



TypeScript(二)使用Webpack搭建环境

原创 coderwhy coderwhy 2019-11-01 18:02

TypeScript环境搭建

 前言：接下来暂停Flutter更新，开始更新TypeScript教程**更新地点：**首发于公众号，第二天更新于掘金、简书等地方；**更多交流：**可以添加我的微信 372623326，关注我的微博：coderwhy希望大家可以 **帮忙转发，点击在看**，给我更多的创作动力。 //

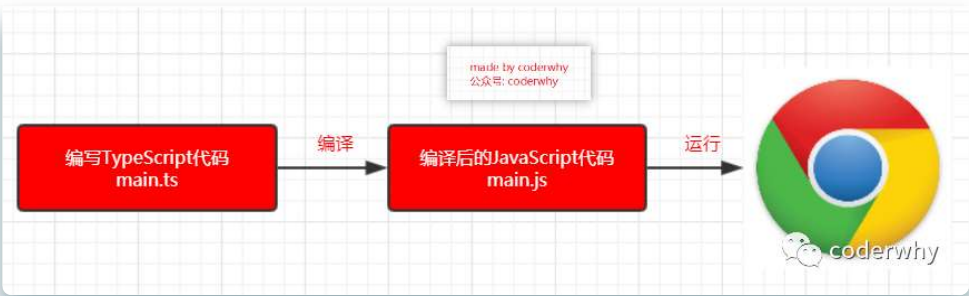
 学习任何的开发，我们都需要对应的环境：包括TypeScript的编译环境和开发工具。这个章节里面，我们就来完成它们的搭建，后续就可以愉快的来写TypeScript代码了。 //

一. 环境搭建

1.1. TypeScript环境安装

已经配置好的环境，大家可以直接下载：<https://github.com/coderwhy/HYLearnTS.git>

在上一个章节中我们说过，TypeScript最终会被编译成JavaScript代码：



TypeScript运行流程

那么我们必然需要对应的编译环境：

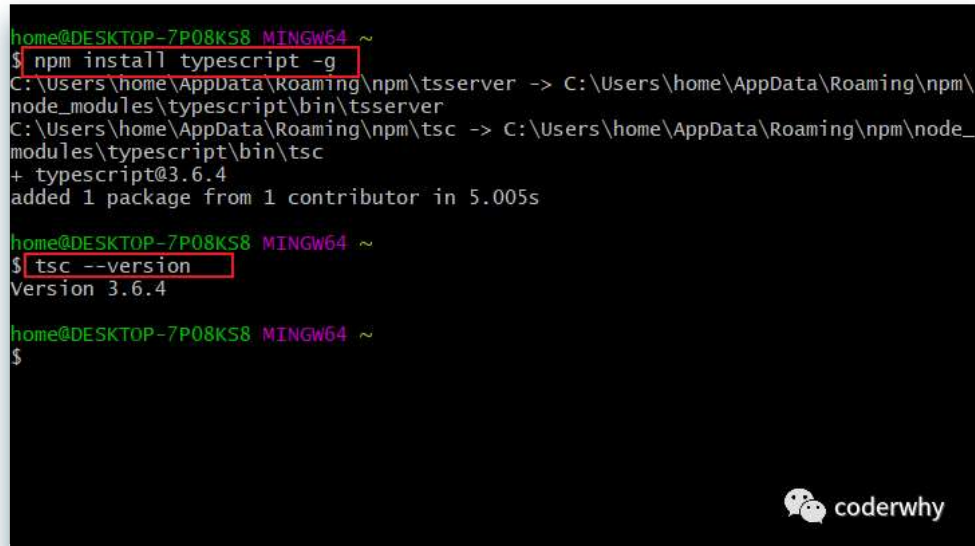
- 首先，TypeScript的环境安装依赖Node，所以需要先保证电脑上有Node和NPM环境；
- 其次，可以通过NPM来安装TypeScript，之后就可以通过 `tsc` 来编译TypeScript的代码；

