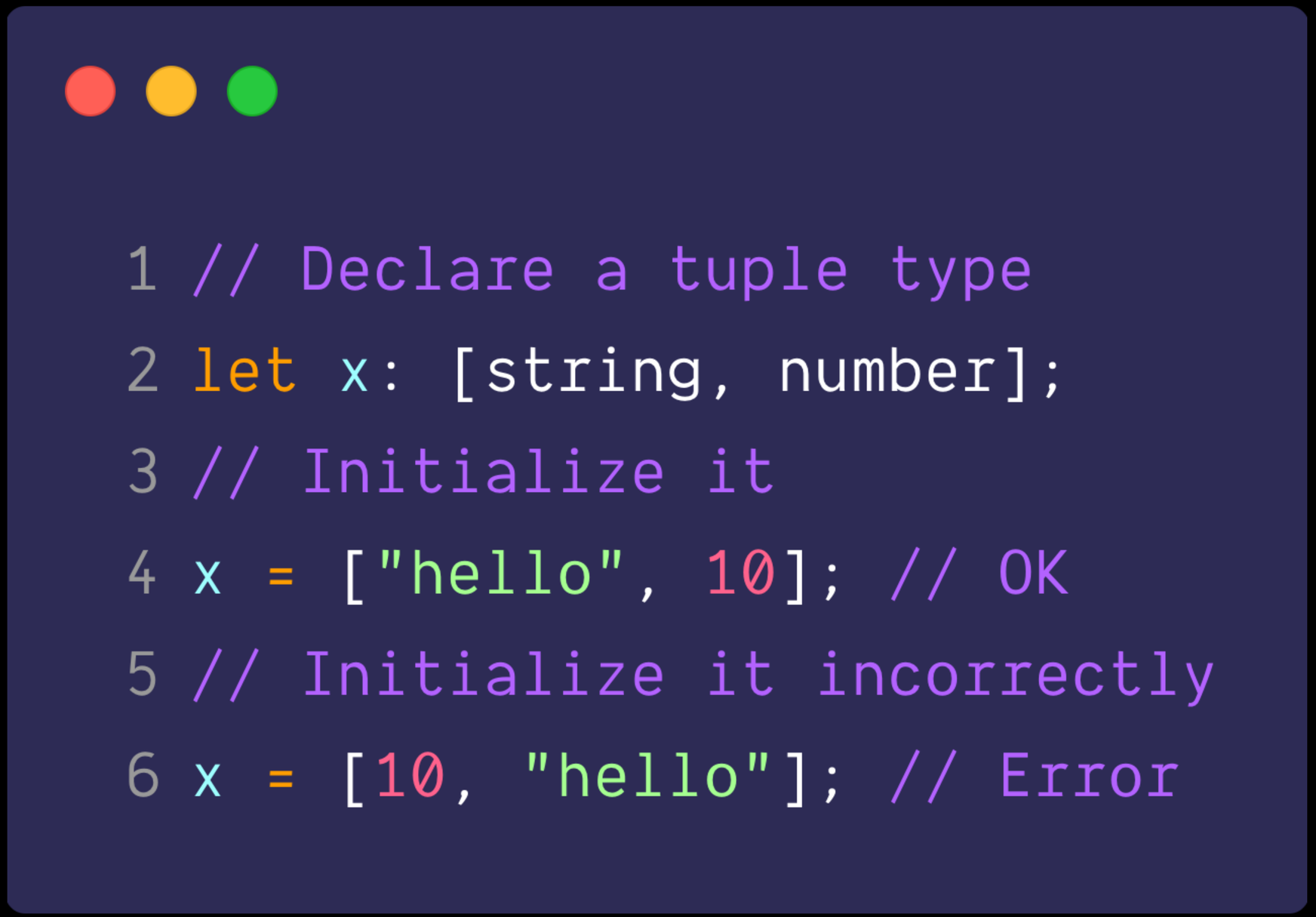
TypeScript 给我们带来了什么？

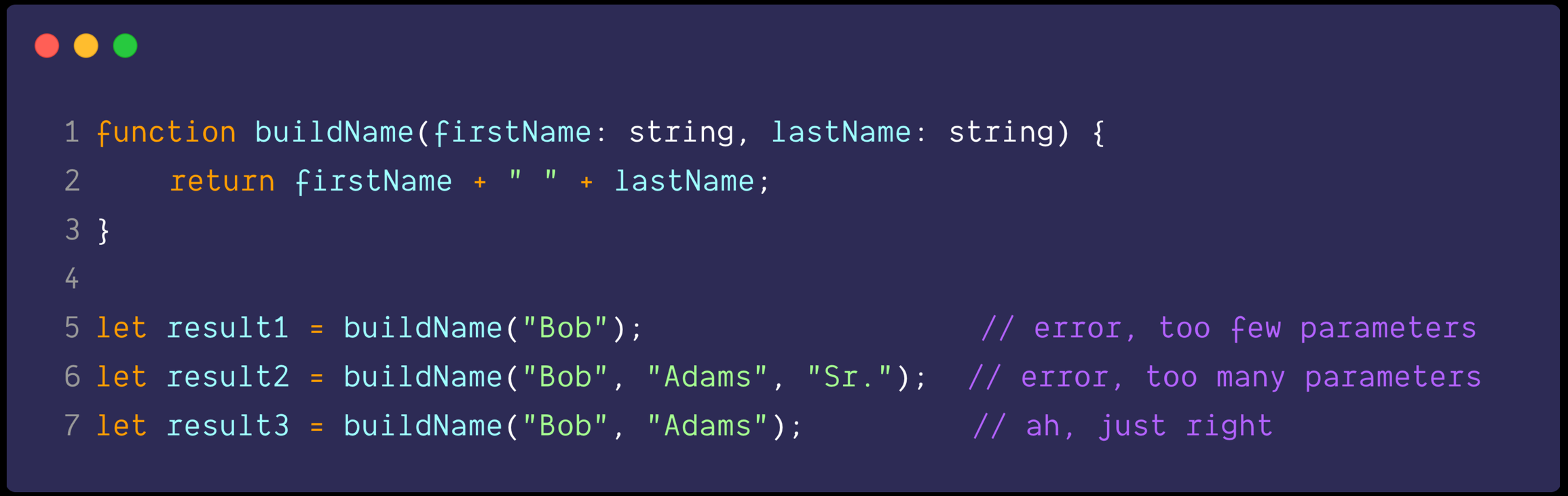
https://github.com/zongzi531

簡単回顧

- TypeScript 发行于 2012 年 10 月 1 日（ 7 年前）—— 维基百科
- TypeScript 是 JavaScript 的类型化超集 —— 官网
- “ AnyScript ” —— 口口相传

类型系统


// Declare a tuple type
let x: [string, number];
// Initialize it
x = ["hello", 10]; // OK
// Initialize it incorrectly
x = [10, "hello"]; // Error



```
function buildName(firstName: string, lastName: string) {
    return firstName + " " + lastName;
}

let result1 = buildName("Bob");                  // error, too few parameters
let result2 = buildName("Bob", "Adams", "Sr.");  // error, too many parameters
let result3 = buildName("Bob", "Adams");         // ah, just right
```

TypeScript 增加了代码的可读性和可维护性

- 类型系统实际上是最好的文档，大部分的函数看看类型的定义就可以知道如何使用了
- 可以在编译阶段就发现大部分错误，这总比在运行时候出错好
- 增强了编辑器和 IDE 的功能，包括代码补全、接口提示、跳转到定义、重构等

任何事物都是有两面性

- 有一定的学习成本，需要理解接口（Interfaces）、泛型（Generics）、类（Classes）、枚举类型（Enums）等
- 短期可能会增加一些开发成本，毕竟要多写一些类型的定义，不过对于一个需要长期维护的项目，TypeScript 能够减少其维护成本
- 集成到构建流程需要一些工作量

有了Babel的话还在使用TypeScript的优势在哪?

太狼
Frontend at day, Rustacean at night.

