主办方：Geekbang. 极客邦科技 InfoQ

QCon

全球软件开发大会【上海站】

TypeScript的发展历程

吴名扬

4:34 PM
100%
发现
大航海时代
100家互联
极客新闻
对落地DevOps理念的一些反思
2017年Elm语言现状调查结果出炉
陆奇：百度的人工智能战略
Java依旧排名第一，然而变化几何?
专栏订阅
4:34 PM
100%
专栏
重拾极客精神
提升技术认知
TO BE A GEEK
TO GAIN THE TECH
陈皓
网名"左耳朵耗子"，资深技术...
左耳听风

主办方 Geekbang> 极客邦科技 InfoQ

ArchSummit

全球架构师峰会2017

12月8–9日 北京 · 国际会议中心

APSEC 2017

南京大學 NANJING UNIVERSITY × Geekbang> 极客邦科技 | InfoQ

APSEC 2017

24th Asia-Pacific Software Engineering Conference
4-8 December 2017, Nanjing, Jiangsu, China

12月4–8日

中国南京

APSEC

了解详情

AiCon

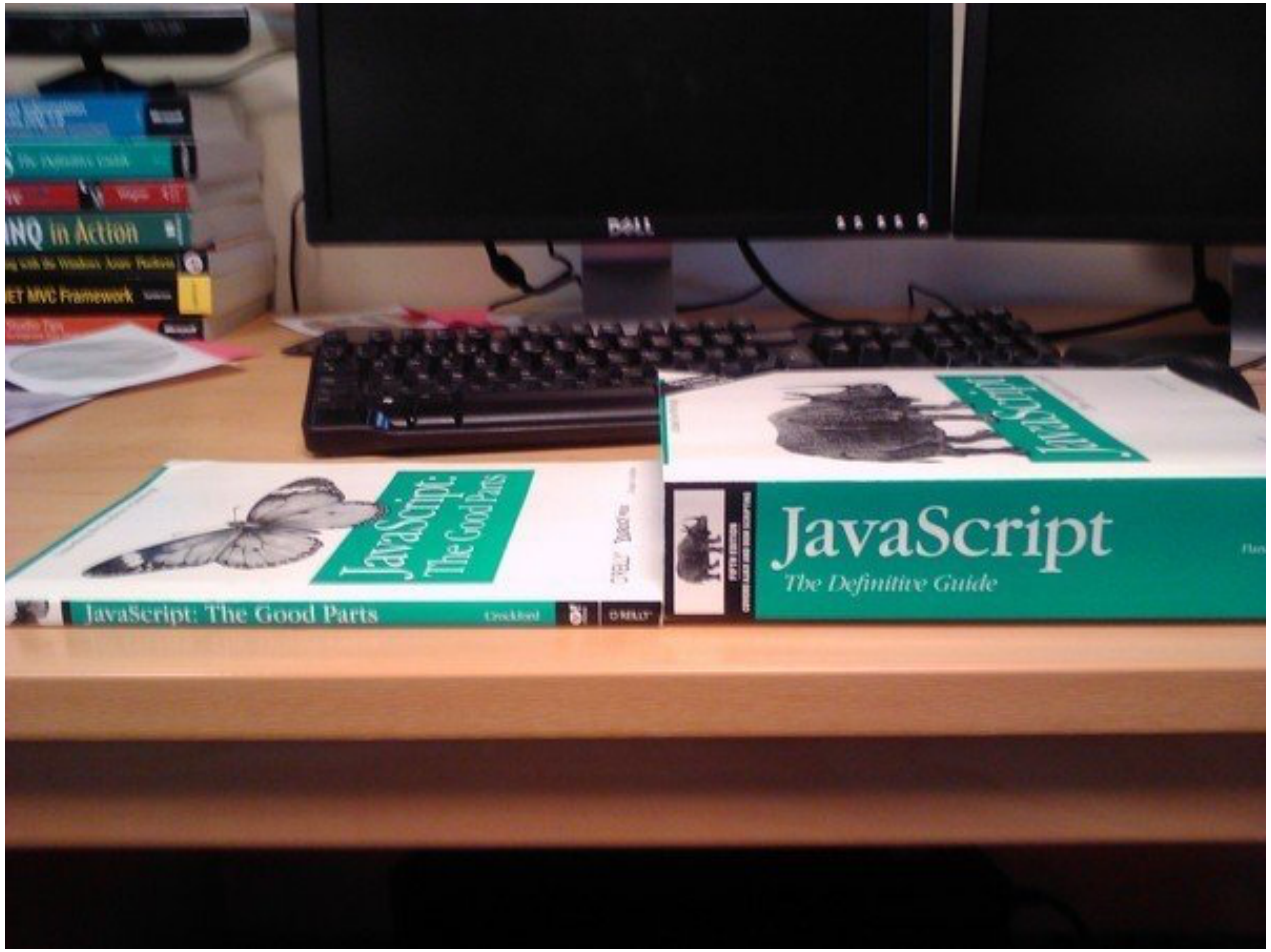
全球人工智能技术大会 2018

助力人工智能落地

2018.1.13 - 1.14　北京国际会议中心

扫描关注大会官网

当年的JavaScript



AltJS

编译到JavaScript的语言
都可称为AltJS

AltJS

Kotlin
by JetBrains
CoffeeScript
ASM.JS
jangaroo
FINALLY
THE BEST OF TWO WORLDS
OBJECTIVE-J
G
DART
TypeScript
PREVIEW
HAXE
pyjs
FündScrip
powered by Fundstream Inc

TypeScript

- JS + 静态类型
- 工具完善
- 贴合语言标准

TS和伙伴们

- 动态类型语法糖： CoffeeScript
- 静态类型新语言: BuckleScript
- 渐进定型: Flow Type

CoffeeScript

- 纯语法糖， 动态类型
- 上手简单
- 工具不良
- 与新标准有冲突

BuckleScript

- 脱胎于OCaml
- 类型系统强大
- 编译高度优化
- 语义语法疏远， 上手难*

Flow Type

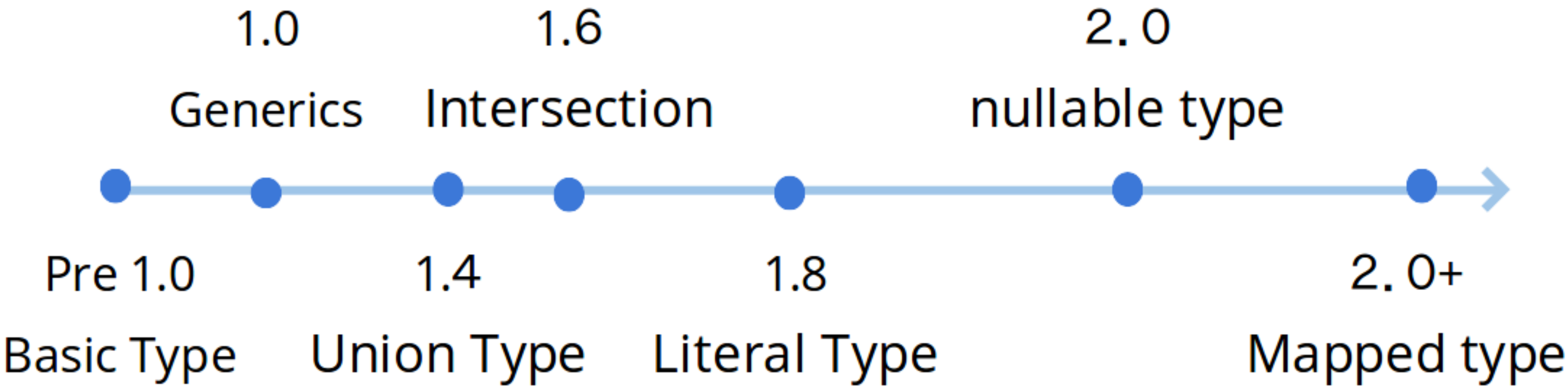
- JS+类型标注
- 工具相对完善
- 上手容易
- 与TS设计相似*

比较	类型系统	难度	工具链
CoffeeScript	动态	低	差
BuckleScript	很强	高	较好
FlowType	强	较低	较好
TypeScript	强	低	很好

