

# TouchTreeGrid

---


**TouchTreeGrid** is an extremely versatile and easy to implement Sencha Touch 2.1x, 2.2x, 2.3x, 2.4x custom component that supports [Tree Grids](#), [Standard Grids](#) and traditional [Accordion](#) view layouts (all by the same component). Column sorting and custom data renderings such as comma formatting, currency, percents and custom colors for negative vs. positive values are included. **TouchTreeGrid** utilizes Sencha's Ext.dataview.List component with all supported configs, methods and events. Most recently support was added to embed forms (readOnly and editable) within Category and Leaf rows.

Refer to "[Basic Features](#)" and "[Notes on Implementation](#)" sections for minimum configurations to get started using TouchTreeGrid.

**TouchTreeGrid** was developed entirely within Sencha Architect designer product and can also be used without Architect. An Architect component (TouchTreeGrid.xdc) is provided for import into your toolbox within each of the project directories (refer to ArchitectComponents directory for different versions). Using Sencha Architect you can implement this component for new grid implementations very rapidly. Scrolling and overall functionality was found to be very fast for a larger data example containing +3000 rows and 3 category levels<sup>1</sup>. This software can be downloaded at <https://github.com/swluken/TouchTreeGrid> and is free to use (refer to associated MIT.LICENSE) . Links to live demo contained within READ.ME file on the GitHub site.

Refer to CalendarPicker.pdf for additional documentation.

Following project directories are included in the download (Architect version 2.2.3 Build 1044 unless noted; Touch version of TouchTreeGrid.xdc and CalendarPicker.xdc as noted) :

Directory	Description
TTG_Slider_2.4	"Kitchen Sink" slide navigation examples using slide menu component , Sencha Touch 2.4 and Architect v3.1.0 Build 1943. TouchTreeGrid non-collapsible treegrid is used for the menu options. This project includes the most updated versions of all examples (see also TouchTreeGrid_FreezeColumn). Be sure to click on  within each example for summary of features highlighted in each example.
TTG_Forms	Refer to TTG_Forms project directory and /TTG_Forms/TTG_Forms.pdf for documentation on examples and on steps to embed forms within Category and Content (aka leaf) rows. These examples also added to TTG_Slider_2.4 project, but not documented in this document.
DynamicLoadEx2C	Dynamically load Nodes when expanded for large TreeStores. Further discussion at: <a href="http://www.sencha.com/forum/showthread.php?259616-TouchTreeGrid-component&amp;p=1058383&amp;viewfull=1#post1058383">http://www.sencha.com/forum/showthread.php?259616-TouchTreeGrid-component&amp;p=1058383&amp;viewfull=1#post1058383</a>

---

<sup>1</sup> Devices tested: Android, iPhones and iPads

# TouchTreeGrid

---

MenuEx	Simplified example of how to utilize new Menu component included in Touch 2.3 for various implementations and using TouchTreeGrid for left/right menu items.
TouchTreeGrid_FreezeColumn	Freeze Column with horizontal scrolling columns example simulates EXTJS 4.2 Kitchen Sink Big Data grid example...refer to TouchTreeGrid_FreezeColumn.pdf for additional documentation. Touch 2.3.0
CalendarPicker	Full set of Calendar Picker examples. Touch 2.3
CalendarPicer_Basic	Basic example of Calendar Pickers used for date selection. Touch 2.3
TouchTreeGrid_Advanced_22	Original Tab Panel version of Examples implemented for Touch 2.2.1
TouchTreeGrid_Advanced_21	Original Tab Panel version of Examples implemented for Touch 2.1 (Architect Version: 2.2.2 Build: 991)
TouchTreeGrid_Basic_22	Basic TreeGrid example for Touch 2.2.1
TouchTreeGrid_Basic_21	Basic TreeGrid example for Touch 2.1 (Architect Version: 2.2.2 Build: 991)

## Contents

TouchTreeGrid used for Slide Navigation Menu.....	4
SAMPLE: Basic Accordion and Standard Grid Examples (with Column Sort).....	5
SAMPLE: Standard Grid with Grouper, Categorized and Horizontal Scrolling.....	5
SAMPLE: Project Status – 3 Variations using CSS.....	6
SAMPLE: US Census – Phone Portrait .....	6
SAMPLE: US Census – Phone Landscape.....	7
SAMPLE: US Census – Tablet Landscape with TreeStore Filtering.....	7
SAMPLE: US Census – Tablet Portrait .....	7
SAMPLE: US Census – Tablet Landscape TreeGrid with Freeze Column.....	8
Intended Audience.....	9
Credits .....	9
Basic Features .....	10
Advanced Features .....	12
Summary of Provided Examples .....	19
Touch 2.3 Menu Implementation using TouchTreeGrid.....	19
TaskList => Accordion.....	20

# TouchTreeGrid

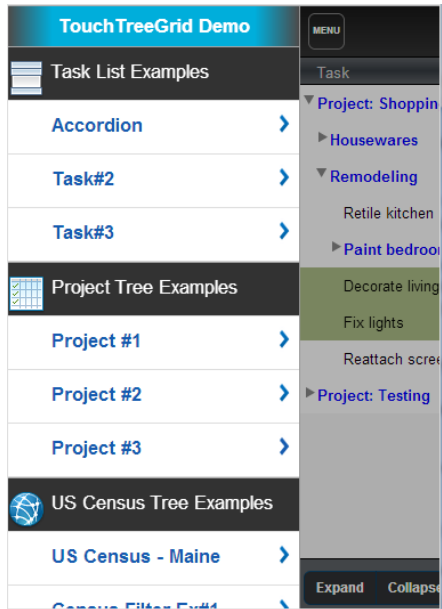
---

TaskList => Tasks#2 .....	22
TaskList => Tasks#3 .....	24
Project => Ex#1.....	26
Project => Ex#2.....	28
Project => Ex#3.....	30
Census Grid .....	33
Census Filter Example #1 .....	36
Census Freeze Column Example .....	38
Lists => Basic #1 (DOW History - 2012 Closing Prices data) .....	41
Lists => Basic #2.....	43
Lists => Grouper (DOW History Example with Grouper feature).....	48
Lists => Grouper#2 (TreeGrid serving 'grouper' feature with expand/collapse).....	51
Lists => Horiz (DOW History Example with Horizontal Scrolling).....	54
Lists => Dynamic (Same as Horiz but defined dynamically from Server).....	57
Override Grid .....	60
Notes on Implementation.....	62
Additional Functionality included in Advanced Example:.....	65
APPENDIX A – Columns Array Definitions.....	70
APPENDIX B – CSS Styling.....	72
APPENDIX C – Upgrading TouchTreeGrid component.....	80
APPENDIX D – TouchTreeGrid Config Definitions.....	82

# TouchTreeGrid

## TouchTreeGrid used for Slide Navigation Menu

Phone:



Tablets:

The screenshot shows a tablet application interface. On the left is a slide navigation menu with a blue header 'TouchTreeGrid Demo'. The menu has three main sections: 'Task List Examples' (with sub-items 'Accordion', 'Task#2', 'Task#3'), 'Project Tree Examples' (with sub-items 'Project #1', 'Project #2', 'Project #3'), and 'US Census Tree Examples' (with sub-items 'US Census - Maine', 'Census Filter Ex#1'). On the right, the main content area shows a table view. The table has a header 'Project #3' and columns 'Task', 'User', 'Dur', and 'Done?'. The table data is as follows:

Task	User	Dur	Done?
Project: Shopping	Tommy Maintz	13.25	
Housewares	Tommy Maintz	1.25	
Kitchen supplies	Tommy Maintz	0.25	
Groceries	Tommy Maintz	0.4	<input checked="" type="checkbox"/>
Cleaning supplies	Tommy Maintz	0.4	
Office supplies	Tommy Maintz	0.2	
Remodeling	Tommy Maintz	12	
Retile kitchen	Tommy Maintz	6.5	
Paint bedroom	Tommy Maintz	2.75	
Decorate living room	Tommy Maintz	2.75	<input checked="" type="checkbox"/>
Fix lights	Tommy Maintz	0.75	<input checked="" type="checkbox"/>
Reattach screen door	Tommy Maintz	2	
Project: Testing	Core Team	2	

At the bottom of the main content area, there are 'Expand' and 'Collapse' buttons, and a status bar showing '2' and '3'.

# TouchTreeGrid

## SAMPLE: Basic Accordion and Standard Grid Examples (with Column Sort)

TouchTreeGrid Demo		Basic List #2				
<div> <div>?</div> <div>Accordion</div> <div>Tasks#2</div> <div>Tasks#3</div> </div>		<div> <div>MENU</div> <div>Basic List #2</div> <div>?</div> </div>				
		CloseDate	Close	Chg	Chg%	Flag
▶ Today		12/28/2012	12,938	-158	-1.21%	<input type="checkbox"/>
▼ Tomorrow		12/27/2012	13,096	-18	-0.14%	<input type="checkbox"/>
Frisbee		12/26/2012	13,115	-24	-0.19%	<input type="checkbox"/>
Cookout		12/24/2012	13,139	-52	-0.39%	<input type="checkbox"/>
		12/21/2012	13,191	-121	-0.91%	<input type="checkbox"/>
▶ This week		12/20/2012	13,312	60	0.45%	<input checked="" type="checkbox"/>
▶ Later		12/19/2012	13,252	-99	-0.74%	<input checked="" type="checkbox"/>
		12/18/2012	13,351	116	0.87%	<input checked="" type="checkbox"/>
		12/17/2012	13,235	100	0.76%	<input type="checkbox"/>
		12/14/2012	13,135	-36	-0.27%	<input type="checkbox"/>
		12/13/2012	13,171	-75	-0.56%	<input type="checkbox"/>
		12/12/2012	13,245	-3	-0.02%	<input type="checkbox"/>
		12/11/2012	13,248	79	0.60%	<input type="checkbox"/>

Expand

Collapse

TaskList

Project

CENSUS

Lists...

Override

Basic List #2 demonstrates checkcolumn and custom row rendering (highlight rows Close Price > 13200).

## SAMPLE: Standard Grid with Grouper, Categorized and Horizontal Scrolling

TouchTreeGrid Demo					TouchTreeGrid Demo					TouchTreeGrid Demo					
<div> <div>?</div> <div>Basic</div> <div>Grouper</div> <div>Horiz</div> <div>Dynamic</div> </div>					<div> <div>?</div> <div>Basic</div> <div>Grouper</div> <div>Grp#2</div> <div>Horiz</div> <div>Dynamic</div> </div>					<div> <div>?</div> <div>Basic</div> <div>Grouper</div> <div>Horiz</div> <div>Dynamic</div> </div>					
CloseDate	Close	Chg	Chg%		CloseDate	Close	Chg	Chg%		Close Date	Open Price	High Price	Low Price	Close Price	Vol
December (Max: 13350.96 / Min: 12938.11 )					▶ December (Max: 13350.96 / Min: 12938.11 )					12/18/2012	13,237	13,366	13,233	13,351	1,5
12/18/2012	13,351	116	0.87%							12/19/2012	13,351	13,358	13,252	13,252	1,4
12/20/2012	13,312	60	0.45%		▼ October (Max: 13610.15 / Min: 13077.34 )					12/12/2012	13,250	13,329	13,227	13,245	1,2
12/19/2012	13,252	-99	-0.74%		10/31/2012	13,096	-11	-0.08%		12/20/2012	13,247	13,315	13,216	13,312	1,1
12/11/2012	13,248	79	0.60%		10/26/2012	13,107	4	0.03%		12/21/2012	13,310	13,310	13,123	13,191	4,1
12/12/2012	13,245	-3	-0.02%		10/25/2012	13,104	26	0.20%		12/11/2012	13,170	13,307	13,170	13,248	1,2
12/17/2012	13,235	100	0.76%												

Expand

Collapse

TaskList

Project

CENSUS

Lists...

Override

# TouchTreeGrid

## SAMPLE: Project Status – 3 Variations using CSS

Menu

Project #1

?

Task	User	Dur.
▼ Project: Shopping	Tommy Maintz	13.25
► Housewares	Tommy Maintz	1.25
▼ Remodeling	Tommy Maintz	12
Retile kitchen	Tommy Maintz	6.5
► Paint bedroom	Tommy Maintz	2.75
Decorate living room	Tommy Maintz	2.75
Fix lights	Tommy Maintz	0.75
Reattach screen door	Tommy Maintz	2
► Project: Testing	Core Team	2

Expand

Collapse

2

3

Menu

Project #2

?

Task	User	Dur.
▼ Project: Shopping	Tommy Maintz	13.25
► Housewares	Tommy Maintz	1.25
▼ Remodeling	Tommy Maintz	12
Retile kitchen	Tommy Maintz	6.5
► Paint bedroom	Tommy Maintz	2.75
Decorate living room	Tommy Maintz	2.75
Fix lights	Tommy Maintz	0.75
Reattach screen door	Tommy Maintz	2
► Project: Testing	Core Team	2

Expand

Collapse

2

3

Menu

Project #3

?

Task	User	Dur.	Done?
▼ Project: Shopping	Tommy Maintz	13.25	
▼ Housewares	Tommy Maintz	1.25	✓
Kitchen supplies	Tommy Maintz	0.25	✓
Groceries	Tommy Maintz	0.4	✓
Cleaning supplies	Tommy Maintz	0.4	✓
Office supplies	Tommy Maintz	0.2	✓
▼ Remodeling	Tommy Maintz	12	
Retile kitchen	Tommy Maintz	6.5	
► Paint bedroom	Tommy Maintz	2.75	
Ceiling	Tommy Maintz	1.25	✓
Walls	Tommy Maintz	1.5	

Expand

Collapse

2

3

Project #3 demonstrates checkcolumn feature and conditional row highlighting for Tree Grids.

## SAMPLE: US Census – Phone Portrait

Menu

US Census - Maine

?

Population

▼ Maine1,274,923

► Androscoggin County103,793

► Aroostook County73,938

► Cumberland County265,612

► Franklin County29,467

► Hancock County51,791

► Kennebec County117,114

ExpandCollapse2

Menu

US Census - Maine

?

Back

Androscoggin County

NumberPercent

Total Population103,793100.0%

Male50,38548.5%

Female53,40851.5%

Under 5 years6,1225.9%

5 to 96,8986.6%

10 to 147,4147.1%

15 to 197,4637.2%

20 to 246,3946.2%

# TouchTreeGrid

## SAMPLE: US Census – Phone Landscape

US Census - Maine				
	Population	Males	Females	Median
▼ Maine	1,274,923	620,309	654,614	38
▶ Androscoggin County	103,793	50,385	53,408	37
▶ Aroostook County	73,938	36,095	37,843	40
▶ Cumberland County	265,612	128,589	137,023	37
Expand Collapse 2				

## SAMPLE: US Census – Tablet Landscape with TreeStore Filtering

Census Filter Ex#1									
	Population	Males	Females	Median	18+	21+	62+	65+	
▼ Maine	1,274,923	620,309	654,614	38	973,685	924,108	215,732	183,402	→
Androscoggin Coun	103,793	50,385	53,408	37	78,957	74,488	17,432	14,962	→
Aroostook County	73,938	36,095	37,843	40	57,218	54,367	14,772	12,551	→
▼ Cumberland County	265,612	128,589	137,023	37	203,650	193,206	41,129	35,324	→
Brunswick town	21,172	10,210	10,962	35	16,301	14,676	3,773	3,272	→
Falmouth town	10,310	4,911	5,399	40	7,499	7,323	1,904	1,667	→
Gorham town	14,141	6,846	7,295	34	10,479	9,237	1,661	1,413	→
Scarborough town	16,970	8,296	8,674	38	12,571	12,203	2,606	2,211	→
Windham town	14,904	7,562	7,342	36	11,282	10,794	1,761	1,473	→
Expand Collapse Population > 10k									

Census Filter Example

>Variation of Census Maine Example.

>Applies easy to implement TreeStore filtering using provided TreeStore generation algorithm where any custom filter function can be defined (refer to Discussion of this example in "TouchTreeGrid - Documentation.pdf")

>Test Filter Options:  
"Males > Females"  
"Females > Males"  
"Population > 10k"

>Only children that pass filter are included.

>Nodes are always included if any of their children are included

>Option to display

## SAMPLE: US Census – Tablet Portrait

(# of columns auto-updated as orientation is changed from Landscape to Portrait)

# TouchTreeGrid

Census Filter Ex#1						
	Population	Males	Females	Median	18+	21+
▼ <b>Maine</b>	1,274,923	620,309	654,614	38	973,685	924,108
Androscoggin C	103,793	50,385	53,408	37	78,957	74,488
Aroostook Coun	73,938	36,095	37,843	40	57,218	54,367
▼ <b>Cumberland Co</b>	265,612	128,589	137,023	37	203,650	193,206
Brunswick tow	21,172	10,210	10,962	35	16,301	14,676
Falmouth town	10,310	4,911	5,399	40	7,499	7,323
Gorham town	14,141	6,846	7,295	34	10,479	9,237
Scarborough t	16,970	8,296	8,674	38	12,571	12,203
Windham town	14,904	7,562	7,342	36	11,282	10,794
Franklin County	29,467	14,228	15,239	38	22,538	20,827
Hancock County	51,791	25,324	26,467	40	40,248	38,366

Expand Collapse Population > 10k

## SAMPLE: US Census – Tablet Landscape TreeGrid with Freeze Column

Freeze Column						Freeze Column Example
Region	Total	Male	Female	<5yrs	5-9	
▼ <b>Maine</b>	1,274,923	620,309	654,614	70,726	83,022	<p>&gt;Horizontal scrolling with Freeze Column for large 600 data row with 28 data elements per row Tree Grid example.</p> <p>&gt;Multiple freeze columns can be implemented simply by instantiating new instances of TouchTreeGrid for specific columns and referencing the same TreeStore</p> <p>&gt;Recommended configs for all columns when specifying widths in pixels or em's:  code&gt;categIndentPct='0',  code&gt;colNumberToTruncateForIndex</p> <p>&gt;Recommended configs for 1st displayed fixed column container:  code&gt;width/minWidth a little larger than the actual column width for right padding,  code&gt;arrowPctWidth='6' (or large enough for reduced display width)</p>
▶ <b>Androscoggin County</b>	103,793	50,385	53,408	6,122	6,898	
▶ <b>Aroostook County</b>	73,938	36,095	37,843	3,730	4,459	
▶ <b>Cumberland County</b>	265,612	128,589	137,023	15,443	17,507	
▶ <b>Franklin County</b>	29,467	14,228	15,239	1,514	1,856	
▼ <b>Hancock County</b>	51,791	25,324	26,467	2,516	3,069	
Amherst town	230	127	103	9	9	
Aurora town	121	67	54	11	10	
Bar Harbor town	4,820	2,249	2,571	218	242	
Blue Hill town	2,390	1,128	1,262	85	134	

Expand Collapse 2



# TouchTreeGrid

---

## Intended Audience

This document is laid out in such a way that it is hopefully useful for beginner and advanced developers as well as for designers who want to learn what the capabilities are.

## Credits

TouchTreeGrid design was modeled after Mitchell Simoens's extremely popular Ext.ux.touch.grid (<https://github.com/mitchellsimoens/Ext.ux.touch.grid> and <http://www.sencha.com/forum/showthread.php?150431-Ext.ux.touch.grid>)

Many thanks to Shinobu Kawano's initial work on how tree stores can be used in Sencha Touch for this purpose (<https://github.com/kawanoshinobu/Ext.ux.AccordionList>).

Thank you to Sencha Team for their continued work on such an awesome suite of products! Of course if you are reading this then you already know that ☺

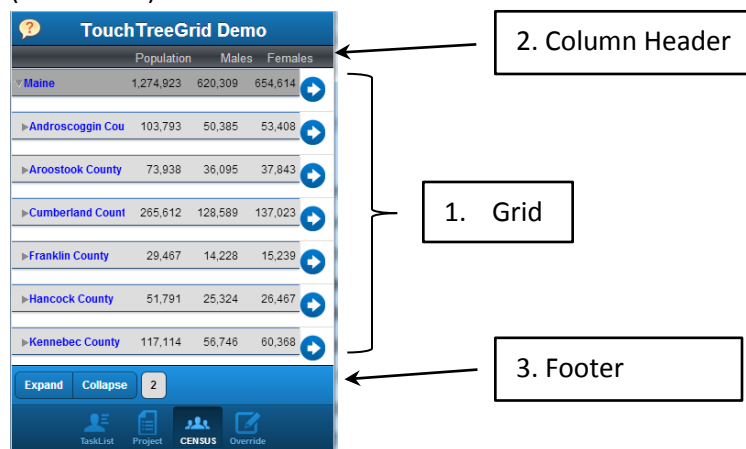
# TouchTreeGrid

TouchTreeGrid includes the following **Basic** and **Advanced** features:

## Basic Features

- The TouchTreeGrid component automatically creates following components simply by linking it as a child to a parent container and specifying a few configurations:

1. Grid displaying Detail and collapsible Category rows
2. Optional header row for column headers
3. Optional footer containing buttons to expand all, collapse all, collapse to specific depth (aka "level").



