Analyzing Code in .NET

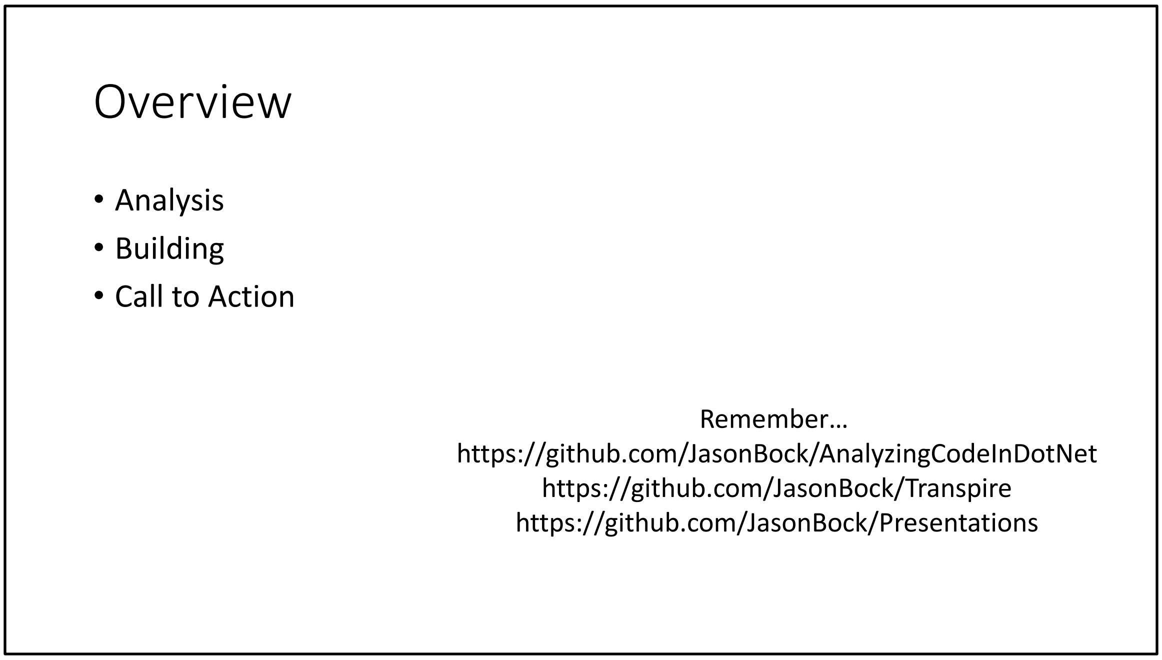
Jason Bock
Developer Advocate
Rocket Mortgage

Personal Info

- http://www.jasonbock.net
- https://www.twitter.com/jasonbock
- https://www.github.com/jasonbock
- https://www.youtube.com/c/JasonBock
- jason.r.bock@outlook.com

Downloads

https://github.com/JasonBock/AnalyzingCodeInDotNet
https://github.com/JasonBock/Transpire
https://github.com/JasonBock/Presentations

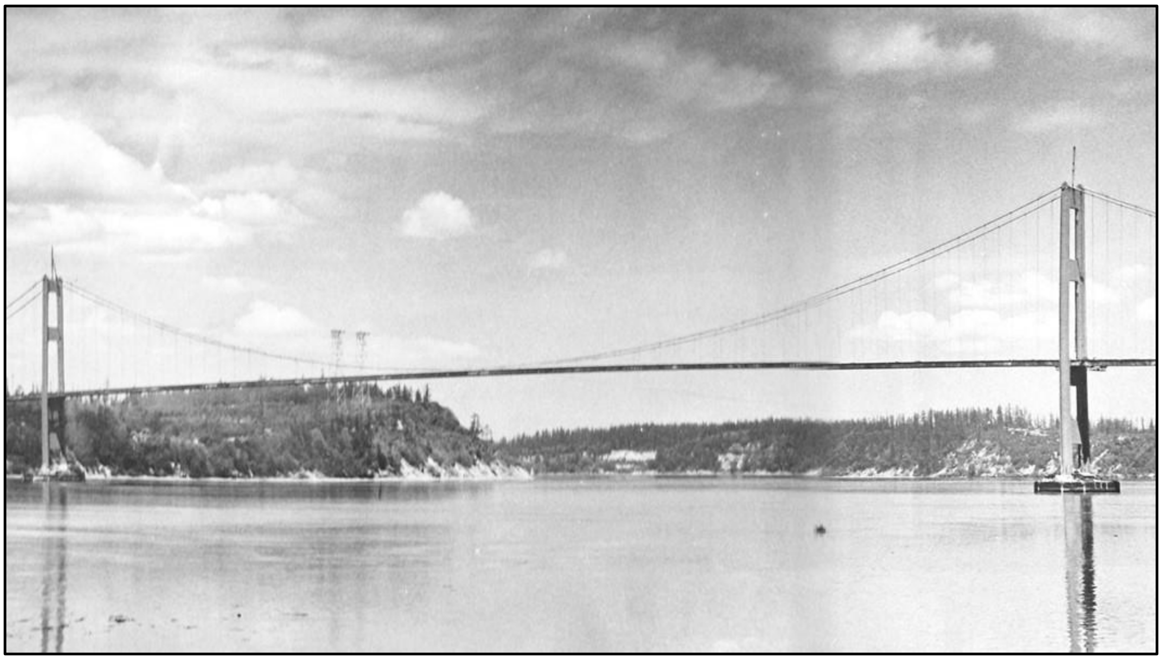


Overview

- Analysis
- Building
- Call to Action

Remember…
https://github.com/JasonBock/AnalyzingCodeInDotNet
https://github.com/JasonBock/Transpire
https://github.com/JasonBock/Presentations

Analysis



Have you ever failed in public? It can be awkward, embarrassing, and when you’re building a project, it can be damaging in various ways. We have the best of intentions, and we are smart, intelligent folks. But we're fallible. Think of the Tacoma Narrows bridge, a famous example of engineering gone wrong.

