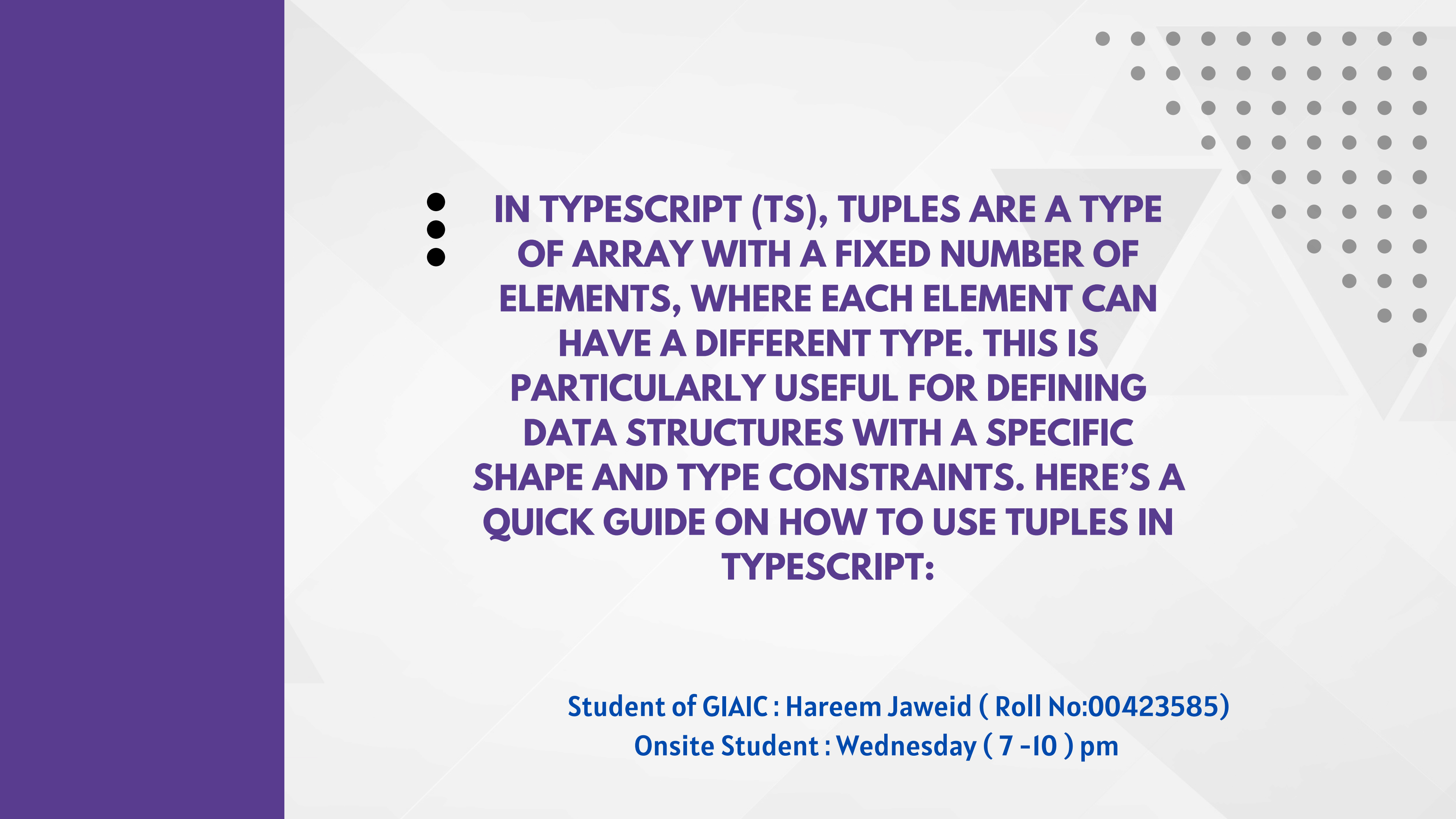


TUPLES IN TYPESCRIPT

Student of GIAIC : Hareem Jaweid (Roll No:00423585)