Category rows are indented for each depth. A smaller arrow icon is used to minimize display consumption. By default the first column of data is truncated for indented depths so that subsequent columns line up.

- **Following are minimum recommended configurations to create a Tree Grid:**

1. store- must of type Ext.data.TreeStore
2. defaultCollapseLevel = 1,2 3, etc... (99 for fully expanded)
3. itemId and listItemId
4. columns array (refer below)

- **Following are minimum recommended configurations to create a basic Grid:**

1. store- any store type can be used except Ext.data.TreeStore
2. simpleList = true
3. columnSorting = true
4. itemId and listItemId
5. columns array (refer below)

# TouchTreeGrid

---

- **Following are minimum recommended configurations to create a basic single expanded style Accordion:**

1. **store- must of type Ext.data.TreeStore**
2. **singleExpand = true**
3. **defaultCollapseLevel = 1**
4. **itemId and listItemId**
5. **columns array (refer below)**

- Column configurations are defined within a COLUMNS array (refer to Appendix A for examples and complete list):

1. header - Text to include in column header
2. dataIndex - data column from treestore/store
3. width - CSS format width for this component (suggest % or em's)
4. style - Additional CSS styling commands for detail rows
5. categStyle - Additional CSS styling commands for category rows  
(categStyle not used/required for Simple Grids)
6. headerStyle - Additional CSS styling commands for column header row
7. sortable - allow sorting for this column

- Column arrays can be defined and applied based on device configuration. For example Phone portrait, Phone landscape, Tablet portrait, Tablet landscape could each display a different number of columns based on screen width. Refer to Census example to see how number of grid columns is auto-updated as device is rotated. Run phone vs. tablet in browser and resize to simulate orientation similar to:

[http://localhost/TouchTreeGrid-master/TouchTreeGrid\\_Advanced/app.html?deviceType=Phone](http://localhost/TouchTreeGrid-master/TouchTreeGrid_Advanced/app.html?deviceType=Phone)

[http://localhost/TouchTreeGrid-master/TouchTreeGrid\\_Advanced/app.html?deviceType=Tablet](http://localhost/TouchTreeGrid-master/TouchTreeGrid_Advanced/app.html?deviceType=Tablet)

For larger applications it is suggested that you store all column configurations in a table on the server. The column arrays could be returned in the same back-end request that returns the data. Definitions would include column configurations, identifier for each grid, device/orientation, and possibly client-specific configurations. This reduces the footprint size that is downloaded on the device and allows for easy configuration changes without a code change.

- Disclosure icon can be included on detail and category rows where the “disclose” event would be processed within a controller (onItemDisclosure=true).
- All styling defined within CSS file (refer to Appendix B for detailed documentation)

# TouchTreeGrid

---

## Advanced Features

- Refer to [CalendarPicker.pdf](#) for advanced usage and documentation of TouchTreeGrid.
- Refer to APPENDIX D for detailed config documentation
  - Special note regarding “**additionalListConfigs**” - Any config supported by Ext.dataview.List component not directly supported in Appendix D can be applied to the auto-generated Grid list component. Configs defined via additionalListConfigs would override any similar configs defaulted by TouchTreeGrid.
- Option to assign unique colors to Category rows by expansion depth:  

```
categDepthColorsArr: ['#808127', '#949569', '#C5C678'] <= Depths 1,2 and 3 colors
```
- Specify the screen width percentage stepsize for each expanded depth:
  - categIndentPct – defaults to 3%
  - colNumberToTruncateForIndents – by default column #1 is truncated by (categIndentPct \* depth) to line up subsequent columns.
- Override default screen width taken by expand/collapse arrow via arrowPctWidth config:
  - Default = 4%
  - 2% would be sufficient for Tablet displays
- Item Disclosure functionality:
  - Specify display of disclose icon for leafs only via disclosureProperty='leaf'
  - Use CSS class “.touchtreegrid-disclose-spacer “ to provide percent width spacing in header row to line up columns when disclosure icon is displayed (refer to Appendix B)
- One or more synchronized sets of scrolling columns supported. Both Freeze and Horizontal Scrolling columns supported. Refer to following configs in Appendix D and documented Census Freeze Column example for more information:
  - linkedGridsParentItemId
  - linkedGridsArr[]
- Custom column render functions are supported a little differently than in Mitchell Simoen’s Ext.ux.touch.grid.

# TouchTreeGrid

---

- Instead of defining the function within the Columns Array for each column, an list of custom render functions can be defined in 'renderers' object within each linked instance of TouchTreeGrid<sup>2</sup>:

```
renderers:
{
  renderer_displayIn1000s: function(value)
  {return this.formatNumbers(Math.round(Number(value)/1000), 0);},

  renderer_formatWithColor: function (value)
  {var clr = (value >= 0) ? 'green' : 'red';
   return '<span style="color:' + clr + ';">' + this.formatNumbers(value,0) +
   '</span>';}
},
```

- Examples and Notes regarding functions:
  - `this.renderer_displayIn1000s(1234567) = 1,235`
  - `this.renderer_formatWithColor(-23.45)` displays cell value in **red** (vs. **green** for positive)
  - custom function parameters are passed in the function calling string (refer below)
  - You could add all common functions used across your application directly in TouchTreeGrid.js renderer object and they would be available for all grid implementations (but take care when updating new version of this component!)  
... I will come up with better way to do this in future release to simplify upgrades.
- Note that custom rendering functions can call other built-in functions or `formatNumbers()` function provided with TouchTreeGrid (refer below)
- The Columns array would include a string representing the function call to make:

```
columns : [{
  header: 'Holdings(k)',
  dataIndex: 'HOLDINGS',
  width: '15%',
  style: 'text-align: right;',
  categStyle: 'text-align: right;',
  headerStyle: 'text-align: right; color: #ccc;',
  renderer: 'this.renderer_displayIn1000s(values.HOLDINGS)'
},
[
  header: '%Change',
  dataIndex: 'PCT_CHG',
  width: '15%',
  style: 'text-align: right;',
  categStyle: 'text-align: right;',
  headerStyle: 'text-align: right; color: #ccc;',
  renderer: 'this.renderer_formatWithColor (values.PCT_CHG)'
],
```

---

<sup>2</sup> One reason for this is that we can't easily store and pass functions to column array if stored in a table on the server. This also simplifies repetitive assignments within the array

# TouchTreeGrid

---

Etc...

- Notes regarding renderer function calls:
  - 'values' is an object containing elements for every item of each data record. Hence, every data field value is available for use within renderer functions.
  - According to Ext.data.NodeInterface documentation the following fields are automatically added to the tree store upon load if not already existing and can also be referenced:  
parentId, index, **depth**, **expanded**, expandable, **checked**, **leaf**, cls, iconCls, root, isLast, isFirst, allowDrop, allowDrag, loaded, loading, href, hrefTarget, qtip, qtitle
- Built-in **formatNumbers(value, decPlaces, prefix, suffix, thouSeparator, decSeparator)** column formatting function provided with TouchTreeGrid
  - Parameters (all fields optional except value)
    - value – number to format
    - decPlaces - # decimals places (default is 2 decimals if omitted)
    - prefix- use to prefix '\$' sign or other to number
    - suffix – use to append '%' sign or other to number
    - thouSeparator – default = ','
    - decSeparator – default = '.'
  - Examples
    - renderer: 'this.formatNumbers(values.TotalPopulation, 0)' => 1,234,567
    - renderer: 'this.formatNumbers(values.NAV\_PCT, 4, "", "%")' => 12.3456%
    - renderer: 'this.formatNumbers(values.NAV\_AMT, 0, "\$")' => \$1,234,567
- Events fired for each item tap for custom processing: **"leafItemTap"** and **"nodeItemtap"**. See Task example controller functions onTask2LeafItemTap () and onTask2NodeItemTap ()
- Default sort functionality can be overridden by specifying **customColumnSortEvent** config. Event specified by this config will be fired and can be processed within controller for custom sorting. This allows for custom column sort functionality without modifying the component. Example would be when sort needs to occur on the server-side or to toggle between more states: ASC, DESC, ABS(ASC), ABS(DESC), etc... Refer to logic in handleColumnSort() method within TouchTreeGrid for parameters passed with event.
- Following additional styling configs supported in columns[] array for highlighting sorted columns:
  - styleSorted (overrides 'style' for sorted column)
  - cateStyleSorted (overrides 'cateStyle')

# TouchTreeGrid

- headerStyleSorted (overrides 'headerStyle')
- categColumns[] config array supported as means to defined category styling totally unique from columns defined by width attribute in columns[] config array. This allows us for example to use TreeGrid to support “grouper” functionality , with expand/collapse functionality. The grouper expression can be any long text totally independent from leaf rows:

The screenshot shows a mobile application interface titled "TouchTreeGrid Demo". It features a top navigation bar with a question mark icon and five tabs: "Basic", "Grouper", "Grp#2", "Horiz", and "Dynamic". Below the tabs is a table with columns: "CloseDate", "Close", "Chg", and "Chg%". The table displays a tree structure with expandable rows for "December", "November", and "October". Each month row shows a range of values in parentheses, e.g., "December (Max: 13350.96 / Min: 12938.11)". Below the table are "Expand" and "Collapse" buttons. At the bottom is a footer with icons and labels for "TaskList", "Project", "CENSUS", "Lists...", and "Override".

CloseDate	Close	Chg	Chg%
▶ December (Max: 13350.96 / Min: 12938.11 )			
▶ November (Max: 13245.68 / Min: 12542.38 )			
▼ October (Max: 13610.15 / Min: 13077.34 )			
10/31/2012	13,096	-11	-0.08%
10/26/2012	13,107	4	0.03%
10/25/2012	13,104	26	0.20%

- Supports **disableExpandCollapse** config to disable expand/collapse functionality.
- PullRefresh implemented as follows:
  - Define 'plugins' object in “linked” TouchTreeGrid (also refer to Example2)

```
itemId: myExample,  
listPlugins: {  
    xclass: 'Ext.plugin.PullRefresh',  
    listeners : {  
        latestfetched : function()  
{this.up('touchtreegrid').fireEvent('pullrefresh', this.up('touchtreegrid'));}  
    }  
}  
/* refreshFn not supported for Touch 2.2 */  
/* refreshFn: function(plugin) {this.up('touchtreegrid').fireEvent('pullrefresh',  
this.up('touchtreegrid'));} */  
}
```

- Within ProjectController:

```
refs: {  
    example2: '#example2',  
  
    control: {  
        "container#example2": {  
            pullrefresh: 'onExample2ListPullrefresh'  
        },  
    },  
}
```

- onMyExampleListPullrefresh() function
  - calls loadExample2Store() which is same function used to initially load data

# TouchTreeGrid

---

- Implement custom Expand/Collapse functionality. By default Touch's built-in Node expand/collapse methods are used. For larger data sets I found it was faster to reset the expand depths manually in the controller and reload the data. Hence, a "customExpCollapseEvent" configuration is provided for this purpose. If defined, an event by the specified name will be fired (i.e. instead of default collapse logic) which can be picked up in the controller.

Note: actual example not provided, but following is guide for implementation:

- Within linked TouchTreeGrid instance
  - customExpCollapseEvent : 'myExampleExpCollapseEvent'
  - itemId : 'myExample'
- Within Controller:

```
refs: {  
    myExample: '#myExample', etc...}  
  
control : {  
    "container#myExample": {  
        myExampleExpCollapseEvent: 'onMyExampleExpCollapse'  
    }, etc...}
```

- 'onMyExampleExpCollapse' function
    - Rebuild tree specifying 'expanded' attribute
    - Update treestore
- 
- Define what depth to auto-expand/collapse depths to upon initialization/refresh:
    - Specify default collapse-to depth via: defaultCollapseLevel = 0 to 99
      - 0 for fully collapsed
      - 1 to collapse to 1<sup>st</sup> level (would be same as 0 if a single root node did not exist)
      - 99 for fully expanded (default)
    - Disable this feature via: applyDefaultCollapseLevel=false  
(this would be used if collapse level defined when tree is created)
- 
- Control display of collapse-to buttons
    - Disable display via, includeFooterLevels = false (default is true)
    - Disable application of colors defined by categDepthColorsArr[] for category row colors to the collapse-to buttons via, categoryDepthColorButtons= false (default = true)
- 
- List item **pressedCls** is supported and defaults to: 'touchtreegrid-item-pressed'
    - If disableSelection=true, the pressedCls is still applied momentarily as user is pressing a list item for disclosure.
    - TouchTreeGrid updateStore function contains logic to update this to empty string if disabledSelection=true thereby not applying any class to the press event.
    - Otherwise user can specify own pressedCls and define in CSS file if disableSelection=false.



# TouchTreeGrid

---

- Refer to TouchTreeGrid.css for examples on it's use.
- List item selectedCls is supported and defaults to: 'touchtreegrid-item-selected'
  - Refer to TouchTreeGrid.css for examples on it's use.
- Manual HTML specifications for Header, Category and Detail rows
  - Manually provide HTML in following configurations of linked instance of TouchTreeGrid: headerTplOverride, categlItemTplOverride, contentItemTplOverride
  - Within Architect's config panel you can specify the data type as Object to edit formatted text
  - Refer to "Manual" example.
- itemHeight and variableHeights
  - List itemHeight within TouchTreeGrid defaults to Sencha's default value of 47 pixels and can be updated. However, since only pixels can be defined at time of this writing (because of scroller implementation as I understand), I found that playing with this number does not always render as expected on some displays (Android for example). I would wait until 'em' is supported before using this.
  - variableHeights = true is the default. Refer to Sencha documentation, but as I understand setting to false along with customizing itemHeight improves performance.
- Header, Category and Detail (aka 'content') rows all utilize outer <div> element which represents the shaded portion of category rows for instance:
  - <div style="display: -webkit-box; -webkit-box-orient: horizontal;">
  - Additional custom styling can be appended to this div statement specifying one or more of the following congurations:
    - styleHeaderRow
    - styleCategRow (not applicable for simpleList = true)
    - styleContentRow
- If you want to change the "Recycle" icon when rotating phones to landscape (TabPanel Advanced example only) you can specify your own url in the linked instance of TouchTreeGrid

# TouchTreeGrid

---

via, landscapelcon = './resources/images/mylcon.png'

	Population	Males	Females	Median	
▼ <a href="#">Maine</a>	1,274,923	620,309	654,614	39	➔
▶ <a href="#">Androscoggin County</a>	103,793	50,385	53,408	37	➔
▶ <a href="#">Aroostook County</a>	73,938	36,095	37,843	41	➔
▶ <a href="#">Cumberland County</a>	265,612	128,589	137,023	38	➔
▶ <a href="#">Franklin County</a>	29,467	14,228	15,239	38	➔

Expand Collapse 2 Rotate for Menu

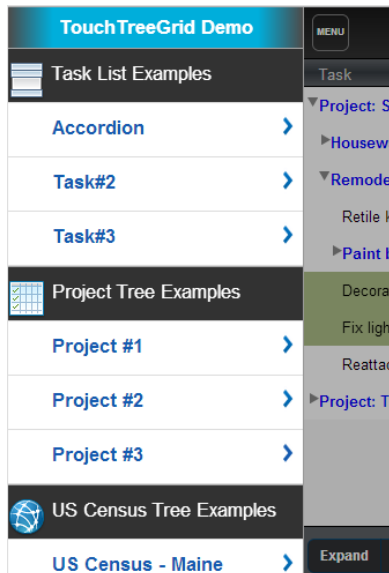
# TouchTreeGrid

---

## Summary of Provided Examples

(refer below for detailed configuration documentation)

### Touch 2.3 Menu Implementation using TouchTreeGrid



Touch 2.3 Ext.Menu Component is implemented as follows:

(Also refer to ./MenuEx/ project directory for simplified example of Menu implementation)

1. MainContainer.js is the main view for the project
  - a. Requires: 'Ext.Menu'
  - b. Contains menu-related methods: doSetHidden(), menuForSide() and myMenu()
  - c. doSetHidden(hidden) method called once by the framework when MainContainer.js is defined. Following configuration options are supported:
    - i. cover = false (slide navigation)
    - ii. reveal = true (reveal navigation .. menu appears to be underneath current window)
    - iii. cover = true (cover navigation ... menu appears overtop current window)
  - d. doSetHidden calls menuForSide('left') method
  - e. menuForSide() method creates initial menu from container defined in myMenu() method
  - f. myMenu() method creates menu container as non-collapsible TouchTreeGrid menu with title container
2. CommonController.js launch() method initializes selection of "Accordion" menu option via:

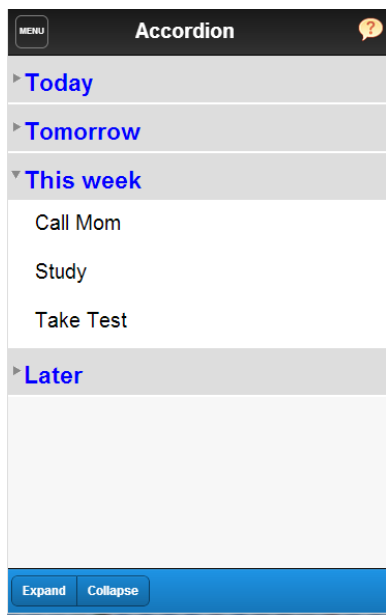
```
var list = this.getMenuopt().down('#menuoptlist');
var idx = list.getStore().find('Text', 'Accordion');
list.select(idx);    <= fires list "select" event
```

# TouchTreeGrid

---

3. CommonController.js onMenuOptListSelect() method listens for “select” event on #menuoptlist list item
4. onMenuOptListSelect() processes each menu selection (including initialized selection) as follows:
  - a. Adds appropriate grid example view to #menucontswap container within MainContainer.js
  - b. Toggles menu via: Ext.Viewport.toggleMenu('left');
    - i. Initially menu is shown at launch time since it was not present.
    - ii. Subsequent menu selections cause menu to be hidden.
  - c. Updates appropriate help menu
5. Menu button processed by onMenuButtonTap() method within CommonController.js
  - a. Calls Ext.Viewport.toggleMenu('left');
  - b. Menu is restored

## TaskList => Accordion



Basic accordion example demonstrates:

- Single expand accordion style treegrid
- No lines separating leaf rows

Program flow:

- Controller: TasksController.js
- TabPanel example View config: Main.js => TasksContainer.js => itemId = 'firstexample'
- Slide Navigation example: AccordionCont.js

# TouchTreeGrid

---

- Data:
  1. TouchTreeGrid.store.TaskAccordion (storeId = TaskAccordionStore)
  2. Model: TouchTreeGrid.model.Task
  3. TreeStore (data defined directly in root property)
  4. Autoload = true

Notable configs:

- singleExpand = true
- defaultCollapseLevel = 1
- style: 'text-align: left; font-size: 1.5em;'
- cateStyle: 'font-weight: bold; font-size: 2em; padding-top: .2em; text-align: left; color: blue;'
- cls: ['x-touchtreegrid-list', 'x-touchtreegrid-list-accordion']

Notable CSS (defaults defined in TouchTreeGrid.css, customized by treegriddemo.css)

```
/* 2.1 selectors to eliminate lines ... 2nd set is for Touch 2.2 */
.x-touchtreegrid-list-accordion .x-list-normal .x-list-item .x-dock-horizontal,
.x-touchtreegrid-list-accordion .x-list-normal .x-list-item.x-list-item-tpl{
    border: none;
}

.x-touchtreegrid-list-accordion .touchtreegrid-list-content{
    padding:.6em 0 0 0;
}

.x-touchtreegrid-list-accordion .touchtreegrid-list-categ {
    border-top: none;
    border-bottom: none;
    -webkit-box-shadow: none;
    min-height: 45px !important;
}

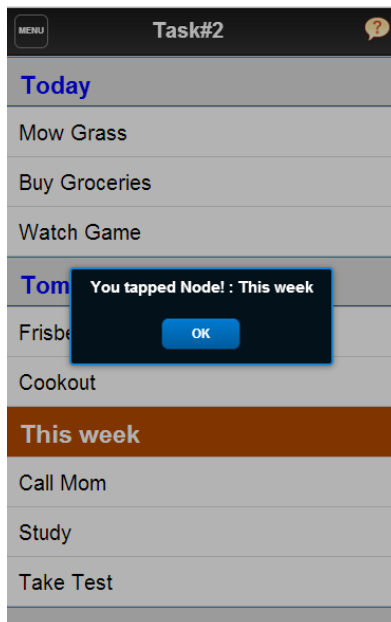
.x-touchtreegrid-list-accordion .touchtreegrid-details-img {
    /* category row arrow centered in row */
    margin-top: .5em !important;
}
```

