


# JSDataSet

 Apr 09, 2021 22:33

## Supported Clients

SmartClient WebClient NGClient

## Property Summary

[Number](#) `rowIndex` Get or set the record index of the dataset.

## Methods Summary

<a href="#">Boolean</a>	<code>addColumn(name)</code>	adds a column with the specified name to the dataset.
<a href="#">Boolean</a>	<code>addColumn(name, index)</code>	adds a column with the specified name to the dataset.
<a href="#">Boolean</a>	<code>addColumn(name, index, type)</code>	adds a column with the specified name to the dataset.
<a href="#">void</a>	<code>addHTMLProperty(row, col, name, value)</code>	Add an HTML property to an HTML tag produced in <code>getAsHTML()</code> .
<a href="#">void</a>	<code>addRow(index, array)</code>	Add a row to the dataset.
<a href="#">void</a>	<code>addRow(array)</code>	Add a row to the dataset.
<a href="#">String</a>	<code>createDataSource(name)</code>	Create a datasource from the data set with specified name and using specified types.
<a href="#">String</a>	<code>createDataSource(name, types)</code>	Create a datasource from the data set with specified name and using specified types.
<a href="#">String</a>	<code>createDataSource(name, types, pkNames)</code>	Create a datasource from the data set with specified name and using specified types.
<a href="#">String</a>	<code>getAsHTML()</code>	Get the dataset as an html table, do not escape values or spaces, no <code>multi_line_markup</code> , do not add indentation, add column names.
<a href="#">String</a>	<code>getAsHTML(escape_values)</code>	Get the dataset as an html table, do not escape spaces, no <code>multi_line_markup</code> , do not add indentation, add column names.
<a href="#">String</a>	<code>getAsHTML(escape_values, escape_spaces)</code>	Get the dataset as an html table, no <code>multi_line_markup</code> , do not add indentation, add column names.
<a href="#">String</a>	<code>getAsHTML(escape_values, escape_spaces, multi_line_markup)</code>	Get the dataset as an html table, do not add indentation, add column names.
<a href="#">String</a>	<code>getAsHTML(escape_values, escape_spaces, multi_line_markup, pretty_indent)</code>	Get the dataset as an html table, add column names.
<a href="#">String</a>	<code>getAsHTML(escape_values, escape_spaces, multi_line_markup, pretty_indent, add_column_names)</code>	Get the dataset as an html table.
<a href="#">String</a>	<code>getAsText(column_separator, row_separator, value_delimiter, add_column_names)</code>	Get the dataset as formatted text.
<a href="#">Array</a>	<code>getColumnAsArray(index)</code>	Get the column data of a dataset as an Array.
<a href="#">String</a>	<code>getColumnName(index)</code>	Get a column name based on index.
<a href="#">Array</a>	<code>getColumnNames()</code>	Get the column names of a dataset.
<a href="#">Number</a>	<code>getColumnType(index)</code>	Get a column type based on index.
<a href="#">Exception</a>	<code>getException()</code>	Get the database exception if an error occurred.
<a href="#">Number</a>	<code>getMaxColumnIndex()</code>	Get the number of columns in the dataset.
<a href="#">Number</a>	<code>getMaxRowIndex()</code>	Get the number of rows in the dataset.
<a href="#">Array</a>	<code>getRowAsArray(index)</code>	Get the row data of a dataset as an Array.
<a href="#">Object</a>	<code>getValue(row, col)</code>	Get the value specified by row and column position from the dataset.
<a href="#">Boolean</a>	<code>hadMoreData()</code>	Return true if there is more data in the resultset then specified by <code>maxReturnedRows</code> at query time.
<a href="#">Boolean</a>	<code>removeColumn(index)</code>	Remove a column by index from the dataset.
<a href="#">void</a>	<code>removeRow(row)</code>	Remove a row from the dataset.
<a href="#">void</a>	<code>setColumnName(index, columnName)</code>	Set a column name based on index.
<a href="#">void</a>	<code>setValue(row, col, obj)</code>	Set the value specified by row and column position from the dataset.
<a href="#">void</a>	<code>sort(col, sort_direction)</code>	Sort the dataset on the given column (1-based) in ascending or descending.
<a href="#">void</a>	<code>sort(comparator)</code>	Sort the dataset using the function as comparator.