TS发展史

- 简单易用,不求完美
- 原汁原味， 贴合JS
- 从简单到强大

TS发展史



Pre 1.0
Basic Type

1.0
Generics

1.4
Union Type

1.6
Intersection

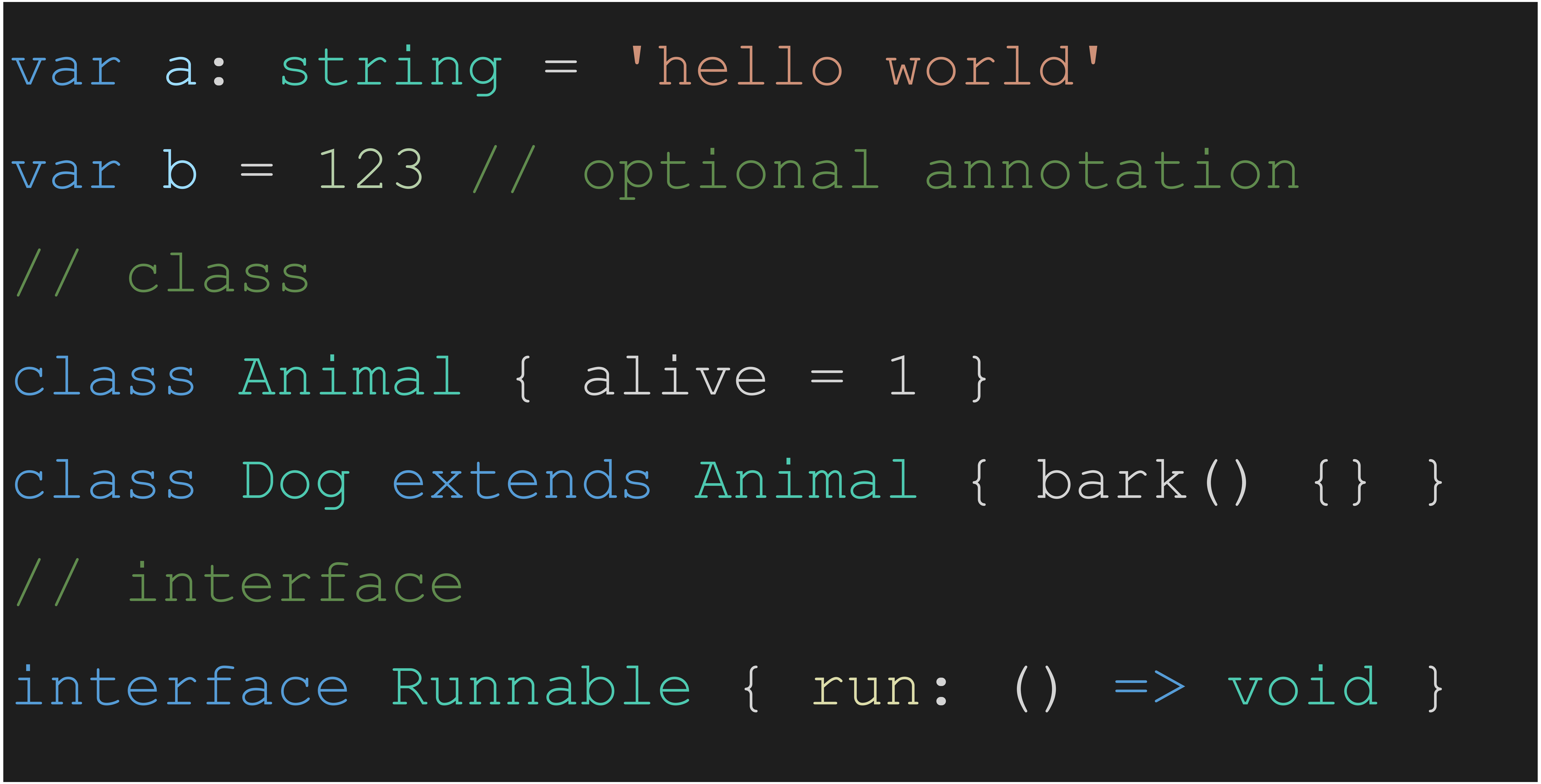
1.8
Literal Type

2. 0
nullable type

2. 0+
Mapped type

Pre 1.0

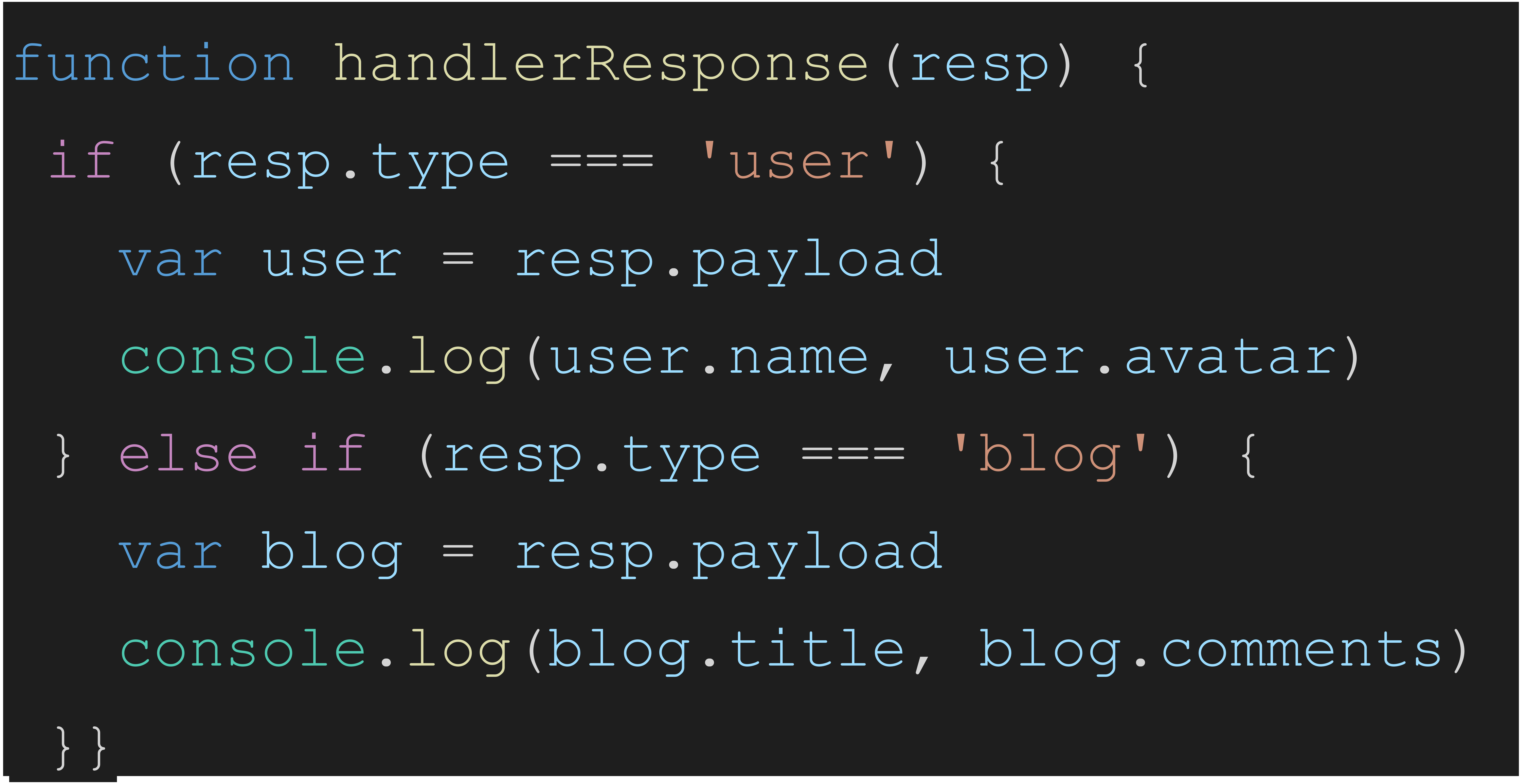
- 简单
- 无泛型



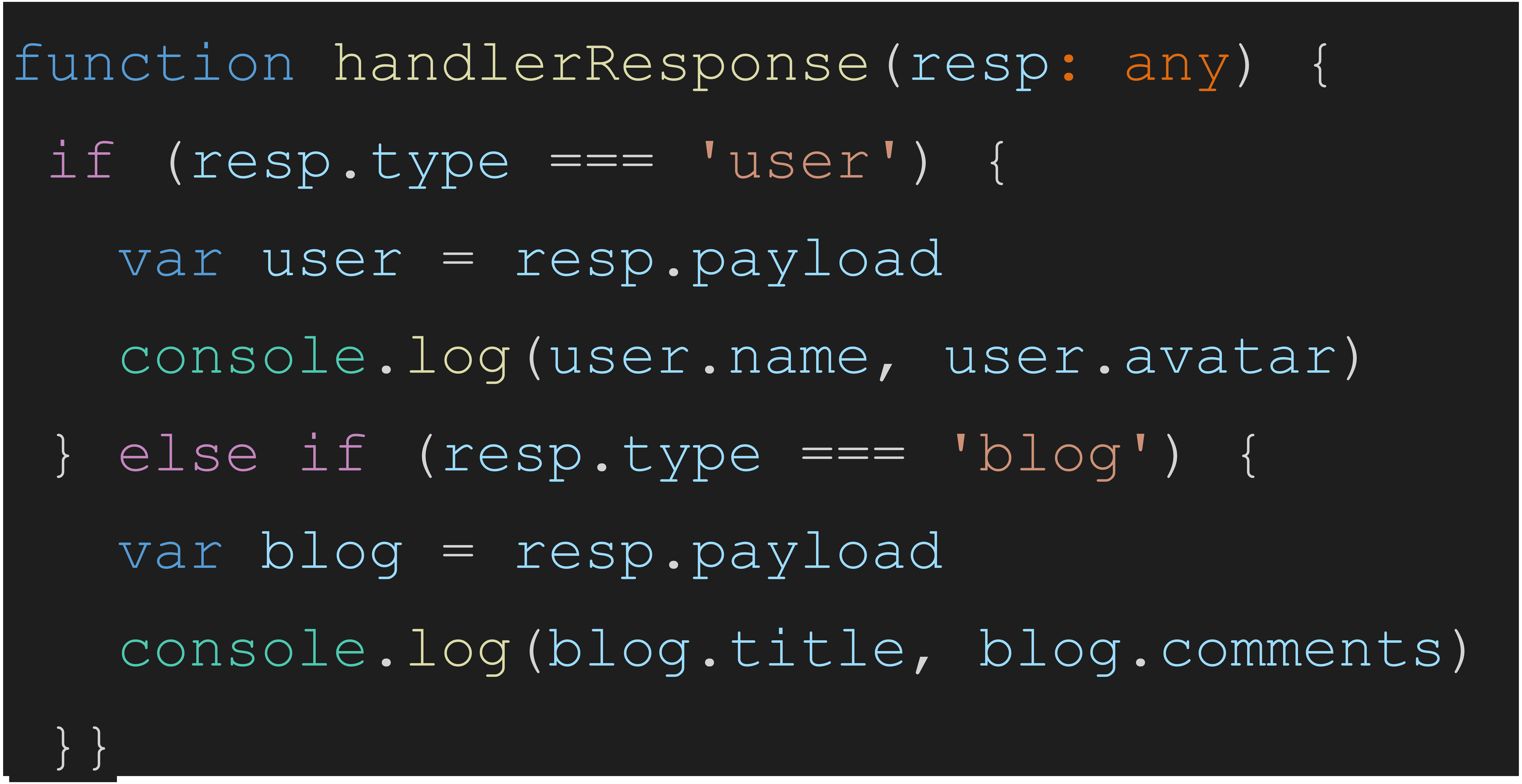
```
var a: string = 'hello world'
var b = 123 // optional annotation
// class
class Animal { alive = 1 }
class Dog extends Animal { bark() {} }
// interface
interface Runnable { run: () => void }
```

