System.Math Class

```
[ILASM]
.class public sealed Math extends System.Object

[C#]
public sealed class Math
```

Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- *Version:* 1.0.x.x
- *Attributes:*
  - CLSCompliantAttribute(true)

Summary

Provides constants and static methods for trigonometric, logarithmic, and other common mathematical functions.

Inherits From: System.Object

Library: ExtendedNumerics

Thread Safety: All public static members of this type are safe for multithreaded operations. No instance members are guaranteed to be thread safe.

Math.E Field

```
[ILASM]
.field public static literal float64 E = 2.71828182845905

[C#]
public const double E = 2.71828182845905
```

Summary

A constant, **e**, which specifies the natural logarithmic base rounded to double precision.

Description

The value of this constant is 2.7182818284590452354 converted to **System.Double**.

Math.PI Field

```
[ILASM]
.field public static literal float64 PI = 3.14159265358979

[C#]
public const double PI = 3.14159265358979
```

Summary

A constant, Đ, which specifies the ratio of the circumference of a circle to its diameter rounded to double precision.

Description

The value of this constant is 3.14159265358979323846 converted to **System.Double**.

Math.Abs(System.SByte) Method

[ILASM]

```
.method public hidebysig static int8 Abs(int8 value)
```

[C#]

```
public static sbyte Abs(sbyte value)
```

Summary

Returns the absolute value of the specified **System.SByte**.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
value	A **System.SByte**.

Return Value

A **System.SByte** containing the absolute value of *value*.

Description

This method is not CLS-compliant. For a CLS-compliant alternative, use **System.Math.Abs**(**System.Int16**).

Exceptions

Exception	Condition
System.OverflowException	*value* equals **System.SByte.MinValue**.

Math.Abs(System.Int16) Method

[ILASM]

```
.method public hidebysig static int16 Abs(int16 value)
```

[C#]

```
public static short Abs(short value)
```

Summary

Returns the absolute value of the specified **System.Int16**.

Parameters

Parameter	Description
value	A **System.Int16**.

Return Value

A **System.Int16** containing the absolute value of *value*.

Exceptions

Exception	Condition
System.OverflowException	*value* equals **System.Int16.MinValue**.

Math.Abs(System.Int32) Method

```
[ILASM]
.method public hidebysig static int32 Abs(int32 value)

[C#]
public static int Abs(int value)
```

Summary

Returns the absolute value of the specified **System.Int32**.

Parameters

Parameter	Description
value	A **System.Int32**.

Return Value

A **System.Int32** containing the absolute value of *value*.

Exceptions

Exception	Condition
System.OverflowException	*value* equals **System.Int32.MinValue**.

Math.Abs(System.Int64) Method

```
[ILASM]
.method public hidebysig static int64 Abs(int64 value)

[C#]
public static long Abs(long value)
```

Summary

Returns the absolute value of the specified **System.Int64**.

Parameters

Parameter	Description
value	A **System.Int64**.

Return Value

A **System.Int64** containing the absolute value of *value*.

Exceptions

Exception	Condition
System.OverflowException	*value* equals **System.Int64.MinValue**.

Math.Abs(System.Single) Method

[ILASM]
```
.method public hidebysig static float32 Abs(float32 value)
```

[C#]
```
public static float Abs(float value)
```

Summary

Returns the absolute value of the specified **System.Single**.

Parameters

Parameter	Description
value	A **System.Single**.

Return Value

A **System.Single** containing the absolute value of *value*. If *value* is equal to **System.Single.NegativeInfinity** or **System.Single.PositiveInfinity**, returns **System.Single.PositiveInfinity**. If *value* is equal to **System.Single.NaN**, returns **System.Single.NaN**.

Math.Abs(System.Double) Method

```
[ILASM]
.method public hidebysig static float64 Abs(float64 value)

[C#]
public static double Abs(double value)
```

Summary

Returns the absolute value of the specified **System.Double**.

Parameters

Parameter	Description
value	A **System.Double**.

Return Value

A **System.Double** containing the absolute value of *value*. If *value* is equal to **System.Double.NegativeInfinity** or **System.Double.PositiveInfinity**, returns **System.Double.PositiveInfinity**. If value is equal to **System.Double.NaN**, returns **System.Double.NaN**.

Math.Abs(System.Decimal) Method

[ILASM]

```
.method public hidebysig static decimal Abs(decimal value)
```

[C#]

```
public static decimal Abs(decimal value)
```

Summary

Returns the absolute value of the specified **System.Decimal**.

Parameters

Parameter	Description
value	A **System.Decimal**.

Return Value

A **System.Decimal** containing the absolute value of *value*.

Example

The following example demonstrates the **System.Math.Abs**(**System.Decimal**) method.

[C#]

```
using System;

public class MathAbsExample
{
   public static void Main()
   {
      Decimal d1 = Math.Abs((Decimal)0.00);
      Decimal d2 = Math.Abs((Decimal)(-1.23));
      Console.WriteLine("Math.Abs((Decimal)0.00) returns
{0}",d1);
      Console.WriteLine("Math.Abs((Decimal)(-1.23)) returns
{0}",d2);
   }
}
```

The output is

```
Math.Abs((Decimal)0.00) returns 0


Math.Abs((Decimal)(-1.23)) returns 1.23
```

degrees.]

Math.Asin(System.Double) Method

[ILASM]

```
.method public hidebysig static float64 Asin(float64 d)
```

[C#]

```
public static double Asin(double d)
```

Summary

Returns the angle whose sine is the specified **System.Double**.

Parameters

Parameter	Description
d	A **System.Double** representing a sine, where -1 <= *d* <= 1.

Return Value

A **System.Double**
radians, for which *d* is the sine, such that - *d* < -
1, *d* > 1, or *d* = **System.Double.NaN**, returns **System.Double.NaN**.

Description

[*Note:* A positive return value represents a counterclockwise angle from the positive x-axis; a negative return value represents a clockwise angle.

degrees.]

The following table specifies the return value if d is equal to **System.Double.NaN**, **System.Double.NegativeInfinity**, or **System.Double.PositiveInfinity**.

Return Value	**Condition**
System.Double.NaN	*d* is equal to **System.Double.NaN**.
- double precision (-1.5707963267949)	*d* is equal to **System.Double.NegativeInfinity**.
(1.5707963267949)	*d* is equal to **System.Double.PositiveInfinity**.

