

C# 8.0 Features

Или рассказ о том, что уже давно надо было добавить в C#

Default in deconstruction

```
(string s, int i, bool b, char c, byte by) = (default, default, default, default,  
                                              default);
```

```
(string s, int i, bool b, char c, byte by) = default;
```

Generic attributes

```
class GenericAttribute<T> : Attribute  
{  
}
```

Caller expression attribute

```
public static class Debug
{
    public static void Assert(bool condition,
[CallerArgumentExpression(nameof(condition))] string message = null)
    { }

    static void Test()
    {
        var array = new int[1];
        Debug.Assert(array.Length == 1);
        Debug.Assert(array.Length == 1, "array.Length == 1");
    }
}
```

Relax ordering of ref and partials modifiers

```
public ref partial class RefPartial { }
```

```
public partial ref class PartialRef { }
```

Mass killing feature C# 8.0

- Помните ли вы порядок следования собачки и доллара?

```
static void InterpolatedVerbatimString()  
{  
    var stringA = @$"True verbatim interpolated string";  
    var stringB = $"Another true verbatim interpolated string";  
}
```

- Свободный порядок объявления interpolated verbatim string

Duck Disposable

```
using (Duck wd = new Duck())  
{  
}
```

```
class Duck  
{  
    public void Dispose()  
    {  
    }  
}
```



The best Duck Disposable case



The best Duck Disposable case

Code C# Create Gist

C# 8.0: Enhanced using (9 Nov 2017)

Results Run

```
using System;

public class Foo
{
    public int Prop { get; set; } return: 42
}

public static class FooExtension
{
    public static void Dispose(this Foo foo) foo: Foo
    {
        Console.WriteLine("Calling dispose for Foo instance");
        Console.WriteLine($"The answer to life the universe and everything = {foo.Prop}");
    }
}

class Bar
{
    static void Main()
    {
        using (var foo = new Foo()) foo: Foo
        {
            foo.Prop = 42;
        }
    }
}
```

This is a new feature — might be unstable/too strict. Please [report](#) any issues.
Calling dispose for Foo instance
The answer to life the universe and everything = 42

Питониза(д)ция

```
string source = "Python? In my c-sharp? No way!";
```

- 1) `string text1 = source[4..8];` `=on? I`
- 2) `string text1_1 = source.AsSpan(4, 8).ToString();` `=on? In my`
- 3) `string text2 = source[^4..^1];` `= way`
- 4) `string text3 = source[^4..^0];` `= way!`
- 5) `string text4 = source[..];` `=Python? In my c-sharp? No way!`

И треснул мир напополам



Типы, которые мы потеряли

```
var list = new List<MyCustomType>()  
{  
    new MyCustomType ("John", "Skeet"),  
    new MyCustomType ("Elizaveta", "Nikita"),  
};
```

```
List<MyCustomType> list = new ()  
{  
    new ("Дмитрий Н.", "Хельсинки"),  
    new ("Райф", «Микросервисы»)  
};
```

Типы, которые мы потеряли

```
var customType = new ();
```



- ❌ CS8124 Кортеж должен содержать по меньшей мере два элемента.
- ❌ CS1526 В выражении `new` после типа требуется `()`, `[]` или `{}`.
- ❌ CS8181 `"new"` невозможно использовать с типом кортежа. Вместо этого используйте литеральное выражение кортежа.

Switch expression

```
static string TestEnumOld(Ценность value)
{
    switch (value)
    {
        case Ценность.Православие:
            return "1";
        case Ценность.Самодержавие:
            return "2";
        case Ценность.Народность:
            return "3";
        default:
            return "-1";
    }
}
```

```
static string TestEnumNew(Ценность value ) =>
    value switch
    {
        Ценность.Либерализм => "1",
        Ценность.Демократия => "2",
        Ценность.Толерантность => "3"
    };
```

Recursive patterns

```
static string RecursivePattern(Shape shape) =>
    shape switch
    {
        CombinedShape (var shape1, var (pos, _)) =>
            $"Combined shape - shape1: {shape1.Name}, pos of shape2: {pos}",
        { Size: (200, 200), Position: var pos } =>
            $"Shape with size 200x200 at position {pos.X}:{pos.Y}",
        Ellipse (var pos, var size) =>
            $"Elliplse with size {size} as position {pos}",
        Rectangle (_, var size) =>
            $"Rectangle with size {size}",
        _ => "Another shape"
    };
```

Обратимся к истокам

```
Action<int, int, int, int, int, int, int, int, int, int, int, int, int, int, int, int> dummyDelegate =  
  
delegate (int x1, int x2, int x3, int x4, int x5, int x6, int x7, int x8, int x9,  
int x10, int x11, int x12, int x13, int x14, int x15, int x16)  
  
{  
  
    Console.WriteLine(  
  
        "Делегат настолько же большой, как и спецификация к новым фичам C# 8.0");  
  
};
```