渐进定型

Gradual Typing



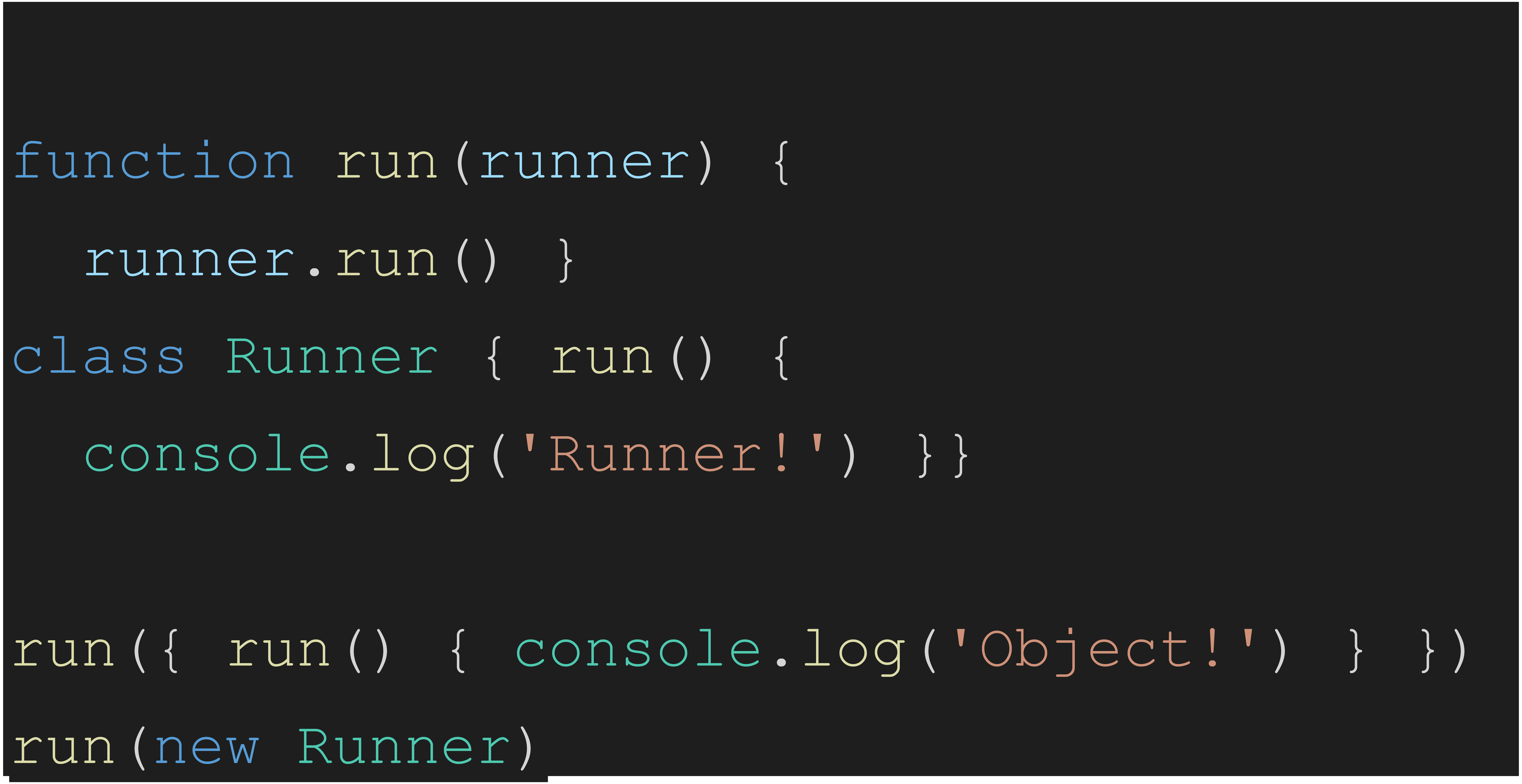
```
function handlerResponse(resp) {
 if (resp.type === 'user') {
   var user = resp.payload
   console.log(user.name, user.avatar)
 } else if (resp.type === 'blog') {
   var blog = resp.payload
   console.log(blog.title, blog.comments)
 }}
```



```
function handlerResponse(resp: any) {
 if (resp.type === 'user') {
   var user = resp.payload
   console.log(user.name, user.avatar)
 } else if (resp.type === 'blog') {
   var blog = resp.payload
   console.log(blog.title, blog.comments)
 }}
```

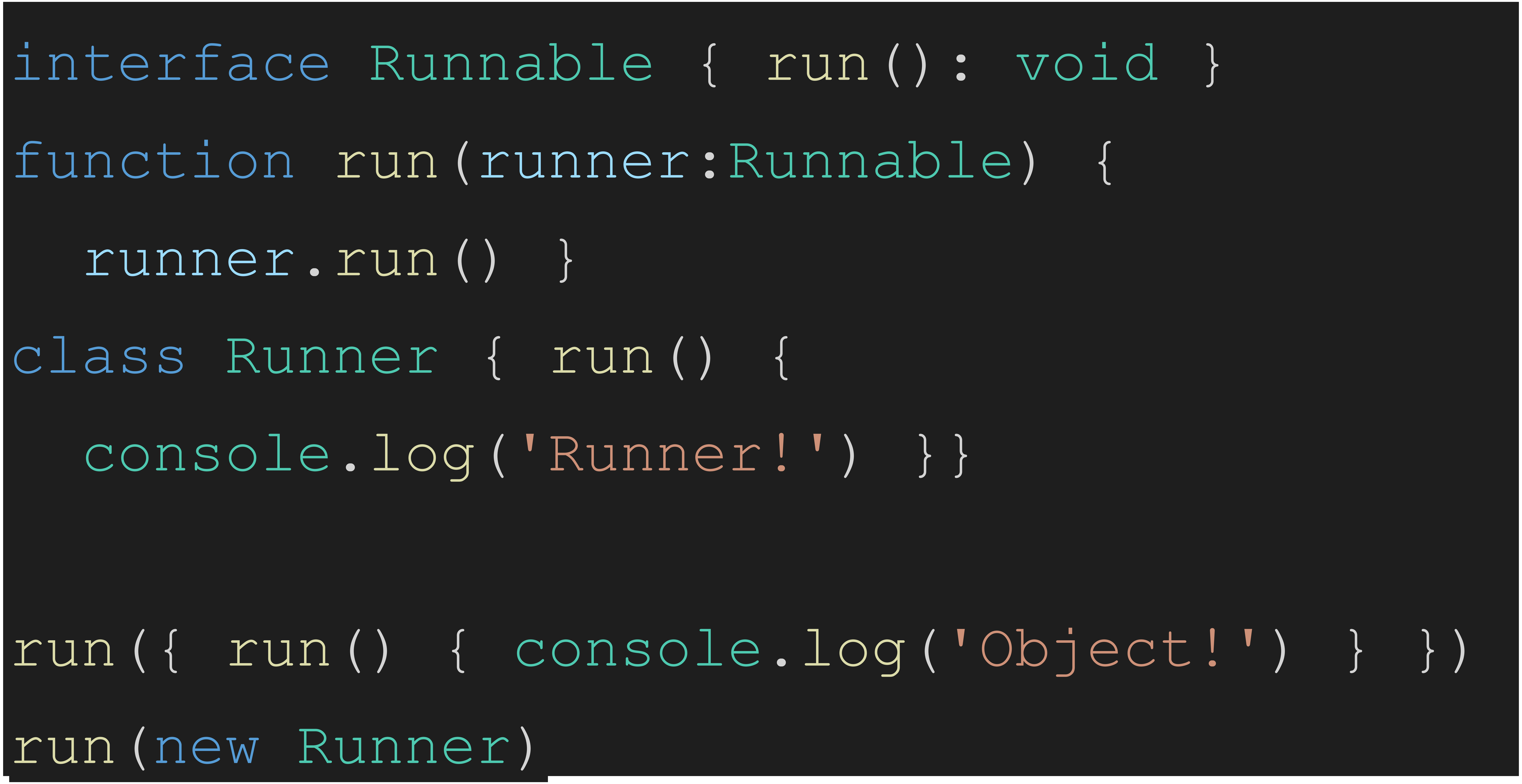
结构定型

Structral Typing



```
function run(runner) {
  runner.run() }
class Runner { run() {
  console.log('Runner!') }}

run({ run() { console.log('Object!') } })
run(new Runner)
```



```
interface Runnable { run(): void }
function run(runner:Runnable) {
  runner.run() }
class Runner { run() {
  console.log('Runner!') }}


run({ run() { console.log('Object!') } })
run(new Runner)
```

