

Creating and Deconstructing Tuples



Jesse Liberty

MICROSOFT & XAMARIN MVP

@jesseliberty <http://jesseliberty.me>

What Problem Are We Trying to Solve?



- Getting more than one value returned from a method
- Out parameters don't cut it
 - They are clunky
 - They cannot be used with async methods

What Problem Are We Trying to Solve?

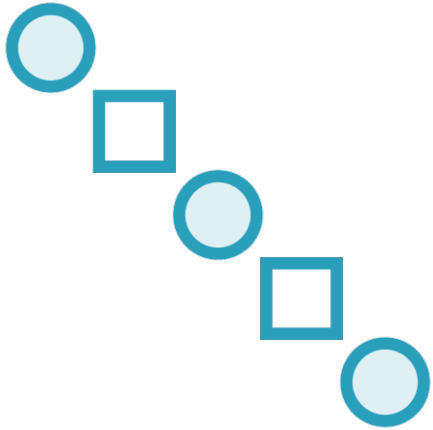


- `System.Tuple<T>`
 - verbose and require allocation of tuple object
- Anonymous types returned through dynamic return type
 - High performance overhead
 - No static type checking

Tuples can be a return type

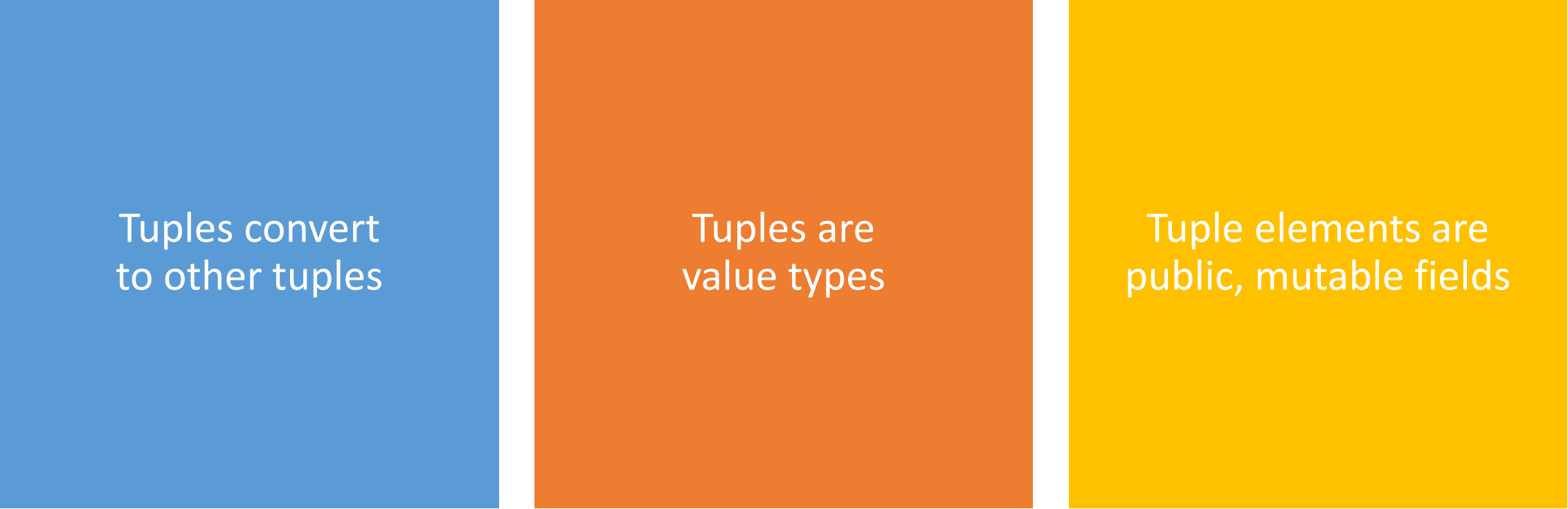
Tuples can be a literal

Tuples



- Each element in a tuple can be accessed with dot notation
- The tuple parts are automatically named Item1, Item2, etc.
- You can name the return tuple parts

```
(string firstName,  
string middleInitial,  
string lastName) myMethod();
```

The image consists of three solid-colored squares arranged horizontally. The left square is blue, the middle square is orange, and the right square is yellow. Each square contains a line of text in white. The text in the blue square reads 'Tuples convert to other tuples'. The text in the orange square reads 'Tuples are value types'. The text in the yellow square reads 'Tuple elements are public, mutable fields'.

Tuples convert
to other tuples

Tuples are
value types

Tuple elements are
public, mutable fields

Demo



- Tuples

Consume Tuples Through Deconstruction

Splits a tuple into new variables

You can use var for the deconstructing declaration

```
(var first, var middle, var last) = GetName(id);
```

You can even put the var outside the parentheses as shorthand

```
var(first, middle, last) = GetName(id);
```

You can deconstruct into existing variables

Demo



- Deconstruction