## Property Details

### `rowIndex`

Get or set the record index of the dataset.

### Returns

[Number](#)

### Supported Clients

SmartClient,WebClient,NGClient

**Sample**

```
//assuming the variable dataset contains a dataset
//to set the rowIndex:
dataset.rowIndex = 1 //sets the rowIndex to the first row (dataset is 1-based)
//to retrieve the rowIndex of the currently selected row
var currRow = dataset.rowIndex
```

**Methods Details****addColumn(name)**

adds a column with the specified name to the dataset.

**Parameters**

**String** name column name.

**Returns**

**Boolean** true if succeeded, else false.

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
//assuming the variable dataset contains a dataset
var success = dataset.addColumn('columnName',1);
```

**addColumn(name, index)**

adds a column with the specified name to the dataset.

**Parameters**

**String** name column name.

**Number** index column index number between 1 and getMaxColumnIndex().

**Returns**

**Boolean** true if succeeded, else false.

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
//assuming the variable dataset contains a dataset
var success = dataset.addColumn('columnName',1);
```

**addColumn(name, index, type)**

adds a column with the specified name to the dataset.

**Parameters**

**String** name column name.

**Number** index column index number between 1 and getMaxColumnIndex().

**Number** type the type of column, see JSColumn constants.

**Returns**

**Boolean** true if succeeded, else false.

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
//assuming the variable dataset contains a dataset
var success = dataset.addColumn('columnName',1);
```

**addHTMLProperty(row, col, name, value)**

---

Add an HTML property to an HTML tag produced in `getAsHTML()`.

For row and col parameters use:  
 1 = applies to the container  
 0 = applies to all  
 >0 = applies to specific cell

#### Parameters

**Number** row row number  
**Number** col column number  
**String** name String property name  
**String** value String property value

#### Supported Clients

SmartClient,WebClient,NGClient

#### Sample

```
//adds a container property (to TABLE tag)
dataset.addHTMLProperty(-1,-1,'cellspacing','3');
dataset.addHTMLProperty(-1,-1,'style','border-collapse:collapse;'); //to have a single line border

//adds a row property to all rows (to TR tag)
dataset.addHTMLProperty(0,0,'class','text');

//adds a row property to second row (to TR tag)
dataset.addHTMLProperty(2,0,'class','text');

//adds a column property to all 3rd columns (to TD tag)
dataset.addHTMLProperty(0,3,'class','redcolumn') ;

//adds a specific cell property (to TD tag)
dataset.addHTMLProperty(2,4,'color','blue');

scopes.globals.html_field = '<html>'+dataset.getAsHTML()+'</html>';
```

### addRow(index, array)

Add a row to the dataset.

#### Parameters

**Number** index index to add row (1-based)  
**Array** array row data

#### Supported Clients

SmartClient,WebClient,NGClient

#### Sample

```
//assuming the variable dataset contains a dataset
dataset.addRow(new Array(1,2,3,4,5,6,7,7)); //adds a row with 8 columns
dataset.addRow(2, new Array(1,2,3,4,5,6,7,7)); //adds a row with 8 columns at row 2
```

### addRow(array)

Add a row to the dataset. The row will be added as the last row.

#### Parameters

**Array** array row data

#### Supported Clients

SmartClient,WebClient,NGClient

#### Sample