Onsite Student : Wednesday (7 -10) pm



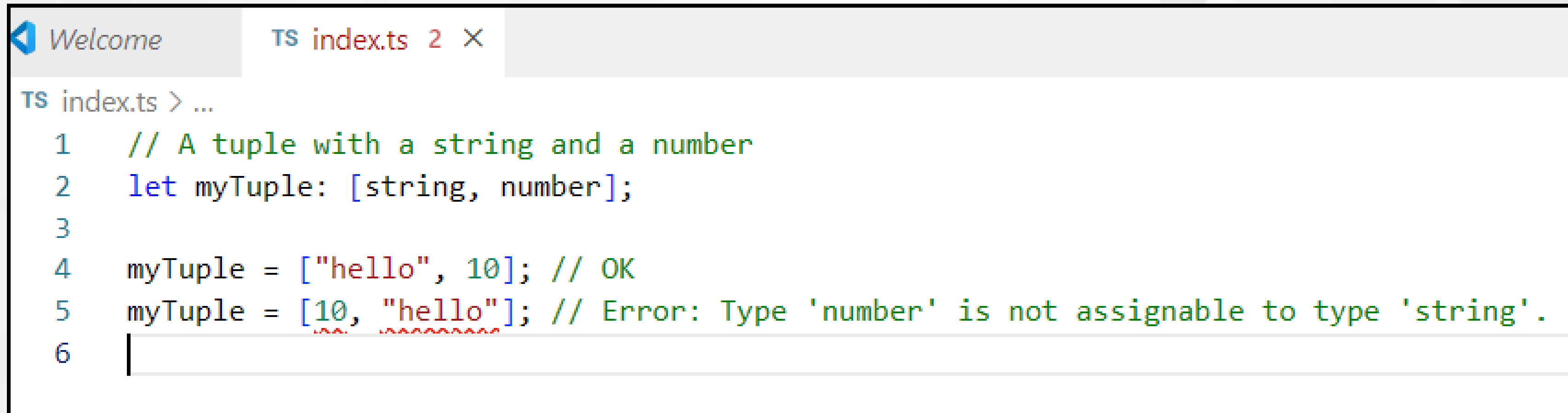
- **IN TYPESCRIPT (TS), TUPLES ARE A TYPE**
- **OF ARRAY WITH A FIXED NUMBER OF**
- **ELEMENTS, WHERE EACH ELEMENT CAN**
- HAVE A DIFFERENT TYPE. THIS IS**
- PARTICULARLY USEFUL FOR DEFINING**
- DATA STRUCTURES WITH A SPECIFIC**
- SHAPE AND TYPE CONSTRAINTS. HERE'S A**
- QUICK GUIDE ON HOW TO USE TUPLES IN**
- TYPESCRIPT:**

Student of GIAIC : Hareem Jaweid (Roll No:00423585)

Onsite Student : Wednesday (7 -10) pm

-
-
- ## DEFINING A TUPLE

To define a tuple, you use the array syntax but specify the types of each element in the array.



The screenshot shows a code editor with a tab labeled 'TS index.ts 2 X'. The code content is as follows:

```
TS index.ts > ...
1  // A tuple with a string and a number
2  let myTuple: [string, number];
3
4  myTuple = ["hello", 10]; // OK
5  myTuple = [10, "hello"]; // Error: Type 'number' is not assignable to type 'string'.
6  |
```

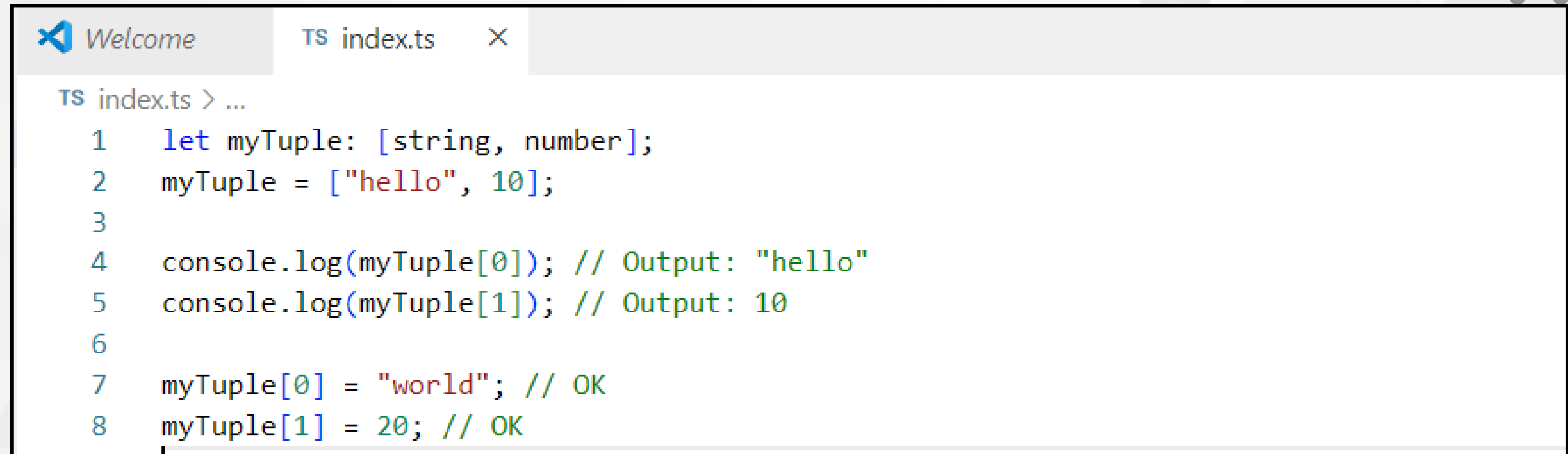
Line 5 shows a TypeScript error because the value 10 (a number) is being assigned to a position in the tuple that is typed as 'string'.

