TypeScript 语言简介

概述

TypeScript（简称 TS）是微软公司开发的一种基于 JavaScript （简称 JS）语言的编程语言。

它的目的并不是创造一种全新语言，而是增强 JavaScript 的功能，使其更适合多人合作的企业级项目。

TypeScript 可以看成是 JavaScript 的超集（superset），即它继承了后者的全部语法，所有 JavaScript 脚本都可以当作 TypeScript 脚本（但是可能会报错），此外它再增加了一些自己的语法。

TypeScript 对 JavaScript 添加的最主要部分，就是一个独立的类型系统。

类型的概念

类型（type）指的是一组具有相同特征的值。如果两个值具有某种共同的特征，就可以说，它们属于同一种类型。

举例来说， `123` 和 `456` 这两个值，共同特征是都能进行数值运算，所以都属于“数值”（number）这个类型。

一旦确定某个值的类型，就意味着，这个值具有该类型的所有特征，可以进行该类型的所有运算。凡是适用该类型的地方，都可以使用这个值；凡是不适用该类型的地方，使用这个值都会报错。

可以这样理解，**类型是人为添加的一种编程约束和用法提示。** 主要目的是在软件开发过程中，为编译器和开发工具提供更多的验证和帮助，帮助提高代码质量，减少错误。

下面是一段简单的 TypeScript 代码，演示一下类型系统的作用。

typescript

```typescript
function addOne(n: number) {
  return n + 1;
}
```

上面示例中，函数 `addOne()` 有一个参数 `n` ，类型为数值（number），表示这个位置只能使用数值，传入其他类型的值就会报错。

typescript

```typescript
addOne("hello"); // 报错
```

上面示例中，函数 `addOne()` 传入了一个字符串 `hello` ，TypeScript 发现类型不对，就报错了，指出这个位置只能传入数值，不能传入字符串。

JavaScript 语言就没有这个功能，不会检查类型对不对。开发阶段很可能发现不了这个问题，代码也许就会原样发布，导致用户在使用时遇到错误。

作为比较，TypeScript 是在开发阶段报错，这样有利于提早发现错误，避免使用时报错。另一方面，函数定义里面加入类型，具有提示作用，可以告诉开发者这个函数怎么用。

动态类型与静态类型

前面说了，TypeScript 的主要功能是为 JavaScript 添加类型系统。大家可能知道，JavaScript 语言本身就有一套自己的类型系统，比如数值 `123` 和字符串 `Hello` 。

但是，JavaScript 的类型系统非常弱，而且没有使用限制，运算符可以接受各种类型的值。在语法上，JavaScript 属于动态类型语言。

请看下面的 JavaScript 代码。

javascript

```javascript
// 例一
let x = 1;
x = "hello";

// 例二
let y = { foo: 1 };
delete y.foo;
y.bar = 2;
```

上面的例一，变量 `x` 声明时，值的类型是数值，但是后面可以改成字符串。所以，无法提前知道变量的类型是什么，也就是说，变量的类型是动态的。

上面的例二，变量 `y` 是一个对象，有一个属性 `foo` ，但是这个属性是可以删掉的，并且还可以新增其他属性。所以，对象有什么属性，这个属性还在不在，也是动态的，没法提前知道。

正是因为存在这些动态变化，所以 JavaScript 的类型系统是动态的，不具有很强的约束性。这对于提前发现代码错误，非常不利。

TypeScript 引入了一个更强大、更严格的类型系统，属于静态类型语言。

上面的代码在 TypeScript 里面都会报错。

javascript

```
// 例一
let x = 1;
x = "hello"; // 报错

// 例二
let y = { foo: 1 };
delete y.foo; // 报错
y.bar = 2; // 报错
```

上面示例中，例一的报错是因为变量赋值时，TypeScript 已经推断确定了类型，后面就不允许再赋值为其他类型的值，即变量的类型是静态的。例二的报错是因为对象的属性也是静态的，不允许随意增删。

TypeScript 的作用，就是为 JavaScript 引入这种静态类型特征。

静态类型的优点

静态类型有很多好处，这也是 TypeScript 想要达到的目的。

（1）有利于代码的静态分析。

有了静态类型，不必运行代码，就可以确定变量的类型，从而推断代码有没有错误。这就叫做代码的静态分析。

这对于大型项目非常重要，单单在开发阶段运行静态检查，就可以发现很多问题，避免交付有问题的代码，大大降低了线上风险。

（2）有利于发现错误。

由于每个值、每个变量、每个运算符都有严格的类型约束，TypeScript 就能轻松发现拼写错误、语义错误和方法调用错误，节省程序员的时间。

typescript

```
let obj = { message: "" };
console.log(obj.messege); // 报错
```

上面示例中，不小心把 `message` 拼错了，写成 `messege` 。TypeScript 就会报错，指出没有定义过这个属性。JavaScript 遇到这种情况是不报错的。

typescript

```
const a = 0;
const b = true;
const result = a + b; // 报错
```

上面示例是合法的 JavaScript 代码，但是没有意义，不应该将数值 `a` 与布尔值 `b` 相加。TypeScript 就会直接报错，提示运算符 `+` 不能用于数值和布尔值的相加。

typescript

```
function hello() {
  return "hello world";
}

hello().find("hello"); // 报错
```

