Namespace FireBlade.CsTools

Classes

StringExtensions

Provides extension methods for strings.

Enums

StringCasing

Specifies the casing of a string.

Enum StringCasing

Namespace: FireBlade.CsTools

Assembly: CsTools.dll

Specifies the casing of a string.

```
public enum StringCasing
```

Fields

`Alternating = 8`

Every 1st character is uppercase, and every 2nd character is lowercase, or reverse. Cannot be set for SetCasing(string, StringCasing), but can be retrieved through GetCasing(string).

`AlternatingNormal = 9`

Every 1st character is uppercase, and every 2nd character is lowercase, or reverse. Cannot be retrieved through GetCasing(string) (see Alternating instead).

`AlternatingReverse = 10`

Every 2nd character is uppercase, and every 1st character is lowercase, or reverse. Cannot be retrieved through GetCasing(string) (see Alternating instead).

`Camel = 5`

The same as Pascal, but the first word is not capitalized.

`Inverse = 7`

The inverse of Title.

`Lower = 2`

All characters are lowercase.

`Other = 0`

The capitalization mode doesn't match any default mode. Cannot be set for SetCasing(string, StringCasing), but can be retrieved through GetCasing(string).

`Pascal = 4`

The first letter of every word is capitalized, but words are NOT seperated by spaces.

`Snake = 6`

Words can have any captialization, but they are seperated by underscores.

`Title = 3`

The first letter of every word is capitalized and words are seperated by spaces.

`Upper = 1`

All characters are uppercase.

Class StringExtensions

Namespace: FireBlade.CsTools

Assembly: CsTools.dll

Provides extension methods for strings.

```
public static class StringExtensions
```

Inheritance

object ← StringExtensions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

GetCasing(string)

Gets the casing of a string.

```
public static StringCasing GetCasing(this string s)
```

Parameters

`s` string

The string to check.

Returns

StringCasing

The casing of the string, or Other if a match wasn't found.

GetString(IEnumerable<char>)

Creates a string from a char collection.

```
public static string GetString(this IEnumerable<char> chars)
```

Parameters

`chars` IEnumerable<char>

The characters to convert.

Returns

string

The collection `chars`, converted to a string.

IsNotNullOrEmpty(string?)

Indicates whether the string is not null or an empty string (Empty).

```
public static bool IsNotNullOrEmpty(this string? s)
```

Parameters

`s` string

The string to check.

Returns

bool

true if the string is not null or an empty string; otherwise, false.

IsNotNullOrWhiteSpace(string?)

Indicates whether the string is not null, empty (Empty), or doesn't consist of only whitespace characters.

```
public static bool IsNotNullOrWhiteSpace(this string? s)
```

Parameters

`s` string

The string to check.

Returns

bool

true if the string is not null or an empty string, or if the string doesn't consist exclusively of whitespace characters; otherwise, false.

IsNullOrEmpty(string?)

Indicates whether the string is null or an empty string (Empty).

```
public static bool IsNullOrEmpty(this string? s)
```

Parameters

`s` string

The string to check.

Returns

bool

true if the string is null or an empty string; otherwise, false.

IsNullOrWhiteSpace(string?)

Indicates whether the string is null, empty (Empty), or consists of only whitespace characters.

```
public static bool IsNullOrWhiteSpace(this string? s)
```

Parameters

`s` string

The string to check.

Returns

bool

true if the string is null or an empty string, or if the string consists exclusively of whitespace characters; otherwise, false.

IsPalindrome(string)

Determines whether the string reads the same forwards and backwards.

```
public static bool IsPalindrome(this string s)
```

Parameters

`s` string

The string to check.

Returns

bool

true if the string reads the same backwards; otherwise, false.

ParseNumber<TNum>(string)

Parses the string into a number.

```
public static TNum ParseNumber<TNum>(this string s) where TNum : INumber<TNum>
```

Parameters

`s` string

The string to parse.

Returns

TNum

The string `s`, converted to a `TNum`.

Type Parameters

`TNum`

The number type.

ParseNumber<TNum>(string, NumberStyles)

Parses the string into a number.

```
public static TNum ParseNumber<TNum>(this string s, NumberStyles style) where TNum
: INumber<TNum>
```

Parameters

`s` string

The string to parse.

`style` NumberStyles

A bitwise combination of styles present in the string.

Returns

TNum

The string `s`, converted to a `TNum`.

Type Parameters

`TNum`

The number type.

ParseNumber<TNum>(string, NumberStyles, IFormatProvider)

Parses the string into a number.

```
public static TNum ParseNumber<TNum>(this string s, NumberStyles style, IFormatProvider
provider) where TNum : INumber<TNum>
```

Parameters

`s` string

The string to parse.

`style` NumberStyles

A bitwise combination of styles present in the string.

