Hello!

I love TypeScript!

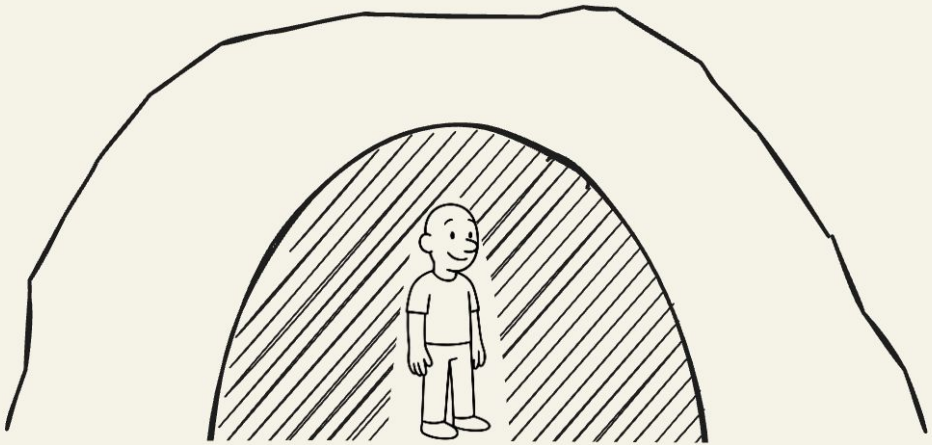

RELEASE
ME
TS
ME
RELEASE
imgflip.com

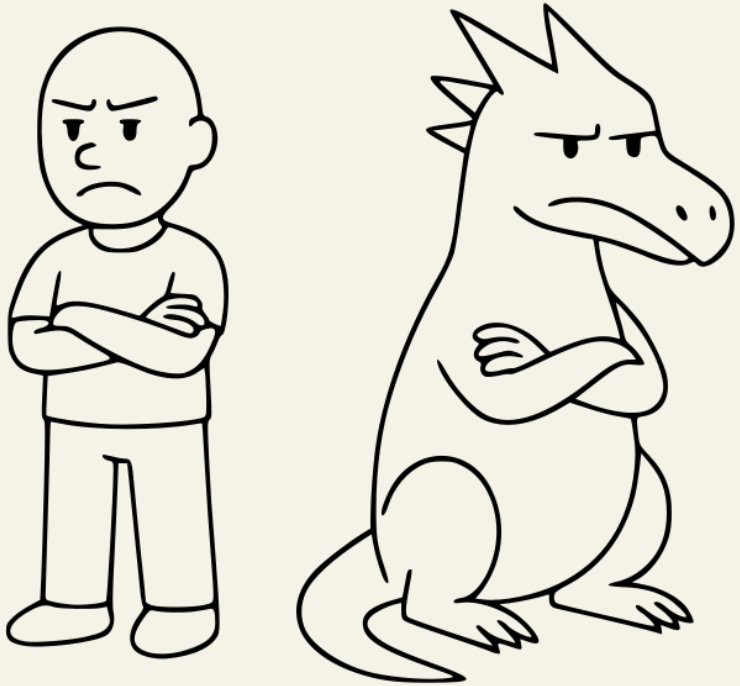
`// @ts-ignore`

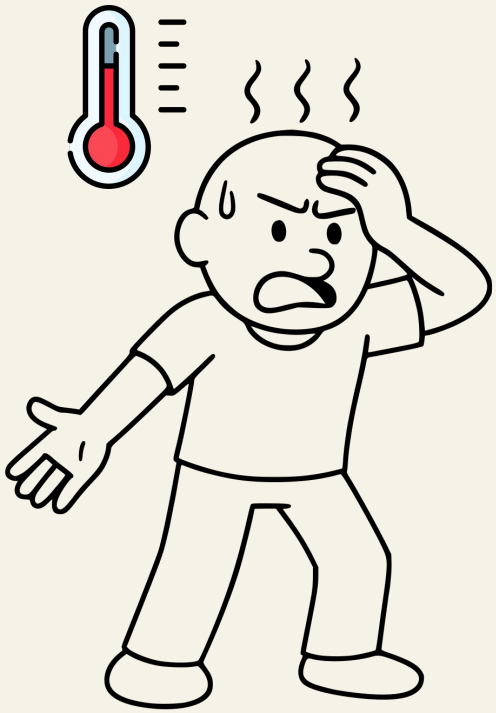
`any`

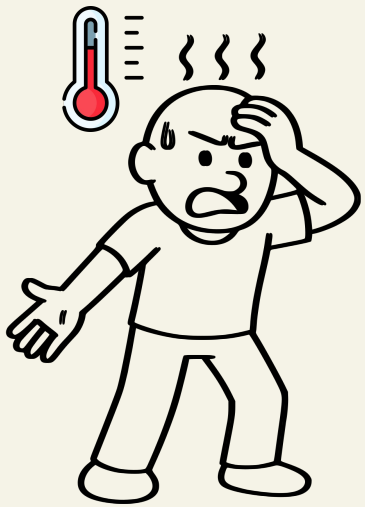
`as Type`

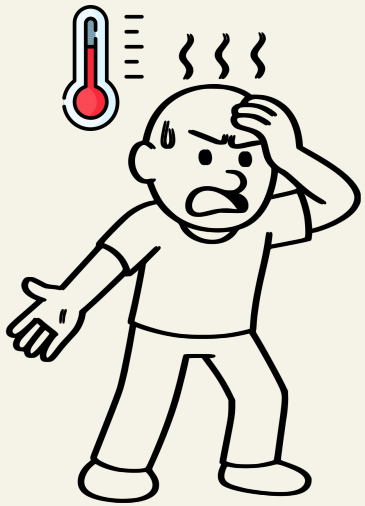
Let’s start with a story
about perspective



