QOwnNotes Documentation

Release develop

Patrizio Bekerle

Sep 22, 2019

Contents

CHAPTER 1

QOwnNotes Scripting

- QOwnNotes scripts consist basically of JavaScript embedded in QML files.
- Take a look at the example scripts to get started fast.
- If you need access to a certain functionality in QOwnNotes or have questions or ideas please open an issue on the QOwnNotes issue page.
- Since debug output is disabled in the releases of QOwnNotes, so you might want to use `console.warn()` instead of `console.log()` to actually see an output.
  - Additionally you can also use the `script.log()` command to log to the log widget.

1.1 Methods and objects QOwnNotes provides

1.1.1 Starting an external program in the background

Parameters

```
/**
 * QML wrapper to start a detached process
 *
 * @param executablePath the path of the executable
 * @param parameters a list of parameter strings
 * @return true on success, false otherwise
 */
bool startDetachedProcess(QString executablePath, QStringList parameters);
```

Usage in QML

```
script.startDetachedProcess("/path/to/my/program", ["my parameter"]);
```

You may want to take a look at the example custom-actions.qml or execute-command-after-note-update.qml.

```
/**
 * QML wrapper to start a synchronous process
 *
 * @param executablePath the path of the executable
 * @param parameters a list of parameter strings
 * @param data the data that will be written to the process (optional)
 * @return the text that was returned by the process
QByteArray startSynchronousProcess(QString executablePath, QStringList parameters,␣
↪QByteArray data);
```

1.1.2 Starting an external program and wait for the output

Usage in QML

```
var result = script.startSynchronousProcess("/path/to/my/program", ["my parameter"],
↪"data");
```

You may want to take a look at the example encryption-keybase.qml.

1.1.3 Getting the path of the current note folder

Parameters

```
/**
 * QML wrapper to get the current note folder path
 *
 * @return the path of the current note folder
 */
QString currentNoteFolderPath();
```

Usage in QML

```
var path = script.currentNoteFolderPath();
```

You may want to take a look at the example absolute-media-links.qml.

1.1.4 Getting the current note

Parameters

```
/**
 * QML wrapper to get the current note
 *
 * @returns {NoteApi} the the current note object
 */
NoteApi currentNote();
```

Usage in QML

```
var note = script.currentNote();
```

You may want to take a look at the example custom-actions.qml.

1.1.5 Logging to the log widget

Parameters

```
/**
 * QML wrapper to log to the log widget
 *
 * @param text
 */
void log(QString text);
```

Usage in QML

```
script.log("my text");
```

1.1.6 Downloading an url to a string

Parameters

```
/**
 * QML wrapper to download an url and returning it as text
 *
 * @param url
 * @return {QString} the content of the downloaded url
 */
QString downloadUrlToString(QUrl url);
```

Usage in QML

```
var html = script.downloadUrlToString("https://www.qownnotes.org");
```

You may want to take a look at the example insert-headline-with-link-from-github-url.qml.

1.1.7 Downloading an url to the media folder

Parameters

```
/**
 * QML wrapper to download an url to the media folder and returning the media
 * url or the markdown image text of the media relative to the current note
 *
```

(continues on next page)

(continued from previous page)

```
 * @param {QString} url
 * @param {bool} returnUrlOnly if true only the media url will be returned (default␣
↪false)
 * @return {QString} the media markdown or url
 */
QString downloadUrlToMedia(QUrl url, bool returnUrlOnly);
```

Usage in QML

```
var markdown = script.downloadUrlToMedia("http://latex.codecogs.com/gif.latex?\frac{1}
↪{1+sin(x)}");
```

You may want to take a look at the example paste-latex-image.qml.

1.1.8 Inserting a media file into the media folder

Parameters

```
/**
 * QML wrapper to insert a media file into the media folder and returning
 * the media url or the markdown image text of the media  relative to the current note
 *
 * @param {QString} mediaFilePath
 * @param {bool} returnUrlOnly if true only the media url will be returned (default␣
↪false)
 * @return {QString} the media markdown or url
 */
QString ScriptingService::insertMediaFile(QString mediaFilePath,
                                          bool returnUrlOnly);
```

Usage in QML

```
var markdown = script.insertMediaFile("/path/to/your/image.png");
```

You may want to take a look at the example scribble.qml.

1.1.9 Regenerating the note preview

Refreshes the note preview.

Parameters

```
/**
 * Regenerates the note preview
 */
QString ScriptingService::regenerateNotePreview();
```

Usage in QML

```
script.regenerateNotePreview();
```

You may want to take a look at the example scribble.qml.

1.1.10 Registering a custom action

Parameters

```
/**
 * Registers a custom action
 *
 * @param identifier the identifier of the action
 * @param menuText the text shown in the menu
 * @param buttonText the text shown in the button
 *                   (no button will be viewed if empty)
 * @param icon the icon file path or the name of a freedesktop theme icon
 *             you will find a list of icons here:
 *             https://specifications.freedesktop.org/icon-naming-spec/icon-naming-
↪spec-latest.html
 * @param useInNoteEditContextMenu if true use the action in the note edit
 *                                 context menu (default: false)
 * @param hideButtonInToolbar if true the button will not be shown in the
 *                            custom action toolbar (default: false)
 * @param useInNoteListContextMenu if true use the action in the note list
 *                                 context menu (default: false)
 */
void ScriptingService::registerCustomAction(QString identifier,
                                            QString menuText,
                                            QString buttonText,
                                            QString icon,
                                            bool useInNoteEditContextMenu,
                                            bool hideButtonInToolbar,
                                            bool useInNoteListContextMenu);
```

Usage in QML

```
// add a custom action without a button
script.registerCustomAction("mycustomaction1", "Menu text");

// add a custom action with a button
script.registerCustomAction("mycustomaction1", "Menu text", "Button text");

// add a custom action with a button and freedesktop theme icon
script.registerCustomAction("mycustomaction1", "Menu text", "Button text", "task-new
↪");

// add a custom action with a button and an icon from a file
script.registerCustomAction("mycustomaction1", "Menu text", "Button text", "/usr/
↪share/icons/breeze/actions/24/view-calendar-tasks.svg");
```

You may then want to use the identifier with function `customActionInvoked` in a script like custom-actions.qml.