Sample data stored directly in Root of store: TouchTreeGrid.store.TaskAccordion

# TouchTreeGrid

---

## TaskList => Tasks#2



- a. Demonstrates:
  - i. Disabled expand/collapse within scrolling window
  - ii. Leaf plus node row selection with custom CSS styling
- b. Program Flow
  - i. Controller: TasksController.js
  - ii. TabPanel example View config: Main.js => TasksContainer.js => itemId = 'task2'
  - iii. Slide Navigation example: Task2Cont.js
  - iv. Data:
    1. TouchTreeGrid.store.Task2Store (storeId = task2Store)
    2. Model: TouchTreeGrid.model.Task
    3. TreeStore (data defined directly in root property)
    4. Autoload = true
  - v. Events:

```
control: {...  
  "container#task2": { // Trapping leaf and node taps !!  
    leafItemTap: 'onTask2LeafItemTap',  
    nodeItemTap: 'onTask2NodeItemTap'  
  },  
}
```
- c. Notable Configs:
  - i. disableSelection = false
  - ii. arrowPctWidth = '0'

# TouchTreeGrid

---

iii. `disableExpandCollapse = true`

d. CSS (defaults defined in TouchTreeGrid.css, customized by treegriddemo.css)

```
.x-touchtreegrid-list-task2 .touchtreegrid-list-categ {
    min-height: 47px !important;
    /* Selected row is still painted, but underneath webkit box which is sized to hide selection */
}

/** Begin Task2 CSS required to support row selection for category and content rows */
.x-touchtreegrid-list-task2 .touchtreegrid-list-content{ /* Detail (leaf) rows */
    padding:.6em 0 0 0;
    background-color: transparent; /* Needed for row selection */
}

.x-touchtreegrid-list-task2 .x-touchtreegrid-item .touchtreegrid-item-selected,
.x-touchtreegrid-list-task2 .touchtreegrid-item-selected,
.x-touchtreegrid-list-task2 .touchtreegrid-item-selected .touchtreegrid-list-content-cell {
    background-color: #b64b00 !important; /* color for selected row */
    color: white !important; /* Change selected text color to white */
}

.x-touchtreegrid-list-task2 .touchtreegrid-item-pressed {
    background-color: #ecd2bf !important; /* color for selected row */
}

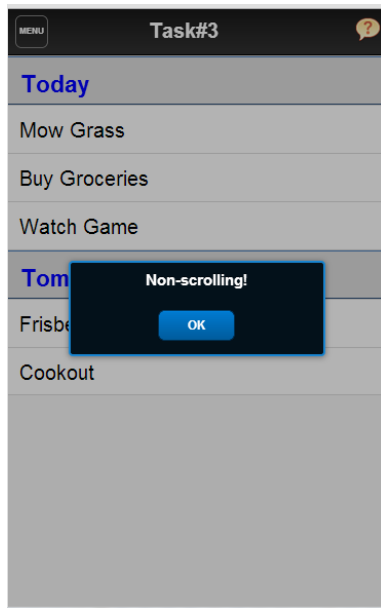
.x-touchtreegrid-list-task2 .touchtreegrid-item-selected .touchtreegrid-list-categ {
    background-color: transparent !important;
    color: white !important;
    border-top: none;
    border-bottom: none;
    -webkit-box-shadow: none;
}

.x-touchtreegrid-list-task2 .touchtreegrid-item-selected .touchtreegrid-list-categ-cell {
    color: white !important;
}
```

# TouchTreeGrid

---

## TaskList => Tasks#3



- a. Demonstrates
  - i. Same as Tasks#2 except non-scrolling and row selection for leafs only
- b. Program Flow
  - i. Controller: TasksController.js
  - ii. TabPanel example View config: Main.js => TasksContainer.js => itemId = 'task3'
  - iii. Slide Navigation example: Task3Cont.js
  - iv. Data:
    - i. TouchTreeGrid.store.Task3Store (storeId = task3Store)
    - ii. Model: TouchTreeGrid.model.Task
    - iii. TreeStore (data defined directly in root property)
    - iv. Autoload = true
  - v. Events:

```
control: {.....
  "tabpanel#tasksTabPanel": { // TabPanel example
    activeitemchange: 'onTasksTabpanelActiveItemChange'
  },
  "container#task3": { // Only trapping leaf taps for this example !
    leafItemTap: 'onTask3LeafItemTap'
  }
}
```

### Activeitemchange()

```
if (newcont === 'task3cont'){
```

```
// One way to disable as list config "scrollable: false" does not work as expected
```

```
grid.getScrollable().getScroller().setDisabled(true);
```

```
Ext.Msg.alert('Non-scrolling!');
```



# TouchTreeGrid

---

}

## c. Notable Configs

- i. disableSelection = false
- ii. disableExpandCollapse = false

## d. CSS (defaults defined in TouchTreeGrid.css, customized by treegriddemo.css)

```
.x-touchtreegrid-list-task3 .touchtreegrid-list-categ {
    min-height: 47px !important;
/* Selected row is still painted, but underneath webkit box which is sized to hide selection */
}

/** Begin Task2 CSS required to support row selection for category and content rows **/
.x-touchtreegrid-list-task3 .touchtreegrid-list-content{ /* Detail (leaf) rows */
    padding:.6em 0 0 0;
    background-color: transparent; /* Needed for row selection */
}

.x-touchtreegrid-list-task3 .x-touchtreegrid-item .touchtreegrid-item-selected,
.x-touchtreegrid-list-task3 .touchtreegrid-item-selected,
.x-touchtreegrid-list-task3 .touchtreegrid-item-selected .touchtreegrid-list-content-cell {
    background-color: #b64b00 !important; /* color for selected row */
    color: white !important; /* Change selected text color to white */
}

.x-touchtreegrid-list-task3 .touchtreegrid-item-pressed {
    background-color: #ecd2bf !important; /* color for selected row */
}
```

# TouchTreeGrid

---

## Project => Ex#1

Task	User	Dur
▼ Project: Shopping	Tommy Maintz	13.25
▶ Housewares	Tommy Maintz	1.25
▼ Remodeling	Tommy Maintz	12
Retile kitchen	Tommy Maintz	6.5
▶ Paint bedroom	Tommy Maintz	2.75
Decorate living roo	Tommy Maintz	2.75
Fix lights	Tommy Maintz	0.75
Reattach screen d	Tommy Maintz	2
▶ Project: Testing	Core Team	2

### a. Demonstrates

- Initially expands nodes as individually defined in JSON tree store
- color-specific category rows with matching colors on collapse-to buttons
- overridden color scheme for expand/collapse button and toolbar (treegriddemo.css)
- custom iPhone-style disclose icon (treegriddemo.css)  
Note: you may or may not want to use this icon depending on your experience with sensitivity when touching icon on your particular device.
- The example also implements handler for itemtaphold event which triggers when user presses anywhere on a disclosure row for more than one second as an alternative (refer to onExample2ListItemTaphold() function within controller).
- On Disclose slides in detail panel showing all data columns for selected row
- Disclose only implemented for detail (leaf) rows for this particular example.
- Implements pull-refresh which sends “pullrefresh” event to controller to re-load data

### b. Program Flow

- Controller: ProjectController.js
- TabPanel example View Config: Main.js => ProjectContainer.js => itemId = 'example2'
- Slide Navigation example: Example2Bcontainer.js
- Data:
  - i. TouchTreeGrid.store.ProjectsStore (storeId = projectsStore)
  - ii. Model: TouchTreeGrid.model.Projects

# TouchTreeGrid

- iii. TreeStore (./data/treegrid.json). Data loaded via CommonController within onMainTabpanelActiveItemChange() method

```
55 if (newcont === 'projecttab'){
56     // Check store for data and load if empty (only)
57     numNodes = grid.getStore().getData().length;
58     if (numNodes === 0) {projectController.loadExample2Store(gridcont);}
59 }
60 }
```

For TabPanel example, or within onMenuOptListSelect() method within CommonController for Slide Navigation example.

projectController.loadExample2Store => commonController.loadStore => commonController.postLoadProcess =>

```
else if ((gridListItemId === 'example2list') ||
(gridListItemId === 'example2Blist') ||
(gridListItemId === 'example2Clist')) {
    if (gridListItemId === 'example2list') {
        Ext.Msg.alert('Custom Expand levels!');
    }

    projectController.getProjectContainer().down('#example2list').up('touchtreegrid').doRefreshList();
    projectController.getProjectContainer().down('#example2Blist').up('touchtreegrid').doRefreshList();
    projectController.getProjectContainer().down('#example2Clist').up('touchtreegrid').doRefreshList();

    // workaround to get Touch 2.2 pullrefresh plugin to auto-snapBack
    scroller.scrollTo(0,1);
}
```

- Events:

```
control: {
    "list#example2list": {
        disclose: 'onExample2ListDisclose',
        itemtaphold: 'onExample2ListItemTaphold'
    },
    "container#example2": {
        pullrefresh: 'onExample2ListPullrefresh'
    }
}
```

## c. Notable Configs

- categDepthColors = true
- onItemDisclosure = true
- disclosureProperty = 'leaf' (disclosure for leafs, but not categories)
- categDepthColorsArr = ['#808127', '#949569', '#C5C678']
- applyDefaultCollapseLevel = false (expand levels defined in store)
- listPlugins = see code

## d. CSS (defaults defined in TouchTreeGrid.css, customized by treegriddemo.css)

```
/* Remove -or- rename this to use default Sencha disclosure icon if concerns with sensitivity of
smaller iphone style icon */
.x-touchtreegrid-list-example2 .x-list-disclosure{
    width: 12px;
    height: 1.5em;
    margin: .4em 0 0 .5em;
    -webkit-mask: none;
    -webkit-mask-box-image: (see treegriddemo.css...)
}
.x-touchtreegrid-list-example2 .x-list-disclosure:before{
/* Disable Touch 2.2 arrow picto for disclosure icon */
    content: '';
    font-family: '';
}
```

# TouchTreeGrid

## Project => Ex#2

Task	User	Dur
▼ Project: Shopping	Tommy Maintz	13.25
▶ Housewares	Tommy Maintz	1.25
▼ Remodeling	Tommy Maintz	12
Retile kitchen	Tommy Maintz	6.5
▶ Paint bedroom	Tommy Maintz	2.75
Decorate living roo	Tommy Maintz	2.75
Fix lights	Tommy Maintz	0.75
Reattach screen d	Tommy Maintz	2
▶ Project: Testing	Core Team	2

- a. Demonstrates
  - a. Variation of Project #1 where itemHeight=32 and variableHeights=false to tighten space between rows.
  - b. onDisclosure for leaf and category rows
  - c. Utilizes config categCssArr[] to specify different CSS selector for 1<sup>st</sup> category level from the rest.
    - i. categCssArr = ['touchtreegrid-list-categ0', 'touchtreegrid-list-categ', 'touchtreegrid-list-categ']
    - ii. CSS selector 'touchtreegrid-list-categ0' defined in treegriddemo.css and renders text in white italics
- b. Program Flow
  - a. Controller: ProjectController.js
  - b. View Config: Main.js => ProjectContainer.js => itemId = 'example2B'
  - c. Data:
    - i. TouchTreeGrid.store.ProjectsStore (storeId = projectsStore)
    - ii. Model: TouchTreeGrid.model.Projects
    - iii. TreeStore (./data/treegrid.json). Data loaded via CommonController within onMainTabpanelActiveItemChange() method

```
55 if (newcont === 'projecttab'){
56     // Check store for data and load if empty (only)
57     numNodes = grid.getStore().getData().length;
58     if (numNodes === 0) {projectController.loadExample2Store(gridcont);}
59 }
60
```

For TabPanel example, or within onMenuOptListSelect() method within CommonController for Slide Navigation example.

# TouchTreeGrid

---

projectController.loadExample2Store => commonController.loadStore =>  
commonController.postLoadProcess =>

```
else if ((gridListItemId === 'example2list') ||  
(gridListItemId === 'example2Blist') ||  
(gridListItemId === 'example2Clist')) {  
    if (gridListItemId === 'example2list') {  
        Ext.Msg.alert('Custom Expand Levels!');  
    }  
  
    projectController.getProjectContainer().down('#example2list').up('touchtreegrid').doRefreshList();  
    projectController.getProjectContainer().down('#example2Blist').up('touchtreegrid').doRefreshList();  
    projectController.getProjectContainer().down('#example2Clist').up('touchtreegrid').doRefreshList();  
  
    // workaround to get Touch 2.2 pullrefresh plugin to auto-snapBack  
    scroller.scrollTo(0,1);  
}
```

d. Events:

```
"list#example2Blist": {  
    disclose: 'onExample2BlistDisclose'  
},
```

c. Notable Configs

- categDepthColors = true
- onItemDisclosure = true
- categDepthColorsArr = ['#808127', '#949569', '#C5C678']
- categCssArr (discussed above)
- applyDefaultCollapseLevel = false (expand levels defined in store)

d. CSS (defaults defined in TouchTreeGrid.css, customized by treegriddemo.css)

```
.x-touchtreegrid-list-example2B .x-list-disclosure{  
    margin-top: .1em;  
}  
.x-touchtreegrid-list-example2B .x-list-normal .x-list-item .x-dock-horizontal,  
.x-touchtreegrid-list-example2B .x-list-normal .x-list-item.x-list-item-tp1 {  
    border: none;  
}  
.x-touchtreegrid-list-example2B .touchtreegrid-list-content{ /* Detail (leaf) rows */  
    padding:.6em 0 0 0;  
}  
  
.x-touchtreegrid-list-example2B .touchtreegrid-list-categ0 {  
/* Default for Category rows (and depth=0 category when custom categDepthColors=true) */  
    color: white;  
    font-weight: bold;  
    font-style: italic;  
  
    background-color: #ddd !important;  
    background-image: none !important;  
    border-top: 1px solid #4D80BC;  
    border-bottom: 1px solid #2D4E76;  
    min-height:2.6em !important;  
    padding:0.6em 0 0 0 !important;  
    font-size: .7em !important;  
    -webkit-box-shadow: 0px 0.1em 0.3em rgba(0, 0, 0, 0.3);  
}
```

# TouchTreeGrid

## Project => Ex#3

Task	User	Dur	Done?
▼ Project: Shopping	Tommy Maintz	13.25	
▼ Housewares	Tommy Maintz	1.25	✓
Kitchen supplies	Tommy Maintz	0.25	✓
Groceries	Tommy Maintz	0.4	✓
Cleaning supplies	Tommy Maintz	0.4	✓
Office supplies	Tommy Maintz	0.2	✓
▼ Remodeling	Tommy Maintz	12	
Retile kitchen	Tommy Maintz	6.5	
▼ Paint bedroom	Tommy Maintz	2.75	
Ceiling	Tommy Maintz	1.25	✓
Walls	Tommy Maintz	1.5	

### a. Demonstrates

- Variation of Project #2 where row shading is removed.
- Disclosure icons removed from all rows
- Coloring of Expand-to buttons to match category row colors disabled
- Slide Navigation example implements row rendering for leaf rows where done=true using **styleContentRow** config:  
**background-color: {[values.done && values.leaf ? "#adbe87" : "none"]};**
- Slide Navigation example also conditionally displays checkmark font using cell rendering (refer to List Basic #2 example for checkcolumn for standard grids):

#### i. Columns[] Array

```
{
  header: 'Done?',
  dataIndex: 'done',
  width: '25%',
  style: 'padding-left: 10%; margin: 0px 0 0px 0;',
  addDataIndexToDiv: true,
  // required to listen to taps on this column
  headerStyle: 'text-align: center; color: #ccc;',
  renderer: 'this.renderer_checkMark(values)'
}
```

#### ii. Renderer{} Object

```
{renderer_checkMark: function (values) {
  if (values.done && values.leaf) {
    return ( "<span class='flag-checked'> </span>");
  }
  else if (!values.done && values.leaf) {
    return ( "<span class='flag-not-checked'> </span>");
  }
  else if (values.done && !values.leaf) {
    return ( "<span class='node-flag-checked'> </span>");
  }
  else if (!values.done && !values.leaf) {
    return ( "<span class='node-flag-not-checked'> </span>");
  }
}
```

# TouchTreeGrid

---

### iii. CSS to support checkcolumn

(refer to <http://pictos.cc/font/> for fonts bundled with Touch)

```
.x-touchtreegrid-list-example2C .flag-checked:before {
    font-family: "Pictos";
    content: "3";
    color: gray;
    font-size: 18px;
    border: 1px solid gray;
}
.x-touchtreegrid-list-example2C .flag-not-checked:before {
    font-family: "Pictos";
    content: "3";
    color: transparent;
    font-size: 18px;
    border: 1px solid gray;
}
.x-touchtreegrid-list-example2C .node-flag-checked:before {
    font-family: "Pictos";
    content: "3";
    color: green;
    font-size: 22px;
}
.x-touchtreegrid-list-example2C .node-flag-not-checked:before {
    font-family: "Pictos";
    content: "3";
    color: transparent;
    font-size: 18px;
}
```

### b. Program Flow

- Controller: ProjectController.js
- View Config: Main.js => ProjectContainer.js => itemId = 'example2C'
- Data:
  - i. TouchTreeGrid.store.Task3Store (storeId = task3Store)
  - ii. Model: TouchTreeGrid.model.Task
  - iii. TreeStore (./data/treegrid.json). Data loaded via CommonController within onMainTabpanelActiveItemChange() method

```
55 if (newcont === 'projecttab'){
56     // Check store for data and load if empty (only)
57     numNodes = grid.getStore().getData().length;
58     if (numNodes === 0) {projectController.loadExample2Store(gridcont);}
59 }
60
```

For TabPanel example, or within onMenuOptListSelect() method within CommonController for Slide Navigation example.

projectController.loadExample2Store => commonController.loadStore => commonController.postLoadProcess =>

```
else if ((gridListItemId === 'example2list') ||
(gridListItemId === 'example2blist') ||
(gridListItemId === 'example2clist')) {
    if (gridListItemId === 'example2list') {
        Ext.Msg.alert('Custom Expand levels!');
    }

    projectController.getProjectContainer().down('#example2list').up('touchtreegrid').doRefreshList();
    projectController.getProjectContainer().down('#example2blist').up('touchtreegrid').doRefreshList();
    projectController.getProjectContainer().down('#example2clist').up('touchtreegrid').doRefreshList();

    // workaround to get Touch 2.2 pullrefresh plugin to auto-snapBack
    scroller.scrollTo(0,1);
}
```

# TouchTreeGrid

---

- Events:
  - i. ListsController onExample2ClistTap() listens for itemtap event.
  - ii. If user taps on DIV containing attribute “dataindex=done” (which is the checkbox div), then record updated:  
record.set('done', !done);
  - iii. The grid being bound to the store is auto-updated
  - iv. updateParents() method is called recursively to update “done” for all applicable parents and grid is also refreshed.

c. Notable Configs

- i. categDepthColors = true
- ii. categDepthColorsArr = ['white', 'white', 'white']
- iii. categDepthColorButtons = false
- iv. renderers{} – see above for Slide Navigation example
- v. styleContentRow (Slide Navigation example)  
**min-height: 32px !important;**  
display: -webkit-box; -webkit-box-orient: horizontal;  
background-color: {[values.done && values.leaf ? "#adbe87" : "none"]};
- vi. styleCategRow (Slide Navigation example)  
**min-height: 32px !important;**  
display: -webkit-box; -webkit-box-orient: horizontal;

d. CSS (defaults defined in TouchTreeGrid.css, customized by treegriddemo.css)

```
.x-touchtreegrid-list-example2C .x-list-normal .x-list-item .x-dock-horizontal,
.x-touchtreegrid-list-example2C .x-list-normal .x-list-item.x-list-item-tp1 {
    border: none;
}

.x-touchtreegrid-list-example2C .touchtreegrid-list-categ {
/* Default for Category rows (and depth=0 category when custom categDepthColors=true) */
padding:0 0 0 0 !important;
background-color: transparent !important;
line-height: 32px !important;
min-height: 32px !important;
font-size: .8em !important;
border: none;
-webkit-box-shadow: none;
}

.x-touchtreegrid-list-example2C .touchtreegrid-list-content {
/* Detail (leaf) rows */
padding:.6em 0 0 0;
padding:0 0 0 0 !important;
background-color: transparent;
line-height: 32px !important;
min-height: 32px !important;
font-size: .8em !important;
}

.x-touchtreegrid-list-example2C .touchtreegrid-details-img {
/* category row arrow centered with row text */
margin-top: .4em;
}
```



