

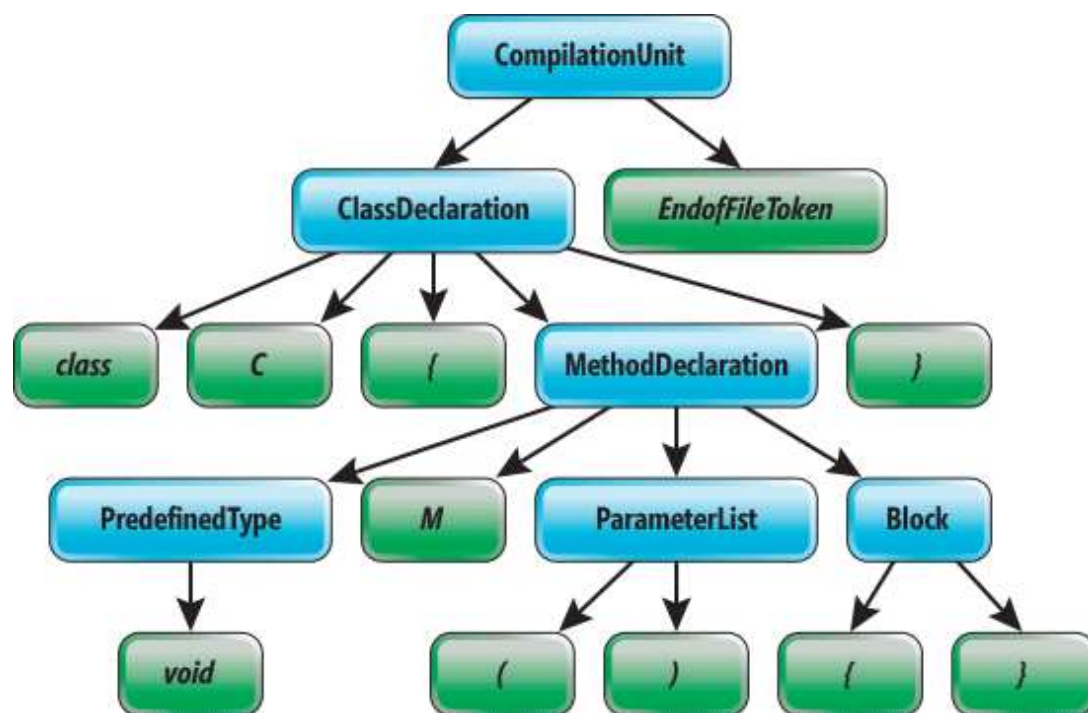
Roslyn

Очевидные неочевидности

Кугушев Александр



Ожидание

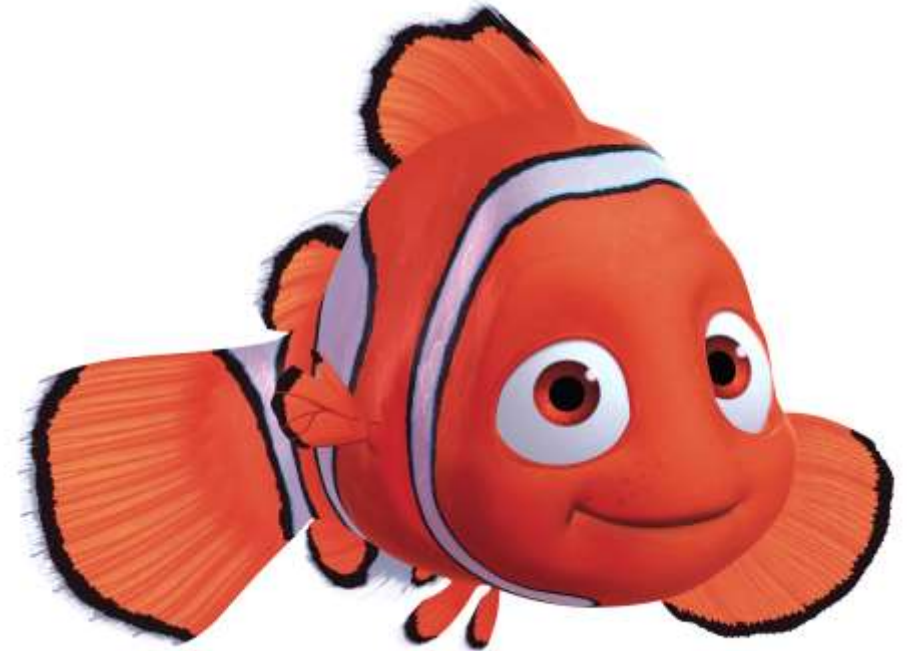


Реальность

[illegible]

