Designer API

Functions

After()

Insert content after each element in the set of matched elements. See also: before()

- after(content)

after(content)

Insert content after each element in the set of matched elements. After creates a new resultset which is used in chained functions. In case a plain text string (text node) is provided the content is automatically wrapped in a elemement to avoid orphan text nodes directly in the body element.

- **content** HTML string, string or HTML string to insert after the matched elements.

Examples	Before	After
query("#salesrep").after(" Lorem ipsum ");	Peter Parker	Peter Parker Lorem ipsum
query("#salesrep").after(" Lorem ipsum ").css("color","red");	Peter Parker	Peter Parker Lorem ipsum
results.after(" Lorem Ipsum ");	Peter Parker	Peter Parker Lorem ipsum
results.find("Lorem ").after("ipsum");	Lorem dolor sit amet, consectetur adipiscing elit.	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
query("#salesrep").after("Lorem Ipsum");	Peter Parker	Peter Parker Lorem Ipsum

Before()

Insert content before each element in the set of matched elements. See also: after()

- before(content)

before(content)

Insert content before each element in the set of matched elements. After creates a new resultset which is used in chained functions. In case a plain text string (text node) is provided the content is automatically wrapped in a elemement to avoid orphan text nodes directly in the body element.

- **content** HTML string, string or HTML string to insert after the matched elements.

Examples	Before	After
query("#salesrep").before(" Lorem ipsum ");	Peter Parker	Lorem ipsum Peter Parker
query("#salesrep").before(" Lorem ipsum ").css("color","red");	Peter Parker	Lorem ipsum Peter Parker
results.before(" Lorem Ipsum ");	Peter Parker	Lorem ipsum Peter Parker
results.find("ipsum").before("Lorem ");	ipsum dolor sit amet, consectetur adipiscing elit.	Lorem ipsum dolor sit amet, consectetur adipiscing elit.
query("#salesrep").before("Lorem Ipsum");	Peter Parker	Lorem Ipsum Peter Parker

Append()

Insert content at the end of each element in the set of matched elements. See also: prepend()

- append(content)

append(content)

Insert content as the last element to each element in the set of matched elements. *Append* creates a new resultset which is used in chained functions. In case a plain text string (text node) is provided the content is automatically wrapped in a elemement to avoid orphan text nodes directly in the body element.

- **content** HTML string, string or HTML string to insert after the matched elements.

Examples	Before	After

Find: #box, Type: selector
results.append(" Peter Parker ");
	
	 Personal information
	
		 Personal information
		 Peter Parker

Find: .name, Type: selector results.append("Peter Parker");	 Personal information Name: 	 Personal information Name: Peter Parker
Insert content to multiple elements at once *Find: div, Type: selector* results.append(" Peter Parker ");	 Personal information Personal information 	 Personal information Peter Parker Personal information Peter Parker

Append snippet
 	
var a = loadhtml('snippets/snippet_name.html');
 	
results.append(a);
	
	 Personal information
	
		 Personal information
		 Peter Parker

query("#box").append(" Peter Parker ");	 Personal information 	 Personal information Peter Parker
query("#box").append(" Peter Parker ").css("color","red");	 Personal information 	 Personal information Peter Parker

Prepend()

Insert content at the beginning of each element in the set of matched elements. See also: append()

- prepend(content)

prepend(content)

Insert content as the first element to each element in the set of matched elements. *Append* creates a new resultset which is used in chained functions. In case a plain text string (text node) is provided the content is automatically wrapped in a elemement to avoid orphan text nodes directly in the body element.

- **content** HTML string, string or HTML string to insert after the matched elements.

Examples	Before	After

Find: #box, Type: selector
results.prepend(" Personal information ");
	
	 Peter Parker
	
		 Personal information
		 Peter Parker

Find: .name, Type: selector results.prepend("Name: ");	 Personal information Peter Parker 	 Personal information Name: Peter Parker
Insert content to multiple elements at once *Find: div, Type: selector* results.prepend(" Personal information ");	 Peter Parker Peter Parker 	 Personal information Peter Parker Personal information Peter Parker

Prepend snippet
 	
var a = loadhtml('snippets/snippet.html');
 	
results.prepend(a);
	
	 Peter Parker
	
		 Personal information
		 Peter Parker

query("#box").prepend(" Personal information ");	 Peter Parker 	 Personal information Peter Parker
query("#box").prepend(" Personal information ").css("color","red");	 Peter Parker 	 Personal information Peter Parker

Add()

The designer script engine a function that allows you to subtract elements from a result set (called "filter"). The add() function allows you to add elements to a result set. This option emulates a part of jQuery's "add" function: see https://api.jquery.com/add/.

