

Hello!

I love TypeScript!

I love TypeScript!

But...



// @ts-ignore

any

as Type



I'll do it right...

I'll do it right...

... after the release

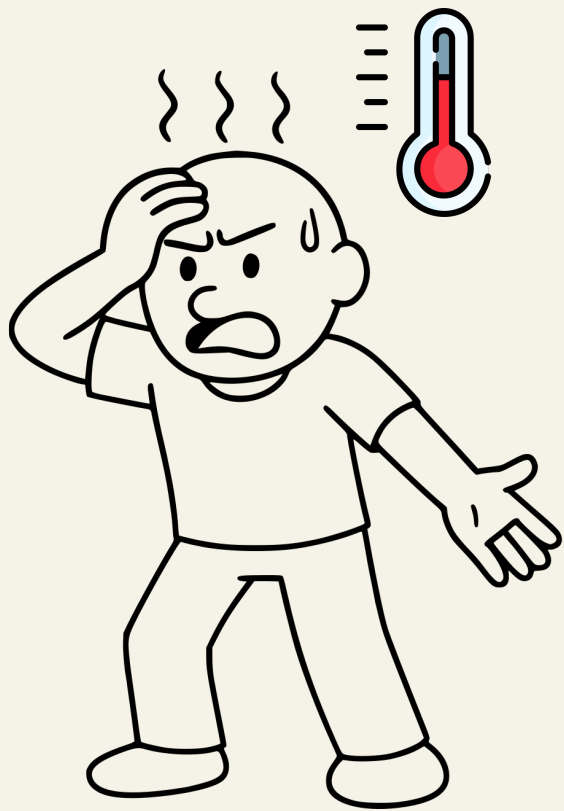
A story about perspective



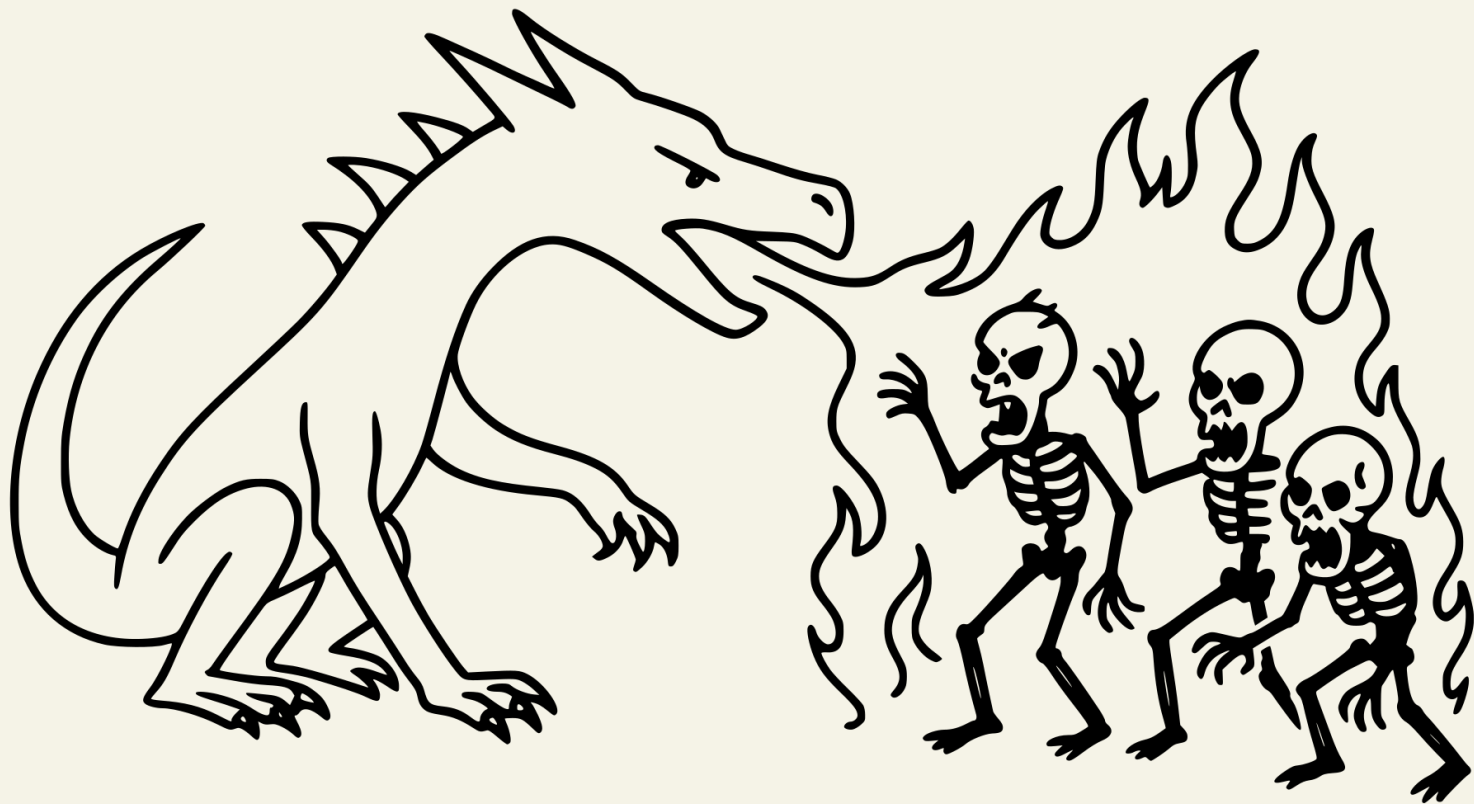


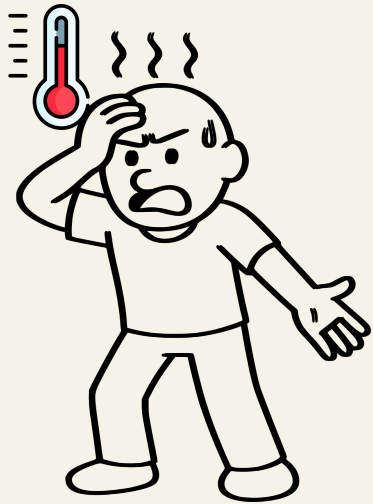


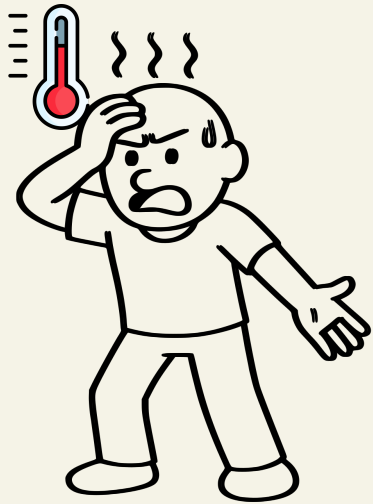






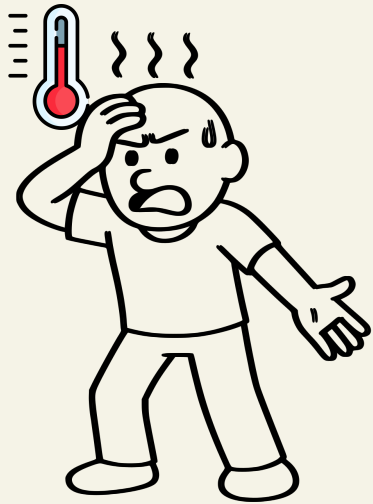




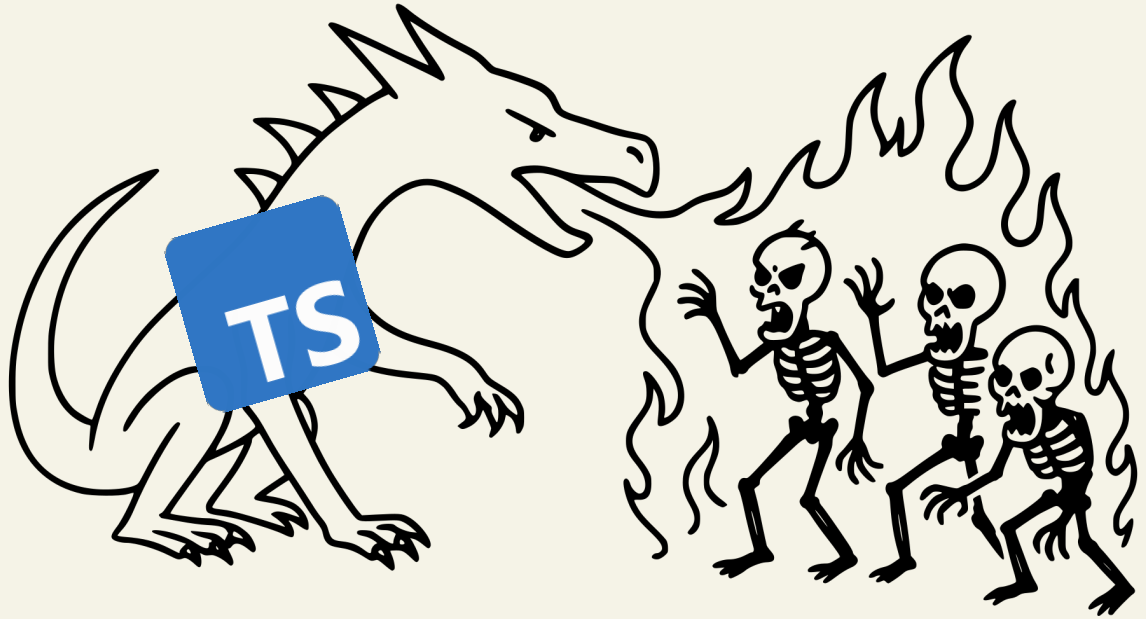


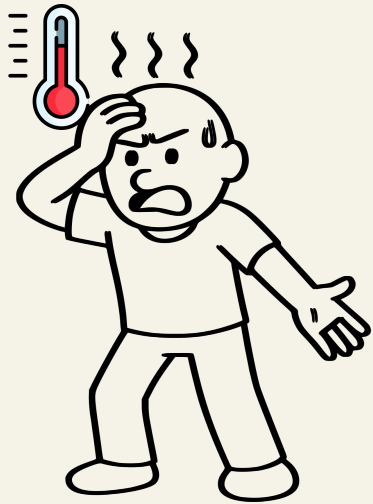
You



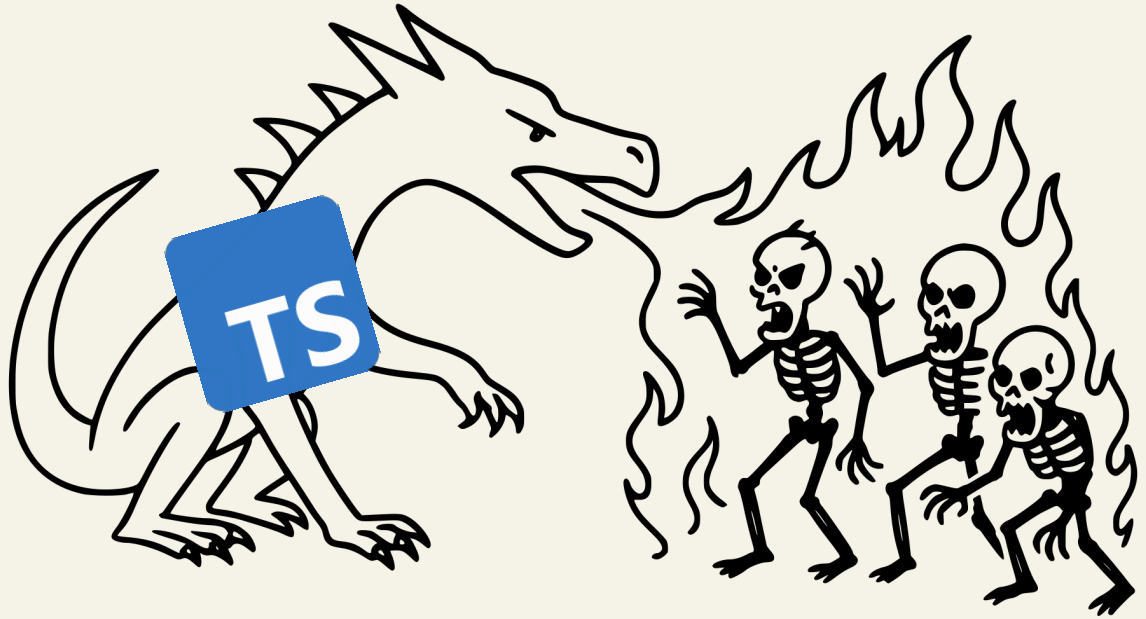


You

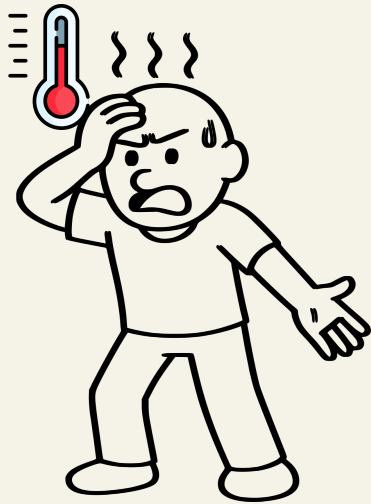




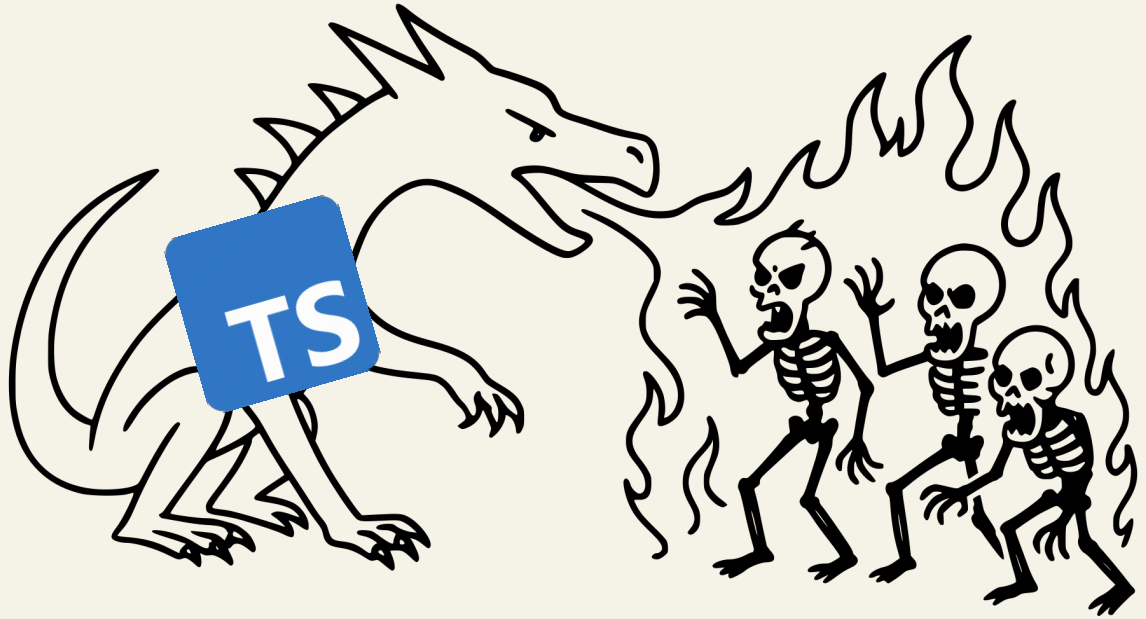
You



???

