FathGrid

developer documentation

www.fathsoft.com

Contents

Installation

Just add fathgrid.js to your page HEAD section

```
<script src="fathgrid.js"></script>
```

Initialization

First, make sure you a have a table with an ID attribute in your HTML:

```
<table  id="table1" class="table table-hover table-bordered ">
  <thead class="thead-light"></thead>
  <tbody></tbody>
</table>
```

To initialize FathGrid object in place of a table, use JavaScript code:

```
var t1=FathGrid("table1",{ …configuration_options });
```

Configuration options

Configuration options is an object set as second parameter to FathGrid initialization call. It can contain the following fields:

Field	***Description***	***Default***
size	page size	10
page	page number	1
filterable	allow filters in table header	true
editable	allow editing when clicking on cells	false
pageable	allow paginator	true
sortable	allow sorting by clicking on column headers	true
showFooter	show table footer	false
showGroupFooter	show group footer	false
data	data array	[]
rowClass	function (item, idx) which returns string of class names to add to each TR element in table body	
onChange	function to be called whenever cell data is changed	function(item, col, oldval, newval)
groupOn	function (item, index) which returns grouping string	function(item. idx){}
sortBy	array of column indices for fixed filter	[]
columns	array of column configuration objects	[]

serverURL	templated URL to retrieve JSON data from	null
prepareData	custom async function which converts received JSON object into data array	async (json)=> json
loading	message to show while loading data	'Loading…'
rtl	set to true for right-to-left languages	false
selectColumns	allow hiding/showing columns	false
onInitFilter	function which is called when filter row is created. Useful for customizing appearance of filter controls	function(tr_elment){}
onInitTable	function which is called when table body is rendered. Useful for customizing appearance	function(tbody_elment){}
onInitInput	function called after table cell editor is initialized. Function parameters are: - **item** : data item - **idx** : column index - **el** : TD cell element in which input is located	function(data_item, column_index, TDElement){}
template	Templated HTML string for grid wrapper element. Use this to insert custom HTML between grid elements.	{tools}{info}{graph}{table}{pager}
graphType	graph type - line, bar, pie, etc.	line

graphValues	function which returns an object to initialize data graph	{
		title: 'Graph Title',
		labels:
		['Jan','Feb','Mar',...],
		values:
		[100,200,300,...]
		}

Column configuration options

Column configuration object can contain the following fields:

Field	*Description*	*Default*
name	field name	
visible	if false, hide this column	true
filter	list-of-values for filter or **null** to automatically create list from table data	undefined
value	function(item) which returns text to display in column	
html	function(item) which return HTML for column cell	
header	text to show in column header	
footer	function(items, element) which returns text to display in footer cell	
groupFooter	expression or a function(items) which returns grouping value	
editable	allow cell editing when clicked	false
type	HTML input type for cell editor	text
pattern	HTML validation pattern for input element	
title	input title or a message to show when editing	
listOfValues	array of values to choose from when editing cell value	
class	column class string. Useful to add text-alignment to the column	

Grid data

Table data can be loaded from:

1. Inline HTML table
2. Dynamic JavaScript array
3. Server-side API endpoint

Inline data

Simply put data in table HTML in TBODY TR TD elements:

```
    <table  id="table1" class="table table-hover table-bordered ">
      <thead class="thead-light">
        <tr><th>Code</th><th>Description</th><th>Qty</th><th>Price</th></tr>

      </thead>
      <tbody>
        <tr><td>100</td><td>first desciption</td><td>1</td><td>30</td></tr>
        <tr><td>200</td><td>other description</td><td>2</td><td>20</td></tr>
        <tr><td>300</td><td>some text</td><td>3</td><td>10</td></tr>
      </tbody>
    </table>
```

Dynamic data

Configuration option “data” allows setting data from JS array:

```
var demodata=[
  ['100','Service title 100', 2, 134],
  ['120','Some stuff', 10, 160],
  ['310','Goods', 5, 2000],
];
var t1=FathGrid("table1",{
      data: demodata
};
```