1.1.11 Registering a label

Parameters

```
/**
 * Registers a label to write to
 *
 * @param identifier the identifier of the label
 * @param text the text shown in the label (optional)
 */
void ScriptingService::registerLabel(QString identifier, QString text);
```

Usage in QML

```
script.registerLabel("html-label", "<strong>Strong</strong> HTML text<br />with three␣
↪lines<br />and a <a href='https://www.qownnotes.org'>link to a website</a>.");

script.registerLabel("long-label", "an other very long text, an other very long text,␣
↪an other very long text, an other very long text, an other very long text, an other␣
↪very long text, an other very long text, an other very long text, an other very␣
↪long text, an other very long text, an other very long text that will wrap");

script.registerLabel("counter-label");
```

The labels will be visible in the scripting dock widget.

You can use both plain text or html in the labels. The text will be selectable and links can be clicked.

You may then want to take a look at the example script scripting-label-demo.qml.

1.1.12 Setting the text of a registered label

Parameters

```
/**
 * Sets the text of a registered label
 *
 * @param identifier the identifier of the label
 * @param text the text shown in the label
 */
void ScriptingService::setLabelText(QString identifier, QString text);
```

Usage in QML

```
script.setLabelText("counter-label", "counter text");
```

You can use both plain text or html in the labels. The text will be selectable and links can be clicked.

You may then want to take a look at the example script scripting-label-demo.qml.

1.1.13 Creating a new note

Parameters

```
/**
 * Creates a new note
 *
 * @param text the note text
 */
void ScriptingService::createNote(QString text);
```

Usage in QML

```
script.createNote("My note headline\n===\n\nMy text");
```

You may want to take a look at the example custom-actions.qml.

1.1.14 Accessing the clipboard

Parameters

```
/**
 * Returns the content of the clipboard as text or html
 *
 * @param asHtml returns the clipboard content as html instead of text
 */
QString ScriptingService::clipboard(bool asHtml);
```

Usage in QML

```
var clipboardText = script.clipboard();
var clipboardHtml = script.clipboard(true);
```

You may want to take a look at the example custom-actions.qml.

1.1.15 Write text to the note text edit

Parameters

```
/**
 * Writes text to the current cursor position in the note text edit
 *
 * @param text
 */
void ScriptingService::noteTextEditWrite(QString text);
```

Usage in QML

```
// write text to the note text edit
script.noteTextEditWrite("My custom text");
```

You might want to look at the custom action `transformTextRot13` in the example custom-actions.qml.

You can use this together with `noteTextEditSelectAll` to overwrite the whole text of the current note.

1.1.16 Read the selected text in the note text edit

Parameters

```
/**
 * Reads the selected text in the note text edit
 *
 * @return
 */
QString ScriptingService::noteTextEditSelectedText();
```

Usage in QML

```
// read the selected text from the note text edit
var text = script.noteTextEditSelectedText();
```

You might want to look at the custom action `transformTextRot13` in the example custom-actions.qml.

1.1.17 Select all text in the note text edit

Parameters

```
/**
 * Selects all text in the note text edit
 */
void ScriptingService::noteTextEditSelectAll();
```

Usage in QML

```
script.noteTextEditSelectAll();
```

You can use this together with `noteTextEditWrite` to overwrite the whole text of the current note.

1.1.18 Select the current line in the note text edit

Parameters

```
/**
 * Selects the current line in the note text edit
 */
void ScriptingService::noteTextEditSelectCurrentLine();
```

Usage in QML

```
script.noteTextEditSelectCurrentLine();
```

1.1.19 Set the currently selected text in the note text edit

Parameters

```
/**
 * Sets the currently selected text in the note text edit
 *
 * @param start
 * @param end
 */
void ScriptingService::noteTextEditSetSelection(int start, int end);
```

Usage in QML

```
// expands the current selection by one character
script.noteTextEditSetSelection(
    script.noteTextEditSelectionStart() - 1,
    script.noteTextEditSelectionEnd() + 1);
```

1.1.20 Get the start position of the current selection in the note text edit

Parameters

```
/**
 * Returns the start position of the current selection in the note text edit
 */
int ScriptingService::noteTextEditSelectionStart();
```

Usage in QML

```
script.log(script.noteTextEditSelectionStart());
```

1.1.21 Get the end position of the current selection in the note text edit

Parameters

```
/**
 * Returns the end position of the current selection in the note text edit
 */
int ScriptingService::noteTextEditSelectionEnd();
```

Usage in QML

```
script.log(script.noteTextEditSelectionEnd());
```

1.1.22 Read the current word from the note text edit

Parameters

```
/**
 * Reads the current word in the note text edit
 *
 * @param withPreviousCharacters also get more characters at the beginning
 *                               to get characters like "@" that are not
 *                               word-characters
 * @return
 */
QString ScriptingService::noteTextEditCurrentWord(bool withPreviousCharacters);
```

Usage in QML

```
// read the current word in the note text edit
var text = script.noteTextEditCurrentWord();
```

You may want to take a look at the example autocompletion.qml.

1.1.23 Check whether platform is Linux, OS X or Windows

Parameters

```
bool ScriptingService::platformIsLinux();
bool ScriptingService::platformIsOSX();
bool ScriptingService::platformIsWindows();
```

Usage in QML

```
if (script.platformIsLinux()) {
    // only will be executed if under Linux
}
```

1.1.24 Tag the current note

Parameters

```
/**
 * Tags the current note with a tag named tagName
 *
 * @param tagName
 */
void ScriptingService::tagCurrentNote(QString tagName);
```

Usage in QML

```
// add a "favorite" tag to the current note
script.tagCurrentNote("favorite");
```

You might want to look at the custom action `favoriteNote` in the example favorite-note.qml.