```
//assuming the variable dataset contains a dataset
dataset.addRow(new Array(1,2,3,4,5,6,7,7)); //adds a row with 8 columns
dataset.addRow(2, new Array(1,2,3,4,5,6,7,7)); //adds a row with 8 columns at row 2
```

### createDataSource(name)

Create a datasource from the data set with specified name and using specified types.  
The types are inferred from the data if possible.

A temporary datasource cannot be removed because once created there may always be forms or relations that refer to it.

When the client exits, all datasources used by that client are removed automatically.

Most resources used by the datasource can be released by deleting all records:  
dataset.removeRow(-1) or databaseManager.getFoundSet(datasource).deleteAllRecords()

#### Parameters

[String](#) name datasource name

#### Returns

[String](#) String uri reference to the created datasource.

#### Supported Clients

SmartClient,WebClient,NGClient

#### Sample

```
ds.addColumn('my_id'); // note: use regular javascript identifiers so they can be used in scripting
ds.addColumn('my_label');
var uri = ds.createDataSource('mydata', [JSColumn.INTEGER, JSColumn.TEXT]);
var jsform = solutionModel.newForm(fname, uri, null, true, 300, 300);

var query = 'select customerid, address, city, country from customers';
var ds2 = databaseManager.getDataSetByQuery('example_data', query, null, 999);
var uri2 = ds2.createDataSource('mydata2'); // types are inferred from query result
```

### createDataSource(name, types)

Create a datasource from the data set with specified name and using specified types.

A temporary datasource cannot be removed because once created there may always be forms or relations that refer to it.

When the client exits, all datasources used by that client are removed automatically.

Most resources used by the datasource can be released by deleting all records:  
dataset.removeRow(-1) or databaseManager.getFoundSet(datasource).deleteAllRecords()

A datasource can be reused if the data has the same signature (column names and types).  
A new createDataSource() call will clear the datasource contents from a previous call and insert the current data.

#### Parameters

[String](#) name datasource name

[Object](#) types array of types as defined in JSColumn

#### Returns

[String](#) String uri reference to the created datasource.

#### Supported Clients

SmartClient,WebClient,NGClient

#### Sample

```
ds.addColumn('my_id'); // note: use regular javascript identifiers so they can be used in scripting
ds.addColumn('my_label');
var uri = ds.createDataSource('mydata', [JSColumn.INTEGER, JSColumn.TEXT]);
var jsform = solutionModel.newForm(fname, uri, null, true, 300, 300);

var query = 'select customerid, address, city, country from customers';
var ds2 = databaseManager.getDataSetByQuery('example_data', query, null, 999);
var uri2 = ds2.createDataSource('mydata2'); // types are inferred from query result
```

### createDataSource(name, types, pkNames)

Create a datasource from the data set with specified name and using specified types.

A temporary datasource cannot be removed because once created there may always be forms or relations that refer to it.

When the client exits, all datasources used by that client are removed automatically.

Most resources used by the datasource can be released by deleting all records:  
dataset.removeRow(-1) or databaseManager.getFoundSet(datasource).deleteAllRecords()

**Parameters**

**String** name   datasource name  
**Object** types   array of types as defined in JSColumn, when null types are inferred from the query result  
**Array** pkNames array of pk names, when null a hidden pk-column will be added

**Returns**

**String** String uri reference to the created datasource.

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
ds.addColumn('my_id'); // note: use regular javascript identifiers so they can be used in scripting
ds.addColumn('my_label');
var uri = ds.createDataSource('mydata', [JSColumn.INTEGER, JSColumn.TEXT], ['my_id']);
var jsform = solutionModel.newForm(fname, uri, null, true, 300, 300);

var query = 'select customerid, address, city, country from customers';
var ds2 = databaseManager.getDataSetByQuery('example_data', query, null, 999);
var uri2 = ds2.createDataSource('mydata2', null, ['customerid']); // types are inferred from query result, use
customerid as pk
```

**getAsHTML()**

Get the dataset as an html table, do not escape values or spaces, no multi\_line\_markup, do not add indentation, add column names.

