06. DATATABLES Options.

6.1. Options

DataTables' huge range of options can be used to customise the way that it will present its interface, and the features available, to the end user. This is done through its **configuration options**, which are set **at initialisation time**.

The **DataTables extensions** also each provide **their own options** that can be set in the DataTables configuration object.

6.2. Setting options

Configuration of DataTables is done by passing the options you want defined into the DataTables constructor as an object. For example:

Example:

```
$('#example').DataTable( {
    paging: false
} );
```

This uses the paging option to disable paging for the table.

Let's say you want to **enable scrolling in the table** - you would use the **scrollY** option:

Example:

```
$('#example').DataTable( {
    scrollY: 400
} );
```

Extending that, you can combine multiple options into a single object. In this case we **enable scrolling and disable paging**:

Example:

```
$('#example').DataTable( {
    paging: false,
    scrollY: 400
} );
```

The object being passed in is just a standard Javascript object and can be treated as such. Add as many options as you wish!

For the full range of configuration options available for DataTables, please refer to the options reference section of this web-site.

https://datatables.net/reference/option/

6.3. HTML 5 data attributes

As of v1.10.5 DataTables can also use initialisation options read from **HTML5 data-* attributes**. This provides a mechanism for setting options directly in your HTML, rather than using Javascript (as above). Consider for example the following table:

Example:

```
<table data-order='[[ 1, "asc" ]]' data-page-length='25'>
    <thead>
        <tr>
            <th>Name</th>
            <th>Position</th>
            <th>Office</th>
            <th>Age</th>
            <th>Start date</th>
            <th data-class-name="priority">Salary</th>
        </tr>
    </thead>
</table>
```

When DataTables is initialised on this table it will set **pageLength** to 25, order by the second column automatically (*order*) and set a class name on the final column of the table (*columns.className*).

There are two important points to consider when using **data-*** attributes as initialisation options:

- jQuery will automatically convert from dashed strings to the camel case notation used by DataTables (*e.g. use data-page-length for pageLength*).

- If using a string inside the attribute it must be in double quotes (*and therefore the attribute as a whole in single quotes*). This is another requirement of jQuery's due to the processing of JSON data.

6.4. Common options

Some of the options you might find particularly useful are:

- **ajax** - Ajax data source configuration
- **data** - Javascript sourced data
- **serverSide** - Enable server-side processing
- **columns.data** - Data source options for a column
- **scrollX** - Horizontal scrolling
- **scrollY** - Vertical scrolling

Full list of options: https://datatables.net/reference/option

6.5. Setting defaults

When on projects that use multiple DataTables it is often useful to set the initialisation defaults to common values (*for example you might want to set the dom option to a common value so all tables get the same layout*). This can be done using the **$.fn.dataTable.defaults** object. This object takes all of the same parameters as the DataTables **initialisation** object, but in this case you are setting the default for all future initialisations of DataTables.

In this example we disable the searching and ordering features of DataTables by default, but when the table is initialised, it is initialised with ordering enabled (*overriding the default*).

Example:

```
// Disable search and ordering by default
$.extend( $.fn.dataTable.defaults, {
    searching: false,
    ordering:  false
} );

// For this specific table we are going to enable ordering
// (searching is still disabled)
$('#example').DataTable( {
    ordering: true
} );
```

6.6. Extensions

Many of DataTables extensions can also be configured through the DataTables configuration object, initialising the extension and configuring it as required. The properties available depend upon the extensions used, and the extension Javascript must be loaded in order for those options to operate.

For example, consider the **Select** extension which provides the end user with the ability to dynamically select rows, columns and cells in the table. It presents the select option which can be set to true to enable selection:

https://datatables.net/extensions/select

Example:

```
$('#myTable').DataTable( {
    select: true
} );
```

The select option can also be given as an object to give fine grained control over the selection options and of course can be combined with the other DataTables options.

The **options reference** provides a searchable list of all options from DataTables and the extensions.

https://datatables.net/reference/option