NINNES

dotNet and Roslyn behind the Curtains

Germán Valencia - @XMachinarius

November 8, 2018

Growth Acceleration Partners

Table of contents

Introduction

What are we here for?

- NINNES project
- dotNet compilation and execution internals
- Roslyn internals
- Cool demo

NINNES Project

1. Bringing C# into NES game programming
2. Customizing the dotNet compilation process
3. Programming CPU-equivalent libraries

C# Code Tailoring

Cutting C# down to size

https://www.pexels.com/photo/fabric-scissors-needle-needles-scissors-461035/

C# Code Tailoring

1. Remove invalid instructions (and substitute them where possible)
2. Assist the programmer with keeping his C# code compatible with the NES CPU
3. Invalidate code that is otherwise perfectly fine

NES dotNet Shims Platform

Bridging the gap to the unknown

https://www.pexels.com/photo/architecture-bridge-fog-ocean-285283/

NES dotNet Shims Platform

1. NuGet Package exposing dotNet-friendly abstractions over the NES CPU, APU and PPU
2. F5 into a CLR-based rendering of the code, XNA-style

C# Compilation to the NES CPU

One Ace up the sleeve.

https://www.pexels.com/photo/actor-adult-business-cards-547593/

Why?

The main objective of this project is to showcase Roslyn's flexibility for custom scenarios with an extreme case study relative to the expected use cases on the usual day to day development, so as to produce a library of Roslyn customization examples covering a wide range of scenarios.

And for 1337 internet points, of course.

From Visual Studio to F5, dotNet behind the curtains

The show begins

From idea to execution, a play by Microsoft

https://www.pexels.com/photo/people-at-theater-713149/

C#, the Idiom and Culture

The programming language we all know and love.

BCL, the Acting Academy

- Base Class Library
- Basically, the System namespace and basic types
- Extended with Operating System service wrappers like Windows Forms and WPF via P/Invoke

Roslyn, Costumer Extraordinaire

- ”.Net Compiler Platform”, excepts everybody calls it Roslyn per the project Codename
- Developed as a Tour de Force to show the maturity of C# and dotNet
- ”Strengthening the ecosystem and becoming the best tooled language on the planet” - Mads Torgensen, C# designer

Roslyn, Costumer Extraordinaire

- Not only a compiler, a dotNet Compilation SDK
- Exposes multiple services to external code, allowing for extension and modification
- Used by Visual Studio for Windows, Mac and VS Code/Sublime Text/Atom... (Via OmniSharp)

MSIL, the Script

- Provides dotNet with ABI/Platform independence
- Machine-agnostic, Portable language
- JIT-ed into machine code

MSBuild, the Playwright

- Source of Truth for the dotNet project models
- Coordinator of project-associated tooling

RyuJIT, the Director

RyuJIT falls in the Just In Time model of dotNet execution as the translator from MSIL to native CPU code.

CLI, the Stage

- ECMA 335, standard since dotNet 1.0
- Open Standard describing the execution of dotNet/MSIL code
- Implemented by Microsoft with the CLR and by Xamarin with Mono

DLR, the Improv Guide

- Dynamic Language Runtime
- Extendes the CLI implementations to add support for dynamic languages like IronPython and IronRuby

dotNet Native, the Cameraman

Invokes RyuJIT ahead of time to produce a native yet CLI-dependant image for startup-time sensitive workloads

From Visual Studio to the NES

The Limits of the NES

The NES runs on a MOS6502 8-bit CPU implementation from Ricoh, albeit without the decimal/floating point mode enabled. This brings forth several limitations:

- No floating point calculations
- No multiplication or division implemented in hardware
- No Operating System
- Limited memory access capabilities

Removing dotNet from C#, 8 bits at a time

To be able to fit a C# program inside the NES some concessions must be made, including:

- No Garbage Collection
- No Threading
- No Operating System services
- No DLR
- No Reflection
- No Sockets/Communication

