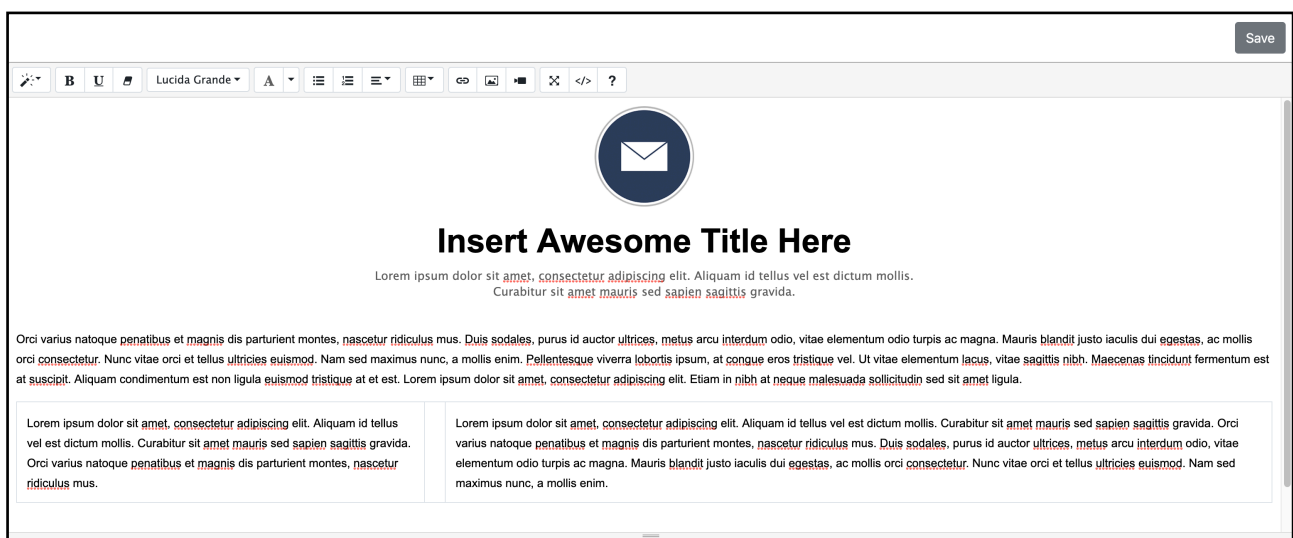


HTML Editors in Mailit 11

Mailit 11 comes with four new email editors that can be accessed through the FileMaker® Web Viewer: The Summernote editor and FileMaker Rich Text Editor allows for WYSIWYG editing of rich HTML emails. The Automizy and Unlayer editor lets you build professional emails including multi-column layouts from templates via drag & drop. This document explains how to integrate these editors into your FileMaker® custom app.

What is an Summernote?

Summernote is a JavaScript library that helps you create WYSIWYG editors.



