

.Net

- .Net
 - Литература
 - Использование и применение
 - Применение
 - .NET Framework
 - CLR
 - История версий
 - .NET Core
 - Command-line interface tools
 - .NET Standard
 - .NET Native
 - IDE
 - Nuget

Литература

- [.NET Documentation](#)
- Jeffrey Richter, CLR Via C# (4th edition)
- Jon Skeet, C# in Depth
- Andrew Troelsen, C# 6.0 and the .NET 4.6 Framework (Самый низкий порог вхождения в изучение C#)
- Сергей Тепляков, [Набор доступных статей про принципам проектирования](#), книга "Паттерны проектирования на платформе .NET"

Использование и применение

- Высокоуровневый ооп язык
- Строгая типизация
- Автоматическое управление памятью

Плюсы:

- Синтаксис и возможности
- Быстрое развитие
- IDE
- Высокая производительность по сравнению с динамически-типизированными языками
- Отсутствие проблем с версиями (DLL хелл)

Минусы:

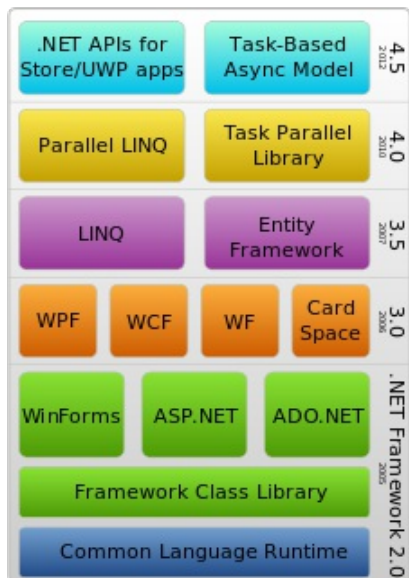
- До .NET Core сильнейшая привязка к Windows
- Бардак с большим количеством фреймворков, которые живут одновременно
- Высочайшая скорость разработки новых фреймворков

[Java vs C# Stackoverflow](#)

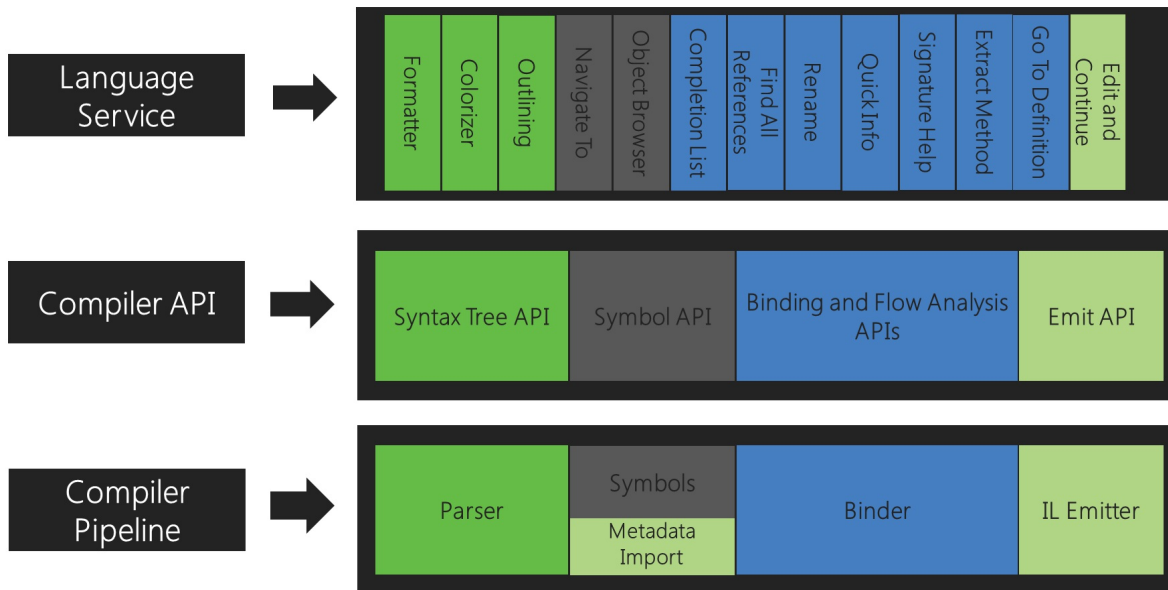
Применение

- ServerSide
- GameDev (Unity, ServerSide, etc)
- UWP / WPF / WinForms Application

.NET Framework



Код собирается компилятором языков ([Roslyn](#)) в промежуточный байт-код CIL ([Common Intermediate Language](#)), который именуется сборкой (assembly). У Roslyn есть собственное API:



Пример кода CIL языка (IL), который получается:

```
.class public Foo
{
    .method public static int32 Add(int32, int32) cil managed
    {
        .maxstack 2
        ldarg.0 // load the first argument;
        ldarg.1 // load the second argument;
        add     // add them;
        ret     // return the result;
    }
}
```

```
int r = Foo.Add(2, 3);    // 5
```

In CIL:

```
ldc.i4.2  
ldc.i4.3  
call int32 Foo::Add(int32, int32)  
stloc.0
```



```

.class public Car
{
    .method public specialname rtspecialname instance void .ctor(int32, int32) cil
managed
    {
        /* Constructor */
    }

    .method public void Move(int32) cil managed
    {
        /* Omitting implementation */
    }

    .method public void TurnRight() cil managed
    {
        /* Omitting implementation */
    }
}

```

Microsoft заложила возможность мультиплатформенности - стандарт CLI ([Common Language Infrastructure](#)), но по факту получился windows-only.

Сейчас область применения .NET Framework постепенно сужается. Он уходит из бэкенда, но остается в windows-desktop, gamedev

CLR

[CLR](#) - исполняющая среда для выполнения CIL. JIT компилятор - часть CLR.

.Net совместимые языки: C#, F#, C++/CLI (legacy name "Managed C++"), VB.Net

Managed Code - код, который может запускаться только из-под CLR/Mono/etc

Stroustrup: "On the difficult and controversial question of what the CLI binding/extensions to C++ is to be called, I prefer C++/CLI as a shorthand for "The CLI extensions to ISO C++". Keeping C++ as part of the name reminds people what is the base language and will help keep C++ a proper subset of C++ with the C++/CLI extensions."

История версий

C#	C# 1.0	C# 2.0	C# 3.0	C# 4.0	C# 5.0	C# 6.0	C# 7.0
.NET Framework	1.0/1.1	2.0	3.0/3.5	4.0	4.5	4.5/4.6	4.5- 4.7
Visual Studio	2002	2005	2008	2010	2012/13	2013/2015	2017
Net Core	-	-	-	-	-	1.0	1.1/2.0
Features	Basic	Generics Partial Nullable Properties Static Delegates	AnonTypes Extensions QueryExp Lambda	dynamic OptionalArgs Generic covariance	Async	C# 6.0 New	C# 7.0 New

- [.NET Framework Guide](#)
- [.NET Core Roadmap & Supported Platforms](#)
- [C#7.0 with .Net Framework 4.0/4.5](#)

.NET Core

Платформа для разработки:

- Кроссплатформенная
- Все CLI команды могут быть реализованы через командную строку command-line interface (CLI) tools
- OpenSource: MIT + Apache2.0

CLR -> CoreCLR, включает новый JIT компилятор RyuJIT, GC, etc

FCL -> .NET Core Libraries (CoreFx): System.Collections, System.IO, System.Xml, etc

- Удалили много лишнего (WebForms, WinForms, WPF, WCF, EF) по сравнению с .NET Framework, сделали код OpenSource, ускорили.
- Отдельные классы из старого фреймворка выложили отдельными напетами.

MS не стремится сделать вес фреймворка меньше, а хочет, чтобы не было лишних зависимостей

- .NET Framework 4.6 - 200 MB
- .NET Core 1.0 - 11 MB / .NET Core 2.0 - 112 MB

Command-line interface tools

Портировали MSBuild на .NET Core и добавили в него новые методы.

Базовые CLI команды:

- new
- restore
- run
- build
- publish
- test
- pack

Примеры вызова через командную строку:

- `dotnet restore`
- `dotnet <command> --help`
- `dotnet publish -o pub -c Release`

.NET Standard

Microsoft управляет несколькими .NET фреймворками: .NET Framework, .NET Core, Xamarin, etc. При этом ядро фреймворков начало расходиться и нужно было реализовать возможность писать портируемый код между платформами.

Придумали .NET Standard. Сам по себе он не содержит реализации.

Это список типов и интерфейсов, которые он требует для реализации тех, кто хочет его поддерживать.

.NET Standard расширяется и новые ветки фреймворков реализуют у себя его требования и поддерживают все более и более широкую его версию.

.NET Standard позволяет создавать библиотеку, которую можно использовать в разных фреймворках.

.NET implementation support

The following table lists all versions of .NET Standard and the platforms supported:

.NET Standard	1.0	1.1	1.2	1.3	1.4	1.5	1.6	2.0
.NET Core	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0
.NET Framework (with .NET Core 1.x SDK)	4.5	4.5	4.5.1	4.6	4.6.1	4.6.2		
.NET Framework (with .NET Core 2.0 SDK)	4.5	4.5	4.5.1	4.6	4.6.1	4.6.1	4.6.1	4.6.1
Mono	4.6	4.6	4.6	4.6	4.6	4.6	4.6	5.4
Xamarin.iOS	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.14
Xamarin.Mac	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.8
Xamarin.Android	7.0	7.0	7.0	7.0	7.0	7.0	7.0	7.5
Universal Windows Platform	10.0	10.0	10.0	10.0	10.0	vNext	vNext	vNext
Windows	8.0	8.0	8.1					
Windows Phone	8.1	8.1	8.1					
Windows Phone Silverlight	8.0							

.NET Native

Технология компиляции в нативный код ahead-of-time.