没有泛型怎么够

Go

TS 1.0

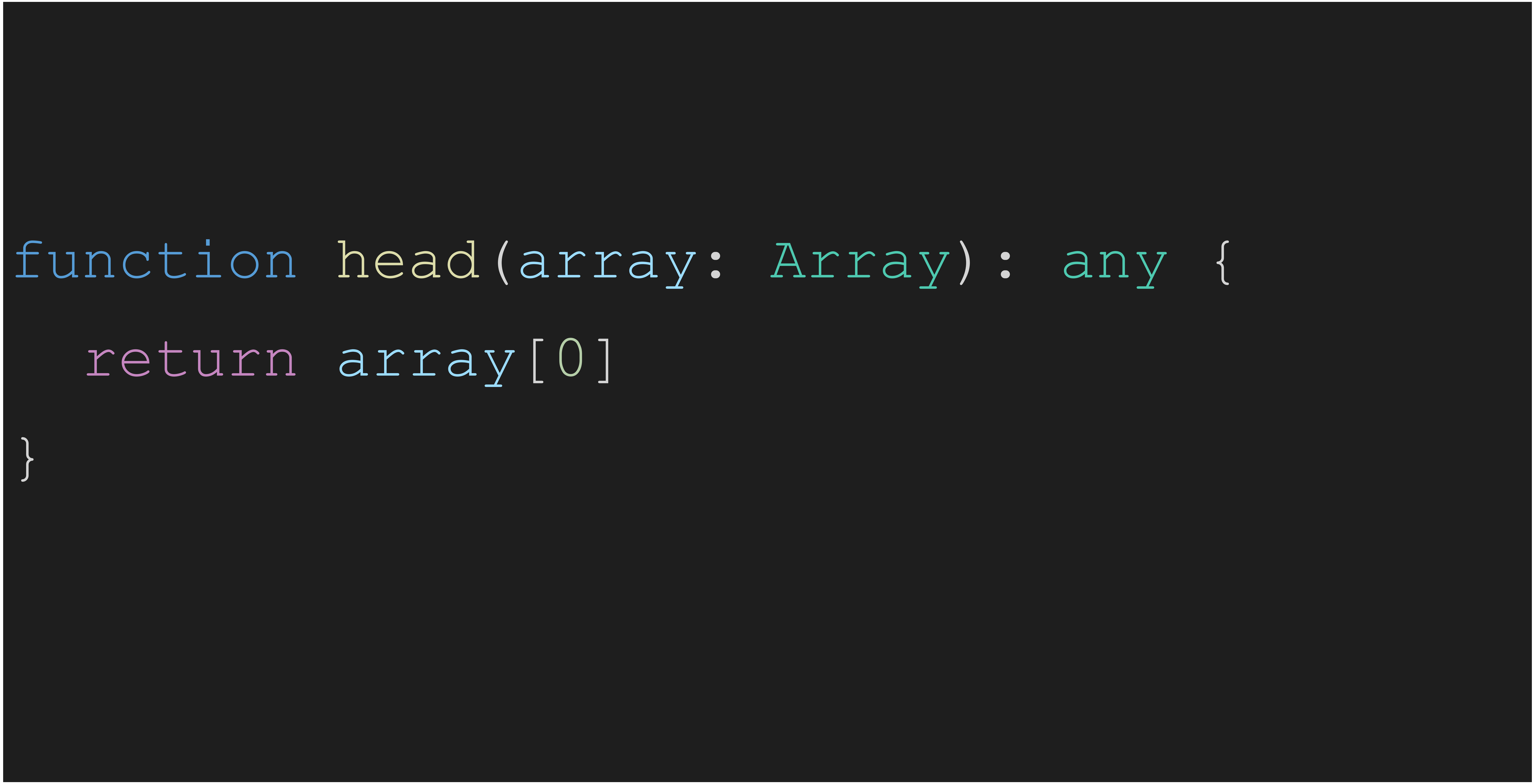
- 引入泛型

- 与Java1.5/C#3.0相当

泛型

Generic

```
function head(array) {
  return array[0]
}
```



```
function head(array: Array): any {
  return array[0]
}
```

```
function head<T>(array: Array<T>): T {
  return array[0]
}
```

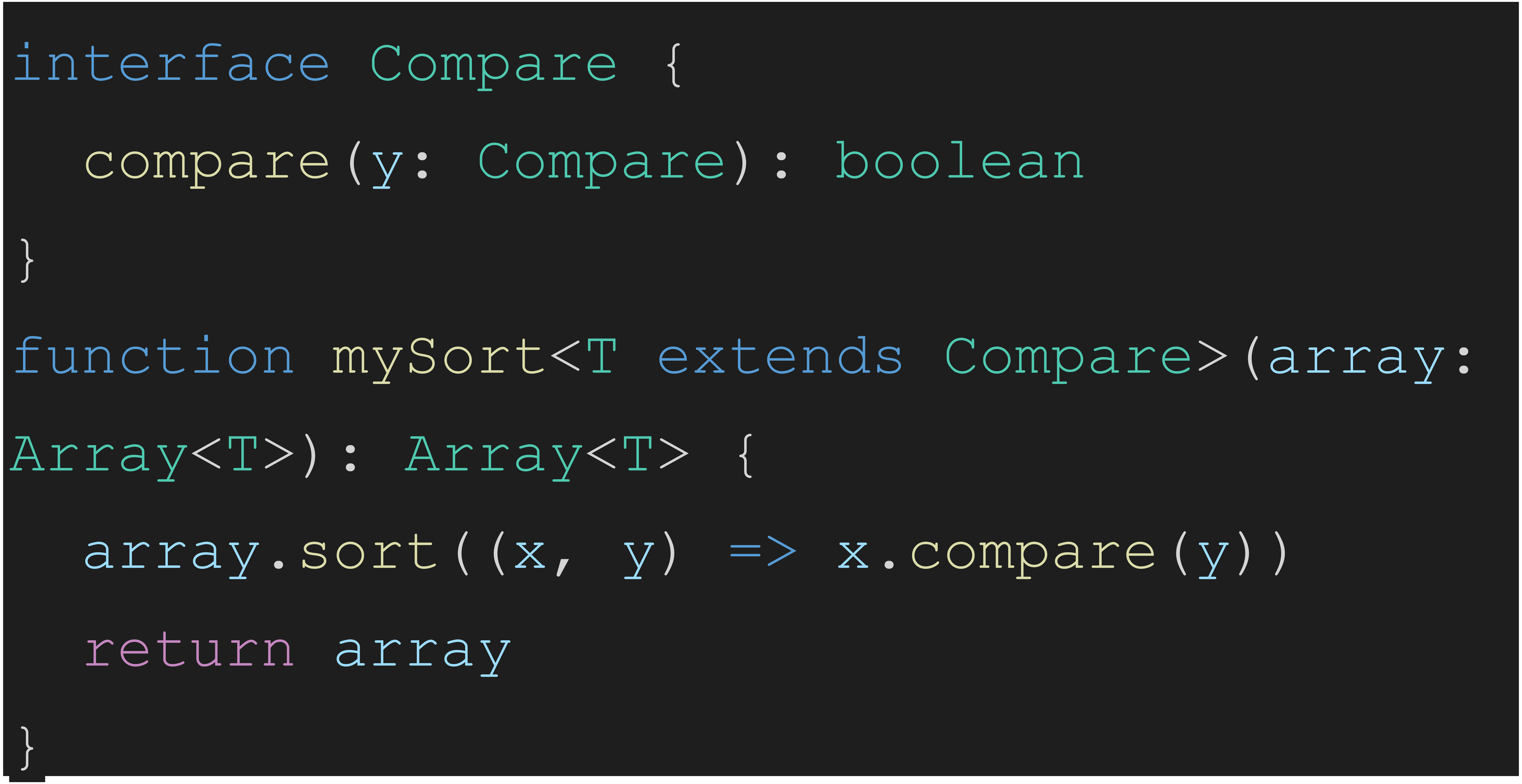
泛型上界

Bounded Generic

```
function mySort(array) {
  array.sort((x, y) => x.compare(y))
  return array
}
```

```
interface Comparable {
  compare(y: Comparable): boolean
}
```

```
interface Compare {
  compare(y: Compare): boolean
}
function mySort(array: Array<Compare>):
Array<Compare> {
  array.sort((x, y) => x.compare(y))
  return array
}
```



```
interface Compare {
  compare(y: Compare): boolean
}
function mySort<T extends Compare>(array:
Array<T>): Array<T> {
  array.sort((x, y) => x.compare(y))
  return array
}
```

