📘 Chapter 8: Tuples

A tuple is a fixed-length array where each item has a specific type, and the order of values matters.

**Basic Syntax -**



You are declaring -

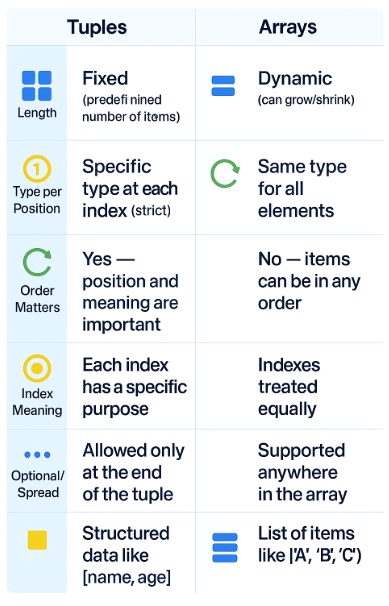
* position 0 must be a string
* position 1 must be a number

Not allowed -



Tuple is a stricter version of an array. It is very particular about the placement of items and their types - the values must follow the same order as defined in the tuple type.

**Tuple vs Array -**



**Named Tuples (Added in TypeScript 4.0+) -**

While defining a tuple, we provide the type for each element inside square brackets [ ]. A **named tuple** simply adds a label to each element, and we define the label along with its type.

This is introduced in Typescript 4.0.



### Benefits - Improves Intellisense.

### 

If you hover over p[0] in VS Code, even though it's just a value in an array, IntelliSense labels it as name instead of just string.

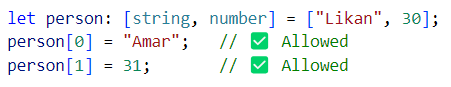
That’s super useful for understanding what the value actually represents.

### At Runtime: The labels do not exist - they are completely removed. So [name: string, age: number] becomes just [string, number].

**Readonly Tuples –**

**By default, tuples in TypeScript are mutable**, which means you **can change** their values **unless** they are marked as readonly.

Mutable Tuple (Default) -



You can reassign individual values just like an array.

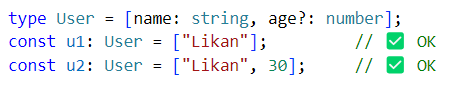
Readonly Tuple -



You cannot change values if the tuple is marked readonly. Use readonly to freeze tuples.

**Optional Tuple Elements -**

In TypeScript, you can make tuple elements **optional** using ?.



Rule- **Optional elements must be placed at the end** of the tuple — you **cannot** have required elements after optional ones.!

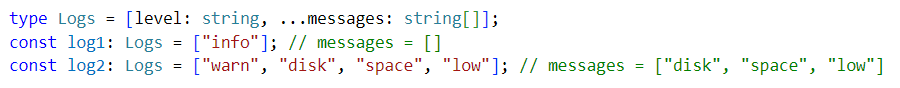
Invalid -



This is not allowed because name is required but comes after an optional value.

**Spread Elements (Rest Tuples) -**

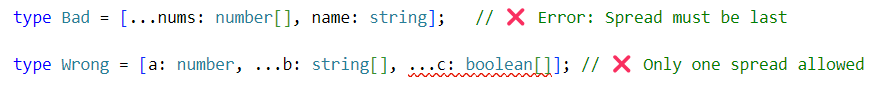
You can use ... in tuple types to allow **flexible tail elements**, just like rest parameters in functions.



messages can hold **zero or more** strings after the first number.

Rule - The spread must come at the end of the tuple - and only one spread element is allowed.

Invalid -



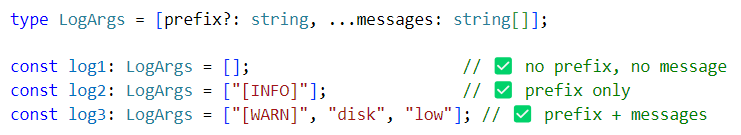
**Combine Optional + Rest –**

TypeScript allows you to mix **optional elements** (?) with a **rest element** (...), but **with strict rules**

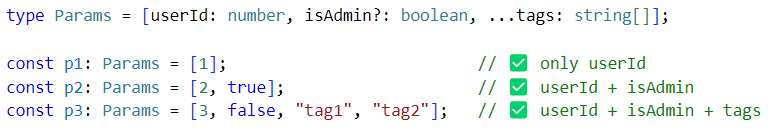
### **Rules**:

1. ✅ Optional elements must come **before** the rest element.
2. ✅ Only one rest element allowed — and it must be **last**.
3. ❌ You can’t put optional elements **after** the rest.

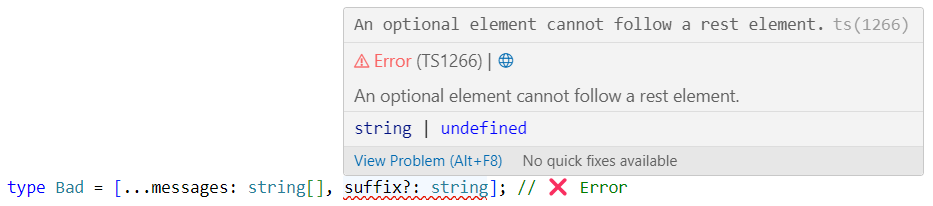
Example 1 — Optional Element + Rest -



Example 2 - Multiple Parameters with Optional and Rest -

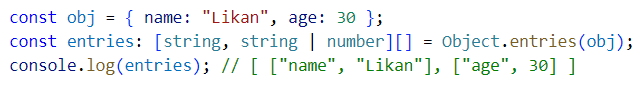


Invalid - Optional **after** Rest (Not Allowed) –



**Tuple Use Case: Object.entries -**

Object.entries returns a tuple array!



When you pass an object to Object.entries(), it returns an **array of tuples**, where:

* Each **tuple** is [key, value]
* ✅ The **key is always a** string (even if the object was defined with symbols or numbers)
* ✅ The **value can be of any type** (string, number, object, boolean, etc.)

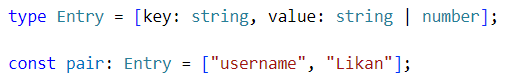
Array of tuples Type -



Array of string Type -



**Deep Dive: Tuple as Key-Value Pairs**



Ideal for dictionary-style access, key-value maps, API responses, and similar use cases.

Interview Questions

***Q1. What’s the difference between a tuple and an array?***

Tuples have fixed length and types per index; arrays are flexible and uniform.

***Q2. Can a tuple have optional and spread elements together?***

Yes! Optional comes before, spread at the end.

***Q3. What is a named tuple?***

A tuple with labels for each element — helps tooling, doesn’t affect runtime.

***Q4. Can I mutate a tuple?***

Yes, unless it's readonly.

***Q5. What happens if you access beyond a tuple’s defined length?***

TypeScript throws a compile-time error.