首先来进行全局安装：

```
# 安装命令
npm install typescript -g

# 查看版本
tsc --version
```

注意：这里我使用了git bash的终端，你可以直接使用windows的命令行工具



```
home@DESKTOP-7P08KS8 MINGW64 ~
$ npm install typescript -g
C:\Users\home\AppData\Roaming\npm\tsserver -> C:\Users\home\AppData\Roaming\npm\node_modules\typescript\bin\tsserver
C:\Users\home\AppData\Roaming\npm\tsc -> C:\Users\home\AppData\Roaming\npm\node_modules\typescript\bin\tsc
+ typescript@3.6.4
added 1 package from 1 contributor in 5.005s

home@DESKTOP-7P08KS8 MINGW64 ~
$ tsc --version
Version 3.6.4

home@DESKTOP-7P08KS8 MINGW64 ~
$
```

安装TypeScript

1.2. VSCode环境搭建

学习或者使用TypeScript有很多编辑器可以供我们选择，目前前端开发比较常用的是两个：

- WebStorm: JetBrains 公司的产品，用法和PHPStorm、PyCharm、IDEA用法基本一致；
- VSCode: Microsoft 公司的产品，目前可以说已经成为最流行的前端工具，并且本身就是TypeScript编写的；

在之前的Flutter文章中我有对比过VSCode和Android Studio的优缺点，其实和VSCode和WebStorm的优缺点对比是相似的，这里不再详细对比。

而以后学习和使用TypeScript，我都会优先选择VSCode，所以这里我们主要介绍VSCode的环境搭建。

1. 下载安装VSCode

- 下载地址: <https://code.visualstudio.com/>
- 下载后直接安装即可

2. 安装对应的插件

- VSCode的另外一个强大的地方就是已经有很多好用的插件了；

- 我个人针对于前端安装过的插件有：open in browser、Vetur、TSLint、Bracket Pair Colorizer等；
- 这里我不再给出截图，因为我还针对Python、Java、Flutter装过很多的插件，会干扰大家；

1.3. tsc简单代码测试

上面步骤完成后，我们就可以在VSCode中编写我们的TypeScript代码，并且通过一些方法来进行测试。

1. 打开VSCode，并且新建两个文件：index.ts

index.ts代码如下：

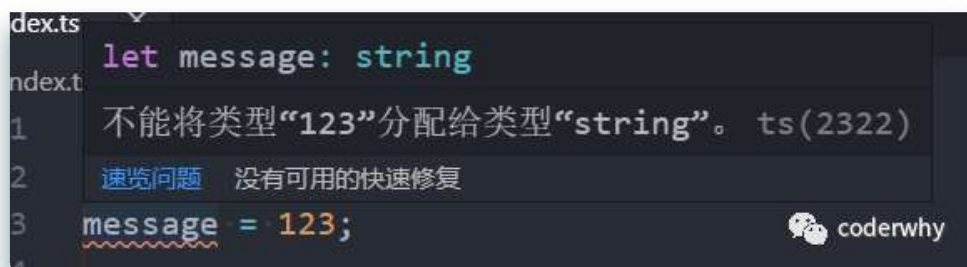
```
// 定义一个变量
let message: string = "abc";
message = 123;

// 定义一个函数
function sum(num1: number, num2: number): number {
    return num1 + num2;
}

sum("abc", "cba");
```

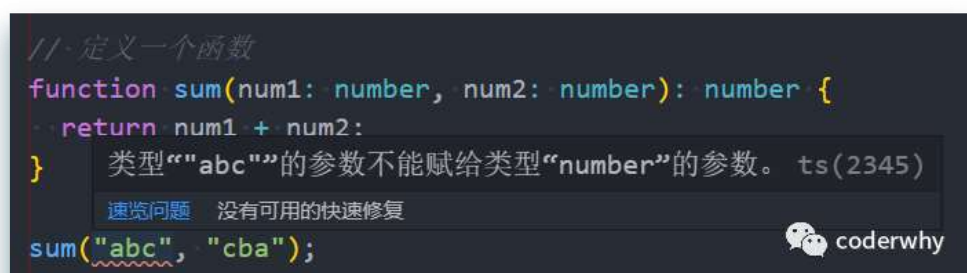
我们会发现有两个地方都会报错：

错误一：不能将类型“123”分配给类型“string”



