FREE eBook

LEARNING datatables

Free unaffiliated eBook created from
Stack Overflow contributors.

#datatables

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: datatables

It is an unofficial and free datatables ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official datatables.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with datatables

Remarks

Key features

- Variable length pagination
- On-the-fly filtering
- Multi-column sorting with data type detection
- Smart handling of column widths
- Display data from almost any data source
- DOM, JavaScript array, Ajax file and server-side processing
- Scrolling options for table viewport
- Fully internationalisable
- jQuery UI ThemeRoller support
- Wide variety of plug-ins
- It's free

Resources

- Usage
- Examples
- styling
- API
- Development
- Extras
- Plug-ins
- Blog
- FAQ's
- Forums
- Server-side examples

Versions

Version	Release notes	Release Date
1.9 and earlier (legacy)	https://datatables.net/forums/discussion/8332/datatables-1-9-0-released	2014-02-01
1.10 and later	https://datatables.net/new/1.10	2015-08-12

Examples

Installation

Have required `JavaScript` and `CSS` files included in your `index.html`. You can do this by either using the CDN files availiable at the following paths:

```
<link rel="stylesheet" type="text/css"
href="//cdn.datatables.net/1.10.12/css/jquery.dataTables.css">

<script type="text/javascript" charset="utf8"
src="//cdn.datatables.net/1.10.12/js/jquery.dataTables.js"></script>
```

Or by downloading individual local files and hosting them yourself. To get a comprehensive package of required `JavaScript` and `CSS` files visit the DataTables download builder which will allow you to pick and choose features you need and condense them into a single package (or offer individual files). Include these in the order displayed at the bottom of the page.

DataTables depends on `jQuery`, so include it before `jquery.dataTables.js`:

```
<script type="text/javascript" src="https://code.jquery.com/jquery-3.1.0.min.js"></script>
```

DataTables are also available through NPM

```
npm install datatables.net    # Core library
npm install datatables.net-dt # Styling
```

and Bower

```
bower install --save datatables.net
bower install --save datatables.net-dt
```

Initializing a minimal DataTables:

The below code will turn the table with an id of `tableid` into a DataTable, as well as return a DataTables API instance:

```
$(document).ready(function() {
    $('#tableid').DataTable();
});
```

Compare this to the below code, which will turn the table into a DataTable but will not return a DataTables API instance:

```
$(document).ready(function() {
    $('#tableid').dataTable();
});
```

See the DataTables API documentation section for more details on what can be done with the DataTables API instance.

Feature Enable/Disable (DataTables Options)

DataTables has the capability to enable or disable a number of its features, such as paging or searching. To choose these options, simply select them in your initialization:

```
$(document).ready(function() {
    $('#tableid').DataTable( {
        "paging":   false, //Turn off paging, all records on one page
        "ordering": false, //Turn off ordering of records
        "info":     false  //Turn off table information
    } );
} );
```

Note that the quotation marks around the option names are optional:

```
paging: false,
ordering: false,
info: false
```

Is also perfectly valid.

A full list of options can be found here, along with descriptions of the uses of each option.

These options can only be set once, when the table is initialised. However, you can work around this limitation by adding:

```
 destroy: true
```

DataTables API

DataTables comes with an extensive API which is used to manipulate or obtain information about the DataTables on a page.

The API can be accessed in 3 ways:

```
var table = $('#tableid').DataTable(); //DataTable() returns an API instance immediately
var table = $('#tableid').dataTable().api(); //dataTable() returns a jQuery object
var table = new $.fn.dataTable.Api('#tableid');
```

Once the object has been set, you can call any of the API functions on that object.

```
var columns = table.columns();
```

A more complex example is adding some rows to your table:

```
table.rows.add( [ {
        "name":       "John Doe",
        "employee_id":   "15135",
        "department":     "development",
    }, {
```

```
        "name":       "Jane Smith",
        "employee_id":   "57432",
        "department":     "quality assurance",
    } ] )
    .draw();
```

The full list of API functions can be found here.

Read Getting started with datatables online: https://riptutorial.com/datatables/topic/1844/getting-started-with-datatables

Chapter 2: Add export buttons to table in Bootstrap 4

Introduction

With the datatables plugin you can add export buttons to your table.

You can export your table data to excel, pdf or copy it to the clipboard.

