Erik Skogström
Squeed, Göteborg

C#

squeed

C# 9

- **Tyngdpunkt**

Förbättringar och förenklingar

Immutable

squeed

C# 9

- Tyngdpunkt
- **Hur installerar man det**

.NET 5

Visual Studio 16.7 eller senare

squeed

C# 9

- Tyngdpunkt
- Hur installerar man det
- **Features**

- Records
- Init only setters
- Top-level statements
- Pattern matching enhancements
- Target-typed new expressions

squeed

C# 9

- Tyngdpunkt
- Hur installerar man det
- **Features**
  - **Records**

- Immutable
- Ny/uppdaterade hjälpmetoder:
  - Värdebaserad jämförelse
  - *GetHashCode()*
  - *ToString()*
  - Kopiera och klona, *with { }*

squeed

C# 9

- Tyngdpunkt
- Hur installerar man det
- **Features**
  - **Records**

```
public record Vehicle
{
    public Vehicle(string color, string brand)
        => (Color, Brand) = (color, brand);

    public string Color { get; }
    public string Brand { get; }
}
```

```
public record Vehicle (string Color, string Brand);
```

squeed

C# 9

- Tyngdpunkt
- Hur installerar man det
- **Features**
  - **Records**

.ToString()

```
var car = new Car
{
    Brand = "Volvo",
    Color = "White"
};

car.ToString();

"Car { Color = White, Brand = Volvo }"
```

squeed

C# 9

Kopiera och klona

- Tyngdpunkt
- Hur installerar man det
- **Features**
  - **Records**

```
var car = new Car
{
    Brand = "Volvo",
    Color = "White"
};

var clone = car with { };

var blackCar = car with { Color = "Black" };
```

squeed

C# 9

- Tyngdpunkt
- Hur installerar man det
- **Features**
  - Records
  - **Init only setters**

```
public record Vehicle
{
    public string Color { get; init; }
    public string Brand { get; init; }
}

var car = new Vehicle
{
    Brand = "Volvo",
    Color = "White"
};

var blackCar = car with { Color = "Black" };
```

squeed

C# 9

- Tyngdpunkt
- Hur installerar man det
- **Features**
  - Records
  - Init only setters
  - **Top-level statements**

```
using System;

namespace TopLevelStatements
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

```
System.Console.WriteLine("Hello World!");
```

squeed

C# 9

- Tyngdpunkt
- Hur installerar man det
- **Features**
  - Records
  - Init only setters
  - Top-level statements
  - **Pattern matching enhancements**

- Typer
- Parenteser
- *and* - uttryck
- *or* - uttryck
- Förhållande, mindre än, större än osv.
- *not* - uttryck

```
public static bool IsLetterOrSeparator(this char c) =>
    c is (>= 'a' and <= 'z') or (>= 'A' and <= 'Z') or '.' or ',';

if (car is not null)
{
    // ...
}
```

squeed

C# 9

- Tyngdpunkt
- Hur installerar man det
- **Features**
  - Records
  - Init only setters
  - Top-level statements
  - Pattern matching enhancements
  - **Target-typed new expressions**

```
Car car = new Car();

var car = new Car();

Car car = new();

List<Car> cars = new()
{
    car,
    new()
};

Garage garage = new()
{
    Car = new()
};

var newGarage = garage with { Car = new() };
```

squeed

C# 8

- **Tyngdpunkt**

Förbättringar och förenklingar

Nullable reference types

C# 8

- Tyngdpunkt
- **Features**

- Default interface methods
- Pattern matching enhancements
  - Switch expressions
  - Property patterns
  - Tuple patterns
  - Positional patterns
- Using declarations
- Nullable reference types
- Null-coalescing assignment

squeed

C# 8

- Tyngdpunkt
- **Features**
  - **Default interface methods**

