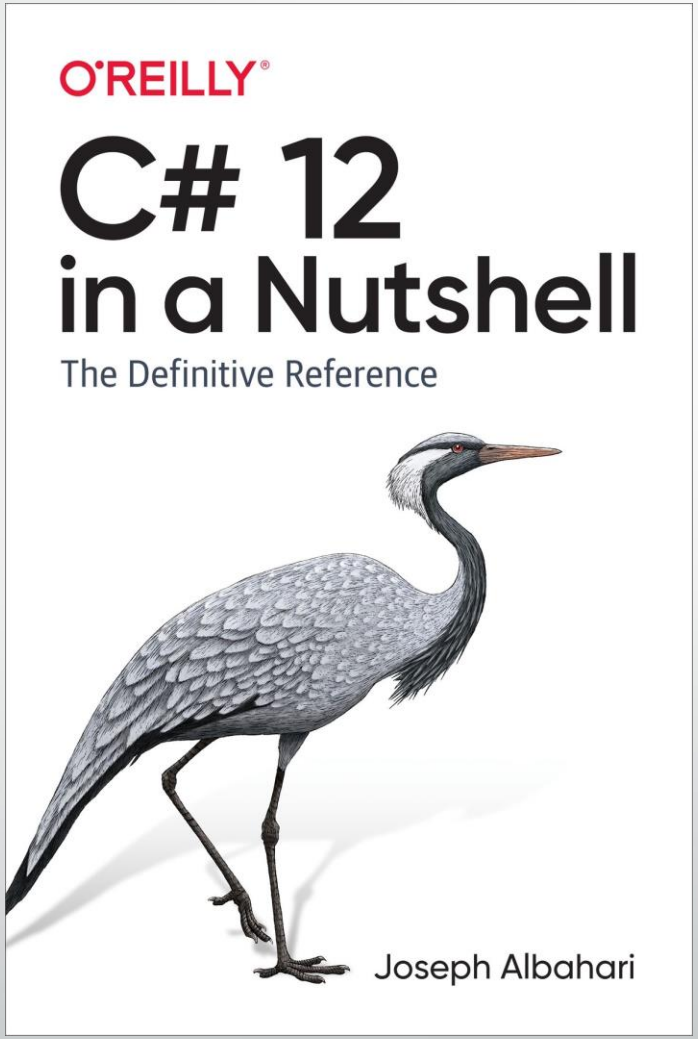
فصل اول – بخش سوم



- What's New in C# 12
- What's New in C# 11
- What's New in C# 10
- What's New in C# 9
- What's New in C# 8
- What's New in C# 7
- What's New in C# 6
- What's New in C# 5
- What's New in C# 4
- What's New in C# 3
- What's New in C# 2

What's New in C# 12

Collection expressions

```
// before C# 12
int[] arrayBefore = { 1, 2, 3 };

// C# 12
int[] arrayInCsharp12 = [1, 2, 3];
```

قبلا برای مقداردهی اولیه آرایه باید از آکولاد {} استفاده میکردیم

ولی الان میتونیم از براکت [] استفاده کنیم.

حالا این چه فایده ای داره؟

```
List<char> list         = ['a','e','i','o','u'];
HashSet<char> set       = ['a','e','i','o','u'];
ReadOnlySpan<char> span = ['a','e','i','o','u'];
```

اول اینکه از نظر سینتکس، شبیه بقیه کالکشن ها میشه.

دوم اینکه میتونید به صورت مستقیم از این ساختار برای وروی توابع استفاده کنید.

```
2 references | - changes | -authors, -changes
int getLength(int[] numbers)
{
    return numbers.Length;
}

getLength({ 1, 2, 3 });
getLength([1, 2, 3]);
```

