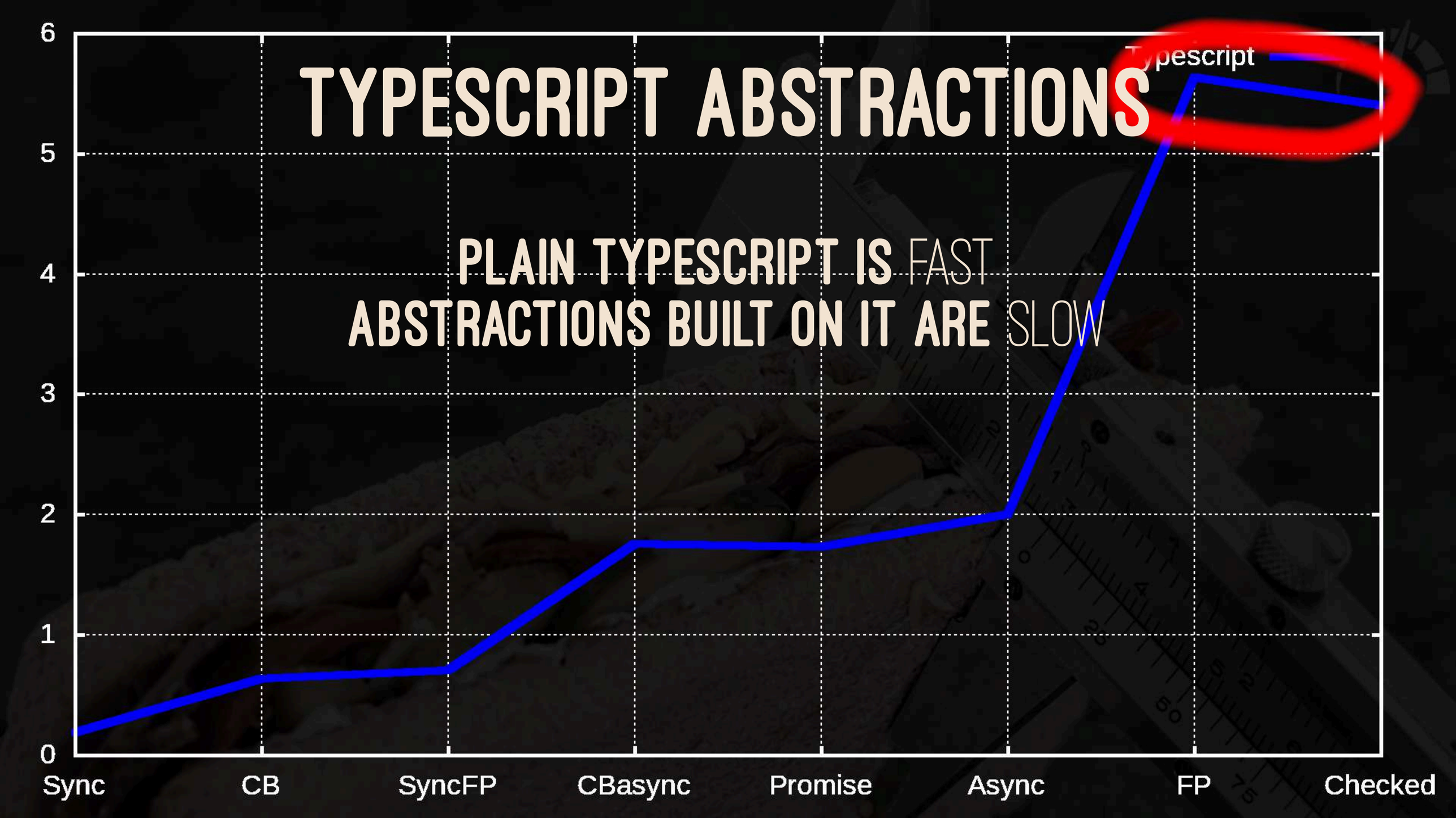

TYPESCRIPT
ABSTRACTIONS
Typescript
6
5
4
3
2
1
0
Sync
CB
SyncFP
CBasync
Promise
Async
FP
Checked