**Returns**

**String** String html.

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
//gets a dataset based on a query
//useful to limit the number of rows
var maxReturnedRows = 10;
var query = 'select c1,c2,c3 from test_table where start_date = ?';

//to access data by name, do not use '.' or special characters in names or aliases
var args = new Array();
args[0] = order_date //or new Date();
var dataset = databaseManager.getDataSetByQuery(databaseManager.getDataSourceServerName(controller.
getDataSource()), query, args, maxReturnedRows);

// gets a dataset with escape values; escape spaces (lines will not wrap); no multi-line markup; with pretty
indentation; shows column names
var htmlTable = dataset.getAsHTML(true, true, false, true, true);

//assigns the dataset to a field and sets the display type to HTML_AREA
//assuming the html_field is a global text variable
scopes.globals.html_field = '<html>'+dataset.getAsHTML()+'</html>';

//Note: To display an HTML_AREA field as an HTML page, add HTML tags at the beginning '<html>' and at the end '<
/html>'.
```

**getAsHTML(escape\_values)**

Get the dataset as an html table, do not escape spaces, no multi\_line\_markup, do not add indentation, add column names.

**Parameters**

**Boolean** escape\_values if true, replaces illegal HTML characters with corresponding valid escape sequences.

**Returns**

**String** String html.

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
//gets a dataset based on a query
//useful to limit the number of rows
var maxReturnedRows = 10;
var query = 'select c1,c2,c3 from test_table where start_date = ?';

//to access data by name, do not use '.' or special characters in names or aliases
var args = new Array();
args[0] = order_date //or new Date();
var dataset = databaseManager.getDataSetByQuery(databaseManager.getDataSourceServerName(controller.
getDataSource()),query,args,maxReturnedRows);

// gets a dataset with escape values; escape spaces (lines will not wrap); no multi-line markup; with pretty
indentation; shows column names
var htmlTable = dataset.getAsHTML(true, true, false, true, true);

//assigns the dataset to a field and sets the display type to HTML_AREA
//assuming the html_field is a global text variable
scopes.globals.html_field = '<html>'+dataset.getAsHTML()+ '</html>';

//Note: To display an HTML_AREA field as an HTML page, add HTML tags at the beginning '<html>' and at the end '<
/html>'.
```

**getAsHTML(escape\_values, escape\_spaces)**

Get the dataset as an html table, no multi\_line\_markup, do not add indentation, add column names.

**Parameters**

- Boolean** escape\_values if true, replaces illegal HTML characters with corresponding valid escape sequences.
- Boolean** escape\_spaces if true, replaces text spaces with non-breaking space tags ( ) and tabs by four non-breaking space tags.

**Returns**

**String** String html.

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
//gets a dataset based on a query
//useful to limit the number of rows
var maxReturnedRows = 10;
var query = 'select c1,c2,c3 from test_table where start_date = ?';

//to access data by name, do not use '.' or special characters in names or aliases
var args = new Array();
args[0] = order_date //or new Date();
var dataset = databaseManager.getDataSetByQuery(databaseManager.getDataSourceServerName(controller.
getDataSource()),query,args,maxReturnedRows);

// gets a dataset with escape values; escape spaces (lines will not wrap); no multi-line markup; with pretty
indentation; shows column names
var htmlTable = dataset.getAsHTML(true, true, false, true, true);

//assigns the dataset to a field and sets the display type to HTML_AREA
//assuming the html_field is a global text variable
scopes.globals.html_field = '<html>'+dataset.getAsHTML()+ '</html>';