What’s New in C# 12

Primary constructors in classes and structs

این ویژگی خیلی شبیه تعریف record هست که در c# 9 معرفی شد، با این ویژگی شما میتونید بلافاصله بعد از تعریف class یا struct بهش یه سری پارامتر ورودی بدید. (تفاوت این ویژگی با record این است که پارامترها به صورت پیش فرض تعریف نمیشوند و برای دسترسی به آنها باید صراحتا در class یا struct تعریف شوند)

```
1 reference | 0 changes | 0 authors, 0 changes
internal class ClassWithPrimaryConstructor(string firstName, string lastName)
{
    0 references | 0 changes | 0 authors, 0 changes
    public string FirstName { get; set; } = firstName;
    0 references | 0 changes | 0 authors, 0 changes
    public string LastName { get; set; } = lastName;
    1 reference | 0 changes | 0 authors, 0 changes
    public string SayHello() => $"Hello {firstName} {lastName}";
}

1 reference | 0 changes | 0 authors, 0 changes
internal struct StructWithPrimaryConstructor(string firstName, string lastName)
{
    0 references | 0 changes | 0 authors, 0 changes
    public string FirstName { get; set; } = firstName;
    0 references | 0 changes | 0 authors, 0 changes
    public string LastName { get; set; } = lastName;
    1 reference | 0 changes | 0 authors, 0 changes
    public readonly string SayHello() => $"Hello {firstName} {lastName}";
}
```

```
Console.WriteLine(new ClassWithPrimaryConstructor("C#", "Class").SayHello());
Console.WriteLine(new StructWithPrimaryConstructor("C#", "Struct").SayHello());
```

What’s New in C# 12

Default lambda parameters

قبلا فقط برای توابع که با فرمت معمولی تعریف میکردیم میتونستیم مقدار پیش فرض بدیم، ولی الان برای arrow function ها هم میتونیم برای مقدار پارامترهای ورودی پیش فرض تعیین کنیم.

```
1 reference | - changes | -authors, -changes
string SayHelloWithoutLambda(string name = "")
{
    return $"Hi {name}".Trim();
}

var SayHelloWithLambda = (string name = "") =>
{
    return $"Hi {name}".Trim();
};

Console.WriteLine(SayHelloWithoutLambda());
Console.WriteLine(SayHelloWithLambda());
```

What’s New in C# 12

Alias any type

C# همیشه اجازه میداد که برای نوع های ساده یا جنریک ها alias تعریف کرد، توی C# 12 نوع های جدیدی مثل tuple و arrays رو هم پشتیبانی میکنه.

```
//enable before c# 12
using ListOfInt = System.Collections.Generic.List<int>;

//enable c# 12
using NumberList = double[];
using Point = (int X, int Y);
```

```
var list = new ListOfInt();
NumberList numbers = { 2.5, 3.5 };
Point p = (3, 4);
```

What’s New in C# 12
inline arrays

یک atribute جدید توسط c# 12 معرفی شده که تا جایی که مطالعه کردم روی performance خیلی تاثیر داره ولی حقیقتش خیلی ازش سر در نیاوردم، میذارمش اینجا لطفا شما کمک کنید تا باهم پی به رازش ببریم.

```
[System.Runtime.CompilerServices.InlineArray(4)]
5 references | 0 changes | 0 authors, 0 changes
struct Point
{
    4 references | 0 changes | 0 authors, 0 changes
    public int X { get; set; }
    4 references | 0 changes | 0 authors, 0 changes
    public int Y { get; set; }
}
```

```
Point p1 = new Point() { X = 1, Y = 2 };
Point p2 = new Point() { X = 3, Y = 4 };

double distance = Math.Sqrt(Math.Pow(p1.X - p2.X, 2) + Math.Pow(p1.Y - p2.Y, 2));
```

What’s New in C# 11

Raw string literals

با استفاده از سه عدد یا بیشتر کاراکتر " میتوانید یک raw string ایجاد کنید. در این ساختار تقریبا همه کاراکترها رو میتونید استفاده کنید. با این ساختار راحت تر میتونید عبارات json یا xml رو تایپ کنید.

اگر در ابتدای این ساختار از کاراکتر $ استفاده کنید میتونید رشته مدنظرتونو در چند خط بنویسید.

اگر از دو عدد یا بیشتر کاراکتر $ استفاده کنید میتونید در رشته ای که مینویسید به راحتی از کاراکتر } و { استفاده کنید.

```
// use three or more quote characters create a raw string
string raw = """<file path="c:\temp\test.txt"></file>""";
Console.WriteLine(raw);

// use $ with raw string you can write multi line string
string multiLineRaw = $"""
    Line 1
    Line 2
    The date and time is {DateTime.Now}
    """;
Console.WriteLine(multiLineRaw);

// use two or more $ in a raw string, allowing you to include braces in the string itself:
Console.WriteLine($$"""{ "TimeStamp": "{{DateTime.Now}}" }""");
// Output: { "TimeStamp": "01/01/2024 12:13:25 PM" }
```

What’s New in C# 11
UTF-8 strings

با استفاده از پسوند utf8 میتونید یه رشته رو encode کنید، توی این کد کاراکتر→ سه بایت فضا اشغال میکنه وقتی encode میشه به utf8. همینطور فرمت خروجی ReadOnlySpan<byte> خواهد بود.

```
var utf8 = "ab→cd"u8;  // Arrow symbol consumes 3 bytes
var str = "ab→cd";  // Arrow symbol consumes 3 bytes
Console.WriteLine("UTF8: " + utf8.Length);      // 7
Console.WriteLine("String: " + str.Length);      // 5
```

یکی از کاربردهای این ویژگی در serialize و deserialize کردن json هست و performance بالایی داره.

