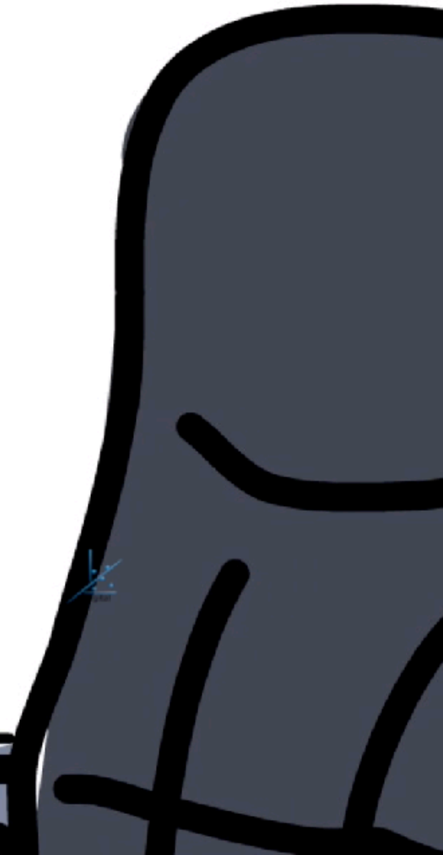
C# Structs Documentation

What is a struct?

Like a class a struct is a data type that can store multiple related items.
Also like a class a struct can implement properties, methods, events etc...

Unlike a class a struct is a value type.

A class is a reference type.



Default Values for C# Value Type Data Types

Value type	Default value
bool	FALSE
byte	0
char	'\0'
decimal	0M
double	0.0D
enum	The value produced by the expression `(E)0`, where `E` is the enum identifier.
float	0.0F
int	0
long	0L
sbyte	0
short	0
struct	The value produced by setting all value-type fields to their default values and all reference-type fields to `null`.
uint	0
ulong	0
ushort	0

Summary

- We have discussed that a struct is like a class in that it can store multiple related items and can also implement object behaviours through for e.g. methods.
- We have discussed that the fundamental differences between a class and a struct is that a class is a reference type and a struct is a value type and unlike a class an object can be instantiated from a struct without the use of the ‘new' keyword.
- We discussed that depending on the requirement it may be more efficient to use a struct instead of a class to store light weight objects.
- The differences in semantics regarding classes and structs were demonstrated.
- We discussed that unlike a class a struct cannot be the base for another class or struct and a struct cannot inherit from another struct or class.
- The fact that, like a class, a struct can implement interfaces was demonstrated.
- We proved through a demonstration that a struct is a value type and a class is a reference type.


gl
Digital