# TouchTreeGrid

## Census Grid

US Census - Maine									
	Population	Males	Females	Median	18+	21+	62+	65+	
▼ Maine	1,274,923	620,309	654,614	38	973,685	924,108	215,732	183,402	→
▶ Androscoggin County	103,793	50,385	53,408	37	78,957	74,488	17,432	14,962	→
▶ Aroostook County	73,938	36,095	37,843	40	57,218	54,367	14,772	12,551	→
▶ Cumberland County	265,612	128,589	137,023	37	203,650	193,206	41,129	35,324	→
▼ Franklin County	29,467	14,228	15,239	38	22,538	20,827	4,922	4,184	→
Avon town	504	244	260	38	371	355	77	64	→
Carrabassett Valley tov	399	210	189	39	319	302	46	35	→
Carthage town	520	279	241	36	380	367	72	63	→
Chesterville town	1,170	572	598	38	877	835	159	128	→
Conlin plantation	135	74	61	41	103	102	21	20	→
Expand Collapse 2									

- Demonstrates
  - ~600 total row grid example with 28 data columns per row (no performance issues)
  - Uses TouchTreeGrid default color schemes for category rows and collapse-to buttons
  - Loads data within controller via JSON tree store
  - Initially collapses to depth 2.
  - Disclose implemented for category and detail rows to view all 28 columns in custom formatted grid. Grid layout created using Architect layouts and onCensusMaineListDisclose() function within controller defines column textfield details.
  - Grid and detail window utilize formatNumbers() function included with TouchTreeGrid to format numbers with commas.
  - Different COLUMN configurations for Phone-Portrait, Phone-Landscape, Tablet-Portrait, Tablet-Landscape. Refer to loadColumnsCensusMaine() function in CensusController.js for example on how to specify different column configurations. Test using webkit-enabled browser by manually resizing window:
    - [http://localhost/TouchTreeGrid-master/TTG\\_Slider/app.html?deviceType=Phone](http://localhost/TouchTreeGrid-master/TTG_Slider/app.html?deviceType=Phone)
    - [http://localhost/TouchTreeGrid-master/TTG\\_Slider/app.html?deviceType=Tablet](http://localhost/TouchTreeGrid-master/TTG_Slider/app.html?deviceType=Tablet)
  - Also note custom implementation in Phone-Landscape mode where Titlebar and TabBars are hidden allowing more display (rotate to Portrait to restore) (TabPanel example only):

# TouchTreeGrid

	Population	Males	Females	Median	
▼ Maine	1,274,923	620,309	654,614	39	➔
▶ Androscoggin County	103,793	50,385	53,408	37	➔
▶ Aroostook County	73,938	36,095	37,843	41	➔
▶ Cumberland County	265,612	128,589	137,023	38	➔
▶ Franklin County	29,467	14,228	15,239	38	➔

Expand Collapse 2 Rotate for Menu

Only displays  
when  
Titlebar/Toolbar  
are hidden in  
Landscape

Note: if app is launched in phone landscape mode the menu will not be hidden until you rotate to portrait, then back to landscape.

## b. Program Flow

- Controller: CensusController.js
- TabPanel example View Config: Main.js => CensusContainer.js => itemId = 'censusmaine'
- Slide Navigation example: CensusMaineContainer.js
- Data:
  - i. TouchTreeGrid.store.CensusMaine2000 (storeId = censusmaine2000store)
  - ii. Model: TouchTreeGrid.model.Census
  - iii. TreeStore (./data/censusmaine2000TREE.json)
  - iv. Data loaded via CommonController within onMainTabpanelActiveItemChange() method

```
if (newcont === 'censusmainecontainer') {  
    // Check store for data and load if empty (only)  
    numNodes = grid.getStore().getData().length;  
    if (numNodes === 0) {censusController.loadCensusMaine2000Store();}  
}
```

For TabPanel example, or within onMenuOptListSelect() method within CommonController for Slide Navigation example.

censusController.loadCensusMaine2000Store=> commonController.loadStore  
=> commonController.postLoadProcess=>

```
if (gridListItemId === 'censusmainelist') {  
    // Collapse nodes to defined level  
    var depth = gridcont.getDefaultCollapseLevel();  
    if (depth !== 99) {gridcont.doExpandDepth(depth);}  
  
    censusController.loadColumnsCensusMaine(); // also refreshes list  
}
```

- v. censusController.loadColumnsCensusMaine() loads columns[] array for list based on phone vs. tablet and portrait vs. landscape

# TouchTreeGrid

---

- Events:

```
control: {
  "list#censusmainlist": {
    disclose: 'onCensusMainListDisclose'
  },
  "button#censusdetailbackbtn": {
    tap: 'onCensusDetailBackButtonTap'
  }
}
```

## c. Notable Configs

- categIndentPct = '2' (could create 2 different linked instances, tablets 2% and phones 4% are practical)
- onItemDisclosure = true
- categDepthColors = true
- defaultCollapseLevel = 2
- categDepthColorsArr: ['#904848', '#dddddd', 'white']
  - category row colors are defined in touchtreegrid-list-categX selectors via categCssArr config (below). This config also applies to collapse-to button colors.
  - **Note: this (along with categCssArr[] below) is suggested method for styling category rows going forward**
- categCssArr: ['touchtreegrid-list-categ0', 'touchtreegrid-list-categ1', 'touchtreegrid-list-categ'] =>
  - Level 1 category rows styled via 'touchtreegrid-list-categ0' selector (maroon rows with bold white text)
  - Level 2 category rows styled via 'touchtreegrid-list-categ1' selector (gray rows with blue bold text)
  - Level 3 category rows not applicable in this case but would be styled by default 'touchtreegrid-list-categ' selector

## d. CSS – overrides default CSS defined in TouchTreeGrid.css via '.x-touchtreegrid-list-censusmaine' selector. Refer to treegriddemo.css:

- .x-touchtreegrid-list-censusmaine .touchtreegrid-list-categ0 {...}
- .x-touchtreegrid-list-censusmaine .touchtreegrid-list-categ1 {...}
- .x-touchtreegrid-list-censusmaine .touchtreegrid-list-content{...}
- .x-touchtreegrid-list-censusmaine .touchtreegrid-header {...}

# TouchTreeGrid

## Census Filter Example #1

Census Filter Ex#1										?	Census Filter Example
	Population	Males	Females	Median	18+	21+	62+	65+			
▼ Maine	1,274,923	620,309	654,614	38	973,685	924,108	215,732	183,402	→		>Variation of Census Maine Example.
Androscoggin Coun	103,793	50,385	53,408	37	78,957	74,488	17,432	14,962	→		>Applies easy to implement TreeStore filtering using provided TreeStore generation algorithm where any custom filter function can be defined (refer to Discussion of this example in "TouchTreeGrid - Documentation.pdf"
Aroostook County	73,938	36,095	37,843	40	57,218	54,367	14,772	12,551	→		
▼ Cumberland County	265,612	128,589	137,023	37	203,650	193,206	41,129	35,324	→		>Test Filter Options: "Males > Females" "Females > Males" "Population > 10k"
Brunswick town	21,172	10,210	10,962	35	16,301	14,676	3,773	3,272	→		>Only children that pass filter are included.
Falmouth town	10,310	4,911	5,399	40	7,499	7,323	1,904	1,667	→		
Gorham town	14,141	6,846	7,295	34	10,479	9,237	1,661	1,413	→		
Scarborough town	16,970	8,296	8,674	38	12,571	12,203	2,606	2,211	→		>Nodes are always included if any of their children are included
Windham town	14,904	7,562	7,342	36	11,282	10,794	1,761	1,473	→		
Franklin County	29,467	14,228	15,239	38	22,538	20,827	4,922	4,184	→		>Option to display node as a leaf if the node is included, but
Expand	Collapse	Population > 10k ▼									

### a. Demonstrates

- Variation of Census Maine example but with TreeStore filtering
- I found a few threads out there and Sencha Support responded that TreeStores don't support filtering for Touch or EXTJS:
  - i. <http://www.sencha.com/forum/showthread.php?257019>
  - ii. <http://www.sencha.com/forum/showthread.php?253644>
- This example demonstrates one easy to implement solution that uses provided TreeStore generation algorithm where any custom filter function can be defined. Refer to discussion of "Lists => Grouper#2 (TreeGrid serving 'grouper' feature with expand/collapse for how to generate TreeStore on client from array that contains ID and PARENT\_ID columns to association leafs with categories. The array is stored to the linked instance of TouchTreeGrid component and re-used for filtering and sorting purposes. This TreeStore generation algorithm was modified to accept myFilter{} object which defines user-defined function for filtering TreeStores when regenerating the TreeStore from the stored array. The following conventions are applied when filtering TreeStores:
  - i. Only children that pass filter are included.
  - ii. Nodes are always included if any of their children are included (i.e. whether the node itself passes the filter condition or not)
  - iii. Option to display node as a leaf if the node is included, but none of it's children are included

# TouchTreeGrid

---

- myFilt{} object members:
  - i. enabled: true or false
  - ii. displayNodesWithAllMembersFilteredAsLeafs: true or false
  - iii. filterFn()
- Sample filter function to include Census Data where Population > 10,000 people:

```
myFilt.filterFn = function (rowObj)
{return (parseInt(rowObj.TotalPopulation) >= 10000)};
```
- Sample filter function to include where Males > Females:

```
var v1 = 'Male';
var v2 = 'Female';
var oper = '>=';

// var func = new Function("x", "y", "return x*y;");
myFilt.filterFn = new Function("rowObj", "return (parseInt(rowObj." + v1 + ') ' +
oper + ' parseInt(rowObj.' + v2 + '));");
```
- Sample filter function to include where Female > Males:

```
myFilt.filterFn = function (rowObj)
{return (parseInt(rowObj.Female) >= parseInt(rowObj.Male))};;
```
- Refer to onCensusFilterSelectfieldChange() and applyCensusFilter() methods within commonController.js for example on implementation

## b. Program Flow

- Controller: CensusController.js
- TabPanel example View Config: Main.js => CensusContainer.js => itemId = 'censusfilter'
- Slide Navigation example: CensusFilterContainer.js
- Data:
  - i. TouchTreeGrid.store.CensusFilterStore (storeId = censusFilterStore)
  - ii. Model: TouchTreeGrid.model.Census
  - iii. TreeStore (./data/censusmaine2000TREE.json)
  - iv. Data loaded via CommonController within onMainTabpanelActiveItemChange() method

```
if (newcont === 'censusfiltercont'){
    numRecords = grid.getStore().getData().length;
    if (numRecords === 0) {
        me.loadCensusFilterStore(gridcont, grid);
    }
}
```

For TabPanel example, or within onMenuOptListSelect() method within CommonController for Slide Navigation example.

# TouchTreeGrid

```
censusController.loadCensusFilterStore=> commonController.loadStore =>  
commonController.postLoadProcess =>
```

```
else if (gridListItemId === 'censusfilterlist') {  
    collapseLvl = (Ext.isEmpty(gridcont.collapseLevel) ? 1 : gridcont.collapseLevel);  
    // Refer to expCollapse() method where collapseLevel could be updated for manual expand processing  
  
    myFilt = {};  
    censusController.loadColumnsCensusMaine(gridcont, true); // also refreshes list  
    commonController.loadTree(collapseLvl, griddata.datalist, [], gridcont, null, true, myFilt); // collapse on initial load  
    Ext.Viewport.setMasked(false);  
}
```

- v. censusController.loadColumnsCensusMaine() loads columns[] array for list based on phone vs. tablet and portrait vs. landscape

- Events:

```
control: {  
    "list#censusmainelist": {  
        disclose: 'onCensusMaineListDisclose'  
    },  
    "button#censusdetailbackbtn": {  
        tap: 'onCensusDetailBackButtonTap'  
    }  
},  
  
//  
"selectfield#censusfilterselect": {  
    change: 'onCensusFilterSelectfieldChange'  
}
```

- c. Notable Configs

- i. Refer to CensusMaine above

- d. CSS – refer to CensusMaine above

## Census Freeze Column Example

Freeze Column						Freeze Column Example
Region	Total	Male	Female	<5yrs	5-9	>Horizontal scrolling with Freeze Column for large 600 data row with 28 data elements per row Tree Grid example.
▼ Maine	1,274,923	620,309	654,614	70,726	83,022	
▶ Androscoggin County	103,793	50,385	53,408	6,122	6,898	>Multiple freeze columns can be implemented simply by instantiating new instances of TouchTreeGrid for specific columns and referencing the same TreeStore
▶ Aroostook County	73,938	36,095	37,843	3,730	4,459	
▶ Cumberland County	265,612	128,589	137,023	15,443	17,507	
▶ Franklin County	29,467	14,228	15,239	1,514	1,856	
▼ Hancock County	51,791	25,324	26,467	2,516	3,069	>Recommended configs for all columns when specifying widths in pixels or em's: categorIndntPct='0', colNumberToTruncateForIndex
Amherst town	230	127	103	9	9	
Aurora town	121	67	54	11	10	>Recommended configs for 1st displayed fixed column container: width/minWidth a little larger than the actual column width for right padding, arrowPctWidth='6' (or large enough for reduced display width)
Bar Harbor town	4,820	2,249	2,571	218	242	
Blue Hill town	2,390	1,128	1,262	85	134	
Expand Collapse 2						

# TouchTreeGrid

---

- e. Demonstrates
  - i. Horizontal scrolling for Tree Grids with synchronized vertical scrolling and expand/collapse functionality.
  - ii. Each column set references the same TreeStore, but with different column configurations
  - iii. Multiple freeze and horizontal scrolling column sets can be implemented

- f. Program Flow

- i. Controller: CensusController.js
- ii. Slide Navigation example: CensusFreezeCont.js
- iii. Data:
  - TouchTreeGrid.store.CensusFreezeStore (storeId = censusFreezeStore)
  - Model: TouchTreeGrid.model.Census
  - TreeStore (./data/censusmaine2000TREE.json)
  - Data loaded via onMenuOptListSelect() method within CommonController.js for Slide Navigation example:

```
censusController.loadCensusFreezeStore =>  
commonController.loadStore =>  
commonController.postLoadProcess =>  
censusController.loadColumnsCensusFreeze()
```

- censusController.loadColumnsCensusFreeze() loads columns[] array defined in pixels. Note pixels are preferred over em's in this case so that the column headers line up with the grid columns. This is necessary when font sizes are different between the two. CensusFreezeCont.js has initialize() method that reduces the column width of the initial freeze column for phone portrait orientations to allow more scrolling width for the remaining columns.
- **IMPORTANT: When specifying column widths in pixels or em's for TreeGrids you MUST set following configs to avoid TPL compile error within doRefreshList() method:**

```
categIndentPct = '0'  
colNumberToTruncateForIndents = 0
```

- g. Notable Configs

- CensusFreezeCont.js is outermost container for all scrolling grids. Unlimited combinations of freeze columns and horizontal scrolling column sets supported.
  - **itemId = 'censusFreezeCont'**
  - **Layout = 'hbox'**
- **Freeze column**
  - xtype = 'touchtreegrid'
  - itemId = 'censusfreezeX'
  - listItemId = 'censusfreezelistX'

# TouchTreeGrid

---

- **linkedGridsParentItemId** = 'censusFreezeCont'  
(ultimate parent container itemId for all scrolling grids)
- **linkedGridsArr** = [{itemId: 'censusfreeze' }]  
(itemId of other scrolling grid)
- Columns[] array defines single dataIndex = 'CATEG' column (width = '200px').
- categIndentPct = '0' (**REQUIRED FOR PIXEL/EM WIDTHS !!**)
- colNumberToTruncateForIndents = 0 (**REQUIRED FOR PIXEL/EM WIDTHS !!**)
- arrowPctWidth = '6' (need to choose percent that allows arrow to remain visible for narrower panel width)
- Footer related configs
- includeFooterLevels = true  
(we want to include expand/collapse and footer levels in freeze column only)
- includeFooter = true (default and not required)
- minWidth/width = '220px' (includes 20px right padding)
- style = 'border-right: 2px solid gray;' (line border between panels)
- **Phone-specific configuration**
- CensusFreezeCont.js initialize() method contains logic to reduce width of freeze column for Phone Portrait configurations to allow more scrolling width for scrolling columns
- **Horizontal Scrolling columns**
  - Horizontal scrolling columns require a parent container with following configs:
  - Flex = 1 (optional, but at least one set of scrolling grids should have flex=1)
  - Layout = 'hbox'
  - Scrollable = { direction: 'horizontal', directionLock: true }
  - Child container to this parent container is linked instance of TouchTreeGrid
  - xtype = 'touchtreegrid'
  - itemId = 'censusfreeze'
  - listItemId = 'censusfreezelist'
  - **flex = 1** (required here)
  - **linkedGridsParentItemId** = 'censusFreezeCont'  
(ultimate parent container itemId for all scrolling grids)
  - **linkedGridsArr** = [{itemId: 'censusfreezeX' }]  
(itemId of other scrolling grid)
  - Columns[] array defines all scrolling columns (except frozen CATEG column and is loaded within loadColumnsCensusFreeze() method of



# TouchTreeGrid

CensusController.js. All widths defined in pixels (vs. em's) so that detail columns line up with header columns (which have different font-size).

- cateIndentPct = '0' (REQUIRED FOR PIXEL/EM WIDTHS !!)
- colNumberToTruncateForIndents = 0 (REQUIRED FOR PIXEL/EM WIDTHS !!)
- arrowPctWidth = '6' (need to choose percent that allows arrow to remain visible for narrower panel width)
- Footer related configs
  - includeFooter = true (default)
  - hideExpandCollapseBtns = true  
(we want footer bar for consistency across both scrolling grids, but we don't want expand/collapse and collapse-to button repeated on second panel)
  - includeFooterLevels = false
- minWidth/width = '220px' (includes 20px right padding)
- style = 'border-right: 2px solid gray;' (line border between panels)

h. CSS - refer to CensusMaine above

## Lists => Basic #1 (DOW History - 2012 Closing Prices data)

Basic List #1			
CloseDate	Close	Chg	Chg%
10/5/2012	13,610	35	0.26%
9/20/2012	13,597	19	0.14%
9/14/2012	13,593	54	0.40%
10/8/2012	13,584	27	-0.19%
9/21/2012	13,579	-17	-0.13%
9/19/2012	13,578	13	0.10%
10/4/2012	13,575	81	0.60%
9/18/2012	13,565	12	0.09%
9/24/2012	13,559	-21	-0.15%
10/17/2012	13,557	5	0.04%
9/17/2012	13,553	-40	-0.30%

- a. Demonstrates
- Standard Grid example using non-TreeStore.
  - Row selection with required CSS to support
  - Expand/Collapse footer toolbar omitted as not applicable
  - Column sorting with custom column shading
  - Row selection

# TouchTreeGrid

---

- vi. Zebra striping supported by following CSS defined in treegriddemo.css:

```
106
107 /***** DOW2012 GRID OVERRIDES *****/
108 .x-touchtreegrid-list-dow2012 .touchtreegrid-simplelist-cell { /* simple list cells */
109     background-color: transparent; /* needed for alternate shading */
110 }
111
112 .x-touchtreegrid-list-dow2012 .x-touchtreegrid-item:nth-child(even) { /* for alternate shading */
113     background-color: #f5f5f5 !important;
114 }
115
```

- vii. Custom renderer functions easily supported:

- i. Format date as mm/dd/yyyy
- ii. Comma formatted closing price
- iii. Change and Chg% rendered as red for negative and green for positive

b. Program Flow

- i. Controller: ListsController.js
- ii. TabPanel example View Config: Main.js => ListsContainer.js => itemId = 'dow2012'
- iii. Slide Navigation example: Dow2012Cont.js
- iv. Data:
  - i. TouchTreeGrid.store.Dow2012 (storeId = dow2012store)
  - ii. Model: TouchTreeGrid.model.Dow2012
  - iii. TreeStore (./data/dow2012.json)

1. commonController. onMainTabpanelActiveItemChange()

```
if (newcont === 'listscontainer'){
    numRecords = grid.getStore().getData().length;
    if (numRecords === 0) {
        Ext.Viewport.setMasked({
            xtype: 'loadmask',
            message: 'Loading Basic...'
        });
        grid.getStore().load();
        gridcont.doRefreshList();
        Ext.Viewport.setMasked(false);
    }
}
```

For TabPanel example, or within onMenuOptListSelect() method within CommonController for Slide Navigation example.

