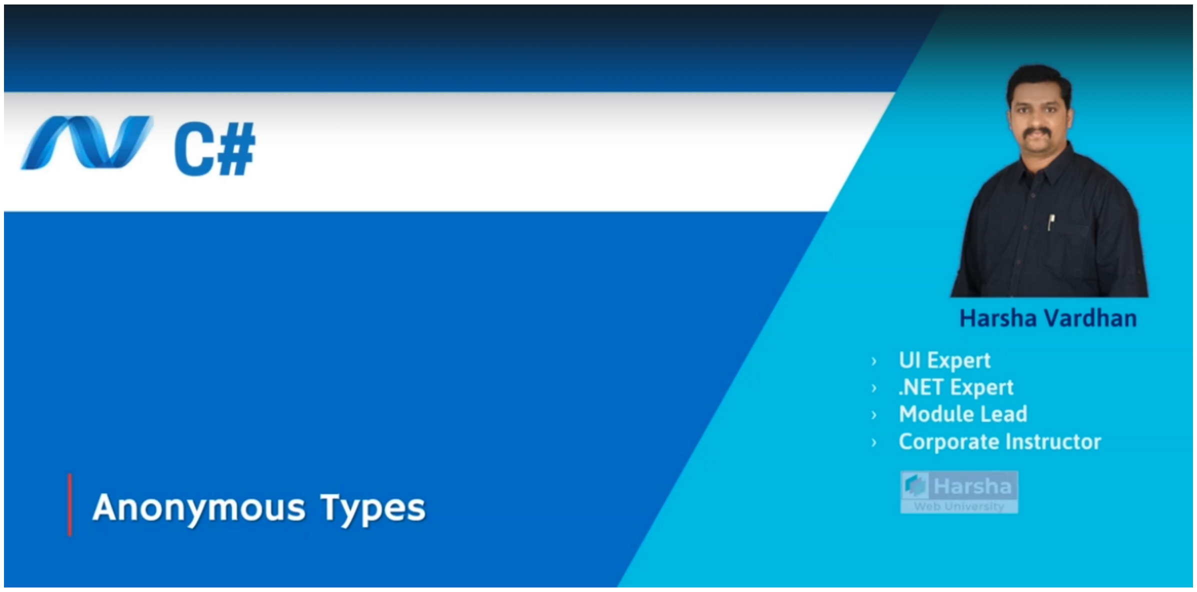
Section 23 : Anonymous Types

28 July 2024 16:35

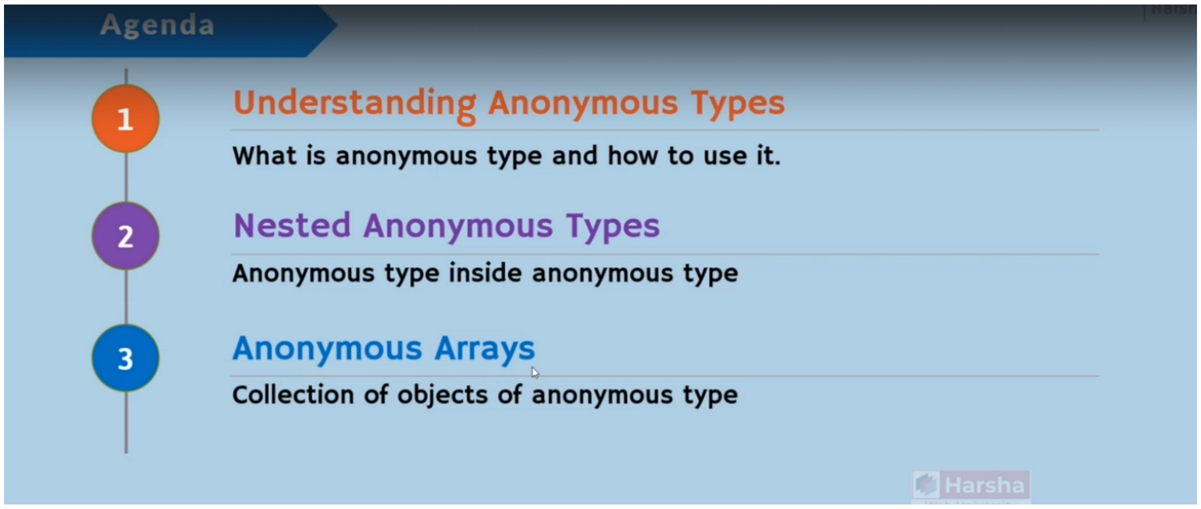


C#

Anonymous Types

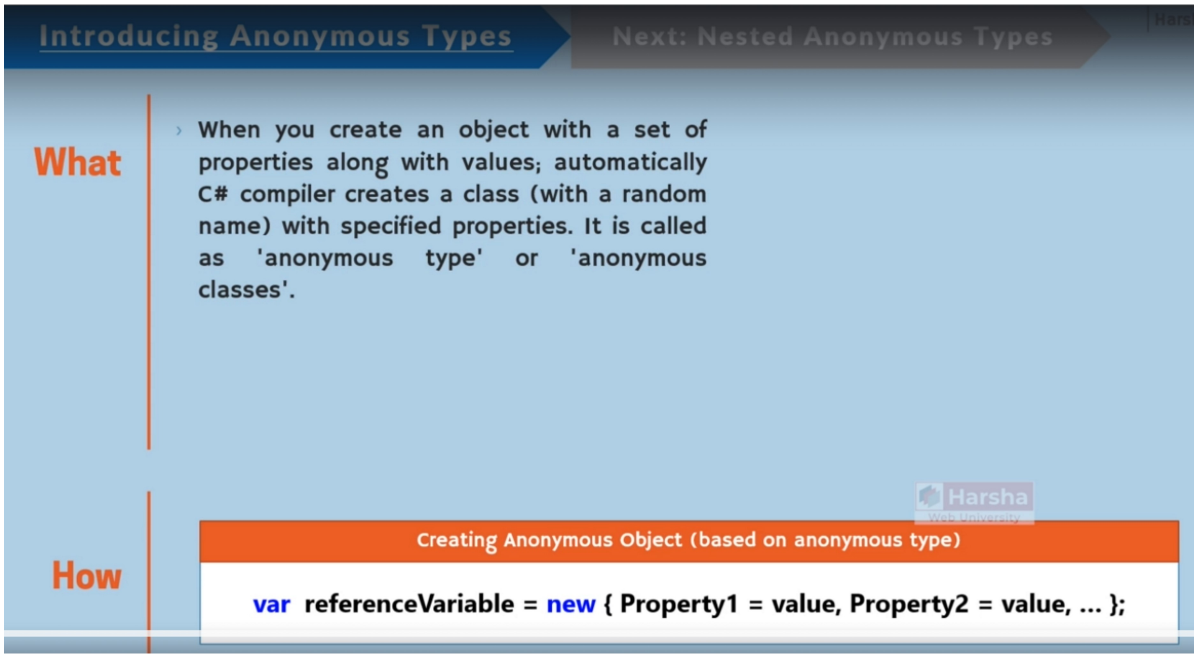
Harsha Vardhan

- UI Expert
- .NET Expert
- Module Lead
- Corporate Instructor



Agenda

1. **Understanding Anonymous Types**
   What is anonymous type and how to use it.
2. **Nested Anonymous Types**
   Anonymous type inside anonymous type
3. **Anonymous Arrays**
   Collection of objects of anonymous type



Introducing Anonymous Types

Next: Nested Anonymous Types

What

› When you create an object with a set of properties along with values; automatically C# compiler creates a class (with a random name) with specified properties. It is called as 'anonymous type' or 'anonymous classes'.

