13. DATATABLES Select Extension.

13.1. Select Extension.

***Select* adds item selection capabilities to a DataTable**. Items can be **rows**, **columns** or **cells**, which can be selected independently, or together. Item selection can be particularly useful in interactive tables where users can perform some action on the table, such as editing rows or marking items to perform an action on.

Item selection can be performed by the end user in one of three different modes:

- ***Operating system (os) style*** - where a single click will select a single item only, deselecting any others that were previously selected, a shift click will select a range of items and a ctrl / cmd click will add and remove items from the selection.
- ***Single item select (single)*** - only a single item can be selected at any one time. Any previously selected items will be deselected.
- ***Multi-item selection (multi)*** - item selection is toggled with a single click.

It is also possible to select items through the DataTables API (***rows().select(), columns().select() and cells().select()***).

Select also triggers a number of **events** letting you know when items have been selected and deselected, and has built-in button types for the **Buttons** extension allowing select all / none and other actions to be performed very easily.

13.2. Initialization

If you include the ***Select* extension** on your page, it is automatically available for every *DataTable* on that page through the API - no additional initialisation is required. If you would like to make item selection possible for your end users through actions such as click and tap, the **select** option can be set to boolean **true**:

Example:

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

This will automatically enable the operating system style row selection. Other styles and item selection options, including which columns in the table will trigger item selection, can be enabled using the select option as an object. Please refer to the reference documentation for full details of the options available.

13.3. Features

Select adds the following features:

- Select rows, columns and cells in a DataTable
- Tight integration with DataTables' API
- Full integration with the Buttons extension
- Multiple selection styles
- Checkbox column selection option
- Fully internationalisable
- Full integration with Bootstrap

13.4. API integration

Select presents an interface to the end user to allow item selection by clicking on table elements. This is all the end user really needs to know about the software, but it is only half the story! You, the programmer, need to know when items have been selected and you need to able to retrieve those items to process that selection! This is done through the **DataTables API** which Select augments to make these manipulations possible.

Selected items retrieval

DataTables has three primary methods for obtaining items from a DataTable (and three matching singular items):

- **rows**() and **row**() to select rows
- **columns**() and **column**() to select columns
- **cells**() and **cell**() to select cells

Each of these methods has their own options that can be used to select specific items, but they all provide the option to pass in a **selector-modifier** option. This option is used to tell the API how to order and filter the items that can be selected from. Select augments this by adding a selected option to the **selector-modifier** object. When true it will return the selected items, when false it will return the items which are not selected. If undefined it will not perform any actions.

Consider for example the following:

Example:

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

table.rows( { selected: true } ).data();
```

This will retrieve the data (using rows().data()) for any rows that are selected. We can extend that with the other selector-modifier options if required:

Example:

```
table
    .rows( {
        selected: true,
        page: 'current'
    } )
    .nodes();
```

In this case the row nodes (***rows().nodes()***) are returned for the selected items on the current page.

This can of course be combined with a regular **row** selector:

Example:

```
table.rows( '.important', { selected: true } ).data();
```

will get the data for any rows which are selected and have the class important.

Similarly we can perform similar actions with the other table selection API methods - for example:

Example:

```
table.cells( { selected: true } ).data();
```

to get the cells that are selected if using the cell selection options.

While working with the API and selected rows it is worth noting that the ***any***() and ***count***() helper methods can be useful to know if and how many items are selected.

Item selection

As well as being able to retrieve the selected items using the DataTables API, it is also possible to select and deselect items using the API. This is done using the following plural functions and their singular counterparts:

- ***rows().select() / rows().deselect()***
- ***columns().select() / columns().deselect()***
- ***cells().select() / cells().deselect()***

For example, to select all rows with the class important use:

Example:

```
table.rows( '.important' ).select();
```

This can also be used to implement your own item selection interface if you prefer instead of using the options made available by *Select*. For example consider the following:

Example:

```
$('#myTable').on( 'click', 'tbody tr', function () {
    if ( table.row( this, { selected: true } ).any() ) {
        table.row( this ).deselect();
    }
    else {
        table.row( this ).select();
    }
} );
```

This code will toggle the selection of rows in the table. Line by line:

- Line 1 - Attach a delegated event listener to the rows in the table
- Line 2 - Check if the row is selected
- Line 3 - If it is selected, deselect it
- Line 6 - If not selected, select it

Events

The final piece of the puzzle is to to know when items are selected. This is done using events that are emitted by *Select* - specifically the ***select and deselect events***. These can be listened for using the ***standard DataTables event model***.

When using *Select* it is also quite natural to use the **Buttons** extension for DataTables - allowing the user to select items and then click a button to trigger an action on those items.

Select provides the ***selected*** and ***selectedSingle*** buttons for use with Buttons, which can be used to easily provide your own actions when items are selected. These buttons are automatically deactivated when there are no items selected (in the case of *selectedSingle* when there is more than one item selected as well) and activated when items are selected.

Consider for example the following which uses the selected button to simply tell you how many rows are selected:

Example:

```
$('#myTable').DataTable( {
    select: true,
    buttons: [
        {
            extend: 'selected',
            action: function ( e, dt, node, config ) {
                var rows = dt.rows( { selected: true } ).count();

                alert( 'There are '+rows+'(s) selected in table' );
            }
        }
    ]
} );
```