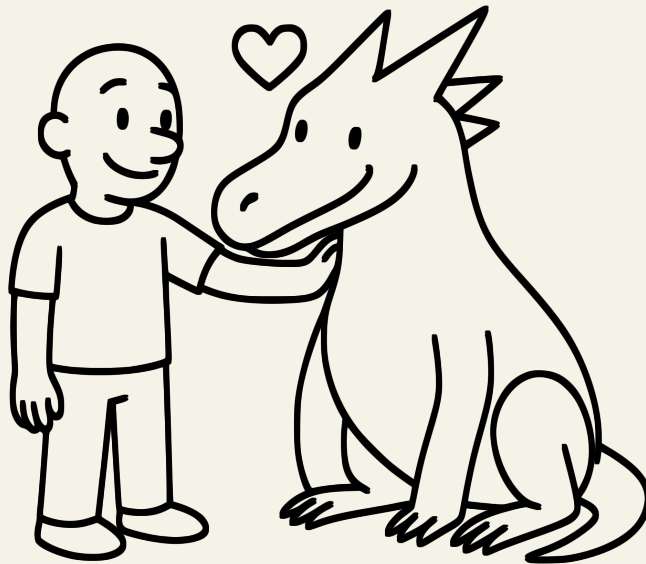


You



Our topic

Treat the Compiler With Compassion



Why do I care?



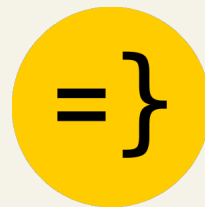
Filip Sodic

Working with compilers at [Wasp](#) and for fun
Teaching Haskell at UniZG



sodic.dev

[Wasp](#)



TypeScript



Haskell





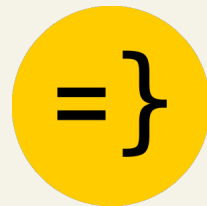
Filip Sodic

Working with compilers at [Wasp](#) and for fun
Teaching Haskell at UniZG



sodic.dev

[Wasp](#)



TypeScript



Haskell

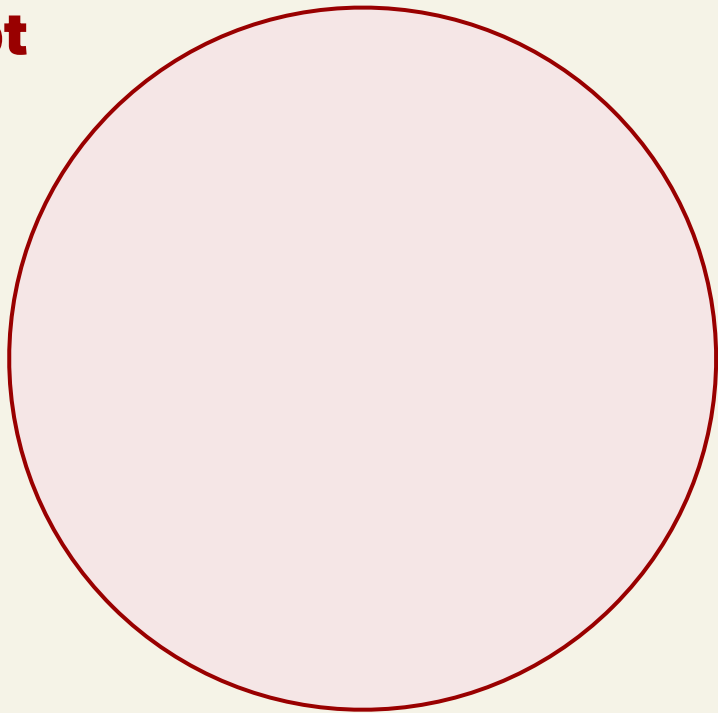


***Why* do we get frustrated?**

Why do we get frustrated?

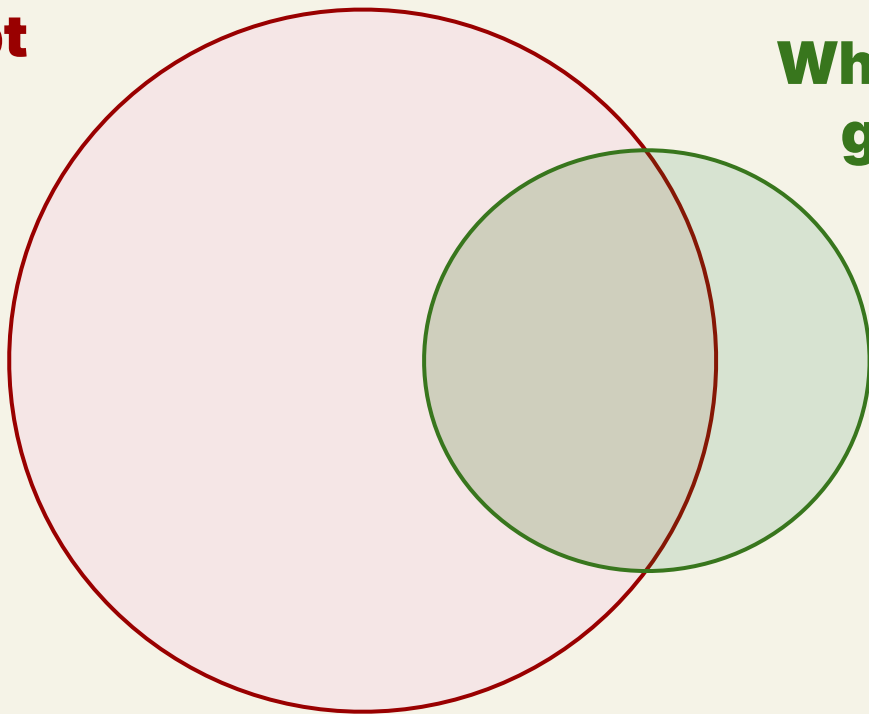
...Because we don't get what we want

What we want from TypeScript



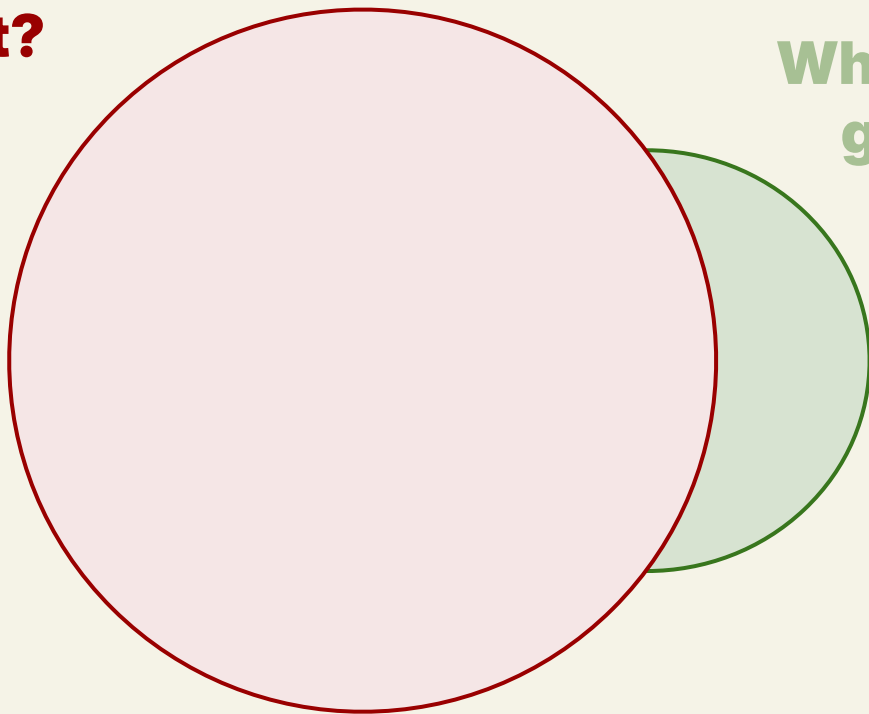
**What we want from
TypeScript**

**What's actually
good for us**



**What do we want from
TypeScript?**

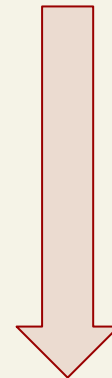
**What's actually
good for us**



```
const conference = {  
  name: 'JSDay',  
  month: 'April',  
}
```

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

```
const conference = {  
  name: 'JSDay',  
  month: 'April',  
}
```



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

```
const conference = {  
  name: 'JSDay',  
  month: 'April',  
}
```

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

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

```
const conference = {  
  name: 'JSDay',  
  month: 'April',  
}
```

