TypeScript Tuples – Syntax and Examples

TypeScript Tuples

In our previous tutorial, we have learnt about TypeScript Arrays. Arrays hold multiple values of same datatype. But at times, we need to store multiple fields of different datatypes in a single variable. TypeScript Tuples are meant for this purpose. Tuples can store multiple fields that may belong to varied datatypes. They resemble structures in C programming language.

Tuples can be used like any other variables. They can be passed as arguments in a function call.

In this tutorial, we shall learn all the CRUD operations with TypeScript Tuples.

- Create or Initialize a Tuple
- Read or Access fields of a Tuple
- Update or Modify fields of a Tuple
- Delete or Clear the fields of a Tuple

Tuples fields are index based. If there are `N` fields in a tuple, their index starts from `0` to `N-1`.

Create or Initialize a Tuple

To create or initialize a tuple, declare tuple as a variable and load the fields into tuple using assignment operator (=) as shown in the following :

```
var tuple_name = [field1, field2, .., fieldn]
```

Fields are separated by command and are enclosed in square brackets as a single unit.

`var` is the keyword

`tuple_name` is the name of the tuple which is used for referencing the tuple in the program thereafter.

`fieldn` is the nth field value.

Following is an example tuple named student1.

var student1 = [“Roshan”, 15, “AB School”]

student1 is a tuple with three fields. Field 1 is name of the student. Field 2 is age of the student. Field 3 is the school the student is studying.

In the above example, both declaration and initialization are done in a single statement. Declaration and initialization of a tuple can be done separately by initially declaring the tuple as empty.

example.ts

```
var student1 = []

student1[0] = "Roshan"
student1[1] = 15
student1[2] = "AB School"
```

Read or Access fields of a Tuple

To read or access the fields of a TypeScript Tuple, use the **index** of the fields as shown in the following.

example.ts

```
var student1 = ["Roshan", 15, "AB School"]

console.log("Name of the student is : "+student1[0])
console.log("Age of the student is : "+student1[1])
console.log(student1[0]+" is studying in "+student1[1])
```

Output

```
Name of the student is : Roshan
Age of the student is : 15
Roshan is studying in AB School
```

A tuple can be de-structured using a tuple with variable names as shown below:

example.ts

```
var student1 = ["Roshan", 15, "AB School"]

var [student_name, age, school] = student1

console.log("Name of the student is : "+student_name)
console.log("Age of the student is : "+age)
console.log(student_name+" is studying in "+school)
```

Update or Modify fields of a Tuple

To update or modify the fields of a TypeScript Tuple, use the **index** of the fields and assignment operator as shown in the following.

example.ts

```
var student1 = ["Roshan", 15, "AB School"]

student1[1] = 16

console.log("Name of the student is : "+student1[0])
console.log("Age of the student is : "+student1[1])
console.log(student1[0]+" is studying in "+student1[1])
```

Output

```
Name of the student is : Roshan
Age of the student is : 16
Roshan is studying in AB School
```

Although the age of a student may not change while executing a program, but the example should demonstrate how a tuple could be updated.

Delete or Clear the fields of a Tuple

A tuple variable cannot be deleted, but its fields could be cleared. To clear the fields of a tuple, assign it with an empty tuple field set as shown in the following.

example.ts

```
var student1 = ["Roshan", 15, "AB School"]
student1 = []
```

Conclusion

In this TypeScript Tutorial, we have learnt how to initialize a tuple, read field members of it, update the fields on the go and clear the fields if necessary with example TypeScript programs.

TypeScript

- TypeScript Tutorial
- TypeScript Variables
- TypeScript if
- TypeScript if-else
- TypeScript Switch
- TypeScript For Loop
- TypeScript While Loop
- TypeScript Do-While Loop
- TypeScript Arrays
- TypeScript Tuples
- TypeScript Unions
- TypeScript Functions
- TypeScript Anonymous Functions

TypeScript Object Oriented Concepts

- TypeScript Class
- TypeScript Inheritance
- TypeScript Method Overriding
- TypeScript Interface

TypeScript String Operations

- TypeScript String Length
- TypeScript String Split