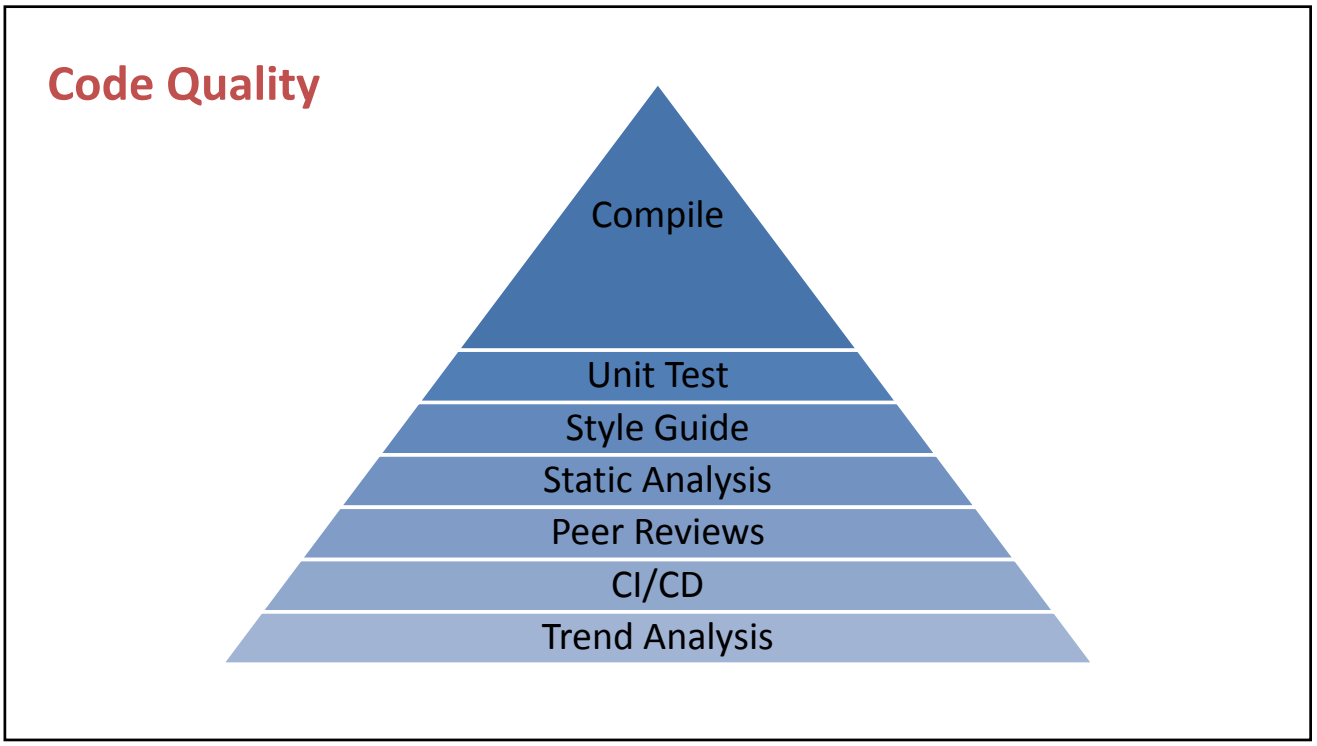

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS
San Diego
Improving code quality with Static Analysis
Jim Wooley
Solution Architect
Slalom
Level: Intermediate, etc.
Code Again for the First Time!
25 YEARS OF CODING INNOVATION


Code Quality
Compile
Unit Test
Style Guide
Static Analysis
Peer Reviews
CI/CD
Trend Analysis

Why use

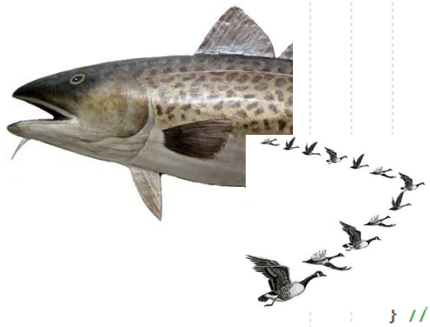
- Standardize within team
- Improve maintainability
- Allow code review to focus on domain/logic issues
- Detect common errors not caught by compiler
- Fix code smells