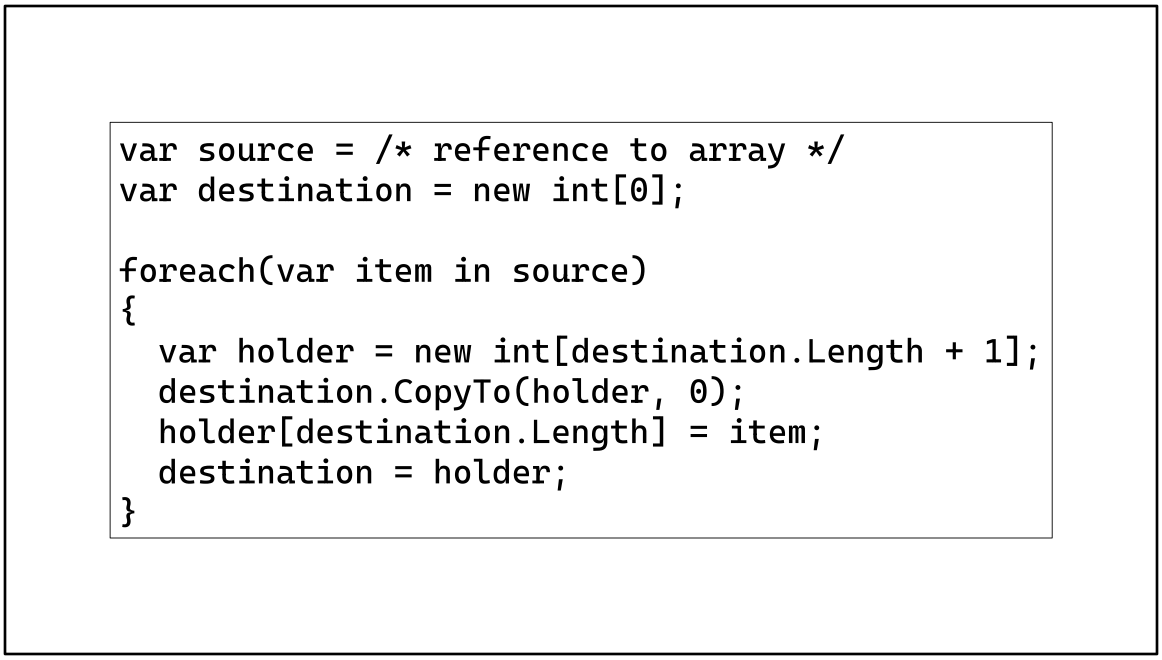
http://www.tacomanarrowsbridge.org/z1940-100.jpg

If there's a truism in code, it's that we'll write code that fails. And sometimes that happens spectacularly.

http://www.tacomanarrowsbridge.org/z1940-34.jpg

Failing in production is something we want to avoid. One technique is to do manual code reviews, either as an individual or as a group.

https://www.pexels.com/photo/photo-of-people-doing-handshakes-3183197/



```
var source = /* reference to array */
var destination = new int[0];

foreach(var item in source)
{
  var holder = new int[destination.Length + 1];
  destination.CopyTo(holder, 0);
  holder[destination.Length] = item;
  destination = holder;
}
```

Sometimes, it’s obvious when someone has written a train wreck. This code copies the contents of an array (source) to destination. But….all it needed to do was call:

source.CopyTo(destination, 0); // assume destination is the same size as source.

(Yes, this was code I found in a production app that had been live for at least 10 years. Copies of it were all over the place.)

Other times, it’s much harder to see the bug.

https://unsplash.com/photos/_OyMf5BbAxA

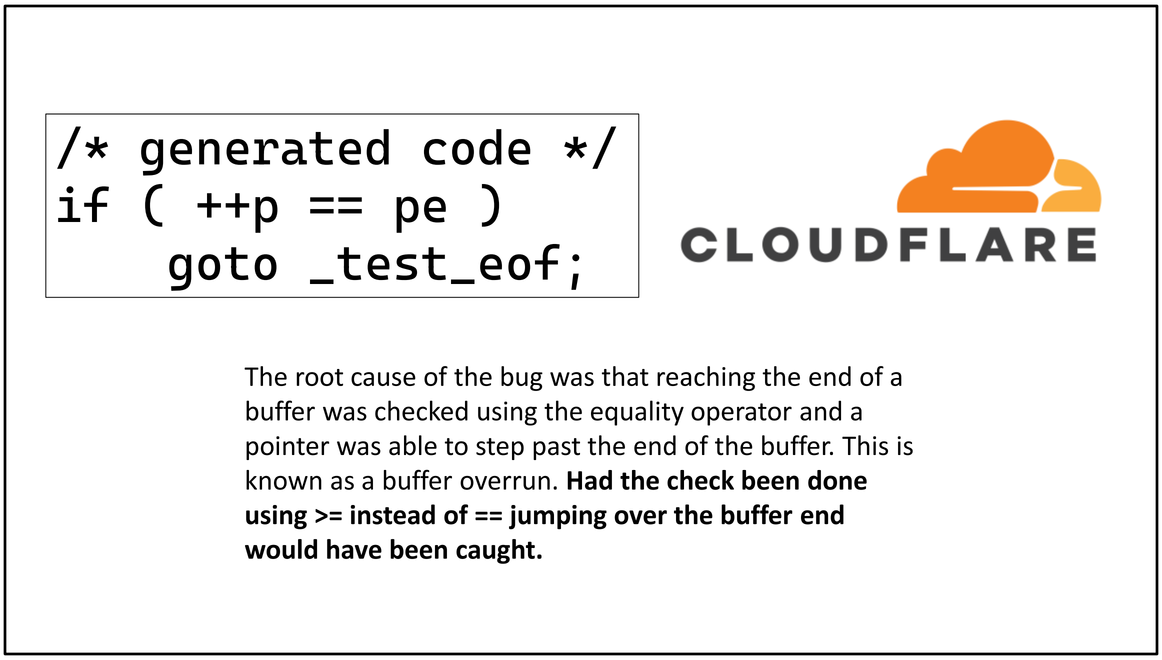
```
year = ORIGINYEAR; /* = 1980 */

while (days > 365)
{
  if (IsLeapYear(year))
  {
    if (days > 366)
    {
      days -= 366;
      year += 1;
    }
  }
  else
  {
    days -= 365;
    year += 1;
  }
}
```

"A bug in the internal clock driver related to the way the device handles a leap year affected Zune users," said the company in a statement. “That being the case, **the issue should be resolved over the next 24 hours as the time change moves to January 1, 2009**."

This code is from the Zune. Do you think you could’ve caught the error at first glance?

https://www.hanselman.com/blog/back-to-basics-explore-the-edge-cases-or-date-math-will-get-you
https://www.wired.com/2008/12/zune-freeze-res/