1.1.25 Search for tags by name

Parameters

```
/**
 * Fetches all tags by doing a substring search on the name field
 *
 * @param name {QString} name to search for
 * @return {QStringList} list of tag names
 */
QStringList ScriptingService::searchTagsByName(QString name);
```

Usage in QML

```
// searches for all tags with the word game in it
var tags = script.searchTagsByName("game");
```

You may want to take a look at the example autocompletion.qml.

1.1.26 Search for notes by note text

Parameters

```
/**
 * Returns a list of note ids of all notes with a certain text in the note text
 *
 * Unfortunately there is no easy way to use a QList<NoteApi*> in QML, so we
 * can only transfer the note ids
 *
 * @return {QList<int>} list of note ids
 */
QList<int> ScriptingService::fetchNoteIdsByNoteTextPart(QString text);
```

Usage in QML

```
var noteIds = script.fetchNoteIdsByNoteTextPart("mytext");

noteIds.forEach(function (noteId){
    var note = script.fetchNoteById(noteId);

    // do something with the note
});
```

You may want to take a look at the example unique-note-id.qml.

1.1.27 Add a custom stylesheet

Parameters

```
/**
 * Adds a custom stylesheet to the application
 *
 * @param stylesheet
 */
void ScriptingService::addStyleSheet(QString stylesheet);
```

Usage in QML

```
// make the text in the note list bigger
script.addStyleSheet("QTreeWidget#noteTreeWidget {font-size: 30px;}");
```

You may want to take a look at the example custom-stylesheet.qml.

You can get the object names from the `*.ui` files, for example mainwindow.ui.

Take a look at Style Sheet Reference for a reference of what styles are available.

If you want to inject styles into html preview to alter the way notes are previewed please look at *notetomarkdownhtml-hook*.

1.1.28 Reloading the scripting engine

Parameters

```
/**
 * Reloads the scripting engine
 */
void ScriptingService::reloadScriptingEngine();
```

Usage in QML

```
// reload the scripting engine
script.reloadScriptingEngine();
```

1.1.29 Fetching a note by its file name

Parameters

```
/**
 * Fetches a note by its file name
 *
 * @param fileName string the file name of the note (mandatory)
 * @param noteSubFolderId integer id of the note subfolder
 * @return NoteApi*
 */
NoteApi* ScriptingService::fetchNoteByFileName(QString fileName,
                                               int noteSubFolderId);
```

Usage in QML

```
// fetch note by file name
script.fetchNoteByFileName("my note.md");
```

1.1.30 Fetching a note by its id

Parameters

```
/**
 * Fetches a note by its id
 *
 * @param id int the id of the note
 * @return NoteApi*
 */
NoteApi* ScriptingService::fetchNoteById(int id);
```

Usage in QML

```
// fetch note by id
script.fetchNoteById(243);
```

You may want to take a look at the example export-notes-as-one-html.qml.

1.1.31 Checking if a note exists by its file name

Parameters

```
/**
 * Checks if a note file exists by its file name
 *
 * @param fileName string the file name of the note (mandatory)
 * @param ignoreNoteId integer id of a note to ignore in the check
 * @param noteSubFolderId integer id of the note subfolder
 * @return bool
```

(continues on next page)

(continued from previous page)

```
 */
bool ScriptingService::noteExistsByFileName(QString fileName,
                                            int ignoreNoteId,
                                            int noteSubFolderId);
```

Usage in QML

```
// check if note exists, but ignore the id of "note"
script.noteExistsByFileName("my note.md", note.id);
```

You may want to take a look at the example use-tag-names-in-filename.qml.

1.1.32 Copying text into the clipboard

Parameters

```
/**
 * Copies text into the clipboard as plain text or html mime data
 *
 * @param text string text to put into the clipboard
 * @param asHtml bool if true the text will be set as html mime data
 */
void ScriptingService::setClipboardText(QString text, bool asHtml);
```

Usage in QML

```
// copy text to the clipboard
script.setClipboardText("text to copy");
```

You may want to take a look at the example selected-markdown-to-bbcode.qml.

1.1.33 Jumping to a note

Parameters

```
/**
 * Sets the current note if the note is visible in the note list
 *
 * @param note NoteApi note to jump to
 */
void ScriptingService::setCurrentNote(NoteApi *note);
```

Usage in QML

```
// jump to the note
script.setCurrentNote(note);
```

You may want to take a look at the example journal-entry.qml.

1.1.34 Jumping to a note subfolder

Parameters

```
/**
 * Jumps to a note subfolder
 *
 * @param noteSubFolderPath {QString} path of the subfolder, relative to the note␣
↪folder
 * @param separator {QString} separator between parts of the path, default "/"
 * @return true if jump was successful
 */
bool ScriptingService::jumpToNoteSubFolder(const QString &noteSubFolderPath,
                                           QString separator);
```

Usage in QML

```
// jump to the note subfolder "a sub folder"
script.jumpToNoteSubFolder("a sub folder");

// jump to the note subfolder "sub" inside of "a sub folder"
script.jumpToNoteSubFolder("a sub folder/sub");
```

1.1.35 Showing an information message box

Parameters

```
/**
 * Shows an information message box
 *
 * @param text
 * @param title (optional)
 */
void ScriptingService::informationMessageBox(QString text, QString title);
```

Usage in QML

```
// show a information message box
script.informationMessageBox("The text I want to show", "Some optional title");
```

1.1.36 Showing a question message box

Parameters

```
/**
 * Shows a question message box
 *
 * For information about buttons see:
```

(continues on next page)

(continued from previous page)

```
 * https://doc.qt.io/qt-5/qmessagebox.html#StandardButton-enum
 *
 * @param text
 * @param title (optional)
 * @param buttons buttons that should be shown (optional)
 * @param defaultButton default button that will be selected (optional)
 * @return id of pressed button
 */
int ScriptingService::questionMessageBox(
        QString text, QString title, int buttons, int defaultButton);
```

Usage in QML

```
// show a question message box with an apply and a help button
// see: https://doc.qt.io/qt-5/qmessagebox.html#StandardButton-enum
var result = script.questionMessageBox(
    "The text I want to show", "Some optional title", 0x01000000|0x02000000,␣
↪0x02000000);
script.log(result);
```

For information about buttons see StandardButton.