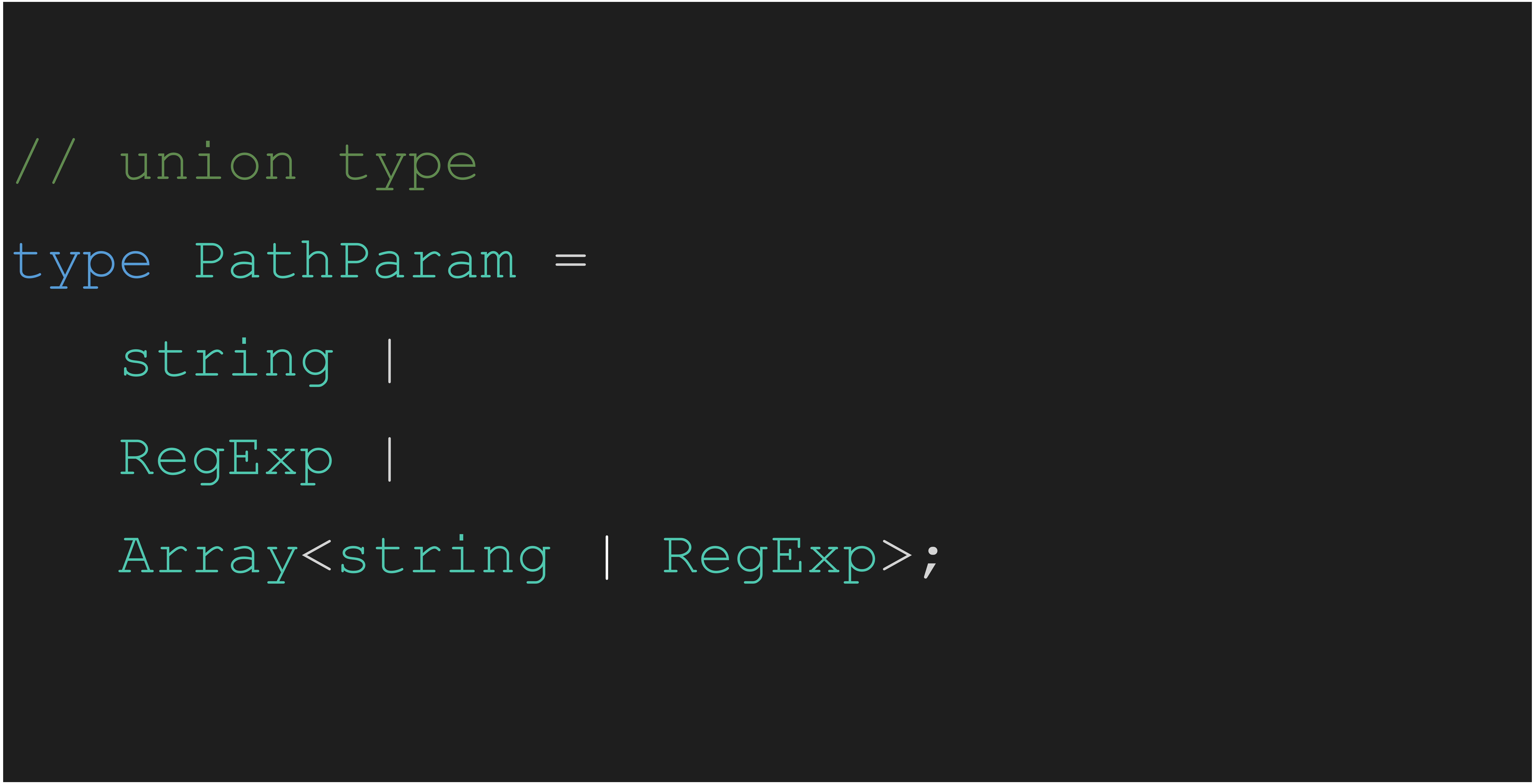
TS 1.4

- 引入联合类型
- 表达力介于Java-Kotlin之间

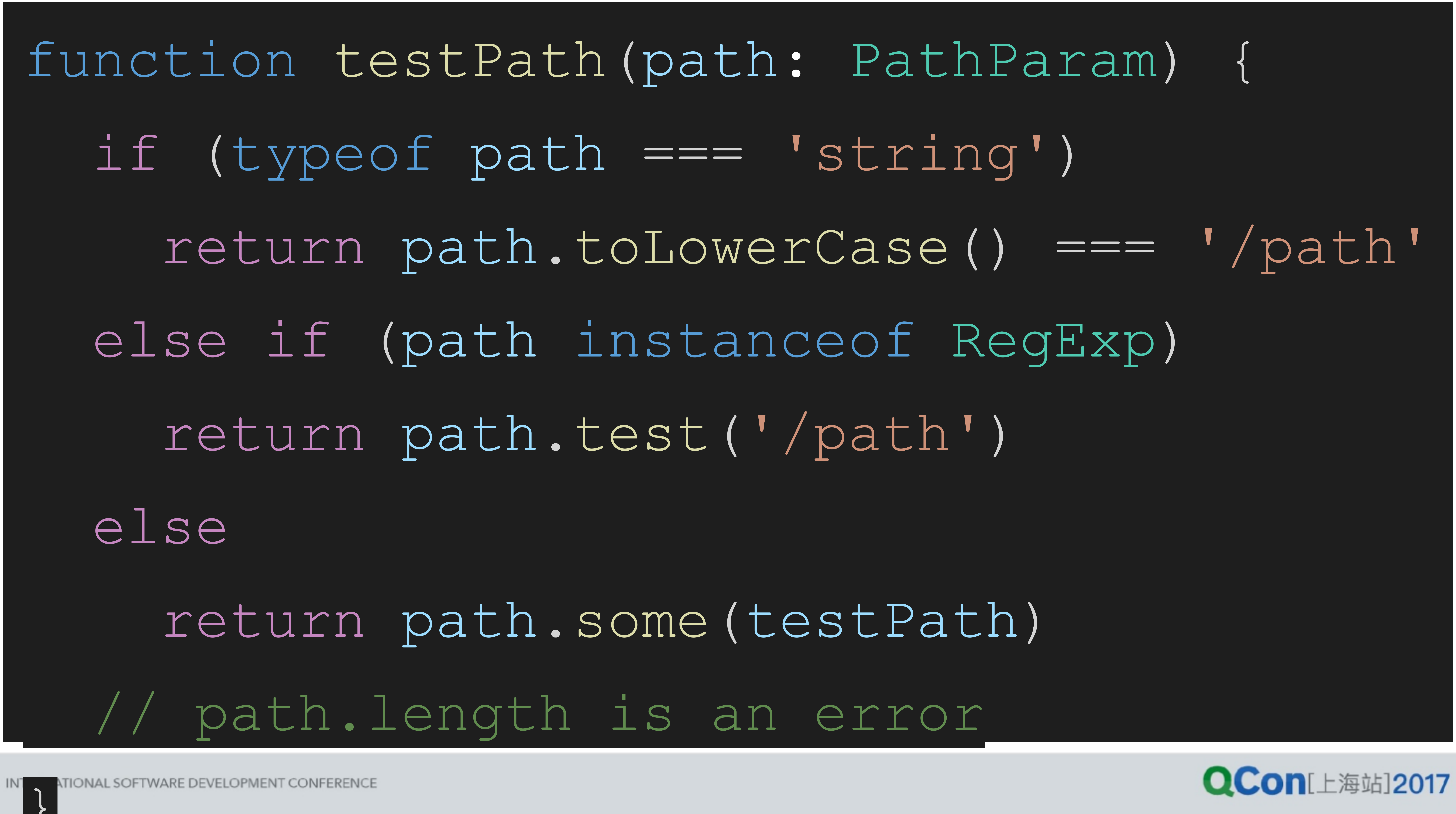
```
// express like API
// string, regex or array

testPath('/path')
testPath(/path/)
testPath(['path', /path/])
```

```
function testPath(path) {
  if (typeof path === 'string')
    return path.toLowerCase() === '/path'
  else if (path instanceof RegExp)
    return path.test('/path')
  else
    return path.some(testPath)
}
```



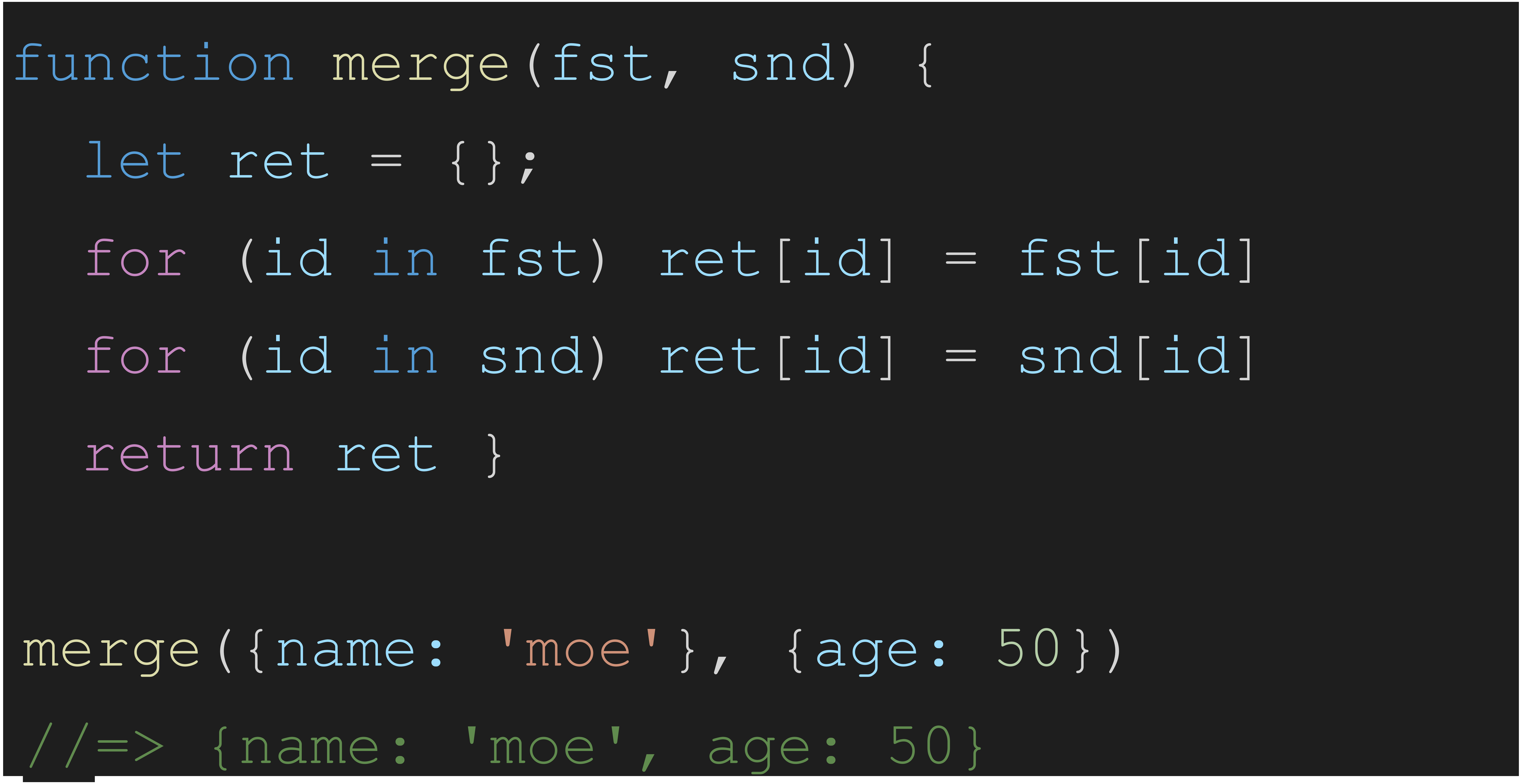
```
// union type
type PathParam =
    string |
    RegExp |
    Array<string | RegExp>;
```



```
function testPath(path: PathParam) {
  if (typeof path === 'string')
    return path.toLowerCase() === '/path'
  else if (path instanceof RegExp)
    return path.test('/path')
  else
    return path.some(testPath)
  // path.length is an error
}
```

TS 1.6

- 引入交集类型
- 源自Flow Type



```
function merge(fst, snd) {
  let ret = {};
  for (id in fst) ret[id] = fst[id]
  for (id in snd) ret[id] = snd[id]
  return ret }


merge({name: 'moe'}, {age: 50})
//=> {name: 'moe', age: 50}
```

```
// manual annotation
interface Merged {
 name: string
 age: number
}

merge<Merged>({name: 'moe'}, {age: 50})
```