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

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

```
const conference = {  
  name: 'JSDay',  
  month: 'April',  
}
```

```
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

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

Never report errors that don't exist

Never crash or hang

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

Never report errors that don't exist

Never crash or hang

Sound

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

Never report errors that don't exist

Never crash or hang

Sound

If the code has type errors,

always report them

Never report errors that don't exist

Never crash or hang

Sound

If the code has type errors,
always report them

Never report errors that don't exist

Never crash or hang

Sound

If the code has type errors,
always report them

Complete

Never report errors that don't exist

Never crash or hang

Sound

If the code has type errors,
always report them

Complete

Never report errors that
don't exist

Never crash or hang

Sound

If the code has type errors,
always report them

Complete

Never report errors that
don't exist

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

Sound

If the code has type errors,
always report them

Complete

Never report errors that
don't exist

Decidable

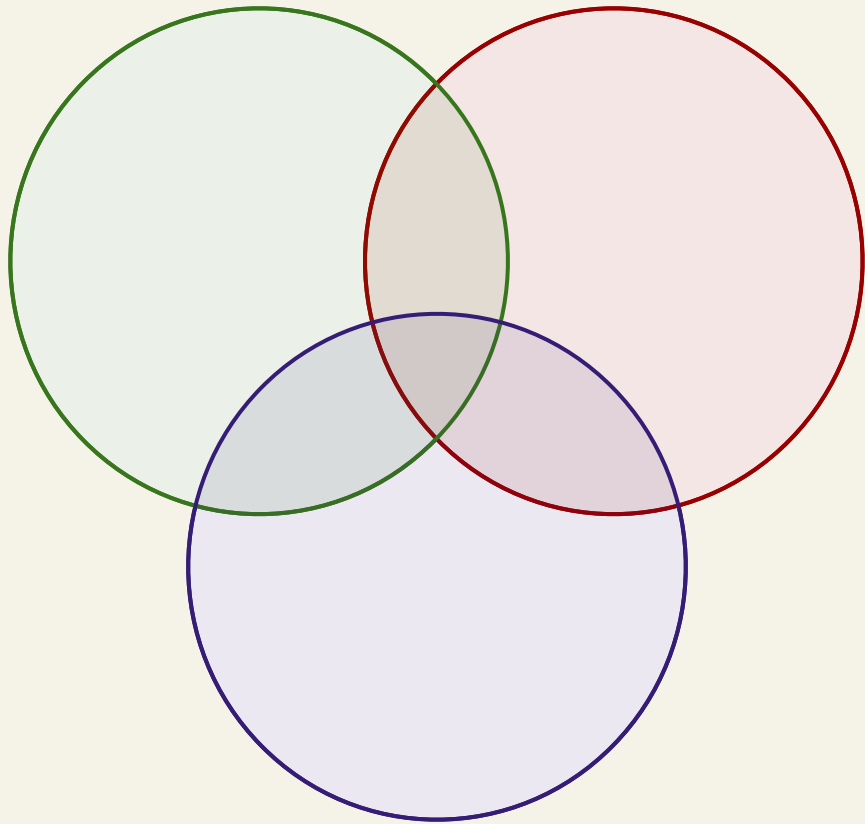
Never crash or hang

Sound

If the code has type errors,
always report them

Complete

Never report errors that
don't exist



Decidable

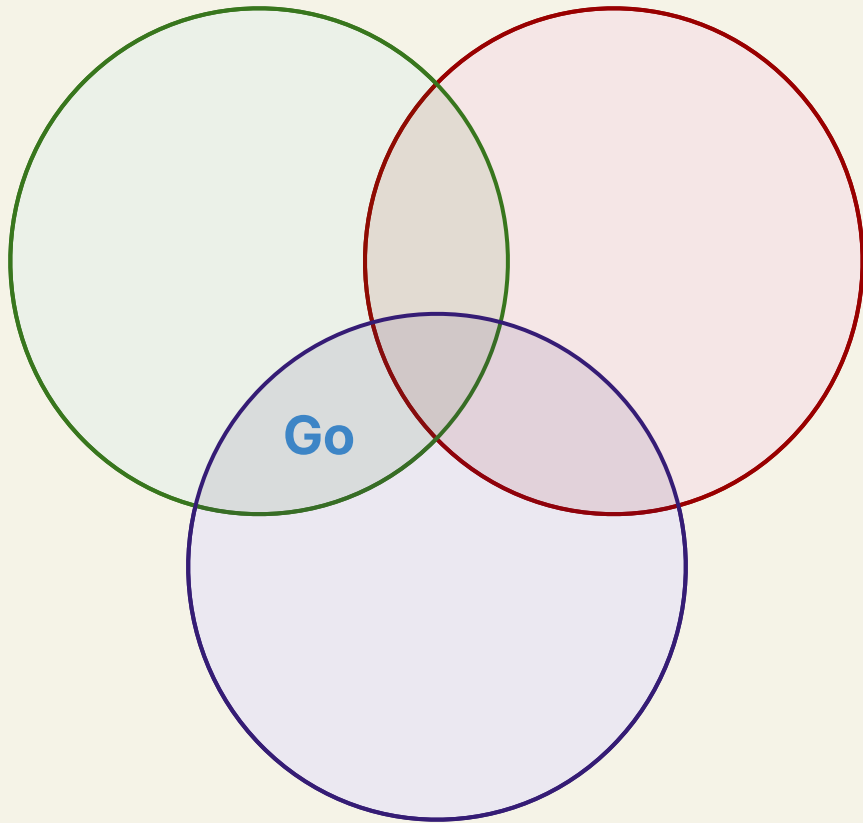
Never crash or hang

Sound

If the code has type errors,
always report them

Complete

Never report errors that
don't exist



Go

Decidable

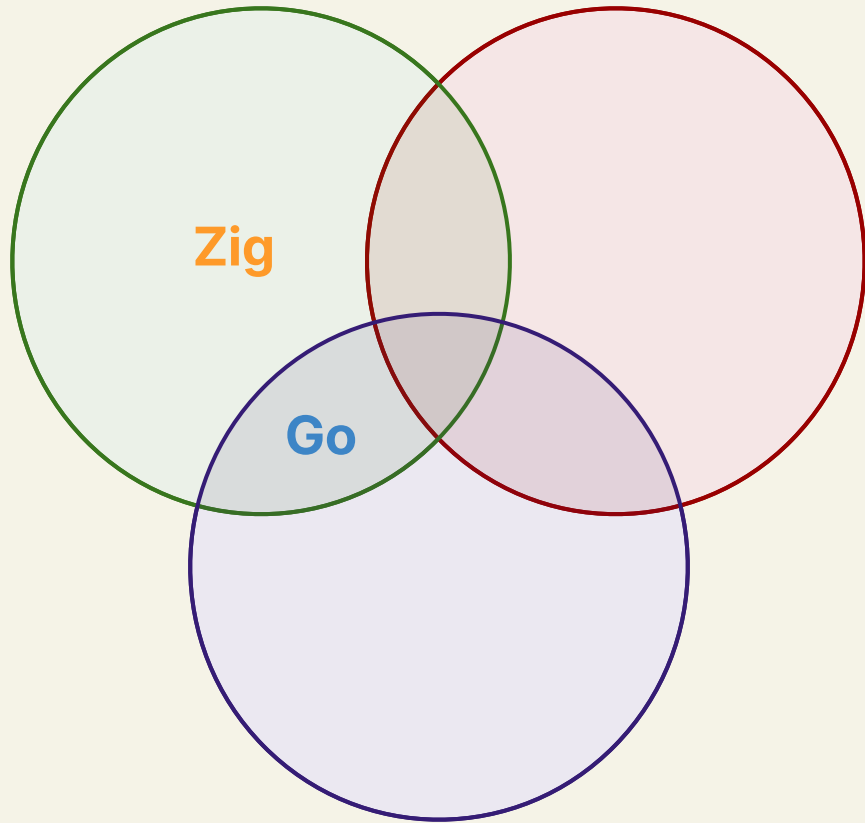
Never crash or hang

Sound

If the code has type errors,
always report them

Complete

Never report errors that
don't exist



Decidable

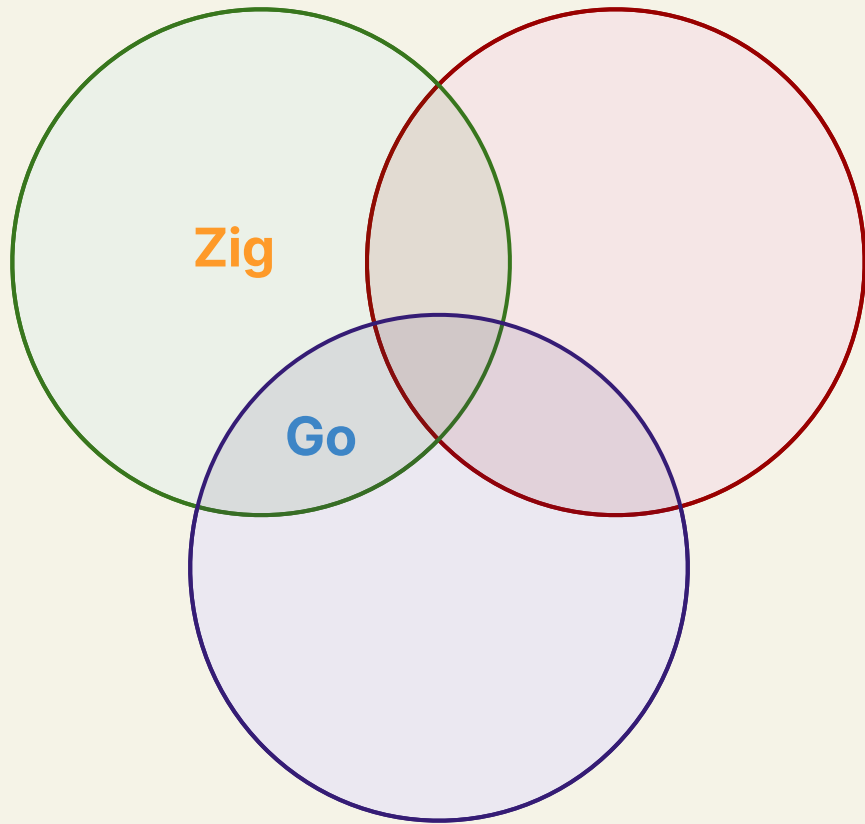
Never crash or hang

Sound

If the code has type errors,
always report them

Complete

Never report errors that
don't exist



Zig

Go

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

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

Never report errors that don't exist

Never crash or hang

If the code has type errors, ^{*often*} ~~always~~ report them

Never report errors that don't exist

Never crash or hang

If the code has type errors, ^{often} ~~always~~ report them

Sometimes

~~Never~~ report errors that don't exist

Never crash or hang

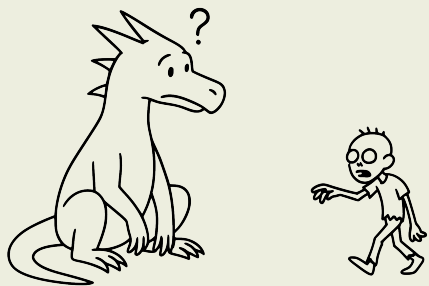
If the code has type errors, ^{often} ~~always~~ report them

Sometimes

~~Never~~ report errors that don't exist

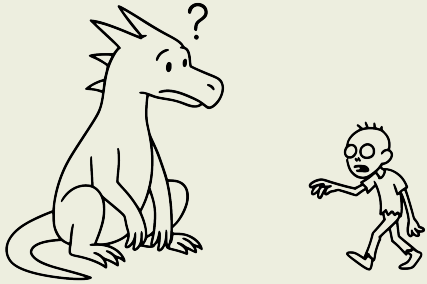
Sometimes

~~Never~~ crash or hang



*Why didn't you catch
this?*

Not Sound

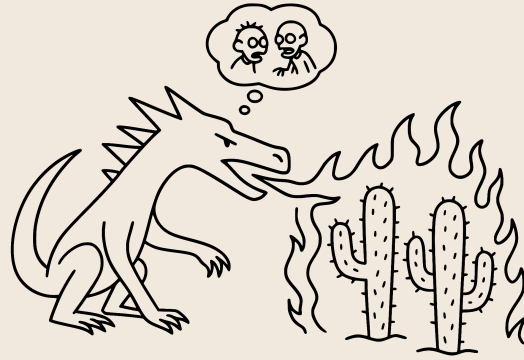


*Why didn't you catch
this?*

Not Sound



*Why didn't you catch
this?*



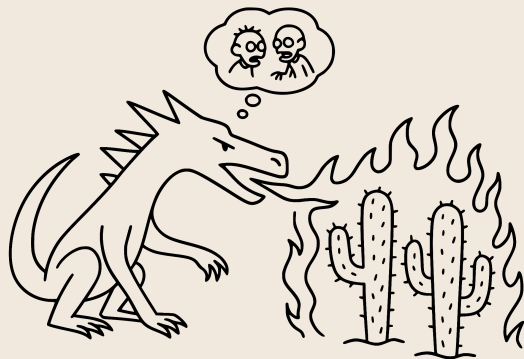
*This works...
why are you bullying me!?*

Not Sound



*Why didn't you catch
this?*

Not Complete



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

Not Sound



*Why didn't you catch
this?*

Not Complete



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



*Mom, the compiler is
broken!*

Not Sound



*Why didn't you catch
this?*

Not Complete



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

Not Decidable



*Mom, the compiler is
broken!*

Not Sound



*Why didn't you catch
this?*

Not Complete



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

Not Sound



*Why didn't you catch
this?*

Not Complete



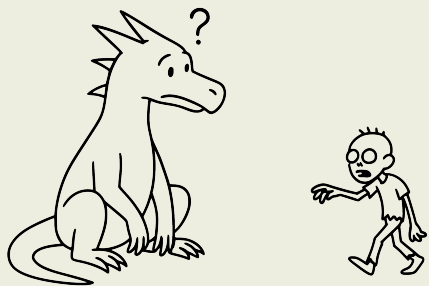
*This works...
why are you bullying me!?*

Something else



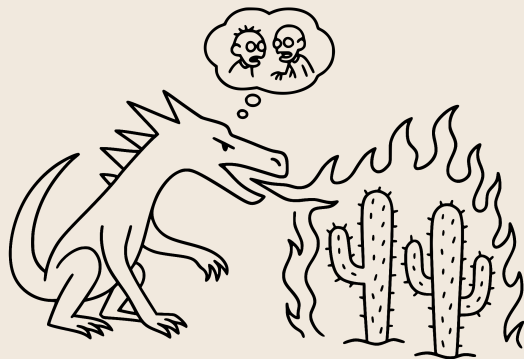
???

Not Sound



*Why didn't you catch
this?*

Not Complete



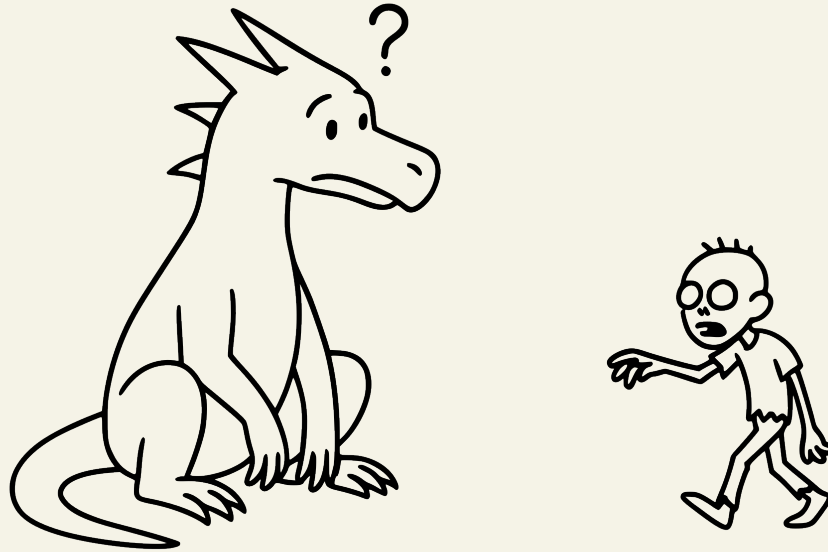
*This works...
why are you bullying me!?*

Something else



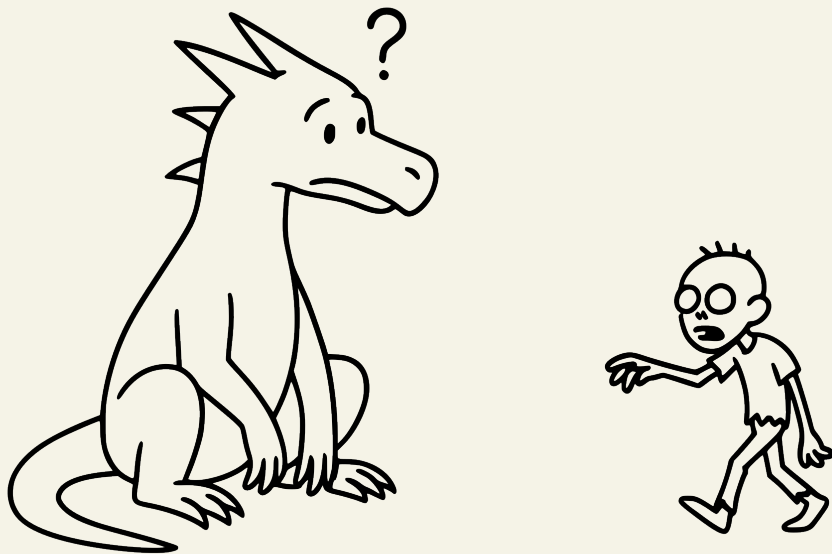
???

Not Sound



Why didn't you catch this?

Why is TypeScript **Not** Sound?



Why didn't you catch this?

Is it an accident?

Is it an accident?

No!

**TypeScript is not sound
on purpose!**

Non-goals

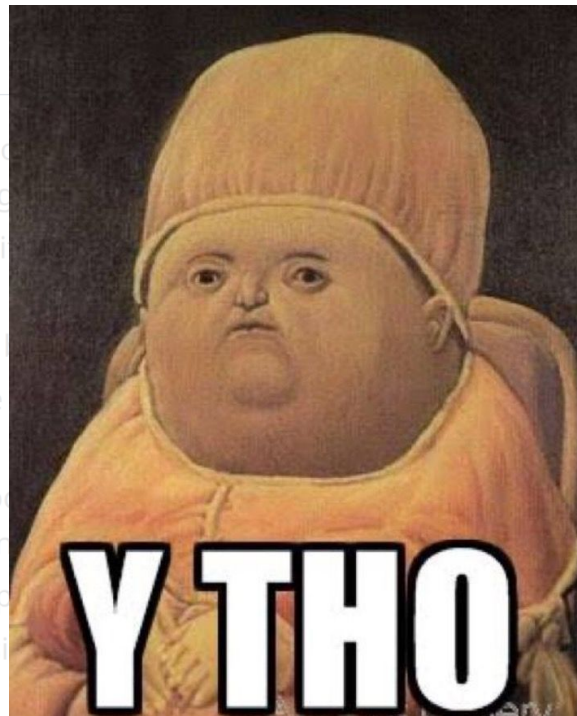
1. Exactly mimic the design of existing languages. Instead, use the behavior of JavaScript and the intentions of program authors as a guide for what makes the most sense in the language.
2. Aggressively optimize the runtime performance of programs. Instead, emit idiomatic JavaScript code that plays well 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.

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 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.

Non-goals

1. Exactly mimic the design of existing languages. Instead, use the behavior 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 code that works well with the performance characteristics of runtime platforms.
3. Apply a sound or "provably correct" type system. Instead, strike a balance between safety and flexibility.
4. Provide an end-to-end build pipeline. Instead, make the system extensible enough to support a compiler for more complex build workflows.
5. Add or rely on run-time information in programs, or emit different code for different run-time systems. Instead, encourage programming patterns that do not require run-time information.
6. Provide additional runtime functionality or libraries. Instead, use TypeScript's built-in functionality.
7. Introduce behaviour that is likely to surprise users. Instead have due consideration for how commonly-used languages behave.



[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  
}
```

```
interface Country {
```

```
name: string
```

```
capital: string | null
```