How

Creating Anonymous Object (based on anonymous type)

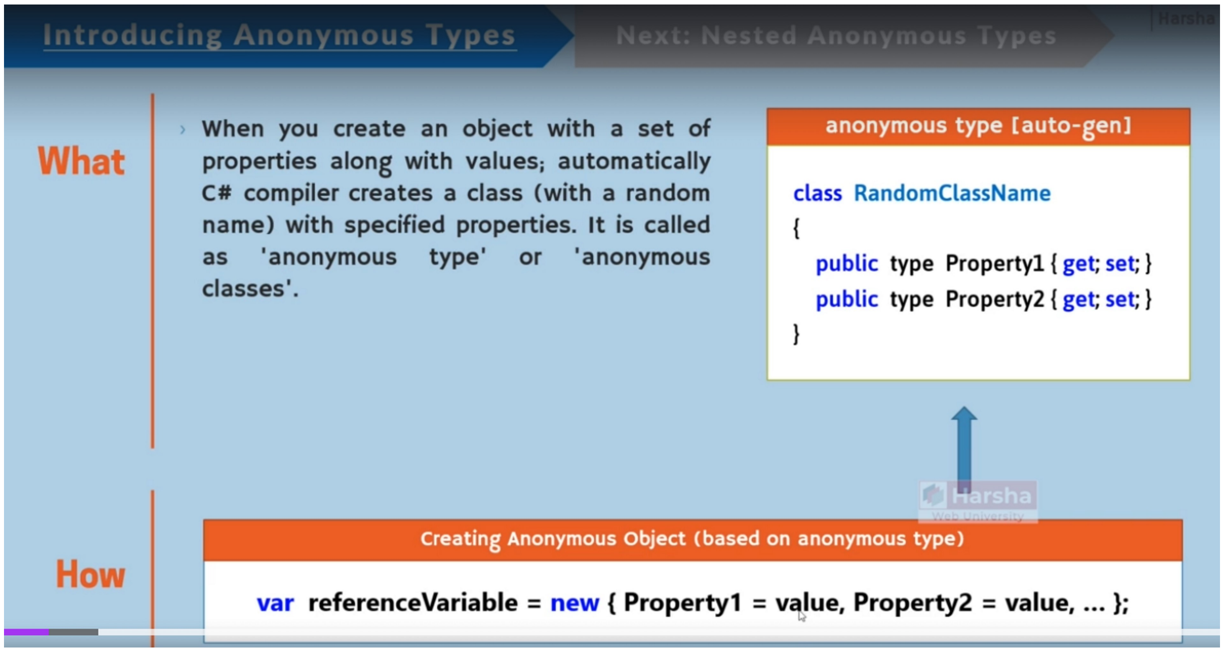
```
var referenceVariable = new { Property1 = value, Property2 = value, ... };
```

sometimes you want to quickly group a set of independent values such as strings and numbers together
but you don't have enough time or necessity to create a separate class with those properties

for example you want to store a string value and integer value as a group but instead of creating a separate class for
that with those properties you want to quickly group them as a single unit that means as an object for this kind of situations you can use the anonymous object

for example there is a database or any other data source you are retrieving some details about the particular person which includes the name and age of the person here name is the string data type and is the integer data type so you want to store them temporarily as a single object actually if you want to create an object.

you should create a separate class and it should contain two properties that is person name and person age and come back to the same place where you're writing the code then you should create the object for that specific class
and assign the values into the name and age properties of the newly created object so it's a lengthy process right?



Introducing Anonymous Types

Next: Nested Anonymous Types

What

› When you create an object with a set of properties along with values; automatically C# compiler creates a class (with a random name) with specified properties. It is called as 'anonymous type' or 'anonymous classes'.

anonymous type [auto-gen]

