

# Что нового в .NET 7 и C# 11

Игорь Лабутин

   @ilabutin

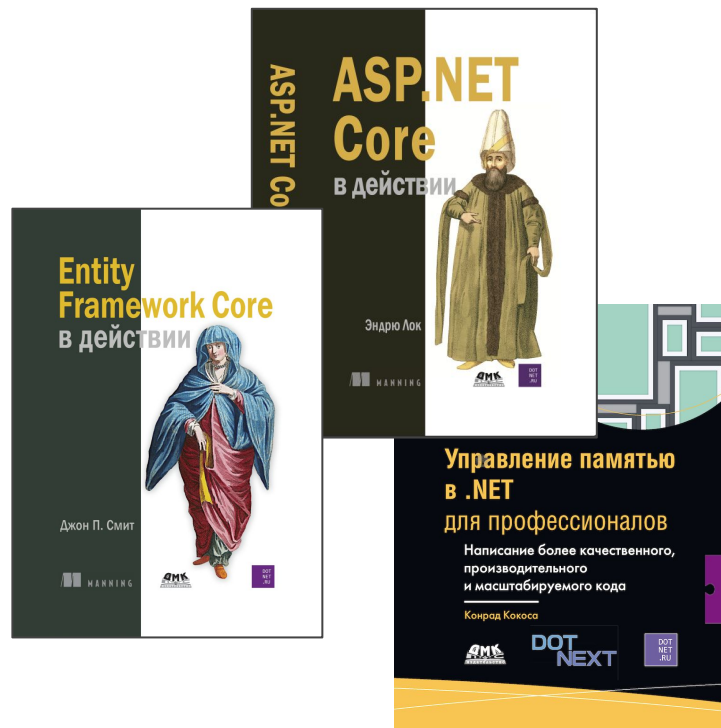
Ноябрь 2022

DOTNEXT

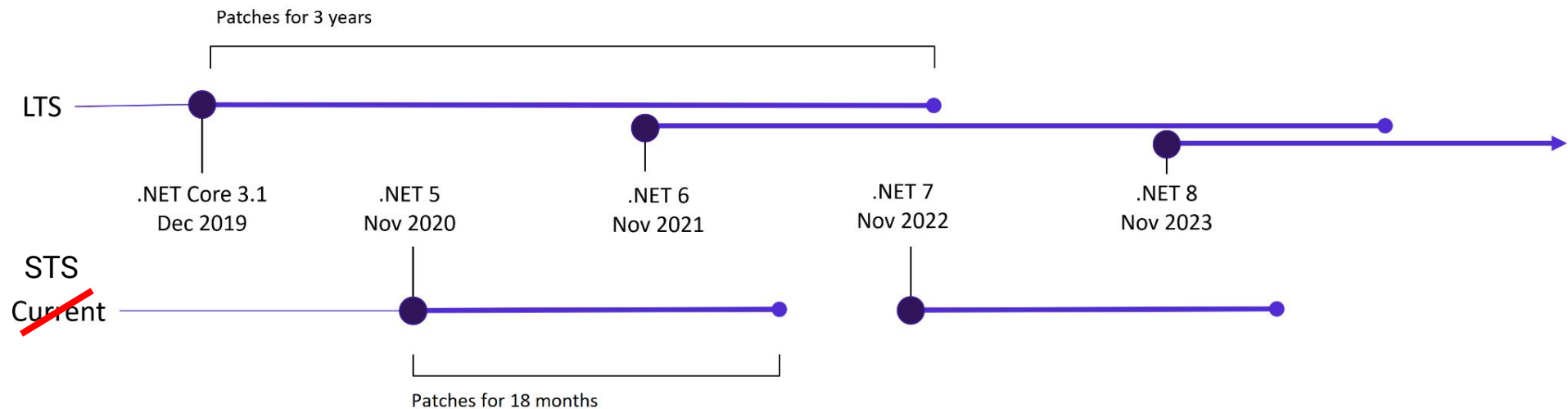
DOT  
NET  
.RU

RADIO  
 DOT  
NET

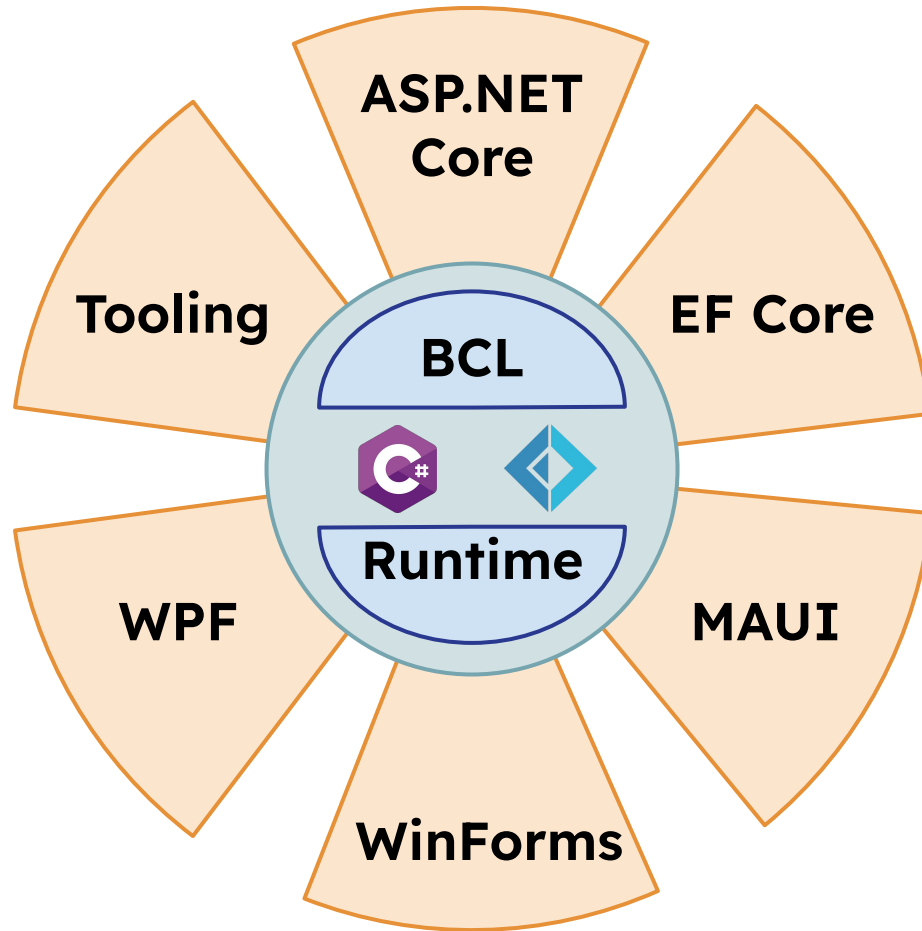
  
**Lanit-Tercom**  
smart software solutions

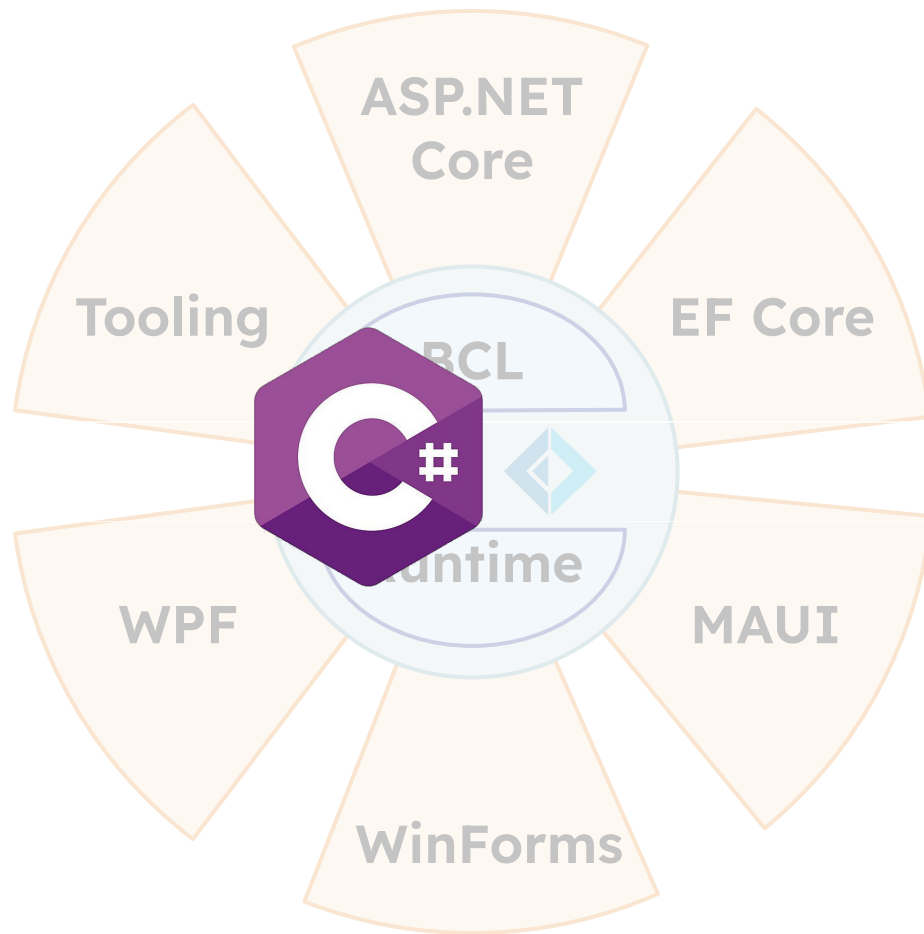


# .NET 7



“The quality of all releases are exactly the same”

