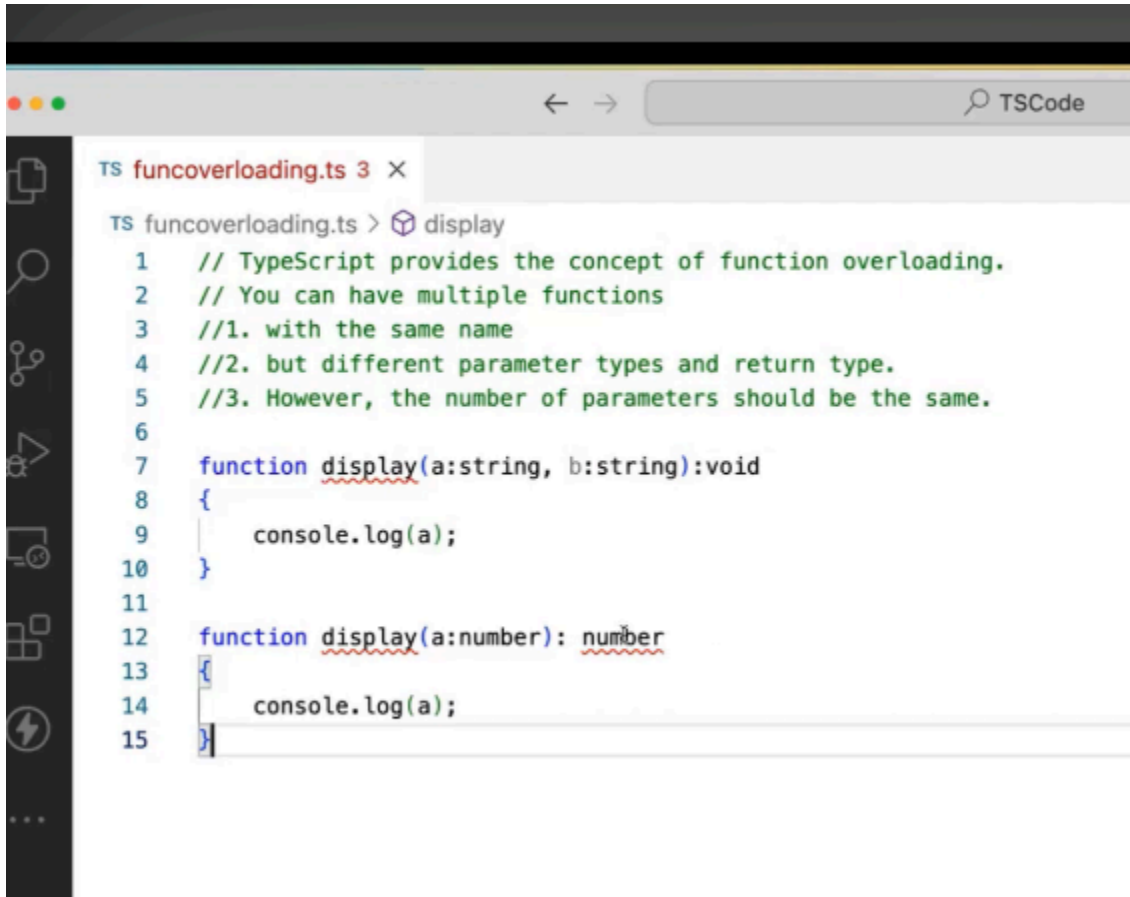


Possible in ts not possible in js - function overloading.

Error cases - wrong use of function overloading in ts.




```
TS funcoverloading.ts 3 x
TS funcoverloading.ts > display
1 // TypeScript provides the concept of function overloading.
2 // You can have multiple functions
3 //1. with the same name
4 //2. but different parameter types and return type.
5 //3. However, the number of parameters should be the same.
6
7 function display(a:string, b:string):void
8 {
9     console.log(a);
10 }
11
12 function display(a:number): number
13 {
14     console.log(a);
15 }
```

← →

TSCode

TS funcoverloading.ts 2 ×

TS funcoverloading.ts >  display

1

// TypeScript provides the concept of function overloading.