错误一

错误二：类型“"abc"”的参数不能赋给类型“number”的参数



错误二

上面两个错误都是因为我们的代码已经增加了类型约束，不能随便赋值其他类型给我们的变量。

将代码修改正确如下：

```
// 定义一个变量
let message: string = "abc";
message = "Hello World";

// 定义一个函数
function sum(num1: number, num2: number): number {
    return num1 + num2;
}

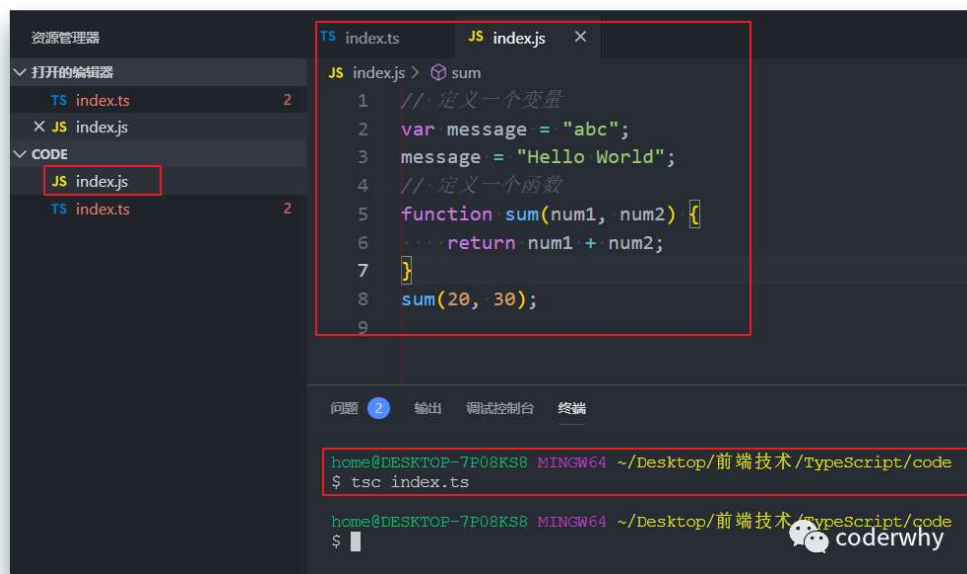
sum(20, 30);
```

2. 将代码编译为JavaScript的代码

因为我们说过，代码最终运行在浏览器上，而浏览器是不识别TypeScript代码的，我们需要对他们进行编译：

打开VSCode的终端，在其中输入如下命令来编译我们的TypeScript：

```
tsc index.ts
```



查看结果

我们会发现，生成了一个index.js文件，并且其中的代码就是普通的JavaScript代码。

3. JavaScript代码的测试

如果我们希望测试这段JavaScript代码就非常简单了，两种方式都可以：

- 方式一：使用node直接执行JavaScript代码；
- 方式二：创建一个html文件，在其中引入index.js文件，并且在浏览器中进行测试；

这里不再给出具体的步骤，大家可以自行去测试

问题：每次都这样测试会不会太麻烦了呢？

如果每次我们写完一个TypeScript代码都需要像上面的步骤一样，一点点去完成测试就会过于麻烦，我们可以怎么做呢？

- 直接配置webpack，让webpack对我们编写的代码进行一个编译，并且自动引入编译后的js文件；
- 而且webpack可以在代码修改后重新帮助我们进行编译，并且自动刷新浏览器，不需要手动操作；

二. 项目环境

 如果实在不会搭建，可以从我的GitHub上直接下载我已经搭建好的模板：记得点个starGitHub地址：
//

2.1. 项目环境的基础配置

为了我们之后的学习和使用方便，我们来配置一个webpack的环境：

- 在环境中我们编写对应的TypeScript代码，让webpack自动帮助我们编译，并且在浏览器中查看结果

注意：这里可能需要大家对npm和webpack有一些简单的了解，不会非常复杂（如果完全不懂，按照我给出的步骤来做即可，后续自己进行一些知识的补充）

1. 创建一个简单的项目目录结构

新建一个新的目录：LearnTypeScript，并且创建如下的目录结构

```
| index.html
├─build
|   webpack.config.js
└─src
    main.ts
```

目录和文件夹结构分析：

- index.html是跑在浏览器上的模块文件
- build文件夹中用于存放webpack的配置信息
- src用于存放我们之后编写的所有TypeScript代码