What’s New in C# 11

List patterns

این ویژگی بهمون کمک میکنه تا بتونیم یه آرایه یا لیست رو با یه آرایه یا لیست دیگه مقایسه کنیم وببینیم شباهت مدنظر ما رو دارند یا نه،

علامت _ یعنی مقدار اون خونه مهم نیست هرچی باشه.

علامت .. بین دو مقدار یعنی بین اون دو آیتم صفر عدد یا بیشتر آیتم وجود داره که مقادیرش اهمیتی نداره. این علامت میتونه ابتدا، وسط و یا انتهای الگو قرار بگیره ولی فقط یک بار میشه ازش در الگو استفاده کرد.

```
int[] arryAges = { 1, 2, 3, 4, 5 };
Console.WriteLine($"List patterns (array)");

Console.WriteLine(arryAges is [1, 2, 3, 4]);   // False
Console.WriteLine(arryAges is [1, 2, 3, 4, 5]);   // True
Console.WriteLine(arryAges is [1, .., 5]);   // True
Console.WriteLine(arryAges is [.., 5]);   // True
Console.WriteLine(arryAges is [1, ..]);   // True
Console.WriteLine(arryAges is [_, 2, .., 5]);   // True
Console.WriteLine(arryAges is [_, _, 3, .., 5]);   // True
```

What's New in C# 11
Required members

این ویژگی رو خودم خیلی باهاش حال کردم و به نظرم کاربردیه، اینجوریه که با استفاده از کلمه کلیدی required قبل از یک پراپرتی در یک کلاس میتونید اونو اجباریش کنید و بدین شکل اجازه ساختن instance از اون کلاس بدون پر کردن اون فیلد خاص رو به کاربر نمیدید. به نظرم میتونه در نوشتن کد تمیز کمک کنه بهمون.

```
1 reference | 0 changes | 0 authors, 0 changes
internal class SampleRequiredMembers
{
    1 reference | 0 changes | 0 authors, 0 changes
    public required string FirstName { get; set; }
    1 reference | 0 changes | 0 authors, 0 changes
    public required string LastName { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public int Age { get; set; }
}
```

```
var obj1 = new SampleRequiredMembers();
```

SampleRequiredMembers.SampleRequiredMembers()

CS9035: Required member 'SampleRequiredMembers.FirstName' must be set in the object initializer or attribute constructor.

CS9035: Required member 'SampleRequiredMembers.LastName' must be set in the object initializer or attribute constructor.

```
var obj2 = new SampleRequiredMembers { FirstName = "", LastName = "" };
```

What's New in C# 11

Static virtual/abstract interface members

این قابلیت به شما اجازه میده که تو interface اعضای abstract یا virtual رو به صورت static تعریف کنید.

```
1 reference | 0 changes | 0 authors, 0 changes
public interface IParsable
{
    2 references | 0 changes | 0 authors, 0 changes
    static abstract string Parse(string s);
}

2 references | 0 changes | 0 authors, 0 changes
public class ParseString : IParsable
{
    2 references | 0 changes | 0 authors, 0 changes
    public static string Parse(string s)
    {
        return $"{string.Join(",", s.ToArray())}";
    }
}
```

```
Console.WriteLine(ParseString.Parse("Test Parse String"));
```

What’s New in C# 11

Generic math

اینترفیس System.Numerics.INumber<TSelf> با هدف یکپارچه سازی نوع های از جنس عدد ارائه شده و اجازه میده تا بتونید متدهای جنریک به شکل زیر بنویسید:

```
3 references | - changes | -authors, -changes
T Sum<T>(T[] numbers) where T : INumber<T>
{
    T total = T.Zero;
    foreach (T n in numbers)
        total += n;      // Invokes addition operator for any numeric type
    return total;
}

int intSum = Sum([3, 5, 7]);
double doubleSum = Sum([3.2, 5.3, 7.1]);
decimal decimalSum = Sum([3.2m, 5.3m, 7.1m]);
```