Home page: <https://summernote.org>

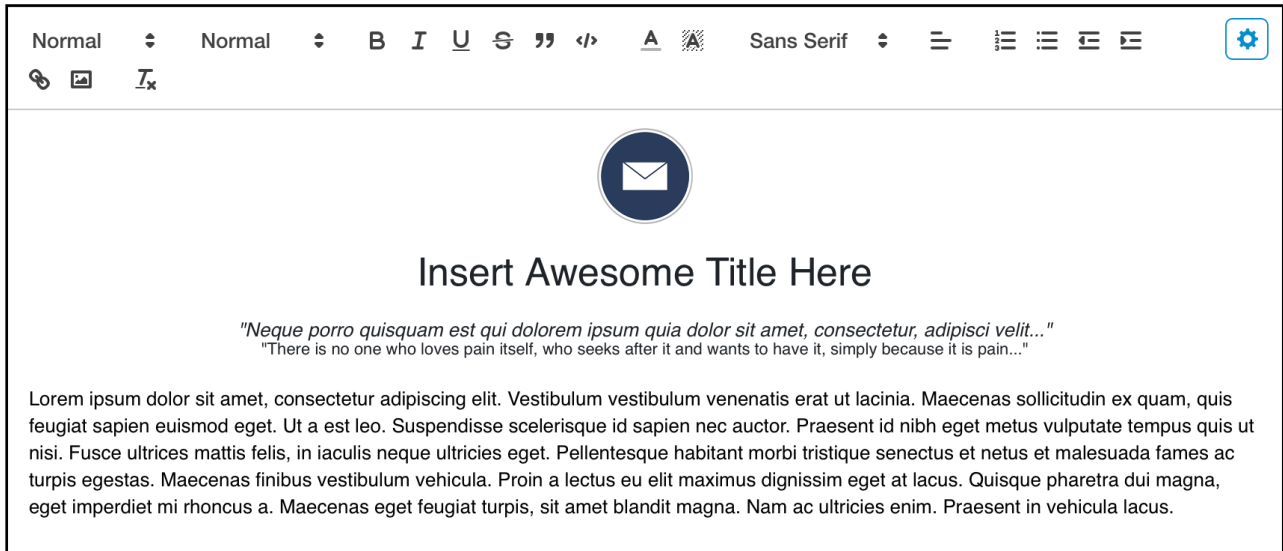
GitHub page: <https://github.com/summernote/summernote>

Summernote has a few special features:

- Paste images from clipboard;
- Saves images directly in the content of the field using base64 encoding, so you don't need to implement image handling at all
- Simple UI
- Interactive WYSIWYG editing
- Supports Bootstrap 3 and 4 integrities
- Lots of plugins and connectors provided together <https://github.com/summernote/awesome-summernote>

What is an FileMaker Rich Text Editor?

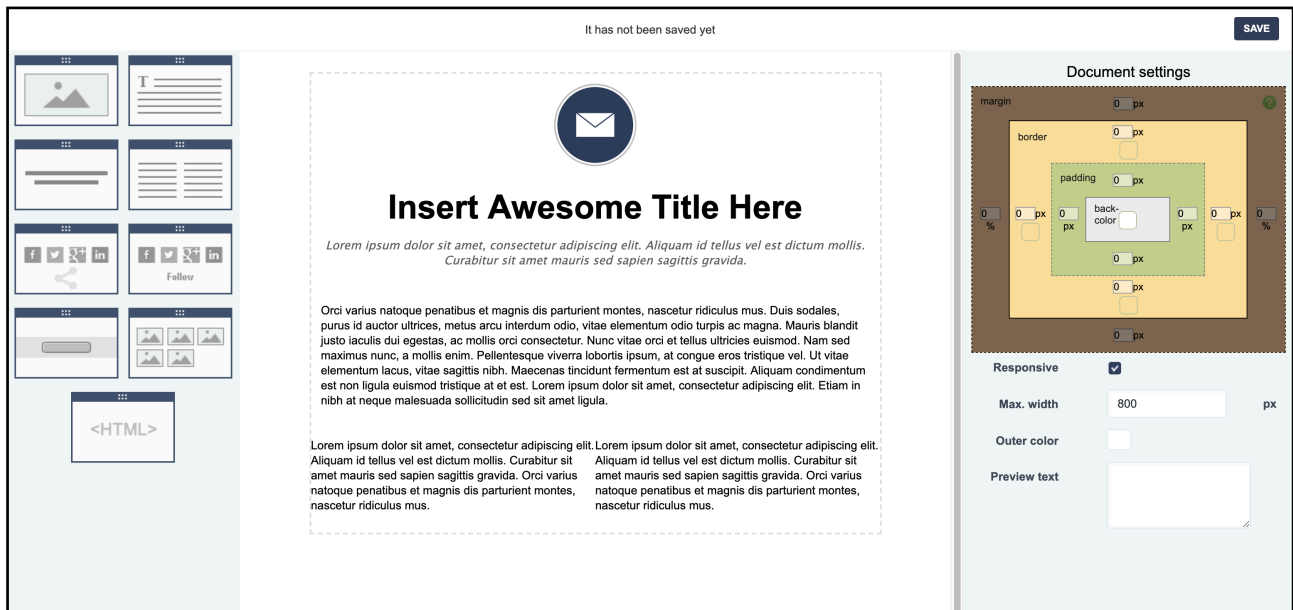
FileMaker Rich Text Editor is a FileMaker add-on which is based on the Quill editor (<https://quilljs.com/>) is a free and open source WYSIWYG editor that can be used for daily mailing.



