FGSourceList

Version 1.0.6

(Professional license)

www.madebyfiga.com

Contents

About the class

FGSourceList is a REAL Studio drop-in control that is intended to provide the functionality of the left-hand sidebar (known as a source list) found in many Apple Macintosh applications. It can be used to reproduce the navigation browser in iTunes or Apple Mail or the sidebar in the Finder. It is extremely flexible and can be customised for use in your own applications to provide and attractive navigation class.

It will compile to all three platforms supported by REAL Studio (Mac, Windows and Linux). It is written entirely in native REALbasic code and requires **no** additional plugins to function.

The FGSourceList control comprises two parts - the source list itself (FGSourceList) and a helper class (FGSourceListItem). Also included in your purchase are a collection of PNG images that you can use in your source list. These are not required for the class to function and you are free to provide your own 16x16 pixel images.

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

Adding the class to your projects

To use the FGSourceList in a REALbasic project you must add all of it’s required components to the ‘Project’ tab in the REALbasic IDE. The easiest way to do this is to put the folder entitled ‘FGSourceList Components’ (found in your download) into 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 to your project.

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

If you would like to enable drag re-ordering of FGSourceList items then you must enable the EnableDragReorder properties in the properties inspector for the FGSourceList.

If you have any problems regarding the use of the FGSourceList 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.

The structure of a source list

FGSourceList is based on a tree-like structure composed of various levels of FGSourceListItems.

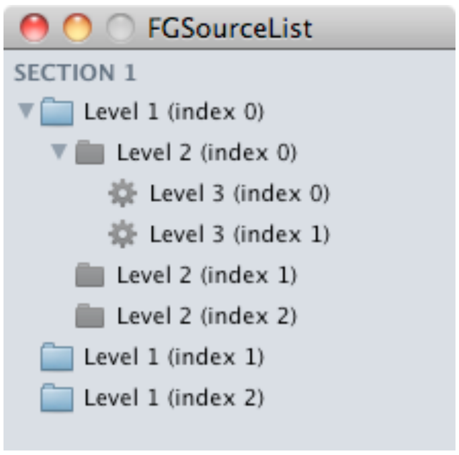
Every source list has a *root* item that is created automatically when you create an instance of FGSourceList. You do not have access to this root item directly, nor should you ever need to.

Source lists are composed of *items* (FGSourceListItems to be precise). An item can either be a *section* or just a plain item. Sections are top-level items and cannot be nested. Section names must be unique within a source list. A section is at level 0.

Sections can contain any number of *children*. All children are known as *items* and are instances of FGSourceListItem.

Source lists have a maximum *depth*. There is a maximum depth of three levels. Why three? Well, Apple's human interface guidelines stipulate that a source list should not contain more than two levels of hierarchy so that's how we've designed FGSourceList.

Here's a picture to better illustrate what we've talked about:


FGSourceList
SECTION 1
Level 1 (index 0)
Level 2 (index 0)
Level 3 (index 0)
Level 3 (index 1)
Level 2 (index 1)
Level 2 (index 2)
Level 1 (index 1)
Level 1 (index 2)

This picture also introduces the concept of an *index*. An item's index describes it's unique position within the tree. An index describes the location of an item within it's parent's *Children* array. This is zero-based.

A level 1 item will have a Level1Index (the location within it's section's Children's array).

A level 2 item will have both a Level1Index and a Level2Index. The Level2Index being the location of this item within it's level 1 parent's Children array and the Level1Index being the location of it's parent in it's section.

Finally, a level 3 item will have a Level1Index, a Level2Index and a Level3Index. Clear as mud?

FGSourceList Events

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

- CellClick
- DoubleClick
- DragReorderRows
- DragRow

Some events are unique to the FGSourceList:

- ClickedButton
- ClickedEjectButton (Mac only - see dedicated entry later in documentation)
- CollapsedSection
- ContextualClickedItem
- DoubleClickedItem
- DraggingItem
- ExpandedSection
- SelectedItem

ClickedButton

If an item has a custom button that is active, then this event is fired when the user clicks on it.

```
ClickedButton( Item as FGSourceListItem )
```

Parameters
Item
A pointer to the FGSourceListItem whose custom button has been clicked in the source list.

Notes
Item.ButtonActive = false then this event will not fire.

