

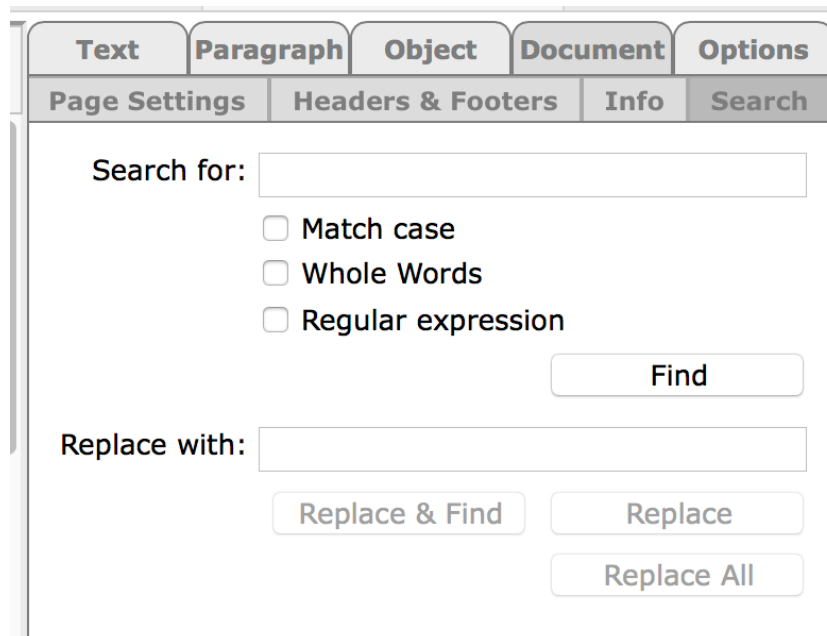
Client-Side Find & Replace

JS-OWrite version 4.5 implements basic find-replace functionality without involving the server. The new examples provide a working user interface for the most common *Find & Replace* actions.

We have broken down this document into three distinct parts. Part 1 explains the example user interface, part 2 ventures behind the scenes for a look at the Omnis code and part 3 explains in detail the new JS-OWrite client method `$searchex()`, which lies at the heart of the new *Find & Replace* feature.

1. The user interface

The updated example search tab implements the various find and replace buttons as well as search options including a regular expression option.



The screenshot shows a user interface for finding and replacing text. It features a tabbed interface with tabs for 'Text', 'Paragraph', 'Object', 'Document', and 'Options'. The 'Options' tab is active, showing a 'Search' sub-tab. The search area includes a 'Search for:' text input field, three checkboxes for 'Match case', 'Whole Words', and 'Regular expression', and a 'Find' button. Below this is a 'Replace with:' text input field, and three buttons: 'Replace & Find', 'Replace', and 'Replace All'.

1.1 Simple Find

Entering a search string and clicking the “Find” button will perform a simple search of the document. Repeatedly pressing the “Find” button will select subsequent matches until the search returns to the starting point. When a search has gone full circle or if no matches are found, appropriate messages are displayed.

1.2 Regular Expressions

JS-OWrite can also search the document using regular expressions such as “(PDFDevice|OSpell2)”. This particular expression will search for either specified word. Again, repeatedly pressing the “Find” button will select subsequent matches until the search returns to the starting point.

JS-OWrite expressions are javascript style regular expressions and detailed documentation of what expressions are supported can be found at https://www.w3schools.com/jsref/jsref_obj_regexp.asp.

1.3 Controlled Find & Replace

Performing controlled replace actions start with the search. Entering the search criteria followed by clicking the “Find” button will execute the search and select the first match. Subsequently, the “Replace & Find” button can be clicked to replace the selection and find the next match. Alternatively, the “Find” button can be clicked to move onto the next match without replacing the current selection. When the search returns to the starting point or if no matches are found, appropriate messages are displayed.

1.4 Replace All

The “Replace All” button will replace all matches with the specified text. When all matches have been replaced, a message is displayed telling the user how many matches have been replaced.