FileMaker Rich Text Editor (<https://marketplace.claris.com/detail/1551.html>)
Home page Quill: <https://quilljs.com/>
GitHub page Quill: <https://github.com/quilljs/quill/>

What is an Automizy Email Editor?

Automizy Email Editor is an easily integrable JavaScript application which helps you to create HTML emails effortlessly. Design mobile friendly HTML emails in your own application.



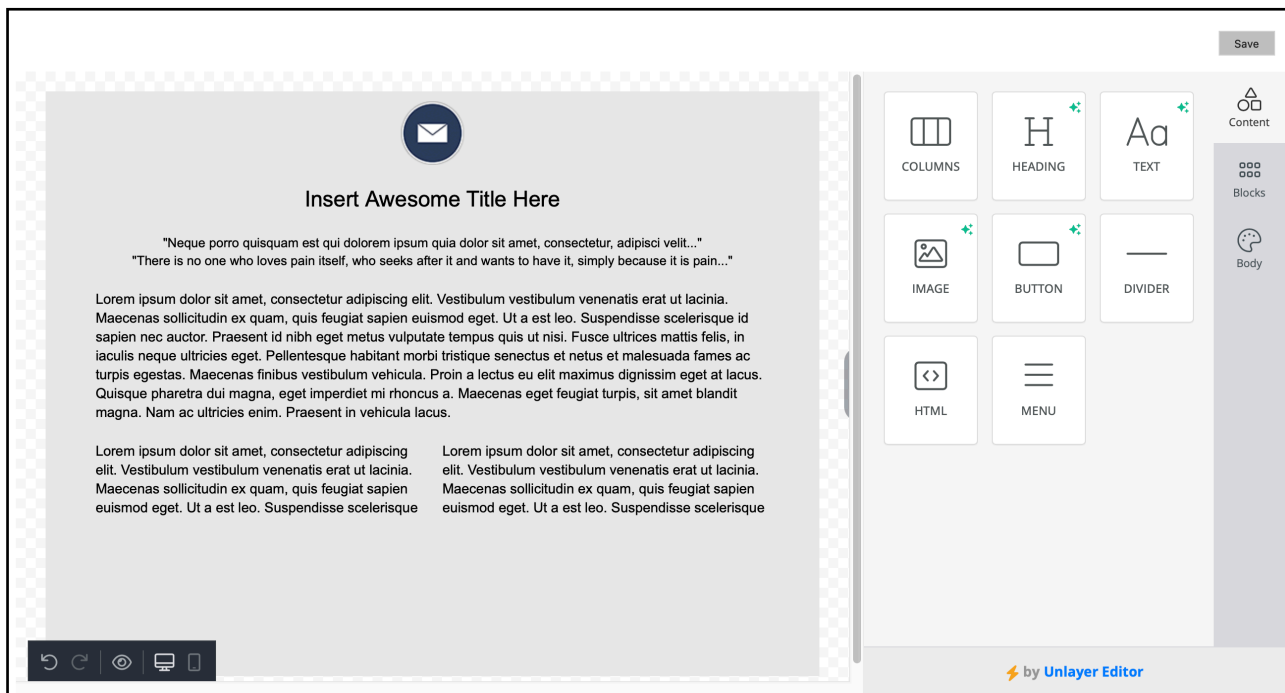
GitHub page: <https://github.com/Automizy/Automizy-Email-Editor>

Automizy Email Editor has a few special features:

- Easy usage
- User-friendly design
- Create responsive email
- Easy integration
- Method chaining & dynamic modifying

What is an Unlayer Email Editor (embed)?