You may also want to take a look at the example input-dialogs.qml.

1.1.37 Showing an open file dialog

Properties

```
/**
 * Shows an open file dialog
 *
 * @param caption (optional)
 * @param dir (optional)
 * @param filter (optional)
 * @return QString
 */
QString ScriptingService::getOpenFileName(QString caption, QString dir,
                                           QString filter);
```

Usage in QML

```
// show an open file dialog
var fileName = script.getOpenFileName("Please select an image", "/home/user/images",
↪"Images (*.png *.xpm *.jpg)");
```

1.1.38 Showing a save file dialog

Properties

```
/**
 * Shows a save file dialog
 *
 * @param caption (optional)
 * @param dir (optional)
 * @param filter (optional)
 * @return QString
 */
QString ScriptingService::getSaveFileName(QString caption, QString dir,
                                           QString filter);
```

Usage in QML

```
// show a save file dialog
var fileName = script.getSaveFileName("Please select HTML file to save", "output.html
↪", "HTML (*.html)");
```

You may want to take a look at the example export-notes-as-one-html.qml.

1.1.39 Registering script settings variables

You need to define properties in your script and register them in an further property named `settingsVariables`.

The user can then set these properties in the script settings.

```
// you have to define your registered variables so you can access them later
property string myString;
property bool myBoolean;
property string myText;
property int myInt;
property string myFile;
property string mySelection;

// register your settings variables so the user can set them in the script settings
// use this property if you don't need
//
// unfortunately there is no QVariantHash in Qt, we only can use
// QVariantMap (that has no arbitrary ordering) or QVariantList (which at
// least can be ordered arbitrarily)
property variant settingsVariables: [
    {
        "identifier": "myString",
        "name": "I am a line edit",
        "description": "Please enter a valid string:",
        "type": "string",
        "default": "My default value",
    },
    {
        "identifier": "myBoolean",
        "name": "I am a checkbox",
        "description": "Some description",
        "text": "Check this checkbox",
        "type": "boolean",
```

(continues on next page)

(continued from previous page)

```
        "default": true,
    },
    {
        "identifier": "myText",
        "name": "I am textbox",
        "description": "Please enter your text:",
        "type": "text",
        "default": "This can be a really long text\nwith multiple lines.",
    },
    {
        "identifier": "myInt",
        "name": "I am a number selector",
        "description": "Please enter a number:",
        "type": "integer",
        "default": 42,
    },
    {
        "identifier": "myFile",
        "name": "I am a file selector",
        "description": "Please select the file:",
        "type": "file",
        "default": "pandoc",
    },
    {
        "identifier": "mySelection",
        "name": "I am an item selector",
        "description": "Please select an item:",
        "type": "selection",
        "default": "option2",
        "items": {"option1": "Text for option 1", "option2": "Text for option 2",
↪"option3": "Text for option 3"},
    }
];
```

In addition you can override the `settingsVariables` with a special function `registerSettingsVariables()` like this:

```
/**
 * Registers the settings variables again
 *
 * Use this method if you want to use code to override your variables, like setting
 * default values depended on the operating system.
 */
function registerSettingsVariables() {
    if (script.platformIsWindows()) {
        // override the myFile default value
        settingsVariables[3].default = "pandoc.exe"
    }
}
```

You may also want to take a look at the example variables.qml.

1.1.40 Storing and loading persistent variables

Properties

```
/**
 * Stores a persistent variable
 * These variables are accessible globally over all scripts
 * Please use a meaningful prefix in your key like "PersistentVariablesTest/myVar"
 *
 * @param key {QString}
 * @param value {QVariant}
 */
void ScriptingService::setPersistentVariable(const QString &key,
                                             const QVariant &value);

/**
 * Loads a persistent variable
 * These variables are accessible globally over all scripts
 *
 * @param key {QString}
 * @param defaultValue {QVariant} return value if the setting doesn't exist (optional)
 * @return
 */
QVariant ScriptingService::getPersistentVariable(const QString &key,
                                                 const QVariant &defaultValue);
```

Usage in QML

```
// store persistent variable
script.setPersistentVariable("PersistentVariablesTest/myVar", result);

// load and log persistent variable
script.log(script.getPersistentVariable("PersistentVariablesTest/myVar", "nothing␣
↪here yet"));
```

Please make sure to use a meaningful prefix in your key like `PersistentVariablesTest/myVar` because the variables are accessible from all scripts.

You may also want to take a look at the example persistent-variables.qml.

1.1.41 Loading application settings variables

Properties

```
/**
 * Loads an application settings variable
 *
 * @param key {QString}
 * @param defaultValue {QVariant} return value if the setting doesn't exist (optional)
 * @return
 */
QVariant ScriptingService::getApplicationSettingsVariable(const QString &key,
                                                          const QVariant &
↪defaultValue);
```

Usage in QML

```
// load and log an application settings variable
script.log(script.getApplicationSettingsVariable("gitExecutablePath"));
```

Keep in mind that settings actually can be empty, you have to take care about that yourself. `defaultValue` is only used if the setting doesn't exist at all.

1.1.42 Reading the path to the directory of your script

If you need to get the path to the directory where your script is placed to for example load other files you have to register a `property string scriptDirPath;`. This property will be set with the path to the script's directory.