Use case: construct a complex result set before calling "after", ensuring that the order in which elements are inserted is predictable.

- add(content)

add(content)

Returns the union of this result set and other content. The supplied content can be an HTML string or a result set.

- **content** QueryResult

Examples	Before	After
Add one query result to another query("#test1").add(query("#test2")).css("background", "yellow");		

Load snippets in an iteration and use add its elements to an empty result set (query()). Replace a placeholder in the template with the new result
 	
Find selector: #chapters
 	
var chapters = query();
for (var i = 1; i <= 4; i++) {
  chapters = chapters.add(loadhtml('snippets/Chapter' + i + '.html'));
}
 	
results.replaceWith(chapters);
	{{chapters}}	Chapter 1
		 Lorem ipsum...
		 Chapter 2
		 Lorem ipsum...
		 Chapter 3
		 Lorem ipsum...
		 Chapter 4
		 Lorem ipsum...

css()

Get the value of a style property for the first element in the set of matched elements or set one or more CSS properties for every matched element.

- css(propertyName)
- css(propertyName, value)
- css(properties)

css(propertyName)

Returns the value of the specified CSS property.

- **propertyName** string, the name of the CSS property

Examples
var textcolor = results.css("color");
var backgroundcolor = query("#calloutbox").css("background-color");

css(propertyName, value)

Function to set a CSS property.

- **propertyName** string, the name of the CSS property
- **value** string, value for the CSS property or a map of property-value pairs to set

Examples
query("#callout p").css('color' , 'red');
results.css('color' , '#669900');

if(record.fields.accountype == "PRO") {
`query("#callout p").css("color","red");`
 	
`}`

// Load a snippet into variable replace find/replace text in the snippet and apply a css property to the replacement text
 	
var mysnippet = loadhtml('snippets/snippet vars.html');
mysnippet.find('@var@').text('OL Connect').css('text-decoration','underline');
results.replaceWith(mysnippet);

css(properties)

Function to set one or multiple CSS properties.

- **properties** array, map of property-value pairs to set

Examples
results.css({'color' : 'red', 'font-weight' : 'bold'});

addClass()

Adds the specified class(es) to each element in this result set. Has no effect if the class is already present.

- addClass(classname)

addClass(classname)

Adds the specified class(es) to each element in this result set. Has no effect if the class is already present.

- **classname** string, space separated list of classnames

Examples	Before	After
Selector:p results.addClass("foo");	Hello world	Hello world
Selector:p results.addClass("foo bar");	Hello world	Hello world

removeClass()

Removes the specified class from each element in this result set. Has no effect if the class is not present.

- removeClass(classname)

Clone

If the user does want to duplicate an existing DOM element he should clone it before calling append().

- clone()

clone()

Placeholder

If "results" consists of more than one target element, for the second target element and each subsequent target element we should insert *copies* of the source elements. This conforms to the behavior of jQuery (see http://jsfiddle.net/kwbb25hb/)

Examples

Simpel iteration over the elements in the result set
 	
`var row = query("tbody tr", results).clone();`
`query("tbody", results).append(row);`

Clone an existing table row to match the number of rows in a detail table. Afterwards iterate over the rows to populate the fields
 	
// Create the number of rows based on the records in the detail table
// We start at 1 so the boilerplate row is used too and there is no need to delete that row
for(var r = 1; r < record.tables['detail'].length; r++) {
  results.parent().append(results.clone());
}
 	
// Iterate over the rows and populate them with the data from the accompanying data row
query("#table_2 > tbody > tr").each(function(i) {
  this.find('@ItemNumber@').text(record.tables['detail'][i].fields["ItemNumber"]);
  this.find('@ItemOrdered@').text(record.tables['detail'][i].fields["ItemOrdered"]);
  this.find('@ItemTotal@').text(record.tables['detail'][i].fields["ItemTotal"]);
  this.find('@ItemDesc@').text(record.tables['detail'][i].fields["ItemDesc"]);
  this.find('@nr@').text(i);
});

Clone and populate a boilerplate row. Once completed you will need to hide the boilerplate row
 	
for(var i = 0; i < record.tables['detail'].length; i++) {
 	
  var row = results.clone(); //Clone our boilerplate row
 	
  row.find('@ItemNumber@').text(record.tables['detail'][i].fields["ItemNumber"]);
  row.find('@ItemOrdered@').text(record.tables['detail'][i].fields["ItemOrdered"]);
  row.find('@ItemTotal@').text(record.tables['detail'][i].fields["ItemTotal"]);
  row.find('@ItemDesc@').text(record.tables['detail'][i].fields["ItemDesc"]);
  row.find('@nr@').text(i);
 	
  results.parent().append(row);
 	
}
 	
// Hide our boilerplate row (note that this doesn't really delete the row).
results.hide();

Remove

Remove the set of matched elements from the DOM.