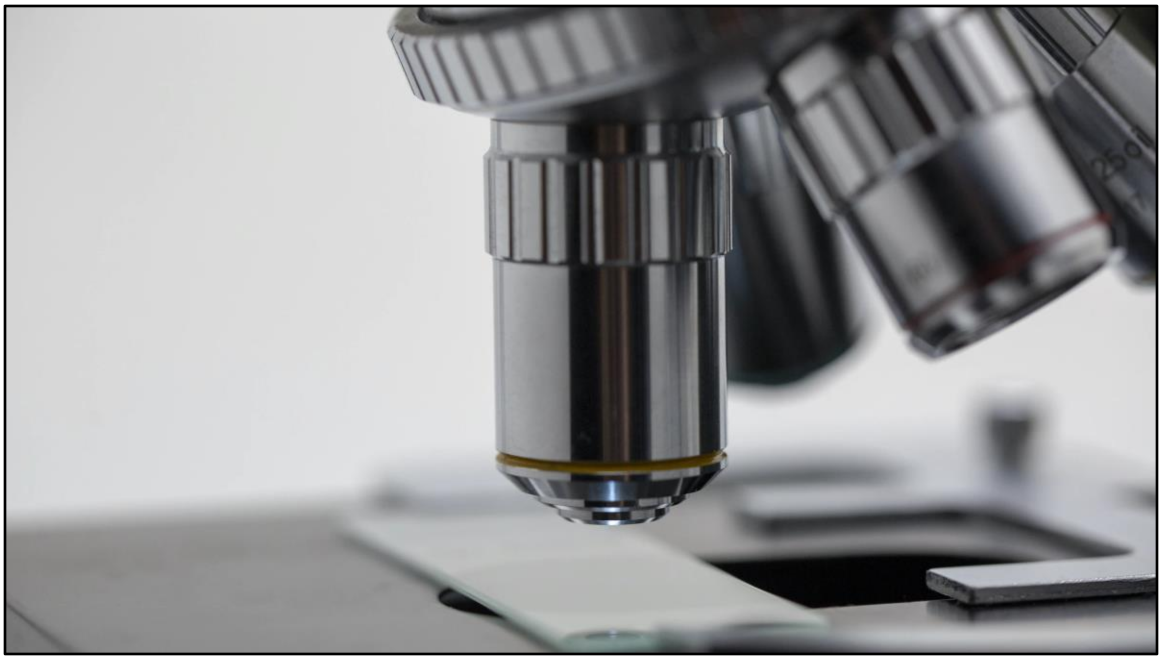
```
/* generated code */
if ( ++p == pe )
    goto _test_eof;
```

CLOUDFLARE

The root cause of the bug was that reaching the end of a buffer was checked using the equality operator and a pointer was able to step past the end of the buffer. This is known as a buffer overrun. **Had the check been done using >= instead of == jumping over the buffer end would have been caught.**

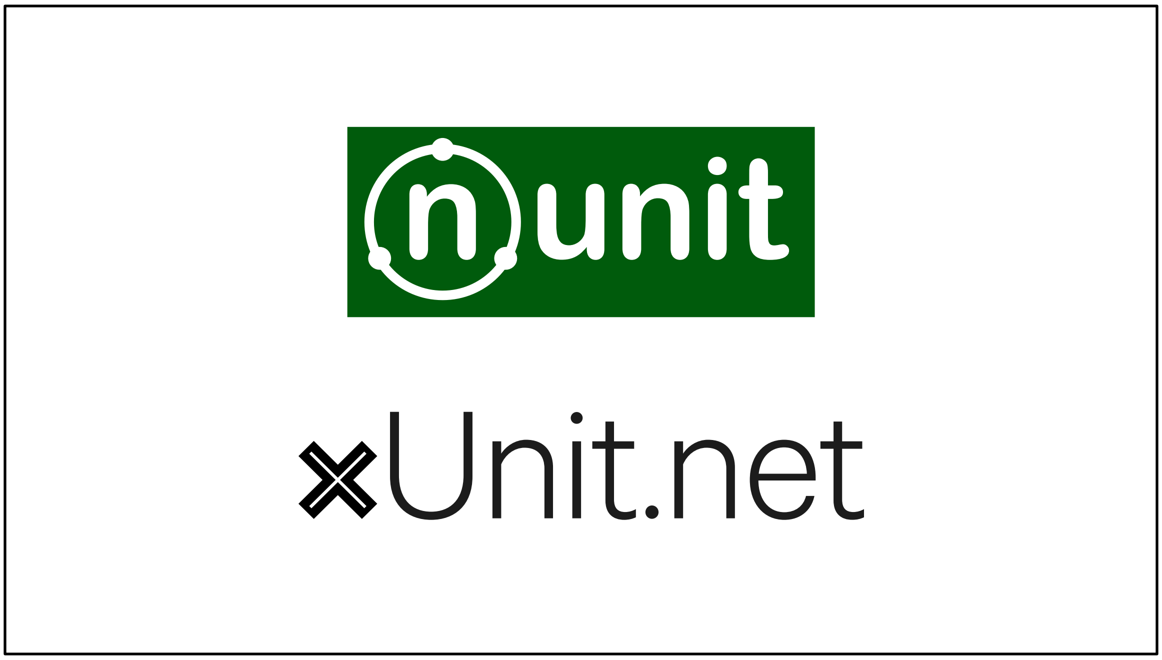
This is arguably even worse. Remember Cloudbleed?

https://blog.cloudflare.com/incident-report-on-memory-leak-caused-by-cloudflare-parser-bug/



We want to run our code through a microscope to let us find these errors as quickly as possible.

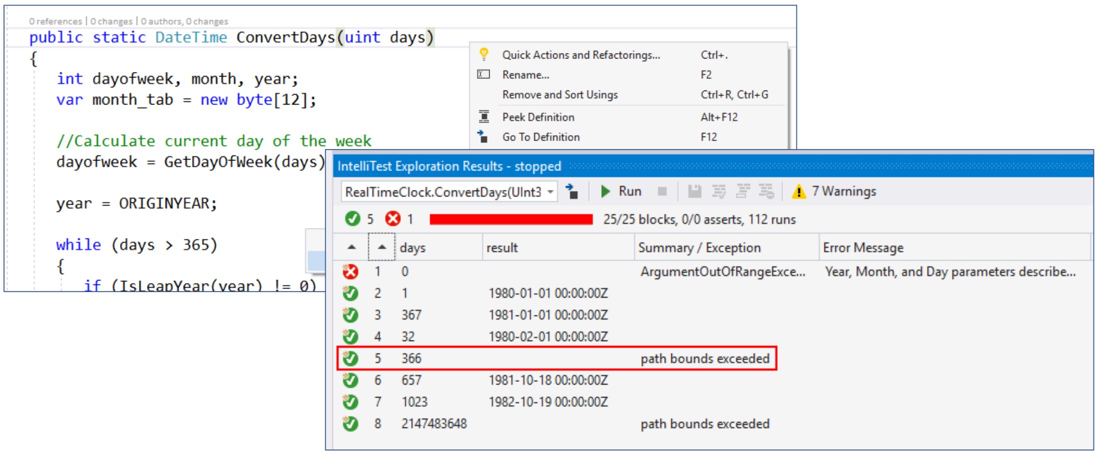
https://unsplash.com/photos/L9EV3OogLh0


nunit

xUnit.net

One technique is testing. We can (and should) write tests for our code. They help exercise our code to do what we think it should do, and if we change code, it can detect regressions and unexpected failures.

https://nunit.org/
https://xunit.net/



```
0 references | 0 changes | 0 authors, 0 changes
public static DateTime ConvertDays(uint days)
{
    int dayofweek, month, year;
    var month_tab = new byte[12];

    //Calculate current day of the week
    dayofweek = GetDayOfWeek(days)

    year = ORIGINYEAR;

    while (days > 365)
    {
        if (IsLeapYear(year) != 0)
```