TYPESCRIPT ABSTRACTIONS

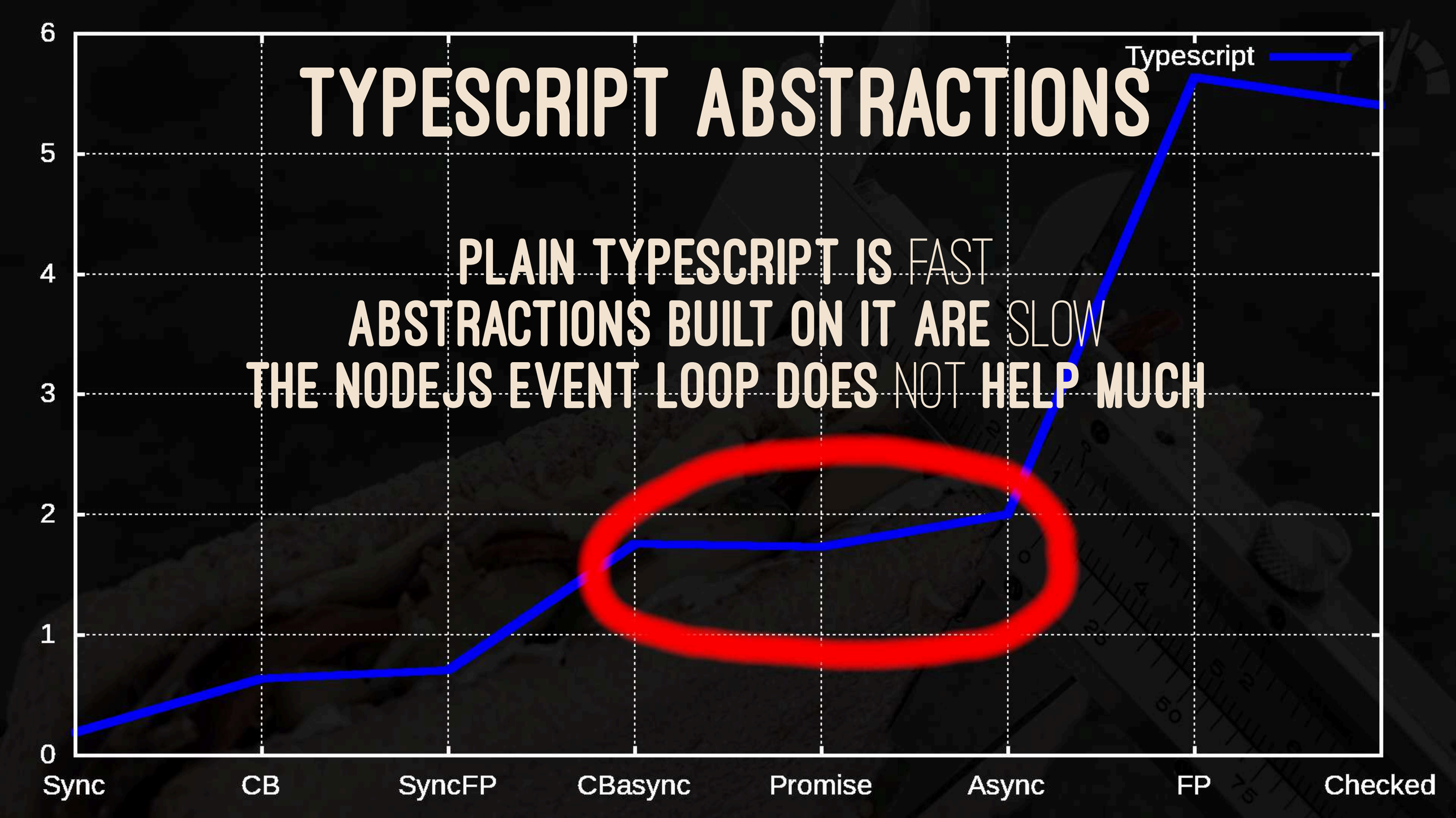
PLAIN TYPESCRIPT IS FAST



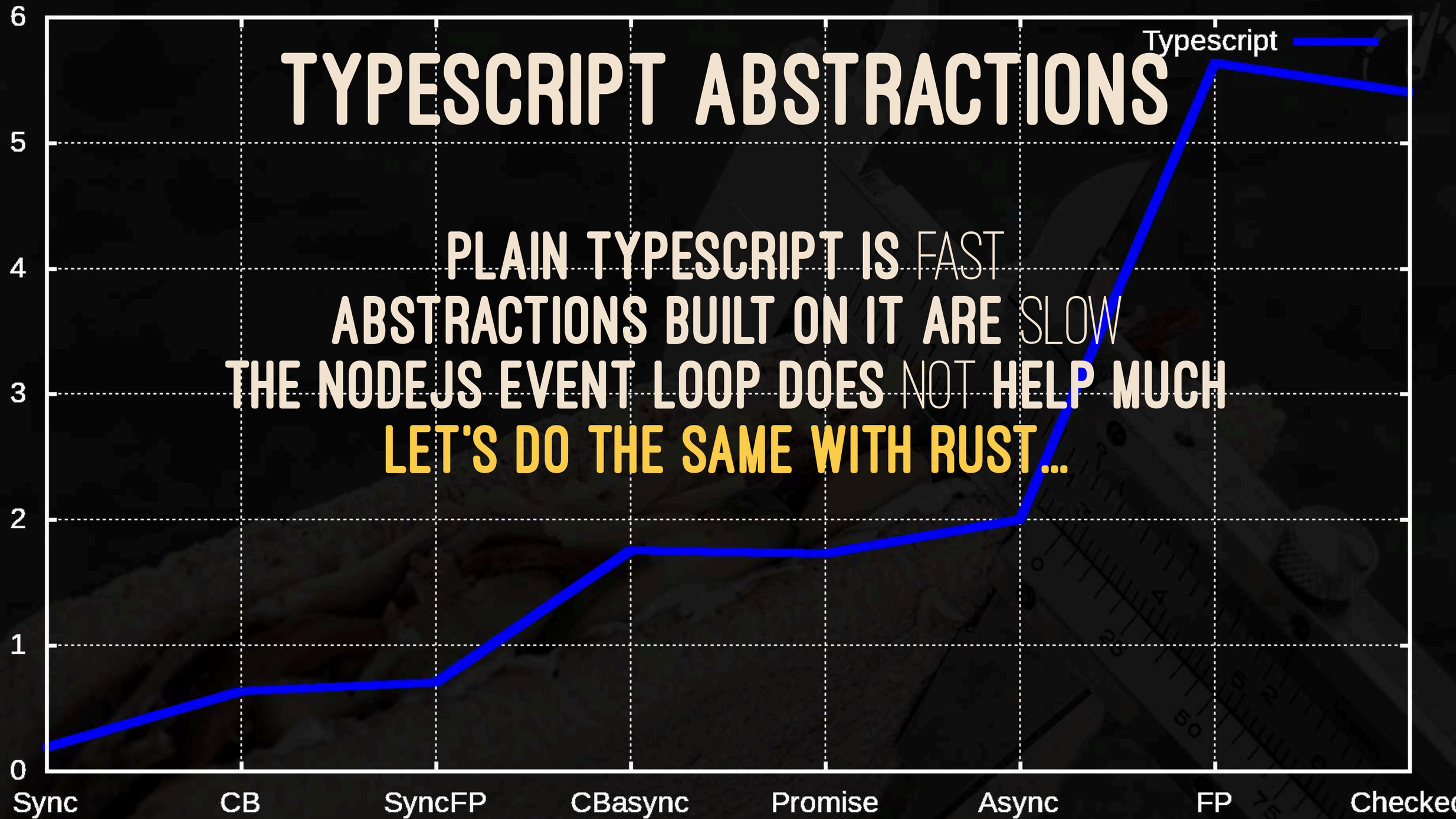
TYPESCRIPT ABSTRACTIONS



PLAIN TYPESCRIPT IS FAST
ABSTRACTIONS BUILT ON IT ARE SLOW


TYPESCRIPT ABSTRACTIONS
PLAIN TYPESCRIPT IS FAST
ABSTRACTIONS BUILT ON IT ARE SLOW
THE NODEJS EVENT LOOP DOES NOT HELP MUCH
Typescript
6
5
4
3
2
1
0
Sync
CB
SyncFP
CBasync
Promise
Async
FP
Checked

TYPESCRIPT ABSTRACTIONS



PLAIN TYPESCRIPT IS FAST
ABSTRACTIONS BUILT ON IT ARE SLOW
THE NODEJS EVENT LOOP DOES NOT HELP MUCH
LET'S DO THE SAME WITH RUST...

RUST. PLAIN

```
pub async fn process(order_id: &String) -> Result<f64, ()> {
    match order_service(order_id).await {
        Some(order) => {
            let validation = validation_service(&order).await;
            match validation {
                Ok(order) => match place_order_service(order).await {
                    Ok(res) => Ok(res.amount),
                    Err(_) => Err(()),
                },
                _ => Err(()),
            }
        }
        _ => Err(()),
    }
}
```

RUST. IDIOMATIC

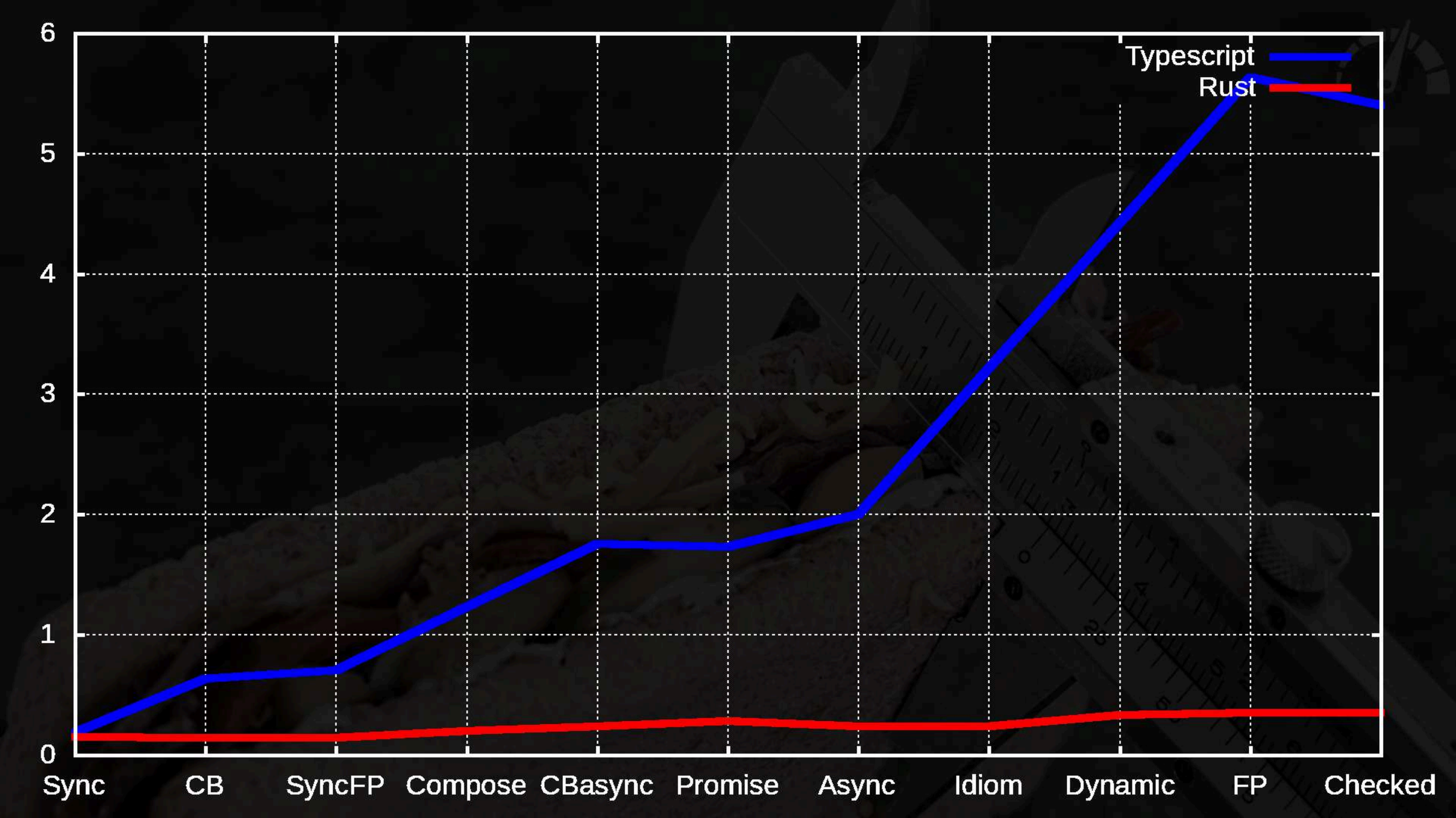
```rust
pub async fn process(order_id: &String) -> Result<f64, ()> {
    let order = order_service(order_id).await;
    let validated = validation_service(order).await.map_err(|_| ())?;
    Ok(place_order_service(validated).await.map_err(|_| ())?.amount)
}
```

RUST. COMPOSABLE

```
pub fn process(order_id: &'static String) ->
        impl Future<Output = Result<f64, ()>> {
    compose!(
        order_service,
        validation_service,
        place_order_service,
        map_order_amount
    )(order_id)
}
```

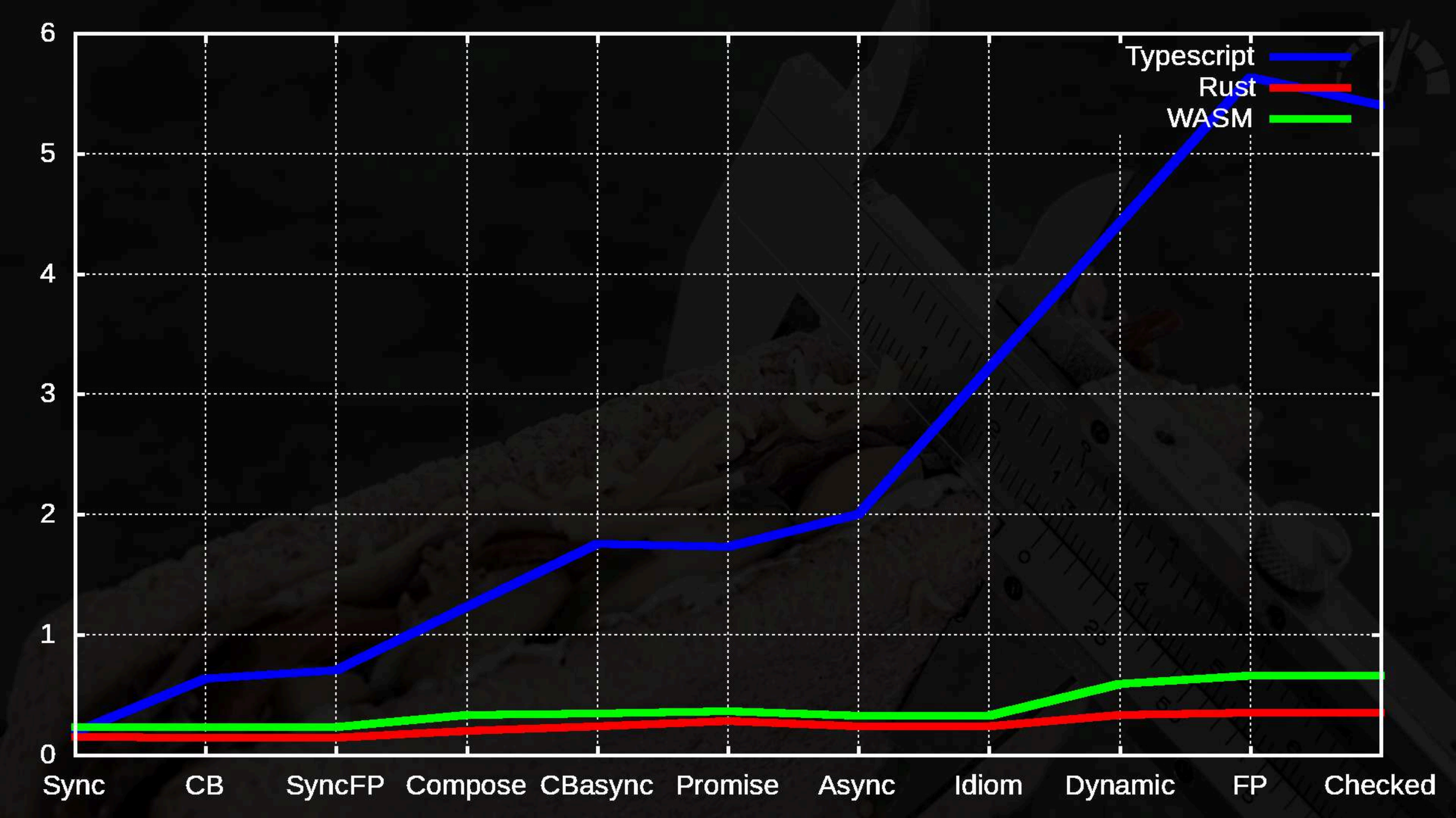
WE HAVE SEEN RUST...

