

AppleListbox

version 1.06
(Standard license)

About the class

The AppleListbox is a REALbasic drop-in control that is intended to mimic the left-hand sidebar found in many Apple Macintosh applications. It can be used to mimic the functionality of the browser in iTunes or Apple Mail or the sidebar in the Apple Finder. It can also be fully customised for use in your own applications to provide an attractive navigation class.

It will compile to all three platforms supported by REALbasic (Mac, Windows and Linux).

The AppleListbox is composed of three parts - the listbox itself (AppleListbox), a helper class (AppleListboxRow) and a collection of images. Included in your purchase are a collection of button images (and their masks) and these are required for the functionality of the class. The collection of assorted icon images are merely provided for illustration and you are free to use these as you wish.

Included in your purchase is an example project that illustrates the use of the AppleListbox.

Adding the class to your projects

To use the AppleListbox in a REALbasic project you must add all of its required components to the 'Project' tab in the REALbasic IDE. The easiest way to do this is to put the folder entitled 'AppleListbox Components' (found in your download) in the same directory as your project and then drag it from here onto the 'Project' tab in the REALbasic IDE. This will add the two required classes and the images to your project.

As you have purchased a standard license, all of the components of the AppleListbox are encrypted. This means that you are not able to view or modify the source code of the classes. In order to view the source code you must purchase a professional license. This can be purchased from our website (www.madebyfiga.com) or by contacting customer support at sales@madebyfiga.com.

To add an AppleListbox to your project simply drag an instance of one onto a window of your choosing in your project. You can find the AppleListbox control from the window editor by clicking on the 'Project Controls' popup menu on the left-hand side of the editor.

Please note that you *must* ensure that the parent window's Composite property is set to true on the Mac. If you do not do this - the disclosure triangles on folders in the listbox will be surrounded by a white square.

If you would like to enable drag reordering of AppleListboxRows you must enable the EnableDrag and EnableDragReorder properties in the properties inspector for the AppleListbox.

If you have any problems regarding the use of the AppleListbox control or would like to make suggestions or file a bug report then please either email us at support@madebyfiga.com or visit our support forums at www.madebyfiga.com/forums.

AppleListbox Events

Although the AppleListbox inherits from the built-in Listbox, the following events have been **removed**:

- DragRow

Some existing events have been **replaced** (that is, they have different arguments):

- DoubleClick
- DragReorderRows

Some events are unique to the AppleListbox:

- ClickedEjectButton
- ClickedLinkButton
- ClickedUpdateButton
- DraggingRow
- HoveringOverEjectButton
- HoveringOverLinkButton
- HoveringOverUpdateButton

AppleListbox.DoubleClick

This event is fired when the user double-clicks on an AppleListboxRow.

```
DoubleClick( theRow as AppleListboxRow )
```

Parameters

theRow

A pointer to the AppleListboxRow object double-clicked on in the listbox. Is *nil* if a row was not double-clicked.

AppleListbox.DragReorderRows

This event is fired when a row drag has been completed.

```
DragReorderRows(  
  rowDragged as AppleListboxRow,  
  destinationParent as AppleListboxRow,  
  oldParent as AppleListboxRow )
```

Parameters

rowDragged

The AppleListboxRow object that was dragged

destinationParent

The parent AppleListboxRow of *rowDragged* at its final destination

oldParent

The AppleListboxRow object that used to be the parent of *rowDragged*

AppleListbox.ClickedEjectButton

This event is fired when the user clicks on a visible row's eject button.

```
ClickedEjectButton( theRow as AppleListboxRow )
```

Parameters

theRow

The AppleListboxRow object whose eject button was clicked

AppleListbox.ClickedLinkButton

This event is fired when the user clicks on a visible row's link button.

```
ClickedLinkButton( theRow as AppleListboxRow )
```

Parameters

theRow

The AppleListboxRow object whose link button was clicked

Notes

If a row has a link button, it is only visible and, therefore, only able to respond to mouse clicks when the row is selected.

AppleListbox.ClickedUpdateButton

This event is fired when the user clicks on a visible row's update button.

```
ClickedUpdateButton( theRow as AppleListboxRow )
```

Parameters

theRow

The AppleListboxRow object whose update button was clicked

Notes

If a row has an update button, it is only visible and, therefore, only able to respond to mouse clicks when the row is selected.

AppleListbox.DraggingRow

This event fires when a user is dragging a row in the listbox.

```
DraggingRow( row as Integer )
```

Parameters

row

The row number of the row being dragged

AppleListbox.HoveringOverEjectButton

This event fires when the mouse is hovering overing a visible eject button in the listbox.

```
HoveringOverEjectButton( theRow as AppleListboxRow )
```

Parameters

theRow

The AppleListboxRow object to which the eject button belongs to.