ClickedEjectButton

If an item has an eject button, then this event is fired when the user clicks on it.

```
ClickedEjectButton( Item as FGSourceListItem )
```

Parameters
Item
A pointer to the FGSourceListItem whose eject button has been clicked.

Notes
Due to problems with unicode font handling, this event is not implemented on Windows OS. See FGSourceListItem.EjectButton for a workaround.

<u>CollapsedItem</u>

Fires when an item has been collapsed/hidden. Occurs when the user has clicks on a section's show/hide button or an item's disclosure triangle.

```
CollapsedItem( Item as FGSourceListItem )
```

<u>Parameters</u>
Item
A pointer to the FGSourceListItem that has just been collapsed/hidden.

<u>ContextualClickedItem</u>

Fires when the user contextual clicks on an item in the source list.

```
ContextualClickedItem( Item as FGSourceListItem )
```

<u>Parameters</u>
Item
A pointer to the FGSourceListItem contextually-clicked.

Notes
A contextual click is more commonly known as a *control-clicking* on the Mac and *right-clicking* on Windows.

<u>DoubleClickedItem</u>

Fires when the user double-clicks on an item in the source list.

```
DoubleClickedItem( Item as FGSourceListItem )
```

Parameters
Item
A pointer to the FGSourceListItem double-clicked in the source list.

<u>DraggingItem</u>

Fires when the user begins dragging an item in the source list to a new position.

```
DraggingItem( Item as FGSourceListItem )
```

<u>Parameters</u>
Item
A pointer to the FGSourceListItem being dragged by the user.

<u>Notes</u>
This event only fires if FGSourceList.EnableDragReorder is true. This event replaces the built-in Listbox's DragRow event.

ExpandedItem

Fires when an item has been expanded/shown. Occurs when the user has clicks on a section's show/hide button or item's disclosure triangle.

```
ExpandedItem( Item as FGSourceListItem )
```

Parameters
Item
A pointer to the FGSourceListItem that has just been expanded/shown.

SelectedItem

Fires when a user selects an item in the source list.

```
SelectedItem( Item as FGSourceListItem )
```

Parameters
Item
A pointer to the FGSourceListItem that has just been selected in the source list.

FGSourceList Methods

AppendItem

Used to append an FGSourceListItem to an existing section.

```
AppendItem( Item as FGSourceListItem, SectionName as String, Level1Index as Integer = -1,
Level2Index as Integer = -2 )
```

Parameters
Item
The FGSourceListItem to append. *Item* cannot be nil and must have both a name and an icon.

SectionName
The name of the section to append *Item* to. This section must have already been added to the source list.

[optional] *Level1Index*
To append a level 2 item to an existing section, you need to specify the index of the item's parent (a level 1 item) in it's section.

[optional] *Level2Index*
To append a level 3 item to an existing section, you need to specify the index of the item's parent (a level 2 item) and it's parent's parent (a level 1 item) in it's section.

Notes
Indexes are zero-based.
For clarification of the various item levels in a source list, see the page entitled "The structure of a source list".

AppendSection

Used to create and append an empty section to the source list.

```
AppendSection( SectionName as String, Collapsible as Boolean = false, AllowChildReordering
as Boolean = false )
```

Parameters
SectionName
The name of the new section to create. Section names must be unique in a source list. Generally, names should be in uppercase.

[optional] *Collapsible*
If true then this section can be collapsed/hidden by the user. Defaults to false.

[optional] AllowChildReordering
If true **and** FGSourceList.EnableDragReorder = true then this section's child items can be re-ordered by dragging them in the source list.

Clear

This method is used to remove all items from the source list.

```
Clear()
```

Notes
The root FGSourceList is recreated.

CollapseItemAtIndex

This method collapses an item at the specified index.

```
CollapseItemAtIndex( CollapseChildren as Boolean, SectionName as String, Level1Index as
Integer, Level2Index as Integer = -1 )
```

Parameters
CollapseChildren
If true then all of the children of the specified item are also collapsed.

SectionName
The name of the section that this item is within.

Level1Index
The level 1 index of the item.

[optional] *Level2Index*
The level 2 index of the item.

Notes
To collapse a section, use the CollapseSection method.
For clarification of the various item levels in a source list, see the page entitled "The structure of a source list".

CollapseItemAtRow

Collapses the item at the passed row.

```
CollapseItemAtRow( Row as Integer )
```

Parameters
Row
The row number of the item to collapse.