Example

```
import QtQml 2.0
import QOwnNotesTypes 1.0

Script {
    // the path to the script's directory will be set here
    property string scriptDirPath;

    function init() {
        script.log(scriptDirPath);
    }
}
```

1.1.43 Converting path separators to native ones

Properties

```
/**
 * Returns path with the '/' separators converted to separators that are
 * appropriate for the underlying operating system.
 *
 * On Windows, toNativeDirSeparators("c:/winnt/system32") returns
 * "c:\winnt\system32".
 *
 * @param path
 * @return
 */
QString ScriptingService::toNativeDirSeparators(QString path);
```

Usage in QML

```
// will return "c:\winnt\system32" on Windows
script.log(script.toNativeDirSeparators("c:/winnt/system32"));
```

1.1.44 Converting path separators from native ones

Properties

```
/**
 * Returns path using '/' as file separator.
 * On Windows, for instance, fromNativeDirSeparators("c:\\winnt\\system32")
 * returns "c:/winnt/system32".
 *
 * @param path
 * @return
 */
QString ScriptingService::fromNativeDirSeparators(QString path);
```

Usage in QML

```
// will return "c:/winnt/system32" on Windows
script.log(script.fromNativeDirSeparators("c:\\winnt\\system32"));
```

1.1.45 Getting the native directory separator

Properties

```
/**
 * Returns the native directory separator "/" or "\" on Windows
 *
 * @return
 */
QString ScriptingService::dirSeparator();
```

Usage in QML

```
// will return "\" on Windows
script.log(script.dirSeparator());
```

1.1.46 Getting a list of the paths of all selected notes

Properties

```
/**
 * Returns a list of the paths of all selected notes
 *
 * @return {QStringList} list of selected note paths
 */
QStringList ScriptingService::selectedNotesPaths();
```

Usage in QML

```
// returns a list of the paths of all selected notes
script.log(script.selectedNotesPaths());
```

You may want to take a look at the example external-note-diff.qml.

1.1.47 Getting a list of the ids of all selected notes

Properties

```
/**
 * Returns a list of the ids of all selected notes
 *
 * @return {QList<int>} list of selected note ids
 */
QList<int> ScriptingService::selectedNotesIds();
```

Usage in QML

```
// returns a list of the ids of all selected notes
script.log(script.selectedNotesIds());
```

You may want to take a look at the example export-notes-as-one-html.qml.

1.1.48 Triggering a menu action

Properties

```
/**
 * Triggers a menu action
 *
 * @param objectName {QString} object name of the action to trigger
 * @param checked {QString} only trigger the action if checked-state is
 *                          different than this parameter (optional, can be 0 or 1)
 */
void ScriptingService::triggerMenuAction(QString objectName, QString checked);
```

Usage in QML

```
// toggle the read-only mode
script.triggerMenuAction("actionAllow_note_editing");

// disable the read-only mode
script.triggerMenuAction("actionAllow_note_editing", 1);
```

You may want to take a look at the example disable-readonly-mode.qml.

You can get the object names of the menu action from mainwindow.ui.

1.1.49 Opening an input dialog with a select box

Properties

```
/**
 * Opens an input dialog with a select box
 *
 * @param title {QString} title of the dialog
 * @param label {QString} label text of the dialog
 * @param items {QStringList} list of items to select
 * @param current {int} index of the item that should be selected (default: 0)
 * @param editable {bool} if true the text in the dialog can be edited (default:
↪false)
 * @return {QString} text of the selected item
 */
QString ScriptingService::inputDialogGetItem(
        const QString &title, const QString &label, const QStringList &items,
        int current, bool editable);
```

Usage in QML

```
var result = script.inputDialogGetItem(
    "combo box", "Please select an item", ["Item 1", "Item 2", "Item 3"]);
script.log(result);
```

You may want to take a look at the example input-dialogs.qml.

1.1.50 Opening an input dialog with a line edit

Properties

```
/**
 * Opens an input dialog with a line edit
 *
 * @param title {QString} title of the dialog
 * @param label {QString} label text of the dialog
 * @param text {QString} text in the dialog (optional)
 * @return
 */
QString ScriptingService::inputDialogGetText(
        const QString &title, const QString &label, const QString &text);
```

Usage in QML

```
var result = script.inputDialogGetText(
    "line edit", "Please enter a name", "current text");
script.log(result);
```

1.1.51 Writing text to a file

Properties

```
/**
 * Writes a text to a file
 *
 * @param filePath
 * @param data
 * @return
 */
bool ScriptingService::writeToFile(const QString &filePath, const QString &data);
```

Usage in QML

```
var result = script.writeToFile(filePath, html);;
script.log(result);
```

You may want to take a look at the example export-notes-as-one-html.qml.

1.1.52 Working with websockets

You can remote control QOwnNotes by using `WebSocketServer`.

Please take a look at the example websocket-server.qml. You can test the socket server by connecting to it on Websocket test.

You can also listen to sockets with `WebSocket`. Please take look at the example websocket-client.qml.

Keep in mind that you need to have Qt's QML `websocket` library installed to use this. For example under Ubuntu Linux you can install `qml-module-qtwebsockets`.

1.2 Hooks

1.2.1 onNoteStored

```
/**
 * This function is called when a note gets stored to disk
 * You cannot modify stored notes, that would be a mess since
 * you are most likely editing them by hand at the same time
 *
 * @param {NoteApi} note - the note object of the stored note
 */
function onNoteStored(note);
```

You may want to take a look at the example on-note-opened.qml.

1.2.2 noteOpenedHook

```
/**
 * This function is called after a note was opened
 *
```

(continues on next page)

(continued from previous page)

```
 * @param {NoteApi} note - the note object that was opened
 */
function noteOpenedHook(note);
```

You may want to take a look at the example on-note-opened.qml.

1.2.3 noteDoubleClickedHook

```
/**
 * This function is called after a note was double clicked
 *
 * @param {NoteApi} note - the note object that was clicked
 */
function noteDoubleClickedHook(note);
```

You may want to take a look at the example external-note-open.qml.

1.2.4 insertMediaHook

```
/**
 * This function is called when media file is inserted into the note
 * If this function is defined in multiple scripts, then the first script that␣
↪returns a non-empty string wins
 *
 * @param fileName string the file path of the source media file before it was copied␣
↪to the media folder
 * @param mediaMarkdownText string the markdown text of the media file, e.g. ![my-
↪image](file://media/505671508.jpg)
 * @return string the new markdown text of the media file
 */
function insertMediaHook(fileName, mediaMarkdownText);
```

You may want to take a look at the example example.qml.