NOTE: The replace string must be plain text and cannot be a regular expression.

2. Behind the scene

All *Find & Replace* code in the example library is implemented to execute on the client. The beating heart of the new functionality is the new method `$searchex()` which is documented in detail in section 3. The previous method `$search()` is now obsolete but is still present for the time being. This section will show some of the code behind each user action as it was described in section 1.

Note: To locate all relevant code in the example library you can search the library for the string “change_2190531”.

2.1 Simple Find & Regular Expression Find

The “Find” button executes the `rfOWFormatDocument.doFind`. This method prepares the various `$searchex` parameters based on the criteria selected by the user and calls `$searchex()` which executes asynchronously.

Variable	Type	Subtype	Init.Val/Calc	Description	
1	searchFlags	Var	N/A	"g"	always require 'g' flag
2	searchMode	Var	N/A		
3	searchType	Var	N/A		

Task	Class	Instance	Local	Parameter	Notes
▼ Class methods					<code>; called by find button</code>
<code>\$event</code>					
<code>\$construct</code>					<code>; calculate search flags and search type</code>
<code>\$destruct</code>					Calculate <code>searchFlags</code> as <code>pick(ivSearchOptMatchCase,con(searchFlags,"i"),searchFlags)</code>
<code>\$init</code>					Calculate <code>searchFlags</code> as <code>pick(ivSearchOptWholeWord,searchFlags,con(searchFlags,"w"))</code>
<code>\$asyncDone</code>					Calculate <code>searchType</code> as <code>pick(ivSearchOptRegExp,"text","textRegExp")</code>
<code>\$clearSearch</code>					Calculate <code>searchMode</code> as <code>pick(ivSearchText=ivSearchTextFound,"New","Next")</code> <code>;; if ivSearchText does not match ivSearch</code>
<code>buildPaperList</code>					<code>; call overwrite \$search</code>
<code>setReplaceButtonStates</code>					<code>; JavaScript: console.log("type:"+searchType+" flags:"+searchFlags+" mode:"+searchMode)</code>
<code>doFind</code>					Do <code>\$cwind.\$objs.jsOWrite.\$searchex(searchType,ivSearchText,searchFlags,searchMode,ivReplaceText,"find",ivSearchText)</code>
<code>doReplace</code>					<code>; Note: last two parameters are the async parameters (the action and the action data)</code>
<code>doReplaceAll</code>					
▶ ButtonStripBackground					

JS-OWrite will generate the `evAsyncDone` event when the search action has completed. Our example `evAsyncDone` case calls the method `rfOWFormatDocument.$asyncDone` to handle the event.

<code>\$setButtonS</code>	On <code>evExecuteContextMenu</code>	Do method <code>\$cinst.\$handleExecContextMenu (pCommandID,\$cfield.\$curobjtype)</code>
<code>\$serverSave</code>		
▼ <code>jsOWrite</code>		
<code>\$event</code>	On <code>evModified</code>	Calculate <code>\$cinst.\$objs.btnSave.\$visible</code> as <code>kTrue</code>
▶ <code>btnSave</code>		
▶ <code>btnNewDocum</code>		
▶ <code>btnDeleteDocu</code>	On <code>evAsyncDone</code>	Switch (<code>pAsyncId</code>) <code>;; change_20190531 - find & replace example</code>
▶ <code>btnMergeData</code>		Case <code>"find","replaceAll"</code>
▶ <code>btnClearMerge</code>		Do <code>\$cinst.\$objs.FormatSubform.\$subinst.\$asyncDone(pAsyncId,pAsyncData,pAsyncResultData)</code>
▶ <code>btnShowPDF</code>		End Switch
▶ <code>propEnabled</code>		
▶ <code>DocumentList</code>		
▶ <code>FormatTabs</code>	On <code>evGetDataFromSrc</code>	<code>; OK message {[pObjID] - [pObjName] - [pObjCalc]}</code>
▶ <code>labelDocument</code>		

The *\$asyncDone* method displays the appropriate message if no match was found. Whether a match was found or not, the method *setReplaceButtonStates* is called to enable or disable the replace buttons accordingly.