Fixing Code Smells

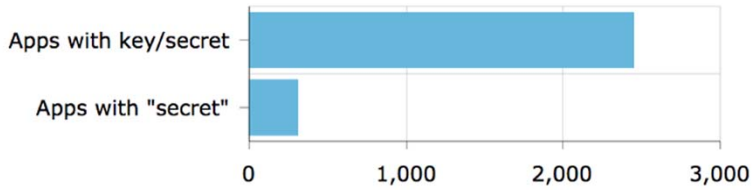
```
void ProcessCustomers(dynamic customers)
{
    if (customers != nu
    {
        foreach(var c i
        {
            DoSomething
            foreach (va
            {
                DoSomet
                if (o.t
                {
                    //
                }
                else
                {
                    DoS
                    whi
                    {
                    }
                }
            } // foreac
        } // foreach customers
    } // If customers != null
} // ProcessCustomers
```

```
void ProcessBetter(dynamic customers)
{
    if (customers == null) return;
    foreach (var c in customers)
    {
        DoSomethingComplex(c);
        foreach (var o in c.Orders)
        {
            ProcessOrder(o);
        }
    }
}

void ProcessOrder(dynamic order)
{
    DoSomethingComplex(order);
    if (order.total <= 1000)
    {
        DoSomethingComplex(order.OrderDetails);
        return;
    }
}
```



Detect security vulnerabilities



Out of 16,000 apps — apps with keys or secrets

https://hackernoon.com/we-reverse-engineered-16k-apps-heres-what-we-found-51bdf3b456bb#.eechvyizf

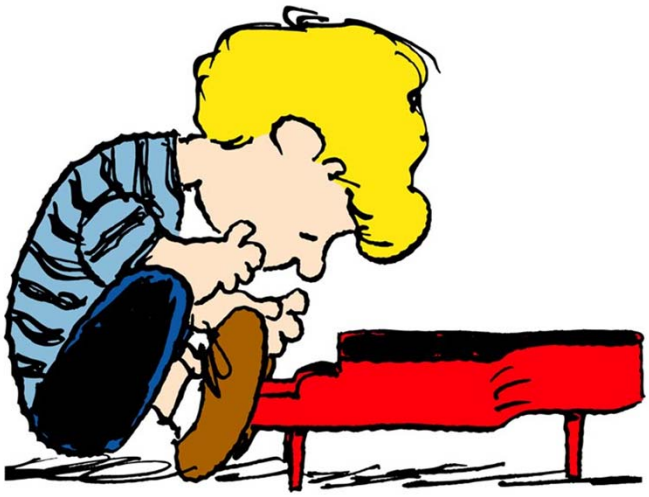
Detect domain specific errors

```
public class TestControllerMisnamed : Controller
{
    public TestControllerMisnamed()
    {

    }

    // GET: Test
    public ActionResult Index()
    {
        return View();
    }
}
```

Recommend best practices



Detect maintainability issues

Code Metrics Results

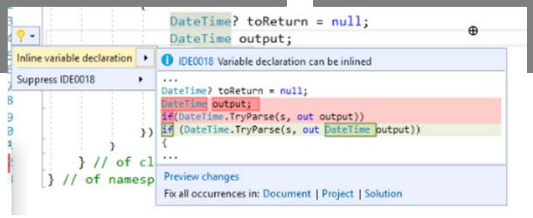
Filter: None Min: Max:

Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
One or more projects were skipped. Code metrics are available only for C#, Visu					
ApiDll (Debug)	69	1,945	4	1,233	12,341
APITests (Debug)	75	1,009	5	533	3,918
Common (Debug)	73	660	3	263	1,232
CommonControls (Debug)	84	223	1	104	471
Data (Debug)	92	2,837	2	211	3,412
Documentation\Modeling (Debug)	84	3,249	2	80	5,104
Email (Debug)	68	25	1	28	53
InternalManagementAppWeb (Debug)	86	1,427	4	648	3,187
KofaxImport\AzureUpload (Debug)	85	234	9	126	573
KofaxImport\AzureUpload.Core (Debug)	76	99	1	34	174
KofaxImport\FileImport (Debug)	75	97	1	80	319
LetterModels (Debug)	93	129	1	13	131
Models (Debug)	93	3,159	2	95	3,271
Tests (Debug)	83	234	1	99	483
WebJobs\AuditJob (Debug)	71	93	1	114	713
WebJobs\NivDocCleanup (Debug)	86	26	1	39	36
WebJobs\ScheduledTasks (Debug)	86	20	1	23	38
WebJobs\SendEmails (Debug)	89	16	1	15	21