Цели

- Рассмотреть топ неочевидных особенностей CodeAnalysis
- Синхронизировать свое мышление со спецификой рассматриваемого API
- Написать шпаргалку молодого аналитаторописателя



Service Locator Fixer

До

```
class Class
{
    public void HelloWorld()
    {
        ServiceLocator
            .Resolve<Console>()
            .WriteLine("Hello World");
    }
}
```

После

```
class Class
{
    public Class(Console console)
    {
        _console = console;
    }

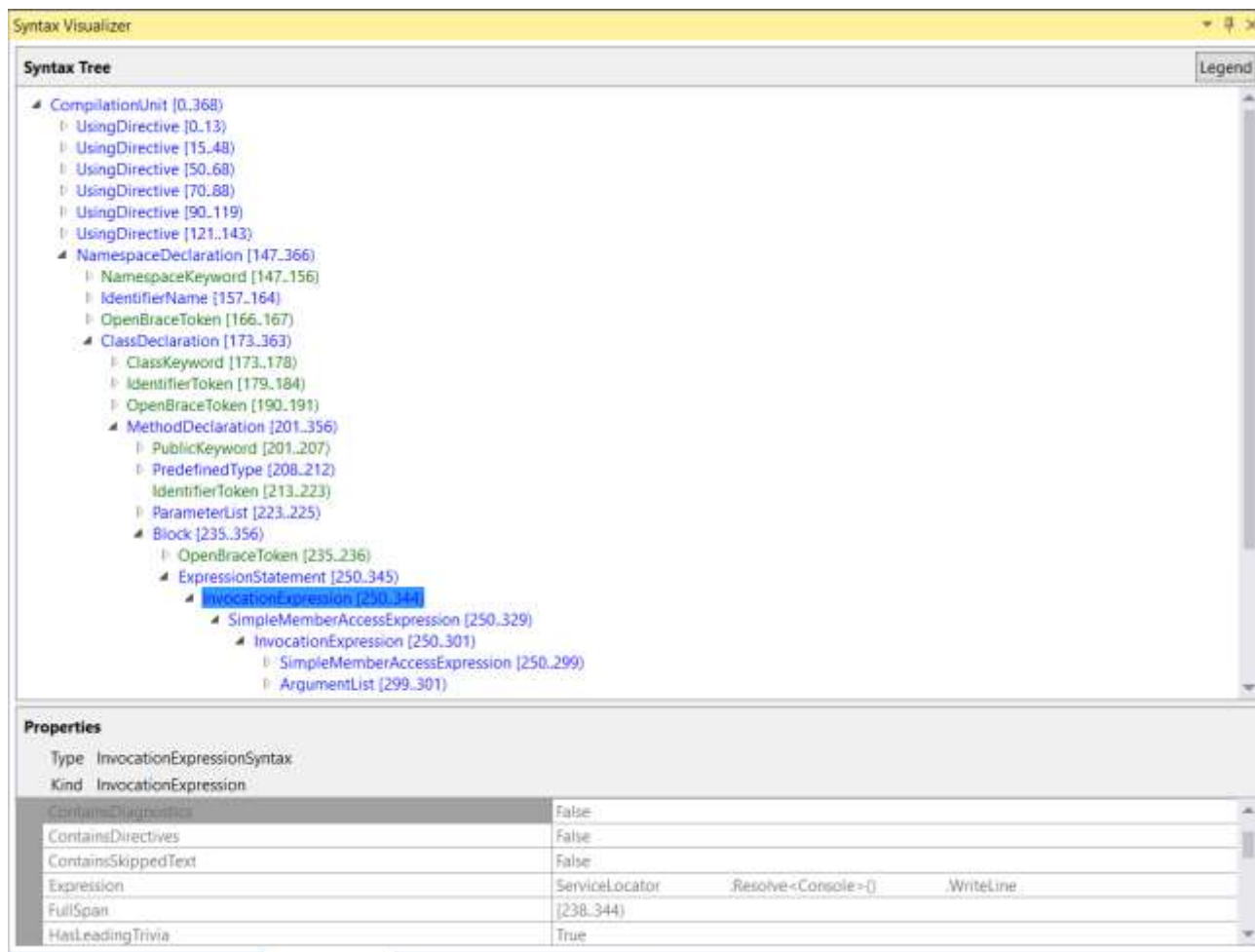
    Console _console;

    public void HelloWorld()
    {
        _console.WriteLine("Hello World");
    }
}
```