Description

[*Note:* A positive return value represents a counterclockwise angle from the positive x-axis; a negative return value represents a clockwise angle.

degrees.]

Math.Atan2(System.Double, System.Double) Method

```
[ILASM]
.method public hidebysig static float64 Atan2(float64 y,
float64 x)

[C#]
public static double Atan2(double y, double x)
```

Summary

Returns the angle whose tangent is the quotient of two specified **System.Double** values.

Parameters

Parameter	Description
y	A **System.Double** representing the y coordinate of a point.
x	A **System.Double** representing the x coordinate of a point.

Return Value

A **System.Double**
radians, such that - *y*/*x*, where (*x*, *y*) is a point in the Cartesian plane.

If both *x* and *y* are any combination of **System.Double.NegativeInfinity** and **System.Double.PositiveInfinity**, **System.Double.NaN** is returned.

If either *x* or *y* is equal to **System.Double.NaN**, **System.Double.NaN** is returned.

The following table specifies the return value if *x* or *y* is equal to **System.Double.NegativeInfinity** or **System.Double.PositiveInfinity**.

Condition	Return Value
y is equal to **System.Double.PositiveInfinity** or **System.Double.NegativeInfinity**, and *x* is equal to **System.Double.PositiveInfinity** or **System.Double.NegativeInfinity**.	**System.Double.NaN**.
y is equal to **System.Double.NegativeInfinity**, and	-**System.Math.PI**/2.

- For (*x*, *y*) in quadrant 3, -   -
- For (*x*, *y*) in quadrant 4, -

]

Example

The following example demonstrates using the **System.Math.Atan2** method.

[C#]

```
using System;

public class MathAtan2Example
{

   public static void Main()
   {

      Double d1 = Math.Atan2(2,0);
      Double d2 = Math.Atan2(0,0);
      Console.WriteLine("Math.Atan2(2,0) returns {0}", d1);
      Console.WriteLine("Math.Atan2(0,0) returns {0}", d2);

   }

}
```

The output is

```
Math.Atan2(2,0) returns 1.5707963267949


Math.Atan2(0,0) returns 0
```

Math.BigMul(System.Int32, System.Int32) Method

[ILASM]

```
.method public hidebysig static int64 BigMul(int32 a,int32 b)
```

[C#]

```
public static long BigMul(int a, int b)
```

Summary

Produces the full product of two 32-bit numbers.

Parameters

Parameter	Description
a	The first **System.Int32** to multiply.
b	The second **System.Int32** to multiply.

Return Value

A **System.Int64** containing the product of the specified numbers.

Math.Ceiling(System.Double) Method

[ILASM]

```
.method public hidebysig static float64 Ceiling(float64 a)
```

[C#]

```
public static double Ceiling(double a)
```

Summary

Returns the smallest integer greater than or equal to the specified **System.Double**.

Parameters

Parameter	Description
a	A **System.Double**.

Return Value

A **System.Double** containing the value of the smallest integer greater than or equal to *a*. If *a* is equal to **System.Double.NaN**, **System.Double.NegativeInfinity**, or **System.Double.PositiveInfinity**, that value is returned.

Example

The following example demonstrates using the **System.Math.Ceiling** method.

[C#]

```
using System;

public class MathCeilingExample
{

   public static void Main()
   {

      Double d1 = Math.Ceiling(3.4);
      Double d2 = Math.Ceiling(-3.4);
      Console.WriteLine("Math.Ceiling(3.4) returns {0}",
d1);
      Console.WriteLine("Math.Ceiling(-3.4) returns {0}",
d2);

   }
}
```

The output is

```
Math.Ceiling(3.4) returns 4


Math.Ceiling(-3.4) returns -3
```

Math.Cos(System.Double) Method

[ILASM]
```
.method public hidebysig static float64 Cos(float64 d)
```

[C#]
```
public static double Cos(double d)
```

Summary

Returns the cosine of the specified **System.Double** that represents an angle.

Parameters

Parameter	Description
d	A **System.Double** that represents an angle measured in radians.

Return Value

A **System.Double** containing the value of the cosine of *d*. If *d* is equal to **System.Double.NaN**, **System.Double.NegativeInfinity**, or **System.Double.PositiveInfinity**, returns **System.Double.NaN**.

Description

[*Note:*

Math.Cosh(System.Double) Method

[ILASM]

```
.method public hidebysig static float64 Cosh(float64 value)
```

[C#]

```
public static double Cosh(double value)
```

Summary

Returns the hyperbolic cosine of the specified **System.Double** that represents an angle.

Parameters

Parameter	Description
value	A **System.Double** that represents an angle measured in radians.

Return Value

The hyperbolic cosine of *value*. If *value* is equal to **System.Double.NegativeInfinity** or **System.Double.PositiveInfinity**, returns **System.Double.PositiveInfinity**. If *value* is equal to **System.Double.NaN**, returns **System.Double.NaN**.

Description

[*Note:*

Math.DivRem(System.Int32, System.Int32, System.Int32) Method

[ILASM]

```
.method public hidebysig static int32 DivRem(int32 a,int32
b, [out] int32 &result)
```

[C#]

```
public static int DivRem(int a, int b, out int result)
```

Summary

Returns the quotient of two numbers, also passing the remainder as an output parameter.

Parameters

Parameter	Description
a	A **System.Int32** that contains the dividend.
b	A **System.Int32** that contains the divisor.
result	A **System.Int32** that receives the remainder.

Return Value

A **System.Int32** containing the quotient of the specified numbers.

Math.DivRem(System.Int64, System.Int64, System.Int64) Method

[ILASM]

```
.method public hidebysig static int64 DivRem(int64 a,int64
b,[out] int64 &result)
```

[C#]

```
public static long DivRem(long a, long b, out long result)
```

Summary

Returns the quotient of two numbers, also passing the remainder as an output parameter.

Parameters

Parameter	Description
a	A **System.Int64** that contains the dividend.
b	A **System.Int64** that contains the divisor.
result	A **System.Int64** that receives the remainder.

Return Value

A **System.Int64** containing the quotient of the specified numbers.