Recommend modernized syntax

```
DateTime? toReturn = null;
DateTime output;
if(DateTime.TryParse(s, out output))
{
    toReturn = output;
}
return toReturn;
```

```
DateTime? toReturn = null;
if (DateTime.TryParse(s, out DateTime output))
{
    toReturn = output;
}
return toReturn;
```



13

Tools

- JsLint/JsHint
- CssLint/ScssLint/PostCSS/StyleLint
- SonarQube/Sonargraph/SonarLint
- Roslyn/FxCop/StyleCop/Code Cracker
- Code Rush/Resharper
- Your favorite tool?

https://en.wikipedia.org/wiki/List_of_tools_for_static_code_analysis

JavaScript – JsLint/JsHint

```
function main() {
  return 'Hello, World!';
  var x = 1;
  if (x == "2")
  {
    console.log(X)
  }
}
```

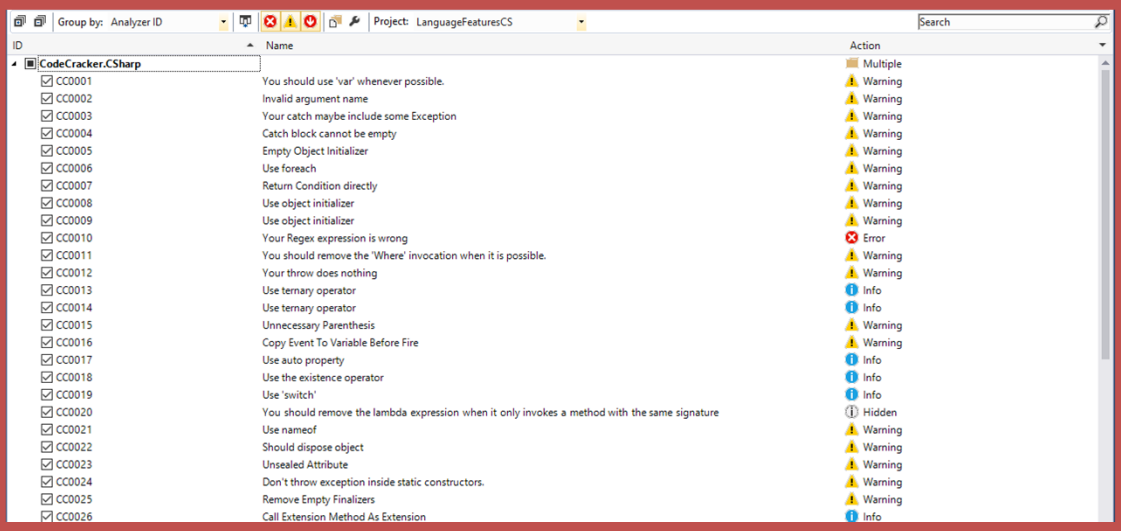
```
1  function main() {
2    return 'Hello, World!';
3    var x = 1;
4    if (x == "2")
5    {
6      console.log(X)
7    }
8  }
```

```
function main() {
    "use strict";
    var x = 1;
    if (x === 2) {
        console.log(x);
    }
}
```