Раньше для этого использовался `ngen`, но у них есть существенные отличия (не использует CLR и JIT вообще, вместо него обрезанный, отрефакторенный runtime).

Ускоряет первый запуск программы.

Проблемы с рефлексомом: все машинные коды должны быть сгенерированы заранее, эвристика компилятора не может угадать все возможные сценарии метапрограммирования.

Не работает в .net core 2.0.

IDE

- [Visual Studio 2017](#) + [Resharper](#)
- [Visual Studio Code](#)
- [Visual Studio for Mac](#) (до этого - Xamarin)
- [JetBrains Rider](#)
- MonoDevelop, SharpDevelop, etc
- [LINQPad](#) - для тестирования и быстрой отладки

Установка Visual Studio отличается красивым выбором Workloads:

Workloads

Individual components

Language packs

Windows (3)



Universal Windows Platform development
Create applications for the Universal Windows Platform with C#, VB, JavaScript, or optional...



.NET desktop development
Build WPF, Windows Forms and console applications using the .NET Framework.



Desktop development with C++
Build classic Windows-based applications using the power of the Visual C++ toolset...



Web & Cloud (5)



ASP.NET and web development
Build web applications using ASP.NET, ASP.NET Core, HTML, JavaScript, and CSS.



Azure development
Azure SDK, tools, and projects for developing cloud apps and creating resources.



Node.js development
Build scalable network applications using Node.js, an asynchronous event-driven...



Data storage and processing
Connect, develop and test data solutions using SQL Server, Azure Data Lake, Hadoop ...



Office/SharePoint development
Create Office and SharePoint add-ins, SharePoint solutions, and VSTO add-ins usin...



Mobile & Gaming (5)



Mobile development with .NET

Build cross-platform applications for iOS, Android or Windows using Xamarin.



Game development with Unity

Create 2D and 3D games with Unity, a powerful cross-platform development...



Mobile development with JavaScript

Build Android, iOS and UWP apps using Tools for Apache Cordova.



Mobile development with C++

Build cross-platform applications for iOS, Android or Windows using C++.



Game development with C++

Use the full power of C++ to build professional games powered by DirectX...



Other Toolsets (3)



Visual Studio extension development

Create add-ons and extensions for Visual Studio, including new commands, code...



Linux development with C++

Create and debug applications running in a Linux environment.