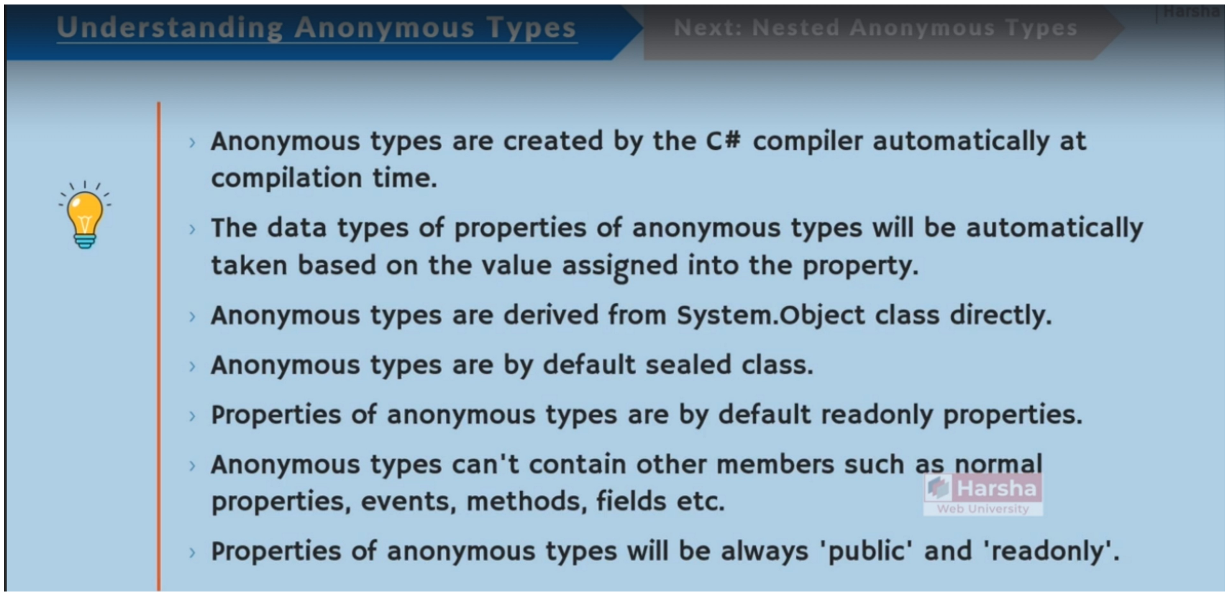
```
class RandomClassName
{
   public type Property1 { get; set; }
   public type Property2 { get; set; }
}
```

How

Creating Anonymous Object (based on anonymous type)

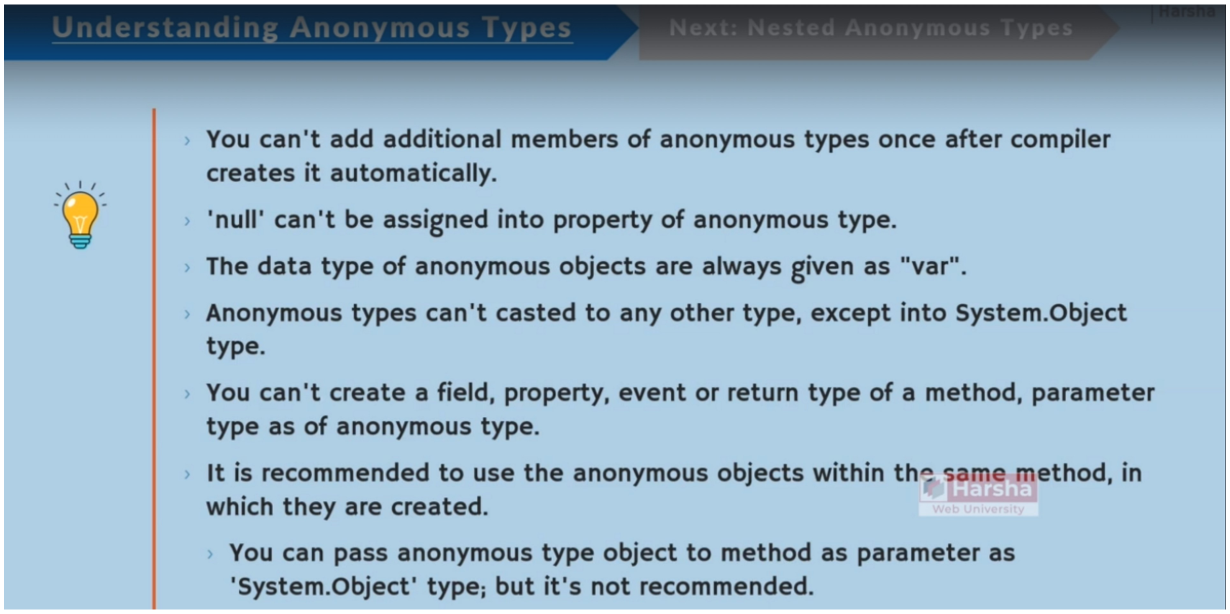
```
var referenceVariable = new { Property1 = value, Property2 = value, ... };
```

Alternatively whenever you want to represent same set of data in various methods then it is recommended to create a separate class manually with those specified properties
for example in various portions of the same project you want to represent person name and age as a group.