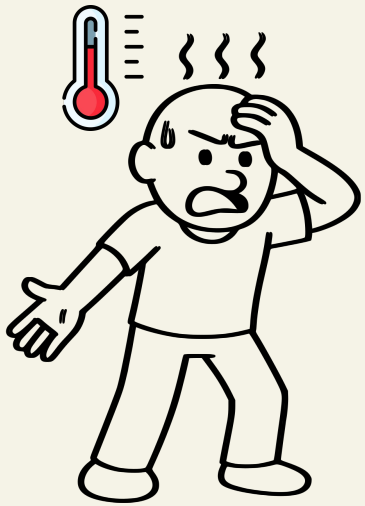




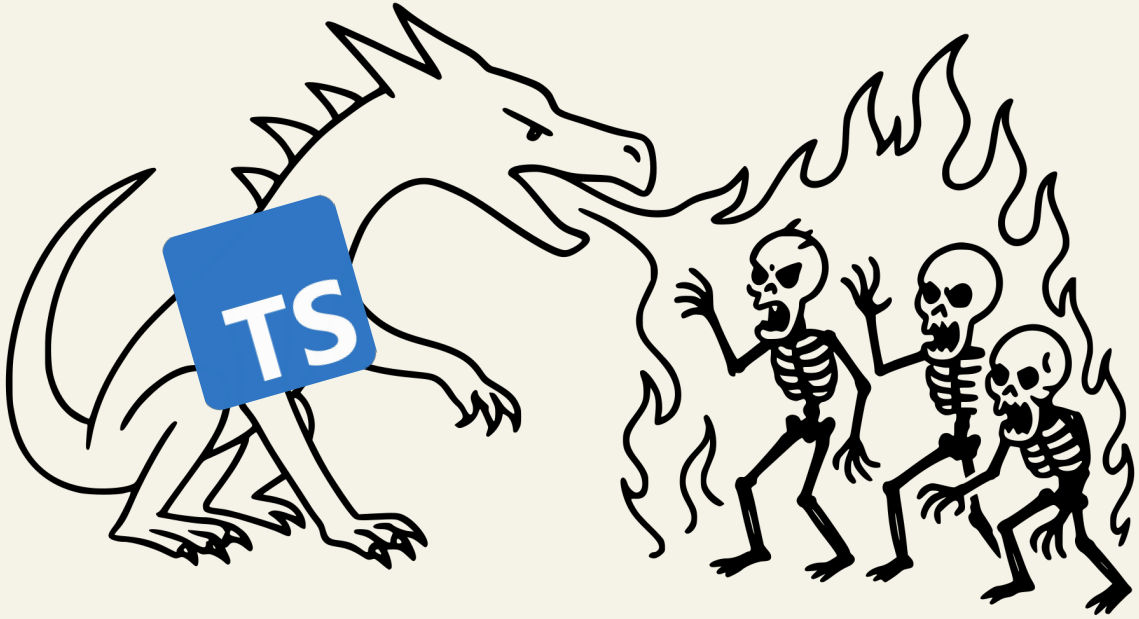


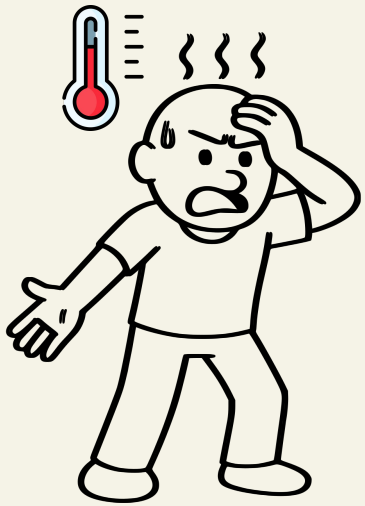


You

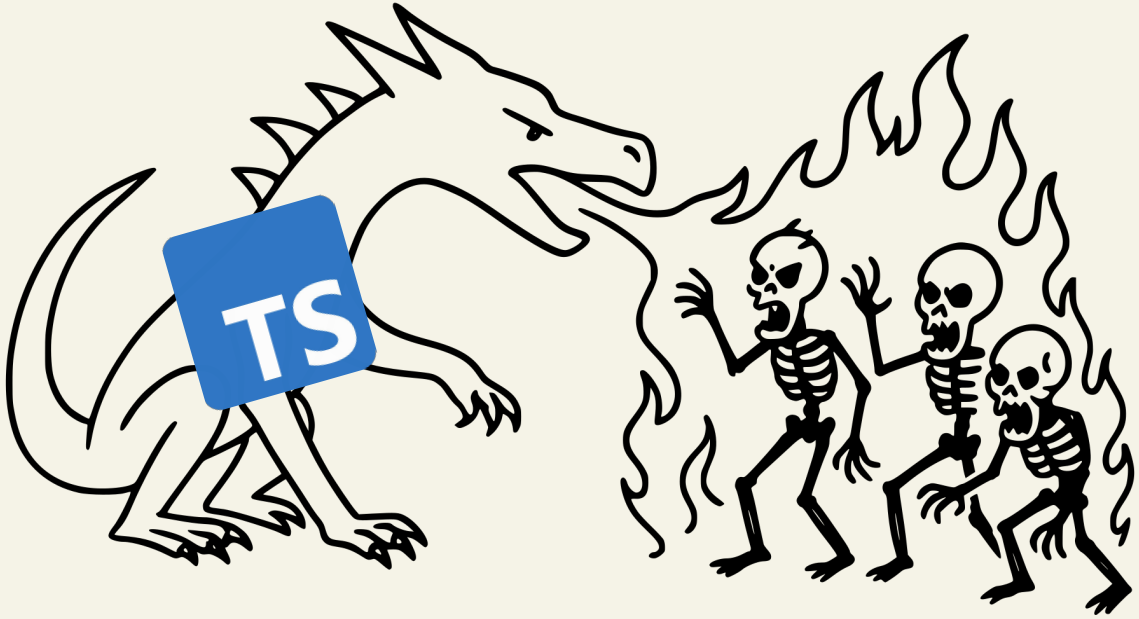


You



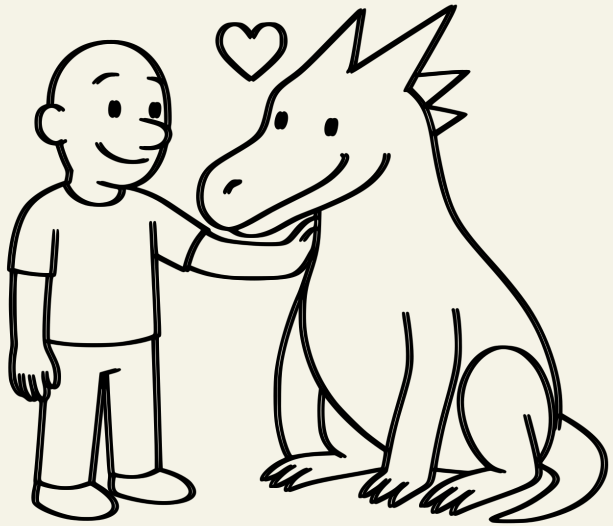


You



Today’s topic

Treat the Compiler With Compassion



Filip Sodic

Working with compilers at Wasp and for fun
Teaching Haskell at UZG

sodic.dev

@filipsodic

@WaspLang

Wasp

TypeScript

Haskell

Filip Sodic

Working with compilers at Wasp and for fun
Teaching Haskell at UZG

sodic.dev

@filipsodic

@WaspLang

Wasp

TypeScript

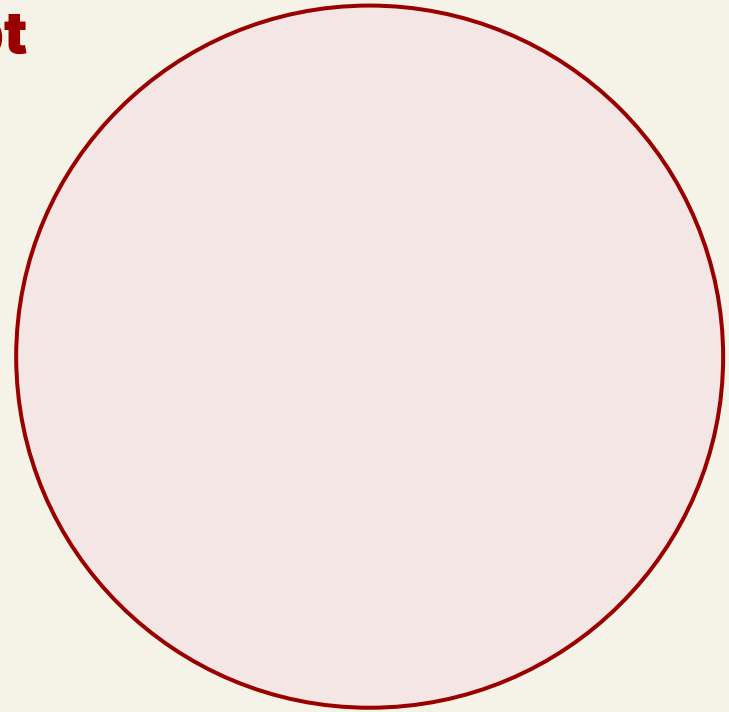
Haskell

Why do we get frustrated?

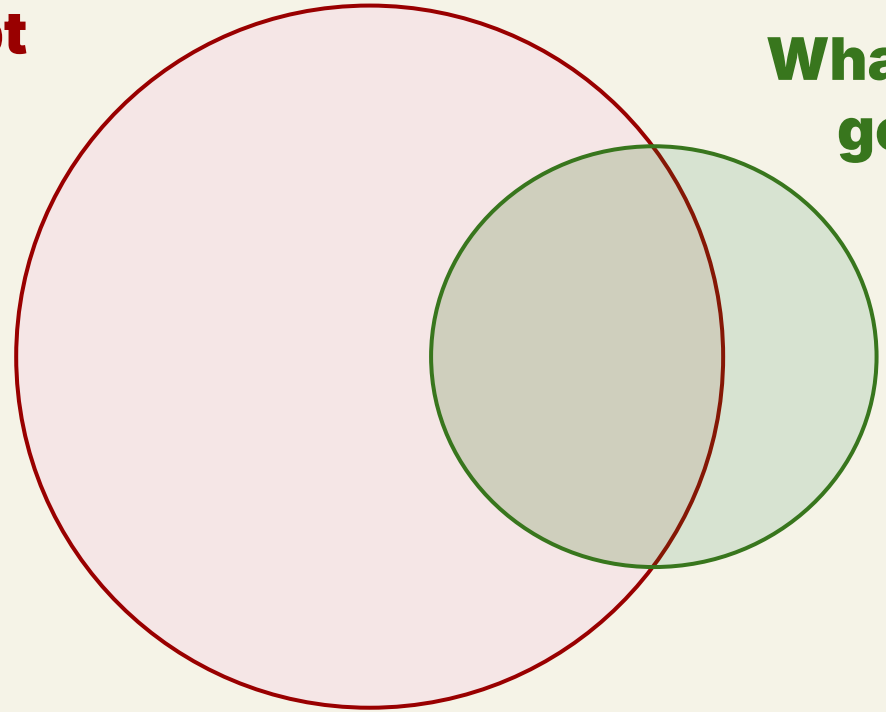
Why do we get frustrated?