SHOULD WE MEASURE IT?


Typescript
Rust
6
5
4
3
2
1
0
Sync
CB
SyncFP
Compose
CBasync
Promise
Async
Idiom
Dynamic
FP
Checked

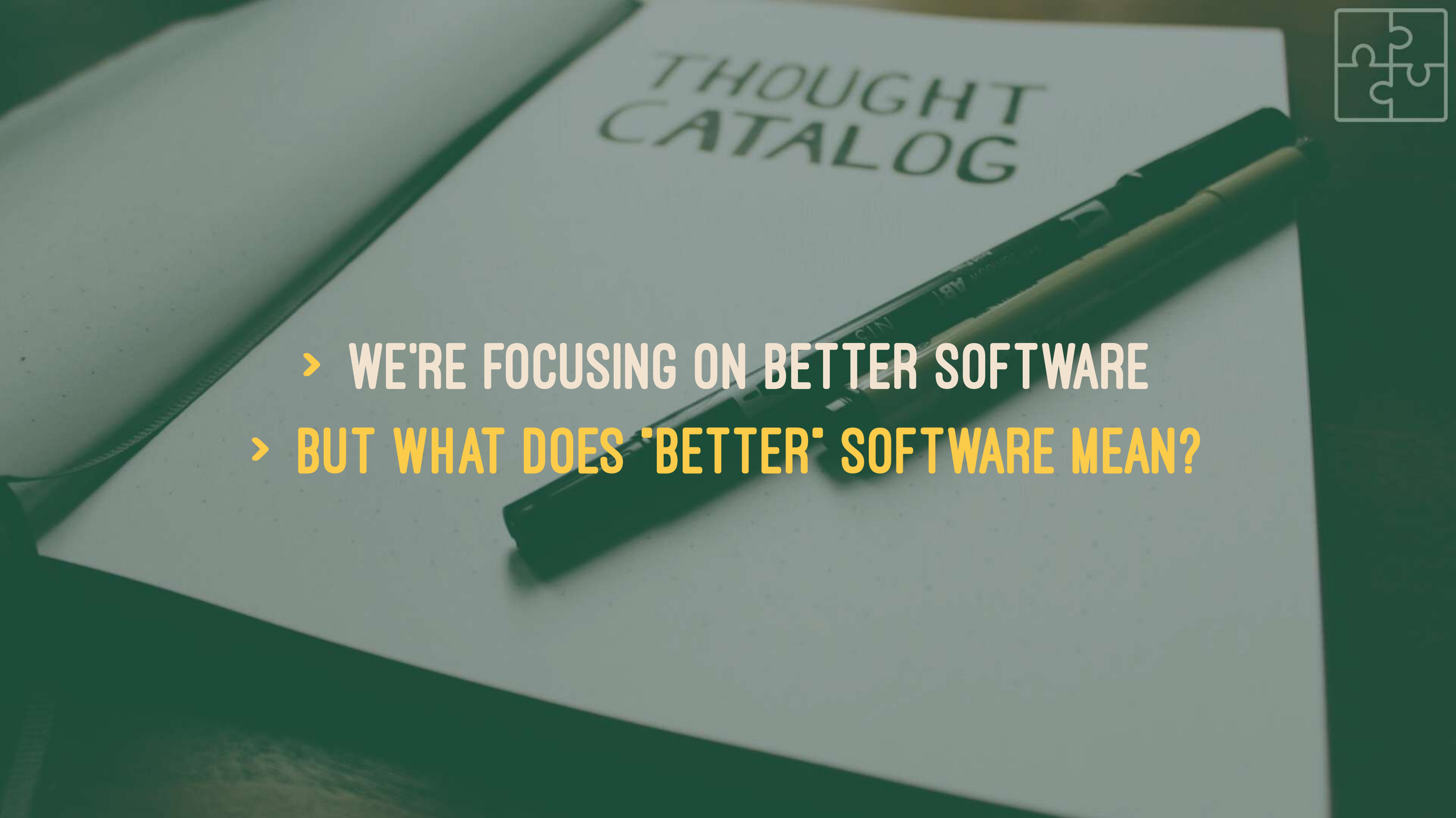
WHAT ABOUT THE WEB?


NEXT STEP
SAME RUST CODE RUNNING ON WEBASSEMBLY


Typescript
Rust
WASM
0
1
2
3
4
5
6
Sync
CB
SyncFP
Compose
CBasync
Promise
Async
Idiom
Dynamic
FP
Checked


THOUGHT
CATALOG

> WE'RE FOCUSING ON BETTER SOFTWARE



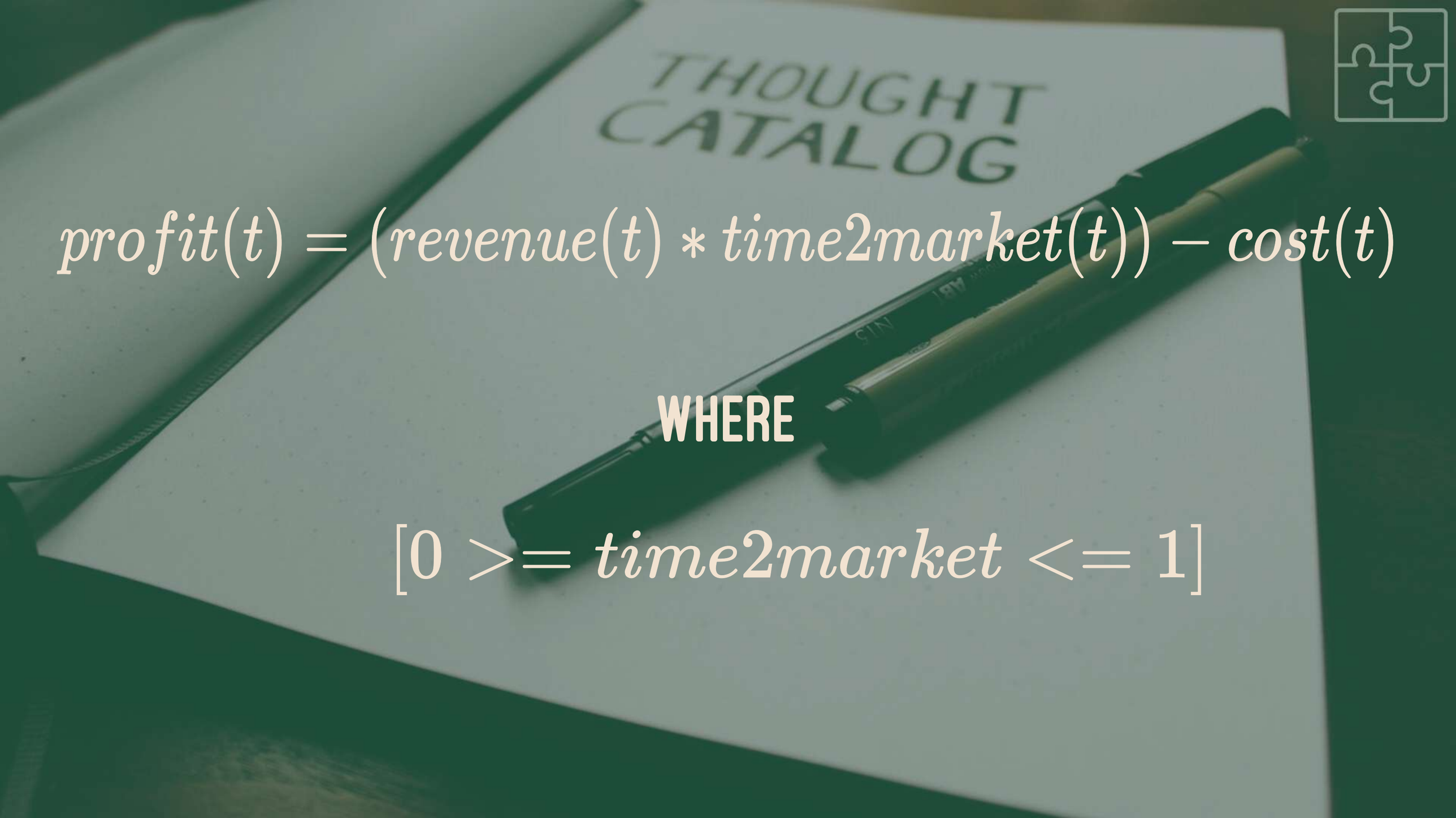
- WE'RE FOCUSING ON BETTER SOFTWARE
- BUT WHAT DOES "BETTER" SOFTWARE MEAN?