Math.Exp(System.Double) Method

```
[ILASM]
.method public hidebysig static float64 Exp(float64 d)

[C#]
public static double Exp(double d)
```

Summary

Returns **e** raised to the specified **System.Double** that represents an exponent.

Parameters

Parameter	Description
d	A **System.Double** that represents an exponent.

Return Value

A **System.Double** equal to the number **e** raised to the power of *d*. If *d* equals **System.Double.NaN** or **System.Double.PositiveInfinity**, returns that value. If *d* equals **System.Double.NegativeInfinity**, returns 0.

Description

[*Note:* Use the **System.Math.Pow** method to calculate powers of other bases.

System.Math.Exp is the inverse of **System.Math.Log**.]

Math.Floor(System.Double) Method

[ILASM]

```
.method public hidebysig static float64 Floor(float64 d)
```

[C#]

```
public static double Floor(double d)
```

Summary

Returns the largest integer less than or equal to the specified **System.Double**.

Parameters

Parameter	Description
d	A **System.Double**.

Return Value

A **System.Double** containing the value of the largest integer less than or equal to *d*. If *d* is equal to **System.Double.NaN**, **System.Double.NegativeInfinity**, or **System.Double.PositiveInfinity**, that value is returned..

Description

The behavior of this method follows IEEE Standard 754, section 4.

Example

The following example demonstrates using the **System.Math.Floor** method.

[C#]

```
using System;

public class MathFloorExample
{

   public static void Main()
   {

      Double d1 = Math.Floor(3.4);
      Double d2 = Math.Floor(-3.4);
      Console.WriteLine("Math.Floor(3.4) returns {0}", d1);
```

```
            Console.WriteLine("Math.Floor(-3.4) returns {0}",
      d2);

         }

      }
```

The output is

```
Math.Floor(3.4) returns 3


Math.Floor(-3.4) returns -4
```

Math.IEEERemainder(System.Double, System.Double) Method

[ILASM]

```
.method public hidebysig static float64
IEEERemainder(float64 x, float64 y)
```

[C#]

```
public static double IEEERemainder(double x, double y)
```

Summary

Returns the remainder resulting from the division of one specified **System.Double** by another specified **System.Double**.

Parameters

Parameter	Description
x	A **System.Double** that represents a dividend.
y	A **System.Double** that represents a divisor.

Return Value

A **System.Double** whose value is as follows:

Value	Description
x - (*y* Q),	Q is the quotient of *x*/*y* rounded to the nearest integer (if *x*/*y* is exactly halfway between two integers, the even integer is returned).
+0	Q is the quotient of *x*/*y* rounded to the nearest integer (if *x*/*y* is exactly halfway between two integers, the even integer is returned), *x* - (y Q) is zero, and *x* is positive.
-0	Q is the quotient of *x*/*y* rounded to the nearest integer (if *x*/*y* is exactly halfway between two integers, the even integer is returned), *x* - (y Q) is zero, and *x* is negative.
System.Double.NaN	y = 0.

Description

This operation complies with the remainder operation defined in Section 5.1 of ANSI/IEEE Std 754-1985; IEEE Standard for Binary Floating-Point Arithmetic; Institute of Electrical and Electronics

Engineers, Inc; 1985.

[*Note:* For more information regarding the use of +0 and -0, see Section 3.1 of ANSI/IEEE Std 754-1985; IEEE Standard for Binary Floating-Point Arithmetic; Institute of Electrical and Electronics Engineers, Inc; 1985.]

Example

The following example demonstrates using the **System.Math.IEEERemainder** method.

[C#]

```
using System;

public class MathIEEERemainderExample
{

   public static void Main()
   {

      Double d1 = Math.IEEERemainder(3.54,0);
      Double d2 = Math.IEEERemainder(9.99,-3.33);
      Double d3 = Math.IEEERemainder(-9.99,3.33);
      Double d4 = Math.IEEERemainder(9.5,1.5);
      Console.WriteLine("Math.IEEERemainder(3.54,0) returns
{0}", d1);
      Console.WriteLine("Math.IEEERemainder(9.99,-3.33)
returns {0}", d2);
      Console.WriteLine("Math.IEEERemainder(-9.99,3.33)
returns {0}", d3);
      Console.WriteLine("Math.IEEERemainder(9.5,1.5)
returns {0}", d4);

   }

}
```

The output is

```
Math.IEEERemainder(3.54,0) returns NaN


Math.IEEERemainder(9.99,-3.33) returns 0
Math.IEEERemainder(-9.99,3.33) returns 0
```

```
Math.IEEERemainder(9.5,1.5) returns 0.5
```

Math.Log(System.Double) Method

```
[ILASM]
.method public hidebysig static float64 Log(float64 d)

[C#]
public static double Log(double d)
```

Summary

Returns the natural logarithm of the specified **System.Double**.

Parameters

Parameter	Description
d	A **System.Double** whose natural logarithm is to be found.

Return Value

Returns a **System.Double** whose value is as follows.

Condition	Returns
d > 0.	The value of the natural logarithm of *d*.
d == 0.	**System.Double.NegativeInfinity**.
d < 0. -or- *d* is equal to **System.Double.NegativeInfinity**. -or- *d* is equal to **System.Double.NaN**.	**System.Double.NaN**.
d is equal to **System.Double.PositiveInfinity**.	**System.Double.PositiveInfinity**.

Description

d is specified as a base 10 number.

Math.Log(System.Double, System.Double) Method

```
[ILASM]
.method public hidebysig static float64 Log(float64 a,
float64 newBase)

[C#]
public static double Log(double a, double newBase)
```

Summary

Returns the logarithm of the specified **System.Double** in the specified base.

Parameters

Parameter	Description
a	A **System.Double** whose logarithm is to be found.
newBase	A **System.Double** containing the value of the base of the logarithm.

Return Value

Returns a **System.Double** whose value is as follows:

Condition	Returns
a > 0.	The value of $\text{Log}_{newBase}a$, if and only if *newBase* is greater than or equal to 0; otherwise, **System.Double.NaN**.
a == 0.	**System.Double.NegativeInfinity**.
a > 0.	**System.Double.NaN**.