Student of GIAIC : Hareem Jaweid (Roll No:00423585)

Onsite Student : Wednesday (7 -10) pm

ACCESSING AND MODIFYING TUPLE ELEMENTS

You can access and modify elements in a tuple just like you would with an array.

A screenshot of a code editor window with two tabs: 'Welcome' and 'TS index.ts'. The 'TS index.ts' tab is active, showing a TypeScript file. The code defines a tuple 'myTuple' of type [string, number], initializes it with 'hello' and 10, logs its elements, and then modifies the first element to 'world' and the second element to 20. Line numbers 1 through 8 are visible on the left side of the code.

```
TS index.ts > ...
1  let myTuple: [string, number];
2  myTuple = ["hello", 10];
3
4  console.log(myTuple[0]); // Output: "hello"
5  console.log(myTuple[1]); // Output: 10
6
7  myTuple[0] = "world"; // OK
8  myTuple[1] = 20; // OK
```

Student of GIAIC : Hareem Jaweid (Roll No:00423585)

Onsite Student : Wednesday (7 -10) pm

● ● ● USING TUPLE TYPES IN FUNCTIONS

Tuples can be used as parameter types and return types for functions.

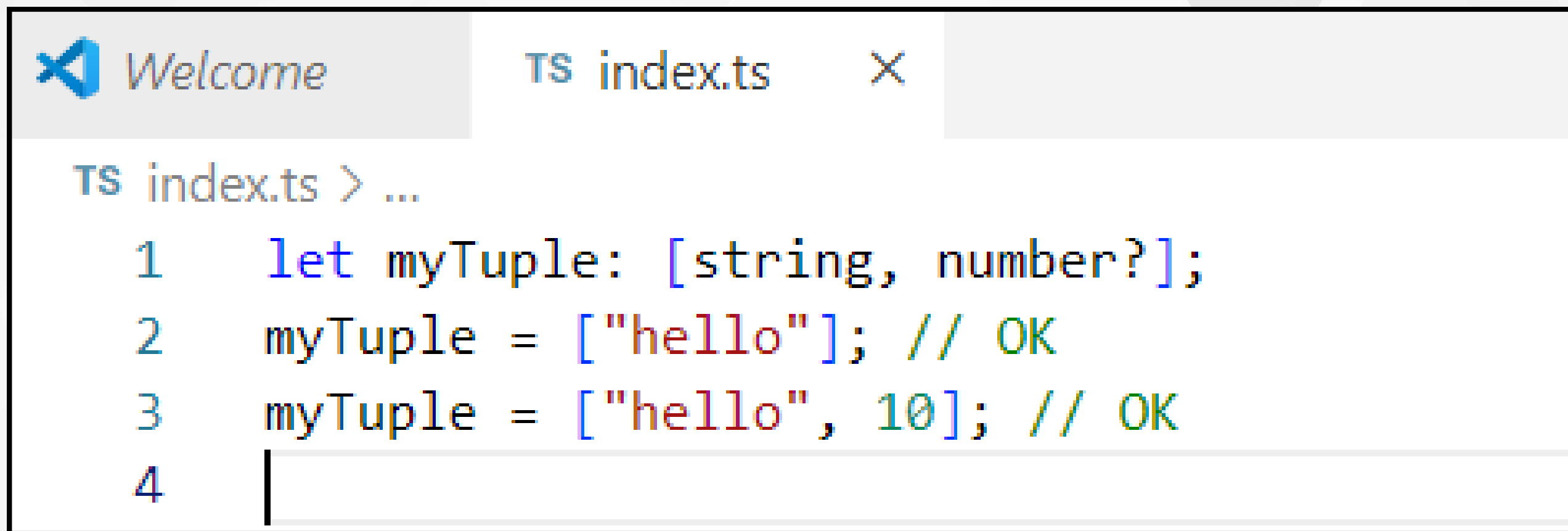
```
TS index.ts > ...
1  // Function that returns a tuple
2  function getTuple(): [string, number] {
3      |   return ["hello", 10];
4      |
5      |
6  // Function that takes a tuple as a parameter
7  function printTuple(tuple: [string, number]) {
8      |   console.log(`String value: ${tuple[0]}`);
9      |   console.log(`Number value: ${tuple[1]}`);
10     |
11     |
12     let myTuple = getTuple();
13     printTuple(myTuple);
14
```

