**String class**.

**Properties**

**Length** : Gets length of String

**Char**: string[i] returns char at index i.

**Static members**

**String.Compare**(strA, StrB, *ignoreCase, indexesToStart, CultureInfo, CompareOptionsEnum*) Compares 2 strings optionally passing in where to start and whether to ignore case etc. Returns int indicating which string is smaller

**String.CompareOrdinal**(strA, strB, *indexesToStart, length*) Compare strings by their ascii code. So A < a.

**String.Concat**(*strings*) Joins number of strings together. And returns string

String.Equals(str1, str2) Determines if 2 string are same.

**String.Format**(format, *strings*) Formats string(s) and returns formatted string

**String.Copy**(str) Creates new instance of str.

**String.Intern** Retrieves interned string

**String.IsInterned** Is interned.

**String.IsNullorWhiteSpace/String.IsNullorEmpty**

**String.Empty** Returns empty string.

**String.Join**(stringSeperator, strings[]) Joins elements in array using seperator

**Instance Members**

**Contains**(str) True if str is within string

**EndsWith**(str, *ignoreCase*, *CompareOptionsEnum, CultureInfo*) Determines if end of string = str.

**StartsWith**(str, *ignoreCase*, *CompareOptionsEnum, CultureInfo*) Determines if start of string = str.

**IndexOf**(str, *startIndex, endIndex, ignoreCase, CompareOptionsEnum*). Reports first occurrence of str.

**IndexOfAny**(str, char[]) Reports first occurrence in array of chars

**LastIndexOf**(..) Like IndexOf except starts from the end of string

**LastIndexOfAny**(str, char[]) Like IndexOfAny excepts starts from last array entry.

**Substring**(index, *startIndex*) Retrieves substring from string starting at index

**Insert**(startIndex, str) returns new string with str inserted at startIndex

**Remove**(index, *endIndex*) Returns a string with certain length of it removed

**Replace**(str1, str2) Returns new string will all occurrences of str1 replaced with str2

**Split**(delimiters[]) Splits string into array based on delimiters

**PadLeft/PadRight**(int, char) pads string with char a specified number of times

**ToLower()/ToUpper()/ToString()**

**Trim/TrimStart/TrimEnd**

**Array Class**

**Properties**

**Length**

**Rank** # of dimensions

**Static members**

**BinarySearch**(Array, value, *index, length, IComparer*) Searches for value using Binary search

**Sort**(Array,*index, length, IComparer*) Sorts using Insertion, Heap then Quicksort.

**Clear**(Array, index, length) Clears Array of length starting at index.

**ConvertAll**(Tin,Tout) Converts elements in array from Tin type to Tout.

**Copy**(Array, *index*, Array2, *index, length*) Copies Array to Array2

**CreateInstance**(T, length, *length[]*) Creates 1 dimensional (or multiDim) array of T with length.

**Exists**(Array, predicate) Determines if Array passes predicate.

**TrueForAll**(Array, predicate) returns true if all elements in array pass predicate.

**Find**(Array, predicate) Returns first occurrence of predicate.

**FindLast**(Array, predicate) Returns last occurrence of predicate.

**FindAll**(Array, predicate) Returns all occurrence’s of predicate.

**FindIndex**(Array, predicate) Returns index of first occurrence of predicate.

**FindLastIndex**(Array, predicate) Returns last index of occurrence of predicate.

**ForEach**(Array, action) performs action on every entry in Array.

**IndexOf**(Array, value, *start, length*) returns index of value in array

**LastIndexOf**(Array, value, *start, length*) returns last index of value in array

**Resize**(Array, newSize) Resizes Array to new size

**Reverse**(Array, *start, length*) reverses array.

**Instance Members**

**CopyTo**(Array, *index*) Copies array to Array starting at index

**GetLength, GetUpperBound, GetLowerBound**

**GetValue**(index) Returns value at index.

**SetValue**(value, index) Sets value at index

**CreateInstance**(T, length) Creates and returns new array

**Properties**

**Dispatcher** :Gets dispatcher to send actions to from background thread.

**Instance Methods**

**ClearValue**(dp) Clears dp’s value

**CoerceValue**(dp) Coerces value of dp

**GetValue**(dp) Gets current value of dp

**ReadLocalValue**(dp) returns local value of dp if exists

**SetValue**(dp, value) Sets dp with new value

**InvalidateProperty**(dp) Reevaluates the value for the dp

**OnPropertyChanged**()

**DependencyObject**

|  |  |
| --- | --- |
| **Members** | **Determining value** |
| **Properties**  **Dispatcher** :Gets dispatcher to send actions to from background thread.  **Instance Methods**  **ClearValue**(dp) Clears dp’s value  **CoerceValue**(dp) Coerces value of dp  **GetValue**(dp) Gets current value of dp  **ReadLocalValue**(dp) returns local value of dp if exists  **SetValue**(dp, value) Sets dp with new value  **InvalidateProperty**(dp) Reevaluates the value for the dp  **OnPropertyChanged**() | Default base value for property are set by following factors  0. Default value  1.Property value inheritance (if the FrameworkPropertyMetadata.Inherits is set)  2. Theme style setters  3. Theme style triggers  4. Style setters  5. Template triggers  6. Style triggers  7. Parent template  8. Parent template trigger  9. Local value    When getting final value you go through the following if they pertain  1. Get value from above  2. Evaluate expression if any  3. Run animation if any  4. Run CoerceValueCallback to correct value  5. Run ValidateValueCallback to correct value |

Property value inheritance (if the FrameworkPropertyMetadata.Inherits is set)

**DependencyProperty Class**

**Properties**

**UnsetValue**

**ValidateValueCallback**

**Static Members**

**Register**

**RegisterAttached**

**RegisterReadOnly**

**RegisterAttachedReadOnly**

**Instance Members**

**GetMetadata(Dp)**

**OverrideMetadata**

**FrameworkElement**

**Properties**

Height, Width, Margin, ToolTip, Horizontal/VerticalAlignment, Max/Min/Actual Height/weight

**DataContext:** The data for binding

**Resources:** Resource dictionary

**Style:** Style used when it’s rendered

**Methods**

**GetBindingExpression:** returns BindingExpression of elements binding if it exists

**SetBinding:** Attaches binding to element

**Find/TryFindResource**

**MeasureOveride(size):** Measures child controls

**ArrangeOverride(size)**: Arranges children(Used by Panel class)

**Control**

**Properties**

Background/Foreground

BorderThickness/Brush

Font props

Horizontal/VerticalContentAlignment

Padding

Template: gets or sets the controls template