AppleListbox.HoveringOverLinkButton

This event fires when the mouse is hovering overing a visible link button in the listbox.

```
HoveringOverLinkButton( theRow as AppleListboxRow )
```

Parameters

theRow

The AppleListboxRow object to which the link button belongs to.

AppleListbox.HoveringOverUpdateButton

This event fires when the mouse is hovering overing a visible update button in the listbox.

```
HoveringOverUpdateButton( theRow as AppleListboxRow )
```

Parameters

theRow

The AppleListboxRow object to which the update button belongs to.

AppleListbox Properties

AppleListbox.HasFocus

When true, the listbox currently has the focus.

HasFocus as Boolean

AppleListbox.MouseOverRow

The row number that the mouse is currently hovering over.

MouseOverRow as Integer

AppleListbox.ShowNewItem

When set to true, the listbox will display the value of each AppleListboxRow's NumberOfNewItem to the right of it. For collapsed folder rows, the cumulative total of all of it's children's NumberOfNewItem is displayed.

ShowNewItem as Boolean

AppleListbox.Root

Represents the top-level of the listbox.

Root as AppleListboxRow

Notes

Root is automatically created when the AppleListbox is first instanced. There should be no need to alter it directly.

AppleListbox.Style

The current style of the AppleListbox

Style as Integer

Notes

Use the following class constants to set the Style:

AppleListbox.Style_iTunes (0) : Dark blue gradient highlight effect for selected rows

AppleListbox.Style_Mail (1) : Light blue/grey gradient highlight effect for selected rows

AppleListbox Methods

AppleListbox.AddChildAtIndex

Adds a row to the listbox as a child of the specified parent at the requested index.

```
AddChildAtIndex(  
    rowToAdd as AppleListboxRow,  
    parentRow as AppleListboxRow,  
    index as Integer )
```

Parameters

rowToAdd

The AppleListboxRow object to add

parentRow

The AppleListboxRow object (that already exists in the listbox) to which you wish to add the *rowToAdd* object to

index

The position of *rowToAdd* (where 0 = the first child)

Notes

Maintains the current row selected in the listbox.

AppleListbox.AddRowAsSibling

Adds a row as a sibling of the specified row

```
AddRowAsSibling(  
    rowToAdd as AppleListboxRow,  
    sibling as AppleListboxRow )
```

Parameters

rowToAdd

The row to be added to the AppleListbox

sibling

This row's parent will have *rowToAdd* appended as a child row

Notes

AppleListbox.Root is not allowed any siblings.

AppleListbox.Append

Appends the passed row to the end of the AppleListbox.

```
Append( rowToAppend as AppleListboxRow )
```

Parameters

rowToAppend

The row to append

Notes

To allow customisation, non-category rows are permitted to be appended to the end of the tree however this behaviour is not really seen in any of Apple's sidebars.

AppleListbox.AppendRowToChild

Appends a row as a child of the passed parent row.

```
AppendRowToChild(  
    rowToAdd as AppleListboxRow,  
    parentRow as AppleListboxRow )
```

Parameters

rowToAdd

The row to add

parentRow

The AppleListboxRow object (that must already exist in the AppleListbox) that we will append *rowToAdd* to

AppleListbox.Collapse

Collapses the specified row

```
Collapse( row as Integer )
```

Parameters

row

The row number to collapse in the AppleListbox

Notes

Performs the same function as clicking on a disclosure triangle whilst it is in the expanded state.

AppleListbox.Expand

Expands the specified row

```
Expand( row as Integer )
```

Parameters

row

The row number to expand in the AppleListbox

Notes

Performs the same function as clicking on a disclosure triangle whilst it is in the collapsed state.

AppleListbox.GetChildAtRowNumber

Returns the AppleListboxRow object at the specified row number

```
result = GetChildAtRowNumber( row as Integer ) as AppleListboxRow
```

Parameters

row

The row number of the AppleListboxRow that you wish returned

Notes

Only top-level rows and rows inside expanded rows are counted in the row number. Row 0 is the first visible row.

AppleListbox.GetChildRowNumber

Returns the row number of the passed AppleListboxRow

```
result = GetChildRowNumber( theChild as AppleListboxRow ) as Integer
```

Parameters

theChild

The AppleListboxRow object that you would like the row number of

Notes

Only top-level rows and rows inside expanded rows are counted in the row number. Row 0 is the first visible row. If theChild is not found then this method returns -1.