...Because we don’t get what we want

What we want from TypeScript

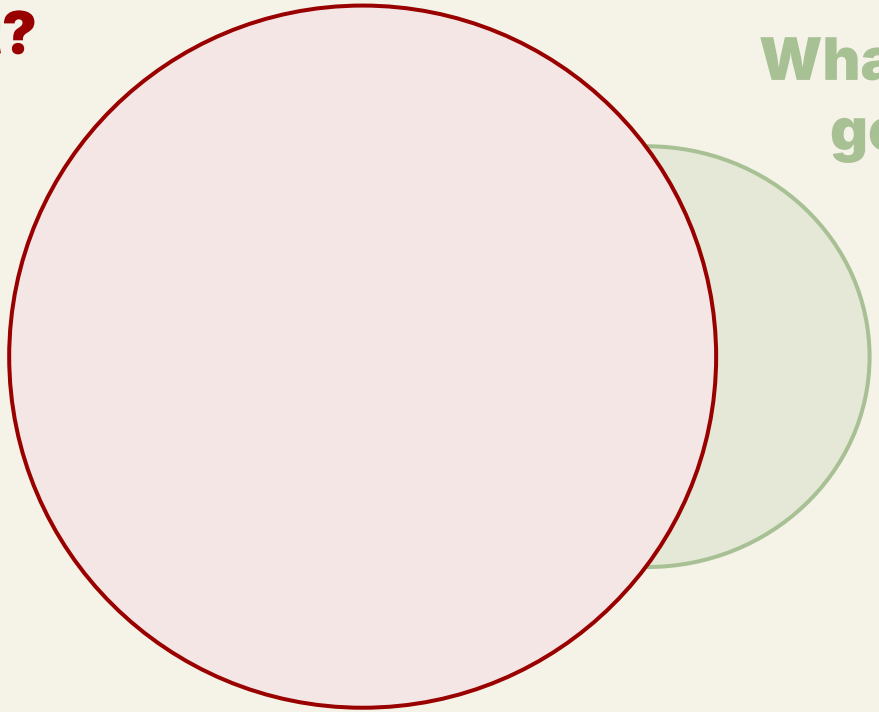


What we want from TypeScript



What's actually good for us

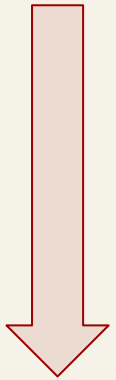
What *do* we want from TypeScript?



What's actually good for us

```
const conference = {
  name: 'JSHeroes',
  month: 'May',
}

console.log(`${conference.name} is in ${conference.moth}`)
```

```
const conference = {
  name: 'JSHeroes',
  month: 'May',
}

console.log(`${conference.name} is in ${conference.moth}`)
```



```
const conference = {
  name: 'JSHeroes',
  month: 'May',
}

console.log(`${conference.name} is in ${conference.moth}`)
```

ERROR: Property 'moon' does not exist on type '{ name: string; month: string; }'.(2339)

```
const conference = {
  name: 'JSHeroes',
  month: 'May',
}

console.log(`${conference.name} is in ${conference.month}`)
```

ERROR: Property 'moon' does not exist on type '{ name: string; month: string; }'.(2339)

```
const conference = {
  name: 'JSHeroes',
  month: 'May',
}

console.log(`${conference.name} is in ${conference.month}`)
```

What do we want from TypeScript?

What do we want from TypeScript?

1. If the code has type errors, report them
2. Don’t report errors that don’t exist

What do we want from TypeScript?

1. If the code has type errors, report them
2. Don’t report errors that don’t exist
3. **Don’t crash or hang**

What do we want from TypeScript?

1. If the code has type errors, report them
2. Don’t report errors that don’t exist
3. Don’t crash or hang

What do we want from TypeScript?

1. If the code has type errors, always report them
2. Don’t report errors that don’t exist
3. Don’t crash or hang

What do we want from TypeScript?

1. If the code has type errors, **always** report them
2. **Never** report errors that don’t exist
3. Don’t crash or hang

What do we want from TypeScript?

1. If the code has type errors, **always** report them
2. **Never** report errors that don't exist
3. **Never** crash or hang

What do we want from TypeScript?

1. If the code has type errors, always report them
2. Never report errors that don't exist
3. Never crash or hang

Sound

What do we want from TypeScript?

1. If the code has type errors, always report them
2. **Never** report errors that don't exist
3. Never crash or hang

Complete

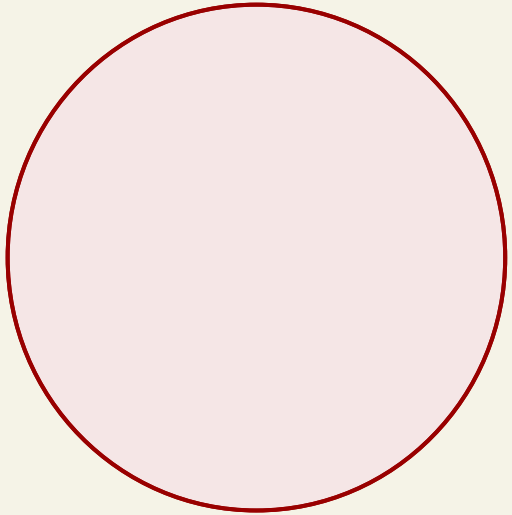
What do we want from TypeScript?