Quick Actions and Refactorings... Ctrl+.
Rename... F2
Remove and Sort Usings Ctrl+R, Ctrl+G
Peek Definition Alt+F12
Go To Definition F12
IntelliTest Exploration Results - stopped
RealTimeClock.ConvertDays(UInt3
Run
7 Warnings

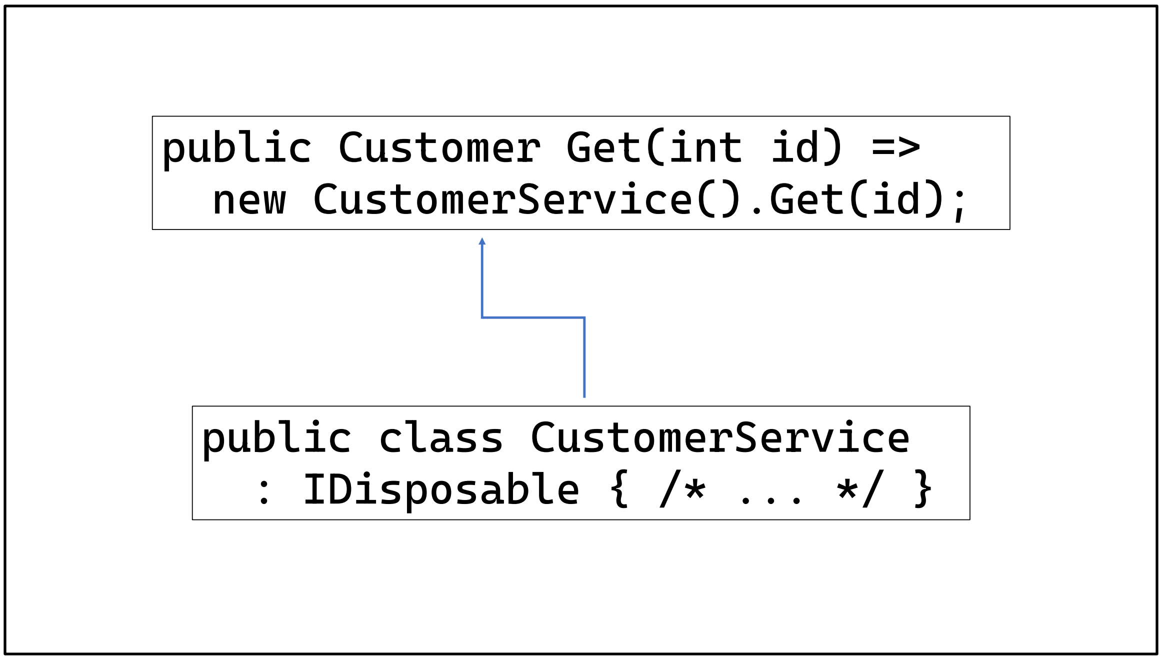
✓ 5 ✗ 1 25/25 blocks, 0/0 asserts, 112 runs

		days	result	Summary / Exception	Error Message
✗	1	0		ArgumentOutOfRangeExce...	Year, Month, and Day parameters describe...
✓	2	1	1980-01-01 00:00:00Z		
✓	3	367	1981-01-01 00:00:00Z		
✓	4	32	1980-02-01 00:00:00Z		
✓	5	366		path bounds exceeded	
✓	6	657	1981-10-18 00:00:00Z		
✓	7	1023	1982-10-19 00:00:00Z		
✓	8	2147483648		path bounds exceeded	

We can also use Intellitest to generate tests for us and exercise our code's boundary conditions (show what happens with the Zune bug). Note that this requires Enterprise SKU and it only works on code targeting the .NET Framework, so that's probably not doable for everyone, but it is an option.

I mentioned manual code reviews before. While they're useful, humans can be frustrated, or sick, and we can miss things.

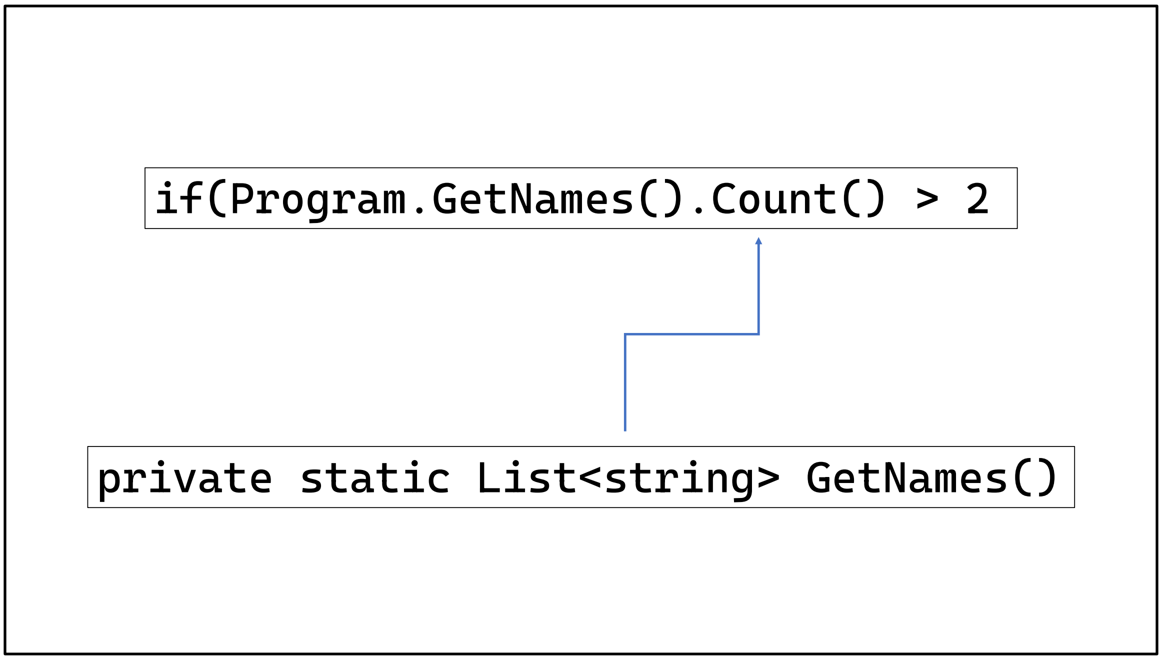
https://unsplash.com/photos/-2vD8lIhdnw



```
public Customer Get(int id) =>
  new CustomerService().Get(id);
```

```
public class CustomerService
  : IDisposable { /* ... */ }
```

Would you be able to see this issue out of the gate with not disposing something that implements IDisposable?