Unlayer Email Editor (embed) is an easy-to-integrate editor with which you can easily construct HTML messages from blocks.



Home page: <https://unlayer.com/embed>

Pricing page: <https://unlayer.com/embed/pricing>

Unlayer Editor has a few special features:

- Easy usage
- User-friendly design
- Create responsive email
- Easy integration
- Method chaining & dynamic modifying

Initial setup

First you need to install the plugin MailIt and additional files for the plugin, to do this, open the file from the archive “../Plugin/Install Plugin.fmp12” and click the “Install Plug-in” button to install the plugin and additional plugin files. The plugin will be installed in the FileMaker plugins folder “Extensions”. You can open this folder in Explorer if you open "Preferences" on the "Plug-ins" tab, click the "Reveal Plug-in Folder" in FileMaker.

In the “Extensions” folder there will be a plugin file and in the same folder the “Mailit Assets” folder should appear in which there are additional files with editors.

Test-drive the new editors

The Mailit 11 download package includes example FileMaker apps demonstrating the WYSIWYG Summernote editor, FileMaker Rich Text Editor, Unlayer Email Editor, and Automizy Email Editor.

All simple editor examples are now combined into a single FileMaker file:

```
"../Email Apps for FileMaker@/Simple Sending.fmp12"
```

This file includes examples for the Summernote WYSIWYG Editor, FileMaker Rich Text Editor, Unlayer Email Editor, and Automizy Email Editor.

These editors are also available in the mass-mail example with HTML templates:

```
"../Email Apps for FileMaker@/Simple Mass Mail with HTML Templates/Simple Mass Mail.fmp12"
```

Details: Sending and receiving data in FileMaker® Web Viewer and JavaScript using the Web server built into MailIt plug-in

The Mailit 11 plugin has a built-in simple static web server. The word “static” means that the server only works with static files, the server does not support working with server languages such as PHP, Ruby, Java, Python, Perl.

To test the examples that are in the Mailit download package, the "Mailit Assets" folder must be copied together with the Mailit plug-in to the FileMaker® Extensions folder. The MailIt plugin automatically looks for the "Mailit Assets" folder within the FileMaker® Extensions folder. If necessary, you can change the name of the "Mailit Assets" folder to any other one and place the directory in any convenient place on your local drive. But when changing the name or location of the folder, you need to pass in the third parameter of the function, "path", the new location and name of the folder in the format of the FileMaker® paths using an absolute path, for example:
filewin:/driveletter:/directoryName/
filemac:/volumeName/directoryName/

The Mailit Assets folder contains:

- The "../Mailit Assets/editor/automizy" folder contains a library and related files for the "Automizy Editor".

- The "../Mailit Assets/editor/summernote" folder contains a library and related files for "Summernote".
- The "../Mailit Assets/editor/unlayer" folder contains the index.html file, which is a wrapper for the embeddable version of the Unlayer editor.
- The "../Mailit Assets/editor/upload" folder is used for temporary storage of images when working with "Summernote" and "Automizy Editor".

When starting the server, the plugin has several reserved addresses:

http://localhost:1234/ - default server address, you can change it in the settings. HTML, image, text files and directories in the directory "../Mailit Assets/" are available at this address. Only the GET request type is supported.

http://localhost:1234/editor/upload - used in JavaScript AJAX requests to upload images to a temporary folder that is predefined and located on the path "../Mailit Assets/editor/upload/", the location of the folder cannot be changed. Only the POST request type is supported.

http://localhost:1234/editor/load - used in JavaScript AJAX requests to retrieve text from fields in the FileMaker® database. Only the POST request type is supported. In the request, you need to pass the fieldName parameter with the name of the field in the database from which you want to return the result in response, for example fieldName = Messages :: Body, it will return the current active content of the "Body" field from the "Messages" table as a text string, provided that the user currently on the layout in which the table and field are available. To get the value "Messages :: Body", use the functions from the FileMaker® API "Evaluate"

http://localhost:1234/editor/save - used in JavaScript AJAX requests to transfer text to FileMaker® by calling a database script with parameters. Only the POST request type is supported. In the request, you need to pass the parameters: scriptFileName name of the database file; scriptName the name of the script in the database; scriptParameter parameters that will be passed to the executable script.