}

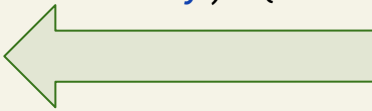
```
function tryGoingToCapitalOf(country: Country) {
```

}



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

```
function tryGoingToCapitalOf(country: Country) {  
    if (country.capital !== null) {  
  
    }  
}
```



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

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



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

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

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

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

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

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

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

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

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

```
tryGoingToCapitalOf({ name: 'Italy', capital: 'Rome' })
```



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

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

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

```
tryGoingToCapitalOf({ name: 'Italy', capital: 'Rome' })  
tryGoingToCapitalOf({ name: 'Nauru', capital: null })
```



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

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

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

[Playground](#)

```
tryGoingToCapitalOf({ name: 'Italy', capital: 'Rome' })  
tryGoingToCapitalOf({ name: 'Nauru', capital: null })
```



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

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

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

```
function tryGoingToCapitalOf(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!

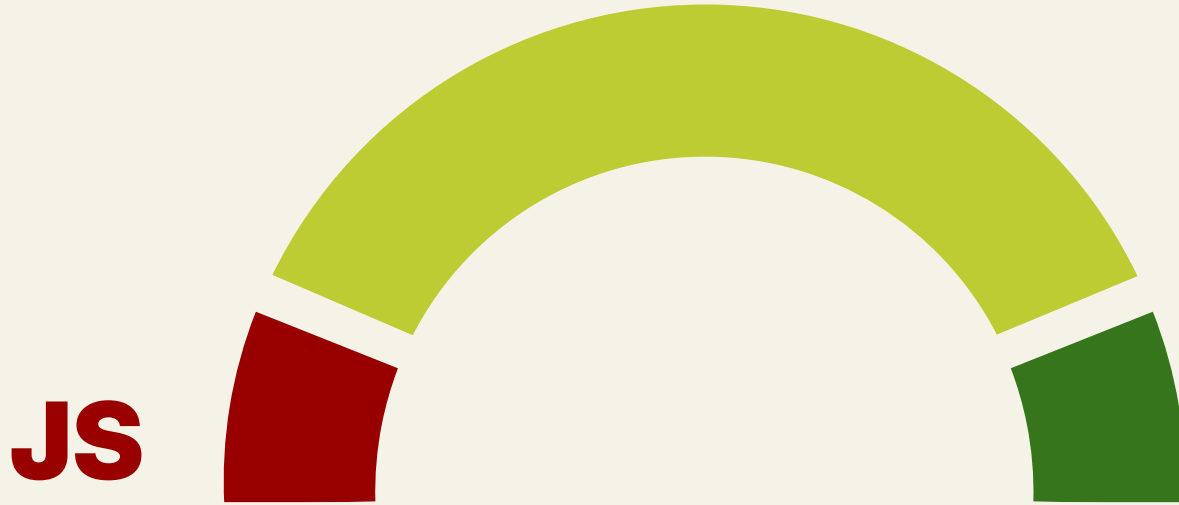
[Source](#)

Practicality **vs** Soundness

Practicality **vs** Soundness



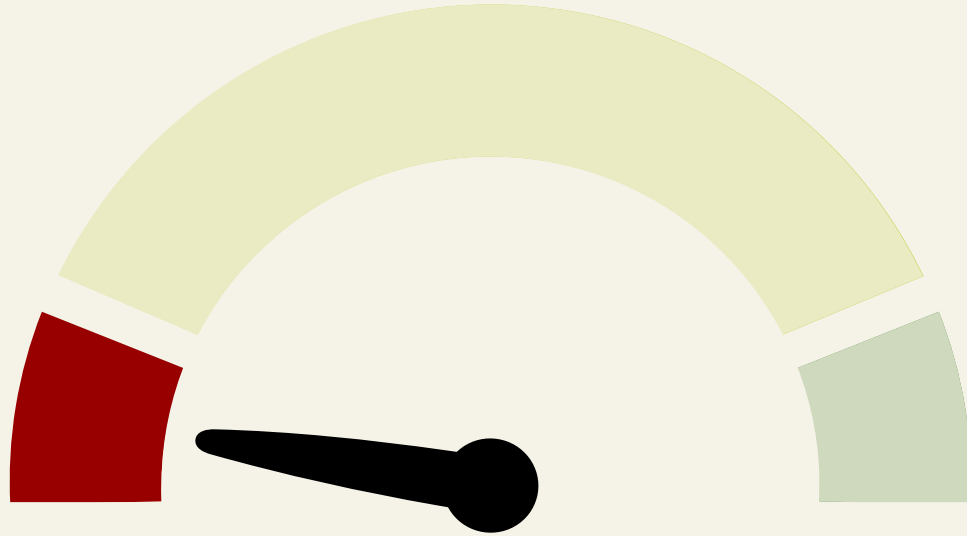
Practicality **vs** Soundness



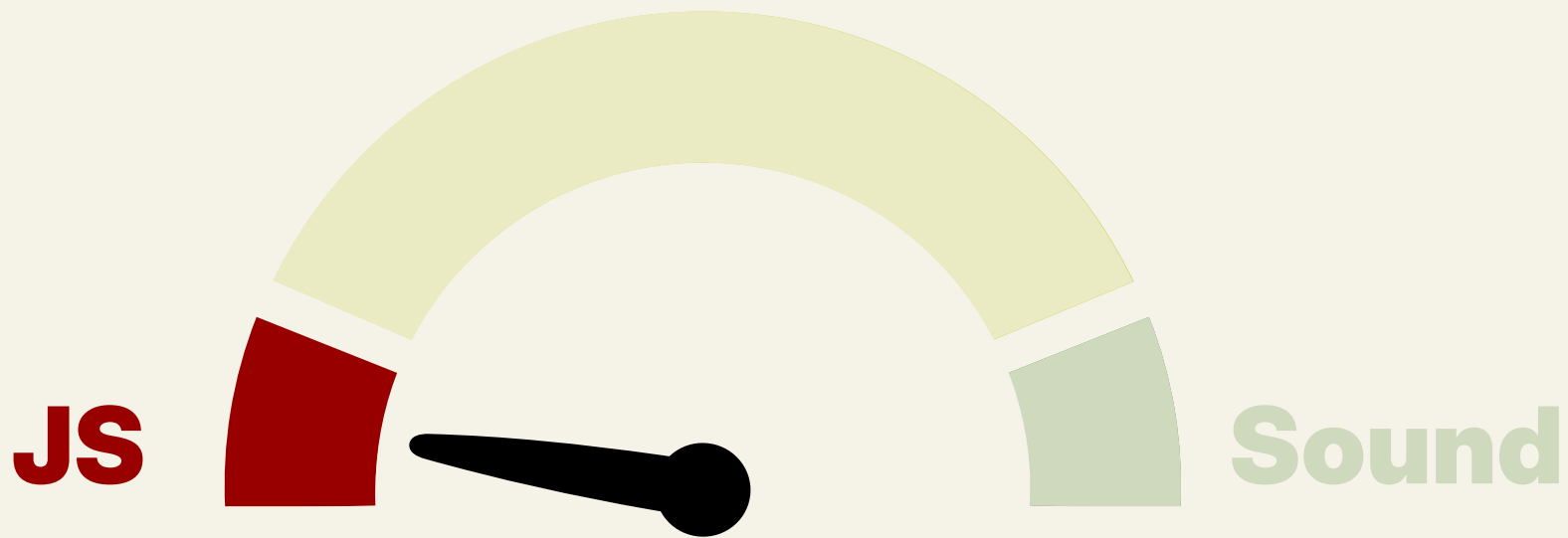
Practicality **vs** Soundness

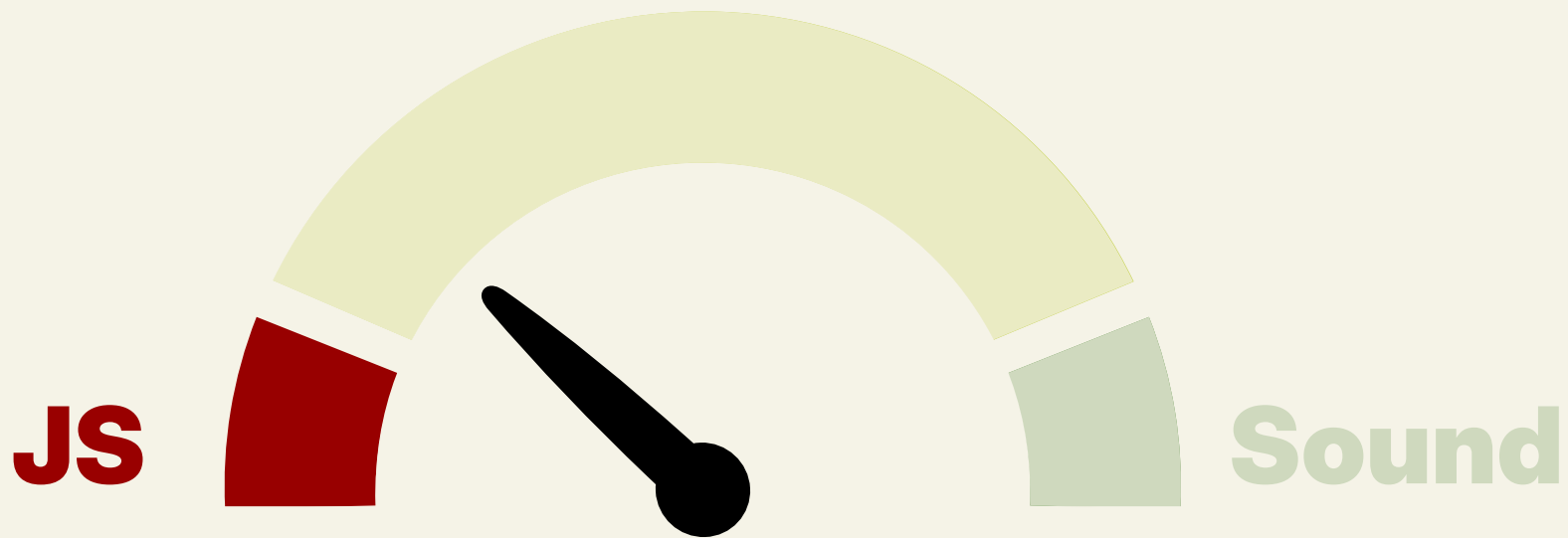


JS



Sound













strict: **true**





strict: **true**





strict: **true**

noUncheckedIndexedAccess: **true**





strict: **true**

noUncheckedIndexedAccess: **true**





strict: **true**

noUncheckedIndexedAccess: **true**



JS



Sound



strict: **true**

noUncheckedIndexAccess: **true**



no-unsafe-*: **true**





strict: **true**

noUncheckedIndexAccess: **true**



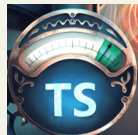
no-unsafe-*: **true**





strict: **true**

noUncheckedIndexAccess: **true**



no-unsafe-*: **true**





strict: **true**

noUncheckedIndexAccess: **true**



TS Reset: **On**



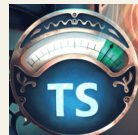
no-unsafe-*: **true**





strict: **true**

noUncheckedIndexAccess: **true**



TS Reset: **On**



no-unsafe-*: **true**

JS

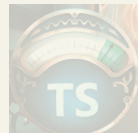


Sound



strict: **true**

noUncheckedIndexAccess: **true**

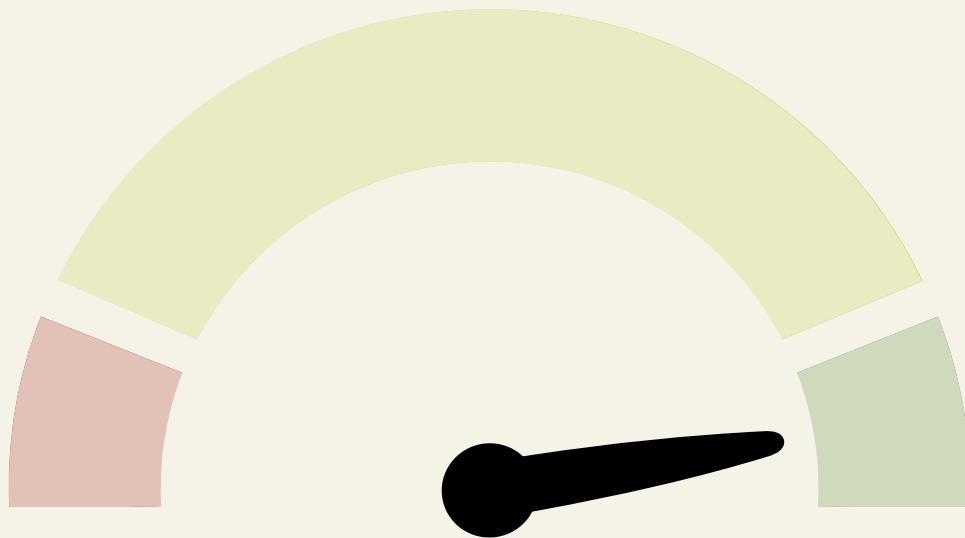


TS Reset: **On**



no-unsafe-*: **true**

JS

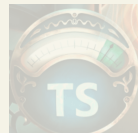


Sound



strict: **true**

noUncheckedIndexAccess: **true**

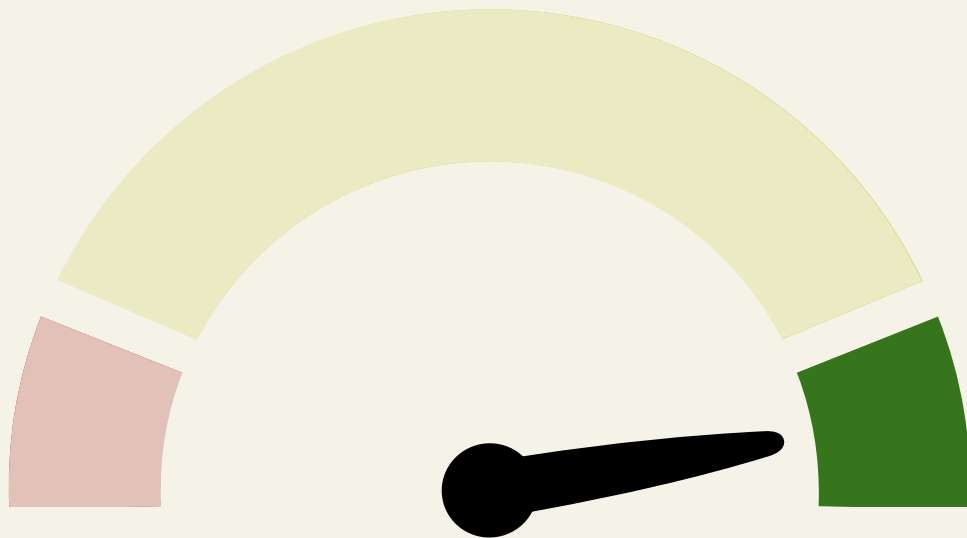


TS Reset: **On**



no-unsafe-*: **true**

JS



Sound



strict: **true**

noUncheckedIndexAccess: **true**

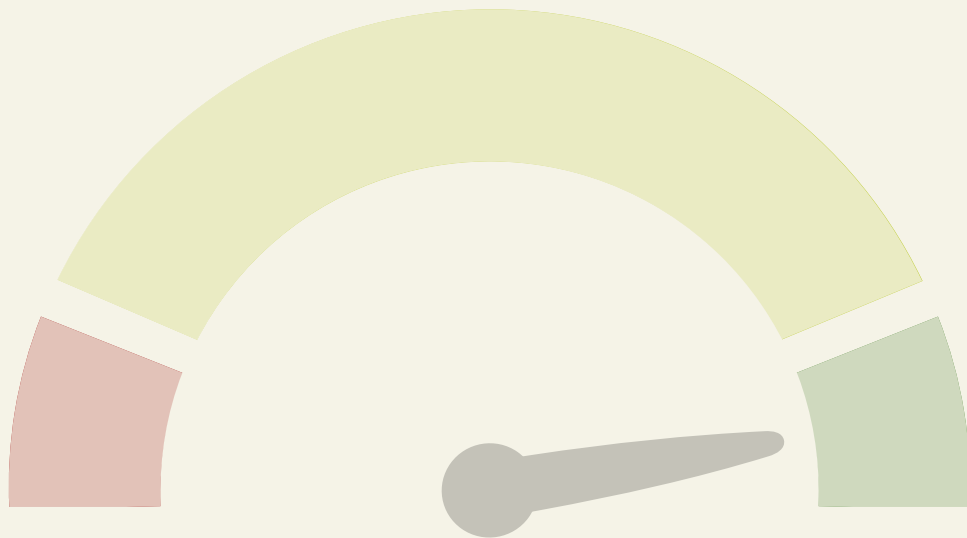


TS Reset: **On**



no-unsafe-*: **true**

JS



Sound