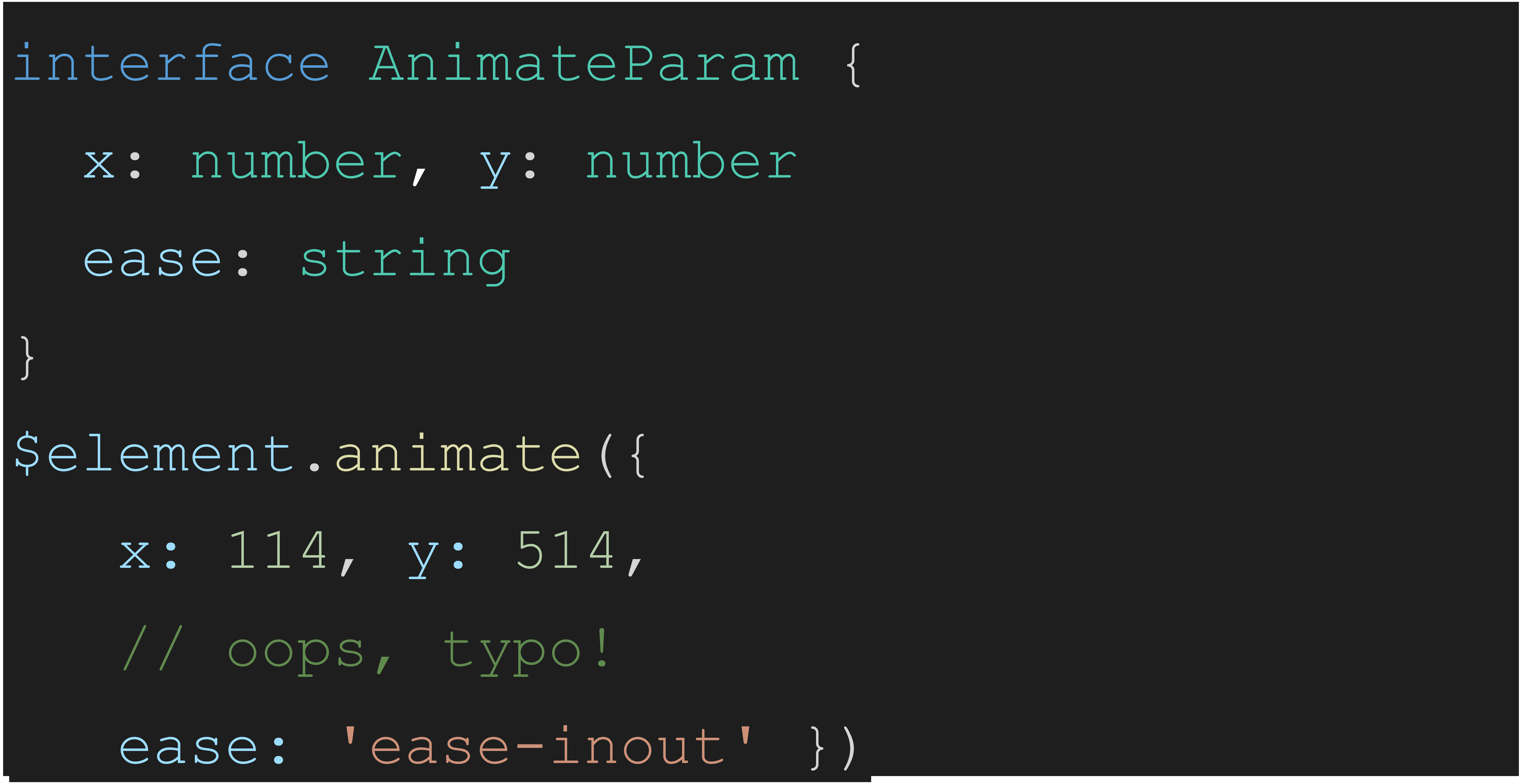
```
function merge<T, U>(fst:T,snd:U):T&U {
  let ret = <any> {}
  for (let id in fst) ret[id] = fst[id]
  for (let id in snd) ret[id] = snd[id]
  return ret
}
// no annotation!
merge({name: 'moe'}, {age: 50})
```

TS 1.8

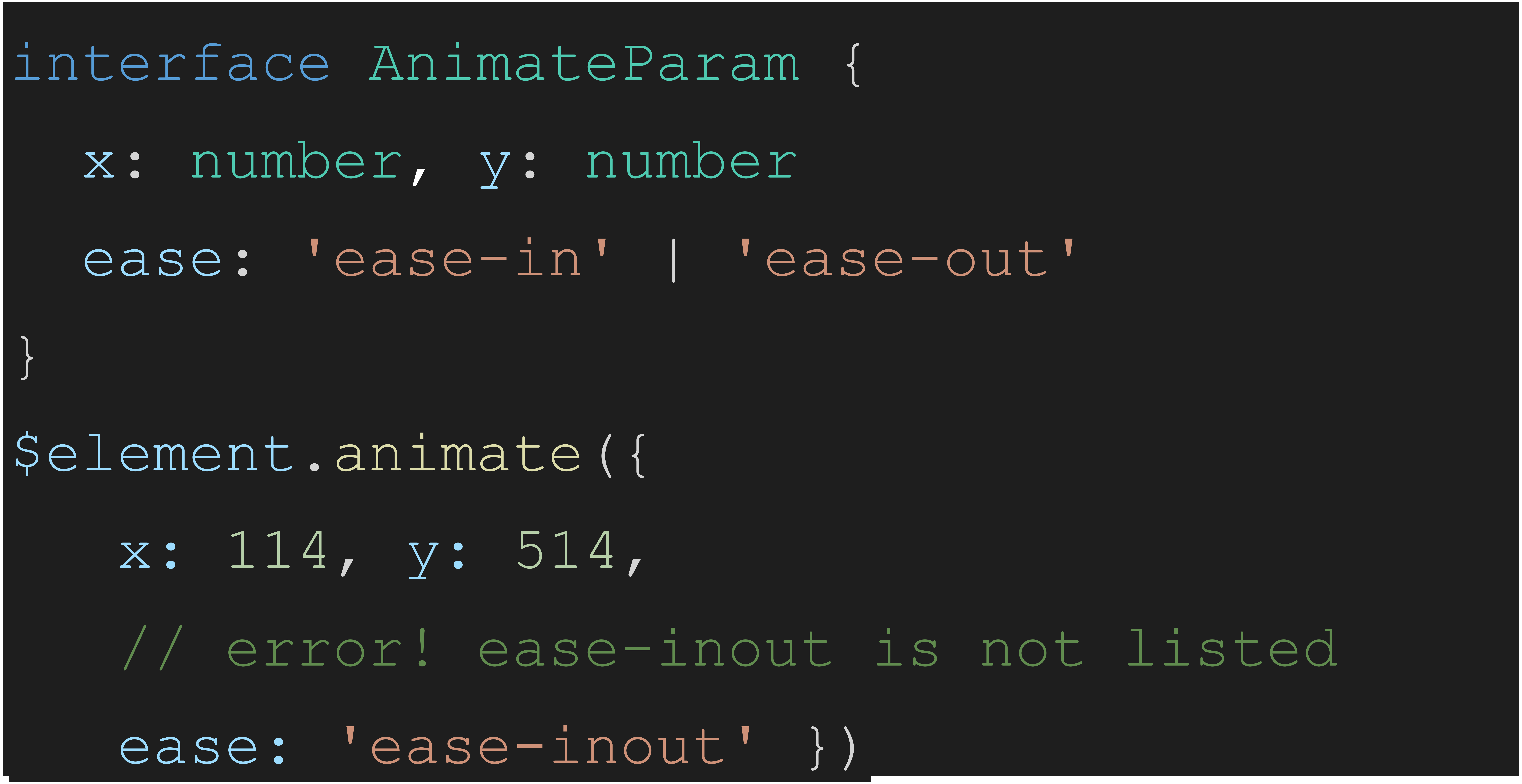
- 字面量类型
- JS特有类型系统的第一步

```
$element.animate({
  x: 114,
  y: 514,
  // ease-in, ease-out
  ease: 'ease-in'
})
```

```
interface AnimateParam {
  x: number, y: number
  ease: string
}
```



```
interface AnimateParam {
  x: number, y: number
  ease: string
}
$element.animate({
   x: 114, y: 514,
   // oops, typo!
   ease: 'ease-inout' })
```



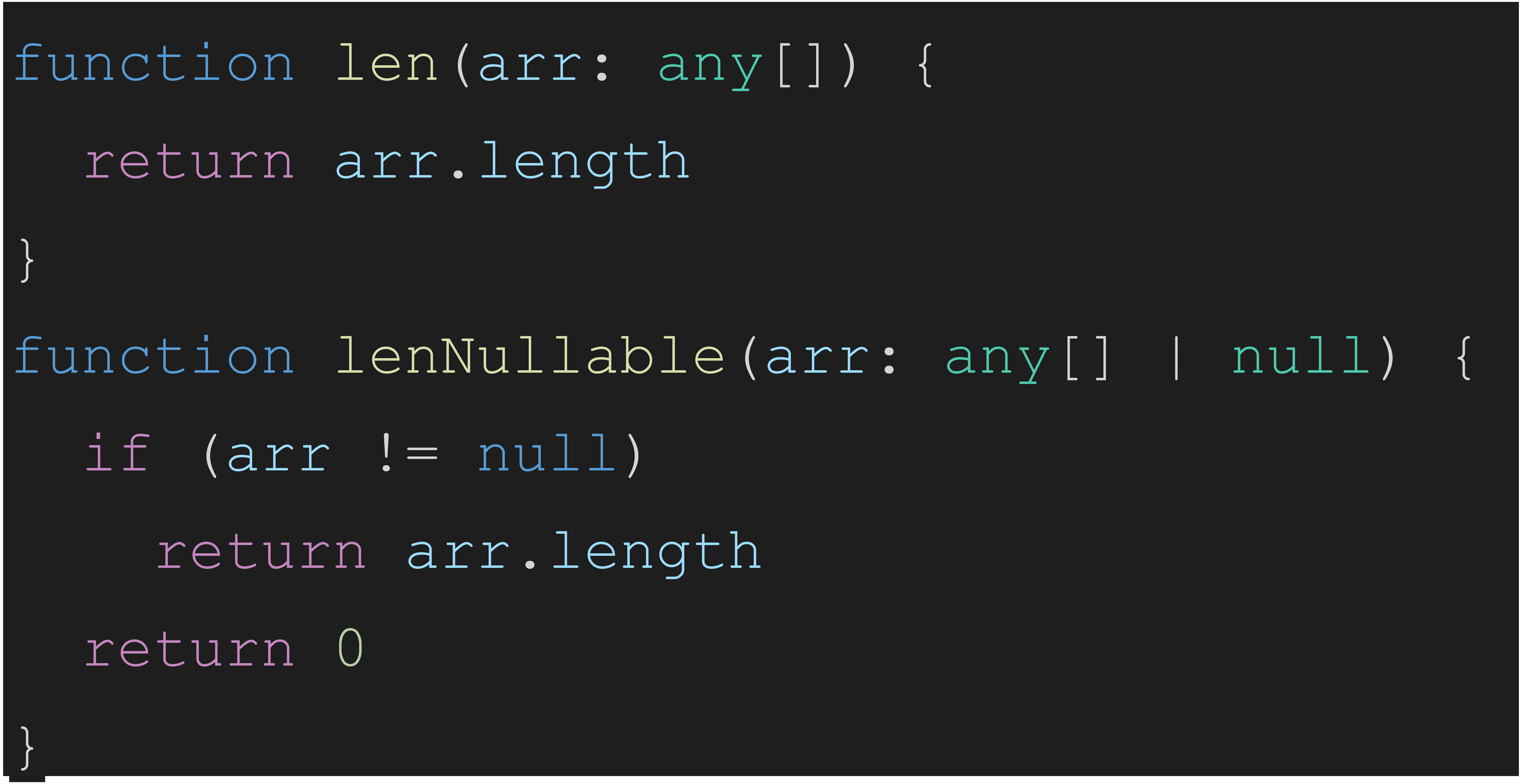
```
interface AnimateParam {
  x: number, y: number
  ease: 'ease-in' | 'ease-out'
}
$element.animate({
   x: 114, y: 514,
   // error! ease-inout is not listed
   ease: 'ease-inout' })
```