Variable	Type	Subtype	Init.Val/Calc	Descr
_occurrences_of	Var	N/A	pick(ivSearchOptRegExp,"occurrences of","occurrences of the regular expression")	
_searching_for	Var	N/A	pick(ivSearchOptRegExp,"searching for","searching for the regular expression")	

Task	Class	Instance	Local	Parameter	Notes
<pre> Class methods \$event \$construct \$ddestruct \$init \$asyncDone \$clearSearch buildPaperList setReplaceButtonStates doFind doReplace doReplaceAll ButtonStripBackground propPaper propOrientation propPaperLength propPaperWidth labelMargins propTopMargin propLeftmargin propRightMargin propBottomMargin btnUpdateDocProps propHeaderMargin propFooterMargin propHeadFootOddEven propHeadFootFirstPage FormatTabs FormatDocumentPages propPaperContinuous propDocInfo propCurBookmark </pre>					
<pre> ; called by parent form when our evAsyncDone event is received by the owrite \$event method Switch pAsyncId Case "find" If not(pAsyncResultData) ; search failed If ivSearchTextFound=pAsyncData ; after a successfull search, failure means we have wrapped around to were we ; started searching with a "New" search mode OK message {Finished [_searching_for] "[pAsyncData]"} Else ; no matches found at all OK message {No matches found [_searching_for] "[pAsyncData]"} End If Calculate ivSearchTextFound as "" Do method setReplaceButtonStates Else ; we found something Calculate ivSearchTextFound as pAsyncData Do method setReplaceButtonStates End If Case "replace" ; nothing to do really Case "replaceAll" If pAsyncResultData ; some occurrences have been replaced (pAsyncResultData has the count) OK message {Replaced [pAsyncResultData] [_occurrences_of] "[pAsyncData]"} Else ; nothing has been replaced OK message {No [_occurrences_of] "[pAsyncData]" found} End If End Switch </pre>					

This completes the behind-the-scenes look of the simple and regular expression find.

2.2 Controlled Find & Replace

A controlled *Find & Replace* typically involves two actions after the initial find. Clicking the “Replace & Find” button calls the method *rfOWFormatDocument.doReplace* which is followed by a call to the method *rfOWFormatDocument.doFind*.

Task	Class	Instance	Local	Parameter	Notes
<pre> ; called after a successful search ;; use insert to replace current selection ; ; Do \$wind.\$objs.jsOWrite.\$insert(kWriObjTypeText,ivReplaceText,kWriInsertOver,"replace",ivReplaceText) ; ;; Note: last two parameters are the async parameters (the action and the action data) </pre>					

The *doReplace* method simply executes a \$insert to replace the current selection and although we provide async information, the case for a controlled replace action in *rfOWFormatDocument.doReplace*, implements no code as there is nothing else to be done after the selection has been replaced.

```

Calculate ivSearchTextFound as pAsyncData
Do method setReplaceButtonStates
End If
Case "replace" ← Async ID
; nothing to do really
Case "replaceAll"
If pAsyncResultData
; some occurrences have been replaced (pAsyncResultData has the count)
OK message {Replaced [pAsyncResultData] [_occurrences_of] "[pAsyncData]"}

```

The subsequent find action will follow the same route as described in section 2.1. This completes the behind-the-scenes look of the controlled *Find & Replace*.

2.3 Replace All

The “Replace All” button calls the method *rfOWFormatDocument.doReplaceAll*. In many ways this method appears very similar to the method *rfOWFormatDocument.doFind*. It prepares the various \$searchex parameters based on the *Find & Replace* criteria selected by the user and calls \$searchex().

