09. DATATABLES Events.

9.1. Events

It can often be useful to know when DataTables or one of its extensions has performed a particular operation, for example a page draw, so other dependent elements can be updated to take account of the change. To provide this ability, DataTables will fire **custom DOM events** which can be listened for, and then acted upon, using either the **on**() method or jQuery's **on**() method. DataTables's custom events work in exactly the same way as standard DOM events, and allow event driven actions, which is particularly useful for plug-ins.

For a full list of the events that DataTables and its extensions will trigger, please refer to the event reference documentation.

https://datatables.net/reference/event

9.2. Listening for events

As noted above, **you can use either the on() or jQuery's on() method to listen for events**. on() works in exactly the same way as ***$().on()***, with provision for namespaces and multiple events.

Please be aware that all DataTables events are triggered with the **dt namespace**. This namespacing of events is to prevent clashes with custom events triggered by other Javascript libraries. As such, you should append .dt to the name of the event(s) that you are listening for (*when using on() the namespace is automatically appended if required*). Because of the way namespaces work in jQuery, you can use the **dt namespace** and your own custom namespace(s) if you wish to use a namespace.

For example, to listen for the draw event in a DataTable:

Example:

```
var table = $('#example').DataTable();

table.on( 'draw', function () {
    alert( 'Table redrawn' );
} );
```

This could also be written as:

Example:

```
$('#example').on( 'draw.dt', function () {
    alert( 'Table redraw' );
} );
```

Note the use of the dt namespace when using the jQuery on() method, while the on() method will automatically append the namespace for you.

9.3. Removing events

As with ***$().on()*** DataTables events can be removed with **off()** and **$().off()**. It is important to remove events from objects which no longer exist (*before they are destroye*d) to allow the Javascript engine's garbage collector to release the memory allocated for the events and the objects it has attached to.

Further to this, a single event can be listened for with **one()** or **$().one()**, where the event handler will be removed immediately after the event has been triggered for the first time.

9.4. Bubbling

As with typical DOM events, the DataTables custom events bubble up through the document, so you can listen for events using the delegate form of **$().on()** or **on** other elements which are higher up the DOM tree.

This can be useful, for example, to know when a new DataTable has been created, which can be listened for using the **init event** thus:

Example:

```
$(document).on( 'init.dt', function ( e, settings ) {
    var api = new $.fn.dataTable.Api( settings );

    console.log( 'New DataTable created:', api.table().node() );
} );
```

Similarly, this method could also be useful with the **xhr** event which will let you know what JSON data was returned from the server from the last DataTables initiated Ajax query.

A full list of the events that DataTables and its extensions can trigger is available in the event reference documentation. https://datatables.net/reference/event

DataTables	
`column-sizing`	Column sizing event - fired when the column widths are recalculated.
`column-visibility`	Column visibility event - fired when the visibility of a column changes.
`destroy`	Table destroy event - fired when a table is destroyed.
`draw`	Draw event - fired once the table has completed a draw.
`error`	Error event - An error has occurred during DataTables' processing of data.
`init`	Initialisation complete event - fired when DataTables has been fully initialised and data loaded.
`length`	Page length change event - fired when the page length is changed.
`order`	order event - fired when the data contained in the table is ordered.
`page`	Page change event - fired when the table's paging is updated.
`preDraw`	Pre-draw event - triggered as the table is about to be redrawn.
`preInit`	Initialisation started event - triggered immediately before data load.
`preXhr`	Ajax event - fired before an Ajax request is made
`processing`	Processing event - fired when DataTables is processing data
`search`	Search event - fired when the table is filtered.
`stateLoaded`	State loaded event - fired once state has been loaded and applied.
`stateLoadParams`	State load event - fired when loading state from storage.
`stateSaveParams`	State save event - fired when saving table state information.
`xhr`	Ajax event - fired when an Ajax request is completed
AutoFill	
`autoFill`	An auto fill action has been completed
`preAutoFill`	An auto fill action is about to be applied to the table
Buttons	
`button-action`	A button's action method has been triggered
ColReorder	
`column-reorder`	Columns have been reordered by the end user or API
KeyTable	
`key-blur`	KeyTable has blurred focus from a cell
`key-focus`	KeyTable has focused a cell

DataTables	
`key-refocus`	KeyTable has refocused a cell
`key`	A key event has been detected on the table and is *not* handled by KeyTable
Responsive	
`responsive-display`	The details for a row have been displayed, updated or hidden
`responsive-resize`	The columns displayed by Responsive has changed due to a resize
RowGroup	
`rowgroup-datasrc`	Grouping data property has been changed
RowReorder	
`pre-row-reorder`	A row reordered action has been initiated by the end user.
`row-reorder`	Rows have been reordered by the end user
`row-reordered`	After rows have been reordered by the end user
Select	
`deselect`	Items (rows, columns or cells) have been deselected
`select`	Items (rows, columns or cells) have been selected
`selectItems`	Select items type change event.
`selectStyle`	Select style change event.
`user-select`	A user action will cause items to be selected in the table