142 人赞同了该回答

随便说一点
公司现在的前端项目大概有十几万行代码，各种从后端拿到的数据类型有上百种
以前后端接口一改，要改字段，瞬间懵逼。
全局搜索，一个个改，各种牵扯到的东西改下来再测试一顿估计小半天没了。
用了 TypeScript 之后，把数据对应的 interface 改掉，然后重新编译一次，把编译失败的地方全部改掉就好了。

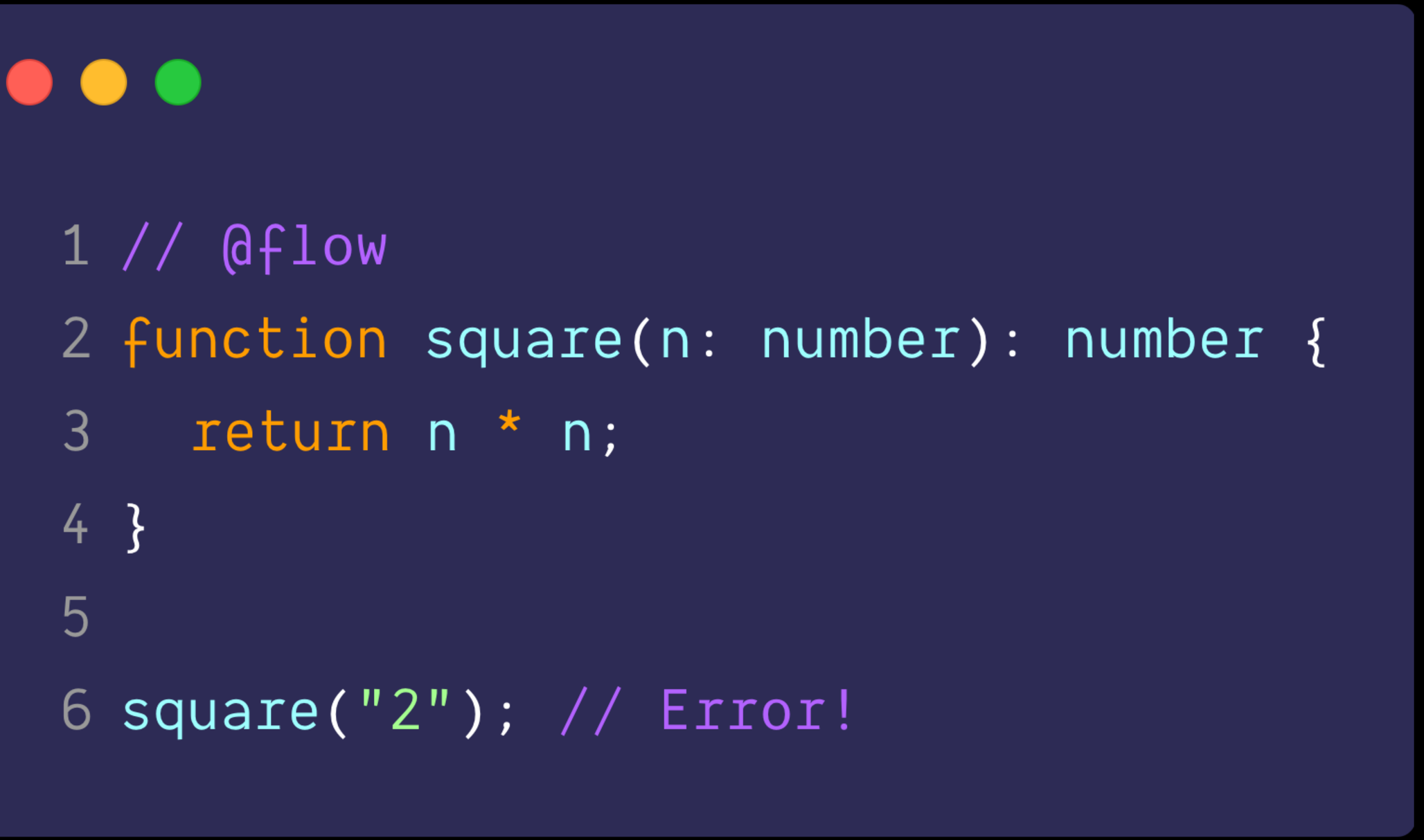
而且在优秀的 TypeScript 架构中，业务开发基本不需要写类型，所有外部输入的类型都可以自动拿到，只需要把一些 local variable 和 output 的类型定义一下就好了，基本跟手写 ES 6 没有区别。
写代码的过程中各种错误在越早期修改的成本就越低。试想没有静态检查跑一遍代码进某个奇怪的 case 才能复现的错误在写代码时期就直接给你的个错误提示，将是多么省时省力省钱。

