

Customizing the Help Browser

Pavi Sandhu

■ Introduction

The Help Browser provides convenient access to all of *Mathematica*'s online documentation. You can browse through a hierarchy of topics or search for information on a specific keyword. You can also edit and evaluate the examples or print any topic displayed in the Help Browser, just like in a notebook.

A little-known feature of the Help Browser is that you can customize it. You can add your own reference material to the existing help files, add your own keywords to the Master Index, or set up hyperlinks between different categories of the Help Browser. You can even change the names and number of the buttons at the top of the Help Browser

These types of changes are useful if you are creating an add-on package or a custom application. You can then integrate your documentation into the Help Browser. This article provides detailed information on the structure of the Help Browser's files and the steps needed to modify them.

All information to be displayed in the Help Browser must be stored in the form of notebooks inside the Documentation directory in the *Mathematica* layout. The Documentation directory can contain several subdirectories, each with the name of a specific language. For example, all documentation in English should be placed in the Documentation/English directory.

The Documentation/English directory in turn contains five subdirectories, each corresponding to a specific button in the Help Browser:

- AddOns
- Demos
- GettingStarted
- MainBook
- OtherInformation
- RefGuide
- Tour

In addition to the notebooks containing the help material, each of these directories must contain its own copy of the following two files:

- `BrowserCategories.m`—This is a plain text file that specifies the categories to be displayed in the Browser window.
- `BrowserIndex.nb`—This is a notebook that specifies the entries that should appear in the Master Index.

To add new reference material to the Help Browser:

1. Place the notebooks containing the new material inside the `AddOns` subdirectory of the `Documentation/English` directory.
2. Organize the material into a set of topics and subtopics corresponding to the categories that you want displayed in the Help Browser.
3. Add descriptive cell tags to the reference material to identify the cells that will be displayed under each category.
4. Edit the `BrowserCategories.m` file to create the categories that will appear in the Browser window. You should first save a copy of the existing `BrowserCategories.m` file in the same location under a different name, in case you need to revert to it later.
5. Edit the `BrowserIndex.nb` file to specify the keywords that will appear in the Help Browser's Master Index.

Specific details on each of these steps is provided in the rest of this article.

Note: Each time you modify any of the help files, you should click the `Help ▸ Rebuild Help Index` menu command for your changes to take effect.

■ Setting Up New Browser Categories

□ Introduction

The four columns in the Help Browser display the categories into which the reference material is organized. These categories are arranged in a hierarchy with the first column showing the top-level categories. Clicking a topic either opens more subtopics in the next column or causes reference material about that topic to be displayed in the Browser window.

To make your own material accessible from the Help Browser, you need to define a set of topics that will be displayed in the Help Browser's columns. To do this, you must edit one of the `BrowserCategories.m` files contained in the `Documentation` directory of your *Mathematica* layout.

□ The `BrowserCategory` Command

The following example shows the contents of a simple `BrowserCategories.m` file.

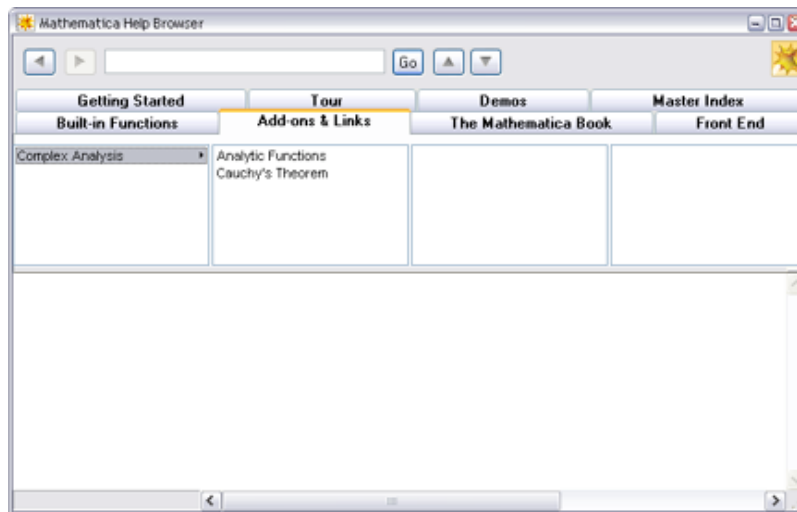
```

BrowserCategory["Add-ons", None,
  {BrowserCategory["Complex Analysis", None,
    {Item["Analytic Functions", "MyHelp.nb"],
     Item["Cauchy's Theorem", "MyHelp.nb"]}]}
}]