TS 2.0

- 可空类型: nullable type
- 价值百万的类型

```
function len(arr) {
  return arr.length
}
function lenNullable(arr) {
  if (arr != null)
    return arr.length
  return 0
}
```

```
len([1, 2, 3]) // ok
len(null) // not ok, but no error
lenNullable([1, 2, 3]) // ok
lenNullable(null) // ok
```

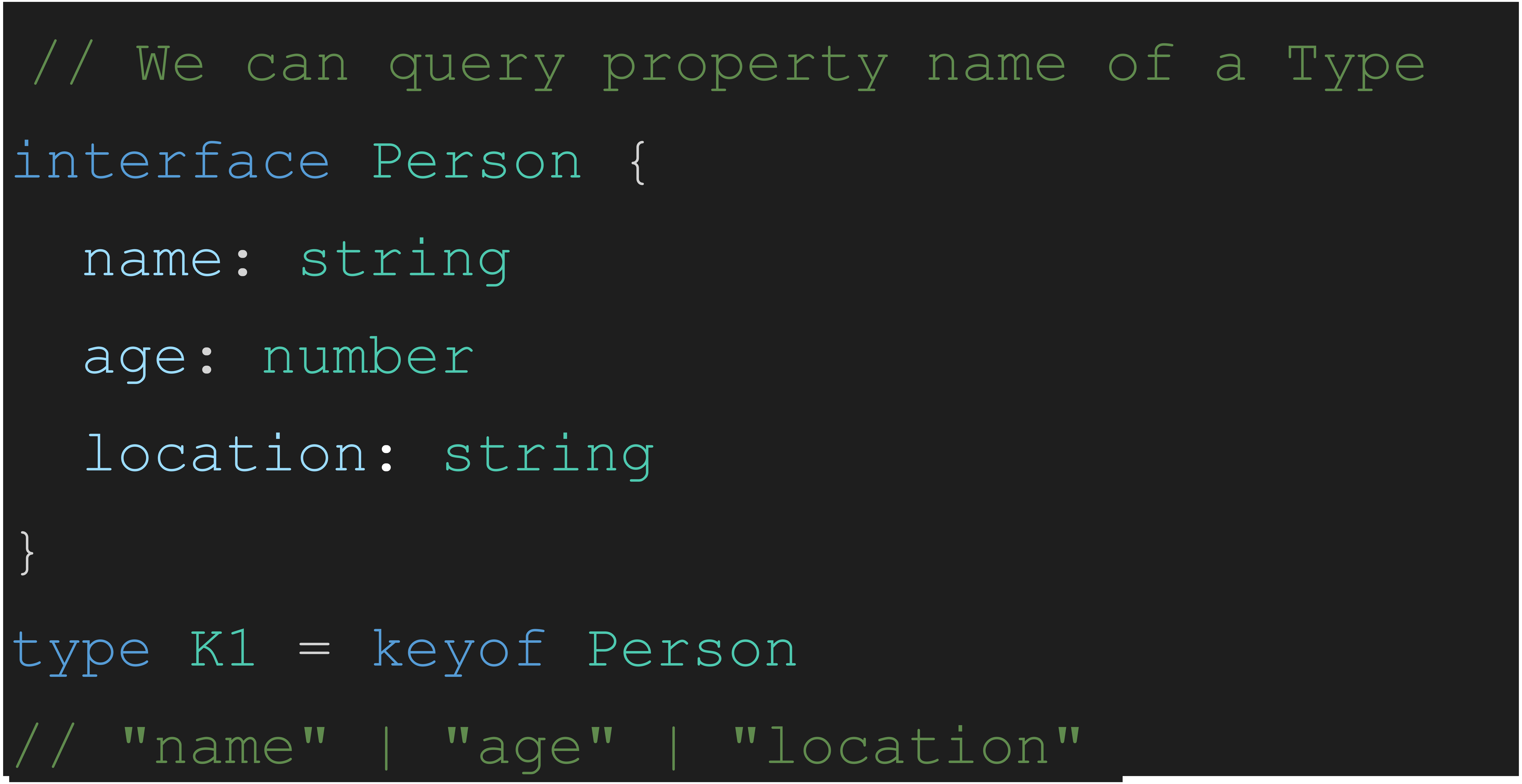


```
function len(arr: any[]) {
  return arr.length
}
function lenNullable(arr: any[] | null) {
  if (arr != null)
    return arr.length
  return 0
}
```

```
len([1, 2, 3]) // ok
len(null) // compile error
lenNullable([1, 2, 3]) // ok
lenNullable(null) // ok
```