1. If the code has type errors, always report them
2. Never report errors that don't exist
3. Never crash or hang

Decidable

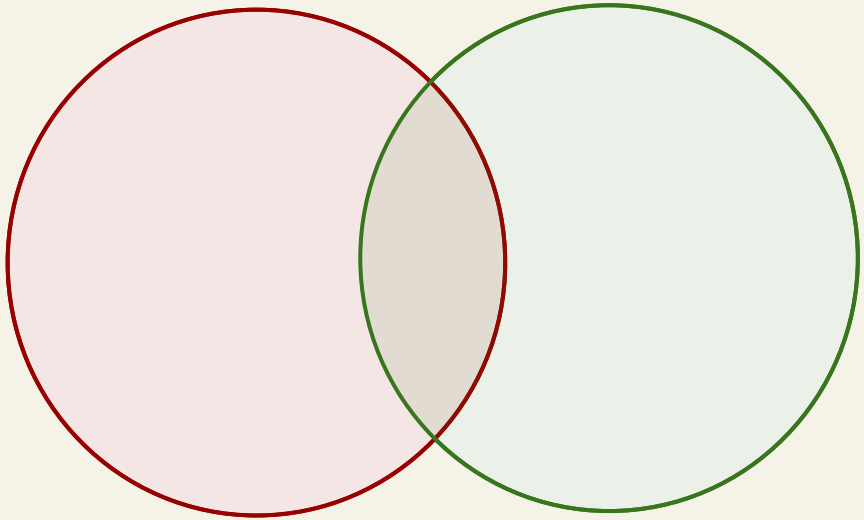
Sound

If the code has type errors,
always report them



Sound

If the code has type errors,
always report them

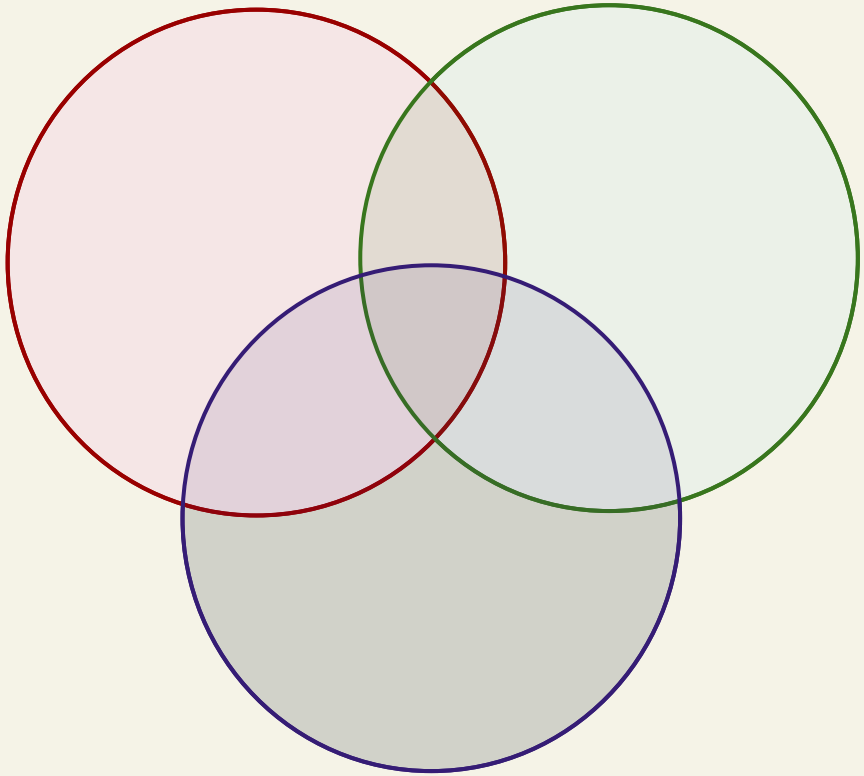


Complete

Never report errors
that don't exist

Sound

If the code has type errors,
always report them



Complete

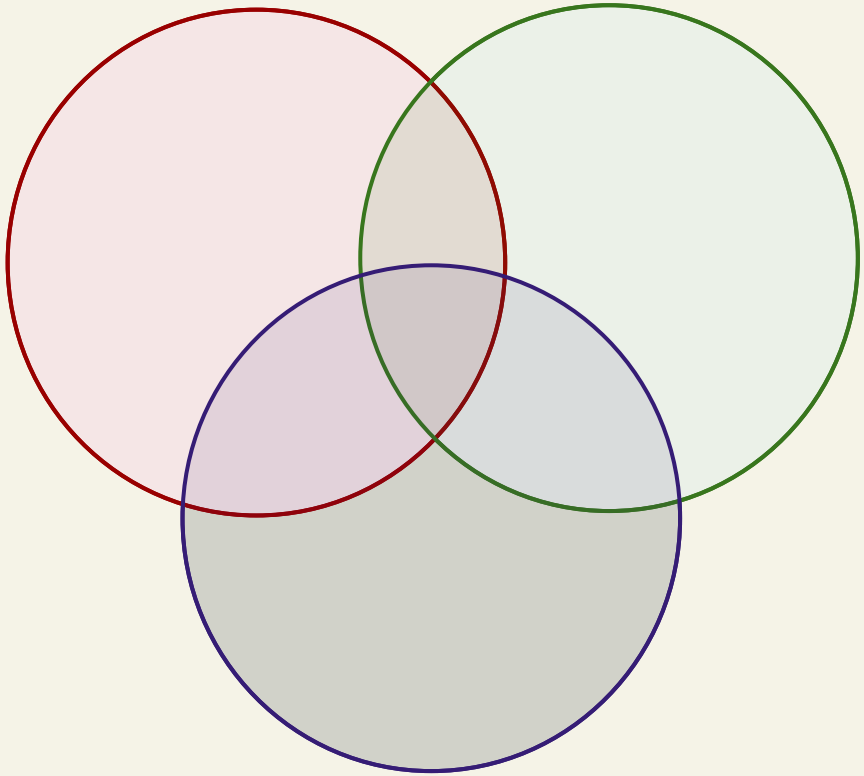
Never report errors
that don't exist

Decidable

Never crash or hang

Sound

If the code has type errors,
always report them



Complete

Never report errors
that don't exist

Decidable

Never crash or hang

What TypeScript actually does

1. If the code has type errors, **always** reports them
2. **Never** reports errors that don’t exist
3. **Never** crashes or hangs

What TypeScript actually does

1. If the code has type errors, often ~~always~~ reports them
2. **Never** reports errors that don't exist
3. **Never** crashes or hangs

What TypeScript actually does

1. If the code has type errors, ~~always~~ often reports them
2. ~~Never~~ Sometimes reports errors that don't exist
3. Never crashes or hangs

What TypeScript actually does

1. If the code has type errors, ~~always~~ often reports them
2. ~~Never~~ Sometimes reports errors that don't exist
3. ~~Never~~ Sometimes crashes or hangs

Frustrations happen because TypeScript is...

Frustrations happen because TypeScript is...

Why didn't you catch this?

Frustrations happen because TypeScript is...

Not sound

Why didn't you catch this?

Frustrations happen because TypeScript is...

Not sound

Why didn't you catch this?

This works... why are you bullying me!?

Frustrations happen because TypeScript is...

Not sound

Why didn't you catch this?

Not complete

This works... why are you bullying me!?

Frustrations happen because TypeScript is...

Not sound

Why didn't you catch this?

Not complete

This works... why are you bullying me!?

Mom, the compiler is broken!

Frustrations happen because TypeScript is...

Not sound

Why didn't you catch this?

Not complete

This works... why are you bullying me!?

Not decidable

Mom, the compiler is broken!

Frustrations happen because TypeScript is...

Not sound

Why didn't you catch this?

Not complete

This works... why are you bullying me!?

Frustrations happen because TypeScript is...

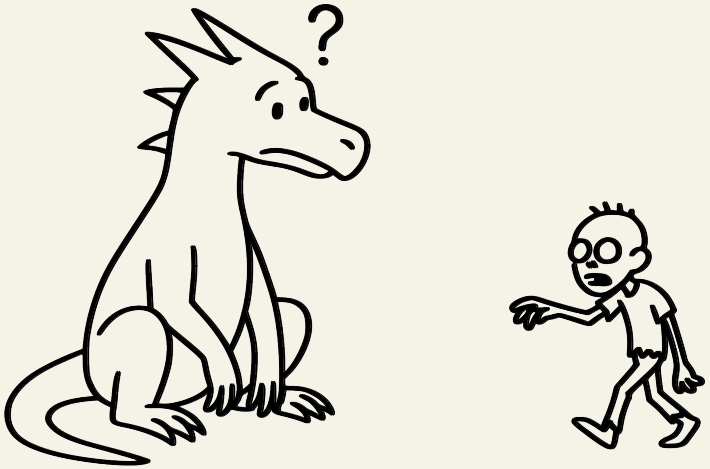
Not sound

Why didn't you catch this?

Not complete

This works... why are you bullying me!?

Why is TypeScript not sound ?



Why didn’t you catch this?

Is it an accident?

Typescript is not sound on purpose!

Non-goals

1. Exactly mimic the design of existing languages. Instead, use the behavior of JavaScript and the intentions of program authors as a guide for what makes the most sense in the language.
2. Aggressively optimize the runtime performance of programs. Instead, emit idiomatic JavaScript code that plays well with the performance characteristics of runtime platforms.
3. Apply a sound or "provably correct" type system. Instead, strike a balance between correctness and productivity.
4. Provide an end-to-end build pipeline. Instead, make the system extensible so that external tools can use the compiler for more complex build workflows.
5. Add or rely on run-time type information in programs, or emit different code based on the results of the type system. Instead, encourage programming patterns that do not require run-time metadata.
6. Provide additional runtime functionality or libraries. Instead, use TypeScript to describe existing libraries.
7. Introduce behaviour that is likely to surprise users. Instead have due consideration for patterns adopted by other commonly-used languages.

