**函数定义**

语法格式如下所示：

function function\_name()

{

    // 执行代码

}

**函数返回值**

function function\_name(): return\_type {

    // 语句

    return value;

}

**实例**

// 函数定义

function greet(): string { // 返回一个字符串

    return "Hello World"

}

function caller() {

    var msg = greet() // 调用 greet() 函数

    console.log(msg)

}

// 调用函数

caller()

**带参数函数**

function func\_name( param1 [:datatype], param2 [:datatype]) {

}

**可选参数和默认参数**

**可选参数**

在 TypeScript 函数里，如果我们定义了参数，则我们必须传入这些参数，除非将这些参数设置为可选，可选参数使用问号标识 ？。

function buildName(firstName: string, lastName?: string) {

    if (lastName)

        return firstName + " " + lastName;

    else

        return firstName;

  }

  let result1 = buildName("Bob");  // 正确

  let result2 = buildName("Bob", "Adams", "Sr.");  // 错误，参数太多了

  let result3 = buildName("Bob", "Adams");  // 正确

**默认参数**

function function\_name(param1[:type],param2[:type] = default\_value) {

}

**剩余参数**

function buildName(firstName: string, ...restOfName: string[]) {

    return firstName + " " + restOfName.join(" ");

}

let employeeName = buildName("Joseph", "Samuel", "Lucas", "MacKinzie");

函数的最后一个命名参数 restOfName 以 ... 为前缀，它将成为一个由剩余参数组成的数组，索引值从0（包括）到 restOfName.length（不包括）。

**匿名函数**

匿名函数在程序运行时动态声明，除了没有函数名外，其他的与标准函数一样。

我们可以将匿名函数赋值给一个变量，这种表达式就成为函数表达式。

var res = function ([arguments]) { ... }

**匿名函数自调用**

匿名函数自调用在函数后使用 () 即可：

(function () { var x = "Hello!!"; console.log(x) })()

**递归函数**

递归函数即在函数内调用函数本身。

function factorial(number) {

    if (number <= 0) {         // 停止执行

        return 1;

    } else {

        return (number \* factorial(number - 1));     // 调用自身

    }

};

console.log(factorial(6));      // 输出 720

**Lambda 函数**

Lambda 函数也称之为箭头函数。

([param1, parma2, …param n]) => statement;

**函数重载**

重载是方法名字相同，而参数不同，返回类型可以相同也可以不同。