发布于 2016-08-07

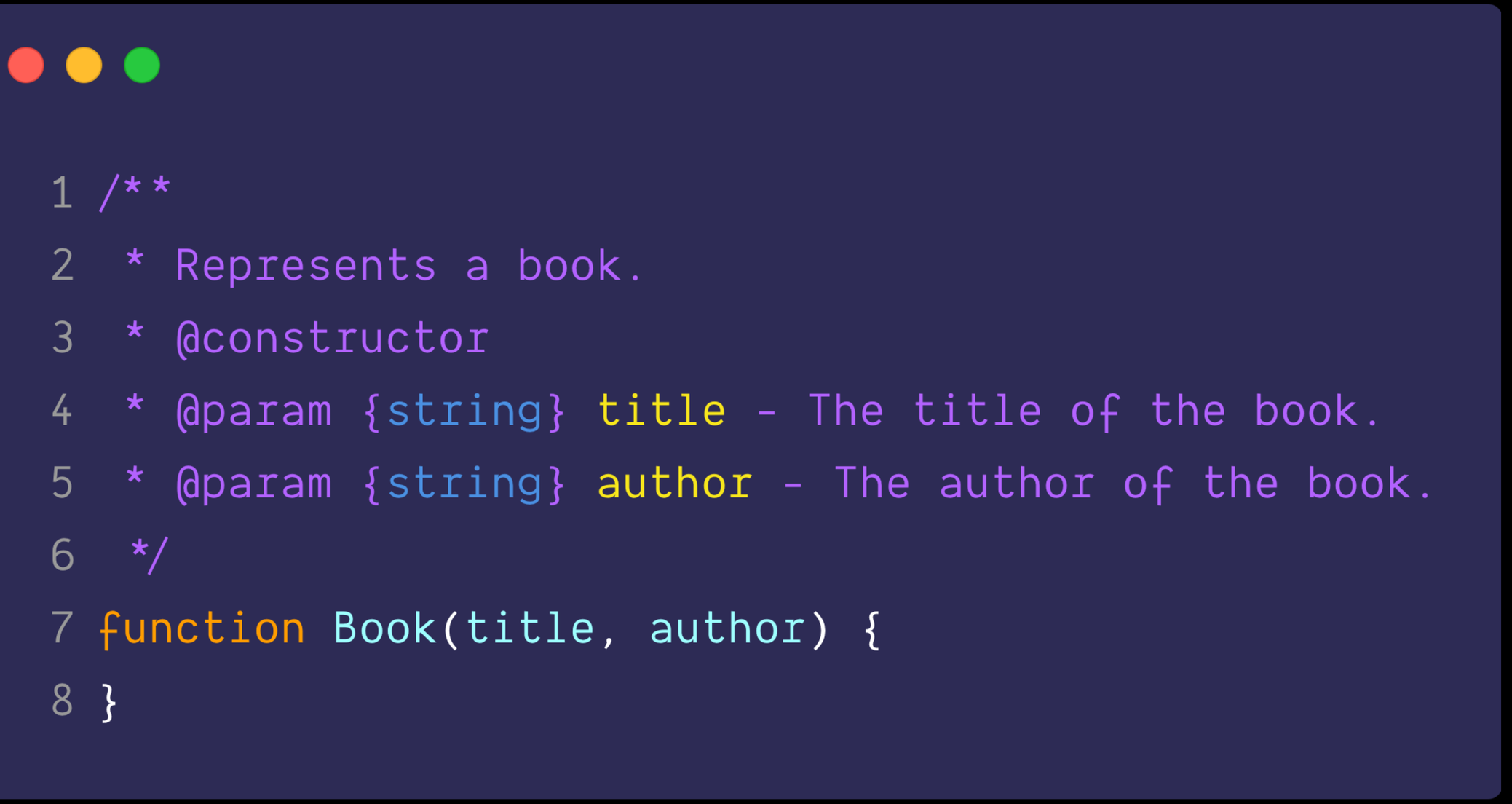
▲ 赞同 142 ▼ 20 条评论 分享 收藏 喜欢 ···

有替代品吗?

“有”


// @flow
function square(n: number): number {
  return n * n;
}

square("2"); // Error!


/**
* Represents a book.
* @constructor
* @param {string} title - The title of the book.
* @param {string} author - The author of the book.
*/
function Book(title, author) {
}

本质

所以，TypeScript 给我们带来了什么？

“类型系统是把双刃剑，以及选型的重要性”

谢谢！