```
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 ) {  
                // ...  
            }  
        }  
    }  
}
```

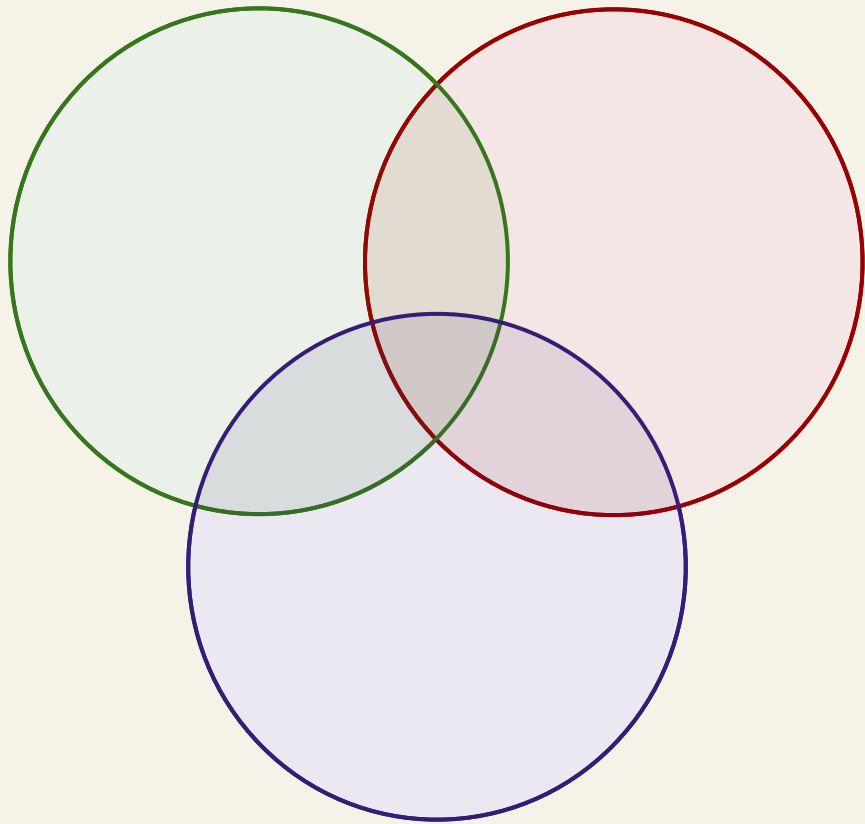
There is social proof

Sound

If the code has type errors,
always report them

Complete

Never report errors that
don't exist



Decidable

Never crash or hang

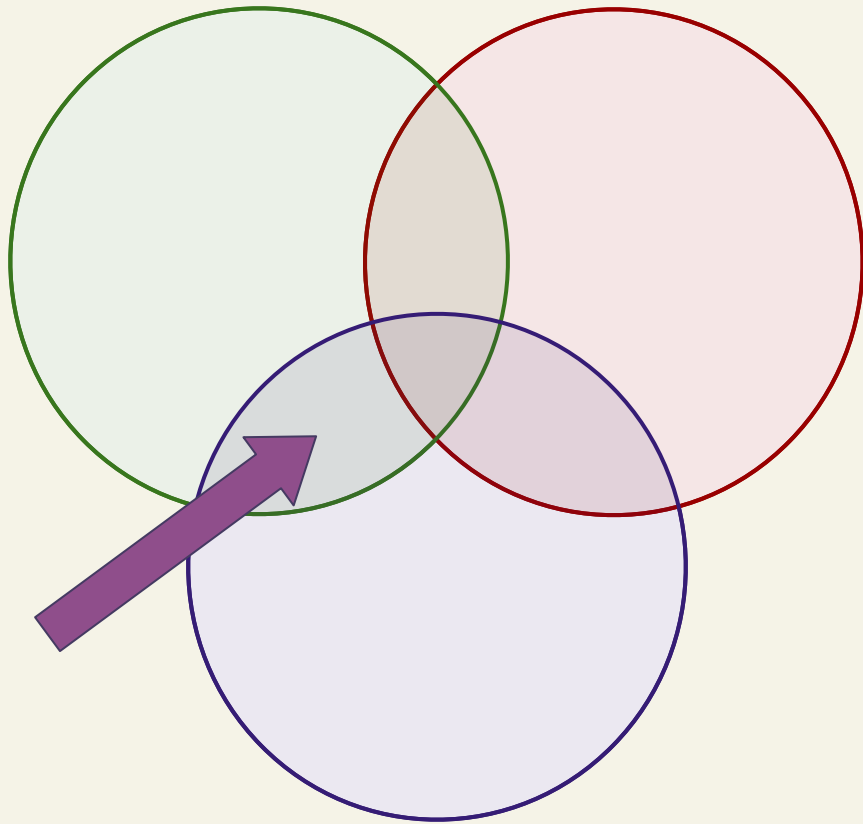


Sound

If the code has type errors,
always report them

Complete

Never report errors that
don't exist



Decidable

Never crash or hang

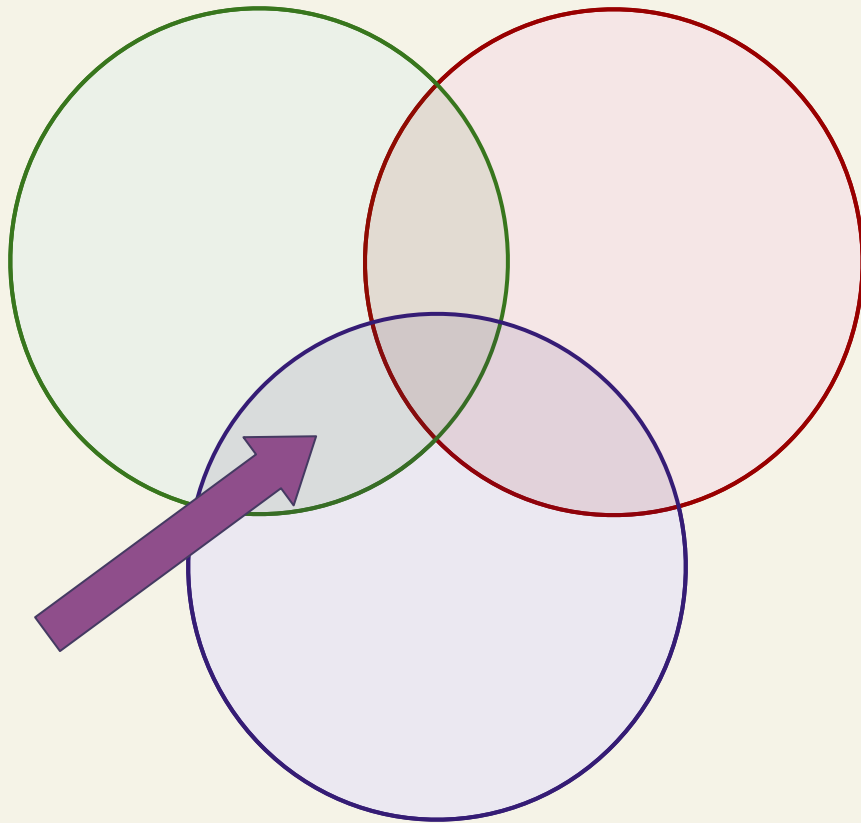


Sound

If the code has type errors,
always report them



Flow



Complete

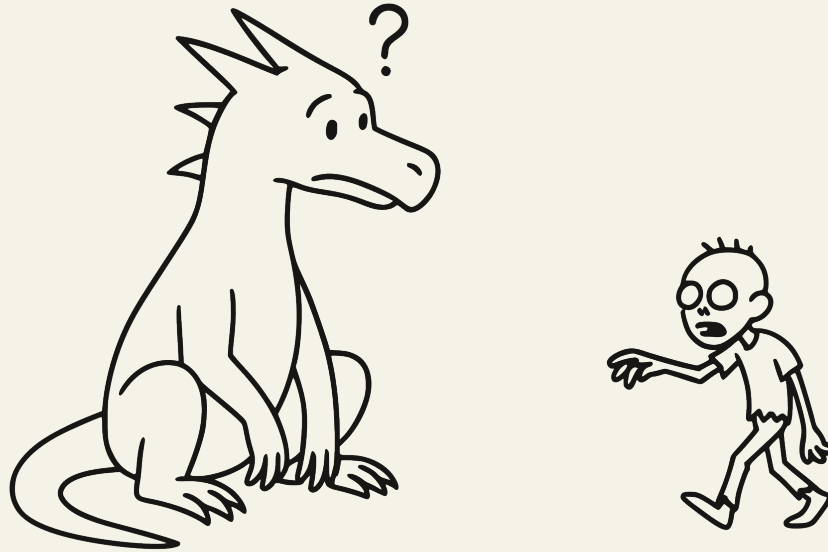
Never report errors that
don't exist



Decidable

Never crash or hang

Not Sound



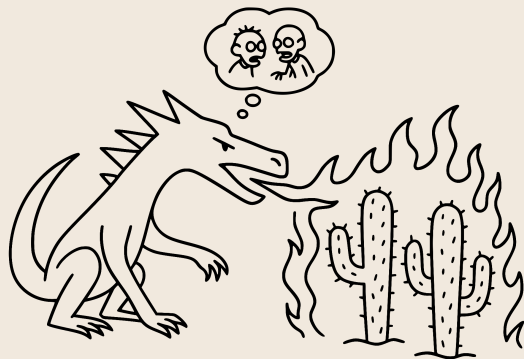
Why didn't you catch this?

Not Sound



*Why didn't you catch
this?*

Not Complete



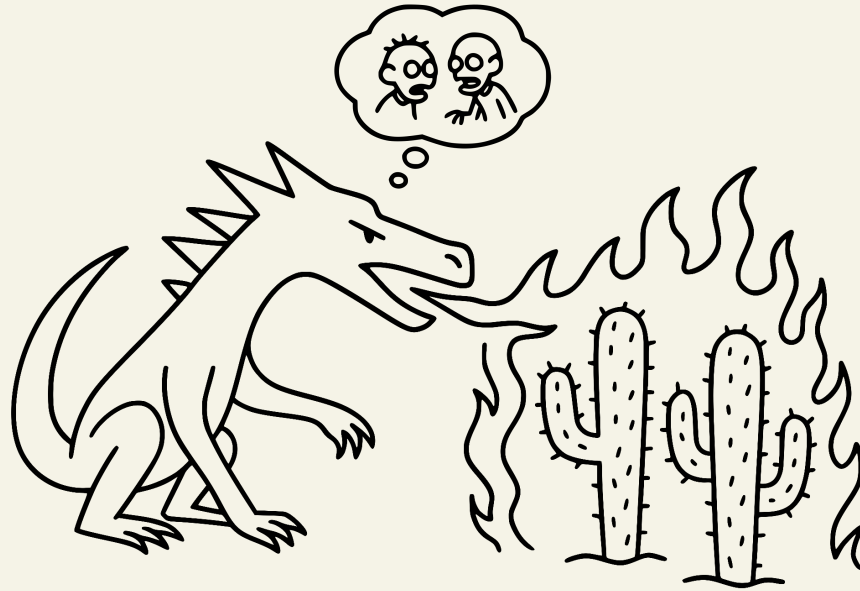
*This works...
why are you bullying me!?*

Something else



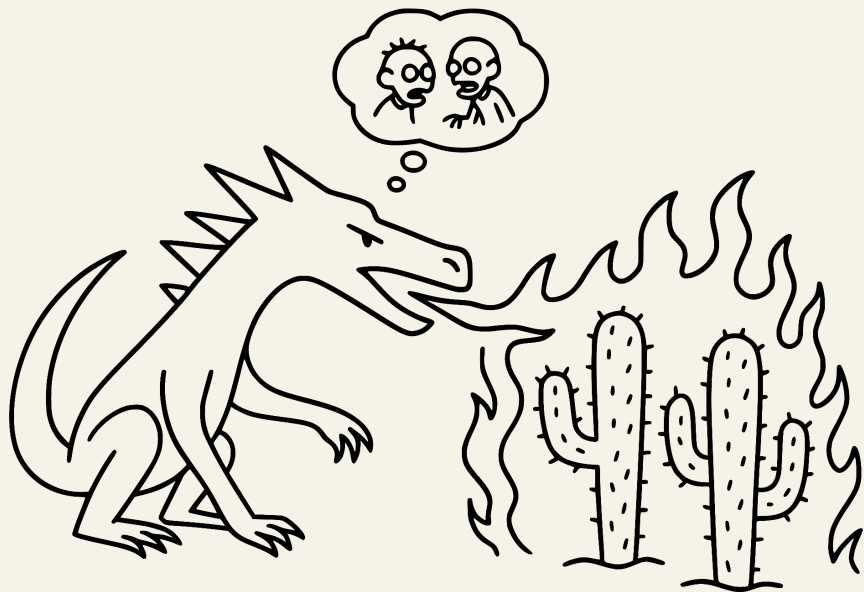
???

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 = 340110998  
setPopulation(data)
```

```
data = "English"  
setLanguage(data)
```