Student of GIAIC : Hareem Jaweid (Roll No:00423585)

Onsite Student : Wednesday (7 -10) pm

● ● ● OPTIONAL ELEMENTS

You can define optional elements in a tuple using the question mark ? syntax.



```
TS index.ts > ...  
1  let myTuple: [string, number?];  
2  myTuple = ["hello"]; // OK  
3  myTuple = ["hello", 10]; // OK  
4  |
```

Student of GIAIC : Hareem Jaweid (Roll No:00423585)

Onsite Student : Wednesday (7 -10) pm

REST ELEMENTS

You can use the rest syntax to define tuples with a variable number of elements of a specific type.



Welcome

TS index.ts



TS index.ts > ...

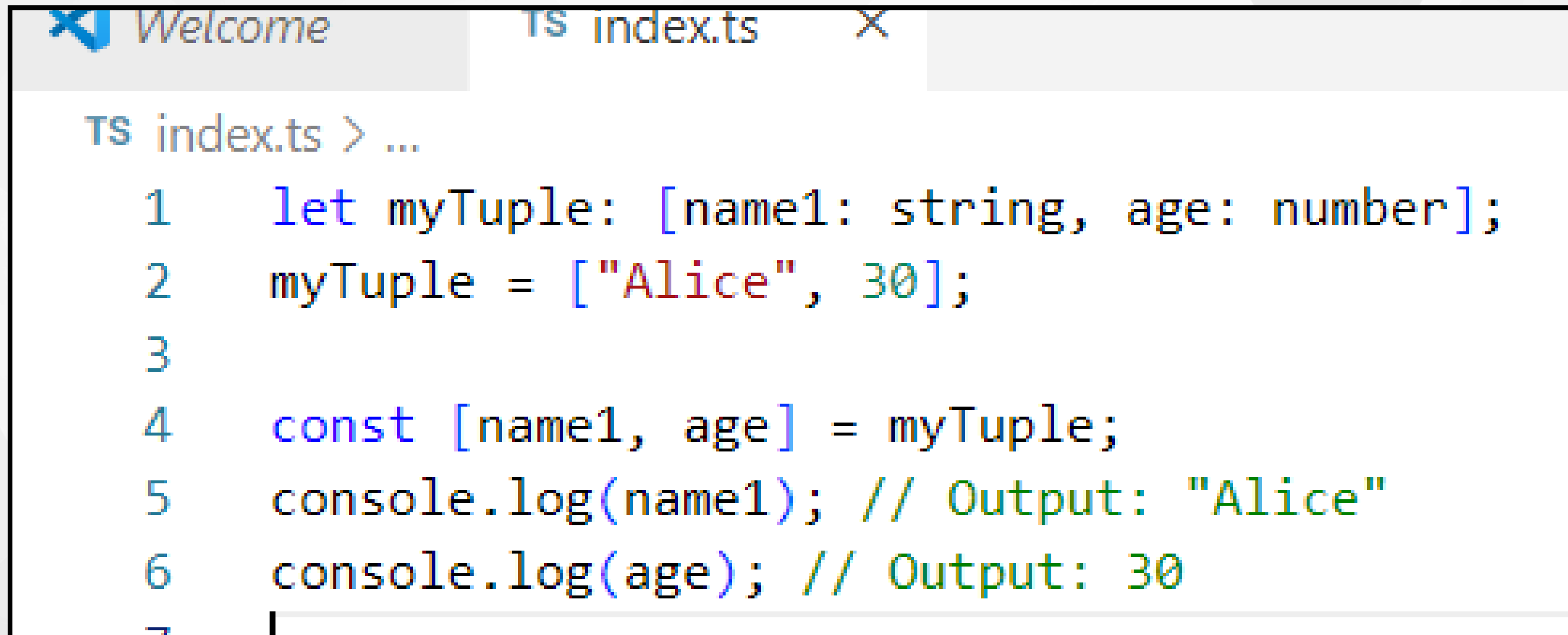
```
1  let myTuple: [string, ...number[]];
2  myTuple = ["hello", 1, 2, 3]; // OK
3  myTuple = ["hello"]; // OK
4  |
```