- 2.

- v. Events: (none defined)

c. Notable Configs

- i. simpleList = true
- ii. columnSorting = true
- iii. disableSelection = false
- iv. header : {..., minHeight: '2.6em', ...} (increase height to improve touch for sorting)
- v. listPlugin : {xclass: 'Ext.plugin.PullRefresh'}
- vi. refer to columns[] array for:
  - i. style vs. styleSorted definitions by column
  - ii. sorted=true

# TouchTreeGrid

## d. CSS (defaults defined in TouchTreeGrid.css, customized by treegriddemo.css)

```
.x-touchtreegrid-list-dow2012 .touchtreegrid-simplelist-cell{ /* simple list cells */
    background-color: transparent; /* needed for alternate shading */
    padding-top: .5em !important;
    line-height: 2.6em;
}

.x-touchtreegrid-list-dow2012 .x-touchtreegrid-item:nth-child(even) { /* for alternate shading */
    background-color: #f5f5f5 !important;
}

/** Begin DOW2012 CSS required to support row selection for simplelist rows **/
.x-touchtreegrid-list-dow2012 .touchtreegrid-item-selected,
.x-touchtreegrid-list-dow2012 .touchtreegrid-item-selected.x-touchtreegrid-item:nth-child(even) {
/* Notice no space b/t last 2 selectors .. not sure why this is necessary but it works for shaded rows*/
    background-color: #006bb6 !important; /* color for selected row */
}

.x-touchtreegrid-list-dow2012 .touchtreegrid-item-selected .touchtreegrid-simplelist-cell{
    background-color: #006bb6 !important; /* cells need to match row color */
    color: white !important; /* Change selected text color to white */
}

.x-touchtreegrid-list-dow2012 .touchtreegrid-item-pressed,
.x-touchtreegrid-list-dow2012 .touchtreegrid-item-pressed.x-touchtreegrid-item:nth-child(even),
.x-touchtreegrid-list-dow2012 .touchtreegrid-item-pressed .touchtreegrid-simplelist-cell {
    background-color: #b6e1ff !important; /* color for selected row */
}

/* Below would control selected color of Disclosure Icon if onDisclosure=true */
.x-touchtreegrid-list-dow2012 .touchtreegrid-item-selected .x-list-disclosure{
    background-color: white;
}

.x-touchtreegrid-list-dow2012 .touchtreegrid-item-selected .x-list-disclosure:before{
    color: #006bb6;
}

/** End DOW2012 CSS required to support row selection for simplelist rows **/
```

## Lists => Basic #2

Basic List #2					
CloseDate	Close	Chg	Chg%	Flag	
12/28/2012	12,938	-158	-1.21%		
12/27/2012	13,096	-18	-0.14%		
12/26/2012	13,115	-24	-0.19%		
12/24/2012	13,139	-52	-0.39%		
12/21/2012	13,191	-121	-0.91%		
12/20/2012	13,312	60	0.45%		
12/19/2012	13,252	-99	-0.74%		
12/18/2012	13,351	116	0.87%		
12/17/2012	13,235	100	0.76%		
12/14/2012	13,135	-36	-0.27%		
12/13/2012	13,171	-75	-0.56%		
12/12/2012	13,245	-3	-0.02%		
12/11/2012	13,248	79	0.60%		

# TouchTreeGrid

---

## e. Demonstrates

- Variation of Basic List #1 Example.
- 32 pixel row height
- Checkcolumn selection for Standard Grid (refer to Project #3 example for checkcolumn for Tree Grids)
- Conditional row highlighting (Close Price >=13200)... refer to cssSimpleRow config below.
- Row selection
- Pull Refresh
- Example of custom "css" (vs. "style") renderers for positive as green and negative as red, but allows selected row to display as white

## f. Program Flow

- i. Controller: ListsController.js
- ii. TabPanel example View Config: Main.js => ListsContainer.js => itemId = 'dow2012'
- iii. Slide Navigation example: Dow2012Cont.js
- iv. Data:
  - i. TouchTreeGrid.store.Dow2012 (storeId = dow2012store)
  - ii. Model: TouchTreeGrid.model.Dow2012
  - iii. TreeStore (./data/dow2012.json)

### 1. commonController. onMainTabpanelActiveItemChange()

```
if (newcont === 'listscontainer'){
    numRecords = grid.getStore().getData().length;
    if (numRecords === 0) {
        Ext.Viewport.setMasked({
            xtype: 'loadmask',
            message: 'Loading Basic...'
        });
        grid.getStore().load();
        gridcont.doRefreshList();
        Ext.Viewport.setMasked(false);
    }
}
```

For TabPanel example, or within onMenuOptListSelect() method within CommonController for Slide Navigation example.

### 2.

## v. Events:

- i. ListController onDow2012Basic2ListItemTap() method listens for itemtap on #dow2012Basic2list
- ii. If user taps on Flag cell (where attribute "dataindex=Flag") .. the cell defining the checkbox, then: record.set('Flag', !Flag);
- iii. The grid is bound to the store and is therefore updated.

## g. Notable Configs

- Refer to Basic #1
  - cssSimpleRow: '[[{values.Close >= 13200 ? "highlight-rows-over-13200" : ""}],[" : ""]]'
- (conditional row highlighting)

# TouchTreeGrid

---

- CSS selector “highlight-rows-over-13200”

```
.x-touchtreegrid-list-dow2012Basic2 .highlight-rows-over-13200{
    background-color: #FFFFB2;
    font-style: italic;
    width: 100%;
}
```

- Column config for checkcolumn

```
{
    header: 'Flag',
    headerStyle: 'text-align: left; color: #ccc;',
    dataIndex: 'Flag',
    width: '35px',
    sortable: true,
    addDataIndexToDiv: true,
    // required to listen to taps on this column
    renderer: 'this.renderer_flag(values)',
    style: 'text-align: center;'
}
```

- Render Object for checkcolumn:

```
renderer_flag: function (values) {
    if (values.Flag) {
        return ( "<span class='flag-checked'> </span>" );
    }
    else {
        return ( "<span class='flag-not-checked'> </span>" );
    }
}
```

- CSS for checkcolumn:

(refer to <http://pictos.cc/font/> for fonts bundled with Touch)

```
.x-touchtreegrid-list-dow2012Basic2 .flag-checked:before {
    font-family: "Pictos";
    content: "3";
    color: gray;
    font-size: 18px;
    border: 1px solid gray;
}
.x-touchtreegrid-list-dow2012Basic2 .flag-not-checked:before {
    font-family: "Pictos";
    content: "3";
    color: transparent;
    font-size: 18px;
    border: 1px solid gray;
}
```

- Renderer Object which is CSS based (as opposed to style-based) for pos/neg styling:

```
{renderer_formatWithColorCls: function (value, decPlaces, prefix, suffix,
thouSeparator, decSeparator)
{var cls = (value >= 0) ? 'cellrend-positive' : 'cellrend-negative';
return '<span class="' + cls + '">' + this.formatNumbers(value, decPlaces, prefix,
suffix, thouSeparator, decSeparator) +
'</span>';}}
```

## h. CSS (defaults defined in TouchTreeGrid.css, customized by treegriddemo.css)

```
/****** DOW2012Basic2 GRID OVERRIDES *****/
.x-touchtreegrid-list-dow2012Basic2 .touchtreegrid-simplelist-cell{ /* simple list cells */
    background-color: transparent; /* needed for alternate shading and row rendering via
cssStyleRow config*/
    padding-top:0 !important;
    line-height: 32px;
```

# TouchTreeGrid

---

```
}

.x-touchtreegrid-list-dow2012Basic2 .cellrend-positive {
    color: green;
}
.x-touchtreegrid-list-dow2012Basic2 .cellrend-negative {
    color: red;
}
.x-touchtreegrid-list-dow2012Basic2 .touchtreegrid-item-selected .cellrend-positive {
    color: white !important;
}
.x-touchtreegrid-list-dow2012Basic2 .touchtreegrid-item-selected .cellrend-negative {
    color: white !important;
}

.x-touchtreegrid-list-dow2012Basic2 .flag-checked:before {
    font-family: "Pictos";
    content: "3";
    color: gray;
    font-size: 18px;
    border: 1px solid gray;
}
.x-touchtreegrid-list-dow2012Basic2 .flag-not-checked:before {
    font-family: "Pictos";
    content: "3";
    color: transparent;
    font-size: 18px;
    border: 1px solid gray;
}
.x-touchtreegrid-list-dow2012Basic2 .touchtreegrid-item-selected .flag-checked:before {
    color: white;
    border: 1px solid white;
}
.x-touchtreegrid-list-dow2012Basic2 .touchtreegrid-item-selected .flag-not-checked:before {
    border: 1px solid white;
}

.x-touchtreegrid-list-dow2012Basic2 .x-touchtreegrid-item:nth-child(even) { /* for alternate shading */
    /* background-color: #f5f5f5 !important; */
}
.x-touchtreegrid-list-dow2012Basic2 .highlight-rows-over-13200{
    background-color: #FFFFB2;
    font-style: italic;
    width: 100%;
}

.x-touchtreegrid-list-dow2012Basic2 .touchtreegrid-item-selected .touchtreegrid-simplelist-cell{
    /* background-color: #006bb6 !important; /* cells need to match row color */
    color: white !important; /* Change selected text color to white */
}

/** Begin DOW2012 CSS required to support row selection for simplelist rows */
.x-touchtreegrid-list-dow2012Basic2 .touchtreegrid-item-selected,
.x-touchtreegrid-list-dow2012Basic2 .touchtreegrid-item-selected.x-touchtreegrid-item:nth-child(even),
.x-touchtreegrid-list-dow2012Basic2 .touchtreegrid-item-selected.x-touchtreegrid-item:nth-child(odd){
/* Notice no space b/t last 2 selectors .. not sure why this is necessary but it works for shaded rows*/
    background-color: #006bb6 !important; /* color for selected row */
}

.x-touchtreegrid-list-dow2012Basic2 .touchtreegrid-item-pressed,
.x-touchtreegrid-list-dow2012Basic2 .touchtreegrid-item-pressed .highlight-rows-over-13200,
```

# TouchTreeGrid

---

```
.x-touchtreegrid-list-dow2012Basic2 .touchtreegrid-item-pressed.x-touchtreegrid-item:nth-
child(even),
.x-touchtreegrid-list-dow2012Basic2 .touchtreegrid-item-pressed.x-touchtreegrid-item:nth-
child(odd),
.x-touchtreegrid-list-dow2012Basic2 .touchtreegrid-item-pressed .touchtreegrid-simplelist-cell {
    background-color: #b6e1ff !important; /* color for selected row */
}

.x-touchtreegrid-list-dow2012Basic2 .touchtreegrid-item-selected,
.x-touchtreegrid-list-dow2012Basic2 .touchtreegrid-item-selected .highlight-rows-over-13200 {
/* Notice no space b/t last 2 selectors .. not sure why this is necessary but it works for shaded
rows*/
    background-color: #006bb6 !important; /* color for selected row */
}

/* Below would control selected color of Disclosure Icon if onDisclosure=true */
.x-touchtreegrid-list-dow2012Basic2 .touchtreegrid-item-selected .x-list-disclosure{
    background-color: white;
}
.x-touchtreegrid-list-dow2012Basic2 .touchtreegrid-item-selected .x-list-disclosure:before{
    color: #006bb6;
}
/** End DOW2012 CSS required to support row selection for simplelist rows **/
```

# TouchTreeGrid

## Lists => Grouper (DOW History Example with Grouper feature)

Grouper #1			
CloseDate	Close	Chg	Chg%
December (Max: 13350.96 / Min: 12938.11 )			
12/18/2012	13,351	116	0.87%
12/20/2012	13,312	60	0.45%
12/19/2012	13,252	-99	-0.74%
12/11/2012	13,248	79	0.60%
12/12/2012	13,245	-3	-0.02%
12/17/2012	13,235	100	0.76%
12/21/2012	13,191	-121	-0.91%
12/13/2012	13,171	-75	-0.56%
12/10/2012	13,170	15	0.11%
12/7/2012	13,155	81	0.62%
12/5/2012	13,138	-59	-0.45%

- a. Demonstrates
  - i. Same as “Basic” DOW example but with Grouper feature.
  - ii. Grouper defined by custom field provides min/max Closing prices for each month
  - iii. Columns sort within each group when column header is tapped (as opposed to sorting across all data)
  - iv. “grouped: true” must be defined within additionalListConfigs config:

```
simpleList: true,  
columnSorting: true,  
additionalListConfigs: {  
  grouped: true  
},  
cls: [  
  'x-touchtreegrid-list',  
  'x-touchtreegrid-list-dow2012-grouper'  
],  
itemId: 'dow2012grouper'
```

- v. Store must define grouper “property” and “sortProperty” and initial sorters “direction” and “property” (if columnSorting: true):



# TouchTreeGrid

---

```
1 Ext.define('TouchTreeGrid.store.Dow2012grouper', {
2     extend: 'Ext.data.Store',
3
4     requires: [
5         'TouchTreeGrid.model.Dow2012'
6     ],
7
8     config: {
9         model: 'TouchTreeGrid.model.Dow2012',
10        storeId: 'Dow2012grouper',
11        proxy: {
12            type: 'ajax',
13            url: './data/dow2012grouper.json',
14            reader: {
15                type: 'json',
16                rootProperty: 'datalist'
17            }
18        },
19        grouper: {
20            direction: 'DESC',
21            property: 'grouper',
22            sortProperty: 'YearMonth'
23        },
24        sorters: {
25            direction: 'DESC',
26            property: 'CloseDate'
27        }
28    }
29 });
```

## b. Program Flow

- a. Controller: ListsController.js
- b. TabPanel example View Config: Main.js => ListsContainer.js => itemId = 'dow2012grouper'
- c. Slide Navigation example: Dow2012grouperCont.js
- d. Data:
  - i. TouchTreeGrid.store.Dow2012grouper (storeId = Dow2012grouper)
  - ii. Model: TouchTreeGrid.model.Dow2012
  - iii. TreeStore (./data/dow2012grouper.json)

### 1. listsController. onListsTabpanelActiveItemChange()

```
if (newcont === 'dow2012grouperCont'){
    numRecords = grid.getStore().getData().length;
    if (numRecords === 0) {
        Ext.Viewport.setMasked({
            xtype: 'loadmask',
            message: 'Loading Grouper ...'
        });
        grid.getStore().load();
        gridcont.doRefreshList();
        Ext.Viewport.setMasked(false);
    }
}
```

For TabPanel example, or within onMenuOptListSelect() method within CommonController for Slide Navigation example.

### 2.

## e. Events: (TabPanel example)

```
control: {
    "tabpanel#listtabpanel": {
        activeitemchange: 'onListsTabpanelActiveItemChange'
    },
}
```

## c. Notable Configs

- a. additionalListConfigs : {grouped : true}
- b. simpleList = true

# TouchTreeGrid

---

- c. `columnSorting = true`
  - d. `disableSelection = true`
- d. CSS – uses default CSS defined in `TouchTreeGrid.css`

# TouchTreeGrid

## Lists => Grouper#2 (TreeGrid serving 'grouper' feature with expand/collapse)



CloseDate	Close	Chg	Chg%
► December (Max: 13350.96 / Min: 12938.11 )			
► November (Max: 13245.68 / Min: 12542.38 )			
► October (Max: 13610.15 / Min: 13077.34 )			
► September (Max: 13596.93 / Min: 13035.94 )			
▼ August (Max: 13275.2 / Min: 12878.88 )			
8/31/2012	13,091	90	0.69%
8/30/2012	13,001	-107	-0.81%
8/29/2012	13,107	4	0.03%
8/28/2012	13,103	-22	-0.17%
8/27/2012	13,125	-33	-0.25%

### a. Demonstrates

- Uses TreeGrid for 'grouper' type feature but supports expand/collapse
- Leverages optional `catColumns[]` array to define column widths and styling for category row independent of leaf rows (which are defined in `columns[]` array)
- Generates TreeStore from flat file containing ID and PARENT\_ID columns
- Demonstrates use of custom `customExpCollapseEvent` config to improve performance over Sencha default Node expand/collapse methods. Essentially we are rebuilding the TreeStore using the provided TreeStore JavaScript generation methods and reloading the store based on the desired collapse-to level.
- Note: I found an issue with Touch 2.2 where you can't use the same model for Store + TreeStore examples as the Node Interface object becomes corrupt. The solution is simply to create separate Models even if the data columns are the same.

### b. Program Flow

- Controller: `ListsController.js`
- TabPanel example View Config: `Main.js => ListsContainer.js => itemId = 'dow2012grouper2'`
- Slide Navigation example: `Dow2012grouper2Cont.js`
- Data:
  - `TouchTreeGrid.store.dow2012TreeStore` (`storeId = dow2012treestore`)
  - Model: `TouchTreeGrid.model.Dow2012Tree`
  - TreeStore (`./data/dow2012categ.json`)

# TouchTreeGrid

1. Here we are demonstrating provided tree generation methods which create TreeStore from flat file containing **ID** and **PARENT\_ID** columns to link children to parents:

```
{ "success": true, "datalist": [
  { "grouper": "December (Max: 12294 \\/ Min: 11766.26 )",
    "Open": null, ..., "ID": "201112", "PARENT_ID": null }
  , { "CloseDate": "1\\10\\2012", "Open": "12394.51", ..., "ID": "44", "PARENT_ID": "201201" }
  , { "CloseDate": "1\\11\\2012", "Open": "12459.52", ..., "ID": "43", "PARENT_ID": "201201" }
  etc...
```

2. listsController.onListsTabpanelActiveItemChange()

```
if (newcont === 'dow2012grouper2Cont') {
  numRecords = grid.getStore().getData().length;
  if (numRecords === 0) {
    Ext.Viewport.setMasked({
      xtype: 'loadmask',
      message: 'Loading Grouper#2 ...'
    });
    this.loadDow2012Grouper2Store(gridcont, grid);
  }
}
```

For TabPanel example, or within onMenuOptListSelect() method within CommonController for Slide Navigation example.