This manual is intended for the bootstrap 4 framework.

Examples

Add buttons to table

In your JS File add this **option** to your datatable:

```
  buttons: [ 'excel', 'pdf', 'copy' ]
```

It will look like:

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

Add the necessary **css** files for the datatable with the buttons:

```
<link rel="stylesheet" type="text/css"
href="//cdn.datatables.net/1.10.15/css/dataTables.bootstrap4.min.css"/>
<link rel="stylesheet" type="text/css"
href="//cdn.datatables.net/buttons/1.3.1/css/buttons.bootstrap4.min.css"/>
```

Add the necessary **javascript** files for the datatable with the buttons:

```
<script type="text/javascript"
src="//cdn.datatables.net/1.10.15/js/jquery.dataTables.min.js"></script>
<script type="text/javascript"
src="//cdn.datatables.net/1.10.15/js/dataTables.bootstrap4.min.js"></script>
<script type="text/javascript"
src="//cdn.datatables.net/buttons/1.3.1/js/dataTables.buttons.min.js"></script>
<script type="text/javascript"
src="//cdn.datatables.net/buttons/1.3.1/js/buttons.bootstrap4.min.js"></script>

<script type="text/javascript"
src="//cdnjs.cloudflare.com/ajax/libs/jszip/3.1.3/jszip.min.js"></script>
<script type="text/javascript"
src="//cdn.rawgit.com/bpampuch/pdfmake/0.1.27/build/pdfmake.min.js"></script>
<script type="text/javascript"
```

```
src="//cdn.rawgit.com/bpampuch/pdfmake/0.1.27/build/vfs_fonts.js"></script>
<script type="text/javascript"
src="//cdn.datatables.net/buttons/1.3.1/js/buttons.html5.min.js"></script>
```

It will look like this picture:

If you doesn't see the buttons, add this option:

```
dom: 'Blfrtip',
```

to the datatable options list. So it looks like:

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

You will find more informations to define the table control elements to appear on the page and in what order on this page.

Note: The prerequisite is, that the jQuery and bootstrap4 files are installed in your project.

Read Add export buttons to table in Bootstrap 4 online:
https://riptutorial.com/datatables/topic/10149/add-export-buttons-to-table-in-bootstrap-4

Chapter 3: datatables - Show Selected Rows option

Examples

Show Selected Rows only

Its common for datatables to have a checkbox to select multiple rows. If the data is spread across multiple pages, it could be difficult for the user to view the records he selected. To enable the user view all the selected records in one go, we usually use a hyperlink that when clicked displays only the selected rows from the datatable. This link can be used a toggle between viewing selected records and all records.

```
$(sAnchor).click(function() {
                $('#show_selected').text(function(_,txt) {
                    var checked = 0;
                    var ret='';
                    var dataTable = $("#report_table").dataTable();
                    if ( txt == 'Show Selected Reports' ) {
                          dataTable.fnFilter('Checked',8);
                          ret = 'Show All Reports';
                    }else{
                          dataTable.fnFilter('',8);
                          ret = 'Show Selected Reports';
                    }
                    return ret;
                 });
 });
```

In the above method, on click of the Select All hyperlink, If the hyperlink div text is 'Show Selected Reports' we filter the datatable to display only those rows for which the checkbox is checked. We use the inbuilt Datatables API function **fnFilter**.

We pass 2 parameters to this method - query string and the index of the column to filter. In this case the value of the checkbox will be **'Checked'** if its selected on the UI and the index of the column containing the checkboxes is 8. Hence we are passing 'Checked'and 8 as pamaeters to the fnFilter function. After filter, we toggle the link to display 'Show All Reports.'

When the user clicks on Show All Reports, we pass an empty string to the fnFilter function as the query string. So it displays all the records.

Read datatables - Show Selected Rows option online:
https://riptutorial.com/datatables/topic/6805/datatables---show-selected-rows-option

Chapter 4: datatables search input box for a realtime search

Examples

Search input box for progressive search on the datatable

Below is an example for implementing a Search input box that helps users to search the occurances of a particular value across the datatable.

