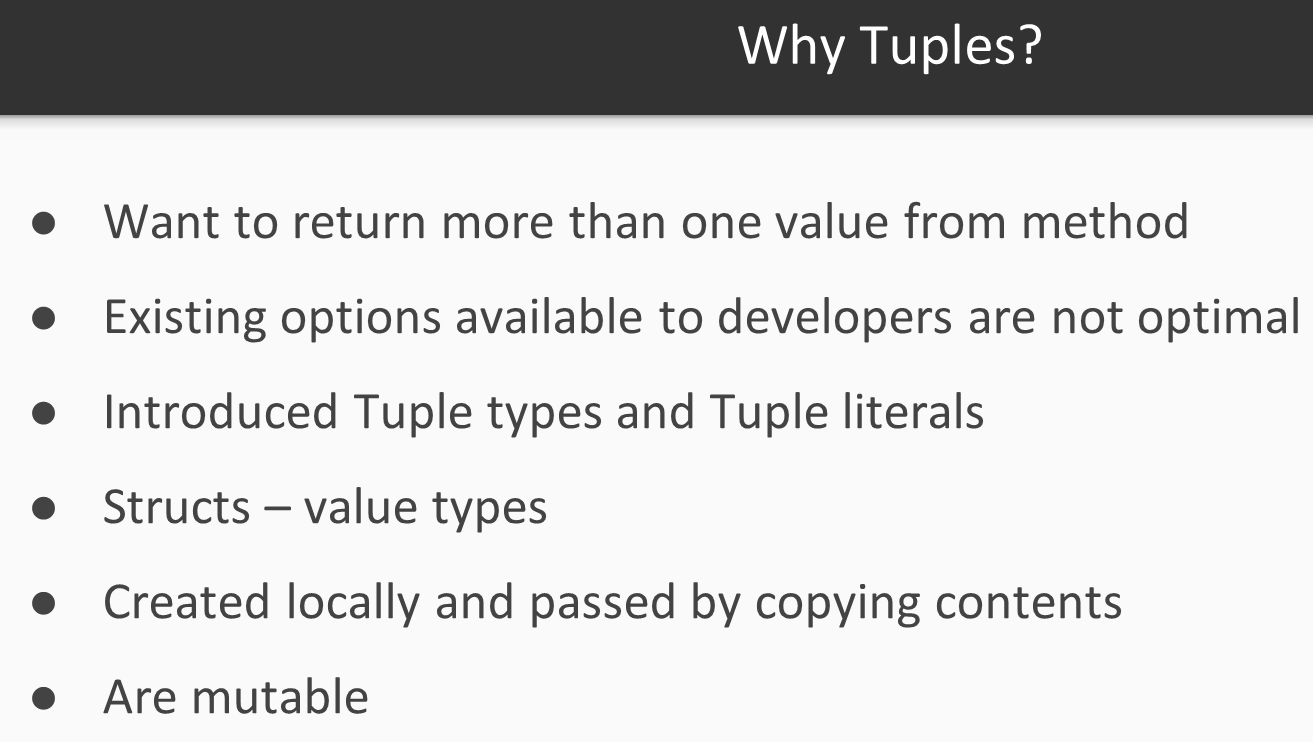
SUMMARY OF TUPLES

02.Working with Tuples.mp4



* - -

C# tuples are types that you define using a lightweight syntax. The advantages include a simpler syntax, rules for conversions based on number (referred to as cardinality) and types of elements, and consistent rules for copies, equality tests, and assignments. As a tradeoff, tuples do not support some of the object-oriented idioms associated with inheritance.

* - -

Ranjit -> I believe

TUPLES == returning Anonymous Objects

* - -

tuples features require the [**ValueTuple**](https://docs.microsoft.com/en-us/dotnet/api/system.valuetuple) types.

ValueTuple structs

* - -

Differences Tuple Types vs Tuple classes

|  |  |  |
| --- | --- | --- |
| Sl.No | Tuple Types | Tuple classes |
| 1 | Tuple types does not enable communicating the meaning of each of the properties. The new language features enable you to declare and use semantically meaningful names for the elements in a tuple. | the Tuple classes named their properties Item1, Item2, and so on. Those names carry no semantic information. |
|  | Using one of the Tuple types means allocating objects. | more performance concerns because they are reference types |

* - -

The language features and the ValueTuple generic structs enforce the rule that you cannot add any behavior (methods) to these tuple types.

* - - -

The ValueTuple struct has fields named Item1, Item2, Item3, and so on

These names are the only names you can use for unnamed tuples.

var unnamed = ("one", "two");

give better names to each field. Doing so creates a named tuple.

var named = (first: "one", second: "two");

* - - -

the field names for a tuple may be provided from the variables used to initialize the tuple. This is referred to as [**tuple projection initializers**](https://docs.microsoft.com/en-us/dotnet/csharp/tuples#tuple-projection-initializers).

var sum = 12.5;

var count = 5;

var accumulation = (count, sum);

* - - - -

IN PROGRESS