Understanding Anonymous Types

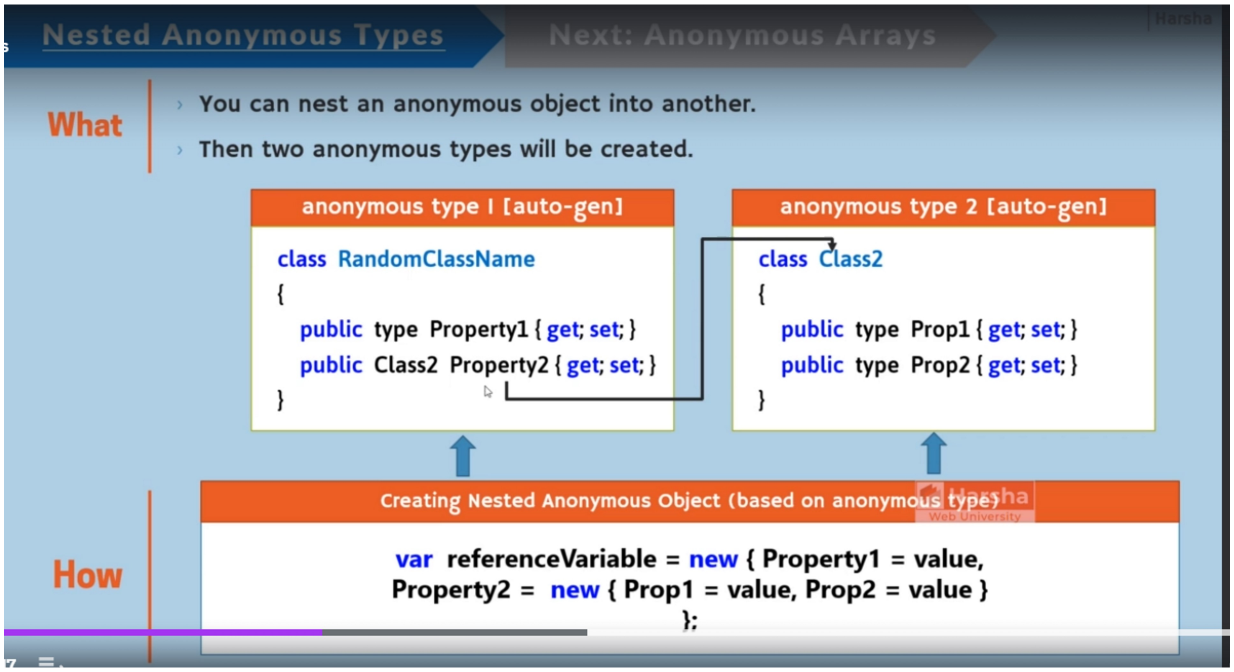
- Anonymous types are created by the C# compiler automatically at compilation time.
- The data types of properties of anonymous types will be automatically taken based on the value assigned into the property.
- Anonymous types are derived from System.Object class directly.
- Anonymous types are by default sealed class.
- Properties of anonymous types are by default readonly properties.
- Anonymous types can't contain other members such as normal properties, events, methods, fields etc.
- Properties of anonymous types will be always 'public' and 'readonly'.



Understanding Anonymous Types

- You can't add additional members of anonymous types once after compiler creates it automatically.
- 'null' can't be assigned into property of anonymous type.
- The data type of anonymous objects are always given as "var".
- Anonymous types can't casted to any other type, except into System.Object type.
- You can't create a field, property, event or return type of a method, parameter type as of anonymous type.
- It is recommended to use the anonymous objects within the same method, in which they are created.
  - You can pass anonymous type object to method as parameter as 'System.Object' type; but it's not recommended.