```

The contents of a simple `BrowserCategories.m` file.

To investigate the effect of this file, copy it into the `Documentation/English/Add-Ons` directory of your *Mathematica* layout. You should first save a copy of the existing `BrowserCategories.m` file in the same location under a different name so that you can revert to it later. If you then rebuild the Help Browser (using the `Rebuild Help Index` command under the `Help` menu), the categories defined by the new `BrowserCategories.m` file will appear in the Browser window.



Every `BrowserCategories.m` file consists of a single `BrowserCategory` command of the form: `BrowserCategory["category name", name, list]`.

The arguments in a `BrowserCategory` command are as follows:

1. *category name*—This is a string that specifies the category name to be displayed in the Help Browser column.
2. *name*—If set to `None`, this option indicates that the source files are located in the same directory as the `BrowserCategory.m` file being edited. Alternatively, the source files can be kept in a subdirectory one level below the current directory. In this case *name* is set to the name of the subdirectory.
3. *list*—This is a list whose elements can be commands of the following types:
 - `Item["subcategory name", "file name", options]`—This command specifies the name of a terminal subcategory in the Help Browser.

Clicking the subcategory name causes information about it to be displayed in the Help Browser.

- `Item[Delimiter]`—This command inserts a horizontal line in the column where the subcategories are listed. It is useful for separating subcategories.
- `HelpDirectoryListing["directory", arg2, arg3]`—This command creates categories in the Help Browser's columns taken from a different `BrowserCategories.m` file. The location of the file to be used is specified by the three arguments.
- `BrowserCategory[]`—This command is of exactly the same form as the top-level `BrowserCategory` command, except that it is nested one level deeper in the hierarchy, so that its category name will be displayed one column to the right.

Note: Since there are only four columns in the Help Browser, a `BrowserCategory` command can be nested at most four levels deep.

□ The `HelpDirectoryListing` Command

The `HelpDirectoryListing` command is used to create categories in the Help Browser's columns based on a `BrowserCategories.m` file at a different location. It has the form: `HelpDirectoryListing[directory, arg2, arg3]`.

The three arguments *directory*, *arg2*, and *arg3* together specify the location of the `BrowserCategories.m` file to be used. The first argument, *directory*, specifies a directory location using the `FrontEnd`FileName` function. The absolute pathname of the directory must be given. Otherwise, the pathname specified is interpreted to be relative to the `Documentation/English` directory inside the top-level *Mathematica* directory.

If *arg2* is set to `False`, the Browser searches for a `BrowserCategories.m` file at the top level of "*directory*". If it is set to `True`, the Browser searches all subdirectories located one level below *directory*.

If *arg3* is set to `False`, the location of the `BrowserCategories.m` file is completely specified by the first two arguments. If it is set to `True`, the `BrowserCategories.m` file is taken to be in a `Documentation/English` directory located in the directory specified by the first two arguments.

If either of the last two arguments is omitted, they assume their default values. The default value of *arg2* is `True`. The default value of *arg3* is the current value of *arg2*.

These comments are summarized in the following table, which shows the directory that is searched to locate a `BrowserCategories.m` file for different values of *arg1* and *arg2*.