Demo

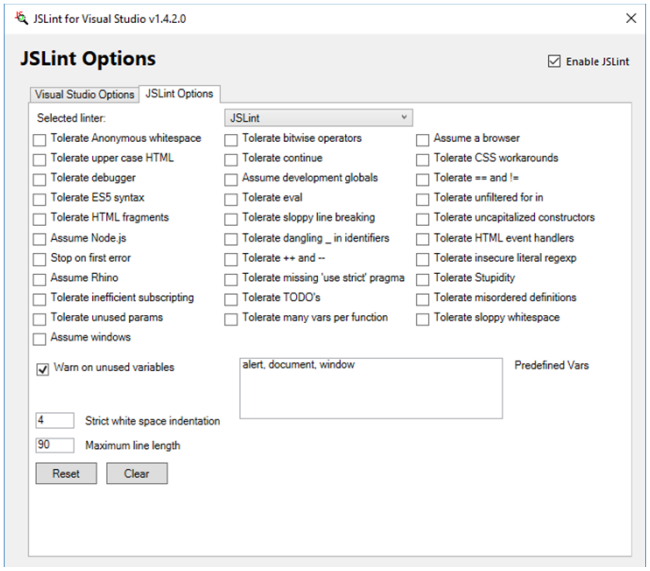
JSLint, CSSLint, Code Cracker

Extensible and Customizable



slalom.com

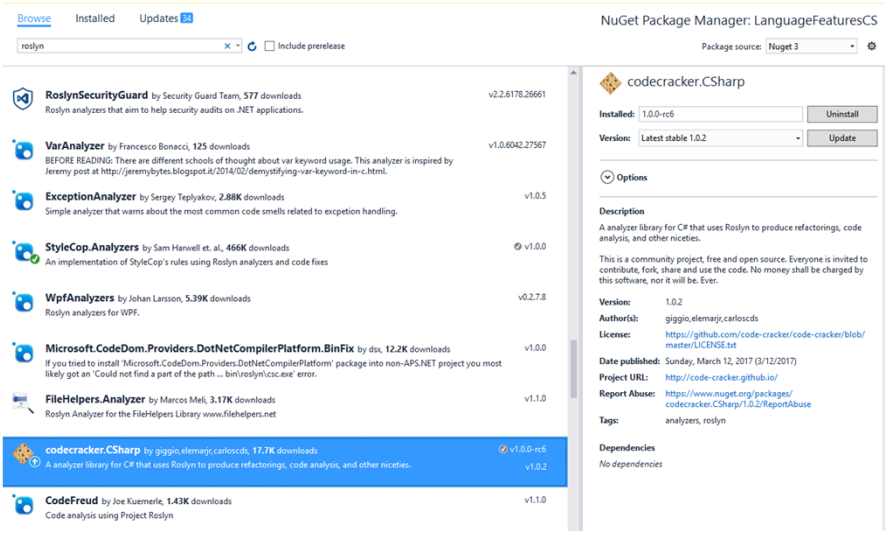
Configuring jsLint



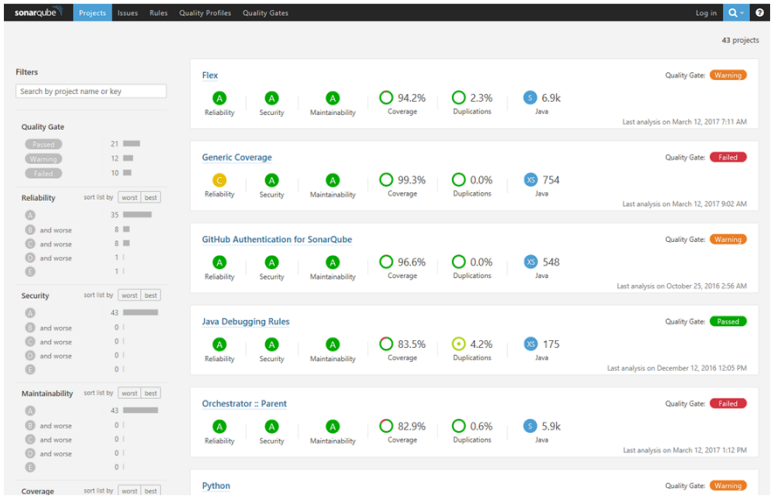
.jshintrc

```
{
  "undef": true,
  "unused": true,
  "globals": {
    "MY_GLOBAL": true
  }
}
```