CollapseSection

Collapses the specified section only if it is collapsible.

```
CollapseSection( SectionName as String )
```

Parameters
SectionName
The name of the section to collapse.

Notes
Collapsing a section does not affect the collapsed/expanded state of it's children. This means that when the section is expanded following a collapse, the states of it's children are preserved.

ExpandItemAtIndex

This method expands an item at the specified index.

```
ExpandItemAtIndex( ExpandChildren as Boolean, SectionName as String, Level1Index as
Integer, Level2Index as Integer = -1 )
```

Parameters
ExpandChildren
If true then all of the children of the specified item are also expanded.

SectionName
The name of the section that this item is within.

Level1Index
The level 1 index of the item.

[optional] *Level2Index*
The level 2 index of the item.

Notes
To expand a section, use the ExpandSection method.
For clarification of the various item levels in a source list, see the page entitled "The structure of a source list".

ExpandItemAtRow

Expands the item at the passed row.

```
ExpandItemAtRow( Row as Integer )
```

Parameters
Row
The row number of the item to expand.

ExpandSection

Expands the specified section.

```
ExpandSection( SectionName as String )
```

Parameters
SectionName
The name of the section to expand.

Notes
Expanding a section does not affect the collapsed/expanded state of it's children. This means that when the section is expanded following a collapse, the states of it's children are restored.

GetSection

This method returns the requested section.

```
result = GetSection( SectionName as String ) as FGSourceListItem
```

Parameters
SectionName
The name of the section required.

result
A pointer to the FGSourceListItem that represents the section in question. Returns nil if not found.

InsertEmptySection

Creates an empty section with the specified name and inserts it at the specified index.

```
InsertEmptySection( SectionName as String, Index as Integer, Collapsible as Boolean = False )
```

Parameters
SectionName
The name of the section to be inserted. Section names must be unique within a source list.

Index
The zero-based index of where in the source list to insert this section.

Collapsible
If true then then section will be capable of collapsing/hiding in response to user-interaction. Defaults to false.

InsertItem

This method inserts an FGSourceListItem at the specified index in the specified section.

```
InsertItem( Item as FGSourceListItem, SectionName as String, Level1Index as Integer,
Level2Index as Integer = -1, Level3Index as Integer = -1 )
```

Parameters
Item
The FGSourceListItem to insert.

SectionName
The section name to insert into.

Level1Index
The level 1 index of *item*.

[optional] *Level2Index*
The level 2 index of *item*.

[optional] *Level3Index*
The level 3 index of *item*.

Notes
It is imperative that all of *Item*'s properties are set correctly. No rigorous error checking is made by the method. Please ensure that *Item.Parent* and *Item.Level* are set correctly. You must also ensure the same for all of *Item*'s children.

For clarification of the various item levels in a source list, see the page entitled “The structure of a source list”.

InsertSection

This method takes a user-created section in the form of a FGSourceListItem and inserts it at the specified index.

```
InsertSection( Section as FGSourceListItem, Index as Integer )
```

Parameters
Section
The section to insert. Please note that *Section.Name* must not be the same as any other section in the source list.

Index
The zero-based index within *root* to insert *Section* at.

ItemAtRowNumber

Returns the item at the specified row in the source list.

```
result = ItemAtRowNumber( Row as Integer ) as FGSourceListItem
```

Parameters
Row
The row number in the source list whose item you want. Zero-based. Only visible rows are counted.

result
A pointer to the FGSourceListItem specified.

Rebuild

Re-draws the entire source list using the data stored in the *root* item.

```
Rebuild()
```

Notes
Generally, you don't need to call this method directly. The only exception to this rule is *after* calling the FGSourceList.RemoveItem method.

RemoveItem

Please note, there are two RemoveItem methods that take different parameters but achieve the same goal.

Used to remove the specified item from the source list

```
RemoveItem( Item as FGSourceListItem )
```

Parameters
Item
The item to remove from the source list.

Notes
You need to call FGSourceList.Rebuild after this method to correctly update the source list.

<u>RemoveItem</u>

Used to remove the specified item from the source list given the item’s section name and indexes.

```
RemoveItem( SectionName as String, Level1Index as Integer, Level2Index as Integer = -1,
Level3Index as Integer = -1 )
```

<u>Parameters</u>
SectionName
The name of the section that contains the item to be removed.

Level1Index
The level 1 index of the item to be removed.

[optional] *Level2Index*
The level 2 index of the item to be removed.