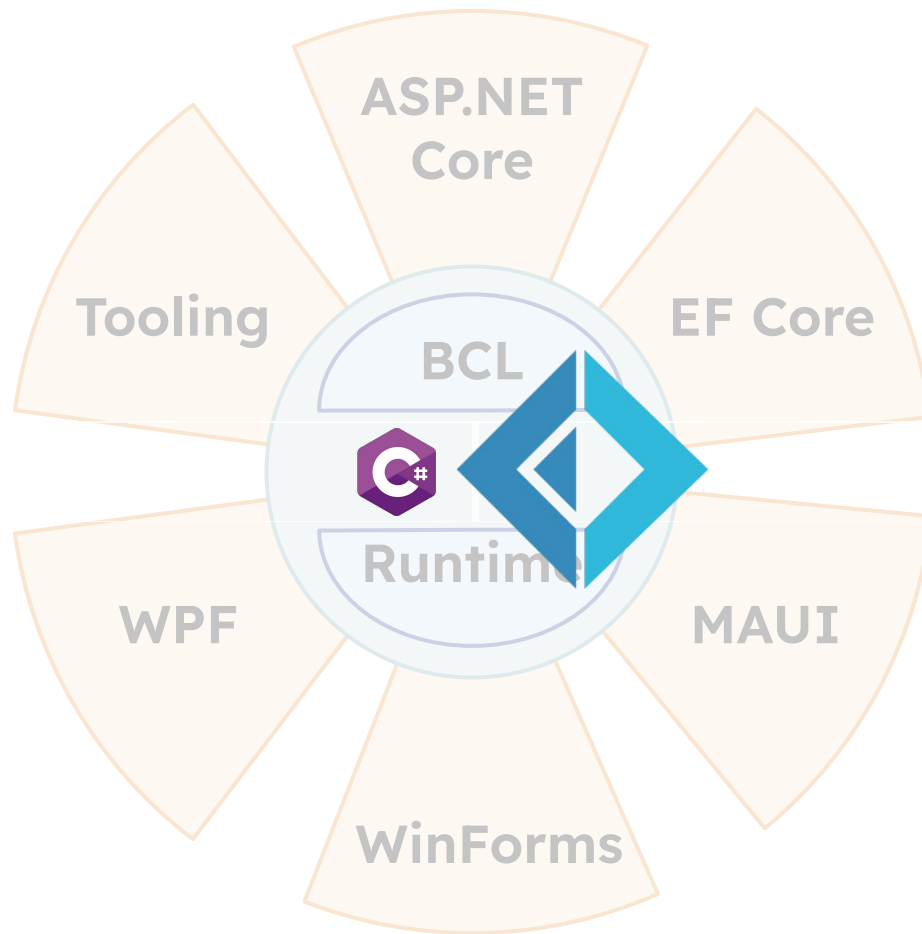




# C# 11

- Required members
- Autodefault structs
- Raw string literals
- Newlines in string interpolation
- Generic attributes
- List patterns
- Static abstract members in interfaces
- Generic math
- Utf-8 string literals
- File-scoped types
- Numeric IntPtr and UIntPtr
- Extended nameof scope
- Unsigned right-shift
- Pattern match on const strings for `Span<char>`

