Event Handling - ClientData

By default the callback mechanism will return:

`_event`	the name of the event
`_what`	the id/name of the element that triggered the event
`_value`	the value of the element that triggered the event (if a value exists)
`_selector`	the selector of the handler

any form data that is on the page is serialized and returned using the names of the form input elements.

You can specify other information to be sent to the server:

`name {selector} {type} {which}`	
`name`	the name to give the data on the server side
`selector`	jQuery/CSS selector of the element from which to get the data if omitted, use the element to which the handler is bound
`type`	the type of data to return. valid types include:

`type =`	**returns**
`attr`	an HTML attribute
`css`	a CSS setting
`html`	the HTML content
`is`	specific settings – see jQuery.is()
`val`	the value of the element
`eval`	the result of the evaluation of a JavaScript string
`string`	constant string
`event`	jQuery event object
`ui`	jQuery ui object
`ejModel`	SyncFusion model object
`argument`	SyncFusion argument object
`serialize`	all data for a form (unnecessary if there is only a single form on the page)

`which`	dependent on `type`

`type =`	**which =**	**Example**
`attr`	the attribute to return	`'attr' 'title'`
`css`	the CSS setting to return	`'css' 'font'`
`html`	"	`'html'`
`is`	the setting to return – see jQuery.is()	`'is' ':checked'`
`val`	"	`'val'`
`eval`	the JavaScript string to evaluate	`'eval' '2+2'`
`string`	the string to return	`'string' 'constant'`
`event`	the element of the event object	see jQueryUI document
`ui`	the element of the ui object	see jQueryUI document
`model`	the element of the model object	see Syncfusion document
`argument`	the element of the argument object	see Syncfusion document

	serialize	''	'serialize'

Example:

```
h←Add Handler            ⍝ add an event handler
h.ClientData←('content' '#div1' 'html')
             ('color' '#div2' 'css' 'background-color')
```

Returns

- a variable named "content" which contains the HTML content of the element with id "div1"
- a variable named "color" with the background color setting of the element with id "div2"

Event Handling - Retrieving Client Data from Callback Functions

Client data can be retrieved in two basic ways:

- Define a public field in your MiPage with the same name as the data you want to retrieve
  For instance, in the specification used above
  ```
        h.ClientData←('content' '#div1' 'html')
                     ('color' '#div2' 'css' 'background-color')
  ```
  You could defined two fields and they would be populated automatically.
  ```
        :field public content
        :field public color
  ```

  When using this method, it is generally important to clear the value of the fields after sending the response to the client so that subsequent requests don't reuse the same data.

- The other way is to use the Get method to retrieve the data by name. The left argument to Get is the default value to use if the data element is not found. Again, using the example from above, the following could be used.
  ```
        '???' Get 'content'
        'white' Get 'color'
  ```

Event Handling – Sending Responses Back to the Client

There are four functions which specify actions to be taken on the client side in response to a callback function.

`r ← selector Replace new`
`r ← selector Append new`
`r ← selector Prepend new`
`r ← Execute javascript`	

`Replace`	Replaces the HTML content of the element specified by `selector` with `new`
`Append`	Appends `new` to the HTML content of the element specified by `selector`
`Prepend`	Prepends `new` to the HTML content of the element specified by `selector`
`Execute`	Executes javascript string (using JavaScript's eval() function)
`selector`	The selector of the elements to update
`new`	The new content with which to update
`javascript`	A character vector of the JavaScript to execute in the client

Example:

```
r ← '#result' Replace _html.h2('Hi')
r,← Execute 'alert("Happy Birthday!")'
```

Callback functions must return a result, though the result could be '' if no action is to be taken on the client side.