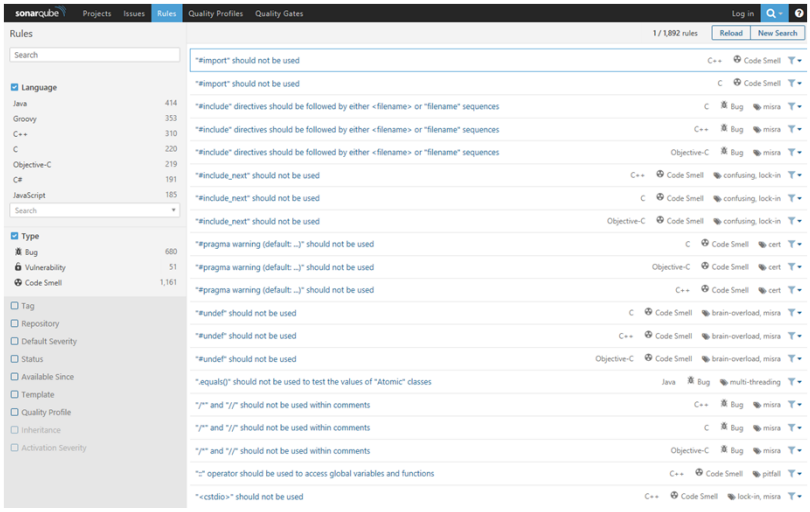
Adding Analyzer from Nuget



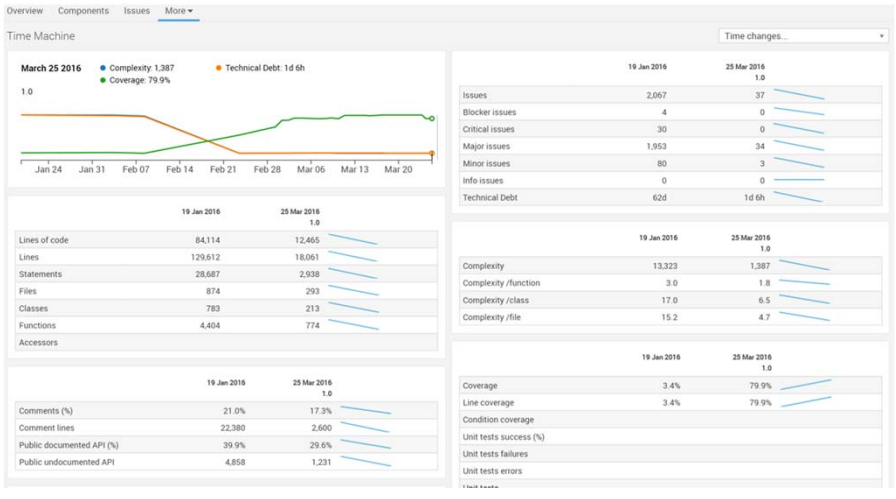
SonarQube



Sonar Rules



Sonar improvements



Detect domain specific errors

```
public class TestControllerMisnamed : Controller
{
    public TestControllerMisnamed()
    {

    }

    // GET: Test
    public ActionResult Index()
    {
        return View();
    }
}
```

Creating your own Roslyn analyzer and Code Fix

```
private void Analyzer(SymbolAnalysisContext context)
{
    var symbol = (INamedTypeSymbol)context.Symbol;
    if (symbol.BaseType == null) return;

    if ((symbol.BaseType.Name == "Controller"
      || symbol.BaseType.Name == "ApiController")
      && !symbol.Name.EndsWith("Controller"))
    {
        var diagnostic = Diagnostic.Create(
            Rule, symbol.Locations[0], symbol.Name);
        context.ReportDiagnostic(diagnostic);
    }
}
```

Custom Diagnostic - CodeFixProvider

```
private async Task<Document> MakeEndInControllerAsync(
  Document document,
  TypeDeclarationSyntax typeDecl,
  CancellationToken cancellationToken)
{
  var identifierToken = typeDecl.Identifier;
  var originalName = identifierToken.Text;

  var nameWithoutController = Regex.Replace(
      originalName, "controller", String.Empty, RegexOptions.IgnoreCase);
  var newName = nameWithoutController + "Controller";

  var newIdentifier = SyntaxFactory.Identifier(newName)
      .WithAdditionalAnnotations(Formatter.Annotation);

  var newDeclaration = typeDecl.ReplaceToken(identifierToken, newIdentifier);

  var root = await document.GetSyntaxRootAsync();
  var newRoot = root.ReplaceNode(typeDecl, newDeclaration);
  var newDocument = document.WithSyntaxRoot(newRoot);
  return newDocument;
}
```

Create ESLint rule

```
module.exports = {
  meta: {
    docs: {
      description: "disallow unnecessary semicolons",
      category: "Possible Errors",
      recommended: true
    },
    fixable: "code",
    schema: [] // no options
  },
  create: function(context) {
    return {
      // callback functionshttp://eslint.org/docs/developer-guide/working-with-rules
    };
  }
};
```

http://eslint.org/docs/developer-guide/working-with-rules

Key Principals

- Start Early
- Run Often
- Customize for your needs
- Augment SDLC
- Focus Code Review on domain issues not code smells

"The key takeaway is to use the tools available to find any defect you can – for even if you don't, your adversaries will."
- Travis Smith

28

Q & A

https://github.com/jwooley/Analyzers

https://github.com/jwooley/RoslynLabs

https://jwooley.github.io

http://www.thinqling.com

http://www.twitter.com/jimwooley