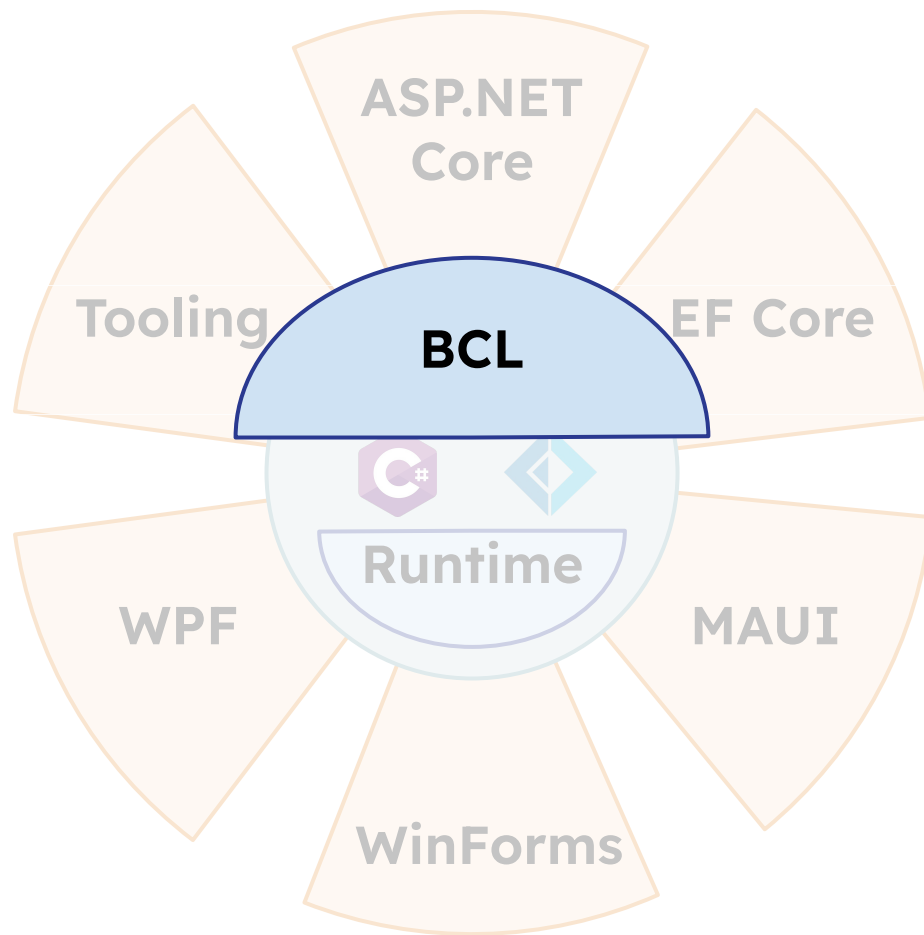
**DEMO TIME**



# F#

- Static abstract members в интерфейсах
- Поддержка required properties
- Упрощение работы с SRTP
- Поддержка reference assemblies
- Сборка self-contained и native AOT
- Поддержка ARM64





# StringSyntaxAttribute

- StringSyntaxAttribute
  - CompositeFormat
  - DateOnlyFormat
  - DateTimeFormat
  - EnumFormat
  - GuidFormat
  - Json
  - NumericFormat
  - Regex
  - TimeOnlyFormat
  - TimeSpanFormat
  - Uri
  - Xml

```
new WebProxy("http://myproxy", false, new[]  
{  
    @"[a-z]+\.\contoso\.com$",  
    @"192\168\.\d{1,3}\.\d{1,3}",  
});
```

\P{...}	negative unicode category ▲
\s	white-space character
\S	non-white-space character
\w	word character
\W	non-word character
	alternation
^	start of string or line
\$	end of string or line ▼

# Регулярные выражения - генератор кода

- Скорость
- Отладка

```
if (Helpers.MyCoolRegex().IsMatch(text))
{
    Console.WriteLine(...)
}
```

2 references

```
partial class Helper
{
    [RegexGenerator("abc|def", RegexOptions.IgnoreCase)]
    internal static partial Regex MyCoolRegex();
}
```

Regex Helpers.MyCoolRegex()

Pattern explanation:

- Match with 2 alternative expressions, atomically.
  - Match a sequence of expressions.
    - Match a character in the set [Aa].
    - Match a character in the set [Bb].
    - Match a character in the set [Cc].
  - Match a sequence of expressions.
    - Match a character in the set [Dd].
    - Match a character in the set [Ee].
    - Match a character in the set [Ff].

# Регулярные выражения - новый режим

Regex: `.+b`  
Input: `aaabcd`

# Регулярные выражения - новый режим

Regex: `.+b`  
Input: `aaabcd`

Шаг 1: пробуем `.+` с `aaabc` -> следующий `d` не подходит

# Регулярные выражения - новый режим

Regex: `.+b`  
Input: `aaabcd`

Шаг 1: пробуем `.+` с `aaabc` -> следующий `d` не подходит  
Шаг 2: пробуем `.+` с `aaab` -> следующий `c` не подходит

# Регулярные выражения - новый режим

Regex: `.+b`  
Input: `aaabcd`

Шаг 1: пробуем `.+` с `aaabc` -> следующий `d` не подходит  
Шаг 2: пробуем `.+` с `aaab` -> следующий `c` не подходит  
Шаг 3: пробуем `.+` с `aaa` -> следующий `b` подходит

# Регулярные выражения - новый режим

Regex: `.+b`  
Input: `aaabcd`

Шаг 1: пробуем `.+` с `aaabc` -> следующий `d` не подходит  
Шаг 2: пробуем `.+` с `aaab` -> следующий `c` не подходит  
Шаг 3: пробуем `.+` с `aaa` -> следующий `b` подходит

- `RegexOptions.NonBacktracking`



# Регулярные выражения - новый режим

Regex: `.+b`  
Input: `aaabcd`

Шаг 1: пробуем `.+` с `aaabc` -> следующий `d` не подходит  
Шаг 2: пробуем `.+` с `aaab` -> следующий `c` не подходит  
Шаг 3: пробуем `.+` с `aaa` -> следующий `b` подходит