1.2.5 insertingFromMimeDataHook

```
/**
 * This function is called when html or a media file is pasted to a note with `Ctrl +␣
↪Shift + V`
 *
 * @param text text of the QMimeData object
 * @param html html of the QMimeData object
 * @returns the string that should be inserted instead of the text from the QMimeData␣
↪object
 */
function insertingFromMimeDataHook(text, html);
```

You may want to take a look at the example example.qml, insert-headline-with-link-from-github-url.qml or note-text-from-5pm-mail.qml.

1.2.6 handleNoteTextFileNameHook

```
/**
 * This function is called when a note gets stored to disk if
 * "Allow note file name to be different from headline" is enabled
 * in the settings
 *
 * It allows you to modify the name of the note file
 * Keep in mind that you have to care about duplicate names yourself!
 *
 * Return an empty string if the file name of the note should
 * not be modified
 *
 * @param {NoteApi} note - the note object of the stored note
 * @return {string} the file name of the note
 */
function handleNoteTextFileNameHook(note);
```

You may want to take a look at the example example.qml or use-tag-names-in-filename.qml.

1.2.7 handleNoteNameHook

```
/**
 * This function is called when the note name is determined for a note
 *
 * It allows you to modify the name of the note that is viewed
 *
 * Return an empty string if the name of the note should not be modified
 *
 * @param {NoteApi} note - the note object of the stored note
 * @return {string} the name of the note
 */
function handleNoteNameHook(note);
```

You may want to take a look at the example example.qml.

It may not be a good idea to use this hook if the setting to use the file name as note name is active.

1.2.8 handleNewNoteHeadlineHook

```
/**
 * This function is called before a note note is created
 *
 * It allows you to modify the headline of the note before it is created
 * Note that you have to take care about a unique note name, otherwise
 * the new note will not be created, it will just be found in the note list
 *
 * You can use this method for creating note templates
 *
 * @param headline text that would be used to create the headline
 * @return {string} the headline of the note
 */
function handleNewNoteHeadlineHook(headline);
```

You may want to take a look at the example custom-new-note-headline.qml.

1.2.9 noteToMarkdownHtmlHook

```
/**
 * This function is called when the markdown html of a note is generated
 *
 * It allows you to modify this html
 * This is for example called before by the note preview
 *
 * The method can be used in multiple scripts to modify the html of the preview
 *
 * @param {NoteApi} note - the note object
 * @param {string} html - the html that is about to being rendered
 * @return {string} the modified html or an empty string if nothing should be modified
 */
function noteToMarkdownHtmlHook(note, html);
```

You may want to take a look at the example example.qml or preview-styling.qml.

Please refer to the Supported HTML Subset documentation for a list of all supported css styles.

1.2.10 encryptionHook

```
/**
 * This function is called when text has to be encrypted or decrypted
 *
 * @param text string the text to encrypt or decrypt
 * @param password string the password
 * @param decrypt bool if false encryption is demanded, if true decryption is demanded
 * @return the encrypted decrypted text
 */
function encryptionHook(text, password, decrypt);
```

You may want to take a look at the example encryption-keybase.qml, encryption-pgp.qml or encryption-rot13.qml.

1.2.11 noteTaggingHook

You can implement your own note tagging mechanism for example with special text in your note like `@tag1`, `@tag2`, `@tag3`.

```
/**
 * Handles note tagging for a note
 *
 * This function is called when tags are added to, removed from or renamed in
 * a note or the tags of a note should be listed
 *
 * @param note
 * @param action can be "add", "remove", "rename" or "list"
 * @param tagName tag name to be added, removed or renamed
 * @param newTagName tag name to be renamed to if action = "rename"
 * @return string or string-list (if action = "list")
 */
function noteTaggingHook(note, action, tagName, newTagName);
```

- as soon as a script is activated that implements the new function `noteTaggingHook` note tagging will be handled by that function

- following features should work via the QOwnNotes user interface
- initially importing tags like `@tag` from your notes and overwriting your current tag assignment
  - you will not loose your tags tree, just the former assignment to notes
  - you can still move tags into other tags
  - if more than one tag has the same name in your tag tree the first hit will be assigned
- adding a tag to a note will add the tag to the note text
- removing a tag from a note will remove the tag from the note text
- removing of tags in the tag list will remove those tags from your notes
- renaming of tags in the tag list will rename those tags in your notes
- bulk tagging of notes in the note list will add those tags to your notes
- bulk removing of tags from notes in the note list will remove those tags from your notes

You may want to take a look at the example note-tagging.qml to implement your own tagging mechanism.

1.2.12 autocompletionHook

You can return a list of strings to be added to the autocompletion list when the autocompletion is invoked.

```
/**
 * Calls the autocompletionHook function for all script components
 * This function is called when autocompletion is invoked in a note
 *
 * @return QStringList of text for the autocomplete list
 */
function callAutocompletionHook();
```

You may want to take a look at the example autocompletion.qml.

1.3 Exposed classes

1.3.1 Note

```
class NoteApi {
    Q_PROPERTY(int id)
    Q_PROPERTY(QString name)
    Q_PROPERTY(QString fileName)
    Q_PROPERTY(QString fullNoteFilePath)
    Q_PROPERTY(int noteSubFolderId)
    Q_PROPERTY(QString noteText)
    Q_PROPERTY(QString decryptedNoteText)
    Q_PROPERTY(bool hasDirtyData)
    Q_PROPERTY(QQmlListProperty<TagApi> tags)
    Q_PROPERTY(QDateTime fileCreated)
    Q_PROPERTY(QDateTime fileLastModified)
    Q_INVOKABLE QStringList tagNames();
    Q_INVOKABLE bool addTag(QString tagName);
    Q_INVOKABLE bool removeTag(QString tagName);
```

(continues on next page)

(continued from previous page)

```
    Q_INVOKABLE QString toMarkdownHtml(bool forExport = true);
};
```

You can use the methods from Date to work with `fileCreated` or `fileLastModified`.