<i>arg2</i>	<i>arg3</i>	Location searched
False	False	<i>directory</i>
False	True	<i>directory</i> / Documentation / English
True	False	<i>directory</i> / *
True	True	<i>directory</i> / * / Documentation / English

Note: Here *directory* / * refers to all subdirectories located one level below *directory*.

Example 1

The `HelpDirectoryListing[]` command is used in the top-level `BrowserCategories.m` file in the Documentation/English directory to define the buttons that appear in the Help Browser window. Here are the contents of this file.

```
BrowserCategory["Help Browser", None, {
  HelpDirectoryListing[{"RefGuide"}, False],
  HelpDirectoryListing[{"AddOns"}, False],
  HelpDirectoryListing[{"MainBook"}, False],
  HelpDirectoryListing[{"OtherInformation"}, False],
  HelpDirectoryListing[{"GettingStarted"}, False],
  HelpDirectoryListing[{"Tour"}, False],
  HelpDirectoryListing[{"Demos"}, False],
  BrowserCategory["Master Index", None, {HelpMasterIndex[]}]
}]
```

Note that in each `HelpDirectoryListing` command here, *arg2* is omitted. This means it has the same value as *arg3* by default.

Example 2

Here is another simple example of a `BrowserCategories.m` file that uses `HelpDirectoryListing` commands. Here it is assumed that each directory, Chapter 2, Chapter 3, and so on, contains a separate `BrowserCategories.m` file.

```
BrowserCategory["Add-ons", None,
{ BrowserCategory["My Book", None, {
: BrowserCategory["Chapter 1", None,
{Item["Section 1.1", "Chapter1.nb"],
Item["Section 1.2", "Chapter1.nb"]}],
Item[Delimiter],
HelpDirectoryListing[{FrontEnd`FileName[{"AddOns",
"Chapter 2"}]}, False],
HelpDirectoryListing[{FrontEnd`FileName[{"AddOns",
"Chapter 3"}]}, False],
HelpDirectoryListing[{FrontEnd`FileName[{"AddOns",
"Chapter 4"}]}, False],
HelpDirectoryListing[{FrontEnd`FileName[{"AddOns",
"Chapter 5"}]}, False]
}]
```

□ The Item Command

The `Item` command has the form: `Item[subcategory name, file name, options]`.

The arguments of the `Item` command specify the target material that is displayed in the Help Browser when the subcategory name associated with that item is clicked.

1. *subcategory name*—This is a string that specifies the subcategory name to be displayed in the Help Browser column.
2. *file name*—This is a string that specifies the name of the notebook from which information pertaining to that subcategory is to be retrieved.
3. *options*—There can be two types of options: `IndexTag` and `CopyTag`:
 - The `IndexTag` serves as a tag to identify the item that is the target of a hyperlink in a notebook. A discussion of this option is deferred to the next section.
 - The `CopyTag` specifies which cells, from the notebook specified by *file name*, appear in the Help Browser when the item is clicked.

The value of `CopyTag` serves as a marker identifying the target cells linked to a specific topic. When an item in the Help Browser is clicked the Browser looks for cells, in the notebook specified by "*file name*", whose cell tags match the value of the `CopyTag` for that item. All such cells are then displayed in the Browser window.

The CopyTag option can be specified in four ways:

1. If `CopyTag` → `None`, the entire notebook specified by *file name* is displayed in the Browser window.
2. If `CopyTag` → "`tag`", the Browser looks for all cells in the notebook specified by *file name* whose cell tag is "`tag`". Only those cells are then displayed in the Browser window.
3. If the `CopyTag` is not specified explicitly, it takes the value of the `IndexTag` option.
4. If both the `CopyTag` and the `IndexTag` are omitted, they assume a default value given by the *subcategory name* specified in the first argument of the `Item` command.

□ An Example with Nested Categories

Here is a more complicated example illustrating how to define subcategories at different levels in the hierarchy. It contains multiple `BrowserCategory` commands, including one that is nested three levels deep. At each level of nesting, the category name associated with the `Item` command shifts one column to the right in the Browser window.

Note: The CopyTag is not specified explicitly for any of the Item commands in this example. It therefore assumes a default value given by the "subcategory name" for each item.

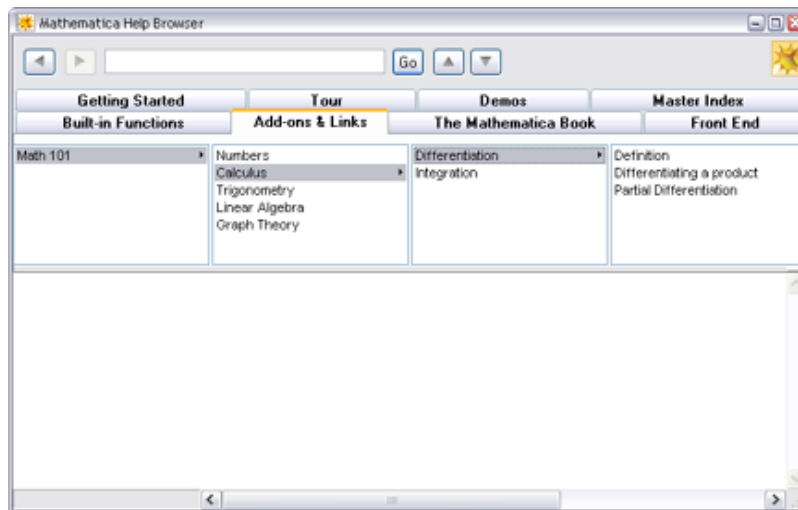
```

BrowserCategory["Add-ons", None,
  BrowserCategory["Math 101", None, {
    {Item["Numbers", "MyMathCourse.nb", IndexTag → "Chapter 1"]},
    {BrowserCategory["Calculus", None,
      {BrowserCategory["Differentiation", None,
        {Item["Definition", "MyMathCourse.nb",
          IndexTag → "Subsection 2.1.1"],
          Item["Differentiating a product", "MyHelp.nb",
            IndexTag → "Subsection 2.1.2"],
          Item["Partial Differentiation", "MyHelp.nb",
            IndexTag → "Subsection 2.1.3"]}}},
      Item["Integration", "MyMathCourse.nb", IndexTag →
        "Section 2.2"]}],
    Item["Trigonometry", "MyMathCourse.nb", IndexTag →
      "Chapter 3"],
    Item["Linear Algebra", "MyMathCourse.nb", IndexTag →
      "Chapter 4"],
    Item["Graph Theory", "MyMathCourse.nb", IndexTag →
      "Chapter 5"]}]
  ]

```

A sample BrowserCategories.m file showing nested commands.

Here are the categories defined by this file as they appear in the Browser window.



You can use the BrowserCategories.m file shown in this example as a template to construct your own Browser categories by replacing the chapters, sections, and

subsections in it with topics and subtopics relevant to your own material. You can add as many extra categories as you wish by adding more `BrowserCategory` or `Item` commands to the list at the level required.

Note: Since there are only four columns in the Help Browser, a `BrowserCategory` command can be nested at most four levels deep.

□ Adding a Sample Document

Here are the steps involved in setting up the Browser categories shown in the first example above.

1. Go to the Documentation/English directory in the top-level *Mathematica* directory and open the AddOns directory.
2. Rename the existing `BrowserCategories.m` file so that you can revert to it later. Then create another file called `BrowserCategories.m` containing the following text.

```
BrowserCategory["Add-ons", None,
{BrowserCategory["Complex Analysis", None,
{Item["Analytic Functions", "MyHelp.nb",
CopyTag→ "SubItem1"],
Item["Cauchy's Theorem", "MyHelp.nb",
CopyTag→ "SubItem2"]}]}]
```

3. Create a new notebook containing the following text.

Here is some reference material about analytic functions.
Here is some reference material about Cauchy's theorem.
4. Assign the cell tag "Complex Analysis" to the first cell and "Cauchy's theorem" to the second cell. To learn how to add a tag to a cell, see *Creating and Using Hyperlinks*.
5. Click the File ▷ SaveAs menu command to bring up the file dialog box.
6. Save this notebook as "MyHelp.nb" in the Documentation/English/Add-Ons directory.
7. Go to the Help menu and select Rebuild Help Index. This command updates the Help Browser so that it reflects your latest changes.