- `RegexOptions.NonBacktracking`
- `NonBacktracking` не совместимо с `Regex.Compiled` и кодогенератором

# Регулярные выражения - оптимизации

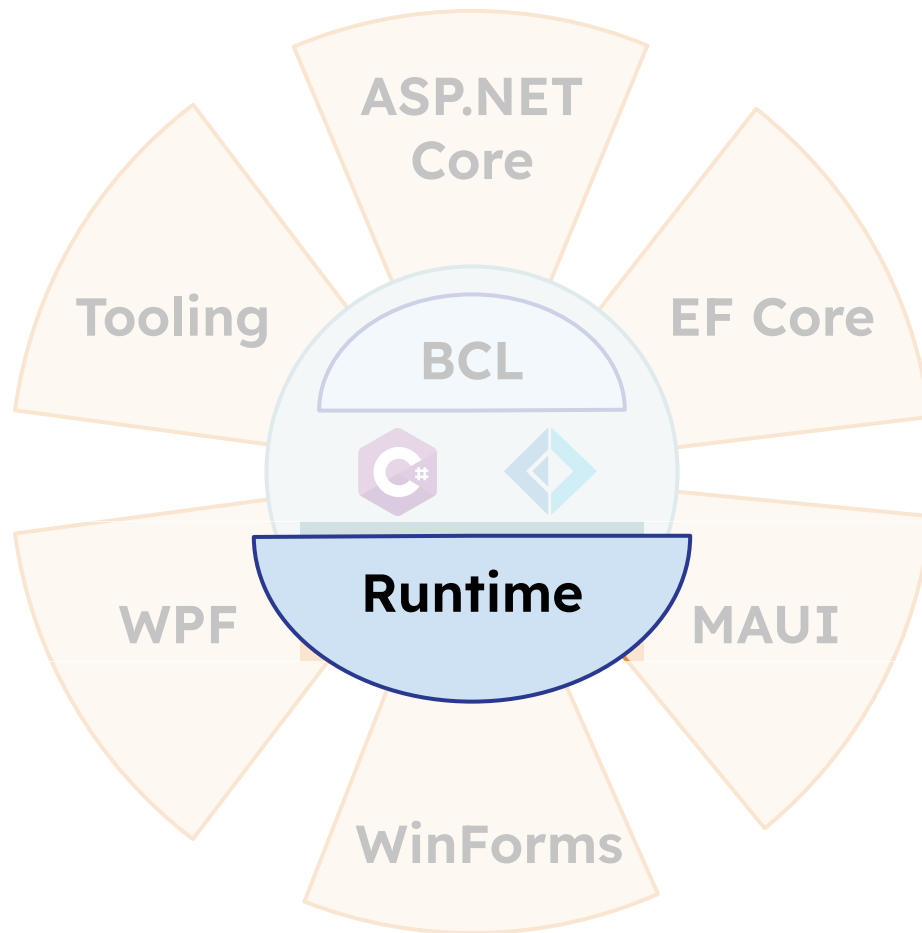
- `RegexOptions.IgnoreCase`
- `Span<T>`
- Векторизация

Method	Runtime	Mean
ManualSet1	.NET 6	85.75 us
IgnoreCase2	.NET 6	235.40 us
ManualSet3	.NET 6	4.312 us
IgnoreCase4	.NET 6	222.387 us
ManualSet1	.NET 7	47.167 us
IgnoreCase2	.NET 7	47.130 us
ManualSet3	.NET 7	4.147 us
IgnoreCase4	.NET 7	4.135 us

# Приятные мелочи

- Nullable аннотации для Microsoft.Extensions
- Поддержка TAR
- DllImport -> LibraryImport
- Микро- и наносекунды
- `System.IO.Stream: ReadAtLeast/ReadExactly`

DEMO TIME



# .NET 7 Performance – APIs

Requests Per Second (Millions)

0.60M

Node.js

2.20M

Java Servlet

7.02M

ASP.NET Core

ASP.NET CORE WEB  
FRAMEWORK IS

>11x

FASTER THAN  
NODE.JS

# Ускоряемся!

- Method group conversion для делегатов
- GC: Регионы
  - Включены по умолчанию
  - Ускоряют фазу разметки
- Reflection
- JIT
  - Dynamic PGO

DEMO TIME

# Native AOT

- .NET давно поддерживает (ngen, crossgen)
  - Все еще требует JIT
- Native AOT это другое!
- Полностью нативное приложение, без зависимостей, без JIT
- Добавляем `<PublishAot>true</PublishAot>` в .csproj
- Собираем `dotnet publish -r linux-arm64 -c Release`
- Можно собирать нативные .DLL/.SO
- Быстрее запуск

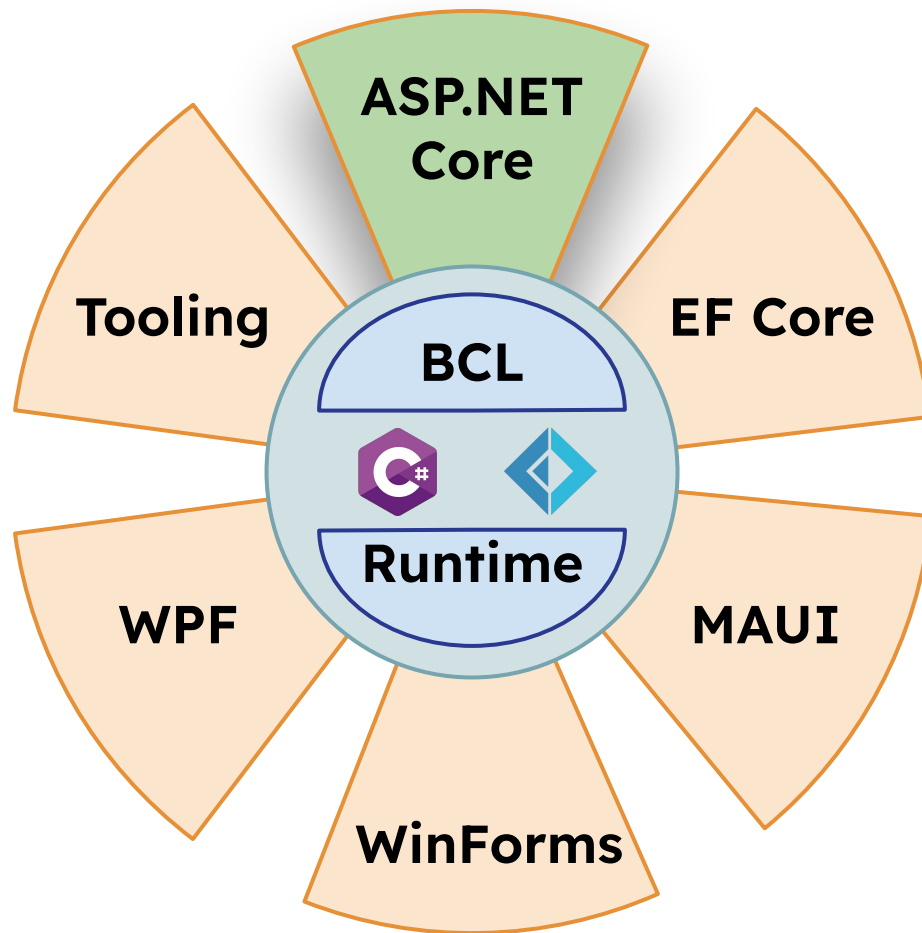
<https://learn.microsoft.com/en-gb/dotnet/core/deploying/native-aot/>