AppleListbox.GetRowCategory

Returns the category row containing the passed row.

```
result = GetRowCategory( theRow as AppleListboxRow ) as AppleListboxRow
```

Parameters

theRow

The AppleListboxRow object whose containing category you wish returned

Notes

If no containing category is found then this method returns nil.

AppleListbox.GetSelectedRow

Returns the currently selected row in the listbox.

```
result = GetSelectedRow( ) as AppleListboxRow
```

Notes

If no row is currently selected in the listbox then *result = nil*.

AppleListbox.LastAddedNode

Returns the last row added to the listbox.

```
result = LastAddedRow( ) as AppleListboxRow
```

AppleListbox.Rebuild

Rebuilds the listbox.

```
Rebuild( )
```

Notes

Rebuilds the entire AppleListbox using the data stored within *AppleListbox.Root*. This should not need to be called by the programmer.

AppleListBox.RemoveChild

Removes the passed row from the listbox.

```
RemoveChild( theChild as AppleListBoxRow )
```

Parameters

theChild

The AppleListBoxRow object to remove from the listbox

AppleListBox.RemoveAllRows

Removes all rows from the listbox.

```
RemoveAllRows( )
```

Notes

Should be used instead of REALbasic's in-built listbox.DeleteAllRows() method.

AppleListBox.SelectRow

Selects the passed row in the listbox.

```
SelectRow( theRow as AppleListBoxRow )
```

Parameters

theRow

The AppleListBoxRow object to select

AppleListBox.Version

Returns the version number of the AppleListBox.

```
result = Version( ) as String
```

AppleListboxRow Properties

AppleListboxRow.Children()

An array containing this row's child rows.

Children() as AppleListboxRow

Notes

The array is zero-based thus Children(0) is the first child.

AppleListboxRow.Editable

Currently not used.

Editable as Boolean

AppleListboxRow.Ejectable

When true, this row will display an eject button in the listbox that responds to mouse clicks and mouse hovers.

Ejectable as Boolean

Notes

Setting Ejectable to True automatically sets AppleListboxRow.HasLinkButton and AppleListboxRow.HasUpdateButton to false.

AppleListboxRow.Expanded

When true, this row is expanded in the tree.

Expanded as Boolean

Notes

Only applies to rows that are folders.

AppleListboxRow.HasLinkButton

When true, this row will display a link button (rightward-facing arrow) in the listbox when this row is selected. The button responds to mouse clicks and mouse hovers.

HasLinkButton as Boolean

Notes

Setting HasLinkButton to True automatically sets AppleListboxRow.Ejectable and AppleListboxRow.HasUpdateButton to false.

AppleListboxRow.HasUpdateButton

When true, this row will display an update button (upward-facing arrow) in the listbox when this row is selected. The button responds to mouse clicks and mouse hovers.

HasUpdateButton as Boolean

Notes

Setting HasUpdateButton to True automatically sets AppleListboxRow.Ejectable and AppleListboxRow.HasLinkButton to false.

AppleListboxRow.Height

The height (in pixels) of this row.

Height as Integer

AppleListboxRow.Icon

This row's picture.

Icon as Picture

Notes

The icon should be 16x16 pixels and may be any image type supported by REALbasic. We recommend using .png images. Remember that these icon pictures require a mask image.

AppleListboxRow.IndentLevel

This row's level of indentation in the tree.

IndentLevel as Integer

Notes

0 = root level (i.e. categories).

This property is essentially for internal use only. Do not alter this property directly - it will have unpredictable results.

AppleListboxRow.Index

This is this row's index in its parent's Children() array.

Index as Integer

Notes

Index is zero-based. E.g. if this row is the second child of its parent row then its index = 1.

AppleListBoxRow.IsCategory

When true, this row represents a category in the listbox.

IsCategory as Boolean

Notes

By setting this value to True, AppleListBoxRow.IsFolder is automatically set to True also. This is because all categories must also be folders.

AppleListBoxRow.IsFolder

When true, this row represents a folder in the listbox.

IsFolder as Boolean

Notes

By setting this value to False, all existing children of this row are permanently removed (as a non-folder row is not allowed to possess any children).

AppleListBoxRow.NewItemsBubbleWidth

The width (in pixels) of this row's new items bubble (if present)

NewItemsBubbleWidth as Integer

Notes

Primarily for internal use only.

If this row does not have any new items to show then NewItemsBubbleWidth = -1.

AppleListBoxRow.NumberOfNewItems

The number of new items that this row has

NumberOfNewItems as Integer

Notes

If AppleListBox.ShowNewItems = True then we display this value to the right of every expanded row in the listbox inside a coloured bubble. If a row is collapsed then the total of each of it's children's NumberOfNewItems values is displayed to the right of the collapsed folder.