Note: You must add cell tags to any target material that is to be displayed when any item in the Browser window is clicked. The cell tags should match the `CopyTag` for that item, as explained in the discussion of the `Item` command.

You are now ready to view your help information.

1. Click Help ▷ Help Browser.
2. Click the Add-ons & Links button.

3. Click "Complex Analysis" in the leftmost category column and then click "Analytic Function" in the second column.

The first cell of your help notebook is displayed in the Browser window. If you click "Cauchy's theorem", the second cell will be displayed.

■ Adding Entries to the Master Index

□ Creating a New Index Entry

Once you have defined Browser categories for your reference material, you might want to incorporate keywords from it into the Master Index. The Master Index provides an alphabetical listing of all keywords, allowing users to quickly locate information on a specific topic.

To add new keywords to the Master Index:

1. Edit the `BrowserIndex.nb` file to add new entries and links.
2. Edit the `BrowserCategories.m` file to define `IndexTag` for each `Item` command.

We first illustrate this process with an example and then give details about the syntax of the `BrowserIndex.nb` file.

□ The `BrowserIndex.nb` file

The `BrowserIndex.nb` file is a *Mathematica* notebook that contains the list of terms that appear in the Master Index, as well as links to information about each item. This file must use the "HelpBrowser.nb" style sheet, which contains definitions for all the special styles needed. Here is an example of a simple `BrowserIndex.nb` file.

□ My Help Documentation

```
Numbers, 1
Calculus, 2
    derivatives, 2.1
    integrals, 2.2
Algebra, 3, A1.1
```

The contents of a simple `BrowserIndex.nb` file.

Every `BrowserIndex.nb` file consists of a Header followed by a list of entries. The first cell in the file is called the Header.

The Header must have the following properties:

- It must be a cell of style `IndexSection`.
- It must have `MasterIndexHeading` as a cell tag.

This is what the expression for the Header in the sample `BrowserIndex.nb` file looks like. You can see that it satisfies both of the previously listed properties.

```
Cell["My Help Documentation", "IndexSection",
CellTags -> "MasterIndexHeading"]
```

The Header cell is followed by a series of entries. Each entry corresponds to a single topic in the Master Index.

An entry has the following properties:

1. It is a cell of `Index` style.
2. It contains two items:
 - Entry name—This is the text that appears in the Master Index heading. It serves as a label for the information that the link points to.
 - Link—This is a special type of hyperlink that brings up reference material corresponding to the entry name and displays it in the Help Browser.
3. It must have a cell tag that is the same as the entry name.

Here is the cell expression for the first entry in the sample `BrowserIndex.nb` file. You can see that it has all the three properties listed earlier.

```
Cell[TextData[{"My first entry, ", ButtonBox["1",
ButtonData:> "Chapter 1", ButtonStyle -> "AddOnsLink"}]],
"Index", CellTags -> "My first entry"]
```

■ Creating Hyperlinks to the Help Browser

□ Creating a Button Box

The `CopyTag` option to the `Item` command allows you to specify which material should show up in the Help Browser when you click on a topic displayed in one of the columns. However, the `Item` command can also take a second type of option called `IndexTag`. Using the latter option, you can construct special hyperlinks in a notebook that point to reference material in the Help Browser.

When one of these hyperlinks is clicked, the Help Browser searches for the `Item` command in the `BrowserCategories.m` file that contains the corresponding `IndexTag`. The cells associated with that item are then displayed in the Help Browser.