3. listsController.loadDow2012Grouper2Store() =>  
commonController.loadStore() => commonController.postLoadProcess

```
else if (gridListItemId === 'dow2012grouper2list') {
  // For each field define level "up to which" values will be included on category rows.
  // '0' means highest root row will include non-nullable values (or as defined in flat file) for this field.
  // '1' means level 1 categories will include data (if defined)
  // '2' means data will only be included for up to level 2. Level 1 and root will not show data for this column.
  // etc...
  fldListArr = [['CloseDate', 1], ['Open', 1], ['High', 1], ['Low', 1], ['Close', 1],
    ['Volume', 1], ['AdjClose', 1], ['Chg', 1], ['ChgPct', 1], ['YearMonth', 0],
    ['grouper', 0]];

  collapseLvl = (Ext.isEmpty(gridcont.collapseLevel) ? 1 : gridcont.collapseLevel);
  // Refer to expCollapse() method where collapseLevel could be updated for manual expand processing

  commonController.loadTree(collapseLvl, griddata.datalist, fldListArr, gridcont, null); // collapse on initial load

  // Sort by YearMonth, then CloseDate
  gridlist.getStore().sort([{'property': 'YearMonth', direction: 'DESC'},
    {'property': 'CloseDate', direction: 'DESC'}]);

  Ext.Viewport.setMasked(false);
}
```

4. commonController.loadTree() => commonController.getTree() =>  
commonController.createTreeStructure() {recursively generates child nodes}

- v. Events: // TabPanel example

```
control: {
  "tabpanel#liststabpanel": {
    activeitemchange: 'onListsTabpanelActiveItemChange'
  },
  "container#dow2012grouper2": {
    expCollapse: 'onDow2012grouper2ExpCollapse'
  },
  ...
}
```

- c. Notable Configs

- i. simpleList = false
- ii. categIndentPct = '0'
- iii. customExpCollapseEvent = 'expCollapse'

# TouchTreeGrid

---

- iv. `applyDefaultCollapseLevel = false`
  - v. `categColumns: [{  
    dataIndex: 'grouper',  
    width: '90%',  
    categStyle: 'text-align: left; color: #006083; font-weight: bolder; font-size: 1.2em;'  
}]`
- d. CSS (defaults defined in TouchTreeGrid.css)

# TouchTreeGrid

## Lists => Horiz (DOW History Example with Horizontal Scrolling)

Horizontal Scrolling						
se te	Open Price	High Price	Low Price	Close Price	Volume	Ch
2012	13,237	13,366	13,233	13,351	1,529,200	1
2012	13,351	13,358	13,252	13,252	1,490,200	-1
2012	13,250	13,329	13,227	13,245	1,275,100	
2012	13,247	13,315	13,216	13,312	1,198,000	
2012	13,310	13,310	13,123	13,191	4,132,700	-1
2012	13,170	13,307	13,170	13,248	1,245,100	
2012	13,113	13,291	13,113	13,246	1,057,100	1
2012	13,233	13,289	13,077	13,093	1,376,600	-1
2012	13,099	13,274	13,099	13,233	1,405,100	1
2012	13,241	13,264	13,147	13,171	1,011,900	-1

- Demonstrates
  - Horizontal scrolling simpleList grid
  - Custom header height, style and wrapping content
  - Column sorting with customized shading
  - Traps cell tap on CloseDate or ClosePrice for controller processing (any cell can be handled this way)
  - Vertical lines defined via styling
  - Column widths defined in pixels or em's instead of percent
  - Total grid width specified in pixels or em'
  - Parent Container must have layout = 'hbox' and horizontal scrolling:

```
xtype: 'container',
title: 'Horiz',
itemId: 'dow2012HorizCont',
layout: {
    type: 'hbox'
},
scrollable: {
    direction: 'horizontal',
    directionLock: true
},
items: [
    {
        xtype: 'touchtreegrid',
        store: 'Dow2012Horiz',
        columns: [
```

# TouchTreeGrid

---

- ix. Vertical lines defined via border-right in Columns array. Example:

```
items: [
  {
    xtype: 'touchtreegrid',
    store: 'Dow2012Horiz',
    columns: [
      {
        header: 'CloseDate',
        dataIndex: 'CloseDate',
        width: '6em',
        style: 'text-align: left; font-weight: bold; color: #008abc; border-right: 1px solid #9b9b9b',
        headerStyle: 'text-align: left; color: #ccc;',
        renderer: 'Ext.Date.format(values.CloseDate, "n/j/Y")',
        sortable: true
      },
      {
        header: 'Open',
        dataIndex: 'Open',
        width: '4.375em',
        style: 'text-align: right; font-weight: normal;',
        headerStyle: 'text-align: right; padding-right: .5em !important; color: #ccc;',
        renderer: 'this.formatNumbers(values.Open, 0)',
        sortable: true
      }
    ]
  }
]
```

- x. Also notice “width” defined in em’s in above example.
- xi. Width and minWidth must be defined in linked instance for Horiz scrolling:

```
helpHtml: './resources/html/DOW2012HorizExample.html',
simpleList: true,
columnSorting: true,
cls: [
  'x-touchtreegrid-list',
  'x-touchtreegrid-list-dow2012Horiz'
],
itemId: 'dow2012horiz',
minWidth: '36em',
width: '36em'
```

## b. Program Flow

- a. Controller: ListsController.js
- b. TabPanel example View Config: Main.js => ListsContainer.js => itemId = 'dow2012horiz'
- c. Slide Navigation example: Dow2012HorizCont.js
- d. Data:
  - TouchTreeGrid.store. Dow2012Horiz (storeId = Dow2012Horiz)
  - Model: TouchTreeGrid.model. Dow2012
  - TreeStore (./data/dow2012horiz.json)

- listsController. onListsTabpanelActiveItemChange()

```
if (newcont === 'dow2012HorizCont') {
  numRecords = grid.getStore().getData().length;
  if (numRecords === 0) {
    Ext.Viewport.setMasked({
      xtype: 'loadmask',
      message: 'Loading Horiz ...'
    });
    grid.getStore().load();
    gridcont.doRefreshList();
    Ext.Viewport.setMasked(false);
  }
}
```

For TabPanel example, or within onMenuOptListSelect() method within CommonController for Slide Navigation example.

▪

# TouchTreeGrid

- e. Events: **// TabPanel example**

```
control: {
    "tabpanel#liststabpanel": {
        activeitemchange: 'onListsTabpanelActiveItemChange'
    },
    "container#dow2012horiz": {
        leafItemTap: 'onHorizGridLeafItemTap'
    },
}
```

- f. Code to trap click event on specific cell (CloseDate and ClosePrice in this case)

```
onHorizGridLeafItemTap: function(me, list, index, target, record, e) {
71 // Example of how we can code to only act if CloseDate or Close (Price) elements are tapped
72 // (requires that columns[] array have "addDataIndexToDiv: true" attribute defined for each of these
73 // columns)
74
75 var myField = e.target.getAttribute('dataIndex');
76 var tapped = true;
77
78 if (myField === 'CloseDate') {
79     var myDate = Ext.Date.format(record.get('CloseDate'), "n/j/Y");
80     Ext.Msg.alert('Close Date: ' + myDate);
81 }
82 else if (myField === 'Close') {
83     Ext.Msg.alert('Close Price: ' + record.get('Close'));
84 }
85 else {
86     // For this example no other fields were defined with dataIndex attribute within DIV
87     tapped = false;
88 }
89
90 if (tapped) {
91     console.log('onHorizGridLeafItemTap record tapped:');
92     console.log(record);
93 }
94 }
```

- g. Notes on how to simulate hyperlink substring within the text of a specific cell.  
You could use a rendering function to combine more than one data fields for display in a given column. Example:

- Store contains fields: LastName, FirstName
- For given column:
  - dataIndex = "LastName"
  - addDataIndexToDiv: true
  - renderer: 'this.renderer\_fullName(values)'
- Config for linked instance of TouchTreeGrid:
  - renderer\_fullName: function (values){  
return '<span style="color: blue;font-weight: bold;">' + LastName +  
'</span> ' + ' , ' + FirstName;
- Cell displays: **Doe, John**
- User taps on "**Doe**" hyperlink
- Sample controller action logic:
  - var myField = e.target.getAttribute('dataIndex');
  - if (myField === 'LastName' && e.target.innerText === record.get("LastName")) {... then we have tapped LastName substring so do something ... }

- c. Notable Configs

- a. simpleList = true
- b. columnSorting = true
- c. minWidth = '36em', width = '36em'
- d. header : { ..., minHeight: '3.2em', ...} to accommodate expanded header



# TouchTreeGrid

e. refer to columns[] array for style, header, styleSorted, headerStyleSorted

d. CSS (defaults defined in TouchTreeGrid.css, customized by treegriddemo.css)

```
.x-touchtreegrid-list-dow2012Horiz .touchtreegrid-simplelist-cell{ /* simple list cells */
    height: 47px !important;
}

.x-touchtreegrid-list-dow2012Horiz .x-grid-sort-desc,
.x-touchtreegrid-list-dow2012Horiz .x-grid-sort-asc {
    margin-top: .2em !important;
    background-position: left top;
}

.x-touchtreegrid-list-dow2012Horiz .touchtreegrid-header { /* Titlebar with column headers */
    /* width:100%; */
    padding:0 0 0 0;
    margin:.2em 0 0 0;
    color: black;
    font-size: 12px;
    background-color: #e5e5e5;
    background: none;
    font-weight: bold;
}
```

## Lists => Dynamic (Same as Horiz but defined dynamically from Server)



Close Date	Open Price	High Price	Low Price	Close Price	Volume
12/28/2012	13,095	13,095	12,927	12,938	859,81
12/27/2012	13,115	13,142	12,964	13,096	1,001,61
12/26/2012	13,139	13,175	13,077	13,115	794,11
12/24/2012					477,11
12/21/2012					4,132,71
12/20/2012	13,247	13,315	13,216	13,312	1,198,01
12/19/2012	13,351	13,358	13,252	13,252	1,490,21
12/18/2012	13,237	13,366	13,233	13,351	1,529,21
12/17/2012	13,135	13,244	13,135	13,235	1,429,81
12/14/2012	13,171	13,190	13,118	13,135	1,176,21

a. Demonstrates

- Same as Horiz example accept columns and fields are retrieved from single AJAX call and Store is created dynamically and loaded with data (from same call)
- Columns[] array utilizes CSS based definitions instead of Styles. Example:

```
{
    "header": "Close<br>Date",
    "dataIndex": "CloseDate",
    "width": "6em",
```

# TouchTreeGrid

---

```
        "css": "dynamic-detail-closedate",
        "cssSorted": "dynamic-detail-closedate-sorted",
        "headerCss": "dynamic-header-center",
        "headerCssSorted": "dynamic-header-center-sorted",
        "renderer": "Ext.Date.format(values.CloseDate, 'n/j/Y')",
        "sortable": "true"
    },
```

## b. Program Flow

- Controller: ListsController.js
- TabPanel example View Config: Main.js => ListsContainer.js => itemId = 'dow2012Dynamic'
- Slide Navigation example: Dow2012DynamicCont.js
- Data:
  - i. Store: no store created or referenced within config as created dynamically.
  - ii. Model: no model used as fields will be dynamically defined when creating store
  - iii. Dynamically loaded TreeStore (./data/dow2012dynamic.json)

Contains 3 arrays of data:

1. dataList[]
2. columnsPhonePortrait[]
3. fields[]

### iv. listsController.onListsTabpanelActiveItemChange()

```
if (newcont === 'dow2012DynamicCont'){
    // Reload each time pressed for dynamic grid example
    this.loadDow2012DynamicStore(gridcont, grid);
}
```

For TabPanel example, or within onMenuOptListSelect() method within CommonController for Slide Navigation example.

- v. listsController.loadDow2012DynamicStore() => commonController.loadDynamicStore()

# TouchTreeGrid

---

```
success: function(response) {
    var alldata = Ext.JSON.decode(response.responseText);
    var griddata = alldata.datalist;
    var columnsPhonePortrait = alldata.columnsPhonePortrait;
    var fields = alldata.fields;

    // NOTE: Could load different device and orientation column configurations
    // and apply based on device/orientation here .. and update again
    // within onOrientationChange(). If you load multiple configurations
    // It is suggested that you store each of the column arrays to gridcont
    // component for easy retrieval when switching.
    //
    // Ex: gridcont.columnsPhonePortrait = columnsPhonePortrait;
    //      gridcont.columnsTabletLandscape = columnsTabletLandscape;
    //      etc...
    //
    // Simply by calling doRefreshList() method after updating columns your
    // grid will immediately reflect the new column configuration.
    //
    // Could support customized user preferences in this same way.

    var gridListItemId = gridcont.getListItemId();
    var gridlist = gridcont.down('#'+gridListItemId);

    gridcont.setColumns(columnsPhonePortrait);

    // Note: we are defining fields directly within Store instead of creating Model
    var gridstore = Ext.create('Ext.data.Store', {fields: fields});

    gridlist.setStore(gridstore);

    if (!loadStoreInPostProcess) {
        gridstore.suspendEvents();
        var gridloaded = gridstore.add(griddata);
        gridstore.resumeEvents();
        if (loadmask) {Ext.Viewport.setMasked(false);}
    }
}
```

c. Notable Configs

- Store = **not defined**
- cls: [

```
    'x-touchtreegrid-list',
    'x-touchtreegrid-list-dow2012Horiz',
    'x-touchtreegrid-list-dow2012Dynamic'
```

```
]
```

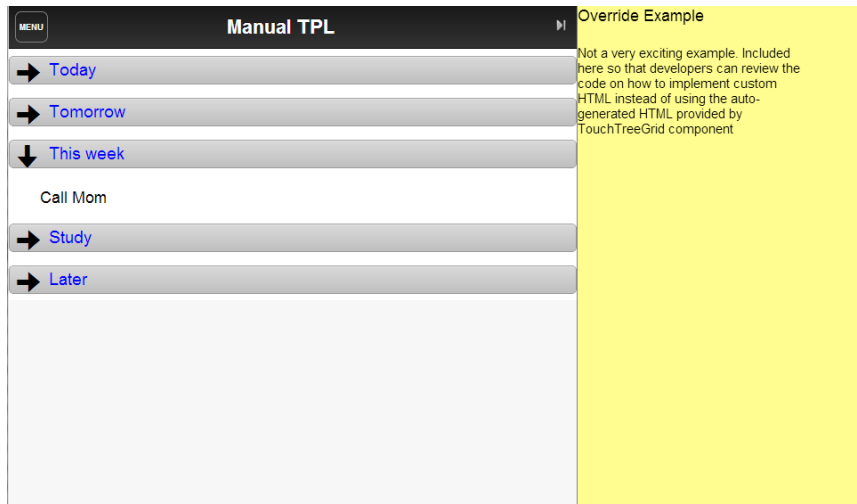
d. CSS (defaults defined in TouchTreeGrid.css, customized by treegriddemo.css)

Uses same CSS as dow2012Horiz. Could be customized further using '**x-touchtreegrid-list-dow2012Dynamic**' selector

# TouchTreeGrid

---

## Override Grid



- a. Demonstrates
  - How automated grid generation can be overridden with use of following configs where developer would specify own TPL for detail, category and header rows (column[] array is not used):
    - i. contentItemTplOverride
    - ii. categlItemTplOverride
    - iii. headerTplOverride
- b. Program Flow
  - Controller: none
  - TabPanel example View Config: Main.js => itemId = 'overrideexample'
  - Slide Navigation example: OverrideExampleCont
  - Data
    - i. TouchTreeGrid.store.OverrideStore (storeId = overrideStore)
    - ii. Model: TouchTreeGrid.model.Task
    - iii. TreeStore (data defined directly in root property)
    - iv. Autoload = true
  - Events: none
- c. Notable Configs
  - includeFooter = false
  - includeHeader = false
  - categlItemTplOverride

# TouchTreeGrid

---

```
categItemTplOverride - object
130 <div style="display : inline-block; white-space : nowrap; width : 100%;font-size: 1.5em;">
131 <tpl if="this.isExpanded(values)">
132 <span class="x-button x-button-small ">
133   <span class="x-button-icon arrow_down x-icon-mask"
134     style="width: .75em; height: .75em; margin-right:0.4em;">
135   </span>
136   <span style="color:blue; padding-left: 1.5em; padding-top: .2em; padding-bottom: .2em;">{text}</span>
137 </span>
138 <tpl else>
139 <span class="x-button x-button-small">
140   <span class="x-button-icon arrow_right x-icon-mask"
141     style="width: .75em; height: .75em; margin-right:0.4em;">
142   </span>
143   <span style="width: 100%; color:blue; padding-left: 1.5em; padding-top: .2em; padding-bottom: .2em;">{text}</span>
144 </span>
145 </tpl>
146 </div>
```

- contentItemTplOverride: '<div style="background-color: white; padding-left: 2em; font-size: 1.2em;">{text}</div>',

d. CSS (defaults defined in TouchTreeGrid.css, customized by treegriddemo.css)

```
.x-touchtreegrid-list-override .touchtreegrid-list-categ {
  background-color: white !important;
  border: none;
  -webkit-box-shadow: none;
}
.x-touchtreegrid-list-override .x-list-normal .x-list-item.x-list-item-tpl{
  border: none;
}
```

# TouchTreeGrid

---

## Notes on Implementation

1. Setup if not using Architect
  - a. Copy 'TouchTreeGrid.js' from ./TouchTreeGrid\_Advanced/app/view/ into similar view directory for your project
  - b. Include 'TouchTreeGrid' in list of 'views' in app.js or your controller
  - c. Copy resource files as discussed below.
  - d. I'm assuming you know what to do from here if not using Architect...
2. Setup if using Sencha Architect
  - a. Import TouchTreeGrid.xdc into your toolbox (via right-click).  
Created using Architect Version: 2.2.2 Build: 991
  - b. Drag this component on top of "Views" in your project inspector to create parent class TouchTreeGrid. This will add a view reference in your "Application" (app.js)
  - c. Suggest copying 'resources' subdirectory from download as is into the same directory level of your project. Key files to include in your project:
    - i. ./resources/css/TouchTreeGrid.css
  - d. Add CSS Resource to 'Resources' section of your project inspector:  
Update url = './resources/css/TouchTreeGrid.css'
  - e. Optionally add custom CSS Resource **after** this one:
    - i. Update url = './resources/css/treegriddemo.css'
    - ii. Definitions in subsequent stack will override prior definitions. When project is saved you will notice in APP.HTML that treegriddemo.css is loaded after TouchTreeGrid.css

### APP.HTML

```
<!DOCTYPE html>

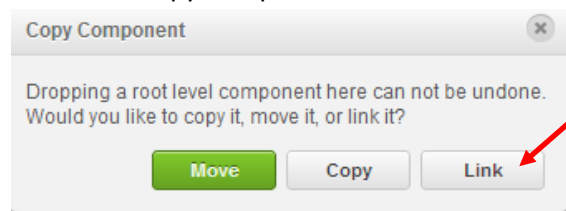
<!-- Auto Generated with Sencha Architect -->
<!-- Modifications to this file will be overwritten. -->
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <title>TouchTreeGrid_Advanced</title>
  <script src="http://cdn.sencha.com/touch/sencha-touch-2.2.1/sencha-touch-all-debug.js">
</script>
  <link rel="stylesheet" href="http://cdn.sencha.com/touch/sencha-touch-2.2.1/
resources/css/sencha-touch.css">
  <link rel="stylesheet" href="./resources/css/TouchTreeGrid.css">
  <link rel="stylesheet" href="./resources/css/treegriddemo.css">
  <script type="text/javascript" src="app.js"></script>
  <script type="text/javascript">
    if (!Ext.browser.is.WebKit) {
      alert("The current browser is unsupported.\n\nSupported browsers:\n" +
        "Google Chrome\n" +
        "Apple Safari\n" +
        "Mobile Safari (iOS)\n" +
        "Android Browser\n" +
        "BlackBerry Browser"
      );
    }
  </script>
```

# TouchTreeGrid

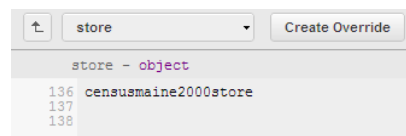
---

```
</head>
<body></body>
</html>
```

- f. Create Model and Store for this grid
  - i. Store must be “Tree Store”.
  - ii. Be sure to define storeId for the Store
- g. Create Container to contain your grid
  - i. Layout = ‘fit’ for standalone implementations
  - ii. Layout = ‘card’ for tabpanel implementations
- h. Drag the ‘TouchTreeGrid’ class on top of this container to make it a child component
  - i. Be sure to Copy Component as a “**Link**”



- i. Click on the added “linked” component under your container and start updating your configs. Minimum required definitions:
  - i. **xtype** : ‘touchtreegrid’ – automatically created when you link the view
  - ii. **itemId** (unique identifier for this grid)
  - iii. **columns** - array defining columns for grid unless loading dynamically (refer to Appendix A)
  - iv. **store** – This is storeId from store. Remove the {} when entering storeId in object edit window



- v. **listItemId** – provide if you want a unique reference to the generated grid list for reference within controller
- vi. **DO NOT** update **createAlias** until you are ready to make this linked instance an “Inline Class” (refer to Appendix C)
- vii. **NOTE:** Linked instances inherit all the configurations defined in TouchTreeGrid component. Architect configuration panel will show these parameters as you search for them and will include the pre-defined default values (which you can override). You only need to define overridden parameters when initially linking to TouchTreeGrid.

# TouchTreeGrid

---

Refer to Appendix C as to how the number of configurations can automatically change when upgrading the component.