In the below example, #report is the div id of the div that contains the search input box. This function is called as soon as the user enters a value in this input box. Since there can be many occurances of a single character, we call the actual search function only when more than 1 character is entered in the search box.

```
$('#report').on('input',function(e){
     if($(this).data("lastval")!= $(this).val()){
     $(this).data("lastval",$(this).val());
        //change action
        if($('#report').val().length >1 ){
            searchTable($(this).val());
        }
     };
 });
```

In the searchTable function we use the inbuilt datatables function **fnFilter** to find the matching occurances of the input string. **We can restrict the search to a particular column by passing the column index.** Here we are passing the column index 2.

```
function searchTable(inputVal) {
    var dataTable = $("#report_table").dataTable();
    dataTable.fnFilter(inputVal,2);
}
```

If you need to search across all the columns **just make sure you do not pass the index parameter**.

```
function searchTable(inputVal) {
    var dataTable = $("#report_table").dataTable();
    dataTable.fnFilter(inputVal);
}
```

Read datatables search input box for a realtime search online:
https://riptutorial.com/datatables/topic/6806/datatables-search-input-box-for-a-realtime-search

Chapter 5: How to get the search value entered in Datatables programmatically?

Examples

Example

This is the code to filter the Datatables [1.10.7] by value programmatically, you can find it on official documentation.

```
function setFilterValue(datatable, value){
    if(datatable !== undefined){
        datatable
            .columns(0)
            .search(value)
            .draw();
    }
}
```

This is the code to get the value by the previous search.

```
function getFilterValue(datatable){
    var value;
    if(datatable !== undefined){
        value = datatable
                    .settings()[0]
                    .oSavedState
                    .columns[0]
                    .search.search;
    }
    return value;
}
```

This approach is useful when you have the cache active (*"stateSave": true*) and you need to know the previous search value after reloaded the page.

Read How to get the search value entered in Datatables programmatically? online: https://riptutorial.com/datatables/topic/6085/how-to-get-the-search-value-entered-in-datatables-programmatically-

Chapter 6: Migration from <1.10 to 1.10 and above

Introduction

Datatables 1.10.x is the latest release as of now. Though it is backwards compatible to the previous versions (1.9 etc.), it is highly advisable to use the latest version which directly returns a `datatable api` object. Another major change, that is the most visible, is the change from Hungarian Notation to camelCase

Syntax

- The only major syntax change is the usage of camelCase everywhere instead of the Hungarian Notation.
- A more detailed guide can be found here which involves the conversion of parameters in <1.10 to 1.10+

Remarks

For more details, try visiting the following pages:

1. Upgrading to Datatables 1.10
2. Changelog Datatables 1.10
3. Convert Parameter Names from 1.9 to 1.10
4. Using the Datatable API

Examples

Initialisation of Datatable 1.10+

Previously, datatables were initialized as follows:

```
var oTable = $("#selector").dataTable();
```

This used to return a jQuery object which would stored in the variable oTable. And then to access the api to modify table properties, we had to initialize the api differently, as shown below:

```
var api = oTable.api()
//r
var api = new $.fn.dataTable.api("#selector")
```

Now, the initialization is changed to the following:

```
var table = $("#selector").DataTable();
```

Note that this returns a `datatable api` instance and you can directly use variable `table` to manipulate datatable properties.

Features Not available in Datatables 1.10+

The following 3 features (that were deprecated in 1.9) are no longer available in 1.10, they are :

1. **fnRender**: According to the deverloper:

   The old fnRender option provided a method of manipulating a cell when it was created. however, it was provided with a confusing list of options as its arguments, and required a particular structure in DataTables internally that caused performance issues. Removal of fnRender has lead to a significant improvement in performance of DataTables with large data sets and the ability to provide object instances to DataTables as data source objects (for example Knockout observable objects).

Alternatives to `fnRender` are available as `columns.render` and `columns.createdCell`

2. **bScrollInfinite**: According to the developer:

   The built-in ability of DataTables 1.9 to show an infinitely scrolling grid through the bScrollInfinite option has been removed due to the inconsistencies it caused in the API. Removal has also helped simply the internal code significantly.

An extension that goes by the name of `Scroller` is available as an alternative.

3. **Cookie based state saving**:

   Cookie based state saving has been replaced with localStorage based state saving in DataTables 1.10. Cookie's, with their 4KiB limit were very limited, and incurred a performance penalty since they were part of every HTTP request. localStorage is much faster and more flexible, and is used as the default storage for state information in DataTables 1.10.