$$profit = revenue - cost$$

$$profit = (revenue * time2market) - cost$$

WHERE

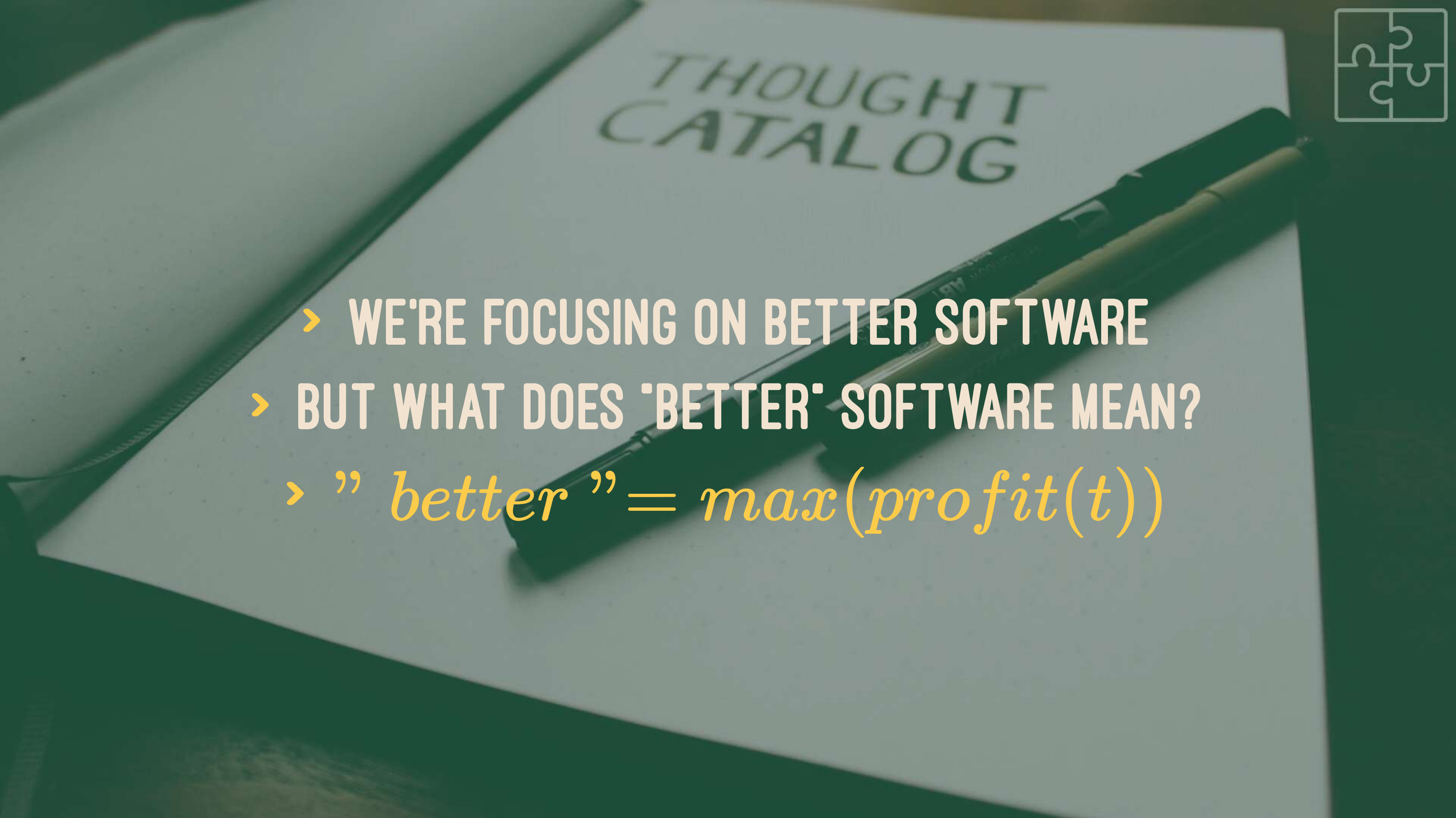
$$[0 >= time2market <= 1]$$



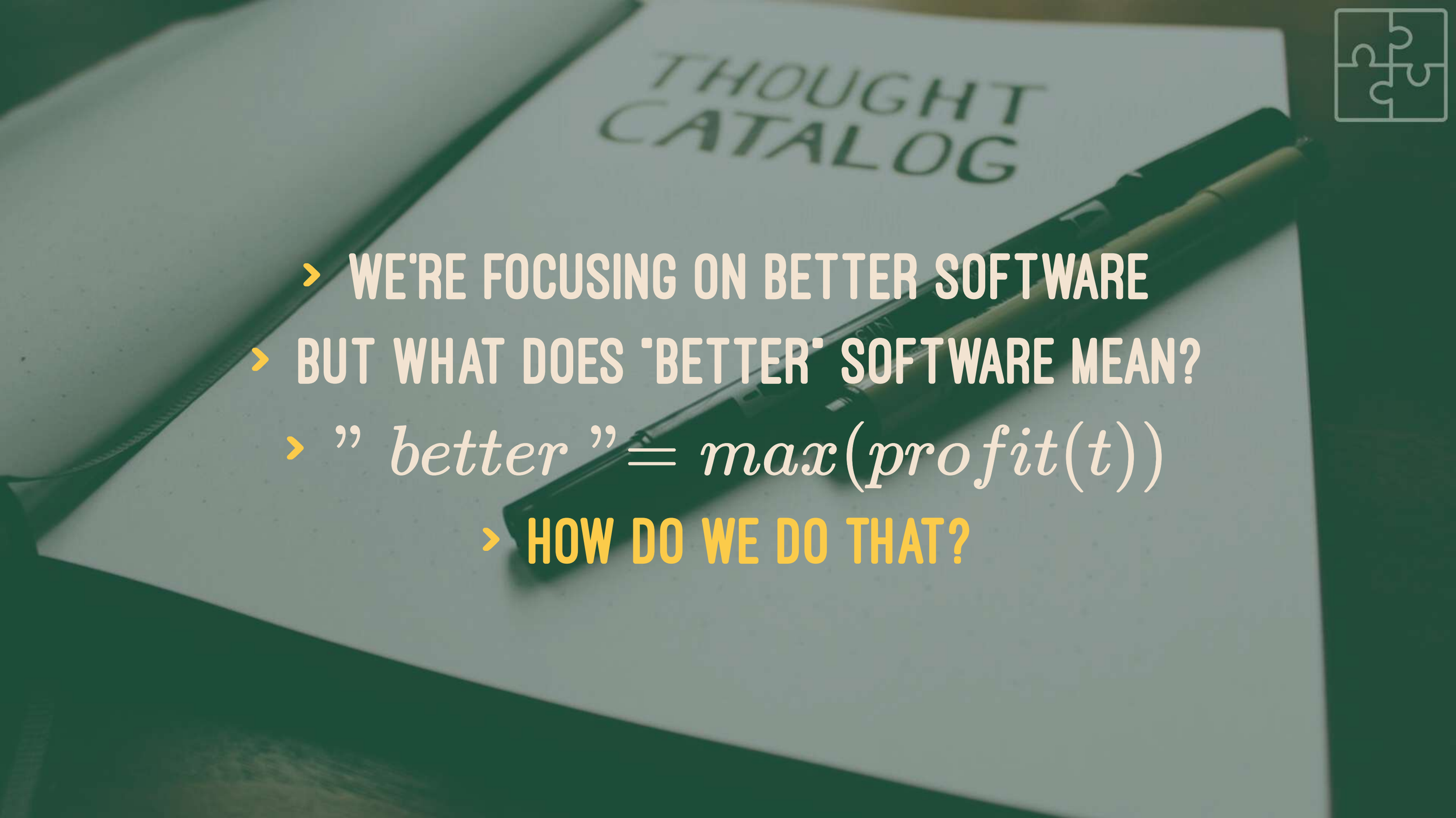
$$profit(t) = (revenue(t) * time2market(t)) - cost(t)$$

WHERE

$$[0 >= time2market <= 1]$$



- WE'RE FOCUSING ON BETTER SOFTWARE
- BUT WHAT DOES "BETTER" SOFTWARE MEAN?
  - $"better" = max(profit(t))$



- WE'RE FOCUSING ON BETTER SOFTWARE
- BUT WHAT DOES "BETTER" SOFTWARE MEAN?
- $" \ better \ " = max(profit(t))$
- HOW DO WE DO THAT?