//Note: To display an HTML_AREA field as an HTML page, add HTML tags at the beginning '<html>' and at the end '<
/html>'.
```

**getAsHTML(escape\_values, escape\_spaces, multi\_line\_markup)**

Get the dataset as an html table, do not add indentation, add column names.

**Parameters**

- Boolean** `escape_values` if true, replaces illegal HTML characters with corresponding valid escape sequences.
- Boolean** `escape_spaces` if true, replaces text spaces with non-breaking space tags ( ) and tabs by four non-breaking space tags.
- Boolean** `multi_line_markup` if true, multiLineMarkup will enforce new lines that are in the text; single new lines will be replaced by <br>, multiple new lines will be replaced by <p>

**Returns**

**String** String html.

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
//gets a dataset based on a query
//useful to limit the number of rows
var maxReturnedRows = 10;
var query = 'select c1,c2,c3 from test_table where start_date = ?';

//to access data by name, do not use '.' or special characters in names or aliases
var args = new Array();
args[0] = order_date //or new Date();
var dataset = databaseManager.getDataSetByQuery(databaseManager.getDataSourceServerName(controller.
getDataSource()),query,args,maxReturnedRows);

// gets a dataset with escape values; escape spaces (lines will not wrap); no multi-line markup; with pretty
indentation; shows column names
var htmlTable = dataset.getAsHTML(true, true, false, true, true);

//assigns the dataset to a field and sets the display type to HTML_AREA
//assuming the html_field is a global text variable
scopes.globals.html_field = '<html>'+dataset.getAsHTML()+'</html>';

//Note: To display an HTML_AREA field as an HTML page, add HTML tags at the beginning '<html>' and at the end '<
/html>'.
```

**getAsHTML(escape\_values, escape\_spaces, multi\_line\_markup, pretty\_indent)**

Get the dataset as an html table, add column names.

**Parameters**

- Boolean** `escape_values` if true, replaces illegal HTML characters with corresponding valid escape sequences.
- Boolean** `escape_spaces` if true, replaces text spaces with non-breaking space tags ( ) and tabs by four non-breaking space tags.
- Boolean** `multi_line_markup` if true, multiLineMarkup will enforce new lines that are in the text; single new lines will be replaced by <br>, multiple new lines will be replaced by <p>
- Boolean** `pretty_indent` if true, adds indentation for more readable HTML code.

**Returns**

**String** String html.

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
//gets a dataset based on a query
//useful to limit the number of rows
var maxReturnedRows = 10;
var query = 'select c1,c2,c3 from test_table where start_date = ?';

//to access data by name, do not use '.' or special characters in names or aliases
var args = new Array();
args[0] = order_date //or new Date();
var dataset = databaseManager.getDataSetByQuery(databaseManager.getDataSourceServerName(controller.
getDataSource()),query,args,maxReturnedRows);

// gets a dataset with escape values; escape spaces (lines will not wrap); no multi-line markup; with pretty
indentation; shows column names
var htmlTable = dataset.getAsHTML(true, true, false, true, true);

//assigns the dataset to a field and sets the display type to HTML_AREA
//assuming the html_field is a global text variable
scopes.globals.html_field = '<html>'+dataset.getAsHTML()+ '</html>';

//Note: To display an HTML_AREA field as an HTML page, add HTML tags at the beginning '<html>' and at the end '<
/html>'.
```

**getAsHTML(escape\_values, escape\_spaces, multi\_line\_markup, pretty\_indent, add\_column\_names)**

Get the dataset as an html table.

**Parameters**

**Boolean** escape\_values if true, replaces illegal HTML characters with corresponding valid escape sequences.  
**Boolean** escape\_spaces if true, replaces text spaces with non-breaking space tags ( ) and tabs by four non-breaking space tags.  
**Boolean** multi\_line\_markup if true, multiLineMarkup will enforce new lines that are in the text; single new lines will be replaced by <br>, multiple new lines will be replaced by <p>  
**Boolean** pretty\_indent if true, adds indentation for more readable HTML code.  
**Boolean** add\_column\_names if false, column headers will not be added to the table.

**Returns**

**String** String html.

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
//gets a dataset based on a query
//useful to limit the number of rows
var maxReturnedRows = 10;
var query = 'select c1,c2,c3 from test_table where start_date = ?';

//to access data by name, do not use '.' or special characters in names or aliases
var args = new Array();
args[0] = order_date //or new Date();
var dataset = databaseManager.getDataSetByQuery(databaseManager.getDataSourceServerName(controller.
getDataSource()),query,args,maxReturnedRows);

// gets a dataset with escape values; escape spaces (lines will not wrap); no multi-line markup; with pretty
indentation; shows column names
var htmlTable = dataset.getAsHTML(true, true, false, true, true);

//assigns the dataset to a field and sets the display type to HTML_AREA
//assuming the html_field is a global text variable
scopes.globals.html_field = '<html>'+dataset.getAsHTML()+ '</html>';

//Note: To display an HTML_AREA field as an HTML page, add HTML tags at the beginning '<html>' and at the end '<
/html>'.
```