If *a* is equal to **System.Double.PositiveInfinity** and *newBase* is not equal **System.Double.PositiveInfinity**, **System.Double.NegativeInfinity**, or **System.Double.NaN**, returns **System.Double.PositiveInfinity**. If *newBase* is equal to **System.Double.PositiveInfinity** and *a* is not equal to **System.Double.PositiveInfinity**, **System.Double.NegativeInfinity**, or **System.Double.NaN**, returns 0. If both *a* and *newBase* are equal to **System.Double.PositiveInfinity**, or *a* or *newBase* is equal to **System.Double.NaN** or **System.Double.NegativeInfinity**, returns **System.Double.NaN**.

Math.Log10(System.Double) Method

```
[ILASM]
.method public hidebysig static float64 Log10(float64 d)

[C#]
public static double Log10(double d)
```

Summary

Returns $\log_{10}$ of the specified **System.Double**.

Parameters

Parameter	Description
d	A **System.Double** whose logarithm is to be found.

Return Value

Returns a **System.Double** as indicated by the following table.

Condition	Returns
d > 0.	A **System.Double** containing the value of $\log_{10}d$.
d == 0.	**System.Double.NegativeInfinity**.
d < 0. -or- *d* is equal to **System.Double.NegativeInfinity**. -or- *d* is equal to **System.Double.NaN**.	**System.Double.NaN**.
d is equal to **System.Double.PositiveInfinity**.	**System.Double.PositiveInfinity**.

Math.Max(System.SByte, System.SByte) Method

```
[ILASM]
.method public hidebysig static int8 Max(int8 val1, int8
val2)

[C#]
public static sbyte Max(sbyte val1, sbyte val2)
```

Summary

Returns the greater of two specified **System.SByte** values.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
val1	The first of two specified **System.Byte** values to compare.
val2	The second of two specified **System.Byte** values to compare.

Return Value

A **System.SByte** that is equal to *val1* if *val1* is greater than or equal to *val2*; otherwise, the return value is equal to *val2*.

Description

This method is not CLS-compliant. For a CLS-compliant alternative, use **System.Math.Max**(**System.Int16**, **System.Int16**).

Math.Max(System.Byte, System.Byte) Method

[ILASM]

```
.method public hidebysig static unsigned int8 Max(unsigned
int8 val1, unsigned int8 val2)
```

[C#]

```
public static byte Max(byte val1, byte val2)
```

Summary

Returns the greater of two specified **System.Byte** values.

Parameters

Parameter	Description
val1	The first of two specified **System.Byte** values to compare.
val2	The second of two specified **System.Byte** values to compare.

Return Value

A **System.Byte** that is equal to *val1* if *val1* is greater than or equal to *val2*; otherwise, the return value is equal to *val2*.

Math.Max(System.Int16, System.Int16) Method

[ILASM]

```
.method public hidebysig static int16 Max(int16 val1, int16
val2)
```

[C#]

```
public static short Max(short val1, short val2)
```

Summary

Returns the greater of two specified **System.Int16** values.

Parameters

Parameter	Description
val1	The first of two specified **System.Int16** values to compare.
val2	The second of two specified **System.Int16** values to compare.

Return Value

A **System.Int16** that is equal to *val1* if *val1* is greater than or equal to *val2*; otherwise, the return value is equal to *val2*.

Math.Max(System.UInt16, System.UInt16) Method

```
[ILASM]
.method public hidebysig static unsigned int16 Max(unsigned
int16 val1, unsigned int16 val2)

[C#]
public static ushort Max(ushort val1, ushort val2)
```

Summary

Returns the greater of two specified **System.UInt16** values.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
val1	The first of two specified **System.UInt16** values to compare.
val2	The second of two specified **System.UInt16** values to compare.

Return Value

A **System.UInt16** that is equal to *val1* if *val1* is greater than or equal to *val2*; otherwise, the return value is equal to *val2*.

Description

This method is not CLS-compliant. For a CLS-compliant alternative, use **System.Math.Max**(**System.Int32**, **System.Int32**).

Math.Max(System.Int32, System.Int32) Method

```
[ILASM]
.method public hidebysig static int32 Max(int32 val1, int32
val2)

[C#]
public static int Max(int val1, int val2)
```

Summary

Returns the greater of two specified **System.Int32** values.

Parameters

Parameter	Description
val1	The first of two specified **System.Int32** values to compare.
val2	The second of two specified **System.Int32** values to compare.

Return Value

A **System.Int32** that is equal to *val1* if *val1* is greater than or equal to *val2*; otherwise, the return value is equal to *val2*.

Math.Max(System.UInt32, System.UInt32) Method

```
[ILASM]
.method public hidebysig static unsigned int32 Max(unsigned
int32 val1, unsigned int32 val2)

[C#]
public static uint Max(uint val1, uint val2)
```

Summary

Returns the greater of two specified **System.UInt32** values.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
val1	The first of two specified **System.UInt32** values to compare.
val2	The second of two specified **System.UInt32** values to compare.

Return Value

A **System.UInt32** that is equal to *val1* if *val1* is greater than or equal to *val2*; otherwise, the return value is equal to *val2*.

Description

This method is not CLS-compliant. For a CLS-compliant alternative, use **System.Math.Max**(**System.Int64**, **System.Int64**).

Math.Max(System.Int64, System.Int64) Method

[ILASM]
```
.method public hidebysig static int64 Max(int64 val1, int64
val2)
```

[C#]
```
public static long Max(long val1, long val2)
```

Summary

Returns the greater of two specified **System.Int64** values.

Parameters

Parameter	Description
val1	The first of two specified **System.Int64** values to compare.
val2	The second of two specified **System.Int64** values to compare.

Return Value

A **System.Int64** that is equal to *val1* if *val1* is greater than or equal to *val2*; otherwise, the return value is equal to *val2*.

Math.Max(System.UInt64, System.UInt64) Method

```
[ILASM]
.method public hidebysig static unsigned int64 Max(unsigned
int64 val1, unsigned int64 val2)

[C#]
public static ulong Max(ulong val1, ulong val2)
```

Summary

Returns the greater of two specified **System.UInt64** values.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
val1	The first of two specified **System.UInt64** values to compare.
val2	The second of two specified **System.UInt64** values to compare.

Return Value

A **System.UInt64** equal to *val1* if *val1* is greater than or equal to *val2*; otherwise, the return value is equal to *val2*.

Description

This method is not CLS-compliant. For a CLS-compliant alternative, use **System.Math.Max**(**System.Decimal**, **System.Decimal**).