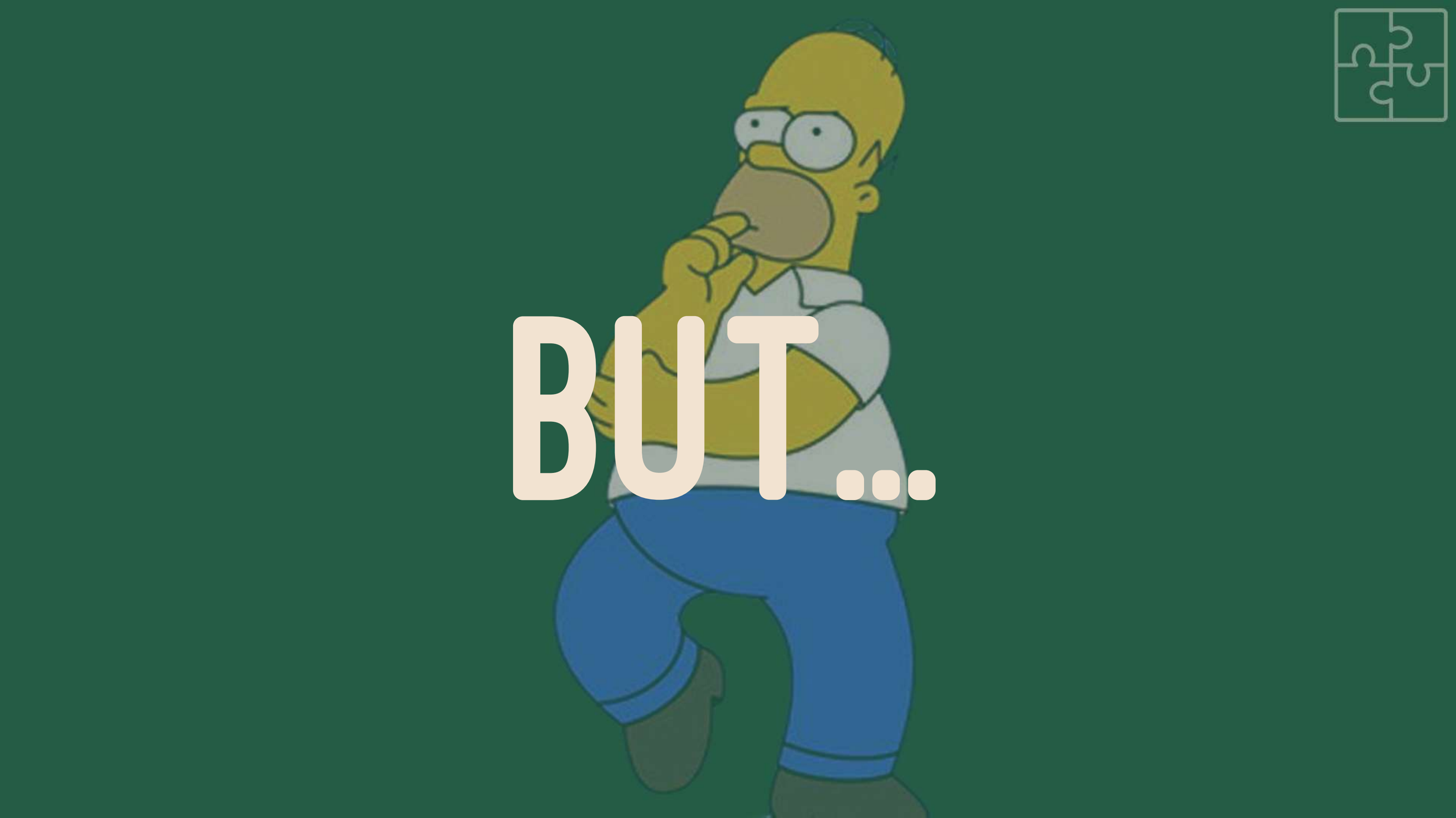
PERFORMANCE
VS
MAINTAINABILITY

DESIGN ABSTRACTION

- 👎 PERFORMANCE
- 👍 MAINTAINABILITY

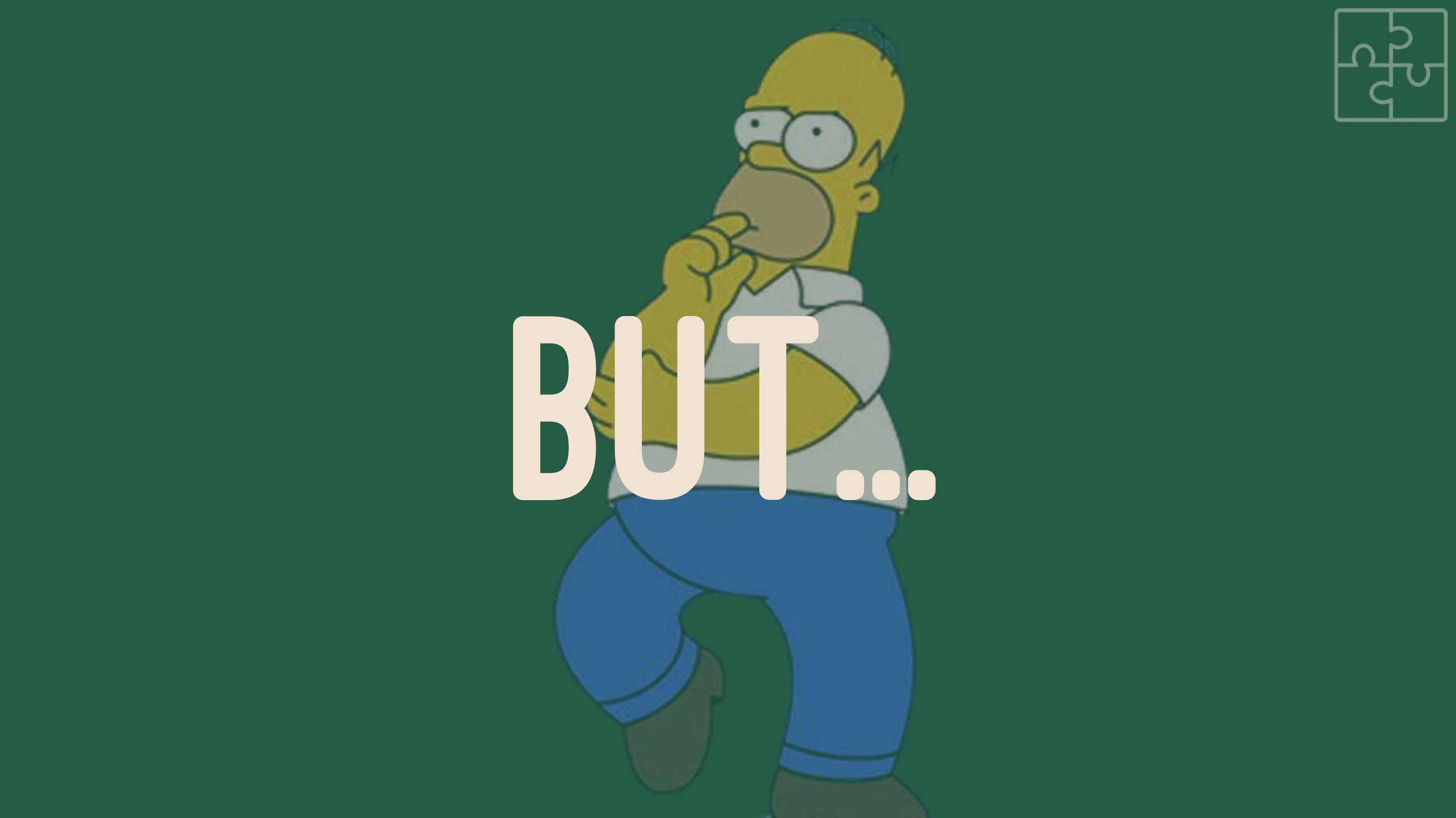
CODE OPTIMISATION ABSTRACTION

- 👍 PERFORMANCE
- 👎 MAINTAINABILITY


BUT...


CAN WE ACHIEVE BOTH PERFORMANCE
AND MAINTAINABILITY?


RUST IS THE ANSWER!
ZERO ~~COST~~ OVERHEAD ABSTRACTION


BUT...

WHAT ABOUT COGNITIVE OVERHEAD?

WRAPPING UP

THERE ARE DIFFERENT KINDS OF ABSTRACTIONS

WRAPPING UP

THERE ARE DIFFERENT KINDS OF ABSTRACTIONS
AND THERE ARE DIFFERENT KINDS OF COSTS

WRAPPING UP

THERE ARE DIFFERENT KINDS OF ABSTRACTIONS
AND THERE ARE DIFFERENT KINDS OF COSTS
DIFFERENT ABSTRACTIONS INVOLVE DIFFERENT COSTS

WRAPPING UP

THERE ARE DIFFERENT KINDS OF ABSTRACTIONS
AND THERE ARE DIFFERENT KINDS OF COSTS
DIFFERENT ABSTRACTIONS INVOLVE DIFFERENT COSTS
THERE ARE NO ZERO COST ABSTRACTIONS, BUT...