2. 使用npm管理项目的依赖

webpack本身需要有很多的依赖，并且之后我们也需要启动node服务来快速浏览index.html模板以及编译后的JavaScript代码。

我们要使用npm来初始化package.json文件：

```
npm init -y
```



初始化package.json

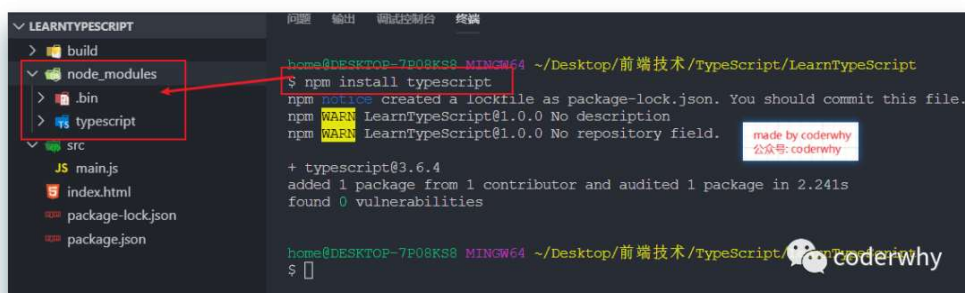
3. 本地依赖TypeScript

为什么需要本地依赖TypeScript：

- 因为我们之后是通过webpack进行编译我们的TypeScript代码的，并不是通过tsc来完成的。（tsc使用的是全局安装的TypeScript依赖）
- 那么webpack会在本地去查找TypeScript的依赖，所以我们是需要本地依赖TypeScript的；