Math.Max(System.Single, System.Single) Method

[ILASM]

```
.method public hidebysig static float32 Max(float32 val1,
float32 val2)
```

[C#]

```
public static float Max(float val1, float val2)
```

Summary

Returns the greater of two specified **System.Single** values.

Parameters

Parameter	Description
val1	The first of two specified **System.Single** values to compare.
val2	The second of two specified **System.Single** values to compare.

Return Value

A **System.Single** equal to *val1* if *val1* is greater than or equal to *val2*; otherwise, the return value is equal to *val2*. If *val1*, *val2*, or both are equal to **System.Single.NaN**, **System.Single.NaN** is returned.

Math.Max(System.Double, System.Double) Method

[ILASM]
```
.method public hidebysig static float64 Max(float64 val1,
float64 val2)
```

[C#]
```
public static double Max(double val1, double val2)
```

Summary

Returns the greater of two specified **System.Double** values.

Parameters

Parameter	Description
val1	The first of two specified **System.Double** values to compare.
val2	The second of two specified **System.Double** values to compare.

Return Value

A **System.Double** equal to *val1* if *val1* is greater than or equal to *val2*; otherwise, the return value is equal to *val2*. If *val1*, *val2*, or both are equal to **System.Double.NaN**, **System.Double.NaN** is returned.

Math.Max(System.Decimal, System.Decimal) Method

[ILASM]

```
.method public hidebysig static decimal Max(decimal val1,
decimal val2)
```

[C#]

```
public static decimal Max(decimal val1, decimal val2)
```

Summary

Returns the greater of two specified **System.Decimal** values.

Parameters

Parameter	Description
val1	The first of two specified **System.Decimal** values to compare.
val2	The second of two specified **System.Decimal** values to compare.

Return Value

A **System.Decimal** that is equal to *val1* if *val1* is greater than or equal to *val2*; otherwise, the return value is equal to *val2*.

Math.Min(System.SByte, System.SByte) Method

```
[ILASM]
.method public hidebysig static int8 Min(int8 val1, int8
val2)

[C#]
public static sbyte Min(sbyte val1, sbyte val2)
```

Summary

Returns the lesser of two specified **System.SByte** values.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
val1	The first of two specified **System.SByte** values to compare.
val2	The second of two specified **System.SByte** values to compare.

Return Value

A **System.SByte** equal to *val1* if *val1* is less than or equal to *val2*; otherwise, the return value is equal to *val2*.

Description

This method is not CLS-compliant. For a CLS-compliant alternative, use **System.Math.Min**(**System.Int16**, **System.Int16**).

Math.Min(System.Byte, System.Byte) Method

[ILASM]

```
.method public hidebysig static unsigned int8 Min(unsigned
int8 val1, unsigned int8 val2)
```

[C#]

```
public static byte Min(byte val1, byte val2)
```

Summary

Returns the lesser of two specified **System.Byte** values.

Parameters

Parameter	Description
val1	The first of two specified **System.Byte** values to compare.
val2	The second of two specified **System.Byte** values to compare.

Return Value

A **System.Byte** equal to *val1* if *val1* is less than or equal to *val2*; otherwise, the return value is equal to *val2*.

Math.Min(System.Int16, System.Int16) Method

[ILASM]
```
.method public hidebysig static int16 Min(int16 val1, int16
val2)
```

[C#]
```
public static short Min(short val1, short val2)
```

Summary

Returns the lesser of two specified **System.Int16** values.

Parameters

Parameter	Description
val1	The first of two specified **System.Int16** values to compare.
val2	The second of two specified **System.Int16** values to compare.

Return Value

A **System.Int16** that is equal to *val1* if *val1* is less than or equal to *val2*; otherwise, the return value is equal to *val2*.

Math.Min(System.UInt16, System.UInt16) Method

[ILASM]

```
.method public hidebysig static unsigned int16 Min(unsigned
int16 val1, unsigned int16 val2)
```

[C#]

```
public static ushort Min(ushort val1, ushort val2)
```

Summary

Returns the lesser of two specified **System.UInt16** values.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
val1	The first of two specified **System.UInt16** values to compare.
val2	The second of two specified **System.UInt16** values to compare.

Return Value

A **System.UInt16** equal to *val1* if *val1* is less than or equal to *val2*; otherwise, the return value is equal to *val2*.

Description

This method is not CLS-compliant. For a CLS-compliant alternative, use **System.Math.Min**(**System.Int32**, **System.Int32**).

Math.Min(System.Int32, System.Int32) Method

[ILASM]

```
.method public hidebysig static int32 Min(int32 val1, int32 val2)
```

[C#]

```
public static int Min(int val1, int val2)
```

Summary

Returns the lesser of two specified **System.Int32** values.

Parameters

Parameter	Description
val1	The first of two specified **System.Int32** values to compare.
val2	The second of two specified **System.Int32** values to compare.

Return Value

A **System.Int32** equal to *val1* if *val1* is less than or equal to *val2*; otherwise, the return value is equal to *val2*.

Math.Min(System.UInt32, System.UInt32) Method

```
[ILASM]
.method public hidebysig static unsigned int32 Min(unsigned
int32 val1, unsigned int32 val2)

[C#]
public static uint Min(uint val1, uint val2)
```

Summary

Returns the lesser of two specified **System.UInt32** values.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
val1	The first of two specified **System.UInt32** values to compare.
val2	The second of two specified **System.UInt32** values to compare.

Return Value

A **System.UInt32** equal to *val1* if *val1* is less than or equal to *val2*; otherwise, the return value is equal to *val2*.

Description

This method is not CLS-compliant. For a CLS-compliant alternative, use **System.Math.Min**(**System.Int64**, **System.Int64**).

Math.Min(System.Int64, System.Int64) Method

```
[ILASM]
.method public hidebysig static int64 Min(int64 val1, int64
val2)

[C#]
public static long Min(long val1, long val2)
```

Summary

Returns the lesser of two specified **System.Int64** values.

Parameters

Parameter	Description
val1	The first of two specified **System.Int64** values to compare.
val2	The second of two specified **System.Int64** values to compare.

Return Value

A **System.Int64** equal to *val1* if *val1* is less than or equal to *val2*; otherwise, the return value is equal to *val2*.