<https://github.com/dotnet/runtime/tree/main/src/coreclr/nativeaot/docs>

# Native AOT - Ограничения

- Нет динамической загрузки сборок
- Нельзя генерировать код в рантайме
- Нет поддержки C++/CLI
- Нет поддержки COM
- Требуется использование assembly trimming
- Сборка в один файл
- Все зависимости включены в итоговый файл





# Что такое ASP.NET Core

- Сервера и рантайм
- MVC
- Minimal APIs
- gRPC
- SignalR
- Blazor

# Начнем с простого

- Параметры
  - Атрибут `FromServices`
  - `IParsable<T>.TryParse` для привязки параметров

# Начнем с простого

- Параметры
  - Атрибут `FromServices`
  - `IParsable<T>.TryParse` для привязки параметров
- Результаты
  - `Result.Stream`
  - Пространство имен `HttpResults` - все реализации `IResult`

# Minimal API

- Типизированные результаты
- Поддержка OpenAPI
- Группировка маршрутов
- Фильтры обработчиков маршрутов
- Привязка параметров (массивы, streams)

**DEMO TIME**

# System.Text.Json

- Полиморфизм
  - [JsonDerivedType] на базовый тип
  - [JsonPolymorphic] для кастомизации поведения

# System.Text.Json

- Полиморфизм
  - `[JsonDerivedType]` на базовый тип
  - `[JsonPolymorphic]` для кастомизации поведения
- Кастомизация контрактов JSON
  - Через `JsonSerializerOptions.TypeInfoResolver`
  - Меняем `JsonPropertyInfo` в резолвере

# Middleware

- Rate limiting
- Output caching
- Request decompression



# gRPC

- gRPC как JSON API

# gRPC

- gRPC как JSON API
  - Изменяем .proto

```
service Greeter {  
  rpc SayHello (HelloRequest) returns (HelloReply) {  
    option (google.api.http) = {  
      get: "/v1/greeter/{name}"  
    };  
  }  
}
```

# gRPC

- gRPC как JSON API

- Изменяем .proto

```
service Greeter {  
  rpc SayHello (HelloRequest) returns (HelloReply) {  
    option (google.api.http) = {  
      get: "/v1/greeter/{name}"  
    };  
  }  
}
```

- Добавляем сервис: `services.AddGrpc().AddJsonTranscoding()`

# gRPC

- gRPC как JSON API

- Изменяем .proto

```
service Greeter {  
  rpc SayHello (HelloRequest) returns (HelloReply) {  
    option (google.api.http) = {  
      get: "/v1/greeter/{name}"  
    };  
  }  
}
```

- Добавляем сервис: `services.AddGrpc().AddJsonTranscoding()`

- Только server-side стриминг

# gRPC

- gRPC как JSON API

- Изменяем .proto

```
service Greeter {  
  rpc SayHello (HelloRequest) returns (HelloReply) {  
    option (google.api.http) = {  
      get: "/v1/greeter/{name}"  
    };  
  }  
}
```

- Добавляем сервис: `services.AddGrpc().AddJsonTranscoding()`
- Только server-side стриминг
- Экспериментальная поддержка OpenAPI: `services.AddGrpcSwagger();`

# Observability

- Поддержка OpenTelemetry
- `System.Diagnostics.Activity`
  - Событие `CurrentChanged`
  - Performant enumerators
- Метрики для `IMemoryCache`