TS 2.1+

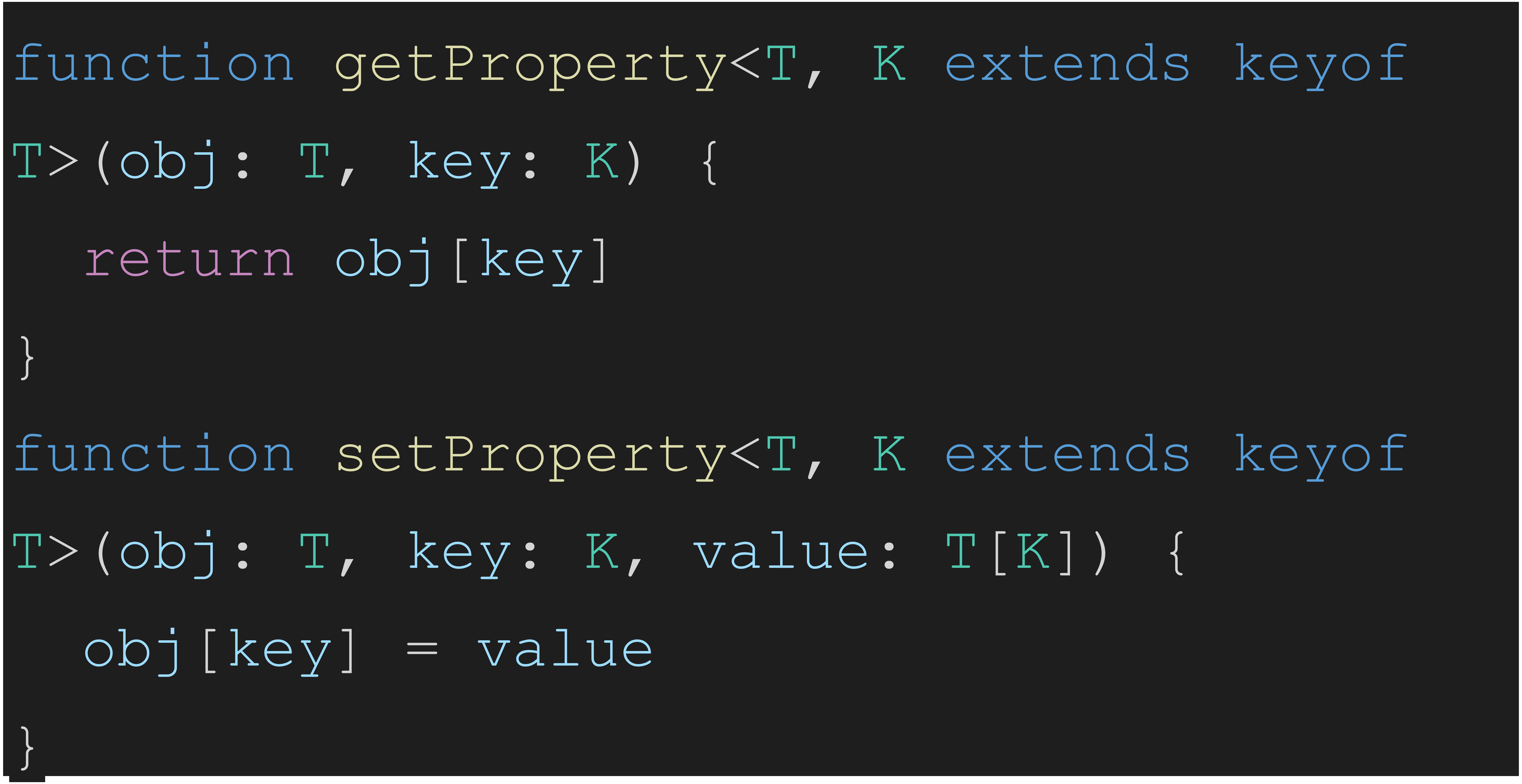
- Lookup Type
- JS特有类型系统的集大成作



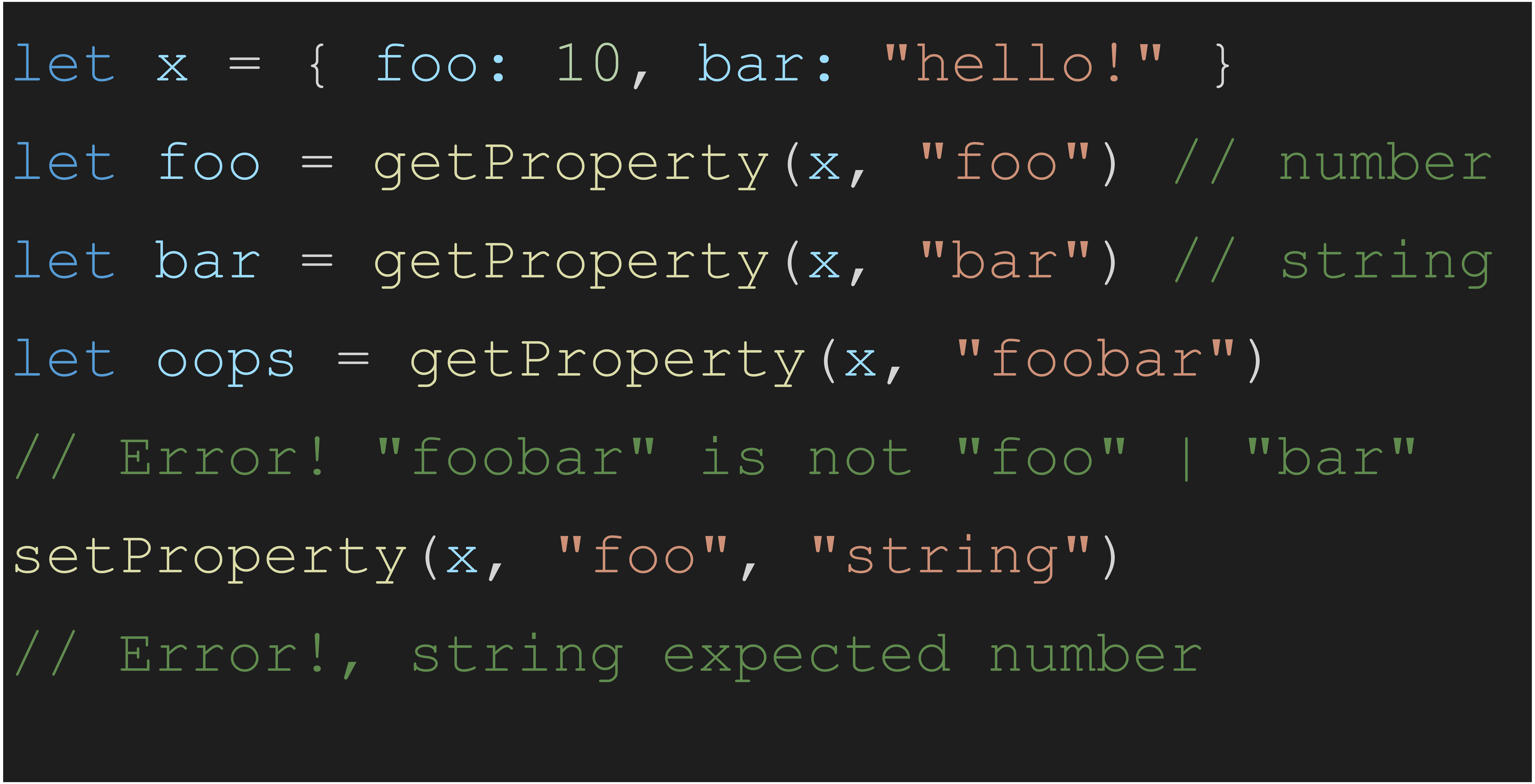
```
 // We can query property name of a Type
interface Person {
  name: string
  age: number
  location: string
}
type K1 = keyof Person
// "name" | "age" | "location"
```

```
type Name = Person['name']
// string
type Age = Person['age']
// number
```

```
type Pluck<T, K extends keyof T> = T[K]
type Location = Pluck<Person, 'location'>
// string
```

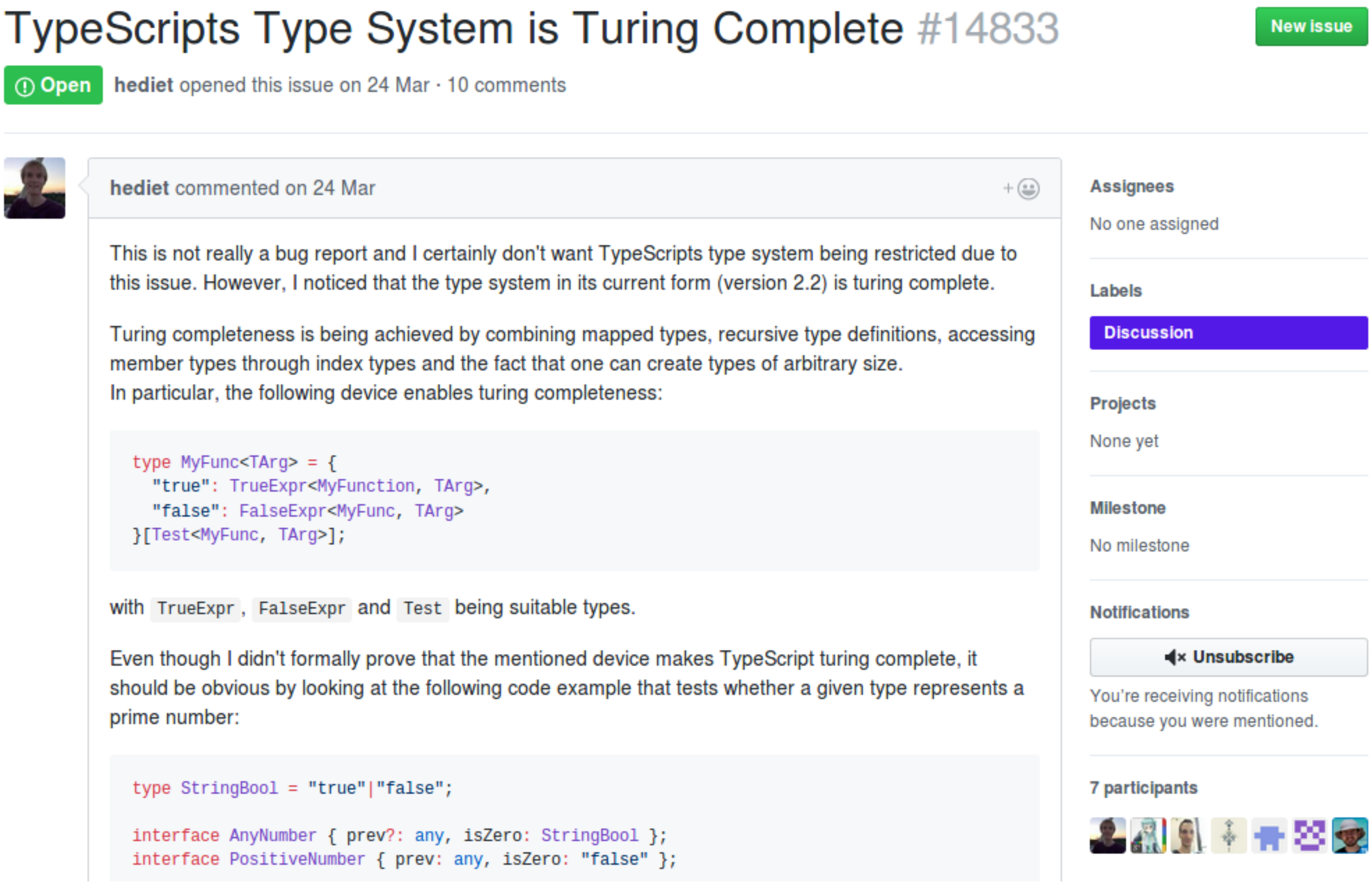


```
function getProperty<T, K extends keyof
T>(obj: T, key: K) {
  return obj[key]
}
function setProperty<T, K extends keyof
T>(obj: T, key: K, value: T[K]) {
  obj[key] = value
}
```



```
let x = { foo: 10, bar: "hello!" }
let foo = getProperty(x, "foo") // number
let bar = getProperty(x, "bar") // string
let oops = getProperty(x, "foobar")
// Error! "foobar" is not "foo" | "bar"
setProperty(x, "foo", "string")
// Error!, string expected number
```

TS惊人的表现力！



TypeScripts Type System is Turing Complete #14833

hediet opened this issue on 24 Mar · 10 comments

hediet commented on 24 Mar

This is not really a bug report and I certainly don't want TypeScripts type system being restricted due to this issue. However, I noticed that the type system in its current form (version 2.2) is turing complete.

Turing completeness is being achieved by combining mapped types, recursive type definitions, accessing member types through index types and the fact that one can create types of arbitrary size.
In particular, the following device enables turing completeness:

```
type MyFunc<TArg> = {
  "true": TrueExpr<MyFunction, TArg>,
  "false": FalseExpr<MyFunc, TArg>
}[Test<MyFunc, TArg>];
```

with `TrueExpr`, `FalseExpr` and `Test` being suitable types.

Even though I didn't formally prove that the mentioned device makes TypeScript turing complete, it should be obvious by looking at the following code example that tests whether a given type represents a prime number:

```
type StringBool = "true"|"false";

interface AnyNumber { prev?: any, isZero: StringBool };
interface PositiveNumber { prev: any, isZero: "false" };
```

Assignees

No one assigned

Labels

Discussion

Projects

None yet

Milestone

No milestone

Notifications

You're receiving notifications because you were mentioned.

7 participants

TS 的未来

- 更健全的类型系统
- 更地道的JS用法

关注QCon微信公众号
获得更多干货!

Thanks!

INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

主办方：Geekbang 极客邦科技 InfoQ