XSLT Stylesheet Scripting Using <msxsl:script>

By dotnet-bot

Archived: 2026-04-05 21:15:46 UTC

The XslTransform class supports embedded scripting using the `script` element.

The XslTransform class supports embedded scripting using the `script` element. When the style sheet is loaded, any defined functions are compiled to Microsoft intermediate language (MSIL) by being wrapped in a class definition and have no performance loss as a result.

The `<msxsl:script>` element is defined below:

```
<msxsl:script language = "language-name" implements-prefix = "prefix of user namespace"> </msxsl:script>
```

where `msxsl` is a prefix bound to the namespace `urn:schemas-microsoft-com:xslt`.

The `language` attribute is not mandatory, but if specified, its value must be one of the following: `C#`, `VB`, `JScript`, `JavaScript`, `VisualBasic`, or `CSharp`. If not specified, the language defaults to JScript. The `language-name` is not case-sensitive, so 'JavaScript' and 'javascript' are equivalent.

The `implements-prefix` attribute is mandatory. This attribute is used to declare a namespace and associate it with the script block. The value of this attribute is the prefix that represents the namespace. This namespace can be defined somewhere in a style sheet.

Because the `msxsl:script` element belongs to the namespace `urn:schemas-microsoft-com:xslt`, the style sheet must include the namespace declaration `xmlns:msxsl=urn:schemas-microsoft-com:xslt`.

If the caller of the script does not have SecurityPermissionFlag access permission, then the script in a style sheet will never compile and the call to Load will fail.

If the caller has `UnmanagedCode` permission, the script compiles, but the operations that are allowed are dependent on the evidence that is supplied at load time.

If you are using one of the Load methods that take an XmlReader or XPathNavigator to load the style sheet, you need to use the Load overload that takes an Evidence parameter as one of its arguments. To provide evidence, the caller must have SecurityPermissionFlag permission to supply `Evidence` for the script assembly. If the caller does not have this permission, then they can set the `Evidence` parameter to `null`. This causes the Load function to fail if it finds script. The `ControlEvidence` permission is considered a very powerful permission that should only be granted to highly trusted code.

To get the evidence from your assembly, use `this.GetType().Assembly.Evidence`. To get the evidence from a Uniform Resource Identifier (URI), use `Evidence e =`

`XmlSecureResolver.CreateEvidenceForUrl(stylesheetURI)`.

If you use Load methods that take an XmlResolver but no `Evidence`, the security zone for the assembly defaults to Full Trust. For more information, see SecurityZone and Named Permission Sets.

Functions can be declared within the `msxsl:script` element. The following table shows the namespaces that are supported by default. You can use classes outside the listed namespaces. However, these classes must be fully qualified.

Default Namespaces	Description
System	System class.
System.Collection	Collection classes.
System.Text	Text classes.
System.Text.RegularExpressions	Regular expression classes.
System.Xml	Core XML classes.
System.Xml.Xsl	XSLT classes.
System.Xml.XPath	XML Path Language (XPath) classes.
Microsoft.VisualBasic	Classes for Microsoft Visual Basic scripts.

When a function is declared, it is contained in a script block. Style sheets can contain multiple script blocks, each operating independent of the other. That means that if you are executing inside a script block, you cannot call a function that you defined in another script block unless it is declared to have the same namespace and the same scripting language. Because each script block can be in its own language, and the block is parsed according to the grammar rules of that language parser, you must use the correct syntax for the language in use. For example, if you are in a C# script block, then it is an error to use an XML comment node `<!-- an XML comment -->` in the block.

The supplied arguments and return values defined by the script functions must be one of the World Wide Web Consortium (W3C) XPath or XSLT types. The following table shows the corresponding W3C types, the equivalent .NET Framework classes (Type), and whether the W3C type is an XPath type or XSLT type.

Type	Equivalent .NET Framework Class (Type)	XPath type or XSLT type
String	System.String	XPath
Boolean	System.Boolean	XPath
Number	System.Double	XPath
Result Tree Fragment	System.Xml.XPath.XPathNavigator	XSLT

Type	Equivalent .NET Framework Class (Type)	XPath type or XSLT type
Node Set	System.Xml.XPath.XPathNodeIterator	XPath

If the script function utilizes one of the following numeric types: Int16, UInt16, Int32, UInt32, Int64, UInt64, Single, or Decimal, they are forced to Double, which maps to the W3C XPath type number. All other types are forced to a string by calling the `ToString` method.

If the script function utilizes a type other than the ones mentioned above, or if the function does not compile when the style sheet is loaded into the XslTransform object, an exception is thrown.

When using the `msxsl:script` element, it is highly recommended that the script, regardless of language, be placed inside a CDATA section. For example, the following XML shows the template of the CDATA section where your code is placed.

```
 <msxsl:script implements-prefix='yourprefix' language='CSharp'>
    <![CDATA[
    ... your code here ...
    ]]>
</msxsl:script>
```

It is highly recommended that all script content be placed in a CDATA section, because operators, identifiers, or delimiters for a given language have the potential of being misinterpreted as XML. The following example shows the use of the logical AND operator in script.

```
 <msxsl:script implements-prefix='yourprefix' language='CSharp'>
    public string book(string abc, string xyz)
    {
        if ((abc == bar) && (abc == xyz)) return bar + xyz;
        else return null;
    }
</msxsl:script>
```

This throws an exception because the ampersands are not escaped. The document is loaded as XML, and no special treatment is applied to the text between the `msxsl:script` element tags.

The following example uses an embedded script to calculate the circumference of a circle given its radius.

```
 using System;
using System.IO;
using System.Xml;
using System.Xml.XPath;
using System.Xml.Xsl;

public class Sample
```

```
{
   private const String filename = "number.xml";
   private const String stylesheet = "calc.xsl";

   public static void Main()
   {
    //Create the XslTransform and load the style sheet.
    XslTransform xslt = new XslTransform();
    xslt.Load(stylesheet);

    //Load the XML data file.
    XPathDocument doc = new XPathDocument(filename);

    //Create an XmlTextWriter to output to the console.
    XmlTextWriter writer = new XmlTextWriter(Console.Out);
    writer.Formatting = Formatting.Indented;

    //Transform the file.
    xslt.Transform(doc, null, writer, null);
    writer.Close();
  }
}
```

number.xml

```
 <?xml version='1.0'?>
<data>
  <circle>
    <radius>12</radius>
  </circle>
  <circle>
    <radius>37.5</radius>
  </circle>
</data>
```

calc.xsl

```
 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:msxsl="urn:schemas-microsoft-com:xslt"
    xmlns:user="urn:my-scripts">

  <msxsl:script language="C#" implements-prefix="user">
     <![CDATA[
     public double circumference(double radius)
     {
       double pi = 3.14;
```

```
        double circ = pi*radius*2;
        return circ;
      }
       ]]>
   </msxsl:script>

  <xsl:template match="data">
  <circles>

  <xsl:for-each select="circle">
    <circle>
    <xsl:copy-of select="node()"/>
       <circumference>
          <xsl:value-of select="user:circumference(radius)"/>
       </circumference>
    </circle>
  </xsl:for-each>
  </circles>
  </xsl:template>
</xsl:stylesheet>
```

```
 <circles xmlns:msxsl="urn:schemas-microsoft-com:xslt" xmlns:user="urn:my-scripts">
  <circle>
    <radius>12</radius>
    <circumference>75.36</circumference>
  </circle>
  <circle>
    <radius>37.5</radius>
    <circumference>235.5</circumference>
  </circle>
</circles>
```

- XslTransform Class Implements the XSLT Processor