Source

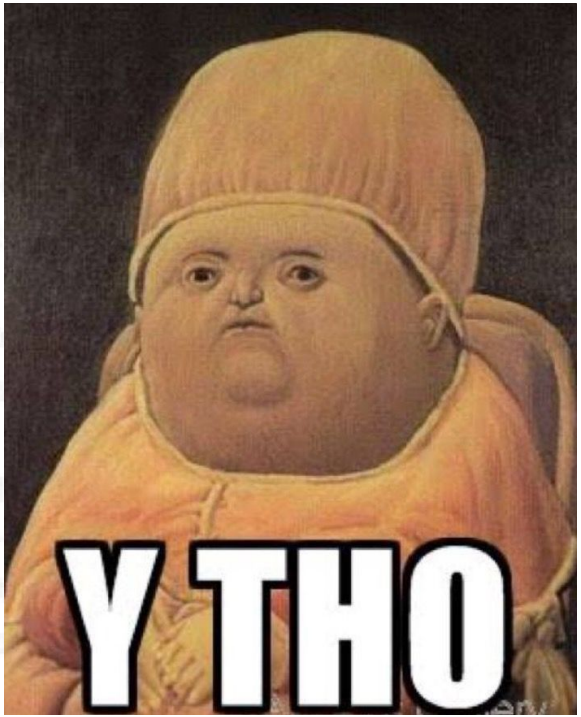
Non-goals

1. Exactly mimic the design of existing languages. Instead, use the behavior of JavaScript and the intentions of program authors as a guide for what makes the most sense in the language.
2. Aggressively optimize the runtime performance of programs. Instead, emit idiomatic JavaScript code that plays well with the performance characteristics of runtime platforms.
3. Apply a sound or "provably correct" type system. Instead, strike a balance between correctness and productivity.
4. Provide an end-to-end build pipeline. Instead, make the system extensible so that external tools can use the compiler for more complex build workflows.
5. Add or rely on run-time type information in programs, or emit different code based on the results of the type system. Instead, encourage programming patterns that do not require run-time metadata.
6. Provide additional runtime functionality or libraries. Instead, use TypeScript to describe existing libraries.
7. Introduce behaviour that is likely to surprise users. Instead have due consideration for patterns adopted by other commonly-used languages.

Source

Non-goals

1. Exactly mimic the design of existing languages. Instead, use the behavior o
   program authors as a guide for what makes the most sense in the language
2. Aggressively optimize the runtime performance of programs. Instead, emit
   well with the performance characteristics of runtime platforms.
3. Apply a sound or "provably correct" type system. Instead, strike a balance t
4. Provide an end-to-end build pipeline. Instead, make the system extensible
   compiler for more complex build workflows.
5. Add or rely on run-time type information in programs, or emit different cod
   system. Instead, encourage programming patterns that do not require run
6. Provide additional runtime functionality or libraries. Instead, use TypeScri
7. Introduce behaviour that is likely to surprise users. Instead have due consi
   commonly-used languages.



Source

It’s a tradeoff...

```
interface Country {



}
```

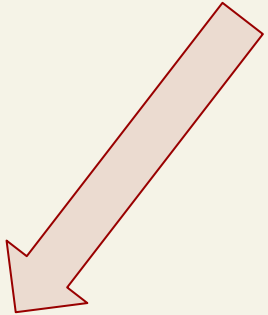
```
interface Country {
    name: string
    capital: string
}
```

```
interface Country {
    name: string
    capital: string
}
```

```
interface Country {
    name: string
    capital: string | null
}
```

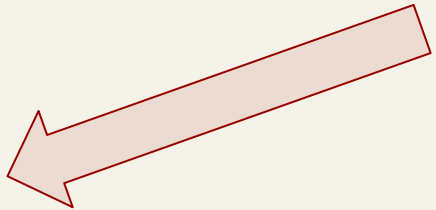
```ts
interface Country {
   name: string
   capital: string | null
}

function goToCapitalOf(country: Country) {




}
```

```ts
interface Country {
    name: string
    capital: string | null
}

function goToCapitalOf(country: Country) {



}
```



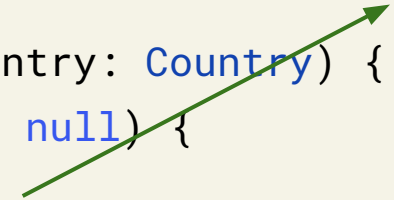
```ts
interface Country {
    name: string
    capital: string | null
}

function goToCapitalOf(country: Country) {
    if (country.capital !== null) {


    }
}
```

```ts
interface Country {
   name: string
   capital: string | null
}

function goToCapitalOf(country: Country) {
   if (country.capital !== null) {
     applyForVisa(country)
     console.log(`Going to ${country.capital.toUpperCase()}!`)
   }
}
```



```ts
interface Country {
  name: string
  capital: string | null
}

function goToCapitalOf(country: Country) {
  if (country.capital !== null) {
    applyForVisa(country)
    console.log(`Going to ${country.capital.toUpperCase()}!`)
  }
}
```

```ts
function applyForVisa(country: Country) {
  // Serious visa stuff
}
```



```ts
interface Country {
  name: string
  capital: string | null
}

function applyForVisa(country: Country) {
  // Serious visa stuff
}

function goToCapitalOf(country: Country) {
  if (country.capital !== null) {
    applyForVisa(country)
    console.log(`Going to ${country.capital.toUpperCase()}!`)
  }
}
```

```ts
interface Country {
   name: string
   capital: string | null
}

function goToCapitalOf(country: Country) {
   if (country.capital !== null) {
     applyForVisa(country)
     console.log(`Going to ${country.capital.toUpperCase()}!`)
   }
}

goToCapitalOf({ name: 'Romania', capital: ‘Bucharest’ })
```

```ts
function applyForVisa(country: Country) {
  // Serious visa stuff
}
```

```ts
interface Country {
   name: string
   capital: string | null
}
```

```ts
function applyForVisa(country: Country) {
  // Serious visa stuff
}
```

```ts
function goToCapitalOf(country: Country) {
   if (country.capital !== null) {
     applyForVisa(country)
     console.log(`Going to ${country.capital.toUpperCase()}!`)
   }
}

goToCapitalOf({ name: 'Romania', capital: ‘Bucharest’ })
goToCapitalOf({ name: 'Nauru', capital: null })
```

Playground

```
function goToCapitalOf(country: Country) {
  if (country.capital !== null) {
    applyForVisa(country)
    console.log(`Going to ${country.capital.toUpperCase()}!`)
  }
}
```

```
function goToCapitalOf(country: Country) {
  if (country.capital !== null) {
    applyForVisa(country)
    if (country.capital !== null ) {
      console.log(`Going to ${country.capital.toUpperCase()}!`)
    }
  }
}
```

```
function goToCapitalOf(country: Country) {
  if (country.capital !== null) {
    applyForVisa(country)
    doSomethingElse(country)
    doAThirdThing(country)
    console.log(`Going to ${country.capital.toUpperCase()}!`)
  }
}
```

```
function goToCapitalOf(country: Country) {
  if (country.capital !== null) {
    applyForVisa(country)
    if (country.capital !== null ) {
      doSomethingElse(country)
      if (country.capital !== null ) {
        doAThirdThing(country)
        if (country.capital !== null ) {
          console.log(`Going to ${country.capital.toUpperCase()}!`)
        }
      }
    }
  }
}
```

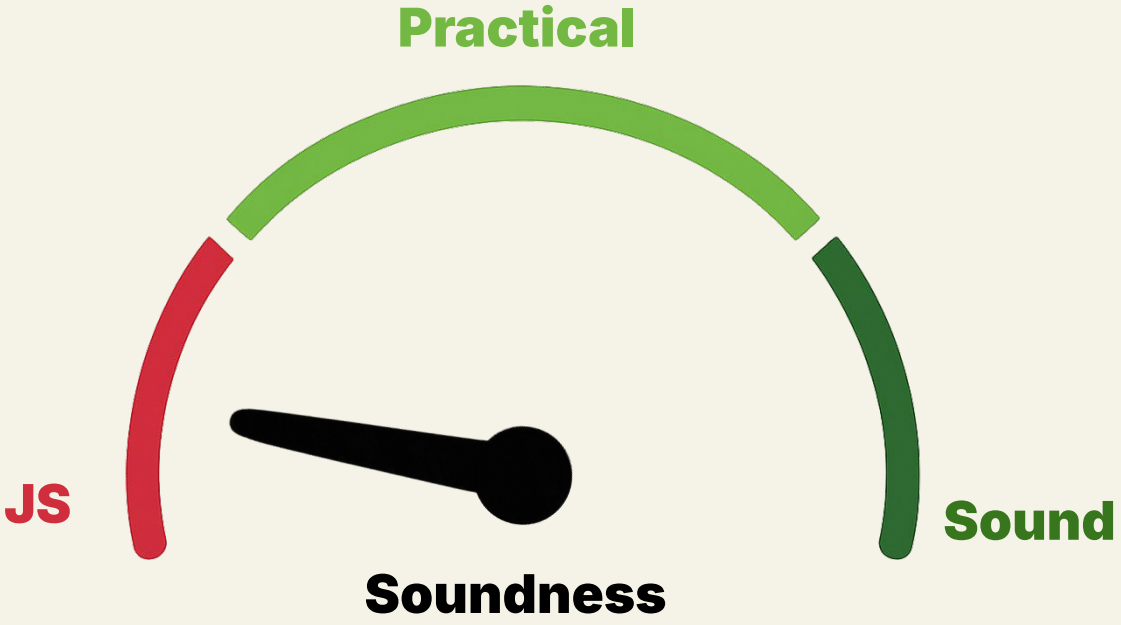
There’s much more!