Server-side data

To load data from server, set **serverURL** configuration option to string template:

```
t1=FathGrid("table1",{
      serverURL:'https://jsonplaceholder.typicode.com/posts?_page=${page}&_lim
it=${size}&_sort=${sort}&_order=${order}&q=${search}&${filters}',
```

note that URL string has these placeholders:

- ${size} replaced with page size setting
- ${page} page number (1,2,3…)
- ${sort} sort field names
- ${order} order of sort, “asc” or “desc” for sort fields
- ${search} search term (if any)
- ${filters} replaced with filer name/value pairs (eg: name=test&price=1)

API

FathGrid object has the following methods:

Method	***Parameters***	***Description***

getData		returns read-only copy of table data
setData	**data**:array	sets new data
export	**format**: 'txt', 'csv', 'xls','pdf'… **filename**: downloaded file name	downloads data-export file
render		re-draws the table
sort	**column_index**: 1-based column **desc**: true for descending order **multi**: true for multicolumn sort	sort data by field
getSort		returns array of field indices by which data is sorted
setSort	**new_sort**: array of column indices	sets sort fields
filter	**column_index**: column **value**: filter value	sets data filtering
getFilter		returns current filter
editCell	**rownum**: 1-based row index **col**: 1-based column index	starts editor in cell
search	**q**: search term	sets instant search term
getSelectedItem		returns data item from currently selected row
deleteRow	**rownum**: 1-based row index	deletes a row
insertRow	**rownum**: 1-based row index **item**: new item data	insert new row
setServerURL	**url**: string	set new server URL
wrapperEl		returns FathGrid wrapper HTML Element
selectRow	**rownum**: 1-based row index	selects specified row
showSubgrid	**subgrid**: details FathGrid **html**: custom HTML to add before sub-grid	insert sub grid below currently selected row

How to...

Display calculated columns

See column configuration **value** and **html** options.

Example which calculates “item total” from “qty” and “price” columns on each row:

```
var t1=FathGrid("table1",{
...
columns:[
  {
    header:'Item Total',
    editable:false,
    value:(x)=>`${(parseFloat(x.qty||0)*parseFloat(x.price||0)).toFixed(2)}`,
    ...
```

Group by expression

See **groupOn** and **showGroupFooter** configuration options.

Example:

```
      //grouping
      sortBy:[1],//group on first column
      groupOn:item=>`${item[0]}`, //group expression
      showGroupFooter:true,

      selectColumns:true,
      columns:[
        {visible:false}, //invisible grouping column
        {
          groupFooter:(data)=>`${data.length} songs`,
        },
```

Add column footer

See **footer** column configuration option.

Example which adds footer with total sum for “item total” column:

```
var t1=FathGrid("table1",{
      ...
      columns:[
        {
         header:'Item Total',
         editable:false,
         value:(x)=>`${x.qty*x.price}`,
         footer:(data,el)=>
```

```
          data.map(x=>(x.qty*x.price))).reduce((x,s)=>x+s,0).toFixed(2)
       },
       ...
```

Change appearance of some rows

See **rowClass** config option.

Example from songs.html sample page:

```
t1=FathGrid("table1",{
    ...
    rowClass: (item,idx)=>
       item[3]=='The Beatles'?
        'table-info' :
        (item[3]=="The Rolling Stones"?
          'table-warning':
          ''
        ),
     ...
```

Editing

If config.editable is true, call **editCell(row,col)** API function to start editor.

Shift+arrow keys move editor across table. Columns with **editable**=false option set are skipped.

Insert new row

Call **insertRow** API function.

Delete row

Call **deleteRow** API function.

Show sub-grid

First, create second FathGrid object which will hold sub-grid data, then, call **showSubgrid** on main grid to insert details grid under currently selected row.