The scriptParameter parameter must be processed by the javascript encodeURIComponent() function, inside the plug-in the value of the scriptParameter parameter will be decoded back.

Let's take a closer look at the example of "Simple Sending Automizy Editor.fmp12".

The local web server starts when you open the example. The server may not start if the default port "1234" is occupied by another application or service.

For tests, there are two buttons "Start" and "Stop". The "Start" button executes a script in which it checks whether the server is running or not, if the server is running, first stops the server and then starts again, this is like restarting, use this button if you will change the server settings so that the changes take effect. The "Stop" button executes the server stop script.

There are three main fields in the server settings:

Local WebServer Host - IP address or host name. Default is "localhost"

Local WebServer Port - The field defines the port through which the local server will be available, by default the port "1234" is used.

Mailit Assets Path - The path to the mount point. By default, this is the path to additional plugin files in the "Mailit Assets" folder, which must be copied in the place with the plugin to the FileMaker® extension directory. If you decide to place the "Mailit Assets" folder in another place, specify the path to the folder in this field. The path is specified in FileMaker® format, that is, with the prefix "filewin://" or "filemac://" depending on the platform you are using.

We figured out the server settings. Go to the "Send Email" layout, on the layout in the WebView component FileMaker®, the page with the "Automizy Editor" should load.

The URL for loading the editor is based on this calculation:

"http://"&System::Local WebServer Host&"&System::Local WebServer Port&

Last Session Transcript
Local Web Server
OAuth 2.0 Client Credentials
OAuth 2.0 Additional settings

Local WebServer Hostlocalhost
Local WebServer Port1234

StartStop

MailIt Assets Path

Auto Export AttachmentOff

Export All Attachments

If you transfer the database to another computer or delete the "upload" folder with images, the images will no longer be displayed in the editor. In this case, there are two options, the first, for the item "Auto Export Attachment" you can set the value to "On", each time you open a message, the export of attachments will occur. Or, for the "Auto Export Attachment" item, set the value to "Off", and click the "Export All Attachments" button to export all attachments at a time.

"/editor/automizy/index.html?scriptFileName=" & Get (FileName) &
"&scriptName=putToFM"&
"&mid=" & Messages::Message ID &
"&hideSaveButton=0"

The result of the calculation:

http://localhost:1234/editor/automizy/index.html?scriptFileName=Simple Sending Automizy Editor&scriptName=putToFM&mid=51&hideSaveButton=0

http://localhost:1234 - the address and port on which the server is running.

/editor/automizy/index.html - relative path to index.html editor file located in the "Mailit Assets" folder.

GET URL variables:

scriptFileName=Simple Sending Automizy Editor - name of the current database file.

scriptName=putToFM - the name of the script in the file to which the result from the editor will be transferred.

mid=51 - the unique identifier of the message we are currently working with, number 51 is used as an example.