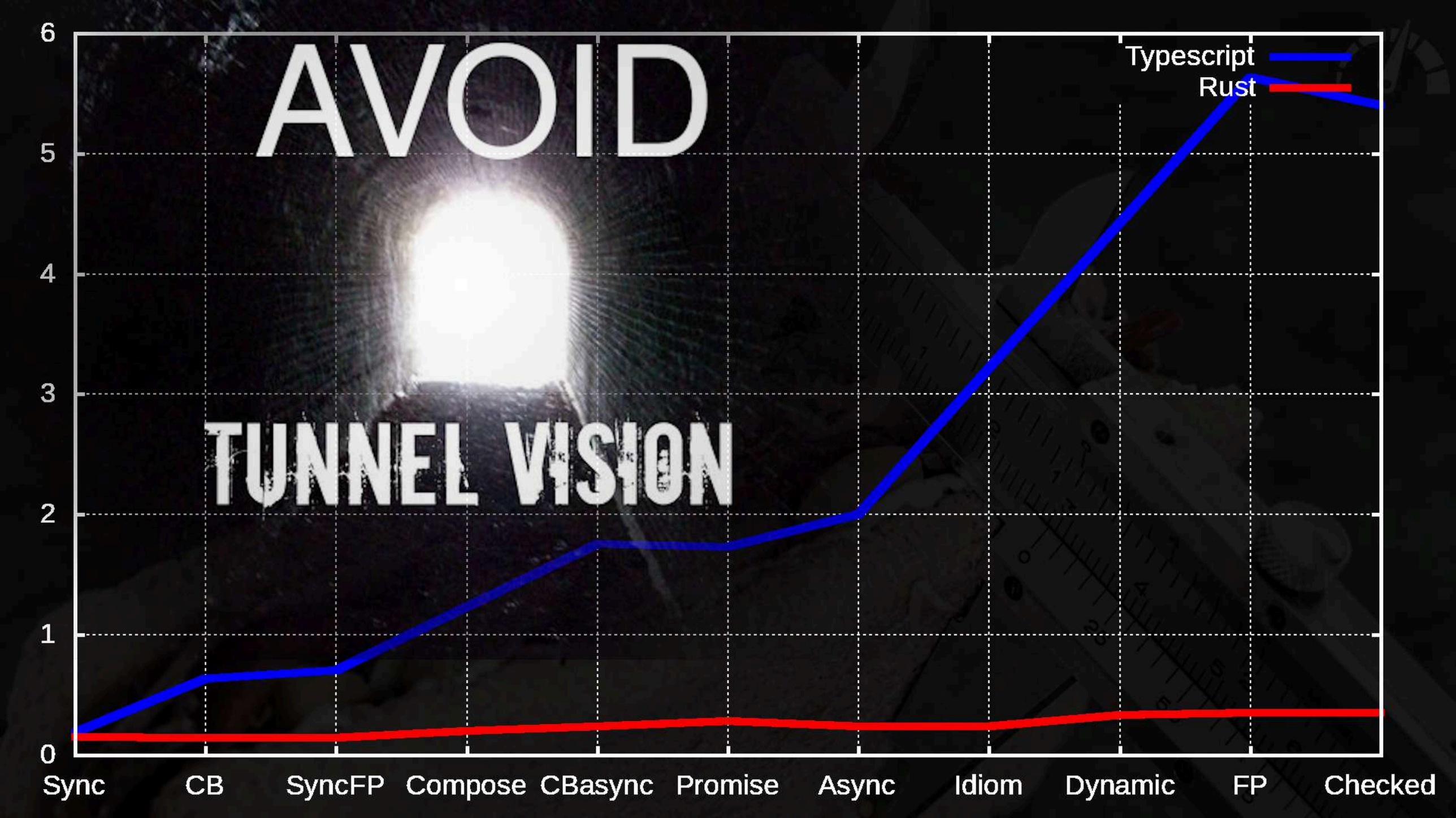
WRAPPING UP

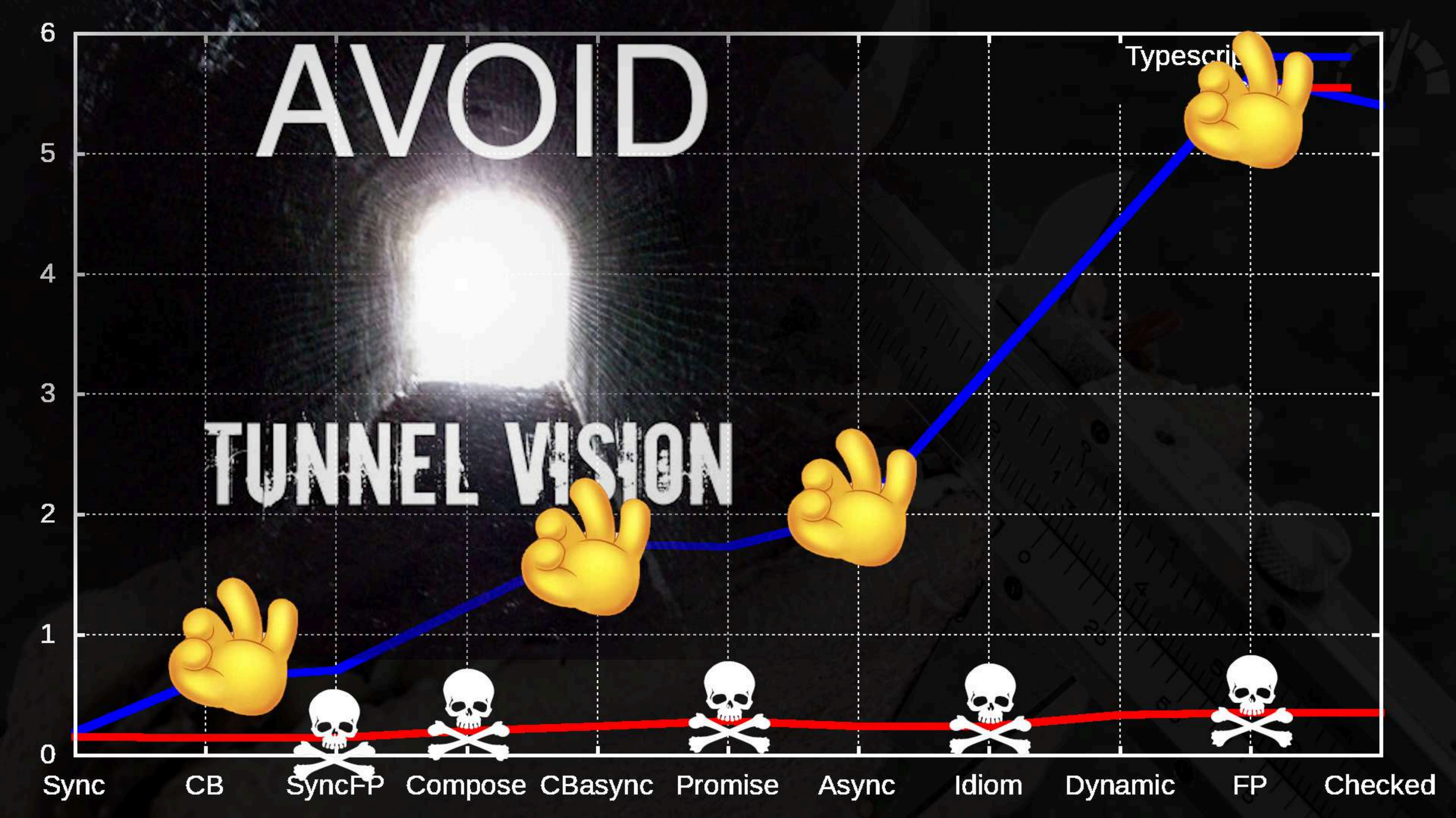
THERE ARE DIFFERENT KINDS OF ABSTRACTIONS
AND THERE ARE DIFFERENT KINDS OF COSTS
DIFFERENT ABSTRACTIONS INVOLVE DIFFERENT COSTS
THERE ARE NO ZERO COST ABSTRACTIONS, BUT…
…WE CAN CHOOSE WHERE TO INCUR COSTS!

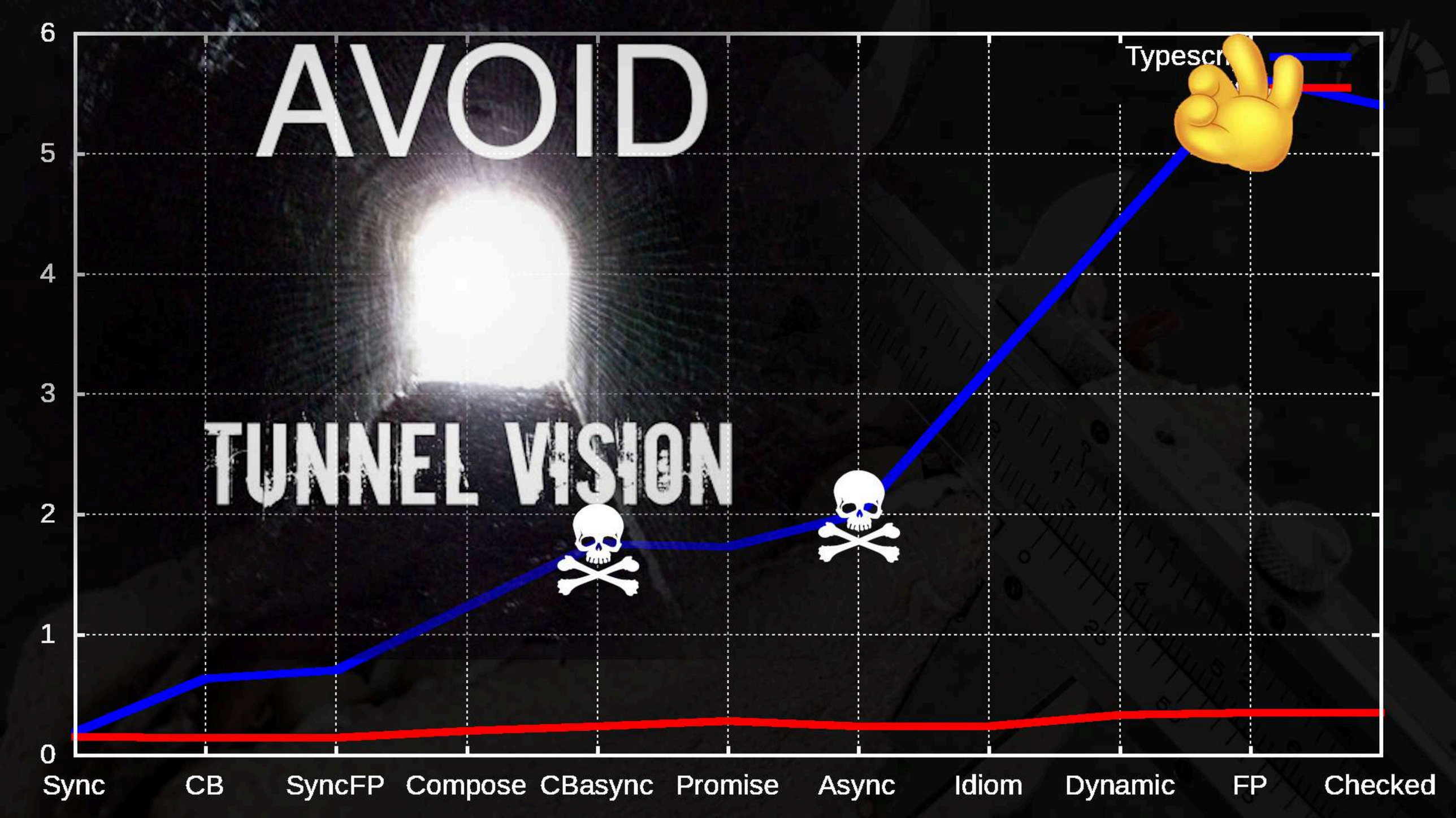
WRAPPING UP

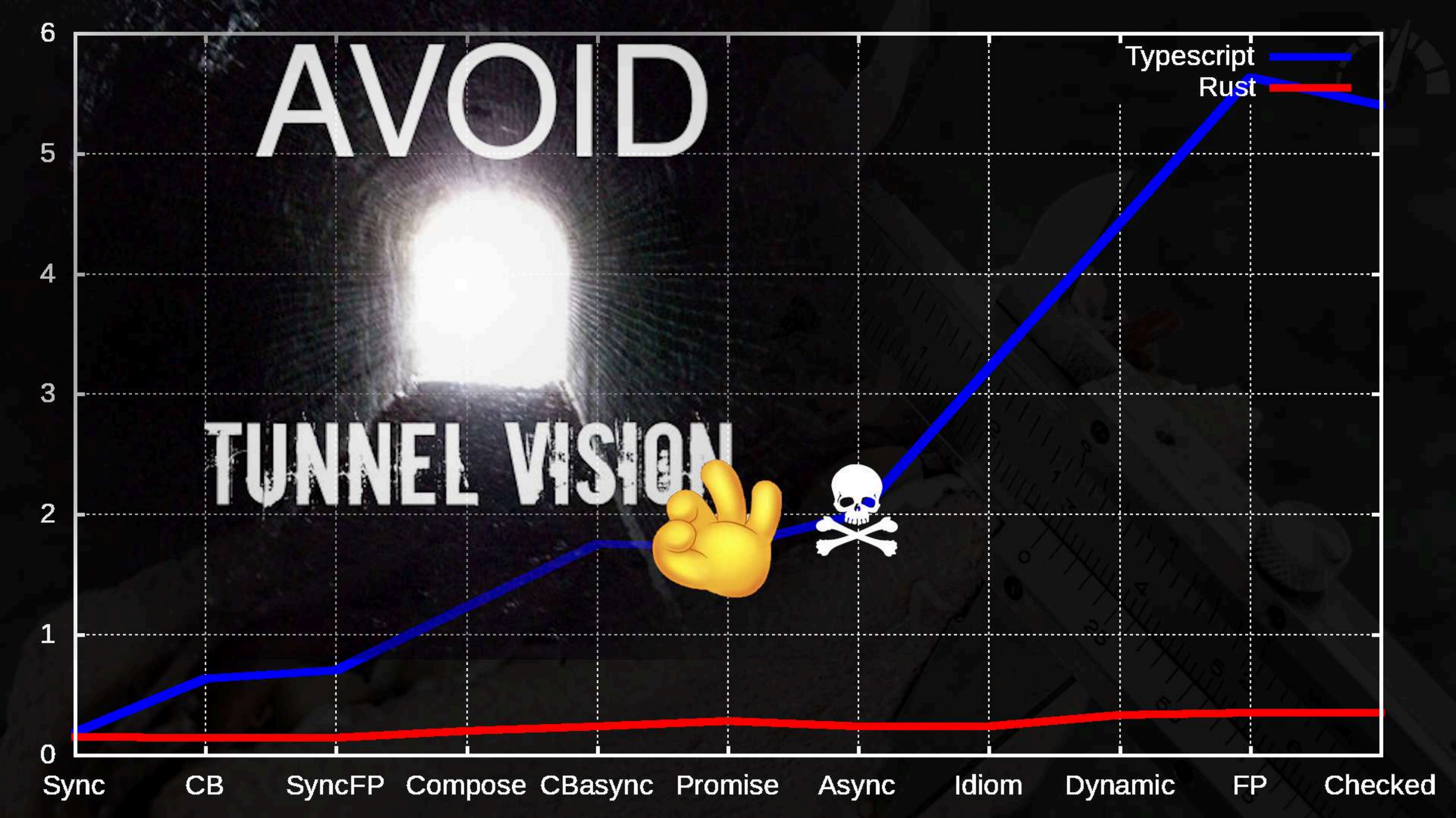
THERE ARE DIFFERENT KINDS OF ABSTRACTIONS
AND THERE ARE DIFFERENT KINDS OF COSTS
DIFFERENT ABSTRACTIONS INVOLVE DIFFERENT COSTS
THERE ARE NO ZERO COST ABSTRACTIONS, BUT…
…WE CAN CHOOSE WHERE TO INCUR COSTS!
HOW DO WE CHOOSE?