What’s New in C# 11
file accessibility modifier

با استفاده از file modifier میتونید کلاسی رو تعریف کنید که فقط در همان فایلی که در آن تعریف شده به آن دسترسی دارید.

```
// In File1.cs:
file interface IWidget
{
    int ProvideAnswer();
}

file class HiddenWidget
{
    public int Work() => 42;
}

public class Widget : IWidget
{
    public int ProvideAnswer()
    {
        var worker = new HiddenWidget();
        return worker.Work();
    }
}
```

```
// In File2.cs:
// Doesn't conflict with HiddenWidget
// declared in File1.cs
public class HiddenWidget
{
    public void RunTask()
    {
        // omitted
    }
}
```

What’s New in C# 10

File-scoped namespaces and The global using directive

معمولا در هر فایل c# یک namesapce استفاده میشه. با استفاده از این قابلیت فرورفتگی های کد رو میتونید کاهش بدید.

```
namespace CSharp12Nutshell.WhatsNew;

0 references | 0 changes | 0 authors, 0 changes
internal class CSharp10
{
}
```

```
global using System;
global using System.Collections.Generic;
global using System.Linq;
global using System.Text;
global using System.Threading.Tasks;
```

با استفاده از کلمه کلیدی global قبل از using میتونید از تکرار using ها در فایلهای مختلف جلوگیری کنید.

Nondestructive mutation for anonymous types:

استفاده از کلمه کلیدی with برای anonymous type ها فراهم شده

```
var a1 = new { A = 1, B = 2, C = 3, D = 4, E = 5 };
var a2 = a1 with { E = 10 };
Console.WriteLine(a2);      // { A = 1, B = 2, C = 3, D = 4, E = 10 }
```

What’s New in C# 10

New deconstruction syntax and Record structs

هنگام کار کردن با tupleها میتونید به این روش هم مقدار یک متغیر رو تغییر بدید یا یک متغیر جدید بسازید.

```
var point = (3, 4);
double x = 0;
(x, double y) = point;
Console.WriteLine($"X={x}, Y={y}");
```

همچنین میتونید رکورد از نوع struct تعریف کنید.

```
0 references | - changes | -authors, -changes
record struct StructRecord(int X, int Y);
```

What's New in C# 10

Lambda expression enhancements

چهار قابلیت برای استفاده از عبارات لامبدا اضافه شده، توضیحات بیشتر در تصویر:

```
// 1. implicit typing (var) is permitted:
var greeter = () => "Hello, world";
var square = (int x) => x * x;

// 2. lambda expression can specify a return type
var sqr = int (int x) => x;

// 3.  you can pass a lambda expression into a method parameter of type object, Delegate, or Expression
M1(() => "test");   // Implicitly typed to Func<string>
M2(() => "test");   // Implicitly typed to Func<string>
M3(() => "test");   // Implicitly typed to Expression<Func<string>>

1 reference | - changes | -authors, -changes
void M1(object x) { }
1 reference | - changes | -authors, -changes
void M2(Delegate x) { }
1 reference | - changes | -authors, -changes
void M3(Expression x) { }

// 4. apply attributes to a lambda expression's compile-generated target method
var a = [Description("test")] () => { };
```

What's New in C# 10

Nested property patterns

این قابلیت رو نمیدونستم و به نظرم خیلی جذاب اومد، با استفاده از property pattern میتونید چک کنید که آبجکت مدنظرتون ویژگی هایی که میخواهید رو داره یا نه.

```
var obj = new Uri("https://www.linqpad.net");
Console.WriteLine(obj is Uri { Scheme.Length: 5 }); //true
Console.WriteLine(obj is Uri { Scheme: { Length: 5 } }); //true

var nestedPropertyPattern = new SampleNestedPropertyPattern
{
    Id = 1,
    Name = "Mohammad",
    Age = 34,
    City = "Tehran",
    Address = "Andisheh"
};

Console.WriteLine(nestedPropertyPattern is SampleNestedPropertyPattern { Name: "Mohammad" }); //true
Console.WriteLine(nestedPropertyPattern is SampleNestedPropertyPattern { Age: 20 }); //false
```

What’s New in C# 10

CallerArgumentExpression

با استفاده از این attribue میتونید به مقداری که در ورودی متد آمده است به صورت یک رشته دسترسی داشته باشید.

```
Print($"{DateTime.Now} - test");

1 reference | - changes | -authors, -changes
void Print(string name, [CallerArgumentExpression(nameof(name))] string expr = null) => Console.WriteLine(expr);

// Output: $"{DateTime.Now} - test"
```