FAQ Archive

Common "Bugs" That Aren't Bugs

> I've found a long-overlooked bug in TypeScript!

Source


Practical
JS
Sound
Soundness


Practical
JS
Sound
Soundness

`strict: true`


```
strict: true
noUncheckedIndexAccess: true
```


```
strict: true
noUncheckedIndexAccess: true
```


```
no-unsafe-*: true
```


```
strict: true
noUncheckedIndexAccess: true
```


```
no-unsafe-*: true
```


```
TS Reset: On
```

```
function applyForVisa(country: Country) {
  country.capital = null
}
```

```
function goToCapitalOf(country: Country) {
  if (country.capital !== null) {
    applyForVisa(country)
    if (country.capital !== null ) {
      doSomethingElse(country)
      if (country.capital !== null ) {
        doAThirdThing(country)
        if (country.capital !== null ) {
          console.log(`Going to ${country.capital.toUpperCase()}!`)
        }
      }
    }
  }
}
```

Frustrations happen because TypeScript is...

Not sound

Why didn't you catch this?

Not complete

This works... why are you bullying me!?

Frustrations happen because TypeScript is...

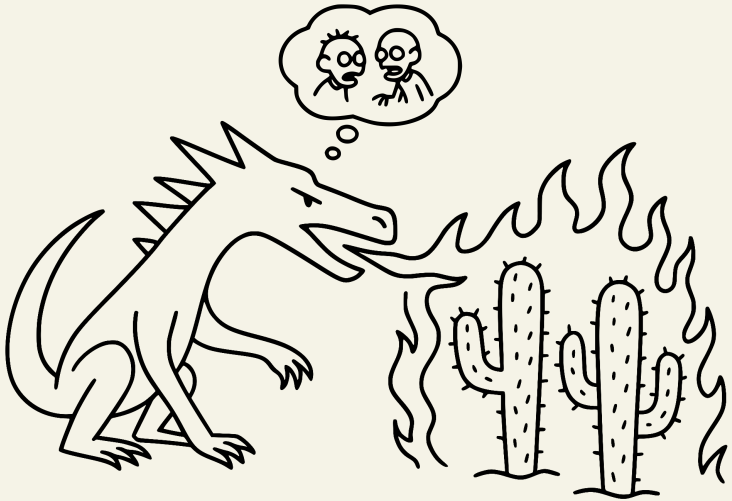
Not sound

Why didn't you catch this?

Not complete

This works... why are you bullying me!?

Why is TypeScript not complete ?



This works! Why are you bullying me!?

```
function setPopulation(population: number) {
 // ...
}

function setLanguage(language: string) {
 // ...
}
```

```
function setPopulation(population: number) {
 // ...
}

function setLanguage(language: string) {
 // ...
}

let data = 19186201
setPopulation(data)

data = "Romanian"
setLanguage(data)
```

```ts
function setPopulation(population: number) {
  // ...
}

function setLanguage(language: string) {
  // ...
}

let data = 19186201
setPopulation(data)

data = "Romanian"
setLanguage(data)
```

ERROR: Type 'string' is not assignable to type 'number'.

ERROR: Argument of type 'number' is not assignable to parameter of type 'string'.

```
function setPopulation(population: number) {
 // ...
}

function setLanguage(language: string) {
 // ...
}

let data = 19186201
setPopulation(data)

data = "Romanian"
setLanguage(data)
```

ERROR: Type 'string' is not assignable to type 'number'.

ERROR: Argument of type 'number' is not assignable to parameter of type 'string'.

```
function setPopulation(population: number) {
 // ...
}

function setLanguage(language: string) {
 // ...
}

let data = 19186201
setPopulation(data)

data = "Romanian"
setLanguage(data)
```

data: number

ERROR: Type 'string' is not assignable to type 'number'.

ERROR: Argument of type 'number' is not assignable to parameter of type 'string'.

```
function setPopulation(population: number) {
  // ...
}

function setLanguage(language: string) {
  // ...
}

let data = 19186201
setPopulation(data)

data = "Romanian"
setLanguage(data)
```

ERROR: Type 'string' is not assignable to type 'number'.

ERROR: Argument of type 'number' is not assignable to parameter of type 'string'.

```ts
function setPopulation(population: number) {
  // ...
}

function setLanguage(language: string) {
  // ...
}

let data = 19186201
setPopulation(data)

data = "Romanian"
setLanguage(data)
```

ERROR: Type 'string' is not assignable to type 'number'.

ERROR: Argument of type 'number' is not assignable to parameter of type 'string'.

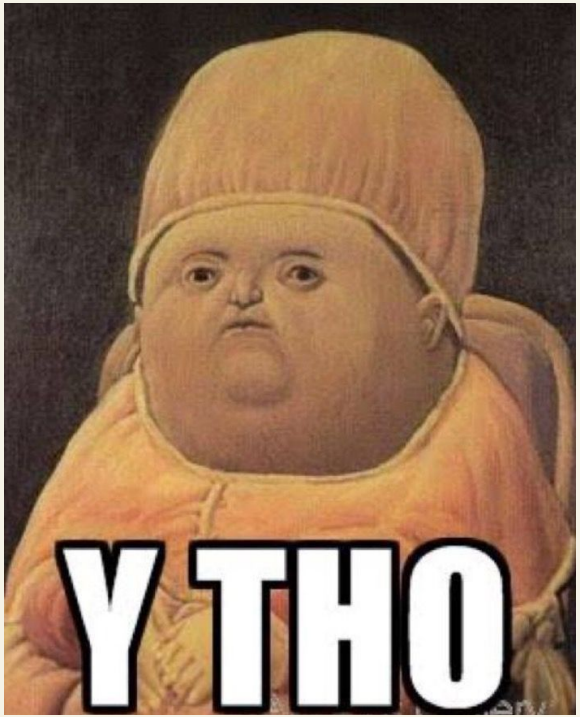
```ts
function setPopulation(population: number) {
  // ...
}

function setLanguage(language: string) {
  // ...
}

let data: string | number = 19186201
setPopulation(data)

data = "Romanian"
setLanguage(data)
```

ERROR: Type 'string' is not assignable to type 'number'.

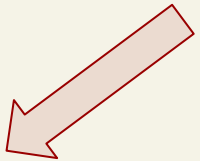
ERROR: Argument of type 'number' is not assignable to parameter of type 'string'.

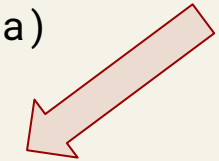
```ts
function setPopulation(population: number) {
 // ...
}

function setLanguage(language: string) {
 // ...
}

let data: string | number = 19186201
setPopulation(data)

data = "Romanian"
setLanguage(data)
```

```ts
function setPopulation(population: number) {
  // ...
}

function setLanguage(language: string) {
  // ...
}

let data: string | number = 19186201
setPopulation(data)

data = "Romanian"
setLanguage(data)
```



```ts
function setPopulation(population: number) {
 // ...
}

function setLanguage(language: string) {
 // ...
}

let data: string | number = 19186201
setPopulation(data)

data = "Romanian"
setLanguage(data)
```





Playground

Why not?

1. Prevents a common JavaScript footgun
2. Simplicity and consistency

Frustrations happen because TypeScript is...

Not sound

Why didn't you catch this?

Not complete

This works... why are you bullying me!?

Frustrations happen because TypeScript is...

Not sound

Why didn't you catch this?