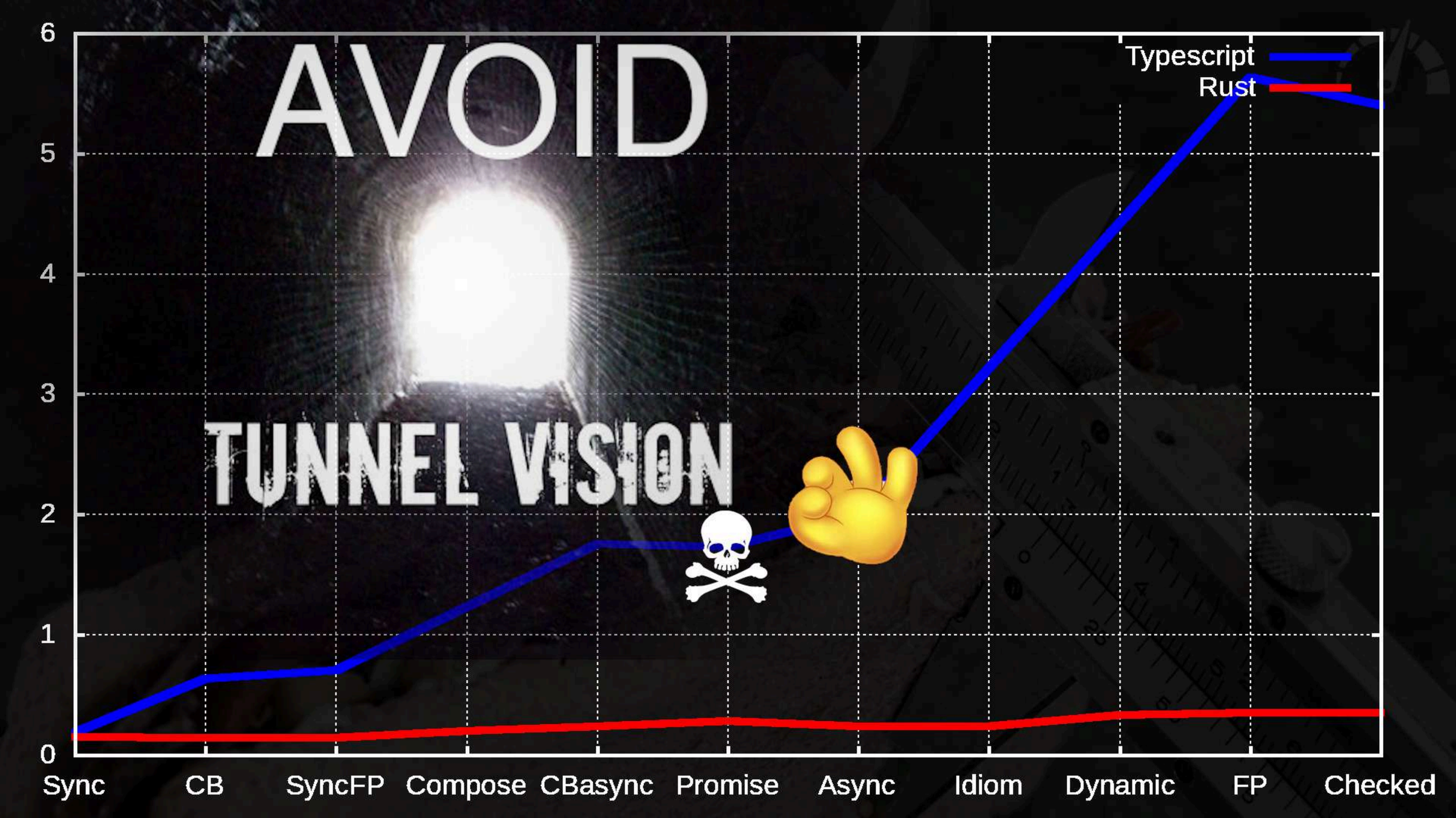

ABSTRACTION
SPACE
Typescript
Ru
6
5
4
3
2
1
0
Sync
CB
SyncFP
Compose
CBasync
Promise
Async
Idiom
Dynamic
FP
Checked


AVOID
TUNNEL VISION
Typescript
Rust
6
5
4
3
2
1
0
Sync
CB
SyncFP
Compose
CBasync
Promise
Async
Idiom
Dynamic
FP
Checked

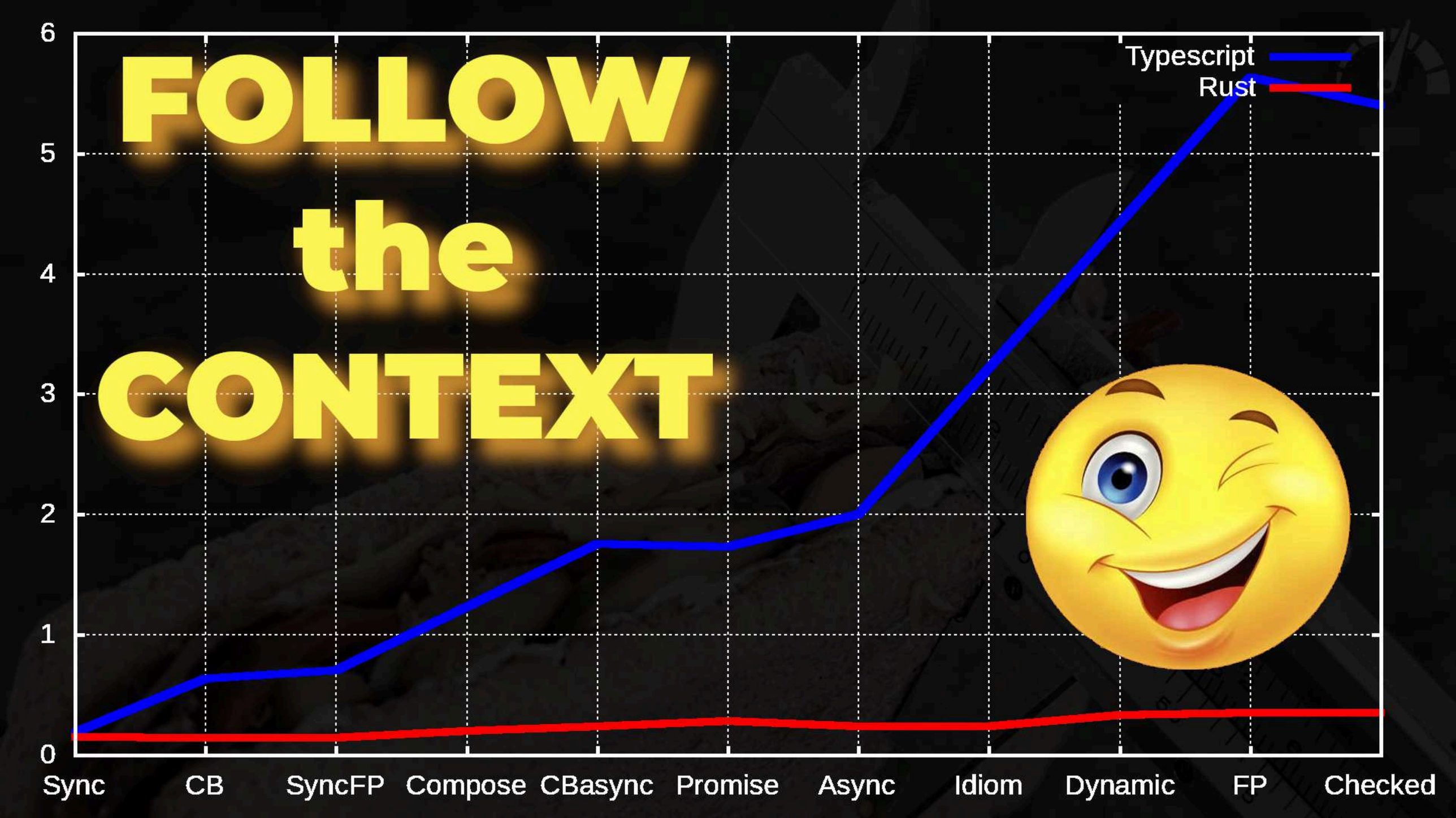

AVOID
TUNNEL VISION
Typescrip
6
5
4
3
2
1
0
Sync
CB
SyncFP
Compose
CBasync
Promise
Async
Idiom
Dynamic
FP
Checked

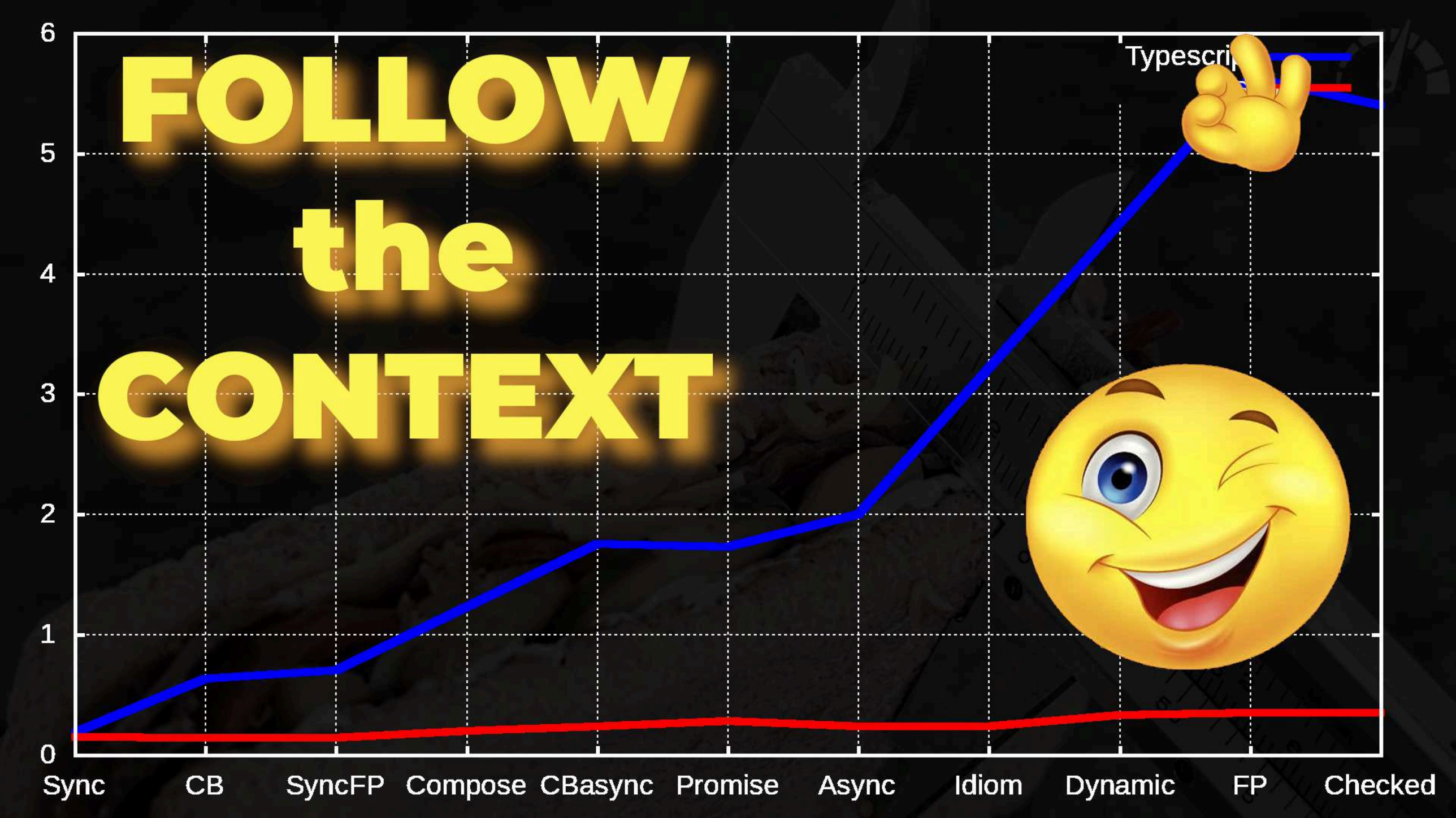

AVOID
TUNNEL VISION
Typescr
6
5
4
3
2
1
0
Sync
CB
SyncFP
Compose
CBasync
Promise
Async
Idiom
Dynamic
FP
Checked