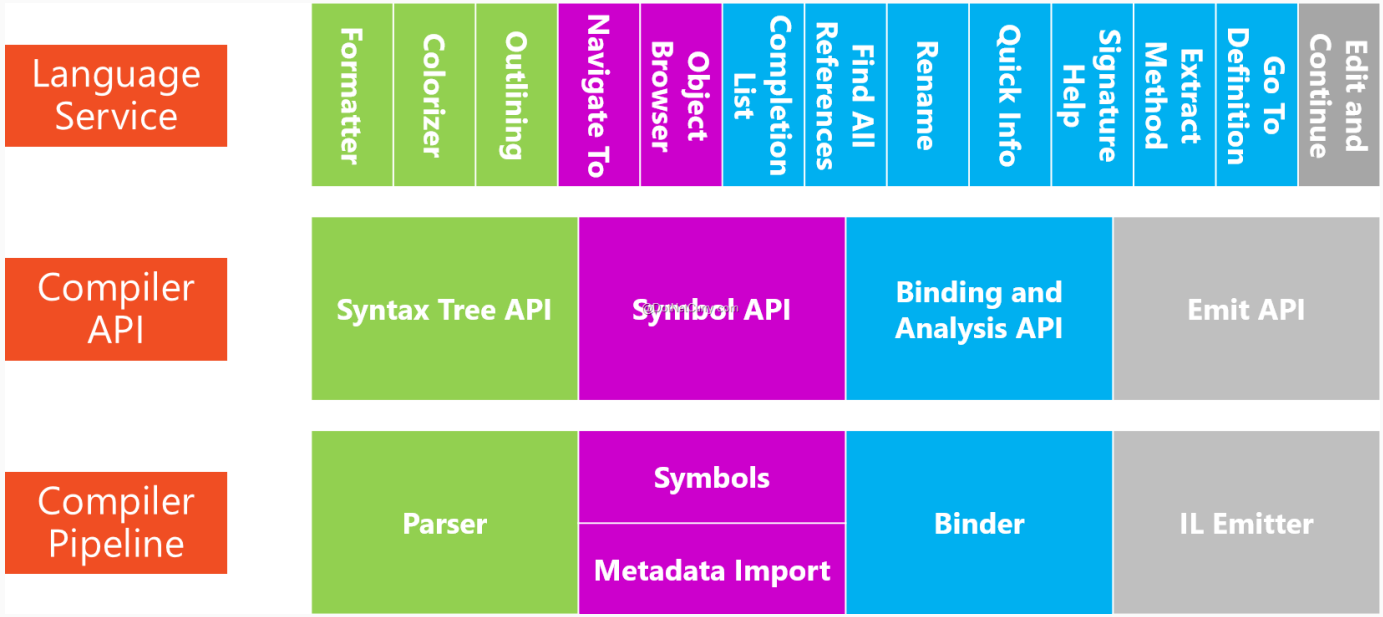
MSIL and 8-bit CPUs don’t mix

Bringing C# compilation output to a MOS6502-compatible format may entail

- Direct ASM transpilation from Roslyn C# ASTs
- C++ transpilation from Roslyn C# ASTs for compilation with a MOS6502-compatible compiler
- MSIL recompilation with LLILC
- Researching the work of the .Net Micro Framework

Roslyn is a Friend, not an Enemy

General Roslyn Services



Some of the services exposed by Roslyn

https://www.dotnetcurry.com/csharp/1258/dotnet-platform-compiler-roslyn-overview

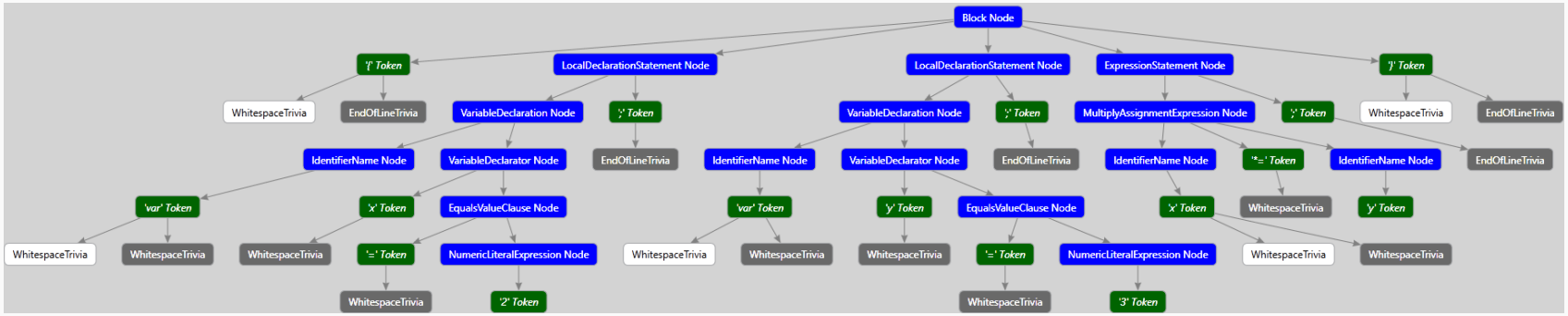
Compiler basics - What is code?

English is the language of the human, Assembler OpCodes is the language of the machine. Compilers must act as a bridge in between these two realms of knowledge, and they too have their own abstraction: The Abstract Syntax Tree.

Compiler basics - Code Sample

```
var x = 2;
var y = 3;
x *= y;
```

Compiler basics - Abstract Syntax Tree



A basic Roslyn AST

Own work

Roslyn Code Model

Roslyn Design Goals:

Speed

Analysis must be performed incrementally and in real time

Memory Economy

Visual Studio can already be daunting for some machines

Roslyn Basics - Code Diagnostics

Two words: CUSTOM SQUIGGLIES

DEMO

Thank you!

Germán Valencia
@XMachinarius
Growth Acceleration Partners


Microsoft
Inter Grupo
ruta n
MEDELLÍN
endava
NAREIA
Fundación
AVANET
UrulT
Ceiba
software
MASGLOBAL
CONSULTING
GROWTH
ACCELERATION
PARTNERS
Soluciones Móviles
areamovil
INTERFAZ
PODCAST
DA7OS
intelligent training
TEAM
INTERNATIONAL
Innovative Software Solutions
iT Synergy
Growing Innovation
.NET Conf CO
v2018