For example:

```
script.log(note.fileCreated.toISOString());
script.log(note.fileLastModified.getFullYear());
```

1.3.2 Tag

```
class TagApi {
    Q_PROPERTY(int id)
    Q_PROPERTY(QString name)
    Q_PROPERTY(int parentId)
};
```

1.3.3 MainWindow

```
class MainWindow {
    Q_INVOKABLE void reloadTagTree();
    Q_INVOKABLE void reloadNoteSubFolderTree();
    Q_INVOKABLE void buildNotesIndexAndLoadNoteDirectoryList(
            bool forceBuild = false, bool forceLoad = false);
    Q_INVOKABLE void focusNoteTextEdit();
};
```

For example:

```
// force a reload of the note list
mainWindow.buildNotesIndexAndLoadNoteDirectoryList(true, true);
```

CHAPTER 2

Command line interface parameters

You can use these parameters on the command line interface:

Parameter	Description
`--help`	Shows the help screen
`--portable`	Runs the application in portable mode
`--clear-settings`	Clears the settings and runs the application
`--dump-settings`	Prints out a dump of the settings and other information about the application and environment in GitHub Markdown and exits the application
`--session <name>`	Runs the application in a different context for settings and internal files
`--allow-multiple-instances`	Allows multiple instances of QOwnNotes to be started even if disallowed in the settings

You may run the application on the command line interface differently on different operating systems:

Operating system	Output
Linux	*QOwnNotes*
macOS	*/Applications/QOwnNotes.app/Contents/MacOS/QOwnNotes*
Windows	*QOwnNotes.exe*

CHAPTER 3

Time formats

You can use your own time format when inserting the current time into a note.

These expressions may be used for the date:

Expression	Output
d	the day as number without a leading zero (1 to 31)
dd	the day as number with a leading zero (01 to 31)
ddd	the abbreviated localized day name (e.g. ‘Mon’ to ‘Sun’).
dddd	the long localized day name (e.g. ‘Monday’ to ‘Sunday’).
M	the month as number without a leading zero (1-12)
MM	the month as number with a leading zero (01-12)
MMM	the abbreviated localized month name (e.g. ‘Jan’ to ‘Dec’).
MMMM	the long localized month name (e.g. ‘January’ to ‘December’).
yy	the year as two digit number (00-99)
yyyy	the year as four digit number

These expressions may be used for the time:

Expression	Output
h	the hour without a leading zero (0 to 23 or 1 to 12 if AM/PM display)
hh	the hour with a leading zero (00 to 23 or 01 to 12 if AM/PM display)
H	the hour without a leading zero (0 to 23, even with AM/PM display)
HH	the hour with a leading zero (00 to 23, even with AM/PM display)
m	the minute without a leading zero (0 to 59)
mm	the minute with a leading zero (00 to 59)
s	the second without a leading zero (0 to 59)
ss	the second with a leading zero (00 to 59)
z	the milliseconds without leading zeroes (0 to 999)
zzz	the milliseconds with leading zeroes (000 to 999)
AP or A	use AM/PM display. *A/AP* will be replaced by either “AM” or “PM”.
ap or a	use am/pm display. *a/ap* will be replaced by either “am” or “pm”.
t	the timezone (for example “CEST”)

Example format strings:

Format	Result
dd.MM.yyyy hh:mm	21.05.2001 14:13
dd.MM.yyyy	21.05.2001
ddd MMMM d yy	Tue May 21 01
hh:mm:ss.zzz	14:13:09.042
h:m:s ap	2:13:9 pm

Take a look at the Qt documentation for more information about time formats.

If you need to make more complex operations to output the current time please consider creating a custom action.

CHAPTER 4

License

```
GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc., <http://fsf.org/>
 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it.  (Some other Free Software Foundation software is covered by
the GNU Lesser General Public License instead.)  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
this service if you wish), that you receive source code or can get it
if you want it, that you can change the software or use pieces of it
in new free programs; and that you know you can do these things.

  To protect your rights, we need to make restrictions that forbid
anyone to deny you these rights or to ask you to surrender the rights.
These restrictions translate to certain responsibilities for you if you
distribute copies of the software, or if you modify it.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must give the recipients all the rights that
```

(continues on next page)

(continued from previous page)

```
you have.  You must make sure that they, too, receive or can get the
source code.  And you must show them these terms so they know their
rights.

  We protect your rights with two steps: (1) copyright the software, and
(2) offer you this license which gives you legal permission to copy,
distribute and/or modify the software.

  Also, for each author's protection and ours, we want to make certain
that everyone understands that there is no warranty for this free
software.  If the software is modified by someone else and passed on, we
want its recipients to know that what they have is not the original, so
that any problems introduced by others will not reflect on the original
authors' reputations.

  Finally, any free program is threatened constantly by software
patents.  We wish to avoid the danger that redistributors of a free
program will individually obtain patent licenses, in effect making the
program proprietary.  To prevent this, we have made it clear that any
patent must be licensed for everyone's free use or not licensed at all.

  The precise terms and conditions for copying, distribution and
modification follow.

                    GNU GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

  0. This License applies to any program or other work which contains
a notice placed by the copyright holder saying it may be distributed
under the terms of this General Public License.  The "Program", below,
refers to any such program or work, and a "work based on the Program"
means either the Program or any derivative work under copyright law:
that is to say, a work containing the Program or a portion of it,
either verbatim or with modifications and/or translated into another
language.  (Hereinafter, translation is included without limitation in
the term "modification".)  Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope.  The act of
running the Program is not restricted, and the output from the Program
is covered only if its contents constitute a work based on the
Program (independent of having been made by running the Program).
Whether that is true depends on what the Program does.

  1. You may copy and distribute verbatim copies of the Program's
source code as you receive it, in any medium, provided that you
conspicuously and appropriately publish on each copy an appropriate
copyright notice and disclaimer of warranty; keep intact all the
notices that refer to this License and to the absence of any warranty;
and give any other recipients of the Program a copy of this License
along with the Program.

You may charge a fee for the physical act of transferring a copy, and
you may at your option offer warranty protection in exchange for a fee.

  2. You may modify your copy or copies of the Program or any portion
of it, thus forming a work based on the Program, and copy and
```