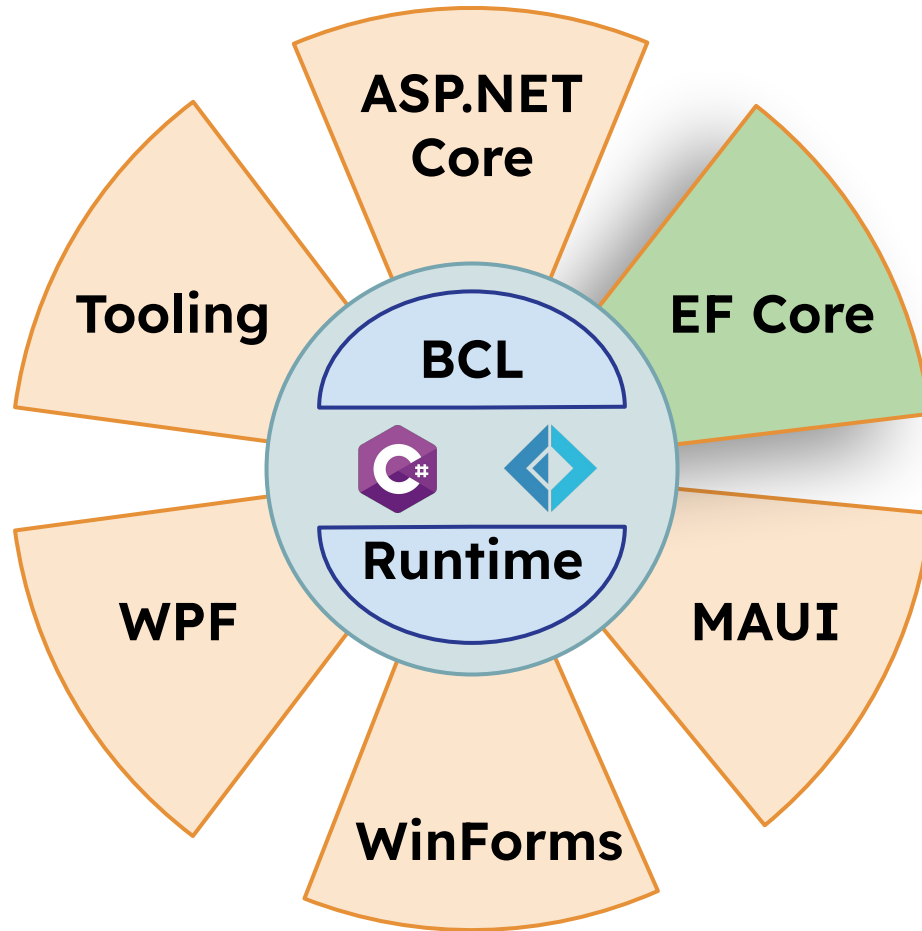
# Blazor WASM

- Интероп с JavaScript: [JSImport] / [JSExport]
- AOT + SIMD
- Исключения в WASM
- Запросы авторизации в рантайме
- Экспериментальная поддержка многопоточности

# Сервера

- WebSockets поверх HTTP/2
- Официально поддержан HTTP/3
- Экспериментальная поддержка WebTransport поверх HTTP/3



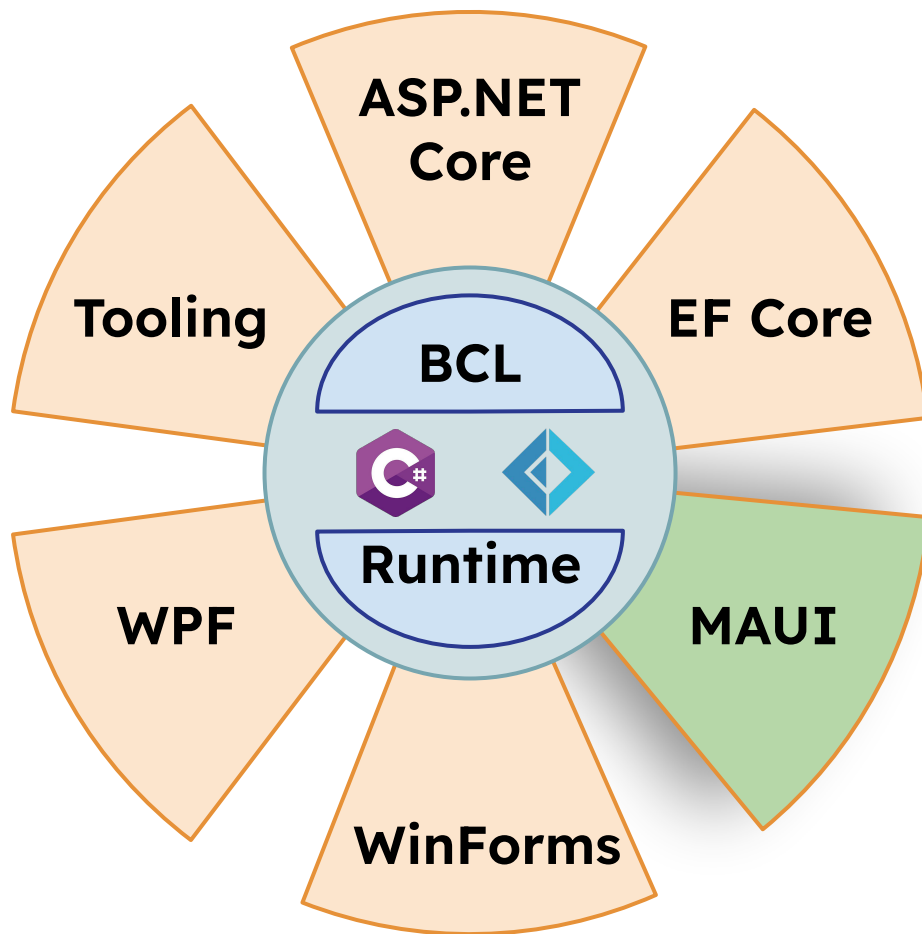


## EF Core - новые фишки

- Поддержка Table-per-concrete-type
- Поддержка JSON колонок
- Работа с хранимыми процедурами для insert/update/delete
- Шаблоны T4 для DB-first

# EF Core - улучшение производительности

- EF Core 6 - про чтение
- EF Core 7 - про запись
- Bulk updates/deletes
- Ускорение SaveChanges

