# 联合类型

联合类型是由两个和以上其他类型组成的类型，比如：

typescript

复制代码

let a: number | string;

变量 a 的值既可以是 number 类型，也可以是 string 类型：

ini

复制代码

a = 10;

a = ‘hello’；

在使用联合类型时，如果**变量的值还没有确定**，也就是此时存在多种可能性，此时**只能访问联合类型中公共的属性和方法**，比如：

一旦联合类型的变量的值确定下来了，此时就**只能访问对应类型的属性和方法**了：

### 实例

|  |
| --- |
| //联合类型就是说一个变量可以允许他的类似是两个或者以上的类型的并集使用|来做并集 let aunion:string|boolean|number aunion = 'hello' console.log(aunion.toUpperCase()); aunion = -35 console.log(Math.abs(aunion)) aunion = false aunion = !aunion console.log(aunion); |

### 运行效果

|  |
| --- |
|  |

### 注意在定义联合类型的变量的时候不需要给他赋值，可以以后再赋值，定义时赋值反而是多余的

## 其实联合类型也可以用来限制变量的取值范围

|  |
| --- |
|  |

## 还可以用在函数中

|  |
| --- |
| const test = (info: string | number) => {  // ...  } |

# 类型别名

当有一个复杂的类型时，或者一个类型被频繁使用时，可以给该类型起一个别名，来简化使用。

比如声明一个描述人的对象，可以这么写：

let p: {name: string, age: number}= {

 name: "John",

 age: 20

}

但是如果要声明多个人时，就有点繁琐了。此时可以使用类型别名：

type Person = {

 name: string,

 age: number

}

​

let p: Person= {

 name: "John",

 age: 20

}

更多例子：

// 基本类型

type Num = Number;

let n: Num = 1;

// 联合类型

type Shape = { kind: "circle"; radius: number } | { kind: "square"; x: number } | { kind: "triangle"; x: number; y: number };

function area(s: Shape) {

if (s.kind === "circle") {

return Math.PI \* s.radius \* s.radius;

} else if (s.kind === "square") {

return s.x \* s.x;

} else {

return (s.x \* s.y) / 2;

}

}

// 对象

type UserInfo = {name: string};

// 函数

type GetInfo = () => string;

// 元祖

type Size = [number, number];

let x: Size = [10, 99.9];

类型别名还经常和联合类型一起使用，比如：

**type union =string |number|boolean**

之后，可以给变量 union 赋值这三种类型中的任意一种值。

let u1:union = 1

let u2:union = '2'

let u3:boolean = true

# 交叉类型

就是将多个类型合并为一个类型，使用&来定义交叉类型

|  |
| --- |
|  |

### 实例

|  |
| --- |
| //交叉类型 interface F1{  name:string,  age:number } interface F2{  gender:string,  email:string,  phone:string }  type MyType = F1 & F2  let john:MyType = {  name:'John',  age:36,  gender:'male',  email:"john123@gmail.com",  phone:'1234567' }  console.log(john); |

#### 效果：

|  |
| --- |
|  |

# type和interface的区别，参考[type和interface的区别](ts学习笔记-type和interface的区别.docx)