Первый шаг

- Получить объект Syntax Tree
- Получить корень этого дерева
- Пройтись от корня по всему дереву в поисках класса
- Внести необходимые изменения в класс
- Заменить класс в исходном синтаксическом дереве
- Вернуть измененное дерево в виде строки





Неочевидность №1: Для всего есть фабрика

Фабричный метод

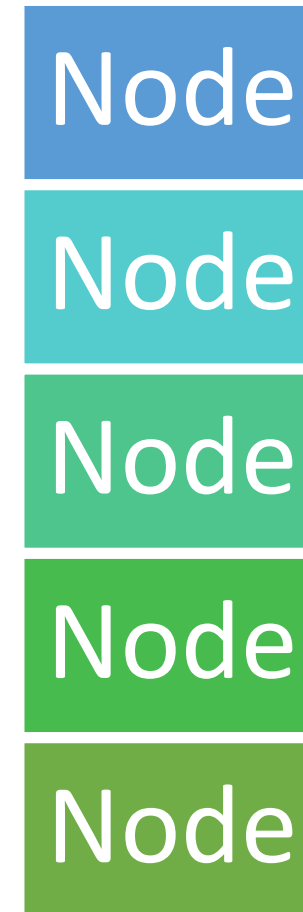
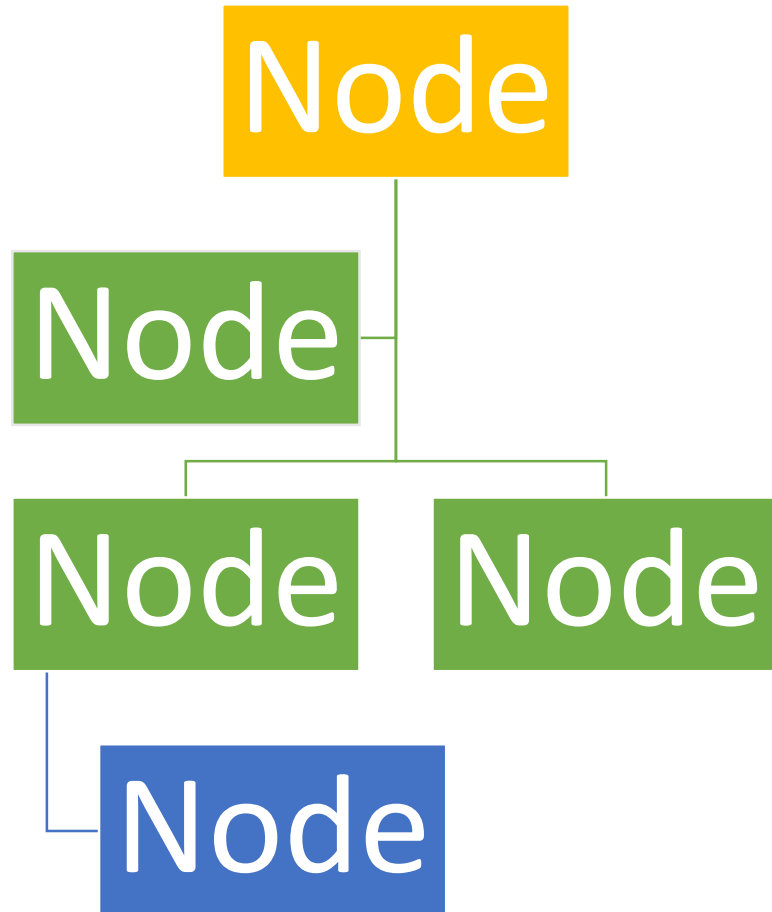
- CSharpSyntaxTree


Статическая фабрика

- SyntaxFactory



Неочевидность №2: DescendantNodes



A vibrant illustration of the Kung Fu Panda family. Po the panda is in the center, looking slightly to the right. He is surrounded by his family: Shifu is on his back, Kaiti is on his chest, and Mei Mei is on his right. Po's mother and father are also present, all looking happy and energetic. The background shows a traditional Chinese village with a temple and mountains.