- j. Repeat this for all other views.
  - k. Save Often !
3. Notes for Tree Store
- a. Look at code for Task store examples to see what a tree store should look like in object format.
  - b. Look at ./data/treegrid.json from examples to see what tree store should look like in JSON format when loading into a store
    - i. This file was obtained from EXTJS download ./examples/tree/ directory
    - ii. Good tool to validate your JSON: <http://jsonlint.com/>
    - iii. I have found, and others have reported there is a bug with Sencha Store component loading JSON files directly from within the Store. Refer to loadStore() function within CommonController for workaround. This approach would be preferred anyway if you were also loading column arrays from server in same call. One requirement I found for this to work is to initialize root config within the Tree Store as follows: **root: {children: []}**
- ```
Ext.define('TouchTreeGrid.store.Task2', {
  extend: 'Ext.data.TreeStore',    <= Tree Store required !
  requires: ['TouchTreeGrid.model.Task2'],
  config: {
    autoLoad: true,
    model: 'TouchTreeGrid.model.Task2',
    storeId: 'Task2Store',    <= referenced in linked grid config
    root: {children: []}
  }
});
```
4. Test your application on actual devices
- a. If you didn't already know this, you can easily test your application on your own devices as you develop if you have flexible use of your company router or most certainly on your home router network as follows:
    - i. Need local webserver running of course and all source code needs to be underneath webserver directory. For XAMPP Apache server this would be: c:/xampp/htdocs/
    - ii. Get your local IP by running 'ipconfig' in your CMD window
    - iii. From phone/tablet browser enter address similar to this (for provided examples assuming they were extracted as such):  
[http://192.168.1.??/TouchTreeGrid-master/TouchTreeGrid\\_Advanced/app.html](http://192.168.1.??/TouchTreeGrid-master/TouchTreeGrid_Advanced/app.html)



# TouchTreeGrid

---

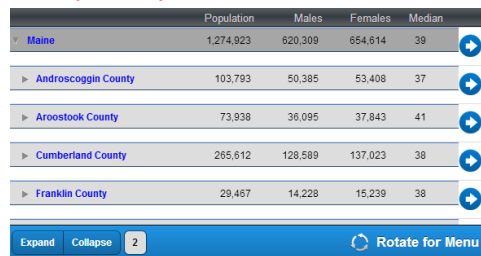
## Additional Functionality included in Advanced Example:

1. Refer to “Summary of Provided Examples” for how JSON data is loaded into TreeStore in varying ways for each of the examples.
2. Example for handling changes in device orientation

- a. “orientationchange” event for viewport is processed within CommonController

```
control: {  
  "viewport": {  
    orientationchange: 'onOrientationChange'  
  }, etc...
```

- b. onOrientationChange() function calls censusController.loadColumnsCensusMaine () function which loads different column configurations for phone (portrait vs. landscape) and tablet (portrait vs. landscape). The number of grid columns dynamically change simply by rotating the device.
- c. onOrientationChange() function calls commonController.hideShowPanels() function to hide titlebar and bottom toolbar for phones (only) when changing from portrait to landscape orientation to allow more vertical display when navigating the grid. **TabPanel example only.**



|                       | Population | Males   | Females | Median |   |
|-----------------------|------------|---------|---------|--------|---|
| ▼ Maine               | 1,274,923  | 620,309 | 654,614 | 39     | ➔ |
| ▶ Androscoggin County | 103,793    | 50,385  | 53,408  | 37     | ➔ |
| ▶ Aroostook County    | 73,938     | 36,095  | 37,843  | 41     | ➔ |
| ▶ Cumberland County   | 265,612    | 128,569 | 137,023 | 38     | ➔ |
| ▶ Franklin County     | 29,467     | 14,228  | 15,239  | 38     | ➔ |

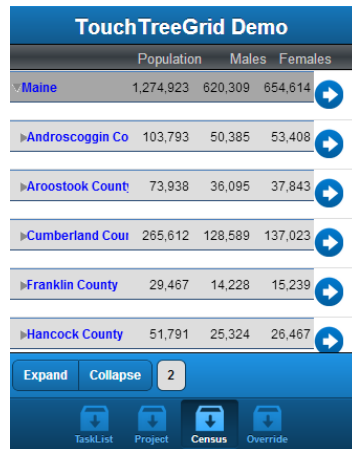
Expand Collapse 2 Rotate for Menu

- d. “Rotate for Menu” text can be customized in hideShowPanels()
- e. Recycle icon can be customized by **landscapelcon='./resources/images/Recycle.png'** config parameter for the linked TouchTreeGrid component.

# TouchTreeGrid

---

- f. Titlebar and Tabbars are unhidden when rotating back to portrait.

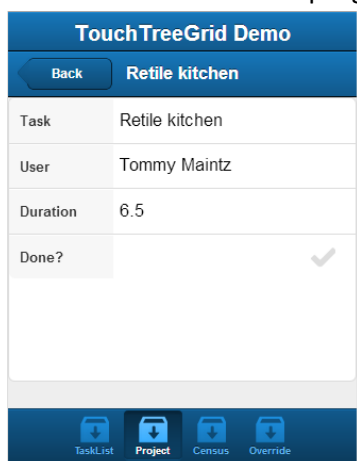


| TouchTreeGrid Demo |            |         |         |
|--------------------|------------|---------|---------|
|                    | Population | Males   | Females |
| ▼ Maine            | 1,274,923  | 620,309 | 654,614 |
| ▶ Androscoggin Co  | 103,793    | 50,385  | 53,408  |
| ▶ Arundel County   | 73,938     | 36,095  | 37,843  |
| ▶ Cumberland Cou   | 265,612    | 128,589 | 137,023 |
| ▶ Franklin County  | 29,467     | 14,228  | 15,239  |
| ▶ Hancock County   | 51,791     | 25,324  | 26,467  |

Expand Collapse 2

TaskList Project Census Override

3. **Griddetailpanel.js** is an example of generic component that can be re-used to display selected row detail fields for multiple grids



| TouchTreeGrid Demo  |                |
|---------------------|----------------|
| Back Retile kitchen |                |
| Task                | Retile kitchen |
| User                | Tommy Maintz   |
| Duration            | 6.5            |
| Done?               | ✓              |

TaskList Project Census Override

- a. TouchTreeGrid.view.griddetailpanel created using Architect and multiple instances of this view can be instantiated. This defines the layout and dummy fieldset which will be overwritten for each implementation. Suggest exporting to file (griddetailpanel.xdc) and importing into toolbox for project re-use
- b. CommonController defines references to parent grid panel (main) and griddetailpanel:

```
refs: {
  main: {
    selector: 'main',
    xtype: 'main'
  },
  griddetailpanel: {
    autoCreate: true,
    selector: 'griddetailpanel',
    xtype: 'griddetailpanel'
  },
}
```

# TouchTreeGrid

---

- c. ProjectController and CommonController includes control definitions:

```
control: {
  "list#example2list": { <= grid listItemId = 'example2list'
    disclose: 'onExample2ListDisclose',
    itemtaphold: 'onExample2ListItemTaphold' <= also trapping taphold
  },
  "button#example2detailbackbtn": { <= back button to navigate back
    tap: 'onExample2GridDetailBackButtonTap'
  },
}
```

- d. onExample2ListDisclose() and onExample2ListItemTaphold() functions both call onExample2ListDiscloseOrHold()

- e. onExample2ListDiscloseOrHold () function instantiates “griddetailpanel” for this particular grid and updates fldSet to display all data fields for this grid.

```
onExample2ListDiscloseOrHold: function(record, target, index) {

  // example2container is itemId of parent container to linked grid instance. We will be
  // swapping the grid instance with the detail panel.

  var swapcont = this.getMain().down('#example2container');
  if (swapcont)
  {
    var newcont = this.getGriddetailpanel(<= create new instance of griddetailpanel
    {
      title : 'Example 2 Detail',
      id : 'example2detail', <= can assign unique ID (not used in this example)
      layout: {type: 'fit'},
      itemId: 'griddetailpanel'
    }
    );

    var gridItemId = swapcont.down('touchtreegrid').getItemId();
    newcont.swapcont = swapcont; <= storing reference to swapcont for generic back button
    newcont.gridItemId = gridItemId; <= storing reference to gridItemId

    if (newcont)
    {
      var newLabel = newcont.down('#griddetaillabel');
      newLabel.setHtml(record.get('task')); <=update label with selected row name

      var fldSet = newcont.down('#griddetailfieldset');
      // Proceed to manually define fields to display all fields for selected row

      var result = fldSet.setConfig({
        items : [
          {label: 'Task', xtype: 'textfield', readOnly: true, value: record.data.task},
          {label: 'User', xtype: 'textfield', readOnly: true, value: record.data.user},
          {label: 'Duration', xtype: 'numberfield', readOnly: true,
            value: record.data.duration},
          {label: 'Done?', xtype: 'checkboxfield', disabledCls: null,
            checked: record.data.done}
        ]});

      swapcont.add(newcont); <= adding new griddetailpanel
      swapcont.setActiveItem(newcont); <= making it the active item for display
    }
  }
}
```

# TouchTreeGrid

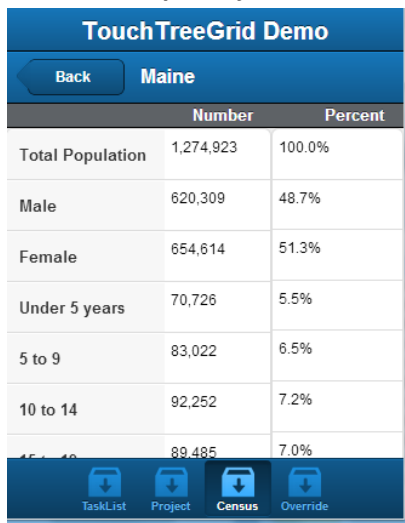
---

- f. `onGridDetailBackButtonTap()` function handles Back button press to navigate from `griddetailpanel` back to the grid

```
// Reusing the Back button for all Project Task examples by storing references when creating
detail panel (list disclose)
var swapcont = button.up('#griddetailpanel').swapcont; <= see onExample2ListDiscloseOrHold()
if (swapcont)
{
    gridItemId = button.up('#griddetailpanel').gridItemId; <= see onExample2ListDiscloseOrHold()
    var newcont = swapcont.down('#'+gridItemId);

    newcont.setShowAnimation({type : "slide", direction : "right"});
    swapcont.setActiveItem(newcont);
}
```

4. `Censusdetailpanel.js` is a custom implementation of `griddetailpanel` documented above



The screenshot shows a mobile application interface titled "TouchTreeGrid Demo". It features a blue header bar with a "Back" button and the word "Maine". Below the header is a table with two columns: "Number" and "Percent". The table contains data for various demographic groups. At the bottom of the screen, there is a navigation bar with four icons labeled "TaskList", "Project", "Census", and "Override".

|                  | Number    | Percent |
|------------------|-----------|---------|
| Total Population | 1,274,923 | 100.0%  |
| Male             | 620,309   | 48.7%   |
| Female           | 654,614   | 51.3%   |
| Under 5 years    | 70,726    | 5.5%    |
| 5 to 9           | 83,022    | 6.5%    |
| 10 to 14         | 92,252    | 7.2%    |
| 15 to 19         | 89,485    | 7.0%    |

- a. Architect is used to quickly create the layouts with dummy fieldsets for Number and for Percent columns.
- b. `Censusdetailpanel` is then instantiated similar to `griddetailpanel`
- c. `CensusController` defines the needed references and controls

```
refs: {
    censusdetailpanel: {
        autoCreate: true,
        selector: 'censusdetailpanel',
        xtype: 'censusdetailpanel'
    }
}, etc...

control: {
    "list#censusmainelist": {
        disclose: 'onCensusMaineListDisclose'
    },
    "button#censusdetailbackbtn": {
        tap: 'onCensusDetailBackButtonTap'
    }, etc...
}
```

# TouchTreeGrid

---

- d. Fieldsets for Number and Percent are manually updated within onCensusMaineListDisclose() function.
  - e. Back button functionality handled within onCensusDetailBackButtonTap()
5. **onMainTabpanelActiveItemChange()** within CommonController contains logic to load Project and Census data only when user clicks on those tabs to speed up initial launch time. **TabPanel example only.**
- a. Data only loaded if Store is empty
6. **onTitlebarGridhelp()** function with CommonController traps touches on TitleBar and launches modal information window custom for each Grid
- a. GridHelpPanel view defines modal panel to display HTML
  - b. HTML files for each grid stored in ./resources/html/ directory
  - c. AJAX call made to load HTML file into panel

```
...see code
gridId = grid.getHelpHtml(); <= URL reference stored to helpHtml config
                                for each linked instance of TouchTreeGrid

Ext.Ajax.request({
    url: gridId,
    method: 'GET',
    callback: function(options, success, response) {

        var help = me.getGridHelpPanel();
        help.setHtml( response.responseText );
        help.showBy(image);
    }
});
```

7.

# TouchTreeGrid

---

## APPENDIX A – Columns Array Definitions

Column configurations are defined within a COLUMNS array for each linked grid instance:

- header - Text to include in column header.
  - Can embed HTML tags for formatting within the string
  - 'My <i>Header</i>' displays as 'My *Header*'
  - 'My <b>Header</b>' displays as 'My **Header**'
- dataIndex - data column from treestore. **Upper/lower case must match model fields!**
- width - CSS format width for this component (suggest % or em's to better support all device types)
- style - Additional CSS styling commands for detail row cells
- css - Additional CSS selector for detail row cells (style take precedence over css)
- styleSorted - style to apply when this column is sorted
- cssSorted - Additional CSS selector for detail row cells when sorted (style take precedence over css)
- categStyle - Additional CSS styling commands for category rows  
(not applicable where simpleList = true)
- categCss - Additional CSS selector for category row cells (style take precedence over css)
- categStyleSorted – style to apply to category row when this column is sorted
- categCssSorted - Additional CSS selector for category row cells when sorted  
(style take precedence over css)
- headerStyle - Additional CSS styling commands for column header row
- headerCss - Additional CSS selector for header row cells (style take precedence over css)
- headerStyleSorted – style to apply to header row when this column is sorted
- headerCssSorted - Additional CSS selector for header row cells when sorted  
(style take precedence over css)
- renderers - refer to “Advanced Features” sections for discussion on use
- sortable - true to make this column sortable (requires columnSorting=true for linked instance of TouchTreeGrid).
- addDataIndexToDiv – Include this attribute and set to “true” if you want to include “dataIndex” attribute in the DIV. Example DIV from Horizontal Scrolling example:

```
<p class="touchtreegrid-simplelist-cell " style="min-width:4.375em
!important;max-width:4.375em !important;width:4.375em !important;text-
align: right;font-weight: bold; padding-right: .4em;"
dataindex="Close">13,139</p>
```

# TouchTreeGrid

---

This would be used to uniquely identify tap on specific cells within a grid. Refer to Horizontal Scrolling and Dynamic Grid examples discussed above for further details.

Columns array example for Project list example:

```
columns: [
  {
    header: 'Task',
    dataIndex: 'task',
    width: '35%',
    style: 'text-align: left;',
    categStyle: 'font-weight: bold; text-align: left; color: blue;',
    headerStyle: 'text-align: left; color: #ccc;'
  },
  {
    header: 'User',
    dataIndex: 'user',
    width: '35%',
    style: 'text-align: left; padding-left: .5em;',
    categStyle: 'text-align: left; padding-left: .5em;',
    headerStyle: 'text-align: left; color: #ccc; padding-left: .5em;'
  },
  {
    header: 'Dur',
    dataIndex: 'duration',
    width: '15%',
    style: 'text-align: right;',
    categStyle: 'text-align: right;',
    headerStyle: 'text-align: right; color: #ccc;'
  }
]
```

Refer to loadColumnsCensusMaine() function within censusController in Advanced example for additional number of column variations for Phone (portrait vs. landscape) and Tablet (portrait vs. landscape).

Refer to Dynamic Grid for example of CSS column styling as opposed to using STYLE-related.

# TouchTreeGrid

---

## APPENDIX B – CSS Styling

Every project should include a copy of **TouchTreeGrid.css** which defines the default styling for TouchTreeGrid component. Any custom overrides should be included in a file loaded after TouchTreeGrid.css. **treegriddemo.css** contains examples of custom styling overrides.

### Excellent reference tools:

- <http://www.w3schools.com/cssref/default.asp> (CSS3 properties reference)
- <http://www.colorschemedesigner.com/> (provides compliment and triad colors to given color)
- <http://www.color-hex.com/> (provides shades and tints of particular color which is useful for gradients)
- <http://www.colorzilla.com/gradient-editor/> (provides tools to create gradient, generates styling for copy/paste into your CSS file)
- <http://www.gimp.org/> (image manipulation program)

### Review of the TouchTreeGrid CSS classes and how they are used by TouchTreeGrid.css:

- **.x-touchtreegrid-list** is the default class for the entire component. Each linked instance should at minimum keep this class and possibly include additional overrides. Example of secondary CSS class reference (Example2):

```
cls: [  
    'x-touchtreegrid-list',  
    'x-touchtreegrid-list-example2'  
],
```

- Every entry in TouchTreeGrid.css is prefixed with **.x-touchtreegrid-list**. Every custom override would be written to treegriddemo.css (or similar) and prefixed with the custom class (i.e. **.x-touchtreegrid-list-example2**) instead of default class (**.x-touchtreegrid-list**), followed by the component specific class name. Example:

- TouchTreeGrid.css contains this definition:  
**.x-touchtreegrid-list .x-touchtreegrid-item** {  
 background-color: **white**;  
}
- If treegriddemo.css contained this definition:  
**.x-touchtreegrid-list-example2 .x-touchtreegrid-item** {  
 background-color: **red**;  
}



# TouchTreeGrid

---

- Then for Example2 the background would be overridden to red.
- Default background color for grid list items:  

```
.x-touchtreegrid-list .x-touchtreegrid-item {  
    background-color: white;  
}
```
- [Touch 2.1] treegriddemo.css contains this definition to hide or modify horizontal lines:  
(both Project #2 and #3 examples have cls : 'x-touchtreegrid-list-example2B')  

```
.x-touchtreegrid-list-example2B .x-list-normal .x-list-item .x-dock-horizontal {  
    border: none;  
}
```
- [Touch 2.2] treegriddemo.css contains this definition to hide or modify horizontal lines:  
(both Project #2 and #3 examples have cls : 'x-touchtreegrid-list-example2B')  

```
.x-touchtreegrid-list-example2B .x-list-normal .x-list-item .x-dock-horizontal ,  
.x-touchtreegrid-list-example2B .x-list-normal .x-list-item.x-list-item-tpl{  
    border: none;  
}
```
- Default layout for category rows:  

```
.x-touchtreegrid-list .touchtreegrid-list-categ {  
    background-color: #ddd !important;  
    background-image: none !important;  
    color: black;  
    border-top: 1px solid #4D80BC;  
    border-bottom: 1px solid #2D4E76;  
    font-weight: normal;  
    min-height: 2.6em !important;  
    padding: 0.6em 0 0 0 !important;  
    width: 100% !important;  
    font-size: .7em !important;  
    -webkit-box-shadow: 0px 0.1em 0.3em rgba(0, 0, 0, 0.3);  
}
```
- Default layout for Detail rows  

```
.x-touchtreegrid-list .touchtreegrid-list-content{  
    color: black;  
    background-color: white;  
    width: 100%;  
    padding: 1em 0 0 0;
```

# TouchTreeGrid

---

```
margin:0;
font-size: .8em !important;
}
```

- Default layout for Header row

```
.x-touchtreegrid-list .touchtreegrid-header {
width:100%;
padding:0.4em 0 0 0;
margin:0;
color: #fff;
height : 1.6em;
font-size: .8em !important;
see CSS file for actual gradient definitions as too large here..
}
```

- Default layout for Category row individual cells

```
.x-touchtreegrid-list .touchtreegrid-list-categ-cell{
overflow : hidden;
text-overflow : clip;
white-space : nowrap;
}
```

- Default layout for Header row individual cells

```
.x-touchtreegrid-list .touchtreegrid-header-cell {
white-space: nowrap;
overflow: hidden;
text-overflow: clip;
}
```

- Default layout for Detail row individual “cells”

```
.x-touchtreegrid-list .touchtreegrid-list-content-cell{
overflow : hidden;
text-overflow : clip;
white-space : nowrap;
}
```

- Default for Simple List “cells”

```
.x-touchtreegrid-list .touchtreegrid-simplelist-cell{ /* simple list cells */
display : inline-block;
overflow : hidden;
text-overflow : clip;
```

# TouchTreeGrid

---

```
white-space : nowrap;
color: black;
background-color:white;
padding:1em 0 0 0;
margin:0;
font-size: .8em !important;
}
```

- Handle colors for simple grid list item "selected" and "pressing" (selectors for Treegrid selection not provided as deemed not appropriate but handled similarly)

```
.x-touchtreegrid-list .touchtreegrid-item-selected,
.x-touchtreegrid-list .touchtreegrid-item-selected .touchtreegrid-simplelist-cell{
    background-color: #006bb6 !important; /* Sencha 2.1 default color for selected */
    color: white !important;
}
.x-touchtreegrid-list .touchtreegrid-item-selected .x-list-disclosure {
    background: #fff none; /* reverses color on disclosure icon for selected rows */
}
.x-touchtreegrid-list .touchtreegrid-item-pressed,
.x-touchtreegrid-list .touchtreegrid-item-pressed .touchtreegrid-simplelist-cell{
    background-color: #b6e1ff !important; /* Sencha 2.1 default color for pressing */
    color: white !important;
}
```

- Default layout for Category row arrow

```
.x-touchtreegrid-list .touchtreegrid-details-img { /* category row arrow */
    width: 18px;
    min-height: 18px;
    height: 100%;
    display: -moz-inline-box;
    vertical-align: top;
    display: inline-block;
    background-image: (see file for embedded sprite ... contains both arrow left and arrow down)
    background-repeat: no-repeat;
    vertical-align: top;
    cursor: pointer;
    border: 0;
    border-image: initial;
}
```

Associated references to sprite:

```
.x-touchtreegrid-list .touchtreegrid-details-img-open {
    /* 2nd arrow sprite pointing down for open category */
    background-position: 0 -18px;
```

# TouchTreeGrid

---

```
}
```

```
.x-touchtreegrid-list .touchtreegrid-details-img-close {  
  /* 1st arrow sprite pointing right for closed category */  
  background-position: 0 0;  
}
```

- Sort up/down arrows defined in TouchTreeGrid.css
  - .x-touchtreegrid-list .x-grid-sort-desc {...}
  - .x-touchtreegrid-list .x-grid-sort-asc {...}
  - background-position: right top;
    - This can be change to locate to different areas within column header cell
    - I found that best appearance achieved by adding "padding-right: .5em !important;" to headerStyle in column[.]
- Defaults for footer toolbar (TouchTreeGrid uses Sencha defaults)

```
.x-touchtreegrid-list .touchtreegrid-footer { <= no changes to Sencha defaults  
}
```

Example2 overrides this:

```
.x-touchtreegrid-list-example2 .touchtreegrid-footer {  
  background: #5e6266; <= gray background instead of Sencha default  
}
```

- Default layout for expand/collapse buttons

```
.x-touchtreegrid-list .touchtreegrid-expand-collapse-buttons {  
  margin-top: .4em;  
}
```

- Default layouts for Content, Category, and Header rows:

```
/* Added 5/26/2013 */  
.x-touchtreegrid-list .touchtreegrid-header-cell {  
  white-space: nowrap;  
  overflow: hidden;  
  text-overflow: clip;  
}
```

```
/* Added 8/19/2013 */  
.x-touchtreegrid-list .css-categ-row {  
  display: flex;  
  display: -webkit-box;  
  display: -ms-flexbox;  
  flex-direction: row;  
  -webkit-box-orient: horizontal;
```

# TouchTreeGrid

---

```
        -ms-flex-direction: row;
    }
    .x-touchtreegrid-list .css-content-row {
        display: flex;
        display: -webkit-box;
        display: -ms-flexbox;
        flex-direction: row;
        -webkit-box-orient: horizontal;
        -ms-flex-direction: row;
    }
    .x-touchtreegrid-list .css-header-row {
        display: flex;
        display: -webkit-box;
        display: -ms-flexbox;
        flex-direction: row;
        -webkit-box-orient: horizontal;
        -ms-flex-direction: row;
    }
}
```

- Default layout for label reading “Rotate for Menu” when phone is rotated from portrait to landscape

```
.x-touchtreegrid-list .touchtreegrid-landscape-label {
    margin: .7em .5em 0 0;
}
```

- Default layout for Recycle icon displayed when phone is rotated from portrait to landscape

```
.x-touchtreegrid-list .touchtreegrid-landscape-icon {
    height: 1.5em !important;
    margin: .5em .5em 0 0;
    width: 1.5em !important;
}
```

- Custom iPhone style Disclose Icon (treegriddemo.css)

```
.x-touchtreegrid-list-example2 .x-list-disclosure{
    width: 12px;
    height: 1.5em;
    margin: .4em 0 0 .5em;
    -webkit-mask: none;
    -webkit-mask-box-image: (see file for embedded image)
}
```

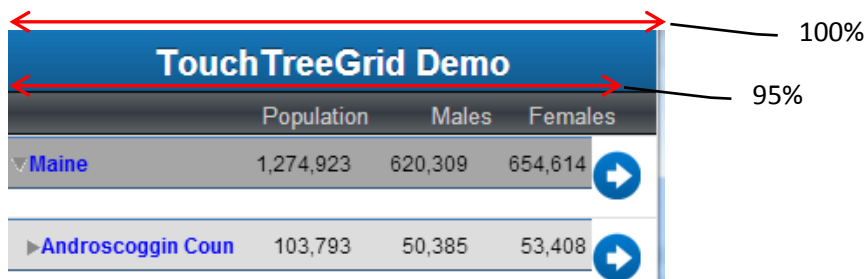
Note: you may or may not want to use this icon depending on your experience with sensitivity when touching icon on your particular device. Project #1 example also implements itemtap hold event as secondary means to react to user’s attempt to navigate.

# TouchTreeGrid

- [Touch 2.2+] Added .x-list-disclosure:before selector to treegriddemo.css to disable arrow picto from overlaying Example #2 I-Phone style icon:

```
.x-touchtreegrid-list-example2 .x-list-disclosure:before{
  content: "";
  font-family: "";
}
```

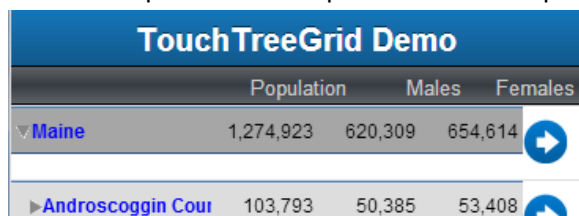
- [Following not applicable for Touch 2.2+ as disclosure icons no longer squeeze list item row] Define spacer width when using disclose icons such that header columns line up with category/detail columns:



	Population	Males	Females
▼ Maine	1,274,923	620,309	654,614
▶ Androscoggin Coun	103,793	50,385	53,408

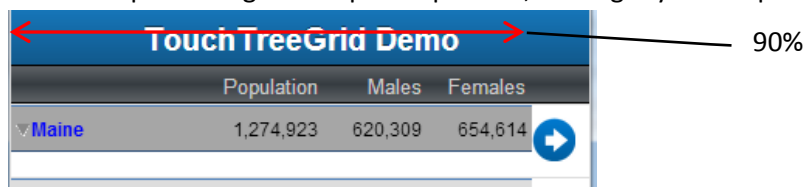
```
.touchtreegrid-disclose-spacer {
  max-width: 95%;
}
```

100% example reveals the problem with no spacer:



	Population	Males	Females
▼ Maine	1,274,923	620,309	654,614
▶ Androscoggin Coun	103,793	50,385	53,408

90% example looks great on phone portrait, but slightly off for phone landscape:



	Population	Males	Females
▼ Maine	1,274,923	620,309	654,614

# TouchTreeGrid

---

	Population	Males	Females	Median	
▼ <b>Maine</b>	1,274,923	620,309	654,614	39	▶
▶ <b>Androscoggin County</b>	103,793	50,385	53,408	37	▶

This is just an approximation and will look slightly different on different devices and orientation, but it is better than no attempt to adjust the display (will spend more time on this later).

Default was left at 95% for phone landscape and tablet configs. Note we implemented a similar change to Mitchell Simeon's Ext.ux.touch.grid when using disclosure icons.

# TouchTreeGrid

---

## APPENDIX C – Upgrading TouchTreeGrid component

The following steps can be used to upgrade to a new version of the TouchTreeGrid component within Architect:

1. **Backup your project!**
2. Update **createAlias='touchtreegrid'** on all linked instances of TouchTreeGrid within Architect's configuration panel (Note: this is an Architect-only setting ... you will not immediately see a code change). You should not update createAlias until you are ready to do step #3 to updating linked instances as "Inline Class" as you could observe issues running your code (i.e. **perform steps #2 and #3 in unison**)
3. Right-click each linked instance and convert to "Inline Class".
  - a. This removes the link and creates a copy of all previously defaulted configurations with the default values. If you want to keep you code size small you can go back and delete the added ones, but this is not necessary.
  - b. xtype = 'touchtreegrid' should remain after removing
    - i. If it doesn't it's because you didn't update createAlias above.
  - c. Be sure to add any new configurations supported by the upgraded component if you do not want to accept the default values
  - d. Prior upgrades that are already Inline do not need to be modified unless you need to add newly supported configurations.
4. **Repeat step 2 and 3 for ALL linked instances (else they will be deleted !!)**
5. Make sure there are no references to 'touchtreegrid' in controller config->"control" section within Architect's config panel. I specifically updated onFirstExampleLeafItemTap() and onFirstExampleNodeItemTap() functions as follows:
  - a. **"targetType"** should be defined as "Ext.Container" instead of "TouchTreeGrid".
  - b. **"control query"** should read "container#firstexample" instead of "touchtreegrid#firstexample"



# TouchTreeGrid

---



- c. If you have references to **touchtreegrid**, any custom parameters will be lost when you load the new component !

## 6. Save the Project !

7. Once you are sure you converted all linked instances, delete the TouchTreeGrid component from you Project Inspector.
8. Right-click in Toolbox and import new version of TouchTreeGrid.xdc (be sure to override default name of 'container' to 'touchtreegrid')
9. Drag the newly imported component ontop of "View" within Project Inspector.
10. Copy/overlay new version of **TouchTreeGrid.css** if updated.

## 11. Save the Project !

12. Confirm that "TouchTreeGrid" is referenced as a view in the Application component within Project Inspector.
  - a. Exception: You may be referencing this within specific controllers and would need to manually add the reference within these controllers as appropriate.

# TouchTreeGrid

## APPENDIX D – TouchTreeGrid Config Definitions

Note: minimum required configurations to define in linked instance of TouchTreeGrid:

- **xtype** : 'touchtreegrid'
- **store**
- **itemId**
- **columns[]** - refer to Appendix A

Configuration	Default	Commonly Updated in Linked Instance	Ignored if simpleList = true?	Purpose
additionalListConfigs	{}			Any config supported by Ext.dataview.List component not already contained in this list can be applied to the auto-generated Grid list component. Configs defined here would override any similar configs defaulted by TouchTreeGrid. Examples: {scrollable: {direction: 'horizontal', directionLock: true}, etc... }  For simpleList=true examples: { grouped: true, indexBar: true, etc... }
applyDefaultCollapseLevel	true		Y	Set to false if collapse levels defined on server side.
arrowPctWidth	4		Y	Width of screen to consume for expand/collapse arrow. 4% good for phones and 2% practical for tablets
categColumns[]	[]		Y	Optional means to define categStyle with different column widths than what would otherwise be defined in columns[] categStyle config.
categCssArr[]	[]		Y	Array defining CSS selector to apply to category depth 1,2,3, etc.. Default selector is '.touchtreegrid-list-categ'. This provides more control than simple color rendering by level supported by categDepthColorsArr[] config.
categDepthColorButtons	true	Y	Y	Set to false to not apply categDepthColorsArr colors to "collapse-to-level" buttons in footer (default sencha colors would then apply)
categDepthColors	false	Y	Y	Set to true to apply default color schemes defined by categDepthColorsArr to category rows.
categDepthColorsArr[]	['#a6a6a6', '#dddddd', 'white']	Y	Y	Array defining colors to apply to category depths 1,2,3, etc... Default color is white for colors not defined for given depth.
categIndentPct	'3'		Y	Each expanded level is indented by 3% of screen by default.
categItemTpl	"		Y	For internal use only.

# TouchTreeGrid

Configuration	Default	Commonly Updated in Linked Instance	Ignored if simpleList = true?	Purpose
catgItemTplOverride	"		Y	Used by TouchTreeGrid but user could provide own TPL for category rows (see Manual example)
cls	'x-touchtreegrid-list'	Y		<p>If specifying custom CSS, always include 'x-touchtreegrid-list' class in array defining any custom class overrides. Example:</p> <pre>[ 'x-touchtreegrid-list',   'x-touchtreegrid-list-example2',   'x-touchtreegrid-list-example2C' ]</pre> <p>Refer to Project #3 example. 'x-touchtreegrid-list' class applies minimum required CSS defined in TouchTreeGrid.css</p> <p>'x-touchtreegrid-list-example2' overrides some of the CSS defined by 'x-touchgrid-list'. Refer to treegriddemo.css. Project #1 CSS overrides are copied to Project #2 and #3. 'x-touchtreegrid-list-example2C' overrides specific configurations from 'x-touchtreegrid-list-example2'. See treegriddemo.css.</p>
colNumberToTruncateForIndents	1		Y	catgIndntPct is added for each indented level. To line up columns, we need to subtract this incremental percentage from another column. Column 1 is the default (i.e. the 1 <sup>st</sup> column).
columns	[]	Y		Refer to Appendix A.
columnSorting	false	Y		Typically set to 'true' when SimpleList=true. Each column in the columns array must also have attribute "sortable: true" (refer to Appendix A)
contentItemTpl	"			For internal use only.
contentItemTplOverride	"			Used by TouchTreeGrid but user could provide own TPL for content/detail rows (see Manual example)
cssContentRow (added 8/25/13)	'css-content-row '			<pre>.x-touchtreegrid-list .css-content-row {   display: flex;   display: -webkit-box;   display: -ms-flexbox;   flex-direction: row;   -webkit-box-orient: horizontal;   -ms-flex-direction: row; }</pre>
cssCategRow (added 8/25/13)	'css-categ-row '		Y	<pre>.x-touchtreegrid-list .css-categ-row {   display: flex;   display: -webkit-box;   display: -ms-flexbox;   flex-direction: row;   -webkit-box-orient: horizontal;   -ms-flex-direction: row; }</pre>

# TouchTreeGrid

Configuration	Default	Commonly Updated in Linked Instance	Ignored if simpleList = true?	Purpose
cssHeaderRow (added 8/25/13)	'css-header-row '			.x-touchtreegrid-list .css-header-row { display: flex; display: -webkit-box; display: -ms-flexbox; flex-direction: row; -webkit-box-orient: horizontal; -ms-flex-direction: row; }
cssSimpleRow (added 11/23/14)	"			CSS selector to conditionally style rows. Refer to Basic #2 example:  cssSimpleRow: '[[{(values.Close >= 13200 ? "highlight-rows-over-13200" : "")}]]',
customColumnSortEvent	"			Developer-specified event to fire for custom column sorting. TouchTreeGrid provided sorting will be disabled. Refer to logic in handleColumnSort() method for parameters passed with event.
customFooterItems (added 8/25/13)	{}		Y	Allows user-defined object of components to be added to footer toolbar (i.e. to same toolbar as expand/collapse buttons). Requires includeCustomFooterItems=true config. Refer to CalendarPicker component for sample usage
customExpCollapseEvent	"		Y	Refer to Advanced Features discussion
defaultCollapseLevel	99	Y	Y	Tree level to expand to for initial display (99 is fully expanded).
disableExpandCollapse	False		Y	Allows expand/collapse feature for TreeGrids to be disabled and 'frozen' in initial rendered state.
disableSelection	true			Recommended to disable selection when using onItemDisclosure. Reason is to avoid confusion where user could select one row and disclose from another. Refer to TouchTreeGrid.css selectors ".touchtreegrid-item-selected", ".touchtreegrid-item-pressed" and ".touchtreegrid-simplelist-cell" on how to specify selection and pressing colors. Also refer to TouchTreeGrid configs: pressedCls and selectedCls on how to define custom selectors if needed.
disclosureProperty	'disclose'			Only applicable for onItemDisclosure=true. 'disclose' used to add disclosure icon for all rows. 'leaf' used to only add disclosure icon on leaf rows.

# TouchTreeGrid

Configuration	Default	Commonly Updated in Linked Instance	Ignored if simpleList = true?	Purpose
footer	{ xtype: 'toolbar', docked: 'bottom', ui: 'light', cls: 'touchtreegrid-footer' }		Y	Used to auto-add toolbar to bottom of grid for expand/collapse buttons. Additional toolbar configurations such as height, color, etc.. can be defined by overriding this.
footerDock (added 8/25/13)	'bottom'		Y	Valid values are 'top' or 'bottom'. Defines how auto-generated footer toolbar containing expand/collapse plus optional custom buttons will be docked. If docked top it will appear above the column header toolbar.
Header (updated 10/26/13: Height configs removed from component styling and added to TouchTreeGrid.css)	{ xtype: 'toolbar', docked: 'top', cls: 'touchtreegrid-header', <del>maxHeight: '1.8em',</del> <del>minHeight: '1.8em'</del> }			Used to auto-add toolbar to top of grid for column headers. Additional toolbar configurations such as height, color, etc.. can be defined by overriding this.
headerTpl	"			For internal use only.
headerTplOverride	"			Used by TouchTreeGrid but user could provide own TPL for header rows (see Manual example)
helpHtml	"			Used for demo only.
hideExpandCollapseBtns (added 8/25/13)	False		Y	When true causes the auto-generated Expand/Collapse buttons to be hidden on the footer toolbar, where custom buttons defined by customFooterItems config would persist.
includeCustomFooterItems (added 8/25/13)	false		Y	Refer to customFooterItems config.
includeFooter	true		Y	Set false to prevent auto-creation of footer. Not applicable for simple grids (simpleList=true).
includeFooterLevels	true		Y	False to not include 'collapse-to-level' buttons on footer.
includeHeader	true			Set false to prevent auto-creation of column header.
infinite	True			[Touch 2.2x] Refer to Sencha Touch documentation.
itemHeight	47			47 pixel height for list items is standard sencha default
landscapelcon	"		Y	Refer to Advanced Features discussion.

# TouchTreeGrid

Configuration	Default	Commonly Updated in Linked Instance	Ignored if simpleList = true?	Purpose
layout	{type: 'vbox'}			For internal use only. <b>Parent container to linked instance should have Layout = 'card' or 'fit' typically. 'hbox' for horizontal scrolling.</b>
linkedGridsArr (added 10/7/13)	[]			<p>linkedGridsArr[], linkedGridsParentItemId and onScrollOptions{} configs apply to linked grids for synchronous scrolling and sorting. Also refer to "Census Freeze Column Example" documentation above.</p> <p><b>linkedGridsParentItemId</b> = itemId of ultimate parent container for all synchronized grids.</p> <p><b>linkedGridsArr[]</b> is array of objects defined for each of the individual scrolling grids and defines the <u>itemIds</u> of the other participating scrolling grids. TouchTreeGrid uses these definitions to obtain the actual object references for each of the grids to support synchronized scrolling. <b>Other</b> attributes internally updated for each object:</p> <ul style="list-style-type: none"> <li>• item = internally updated as actual object reference for itemId</li> <li>• scroller = internally updated as actual scroller object reference for itemId</li> <li>• onScrollOptions = internally updated as onScrollOptions object for itemId (if defined)</li> </ul> <p><b>onScrollOptions</b> object is optional and provides developer option to pass "options" object when processing 'scroll' event. One example of this would be to throttle processing of scroll events. Refer to on() method for Ext.Container in Sencha's documentation for more.</p>
linkedGridsParentItemId (added 10/7/13)	"			Refer to linkedGridsArr[]
list	{}			For internal use only.
listItemId	'touchtreegridlist'	Y		To add any logic in your controller you should assign a unique itemId for each auto-generated grid.
listPlugins	{}			Refer to Advanced Features discussion
listScrollable	true			Generally all lists would be scrollable unless you were implementing something like a dashboard and only wanted the first 4 items for example to appear.
mode	SINGLE			Refer to Sencha Touch documentation for Ext.dataview.List
onItemDisclosure	false	Y		Refer to Sencha Touch documentation for Ext.dataview.List

# TouchTreeGrid

Configuration	Default	Commonly Updated in Linked Instance	Ignored if simpleList = true?	Purpose
onScrollOptions (added 10/7/13)	{}			Refer to linkedGridsArr[]
pressedCls	'touchtreegrid-item-pressed'			TouchTreeGrid.js includes logic to match pressed color to display color to avoid unwanted color flickering for onDisclosure. For cases where row selection is desired (simpleList=true), TouchTreeGrid.css contains 'touchtreegrid-item-pressed' selectors to specify color while pressing an item.
renderers	{}			Refer to Advanced Features discussion
selectedCls	'touchtreegrid-item-selected'			For cases where row selection is desired (simpleList=true), TouchTreeGrid.css contains 'touchtreegrid-item-selected' selectors to specify color for selected items.
simpleList	false	Y		True for basic non-tree grids. Refer to "DOW Hist" example.
singleExpand	false	Y	Y	Typically used for Accordions where only one sibling branch expanded at a time.
store	"	Y		Must be Ext.data.TreeStore for Tree Grids and Accordions (simpleList = false). If not provided then temporary dummy store will be auto-created within updateStore() method to prevent Sencha's List component from generating error upon creation. The only time this should be omitted is when implementing dynamic grids where store is auto-created when data is loaded.
styleCategRow (updated 8/25/13)	<del>'display: webkit-box; webkit-box-orient: horizontal;'</del>		Y	Refer to Advanced Features discussion
styleContentRow (updated 8/25/13)	<del>'display: webkit-box; webkit-box-orient: horizontal;'</del>			Refer to Advanced Features discussion
styleHeaderRow (updated 8/25/13)	<del>'display: webkit-box; webkit-box-orient: horizontal;'</del>			Refer to Advanced Features discussion
styleSimpleRow (added 11/23/14)	"			Refer to cssSimpleRow. Instead of defining CSS this would define Style attributes to conditionally style rows.
useSimpleItems	True			[Touch 2.2x] Refer to Sencha Touch documentation.
variableHeights	true			Refer to Sencha Touch documentation.