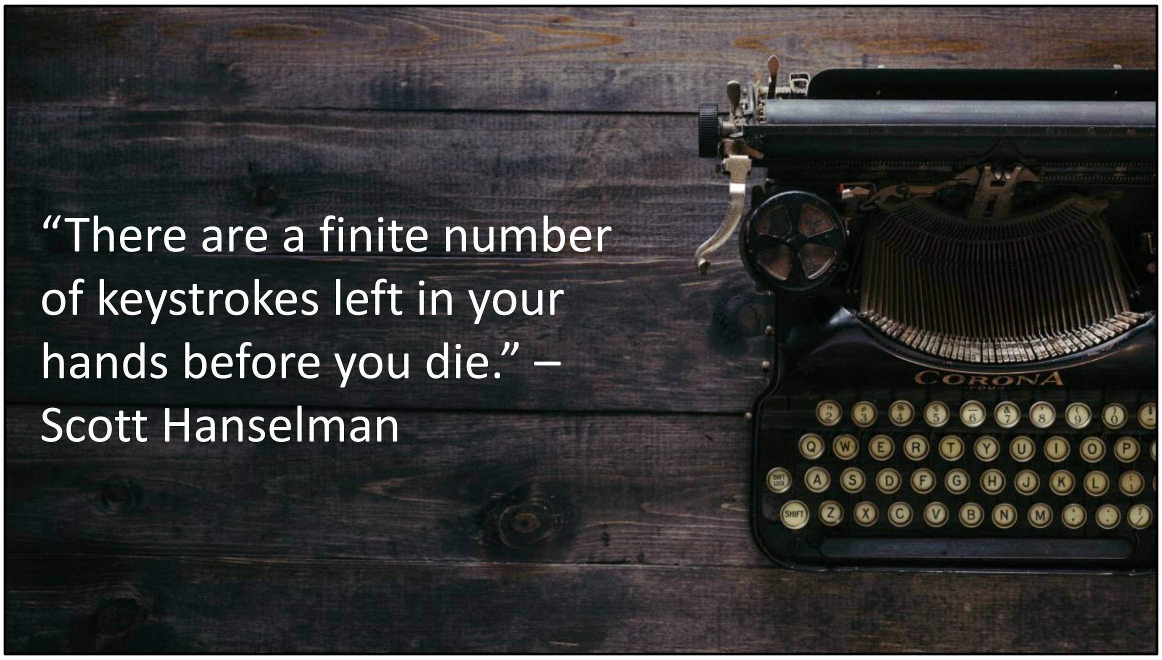


```
if(Program.GetNames().Count() > 2
```

```
private static List<string> GetNames()
```

Or this one? This should really use the ".Count" property, not the ".Count()" LINQ extension method.

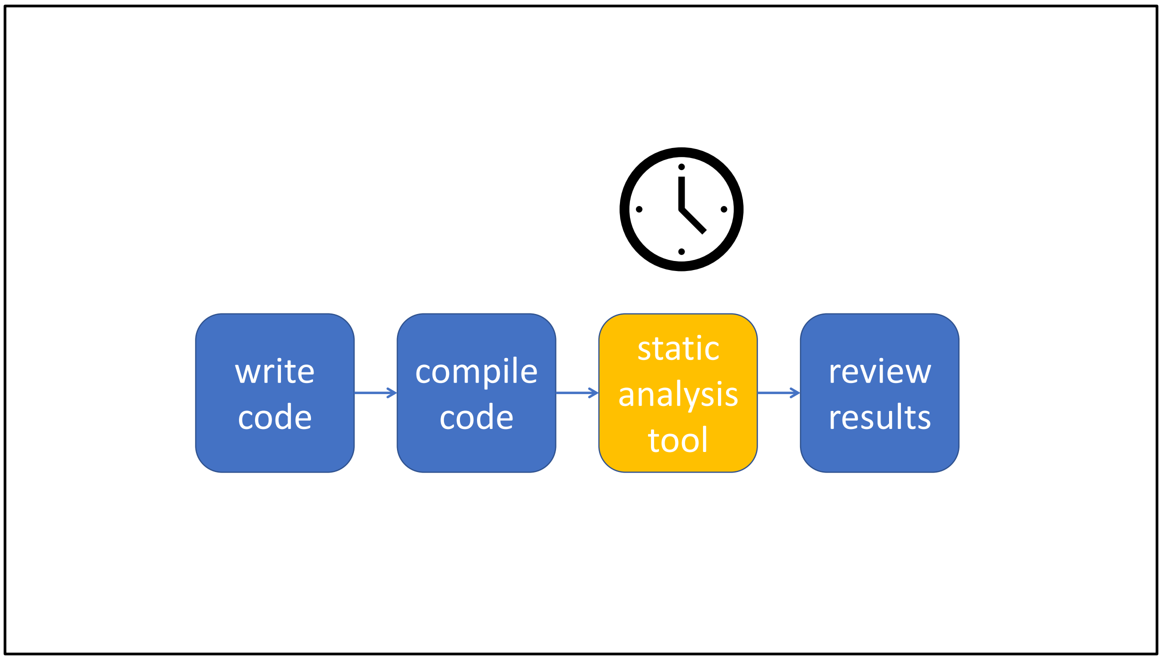
(And yes, in .NET 6, this



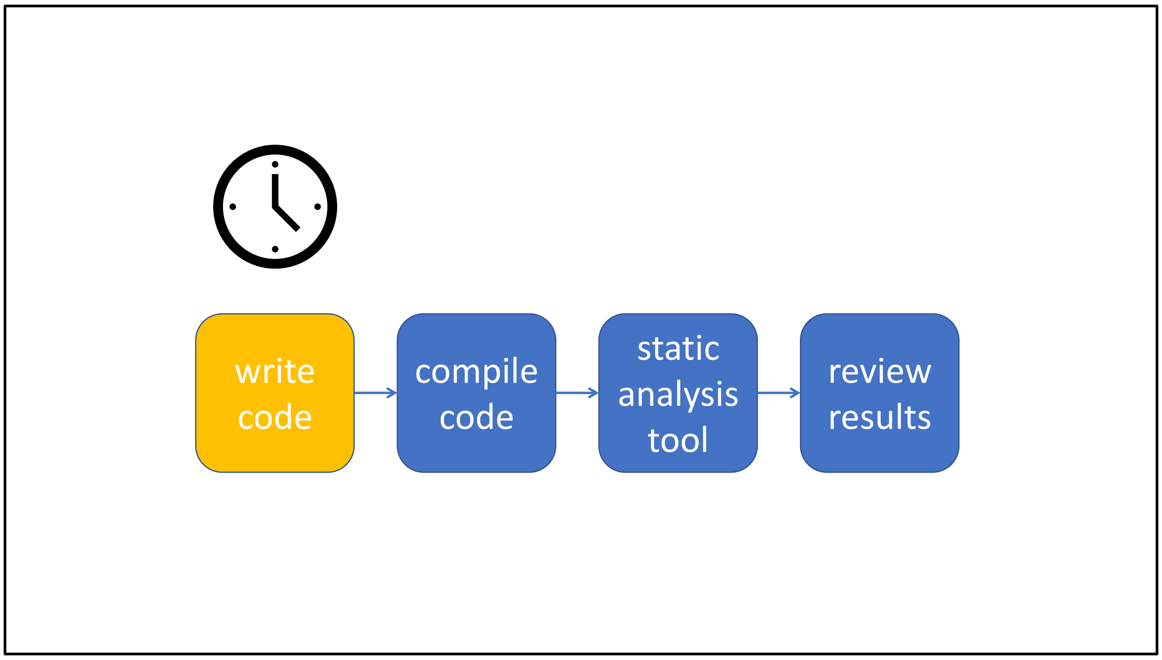
“There are a finite number of keystrokes left in your hands before you die.” – Scott Hanselman

We want to find issues as soon as we type code. The quote may be morbid, but it’s true!

https://www.hanselman.com/blog/do-they-deserve-the-gift-of-your-keystrokes
https://unsplash.com/photos/0gkw_9fy0eQ



There are tools out there that can analyze your code. But the problem is the timing. The usually won't do anything until you successfully compile code. They only works at the assembly level.



