03. DATATABLES Orthogonal Data.

3.1. Data

Data is complex. Not only will the data displayed in your tables have rules relating to how each data point corresponds to the others in your table, but also each data point itself can take multiple forms. Consider for example currency data; for display it might be shown formatted with a currency sign and thousand separators, while sorting should happen numerically and searching on the data will accept either form. In DataTables we term this **orthogonal data**.

DataTables has four built-in data operations, each of which can potentially use an orthogonal (independent) data source. These four operations are:

- ***display*** (String) - Display data.
- ***sort*** (String) - Data used for ordering.
- ***filter*** (String) - Data used for searching.
- ***type*** (String) - Type detection data.

By default DataTables will use the same **data** for all four operations, but this can easily be modified using the **data** and **columns.render** initialisation option, or HTML 5 **data-*** attributes.

3.2. Data Source

Orthogonal data for a table can be provided through the **data source array / object** (note, objects are much easier to work with, since you don't need to remember array indexes!) as predefined values (typically this is done with ajax loaded data or a Javascript provided data source), or they can be computed on-the-fly as they are needed.

Predefined values

If your data source has the orthogonal data that you wish to display already available in it, DataTables can make use of this directly by providing the **columns.data** and / or **columns.render** options as objects.

Consider for example the following **date** structure - note that the **start_date** object has a nicely formatted display property which will be used for display of the data in the table, but also a **timestamp** property. This is useful in this case since the format selected for the date display is not easily sortable:

Example:

```
{
    "name":       "Tiger Nixon",
    "position":   "System Architect",
    "start_date": {
        "display": "Mon 25th Apr 11",
        "timestamp": "1303682400"
    },
    "office":     "Edinburgh"
}
```

To use the data in this format in the table we might use the following options for a **columns** or **columnDefs** column description:

Example:

```
{
    data: 'start_date',
    render: {
        _: 'display',
        sort: 'timestamp'
    }
}
```

Note that the **_ property** must be defined when using **columns.data** or **columns.render** as an object. The _ property is the '*fallback*' if a data option has not been defined (there is no display option in the above object for example).

Example: JavaScript / HTML / AJAX

```
$(document).ready(function() {
    $('#example').DataTable( {
        ajax: "data/orthogonal.txt",
        columns: [
            { data: "name" },
            { data: "position" },
            { data: "office" },
            { data: "extn" },
            { data: {
                _:    "start_date.display",
                sort: "start_date.timestamp"
```

```
                } },
                { data: "salary" }
            ]
        } );
    } );
```

```
    <table id="example" class="display" style="width:100%">
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Position</th>
                    <th>Office</th>
                    <th>Extn.</th>
                    <th>Start date</th>
                    <th>Salary</th>
                </tr>
            </thead>
            <tfoot>
                <tr>
                    <th>Name</th>
                    <th>Position</th>
                    <th>Office</th>
                    <th>Extn.</th>
                    <th>Start date</th>
                    <th>Salary</th>
                </tr>
            </tfoot>
        </table>
```

```
{
  "data": [
    {
      "name": "Tiger Nixon",
      "position": "System Architect",
      "salary": "$320,800",
      "start_date": {
        "display": "Mon 25th Apr 11",
        "timestamp": "1303689600"
      },
      "office": "Edinburgh",
      "extn": "5421"
    },
    …
  ]
}
```

Computed values

If your data source does not contain pre-formatted orthogonal data, the **columns.data** and **columns.render** options can be provided as functions. These functions would be used to compute the data required for display.

For example, consider the following data structure:

Example:

```
{
    "name":       "Tiger Nixon",
    "position":   "System Architect",
    "start_date": "1303682400",
    "office":     "Edinburgh"
}
```

Although the start date is in a nice format for the computer to use, it is virtually useless for humans. To resolve this we can compute the value to be displayed using **columns.render** as a function - the **columns.data** option tells the renderer what data to use:

Example:

```
{
    data: 'start_date',
    render: function ( data, type, row ) {
        if ( type === 'display' || type === 'filter' ) {
            var d = new Date( data * 1000 );
            return d.getDate() +'-'+ (d.getMonth()+1) +'-'+ d.getFullYear();
        }
        return data;
    }
}
```

3.3. HTML 5

The above approach of using orthogonal data from a data source is great if you are ajax loading the data, but it isn't so useful if your table already exists in HTML. For this, DataTables supports **data-* attributes**, which can be used to hold information visible in the **DOM**, but not to the end user.

DataTables will automatically detect the following attributes on HTML cells:

- **data-sort** and **data-order** - for ordering data
- **data-filter** or **data-search** - for search data

For example, consider the following HTML table row:

Example:

```
<tr>
    <td data-search="Tiger Nixon">T. Nixon</td>
    <td>System Architect</td>
    <td>Edinburgh</td>
    <td>61</td>
    <td data-order="1303682400">Mon 25th Apr 11</td>
    <td data-order="3120">$3,120/m</td>
</tr>
```

Note that:

- The first column has a **data-search** attribute, allowing for a full name search in this case, while the display shows an abbreviated form.
- The fifth column has a date which is not directly sortable, so **data-order** is used to provide the unix time representation for ordering.
- The sixth column also uses **data-order**, in this case to provide a numeric form of a formatted number.

In order for the HTML 5 **data-* attribute** detection and processing to work correctly, all cells in a column must have the same attribute available. Without this, DataTables will give a warning.

Example: JavaScript / HTML

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

```
<table id="example" class="display" style="width:100%">
        <thead>
            <tr>
                <th>Name</th>
                <th>Position</th>
                <th>Office</th>
                <th>Age</th>
                <th>Start date</th>
                <th>Salary</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td data-search="Tiger Nixon">T. Nixon</td>
                <td>System Architect</td>
                <td>Edinburgh</td>
                <td>61</td>
```

```
                    <td data-order="1303689600">Mon 25th Apr 11</td>
                    <td data-order="320800">$320,800/y</td>
                </tr>

                 …
               </tbody>
            <tfoot>
                <tr>
                    <th>Name</th>
                    <th>Position</th>
                    <th>Office</th>
                    <th>Age</th>
                    <th>Start date</th>
                    <th>Salary</th>
                </tr>
            </tfoot>
        </table>
```

3.4. API Interface

When DataTables reads data from an HTML table, by default it will **read the data in each row into an array** (although this can be customised with **columns.data**). When the software detects orthogonal HTML5 attributes, it reads the information into an object, allowing multiple data points for each cell (if required).

If orthogonal data exists for a cell, the content of the cell (i.e. what the user sees) will be copied into a **display** property. The HTML5 attributes are copied into **properties** with the same name as the attribute, but with an **@ prefix**.

If we continue the example from above, it will be read in using the following structure:

Example:

```
{
    "0": {
        "display": "T. Nixon",
        "@data-search": "Tiger Nixon"
    },
    "1": "System Architect",
    "2": "Edinburgh",
    "3": "61",
    "4": {
        "display": "Mon 25th Apr 11",
        "@data-order": "1303682400"
    },
    "5": {
        "display": "$3,120/m",
```

```
            "@data-order": "3120"
        }
    }
```

If you don't use the DataTables API to manipulate the data in the table, you'll never need to know the structure of the data. However, if you want to read the data back (e.g. using **row().data()**) or add a new row of data (**row.add()**) you need to make sure that you use the same data structure as DataTables read the data in as. Without that, you will get **Requested unknown parameter errors**.

To check the data structure DataTables is using for a row, use console.log(myTable.row(':eq(0)').data()); to show the data for the first row in the table.

Example:

```
console.log( myTable.row(':eq(0)').data() );
```