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

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

```
let data = 340110998  
setPopulation(data)
```

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

```
data = "English"  
setLanguage(data)
```

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

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

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

```
let data = 340110998  
setPopulation(data)
```

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

```
data = "English"  
setLanguage(data)
```

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

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

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

data: number



```
let data = 340110998  
setPopulation(data)
```

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

```
data = "English"  
setLanguage(data)
```

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

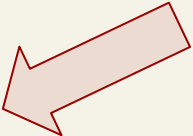
```
function setPopulation(population: number) {  
    // ...  
}
```

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

```
let data = 340110998  
setPopulation(data)
```

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

```
data = "English"  
setLanguage(data)
```



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

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

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

```
let data = 340110998  
setPopulation(data)
```

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

```
data = "English"  
setLanguage(data)
```

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

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

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

```
let data: string | number = 340110998  
setPopulation(data)
```

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

```
data = "English"  
setLanguage(data)
```

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

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

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

```
let data: string | number = 340110998  
setPopulation(data)
```

```
data = "English"  
setLanguage(data)
```

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

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

**Could it have done this
by itself?**

```
let data: string | number = 345175138  
setPopulation(data)
```

```
data = "English"  
setLanguage(data)
```

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

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

Yes, absolutely!

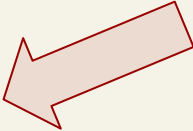
```
let data: string | number = 340110998  
setPopulation(data)
```

```
data = "English"  
setLanguage(data)
```

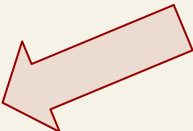
```
function setPopulation(population: number) {  
    // ...  
}
```

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

```
let data: string | number = 340110998  
setPopulation(data)
```



```
data = "English"  
setLanguage(data)
```

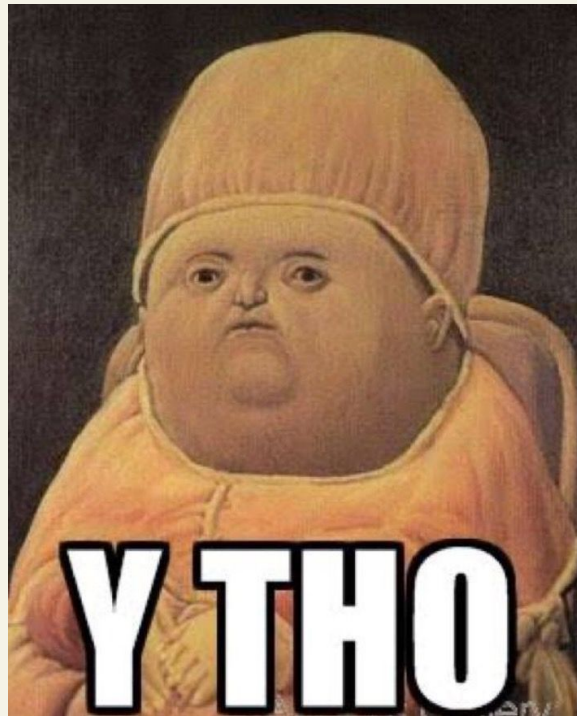


```
function setPopulation(population: number) {  
    // ...  
}
```

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

```
let data: string | number = 340110998  
setPopulation(data)
```

```
data = "English"  
setLanguage(data)
```



Why not infer the correct type?

- 1. Simplicity and consistency**
- 2. It could (or should) be a mistake**

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

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

```
let data: string | number = 340110998  
setPopulation(data)
```

```
data = "English"  
setLanguage(data)
```

[Playground](#)

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

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

```
const population = 340110998  
setPopulation(population)
```

```
const language = "English"  
setLanguage(language)
```

[Playground](#)

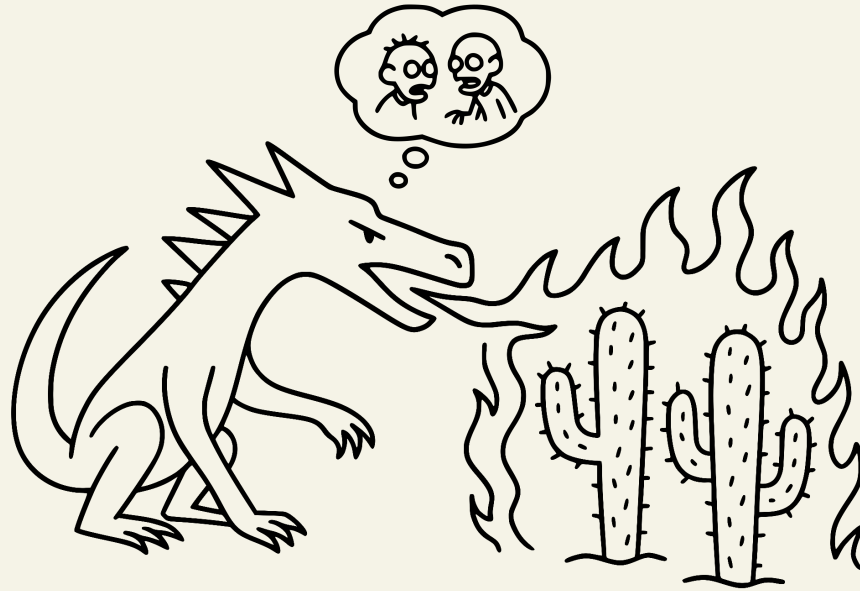
```
function setPopulation(population: number) {  
  // ...  
}
```

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

```
const population = // computation that returns a number  
setPopulation(population)
```

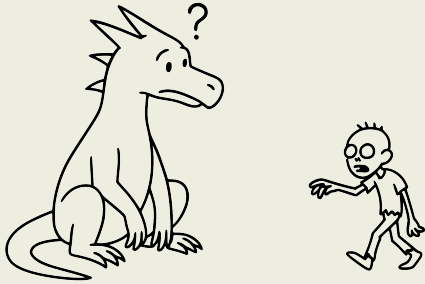
```
const language = // computation that returns a string  
setLanguage(language)
```

Not Complete



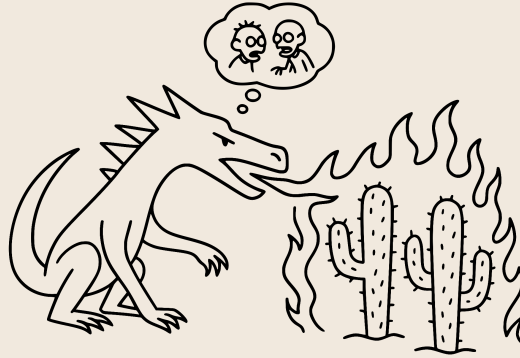
This works... why are you bullying me!?

Not Sound



*Why didn't you catch
this?*

Not Complete



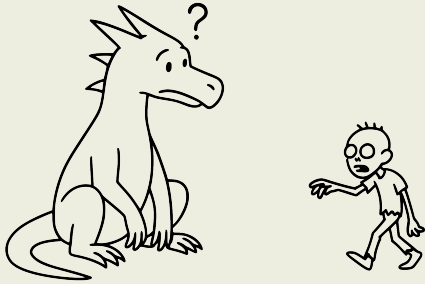
*This works...
why are you bullying me!?*

Something else



???

Not Sound



*Why didn't you catch
this?*

Not Complete



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

Something else



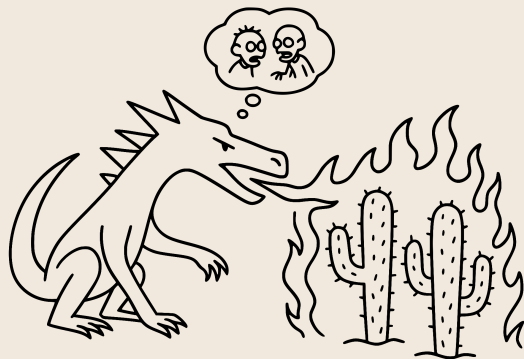
???

Not Sound



*Why didn't you catch
this?*

Not Complete



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

Something else



???

Not Sound



*Why didn't you catch
this?*

Not Complete



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

Something else



???

Not Sound



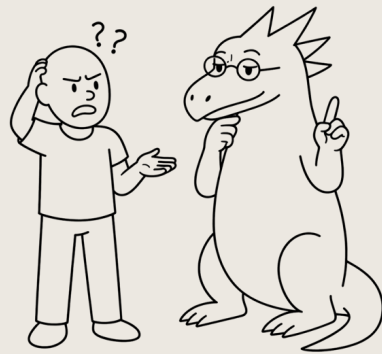
*Why didn't you catch
this?*

Not Complete



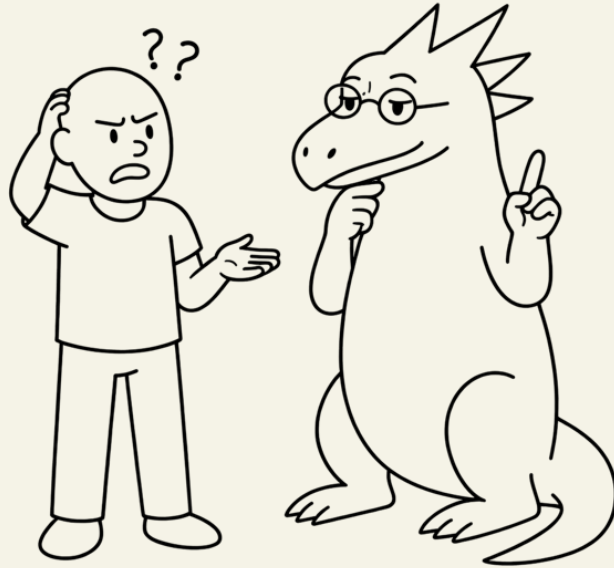
*This works...
why are you bullying me!?*

Actually Right?