**getAsText(column\_separator, row\_separator, value\_delimiter, add\_column\_names)**



Get the dataset as formatted text.

#### Parameters

**String** column\_separator any specified column separator; examples: tab '\t'; comma ','; semicolon ';'; space ' '.  
**String** row\_separator the specified row separator; examples: new line '\n'.  
**String** value\_delimiter the specified value delimiter; null means empty string; example: double quote '"'.  
**Boolean** add\_column\_names if true column names will be added as a first row.

#### Returns

**String** String formatted text.

#### Supported Clients

SmartClient, WebClient, NGClient

#### Sample

```
//assuming the variable dataset contains a dataset
//you can create csv or tab delimited results
var csv = dataset.getAsText(',', '\n', '"', true)
var tab = dataset.getAsText('\t', '\n', '"', true)
```

### getColumnAsArray(index)

Get the column data of a dataset as an Array.

#### Parameters

**Number** index index of column (1-based).

#### Returns

**Array** Object array of data.

#### Supported Clients

SmartClient, WebClient, NGClient

#### Sample

```
//assuming the variable dataset contains a dataset
var dataArray = dataset.getColumnAsArray(1); //puts the contents from the first column of the dataset into an array
//once you have it as an array you can loop through it or feed it to a custom valuelist for example
```

### getColumnName(index)

Get a column name based on index.

#### Parameters

**Number** index index of column (1-based).

#### Returns

**String** String column name.

#### Supported Clients

SmartClient, WebClient, NGClient

#### Sample

```
//assuming the variable dataset contains a dataset
var firstColumnName = dataset.getColumnName(1) //retrieves the first columnname into the variable firstColumnName
//using a loop you can get all columnnames in an array:
var query = 'select * from customers';
var dataset = databaseManager.getDataSetByQuery(databaseManager.getDataSourceServerName(controller.
getDataSource()), query, null, 100);
var colArray = new Array()
for (var i = 1; i <= dataset.getMaxColumnIndex(); i++)
{
    colArray[i-1] = dataset.getColumnName(i)
    //note the -1, because an array is zero based and dataset is 1 based.
}
}
```

### getColumnNames()

Get the column names of a dataset.

**Returns**

[Array](#) String[] column names

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
var query = 'select * from customers';
var dataset = databaseManager.getDataSetByQuery(databaseManager.getDataSourceServerName(controller.
getDataSource()), query, null, 100);
var columnNames = dataset.getColumnNames();
```

**getColumnType(index)**

Get a column type based on index.

**Parameters**

[Number](#) index index of column (1-based).

**Returns**

[Number](#) Number the column type (JSColumn constant)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
//assuming the variable dataset contains a dataset
var firstColumnType = dataset.getColumnType(1) //retrieves the first column's type into the variable
firstColumnType
if (firstColumnType == JSColumn.NUMBER) { }
```

**getException()**

Get the database exception if an error occurred.

**Returns**

[Exception](#) ServoyException exception or null when not available.

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
//assuming the variable dataset contains a dataset
var dbException = dataset.getException();
```

**getMaxColumnIndex()**

Get the number of columns in the dataset.

**Returns**

[Number](#) int number of columns.

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
//assuming the variable dataset contains a dataset
for (var i = 1; i <= dataset.getMaxColumnIndex(); i++)
{
    colArray[i-1] = dataset.getColumnName(i)
    //have to subtract 1, because an array is zero based and a dataset is 1 based.
}
```

**getMaxRowIndex()**

Get the number of rows in the dataset.

---

**Returns**

[Number](#) int number of rows.

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
//assuming the variable dataset contains a dataset
var totalRows = dataset.getMaxRowIndex();
```

**getRowAsArray(index)**

Get the row data of a dataset as an Array.

**Parameters**

[Number](#) index index of row (1-based).

**Returns**

[Array](#) Object array of data.

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
//assuming the variable dataset contains a dataset
var dataArray = dataset.getRowAsArray(1); //puts the contents from the first row of the dataset into an array
//once you have it as an array you can loop through it
```

**getValue(row, col)**

Get the value specified by row and column position from the dataset.

**Parameters**

[Number](#) row row number, 1-based

[Number](#) col column number, 1-based

**Returns**

[Object](#) Object value

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
//assuming the variable dataset contains a dataset
var dataAtRow2Col1 = dataset.getValue(2, 1);
```

**hadMoreData()**

Return true if there is more data in the resultset then specified by maxReturnedRows at query time.

**Returns**

[Boolean](#) boolean more data available

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
var ds = databaseManager.getDataSetByQuery('example_data', 'select order_id from orders', null, 10000)
if (ds.hadMoreData())
{
    // handle large result
}
```

**removeColumn(index)**

Remove a column by index from the dataset.

---

**Parameters**

**Number** index index of column to remove (1-based)

**Returns**

**Boolean** true if succeeded, else false.

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
//assuming the variable dataset contains a dataset
var success = dataset.removeColumn(1); // removes first column
```

**removeRow(row)**

Remove a row from the dataset.

**Parameters**

**Number** row row index to remove, -1 for all rows

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
//assuming the variable dataset contains a dataset
dataset.removeRow(1); //removes the first row
dataset.removeRow(-1); //removes all rows
```

**setColumnName(index, columnName)**

Set a column name based on index.

**Parameters**

**Number** index index of column (1-based).

**String** columnName new column name.

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
var query = 'select customerid, customername from customers';
var dataset = databaseManager.getDataSetByQuery(databaseManager.getDataSourceServerName(controller.
getDataSource()), query, null, -1);
dataset.setColumnName(2, 'name_of_customer') // change the column name for second column.
```

**setValue(row, col, obj)**

Set the value specified by row and column position from the dataset.  
Use row = -1, to set columnnames.

**Parameters**

**Number** row row number, 1-based

**Number** col column number, 1-based

**Object** obj the value to be stored at the given row and column.

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
//assuming the variable dataset contains a dataset
dataset.setValue(2, 1,'data');
```

**sort(col, sort\_direction)**

Sort the dataset on the given column (1-based) in ascending or descending.

---

**Parameters**

**Number** col column number, 1-based  
**Boolean** sort\_direction ascending (true) or descending (false)

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
// sort using column number
//assuming the variable dataset contains a dataset
dataset.sort(1, false)
```

**sort(comparator)**

Sort the dataset using the function as comparator.

The comparator function is called to compare two rows, that are passed as arguments, and it will return -1/0/1 if the first row is less/equal/greater then the second row.

NOTE: starting with 7.2 release, when called on datasource(foundset) dataset, this function doesn't save the data anymore

**Parameters**

**Function** comparator comparator function

**Supported Clients**

SmartClient,WebClient,NGClient

**Sample**

```
//sort using comparator
dataset.sort(mySortFunction);

function mySortFunction(r1, r2)
{
    var o = 0;
    if(r1[0] < r2[0])
    {
        o = -1;
    }
    else if(r1[0] > r2[0])
    {
        o = 1;
    }
    return o;
}
```