

C# 15

Размеченные объединения

A dark blue diagonal gradient bar that starts from the bottom left and extends towards the top right, covering the lower half of the slide.

Марк Шевченко

Московский клуб программистов

Писал или пишу на C, C++, Delphi, Perl,
PHP, Ruby, C#, F#, Rust, Go.



Объединения



Структуры

```
struct date {  
    int year;  
    int month;  
    int day;  
};
```

Объединения

```
union contact {  
    char* email;  
    char phone[12];  
    struct address address;  
};
```

Конструирование типов

```
struct user {  
    struct date birth_date;  
    char* name;  
    union contact contact;  
};
```

Хранение

struct

year

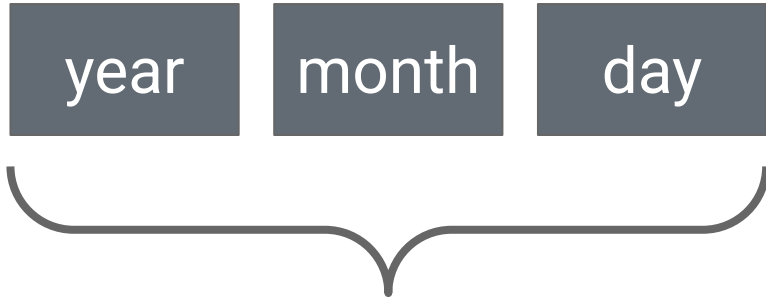
month

day

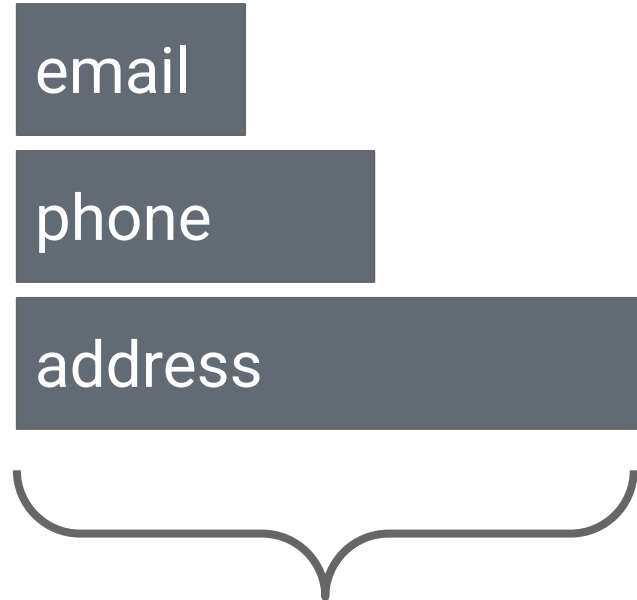


Хранение

struct



union



Тегн

```
struct contact {  
    int tag;  
    union {  
        char* email;  
        char phone[12];  
        struct address address;  
    } value;  
};
```

Tagged Unions
Discriminated Unions

F#

```
type Contact =  
    | Email of string  
    | Phone of int * string  
    | Address of string * string * string  
  
let a = Email("aaa@bbb.com")  
let b = Phone(7, "5551112233")  
let c = Address("103132", "Москва", "Кремль")
```

F#

```
let print contact =  
    match contact with  
    | Email email ->  
        printfn "%s" email  
    | Phone code number ->  
        printfn "+%d%s" code number  
    | Address _ city street ->  
        printfn "%s, %s" city street
```

Классический С#



Interop Services

```
[StructLayout(LayoutKind.Explicit)]  
public struct ArgbUnion  
{  
    [FieldOffset(0)] public uint Argb;  
    [FieldOffset(0)] public byte Blue;  
    [FieldOffset(1)] public byte Green;  
    [FieldOffset(2)] public byte Red;  
    [FieldOffset(3)] public byte Alpha;  
}
```

Иерархия классов

```
public abstract record Contact;
```

```
public sealed record Email(string Email)  
    : Contact;
```

```
public sealed record Phone(int Code,  
    string Number) : Contact;
```

```
public sealed record Address(string Zip,  
    string City, string Street) : Contact;
```