- remove()

remove()

Remove the set of matched elements from the DOM. Use empty() to remove the children/inner HTML of the matched elements, use remove() the remove the elements themselves.

Examples	Before	After
Simple remove Selector: span Script: results.remove();	Lorem ipsum dolor sit amet, consectetuer adipiscing elit.	Lorem ipsum amet, consectetuer adipiscing elit.
Remove versus Empty Selector: span Script: results.empty();	Lorem ipsum dolor sit amet, consectetuer adipiscing elit.	Lorem ipsum amet, consectetuer adipiscing elit.

find()

Performs a deep search for textToFind in the children of each element, and returns a new result set with elements that surround the occurrences.

- find(textToFind)

Examples

Retrieve the inner HTML of the elements in a snippet, perform a find/replace
var snippet = loadhtml('snippets/snippet.html','#foobar').children();
snippet.find('@firstname@').text('foobar');
 	
results.append(snippet);

Global Functions

Loadhtml()

Global function that replaces the content (inner html) of each matched element in the result set, alternatively load the data into a variable. The location should be an URL or a relative file path.

- loadhtml(location)
- loadhtml(location, selector)

Note! Loadhtml() is cached per batch run (based on the URL) in print/email.

loadhtml(location)

Loads all html from the HTML file

- **location** string, can be absolute or relative to the section/context. Use: snippets/<snippet-name> to retrieve the content from a HTML file residing in snippets folder of the Resources panel.

Examples
Loads a local html snippet (Resources panel) directly into the matched elements results.loadhtml("snippets/snippet.html");

Loads a local html snippet (Resources panel) into a variable
The replaceWith() command is used to replace the matched element with the contents of the snippet.
 	
var mysnippet = loadhtml('snippets/snippet.html');
 	
results.replaceWith(mysnippet);
 	
// Same result different notation
 	
results.replaceWith(loadhtml('snippets/snippet.html'));

Load a snippet into a variable and find/replace text in the variable before injecting the content into the page. Second find command adds formatting to the replacement text
 	
var mysnippet = loadhtml('snippets/snippet.html');
 	
mysnippet.find('@var1@').text('OL Connect 1');
 	
mysnippet.find('@var2@').html('OL Connect 2').css('text-decoration','underline');
 	
results.replaceWith(mysnippet);

Load a snippet into a variable and retrieve an element from the snippet using query()
 	
var mysnippet = loadhtml('snippets/text-root-wrapped.html');
var subject = query("#subject", mysnippet).text();
 	
results.append(" " + subject + " ");

loadhtml(location, selector)

Retrieve specific content from the filename.

- **location** string, can be absolute or relative to the section/context. Use: snippets/<snippet-name> to retrieve the content from a HTML file residing in snippets folder of the Resources panel.
- **selector** string, the supplied selector should conform to CSS selector syntax and allows you to retrieve only the content of matching elements.

Examples

// Load a specific element from the snippet
 	
var mysnippet = loadhtml('snippets/snippet-selectors.html','#item3');
 	
results.replaceWith(mysnippet);

Load the children of the selected element
 	
var snippet = loadhtml('snippets/snippet.html','foobar').children();
 	
results.replaceWith(snippet);

Loadjson()

Creates a JSON object based on the text retrieved from the supplied location. The function lets you retrieve content from an JSON enabled server using a standard HTTP request. Popular content management systems, like WordPress (requires JSON API plug-in) and Drupal, provide a JSON service/api to retrieve content.

- loadjson(location)

Note! Loadjson() is cached per batch run (based on the URL) in print/email.

Note! Online JSON viewer (handy to debug JSON data): http://jsonviewer.stack.hu

loadjson(location)

Loads json data from a remote location

- **location** string, the supplied location should be either a URL or a relative file path.