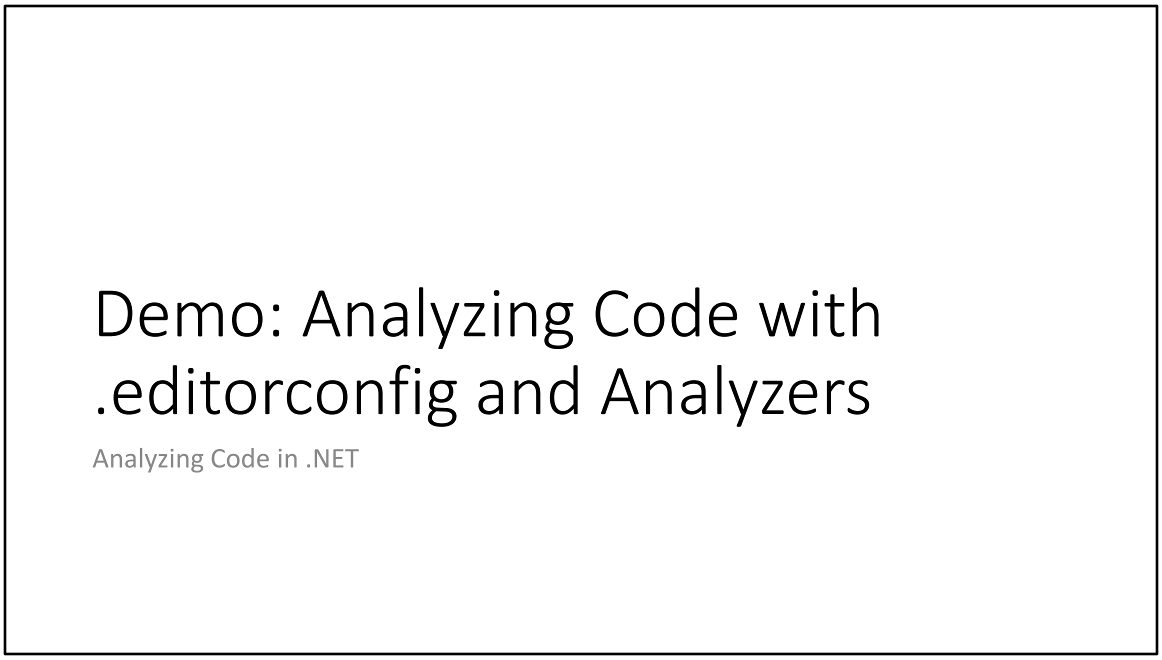
We want to move the needle, and do the analysis as soon as you make the mistake.

Thankfully we now have the Compiler API, which allows us to write analyzers that can find issues as soon as you type.

https://user-images.githubusercontent.com/46729679/109719841-17b7dd00-7b5e-11eb-8f5e-87eb2d4d1be9.png

- Roslyn Analyzers (https://github.com/dotnet/roslyn-analyzers)
- SonarQube (https://www.sonarqube.org)
- NDepend (https://www.ndepend.com)
- DotNetAnalyzers (https://github.com/DotNetAnalyzers)
- Code Cracker (https://code-cracker.github.io)
- CodeIt.Right (https://submain.com/codeit.right/)
- PVS Studio (https://www.viva64.com/en/pvs-studio/)
- Coverity (https://scan.coverity.com)
- Puma Scan (https://pumasecurity.io/)
- And many others…

Also: there are other products you can use. I won't show any of them here, but you are encouraged to check them out and evaluate them.



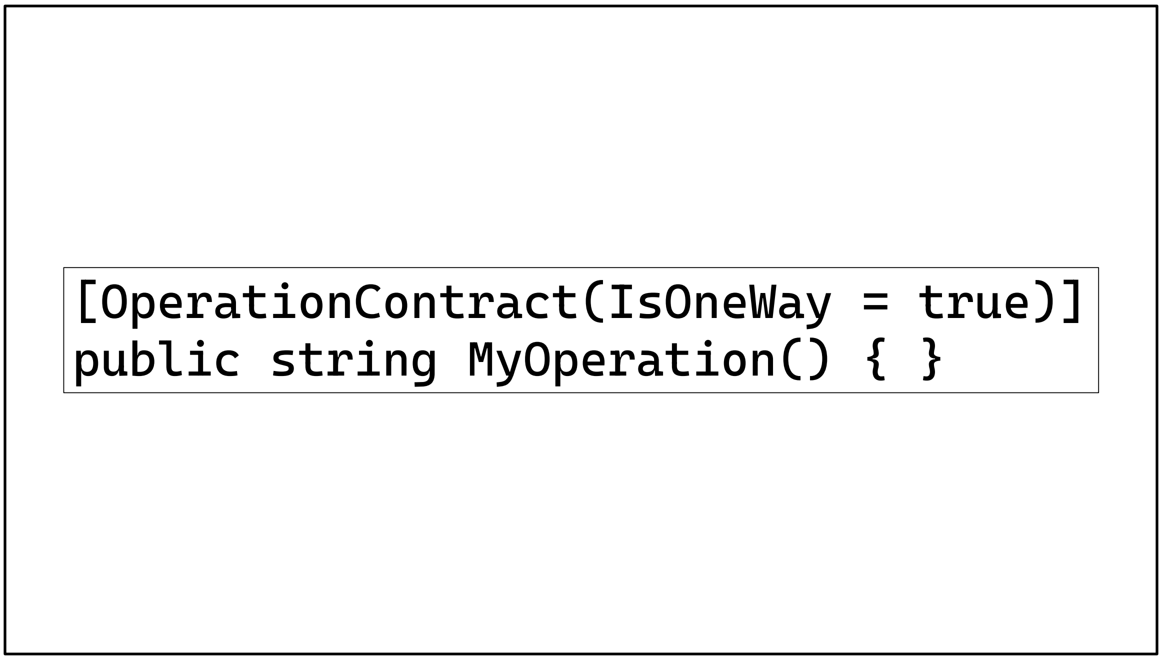
Demo: Analyzing Code with .editorconfig and Analyzers

Analyzing Code in .NET

Let’s see how .editorconfig can affect code styles, and what happens when you add analyzers to a project and what they can find.