4. **`two_button` pagination control**:

   DataTables 1.10 has significantly upgraded the paging controls of DataTables (see pagingType), a consequence of which is that the old built-in two_button form of paging has been removed.

They have taken care of people who still want to use the `two_button` pagination method by providing an extra javascript file called `two_button.js`. Usage is as follows:

   Simply include this file in your document, after you load DataTables but before you initialise your table and the two_button pagination will be restored exactly as it was in 1.9 (including class names etc).

Read Migration from <1.10 to 1.10 and above online:

https://riptutorial.com/datatables/topic/8369/migration-from--1-10-to-1-10-and-above

Chapter 7: Server Side Data Processing

Examples

Load data using ajax with server-side processing.

```
var MY_AJAX_ACTION_URL = "path/to/controller.php";

var table = $('#user_list_table').DataTable({
       "autoWidth": true,
       "paging": true,
       "searching": true,
       "ordering": true,
       "language": {
         "zeroRecords": "No data Found",
         "processing": 'Loading'
       },
       "info": false,
       "stripeClasses": [ "odd nutzer_tr", "even nutzer_tr"],
       "columns": [
           {'data':'uid',"visible": false},
           {'data':'name','orderable': true},
           {'data':'phone','orderable': true},
           {'data':'email','orderable': true },
           {'data':'address','orderable': true}
       ],
       "order": [[ 1, "desc" ]],
       "processing": true,
       "serverSide": true,
       "ajax":MY_AJAX_ACTION_URL
   });
```

The response of the call to `MY_AJAX_ACTION_URL` should be strictly in the below format:

```
{
  "draw": 1,
  "recordsTotal": 2,
  "recordsFiltered": 2,
  "data": [
      {"name":"XYZ","phone":"678654454","email":"xyz@gmail.com","address":"true"},
      {"name":"ABC","phone":"678654455","email":"abc@gmail.com","address":"true"}
  ]
}
```

Note that if the output of the call fails to match the above format, it will result in an error in the initialization of `table`.

DataTables 1.10+ Serverside Processing

Example Table

There are several ways to inject your data into DataTables. Serverside Processing is just one

method. In this manner, DataTables has a pre-configured endpoint to retrieve data from, and that endpoint is responsible for accepting all paging/filtering/sorting requests that DataTables applies. There are a variety of pros and cons to this versus sending your complete dataset back from the server and letting DataTables do it all client-side depending on your use case.

```
var tbl = $('#example').DataTable({
  processing: true,
  serverSide: true,
  ajax: {
    url: '/echo/json/',
    method: 'post'
  },
  columns: [{
    data: 'First',
    title: 'First Name'
  }, {
    data: 'Last',
    title: 'Last Name'
  }]
});
```

Example of hooking preXhr event to send additional data to ajax request

This event fires directly before the ajax call is made allowing you to modify the request body. This is useful if there is a form that influences the data returned to the table (envision maybe a min/max bar to filter within a date range or similar).

```
tbl.on('preXhr.dt', function(ev, settings, data) {
    $.extend(data, {
      min: $('form [name=min]').val(),
      max: $('form [name=max]').val()
    });
});
```

Explanation of server side requirements for processing request

In a typical no-option instance of DataTables, all filtering, sorting, and paging is handled in the client browser. With Serverside Processing enabled, these tasks are shifted to the webserver. For very large datasets which may be inefficient to send in their entirety to the client, this can help.

There are several default parameters which are sent by the Datatables request when you configure an ajax endpoint. It can get very long, so rather than itemizing each, a general overview of the server responsibilities may be more helpful.

Starting with any optional parameters you may have supplied to the request, compile your dataset and prepare it for DataTables operations. Use the `search[value]` and `search[regex]` parameters to apply any filtering across all columns/properties. There are also `columns[i][search][value]` and `columns[i][search][regex]` parameters for individual column filters, though these are not commonly used in simple instances of DataTables.