AppleListboxRow.Left

The left position (in pixels) of this row.

Left as Integer

Notes

The position is relative to the enclosing window.

AppleListboxRow.Parent

This row's parent row in the tree.

Parent as AppleListboxRow

Notes

For internal use only. Altering this property manually will have unpredictable results.

AppleListboxRow.Tag

A string property. Can be used by developers to add proprietary string data to a row (not displayed).

Tag as String

AppleListboxRow.Text

The text to display in the listbox for this row.

Text as String

Notes

If this row is a category, it is recommended to capitalise Text (Apple human interface guidelines).

AppleListboxRow.Top

The top position (in pixels) of this row

Top as Integer

Notes

Relative to the enclosing window.

AppleListboxRow Methods

AppleListboxRow.AppendChild

Appends the passed AppleListboxRow object as a child of this row.

```
AppendChild( theChild as AppleListboxRow )
```

Parameters

theChild

The AppleListboxRow object to append

AppleListboxRow.Clone

Returns a copy of this row that is a unique object.

```
result = Clone( ) as AppleListboxRow
```

Notes

The clone's children and parent are the **same** objects referenced by the original row.

AppleListboxRow.Constructor

Used to construct a new row.

```
result = Constructor(  
  theText as String,  
  folder as Boolean = False,  
  category as Boolean = False,  
  theIcon as Picture = Nil ) as AppleListboxRow
```

Parameters

theText

The string used to represent this row in the listbox

folder

[Optional]. If True then this row will be a folder in the listbox (i.e. have a disclosure triangle and be able to accept child rows)

category

[Optional]. If True then this row will be treated as a category.

theIcon

[Optional]. The picture used to represent this row's icon

Notes

Category rows should not be assigned an icon.

AppleListboxRow.CountAllChildren

Returns the total number of children that this row has.

```
result = CountAllChildren( ) as Integer
```

Notes

Returns the total number of children beneath this row. This includes rows that are not visible in the listbox.

AppleListboxRow.GetChildWithName

Returns the first instance of a child named *childName*.

```
result = GetChildWithName( childName as String ) as AppleListboxRow
```

Notes

If no child of name *childName* is found then this method returns nil.

AppleListboxRow.GetNextChild

Returns the row after the the passed row.

```
result = GetNextChild( theChild as AppleListboxRow ) as AppleListboxRow
```

Parameters

theChild

The reference AppleListboxRow object

Notes

If there is no child after *theChild* then this method returns Nil.

AppleListboxRow.HasChild

Returns true if the passed row is a child of this row (at any depth).

```
result = HasChild( theChild as AppleListboxRow ) as Boolean
```

Parameters

theChild

The AppleListboxRow object in question

AppleListBoxRow.InsertChildAfter

Inserts *childToInsert* after *indexChild*.

```
InsertChildAfter(  
    indexChild as AppleListBoxRow,  
    childToInsert as AppleListBoxRow )
```

Parameters

indexChild

The reference AppleListBoxRow object

childToInsert

The row to insert after *indexChild*

Notes

For this method to succeed, this row must be a folder (i.e. isFolder = True) and indexChild must be a child of this row.

AppleListBoxRow.InsertChildAtIndex

Inserts *theChild* at the specified *index* in Children()

```
InsertChildAfter(  
    theChild as AppleListBoxRow,  
    index as Integer )
```

Parameters

theChild

The row to insert

index

The index in Children() that *theChild* is to be inserted

Notes

Index is zero-based (i.e. the first child is index = 0).

AppleListboxRow.InsertChildBefore

Inserts *childToInsert* before *indexChild*.

```
InsertChildBefore(  
    indexChild as AppleListboxRow,  
    childToInsert as AppleListboxRow )
```

Parameters

indexChild

The reference AppleListboxRow object

childToInsert

The row to insert before *indexChild*

Notes

For this method to succeed, this row must be a folder (i.e. isFolder = True) and indexChild must be a child of this row.

AppleListboxRow.RemoveChild

Removes the passed AppleListboxRow object **if** it can be found beneath this row in the tree.

```
RemoveChild( theChild as AppleListboxRow )
```

Parameters

theChild

The row to remove

AppleListboxRow.RemoveChildAtIndex

Removes the passed AppleListboxRow object at the passed index.

```
RemoveChildAtIndex( index as Integer )
```

Parameters

index

The index of the child to be removed

Notes

To remove nested children, you must call this method from that nested child's parent row.

AppleListBoxRow.TotalNewItems

Returns the total number of new items contained in this row and it's children.

```
result = TotalNewItems( ) as Integer
```

AppleListBoxRow.Version

Returns the version number of this AppleListBoxRow

```
result = Version( ) as String
```