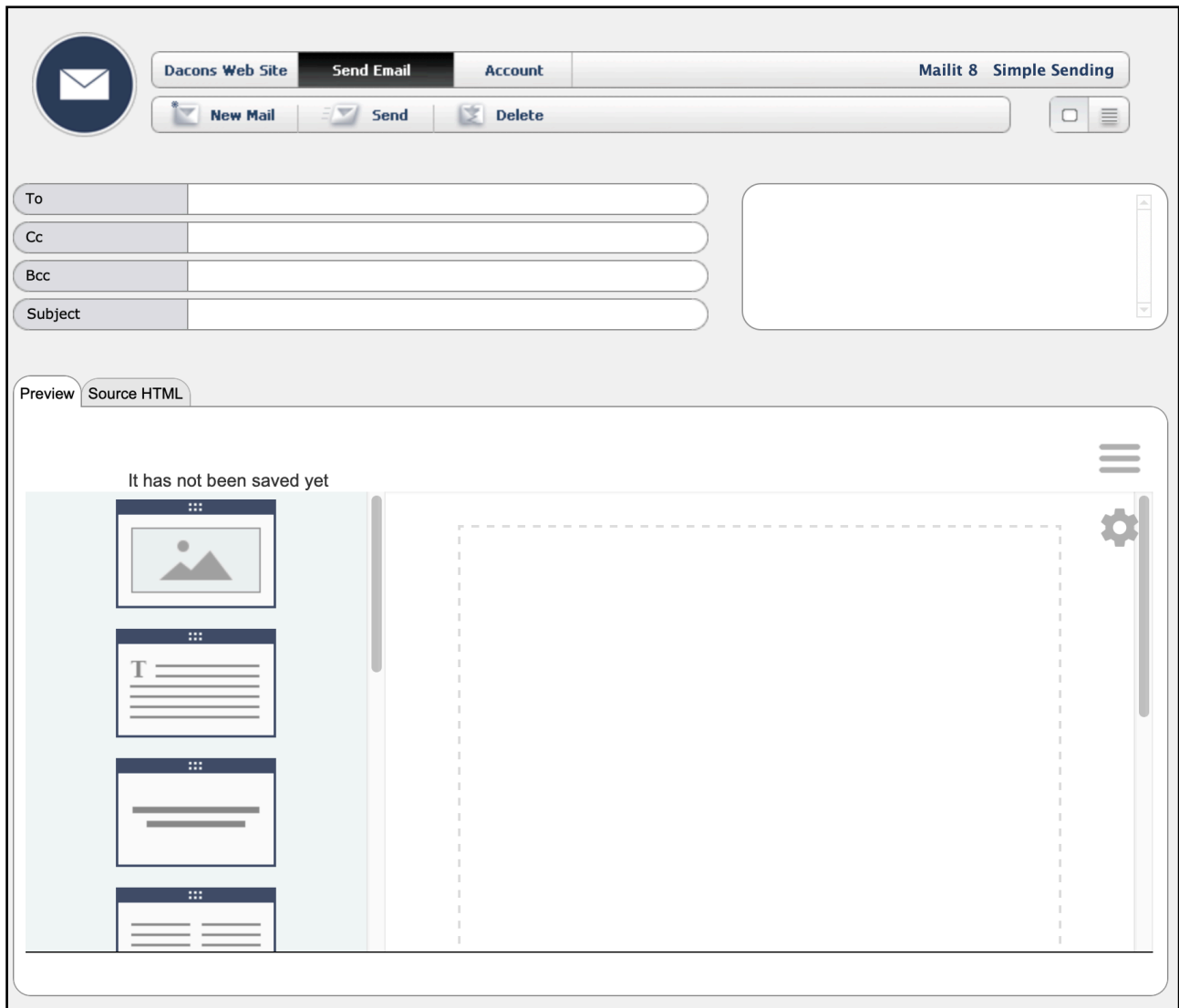
hideSaveButton=0 - a parameter that is responsible for hiding the "Save" button, if the parameter is 1 then the button will be hidden.