Sort your filtered data using the various `order` parameters. As with the search parameters, there will be a set of ordering parameters per column on which sorting is enabled. `order[i][column]` and `order[i][column][dir]`

Once filtering and sorting is complete, it's time to page the data based on the DataTables request using the `start` and `length` parameters. In .NET this could look like:

```
int start = Int32.TryParse(Request["start"]);
int length = Int32.TryParse(Request["length"]);
return MyData.Skip(start).Take(length);
```

Example Response

This is the rough response structure DataTables expects. Whatever your data is (2d array, array of objects, etc) is nested in the `data` property with the other properties seeding DataTable's core to draw information about paging and filtering (e.g. "Showing records 11-20 of 500 (filtered from 1000)")

```
{
  "draw": 1,
  "recordsTotal": 57,
  "recordsFiltered": 57,
  "data": [/*your data goes here*/]
}
```

Also see `.rows.add(data)` and the `data` option for alternate methods of setting the contents of your DataTable using JSON data. Of course DataTables can also be initialized from static HTML or HTML5 `data-` attributes

Get JSON data from MySQL table

On the official website of DataTable is an example of how a server-side process with PHP and MySQL can look. This example is **deprecated** and can no longer be used with PHP 7 (the function "mysql_pconnect" and the associated functions are deprecated, see this post).

So this function gives you a wellformed JSON data as response:

```
<?php
//include database connection file
$servername = "localhost";
$username = "root";
$password = "root";
$dbname = "name";

$conn = mysqli_connect($servername, $username, $password, $dbname) or die("Connection failed:
" . mysqli_connect_error());

/* check connection */
if (mysqli_connect_errno()) {
  printf("Connect failed: %s\n", mysqli_connect_error());
  exit();
}

// initilize all variable
$params = $columns = $totalRecords = $data = array();
$params = $_REQUEST;
//define index of column name
```

```
$columns = array(
    0 =>'id',
    1 =>'name',
    2 =>'salery',
);

$where = $sqlTot = $sqlRec = "";

// check search value exist
if( !empty($params['search']['value']) ) {
    $where .=" WHERE ";
    $where .=" ( id LIKE '".$params['search']['value']."%' ";
    $where .=" OR name LIKE '".$params['search']['value']."%' ";
    $where .=" OR salery LIKE '".$params['search']['value']."%' )";
}

// getting total number records without any search
$sql = "SELECT * FROM `employees` ";
$sqlTot .= $sql;
$sqlRec .= $sql;

//concatenate search sql if value exist
if(isset($where) && $where != '') {
    $sqlTot .= $where;
    $sqlRec .= $where;
}

 $sqlRec .=  " ORDER BY ". $columns[$params['order'][0]['column']]."
".$params['order'][0]['dir']."  LIMIT ".$params['start']." ,".$params['length']." ";

$queryTot = mysqli_query($conn, $sqlTot) or die("database error:". mysqli_error($conn));

$totalRecords = mysqli_num_rows($queryTot);

$queryRecords = mysqli_query($conn, $sqlRec) or die("error to fetch employees data");

while( $row = mysqli_fetch_row($queryRecords) ) {
    $data[] = $row;
}

$json_data = array(
        "draw"            => intval( $params['draw'] ),
        "recordsTotal"    => intval( $totalRecords ),
        "recordsFiltered" => intval($totalRecords),
        "data"            => $data   // total data array
        );

echo json_encode($json_data);  // send data as json format
?>
```

The response looks like and can then be processed by the DataTable:

```
{
  "draw": 1,
  "recordsTotal": 3,
  "recordsFiltered": 2,
  "data": [
      {
        "id":"1",
        "name":"Jim",
```

```
        "salery":"1000"
      },
      {
        "id":"2",
        "name":"Claudia",
        "salery":"3000"
      },
      {
        "id":"3",
        "name":"Tommy",
        "salery":"2000"
      }
  ]
}
```

Read Server Side Data Processing online: https://riptutorial.com/datatables/topic/4176/server-side-data-processing

Credits

S. No	Chapters	Contributors
1	Getting started with datatables	andialles, Andrei Zhytkevich, Chris H., Community, cookypuss
2	Add export buttons to table in Bootstrap 4	Martin
3	datatables - Show Selected Rows option	saikris
4	datatables search input box for a realtime search	saikris
5	How to get the search value entered in Datatables programmatically?	Domenico Campagnolo
6	Migration from <1.10 to 1.10 and above	philantrovert
7	Server Side Data Processing	Aswathy S, BLSully, Martin, philantrovert