Not complete

This works... why are you bullying me!?

```
Object.keys(...)
```

```ts
interface Country {
  name: string
  capital: string | null
}
```

```ts
interface Country {
  name: string
  capital: string | null
}
```

```ts
const country = { name: 'Romania', capital: 'Bucharest' }
logCountry(country)
```

```ts
interface Country {
  name: string
  capital: string | null
}

function logCountry(country: Country) {
}

const country = { name: 'Romania', capital: 'Bucharest' }
logCountry(country)
```

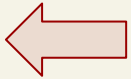
```ts
interface Country {
  name: string
  capital: string | null
}

function logCountry(country: Country) {
  const countryKeys = Object.keys(country)

}

const country = { name: 'Romania', capital: 'Bucharest' }
logCountry(country)
```

```ts
interface Country {
  name: string
  capital: string | null
}

function logCountry(country: Country) {
  const countryKeys = Object.keys(country)
  for (const key of countryKeys) {
    console.log(key, country[key])
  }
}

const country = { name: 'Romania', capital: 'Bucharest' }
logCountry(country)
```

```ts
interface Country {
  name: string
  capital: string | null
}

function logCountry(country: Country) {
  const countryKeys = Object.keys(country)
  for (const key of countryKeys) {
    console.log(key, country[key])
  }
}

const country = { name: 'Romania', capital: 'Bucharest' }
logCountry(country)
```

ERROR: … expression of type 'string' can't be used to index type 'Country'.'.

```ts
interface Country {
  name: string
  capital: string | null
}

function logCountry(country: Country) {
  const countryKeys = Object.keys(country)
  for (const key of countryKeys) {
    console.log(key, country[key])
  }
}

const country = { name: 'Romania', capital: 'Bucharest' }
logCountry(country)
```



```ts
countryKeys: string[]
```

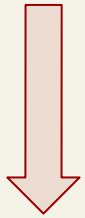
ERROR: … expression of type 'string' can't be used to index type 'Country'.'.

```ts
interface Country {
  name: string
  capital: string | null
}

function logCountry(country: Country) {
  const countryKeys = Object.keys(country)
  for (const key of countryKeys) {
    console.log(key, country[key])
  }
}

const country = { name: 'Romania', capital: 'Bucharest' }
logCountry(country)
```

```ts
interface Country {
  name: string
  capital: string | null
}

function logCountry(country: Country) {
  const countryKeys = Object.keys(country)
  for (const key of countryKeys) {
    console.log(key, country[key])
  }
}

const country = { name: 'Romania', capital: 'Bucharest' }
logCountry(country)
```

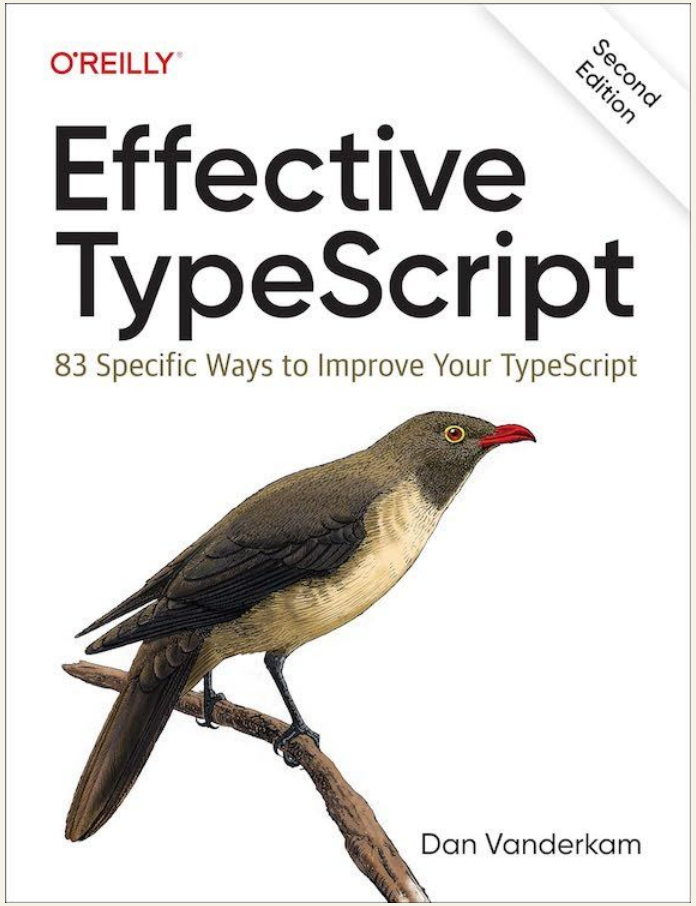
Playground

```ts
interface Country {
  name: string
  capital: string | null
}

function logCountry(country: Country) {
  const countryKeys = Object.keys(country)
  for (const key of countryKeys) {
    console.log(key, country[key])
  }
}
```



```ts
const country = { name: 'Romania', capital: 'Bucharest', population: 19186201 }
logCountry(country)
```

There’s a lot of this stuff!


O'REILLY
Second Edition
Effective TypeScript
83 Specific Ways to Improve Your TypeScript
Dan Vanderkam

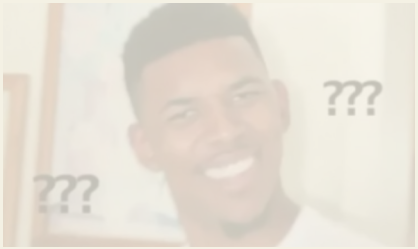
Frustrations happen because TypeScript is...

Not sound

Why didn't you catch this?

Not complete

This works... why are you bullying me!?



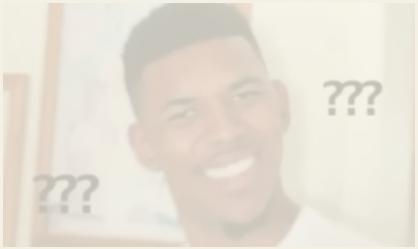
Frustrations happen because TypeScript is...

Not sound

Why didn't you catch this?

Not complete

This works... why are you bullying me!?



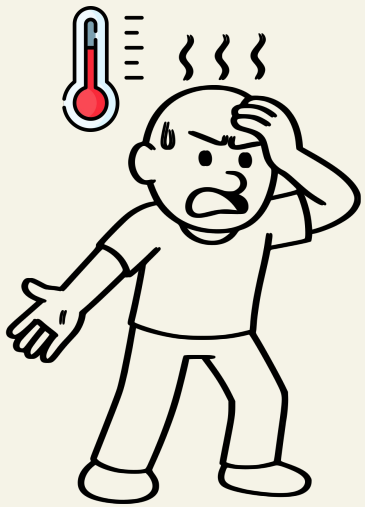
Frustrations happen because TypeScript is...

Not sound

Why didn't you catch this?

Not complete

This works... why are you bullying me!?




The Compiler Is Our Friend

You can reach me at...

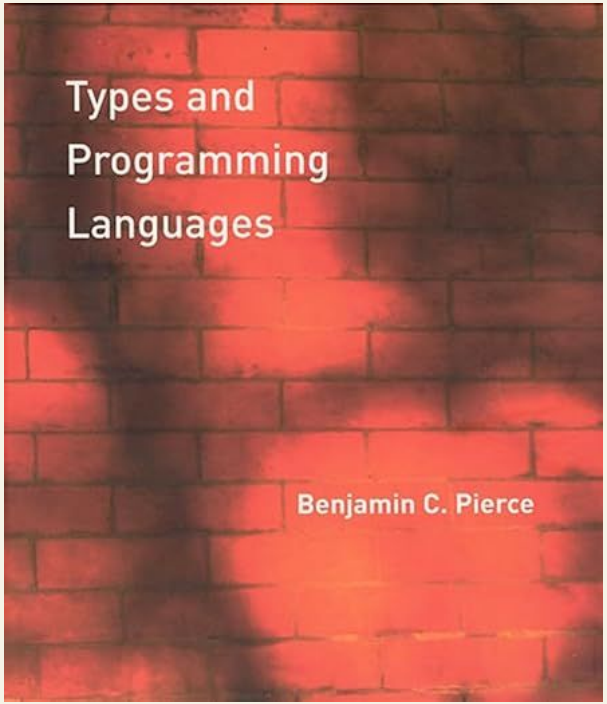
@filipsodic

filip@wasp-lang.dev

... Or come to talk to us in the break

Extra slides

I *really* like this, even beyond TypeScript!