```
public interface IOrder
{
    decimal Cost { get; }
    decimal Discount { get; }

    public decimal GetDiscountPercentage()
        => Discount / Cost * 100;
}

public class Order : IOrder
{
    public decimal Cost { get; set; }
    public decimal Discount { get; set; }
}

private static void Main(string[] args)
{
    IOrder order = new Order {Cost = 100, Discount = 40};

    Console.Write(order.GetDiscountPercentage());
}
```

squeed

C# 8

- Tyngdpunkt
- **Features**
  - Default interface methods
  - **Pattern matching enhancements**
    - **Switch expressions**
    - **Property patterns**
    - **Tuple patterns**
    - **Positional patterns**

squeed

C# 8

- Tyngdpunkt
- **Features**
  - Default interface methods
  - **Pattern matching enhancements**
    - **Switch expressions**
    - Property patterns
    - Tuple patterns
    - Positional patterns

```
public static RGBColor FromRainbow(Rainbow colorBand) =>
    colorBand switch
    {
        Rainbow.Red    => new RGBColor(0xFF, 0x00, 0x00),
        Rainbow.Orange => new RGBColor(0xFF, 0x7F, 0x00),
        Rainbow.Yellow => new RGBColor(0xFF, 0xFF, 0x00),
        Rainbow.Green  => new RGBColor(0x00, 0xFF, 0x00),
        Rainbow.Blue   => new RGBColor(0x00, 0x00, 0xFF),
        Rainbow.Indigo => new RGBColor(0x4B, 0x00, 0x82),
        Rainbow.Violet => new RGBColor(0x94, 0x00, 0xD3),
        _              => throw new ArgumentException(message: "invalid enum value", paramName: nameof(colorBand)),
    };
```

```
public static RGBColor FromRainbowClassic(Rainbow colorBand)
{
    switch (colorBand)
    {
        case Rainbow.Red:
            return new RGBColor(0xFF, 0x00, 0x00);
        case Rainbow.Orange:
            return new RGBColor(0xFF, 0x7F, 0x00);
        case Rainbow.Yellow:
            return new RGBColor(0xFF, 0xFF, 0x00);
        case Rainbow.Green:
            return new RGBColor(0x00, 0xFF, 0x00);
        case Rainbow.Blue:
            return new RGBColor(0x00, 0x00, 0xFF);
        case Rainbow.Indigo:
            return new RGBColor(0x4B, 0x00, 0x82);
        case Rainbow.Violet:
            return new RGBColor(0x94, 0x00, 0xD3);
        default:
            throw new ArgumentException(message: "invalid enum value", paramName: nameof(colorBand));
    };
}
```

squad

C# 8

- Tyngdpunkt
- **Features**
  - Default interface methods
  - **Pattern matching enhancements**
    - Switch expressions
    - **Property patterns**
    - Tuple patterns
    - Positional patterns

```
public static decimal ComputeSalesTax(Address location, decimal salePrice) =>
    location switch
    {
        { State: "WA" } => salePrice * 0.06M,
        { State: "MN" } => salePrice * 0.075M,
        { State: "MI" } => salePrice * 0.05M,
        // other cases removed for brevity...
        _ => 0M
    };
```

C# 8

- Tyngdpunkt
- **Features**
  - Default interface methods
  - **Pattern matching enhancements**
    - Switch expressions
    - Property patterns
    - **Tuple patterns**
    - Positional patterns

```
public static string RockPaperScissors(string first, string second)
    => (first, second) switch
    {
        ("rock", "paper")     => "rock is covered by paper. Paper wins.",
        ("rock", "scissors")  => "rock breaks scissors. Rock wins.",
        ("paper", "rock")     => "paper covers rock. Paper wins.",
        ("paper", "scissors") => "paper is cut by scissors. Scissors wins.",
        ("scissors", "rock")  => "scissors is broken by rock. Rock wins.",
        ("scissors", "paper") => "scissors cuts paper. Scissors wins.",
        (_, _)                => "tie"
    };
```

squad

C# 8

- Tyngdpunkt
- **Features**
  - Default interface methods
  - **Pattern matching enhancements**
    - Switch expressions
    - Property patterns
    - Tuple patterns
    - **Positional patterns**