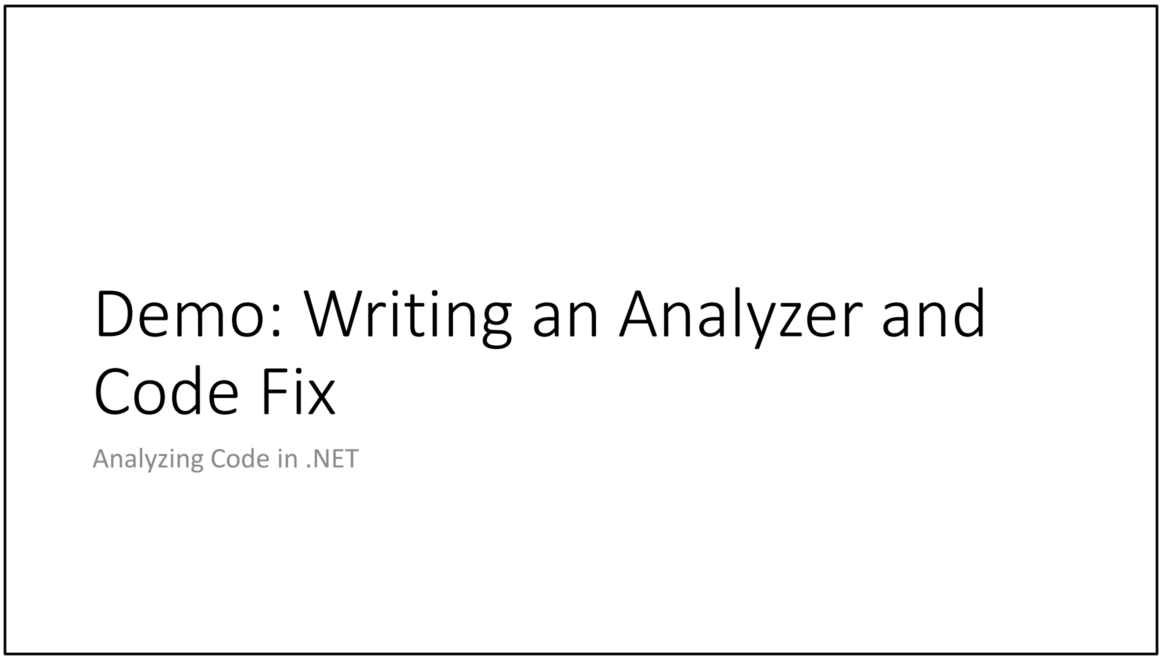
So….that's great. You can write your own .editorconfig, and you can get analyzers from Nuget packages. But….what if you want to write your own? You saw in the demo that the CSLA package had custom analyzers, so, how do you do that?

https://unsplash.com/photos/y02jEX_B0O0



```
[OperationContract(IsOneWay = true)]
public string MyOperation() { }
```

My personal “enlightening” moment was an issue I ran into with WCF. You could attribute methods as a “one-way” operation. This is essentially fire-and-forget – as soon as the service started running, the client could move on. This means that the method could only return void, but as you can see, this code returns a string. This code will build, and if you write a unit test that simply calls the method (i.e. it didn’t host it as a WCF service), it would pass. But as soon as you tried to run it for real, you’d get bad error. This is why we need the ability to write analyzers (and automate code fixes as well if possible)



Demo: Writing an Analyzer and Code Fix

Analyzing Code in .NET

Talk about Transpire, finding “new Guid()” and replacing with “Guid.NewGuid()”

Notes:
Get “.NET Compiler Platform SDK” from the VS Installer to get the Syntax Visualizer.
(There is Syntax Visualizer for Rider, not sure if anything exists for VS Code)
You can use the “Analyzer with Code Fix” project template, which is good to see how stuff is set up, but it’s not required.
http://roslynquoter.azurewebsites.net/ can be used to generate syntax trees, which may be useful for code fixes.

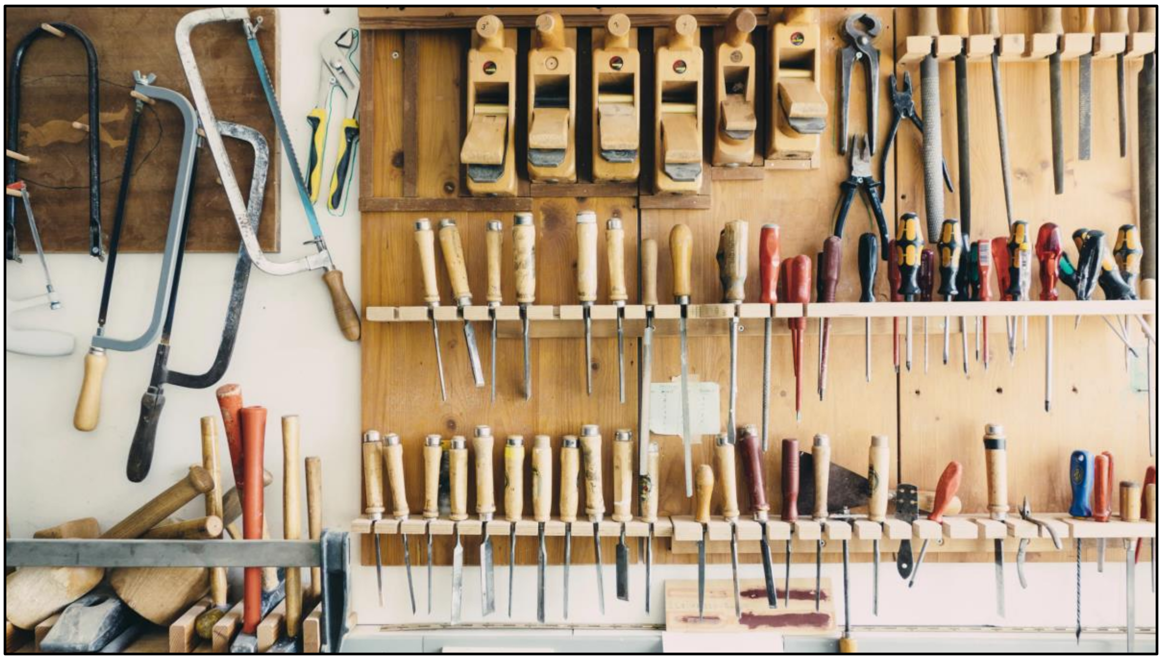
Call To Action

My “call to action” to you is this: climb the mountain. Using code analysis tools may be straightforward but writing them may seem daunting.