(continues on next page)

(continued from previous page)

```
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:

    a) You must cause the modified files to carry prominent notices
    stating that you changed the files and the date of any change.

    b) You must cause any work that you distribute or publish, that in
    whole or in part contains or is derived from the Program or any
    part thereof, to be licensed as a whole at no charge to all third
    parties under the terms of this License.

    c) If the modified program normally reads commands interactively
    when run, you must cause it, when started running for such
    interactive use in the most ordinary way, to print or display an
    announcement including an appropriate copyright notice and a
    notice that there is no warranty (or else, saying that you provide
    a warranty) and that users may redistribute the program under
    these conditions, and telling the user how to view a copy of this
    License.  (Exception: if the Program itself is interactive but
    does not normally print such an announcement, your work based on
    the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole.  If
identifiable sections of that work are not derived from the Program,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works.  But when you
distribute the same sections as part of a whole which is a work based
on the Program, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Program.

In addition, mere aggregation of another work not based on the Program
with the Program (or with a work based on the Program) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.

  3. You may copy and distribute the Program (or a work based on it,
under Section 2) in object code or executable form under the terms of
Sections 1 and 2 above provided that you also do one of the following:

    a) Accompany it with the complete corresponding machine-readable
    source code, which must be distributed under the terms of Sections
    1 and 2 above on a medium customarily used for software interchange; or,

    b) Accompany it with a written offer, valid for at least three
    years, to give any third party, for a charge no more than your
    cost of physically performing source distribution, a complete
    machine-readable copy of the corresponding source code, to be
    distributed under the terms of Sections 1 and 2 above on a medium
    customarily used for software interchange; or,
```

(continues on next page)

(continued from previous page)

```
    c) Accompany it with the information you received as to the offer
    to distribute corresponding source code.  (This alternative is
    allowed only for noncommercial distribution and only if you
    received the program in object code or executable form with such
    an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for
making modifications to it.  For an executable work, complete source
code means all the source code for all modules it contains, plus any
associated interface definition files, plus the scripts used to
control compilation and installation of the executable.  However, as a
special exception, the source code distributed need not include
anything that is normally distributed (in either source or binary
form) with the major components (compiler, kernel, and so on) of the
operating system on which the executable runs, unless that component
itself accompanies the executable.

If distribution of executable or object code is made by offering
access to copy from a designated place, then offering equivalent
access to copy the source code from the same place counts as
distribution of the source code, even though third parties are not
compelled to copy the source along with the object code.

  4. You may not copy, modify, sublicense, or distribute the Program
except as expressly provided under this License.  Any attempt
otherwise to copy, modify, sublicense or distribute the Program is
void, and will automatically terminate your rights under this License.
However, parties who have received copies, or rights, from you under
this License will not have their licenses terminated so long as such
parties remain in full compliance.

  5. You are not required to accept this License, since you have not
signed it.  However, nothing else grants you permission to modify or
distribute the Program or its derivative works.  These actions are
prohibited by law if you do not accept this License.  Therefore, by
modifying or distributing the Program (or any work based on the
Program), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Program or works based on it.

  6. Each time you redistribute the Program (or any work based on the
Program), the recipient automatically receives a license from the
original licensor to copy, distribute or modify the Program subject to
these terms and conditions.  You may not impose any further
restrictions on the recipients' exercise of the rights granted herein.
You are not responsible for enforcing compliance by third parties to
this License.

  7. If, as a consequence of a court judgment or allegation of patent
infringement or for any other reason (not limited to patent issues),
conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Program at all.  For example, if a patent
license would not permit royalty-free redistribution of the Program by
```

(continues on next page)

(continued from previous page)

```
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under
any particular circumstance, the balance of the section is intended to
apply and the section as a whole is intended to apply in other
circumstances.

It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system, which is
implemented by public license practices.  Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.

This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.

  8. If the distribution and/or use of the Program is restricted in
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Program under this License
may add an explicit geographical distribution limitation excluding
those countries, so that distribution is permitted only in or among
countries not thus excluded.  In such case, this License incorporates
the limitation as if written in the body of this License.

  9. The Free Software Foundation may publish revised and/or new versions
of the General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

Each version is given a distinguishing version number.  If the Program
specifies a version number of this License which applies to it and "any
later version", you have the option of following the terms and conditions
either of that version or of any later version published by the Free
Software Foundation.  If the Program does not specify a version number of
this License, you may choose any version ever published by the Free Software
Foundation.

  10. If you wish to incorporate parts of the Program into other free
programs whose distribution conditions are different, write to the author
to ask for permission.  For software which is copyrighted by the Free
Software Foundation, write to the Free Software Foundation; we sometimes
make exceptions for this.  Our decision will be guided by the two goals
of preserving the free status of all derivatives of our free software and
of promoting the sharing and reuse of software generally.

                            NO WARRANTY

  11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY
FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.  EXCEPT WHEN
OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES
```

(continues on next page)

(continued from previous page)

```
PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED
OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE ENTIRE RISK AS
TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU.  SHOULD THE
PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING,
REPAIR OR CORRECTION.

  12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR
REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,
INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING
OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED
TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY
YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER
PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE
POSSIBILITY OF SUCH DAMAGES.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
convey the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    {description}
    Copyright (C) {year}  {fullname}

    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License along
    with this program; if not, write to the Free Software Foundation, Inc.,
    51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

    Gnomovision version 69, Copyright (C) year name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.
```

(continues on next page)

(continued from previous page)

```
The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, the commands you use may
be called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  `Gnomovision' (which makes passes at compilers) written by James Hacker.

  {signature of Ty Coon}, 1 April 1989
  Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.
```