`provider` IFormatProvider

An object that provides culture-specific formatting information.

Returns

TNum

The string `s`, converted to a `TNum`.

Type Parameters

`TNum`

The number type.

ParseNumber<TNum>(string, IFormatProvider)

Parses the string into a number.

```
public static TNum ParseNumber<TNum>(this string s, IFormatProvider provider) where TNum
```

```
: INumber<TNum>
```

Parameters

`s` string

The string to parse.

`provider` IFormatProvider

An object that provides culture-specific formatting information.

Returns

TNum

The string `s`, converted to a `TNum`.

Type Parameters

`TNum`

The number type.

SetCasing(string, StringCasing)

Sets the casing of the string.

```
public static string SetCasing(this string s, StringCasing targetCasing)
```

Parameters

`s` string

The string to change.

`targetCasing` StringCasing

The new casing.

Returns

string

The string `s`, converted to the casing specified in `targetCasing`.

TryParseNumber<TNum>(string, NumberStyles, out TNum)

Tries to parse the string into a number.

```
public static bool TryParseNumber<TNum>(this string s, NumberStyles style, out TNum result)
where TNum : INumber<TNum>
```

Parameters

`s` string

The string to try to parse.

`style` NumberStyles

A bitwise combination of styles present in the string.

`result` TNum

The result, if successful.

Returns

bool

true if the conversion succeeded; otherwise, false.

Type Parameters

`TNum`

The number type.

TryParseNumber<TNum>(string, IFormatProvider, NumberStyles, out TNum)

Tries to parse the string into a number.

```
public static bool TryParseNumber<TNum>(this string s, IFormatProvider provider,
NumberStyles style, out TNum result) where TNum : INumber<TNum>
```

Parameters

`s` string

The string to try to parse.

`provider` IFormatProvider

An object that contains culture-specific formatting information.

`style` NumberStyles

A bitwise combination of styles present in the string.

`result` TNum

The result, if successful.

Returns

bool

true if the conversion succeeded; otherwise, false.

Type Parameters

`TNum`

The number type.

TryParseNumber<TNum>(string, IFormatProvider, out TNum)

Tries to parse the string into a number.

```
public static bool TryParseNumber<TNum>(this string s, IFormatProvider provider, out TNum
result) where TNum : INumber<TNum>
```

Parameters

`s` string

The string to try to parse.

`provider` IFormatProvider

An object that contains culture-specific formatting information.

`result` TNum

The result, if successful.

Returns

bool

true if the conversion succeeded; otherwise, false.

Type Parameters

`TNum`

The number type.

TryParseNumber<TNum>(string, out TNum)

Tries to parse the string into a number.

```
public static bool TryParseNumber<TNum>(this string s, out TNum result) where TNum
: INumber<TNum>
```

Parameters

`s` string

The string to try to parse.

`result` TNum

The result, if successful.

Returns

bool

true if the conversion succeeded; otherwise, false.

Type Parameters

TNum

The number type.

Namespace FireBlade.CsTools.Numbers

Classes

MathTools

Provides mathematical tools.

NumberExtensions

Extends number types such as int, float, or double.

Range<TNum>

Represents a range of numbers.

Class MathTools

Namespace: FireBlade.CsTools.Numbers

Assembly: CsTools.dll

Provides mathematical tools.

```
public static class MathTools
```

Inheritance

object ← MathTools

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

IsEven<TNum>(TNum)

Checks if a number is even.

```
public static bool IsEven<TNum>(TNum val) where TNum : INumber<TNum>
```

Parameters

`val` TNum

The number to check.

Returns

bool

true if the value is even; otherwise, false.

Type Parameters

`TNum`

The number type.

IsOdd<TNum>(TNum)

Checks if a number is odd.

```
public static bool IsOdd<TNum>(TNum val) where TNum : INumber<TNum>
```

Parameters

`val` TNum

The number to check.

Returns

bool

true if the value is odd; otherwise, false.

Type Parameters

`TNum`

The number type.

Map<TNum>(TNum, Range<TNum>, Range<TNum>)

Maps a value from the input range to the output range.

```
public static TNum Map<TNum>(TNum val, Range<TNum> input, Range<TNum> output) where TNum
: INumber<TNum>
```

Parameters

`val` TNum

The value to map.

`input` Range<TNum>

The input range.

`output` Range<TNum>

The output range.

Returns

TNum

`val`, mapped from the `input` range to the `output` range.

Type Parameters

`TNum`

The number type.

Mod<TNum>(TNum, TNum)

Returns the true modulo of a number. Works with negative numbers correctly.

```
public static TNum Mod<TNum>(TNum value, TNum modulus) where TNum : INumber<TNum>
```

Parameters

`value` TNum

The value to apply the modulus operator on.

`modulus` TNum

The modulus to apply.

Returns

TNum

The `modulus` remainder of `value` after division.