When the editor boots up, execute the Javascript code:

```

function loadContent() {
$.ajax({
  type: "POST",
  url: 'http://localhost:1234/editor/load',

```

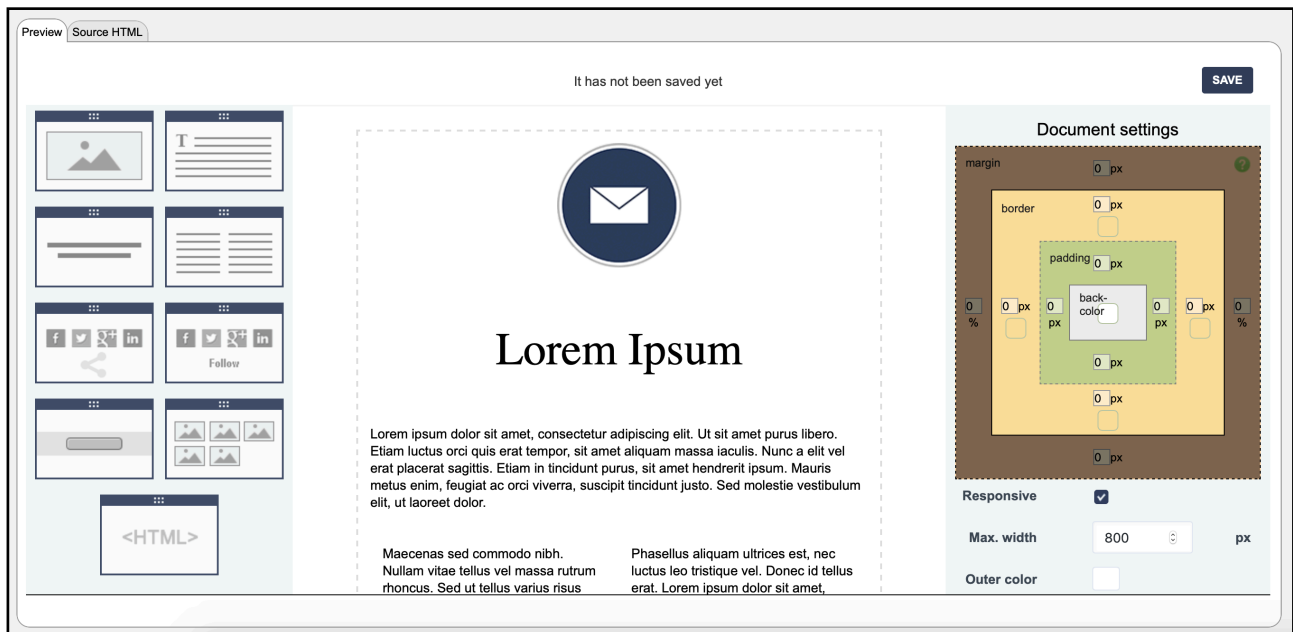


```
data: 'fieldName=Messages::Body',
success: function(data){
    //As a result of the query, the "data" variable will contain the value
of the "Body" field from the "Messages" table.
    $AEE.setEditorCode(data);
},
});
}
```

The code executes an AJAX request to the address <http://localhost:1234/editor/> load with POST parameters `fieldName = Messages :: Body`. The plugin execute the logical function `FileMaker® Evaluate(Messages :: Body)`, the result of the execution will be transmitted in response to the request. After loading, the editor is ready to work.

In the process of creating a message, you can use images. When the image is inserted into the editor, the image is transferred to the server by performing a POST request to the address <http://localhost:1234/editor/upload> with the GET variable `mid = 51`. The image will be uploaded to the `../Mailit Assets/editor/upload/51/image.png` directory. The mid parameter is used to create directories inside the `../Mailit Assets/editor/upload/` folder in order to avoid overwriting images in different messages with the same name. As a result of a successful image upload, the server response will contain the web address of the image, for example <http://localhost:1234/editor/upload/51/image.png>

As soon as the message is created, you must click the “Save” button in order to save the data. If you don’t click on the “Save” button and switch to another layout, all changes in the editor will be lost.