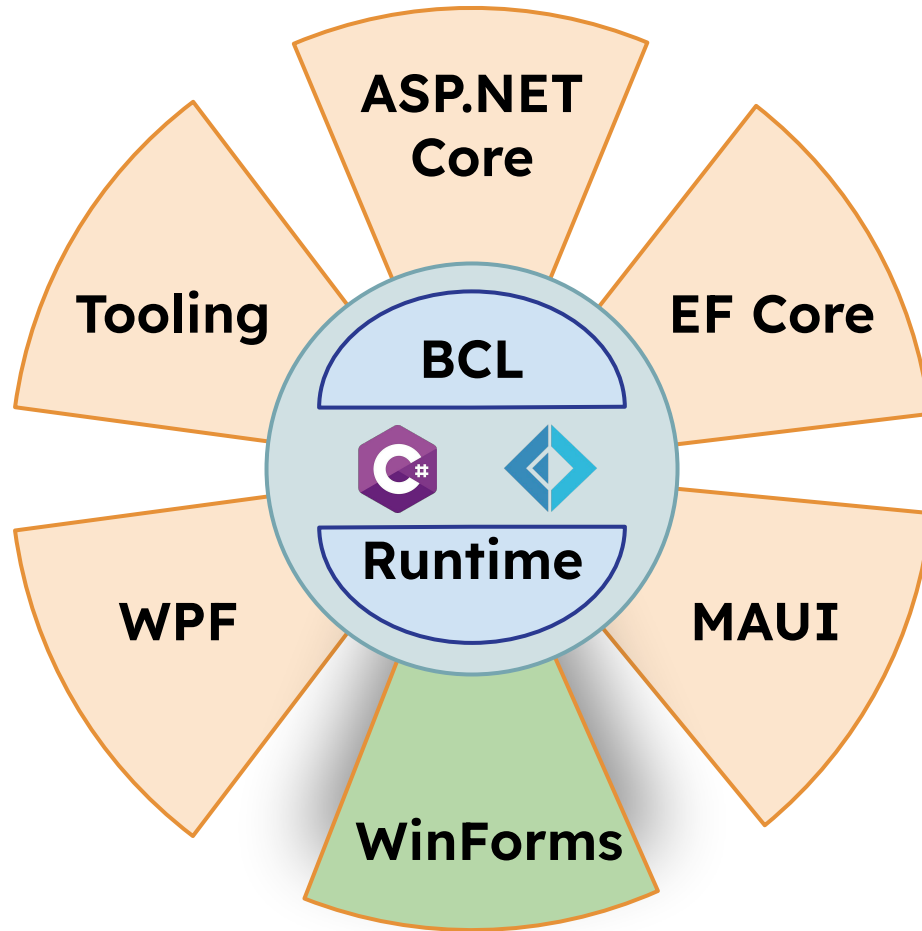


# MAUI

- Поддержка Android, iOS, MacOS, Windows (+Tizen by Samsung)
  - Поддержка Linux - только силами сообщества
- Ускорился рендеринг
- Новые контролы
- Поддержка desktop приложений

# MAUI - примеры

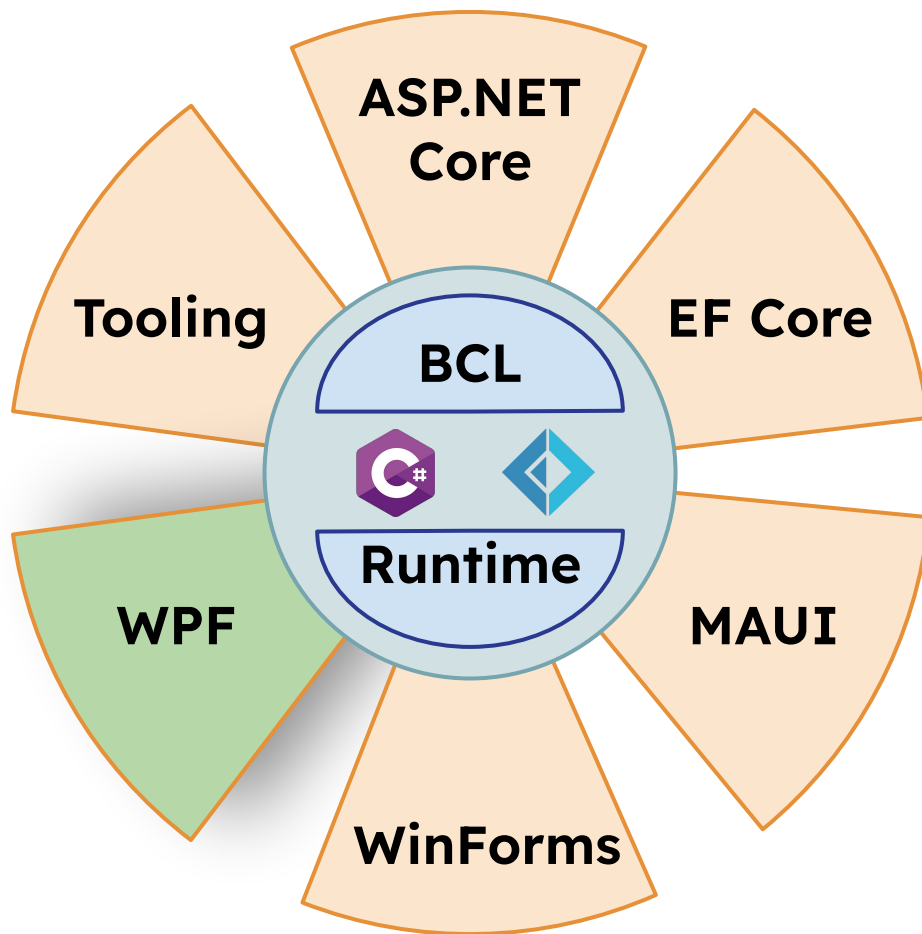
- Новое “модельное” приложение - Подкасты
- ASP.NET Core (Minimal APIs, Razor)
- Blazor WASM
- MAUI app
- Hybrid app



# WinForms

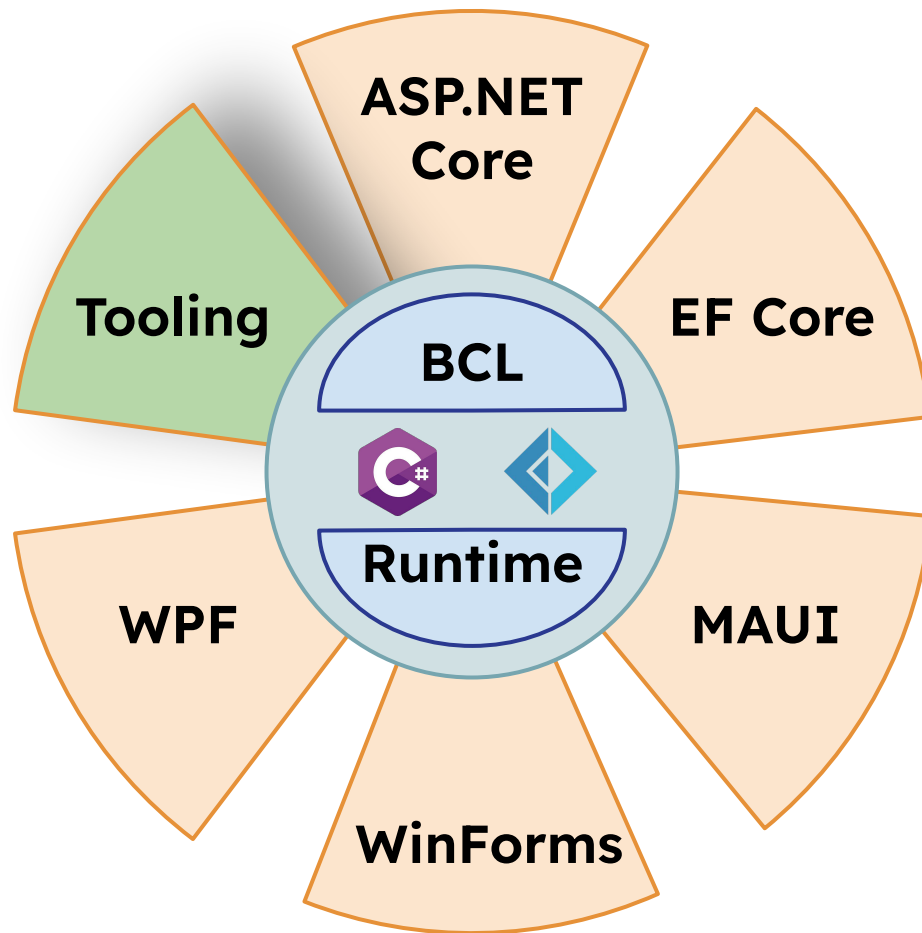
- Доступность (accessibility)
- Поддержка HighDPI
- Экспериментальная поддержка ComWrappers и Native AOT
- Экспериментальная превью поддержка новых data bindings