Type Parameters

`TNum`

The number type.

Class NumberExtensions

Namespace: FireBlade.CsTools.Numbers

Assembly: CsTools.dll

Extends number types such as int, float, or double.

```
public static class NumberExtensions
```

Inheritance

object ← NumberExtensions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

IsInRange<TNum>(TNum, Range<TNum>)

Checks if the specified number is in a range.

```
public static bool IsInRange<TNum>(this TNum val, Range<TNum> range) where TNum
: INumber<TNum>
```

Parameters

`val` TNum

The value to test.

`range` Range<TNum>

The range to check the value against.

Returns

bool

true if the value is in range; otherwise, false.

Type Parameters

`TNum`

The number type.

Exceptions

ArgumentOutOfRangeException

The minimum value is larger than the maximum value.

IsInRange<TNum>(TNum, decimal, decimal)

Checks if the specified number is in a range.

```
public static bool IsInRange<TNum>(this TNum val, decimal min, decimal max) where TNum
: INumber<TNum>
```

Parameters

`val` TNum

The value to test.

`min` decimal

The minimum value of the range.

`max` decimal

The maximum value of the range.

Returns

bool

true if the value is in range; otherwise, false.

Type Parameters

`TNum`

The number type.

Exceptions

ArgumentOutOfRangeException

The minimum value is larger than the maximum value.

IsInRange<TNum>(TNum, double, double)

Checks if the specified number is in a range.

```
public static bool IsInRange<TNum>(this TNum val, double min, double max) where TNum
: INumber<TNum>
```

Parameters

`val` TNum

The value to test.

`min` double

The minimum value of the range.

`max` double

The maximum value of the range.

Returns

bool

true if the value is in range; otherwise, false.

Type Parameters

`TNum`

The number type.

Exceptions

ArgumentOutOfRangeException

The minimum value is larger than the maximum value.

IsInRange<TNum>(TNum, int, int)

Checks if the specified number is in a range.

```
public static bool IsInRange<TNum>(this TNum val, int min, int max) where TNum
: INumber<TNum>
```

Parameters

`val` TNum

The value to test.

`min` int

The minimum value of the range.

`max` int

The maximum value of the range.

Returns

bool

true if the value is in range; otherwise, false.

Type Parameters

`TNum`

The number type.

Exceptions

ArgumentOutOfRangeException

The minimum value is larger than the maximum value.

IsInRange<TNum>(TNum, float, float)

Checks if the specified number is in a range.

```
public static bool IsInRange<TNum>(this TNum val, float min, float max) where TNum
: INumber<TNum>
```

Parameters

`val` TNum

The value to test.

`min` float

The minimum value of the range.

`max` float

The maximum value of the range.

Returns

bool

true if the value is in range; otherwise, false.

Type Parameters

`TNum`

The number type.

Exceptions

ArgumentOutOfRangeException

The minimum value is larger than the maximum value.

IsInRange<TNum>(TNum, TNum, TNum)

Checks if the specified number is in a range.

```
public static bool IsInRange<TNum>(this TNum val, TNum min, TNum max) where TNum
: INumber<TNum>
```

Parameters

`val` TNum

The value to test.

`min` TNum

The minimum value of the range.

`max` TNum

The maximum value of the range.

Returns

bool

true if the value is in range; otherwise, false.

Type Parameters

`TNum`

The number type.

Exceptions

ArgumentOutOfRangeException

The minimum value is larger than the maximum value.

Class Range<TNum>

Namespace: FireBlade.CsTools.Numbers

Assembly: CsTools.dll

Represents a range of numbers.

```
public class Range<TNum> where TNum : INumber<TNum>
```

Type Parameters

`TNum`

The number type.

Inheritance

object ← Range<TNum>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

Range(TNum, TNum)

Creates a new instance of the Range<TNum> class.

```
public Range(TNum min, TNum max)
```

Parameters

`min` TNum

`max` TNum

Properties

Length

Gets the difference between the Maximum value and the Minimum value.

```
public TNum Length { get; }
```

Property Value

TNum

Maximum

Gets or sets the maximum value of the Range<TNum>.

```
public TNum Maximum { get; set; }
```

Property Value

TNum

Exceptions

ArgumentOutOfRangeException

The minimum value is larger than the maximum value.

Minimum

Gets or sets the minimum value of the Range<TNum>.

```
public TNum Minimum { get; set; }
```

Property Value

TNum

Exceptions

ArgumentOutOfRangeException

The minimum value is larger than the maximum value.

Methods

IsInRange(TNum)

Checks if the specified number is in the range.

```
public bool IsInRange(TNum val)
```

Parameters

`val` TNum

The value to test.

Returns

bool

true if the value is in range; otherwise, false.

Exceptions

ArgumentOutOfRangeException

The minimum value is larger than the maximum value.