http://localhost:1234/editor/save - when the “Save” button is clicked, JavaScript code is executed:

```
function saveContentEditorCode(data) {
    $.ajax({
        type: "POST",
        url: 'http://localhost:1234/editor/save',
        data:
        'scriptFileName='+qs.scriptFileName+'&scriptName='+qs.scriptName+'_EditorCode&scriptParameter='+encodeURIComponent(data.editorCode),
        //data: 'scriptFileName=Simple Sending Automizy
        Editor&scriptName=putToFM_EditorContent&scriptParameter=<html>...</html>',
        success: function(response){
            saveContentHtmlCode(data);
        },
    });
}

function saveContentHtmlCode(data) {
    $.ajax({
        type: "POST",
        url: 'http://localhost:1234/editor/save',
        data:
        'scriptFileName='+qs.scriptFileName+'&scriptName='+qs.scriptName+'_HtmlCode&scriptParameter='+encodeURIComponent(data.htmlCode),
        //data: 'scriptFileName=Simple Sending Automizy
        Editor&scriptName=putToFM_Content&scriptParameter=<html>...</html>',
        success: function(response){},
    });
}
```

The code performs two AJAX POST requests to the address **http://localhost:1234/editor/save** with POST parameters: **scriptFileName**, **scriptName**, **scriptParameter**, the MailIt plugin will execute the script specified in the **scriptName** parameter and pass the value of **scriptParameter** as the script call parameter, in the database file whose name is specified in “**scriptFileName**”

The scriptParameter parameter must be processed by the javascript encodeURIComponent() function, inside the plugin the value of the scriptParameter parameter will be decoded back.

In the first request, the HTML code of the editor is saved so that you can edit the message in the future. The HTML code of the editor is passed to the script “putToFM_EditorContent” as a parameter, then the code is stored in the “Messages :: Editor Content” field. The second request saves the HTML code of the finished message. HTML message code is passed to the script “putToFM_Content” as a parameter, then the code is stored in the “Messages :: Message” field.

Details: Sending and receiving data in FileMaker® Web Viewer and JavaScript using the FileMaker JavaScript API

For FileMaker 19.0 and later, communication between FileMaker and JS editors is supported using the FileMaker JS API.

In the examples, index.html is still loaded into the WebView using the local static server built into the plugin, but the exchange of data between the editor and FileMaker takes place without the interaction of the Maillt plugin, via the FileMaker JS API.

When the editor's index.html file is loaded into the WebView, the JS function “loadContent();” is executed. In the loadContent function, a script is called in FileMaker with the name “putToJS”, and in the “putToJS” script, the “Perform JavaScript in Web Viewer” script step is executed in which the contents of the database field are transferred to the JS editor, example:

```
Perform JavaScript in Web Viewer [ Object Name: "WebViewEditor"; Function Name:
"setContent"; Parameter 1: $content ]
```

The setContent function takes data from FileMaker and passes it to the JS editor.

If you leave the layout with the editor, the data from the editor will not be transferred to FileMaker, new changes in the editor will not be saved, so before leaving the layout with the editor, you must save the editor data.

To save data from the editor, each index.html file has a “Save” button that executes the JS function “saveContent(_nextScriptName, _nextScriptOptions)” in which the data is passed to the FileMaker scripts “putToFM_Content” and “putToFM_EditorContent”. Additionally, two parameters _nextScriptName and _nextScriptOptions can be passed to the saveContent function, in which you can specify the script name in FileMaker and parameters as a text string that will be passed to the FileMaker script after the FileMaker scripts “putToFM_Content” and “putToFM_EditorContent” are executed.

An alternative option for saving data from the editor is to create a separate button in FileMaker, when clicked, the JS function “saveContent(_nextScriptName, _nextScriptOptions)” will be executed, this can be done using the script step “Perform JavaScript in Web Viewer”, example:

```
Perform JavaScript in Web Viewer [ Object Name: "WebViewEditor"; Function Name:
"saveContent"; Parameter 1: $scriptName; Parameter 2: $scriptParameter ]
```

If you choose an alternative option, then the save button located in the index.html file can be hidden, for this you need to pass the GET parameter in the URL parameters, hideSaveButton=1, example:

<http://localhost:1234/editor/unlayer/index.html?hideSaveButton=1>