But we just don't get it

Actually Right?



But we just don't get it

When is TypeScript **Actually** Right?



But we just don't get it

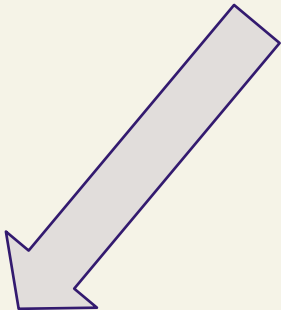
Object.keys(...)

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

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

```
const italy = { name: 'Italy', capital: 'Rome' }  
logCountry(italy)
```

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



```
function logCountry(country: Country) {
```

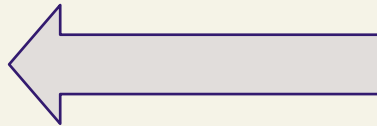
```
}
```

```
const italy = { name: 'Italy', capital: 'Rome' }
```

```
logCountry(italy)
```

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

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

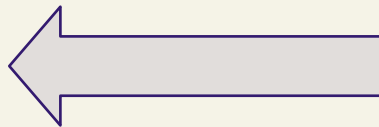


```
}
```

```
const italy = { name: 'Italy', capital: 'Rome' }  
logCountry(italy)
```

```
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 italy = { name: 'Italy', capital: 'Rome' }  
logCountry(italy)
```

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

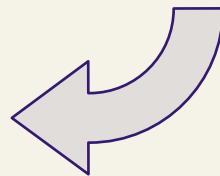
```
function logCountry(country: Country) {  
  const countryKeys = Object.keys(country)  
  for (const key of countryKeys) {  
    console.log(key, country[key])  
  }  
}
```

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

```
const italy = { name: 'Italy', capital: 'Rome' }  
logCountry(italy)
```

```
interface Country {  
  name: string  
  capital: string | null  
}  
countryKeys: string[]
```

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

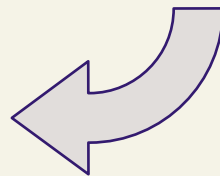


ERROR: ... expression of type 'string' can't be used to index type 'Country'.

```
const italy = { name: 'Italy', capital: 'Rome' }  
logCountry(italy)
```

```
interface Country {  
  name: string  
  capital: string | null  
}  
countryKeys: (keyof Country)[]
```

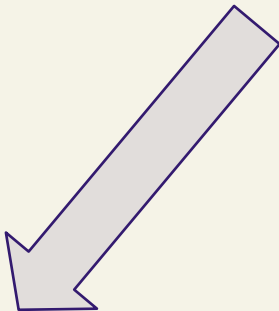
```
function logCountry(country: Country) {  
  const countryKeys = Object.keys(country)  
  for (const key of countryKeys) {  
    console.log(key, country[key])  
  }  
}
```



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

```
const italy = { name: 'Italy', capital: 'Rome' }  
logCountry(italy)
```

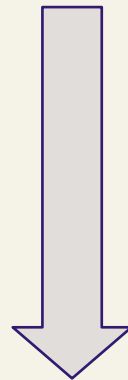
```
interface Country {  
  name: string  
  capital: string | null  
}
```



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

```
const italy = { name: 'Italy', capital: 'Rome' }  
logCountry(italy)
```

```
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 italy = { name: 'Italy', capital: 'Rome', population: 58994323 }  
logCountry(italy)
```

[Playground](#)

There's a lot of this stuff!

O'REILLY®

Second
Edition

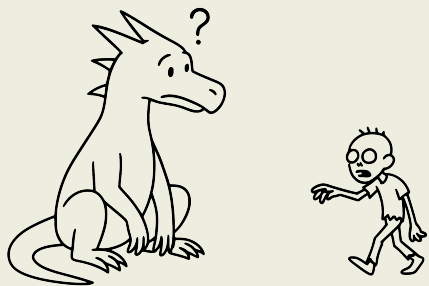
Effective TypeScript

83 Specific Ways to Improve Your TypeScript



Dan Vanderkam

Not Sound



*Why didn't you catch
this?*

Not Complete



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

Actually right?



But we just don't get it

Not Sound



*Why didn't you catch
this?*

Not Complete



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

Actually right?



But we just don't get it

Not Sound



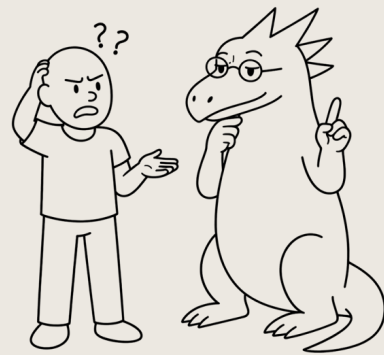
*Why didn't you catch
this?*

Not Complete

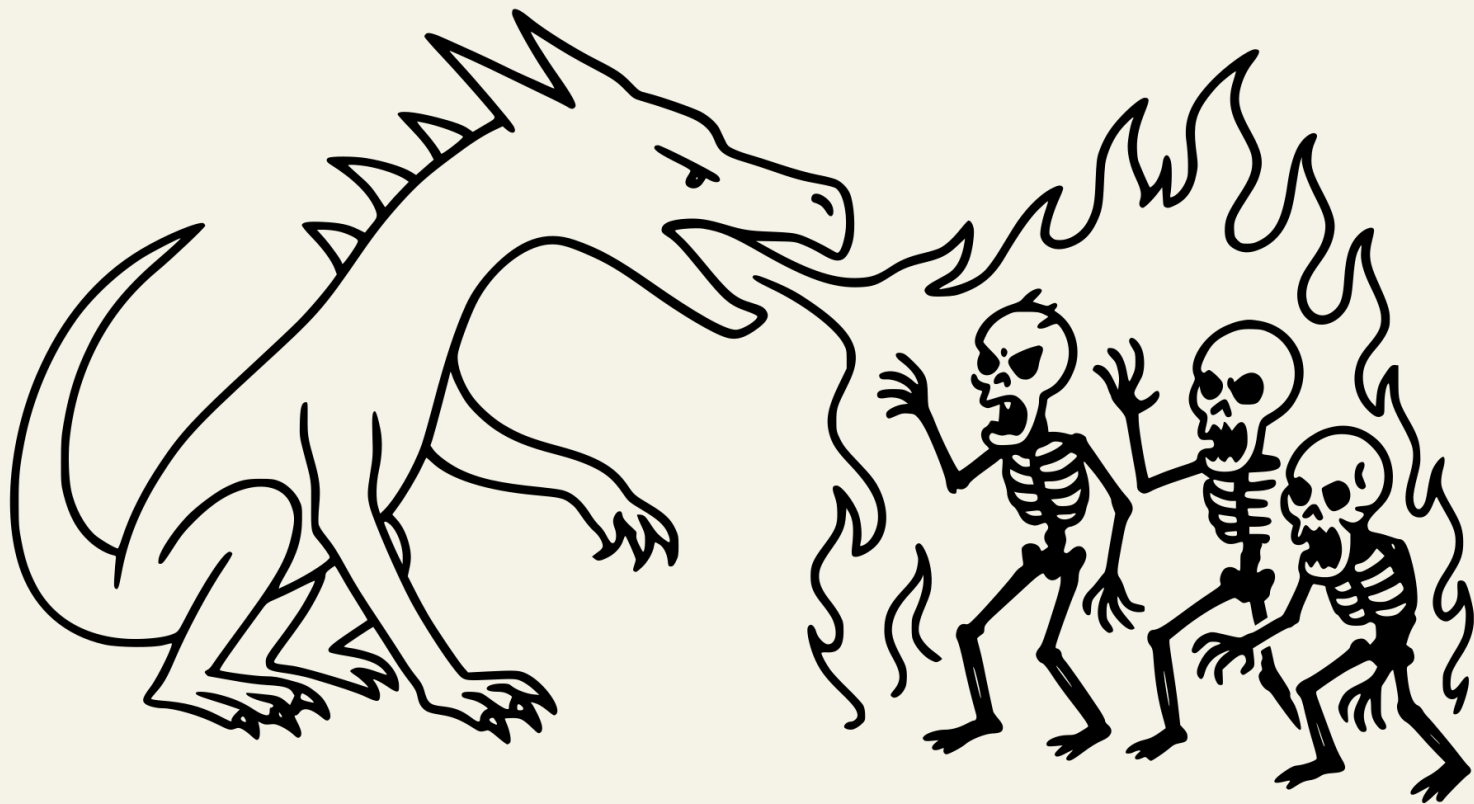


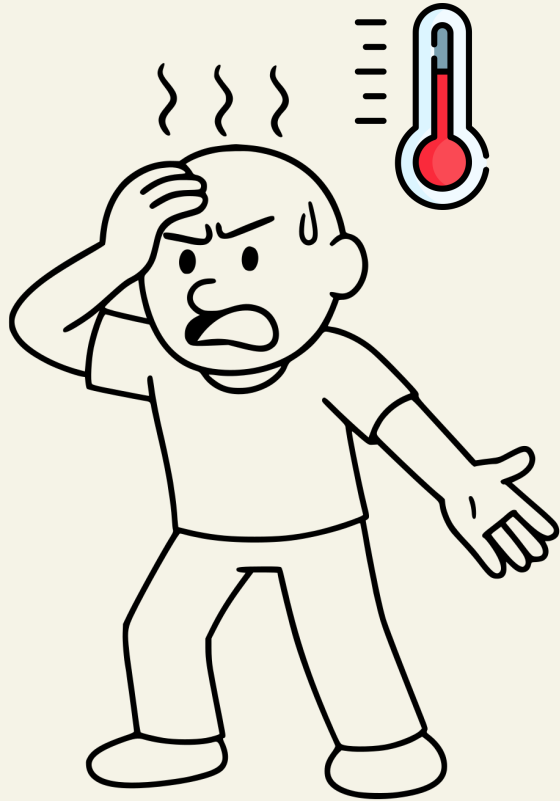
*This works...
why are you bullying me!?*

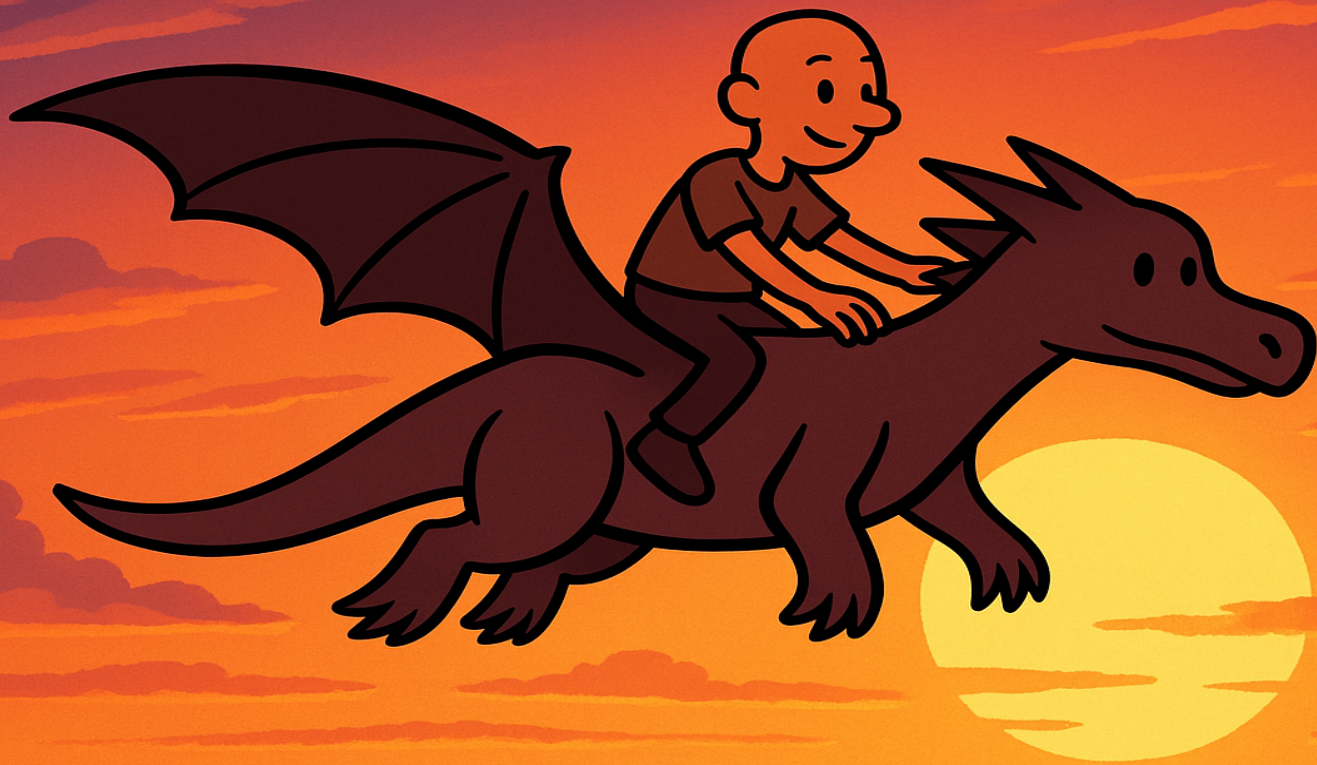
Actually right?