Неочевидность №3: 100 простых анализаторов лучше одного сложного

Шпаргалка №2:

ReplaceNode – первый друг



Шпаргалка №3:

Как получить текст **SyntaxNode**

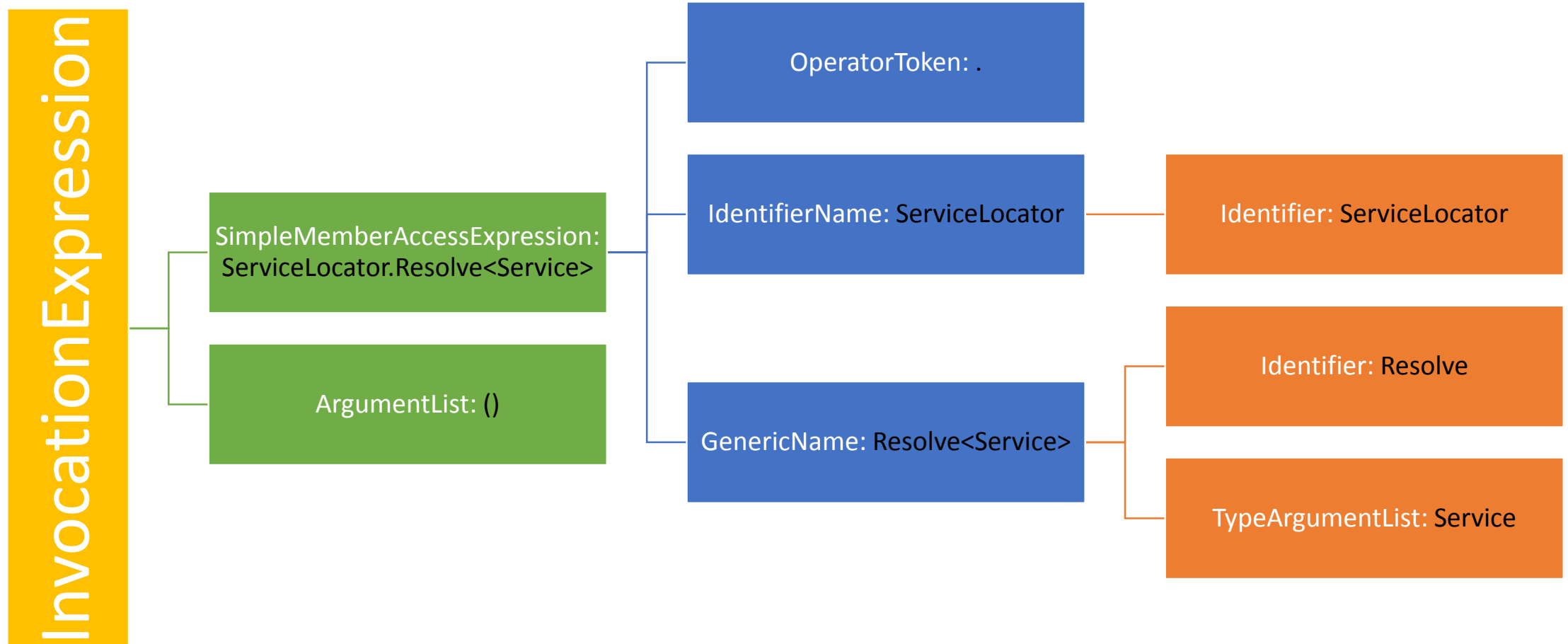
ToString

```
class MyClass  
{  
}
```

ToString

```
// This is my class  
class MyClass  
{  
} // My class is amazing
```

ServiceLocator.Resolve<Service>()





Неочевидность №4: CodeAnalysys очень дружественна к F12

Шпаргалка №4:

Помимо типа узла есть **SyntaxKind**

- В Syntax Visualizer отображается именно SyntaxKind
- Все SyntaxToken различаются именно значением SyntaxKind
- Если чего то нет в SyntaxKind, этого нет в языке

Неочевидность №5:

Где имя, там Identifier



Шпаргалка №5:

Есть Equals, а есть IsEquivalentTo

IsEquivalentTo(false)