IEnumerable

```
public interface IEnumerable<out T>
{
    IEnumerator<T> GetEnumerator();
}
```

```
public interface IEnumerator<out T>
{
    bool MoveNext();
    T Current { get; }
    void Reset();
}
```


IObservable

```
public interface IObservable<out T>
{
    IDisposable Subscribe(IObserver<T> observer);
}
```

```
public interface IObserver<in T>
{
    void OnNext(T value);
    void OnError(Exception error);
    void OnCompleted();
}
```

async/await

```
public static async Task<JObject> GetJsonAsync(Uri uri)
{
    using (var client = new HttpClient())
    {
        var jsonString = await client.GetStringAsync(uri);
        return JObject.Parse(jsonString);
    }
}
```

```
public void Button1_Click(...)
{
    var jsonTask = GetJsonAsync(...);
    textBox1.Text = jsonTask.Result;
}
```

Async streams

```
await foreach (var i in MyIterator())  
{ }
```

```
static async IEnumerable<int> MyIterator()  
{  
    for (int i = 0; i < 100; i++)  
    {  
        await Task.Delay(1000);  
        yield return i;  
    }  
}
```

Dispose, которого мы так *долго* ждали

```
await using (var connection = new VeryLongHttpConnection())  
{ }
```



Default interface methods

```
interface Logger
{
    public void Log(string message) => Console.WriteLine(message);

    void Log(Exception ex) => Log(ex.Message);
}
```



Interface VS abstract class

- Так чем же абстрактный класс отличается от интерфейса?
- Не хранит состояние.
- Описывает только сигнатуры методов/свойств.
- Не имеет модификаторов у методов/свойств.

Interface VS abstract class

```
interface IDefault
{
    void TestMethod() => Console.WriteLine(SomeProperty++);
    static ConditionalWeakTable<IDefault, object> MyWeakTable = new
ConditionalWeakTable<IDefault, object>();
    public int SomeProperty
    {
        get { return (MyWeakTable.TryGetValue(this, out var sp)) ? (int)sp : 0; }
        set
        {
            MyWeakTable.Remove(this);
            MyWeakTable.Add(this, value);
        }
    }
}
```