Math.Min(System.UInt64, System.UInt64) Method

```
[ILASM]
.method public hidebysig static unsigned int64 Min(unsigned
int64 val1, unsigned int64 val2)

[C#]
public static ulong Min(ulong val1, ulong val2)
```

Summary

Returns the lesser of two specified **System.UInt64** values.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
val1	The first of two specified **System.UInt64** values to compare.
val2	The second of two specified **System.UInt64** values to compare.

Return Value

A **System.UInt64** equal to *val1* if *val1* is less than or equal to *val2*; otherwise, the return value is equal to *val2*.

Description

This method is not CLS-compliant. For a CLS-compliant alternative, use **System.Math.Min**(**System.Decimal**, **System.Decimal**).

Math.Min(System.Single, System.Single) Method

[ILASM]

```
.method public hidebysig static float32 Min(float32 val1,
float32 val2)
```

[C#]

```
public static float Min(float val1, float val2)
```

Summary

Returns the lesser of two specified **System.Single** values.

Parameters

Parameter	**Description**
val1	The first of two specified **System.Single** values to compare.
val2	The second of two specified **System.Single** values to compare.

Return Value

A **System.Single** equal to *val1* if *val1* is less than or equal to *val2*; otherwise, the return value is equal to *val2*. If *val1*, *val2*, or both are equal to **System.Single.NaN**, **System.Single.NaN** is returned.

Math.Min(System.Double, System.Double) Method

[ILASM]

```
.method public hidebysig static float64 Min(float64 val1, float64 val2)
```

[C#]

```
public static double Min(double val1, double val2)
```

Summary

Returns the lesser of two specified **System.Double** values.

Parameters

Parameter	Description
val1	The first of two specified **System.Double** values to compare.
val2	The second of two specified **System.Double** values to compare.

Return Value

A **System.Double** equal to *val1* if *val1* is less than or equal to *val2*; otherwise, the return value is equal to *val2*. If *val1*, *val2*, or both are equal to **System.Double.NaN**, **System.Double.NaN** is returned.

Math.Min(System.Decimal, System.Decimal) Method

[ILASM]

```
.method public hidebysig static decimal Min(decimal val1,
decimal val2)
```

[C#]

```
public static decimal Min(decimal val1, decimal val2)
```

Summary

Returns the lesser of two specified **System.Decimal** values.

Parameters

Parameter	Description
val1	The first of two specified **System.Decimal** values to compare.
val2	The second of two specified **System.Decimal** values to compare.

Return Value

A **System.Decimal** equal to *val1* if *val1* is less than or equal to *val2*; otherwise, the return value is equal to *val2*.

Math.Pow(System.Double, System.Double) Method

```
[ILASM]
.method public hidebysig static float64 Pow(float64 x,
float64 y)

[C#]
public static double Pow(double x, double y)
```

Summary

Returns the specified **System.Double** raised to the specified power.

Parameters

Parameter	**Description**
x	A **System.Double** to be raised to a power.
y	A **System.Double** that specifies that power.

Return Value

A **System.Double** equal to *x* raised to the power *y*. The following table specifies the results if *x* or *y* is equal to **System.Double.NaN**, **System.Double.NegativeInfinity**, or **System.Double.PositiveInfinity**.

Parameter Values	**Returns**
x or *y* is equal to **System.Double.NaN**	**System.Double.NaN**.
x is equal to **System.Double.NegativeInfinity**	**System.Double.NegativeInfinity** if *y* is an odd integer; otherwise, **System.Double.PositiveInfinity**.
y is equal to **System.Double.NegativeInfinity**	0.
x is equal to **System.Double.PositiveInfinity**	0 if *y* is equal to **System.Double.NegativeInfinity**; otherwise, **System.Double.PositiveInfinity**.
y is equal to **System.Double.PositiveInfinity**	**System.Double.PositiveInfinity**.

Math.Round(System.Double) Method

[ILASM]

```
.method public hidebysig static float64 Round(float64 a)
```

[C#]

```
public static double Round(double a)
```

Summary

Returns the integer nearest the specified **System.Double**.

Parameters

Parameter	**Description**
a	A **System.Double** to be rounded.

Return Value

A **System.Double** containing the value of the integer nearest *a*. If *a* is exactly halfway between two integers, one of which is even and the other odd, then the even integer is returned.

Description

The behavior of this method follows IEEE Standard 754, section 4.1.

Example

The following example demonstrates using the **System.Math.Round**(**System.Double**) method.

[C#]

```
using System;

public class MathRoundExample
{

   public static void Main()
   {

      Double d1 = Math.Round(4.4);
      Double d2 = Math.Round(4.5);
      Double d3 = Math.Round(4.6);
      Console.WriteLine("Math.Round(4.4) returns {0}", d1);
      Console.WriteLine("Math.Round(4.5) returns {0}", d2);
      Console.WriteLine("Math.Round(4.6) returns {0}", d3);
```

```
    }

}
```

The output is

```
Math.Round(4.4) returns 4


Math.Round(4.5) returns 4


Math.Round(4.6) returns 5
```

Math.Round(System.Double, System.Int32) Method

```
[ILASM]
.method public hidebysig static float64 Round(float64
value, int32 digits)

[C#]
public static double Round(double value, int digits)
```

Summary

Returns the number nearest the specified **System.Double** within the specified precision.

Parameters

Parameter	Description
value	A **System.Double** to be rounded.
digits	A **System.Int32** containing the value of the number of significant fractional digits (precision) in the return value. This number is required to be greater than or equal to 0 and less than or equal to 15.

Return Value

A **System.Double** containing the value of the number nearest *value* with a precision equal to *digits*. If the digit in *value* that is in the $10^{-(digits + 1)}$ place is equal to 5 and there are no non-zero numbers in any less significant place, then the digit in the $10^{-digits}$ place will be unchanged if it is even, else it will be set to the closest even integer value in the direction of the digit in the $10^{-(digits + 1)}$ place. If the precision of *value* is less than *digits*, then *value* is returned unchanged. If *digits* is zero, this method behaves in the same manner as **System.Math.Round** (*value*).

Description

The behavior of this method follows IEEE Standard 754, section 4.1.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*digits* < 0.

	digits > 15.