Nested Anonymous Types



Nested Anonymous Types

Next: Anonymous Arrays

What

- You can nest an anonymous object into another.
- Then two anonymous types will be created.

anonymous type 1 [auto-gen]

```
class RandomClassName
{
   public type Property1 { get; set; }
   public Class2 Property2 { get; set; }
}
```

anonymous type 2 [auto-gen]

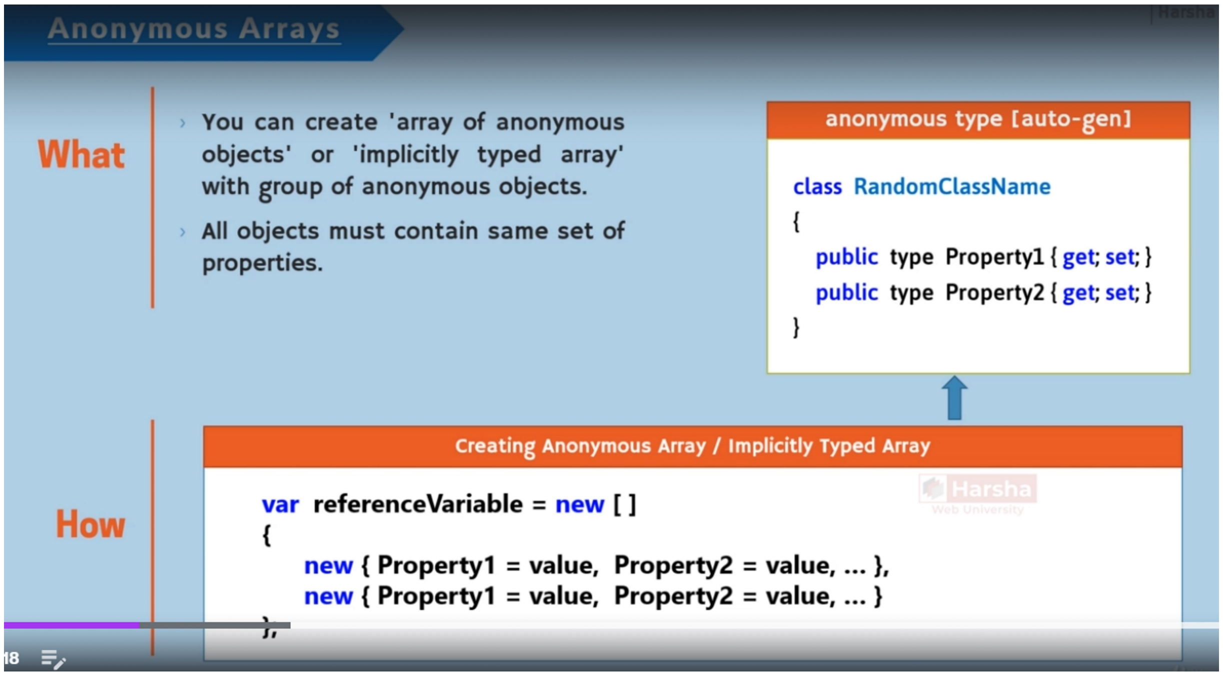
```
class Class2
{
   public type Prop1 { get; set; }
   public type Prop2 { get; set; }
}
```

How

Creating Nested Anonymous Object (based on anonymous type)

```
var referenceVariable = new { Property1 = value,
Property2 =  new { Prop1 = value, Prop2 = value }
};
```

Anonymous Arrays



Anonymous Arrays

What

- You can create 'array of anonymous objects' or 'implicitly typed array' with group of anonymous objects.
- All objects must contain same set of properties.

anonymous type [auto-gen]

```
class RandomClassName
{
   public type Property1 { get; set; }
   public type Property2 { get; set; }
}
```

How

Creating Anonymous Array / Implicitly Typed Array

```
var referenceVariable = new [ ]
{
    new { Property1 = value,  Property2 = value, ... },
    new { Property1 = value,  Property2 = value, ... }
};
```