[optional] *Level3Index*
The level 2 index of the item to be removed.

<u>Notes</u>
You need to call FGSourceList.Rebuild after this method to correctly update the source list.
For clarification of the various item levels in a source list, see the page entitled “The structure of a source list”.

<u>RemoveSection</u>

Removes the specified section from the source list.

```
RemoveSection( SectionName as String )
```

Parameters
SectionName
The name of the section to be removed. All children of this section will also be removed.

<u>SectionCount</u>

Returns the number of sections in the source list.

```
result = SectionCount() as Integer
```

<u>Parameters</u>
result
The number of sections in the source list.

SectionWithNameExists

```
result = SectionWithNameExists( SectionName as String ) as Boolean
```

Parameters
SectionName
The section name.

result
Is set to true if the source list contains a section named *SectionName*, otherwise it returns false.

FGSourceList Constants

Note that all constants begin with a lowercase ‘k’.

Error constants
If an error occurs within a method of FGSourceList then FGSourceList.ErrorCode will be set to one of the following error constants:

- kErrorCannotRemoveRootItem
- kErrorChildNotFound
- kErrorInternal
- kErrorInvalidIndex
- kErrorInvalidItemName
- kErrorInvalidParentIndex
- kErrorInvalidItemDoesNotExist
- kErrorInvalidItemHasNoChildren
- kErrorInvalidNilItem
- kErrorInvalidNone
- kErrorInvalidSectionDoesNotExist
- kErrorInvalidSectionNameInvalid
- kErrorInvalidSectionNameNotUnique
- kErrorInvalidSectionNotCollapsible

Source list styles

The three presentation styles available to FGSourceList are as follows:

- kStyleFinder
- kStyleMail
- kStyleItunes

FGSourceList.Style is automatically set to kStyleFinder if you do not specify one yourself.

FGSourceList Properties

ErrorCode

```
ErrorCode as Integer
```

Read-only. Returns the error code for the last operation performed by FGSourceList.

ErrorMessage

```
ErrorMessage as String
```

Read-only. Returns an error message for the last operation performed by FGSourceList. If no error has occurred then this value is an empty string.

LastItemIndex

```
LastItemIndex as Integer
```

Read-only. Returns the index of the last item added to the source list within it's parent's Children array.

Style

```
Style as Integer
```

Read-write. Used to set or get the style of the source list. There are subtle differences between the presentation of the source list depending on the style.

Version

```
Version as String
```

Read-only. The version of this class.

FGSourceListItem

FGSourceListItem Methods

CollapseChildren

Collapses all of this item’s children.

```
CollapseChildren()
```

ExpandChildren

```
ExpandChildren()
```

RemoveChild

Removes the passed FGSourceListItem if it can be found beneath this item in the tree.

```
result = RemoveChild( Item as FGSourceListItem ) as Integer
```

Parameters
Item
A pointer to the FGSourceListItem to be removed.

result
The error code generated by this method. It is one of the FGSourceList errors (see the FGSourceList documentation for more details).

FGSourceListItem Properties

AllowChildDragReordering

```
AllowChildDragReordering as Boolean = True
```

Read-write. Used to determine whether the user can drag-reorder this item's children within their local hierarchy. Only functions if this item's section's AllowChildDragReordering property also = true **and** FGSourceList.EnableDragReorder = true. Defaults to true.

BadgeColor

```
BadgeColor as Color = kDefaultBadgeColour
```

Read-write. The colour of this item's badge (if one is to be displayed). Defaults to the standard Apple badge colour.

BadgeCount

```
BadgeCount as Integer = 0
```

Read-write. The number to display in this item's badge. Only relevant if FGSourceListItem.ShowBadge is true. Defaults to 0.

BadgeWidth

```
BadgeWidth as Integer
```

Read-only. The width (in pixels) of this item's badge.

Button

```
Button as Boolean = False
```

Read-write. If true then this item will display a custom button to it's right-hand side. If true then FGSourceListItem.ButtonImage must also be set.

If an image is also supplied for FGSourceListItem.ButtonHoverImage then this second image will be drawn when the user hovers the mouse cursor over the custom button.

ButtonActive

```
ButtonActive as Boolean = True
```

Read-write. If true and FGSourceListItem.Button = true then clicking on the custom button will fire the FGSourceList.ClickedButton event in the source list. If false then clicking on the custom button will have no effect (useful if you just want to display an icon). Defaults to true.

