Modifying Data Grid Columns

By Robert Molina, Technical Services Team Member, 4D Inc.

Technical Note 08-11

Abstract

The Technical Note provides information about the new, easy to use, API functions that modify the column properties of the new Data Grid in the 4D Ajax Framework v11 release 1 (11.1). These newly added functionalities allow developers to deliver a rich web experience for users with very minimal code involved. A sample database is included.

Table of Contents

Introduction

With each revision the 4D Ajax Framework (4DAF) continues to deliver new features requested by developers. In the latest version of the framework, the Data Grid has been reworked and redeveloped from the ground up. The result is a feature-rich grid that provides flexibility and functionality with an easy to use API. Through examples, this Technical Note covers modifying column properties of the Data Grid with the new API calls. The examples will cover:

- Hiding and Displaying Columns
- Column Resizing
- Column Locking
- Inline Editing

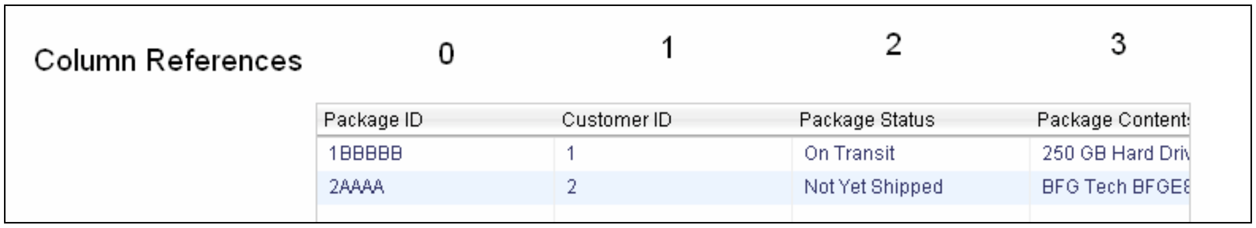
The example consists of the following:

- 4D Database
- 4D Ajax Framework
- Custom HTML pages

In order to combine all of the above components, knowledge of 4D and Javascript is required. In addition, it is recommended to use the Firebug add-on for Firefox to help aid in Javascript development.

Referencing Columns

As the picture below shows, column number starts at 0 and not at 1. In other words, the left most column reference is always 0. This is important to keep in mind when using API calls that require a column number parameter.



Package ID	Customer ID	Package Status	Package Content
1BBBBB	1	On Transit	250 GB Hard Driv
2AAAA	2	Not Yet Shipped	BFG Tech BFGE8

Example Database

The example database included with this document comes in three flavors:

- Interpreted Source Database for 4D 2004
- Interpreted Source Database for 4D v11 SQL
- Merged Database

The interpreted source databases allow you access to the methods used on the backend. Otherwise, feel free to connect to the merged version of this database.

The majority of the coding that went into this example database is actually on the web side of things. Specifically, the JavaScript code in the HTML files included with the merged application.

Installing the 4D Ajax Framework Component

Before using the source database you will need to install the 4D Ajax Framework. This product is part of the Web 2.0 Pack, which is subscription-based therefore we cannot provide the component in the source database. Because of this, only Web 2.0 Pack subscribers will be able to fully utilize the source database. However you can still use the merged application in order to try the examples.

Note: *If you are a Web 2.0 Pack subscriber and would like to install the 4DAF, please follow the instructions supplied with the Web 2.0 Pack.*

Hiding and Displaying Columns

The ability to either hide or display columns is a feature within the new version of the 4D Ajax Framework. The section will provide information regarding the API calls and suggest some practical uses.

Function: .hideColumn()

```
myGrid.hideColumn(colNum)
```

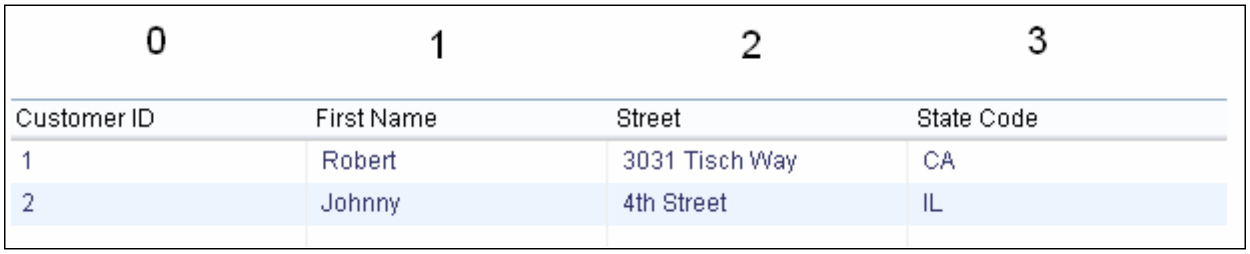
When executed, the above function hides a column in the Data Grid. The pieces involved with the function are:

`myGrid`: The Data Grid object created with new `dax_dataGrid()`.

`hideColumn`: The function that hides the column.

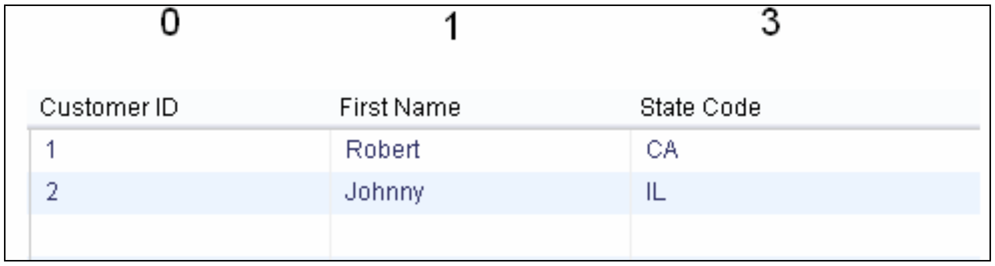
`colNum`: The reference number that corresponds to a column.

When hiding the columns, the reference numbers do not change even though their positions may have when displayed on the page. For instance, below are four columns:



0	1	2	3
Customer ID	First Name	Street	State Code
1	Robert	3031 Tisch Way	CA
2	Johnny	4th Street	IL

From left to right, the reference numbers are 0,1,2,3. If the function is called to hide column 2, the browser will display columns 0,1,3.



0	1	3
Customer ID	First Name	State Code
1	Robert	CA
2	Johnny	IL

Thus, column 3 is now in the position that was once occupied by column 2, but column 3 is still reference with 3 and not 2. Just because the position has changed it does not mean that the column number will change as well.

Function: .showColumn()

```
myGrid.showColumn(colNum)
```

When executed the above function displays a column that is hidden within the Data Grid. The pieces involved with the function are:

`myGrid`: The Data Grid object created with new `dax_dataGrid().`

`showColumn`: The function that will execute and display the column.

`colNum:` The reference number that corresponds to a column.

The result of using this function is the exact opposite of `hideColumn`. When calling this function, it is assumed that the column has already been hidden. When the function is executed, the column will reappear in the browser. The columns will always display in cardinal order from left to right.

Note*: For both functions do not pass a column number that does not exist. Currently the framework does not provide an error when referencing a column out of range.*

***Note:** For both functions, they must be only used after the .go() function has already been executed.*

Note*: The column properties, when hidden stay intact. For instance resizing the column, hiding it, and then displaying it will still retain the previous size.*

Possible Practical Use

These functions can be used to create a custom tailored grid for each user. Since not every user needs to see the exact same View, these functions provide a means to hide or display specific data for each user. For instance user A can only view columns 1 and 3, while user B can view all the columns, these functions allow the developer to deliver a solution that accommodates both users without needing to create a whole new View for each user. The end result is that the developer delivers what his users want with very little work and little time involved.

Column Resizing

This section provides information regarding the API calls that deal with the sizing properties of columns.

Function: .setColumnWidth()

```
myGrid.setColumnWidth(colNum, width)
```

When executed the above function resizes the column width to the specified width value in pixels. The pieces involved with the function are:

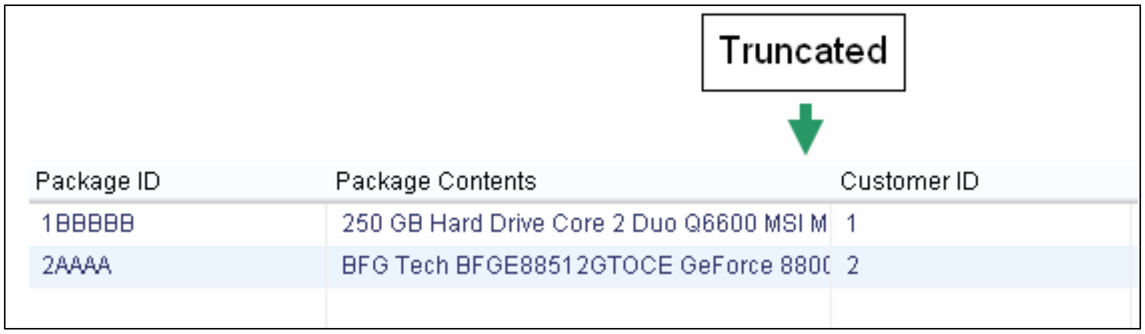
`myGrid`: The Data Grid object created with new `dax_dataGrid()`.

`setColumnWidth`: The function that will execute and set the column width.

`colNum`: The reference number that corresponds to a column.

`Width`: The number in pixels that specifies the width size of the column.

An important note when using this function is that it does not increase the object size when displaying the text. For instance, view the screenshot below:



Package ID	Package Contents	Customer ID
1BBBBB	250 GB Hard Drive Core 2 Duo Q6600 MSI M	1
2AAAA	BFG Tech BFGE88512GTOCE GeForce 880(	2

Trying to resize this through the interface or with the API call may still not display all of the text. This is because the field object containing the text does not automatically increase. Therefore in order to display more of the text to be seen, the `setColumnWidth()` function along with the function `querySetMaxChar()` needs to be used.

***Note**: It is suggested to not resize the right most column. By default, the Data Grid will try to occupy any empty space.*

***Note**: Disabling the resizing of the Data Grid does not affect this function.*

Function: .getColumnWidth ()

`myGrid.getColumnWidth(colNum)`

The above function returns the specified column width, in pixels. The pieces involved with the function are:

`myGrid`: The Data Grid object created with new `dax_dataGrid()`.

`getColumnWidth`: The function that will return the column width.

`colNum`: The reference number that corresponds to a column.

The `getColumnWidth` function provides the developer a means to obtain the current size in pixels of a specific column. The column itself does not need to be visible in order to obtain the column width.

***Note**: By default the column sizes are 150 pixels for text and 105 pixels for numeric.*

Function: .allowColumnResize()

`myGrid.allowColumnResize(allowColumnResize)`

When executed the above function either permits or prevents the resizing of columns. The pieces involved with the function are:

`myGrid`: The Data Grid object created with new `dax_dataGrid()`.

`allowColumnResize`: The function that enables or disables column resizing.

`allowColumnResize`: The parameter is a boolean type and accepts either True or False.

Passing True for the `allowColumnResize` parameter allows resizing of the column. Therefore, the user can place the mouse cursor on a column splitter and be able to increase or decrease the size of the column via the interface. On the other hand if False is passed as the parameter, the user will not be able to modify the column widths via click and drag of a column splitter.

Possible Practical Use

The ability to increase the width of columns can accommodate larger pieces of data. In some cases where data within the grid can be quite large these functions can help in displaying these varying sizes of data.

Column Locking

This section provides information regarding the API calls that lock a column.

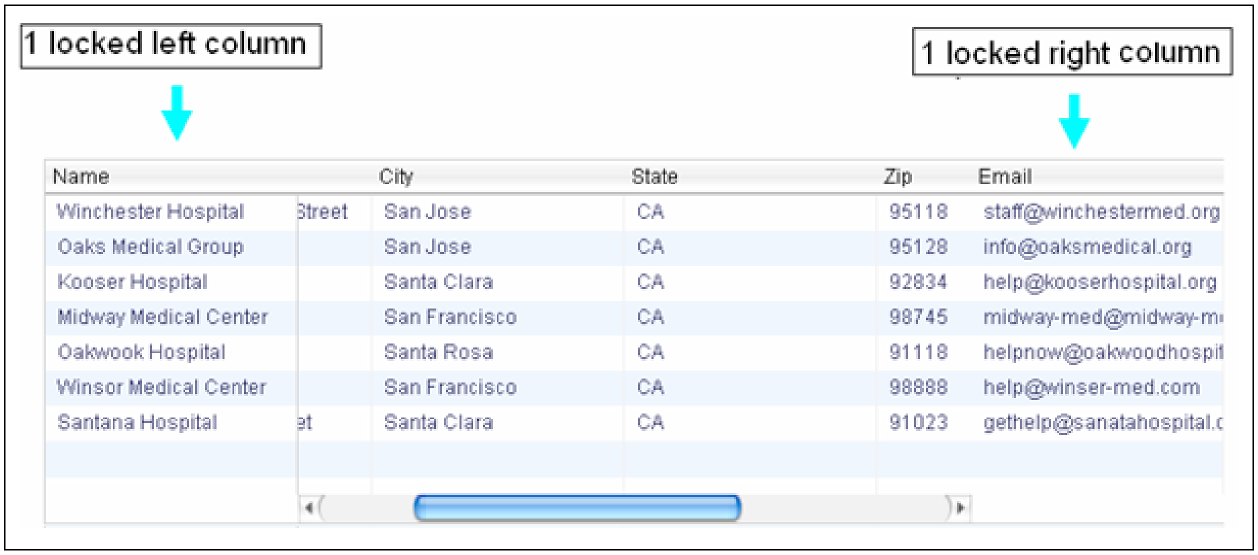
Function: .setRightLockedColumns()

```
myGrid.setRightLockedColumns(number of locked right columns);
```

The above function will lock the column on the right. The pieces involved with the function are:

> `myGrid`: The Data Grid object created with new `dax_dataGrid()`.
>
> `setRightLockedColumns`: The function that sets the number of right locked columns.
>
> `number of locked right columns`: This is the number of columns locked starting from the right of the Data Grid.

The screen shot below displays the right column being locked. This function must be called before the `.go()` function. In the screenshot, notice that the scroll bar is only applied to the middle columns:



Name		City	State	Zip	Email
Winchester Hospital	3treet	San Jose	CA	95118	staff@winchestermed.org
Oaks Medical Group		San Jose	CA	95128	info@oaksmedical.org
Kooser Hospital		Santa Clara	CA	92834	help@kooserhospital.org
Midway Medical Center		San Francisco	CA	98745	midway-med@midway-m
Oakwook Hospital		Santa Rosa	CA	91118	helpnow@oakwoodhospit
Winsor Medical Center		San Francisco	CA	98888	help@winser-med.com
Santana Hospital	et	Santa Clara	CA	91023	gethelp@sanatahospital.c

Function: dax_dataGrid()

```
dax_dataGrid(selection, location, headerRows, lockedLeftColumns,
useControlColumn)
```

This function does several things. One of the properties it sets is the number of locked left most columns.

> ***Note:*** *This function is a constructor and is not called with the same syntax as other functions. (i.e. mygrid.function) This is because this function not only sets properties, but it also creates the Data Grid. Therefore, it is called with the keyword "new". Example:*
>
> ```
> myGrid = new dax_dataGrid(selection, location, headerRows,
> lockedLeftColumns, useControlColumn)
> ```

`lockedLeftColumns`: The parameter specifying number of columns to lock.

For more information regarding the other parameters of this function please view the Data Grid API reference.

The previous screen shot displays a Data Grid with 1 left column being locked.

> ***Note:*** *This function should be called before the* `.go()` *function.*

> ***Note****: Do not lock more columns then you have displayed. For instance, if 4 columns are displayed in the Data Grid, the maximum number of locked right and left columns combined can only be 3 .*

Possible Practical Use

Locking columns is another feature that helps the user work efficiently. With this feature in place, a user can scroll horizontally and can view a specific column constantly without the need to go back and forth. This small feature may appear subtle, but it saves a user the need to constantly keep scrolling back and forth.

Inline Editing

A key feature within the Data Grid is Inline Editing. This feature allows the user to modify the data within the list. Two functions are required to use this feature.

Function: .allowInlineEditing()

```
myGrid.allowInlineEditing(boolean)
```

This function will enable or disable the ability to edit a row within the Data Grid.

`myGrid`: The Data Grid object created with new `dax_dataGrid()`.

`boolean`: The parameter is a boolean type. It accepts either True or False.

When the function is executed the user is able to select a row and click on a field to start editing, similar to the 4D feature, Enterable in List.

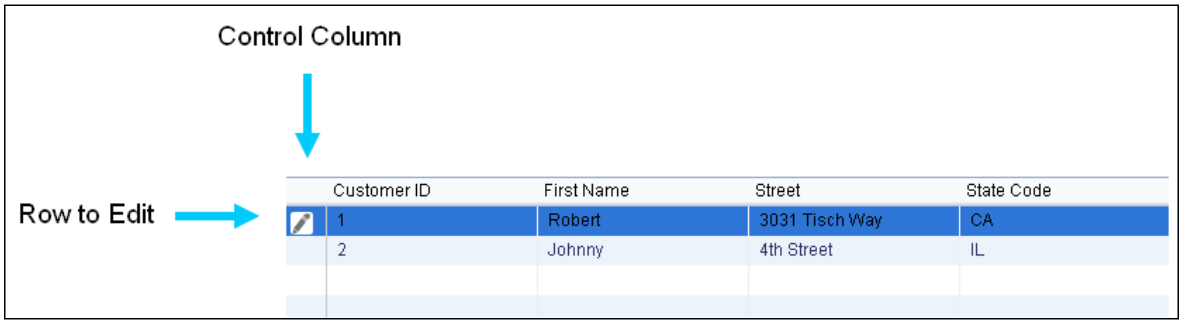
Function: dax_dataGrid()

```
dax_dataGrid(selection, location, headerRows, lockedLeftColumns,
useControlColumn)
```

As mentioned in the previous section this function does several things. One of the properties it sets is the control column.

> `useControlColumn`: Boolean parameter that accepts either True or False. True will display the control column and False will hide the column.

When the function is executed, it will either enable or disable the control column. Below is a screenshot with the control column enabled:



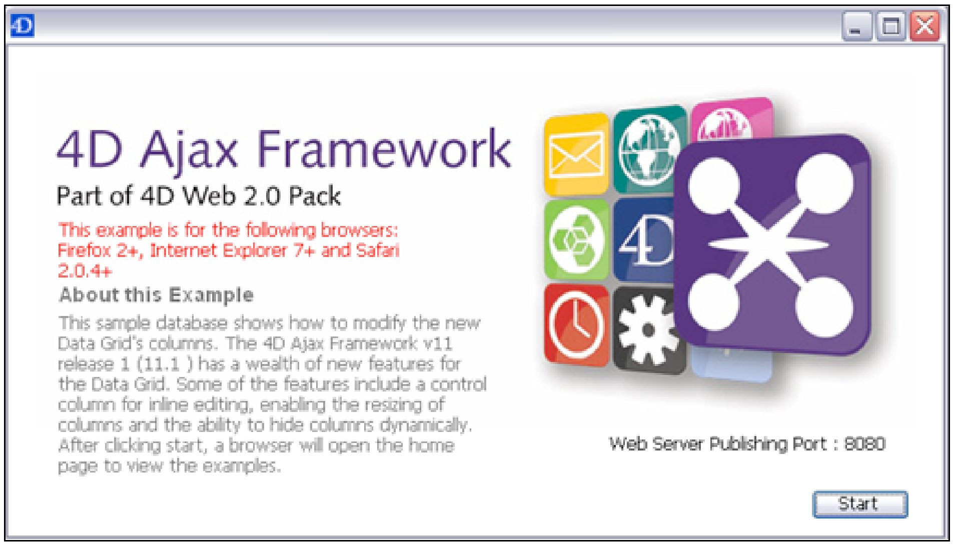
	Customer ID	First Name	Street	State Code
✎	1	Robert	3031 Tisch Way	CA
	2	Johnny	4th Street	IL

Possible Practical Use

These two functions work together to implement the Inline Editing feature. With this feature in place, users have the option to either edit a record directly on the row itself or within the Editor Sheet. When making a quick edit to a record, the inline edit may work best.

Using the Example Database

After launching the example database, the following splash screen appears:



4D Ajax Framework

Part of 4D Web 2.0 Pack

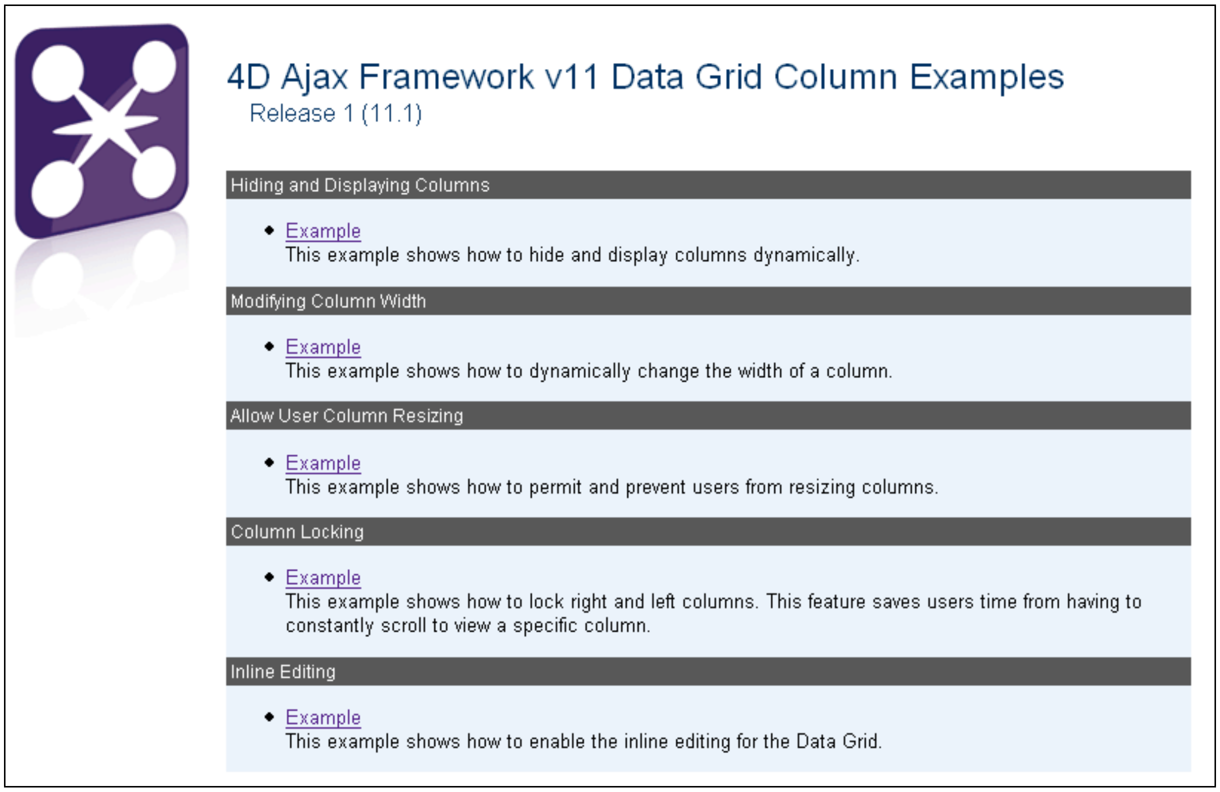
This example is for the following browsers: Firefox 2+, Internet Explorer 7+ and Safari 2.0.4+

About this Example

This sample database shows how to modify the new Data Grid's columns. The 4D Ajax Framework v11 release 1 (11.1) has a wealth of new features for the Data Grid. Some of the features include a control column for inline editing, enabling the resizing of columns and the ability to hide columns dynamically. After clicking start, a browser will open the home page to view the examples.

Web Server Publishing Port : 8080

Clicking on the Start button will open the following webpage:



4D Ajax Framework v11 Data Grid Column Examples

Release 1 (11.1)

Hiding and Displaying Columns

- Example
  This example shows how to hide and display columns dynamically.

Modifying Column Width

- Example
  This example shows how to dynamically change the width of a column.

Allow User Column Resizing

- Example
  This example shows how to permit and prevent users from resizing columns.

Column Locking

- Example
  This example shows how to lock right and left columns. This feature saves users time from having to constantly scroll to view a specific column.

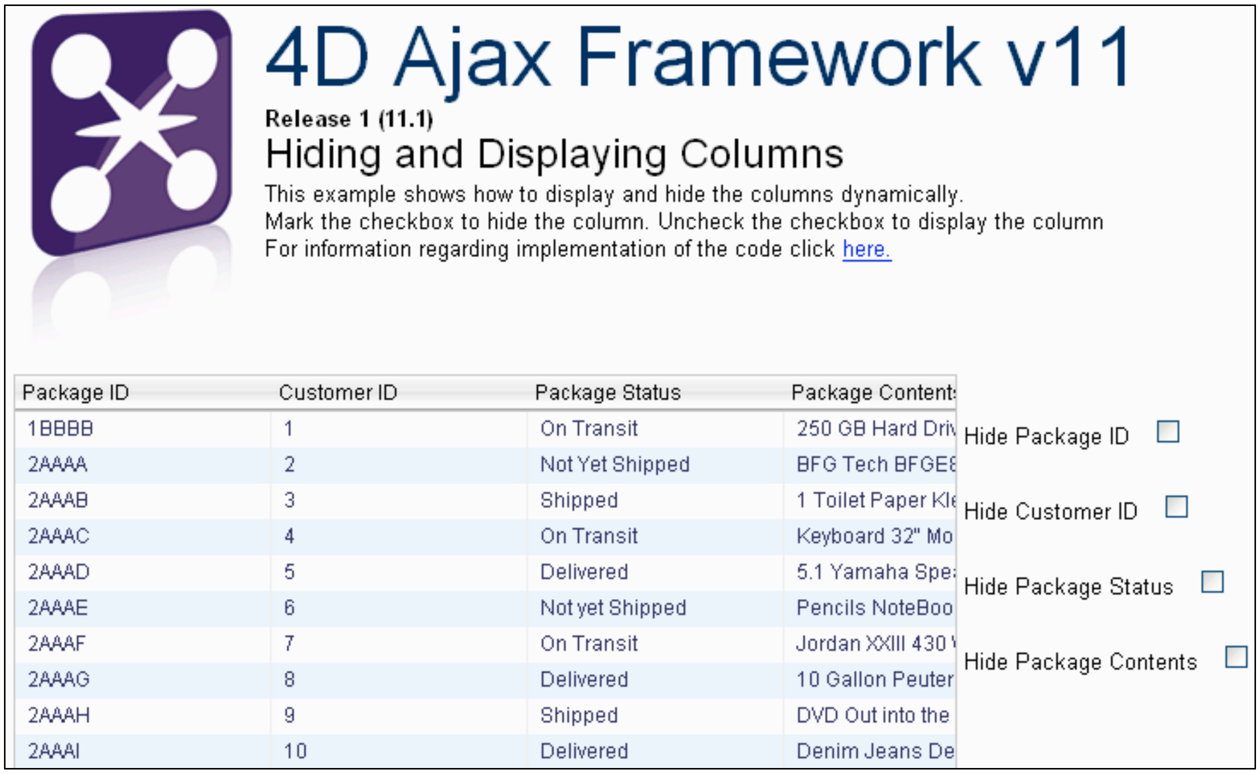
Inline Editing

- Example
  This example shows how to enable the inline editing for the Data Grid.

The above screenshot is the home page. Each section has a quick description of the feature as well as a link to another web page that will provide a demonstration of how the feature is implemented. Each of the sections is explained below.

Example: Hiding and Displaying Columns

At the home page, select the link "Example" in the Hiding and Displaying columns section. The following web page appears:



Hiding and Displaying Columns

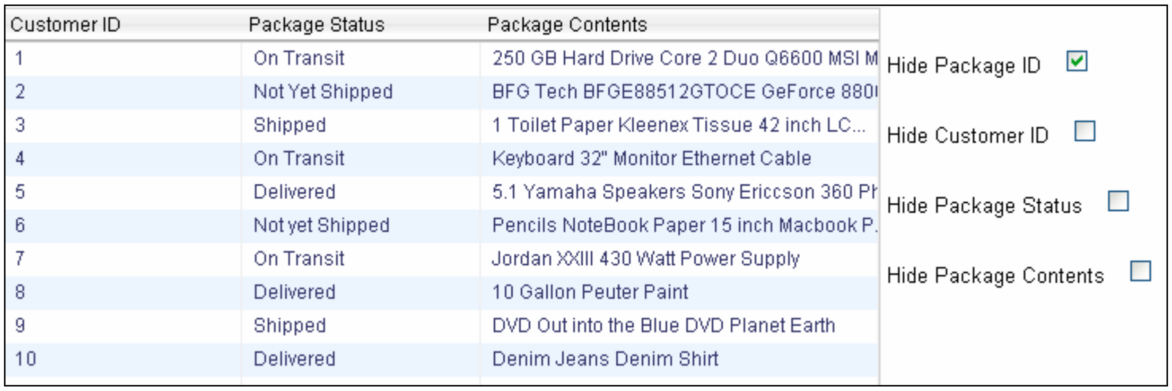
This example shows how to display and hide the columns dynamically.
Mark the checkbox to hide the column. Uncheck the checkbox to display the column
For information regarding implementation of the code click here.

Package ID	Customer ID	Package Status	Package Content
1BBBB	1	On Transit	250 GB Hard Driv
2AAAA	2	Not Yet Shipped	BFG Tech BFGE8
2AAAB	3	Shipped	1 Toilet Paper Kle
2AAAC	4	On Transit	Keyboard 32" Mo
2AAAD	5	Delivered	5.1 Yamaha Spe
2AAAE	6	Not yet Shipped	Pencils NoteBoo
2AAAF	7	On Transit	Jordan XXIII 430
2AAAG	8	Delivered	10 Gallon Peuter
2AAAH	9	Shipped	DVD Out into the
2AAAI	10	Delivered	Denim Jeans De

Hide Package ID ☐
Hide Customer ID ☐
Hide Package Status ☐
Hide Package Contents ☐

How to Use the Example

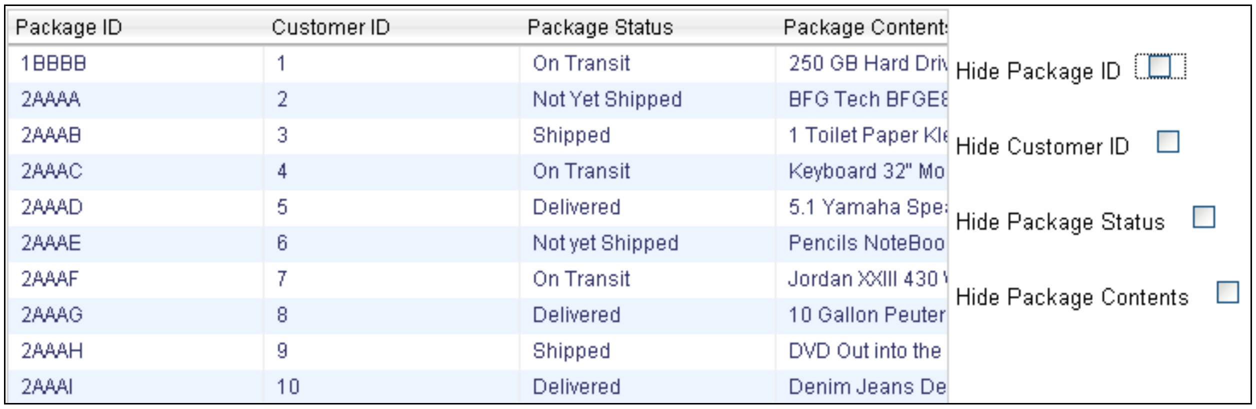
The webpage displays four checkboxes. Checking a checkbox will hide a column and unchecking the checkbox will display the column. For example, enabling the checkbox "Hide Package ID" will hide the "Package ID" coumn:



Customer ID	Package Status	Package Contents
1	On Transit	250 GB Hard Drive Core 2 Duo Q6600 MSI M
2	Not Yet Shipped	BFG Tech BFGE88512GTOCE GeForce 880
3	Shipped	1 Toilet Paper Kleenex Tissue 42 inch LC...
4	On Transit	Keyboard 32" Monitor Ethernet Cable
5	Delivered	5.1 Yamaha Speakers Sony Ericcson 360 Pl
6	Not yet Shipped	Pencils NoteBook Paper 15 inch Macbook P.
7	On Transit	Jordan XXIII 430 Watt Power Supply
8	Delivered	10 Gallon Peuter Paint
9	Shipped	DVD Out into the Blue DVD Planet Earth
10	Delivered	Denim Jeans Denim Shirt

Hide Package ID ✓
Hide Customer ID ☐
Hide Package Status ☐
Hide Package Contents ☐

As the above screenshot displays, the first column "Package ID" is hidden. Unchecking the "Hide Package ID" checkbox will display the column once again:



Package ID	Customer ID	Package Status	Package Content:
1BBBB	1	On Transit	250 GB Hard Driv
2AAAA	2	Not Yet Shipped	BFG Tech BFGE8
2AAAB	3	Shipped	1 Toilet Paper Kle
2AAAC	4	On Transit	Keyboard 32" Mo
2AAAD	5	Delivered	5.1 Yamaha Spe:
2AAAE	6	Not yet Shipped	Pencils NoteBoo
2AAAF	7	On Transit	Jordan XXIII 430 \
2AAAG	8	Delivered	10 Gallon Peuter
2AAAH	9	Shipped	DVD Out into the
2AAAI	10	Delivered	Denim Jeans De

Hide Package ID ☐
Hide Customer ID ☐
Hide Package Status ☐
Hide Package Contents ☐

Implementation

In the example provided, checkbox objects were used to trigger the function which executes `.hideColumn()`. Below is the HTML code for a single checkbox.

```
<input type="checkbox"text" name="hidecol1"
id="hidecolid1"style="width: 30px; height: 15px;margin-top:
10px;"onClick="hide($('hidecolid1'),0);"> </input>
```

The above code is called in four instances within the HTML page, which corresponds to the amount of columns within the Data Grid. The important piece in the above code is the `onClick` attribute which calls the `hide()` function with two parameters.

```
onClick="hide($('hidecolid1'),0);"
```

With this attribute set, it means that when a user clicks on the checkbox object, the function `hide()` will execute with the parameters `$('hidecolid1')` and "0". The value 'hidecolid1' is the id of the checkbox and "0" is the column being referenced from the data grid. Below is the definition of the `hide()` function:

```
   function hide(check,col){
     if(check.checked){
       myDataGrid.hideColumn(col);
     }
     else{
       myDataGrid.showColumn(col);
     }
   }
```

The function first checks if the checkbox is either True or False. It is True if the checkbox is checked and is False when unchecked. Therefore when checked, the line of code executes:

```
myDataGrid.hideColumn(col);
```

The variable `col` contains the value which was passed during the `onclick` event.

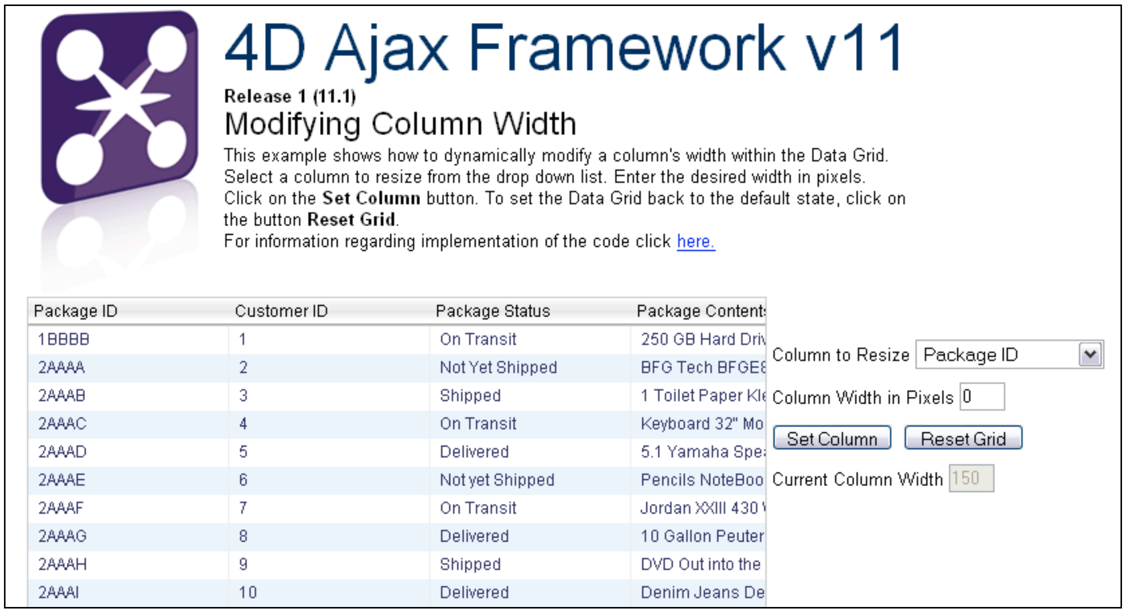
The `.checked` property may also be false in the situation where the checkbox is unchecked. In this case, the else branch is executed:

```
myDataGrid.showColumn(col);
```

As mentioned in the "Hiding and Displaying Columns" section, this function will display the column.

Example: Modifying Column Width

At the home page, select the link "Example" in the Modifying Column Width section. The following web page appears:



Modifying Column Width

This example shows how to dynamically modify a column's width within the Data Grid. Select a column to resize from the drop down list. Enter the desired width in pixels. Click on the **Set Column** button. To set the Data Grid back to the default state, click on the button **Reset Grid**.
For information regarding implementation of the code click here.

Package ID	Customer ID	Package Status	Package Content:
1BBBB	1	On Transit	250 GB Hard Driv
2AAAA	2	Not Yet Shipped	BFG Tech BFGE8
2AAAB	3	Shipped	1 Toilet Paper Kle
2AAAC	4	On Transit	Keyboard 32" Mo
2AAAD	5	Delivered	5.1 Yamaha Spea
2AAAE	6	Not yet Shipped	Pencils NoteBoo
2AAAF	7	On Transit	Jordan XXIII 430 \
2AAAG	8	Delivered	10 Gallon Peuter
2AAAH	9	Shipped	DVD Out into the
2AAAI	10	Delivered	Denim Jeans De

Column to Resize: Package ID
Column Width in Pixels: 0
Set Column | Reset Grid
Current Column Width: 150

How to Use the Example

To change the width of a column:

1. Select the column from the drop down list "Column to Resize". Notice that the "Current Column Width" field displays the width in pixels for the selected column.
2. Enter the size in pixels for the column width within the field "Column Width in Pixels"
3. Click on the set column width button. The Data Grid should reflect the change.

To reset the Data Grid to the default state, click on the "Reset Grid" button.

Implementation

When the page loads the following function runs:

```
function onAfterInit(){
myDataGrid = new dax_dataGrid('View_1', $('griddiv'),1,0,false);
   col=0;
   myDataGrid.go();
   $('colwidthdisplayid').value=myDataGrid.getColumnWidth(col);
   $('columnwidthid').value=0;
}
```

The first line of code within the function is the call to create the grid. The second line of execution will set the global variable `col` to the value of 0. The .`go`() function then is called which displays the grid. The next line of code (pointed with green arrow above) obtains the column width for column 0. The value is saved into the input element with id 'colwidthdisplayid'. Lastly, the input element with id 'columnwidthid' is initialized to 0.

Note: *The expression " $('colwidthdisplayid')" can also be written as:*

document.getElementById('colwidthdisplayid')

After the above code has executed the page is now available to accept input. The first thing a user will do is select a column from the drop down list. Below is the HTML code for the drop down list:

```
<select
name='Columns'onchange='OnChange(this.form.Columns,this.form.colwidthdisplay)'
id="colresize">
<option value="0">Package ID</option>
<option value="1">Customer ID</option>
<option value="2">Package Status</option>
<option value="3">Package Contents</option>
</select>
```

As the code above shows, each option element has a unique value that ranges from 0 to 3, which corresponds to the column reference numbers within the Data Grid. In addition, this select element also calls the function `OnChange()`, which is triggered by the `onchange` event. Thus, changing the value of the drop down list on the web page will trigger this event which will then trigger the function. The `OnChange()` function accepts two parameters; the name of the select element (e.g."Columns") and the name of the input element (e.g. "colwidthdisplay"). Below is the HTML code for the colwidthdisplay input element.

```
<input type="text" disabled="disabled" id="colwidthdisplayid"
name="colwidthdisplay" style="width: 30px; height: 15px;margin-top:
10px;"> </input>
```

Even though this is an input element, no inputs will be accepted since the "disabled" attribute is used. It will strictly be used to display the column width value. Earlier in this section, an explanation was provided of how this object was initialized which gave it a value. Another way this object receives its value is in the following function:

```
function OnChange(dropdown,coldis){
    var myindex  = dropdown.selectedIndex;
    col = dropdown.options[myindex].value;
   coldis.value= myDataGrid.getColumnWidth(col)
   }
```

The first line obtains the index of the drop down list item chosen. The next line then gets the value provided for the item which was provided in the select element tag.

```
<option value="0">Package ID</option>
<option value="1">Customer ID</option>
<option value="2">Package Status</option>
<option value="3">Package Contents</option>
```

$\uparrow$

dropdown.options[myindex].value

That value, which is the column number, is then saved into the `col` variable. That variable can then be passed to the `getColumnWidth()`API function which then will return the column width size value to the `coldis.value` which is essentially the input element named `'colwidthdisplay'`. Now that a column has been selected, a width value can be specified. The width value is stored in the input element name "colwidth".

```
<input type="text" name="colwidth" id="columnwidthid" style="width:
30px; height: 15px;margin-top: 10px;"> </input>
```

After the value is entered, the user can click on the button labeled named "Set Column" which is the HTML input element name "SetCol".

```
<input type="button" name="SetCol" value="Set Column" style="width:
90px; height: 20px;margin-top: 10px;"
onClick="setcolwidth(this.form.colwidth);"> </input>
```

This element has the `onClick` attribute which will call the function `setcolwidth()`. This function has one parameter, which is the value the user entered into the `colwidth` input element. Below is the function definition for `setcolwidth()`:

```
function setcolwidth(num){
  if(isInteger(num.value)){
    if((num.value<10) || (num.value>1000)){
      alert("Value needs to be more than 10 and less than 1000")
    }
    else{
      myDataGrid.setColumnWidth(col, num.value);
      $('colwidthdisplayid').value=myDataGrid.getColumnWidth(col);
```

```
        }
    }
}
```

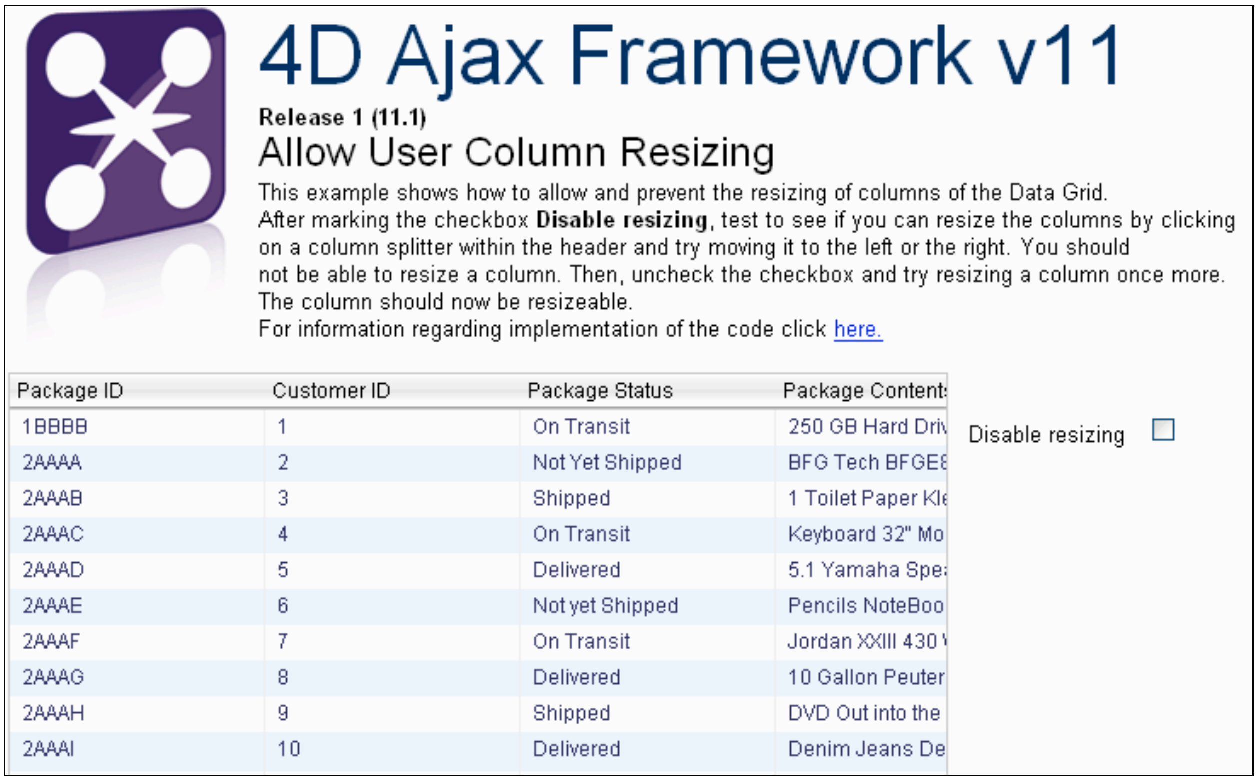
The first line is a conditional statement that calls the `isInteger()` function which returns a boolean. The `isInteger()` function will check to see if the characters passed are integer values. If the values are not integers an alert will be prompted and the function will return false. On the other hand if the values are integers, the function returns true then the code goes onto another conditional statement to check the range of values. If the value is either less than 10 or more than 1000 an alert statement will be prompted asking the user to enter in values in that range. Lastly, if the values are within range the else block executes:

```
        myDataGrid.setColumnWidth(col, num.value);
        $('colwidthdisplayid').value=myDataGrid.getColumnWidth(col);
```

The first line sets the column width of the column specified by the variable `col`. After this line of code executes, the Data Grid will instantly display the changes. The second line of code then goes on to update the input element named `colwidthdisplay`, which will display the recently modified column's widths.

Example: Allow User Column Resizing

At the home page, select the link "Example" from the User Column Resizing section. The following web page appears:



4D Ajax Framework v11

Release 1 (11.1)

Allow User Column Resizing

This example shows how to allow and prevent the resizing of columns of the Data Grid.
After marking the checkbox **Disable resizing**, test to see if you can resize the columns by clicking on a column splitter within the header and try moving it to the left or the right. You should not be able to resize a column. Then, uncheck the checkbox and try resizing a column once more. The column should now be resizeable.
For information regarding implementation of the code click here.

Package ID	Customer ID	Package Status	Package Content
1BBBB	1	On Transit	250 GB Hard Driv
2AAAA	2	Not Yet Shipped	BFG Tech BFGE8
2AAAB	3	Shipped	1 Toilet Paper Kle
2AAAC	4	On Transit	Keyboard 32" Mo
2AAAD	5	Delivered	5.1 Yamaha Spe
2AAAE	6	Not yet Shipped	Pencils NoteBoo
2AAAF	7	On Transit	Jordan XXIII 430
2AAAG	8	Delivered	10 Gallon Peuter
2AAAH	9	Shipped	DVD Out into the
2AAAI	10	Delivered	Denim Jeans De

Disable resizing ☐

How to Use the Example

When the page loads, the Data Grid is displayed. Now do the following:

1. Place the mouse cursor on a splitter.
2. Click and drag the splitter left to right. The column should be resizing.
3. Now click on the checkbox “Disable resizing”
4. Place the mouse cursor on a splitter.
5. Try to click and drag the splitter. The column now is static.

Implementation

By default, when the Data Grid is created, its columns can be resized by the user. In order to disable that behavior the `allowColumnResize()` function is called. In order to call that function the web page uses the input element named `rescheck` with the attribute `type = checkbox`. Below is the HTML code:

```
<input type="checkbox"text" name="rescheck" id="rescheckid"
style="width: 30px; height: 15px;margin-top:
10px;"onClick="allowresize($('rescheckid'));"> </input>
```

The attribute `onClick="allowresize($('rescheckid'))` will call the javascript function `allowresize()` when the user clicks on the check box. The input element with the id `rescheckid` is passed into the `allowresize()` function. Below is the function definition:

```
function allowresize(check){
  if(check.checked){
  myDataGrid.allowColumnResize(false);
  }
  else{
  myDataGrid.allowColumnResize(true);
  }
}
```

The first line within the function is a conditional branch to determine if the check box is either checked or unchecked. The `check.checked` property will be True if the input element passed into the function is checked. If True, the code will execute:

```
myDataGrid.allowColumnResize(false);
```

After this function executes the Data Grid on the web page will reflect the change and prevent users from resizing the column.

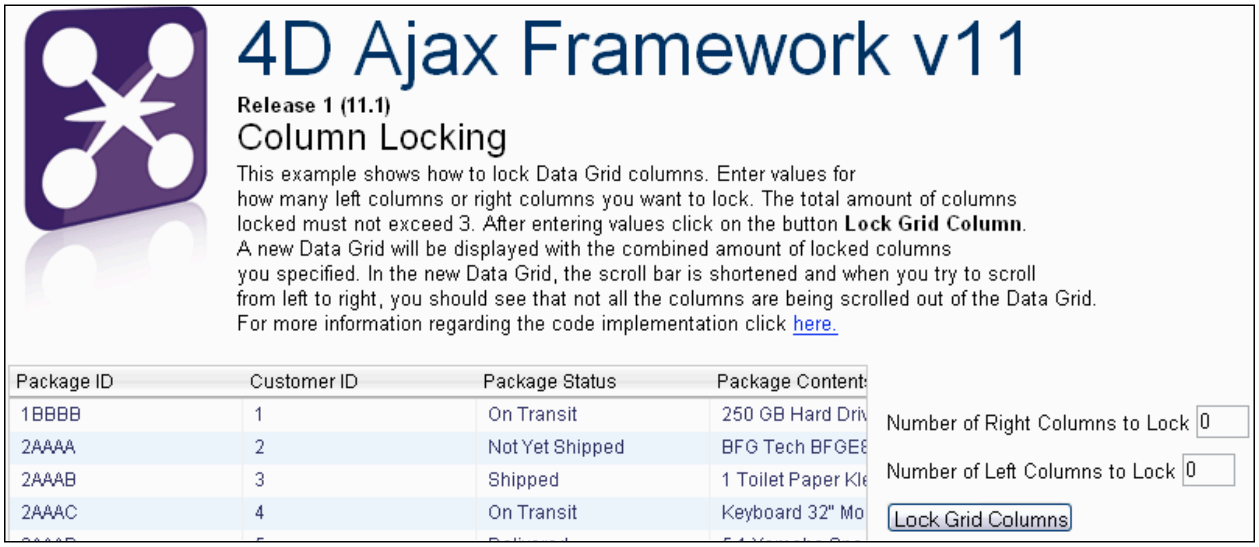
In contrast, if `check.checked` is False the code executed will be the following:

```
myDataGrid.allowColumnResize(true);
```

After the above code executes the Data Grid on the web page will reflect the change and allow users to once again resize the column.

Example: Locking Columns

At the home page, select the link "Example" in the Column Locking section. The following web page appears:



4D Ajax Framework v11

Release 1 (11.1)

Column Locking

This example shows how to lock Data Grid columns. Enter values for how many left columns or right columns you want to lock. The total amount of columns locked must not exceed 3. After entering values click on the button **Lock Grid Column**. A new Data Grid will be displayed with the combined amount of locked columns you specified. In the new Data Grid, the scroll bar is shortened and when you try to scroll from left to right, you should see that not all the columns are being scrolled out of the Data Grid. For more information regarding the code implementation click here.

Package ID	Customer ID	Package Status	Package Content
1BBBB	1	On Transit	250 GB Hard Driv
2AAAA	2	Not Yet Shipped	BFG Tech BFGE8
2AAAB	3	Shipped	1 Toilet Paper Kle
2AAAC	4	On Transit	Keyboard 32" Mo

Number of Right Columns to Lock 0

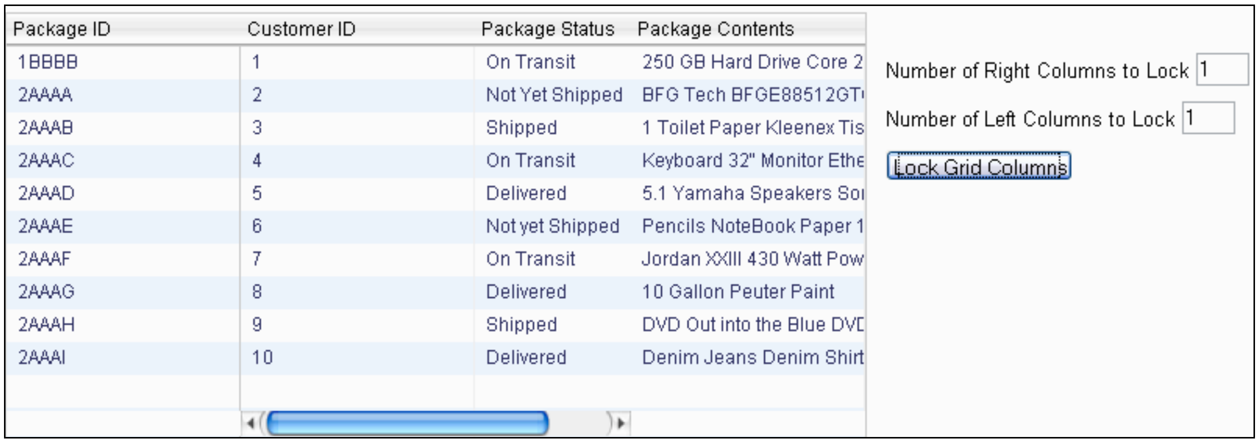
Number of Left Columns to Lock 0

How to Use the Example

The web page displays two input fields, to enter integer values. These two values when added together should never exceed 3. For instance, If the "Number of Right Columns to Lock" has the value 3 then "Number of Left Columns to Lock" has to be 0. This is because there are 4 columns total. The total number of locked columns should always be less than the total number of columns (Left and Right locked columns < Total number of columns).

1. Enter an integer value for "Number of Right Columns to Lock"
2. Enter an integer value for "Number of Left Columns to Lock"
3. Click on the "Lock Grid Columns" button.

Below is a screenshot of 1 locked left column and 1 locked right column:



Package ID	Customer ID	Package Status	Package Contents
1BBBB	1	On Transit	250 GB Hard Drive Core 2
2AAAA	2	Not Yet Shipped	BFG Tech BFGE88512GT
2AAAB	3	Shipped	1 Toilet Paper Kleenex Tis
2AAAC	4	On Transit	Keyboard 32" Monitor Ethe
2AAAD	5	Delivered	5.1 Yamaha Speakers Soi
2AAAE	6	Not yet Shipped	Pencils NoteBook Paper 1
2AAAF	7	On Transit	Jordan XXIII 430 Watt Pow
2AAAG	8	Delivered	10 Gallon Peuter Paint
2AAAH	9	Shipped	DVD Out into the Blue DVD
2AAAI	10	Delivered	Denim Jeans Denim Shirt

Number of Right Columns to Lock 1

Number of Left Columns to Lock 1

Implementation

When the web page loads, no columns are locked. Below is the code that is similar to all the examples provided.

```
function onAfterInit(){
   myDataGrid = new dax_dataGrid('View_1', $('griddiv'),1,0,false);
   myDataGrid.go();
   $('columnsright').value=0;
   $('columnsleft').value=0;
}
```

In the code above, the fourth parameter to `dax_dataGrid()` sets the number of locked left columns to 0. After the page has loaded, the user is given the opportunity to enter how many left or right locked columns. Below is the HTML code for those two input elements.

```
<input type="text" name="colsr" id="columnsright" style="width: 30px;
height: 15px;margin-top: 10px;"> </input>
```

```
<input type="text" name="colsl" id="columnsleft" style="width: 30px;
height: 15px;margin-top: 10px;"> </input>
```

Once values are entered, the button "Lock Grid Columns" is clicked. Below is the HTML code for the button:

```
<input type="button" name="reset" value="Lock Grid Columns"
style="width: 120px; height: 20px;margin-top: 10px;"
onClick="resetGrid(this.form.colsr,this.form.colsl);"> </input>
```

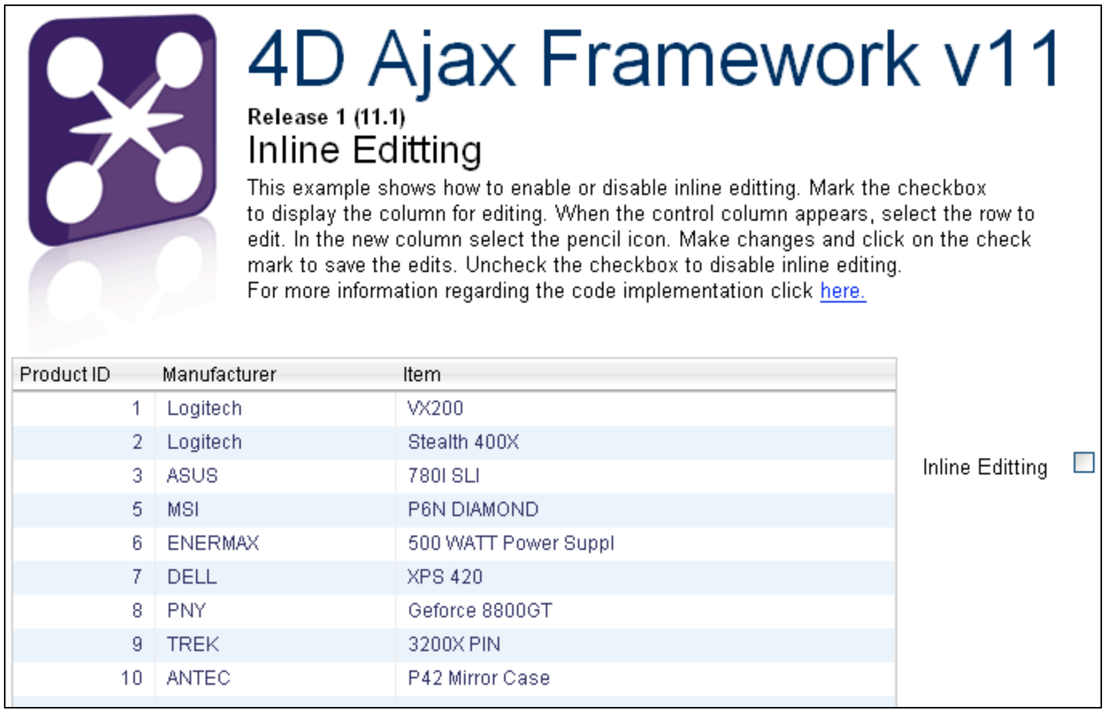
The `onClick` attribute calls the `resetGrid()` function which passes the parameters `this.form.colsr` and `this.form.colsl`, which correspond to the input elements holding the values entered by the user. Below is the function definition where each the values will be used:

```
function resetGrid(colsr,colsl){
  if(isInteger(colsr.value) & isInteger(colsl.value)){
    var result=parseInt(colsr.value) + parseInt(colsl.value);
      if(result <4){
        myDataGrid = null;
        myDataGrid = new dax_dataGrid('View_1',
$('griddiv'),1,colsl.value,false);
        myDataGrid.setRightLockedColumns(colsr.value);
        myDataGrid.go();
      }
      else{
        alert("The number of combined locked columns must be less than
4.  You are trying to lock "+result+" columns")
      }
  }
  else{
    alert("One or both values entered is not a number");
  }
}
```

The first line is a conditional statement that checks if the data entered by the user are actually integer values. If they are not, an alert message will be shown. If the values are verified as integers, the values are added together and stored in a variable. This variable is then used in another "if" statement to determine if the value is less than 3. This is done to prevent locking more columns than there are available. If the variable is more than 3 an alert message will be shown asking the user to enter smaller values. If the value is 3 or less, the `myDataGrid` object is set to null which deletes the Data Grid. After it is cleared, a call to create a new Data Grid is made, but this time the fourth parameter (lockedLeftColumns) is set to a value other than 0. This value is from the input element `colsl`. The next line then sets the number of right locked columns, `myDataGrid.setRightLockedColumns(colsr.value)`. Lastly, `myDataGrid.go()` is executed to display the newly created grid with the specified locked columns.

Example: Inline Editing

At the home page, select the link "Example" from the Inline Editing section. The following web page appears:



4D Ajax Framework v11

Release 1 (11.1)

Inline Editting

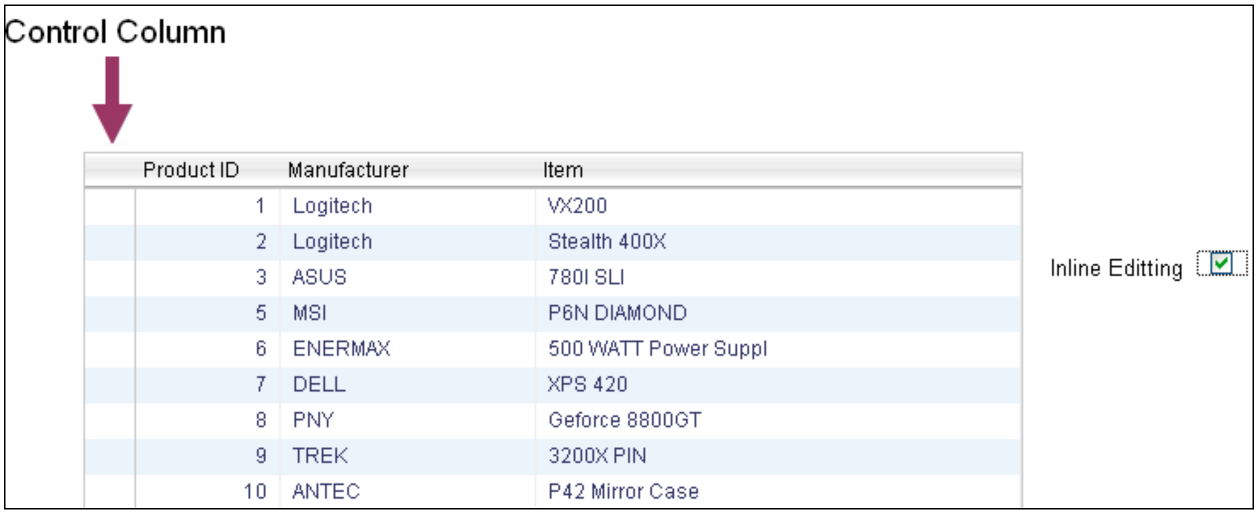
This example shows how to enable or disable inline editting. Mark the checkbox to display the column for editing. When the control column appears, select the row to edit. In the new column select the pencil icon. Make changes and click on the check mark to save the edits. Uncheck the checkbox to disable inline editing. For more information regarding the code implementation click here.

Product ID	Manufacturer	Item
1	Logitech	VX200
2	Logitech	Stealth 400X
3	ASUS	780I SLI
5	MSI	P6N DIAMOND
6	ENERMAX	500 WATT Power Suppl
7	DELL	XPS 420
8	PNY	Geforce 8800GT
9	TREK	3200X PIN
10	ANTEC	P42 Mirror Case

How to Use the Example

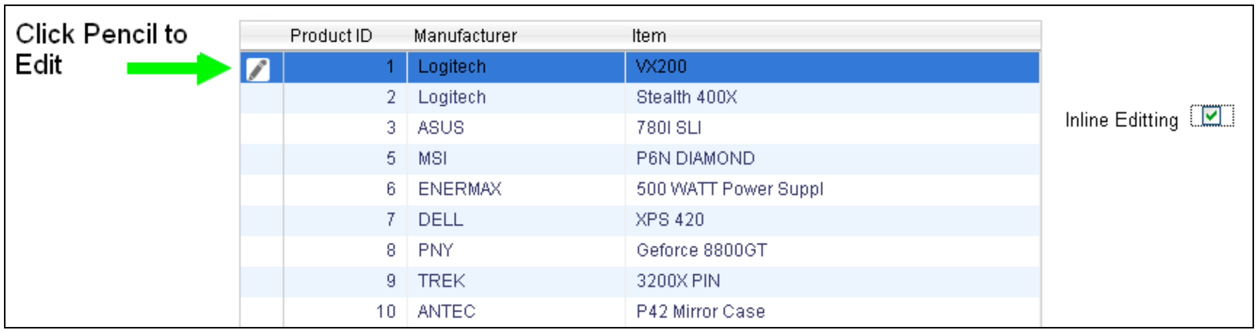
The webpage will display a Data Grid and a checkbox.

1. Check the "Inline Editing" checkbox.
2. A new column appears.



	Product ID	Manufacturer	Item
	1	Logitech	VX200
	2	Logitech	Stealth 400X
	3	ASUS	780I SLI
	5	MSI	P6N DIAMOND
	6	ENERMAX	500 WATT Power Suppl
	7	DELL	XPS 420
	8	PNY	Geforce 8800GT
	9	TREK	3200X PIN
	10	ANTEC	P42 Mirror Case

3. Select a column.



	Product ID	Manufacturer	Item
	1	Logitech	VX200
	2	Logitech	Stealth 400X
	3	ASUS	780I SLI
	5	MSI	P6N DIAMOND
	6	ENERMAX	500 WATT Power Suppl
	7	DELL	XPS 420
	8	PNY	Geforce 8800GT
	9	TREK	3200X PIN
	10	ANTEC	P42 Mirror Case

4. Edit a field.
5. Click on the check mark.

Implementation

In the example, the check box object receives input from the user to either enable or disable the inline editing feature. Below is the HTML code for the checkbox.

```
<input type="checkbox" name="contcol" id="contcolid" style="width:
30px; height: 15px;margin-top:
10px;"onClick="resetGrid($('contcolid'))"> </input>
```

The `onClick` attribute will trigger the function `resetGrid()`. The parameter `$('contcolid')` is passed into the function. Below is the function definition:

```
function resetGrid(check){
  myDataGrid.destroy();
  myDataGrid = new dax_dataGrid('Products',
$('griddiv'),1,0,check.checked);
  myDataGrid.allowInlineEditing(check.checked);
  myDataGrid.go();
}
```

The first line of code will delete the Data Grid. Once deleted, the next line creates a new Data Grid. In the `dax_dataGrid()` call the last parameter determines if the control column will be enabled or disabled. The parameter passed is `check.checked`, which is True if the checkbox is checked, otherwise it is False. Once the control column is added, the Data Grid can be given the property to be editable. This is done with the next line of code:

```
myDataGrid.allowInlineEditing(check.checked);
```

This function also accepts a boolean type parameter thus is given the `check.checked` parameter as well. Lastly, the function `myDataGrid.go()` is called and the newly built Data Grid is displayed with the inline edit feature available and ready for use.

Conclusion

Through the use of examples this Technical Note described how to implement and use the following new column properties of the Data Grid:

- Hiding and Displaying Columns
- Column Resizing
- Column Locking
- Inline Editing

With these new features, developers can create dynamic rich web applications with ease.

A Note about 4D Web 2.0 Pack

The products in 4D Web 2.0 Pack are a departure from most other 4D products. As 4D Web 2.0 pack is a subscription-based product it is expected that incremental releases will be made. Thus please note that this Technical Note is based on 4D Ajax Framework v11 Release 1 (11.1). As new features are implemented this Technical Note may become obsolete (faster than most other 4D products).

Related Resources

4D Ajax Framework Data Grid 2.0:
http://www.daxipedia.com/index.php/Data_Grid_2.0

For the latest information on the 4DAF consult the latest documentation:
http://www.4d.com/support/documentation.html

Also check the 4D Web 2.0 Pack Wiki:
http://daxipedia.4d.com

For the latest news about 4D Web 2.0 Pack or to find out how to purchase it see:
http://www.4d.com/products/4dweb20pack.html