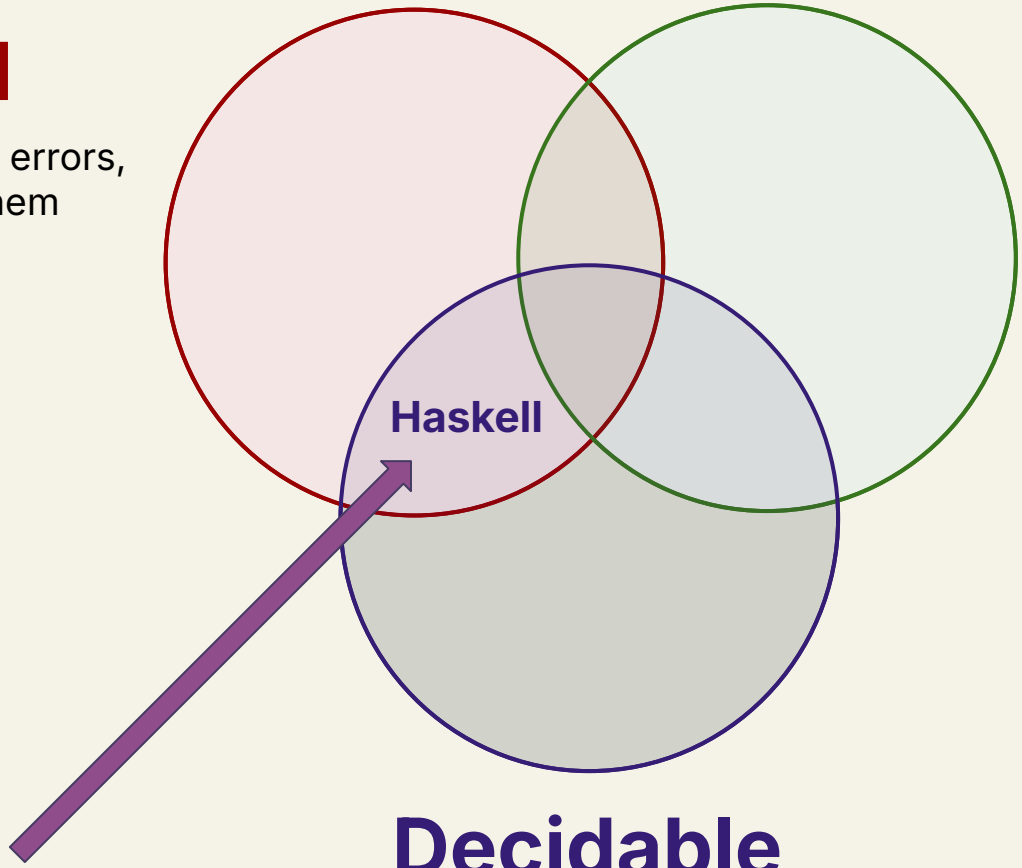
Haskell

Sound

If the code has type errors,
always report them

Complete

Never report errors
that don't exist



Decidable

Never crash or hang

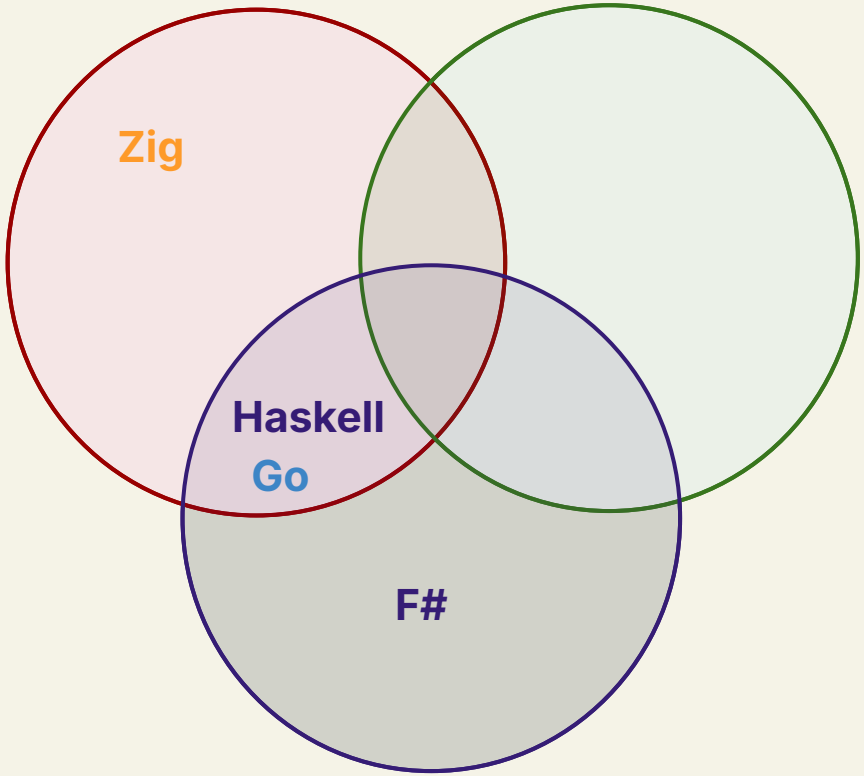
TS

Sound

If the code has type errors, **always** report them

Complete

Never report errors that don't exist



Java

C#

Scala

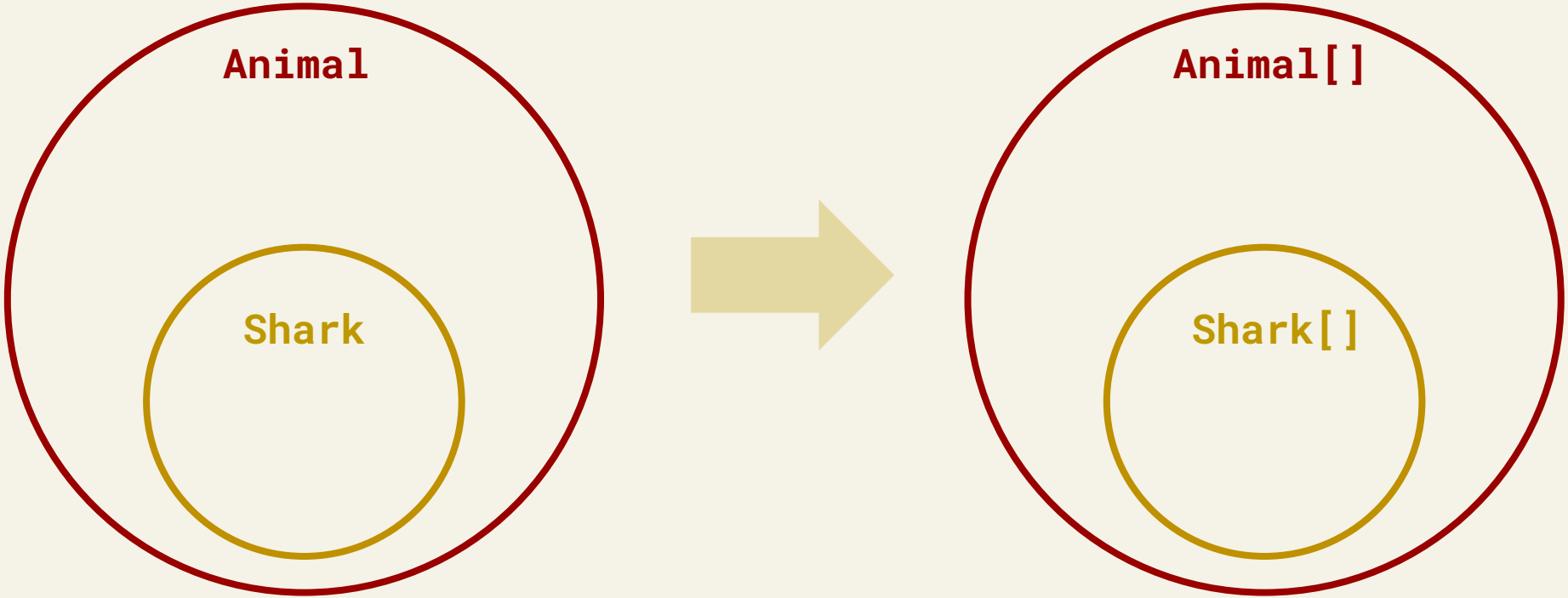
Rust

TypeScript

Decidable

Never crash or hang

Soundness tradeoffs



Playground

```
interface Animal {
 name: string
}

interface Toucan extends Animal {
 beakSize: number
}

interface Shark extends Animal {
 maxDepth: number
}

const toucans: Toucan[] = []
const animals: Animal[] = toucans
const shark: Shark = { name: 'Baby', maxDepth: 2000 }
animals.push(shark)
const beakSizeRounded = toucans[0].beakSize.toFixed(2)
```

References

- TypeScript Handbook
- r/ProgrammingLanguages
- TypeScript FAQ (and the repo in general)
- Typing is Hard
- Effective TypeScript (and Dan’s blog)
- Types and Programming Languages