上面示例中， `hello()` 返回的是一个字符串，TypeScript 发现字符串没有 `find()` 方法，所以报错了。如果是 JavaScript，只有到运行阶段才会报错。

（3）更好的 IDE 支持，做到语法提示和自动补全。

IDE（集成开发环境，比如 VSCode）一般都会利用类型信息，提供语法提示功能（编辑器自动提示函数用法、参数等）和自动补全功能（只键入一部分的变量名或函数名，编辑器补全后面的部分）。

（4）提供了代码文档。

类型信息可以部分替代代码文档，解释应该如何使用这些代码，熟练的开发者往往只看类型，就能大致推断代码的作用。借助类型信息，很多工具能够直接生成文档。

（5）有助于代码重构。

修改他人的 JavaScript 代码，往往非常痛苦，项目越大越痛苦，因为不确定修改后是否会影响到其他部分的代码。

类型信息大大减轻了重构的成本。一般来说，只要函数或对象的参数和返回值保持类型不变，就能基本确定，重构后的代码也能正常运行。如果还有配套的单元测试，就完全可以放心重构。越是大型的、多人合作的项目，类型信息能够提供的帮助越大。

综上所述，TypeScript 有助于提高代码质量，保证代码安全，更适合用在大型的企业级项目。这就是为什么大量 JavaScript 项目转成 TypeScript 的原因。

静态类型的缺点

静态类型也存在一些缺点。

（1）丧失了动态类型的代码灵活性。

动态类型有非常高的灵活性，给予程序员很大的自由，静态类型将这些灵活性都剥夺了。

（2）增加了编程工作量。

有了类型之后，程序员不仅需要编写功能，还需要编写类型声明，确保类型正确。这增加了不少工作量，有时会显著拖长项目的开发时间。

（3）更高的学习成本。

类型系统通常比较复杂，要学习的东西更多，要求开发者付出更高的学习成本。

（4）引入了独立的编译步骤。

原生的 JavaScript 代码，可以直接在 JavaScript 引擎运行。添加类型系统以后，就多出了一个单独的编译步骤，检查类型是否正确，并将 TypeScript 代码转成 JavaScript 代码，这样才能运行。

（5）兼容性问题。

TypeScript 依赖 JavaScript 生态，需要用到很多外部模块。但是，过去大部分 JavaScript 项目都没有做 TypeScript 适配，虽然可以自己动手做适配，不过使用时难免还是会有一些兼容性问题。

总的来说，这些缺点使得 TypeScript 不一定适合那些小型的、短期的个人项目。

TypeScript 的历史

下面简要介绍 TypeScript 的发展历史。

2012 年，微软公司宣布推出 TypeScript 语言，设计者是著名的编程语言设计大师 Anders Hejlsberg，他也是 C# 和 .Net 的设计师。

微软推出这门语言的主要目的，是让 JavaScript 程序员可以参与 Windows 8 应用程序的开发。

当时，Windows 8 即将发布，它的应用程序开发除了使用 C# 和 Visual Basic，还可以使用 HTML + JavaScript。微软希望，TypeScript 既能让 JavaScript 程序员快速上手，也能让 .Net 程序员感到熟悉。

这就是说，TypeScript 的最初动机是减少 .Net 程序员的转移和学习成本。所以，它的很多语法概念跟 .Net 很类似。

另外，TypeScript 是一个开源项目，接受社区的参与，核心的编译器采用 Apache 2.0 许可证。微软希望通过这种做法，迅速提高这门语言在社区的接受度。

2013 年，微软的 Visual Studio 2013 开始内置支持 TypeScript 语言。

2014 年，TypeScript 1.0 版本发布。同年，代码仓库搬到了 GitHub。

2016 年，TypeScript 2.0 版本发布，引入了很多重大的语法功能。

2018 年，TypeScript 3.0 版本发布。

2020 年，TypeScript 4.0 版本发布。

2023 年，TypeScript 5.0 版本发布。

如何学习

学习 TypeScript，必须先了解 JavaScript 的语法。因为真正的实际功能都是 JavaScript 引擎完成的，TypeScript 只是添加了一个类型系统。

本书假定读者已经了解 JavaScript 语言，就不再介绍它的语法了，只介绍 TypeScript 引入的新语法，主要是类型系统。

如果你对 JavaScript 还不熟悉，建议先阅读《JavaScript 教程》和《ES6 教程》，再来阅读本书。

最后更新: 2023/8/13 15:25