Examples

//This sample retrieves JSON data from a snippet
 	
var localJSON = loadjson('snippets/jsonsnippet.html');
 	
if(localJSON.post){
results.html(" " + localJSON.post.title + " " + localJSON.post.modified + " ");
}

// This sample retrieves a post from a WordPress site
 	
var wpPost = loadjson('http://192.168.101.58/2013/06/leave-the-third-dimension-behind-and-focus-on-real-printing-innovation/?json=1');
 	
if(wpPost.post){
results.html(" " + wpPost.post.title + " " + wpPost.post.content);
}

// This sample retrieves a post from a WordPress site
 	
var numPosts = 3;
var wpPost = '';
var wpRecentPosts = loadjson('http://192.168.101.58/?json=get_recent_posts&count=' + numPosts);
 	
if(wpRecentPosts.posts){
for (var i = 0; i < numPosts ; i++) {
wpPost += " " + wpRecentPosts.posts[i].title + " ";
}
}
 	
results.after(wpPost)

query()

Creates a new result set containing the HTML elements that match the supplied CSS selector. The context (optional) allows you to restrict the search to descendants of one or more context elements.

- query(selector)
- query(selector, context)

query(selector)

Creates a new result set containing the HTML elements that match the supplied CSS selector.

Examples	Before	After

Apply css to the queried elements
query("#test1").css("color", "yellow");
query("#test2").css("color","red");
	foo
	 bar
		foo
		 bar

Query an element in a snippet and set its text
var snippet = loadhtml('snippets/mysnippet.html');
query("#foo", snippet).text("bar");
 	
results.replaceWith(snippet);		

children()

Returns the immediate children (inner HTML) of the elements in this result set.

- children()

Examples

Retrieve the inner HTML of an element selector in a snippet
var snippet = loadhtml('snippets/snippet.html','#foobar').children();
 	
results.append(snippet);

Retrieve the inner HTML of the elements, perform a find/replace
var snippet = loadhtml('snippets/snippet.html','#foobar').children();
snippet.find('@firstname@').text('foobar');
 	
results.append(snippet);

Statements

For...in

Can be used to iterate over fields in a dataset or rows in detail table.

- for(variable in object)

for(variable in object)

Iterates over the enumerable properties of an object, in arbitrary order. For each distinct property, statements can be executed.

Examples	Before	After

Iterate over field names in the main record
Find selector: #test
 	
for(var i in record.fields){
  results.after(" " + i+ " ");
}
	<h1 id="test">Fields</h1>	<h1 id="test">Fields</h1>
		 first
		 last
		 email

Iterate over field in the main record retrieve value
Find selector: #test
 	
for(var i in record.fields){
  results.after(" " + record.fields[i] + " ");
}
	<h1 id="test">Fields</h1>	<h1 id="test">Fields</h1>
		 Peter
		 Parker
		 pparker@localhost.com

Iterate over rows in a detail table
Find selector: #countries
 	
for(var i in record.tables['countries']) {
  results.after(" "
+ record.tables['countries'][i].fields['country'] + " ");
}
	<h1 id="countries">Countries</h1>	<h1 id="countries">Countries</h1>
		 The Netherlands
		 Canada
		 Australie

for(var i in record.tables['detail']) {
  var str = record.tables['detail'][i].fields["ItemID2"];
  results.append("<option value='" + str + "'>" + str +
"</option>");
}		

Each

A generic iterator function, to iterate over the elements in the result set. The callback is passed the iteration index and the current element. In scope of the callback, this also refers to the current element.

- each(callbackt)

each(callback)

Iterates over the enumerable properties of an object, in arbitrary order. For each distinct property, statements can be executed.

Examples	Before	After

Simpel iteration over the elements in the result set
 	
results.each(function(index){
  results[index].css('background-color','red');
});		

Iterate over elements in the result and set a the html content of the element with a value from an array.
 	
var strings = loadjson('snippets/' + record.fields.locale +
'.html');
 	
results.each(function(index){
  if(strings[this.attr('data-translate')])
  this.html(strings[this.attr('data-translate')]);
});
	<h1 id="test">Fields</h1>	<h1 id="test">Fields</h1>
		 first
		 last
		 email

Apply a random int
Selector: p
 	
results.each(function(index){
  var test = Math.floor((Math.random() * 10) + 1);
  this.html(test);
});
	
		3
		 1
		 7

Show row index
Selector: p
 	
results.each(function(index){
  this.text(index);
}
	
		0
		 1
		 2