Проверка

```
if (contact is Email e)
    Console.WriteLine($"{e.Email}");
else if (contact is Phone p)
    Console.WriteLine($"{p.Code}{p.Number}");
else if (contact is Address a)
    Console.WriteLine($"{a.City}, {a.Street}");
```

Сравним с F#



Как реализовано в F#

```
[CompilationMapping(SourceConstructFlags.SumType)]  
public abstract class Contact : IEquatable<Contact>,  
..., IStructuralComparable  
{  
    public static class Tags  
    {  
        public const int Email = 0;  
        public const int Phone = 1;  
        public const int Address = 2;  
    }  
    ...  
}
```

Как реализовано в F#

```
public class Email : Contact
{
    public string Item { get; private set; }

    internal Email(string item)
    {
        Item = item;
    }
}
```

Новый С#



СИНТАКСИС

```
public record Email(string Email);  
public record Phone(int Code, string Email);  
public record Address(string Zip,  
    string City, string Street);  
  
public union Contact(Email, Phone, Address);  
  
Contact c = new Email("gmail@chuck.norris");
```

Проверка

```
string s = contact switch
{
    Email e => e.Email,
    Phone p => $" + {p.Code}{p.Number}",
    Address a = $" {a.City}, {a.Street}",
};
```

Реализация

```
[Union]
public record struct Contact : IUnion
{
    public Contact(Email value) =>
        Value = value;
    ...

    public object? Value { get; }
}
```

Проверка

```
string s = contact switch
{
    Email e => e.Email,
    Phone p => $"#{p.Code}{p.Number}",
    Address a => $"#{a.City}, {a.Street}",
    null => "Nowhere",
};
```

Собственная реализация



Пример

```
[StructLayout(LayoutKind.Explicit), Union]  
public struct Variant : IUnion  
{  
    [FieldOffset(0)] private uint _tag;  
    [FieldOffset(4)] private bool _bool;  
    [FieldOffset(4)] private int _int;  
    [FieldOffset(4)] private double _double;  
}
```

Пример

```
public Variant(int value)
{
    _tag = 2;
    _int = value;
}
```

Пример

```
public object? Value => _tag switch
{
    1 => _bool,
    2 => _int,
    3 => _double,
    _ => null
}
```

Пример

```
public bool HasValue => _tag != 0;

public bool TryGetValue(out int value)
{
    value = _int;
    return _tag == 2;
}
```

Пример

```
public string Print(Variant variant) =>
    variant switch
    {
        bool b => $"bool: {b}",
        int i => $"int: {i}",
        double d => $"double: {d}",
        null => "null"
    }
```

Примеры



Nullable<T>, NRT

```
string s1 = "String";
```

```
string? s2 = null;
```

```
int? i = 3;
```

Option (Maybe)

```
public record None;  
public record Some<T>(T Value)  
    where T : notnull;  
public union Option<T>(None, Some<T>);
```

Result (Either)

```
public record Ok<T>(T Value);  
public record Error<E>(E Error);  
public union Result<T, E>(Ok<T>, Error<E>);
```

Make
illegal states
unrepresentable

Пользователь

```
public class User
{
    public string Name { get; }
    public DateOnly BirthDate { get; }
    public Email? Email { get; }
    public Phone? Phone { get; }
    public Address? Address { get; }
}
```