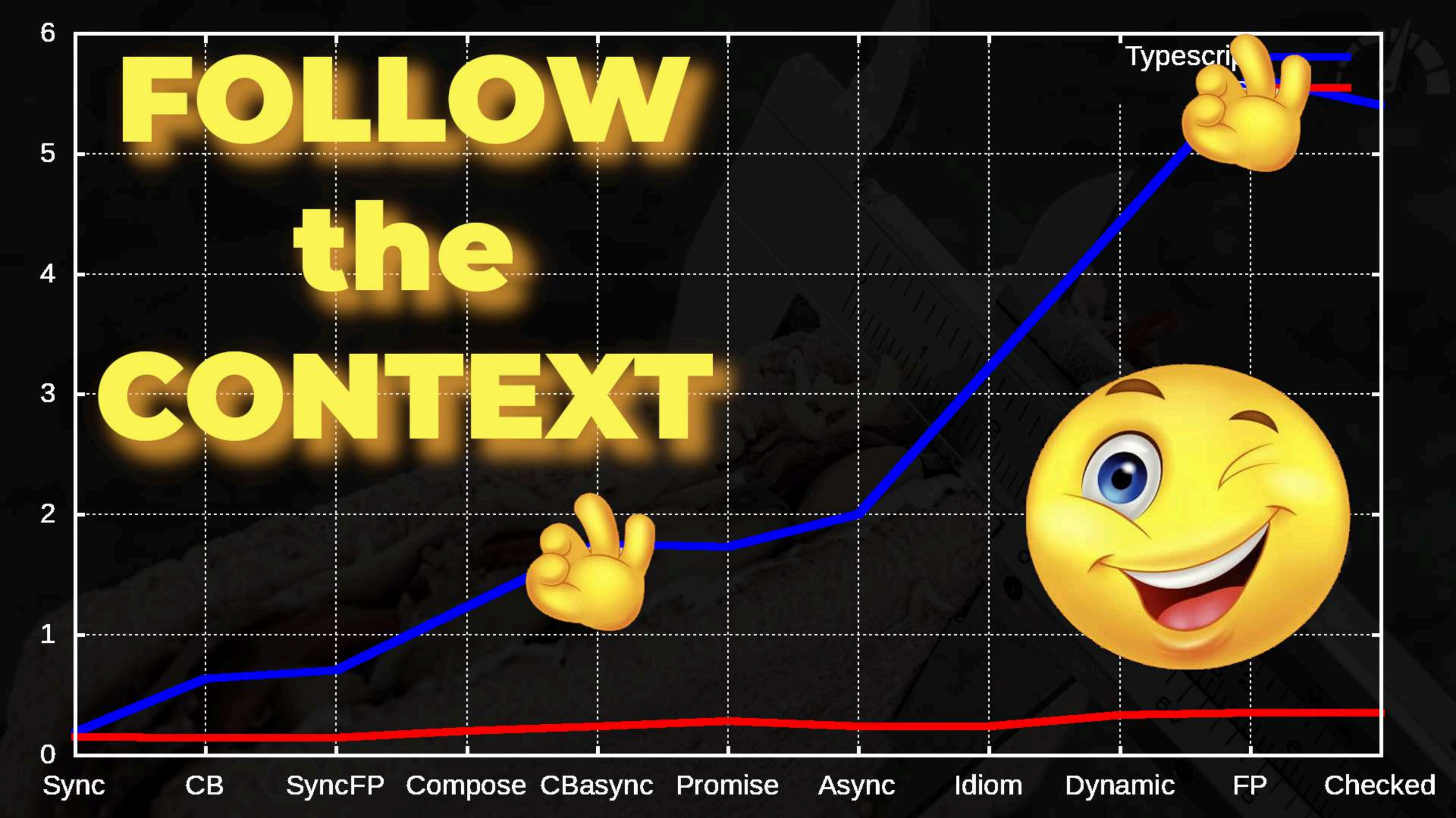

AVOID
TUNNEL VISION
Typescript
Rust
6
5
4
3
2
1
0
Sync
CB
SyncFP
Compose
CBasync
Promise
Async
Idiom
Dynamic
FP
Checked

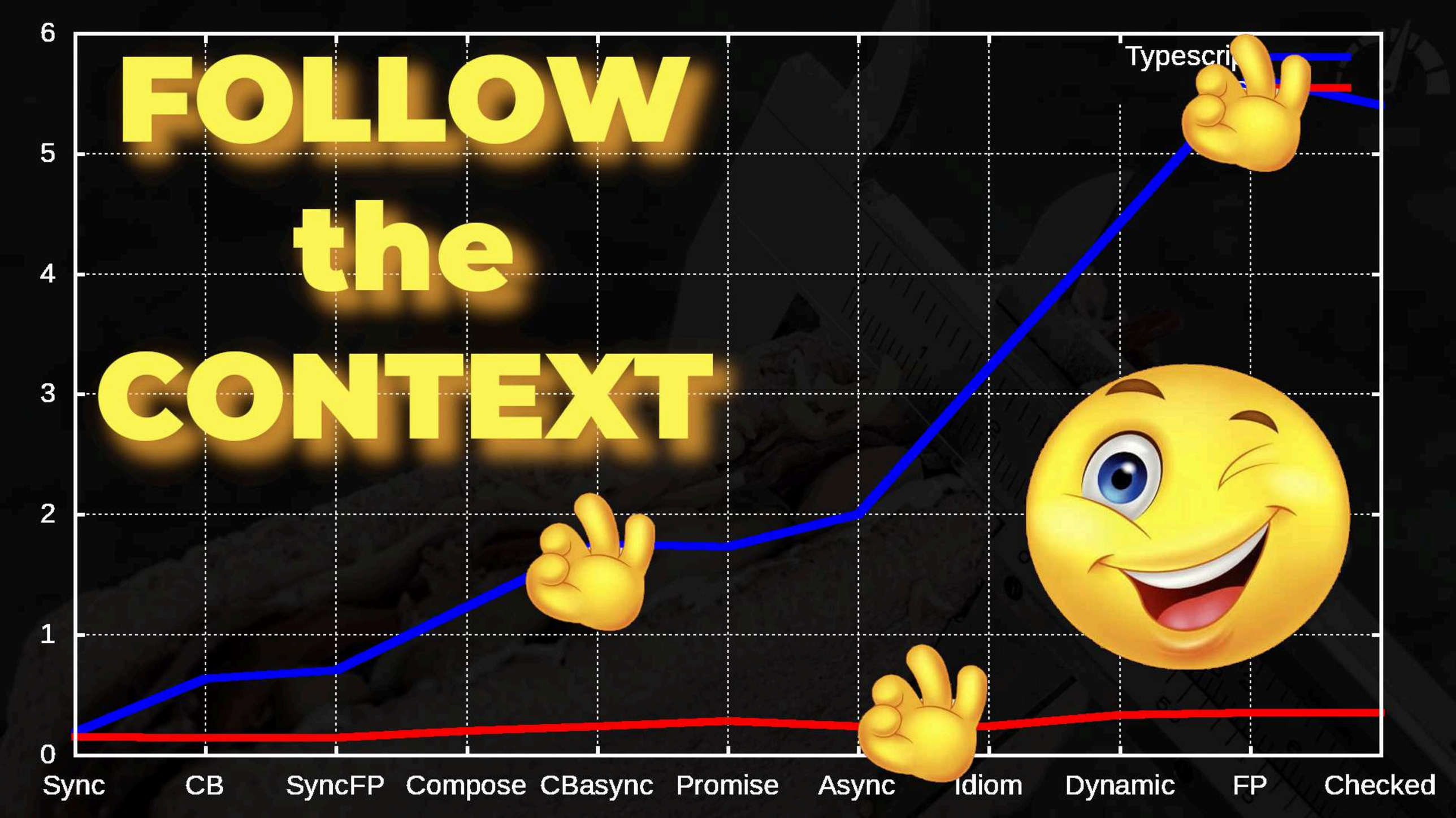

AVOID
TUNNEL VISION
Typescript
Rust
6
5
4
3
2
1
0
Sync
CB
SyncFP
Compose
CBasync
Promise
Async
Idiom
Dynamic
FP
Checked

FOLLOW the CONTEXT




FOLLOW
the
CONTEXT
Typescri
6
5
4
3
2
1
0
Sync
CB
SyncFP
Compose
CBasync
Promise
Async
Idiom
Dynamic
FP
Checked


FOLLOW
the
CONTEXT
Typescri
6
5
4
3
2
1
0
Sync
CB
SyncFP
Compose
CBasync
Promise
Async
Idiom
Dynamic
FP
Checked

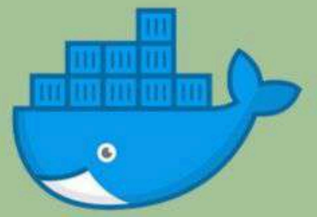

FOLLOW
the
CONTEXT
Typescri
6
5
4
3
2
1
0
Sync
CB
SyncFP
Compose
CBasync
Promise
Async
Idiom
Dynamic
FP
Checked


Thank
you!


QUESTIONS?

slides

code



environment

http://tsconf2019.carucci.org/