Variable	Type	Subtype	Init.Val/Calc	Description
1 searchFlags	Var	N/A	"g"	always require 'g' flag
2 searchMode	Var	N/A		
3 searchType	Var	N/A		

Task	Class	Instance	Local	Parameter	Notes
<pre> ; called by replace all button ; calculate search flags and search type Calculate searchFlags as pick(ivSearchOptMatchCase,con(searchFlags,"i"),searchFlags) Calculate searchFlags as pick(ivSearchOptWholeWord,searchFlags,con(searchFlags,"w")) Calculate searchType as pick(ivSearchOptRegExp,"text","textRegExp") Calculate searchMode as "ReplaceAll" ; call overwrite \$search ; JavaScript: console.log("type:"+searchType+" flags:"+searchFlags+" mode:"+searchMode) Do \$cwind.\$objjs.jsOWrite.\$searchex(searchType,ivSearchText,searchFlags,searchMode,ivReplaceText,"replaceAll",ivSearchText) ; ; Note: last two parameters are the async parameters (the action and the action data) </pre>					

When the replace-all action has been carried out by JS-OWrite, the evAsyncDone event is generated and just as before, the event code calls the \$asyncDone method.

```

Case "replace"
; nothing to do really
Case "replaceAll" ← Async ID
If pAsyncResultData
; some occurrences have been replaced (pAsyncResultData has the count)
OK message {Replaced [pAsyncResultData] [_occurrences_of] "[pAsyncData]"}
Else
; nothing has been replaced
OK message {No [_occurrences_of] "[pAsyncData]" found}
End If
End Switch

```

3. The new search method

The new method `$searchex()` will search for the specified string or regular expression, selecting the next match, or perform a replace-all action. On completion, an async message is generated if the parameters `asyncId` and `asyncData` has been specified. Our *Find & Replace* interface makes use of these async messages to keep track of whether the next click on the search button should supply a mode of “New” or “Next” and to enable or disable the various *Find & Replace* buttons. When receiving the async message `evAsyncDone`, the event parameter `pAsyncResultData` will contain the result of the search or replace-all action. An unsuccessful search (including when a repeated search has wrapped around to the starting point) will set this parameter to zero. The next search action should then start a new search with the mode “New”. A successful search will set `pAsyncResultData` to 1 and the next search should use the modes “Next” or “Prev”. A replace-all action can be performed at any point and on completion `pAsyncResultData` will contain the number of occurrences that have been replaced.

The method syntax

`$searchex(searchType,searchValue,searchFlags,searchMode,replaceValue,asyncId,asyncData)`

The method parameters

searchType: This can be of the strings “text”, “textRegExp”, “class”, “bookmark”. The “textRegExp” type is new and JS regular expressions are fully explained at https://www.w3schools.com/jsref/jsref_obj_regexp.asp. IMPORTANT: regular expressions are explained as in “/pattern/modifiers;”. When calling `$searchex` you only specify the “pattern” for the `searchValue` parameter and “modifiers” for the `searchFlags` parameter, thus excluding the expression separators ‘/’ and ‘;’.

searchValue: The search string when `searchType` is “text” or the regular expression pattern when `searchType` is “textRegExp”.

searchFlags: Additional regular expression modifiers when `searchType` is “textRegExp”. If not supplied the flags default to “gi” (global case insensitive search). IMPORTANT: you may specify the non-standard expression flag ‘w’ for whole word searches. `$searchex` will always append the “g” modifier flag which is essential for our implementation, but it does not have to be specified by the developer. In most cases you may only specify “w” for whole word searches or an empty string for case insensitive searches.

searchMode: One of the strings “Prev”, “Next”, “New” or “ReplaceAll” when `searchType` is “text” or “textRegExp”. Whenever you start a new search (i.e. the search string has changed or you completed the previous search by reaching the end of the document) the mode “New” should be used. After the initial “New” you may continue searching with “Next” or “Prev”.

replaceValue: This specifies the new text for replace actions when `searchMode` is “ReplaceAll”. Only plain text is supported.

asyncId: The async ID that is passed in the pAsyncId event parameter for the evAsyncDone event, when the search or replace action has completed.

asyncData: The async data that is passed in the pAsyncData event parameter for the evAsyncDone event, when the search or replace action has completed.

Document History

02 July 2019: first publication