Student of GIAIC : Hareem Jaweid (Roll No:00423585)

Onsite Student : Wednesday (7 -10) pm

• NAMED TUPLES (WITH LABELLED • TUPLE ELEMENTS)

As of TypeScript 4.0, you can add labels to tuple elements to make the code more readable.

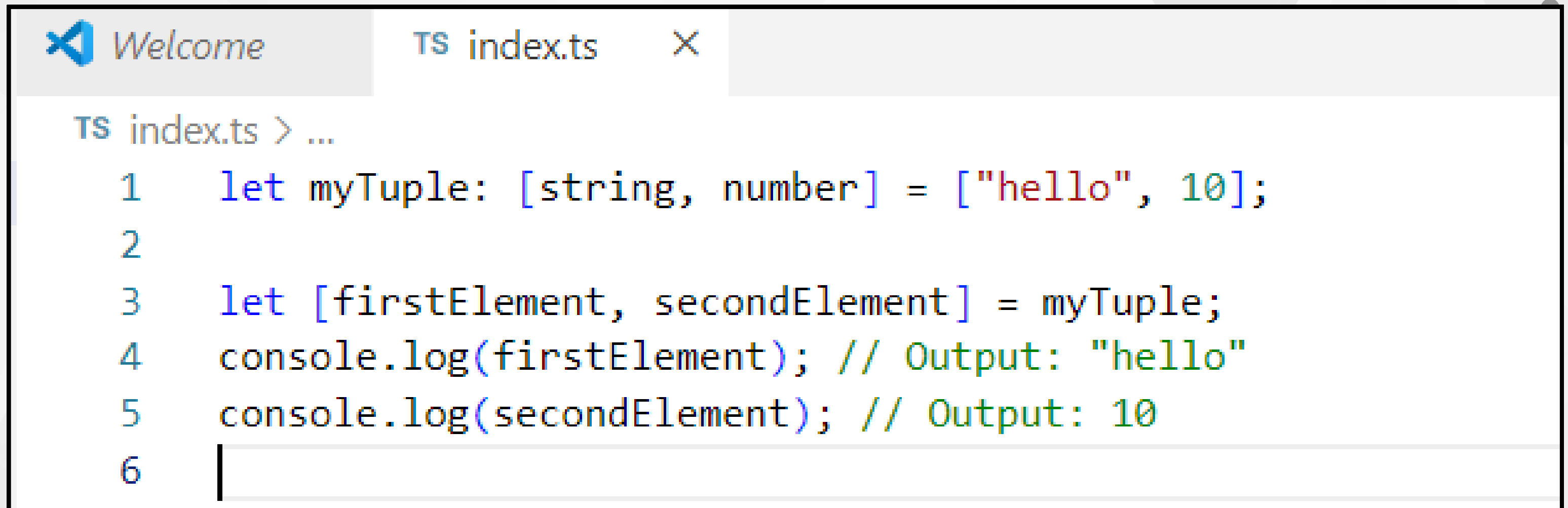
A screenshot of a code editor window. The title bar shows 'Welcome' and 'TS index.ts'. The code is as follows:

```
TS index.ts > ...  
1  let myTuple: [name1: string, age: number];  
2  myTuple = ["Alice", 30];  
3  
4  const [name1, age] = myTuple;  
5  console.log(name1); // Output: "Alice"  
6  console.log(age); // Output: 30  
7
```

Student of GIAIC : Hareem Jaweid (Roll No:00423585)
Onsite Student : Wednesday (7 -10) pm