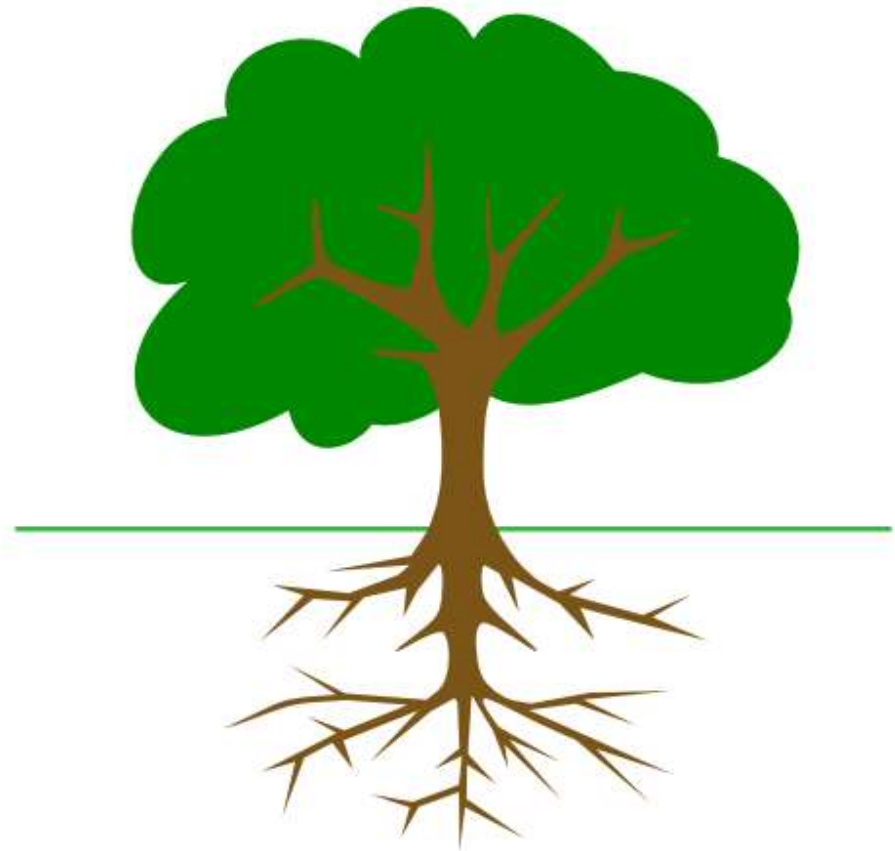


IsEquivalentTo(true)



Как будем убивать ServiceLocator?

- Создадим конструктор если нет подходящего
- Добавим соответствующие поля
- Добавим в конструктор необходимые параметры
- Проинициализируем поля в конструкторе
- Заменим вызовы ServiceLocator обращениями к полям



Шпаргалка №6:

SyntaxFactory – король этого мира



Шпаргалка №7:

With....\Update...\Add...

With...\Update...

- Возвращает новый экземпляр с полностью замененной частью
- Существует версия для любой части синтаксиса конкретной SyntaxNode
- Включает в себя токены

Add...

- Возвращает новый экземпляр с дополненной частью
- Существует если только если в этом есть смысл

Шпаргалка №8:

NormalizeWhitespace()

До

```
class Victim
{
    ~
public Victim
(CookieService cookieService, CookieMonster cookieMonster, CookiesWarehou:
    {
        var cookies = _cookieService;
        _cookieMonster.Notify(cookies.Kind);

        cookies.Consuming += count
            => _cookiesWarehouse.Send(count);

        cookies.Feed(
            _cookieMonster);

        return _cookiesWarehouse.Left;
    }
}
```

После

```
class Victim
{
    public Victim(CookieService cookieService, CookieMonster cookieMc
    {
        _cookieService = cookieService;
        _cookieMonster = cookieMonster;
        _cookiesWarehouse = cookiesWarehouse;
    }

    CookiesWarehouse _cookiesWarehouse;
    CookieMonster _cookieMonster;
    CookieService _cookieService;
    public int FeedMonster()
    {
        var cookies = _cookieService;
        _cookieMonster.Notify(cookies.Kind);
        cookies.Consuming += count => _cookiesWarehouse.Send(count);
        cookies.Feed(_cookieMonster);
        return _cookiesWarehouse.Left;
    }
}
```


Шпаргалка №9:

InsertNodesBefore и InsertNodesAfter

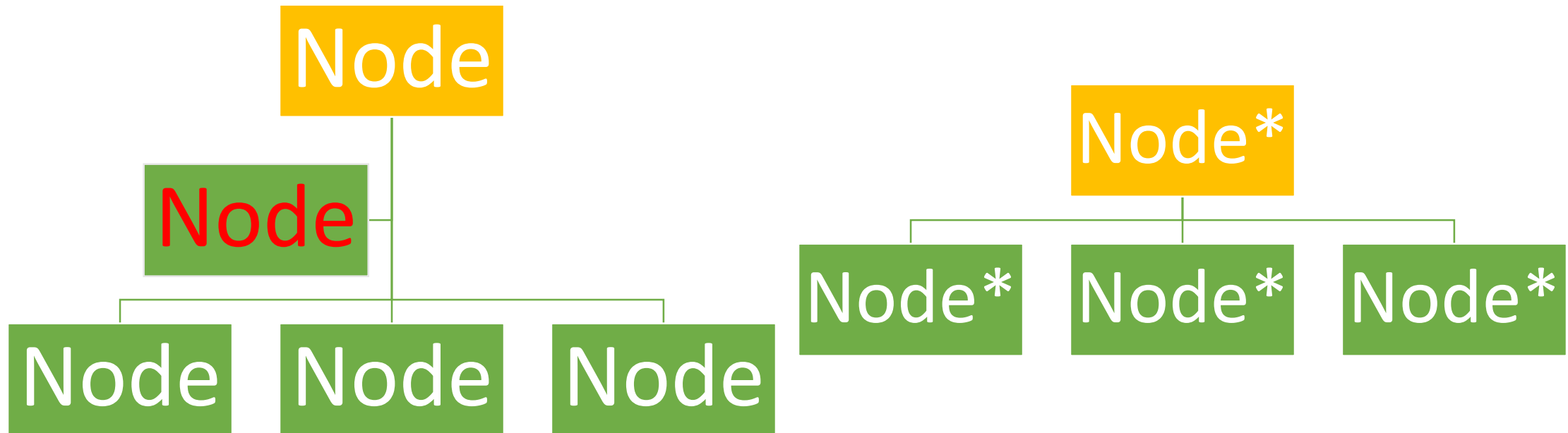


Неочевидность №6: **Immutability, immutability everywhere**



Неочевидность №7:

Если меняешь узел дерева, меняешь все дерево



**Шпаргалка
№10:
ReplaceNodes
хорошо когда
друзей
много**

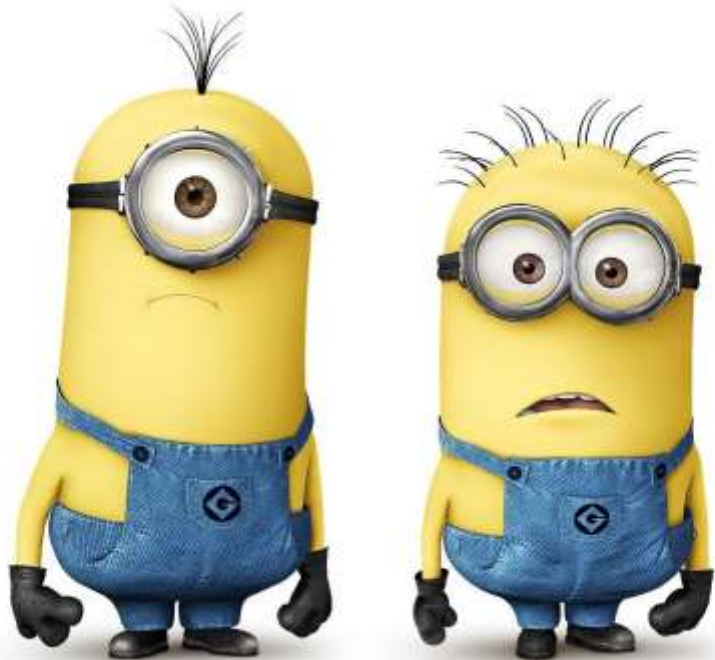


Что мы узнали

1. Не забываем использовать `SyntaxVisualizer` что бы понять структуру кода
2. Узлы с одинаковым типом могут отличаться по `SyntaxKind`
3. Все объекты синтаксической модели создаются средствами `SyntaxFactory`
4. Все синтаксические узлы следуют конвенции именования `Add/Update/With`
5. Есть много полезных функций: `ReplaceNode`, `IsEquivalentTo`, `NormalizeWhitespace` и проч.



Вопросы?



Материалы

- <https://joshvarty.wordpress.com/learn-roslyn-now/>
- <http://sergeyteplyakov.blogspot.ru/2015/10/simple-roslyn-based-analyzer.html>
- <https://github.com/dotnet/roslyn>