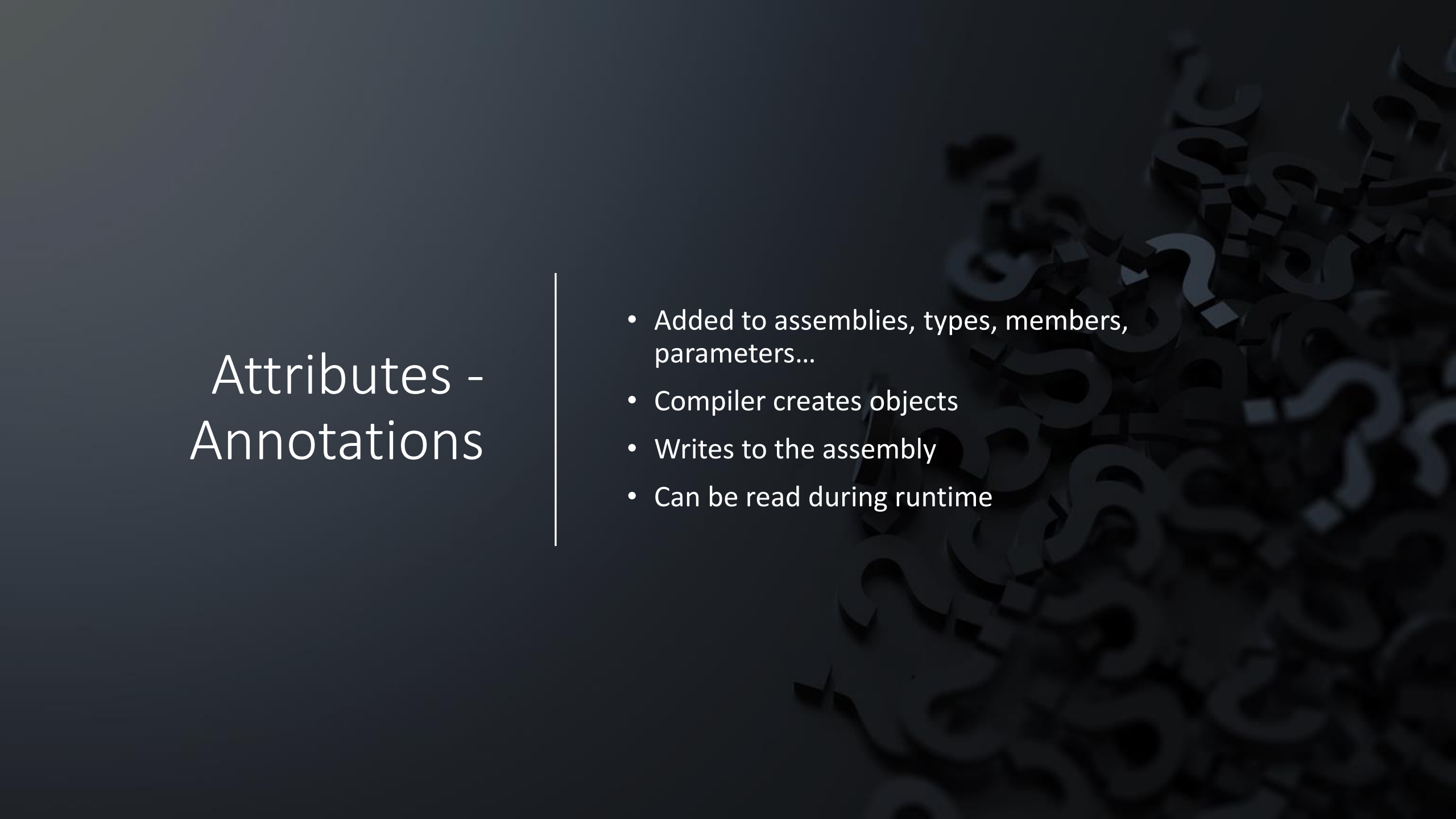
Reflection

Professional C# 2021 - Chapter 12

https://csharp.christiannagel.com

Attributes - Annotations

- Added to assemblies, types, members, parameters…
- Compiler creates objects
- Writes to the assembly
- Can be read during runtime



Custom Attributes

- Attribute Postfix (not required)
- Derives from Attribute
- Specify where to use the attribute - AttributeUsage

Custom Attributes

```
[AttributeUsage(AttributeTargets.Class | AttributeTargets.Method |
  AttributeTargets.Constructor | AttributeTargets.Property, AllowMultiple=true,
  Inherited=false)]
public class LastModifiedAttribute: Attribute
{
  private readonly DateTime _dateModified;
  private readonly string _changes;
  public LastModifiedAttribute(string dateModified, string changes)
  {
    _dateModified = DateTime.Parse(dateModified);
    _changes = changes;
  }

  public DateTime DateModified => _dateModified;

  public string Changes => _changes;

  public string Issues { get; set; }
}

[AttributeUsage(AttributeTargets.Assembly)]
public class SupportsWhatsNewAttribute: Attribute
{
}
```

Using Attributes

- Compiler Information
  - Obsolete, Caller Information, Code Analysis, Type Forward
- Information at Runtime
  - Serialization, Composition, Data Annotations
  - Module Initalization
- Tool Information
  - Designers
  - Property Editor

Reflection – Type Class

- Name
- Members of the Type
- Read Information
- Invoke Members

Assembly Class

- Access Metadata

```
Assembly assembly1 = Assembly.Load("SomeAssembly");
Assembly assembly2 = Assembly.LoadFrom
  (@"C:\My Projects\Software\SomeOtherAssembly");
```

```
Type[] types = theAssembly.GetTypes();
foreach(Type definedType in types)
{
  DoSomethingWith(definedType);
}
```

Create an object dynamically

- Using the Assembly Class

```
Assembly assembly = Assembly.LoadFile(CalculatorLibPath);
object calc = assembly.CreateInstance(CalculatorTypeName);
```

Invoke Members via Reflection

```
object result = calc.GetType().GetMethod("Add")
   .Invoke(calc, new object[] { x, y });
Console.WriteLine($"the result of {x} and {y} is {result}");
```

Dynamic Type

- Reflection
- COM Interop

```
dynamic calc = GetCalculator();
dynamic result = calc.Add(x, y);
```

Summary

Annotations

Reflection

Dynamic Type