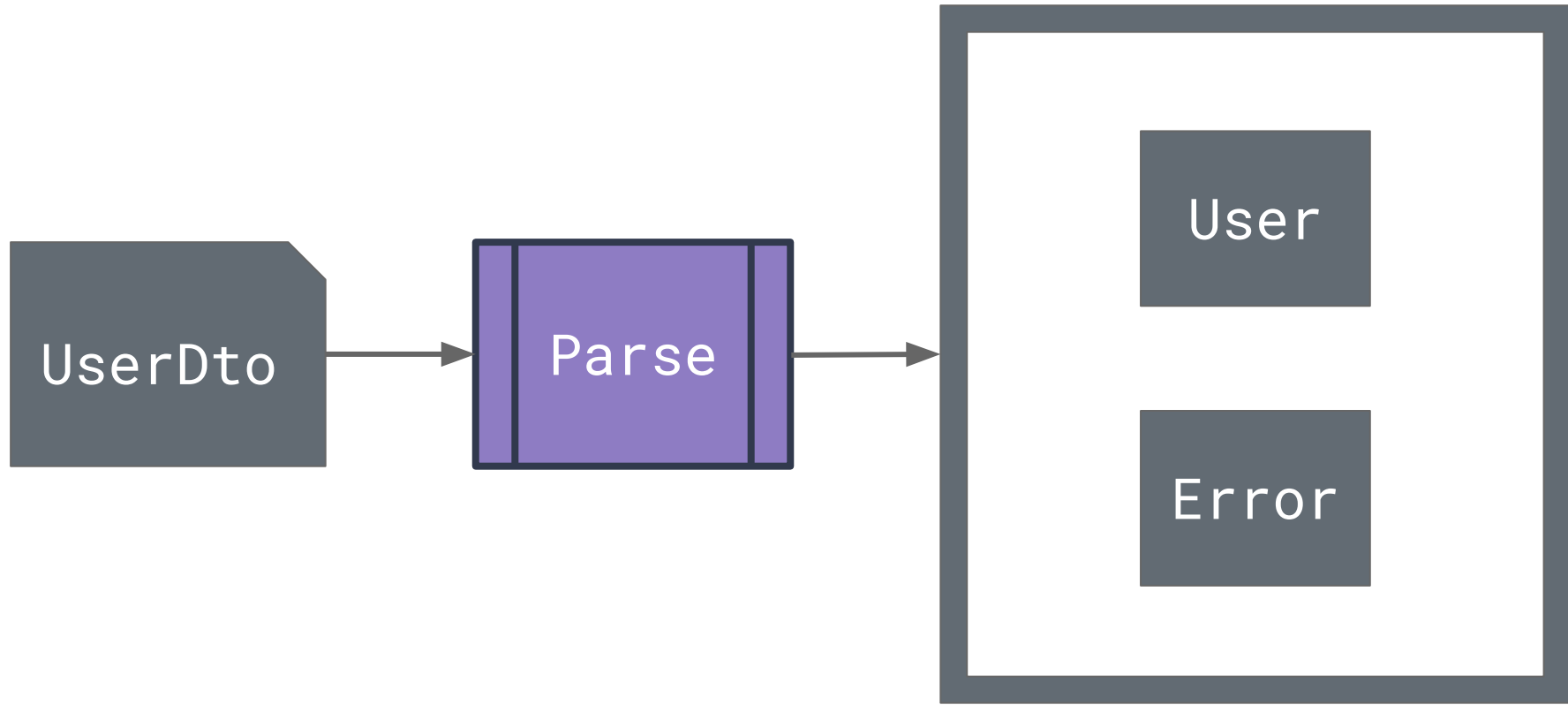
Пользователь

```
public class User
{
    public string Name { get; }
    public DateOnly BirthDate { get; }
    public Email[] Emails { get; }
    public Phone[] Phones { get; }
    public Address[] Addresses { get; }
}
```

Пользователь

```
public class User
{
    public string Name { get; }
    public DateOnly BirthDate { get; }
    public Contact PrimaryContact { get; }
    public Contact[] SecondaryContacts { get; }
}
```