ButtonHoverImage

```
ButtonHoverImage as Picture
```

Read-write. Optional. If FGSourceListItem.Button = true then this image will be drawn when the mouse cursor hovers overs this item's custom button. The image must be 16 pixels in height but (within reason) can be any width.

ButtonImage

```
ButtonImage as Picture
```

Read-write. If FGSourceListItem.Button = true then this must not be nil. The image must be 16 pixels in height but (within reason) can be any width.

Children

```
Children() as FGSourceListItem
```

Read-write. This array stores any children of this item. The array is zero-based.

Collapsible

```
Collapsible as Boolean = False
```

Read-write. If true then this item can be collapsed / hidden. When an item is collapsed, it's name remains visible in the source list but all of it's children are hidden from view. Defaults to false.

EjectButton

```
EjectButton as Boolean
```

Read-write. Mac-only. If true then this item will display an eject button to the far right of it's name. Item badge counts are not displayed if an eject button is visible (custom buttons are).

Due to problems with how Windows supports unicode fonts, this property has no effect on Windows. If you would like to display an eject button on Windows, the simplest way to achieve this is to use a custom button.

Expanded

```
Expanded as Boolean
```

Read-write. Determines whether this item is expanded in the source list or not. Only has an effect if FGSourceListItem.Collapsible = true.

Icon

```
Icon as Picture
```

Read-write. This item’s icon. This is mandatory for all items except sections (which do not display an icon). Icons must be 16x16 pixels.

IndexInParent

```
IndexInParent as Integer
```

Read-only. The index of this item in it’s parent.

Level

```
Level as Integer
```

Read-write. This item’s level in the source list.

Level 0 = A section.
Level 1 = A child of a section.
Level 2 = A child of a level 1 item.
Level 3 = A child of a level 2 item. This is the deepest level possible.

MySection

```
MySection as FGSourceListItem
```

Read-write. This item’s containing section. Returns nil if this item is a section or the root item itself.

Name

```
Name as String
```

Read-write. The text to display in the source list. This must **not** be an empty string.

NumberOfChildren

```
NumberOfChildren as Integer
```

Read-only. The total number of children beneath this item. It includes the children of children.

Parent

```
Parent as FGSourceListItem
```

Read-write. This item’s parent item. Unless this item is root, this value must **not** be nil.

Section

```
Section as Boolean = False
```

Read-write. If this item is a section then this is true. Defaults to false.

ShowBadge

```
ShowBadge as Boolean = True
```

Read-write. By default, items show the total badge count of themselves and their children when they are collapsed. If this is set to true then a badge will never be shown next to this item.

Badges can only be displayed when FGSourceListItem.EjectButton = false.

Tag

```
Tag as Variant
```

This is simply a variant property that can be used to store any form of arbitrary data for this item.

TotalBadgeCount

```
TotalBadgeCount as Integer
```

Read-only. The total badge count for this item **and** all of it's children.

Version

```
Version as String
```

Read-only. The version number of the class.

Release Notes

Version 1.0.6 (16/07/12)

[FIX]	Item badges now render with smooth, rather than ragged edges on Windows
[FIX]	Faster drawing routines thanks to use of #pragma directives
[FIX]	Item reordering now works again in Cocoa (with Real Studio 2012 r1)

Version 1.0.5 (12/01/12)

[FIX]	Removed an erroneous empty event from FGSourceList called ‘Untitled’
[CHANGE]	AutoHideScrollBars property is now editable in the IDE property inspector
[CHANGE]	ScrollBarVertical property is now editable in the IDE property inspector

Version 1.0.4 (23/08/11)

[FIX]	The text within badge numbers is now correctly aligned with Cocoa builds
[CHANGE]	CollapsedSection event has now been replaced by a CollapsedItem event
[CHANGE]	ExpandedSection event has now been replaced by an ExpandedItem event

Version 1.0.3 (18/12/10)

[FIX]	Background colour now drawn correctly when control loses focus.
[FIX]	Clarified drag reordering permissions. Can only reorder within a parent.

Version 1.0.2 (08/11/10)

[FIX]	Background colour now updated when the control’s parent window loses focus.
[CHANGE]	The FGSourceListItem Tag property is now a variant instead of a string.

Version 1.0.1 (23/09/10)

[FIX]	CollapsedSection and ExpandedSection events now fire correctly.

Version 1.0 (21/09/10)

First public release