2

// You can have multiple functions

3

//1. with the same name

4

//2. but different parameter types and return type.

5

//3. However, the number of parameters should be the same.

6


7

function display(a:string):void

8

{

9

 console.log(a);

10

}

11

12

function display(a:number): void

13

{

14

console.log(a);

15

}

4

//2. but different parameter types and return type.

5

//3. However, the number of parameters should be th

6

7

function display(a:string, b:string):void

8

{

9

console.log(a);

10

}

11

12

function display(a:number): number

13

{

14

console.log(a);

15

}

```

3 //1. with the same name
4 //2. but different parameter types and return type.
5 //3. However, the number of parameters should be the same
6
7 function display(a:string, b:string):void
8 {
9     console.log(a);
10 }
11
12 function display(a:number): any
13 {
14     console.log(a);
15 }

```

Right way

```

4 //2. but different parameter types and return type.
5 //3. However, the number of parameters should be the same
6
7 //provide the implementation of the function:
8 function add(a: string, b:string): string;
9 function add(a: number, b:number): number;
10

```

```

10
11 //need to implement this only once:
12 function add(a:any, b:any): any{
13     return a+b;
14 }

```

Changed 7

```

4 //2. but different parameter types and return type.
5 //3. However, the number of parameters should be the same
6
7 //provide the implementation/prototype of the function:
8 function add(a: string, b:string): string;
9 function add(a: number, b:number): number;
10 function add(a: number, b:number): number;
11

```

Changed 13

```

11
12
13 //need to implement with function body this only once:
14 function add(a:any, b:any): any{
15     return a+b;
16 }

```

Changed 10

```

TS funcoverloading.ts > ...
6
7 //provide the implementaion/prototype of the function:
8 function add(a: string, b:string): string;
9 function add(a: number, b:number): number;
10 function add(a: boolean, b:boolean): boolean;
11
16 }
17
18 let s1 = add('hello', 'world');//hello world
19 let s2 = add(10,20); 30
20 let s3 = add(true, false);
21
22 console.log(s1);
23 console.log(s2);
24 console.log(s3);
25

```

Not allowed

```

TS funcoverloading.ts 1 x
TS funcoverloading.ts > ...
6
7 //provide the implementaion/prototype of the function:
8 function add(a: string, b:string): string;
9 function add(a: number, b:number): number;
10 function add(a: boolean, b:boolean): boolean;
11 function add(a: number): boolean;
12

```

```

12
13
14
15 //need to implement with function body this only once:
16 function add(a:any, b:any): any{
17     return a+b;
18 }
19
20 function add(a:any): any{
21     return a;
22 }

```

```
funoverload1.ts 3 x funoverload2.ts 1 funoverload3.ts 1 funoverload4.ts 3 funoverload5.ts 6 funoverload6.ts 1  
funoverload1.ts > display1  
1 //error cases of functional overload.  
2  
3 //not allowed in ts.  
4  
5 function display1(a:string, b:string):void{ //Duplicate function implementation.ts(2393)  
6     console.log(a)  
7 }  
8  
9 function display1(a:number):number{ //Duplicate function implementation.ts(2393)  
10     console.log(a)  
11 }
```

```
funoverload2.ts > display2
1 //allowed in compile time.
2 //during run time get error.
3
4 function display2(a:string):void{
5     console.log(a)
6 }
7
8 // function display(a:number):void{
9 //     //funoverload2.ts:7:10 - error TS2393: Duplicate function implementation.
10 //     console.log(a)
11 // }
12
13 display2('tiger') //tiger
14 // display(-34324.3243)
15 //Argument of type 'number' is not assignable to parameter of type 'string'.ts(2345)
```

```
... TS funoverload3.ts X TS funoverload4.ts 3 TS funoverload5.ts 6 TS funoverload6.ts 6 TS funoverload7.ts 1
TS funoverload3.ts > ...
1 //here also at run time we come to know not at compile time.
2
3 function display3(a:string, b:string):void{
4     console.log(a)
5 }
6
7 // function display(a:number):any{
8 //     //funoverload3.ts:7:10 - error TS2393: Duplicate function implementation.
9 //     //run time error.
10 //     console.log(a)
11 // }
12
13
14 display3('tiger', 'lion')
15 // display(-32434.32434)
16 //Argument of type 'number' is not assignable to parameter of type 'string'.ts(2345)
```

```
... TS funoverload4.ts X TS funoverload5.ts 6 TS funoverload6.ts 6 TS funoverload7.ts 1
TS funoverload4.ts > [i2]
1 //right way.
2
3 //first define the functions to overload.
4
5 function add4(a:string, b:string):string
6 function add4(a:number, b:number):number
7
8 //implement the function only once.
9 function add4(a:any, b:any):any{
10     return a+b
11 }
12
13 let i1=add4(10,20)
14 console.log(i1) //30
15
16 let i2=add4('karan', 'tiger')
17 console.log(i2) //karantiger
```

