

# TYPESCRIPT



# Hello

**Kamil Richert**

Senior Software Engineer at Atlassian

# TYPESCRIPT

TypeScript extends JavaScript with the ability to be typed. We are able to catch more bugs before the software goes into production.



# What does TypeScript give us?

1. Facilitates control over the application
2. Better code readability
3. Force smaller functions
4. Hints in the code editor
5. Validates types during compilation
6. Each JS code is a valid TS code
7. TS is finally compiled into JS

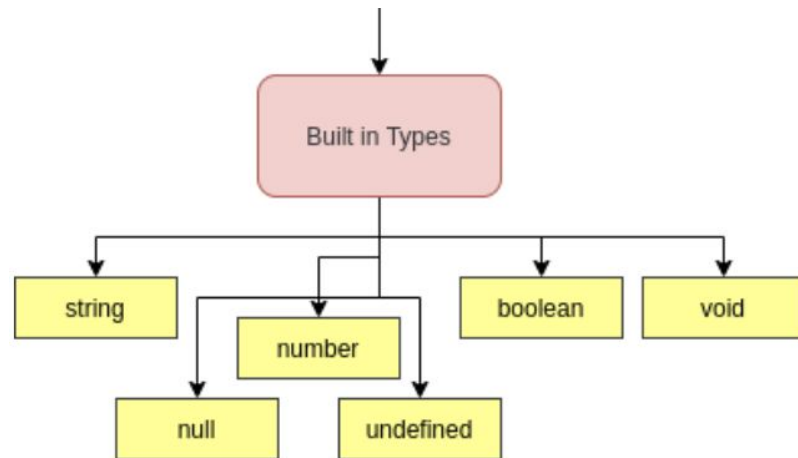
# What are the disadvantages?

1. It slows down software release time
2. Additional configuration at the start of the project

# Basic types

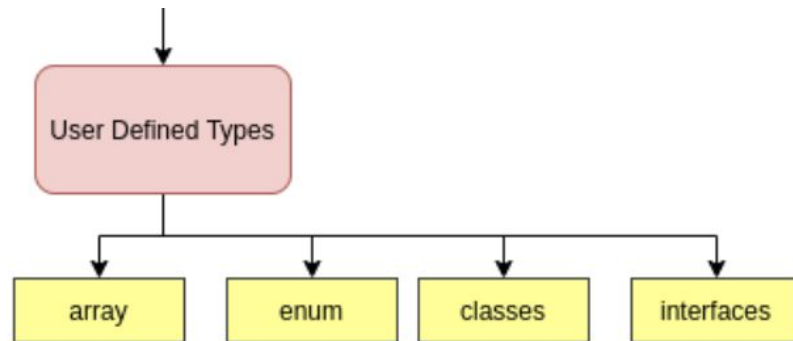
- string
- number
- boolean
- void
- null
- undefined

- never
- any
- unknown



# Complex types

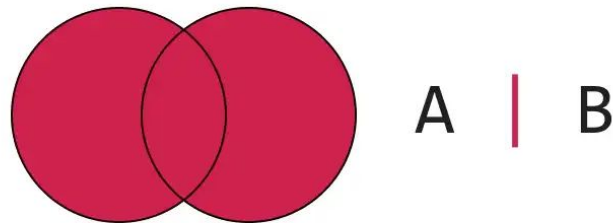
- array
- enum
- classes
- interfaces



# Union & Intersection

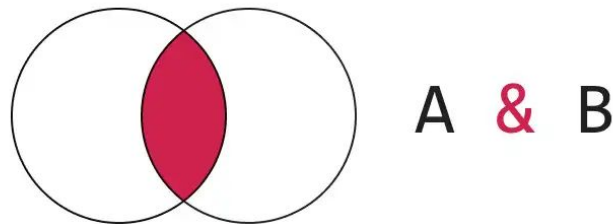
## Union

type C = type A | type B



## Intersection

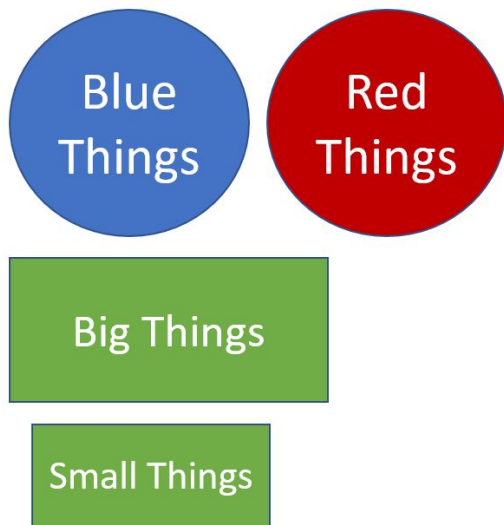
type C = type A & type B



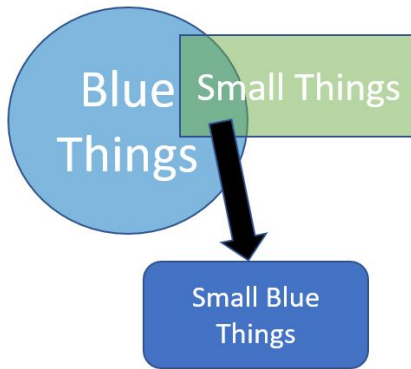


# Union & Intersection

Consider classifying objects four ways: blue, red, big, and small

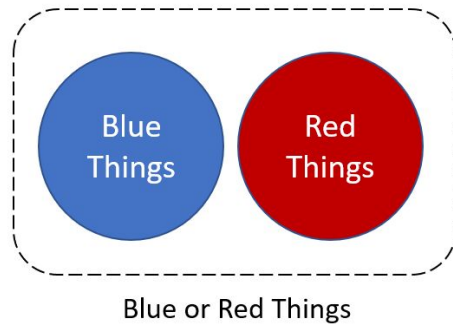


If we *intersect* **blue** with **small**, we get a new set:



The *intersection* of these sets has the *union* of its properties

If we *union* **blue** with **red**, we get a new set:



The *union* of these sets has the *intersection* of its properties, which in this case is empty

# Generic types

If we want to create a reusable type, but one element changes for example, then you can use a generic type.

```
identity<Type>(arg: Type): Type {  
    return arg;  
}
```

# Typing tools

Required

Partial

Pick

Omit

...

<https://www.typescriptlang.org/docs/handbook/utility-types.html>

```
type PartialCars = Partial<Cars>;
```

```
type RequiredUser = Required<User>;
```

```
type PickCars = Pick<Cars, 'model'>
```

```
type OmitCars = Omit<Cars, 'id'>
```



# Thanks

You can find me:

<https://www.linkedin.com/in/kamil-richert/>

<https://github.com/krichert>