14. DATATABLES Buttons Extension.

14.1. Buttons Extension.

A common UI paradigm to use with interactive tables is to present **buttons that will trigger some action** - that may be to alter the table's state, modify the data in the table, gather the data from the table or even to activate some external process. Showing such buttons is an interface that end users are comfortable with, making them feel at home with the table.

The ***Buttons*** library for DataTables provides a framework with common options and API that can be used with DataTables, but is also very extensible, recognising that you will likely want to use buttons which perform an action unique to your applications.

Buttons has four sets of plug-ins that are part of the core software - they are not built into the core, but rather than be included as and when you need them, selecting only the software you require. Other extensions such as Editor and Select also provide buttons for use with this library, with actions unique to their own behaviours. This ensures a consistent interface for the interactions performed with your tables.

14.2. Initialization

The ***Buttons*** library can be initialised and used in two different ways:

1. As part of the ***DataTables constructor*** with the buttons **configuration** option

2. A **new constructor**

It is important to note that **multiple instances of Buttons can be created for use with a DataTable**. This can be particularly useful if you want to present different sets of buttons to the end user - for example above and below the table.

In DataTables

As part of the *DataTables* constructor, the buttons option can be given as an **array of the buttons** you wish to show - this is typically just the button name, although you can provide options to customise the button's actions:

Example:

```
$('#myTable').DataTable( {
    buttons: [
        'copy', 'excel', 'pdf'
    ]
} );
```

When using this method of initialisation, you may also wish to use the **dom** option to tell DataTables where to display the buttons - see below. The **buttons** option can also be given as an object to provide more control over the behaviour of *Buttons*.

Constructor

Button instances can also be created using the Javascript new keyword with the ***$.fn.dataTable.Buttons*** function. This function takes two parameters:

1. The ***DataTable instance*** to apply the buttons to
2. The ***button options*** (this is the same as the options available for the buttons option).

Example:

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

new $.fn.dataTable.Buttons( table, {
    buttons: [
        'copy', 'excel', 'pdf'
    ]
} );
```

This method of initialisation is particularly useful for cases when you wish to present multiple button instances, since only a single instance can be created using the buttons option.

14.3. Displaying the buttons

With the *Buttons* instance created we still need to display the buttons somewhere on the page so the end user can interact with them! Once again, there are two ways of doing this:

1. ***dom*** - *DataTables' DOM control parameter* - this option is only available if you also use buttons. Additionally, if you use anything other than the DataTables default styling, you probably don't want to use this option!
2. **Direct insertion using the API**

dom parameter

DataTables has a number of table control elements available and where they are placed in the DOM (i.e. the order) is defined by the **dom** parameter. This parameter can be a little confusing at first, but simply put, each letter in it is a DataTables feature. For Buttons the **B** character is the letter to use:

Example:

```
$('#myTable').DataTable( {
    dom: 'Bfrtip',
    buttons: [
        'copy', 'excel', 'pdf'
    ]
} );
```

Direct insertion

If you are using one of the styling integration options, such as for **Bootstrap**, or you wish to have multiple button instances available, the ***buttons().container()*** method can be used to obtain a jQuery object that holds the container element of the button set. Thus you can then insert the element anywhere you want:

With **buttons**:

Example:

```
var table = $('#example').DataTable( {
    buttons: [
        'copy', 'excel', 'pdf'
    ]
} );

table.buttons().container()
    .appendTo( $('.col-sm-6:eq(0)', table.table().container() ) );
```

And with a **new constructor** (note how it also uses the DataTables API to obtain the buttons container):

Example:

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

new $.fn.dataTable.Buttons( table, {
```

```
          buttons: [
              'copy', 'excel', 'pdf'
          ]
      } );

      table.buttons().container()
          .appendTo( $('.col-sm-6:eq(0)', table.table().container() ) );
```

14.4. Features

Buttons provides the following features:

- Common interface and framework for DataTables related buttons
- Buttons can be activated with assignable key combinations
- Comprehensive API
- Fully internationalisable
- HTML5 export options for modern browsers
- Flash export options for legacy browsers
- Column visibility control
- Print view