# WPF

- Доступность
- Оптимизация памяти
- Упрощение сторонних контрибьютов
  - Инфраструктура тестов
  - Запуск тестов на PR



# Build/tooling

- Visual Studio 2022 17.4
- Rider 2022.3 EAP
- .NET SDK
  - Дополнительные компоненты через “dotnet workload”

# Поддержка контейнеров

- Сборка контейнера без Dockerfile

```
# add a reference to a (temporary) package that creates the container
```

```
dotnet add package Microsoft.NET.Build.Containers
```

```
# publish your project for linux-x64
```

```
dotnet publish --os linux --arch x64 -c Release -p:PublishProfile=DefaultContainer
```

# Поддержка контейнеров

- Сборка контейнера без Dockerfile

```
# add a reference to a (temporary) package that creates the container
```

```
dotnet add package Microsoft.NET.Build.Containers
```

```
# publish your project for linux-x64
```

```
dotnet publish --os linux --arch x64 -c Release -p:PublishProfile=DefaultContainer
```

- Настройка через свойства в .csproj

# Поддержка контейнеров

- Сборка контейнера без Dockerfile

```
# add a reference to a (temporary) package that creates the container
```

```
dotnet add package Microsoft.NET.Build.Containers
```

```
# publish your project for linux-x64
```

```
dotnet publish --os linux --arch x64 -c Release -p:PublishProfile=DefaultContainer
```

- Настройка через свойства в .csproj

- Но не всё ещё реализовано

# Поддержка контейнеров

- Сборка контейнера без Dockerfile

```
# add a reference to a (temporary) package that creates the container
```

```
dotnet add package Microsoft.NET.Build.Containers
```

```
# publish your project for linux-x64
```

```
dotnet publish --os linux --arch x64 -c Release -p:PublishProfile=DefaultContainer
```

- Настройка через свойства в .csproj
  - Но не всё ещё реализовано
- Сейчас только для Linux-x64



# Поддержка контейнеров

- Сборка контейнера без Dockerfile

```
# add a reference to a (temporary) package that creates the container
```

```
dotnet add package Microsoft.NET.Build.Containers
```

```
# publish your project for linux-x64
```

```
dotnet publish --os linux --arch x64 -c Release -p:PublishProfile=DefaultContainer
```

- Настройка через свойства в .csproj
  - Но не всё ещё реализовано
- Сейчас только для Linux-x64
- Нет авторизации для докера, поэтому push не работает

# Централизованное управление пакетами

- Directory.Packages.props

```
<Project>
  <PropertyGroup>
    <ManagePackageVersionsCentrally>true</ManagePackageVersionsCentrally>
  </PropertyGroup>
  <ItemGroup>
    <PackageVersion Include="Newtonsoft.Json" Version="13.0.1" />
  </ItemGroup>
</Project>
```

# Централизованное управление пакетами

- Directory.Packages.props

```
<Project>
  <PropertyGroup>
    <ManagePackageVersionsCentrally>true</ManagePackageVersionsCentrally>
  </PropertyGroup>
  <ItemGroup>
    <PackageVersion Include="Newtonsoft.Json" Version="13.0.1" />
  </ItemGroup>
</Project>
```

- .csproj

```
<ItemGroup>
  <PackageReference Include="Newtonsoft.Json"/>
</ItemGroup>
```

# Централизованное управление пакетами

- Directory.Packages.props

```
<Project>
  <PropertyGroup>
    <ManagePackageVersionsCentrally>true</ManagePackageVersionsCentrally>
  </PropertyGroup>
  <ItemGroup>
    <PackageVersion Include="Newtonsoft.Json" Version="13.0.1" />
  </ItemGroup>
</Project>
```

- .csproj

```
<ItemGroup>
  <PackageReference Include="Newtonsoft.Json" VersionOverride="12.0.0"/>
</ItemGroup>
```

# Централизованное управление пакетами

- Directory.Packages.props

```
<Project>
  <PropertyGroup>
    <ManagePackageVersionsCentrally>true</ManagePackageVersionsCentrally>
  </PropertyGroup>
  <ItemGroup>
    <PackageVersion Include="Newtonsoft.Json" Version="13.0.1" />
  </ItemGroup>
</Project>
```

- .csproj

```
<ItemGroup>
  <PackageReference Include="Newtonsoft.Json" VersionOverride="12.0.0"/>
</ItemGroup>
```

- Транзитивное закрепление версий

# Централизованное управление пакетами

- Directory.Packages.props

```
<Project>
  <PropertyGroup>
    <ManagePackageVersionsCentrally>true</ManagePackageVersionsCentrally>
  </PropertyGroup>
  <ItemGroup>
    <PackageVersion Include="Newtonsoft.Json" Version="13.0.1" />
  </ItemGroup>
</Project>
```

- .csproj

```
<ItemGroup>
  <PackageReference Include="Newtonsoft.Json" VersionOverride="12.0.0"/>
</ItemGroup>
```

- Транзитивное закрепление версий
- Глобальные ссылки на пакеты

# Итоги

- Скорость
- Удобство
- Эксперименты

<https://devblogs.microsoft.com/dotnet/announcing-dotnet-7/>

<https://devblogs.microsoft.com/dotnet/regular-expression-improvements-in-dotnet-7/>

[https://devblogs.microsoft.com/dotnet/performance\\_improvements\\_in\\_net\\_7/](https://devblogs.microsoft.com/dotnet/performance_improvements_in_net_7/)

<https://devblogs.microsoft.com/dotnet/regular-expression-improvements-in-dotnet-7/>

<https://devblogs.microsoft.com/nuget/announcing-nuget-6-4-signed-central-delivered/>

<https://www.dotnetconf.net/>