```
public class Point
{
    public int X { get; }
    public int Y { get; }

    public Point(int x, int y) => (X, Y) = (x, y);

    public void Deconstruct(out int x, out int y) =>
        (x, y) = (X, Y);
}
```

```
static Quadrant GetQuadrant(Point point) => point switch
{
    (0, 0)                              => Quadrant.Origin,
    var (x, y) when x > 0 && y > 0 => Quadrant.One,
    var (x, y) when x < 0 && y > 0 => Quadrant.Two,
    var (x, y) when x < 0 && y < 0 => Quadrant.Three,
    var (x, y) when x > 0 && y < 0 => Quadrant.Four,
    var (_, _)                          => Quadrant.OnBorder,
    _                                   => Quadrant.Unknown
};
```

squad

C# 8

- Tyngdpunkt
- **Features**
  - Default interface methods
  - Pattern matching enhancements
    - Switch expressions
    - Property patterns
    - Tuple patterns
    - Positional patterns
  - **Using declarations**

```
public void ReadStream()
{
    using (var file = new StreamReader("File.txt"))
    {
        //...
    } // file is disposed here
}
```

```
public void ReadStream()
{
    using var file = new StreamReader("File.txt");

    //...

    // file is disposed here
}
```

squeed

C# 8

- Tyngdpunkt
- **Features**
  - Default interface methods
  - Pattern matching enhancements
    - Switch expressions
    - Property patterns
    - Tuple patterns
    - Positional patterns
  - Using declarations
  - **Nullable reference types**

```
public class Car
{
    public string? Brand { get; set; }
    public string Color { get; set; }
}

car.Brand.Length;

car.Color.Length;

car.Brand!.Length;
```

squeed

C# 8

- Tyngdpunkt
- **Features**
  - Default interface methods
  - Pattern matching enhancements
    - Switch expressions
    - Property patterns
    - Tuple patterns
    - Positional patterns
  - Using declarations
  - **Nullable reference types**

```
#nullable enable
```

```
<Nullable>enable</Nullable>
```

squeed

C# 8

- Tyngdpunkt
- **Features**
  - Default interface methods
  - Pattern matching enhancements
    - Switch expressions
    - Property patterns
    - Tuple patterns
    - Positional patterns
  - Using declarations
  - Nullable reference types
  - **Null-coalescing assignment**

```
List<int> numbers = null;
int? i = null;

numbers ??= new List<int>();
numbers.Add(i ??= 17);
numbers.Add(i ??= 20);

Console.WriteLine(string.Join(" ", numbers)); // output: 17 17
Console.WriteLine(i);                         // output: 17
```

squeed

C# 7.0 - 7.3

- **Tyngdpunkt**

Förbättringar och förenklingar

Pattern Matching

C# 7.0 - 7.3

- Tyngdpunkt
- **Features**
  - out variables
  - private protected access modifier

C# 7.0 - 7.3

- Tyngdpunkt
- **Features**
  - **out variables**

```
if (int.TryParse(input, out var answer))
    Console.WriteLine(answer);
else
    Console.WriteLine("Could not parse input");
```

squeed

C# 7.0 - 7.3

- Tyngdpunkt
- **Features**
  - out variables
  - **private protected access modifier**

```
public class BaseClass
{
    private protected int MyValue = 0;
}

public class DerivedClass1 : BaseClass
{
    public void Access()
    {
        var baseObject = new BaseClass();

        // Error CS1540, because myValue can only be accessed by
        // classes derived from BaseClass.
        baseObject.MyValue = 5;

        // OK, accessed through the current derived class instance
        MyValue = 5;
    }
}
```

squeed

- C# 9.0 - https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-9
- C# 8.0 - https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-8
- C# 7.0-7.3 - https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-7
- Github C# - https://github.com/dotnet/csharplang/tree/master/proposals

FRÅGOR?

squeed

TACK

erikskogstrom/examples

squeed