Example

The following example demonstrates using the **System.Math.Round**(**System.Double**, **System.Int32**) method.

[C#]

```
using System;

public class MathRoundExample
{

   public static void Main()
   {

      Double d1 = Math.Round(3.44,1);
      Double d2 = Math.Round(3.45,1);
      Double d3 = Math.Round(3.55,1);
      Console.WriteLine("Math.Round(3.44, 1) returns {0}",
d1);
      Console.WriteLine("Math.Round(3.45, 1) returns {0}",
d2);
      Console.WriteLine("Math.Round(3.55, 1) returns {0}",
d3);

   }

}
```

The output is

```
Math.Round(3.44, 1) returns 3.4


Math.Round(3.45, 1) returns 3.4


Math.Round(3.55, 1) returns 3.6
```

1

Math.Round(System.Decimal) Method

[ILASM]

```
.method public hidebysig static decimal Round(decimal d)
```

[C#]

```
public static decimal Round(decimal d)
```

Summary

Returns the integer nearest the specified **System.Decimal**.

Parameters

Parameter	Description
d	A **System.Decimal** to be rounded.

Return Value

A **System.Decimal** containing the value of the integer nearest *d*. If *d* is exactly halfway between two integers, one of which is even and the other odd, then the even integer is returned.

Description

The behavior of this method follows IEEE Standard 754, section 4.1.

Example

The following example demonstrates using the **System.Math.Round**(**System.Decimal**) method.

[C#]

```
using System;

public class MathRoundExample
{

   public static void Main()
   {

      Double d1 = Math.Round(4.4);
      Double d2 = Math.Round(4.5);
      Double d3 = Math.Round(4.6);
      Console.WriteLine("Math.Round(4.4) returns {0}", d1);
      Console.WriteLine("Math.Round(4.5) returns {0}", d2);
      Console.WriteLine("Math.Round(4.6) returns {0}", d3);
```

```
    }

}
```

The output is

```
Math.Round(4.4) returns 4


Math.Round(4.5) returns 4


Math.Round(4.6) returns 5
```

Math.Sign(System.SByte) Method

```
[ILASM]
.method public hidebysig static int32 Sign(int8 value)

[C#]
public static int Sign(sbyte value)
```

Summary

Returns a value indicating the sign of the specified **System.SByte**.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
value	A **System.SByte** whose sign is to be determined.

Return Value

A **System.Int32** indicating the sign of *value*.

Number	Description
-1	*value* < 0.
0	*value* == 0.
1	*value* > 0.

Description

This method is not CLS-compliant. For a CLS-compliant alternative, use **System.Math.Sign**(**System.Int16**).

Math.Sign(System.Int16) Method

[ILASM]

```
.method public hidebysig static int32 Sign(int16 value)
```

[C#]

```
public static int Sign(short value)
```

Summary

Returns a value indicating the sign of the specified **System.Int16**.

Parameters

Parameter	Description
value	A **System.Int16** whose sign is to be determined.

Return Value

A **System.Int32** indicating the sign of *value*.

Number	Description
-1	*value* < 0.
0	*value* == 0.
1	*value* > 0.

Math.Sign(System.Int32) Method

```
[ILASM]
.method public hidebysig static int32 Sign(int32 value)

[C#]
public static int Sign(int value)
```

Summary

Returns a value indicating the sign of the specified **System.Int32**.

Parameters

Parameter	Description
value	A **System.Int32** whose sign is to be determined.

Return Value

A **System.Int32** indicating the sign of *value*.

Number	Description
-1	*value* < 0.
0	*value* == 0.
1	*value* > 0.

Math.Sign(System.Int64) Method

```
[ILASM]
.method public hidebysig static int32 Sign(int64 value)

[C#]
public static int Sign(long value)
```

Summary

Returns a value indicating the sign of the specified **System.Int64**.

Parameters

Parameter	Description
value	A **System.Int64** whose sign is to be determined.

Return Value

A **System.Int32** indicating the sign of *value*.

Number	Description
-1	*value* < 0.
0	*value* == 0.
1	*value* > 0.

Math.Sign(System.Single) Method

[ILASM]
```
.method public hidebysig static int32 Sign(float32 value)
```

[C#]
```
public static int Sign(float value)
```

Summary

Returns a value indicating the sign of the specified **System.Single**.

Parameters

Parameter	Description
value	A **System.Single** whose sign is to be determined.

Return Value

A **System.Int32** indicating the sign of value.

Number	Description
-1	*value* < 0.
0	*value* == 0.
1	*value* > 0.

Exceptions

Exception	Condition
System.ArithmeticException	*value* is equal to **System.Single.NaN**.

Math.Sign(System.Double) Method

```
[ILASM]
.method public hidebysig static int32 Sign(float64 value)

[C#]
public static int Sign(double value)
```

Summary

Returns a value indicating the sign of the specified **System.Double**.

Parameters

Parameter	Description
value	A **System.Double** whose sign is to be determined.

Return Value

A **System.Int32** indicating the sign of *value*.

Number	Description
-1	*value* < 0.
0	*value* == 0.
1	*value* > 0.

Exceptions

Exception	Condition
System.ArithmeticException	*value* is equal to **System.Double.NaN**.

Example

The following example demonstrates using the **System.Math.Sign**(**System.Double**) method.

```
[C#]

using System;
```

```
public class MathSignExample
{

   public static void Main()
   {

      Double d1 = Math.Sign(4.4);
      Double d2 = Math.Sign(0.0);
      Double d3 = Math.Sign(-4.5);
      Console.WriteLine("Math.Sign(4.4) returns {0}", d1);
      Console.WriteLine("Math.Sign(0.0) returns {0}", d2);
      Console.WriteLine("Math.Sign(-4.5) returns {0}", d3);

   }

}
```

The output is

```
Math.Sign(4.4) returns 1


Math.Sign(0.0) returns 0


Math.Sign(-4.5) returns -1
```

Math.Sign(System.Decimal) Method

[ILASM]

```
.method public hidebysig static int32 Sign(decimal value)
```

[C#]

```
public static int Sign(decimal value)
```

Summary

Returns a value indicating the sign of the specified **System.Decimal**.

Parameters

Parameter	Description
value	A **System.Decimal** number whose sign is to be determined.

Return Value

A **System.Int32** indicating the sign of *value*.

