一、手册指南

1、基础类型

1）、元组 Tuple

元组类型允许表示一个已知元素数量和类型的数组，各元素的类型不必相同。比如，你可以定义一对值分别为 string和number类型的元组。

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

2）、枚举 enum

enum类型是对JavaScript标准数据类型的一个补充。 像C#等其它语言一样，使用枚举类型可以为一组数值赋予友好的名字。

```
enum Color {Red = 1, Green, Blue}
let colorName: string = Color[2];

console.log(colorName);  // 显示'Green'因为上面代码里它的值是2
```

3）、不定类型 any

有时候，我们会想要为那些在编程阶段还不清楚类型的变量指定一个类型。这些值可能来自于动态的内容，比如来自用户输入或第三方代码库。 这种情况下，我们不希望类型检查器对这些值进行检查而是直接让它们通过编译阶段的检查。 那么我们可以使用 any类型来标记这些变量

在对现有代码进行改写的时候，any类型是十分有用的，它允许你在编译时可选择地包含或移除类型检查。 你可能认为 Object有相似的作用，就像它在其它语言中那样。 但是 Object类型的变量只是允许你给它赋任意值 - 但是却不能够在它上面调用任意的方法，即便它真的有这些方法

当你只知道一部分数据的类型时，any类型也是有用的。 比如，你有一个数组，它包含了不同的类型的数据：

```
let list: any[] = [1, true, "free"];

list[1] = 100;
```

4）、无返回值

某种程度上来说，void类型像是与any类型相反，它表示没有任何类型。 当一个函数没有返回值时，你通常会见到其返回值类型是 void：

声明一个void类型的变量没有什么大用，因为你只能为它赋予undefined和null：

5）、类型断言（类型转换）as语法或者尖括号（）二者效果等价

类型断言有两种形式。 其一是“尖括号”语法：

```
let someValue: any = "this is a string";

let strLength: number = (<string>someValue).length;
```

另一个为 as 语法：

```
let someValue: any = "this is a string";

let strLength: number = (someValue as string).length;
```

2、变量声明

1）、用var声明变量会发生变量提升（在全局也可以访问到函数内部的变量，使用for方法时有时就会发生，如：for循环挨个儿操作i值），当用let声明一个变量，它使用的是词法作用域或块作用域。 不同于使用 var声明的变量那样可以在包含它们的函数外访问，块作用域变量在包含它们的块或for循环之外是不能访问的。（不存在变量提升）

直到声明它的代码之前的区域都属于 暂时性死区。（暂存死区）

2）、let在相同作用域中不允许重复声明变量

3）、const：它在定义后不能再改变

使用最小特权原则，所有变量除了你计划去修改的都应该使用const。 基本原则就是如果一个变量不需要对它写入，那么其它使用这些代码的人也不能够写入它们，并且要思考为什么会需要对这些变量重新赋值。 使用 const也可以让我们更容易的推测数据的流动