Tuple

This means i want to fix some points in array-like

I want the length of the array to be 2

1st element should be the number for roll Number

The 2nd should be a string for Name.

When we write

Let x:(number | string)

Let x:[number, string)

They have different meaning

function eg1(){

let a = [10, "Good", 50, 40];

console.log(a);

a = [1,2,3,"Correct", 5,6,7];

console.log(a);

let student:[number, string];

student = [101, "Ramesh"];

console.log(student);

//student = [1,2,3]; // error type number is not assignable to type string

//console.log(student);

}

eg1();

/\*

output:

[ 10, 'Good', 50, 40 ]

[ 1, 2, 3, 'Correct', 5, 6, 7 ]

[ 101, 'Ramesh' ]

\*/

We can not assign non tuple values to tuple

function eg2(){

let student:[number, string];

let k = [101, "Sameer"]; // k is not a tuple

student = k;

console.log(student);

}

eg2();

/\*\*

Error:

Type '(string | number)[]' is not assignable to type '[number, string]'.

Target requires 2 element(s) but the source may have fewer.

4 student = k;

~~~~~~~

\*/

Why?

A = [10,20,30];

B = A;

//Here B will start pointing towards A. (B and A are pointers)

//There will not be a new object for B.

If you make any changes in B it will also reflect in A.

Here A and B are different variables with different addresses but pointing towards the same array.

function eg3(){

let a = [10,20,30];

let b=a;

console.log(a);

console.log(b);

b[3] = 400;

console.log(a);

console.log(b);

let student1:[number, string];

let student2:[number, string];

student1=[101, "Sameer"];

student2 = student1;

console.log(student1);

console.log(student2);

student2[0] = 102;

console.log(student1);

console.log(student2);

}

eg3();

/\*

Output:

[ 10, 20, 30 ]

[ 10, 20, 30 ]

[ 10, 20, 30, 400 ]

[ 10, 20, 30, 400 ]

[ 101, 'Sameer' ]

[ 101, 'Sameer' ]

[ 102, 'Sameer' ]

[ 102, 'Sameer' ]

\*/

POC they all are pointers.

Now new Example to pass the argument as a rest parameter while the parameter is not

function eg4\_some(x:number, y:number){

console.log(x+", "+y);

}

function eg4(){

let a = [10,20];

eg4\_some(...a);

}

eg4();

/\*\*

output: ERROR

A spread argument must either have a tuple type or be passed to a rest parameter.

// means either pass tuple

or change the parameter to the rest parameter.

\*/

Answer

function eg5\_some(x:number, y:number){

console.log(x\*y);

}

function eg5(){

let x:[number, number];

x = [10,20];

eg5\_some(...x);

}

eg5();

We can return more then one value at once like

Return [10,20,30]

Now we can receiver in same manner also

Lets take example:

function eg7\_some(x:number){

let square = x\*x;

let e, factorial;

e = 1;

factorial = 1;

while(e<=x){

factorial = e\*factorial;

e++;

}

return [x, square, factorial];

}

function eg7(){

let [num, sqr, fact] = eg7\_some(5);

console.log(num);

console.log(sqr);

console.log(fact);

}

eg7();

/\*

5

25

120

\*/

You can not change values if you use them as const or make them readonly.

function eg9(){

let a = [10, "good"] as const;

let b = a;

console.log(a);

console.log(b);

b[0]=20; // Cannot assign to '0' because it is a read-only property.

let c = 20;

c = a; // Type 'readonly [10, "good"]' is not assignable to type 'number'.

let d:[number, string];

d = a; // The type 'readonly [10, "good"]' is 'readonly' and cannot be assigned to the mutable type '[number, string]'.

Const f = a;

console.log(a) // will work

}

If you initialise any const variable then you need to assign value to it.

const a: [numner, string]; // error

const a: [number, string] = [10, “good”]; // correct

We can not change its value and if you assign this value to another variable then that new variable will have the same type.