https://unsplash.com/photos/5x4U6InVXpc

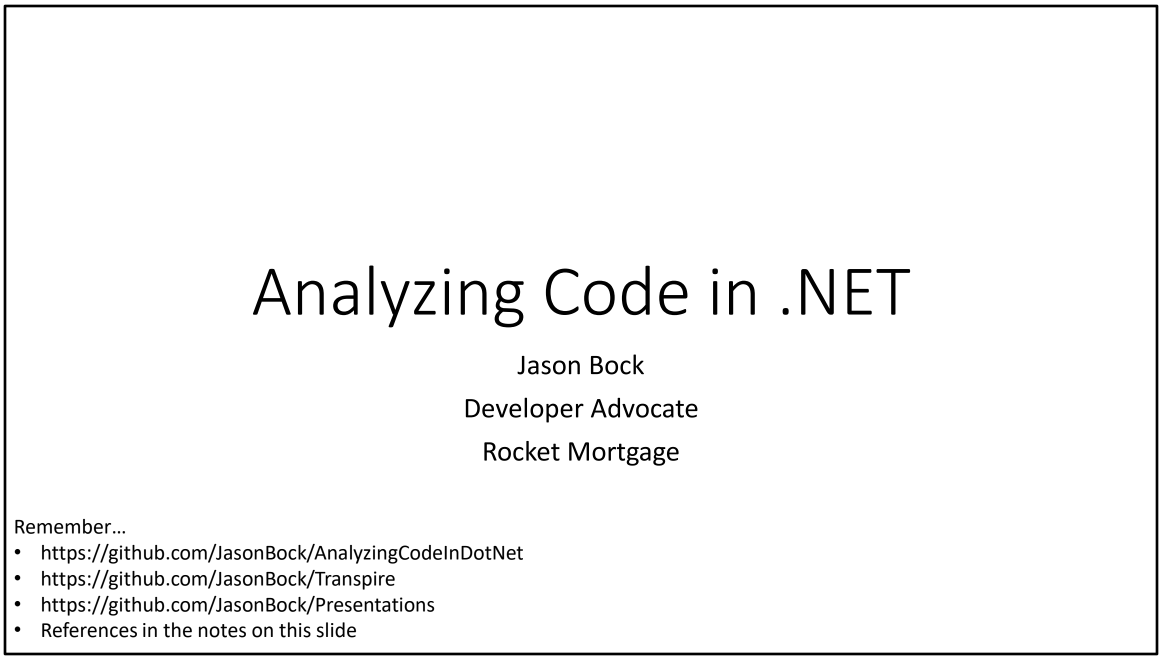
If you stick with it, you’ll have a great view of your code.

https://unsplash.com/photos/z0nVqfrOqWA



Using analyzers and being able to write your own – those are powerful tools. I strongly encourage you to use ones that exist and create your own for cases you're running into with your own code.

https://unsplash.com/photos/t5YUoHW6zRo



Analyzing Code in .NET

Jason Bock

Developer Advocate

Rocket Mortgage

Remember…
- https://github.com/JasonBock/AnalyzingCodeInDotNet
- https://github.com/JasonBock/Transpire
- https://github.com/JasonBock/Presentations
- References in the notes on this slide

## References

* [Overview of .NET source code analysis](https://docs.microsoft.com/en-us/dotnet/fundamentals/code-analysis/overview)
* [The Roslyn analyzers I use in my projects](https://www.meziantou.net/the-roslyn-analyzers-i-use.htm)
* [ANALYZING CODE FOR ISSUES IN .NET 5](https://dotnettips.wordpress.com/2021/01/14/analyzing-code-for-issues-in-net-5/)
* [Static analysis for .NET 5](https://github.com/dotnet/runtime/issues/30740)
* [A Few Billion Lines of Code Later: Using Static Analysis to Find Bugs in the Real World](https://m-cacm.acm.org/magazines/2010/2/69354-a-few-billion-lines-of-code-later/fulltext)
* [Awesome Roslyn](https://github.com/ironcev/awesome-roslyn/blob/master/readme.md)
* [Welcome to roslyn-analyzers documentation](https://roslyn-analyzers.readthedocs.io/en/latest/)
* [Treat Warnings As Errors. I Mean It. You Need to Do This](https://blog.submain.com/treat-warnings-errors/)

* [Raffaele Rialdi — Write your own C# static code analysis tool to drive business rules](https://www.youtube.com/watch?v=HT7k3Qm4uFY)
* [16 New Code Analysis, Testing and Debugging Tools For Visual Studio 2017](https://visualstudiomagazine.com/articles/2018/05/01/vs-analysis-tools.aspx)
* [.NET Development Using the Compiler API](https://www.apress.com/us/book/9781484221105)
* [Automating Code Reviews in .NET](https://magenic.com/media/2669/automating-code-reviews-in-net.pdf)
* [Roslyn Analyzer IDs Problem .NET Core, .NET Standard APIs](https://visualstudiomagazine.com/articles/2017/11/07/api-analyzer.aspx)
* [Rigging up Roslyn Analyzers in .NET Core](https://blogs.msdn.microsoft.com/devfish/2017/11/09/rigging-up-roslyn-analyzers-in-net-core/)
* [Your New Virtual API Review Assistant](https://channel9.msdn.com/coding4fun/blog/Your-New-Virtual-API-Review-Assistant)
* [Code Analysis, Profiling and Refactoring Tools for Visual Studio 2017](https://visualstudiomagazine.com/Articles/2017/10/01/Code-Analysis.aspx)
* [Hands-free Security Scanning within .NET Applications](http://rion.io/2017/10/02/hands-free-security-scanning-within-net-applications/)
* [Amazon Says One Engineer's Simple Mistake Brought the Internet Down](https://gizmodo.com/amazon-says-one-engineers-simple-mistake-brought-the-in-1792907038)
* [Summary of the Amazon S3 Service Disruption in the Northern Virginia (US-EAST-1) Region](https://aws.amazon.com/message/41926)
* [Everything You Need to Know About Cloudbleed, the Latest Internet Security Disaster](http://gizmodo.com/everything-you-need-to-know-about-cloudbleed-the-lates-1792710616)
* [Incident report on memory leak caused by Cloudflare parser bug](https://blog.cloudflare.com/incident-report-on-memory-leak-caused-by-cloudflare-parser-bug/)
* [Windows Azure storage issue: Expired HTTPS certificate possibly at fault](http://www.zdnet.com/article/windows-azure-storage-issue-expired-https-certificate-possibly-at-fault/)
* [Windows Azure Service Disruption from Expired Certificate](https://azure.microsoft.com/en-us/blog/windows-azure-service-disruption-from-expired-certificate/)
* [Back to Basics: Explore the Edge Cases or Date Math will Get

You](https://www.hanselman.com/blog/back-to-basics-explore-the-edge-cases-or-date-math-will-get-you)
* [Zune bug explained in detail](https://techcrunch.com/2008/12/31/zune-bug-explained-in-detail/)