Parse, don't validate

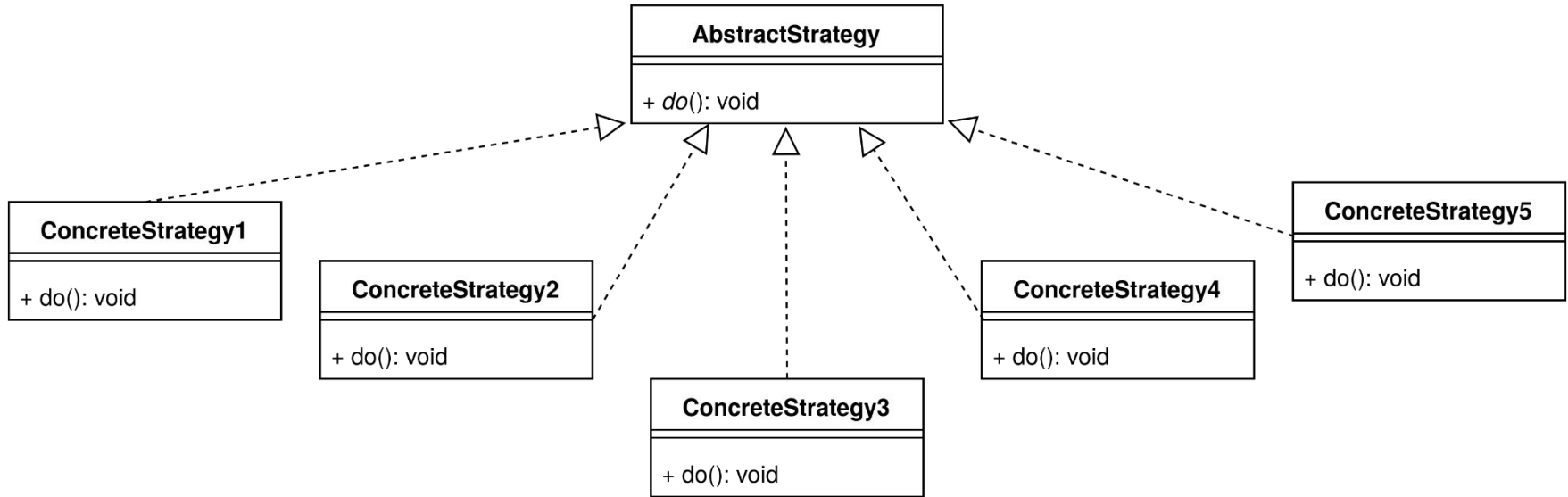


Parse, don't validate

Паттерны



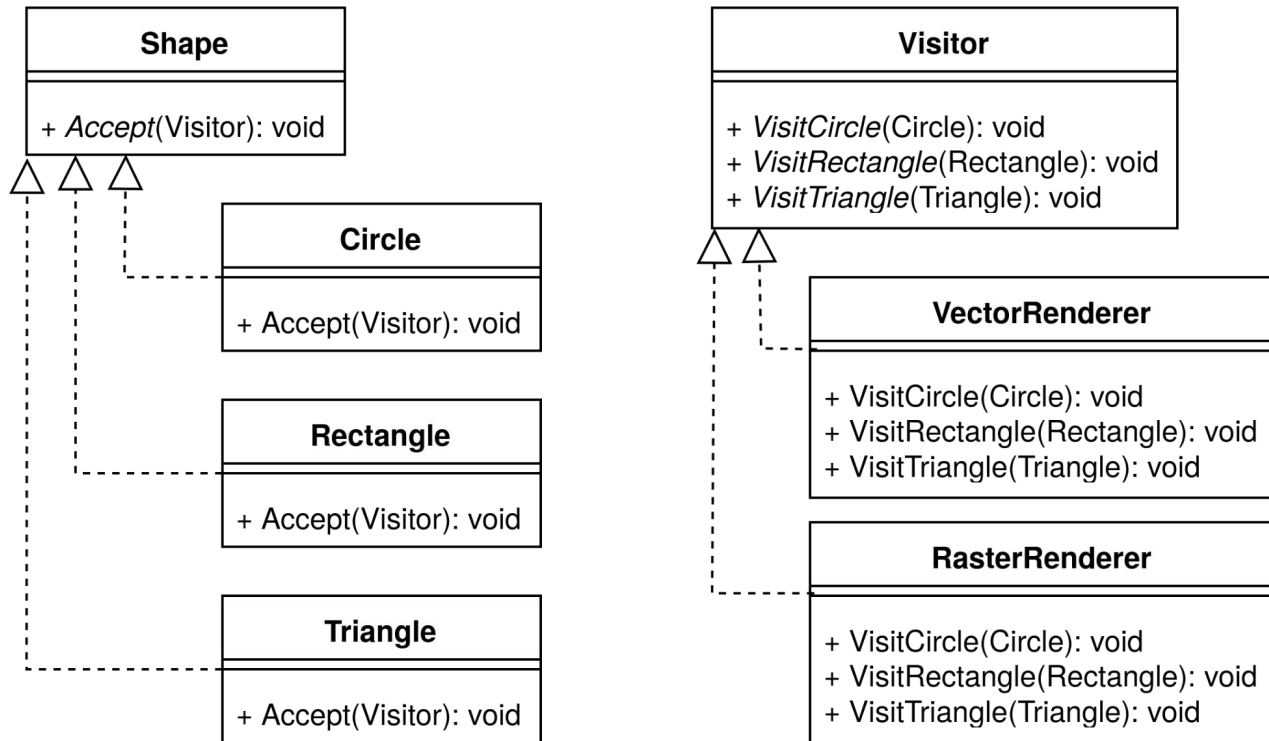
Strategy



Strategy

```
public union Strategy(Case1, Case2, Case3)
{
    public void Do() => Value switch
    {
        Case1 => ...,
        Case2 => ...,
        Case3 => ...,
    };
}
```

Visitor



Visitor

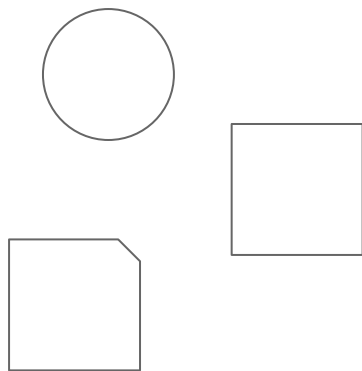
```
public union Shape(Circle, Rectangle, Triangle);  
  
public static void VectorRender(this Shape shape)  
{  
    shape.Value switch  
    {  
        Circle c => ...,  
        Rectangle r => ...,  
        Triangle t => ...,  
    };  
}
```

Недостатки



Реализация

```
type Figure =  
  | Circle of Circle  
  | Rectangle of Rectangle  
  | Folded of Rectangle  
  ...
```



```
public record Folded(Rectangle Rectangle);
```

Сериализация

```
{  
  "contact": {  
    "$type": "email",  
    "value": {  
      "email": "john.smith@mail.ru"  
    }  
  }  
}
```

Сериализация Optional

```
{  
    "optionalInt1": null,  
    "optionalInt2": 3  
}
```

Сериализация Result

```
{  
  "data": 3.1415,  
  "error": null  
}
```

```
{  
  "data": null,  
  "error": "Invalid parameter"  
}
```

Null propagation

do-нотация

```
fn read_file() -> Result<String, Error> {  
    let mut file = File::open("data.txt"?);  
    let mut contents = String::new();  
    file.read_to_string(&mut contents)?;  
    Ok(contents)  
}
```

Источники

- Union Types - C# Reference
<https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/union>
- C# 15 Unions
<https://blog.ndepend.com/csharp-unions/>
- Discriminated Unions: Что не так с реализациями объединений в C#?
<https://habr.com/ru/articles/1025098/>
- Polymorphism inference for closed type hierarchies in System.Text.Json
<https://github.com/dotnet/core/blob/main/release-notes/11.0/preview/preview7/libraries.md#polymorphism-inference-for-closed-type-hierarchies-in-systemtextjson>
- Проектирование с помощью типов
<https://habr.com/ru/articles/1033910/>
- Вычислительные выражения в F#
<https://habr.com/ru/articles/840106/>

Размеченные объединения в C# 15

- [@markshevchenko](https://twitter.com/markshevchenko)
- <https://markshevchenko.pro>
- Московский клуб программистов
[@progmsk](https://twitter.com/progmsk)

Спасибо за внимание!