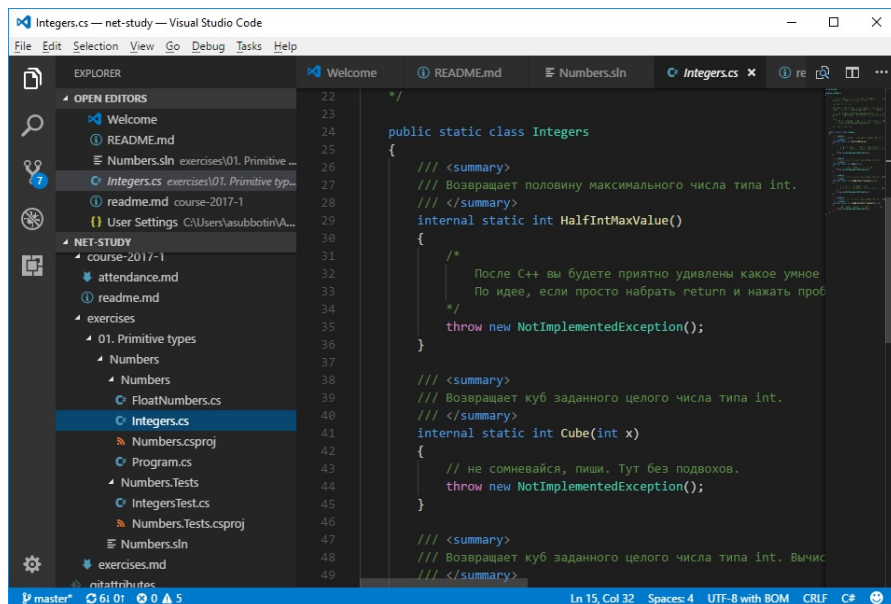


.NET Core cross-platform development

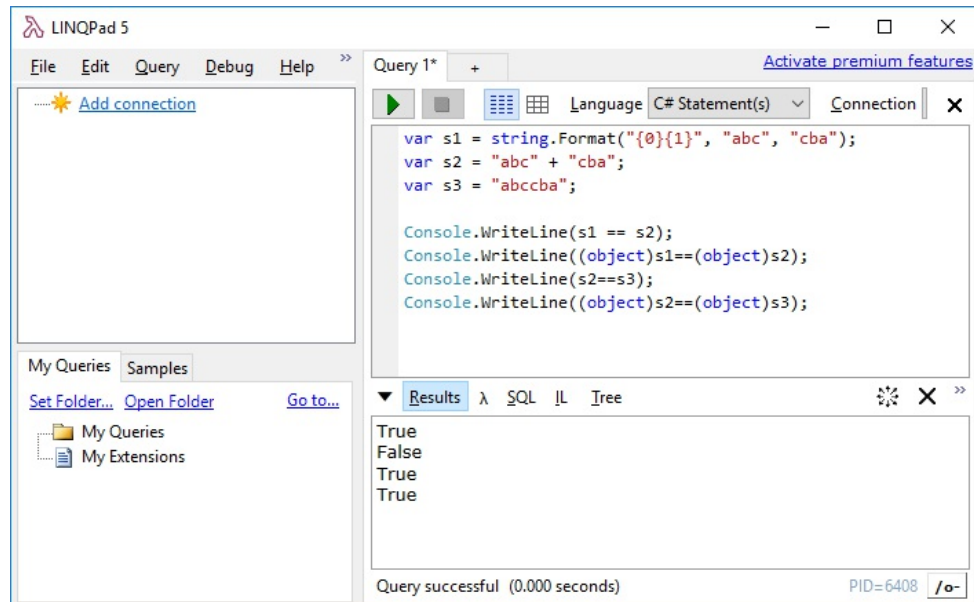
Build cross-platform applications using .NET Core, ASP.NET Core, HTML, JavaScript, and...



Visual Studio Code





LINQPad 5



Nuget

Система публикации сборок.


[Packages](#)
[Upload](#)
[Statistics](#)
[Documentation](#)
[Downloads](#)
[Blog](#)
[Sign in](#) | [Register](#)





Newtonsoft.Json 10.0.3

Json.NET

Json.NET is a popular high-performance JSON framework for .NET

Requires NuGet 2.12 or higher.



.NET CLI

```
PM> Install-Package Newtonsoft.Json -Version 10.0.3
```

Info

🕒 last updated 3 months ago

 Project Site License Info

 [Contact owners](#)

Report

Manual download

> Dependencies

Version History

Version	Downloads	Last updated
10.0.3 (current version)	1 783 896	3 months ago
10.0.2	3 368 472	5 months ago
10.0.1	10 192 688	6 months ago
9.0.1	8 935 855	22.06.2016
8.0.3	4 565 642	14.03.2016

+ Show more

Statistics

↓ 79 687 794 total downloads

1 783 896 downloads of latest version

32 672 downloads per day (avg)

[View full stats](#)

Owners



jamesnk

Authors

B Visual Studio:

- Tools -> NuGet Package Manager
- Right click на проекте -> Manage NuGet Packages

Добавление reference:

The screenshot shows the NuGet Package Manager window in Visual Studio. The left pane displays a list of installed packages, including Newtonsoft.Json, NUnit, EntityFramework, HtmlAgilityPack, jQuery, bootstrap, AutoMapper, and Microsoft.AspNet.Mvc. The right pane shows the details for the selected package, Newtonsoft.Json, including its version (4.5.11), description, and dependencies.

NuGet Package Manager: Abbyy.Online.ApiService

Package source: nuget.org

Newtonsoft.Json

Installed: 4.5.11 [Uninstall]

Version: Latest stable 10.0.3 [Update]

Options

Description

Json.NET is a popular high-performance JSON framework for .NET

Version: 10.0.3

Author(s): James Newton-King

License: <https://raw.githubusercontent.com/JamesNK/Newtonsoft.Json/master/LICENSE.md>

Date published: Sunday, June 18, 2017 (6/18/2017)

Project URL: <http://www.newtonsoft.com/json>

Report Abuse: <https://www.nuget.org/packages/Newtonsoft.Json/10.0.3/ReportAbuse>

Tags: json

Dependencies

.NETFramework.Version=v4.0
No dependencies

.NETStandard.Version=v1.3
Microsoft.CSharp (>= 4.3.0)
System.ComponentModel.TypeConverter (>= 4.3.0)
System.Runtime.Serialization.Formatters (>= 4.3.0)
System.Xml.XmlDocument (>= 4.3.0)
System.Runtime.Serialization.Primitives (>= 4.3.0)
NETStandard.Library (>= 1.6.1)

.NETFramework.Version=v2.0
No dependencies

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

☐ Do not show this again

Настройки nuget:

