Namespace ConsoleAppVisuals

Classes

Core

The Core class is a collection of low-level methods responsible of all the interactions between the library and the console (e.g. changing colors, writing text, etc.).

Window

The Window class is a collection of methods to manage visual elements stored in itself. You may add, remove, or render elements in the console after adding them to the Window class.

Class Core

Namespace: ConsoleAppVisuals

Assembly: ConsoleAppVisuals.dll

The Core class is a collection of low-level methods responsible of all the interactions between the library and the console (e.g. changing colors, writing text, etc.).

```
public static class Core
```

Inheritance

object ← Core

Inherited Members

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

Remarks

For more information, consider visiting the documentation available here.

Fields

NEGATIVE_ANCHOR

Defines the negative anchor to put inside a string to be recognized as negative.

The following line will put " negative " in negative on the screen and not print the "/neg" anchors.

```
string str = "This is a /neg negative /neg string";
```

```
public const string NEGATIVE_ANCHOR = "/neg"
```

Field Value

string

Properties

ColorPanel

Gets the current color panel used in the console.

```
public static (ConsoleColor, ConsoleColor) ColorPanel { get; }
```

Property Value

(ConsoleColor, ConsoleColor)

InitialColorPanel

Gets the initial color panel of the console.

```
public static (ConsoleColor, ConsoleColor) InitialColorPanel { get; }
```

Property Value

(ConsoleColor, ConsoleColor)

Methods

ApplyNegative(bool)

Switches the font and background colors of the console in order to apply a negative to highlight the text.

```
[Visual]
public static void ApplyNegative(bool negative = false)
```

Parameters

`negative` bool

If true, the text is highlighted.

Remarks

For more information, consider visiting the documentation available here.

BannerToString((string, string, string))

Converts a banner tuple into a string.

```
[Visual]
public static string BannerToString(this (string, string, string) banner)
```

Parameters

`banner` (string, string, string)

The banner tuple.

Returns

string

Converts the banner to a string.

Remarks

For more information, consider visiting the documentation available here.

ClearMultiplePositionedLines(Placement, int, params string[])

Clears a multiple lines in the console given a certain starting line.

```
[Visual]
public static void ClearMultiplePositionedLines(Placement placement, int line,
params string[] text)
```

Parameters

`placement` Placement

The placement of the paragraph.

`line` int

The height of the paragraph.

`text` string[]

The lines of the paragraph.

Remarks

For more information, consider visiting the documentation available here.

GetRandomColor()

Gets a random color from a list of predefined colors (the default ones).

```
public static ConsoleColor GetRandomColor()
```

Returns

ConsoleColor

A random color.

Remarks

For more information, consider visiting the documentation available here.

GetRangeAndRemoveNegativeAnchors(string)

Gets the range of a negative sequence in a string and remove the negative anchors.

```
public static (string, (int, int)?) GetRangeAndRemoveNegativeAnchors(this
string str)
```

Parameters

`str` string

The string to check.

Returns

(string, (int, int)?)

The string without the negative anchors and the range of the negative sequence.

Remarks

For more information, consider visiting the documentation available here.

InsertString(string, string, TextAlignment)

Inserts a specified string into another string, at a specified position.

```
public static string InsertString(this string inserted, string toInsert,
TextAlignment align = TextAlignment.Center)
```

Parameters

`inserted` string

The string that receives the other.

`toInsert` string

The string to insert.

`align` TextAlignment

The alignment of the string to insert.

Returns

string

The final string after computing.

Remarks

For more information, consider visiting the documentation available here.

IsScreenUpdated()

Checks if the screen has been updated (colors or dimensions have changed)

```
[Visual]
public static bool IsScreenUpdated()
```

Returns

bool

True if the screen has been updated, false otherwise.

Remarks

For more information, consider visiting the documentation available here.

LoadSavedColorPanel()

Loads the saved color panel (consider saving it beforehand)

```
[Visual]
public static void LoadSavedColorPanel()
```

Remarks

For more information, consider visiting the documentation available here.

LoadTerminalColorPanel()

Loads the terminal color panel.

```
[Visual]
public static void LoadTerminalColorPanel()
```

Remarks

For more information, consider visiting the documentation available here.

ResizeString(string, int, TextAlignment, bool)

Builds a new string with a specific size and a specific placement.

```
public static string ResizeString(this string str, int size, TextAlignment align =
TextAlignment.Center, bool truncate = true)
```

Parameters

`str` string

The string to build.

`size` int

The size of the string.

`align` TextAlignment

The alignment of the string.

`truncate` bool

If true, the string is truncated if it is too long.

Returns

string

The built string.

Remarks

For more information, consider visiting the documentation available here.

RestoreColorPanel()

Restores the default colors of the console with the initial color panel.

```
[Visual]
public static void RestoreColorPanel()
```

Remarks

For more information, consider visiting the documentation available here.

SaveColorPanel()

Saves the current color panel in a variable.

```
[Visual]
public static void SaveColorPanel()
```

Remarks

For more information, consider visiting the documentation available here.

SetBackgroundColor(ConsoleColor)

Changes the background color of the console.

```
[Visual]
public static void SetBackgroundColor(ConsoleColor color)
```

Parameters

`color` ConsoleColor

The new background color.

Remarks

For more information, consider visiting the documentation available here.

SetConsoleColors()

Set the console colors to the Core variables associated. It does not change the actual colors of the console.

```
[Visual]
public static void SetConsoleColors()
```

Remarks

For more information, consider visiting the documentation available here.

SetConsoleDimensions()

Set the dimensions of the console to the Core variables associated. It does not change the actual dimensions of the console.

```
[Visual]
public static void SetConsoleDimensions()
```

Remarks

For more information, consider visiting the documentation available here.

SetForegroundColor(ConsoleColor)

Changes the foreground color of the console.

```
[Visual]
public static void SetForegroundColor(ConsoleColor color)
```

Parameters

`color` ConsoleColor

The new color.

Remarks

For more information, consider visiting the documentation available here.

ToPlacement(TextAlignment)

Converts a TextAlignment into a Placement.

```
public static Placement ToPlacement(this TextAlignment align)
```

Parameters

`align` TextAlignment

The alignment to convert.

Returns

Placement

The converted alignment.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentException

Thrown when the alignment is not valid.

ToTextAlignment(Placement)

Converts a Placement into a TextAlignment.

```
public static TextAlignment ToTextAlignment(this Placement placement)
```

Parameters

`placement` Placement

The placement to convert.

Returns

TextAlignment

The converted placement.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentException

Thrown when the placement is not valid.

WriteContinuousString(string, int?, bool, int, int, int?, TextAlignment, Placement)

Writes a string continuously in the console. The string is written letter by letter on the console.

```
[Visual]
public static void WriteContinuousString(string str, int? line, bool negative
= false, int printTime = 2000, int additionalTime = 1000, int? length
= null, TextAlignment align = TextAlignment.Center, Placement placement
= Placement.TopCenter)
```

Parameters

`str` string

The string to write.

`line` int?

The line where the string is written in the console. If null, will be written where the cursor is.

`negative` bool

If true, the text is highlighted.

`printTime` int

The total time to write the string in ms.

`additionalTime` int

The additional time to wait after the string is written in ms.

`length` int?

The length of the string. If null, the length is the window width.

`align` TextAlignment

The alignment of the string.

`placement` Placement

The placement of the string.

Remarks

For more information, consider visiting the documentation available here.

WriteDebugMessage(Placement, params string[])

[Debugging purposes] It overwrites any text in the console at a specified placement to display a debug message.

```
[Visual]
public static void WriteDebugMessage(Placement placement = Placement.TopRight,
params string[] lines)
```

Parameters

`placement` Placement

The placement of the debug message.

`lines` string[]

The text to display.

Remarks

For more information, consider visiting the documentation available here.

WriteMultiplePositionedLines(bool, TextAlignment, Placement, bool, int?, params string[])

Writes multiple lines in the console given a certain placement and line.

```
[Visual]
public static void WriteMultiplePositionedLines(bool equalizeLengths = true,
TextAlignment align = TextAlignment.Center, Placement placement =
Placement.TopCenter, bool negative = false, int? line = null, params string[] text)
```

Parameters

`equalizeLengths` bool

Whether or not the lines of the paragraph should be equalized to the same length.

`align` TextAlignment

The alignment of the paragraph.

`placement` Placement

The placement of the paragraph.

`negative` bool

If true, the paragraph is printed in the negative colors.

`line` int?

The height of the paragraph.

`text` string[]

The lines of the paragraph.

Remarks

For more information, consider visiting the documentation available here.

WritePositionedString(string, Placement, bool, int?, bool)

Writes a string in the console given a certain placement and line.

```
[Visual]
public static void WritePositionedString(string str, Placement placement =
Placement.TopCenter, bool negative = false, int? line = null, bool writeLine
= false)
```

Parameters

`str` string

The string to write.

`placement` Placement

The placement of the string in the console.

`negative` bool

If true, the text is highlighted.

`line` int?

The line where the string is written in the console. If null, will be written where the cursor is.

`writeLine` bool

If true, the string is written with a line break.

Remarks

For more information, consider visiting the documentation available here.

WritePositionedStyledText(string[]?, int?, int?, int?, TextAlignment, bool)

Writes a styled text in the console. The height will depend on the font used. Font enum for details.

```
[Visual]
public static void WritePositionedStyledText(string[]? text = null, int? line =
null, int? width = null, int? margin = null, TextAlignment align =
TextAlignment.Center, bool negative = false)
```

Parameters

`text` string[]

The styled string to write.

`line` int?

The line where the string is written in the console. If null, will be written from the ContentHeight.

`width` int?

The width of the string. If null, the width is the window width.

`margin` int?

The upper and lower margin.

`align` TextAlignment

The alignment of the string.

`negative` bool

If true, the text is highlighted.

Remarks

For more information, consider visiting the documentation available here.

Class Window

Namespace: ConsoleAppVisuals

Assembly: ConsoleAppVisuals.dll

The Window class is a collection of methods to manage visual elements stored in itself. You may add, remove, or render elements in the console after adding them to the Window class.

```
public static class Window
```

Inheritance

object ← Window

Inherited Members

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

Remarks

For more information, consider visiting the documentation available here.

Fields

INTERVAL_BETWEEN_READS

Defines the interval of milliseconds between different read key of the console (used in the Freeze(ConsoleKey) method).

```
public const int INTERVAL_BETWEEN_READS = 10
```

Field Value

int

Properties

CountElements

Gets the number of elements currently stored in the window.

```
public static int CountElements { get; }
```

Property Value

int

Elements

Gets the list of elements stored in the window.

```
public static List<Element> Elements { get; }
```

Property Value

List<Element>

NextId

Gets the next id number each time a new element is added to the window.

```
public static int NextId { get; }
```

Property Value

int

Methods

ActivateAllElements()

Attempts to toggle the visibility of all Element in the window if the element fits the max number constraint.

```
public static void ActivateAllElements()
```

Remarks

For more information, consider visiting the documentation available here.

ActivateElement(Element, bool)

Attempts to toggle the visibility of the given Element if it fits the max number constraint. The element will be rendered if the render parameter is true.

```
[Visual]
public static void ActivateElement(Element element, bool render = true)
```

Parameters

`element` Element

The Element to be activated.

`render` bool

If true, the Element will be rendered.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ElementNotFoundException

Thrown when the element is invalid.

ActivateElement(int, bool)

Attempts to toggle the visibility of the Element with the given id if the element fits the max number constraint. The element will be rendered if the render parameter is true.

```
[Visual]
public static void ActivateElement(int id, bool render = true)
```

Parameters

`id` int

The id of the Element to activate.

`render` bool

If true, the Element will be rendered.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Thrown when the id is out of the range of the Window elements.

ActivateElement<T>(bool)

Attempts to toggle the visibility of the first Element with the given type if the element fits the max number constraint.

```
[Visual]
public static void ActivateElement<T>(bool render = true) where T : Element
```

Parameters

`render` bool

If true, the element will be rendered.

Type Parameters

`T`

The type of the element.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ElementNotFoundException

Thrown when the element is invalid.

AddElement(params Element[])

Adds one or more Element to the window. If the element is passive, it will try to toggle its visibility to true.

```
public static void AddElement(params Element[] elements)
```

Parameters

`elements` Element[]

The Element to be added.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentException

Thrown when no Element are provided.

DuplicateElementException

Thrown when the Element is already present in the Window.

CheckLine(int?)

Checks if the line is not out of the console range.

```
public static int? CheckLine(int? line)
```

Parameters

`line` int?

The line to be checked.

Returns

int?

The line if it is valid.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

LineOutOfConsoleException

Thrown when the line is out of the console range.

Clear(bool, int?, int?, int)

Clears the console given a range of lines.

```
[Visual]
public static bool Clear(bool continuous = false, int? startLine = null, int? length
= null, int step = 1)
```

Parameters

`continuous` bool

If true, the window will be cleared continuously.

`startLine` int?

The start line of the window to be cleared.

`length` int?

The number of lines to be cleared.

`step` int

The step of the window to be cleared.

Returns

bool

True if the window is successfully cleared, false otherwise.

Remarks

For more information, consider visiting the documentation available here.

Close()

Clears the console, show the cursor, and exit the program.

```
[Visual]
public static void Close()
```

Remarks

For more information, consider visiting the documentation available here.

DeactivateAllElements(bool)

This method deactivate the visibility of all Element.

```
public static void DeactivateAllElements(bool clear = true)
```

Parameters

`clear` bool

Remarks

For more information, consider visiting the documentation available here.

DeactivateElement(Element, bool)

Deactivates the visibility of the given Element. The console space taken by the element will be cleared if the clear parameter is true.

```
public static void DeactivateElement(Element element, bool clear = true)
```

Parameters

`element` Element

The Element to be deactivated.

`clear` bool

If true, the Element space will be cleared from the console.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ElementNotFoundException

Thrown when the element is invalid.

DeactivateElement(int, bool)

Deactivates the visibility of the Element with the given id. The console space taken by the element will be cleared if the clear parameter is true.

```
public static void DeactivateElement(int id, bool clear = true)
```

Parameters

`id` int

The id of the Element.

`clear` bool

If true, the Element space will be cleared from the console.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Thrown when the id is out of the range of the Window elements.

DeactivateElement<T>(bool)

Deactivates the visibility of the first Element with the given type. The console space taken by the element will be cleared if the clear parameter is true.

```
public static void DeactivateElement<T>(bool clear = true) where T : Element
```

Parameters

`clear` bool

If true, the element will be cleared.

Type Parameters

`T`

The type of the element.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ElementNotFoundException

Thrown when the element is invalid.

Freeze(ConsoleKey)

Freezes the execution of the program until the given key is pressed.

```
[Visual]
public static void Freeze(ConsoleKey key = ConsoleKey.Enter)
```

Parameters

`key` ConsoleKey

The key to be pressed to continue the execution.

Remarks

For more information, consider visiting the documentation available here.

GetElement<T>()

Gets the first Element of the given type present in the Window.

```
public static T? GetElement<T>() where T : Element
```

Returns

T

The element with the given type if it exists, null otherwise.

Type Parameters

`T`

The type of the element.

Remarks

For more information, consider visiting the documentation available here.

GetElement<T>(int)

Gets the Element with the given id in the Window.

```
public static T? GetElement<T>(int id) where T : Element
```

Parameters

`id` int

The id of the element.

Returns

T

The Element with the given id if it exists, null otherwise.

Type Parameters

`T`

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Thrown when the id is out of the range of the Window elements.

GetLineAvailable(Placement)

Gets the last line available to render an Element on the console given a placement.

```
public static int GetLineAvailable(Placement placement)
```

Parameters

`placement` Placement

The placement to get the line.

Returns

int

The last line available to render an Element on the console.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Thrown when the placement is invalid.

GetVisibleElement<T>()

Gets the visible Element with the given type in the Window.

```
public static T? GetVisibleElement<T>() where T : Element
```

Returns

T

The visible Element with the given type if it exists, null otherwise.

Type Parameters

`T`

The type of the element.

Remarks

For more information, consider visiting the documentation available here.

InsertElement(Element, int)

Inserts an Element at the given id in the Window.

```
public static void InsertElement(Element element, int id)
```

Parameters

`element` Element

The Element to be inserted.

`id` int

The target id to insert the element.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Thrown when the id is out of the range of the Window elements.

IsElementActivatable(int)

Tests if an Element given by an id can be activated

```
public static bool IsElementActivatable(int id)
```

Parameters

`id` int

The id of the Element to test.

Returns

bool

True if the Element can be activated, false otherwise.

Remarks

For more information, consider visiting the documentation available here.

Open()

Clears the console and hide the cursor.

```
[Visual]
public static void Open()
```

Remarks

For more information, consider visiting the documentation available here.

Range(int, int)

Collects a range of Element given a start id and a count.

```
public static List<Element> Range(int index, int count)
```

Parameters

`index` int

The start id of the range.

`count` int

The number of elements to be collected.

Returns

List<Element>

The list of Element given a range of ids.

Remarks

For more information, consider visiting the documentation available here.

RemoveAllElements()

Removes all Element from the window.

```
public static void RemoveAllElements()
```

Remarks

For more information, consider visiting the documentation available here.

RemoveElement(params Element[])

Removes one or more Element from the Window.

```
public static bool RemoveElement(params Element[] elements)
```

Parameters

`elements` Element[]

The Element to be removed.

Returns

bool

True if the element is successfully removed, false otherwise.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ElementNotFoundException

Thrown when the Element is not found in the Window.

RemoveElement(int)

Removes the Element with the given id from the Window.

```
public static void RemoveElement(int id)
```

Parameters

`id` int

The id of the element to be removed.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Thrown when the id is out of the range of the Window elements.

RemoveElement<T>()

Removes the first Element of the given type from the Window.

```
public static bool RemoveElement<T>() where T : Element
```

Returns

bool

True if the element is successfully removed, false otherwise.

Type Parameters

`T`

The type of the element.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ElementNotFoundException

Thrown when the Element is not found in the Window.

Render()

Displays all the visible Element in the window.

```
[Visual]
public static void Render()
```

Remarks

For more information, consider visiting the documentation available here.

Render(params Element[])

Displays the given Element in the window.

```
[Visual]
public static void Render(params Element[] elements)
```

Parameters

`elements` Element[]

The Element to be displayed.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ElementNotFoundException

Thrown when the element is not found in the Window.

RenderElementsSpace(bool)

Renders the space of all visible Element in the window. May ignore the visibility to display the hidden Element spaces.

```
[Visual]
public static bool RenderElementsSpace(bool ignoreVisibility = false)
```

Parameters

`ignoreVisibility` bool

If true, the space of the hidden Element will be drawn.

Returns

bool

True if the space of the Element is successfully drawn, false otherwise.

Remarks

For more information, consider visiting the documentation available here.

Namespace ConsoleAppVisuals.Animated Elements

Classes

FakeLoadingBar

The FakeLoadingBar is an animated element that simulates a loading bar with a fixed duration.

LoadingBar

The LoadingBar is a passive element that displays a loading bar using a reference to a progress variable.

Class FakeLoadingBar

Namespace: ConsoleAppVisuals.AnimatedElements

Assembly: ConsoleAppVisuals.dll

The FakeLoadingBar is an animated element that simulates a loading bar with a fixed duration.

```
public class FakeLoadingBar : AnimatedElement
```

Inheritance

object ← Element ← AnimatedElement ← FakeLoadingBar

Inherited Members

AnimatedElement.RenderOptionsAfterHand() , AnimatedElement.Type , AnimatedElement.MaxNumberOfThisElement , Element.ToggleVisibility() , Element.RenderElement() , Element.RenderOptionsBeforeHand() , Element.RenderElementSpace(bool) , Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility , Element.TextAlignment , Element.Line , object.GetType() , object.MemberwiseClone() , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

FakeLoadingBar(string, int, Placement, int, int, BordersType)

The FakeLoadingBar is an animated element that simulates a loading bar with a fixed duration.

```
public FakeLoadingBar(string text = "Loading ...", int barWidth = 0, Placement
placement = Placement.TopCenter, int processDuration = 2000, int additionalDuration
= 1000, BordersType bordersType = BordersType.SingleStraight)
```

Parameters

`text` string

The text of the loading bar.

`barWidth` int

The width of the loading bar. If 0 or less than text length, text length is used.

`placement` Placement

The placement of the loading bar.

`processDuration` int

The duration of the loading bar.

`additionalDuration` int

The additional duration of the loading bar at the end.

`bordersType` BordersType

The type of borders to display around the loading bar.

Remarks

For more information, consider visiting the documentation available here.

Properties

AdditionalDuration

Gets the additional duration of the loading bar at the end.

```
public int AdditionalDuration { get; }
```

Property Value

int

BarWidth

Gets the explicit width of the loading bar.

```
public int BarWidth { get; }
```

Property Value

int

Borders

Gets the borders of the loading bar.

```
public Borders Borders { get; }
```

Property Value

Borders

BordersType

Gets the border type of the loading bar.

```
public BordersType BordersType { get; }
```

Property Value

BordersType

Height

Gets the height of the loading bar.

```
public override int Height { get; }
```

Property Value

int

Placement

Gets the placement of the loading bar.

```
public override Placement Placement { get; }
```

Property Value

Placement

ProcessDuration

Gets the duration of the loading bar.

```
public int ProcessDuration { get; }
```

Property Value

int

Text

Gets the text of the loading bar.

```
public string Text { get; }
```

Property Value

string

Width

Gets the width of the loading bar.

```
public override int Width { get; }
```

Property Value

int

Methods

RenderElementActions()

Defines the actions to perform when the element is called to be rendered on the console.

```
[Visual]
protected override void RenderElementActions()
```

UpdateAdditionalDuration(int)

Updates the additional duration of the loading bar.

```
public void UpdateAdditionalDuration(int additionalDuration)
```

Parameters

`additionalDuration` int

The new additional duration of the loading bar.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

The additional duration of the loading bar cannot be negative.

UpdateBarWidth(int)

Updates the width of the loading bar.

```
public void UpdateBarWidth(int barWidth)
```

Parameters

`barWidth` int

The new width of the loading bar. If 0, text length is used.

Remarks

For more information, consider visiting the documentation available here.

UpdateBordersType(BordersType)

Updates the borders type of the loading bar.

```
public void UpdateBordersType(BordersType bordersType)
```

Parameters

`bordersType` BordersType

The new border type of the loading bar.

Remarks

For more information, consider visiting the documentation available here.

UpdatePlacement(Placement)

Updates the placement of the loading bar.

```
public void UpdatePlacement(Placement placement)
```

Parameters

`placement` Placement

The new placement of the loading bar.

Remarks

For more information, consider visiting the documentation available here.

UpdateProcessDuration(int)

Updates the duration of the loading bar.

```
public void UpdateProcessDuration(int processDuration)
```

Parameters

`processDuration` int

The new duration of the loading bar.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Throw when the process duration is negative.

UpdateText(string)

Updates the text of the loading bar.

```
public void UpdateText(string text)
```

Parameters

`text` string

The new text of the loading bar.

Remarks

For more information, consider visiting the documentation available here.

Class LoadingBar

Namespace: ConsoleAppVisuals.AnimatedElements

Assembly: ConsoleAppVisuals.dll

The LoadingBar is a passive element that displays a loading bar using a reference to a progress variable.

```
public class LoadingBar : AnimatedElement
```

Inheritance

object ← Element ← AnimatedElement ← LoadingBar

Inherited Members

AnimatedElement.RenderOptionsAfterHand() , AnimatedElement.Type , AnimatedElement.MaxNumberOfThisElement , Element.ToggleVisibility() , Element.RenderElement() , Element.RenderOptionsBeforeHand() , Element.RenderElementSpace(bool) , Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility , Element.TextAlignment , Element.Line , object.GetType() , object.MemberwiseClone() , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

LoadingBar(string, ref float, Placement, int)

The LoadingBar is a passive element that displays a loading bar using a reference to a progress variable.

```
public LoadingBar(string text, ref float progress, Placement placement =
Placement.TopCenter, int additionalDuration = 1000)
```

Parameters

`text` string

The text of the loading bar.

`progress` float

The reference of the progress of the loading bar (that means that you should put a reference to a variable that will contain the percentage of progress of your process).

`placement` Placement

The placement of the loading bar.

`additionalDuration` int

The duration of the loading bar after the process.

Remarks

For more information, consider visiting the documentation available here.

Properties

AdditionalDuration

Gets the additional duration of the loading bar at the end.

```
public int AdditionalDuration { get; }
```

Property Value

int

Height

Gets the height of the loading bar.

```
public override int Height { get; }
```

Property Value

int

Remarks

One line for the text,one line for the space between and one line for the progress.

Placement

Gets the placement of the loading bar.

```
public override Placement Placement { get; }
```

Property Value

Placement

Progress

Gets the progress of the loading bar.

```
public float Progress { get; }
```

Property Value

float

Text

Gets the text of the loading bar.

```
public string Text { get; }
```

Property Value

string

Width

Gets the width of the loading bar.

```
public override int Width { get; }
```

Property Value

int

Methods

RenderElementActions()

Defines the actions to perform when the element is called to be rendered on the console.

```
[Visual]
protected override void RenderElementActions()
```

UpdateAdditionalDuration(int)

Updates the additional duration of the loading bar.

```
public void UpdateAdditionalDuration(int additionalDuration)
```

Parameters

`additionalDuration` int

The new additional duration of the loading bar.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentException

The additional duration of the loading bar cannot be negative.

UpdatePlacement(Placement)

Updates the placement of the loading bar.

```
public void UpdatePlacement(Placement placement)
```

Parameters

`placement` Placement

The new placement of the loading bar.

Remarks

For more information, consider visiting the documentation available here.

UpdateProgress(float)

Updates the progress of the loading bar.

```
public void UpdateProgress(float progress)
```

Parameters

`progress` float

The new progress of the loading bar.

Remarks

For more information, consider visiting the documentation available here.

UpdateText(string)

Updates the text of the loading bar.

```
public void UpdateText(string text)
```

Parameters

`text` string

The new text of the loading bar.

Remarks

For more information, consider visiting the documentation available here.

Namespace ConsoleAppVisuals.Attributes

Classes

VisualAttribute

The VisualAttribute class is used to mark a class, struct, enum, constructor, method, property, field, event, interface, or delegate as a visual and so interact with the console.

Class VisualAttribute

Namespace: ConsoleAppVisuals.Attributes

Assembly: ConsoleAppVisuals.dll

The VisualAttribute class is used to mark a class, struct, enum, constructor, method, property, field, event, interface, or delegate as a visual and so interact with the console.

```
[AttributeUsage(AttributeTargets.All, Inherited = false)]
public sealed class VisualAttribute : Attribute
```

Inheritance

object ← Attribute ← VisualAttribute

Inherited Members

Attribute.GetCustomAttributes(MemberInfo, Type) ,
Attribute.GetCustomAttributes(MemberInfo, Type, bool) ,
Attribute.GetCustomAttributes(MemberInfo) ,
Attribute.GetCustomAttributes(MemberInfo, bool) ,
Attribute.IsDefined(MemberInfo, Type) , Attribute.IsDefined(MemberInfo, Type, bool) ,
Attribute.GetCustomAttribute(MemberInfo, Type) ,
Attribute.GetCustomAttribute(MemberInfo, Type, bool) ,
Attribute.GetCustomAttributes(ParameterInfo) ,
Attribute.GetCustomAttributes(ParameterInfo, Type) ,
Attribute.GetCustomAttributes(ParameterInfo, Type, bool) ,
Attribute.GetCustomAttributes(ParameterInfo, bool) ,
Attribute.IsDefined(ParameterInfo, Type) ,
Attribute.IsDefined(ParameterInfo, Type, bool) ,
Attribute.GetCustomAttribute(ParameterInfo, Type) ,
Attribute.GetCustomAttribute(ParameterInfo, Type, bool) ,
Attribute.GetCustomAttributes(Module, Type) , Attribute.GetCustomAttributes(Module) ,
Attribute.GetCustomAttributes(Module, bool) ,
Attribute.GetCustomAttributes(Module, Type, bool) , Attribute.IsDefined(Module, Type) ,
Attribute.IsDefined(Module, Type, bool) , Attribute.GetCustomAttribute(Module, Type) ,
Attribute.GetCustomAttribute(Module, Type, bool) ,
Attribute.GetCustomAttributes(Assembly, Type) ,
Attribute.GetCustomAttributes(Assembly, Type, bool) ,
Attribute.GetCustomAttributes(Assembly) ,
Attribute.GetCustomAttributes(Assembly, bool) , Attribute.IsDefined(Assembly, Type) ,

Attribute.IsDefined(Assembly, Type, bool) ,
Attribute.GetCustomAttribute(Assembly, Type) ,
Attribute.GetCustomAttribute(Assembly, Type, bool) , Attribute.Equals(object) ,
Attribute.GetHashCode() , Attribute.Match(object) , Attribute.IsDefaultAttribute() ,
Attribute.TypeId , object.GetType() , object.ToString() , object.Equals(object, object) ,
object.ReferenceEquals(object, object)

Remarks

[WARNING] This element cannot be tested.

Constructors

VisualAttribute()

The VisualAttribute class is used to mark a class, struct, enum, constructor, method, property, field, event, interface, or delegate as a visual and so interact with the console.

```
public VisualAttribute()
```

Remarks

For more information, refer to the following resources:

- Documentation
- Example Project

VisualAttribute(string?)

The VisualAttribute class is used to mark a class, struct, enum, constructor, method, property, field, event, interface, or delegate as a visual and so interact with the console.

```
public VisualAttribute(string? message)
```

Parameters

`message` string

The text string that describes alternative workarounds.

Remarks

For more information, refer to the following resources:

- Documentation
- Example Project

VisualAttribute(string?, bool)

The VisualAttribute class is used to mark a class, struct, enum, constructor, method, property, field, event, interface, or delegate as a visual and so interact with the console.

```
public VisualAttribute(string? message, bool error)
```

Parameters

`message` string

The text string that describes alternative workarounds.

`error` bool

True if the visual element usage generates a compiler error; false if it generates a compiler warning.

Remarks

For more information, refer to the following resources:

- Documentation
- Example Project

Properties

DiagnosticId

Gets or sets the ID that the compiler will use when reporting a use of the API.

```
public string? DiagnosticId { get; set; }
```

Property Value

string

IsError

Gets a value that indicates whether the compiler will treat usage of the visual program element as an error.

```
public bool IsError { get; }
```

Property Value

bool

True if the visual element usage is considered an error; otherwise, false. The default is false.

Message

Gets the workaround message.

```
public string? Message { get; }
```

Property Value

string

The workaround text string.

UrlFormat

Gets or sets the URL for corresponding documentation. The API accepts a format string instead of an actual URL, creating a generic URL that includes the diagnostic ID.

```
public string? UrlFormat { get; set; }
```

Property Value

string

The format string that represents a URL to corresponding documentation.

Namespace ConsoleAppVisuals.Enums

Enums

BordersType
: The BordersType enum defines the type of border to use for embed elements.

DialogOption
: The DialogOption enum defines the outputs of a dialog.

Direction
: The Direction enum defines the direction in some space.

ElementType
: The ElementType enum defines the type of an element.

Font
: The Font enum defines the font used to display styled text. (Used in the Title element)

Placement
: The Placement enum defines the placement of a string in some space. It could be another string or a console line.

PromptInputStyle
: The PromptInputStyle enum defines the style of the prompt input.

Status
: The Status enum represents the exit status of an interaction.

TextAlignment
: The TextAlignment enum defines the alignment of a string in the space.

Enum BordersType

Namespace: ConsoleAppVisuals.Enums

Assembly: ConsoleAppVisuals.dll

The BordersType enum defines the type of border to use for embed elements.

```
public enum BordersType
```

Fields

`ASCII = 4`

ASCII borders (+-|++++)

Universal compatibility.

`DoubleStraight = 3`

Double line borders with straight corners (╔╗╚╝═║╦╩╠╣╬)

Universal compatibility.

`SingleBold = 2`

Single line borders with bold lines (┏┓┗┛━┃┳┻┣┫╋)

May not be supported on PowerShell (Windows).

`SingleRound = 1`

Single line borders with rounded corners (╭╮╰╯─│┬┴├┤┼)

May not be supported on PowerShell (Windows).

`SingleStraight = 0`

Single line borders with straight corners (┌┐└┘─│┬┴├┤┼)

Universal compatibility.

Remarks

For more information, consider visiting the documentation available here.

Enum DialogOption

Namespace: ConsoleAppVisuals.Enums

Assembly: ConsoleAppVisuals.dll

The DialogOption enum defines the outputs of a dialog.

```
public enum DialogOption
```

Fields

`Left = 1`

Left option selected.

`None = 0`

No options are set or escape pressed.

`Right = 2`

Right option selected.

Remarks

For more information, consider visiting the documentation available here.

Enum Direction

Namespace: ConsoleAppVisuals.Enums

Assembly: ConsoleAppVisuals.dll

The Direction enum defines the direction in some space.

```
public enum Direction
```

Fields

`Down = 1`

The down direction.

`Up = 0`

The up direction.

Remarks

For more information, consider visiting the documentation available here.

Enum ElementType

Namespace: ConsoleAppVisuals.Enums

Assembly: ConsoleAppVisuals.dll

The ElementType enum defines the type of an element.

```
public enum ElementType
```

Fields

`Animated = 3`

The animated element type.

`Default = 0`

The default element type, not regarding wether it is passive or interactive.

`Interactive = 2`

The interactive element type.

`Passive = 1`

The passive element type.

Remarks

For more information, consider visiting the documentation available here.

Enum Font

Namespace: ConsoleAppVisuals.Enums

Assembly: ConsoleAppVisuals.dll

The Font enum defines the font used to display styled text. (Used in the Title element)

```
public enum Font
```

Fields

`ANSI_Shadow = 1`

Author: Unknown, Height: 6

`Big = 6`

Author: Glenn Chappell, Height: 8

`Bloody = 5`

Author: Unknown, Height: 8

`Bulbhead = 2`

Author: Jef Poskanzer, Height: 4

`Custom = 0`

Font defined by the user.

`Ghost = 3`

Author: myflix, Height: 9

`Lil_Devil = 7`

Author: myflix, Height: 8

`Merlin = 4`

Author: LG Beard, Height: 8

`Stop = 8`

Author: David Walton, Height: 7

Remarks

For more information, consider visiting the documentation available here.

Enum Placement

Namespace: ConsoleAppVisuals.Enums

Assembly: ConsoleAppVisuals.dll

The Placement enum defines the placement of a string in some space. It could be another string or a console line.

```
public enum Placement
```

Extension Methods
Core.ToTextAlignment(Placement)

Fields

`BottomCenterFullWidth = 4`

The object is placed in the bottom center of the space and takes all the width.

`TopCenter = 0`

The object is placed in the top center of the space.

`TopCenterFullWidth = 3`

The object is placed in the top center of the space and takes all the width.

`TopLeft = 1`

The object is placed in the top left of the space.

`TopRight = 2`

The object is placed in the top right of the space.

Remarks

For more information, consider visiting the documentation available here.

Enum PromptInputStyle

Namespace: ConsoleAppVisuals.Enums

Assembly: ConsoleAppVisuals.dll

The PromptInputStyle enum defines the style of the prompt input.

```
public enum PromptInputStyle
```

Fields

`Default = 0`

The default prompt style.

`Fill = 1`

The form style. (the default text will be "----" for 4 characters)

`Secret = 2`

The secret style. (the input will be hidden with "*" characters)

Remarks

For more information, consider visiting the documentation available here.

Enum Status

Namespace: ConsoleAppVisuals.Enums

Assembly: ConsoleAppVisuals.dll

The Status enum represents the exit status of an interaction.

```
public enum Status
```

Fields

`Deleted = 1`

Pressed the delete key.

`Escaped = 2`

Pressed the escape key.

`Selected = 0`

Pressed the enter key.

Remarks

For more information, consider visiting the documentation available here.

Enum TextAlignment

Namespace: ConsoleAppVisuals.Enums

Assembly: ConsoleAppVisuals.dll

The TextAlignment enum defines the alignment of a string in the space.

```
public enum TextAlignment
```

Extension Methods
Core.ToPlacement(TextAlignment)

Fields

`Center = 0`

The string is placed in the center of the space.

`Left = 1`

The string is placed on the left of the space.

`Right = 2`

The string is placed on the right of the space.

Remarks

For more information, consider visiting the documentation available here.

Namespace ConsoleAppVisuals.Errors

Classes

DuplicateElementException
: Exception thrown when an element is found twice in a collection.

ElementNotFoundException
: Exception thrown when an element is not found in a collection.

EmptyFileException
: Exception thrown when no data is found in a file.

LineOutOfConsoleException
: Exception thrown when a line exceeds the console's height.

NotSupportedCharException
: Exception thrown when a character is not supported by the TextStyler class.

Class DuplicateElementException

Namespace: ConsoleAppVisuals.Errors

Assembly: ConsoleAppVisuals.dll

Exception thrown when an element is found twice in a collection.

```
public class DuplicateElementException : Exception, ISerializable
```

Inheritance

object ← Exception ← DuplicateElementException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.ToString() , Exception.GetType() , Exception.TargetSite , Exception.Message , Exception.Data , Exception.InnerException , Exception.HelpLink , Exception.Source , Exception.HResult , Exception.StackTrace , Exception.SerializeObjectState , object.MemberwiseClone() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

DuplicateElementException()

Exception thrown when an element is found twice in a collection.

```
public DuplicateElementException()
```

DuplicateElementException(string)

Exception thrown when an element is found twice in a collection.

```
public DuplicateElementException(string message)
```

Parameters

`message` string

Message to be displayed.

DuplicateElementException(string, Exception)

Exception thrown when an element is found twice in a collection.

```
public DuplicateElementException(string message, Exception inner)
```

Parameters

`message` string

Message to be displayed.

`inner` Exception

Inner exception.

Class ElementNotFoundException

Namespace: ConsoleAppVisuals.Errors

Assembly: ConsoleAppVisuals.dll

Exception thrown when an element is not found in a collection.

```
public class ElementNotFoundException : Exception, ISerializable
```

Inheritance

object ← Exception ← ElementNotFoundException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.ToString() , Exception.GetType() , Exception.TargetSite , Exception.Message , Exception.Data , Exception.InnerException , Exception.HelpLink , Exception.Source , Exception.HResult , Exception.StackTrace , Exception.SerializeObjectState , object.MemberwiseClone() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

ElementNotFoundException()

Exception thrown when an element is not found in a collection.

```
public ElementNotFoundException()
```

ElementNotFoundException(string)

Exception thrown when an element is not found in a collection.

```
public ElementNotFoundException(string message)
```

Parameters

`message` string

Message to be displayed.

ElementNotFoundException(string, Exception)

Exception thrown when an element is not found in a collection.

```
public ElementNotFoundException(string message, Exception inner)
```

Parameters

`message` string

Message to be displayed.

`inner` Exception

Inner exception.

Class EmptyFileException

Namespace: ConsoleAppVisuals.Errors

Assembly: ConsoleAppVisuals.dll

Exception thrown when no data is found in a file.

```
public class EmptyFileException : Exception, ISerializable
```

Inheritance

object ← Exception ← EmptyFileException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.ToString() , Exception.GetType() , Exception.TargetSite , Exception.Message , Exception.Data , Exception.InnerException , Exception.HelpLink , Exception.Source , Exception.HResult , Exception.StackTrace , Exception.SerializeObjectState , object.MemberwiseClone() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

EmptyFileException()

Exception thrown when no data is found in a file.

```
public EmptyFileException()
```

EmptyFileException(string)

Exception thrown when no data is found in a file.

```
public EmptyFileException(string message)
```

Parameters

`message` string

Message to be displayed.

EmptyFileException(string, Exception)

Exception thrown when no data is found in a file.

```
public EmptyFileException(string message, Exception inner)
```

Parameters

`message` string

Message to be displayed.

`inner` Exception

Inner exception.

Class LineOutOfConsoleException

Namespace: ConsoleAppVisuals.Errors

Assembly: ConsoleAppVisuals.dll

Exception thrown when a line exceeds the console's height.

```
public class LineOutOfConsoleException : Exception, ISerializable
```

Inheritance

object ← Exception ← LineOutOfConsoleException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.ToString() , Exception.GetType() , Exception.TargetSite , Exception.Message , Exception.Data , Exception.InnerException , Exception.HelpLink , Exception.Source , Exception.HResult , Exception.StackTrace , Exception.SerializeObjectState , object.MemberwiseClone() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

LineOutOfConsoleException()

Exception thrown when a line exceeds the console's height.

```
public LineOutOfConsoleException()
```

LineOutOfConsoleException(string)

Exception thrown when a line exceeds the console's height.

```
public LineOutOfConsoleException(string message)
```

Parameters

`message` string

Message to be displayed.

LineOutOfConsoleException(string, Exception)

Exception thrown when a line exceeds the console's height.

```
public LineOutOfConsoleException(string message, Exception inner)
```

Parameters

`message` string

Message to be displayed.

`inner` Exception

Inner exception.

Class NotSupportedCharException

Namespace: ConsoleAppVisuals.Errors

Assembly: ConsoleAppVisuals.dll

Exception thrown when a character is not supported by the TextStyler class.

```
public class NotSupportedCharException : Exception, ISerializable
```

Inheritance

object ← Exception ← NotSupportedCharException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.ToString() , Exception.GetType() , Exception.TargetSite , Exception.Message , Exception.Data , Exception.InnerException , Exception.HelpLink , Exception.Source , Exception.HResult , Exception.StackTrace , Exception.SerializeObjectState , object.MemberwiseClone() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

NotSupportedCharException()

Exception thrown when a character is not supported by the TextStyler class.

```
public NotSupportedCharException()
```

NotSupportedCharException(string)

Exception thrown when a character is not supported by the TextStyler class.

```
public NotSupportedCharException(string message)
```

Parameters

`message` string

Message to be displayed.

NotSupportedCharException(string, Exception)

Exception thrown when a character is not supported by the TextStyler class.

```
public NotSupportedCharException(string message, Exception inner)
```

Parameters

`message` string

Message to be displayed.

`inner` Exception

Inner exception.

Namespace ConsoleAppVisuals.InteractiveElements

Classes

Dialog

The Dialog is an interactive element that displays a dialog bow with one or two choices. See DialogOption enum for the possible outputs of a dialog.

FloatSelector

The FloatSelector is an interactive element that allows the user to select a float value from a range of values.

IntSelector

The IntSelector is an interactive element that allows the user to select an integer value from a range of values.

Prompt

The Prompt is an interactive element that allows the user to input a string response.

ScrollingMenu

The ScrollingMenu is an interactive element that displays a question with multiple scrollable choices.

TableSelector

The TableSelector is an interactive element that displays a table with selectable elements.

Class Dialog

Namespace: ConsoleAppVisuals.InteractiveElements

Assembly: ConsoleAppVisuals.dll

The Dialog is an interactive element that displays a dialog bow with one or two choices. See DialogOption enum for the possible outputs of a dialog.

```
public class Dialog : InteractiveElement<DialogOption>
```

Inheritance

object ← Element ← InteractiveElement<DialogOption> ← Dialog

Inherited Members

InteractiveElement<DialogOption>._interactionResponse ,
InteractiveElement<DialogOption>.SetInteractionResponse(object, InteractionEventArgs<DialogOption>) ,
InteractiveElement<DialogOption>.SendResponse(object, InteractionEventArgs<DialogOption>) ,
InteractiveElement<DialogOption>.GetResponse() ,
InteractiveElement<DialogOption>.GetResponseHistory() ,
InteractiveElement<DialogOption>.RenderOptionsBeforeHand() ,
InteractiveElement<DialogOption>.RenderOptionsAfterHand() ,
InteractiveElement<DialogOption>.Type ,
InteractiveElement<DialogOption>.MaxNumberOfThisElement ,
InteractiveElement<DialogOption>.EndOfInteractionEvent , Element.ToggleVisibility() ,
Element.RenderElement() , Element.RenderElementSpace(bool) ,
Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility ,
Element.Line , object.GetType() , object.MemberwiseClone() , object.ToString() ,
object.Equals(object) , object.Equals(object, object) ,
object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

Dialog(List<string>, string?, string?, TextAlignment, Placement, BordersType)

The Dialog is an interactive element that displays a dialog bow with one or two choices. See DialogOption enum for the possible outputs of a dialog.

```
public Dialog(List<string> lines, string? leftOption = null, string? rightOption = null, TextAlignment align = TextAlignment.Left, Placement placement = Placement.TopCenter, BordersType bordersType = BordersType.SingleStraight)
```

Parameters

`lines` List<string>

The text to display.

`leftOption` string

The text of the left option. Null for no button.

`rightOption` string

The text of the right option. Null for no button.

`align` TextAlignment

The alignment of the Dialog.

`placement` Placement

The placement of the Dialog element.

`bordersType` BordersType

The type of border to display.

Remarks

For more information, consider visiting the documentation available here.

Properties

Borders

Gets the borders of the Dialog.

```
public Borders Borders { get; }
```

Property Value

Borders

BordersType

Gets the border type of the selector.

```
public BordersType BordersType { get; }
```

Property Value

BordersType

Height

Gets the height of the Dialog.

```
public override int Height { get; }
```

Property Value

int

LeftOption

Gets the text of the left option.

```
public string? LeftOption { get; }
```

Property Value

string

Lines

Gets the rows of the Dialog.

```
public List<string> Lines { get; }
```

Property Value

List<string>

Placement

Gets the position of the Dialog.

```
public override Placement Placement { get; }
```

Property Value

Placement

RightOption

Gets the text of the right option.

```
public string? RightOption { get; }
```

Property Value

string

TextAlignment

Gets the alignment of the Dialog.

```
public override TextAlignment TextAlignment { get; }
```

Property Value

TextAlignment

TextToDisplay

Gets the text to display.

```
public List<string>? TextToDisplay { get; }
```

Property Value

List<string>

Width

Gets the width of the Dialog.

```
public override int Width { get; }
```

Property Value

int

Methods

AddLine(string)

Adds a line to the Dialog.

```
public void AddLine(string line)
```

Parameters

`line` string

The line to add.

Remarks

For more information, consider visiting the documentation available here.

InsertLine(int, string)

Inserts a line to the Dialog.

```
public void InsertLine(int index, string line)
```

Parameters

`index` int

The index where to insert the line.

`line` string

The line to insert.

Remarks

For more information, consider visiting the documentation available here.

RemoveLine(int)

Removes a line from the Dialog.

```
public void RemoveLine(int index)
```

Parameters

`index` int

The index of the line to remove.

Remarks

For more information, consider visiting the documentation available here.

RemoveLine(string)

Removes a line from the Dialog.

```
public void RemoveLine(string line)
```

Parameters

`line` string

The line to remove.

Remarks

For more information, consider visiting the documentation available here.

RenderElementActions()

Renders the Dialog.

```
[Visual]
protected override void RenderElementActions()
```

UpdateBordersType(BordersType)

Updates the borders of the Dialog.

```
public void UpdateBordersType(BordersType bordersType)
```

Parameters

`bordersType` BordersType

The new border type of the Dialog.

Remarks

For more information, consider visiting the documentation available here.

UpdateLeftOption(string?)

Updates the text of the second option.

```
public void UpdateLeftOption(string? text)
```

Parameters

`text` string

The new text of the second option.

Remarks

For more information, consider visiting the documentation available here.

UpdateLines(List<string>)

Updates the text of the Dialog.

```
public void UpdateLines(List<string> lines)
```

Parameters

`lines` List<string>

The new text of the Dialog.

Remarks

For more information, consider visiting the documentation available here.

UpdatePlacement(Placement)

Updates the placement of the Dialog.

```
public void UpdatePlacement(Placement placement)
```

Parameters

`placement` Placement

The new placement of the Dialog.

Remarks

For more information, consider visiting the documentation available here.

UpdateRightOption(string?)

Updates the text of the first option.

```
public void UpdateRightOption(string? text)
```

Parameters

`text` string

The new text of the first option.

Remarks

For more information, consider visiting the documentation available here.

UpdateTextAlignment(TextAlignment)

Updates the alignment of the Dialog.

```
public void UpdateTextAlignment(TextAlignment align)
```

Parameters

`align` TextAlignment

The new alignment of the Dialog.

Remarks

For more information, consider visiting the documentation available here.

Class FloatSelector

Namespace: ConsoleAppVisuals.InteractiveElements

Assembly: ConsoleAppVisuals.dll

The FloatSelector is an interactive element that allows the user to select a float value from a range of values.

```
public class FloatSelector : InteractiveElement<float>
```

Inheritance

object ← Element ← InteractiveElement<float> ← FloatSelector

Inherited Members

InteractiveElement<float>._interactionResponse ,
InteractiveElement<float>.SetInteractionResponse(object, InteractionEventArgs<float>) ,
InteractiveElement<float>.SendResponse(object, InteractionEventArgs<float>) ,
InteractiveElement<float>.GetResponse() ,
InteractiveElement<float>.GetResponseHistory() ,
InteractiveElement<float>.RenderOptionsBeforeHand() ,
InteractiveElement<float>.RenderOptionsAfterHand() , InteractiveElement<float>.Type ,
InteractiveElement<float>.MaxNumberOfThisElement ,
InteractiveElement<float>.EndOfInteractionEvent , Element.ToggleVisibility() ,
Element.RenderElement() , Element.RenderElementSpace(bool) ,
Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility ,
Element.TextAlignment , Element.Line , object.GetType() , object.MemberwiseClone() ,
object.ToString() , object.Equals(object) , object.Equals(object, object) ,
object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

FloatSelector(string, float, float, float, float, Placement, BordersType)

The FloatSelector is an interactive element that allows the user to select a float value from a range of values.

```
public FloatSelector(string question, float min, float max, float start,
float step, Placement placement = Placement.TopCenter, BordersType bordersType
= BordersType.SingleStraight)
```

Parameters

`question` string

The question to ask the user.

`min` float

The minimum value of the selector.

`max` float

The maximum value of the selector.

`start` float

The start value of the selector.

`step` float

The step of the selector.

`placement` Placement

The placement of the selector on the console.

`bordersType` BordersType

The type of the borders of the selector.

Remarks

For more information, consider visiting the documentation available here.

Properties

Borders

Gets the borders manager of the selector.

```
public Borders Borders { get; }
```

Property Value

Borders

BordersType

Gets the type of the borders of the selector.

```
public BordersType BordersType { get; }
```

Property Value

BordersType

Height

Gets the height of the selector.

```
public override int Height { get; }
```

Property Value

int

LeftSelector

Gets the left selector of the selector.

```
public char LeftSelector { get; }
```

Property Value

char

Max

Gets the maximum value of the selector.

```
public float Max { get; }
```

Property Value

float

Min

Gets the minimum value of the selector.

```
public float Min { get; }
```

Property Value

float

Placement

Gets the placement of the selector on the console.

```
public override Placement Placement { get; }
```

Property Value

Placement

Question

Gets the question to ask the user.

```
public string Question { get; }
```

Property Value

string

RightSelector

Gets the right selector of the selector.

```
public char RightSelector { get; }
```

Property Value

char

Start

Gets the start value of the selector.

```
public float Start { get; }
```

Property Value

float

Step

Gets the step of the selector.

```
public float Step { get; }
```

Property Value

float

Width

Gets the width of the selector.

```
public override int Width { get; }
```

Property Value

int

Methods

RenderElementActions()

Defines the actions to perform when the element is called to be rendered on the console.

```
[Visual]
protected override void RenderElementActions()
```

UpdateBordersType(BordersType)

Updates the type of the borders of the selector.

```
public void UpdateBordersType(BordersType bordersType)
```

Parameters

`bordersType` BordersType

The new type of the borders of the selector.

Remarks

For more information, consider visiting the documentation available here.

UpdateLeftSelector(char)

Updates the selector of the selector.

```
public void UpdateLeftSelector(char leftSelector = '▶')
```

Parameters

`leftSelector` char

The new left selector of the selector.

Remarks

For more information, consider visiting the documentation available here.

UpdateMax(float)

Updates the maximum value of the selector.

```
public void UpdateMax(float max)
```

Parameters

`max` float

The maximum value of the selector.

Remarks

For more information, consider visiting the documentation available here.

UpdateMin(float)

Updates the minimum value of the selector.

```
public void UpdateMin(float min)
```

Parameters

`min` float

The minimum value of the selector.

Remarks

For more information, consider visiting the documentation available here.

UpdatePlacement(Placement)

Updates the placement of the selector.

```
public void UpdatePlacement(Placement placement)
```

Parameters

`placement` Placement

The placement of the selector on the console.

Remarks

For more information, consider visiting the documentation available here.

UpdateQuestion(string)

Updates the question of the selector.

```
public void UpdateQuestion(string question)
```

Parameters

`question` string

The question to ask the user.

Remarks

For more information, consider visiting the documentation available here.

UpdateRightSelector(char)

Updates the selector of the selector.

```
public void UpdateRightSelector(char rightSelector = '◀')
```

Parameters

`rightSelector` char

The new right selector of the selector.

Remarks

For more information, consider visiting the documentation available here.

UpdateStart(float)

Updates the start value of the selector.

```
public void UpdateStart(float start)
```

Parameters

`start` float

The start value of the selector.

Remarks

For more information, consider visiting the documentation available here.

UpdateStep(float)

Updates the step of the selector.

```
public void UpdateStep(float step)
```

Parameters

`step` float

The step of the selector.

Remarks

For more information, consider visiting the documentation available here.

Class IntSelector

Namespace: ConsoleAppVisuals.InteractiveElements

Assembly: ConsoleAppVisuals.dll

The IntSelector is an interactive element that allows the user to select an integer value from a range of values.

```
public class IntSelector : InteractiveElement<int>
```

Inheritance

object ← Element ← InteractiveElement<int> ← IntSelector

Inherited Members

InteractiveElement<int>._interactionResponse , InteractiveElement<int>.SetInteractionResponse(object, InteractionEventArgs<int>) , InteractiveElement<int>.SendResponse(object, InteractionEventArgs<int>) , InteractiveElement<int>.GetResponse() , InteractiveElement<int>.GetResponseHistory() , InteractiveElement<int>.RenderOptionsBeforeHand() , InteractiveElement<int>.RenderOptionsAfterHand() , InteractiveElement<int>.Type , InteractiveElement<int>.MaxNumberOfThisElement , InteractiveElement<int>.EndOfInteractionEvent , Element.ToggleVisibility() , Element.RenderElement() , Element.RenderElementSpace(bool) , Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility , Element.TextAlignment , Element.Line , object.GetType() , object.MemberwiseClone() , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

IntSelector(string, int, int, int, int, Placement, BordersType)

The IntSelector is an interactive element that allows the user to select an integer value from a range of values.

```
public IntSelector(string question, int min, int max, int start, int
step, Placement placement = Placement.TopCenter, BordersType bordersType
= BordersType.SingleStraight)
```

Parameters

`question` string

The question to ask the user.

`min` int

The minimum value of the selector.

`max` int

The maximum value of the selector.

`start` int

The start value of the selector.

`step` int

The step of the selector.

`placement` Placement

The placement of the selector on the console.

`bordersType` BordersType

The border type of the selector.

Remarks

For more information, consider visiting the documentation available here.

Properties

Borders

Gets the borders manager of the selector.

```
public Borders Borders { get; }
```

Property Value

Borders

BordersType

Gets the border type of the selector.

```
public BordersType BordersType { get; }
```

Property Value

BordersType

Height

Gets the height of the selector.

```
public override int Height { get; }
```

Property Value

int

LeftSelector

Gets the left selector of the selector.

```
public char LeftSelector { get; }
```

Property Value

char

Max

Gets the maximum value of the selector.

```
public int Max { get; }
```

Property Value

int

Min

Gets the minimum value of the selector.

```
public int Min { get; }
```

Property Value

int

Placement

Gets the placement of the selector on the console.

```
public override Placement Placement { get; }
```

Property Value

Placement

Question

Gets the question to ask the user.

```
public string Question { get; }
```

Property Value

string

RightSelector

Gets the right selector of the selector.

```
public char RightSelector { get; }
```

Property Value

char

Start

Gets the start value of the selector.

```
public int Start { get; }
```

Property Value

int

Step

Gets the step of the selector.

```
public int Step { get; }
```

Property Value

int

Width

Gets the width of the selector.

```
public override int Width { get; }
```

Property Value

int

Methods

RenderElementActions()

Defines the actions to perform when the element is called to be rendered on the console.

```
[Visual]
protected override void RenderElementActions()
```

UpdateBordersType(BordersType)

Updates the border type of the selector.

```
public void UpdateBordersType(BordersType bordersType)
```

Parameters

`bordersType` BordersType

The border type of the selector.

Remarks

For more information, consider visiting the documentation available here.

UpdateLeftSelector(char)

Updates the selector of the selector.

```
public void UpdateLeftSelector(char leftSelector = '▶')
```

Parameters

`leftSelector` char

The new left selector of the selector.

Remarks

For more information, consider visiting the documentation available here.

UpdateMax(int)

Updates the maximum value of the selector.

```
public void UpdateMax(int max)
```

Parameters

`max` int

The maximum value of the selector.

Remarks

For more information, consider visiting the documentation available here.

UpdateMin(int)

Updates the minimum value of the selector.

```
public void UpdateMin(int min)
```

Parameters

`min` int

The minimum value of the selector.

Remarks

For more information, consider visiting the documentation available here.

UpdatePlacement(Placement)

Updates the placement of the selector.

```
public void UpdatePlacement(Placement placement)
```

Parameters

`placement` Placement

The placement of the selector on the console.

Remarks

For more information, consider visiting the documentation available here.

UpdateQuestion(string)

Updates the question of the selector.

```
public void UpdateQuestion(string question)
```

Parameters

`question` string

The question to ask the user.

Remarks

For more information, consider visiting the documentation available here.

UpdateRightSelector(char)

Updates the selector of the selector.

```
public void UpdateRightSelector(char rightSelector = '◄')
```

Parameters

`rightSelector` char

The new right selector of the selector.

Remarks

For more information, consider visiting the documentation available here.

UpdateStart(int)

Updates the start value of the selector.

```
public void UpdateStart(int start)
```

Parameters

`start` int

The start value of the selector.

Remarks

For more information, consider visiting the documentation available here.

UpdateStep(int)

Updates the step of the selector.

```
public void UpdateStep(int step)
```

Parameters

`step` int

The step of the selector.

Remarks

For more information, consider visiting the documentation available here.

Class Prompt

Namespace: ConsoleAppVisuals.InteractiveElements

Assembly: ConsoleAppVisuals.dll

The Prompt is an interactive element that allows the user to input a string response.

```
public class Prompt : InteractiveElement<string>
```

Inheritance

object ← Element ← InteractiveElement<string> ← Prompt

Inherited Members

InteractiveElement<string>._interactionResponse ,
InteractiveElement<string>.SetInteractionResponse(object, InteractionEventArgs<string>) ,
InteractiveElement<string>.SendResponse(object, InteractionEventArgs<string>) ,
InteractiveElement<string>.GetResponse() ,
InteractiveElement<string>.GetResponseHistory() ,
InteractiveElement<string>.RenderOptionsBeforeHand() ,
InteractiveElement<string>.RenderOptionsAfterHand() , InteractiveElement<string>.Type ,
InteractiveElement<string>.MaxNumberOfThisElement ,
InteractiveElement<string>.EndOfInteractionEvent , Element.ToggleVisibility() ,
Element.RenderElement() , Element.RenderElementSpace(bool) ,
Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility ,
Element.TextAlignment , Element.Line , object.GetType() , object.MemberwiseClone() ,
object.ToString() , object.Equals(object) , object.Equals(object, object) ,
object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

Prompt(string, string?, Placement, int, PromptInputStyle, BordersType)

The Prompt is an interactive element that allows the user to input a string response.

```
public Prompt(string question, string? defaultValue = null, Placement placement =
Placement.TopCenter, int maxInputLength = 12, PromptInputStyle style =
PromptInputStyle.Default, BordersType borderType = BordersType.SingleStraight)
```

Parameters

`question` string

The text on the left of the prompt element.

`defaultValue` string

The text in the center of the prompt element.

`placement` Placement

The placement of the prompt element.

`maxInputLength` int

The maximum length of the response.

`style` PromptInputStyle

The style of the prompt input.

`borderType` BordersType

The type of border to use for the element.

Remarks

For more information, consider visiting the documentation available here.

Properties

Borders

Gets the borders of the prompt element.

```
public Borders Borders { get; }
```

Property Value

Borders

BordersType

Gets the border type of the selector.

```
public BordersType BordersType { get; }
```

Property Value

BordersType

DefaultValue

Gets the default value of the response.

```
public string DefaultValue { get; }
```

Property Value

string

Height

Gets the height of the prompt element.

```
public override int Height { get; }
```

Property Value

int

MaxInputLength

Gets the maximum length of the response.

```
public int MaxInputLength { get; }
```

Property Value

int

Placement

Gets the placement of the prompt element.

```
public override Placement Placement { get; }
```

Property Value

Placement

Question

Gets the question of the prompt element.

```
public string Question { get; }
```

Property Value

string

Selector

Gets the selector of the prompt element.

```
public char Selector { get; }
```

Property Value

char

Style

Gets the style of the prompt input.

```
public PromptInputStyle Style { get; }
```

Property Value

PromptInputStyle

Width

Gets the width of the prompt element.

```
public override int Width { get; }
```

Property Value

int

Methods

RenderElementActions()

Defines the actions to perform when the element is called to be rendered on the console.

```
[Visual]
protected override void RenderElementActions()
```

UpdateBordersType(BordersType)

Updates the border type of the prompt element.

```
public void UpdateBordersType(BordersType bordersType)
```

Parameters

`bordersType` BordersType

The new border type of the prompt element.

Remarks

For more information, consider visiting the documentation available here.

UpdateDefaultValue(string?)

Updates the default value of the prompt element.

```
public void UpdateDefaultValue(string? defaultValue)
```

Parameters

`defaultValue` string

The new default value of the prompt element.

Remarks

For more information, consider visiting the documentation available here.

UpdateMaxLength(int)

Updates the maximum length of the response.

```
public void UpdateMaxLength(int maxLength)
```

Parameters

`maxLength` int

The new maximum length of the response.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

The maximum length of the response must be greater than 0 and less than the width of the console window.

UpdatePlacement(Placement)

Updates the placement of the prompt element.

```
public void UpdatePlacement(Placement placement)
```

Parameters

`placement` Placement

The new placement of the prompt element.

Remarks

For more information, consider visiting the documentation available here.

UpdateQuestion(string)

Updates the question of the prompt element.

```
public void UpdateQuestion(string question)
```

Parameters

`question` string

The new question of the prompt element.

Remarks

For more information, consider visiting the documentation available here.

UpdateSelector(char)

Updates the selector of the prompt element.

```
public void UpdateSelector(char selector = '▶')
```

Parameters

`selector` char

The new selector of the prompt element.

Remarks

For more information, consider visiting the documentation available here.

UpdateStyle(PromptInputStyle)

Updates the style of the prompt input.

```
public void UpdateStyle(PromptInputStyle style)
```

Parameters

`style` PromptInputStyle

The new style of the prompt input.

Remarks

For more information, consider visiting the documentation available here.

Class ScrollingMenu

Namespace: ConsoleAppVisuals.InteractiveElements

Assembly: ConsoleAppVisuals.dll

The ScrollingMenu is an interactive element that displays a question with multiple scrollable choices.

```
public class ScrollingMenu : InteractiveElement<int>
```

Inheritance

object ← Element ← InteractiveElement<int> ← ScrollingMenu

Inherited Members

InteractiveElement<int>._interactionResponse , InteractiveElement<int>.SetInteractionResponse(object, InteractionEventArgs<int>) , InteractiveElement<int>.SendResponse(object, InteractionEventArgs<int>) , InteractiveElement<int>.GetResponse() , InteractiveElement<int>.GetResponseHistory() , InteractiveElement<int>.RenderOptionsBeforeHand() , InteractiveElement<int>.RenderOptionsAfterHand() , InteractiveElement<int>.Type , InteractiveElement<int>.MaxNumberOfThisElement , InteractiveElement<int>.EndOfInteractionEvent , Element.ToggleVisibility() , Element.RenderElement() , Element.RenderElementSpace(bool) , Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility , Element.TextAlignment , Element.Line , object.GetType() , object.MemberwiseClone() , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

ScrollingMenu(string, int, Placement, params string[])

The ScrollingMenu is an interactive element that displays a question with multiple scrollable choices.

```
public ScrollingMenu(string question, int defaultIndex = 0, Placement placement =
Placement.TopCenter, params string[] choices)
```

Parameters

`question` string

The question to ask the user.

`defaultIndex` int

The index of the default choice(initially 0).

`placement` Placement

The placement of the menu on the console.

`choices` string[]

The different choices of the menu.

Remarks

For more information, consider visiting the documentation available here.

Properties

Choices

Gets the different choices of the menu.

```
public string[] Choices { get; }
```

Property Value

string[]

DefaultIndex

Gets the index of the default choice.

```
public int DefaultIndex { get; }
```

Property Value

int

Height

Gets the height of the menu.

```
public override int Height { get; }
```

Property Value

int

Placement

Gets the placement of the menu on the console.

```
public override Placement Placement { get; }
```

Property Value

Placement

Question

Gets the question to ask the user.

```
public string Question { get; }
```

Property Value

string

Selector

Gets the selector char of the menu.

```
public char Selector { get; }
```

Property Value

char

Width

Gets the width of the menu.

```
public override int Width { get; }
```

Property Value

int

Methods

RenderElementActions()

Defines the actions to perform when the element is called to be rendered on the console.

```
[Visual]
protected override void RenderElementActions()
```

UpdateChoices(params string[])

Updates the choices of the menu.

```
public void UpdateChoices(params string[] choices)
```

Parameters

`choices` string[]

The new choices of the menu.

Remarks

For more information, consider visiting the documentation available here.

UpdateDefaultIndex(int)

Updates the default index of the menu.

```
public void UpdateDefaultIndex(int defaultIndex)
```

Parameters

`defaultIndex` int

The new default index of the menu.

Remarks

For more information, consider visiting the documentation available here.

UpdatePlacement(Placement)

Updates the placement of the menu.

```
public void UpdatePlacement(Placement placement)
```

Parameters

`placement` Placement

The new placement of the menu.

Remarks

For more information, consider visiting the documentation available here.

UpdateQuestion(string)

Updates the question of the menu.

```
public void UpdateQuestion(string question)
```

Parameters

`question` string

The new question of the menu.

Remarks

For more information, consider visiting the documentation available here.

UpdateSelector(char)

Updates the selector of the menu.

```
public void UpdateSelector(char selector = '▶')
```

Parameters

`selector` char

The new selector of the menu.

Remarks

For more information, consider visiting the documentation available here.

Class TableSelector

Namespace: ConsoleAppVisuals.InteractiveElements

Assembly: ConsoleAppVisuals.dll

The TableSelector is an interactive element that displays a table with selectable elements.

```
public class TableSelector : InteractiveElement<int>
```

Inheritance

object ← Element ← InteractiveElement<int> ← TableSelector

Inherited Members

InteractiveElement<int>._interactionResponse , InteractiveElement<int>.SetInteractionResponse(object, InteractionEventArgs<int>) , InteractiveElement<int>.SendResponse(object, InteractionEventArgs<int>) , InteractiveElement<int>.GetResponse() , InteractiveElement<int>.GetResponseHistory() , InteractiveElement<int>.RenderOptionsBeforeHand() , InteractiveElement<int>.RenderOptionsAfterHand() , InteractiveElement<int>.Type , InteractiveElement<int>.MaxNumberOfThisElement , InteractiveElement<int>.EndOfInteractionEvent , Element.ToggleVisibility() , Element.RenderElement() , Element.RenderElementSpace(bool) , Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility , Element.TextAlignment , Element.Line , object.GetType() , object.MemberwiseClone() , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

TableSelector(string?, List<string>?, List<List<string>>?, bool, bool, string?, Placement, BordersType)

The TableSelector is an interactive element that displays a table with selectable elements.

```
public TableSelector(string? title = null, List<string>? headers = null,
List<List<string>>? lines = null, bool excludeHeader = true, bool excludeFooter =
true, string? footerText = null, Placement placement = Placement.TopCenter,
BordersType bordersType = BordersType.SingleStraight)
```

Parameters

`title` string

The title of the table.

`headers` List<string>

The headers of the table.

`lines` List<List<string>>

The lines of the table.

`excludeHeader` bool

Whether to exclude the header from selectable elements.

`excludeFooter` bool

Whether to exclude the footer from selectable elements.

`footerText` string

The text to display in the footer.

`placement` Placement

The placement of the table.

`bordersType` BordersType

The type of the borders of the table.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentException

Thrown when the number of columns in the table is not consistent with itself or with the headers.

NullReferenceException

Thrown when no body lines were provided.

Properties

Borders

Gets the borders manager of the table.

```
public Borders Borders { get; }
```

Property Value

Borders

BordersType

Gets the type of the borders of the table.

```
public BordersType BordersType { get; }
```

Property Value

BordersType

Count

Gets the number of lines in the table.

```
public int Count { get; }
```

Property Value

int

DisplayArray

Gets the display array of the table.

```
public string[]? DisplayArray { get; }
```

Property Value

string[]

ExcludeFooter

Gets if the footer is excluded.

```
public bool ExcludeFooter { get; }
```

Property Value

bool

ExcludeHeader

Gets if the header is excluded.

```
public bool ExcludeHeader { get; }
```

Property Value

bool

FooterText

Gets the text of the footer.

```
public string FooterText { get; }
```

Property Value

string

GetRawHeaders

Gets the headers of the table.

```
public List<string>? GetRawHeaders { get; }
```

Property Value

List<string>

GetRawLines

Gets the lines of the table.

```
public List<List<string>>? GetRawLines { get; }
```

Property Value

List<List<string>>

Height

Gets the height of the table.

```
public override int Height { get; }
```

Property Value

int

Placement

Gets the placement of the table on the console.

```
public override Placement Placement { get; }
```

Property Value

Placement

Title

Gets the title of the table.

```
public string? Title { get; }
```

Property Value

string

Width

Gets the width of the table.

```
public override int Width { get; }
```

Property Value

int

Methods

AddHeaders(List<string>)

Adds headers to the table.

```
public void AddHeaders(List<string> headers)
```

Parameters

`headers` List<string>

The headers to add.

Remarks

For more information, consider visiting the documentation available here.

AddLine(List<string>)

Adds a line to the table.

```
public void AddLine(List<string> line)
```

Parameters

`line` List<string>

The line to add.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentException

Thrown when the number of columns in the table is not consistent with itself or with the headers.

AddTitle(string)

Adds a title to the table.

```
public void AddTitle(string title)
```

Parameters

`title` string

The title to add.

Remarks

For more information, consider visiting the documentation available here.

ClearHeaders()

Clears the headers of the table.

```
public void ClearHeaders()
```

Remarks

For more information, consider visiting the documentation available here.

ClearLines()

Clears the lines of the table.

```
public void ClearLines()
```

Remarks

For more information, consider visiting the documentation available here.

GetColumnData(int)

Gets all the elements from a column given its index.

```
public List<string>? GetColumnData(int index)
```

Parameters

`index` int

The index of the column.

Returns

List<string>

The elements of the column.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Thrown when the index is out of range.

GetColumnData(string)

Gets all the elements from a column given its header.

```
public List<string>? GetColumnData(string header)
```

Parameters

`header` string

The header of the column.

Returns

List<string>

The elements of the column.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentException

Thrown when the header is invalid.

GetLine(int)

Gets the specified line in the table.

```
public List<string> GetLine(int index)
```

Parameters

`index` int

The index of the line to return.

Returns

List<string>

The line at the specified index.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Thrown when the index is out of range.

RemoveLine(int)

Removes a line from the table given its index.

```
public void RemoveLine(int index)
```

Parameters

`index` int

The index of the line to remove.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Thrown when the index is out of range.

RenderElementActions()

Defines the actions to perform when the element is called to be rendered on the console.

```
[Visual]
protected override void RenderElementActions()
```

Reset()

Clears the table.

```
public void Reset()
```

Remarks

For more information, consider visiting the documentation available here.

SetExcludeFooter(bool)

Sets the table to exclude the footer.

```
public void SetExcludeFooter(bool excludeFooter = true)
```

Parameters

`excludeFooter` bool

Whether to exclude the footer or not.

Remarks

For more information, consider visiting the documentation available here.

SetExcludeHeader(bool)

Sets the table to exclude the header.

```
public void SetExcludeHeader(bool excludeHeader = true)
```

Parameters

`excludeHeader` bool

Whether to exclude the header or not.

Remarks

For more information, consider visiting the documentation available here.

UpdateBordersType(BordersType)

Updates the type of the borders of the table.

```
public void UpdateBordersType(BordersType bordersType)
```

Parameters

`bordersType` BordersType

The type of the borders of the table.

Remarks

For more information, consider visiting the documentation available here.

UpdateFooterText(string)

Updates the text of the footer.

```
public void UpdateFooterText(string footerText)
```

Parameters

`footerText` string

The text of the footer.

Remarks

For more information, consider visiting the documentation available here.

UpdateHeaders(List<string>)

Updates the headers of the table.

```
public void UpdateHeaders(List<string> headers)
```

Parameters

`headers` List<string>

The headers to update.

Remarks

For more information, consider visiting the documentation available here.

UpdateLine(int, List<string>)

Updates a line in the table.

```
public void UpdateLine(int index, List<string> line)
```

Parameters

`index` int

The index of the line to update.

`line` List<string>

The new line.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Thrown when the index is out of range.

ArgumentException

Thrown when the number of columns in the table is not consistent with itself or with the headers.

UpdatePlacement(Placement)

Updates the placement of the table.

```
public void UpdatePlacement(Placement placement)
```

Parameters

`placement` Placement

The placement of the table.

Remarks

For more information, consider visiting the documentation available here.

UpdateTitle(string)

Updates the title of the table.

```
public void UpdateTitle(string title)
```

Parameters

`title` string

The title to update.

Remarks

For more information, consider visiting the documentation available here.

Namespace ConsoleAppVisuals.Models

Classes

AnimatedElement

The AnimatedElement class is an abstract class that represents an element that can be rendered on the console and animated.

Borders

The Borders class defines the border characters to use for embed elements.

Element

The Element class is an abstract class that represents an element that can be rendered on the console.

FontYamlFile

The FontYamlFile class defines the structure of a yaml file used to store the height of each character of a font.

InteractionEventArgs<T>

The `InteractionEventArgs` class is a generic class that represents the event arguments for the interactive elements.

InteractiveElement<TResponse>

The `InteractiveElement` class is an abstract class that represents the interactive elements.

PassiveElement

The `PassiveElement` class is an abstract class that represents the passive elements.

TextStyler

The TextStyler class is a class that styles text with the font files.

Structs

Position

The `Position` struct represents a position in the console with a line and a column.

Class AnimatedElement

Namespace: ConsoleAppVisuals.Models

Assembly: ConsoleAppVisuals.dll

The AnimatedElement class is an abstract class that represents an element that can be rendered on the console and animated.

```
public abstract class AnimatedElement : Element
```

Inheritance

object ← Element ← AnimatedElement

Derived

FakeLoadingBar, LoadingBar

Inherited Members

Element.ToggleVisibility() , Element.RenderElement() , Element.RenderElementActions() , Element.RenderOptionsBeforeHand() , Element.RenderElementSpace(bool) , Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility , Element.Height , Element.Width , Element.Placement , Element.TextAlignment , Element.Line , object.GetType() , object.MemberwiseClone() , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

AnimatedElement()

```
protected AnimatedElement()
```

Properties

MaxNumberOfThisElement

Gets the maximum number of this element that can be displayed on the console simultaneously.

```
public override sealed int MaxNumberOfThisElement { get; }
```

Property Value

int

Type

Gets the type of the element.

```
public override sealed ElementType Type { get; }
```

Property Value

ElementType

Methods

RenderOptionsAfterHand()

Deactivates the element after having been rendered.

```
[Visual]
protected override sealed void RenderOptionsAfterHand()
```

Class Borders

Namespace: ConsoleAppVisuals.Models

Assembly: ConsoleAppVisuals.dll

The Borders class defines the border characters to use for embed elements.

```
public class Borders
```

Inheritance

object ← Borders

Inherited Members

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

Remarks

For more information, consider visiting the documentation available here.

Constructors

Borders(BordersType)

The Borders class defines the border characters to use for embed elements.

```
public Borders(BordersType type = BordersType.SingleStraight)
```

Parameters

`type` BordersType

The type of border to use for the element.

Remarks

For more information, consider visiting the documentation available here.

Properties

Bottom

Gets the bottom line of the border. (┴)

```
public char Bottom { get; }
```

Property Value

char

BottomLeft

Gets the bottom-left corner of the border. (└)

```
public char BottomLeft { get; }
```

Property Value

char

BottomRight

Gets the bottom-right corner of the border. (┘)

```
public char BottomRight { get; }
```

Property Value

char

Cross

Gets the cross of the border. (┼)

```
public char Cross { get; }
```

Property Value

char

Horizontal

Gets the horizontal line of the border. (─)

```
public char Horizontal { get; }
```

Property Value

char

Left

Gets the left line of the border. (├)

```
public char Left { get; }
```

Property Value

char

Right

Gets the right line of the border. (┤)

```
public char Right { get; }
```

Property Value

char

Top

Gets the top line of the border. (┬)

```
public char Top { get; }
```

Property Value

char

TopLeft

Gets the top-left corner of the border. (┌)

```
public char TopLeft { get; }
```

Property Value

char

TopRight

Gets the top-right corner of the border. (┐)

```
public char TopRight { get; }
```

Property Value

char

Type

Gets the type of border to use for the element.

```
public BordersType Type { get; }
```

Property Value

BordersType

Vertical

Gets the vertical line of the border. (|)

```
public char Vertical { get; }
```

Property Value

char

Methods

GetBorderChar(int)

Gets the border character at the specified index.

```
public char GetBorderChar(int index)
```

Parameters

`index` int

The index of the border character to get.

Returns

char

The border character at the specified index.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentException

Thrown when the border type is invalid.

ArgumentOutOfRangeException

Thrown when the index is out of range.

UpdateBordersType(BordersType)

Updates the border type of the element.

```
public void UpdateBordersType(BordersType newType)
```

Parameters

`newType` BordersType

The new border type to use.

Remarks

For more information, consider visiting the documentation available here.

Class Element

Namespace: ConsoleAppVisuals.Models

Assembly: ConsoleAppVisuals.dll

The Element class is an abstract class that represents an element that can be rendered on the console.

```
public abstract class Element
```

Inheritance

object ← Element

Derived

AnimatedElement, InteractiveElement<TResponse>, PassiveElement

Inherited Members

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

Remarks

For more information, consider visiting the documentation available here.

Constructors

Element()

```
protected Element()
```

Properties

Height

Gets the height of the element, the vertical number of lines taken in the console.

```
public virtual int Height { get; }
```

Property Value

int

Remarks

This property is marked as virtual. It is recommended to override this property in derived classes to make it more specific.

Id

Gets the id number of the element.

```
public int Id { get; set; }
```

Property Value

int

Remarks

This property is sealed. The ID of an element is automatically generated and managed by the Window class.

Line

Gets a line to place the element in the console.

```
public virtual int Line { get; }
```

Property Value

int

Remarks

ATTENTION: This property is not marked as virtual. Override this property only to give it a constant value.

Exceptions

ArgumentOutOfRangeException

Thrown when the placement of the element is invalid.

MaxNumberOfThisElement

Gets the maximum number of this element that can be drawn on the console.

```
public virtual int MaxNumberOfThisElement { get; }
```

Property Value

int

Remarks

This property is marked as virtual. It is recommended to override this property in derived classes to make it more specific.

Placement

Gets the placement of the element int the console. See the Placement enum to know the possible values.

```
public virtual Placement Placement { get; set; }
```

Property Value

Placement

Remarks

This property is marked as virtual. It is recommended to override this property in derived classes to make it more specific.

TextAlignment

Gets the text alignment of the text of the element. See the TextAlignment enum to know the possible values.

```
public virtual TextAlignment TextAlignment { get; set; }
```

Property Value

TextAlignment

Remarks

This property is marked as virtual. It is recommended to override this property in derived classes to make it more specific.

Type

Gets the type of the element.

```
[Visual]
public virtual ElementType Type { get; }
```

Property Value

ElementType

Visibility

Gets the visibility of the element.

```
public bool Visibility { get; }
```

Property Value

bool

Remarks

This property is sealed. The visibility of an element is managed by the ToggleVisibility() method.

Width

Gets the width of the element, the horizontal number of lines taken in the console.

```
public virtual int Width { get; }
```

Property Value

int

Remarks

This property is marked as virtual. It is recommended to override this property in derived classes to make it more specific.

Methods

Clear()

Clears the space taken by the element on the console.

```
[Visual]
public void Clear()
```

Remarks

For more information, consider visiting the documentation available here.

GetRenderSpace()

Gets the space taken by the element on the console.

```
[Visual]
protected virtual string[] GetRenderSpace()
```

Returns

string[]

The space taken by the element.

Remarks

This method is marked as virtual. It is recommended to override this method in derived classes to make it more specific.

RenderElement()

Renders the element on the console.

```
[Visual]
public void RenderElement()
```

Remarks

For more information, consider visiting the documentation available here.

RenderElementActions()

Defines the actions to perform when the element is called to be rendered on the console.

```
[Visual]
protected virtual void RenderElementActions()
```

Remarks

This method is marked as virtual. It is recommended to override this method in derived classes to make it more specific.

RenderElementSpace(bool)

Renders the space taken by the element on the console.

```
[Visual]
public void RenderElementSpace(bool ignoreVisibility = false)
```

Parameters

`ignoreVisibility` bool

Whether to ignore the visibility of the element or not.

Remarks

For more information, consider visiting the documentation available here.

RenderOptionsAfterHand()

Defines actions to perform after rendering the element on the console.

```
[Visual]
protected virtual void RenderOptionsAfterHand()
```

RenderOptionsBeforeHand()

Defines actions to perform before rendering the element on the console.

```
[Visual]
protected virtual void RenderOptionsBeforeHand()
```

ToggleVisibility()

Toggles the visibility of the element. If the maximum number of this element is reached, an exception is thrown.

```
public void ToggleVisibility()
```

Remarks

This method is effectively sealed. The only way to change the visibility of an element is to use this method.

Exceptions

InvalidOperationException

Thrown when the maximum number of this element is reached.

Class FontYamlFile

Namespace: ConsoleAppVisuals.Models

Assembly: ConsoleAppVisuals.dll

The FontYamlFile class defines the structure of a yaml file used to store the height of each character of a font.

```
public class FontYamlFile
```

Inheritance

object ← FontYamlFile

Inherited Members

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

Remarks

For more information, consider visiting the documentation available here.

Constructors

FontYamlFile()

```
public FontYamlFile()
```

Properties

Author

The author of the font.

```
[YamlMember(Alias = "author", ApplyNamingConventions = false)]
public string? Author { get; set; }
```

Property Value

string

Chars

The height of the elements of the font.

```
public Dictionary<string, string>? Chars { get; set; }
```

Property Value

Dictionary<string, string>

Height

The height of each font element.

```
[YamlMember(Alias = "height", ApplyNamingConventions = false)]
public int? Height { get; set; }
```

Property Value

int?

Name

The name of the font.

```
[YamlMember(Alias = "name", ApplyNamingConventions = false)]
public string? Name { get; set; }
```

Property Value

string

Class InteractionEventArgs<T>

Namespace: ConsoleAppVisuals.Models

Assembly: ConsoleAppVisuals.dll

The `InteractionEventArgs` class is a generic class that represents the event arguments for the interactive elements.

```
public class InteractionEventArgs<T> : EventArgs
```

Type Parameters

`T`

Inheritance

object ← EventArgs ← InteractionEventArgs<T>

Inherited Members

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

Remarks

For more information, consider visiting the documentation available here.

Constructors

InteractionEventArgs(Status, T)

The `InteractionEventArgs` class is a generic class that represents the event arguments for the interactive elements.

```
public InteractionEventArgs(Status status, T value)
```

Parameters

`status` Status

The status of the exit from the menu.

`value` T

The value of the response after exiting the interactive element.

Remarks

For more information, consider visiting the documentation available here.

Properties

Status

Gets the status after exiting the interactive element. See the Status enumeration to know the possible values.

```
public Status Status { get; set; }
```

Property Value

Status

Status.Escaped : pressed escape, Status.Deleted : pressed backspace, Status.Selected : pressed enter

Value

Gets the `T` value of the response after exiting the interactive element.

```
public T Value { get; set; }
```

Property Value

T

Class InteractiveElement<TResponse>

Namespace: ConsoleAppVisuals.Models

Assembly: ConsoleAppVisuals.dll

The `InteractiveElement` class is an abstract class that represents the interactive elements.

```
public abstract class InteractiveElement<TResponse> : Element
```

Type Parameters

`TResponse`

Inheritance

object ← Element ← InteractiveElement<TResponse>

Derived

Dialog, FloatSelector, IntSelector, Prompt, ScrollingMenu, TableSelector

Inherited Members

Element.ToggleVisibility() , Element.RenderElement() , Element.RenderElementActions() , Element.RenderElementSpace(bool) , Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility , Element.Height , Element.Width , Element.Placement , Element.TextAlignment , Element.Line , object.GetType() , object.MemberwiseClone() , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

InteractiveElement()

```
protected InteractiveElement()
```

Fields

_interactionResponse

The response of the user.

```
protected List<InteractionEventArgs<TResponse>?> _interactionResponse
```

Field Value

List<InteractionEventArgs<TResponse>>

Properties

MaxNumberOfThisElement

Gets the maximum number of this element that can be drawn on the console. As an interactive element, the value is 1.

```
public override sealed int MaxNumberOfThisElement { get; }
```

Property Value

int

Type

Gets the type of the element.

```
public override sealed ElementType Type { get; }
```

Property Value

ElementType

Methods

GetResponse()

Gets the response of the user after an interaction.

```
[Visual]
public InteractionEventArgs<TResponse>? GetResponse()
```

Returns

InteractionEventArgs<TResponse>

Null if the user has not interacted with the element, otherwise the response of the user.

Remarks

This sample shows how to use the GetResponse() method using the var keyword:

```
var response = element.GetResponse();
```

For more information, consider visiting the documentation available here.

GetResponseHistory()

Gets the history of the responses of the user after interactions.

```
[Visual]
public List<InteractionEventArgs<TResponse>?> GetResponseHistory()
```

Returns

List<InteractionEventArgs<TResponse>>

The history of the responses of the user after interactions.

Remarks

For more information, consider visiting the documentation available here.

RenderOptionsAfterHand()

Stops listening to the user's interactions with the element.

```
[Visual]
protected override sealed void RenderOptionsAfterHand()
```

Remarks

For more information, consider visiting the documentation available here.

RenderOptionsBeforeHand()

Listens to the user's interactions with the element.

```
[Visual]
protected override sealed void RenderOptionsBeforeHand()
```

Remarks

For more information, consider visiting the documentation available here.

SendResponse(object?, InteractionEventArgs<TResponse>)

Triggers the EndOfInteractionEvent event.

```
[Visual]
protected void SendResponse(object? sender, InteractionEventArgs<TResponse> e)
```

Parameters

`sender` object

The sender of the event.

`e` InteractionEventArgs<TResponse>

The response of the user.

SetInteractionResponse(object?, InteractionEventArgs<TResponse>)

Sets the response of the user in the attribute field.

```
[Visual]
protected void SetInteractionResponse(object? sender,
InteractionEventArgs<TResponse> e)
```

Parameters

`sender` object

The sender of the event.

`e` InteractionEventArgs<TResponse>

The response of the user.

Events

EndOfInteractionEvent

The event that is triggered when the user has interacted with the element.

```
public event EventHandler<InteractionEventArgs<TResponse>>? EndOfInteractionEvent
```

Event Type

EventHandler<InteractionEventArgs<TResponse>>

Class PassiveElement

Namespace: ConsoleAppVisuals.Models

Assembly: ConsoleAppVisuals.dll

The `PassiveElement` class is an abstract class that represents the passive elements.

```
public abstract class PassiveElement : Element
```

Inheritance

object ← Element ← PassiveElement

Derived

ASCIIArt, Banner, ElementsDashboard, ElementsList, EmbedText, Footer, Header, HeightSpacer, Matrix<T>, TableView, Text, Title

Inherited Members

Element.ToggleVisibility() , Element.RenderElement() , Element.RenderElementActions() , Element.RenderOptionsBeforeHand() , Element.RenderOptionsAfterHand() , Element.RenderElementSpace(bool) , Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility , Element.Height , Element.Width , Element.Placement , Element.TextAlignment , Element.MaxNumberOfThisElement , Element.Line , object.GetType() , object.MemberwiseClone() , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

PassiveElement()

```
protected PassiveElement()
```

Properties

Type

Gets the type of the element.

```
public override sealed ElementType Type { get; }
```

Property Value

ElementType

Struct Position

Namespace: ConsoleAppVisuals.Models

Assembly: ConsoleAppVisuals.dll

The `Position` struct represents a position in the console with a line and a column.

```
public struct Position : IEquatable<Position>
```

Implements
IEquatable<Position>

Inherited Members
object.GetType() , object.Equals(object, object) ,
object.ReferenceEquals(object, object)

Remarks

For more information, consider visiting the documentation available here.

Constructors

Position(Position)

Initializes a new instance of the Position class with another instance of the Position class.

```
public Position(Position pos)
```

Parameters

`pos` Position

The position to copy.

Remarks

For more information, consider visiting the documentation available here.

Position(int, int)

Initializes a new instance of the Position class with 2 coordinates.

```
public Position(int x, int y)
```

Parameters

x int

The x coordinate of the position.

y int

The y coordinate of the position.

Remarks

For more information, consider visiting the documentation available here.

Properties

X

Gets the x coordinate of the position.

```
public int X { readonly get; set; }
```

Property Value

int

Y

Gets the y coordinate of the position.

```
public int Y { readonly get; set; }
```

Property Value

int

Methods

Equals(object?)

Compares the position to another position.

```
public override readonly bool Equals(object? obj)
```

Parameters

`obj` object

The position to compare to.

Returns

bool

True if the positions are equal, false otherwise.

Remarks

For more information, consider visiting the documentation available here.

GetHashCode()

Gets the hash code of the position.

```
public override readonly int GetHashCode()
```

Returns

int

An integer representing the hash code of the position.

Remarks

For more information, consider visiting the documentation available here.

ToString()

Converts the position to a string.

```
public override readonly string ToString()
```

Returns

string

The position as a string.

Remarks

For more information, consider visiting the documentation available here.

Operators

operator ==(Position, Position)

Implements the operator to check if the position is equal to another position.

```
public static bool operator ==(Position left, Position right)
```

Parameters

`left` Position

The first position to compare.

`right` Position

The second position to compare.

Returns

bool

True if the positions are equal, false otherwise.

operator !=(Position, Position)

Implements the operator to check if the position is not equal to another position.

```
public static bool operator !=(Position left, Position right)
```

Parameters

`left` Position

The first position to compare.

`right` Position

The second position to compare.

Returns

bool

True if the positions are not equal, false otherwise.

Class TextStyler

Namespace: ConsoleAppVisuals.Models

Assembly: ConsoleAppVisuals.dll

The TextStyler class is a class that styles text with the font files.

```
public class TextStyler
```

Inheritance

object ← TextStyler

Inherited Members

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

Remarks

For more information, consider visiting the documentation available here.

Constructors

TextStyler(Font, string?, Assembly?)

The TextStyler class is a class that styles text with the font files.

```
public TextStyler(Font source = Font.ANSI_Shadow, string? fontPath = null, Assembly?
assembly = null)
```

Parameters

`source` Font

The font to use. Font.Custom if you want to use your own font.

`fontPath` string

ATTENTION: only use the path to the font files for custom fonts.

`assembly` Assembly

ATTENTION: Debug purposes only. Do not update it.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

EmptyFileException

Thrown when the config.yml file is empty.

Properties

Dictionary

Gets the dictionary that stores the characters and their styled equivalent.

```
public Dictionary<char, string> Dictionary { get; }
```

Property Value

Dictionary<char, string>

Font

Gets the font to use. Font.Custom if you want to use your own font.

```
public Font Font { get; }
```

Property Value

Font

FontPath

Gets the path to the font files. Null if the font is not custom.

```
public string? FontPath { get; }
```

Property Value

string

SupportedAlphabet

Gets the supported alphabet by the font.

```
public string SupportedAlphabet { get; }
```

Property Value

string

SupportedChars

Gets all the supported characters by the font.

```
public string SupportedChars { get; }
```

Property Value

string

SupportedNumbers

Gets the supported numbers by the font.

```
public string SupportedNumbers { get; }
```

Property Value

string

SupportedSymbols

Gets the supported symbols by the font.

```
public string SupportedSymbols { get; }
```

Property Value

string

Methods

Style(string)

Styles the given text with the font files.

```
public string[] Style(string text)
```

Parameters

`text` string

The text to style.

Returns

string[]

The styled text as a string array.

Remarks

For more information, consider visiting the documentation available here.

ToString()

Gets the info of the actual style (from the config.yml file).

```
public override string ToString()
```

Returns

string

A string compiling these pieces of information.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

EmptyFileException

Thrown when the config.yml file is empty.

Namespace ConsoleAppVisuals.Passive Elements

Classes

ASCIIArt
The ASCIIArt is a passive element that displays ASCII art from a file or string content.

Banner
The Banner is a passive element that displays a banner on the console.

ElementsDashboard
The ElementsDashboard is a passive element that displays a dashboard of all the elements currently in the Window class.

ElementsList
The ElementsList is a passive element that displays a list of all the elements types available.

EmbedText
The EmbedText is a passive element that displays text in a box.

Footer
The Footer is a passive element that displays a footer on the console.

Header
The Header is a passive element that displays a header on the console.

HeightSpacer
The HeightSpacer is a passive element that displays a space between elements with a fixed height.

Matrix<T>
The Matrix<T> is a passive element that displays a matrix on the console.

TableView
The TableView is a passive element that displays a table on the console.

Text
The Text is an passive element that displays one or multiple lines of text.

Title
The Title is a passive element that displays a title on the console.

Class ASCIIArt

Namespace: ConsoleAppVisuals.PassiveElements

Assembly: ConsoleAppVisuals.dll

The ASCIIArt is a passive element that displays ASCII art from a file or string content.

```
public class ASCIIArt : PassiveElement
```

Inheritance

object ← Element ← PassiveElement ← ASCIIArt

Inherited Members

PassiveElement.Type , Element.ToggleVisibility() , Element.RenderElement() , Element.RenderOptionsBeforeHand() , Element.RenderOptionsAfterHand() , Element.RenderElementSpace(bool) , Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility , Element.MaxNumberOfThisElement , Element.Line , object.GetType() , object.MemberwiseClone() , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

ASCIIArt(List<string>, Placement)

The ASCIIArt is a passive element that displays ASCII art from a list of strings.

```
public ASCIIArt(List<string> artLines, Placement placement = Placement.TopCenter)
```

Parameters

`artLines` List<string>

The lines of ASCII art to display.

`placement` Placement

The placement of the ASCIIArt element on the screen.

Remarks

For more information, consider visiting the documentation available here.

ASCIIArt(string, Placement)

The ASCIIArt is a passive element that displays ASCII art from a file.

```
public ASCIIArt(string filePath, Placement placement = Placement.TopCenter)
```

Parameters

`filePath` string

The path to the file containing the ASCII art.

`placement` Placement

The placement of the ASCIIArt element on the screen.

Remarks

For more information, consider visiting the documentation available here.

Properties

ArtLines

Gets the art lines of the ASCIIArt element.

```
public List<string> ArtLines { get; }
```

Property Value

List<string>

Height

Gets the height of the ASCIIArt element.

```
public override int Height { get; }
```

Property Value

int

Placement

Gets the position of the ASCIIArt element on the screen.

```
public override Placement Placement { get; }
```

Property Value

Placement

TextAlignment

Gets the alignment of the ASCIIArt element.

```
public override TextAlignment TextAlignment { get; }
```

Property Value

TextAlignment

Width

Gets the width of the ASCIIArt element.

```
public override int Width { get; }
```

Property Value

int

Methods

RenderElementActions()

Defines the actions to perform when the element is called to be rendered on the console.

```
[Visual]
protected override void RenderElementActions()
```

UpdateArt(string)

Updates the ASCII art from a multiline string.

```
public void UpdateArt(string art)
```

Parameters

`art` string

The new ASCII art as a multiline string.

Remarks

For more information, consider visiting the documentation available here.

UpdateArtLines(List<string>)

Updates the ASCII art lines.

```
public void UpdateArtLines(List<string> artLines)
```

Parameters

`artLines` List<string>

The new ASCII art lines.

Remarks

For more information, consider visiting the documentation available here.

UpdateFromFile(string)

Updates the ASCII art from a file.

```
public void UpdateFromFile(string filePath)
```

Parameters

`filePath` string

The path to the file containing the new ASCII art.

Remarks

For more information, consider visiting the documentation available here.

UpdatePlacement(Placement)

Updates the placement of the ASCIIArt element on the screen.

```
public void UpdatePlacement(Placement newPlacement)
```

Parameters

`newPlacement` Placement

The new placement of the ASCIIArt element.

Remarks

For more information, consider visiting the documentation available here.

Class Banner

Namespace: ConsoleAppVisuals.PassiveElements

Assembly: ConsoleAppVisuals.dll

The Banner is a passive element that displays a banner on the console.

```
public class Banner : PassiveElement
```

Inheritance

object ← Element ← PassiveElement ← Banner

Inherited Members

PassiveElement.Type , Element.ToggleVisibility() , Element.RenderElement() , Element.RenderOptionsBeforeHand() , Element.RenderOptionsAfterHand() , Element.RenderElementSpace(bool) , Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility , Element.TextAlignment , Element.MaxNumberOfThisElement , Element.Line , object.GetType() , object.MemberwiseClone() , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

Banner(string, string, string, int, int, Placement)

The Banner is a passive element that displays a banner on the console.

```
public Banner(string leftText = "Banner Left", string centerText = "Banner Center",
string rightText = "Banner Right", int upperMargin = 0, int lowerMargin = 0,
Placement placement = Placement.TopCenterFullWidth)
```

Parameters

`leftText` string

The text on the left of the banner.

`centerText` string

The text in the center of the banner.

`rightText` string

The text on the right of the banner.

`upperMargin` int

The upper margin of the banner.

`lowerMargin` int

The lower margin of the banner.

`placement` Placement

The placement of the banner.

Remarks

For more information, consider visiting the documentation available here.

Properties

Height

Gets the height of the banner.

```
public override int Height { get; }
```

Property Value

int

LowerMargin

Gets the lower margin of the banner.

```
public int LowerMargin { get; }
```

Property Value

int

Placement

Gets the placement of the banner.

```
public override Placement Placement { get; }
```

Property Value

Placement

Text

Gets the text of the banner.

```
public (string, string, string) Text { get; }
```

Property Value

(string, string, string)

UpperMargin

Gets the upper margin of the banner.

```
public int UpperMargin { get; }
```

Property Value

int

Width

Gets the width of the banner.

```
public override int Width { get; }
```

Property Value

int

Methods

RenderElementActions()

Defines the actions to perform when the element is called to be rendered on the console.

```
[Visual]
protected override void RenderElementActions()
```

UpdateCenterText(string)

Updates the text in the center of the banner.

```
public void UpdateCenterText(string centerText)
```

Parameters

`centerText` string

The new text in the center of the banner.

Remarks

For more information, consider visiting the documentation available here.

UpdateLeftText(string)

Updates the text on the left of the banner.

```
public void UpdateLeftText(string leftText)
```

Parameters

`leftText` string

The new text on the left of the banner.

Remarks

For more information, consider visiting the documentation available here.

UpdateLowerMargin(int)

Updates the lower margin of the banner.

```
public void UpdateLowerMargin(int lowerMargin)
```

Parameters

`lowerMargin` int

The new lower margin of the banner.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

The lower margin of the banner must be between 0 and the height of the console window.

UpdatePlacement(Placement)

Updates the placement of the banner.

```
public void UpdatePlacement(Placement placement)
```

Parameters

`placement` Placement

The new placement of the banner.

Remarks

For more information, consider visiting the documentation available here.

UpdateRightText(string)

Updates the text on the right of the banner.

```
public void UpdateRightText(string rightText)
```

Parameters

`rightText` string

The new text on the right of the banner.

Remarks

For more information, consider visiting the documentation available here.

UpdateUpperMargin(int)

Updates the upper margin of the banner.

```
public void UpdateUpperMargin(int upperMargin)
```

Parameters

`upperMargin` int

The new upper margin of the banner.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

The upper margin of the banner must be between 0 and the height of the console window.

Class ElementsDashboard

Namespace: ConsoleAppVisuals.PassiveElements

Assembly: ConsoleAppVisuals.dll

The ElementsDashboard is a passive element that displays a dashboard of all the elements currently in the Window class.

```
public class ElementsDashboard : PassiveElement
```

Inheritance

object ← Element ← PassiveElement ← ElementsDashboard

Inherited Members

PassiveElement.Type , Element.ToggleVisibility() , Element.RenderElement() , Element.RenderOptionsBeforeHand() , Element.RenderOptionsAfterHand() , Element.RenderElementSpace(bool) , Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility , Element.TextAlignment , Element.MaxNumberOfThisElement , Element.Line , object.GetType() , object.MemberwiseClone() , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

ElementsDashboard(Placement, BordersType)

The ElementsDashboard is a passive element that displays a dashboard of all the elements currently in the Window class.

```
public ElementsDashboard(Placement placement = Placement.TopCenter, BordersType
bordersType = BordersType.SingleStraight)
```

Parameters

`placement` Placement

The placement of the dashboard.

`bordersType` BordersType

The type of borders to be used in the dashboard.

Remarks

For more information, consider visiting the documentation available here.

Properties

Borders

Gets the border manager of the dashboard.

```
public Borders Borders { get; }
```

Property Value

Borders

BordersType

Gets the type of borders to be used in the dashboard.

```
public BordersType BordersType { get; }
```

Property Value

BordersType

Count

Gets the number of lines in the dashboard.

```
public int Count { get; }
```

Property Value

int

Headers

Gets the headers of the dashboard.

```
public static List<string> Headers { get; }
```

Property Value

List<string>

Height

This property returns the height of the dashboard.

```
public override int Height { get; }
```

Property Value

int

Lines

Gets the lines of the dashboard.

```
public List<List<string>> Lines { get; }
```

Property Value

List<List<string>>

Placement

Gets the title of the dashboard.

```
public override Placement Placement { get; }
```

Property Value

Placement

Title

Gets the title of the dashboard.

```
public static string Title { get; }
```

Property Value

string

Width

This property returns the width of the dashboard.

```
public override int Width { get; }
```

Property Value

int

Methods

RenderElementActions()

Defines the actions to perform when the element is called to be rendered on the console.

```
[Visual]
protected override void RenderElementActions()
```

UpdateBordersType(BordersType)

Updates the type of borders to be used in the dashboard.

```
public void UpdateBordersType(BordersType bordersType)
```

Parameters

`bordersType` BordersType

The new type of borders to be used in the dashboard.

Remarks

For more information, consider visiting the documentation available here.

UpdatePlacement(Placement)

Updates the placement of the dashboard.

```
public void UpdatePlacement(Placement placement)
```

Parameters

`placement` Placement

The new placement of the dashboard.

Remarks

For more information, consider visiting the documentation available here.

Class ElementsList

Namespace: ConsoleAppVisuals.PassiveElements

Assembly: ConsoleAppVisuals.dll

The ElementsList is a passive element that displays a list of all the elements types available.

```
public class ElementsList : PassiveElement
```

Inheritance

object ← Element ← PassiveElement ← ElementsList

Inherited Members

PassiveElement.Type , Element.ToggleVisibility() , Element.RenderElement() , Element.RenderOptionsBeforeHand() , Element.RenderOptionsAfterHand() , Element.RenderElementSpace(bool) , Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility , Element.TextAlignment , Element.MaxNumberOfThisElement , Element.Line , object.GetType() , object.MemberwiseClone() , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

ElementsList(ElementType, Placement, BordersType)

The ElementsList is a passive element that displays a list of all the elements types available.

```
public ElementsList(ElementType elementTypeExpected = ElementType.Default,
Placement placement = Placement.TopCenter, BordersType bordersType
= BordersType.SingleStraight)
```

Parameters

`elementTypeExpected` ElementType

The type of element expected.

`placement` Placement

The placement of the InteractiveList.

`bordersType` BordersType

The type of borders of the InteractiveList.

Remarks

For more information, consider visiting the documentation available here.

Properties

Borders

Gets the borders of the InteractiveList.

```
public Borders Borders { get; }
```

Property Value

Borders

BordersType

Gets the type of borders of the InteractiveList.

```
public BordersType BordersType { get; }
```

Property Value

BordersType

Count

Gets the number of lines in the InteractiveList.

```
public int Count { get; }
```

Property Value

int

ElementsTypeExpected

Gets the type of element expected.

```
public ElementType ElementsTypeExpected { get; }
```

Property Value

ElementType

Headers

Gets the headers of the dashboard.

```
public static List<string> Headers { get; }
```

Property Value

List<string>

Height

Gets the height of the InteractiveList.

```
public override int Height { get; }
```

Property Value

int

Lines

Gets the lines of the InteractiveList.

```
public List<List<string>> Lines { get; }
```

Property Value

List<List<string>>

Placement

Gets the title of the InteractiveList.

```
public override Placement Placement { get; }
```

Property Value

Placement

Title

Gets the title of the InteractiveList.

```
public string Title { get; }
```

Property Value

string

Width

Gets the width of the InteractiveList.

```
public override int Width { get; }
```

Property Value

int

Methods

RenderElementActions()

Defines the actions to perform when the element is called to be rendered on the console.

```
[Visual]
protected override void RenderElementActions()
```

UpdateBordersType(BordersType)

Updates the type of borders of the InteractiveList.

```
public void UpdateBordersType(BordersType bordersType)
```

Parameters

`bordersType` BordersType

The new type of borders of the InteractiveList.

Remarks

For more information, consider visiting the documentation available here.

UpdateElementsTypeExpected(ElementType)

Updates the type of element expected.

```
public void UpdateElementsTypeExpected(ElementType elementsTypeExpected)
```

Parameters

`elementsTypeExpected` ElementType

The new type of element expected.

Remarks

For more information, consider visiting the documentation available here.

UpdatePlacement(Placement)

Updates the placement of the InteractiveList.

```
public void UpdatePlacement(Placement placement)
```

Parameters

`placement` Placement

The new placement of the InteractiveList.

Remarks

For more information, consider visiting the documentation available here.

Class EmbedText

Namespace: ConsoleAppVisuals.PassiveElements

Assembly: ConsoleAppVisuals.dll

The EmbedText is a passive element that displays text in a box.

```
public class EmbedText : PassiveElement
```

Inheritance

object ← Element ← PassiveElement ← EmbedText

Inherited Members

PassiveElement.Type , Element.ToggleVisibility() , Element.RenderElement() , Element.RenderOptionsBeforeHand() , Element.RenderOptionsAfterHand() , Element.RenderElementSpace(bool) , Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility , Element.MaxNumberOfThisElement , Element.Line , object.GetType() , object.MemberwiseClone() , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

EmbedText(List<string>, TextAlignment, Placement, BordersType)

The EmbedText is a passive element that displays text in a box.

```
public EmbedText(List<string> text, TextAlignment align = TextAlignment.Left,
Placement placement = Placement.TopCenter, BordersType bordersType
= BordersType.SingleStraight)
```

Parameters

`text` List<string>

The text to display.

`align` TextAlignment

The alignment of the Embed text.

`placement` Placement

The placement of the Embed text element.

`bordersType` BordersType

The type of border to display.

Remarks

For more information, consider visiting the documentation available here.

Properties

Borders

Gets the borders of the Embed text.

```
public Borders Borders { get; }
```

Property Value

Borders

BordersType

Gets the border type of the selector.

```
public BordersType BordersType { get; }
```

Property Value

BordersType

Height

Gets the height of the Embed text.

```
public override int Height { get; }
```

Property Value

int

Lines

Gets the rows of the Embed text.

```
public List<string> Lines { get; }
```

Property Value

List<string>

Placement

Gets the position of the Embed text.

```
public override Placement Placement { get; }
```

Property Value

Placement

TextAlignment

Gets the alignment of the Embed text.

```
public override TextAlignment TextAlignment { get; }
```

Property Value

TextAlignment

TextToDisplay

Gets the text to display.

```
public List<string>? TextToDisplay { get; }
```

Property Value

List<string>

Width

Gets the width of the Embed text.

```
public override int Width { get; }
```

Property Value

int

Methods

AddLine(string)

Adds a line to the Embed text.

```
public void AddLine(string line)
```

Parameters

`line` string

The line to add.

Remarks

For more information, consider visiting the documentation available here.

InsertLine(int, string)

Inserts a line to the Embed text.

```
public void InsertLine(int index, string line)
```

Parameters

`index` int

The index where to insert the line.

`line` string

The line to insert.

Remarks

For more information, consider visiting the documentation available here.

RemoveLine(int)

Removes a line from the Embed text.

```
public void RemoveLine(int index)
```

Parameters

`index` int

The index of the line to remove.

Remarks

For more information, consider visiting the documentation available here.

RemoveLine(string)

Removes a line from the Embed text.

```
public void RemoveLine(string line)
```

Parameters

`line` string

The line to remove.

Remarks

For more information, consider visiting the documentation available here.

RenderElementActions()

Defines the actions to perform when the element is called to be rendered on the console.

```
[Visual]
protected override void RenderElementActions()
```

UpdateBordersType(BordersType)

Updates the borders of the Embed text.

```
public void UpdateBordersType(BordersType bordersType)
```

Parameters

`bordersType` BordersType

The new border type of the Embed text.

Remarks

For more information, consider visiting the documentation available here.

UpdateLines(List<string>)

Updates the text of the Embed text.

```
public void UpdateLines(List<string> newText)
```

Parameters

`newText` List<string>

The new text of the Embed text.

Remarks

For more information, consider visiting the documentation available here.

UpdatePlacement(Placement)

Updates the placement of the Embed text.

```
public void UpdatePlacement(Placement newPlacement)
```

Parameters

`newPlacement` Placement

The new placement of the Embed text.

Remarks

For more information, consider visiting the documentation available here.

UpdateTextAlignment(TextAlignment)

Updates the alignment of the Embed text.

```
public void UpdateTextAlignment(TextAlignment newAlignment)
```

Parameters

`newAlignment` TextAlignment

The new alignment of the Embed text.

Remarks

For more information, consider visiting the documentation available here.

Class Footer

Namespace: ConsoleAppVisuals.PassiveElements

Assembly: ConsoleAppVisuals.dll

The Footer is a passive element that displays a footer on the console.

```
public class Footer : PassiveElement
```

Inheritance

object ← Element ← PassiveElement ← Footer

Inherited Members

PassiveElement.Type , Element.ToggleVisibility() , Element.RenderElement() , Element.RenderOptionsBeforeHand() , Element.RenderOptionsAfterHand() , Element.RenderElementSpace(bool) , Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility , Element.TextAlignment , Element.Line , object.GetType() , object.MemberwiseClone() , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

Footer(string, string, string)

The Footer is a passive element that displays a footer on the console.

```
public Footer(string leftText = "Footer Left", string centerText = "Footer Center",
string rightText = "Footer Right")
```

Parameters

`leftText` string

The text on the left of the footer.

`centerText` string

The text in the center of the footer.

`rightText` string

The text on the right of the footer.

Remarks

For more information, consider visiting the documentation available here.

Properties

Height

Gets the height of the footer.

```
public override int Height { get; }
```

Property Value

int

MaxNumberOfThisElement

Gets the maximum number of this element.

```
public override int MaxNumberOfThisElement { get; }
```

Property Value

int

Placement

Gets the placement of the footer.

```
public override Placement Placement { get; }
```

Property Value

Placement

Text

Gets the text of the footer.

```
public (string, string, string) Text { get; }
```

Property Value

(string, string, string)

Width

Gets the width of the footer.

```
public override int Width { get; }
```

Property Value

int

Methods

RenderElementActions()

Defines the actions to perform when the element is called to be rendered on the console.

```
[Visual]
protected override void RenderElementActions()
```

UpdateCenterText(string)

Updates the text in the center of the footer.

```
public void UpdateCenterText(string centerText)
```

Parameters

`centerText` string

The new text in the center of the footer.

Remarks

For more information, consider visiting the documentation available here.

UpdateLeftText(string)

Updates the text on the left of the footer.

```
public void UpdateLeftText(string leftText)
```

Parameters

`leftText` string

The new text on the left of the footer.

Remarks

For more information, consider visiting the documentation available here.

UpdateRightText(string)

Updates the text on the right of the footer.

```
public void UpdateRightText(string rightText)
```

Parameters

`rightText` string

The new text on the right of the footer.

Remarks

For more information, consider visiting the documentation available here.

Class Header

Namespace: ConsoleAppVisuals.PassiveElements

Assembly: ConsoleAppVisuals.dll

The Header is a passive element that displays a header on the console.

```
public class Header : PassiveElement
```

Inheritance

object ← Element ← PassiveElement ← Header

Inherited Members

PassiveElement.Type , Element.ToggleVisibility() , Element.RenderElement() , Element.RenderOptionsBeforeHand() , Element.RenderOptionsAfterHand() , Element.RenderElementSpace(bool) , Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility , Element.TextAlignment , Element.Line , object.GetType() , object.MemberwiseClone() , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

Header(string, string, string, int)

The Header is a passive element that displays a header on the console.

```
public Header(string leftText = "Header Left", string centerText = "Header Center",
string rightText = "Header Right", int margin = 1)
```

Parameters

`leftText` string

The text on the left of the header.

`centerText` string

The text in the center of the header.

`rightText` string

The text on the right of the header.

`margin` int

The margin of the header.

Remarks

For more information, consider visiting the documentation available here.

Properties

Height

Gets the height of the header.

```
public override int Height { get; }
```

Property Value

int

Margin

Gets the margin of the header.

```
public int Margin { get; }
```

Property Value

int

MaxNumberOfThisElement

Gets the maximum number of this element.

```
public override int MaxNumberOfThisElement { get; }
```

Property Value

int

Placement

Gets the placement of the header.

```
public override Placement Placement { get; }
```

Property Value

Placement

Text

Gets the text of the header.

```
public (string, string, string) Text { get; }
```

Property Value

(string, string, string)

Width

Gets the width of the header.

```
public override int Width { get; }
```

Property Value

int

Methods

RenderElementActions()

Defines the actions to perform when the element is called to be rendered on the console.

```
[Visual]
protected override void RenderElementActions()
```

UpdateCenterText(string)

Updates the text in the center of the header.

```
public void UpdateCenterText(string centerText)
```

Parameters

`centerText` string

The new text in the center of the header.

Remarks

For more information, consider visiting the documentation available here.

UpdateLeftText(string)

Updates the text on the left of the header.

```
public void UpdateLeftText(string leftText)
```

Parameters

`leftText` string

The new text on the left of the header.

Remarks

For more information, consider visiting the documentation available here.

UpdateMargin(int)

Updates the margin of the header.

```
public void UpdateMargin(int margin)
```

Parameters

`margin` int

The new margin of the header.

Remarks

For more information, consider visiting the documentation available here.

UpdateRightText(string)

Updates the text on the right of the header.

```
public void UpdateRightText(string rightText)
```

Parameters

`rightText` string

The new text on the right of the header.

Remarks

For more information, consider visiting the documentation available here.

Class HeightSpacer

Namespace: ConsoleAppVisuals.PassiveElements

Assembly: ConsoleAppVisuals.dll

The HeightSpacer is a passive element that displays a space between elements with a fixed height.

```
public class HeightSpacer : PassiveElement
```

Inheritance

object ← Element ← PassiveElement ← HeightSpacer

Inherited Members

PassiveElement.Type , Element.ToggleVisibility() , Element.RenderElement() , Element.RenderOptionsBeforeHand() , Element.RenderOptionsAfterHand() , Element.RenderElementSpace(bool) , Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility , Element.Width , Element.TextAlignment , Element.MaxNumberOfThisElement , Element.Line , object.GetType() , object.MemberwiseClone() , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

HeightSpacer(int, Placement)

The HeightSpacer is a passive element that displays a space between elements with a fixed height.

```
public HeightSpacer(int height = 1, Placement placement = Placement.TopCenter)
```

Parameters

`height` int

The height of the spacer.

`placement` Placement

The placement of the spacer.

Remarks

For more information, consider visiting the documentation available here.

Properties

Height

Gets the height of the spacer.

```
public override int Height { get; }
```

Property Value

int

Placement

Gets the placement of the spacer.

```
public override Placement Placement { get; }
```

Property Value

Placement

Methods

RenderElementActions()

Defines the actions to perform when the element is called to be rendered on the console.

```
[Visual]
protected override void RenderElementActions()
```

UpdateHeight(int)

Updates the height of the spacer.

```
public void UpdateHeight(int newHeight)
```

Parameters

`newHeight` int

The new height of the spacer.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Thrown when the new height is negative or greater than the window height.

UpdatePlacement(Placement)

Updates the placement of the spacer.

```
public void UpdatePlacement(Placement newPlacement)
```

Parameters

`newPlacement` Placement

The new placement of the spacer.

Remarks

For more information, consider visiting the documentation available here.

Class Matrix<T>

Namespace: ConsoleAppVisuals.PassiveElements

Assembly: ConsoleAppVisuals.dll

The Matrix<T> is a passive element that displays a matrix on the console.

```
public class Matrix<T> : PassiveElement
```

Type Parameters

T

Inheritance

object ← Element ← PassiveElement ← Matrix<T>

Inherited Members

PassiveElement.Type , Element.ToggleVisibility() , Element.RenderElement() , Element.RenderOptionsBeforeHand() , Element.RenderOptionsAfterHand() , Element.RenderElementSpace(bool) , Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility , Element.TextAlignment , Element.MaxNumberOfThisElement , Element.Line , object.GetType() , object.MemberwiseClone() , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

Matrix(List<List<T?>>?, Placement, BordersType)

The Matrix<T> is a passive element that displays a matrix on the console.

```
public Matrix(List<List<T?>>? rawLines = null, Placement placement =
Placement.TopCenter, BordersType bordersType = BordersType.SingleStraight)
```

Parameters

`rawLines` List<List<T>>

The matrix to be used.

`placement` Placement

The placement of the matrix.

`bordersType` BordersType

The type of borders to use for the matrix.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentException

Thrown when the matrix is empty or not compatible (lines are not of the same length).

Properties

Borders

The border characters to use for the matrix.

```
public Borders Borders { get; }
```

Property Value

Borders

BordersType

The border type of the selector.

```
public BordersType BordersType { get; }
```

Property Value

BordersType

Count

Gets the number of lines in the matrix.

```
public int Count { get; }
```

Property Value

int

Height

Gets the height of the matrix.

```
public override int Height { get; }
```

Property Value

int

Placement

Gets the placement of the matrix.

```
public override Placement Placement { get; }
```

Property Value

Placement

Width

Gets the width of the matrix.

```
public override int Width { get; }
```

Property Value

int

Methods

AddLine(List<T?>)

Adds a line to the matrix.

```
public bool AddLine(List<T?> line)
```

Parameters

`line` List<T>

The line to add.

Returns

bool

True if the line has been added successfully, false otherwise.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentException

Thrown when the line is not of the same length as the other lines.

GetItem(Position)

Gets the element at the specified position.

```
public T? GetItem(Position position)
```

Parameters

`position` Position

The position of the element.

Returns

T

The element at the specified position.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Thrown when the position is out of range.

InsertLine(int, List<T?>)

Inserts a line at the specified index.

```
public bool InsertLine(int index, List<T?> line)
```

Parameters

`index` int

The index of the line to insert.

`line` List<T>

The line to insert.

Returns

bool

True if the line has been inserted successfully, false otherwise.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Thrown when the index is out of range.

ArgumentException

Thrown when the line is not of the same length as the other lines.

RemoveItem(Position)

Removes an element from the matrix.

```
public bool RemoveItem(Position position)
```

Parameters

`position` Position

The position of the element to remove.

Returns

bool

True if the element has been removed successfully, false otherwise.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Thrown when the position is out of range.

RemoveLine(int)

Removes a line from the matrix.

```
public bool RemoveLine(int index)
```

Parameters

`index` int

The index of the line to remove.

Returns

bool

True if the line has been removed successfully, false otherwise.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Thrown when the index is out of range.

RenderElementActions()

Defines the actions to perform when the element is called to be rendered on the console.

```
[Visual]
protected override void RenderElementActions()
```

UpdateBordersType(BordersType)

Updates the borders of the matrix.

```
public void UpdateBordersType(BordersType bordersType)
```

Parameters

`bordersType` BordersType

The new border type of the matrix.

Remarks

For more information, consider visiting the documentation available here.

UpdateItem(Position, T)

Updates an element in the matrix.

```
public bool UpdateItem(Position position, T item)
```

Parameters

`position` Position

The position of the element to update.

`item` T

The new element.

Returns

bool

True if the element has been updated successfully, false otherwise.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Thrown when the position is out of range.

UpdateLine(int, List<T?>)

Updates a line in the matrix.

```
public bool UpdateLine(int index, List<T?> line)
```

Parameters

`index` int

The index of the line to update.

`line` List<T>

The new line.

Returns

bool

True if the line has been updated successfully, false otherwise.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Thrown when the index is out of range.

ArgumentException

Thrown when the line is not of the same length as the other lines.

UpdatePlacement(Placement)

Updates the placement of the matrix.

```
public void UpdatePlacement(Placement placement)
```

Parameters

`placement` Placement

The new placement of the matrix.

Remarks

For more information, consider visiting the documentation available here.

Class TableView

Namespace: ConsoleAppVisuals.PassiveElements

Assembly: ConsoleAppVisuals.dll

The TableView is a passive element that displays a table on the console.

```
public class TableView : PassiveElement
```

Inheritance

object ← Element ← PassiveElement ← TableView

Inherited Members

PassiveElement.Type , Element.ToggleVisibility() , Element.RenderElement() , Element.RenderOptionsBeforeHand() , Element.RenderOptionsAfterHand() , Element.RenderElementSpace(bool) , Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility , Element.TextAlignment , Element.MaxNumberOfThisElement , Element.Line , object.GetType() , object.MemberwiseClone() , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

TableView(string?, List<string>?, List<List<string>>?, Placement, BordersType)

The TableView is a passive element that displays a table on the console.

```
public TableView(string? title = null, List<string>? headers = null,
List<List<string>>? lines = null, Placement placement = Placement.TopCenter,
BordersType bordersType = BordersType.SingleStraight)
```

Parameters

`title` string

The title of the table.

`headers` List<string>

The headers of the table.

`lines` List<List<string>>

The lines of the table.

`placement` Placement

The placement of the table.

`bordersType` BordersType

The type of borders to use for the table.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentException

Is thrown when the number of columns in the table is not consistent with itself or with the headers.

NullReferenceException

Is thrown when no body lines were provided.

Properties

Borders

Gets the borders of the table.

```
public Borders Borders { get; }
```

Property Value

Borders

BordersType

Gets the border type of the selector.

```
public BordersType BordersType { get; }
```

Property Value

BordersType

Count

Gets the number of lines in the table.

```
public int Count { get; }
```

Property Value

int

GetRawHeaders

Gets the headers of the table.

```
public List<string>? GetRawHeaders { get; }
```

Property Value

List<string>

GetRawLines

Gets the lines of the table.

```
public List<List<string>>? GetRawLines { get; }
```

Property Value

List<List<string>>

Height

Gets the height of the table.

```
public override int Height { get; }
```

Property Value

int

Placement

Gets the title of the table.

```
public override Placement Placement { get; }
```

Property Value

Placement

Width

Gets the width of the table.

```
public override int Width { get; }
```

Property Value

int

Methods

AddHeaders(List<string>)

Adds headers to the table.

```
public void AddHeaders(List<string> headers)
```

Parameters

`headers` List<string>

The headers to add.

Remarks

For more information, consider visiting the documentation available here.

AddLine(List<string>)

Adds a line to the table.

```
public void AddLine(List<string> line)
```

Parameters

`line` List<string>

The line to add.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentException

Is thrown when the number of columns in the table is not consistent with itself or with the headers.

AddTitle(string)

Adds a title to the table.

```
public void AddTitle(string title)
```

Parameters

`title` string

The title to add.

Remarks

For more information, consider visiting the documentation available here.

ClearHeaders()

Clears the headers of the table.

```
public void ClearHeaders()
```

Remarks

For more information, consider visiting the documentation available here.

ClearLines()

Clears the lines of the table.

```
public void ClearLines()
```

Remarks

For more information, consider visiting the documentation available here.

ClearTitle()

Clears the title of the table.

```
public void ClearTitle()
```

Remarks

For more information, consider visiting the documentation available here.

GetColumnData(int)

Gets all the elements from a column given its index.

```
public List<string>? GetColumnData(int index)
```

Parameters

`index` int

The index of the column.

Returns

List<string>

The elements of the column.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Is thrown when the index is out of range.

GetColumnData(string)

Gets all the elements from a column given its header.

```
public List<string>? GetColumnData(string header)
```

Parameters

`header` string

The header of the column.

Returns

List<string>

The elements of the column.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

InvalidOperationException

Is thrown when the table is empty.

ArgumentOutOfRangeException

Is thrown when the header is invalid.

GetLine(int)

Gets the specified line in the table.

```
public List<string> GetLine(int index)
```

Parameters

`index` int

The index of the line to return.

Returns

List<string>

The line at the specified index.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Is thrown when the index is out of range.

RemoveLine(int)

Removes a line from the table.

```
public void RemoveLine(int index)
```

Parameters

`index` int

The index of the line to remove.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Is thrown when the index is out of range.

RenderElementActions()

Defines the actions to perform when the element is called to be rendered on the console.

```
[Visual]
protected override void RenderElementActions()
```

Reset()

Clears the table.

```
public void Reset()
```

Remarks

For more information, consider visiting the documentation available here.

UpdateBordersType(BordersType)

Updates the borders of the table.

```
public void UpdateBordersType(BordersType bordersType)
```

Parameters

`bordersType` BordersType

The type of border to use for the table.

Remarks

For more information, consider visiting the documentation available here.

UpdateHeaders(List<string>)

Updates the headers of the table.

```
public void UpdateHeaders(List<string> headers)
```

Parameters

`headers` List<string>

The headers to update.

Remarks

For more information, consider visiting the documentation available here.

UpdateLine(int, List<string>)

Updates a line in the table.

```
public void UpdateLine(int index, List<string> line)
```

Parameters

`index` int

The index of the line to update.

`line` List<string>

The new line.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

Is thrown when the index is out of range.

ArgumentException

Is thrown when the number of columns in the table is not consistent with itself or with the headers.

UpdatePlacement(Placement)

Updates the placement of the table.

```
public void UpdatePlacement(Placement placement)
```

Parameters

`placement` Placement

The new placement of the table.

Remarks

For more information, consider visiting the documentation available here.

UpdateTitle(string)

Updates the title of the table.

```
public void UpdateTitle(string title)
```

Parameters

`title` string

The title to update.

Remarks

For more information, consider visiting the documentation available here.

Class Text

Namespace: ConsoleAppVisuals.PassiveElements

Assembly: ConsoleAppVisuals.dll

The Text is an passive element that displays one or multiple lines of text.

```
public class Text : PassiveElement
```

Inheritance

object ← Element ← PassiveElement ← Text

Inherited Members

PassiveElement.Type , Element.ToggleVisibility() , Element.RenderElement() , Element.RenderOptionsBeforeHand() , Element.RenderOptionsAfterHand() , Element.RenderElementSpace(bool) , Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility , Element.MaxNumberOfThisElement , Element.Line , object.GetType() , object.MemberwiseClone() , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

Text(List<string>, TextAlignment, Placement)

The Text is an passive element that displays one or multiple lines of text.

```
public Text(List<string> lines, TextAlignment align = TextAlignment.Left, Placement
placement = Placement.TopCenter)
```

Parameters

`lines` List<string>

The text to display.

`align` TextAlignment

The alignment of the Text element.

`placement` Placement

The placement of the Text element.

Remarks

For more information, consider visiting the documentation available here.

Properties

Height

Gets the height of the Text element.

```
public override int Height { get; }
```

Property Value

int

Lines

Gets the text of the Text element.

```
public List<string> Lines { get; }
```

Property Value

List<string>

Placement

Gets the position of the Text element.

```
public override Placement Placement { get; }
```

Property Value

Placement

TextAlignment

Gets the alignment of the Text element.

```
public override TextAlignment TextAlignment { get; }
```

Property Value

TextAlignment

TextToDisplay

Gets the text to display.

```
public List<string>? TextToDisplay { get; }
```

Property Value

List<string>

Width

Gets the width of the Text element.

```
public override int Width { get; }
```

Property Value

int

Methods

AddLine(string)

Adds a line to the Text element.

```
public void AddLine(string line)
```

Parameters

`line` string

The line to add.

Remarks

For more information, consider visiting the documentation available here.

InsertLine(int, string)

Inserts a line to the Text element.

```
public void InsertLine(int index, string line)
```

Parameters

`index` int

The index where to insert the line.

`line` string

The line to insert.

Remarks

For more information, consider visiting the documentation available here.

RemoveLine(int)

Removes a line from the Text element.

```
public void RemoveLine(int index)
```

Parameters

`index` int

The index of the line to remove.

Remarks

For more information, consider visiting the documentation available here.

RemoveLine(string)

Removes a line from the Text element.

```
public void RemoveLine(string line)
```

Parameters

`line` string

The line to remove.

Remarks

For more information, consider visiting the documentation available here.

RenderElementActions()

Defines the actions to perform when the element is called to be rendered on the console.

```
[Visual]
protected override void RenderElementActions()
```

UpdateLines(List<string>)

Updates the lines of the Text element.

```
public void UpdateLines(List<string> lines)
```

Parameters

`lines` List<string>

The new text of the Text element.

Remarks

For more information, consider visiting the documentation available here.

UpdatePlacement(Placement)

Updates the placement of the Text element.

```
public void UpdatePlacement(Placement newPlacement)
```

Parameters

`newPlacement` Placement

The new placement of the Text element.

Remarks

For more information, consider visiting the documentation available here.

UpdateTextAlignment(TextAlignment)

Updates the alignment of the Text element.

```
public void UpdateTextAlignment(TextAlignment newAlignment)
```

Parameters

`newAlignment` TextAlignment

The new alignment of the Text element.

Remarks

For more information, consider visiting the documentation available here.

Class Title

Namespace: ConsoleAppVisuals.PassiveElements

Assembly: ConsoleAppVisuals.dll

The Title is a passive element that displays a title on the console.

```
public class Title : PassiveElement
```

Inheritance

object ← Element ← PassiveElement ← Title

Inherited Members

PassiveElement.Type , Element.ToggleVisibility() , Element.RenderElement() , Element.RenderOptionsBeforeHand() , Element.RenderOptionsAfterHand() , Element.RenderElementSpace(bool) , Element.GetRenderSpace() , Element.Clear() , Element.Id , Element.Visibility , Element.TextAlignment , object.GetType() , object.MemberwiseClone() , object.ToString() , object.Equals(object) , object.Equals(object, object) , object.ReferenceEquals(object, object) , object.GetHashCode()

Remarks

For more information, consider visiting the documentation available here.

Constructors

Title(string, int, TextAlignment, Font, string?)

The Title is a passive element that displays a title on the console.

```
public Title(string text, int margin = 1, TextAlignment align =
TextAlignment.Center, Font font = Font.ANSI_Shadow, string? fontPath = null)
```

Parameters

`text` string

The text of the title.

`margin` int

The margin of the title.

`align` TextAlignment

The alignment of the title.

`font` Font

The font of the title.

`fontPath` string

ATTENTION: fill this parameter only if you want to use a custom font (Font.Custom).

Remarks

For more information, consider visiting the documentation available here.

Properties

Font

Gets the font of the title.

```
public Font Font { get; }
```

Property Value

Font

Height

Gets the height of the title.

```
public override int Height { get; }
```

Property Value

int

Line

Gets the line of the title.

```
public override int Line { get; }
```

Property Value

int

MaxNumberOfThisElement

Gets the maximum number of this element.

```
public override int MaxNumberOfThisElement { get; }
```

Property Value

int

Placement

Gets the placement of the title.

```
public override Placement Placement { get; }
```

Property Value

Placement

StyledText

Gets the styled text of the title.

```
public string[] StyledText { get; }
```

Property Value

string[]

Styler

Gets the text styler of the title.

```
public TextStyler Styler { get; }
```

Property Value

TextStyler

Width

Gets the width of the title.

```
public override int Width { get; }
```

Property Value

int

Methods

RenderElementActions()

Defines the actions to perform when the element is called to be rendered on the console.

```
[Visual]
protected override void RenderElementActions()
```

UpdateAlignment(TextAlignment)

Updates the alignment of the title.

```
public void UpdateAlignment(TextAlignment align)
```

Parameters

`align` TextAlignment

The new alignment of the title.

Remarks

For more information, consider visiting the documentation available here.

UpdateFont(Font, string?)

Updates the font of the title.

```
public void UpdateFont(Font font, string? fontPath = null)
```

Parameters

`font` Font

The new font of the title.

`fontPath` string

ATTENTION: fill this parameter only if you want to use a custom font (Font.Custom).

Remarks

For more information, consider visiting the documentation available here.

UpdateMargin(int)

Updates the margin of the title.

```
public void UpdateMargin(int margin)
```

Parameters

`margin` int

The new margin of the title.

Remarks

For more information, consider visiting the documentation available here.

Exceptions

ArgumentOutOfRangeException

The margin must be between 0 and the half of the console height.

UpdateText(string)

Updates the text of the title.

```
public void UpdateText(string text)
```

Parameters

`text` string

The new text of the title.

Remarks

For more information, consider visiting the documentation available here.