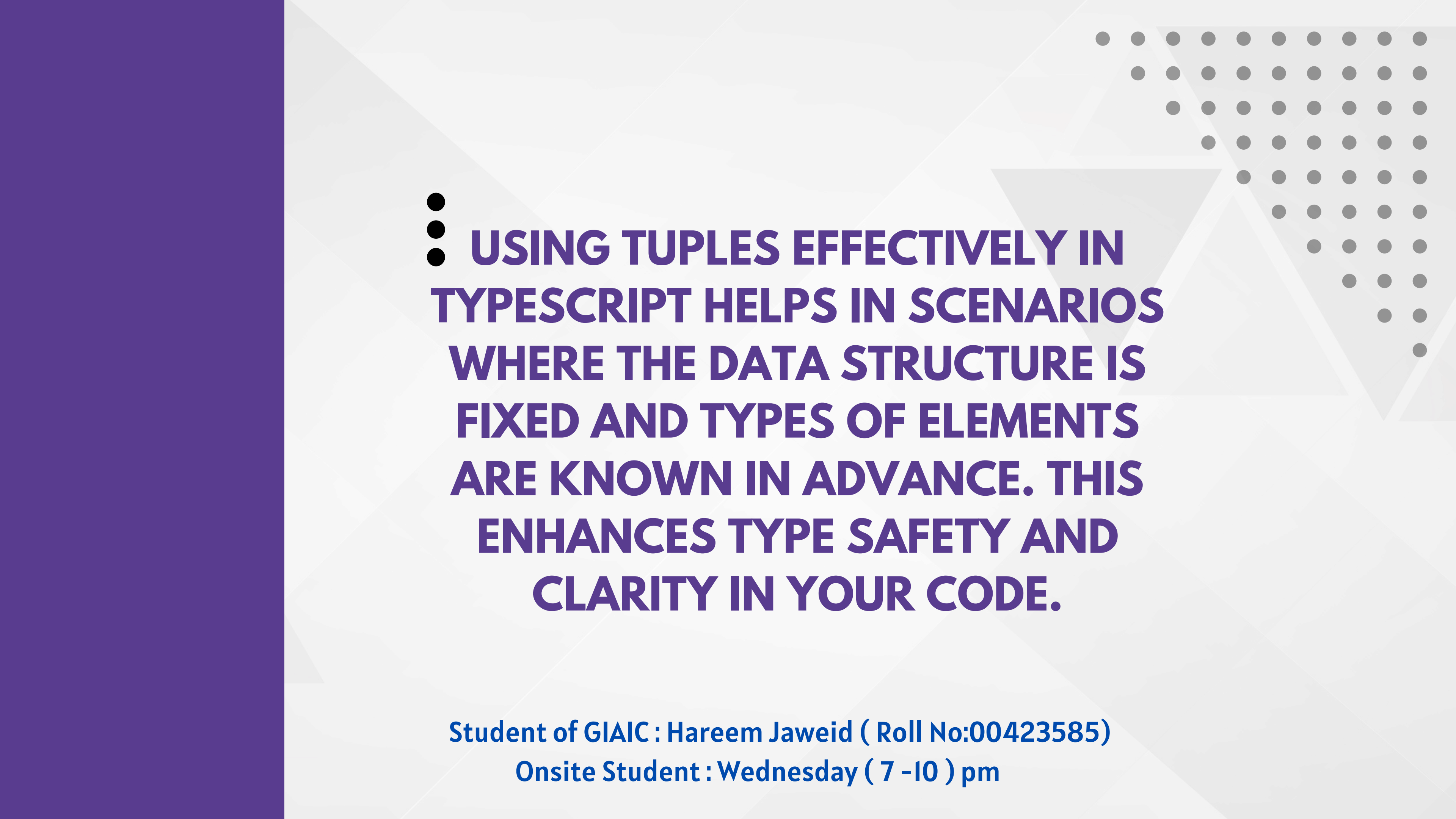
DESTRUCTURING TUPLES

Tuples can be destructured just like arrays.



```
TS index.ts  X
TS index.ts > ...
1  let myTuple: [string, number] = ["hello", 10];
2
3  let [firstElement, secondElement] = myTuple;
4  console.log(firstElement); // Output: "hello"
5  console.log(secondElement); // Output: 10
6  |
```

Student of GIAIC : Hareem Jaweid (Roll No:00423585)
Onsite Student : Wednesday (7 -10) pm

- 
- -
 - **USING TUPLES EFFECTIVELY IN TYPESCRIPT HELPS IN SCENARIOS WHERE THE DATA STRUCTURE IS FIXED AND TYPES OF ELEMENTS ARE KNOWN IN ADVANCE. THIS ENHANCES TYPE SAFETY AND CLARITY IN YOUR CODE.**

Student of GIAIC : Hareem Jaweid (Roll No:00423585)
Onsite Student : Wednesday (7 -10) pm



⋮ **THANK YOU**

Student of GIAIC : Hareem Jaweid (Roll No:00423585)

Onsite Student : Wednesday (7 -10) pm