To construct a hyperlink that uses an index tag, you should know how to unformat a cell and edit the expression representing the cell directly. To see the expression for a cell, select the cell and choose `Format > Show Expression`. The expression for each hyperlink consists of a `ButtonBox` command with the following syntax:

```

ButtonBox["Button text", ButtonData:> "IndexTag",
ButtonStyle → "AddOnsLink"]

```

A `ButtonBox` command can have three arguments:

1. "Button text"—This is the text that forms the hyperlink. It usually appears underlined and in a different color.
2. "ButtonData"—This is an option that is used to specify the material referred to by the hyperlink. It is usually set to the `IndexTag` associated with the cell or group of cells that must be displayed in the Help Browser. If this option is omitted, then it assumes a default value given by "Button text".
3. "ButtonStyle"—This option specifies the directory in which the target material of the hyperlink is stored. It can take any of five values, each named after one of the button categories, that is, `GettingStartedLink`, `MainBookLink`, `OtherInformationLink`, `RefGuideLink`, or `AddOnsLink`.

This `ButtonData` option can be set in three different ways:

- i. If `ButtonData → "tag"`, then the Browser searches the `BrowserCategories.m` file for an item with that `IndexTag`. The cells associated with that item (i.e., whose cell tags match the `CopyTag` or `IndexTag` for the corresponding `Item` command) are then displayed in the Browser window.
- ii. If `ButtonData → {"tag1","tag2"}`, then the Browser searches the `BrowserCategories.m` file for an item with an `IndexTag` of "tag1". All cells associated with that item are then displayed in the Browser window. However, the window scrolls down to the first cell with the additional cell tag of "tag2".
- iii. If you do not set the `ButtonData` option explicitly, it assumes a default value given by the "subcategory name" specified in the first argument of the `Item` command.

An Example

Suppose you want to construct a hyperlink that points to information about the `Option Inspector` menu command under the `Format` menu. The cells with this information are located in the a notebook called "FormatMenu.nb" and have the cell tag "OptionsDialog". This notebook is in the `OtherInformation` directory of the `Documentation/English` directory. Here is the item in the `BrowserCategories.m` file that refers to this topic.

```

Item["Option Inspector","FormatMenu.nb", CopyTag →
"OptionsDialog", IndexTag → "Option Inspector"]

```

Hence the expression for the `ButtonBox` command will be the following.

```
ButtonBox["Option Inspector", ButtonData:> "Option Inspector",
ButtonStyle -> "OtherInformationLink"]
```

Note

1. The first argument of the `ButtonBox` command contains the text that will represent the hyperlink.
2. The setting for the `ButtonData` option matches the `IndexTag` for this item.
3. The setting for the `ButtonStyle` option is `OtherInformationLink` because the target material is located in the `OtherInformation` subdirectory.

Here are the steps involved in creating such a hyperlink:

1. Select the cell in which you want to place the hyperlink, for example, a blank text cell.
2. Unformat the expression for the cell by typing `CTRL+SHIFT+E`. (For Macintosh, type `⌘+SHIFT+E`.)
3. Insert the previous `ButtonBox` expression into the cell.
4. Wrap the command `TextData[]` around the `ButtonBox` expression inside the cell.
5. The cell expression now looks like this.

```
Cell[TextData[ButtonBox["Option Inspector", ButtonData
:> "Option Inspector", ButtonStyle -> "Hyperlink"]],
"Text"]
```

6. Format the cell expression by typing `CTRL+SHIFT+E` again.

This completes the construction of the hyperlink. The text "Option Inspector" now appears underlined and colored. Clicking the link will cause the Help Browser to open and display information about this menu command. A similar procedure can be followed for constructing a hyperlink to any other item in the Help Browser.

About the Author

Pavi Sandhu has a Ph.D. in physics from Boston University. He is the author of *The MathML Handbook*, published by Charles River Media in November 2002.

Pavi Sandhu
Senior Technical Writer
Oracle Corporation
Redwood Shores, CA