...

funoverload5.ts X

funoverload6.ts 6

funoverload7.ts 1

funoverload5.ts > [🔗] i7

1 ✓ //right way.

2

3 //first define the functions to overload.

4 //defined some more combinations.

• 5

6 function add5(a:string, b:string):string

7 function add5(a:number, b:number):number

8 function add5(a:number, b:boolean):any

9 function add5(a:string, b:number):string

10 function add5(a:boolean, b:boolean):boolean

11

12 //implement the function only once.

13 ✓ function add5(a:any, b:any):any{

14 | return a+b

6

15 }

16

1

```
15   }  
16  
17   let i3=add5(10,20)  
18   console.log(i3) //30  
19  
20   let i4=add5('karan', 'tiger')  
21   console.log(i4) //karantiger  
22  
23   let i5=add5(324324, false)  
24   console.log(i5) //324324  
25  
26   let i6=add5('324324', 45)  
27   console.log(i6) //32432445  
28   💡  
29   let i7=add5(true, false)
```

```
29   let i7=add5(true, false)  
30   console.log(i7) //1
```



```
... funoverload6.ts X funoverload7.ts
funoverload6.ts > ...
1 //right way.
2
3 //first define the functions to overload.
4 //defined some more combinations.
5
6 function add6(a:string, b:string):string
7 function add6(a:number, b:number):number
8 function add6(a:number, b:boolean):any
9 function add6(a:string, b:number):string
10 function add6(a:boolean, b:boolean):boolean
11 // function add6(a:boolean):boolean
12 //funoverload6.ts:11:10 - error TS2394: This overload signature
13 // is not compatible with its implementation signature.
14 // function add6(a:number):number
15 //funoverload6.ts:14:10 - error TS2394:
16 // This overload signature is not compatible with its implementation signature.
17
18 //implement the function only once.
19 function add6(a:any, b:any):any{
20     return a+b
```

```
20     return a+b
21 }
22
23 let q1=add6(10,20)
24 console.log(q1) //30
25
26 let q11=add6('karan', 'tiger')
27 console.log(q11) //karantiger
28
29 let q111=add6(324324, false)
30 console.log(q111) //324324
31
32 let q1111=add6('324324', 45)
33 console.log(q1111) //32432445
34
35 let q11111=add6(true, false)
36 console.log(q11111) //1
```

```
● 36 console.log(q11111) //1
37
38 ✓ // let q111111=add6(true)
39 // console.log(q111111)
40
41 // let q1111111=add6(34435)
42 // console.log(q1111111)
```

```
... funoverload7.ts X
funoverload7.ts > ...
1  //cant implement more than once.
2  //compile time - throws error when calling function with one argument.
3  //run time - shows duplicate function.
4
5  function add7(a:any, b:any):any{
6      return a+b
7  }
8
9
10 // function add7(a:any):any{
11 //     //funoverload7.ts:8:10 - error TS2393: Duplicate function implementation.
12 //     return a+b
13 // }
14
15 let r1=add7(1,2)
16 console.log(r1)
17
18 // console.log(r1)
19
20 // let r2=add7(3)
21 // //Expected 2 arguments, but got 1.ts(2554)
22 // // funoverload4.ts(5, 24): An argument for 'b' was not provided.
23 // console.log(r2)
```