安装本地TypeScript依赖

```
npm install typescript
```



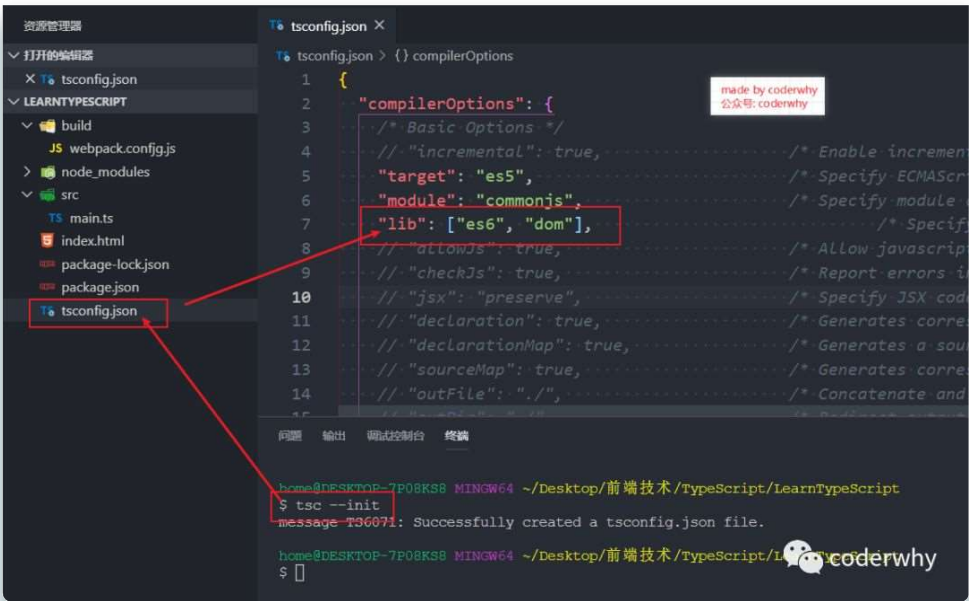
本地安装TypeScript

4. 初始化tsconfig.json文件

在进行TypeScript开发时，我们会针对TypeScript进行相关的配置，而这些配置信息是存放在一个tsconfig.json文件中的

我们并不需要手动去创建它，可以通过命令行直接来生成这样的一个文件：

tsc --init



初始化tsconfig.json

5. 配置tslint来约束代码

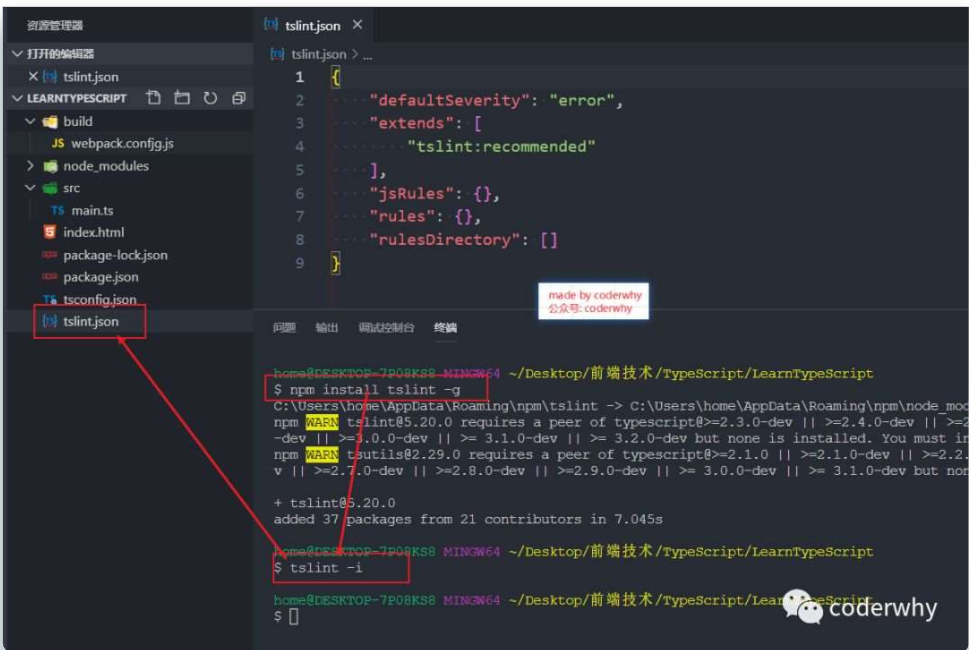
为了让大家按照严格的TypeScript风格学习代码，这里我希望大家可以加入tslint

全局安装tslint：

npm install tslint -g

在项目中初始化tslint的配置文件：tslint.json

tslint -i



2.2. 项目环境的Webpack

下面我们开始配置webpack相关的内容

1. 安装webpack相关的依赖

使用webpack开发和打开，需要依赖webpack、webpack-cli、webpack-dev-server

```
npm install webpack webpack-cli webpack-dev-server -D
```

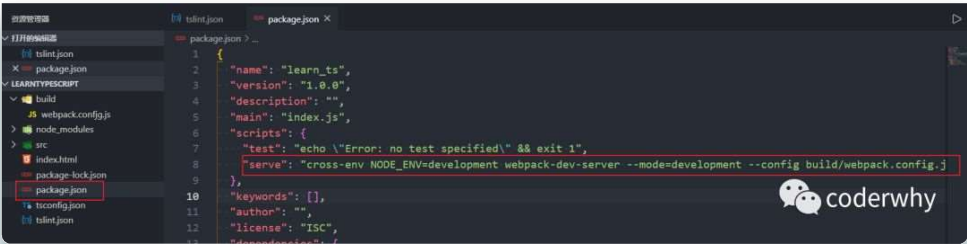


安装webpack依赖

2. 在package.json中添加启动命令

为了方便启动webpack，我们在package.json中添加如下启动命令

```
"scripts": {  
  "test": "echo \"Error: no test specified\" && exit 1",  
  "serve": "cross-env NODE_ENV=development webpack-dev-server --mode=development --config build/webpack.config.js",  
},
```



自定义启动脚本

3. 添加webpack的其他相关依赖

依赖一：cross-env

这里我们用到一个插件 "cross-env" , 这个插件的作用是在可以在 webpack.config.js 中通过 process.env.NODE_ENV 来获取当前是开发还是生产环境, 我们需要这个插件:

```
npm install cross-env -D
```

依赖二: ts-loader

因为我们需要解析.ts文件, 所以需要依赖对应的loader: ts-loader

```
npm install ts-loader -D
```

依赖三: html-webpack-plugin

编译后的代码需要对应的html模块作为它的运行环境, 所以我们需要使用html-webpack-plugin来将它插入到对应的模板中:

```
npm install html-webpack-plugin -D
```

4. 配置webpack.config.js文件

将如下配置到webpack.config.js文件中:

- 这里不再给出详细的说明信息, webpack后面我可能会再开一个专栏来讲解

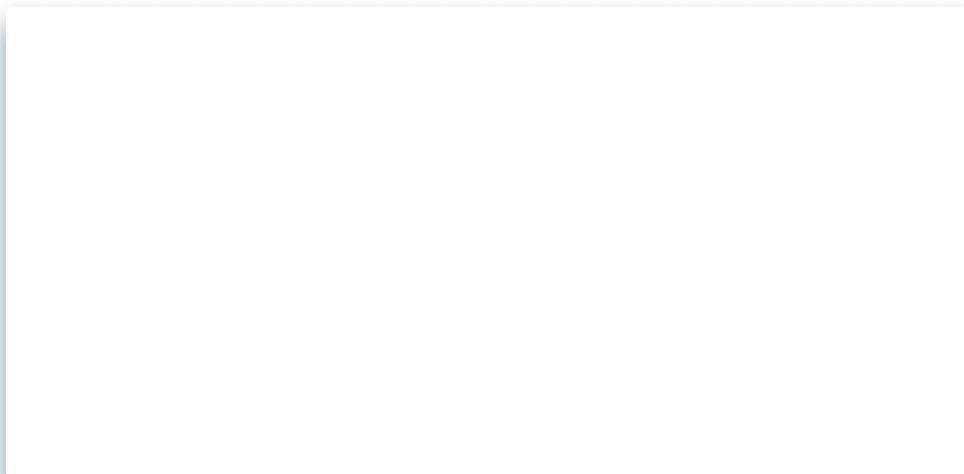
```
const HtmlWebpackPlugin = require("html-webpack-plugin");

module.exports = {
  entry: "./src/main.ts",
  output: {
    filename: "build.js"
  },
  resolve: {
    extensions: [".tsx", ".ts", ".js"]
  },
  module: {
    rules: [
      {
        test: /\.tsx?$/,
        use: "ts-loader",
        exclude: /node_modules/
      }
    ]
  },
  devtool: process.env.NODE_ENV === "production" ? false : "inline-source-map",
  devServer: {
    contentBase: "./dist",
```

```
    stats: "errors-only",
    compress: false,
    host: "localhost",
    port: 8080
  },
  plugins: [
    new HtmlWebpackPlugin({
      template: "./index.html"
    })
  ]
};
```

2.3. 项目环境下代码测试

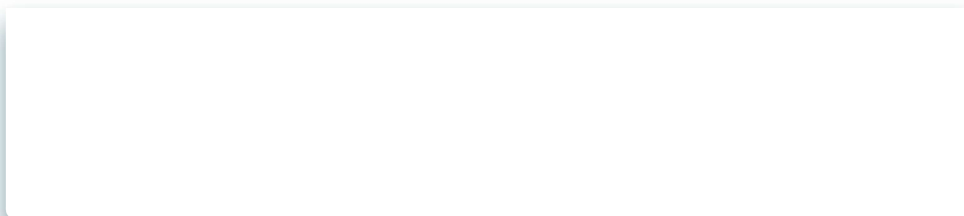
下面我们就可以愉快的在main.ts中编写代码，之后只需要启动服务即可：



测试代码

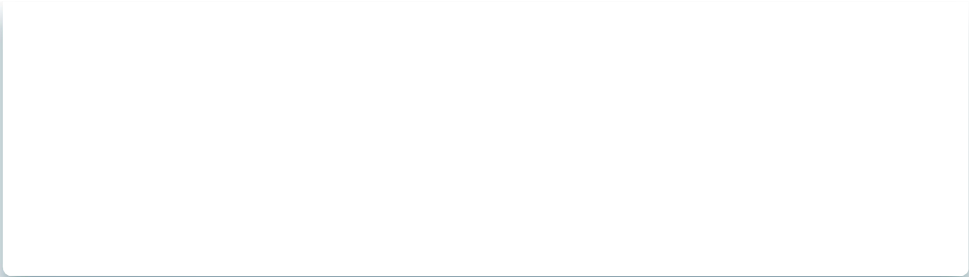
在终端中启动服务：

```
npm run serve
```



程序运行

在浏览器中打开：<http://localhost:8080/>



查看结果

修改代码，直接可以看到修改后的效果：不需要手动进行任何刷新



修改代码



备注：所有内容首发于公众号，之后除了Flutter也会更新其他技术文章，TypeScript、React、Node、数据结构与算法等等，也会更新一些自己的学习心得等，欢迎大家关注 //



公众号

TypeScript教程 4

TypeScript教程 · 目录

上一篇

TypeScript(一)TypeScript培养类型思维

下一篇

TypeScript(三)定义变量和数据类型