Возвращение миллиарда

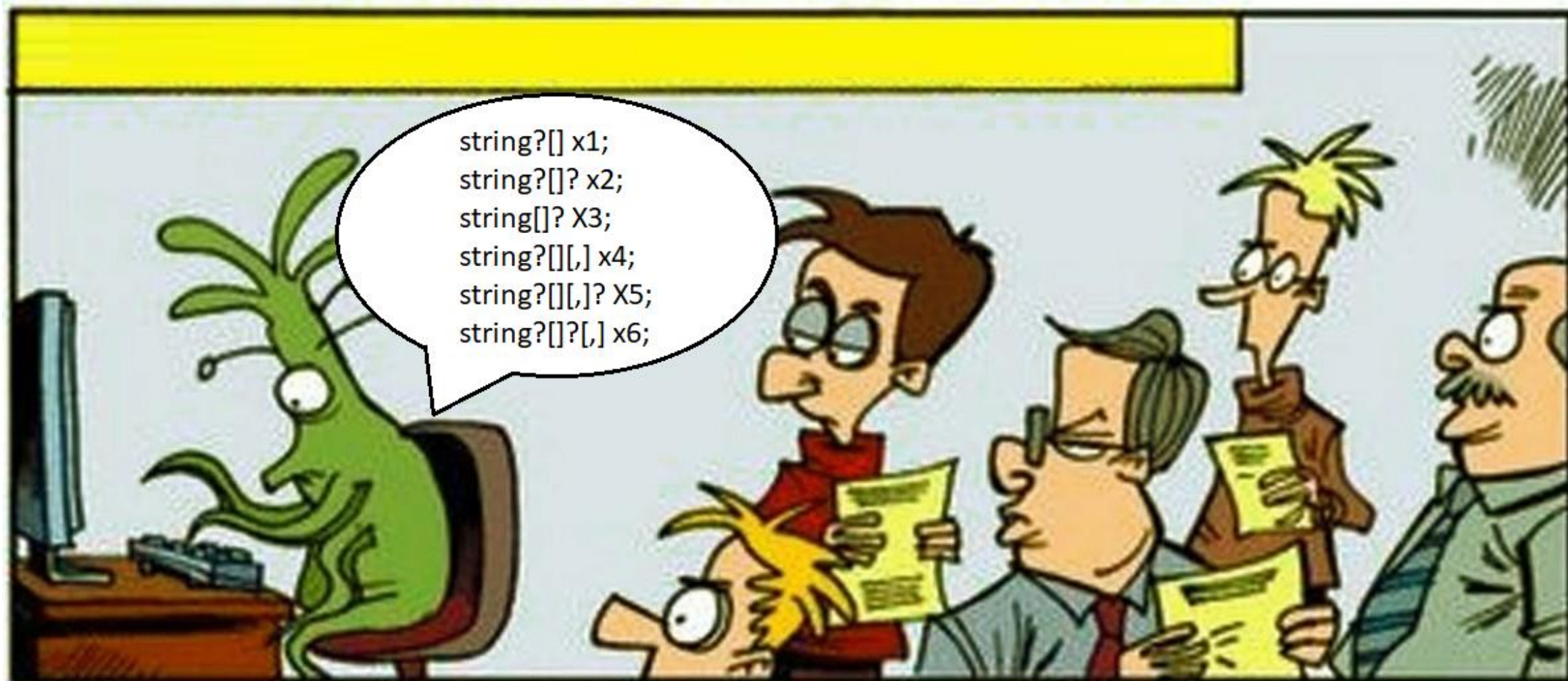


Страшная правда про ссылочные типы

- Nullable `string` теперь - типа `string`?
- Not nullable `string` – конкретный `string`!
- Что же такое обычный `string` – I don't know `string` ?

Учимся говорить правильно и четко

- `string?[] x1;` // Конкретный массив из типа строк
- `string?[]? x2;` // Типа массив из типа строк
- `string[]? x3;` // Типа массив из конкретных строк
- `string?[][,] x4;` // Конкретный двумерный массив конкретных одномерных массивов типа строк
- `string?[][,]? x5;` // Типа двумерный массив конкретных одномерных массивов типа строк
- `string?[]?[,] x6;` // Конкретный двумерный массив типа одномерных массивов типа строк



Вызываем товарища майора

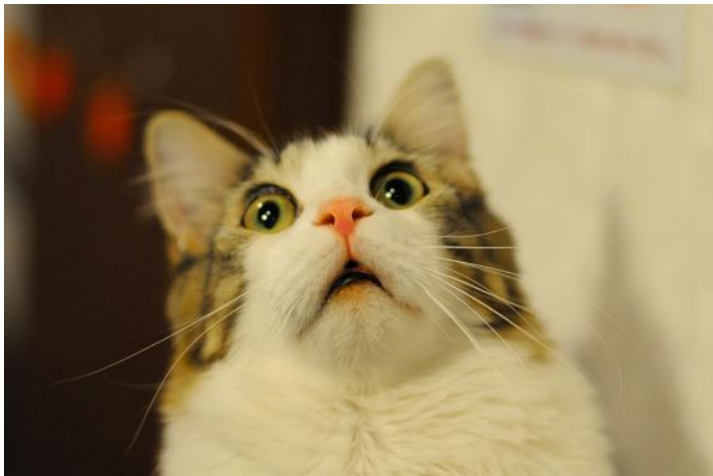
`<TreatWarningsAsErrors>True</TreatWarningsAsErrors>`

```
class Person
{
    public string FirstName { get; set; }
    public string MiddleName { get; set; }
    public string LastName { get; set; }
    public Person(string first, string last)
    {
        FirstName = first;
        LastName = last;
    }
}
```

⊗ `Person.Person(string first, string last)`
Non-nullable property 'MiddleName' is uninitialized.
[Show potential fixes](#) (Alt+Enter or Ctrl+.)

ШОК! Сенсация! 18+

- Fody.NullGuard остался без работы
- Уволен с волчьим билетом как сын предателя open-source



Бдительный оператор

```
string? x = null;  
if (x == null)  
    x = "Hello there";  
if (x == null)  
    x = "General Kenobi";
```

```
string? x = null;  
x ??= "Hello there";  
x ??= "General Kenobi";
```

Оператор «Мамой клянусь!»

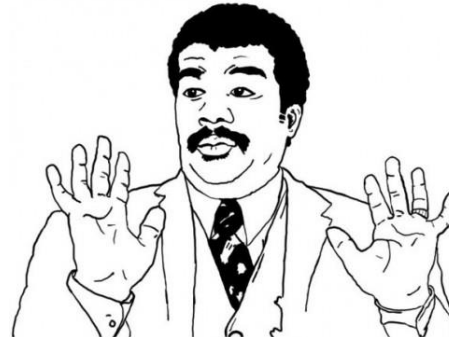
```
string? x = null;
```

```
string y = "Конкретно строка";
```

```
string z1 = y;
```


```
string z2 = x;
```

```
string z3 = x!;
```




Дата релиза C# 8.0

8.0 candidate

 Due by January 01, 2080 6% complete

Candidates for the C# 8.0 release


 Search or jump to... /

dotnet / csharpplang

[Code](#) [Issues 1,387](#) [Pull requests 19](#)

[Labels](#) [Milestones](#)

8.0 candidate

 Due by January 01, 2080 6% complete

Candidates for the C# 8.0 release

[40 Open](#) [3 Closed](#)

Евгений Макаров – оператор лопаты

Вентиляторный совет

Кирилл Маурин – первая лопасть

Никита Цуканов - вторая лопасть

Елизавета Голенок - третья лопасть

Илья Фофанов - ось

Дмитрий Нестерук - отказался участвовать в этом балагане