Number	Description
-1	*value* < 0.
0	*value* == 0.
1	*value* > 0.

Math.Sin(System.Double) Method

[ILASM]
```
.method public hidebysig static float64 Sin(float64 a)
```

[C#]
```
public static double Sin(double a)
```

Summary

Returns the sine of the specified **System.Double** that represents an angle.

Parameters

Parameter	Description
a	A **System.Double** containing the value of an angle measured in radians.

Return Value

A **System.Double** containing the value of the sine of *a*. If *a* is equal to **System.Double.NaN**, **System.Double.NegativeInfinity**, or **System.Double.PositiveInfinity**, returns **System.Double.NaN**.

Description

[*Note:*

Example

The following example demonstrates using the **System.Math.Sin** method.

[C#]

```
using System;

public class MathSinExample
{

   public static void Main()
   {

      Double d1 = Math.Sin(0);
      Double d2 = Math.Sin(Math.PI/2.0);
      Console.WriteLine("Math.Sin(0) returns {0}", d1);
      Console.WriteLine("Math.Sin(Math.PI/2.0) returns
{0}", d2);

   }

}
```

The output is

```
Math.Sin(0) returns 0


Math.Sin(Math.PI/2.0) returns 1
```

Math.Sinh(System.Double) Method

[ILASM]

```
.method public hidebysig static float64 Sinh(float64 value)
```

[C#]

```
public static double Sinh(double value)
```

Summary

Returns the hyperbolic sine of the specified **System.Double** that represents an angle.

Parameters

Parameter	Description
value	A **System.Double** containing the value of an angle measured in radians.

Return Value

A **System.Double** containing the value of the hyperbolic sine of *value*. If *value* is equal to **System.Double.NegativeInfinity**, **System.Double.PositiveInfinity**, or **System.Double.NaN**, returns a **System.Double** equal to *value*.

Description

[*Note:*

Example

The following example demonstrates using the **System.Math.Sinh** method.

[C#]

```
using System;

public class MathSinhExample
{

   public static void Main()
   {

      Double d1 = Math.Sinh(0);
      Double d2 = Math.Sinh(Math.PI);
      Console.WriteLine("Math.Sinh(0) returns {0}", d1);
```

```
            Console.WriteLine("Math.Sinh(Math.PI) returns {0}",
d2);

    }

}
```

The output is

```
Math.Sinh(0) returns 0


Math.Sinh(Math.PI) returns 11.5487393572577
```

Math.Sqrt(System.Double) Method

[ILASM]

```
.method public hidebysig static float64 Sqrt(float64 d)
```

[C#]

```
public static double Sqrt(double d)
```

Summary

Returns the square root of the specified **System.Double**.

Parameters

Parameter	Description
d	A **System.Double**.

Return Value

A **System.Double** whose value is indicated as follows:

Condition	Returns
d == 0	A **System.Double** containing the positive square root of *d*.
d < 0 *d* is equal to **System.Double.NegativeInfinity**. *d* is equal to **System.Double.NaN**.	**System.Double.NaN**.
d is equal to **System.Double.PositiveInfinity**	**System.Double.PositiveInfinity**.

Example

The following example demonstrates using the **System.Math.Sqrt** method.

[C#]

```
using System;
```

```
public class MathSqrtExample
{

   public static void Main()
   {

      Double d1 = Math.Sqrt(16.0);
      Double d2 = Math.Sqrt(0.0);
      Double d3 = Math.Sqrt(-10.0);
      Console.WriteLine("Math.Sqrt(16.0) returns {0}", d1);
      Console.WriteLine("Math.Sqrt(0.0) returns {0}", d2);
      Console.WriteLine("Math.Sqrt(-10.0) returns {0}",
d3);

   }

}
```

The output is

```
Math.Sqrt(16.0) returns 4


Math.Sqrt(0.0) returns 0


Math.Sqrt(-10.0) returns NaN
```

Math.Tan(System.Double) Method

[ILASM]

```
.method public hidebysig static float64 Tan(float64 a)
```

[C#]

```
public static double Tan(double a)
```

Summary

Returns the tangent of the specified **System.Double** that represents an angle.

Parameters

Parameter	Description
a	A **System.Double** that represents an angle measured in radians.

Return Value

A **System.Double** containing the value of the tangent of *a*. If a is equal to **System.Double.NaN**, **System.Double.NegativeInfinity**, or **System.Double.PositiveInfinity**, returns **System.Double.NaN**.

Description

[*Note:*

Example

The following example demonstrates using the **System.Math.Tan** method.

[C#]

```
using System;

public class MathTanExample
{

   public static void Main()
   {

      Double d1 = Math.Tan(0);
      Double d2 = Math.Tan(Math.PI/2.0);
      Console.WriteLine("Math.Tan(0) returns {0}", d1);
      Console.WriteLine("Math.Tan(Math.PI/2.0) returns
{0}", d2);
```

```
    }
}
```

The output is

```
Math.Tan(0) returns 0


Math.Tan(Math.PI/2.0) returns 1.63317787283838E+16
```

Math.Tanh(System.Double) Method

[ILASM]
```
.method public hidebysig static float64 Tanh(float64 value)
```

[C#]
```
public static double Tanh(double value)
```

Summary

Returns the hyperbolic tangent of the specified **System.Double** that represents an angle.

Parameters

Parameter	Description
value	A **System.Double** that represents an angle measured in radians.

Return Value

A **System.Double** containing the value of the hyperbolic tangent of *value*. If *value* is equal to **System.Double.NegativeInfinity**, returns -1. If value is equal to **System.Double.PositiveInfinity**, returns 1. If value is equal to **System.Double.NaN**, returns **System.Double.NaN**.

Description

[*Note:*

Example

The following example demonstrates using the **System.Math.Tanh** method.

[C#]

```
using System;

public class MathTanhExample
{

   public static void Main()
   {

      Double d1 = Math.Tanh(0);
      Double d2 = Math.Tanh(Math.PI);
      Console.WriteLine("Math.Tanh(0) returns {0}", d1);
```

```
            Console.WriteLine("Math.Tanh(Math.PI) returns {0}",
d2);

      }

}
```

The output is

```
Math.Tanh(0) returns 0


Math.Tanh(Math.PI) returns 0.99627207622075
```