But we just don't get it







The Compiler Is Our Friend

Feedback and questions:



[in/filipsodic/](https://www.linkedin.com/company/filipsodic/)



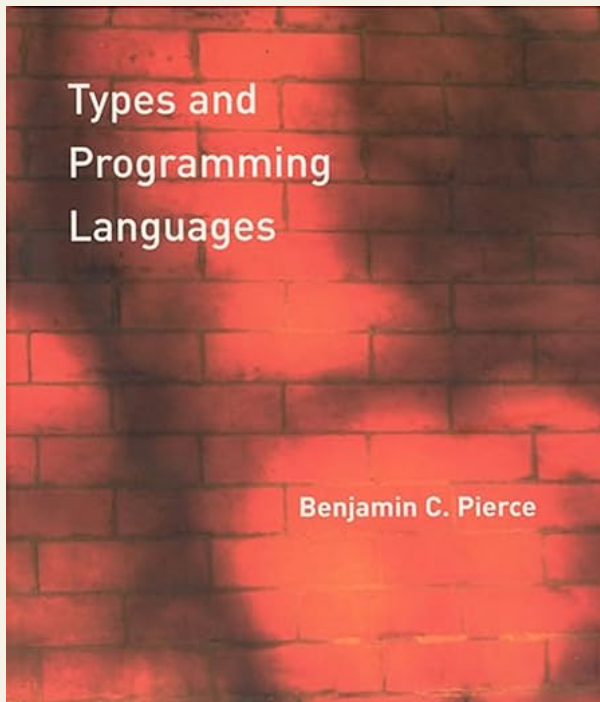
filip@wasp-lang.dev



[@filipsodic](https://twitter.com/filipsodic)
[@WaspLang](https://twitter.com/WaspLang)

Extra slides

I *really* like this, even beyond TypeScript!



Haskell

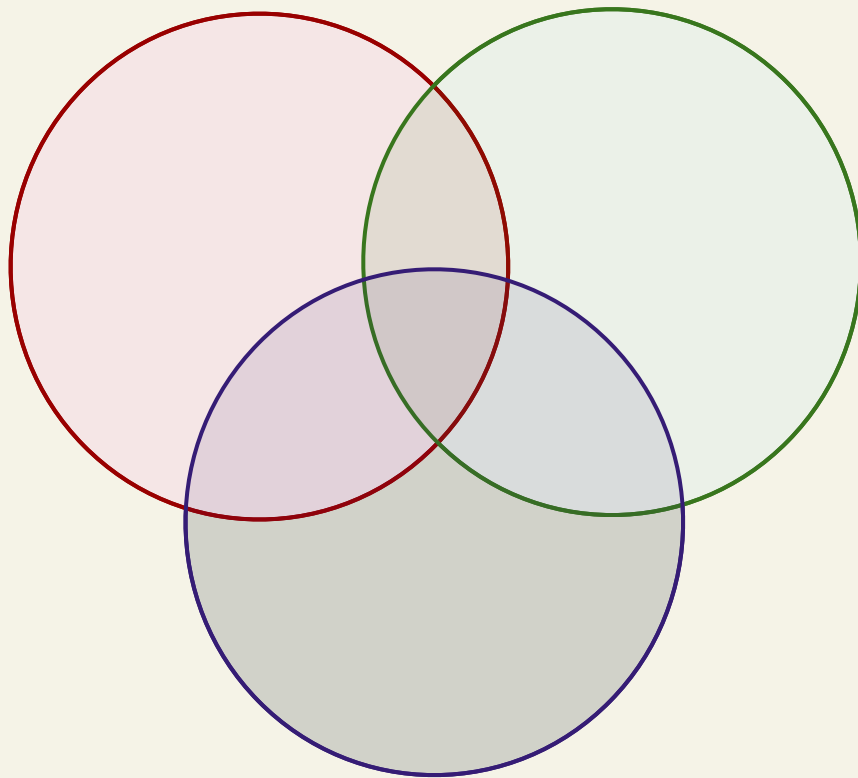


Sound

If the code has type errors,
always report them

Complete

Never report errors
that don't exist



Decidable

Never crash or hang

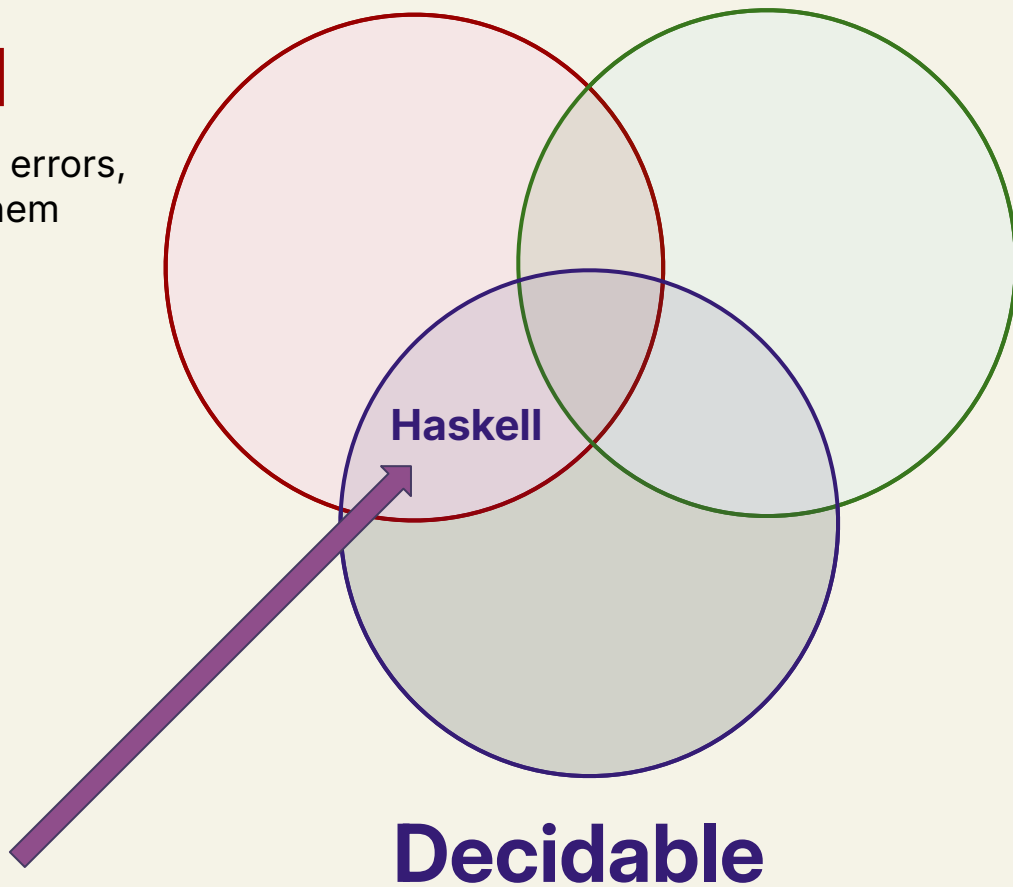


Sound

If the code has type errors,
always report them

Complete

Never report errors
that don't exist



Haskell

Decidable

Never crash or hang



Sound

If the code has type errors,
always report them

Complete

Never report errors
that don't exist

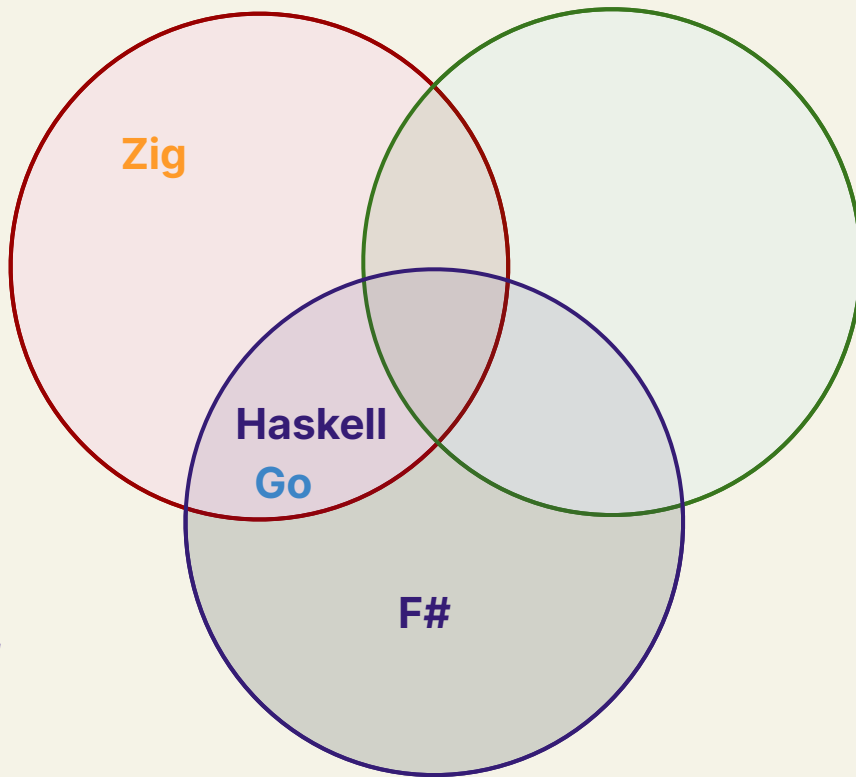
Scala

Java

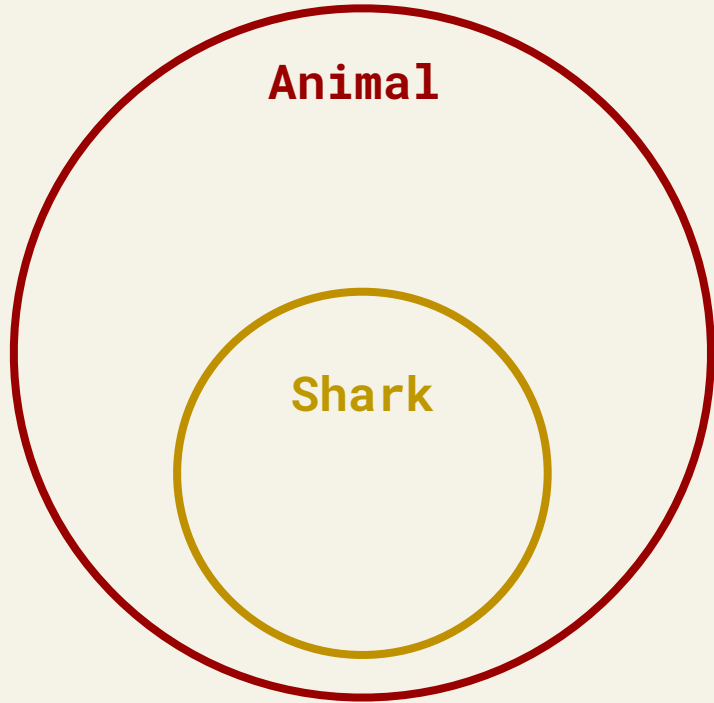
C#

Decidable

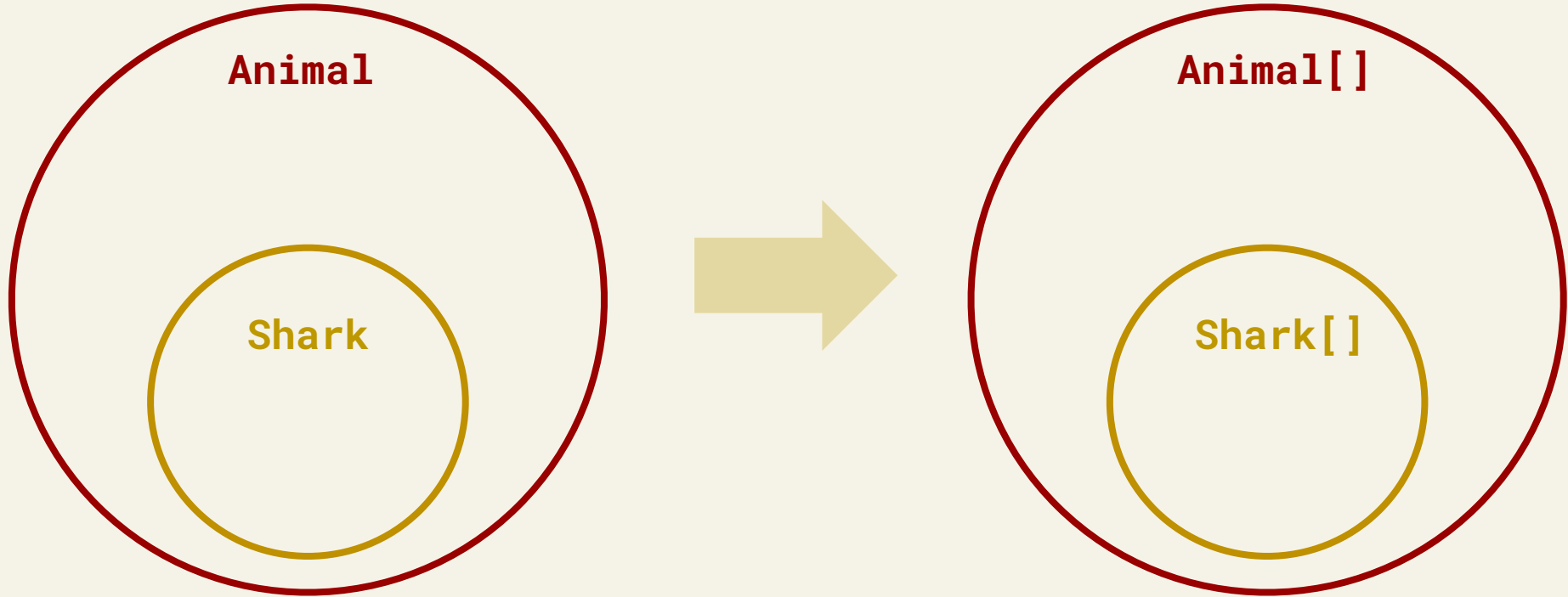
Never crash or hang



Soundness tradeoffs



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](#)