

TouchTreeGrid

TouchTreeGrid is an extremely versatile and easy to implement Sencha Touch 2.1x custom component that supports Tree and Accordion grid view layouts. It utilizes Sencha's Ext.dataview.List component and Disclosure events are supported for leafs and optionally for category rows. This software can be downloaded at <https://github.com/swluken/TouchTreeGrid> and is free to use (refer to associated MIT.LISCENSE) . **TouchTreeGrid** was developed entirely within Sencha Architect (v2.2) designer product. An Architect component is provided for import into your toolbox (TouchTreeGrid.xdc). Using Sencha Architect you can implement this component for new grid implementations very rapidly. Both basic and advanced examples are included in the download. Scrolling and overall functionality was found to be very fast for a larger data example containing +3000 rows and 3 category levels¹.

Contents

SAMPLE: Project Status – 3 Variations.....	2
SAMPLE: US Census – Phone Portrait	2
SAMPLE: US Census – Tablet Landscape.....	3
SAMPLE: US Census – Tablet Portrait	4
Intended Audience.....	4
Credits	4
Basic Features	5
Advanced Features	6
Summary of Provided Examples	10
Notes on Implementation.....	12
Configuration: Project Status Grid #1	15
Configuration: Project Status Grid #2 (differences from #1)	16
Configuration: Project Status Grid #3 (differences from #2)	17
Configuration: CENSUS Grid.....	17
Configuration: Manual Grid	18
Additional Functionality included in Advanced Example:.....	18
APPENDIX A – Columns Array Definitions.....	26
APPENDIX B – CSS Styling.....	27
APPENDIX C – Upgrading TouchTreeGrid component.....	33

¹ Devices tested: Android 4, iPhone 5 and iPad (3rd generation)

TouchTreeGrid

SAMPLE: Project Status – 3 Variations

TouchTreeGrid Demo	TouchTreeGrid Demo	TouchTreeGrid Demo
Ex #1 Ex #2 Ex #3	Ex #1 Ex #2 Ex #3	Ex #1 Ex #2 Ex #3
Task User Dur	Task User Dur	Task User Dur
▼Project: Shopping Tommy Maintz 13.25	▼Project: Shopping Tommy Maintz 13.25	▼Project: Shopping Tommy Maintz 13.25
►Housewares Tommy Maintz 1.25	►Housewares Tommy Maintz 1.25	►Housewares Tommy Maintz 1.25
▼Remodeling Tommy Maintz 12	▼Remodeling Tommy Maintz 12	▼Remodeling Tommy Maintz 12
Retile kitchen Tommy Maintz 6.5	Retile kitchen Tommy Maintz 6.5	Retile kitchen Tommy Maintz 6.5
►Paint bedroom Tommy Maintz 2.75	►Paint bedroom Tommy Maintz 2.75	►Paint bedroom Tommy Maintz 2.75
Decorate living r Tommy Maintz 2.75	Decorate living r Tommy Maintz 2.75	Decorate living r Tommy Maintz 2.75
Fix lights Tommy Maintz 0.75	Fix lights Tommy Maintz 0.75	Fix lights Tommy Maintz 0.75
Reattach screen Tommy Maintz 2	Reattach screen Tommy Maintz 2	Reattach screen Tommy Maintz 2
►Project: Testing Core Team 2	►Project: Testing Core Team 2	►Project: Testing Core Team 2
Expand Collapse 2 3	Expand Collapse 2 3	Expand Collapse 2 3
TaskList Project CENSUS Override	TaskList Project CENSUS Override	TaskList Project CENSUS Override

SAMPLE: US Census – Phone Portrait

TouchTreeGrid Demo	TouchTreeGrid Demo
Population Males Females	Back Maine
▼Maine 1,274,923 620,309 654,614	Number Percent
►Androscoggin Cou 103,793 50,385 53,408	Total Population 1,274,923 100.0%
►Aroostook County 73,938 36,095 37,843	Male 620,309 48.7%
►Cumberland Count 265,612 128,589 137,023	Female 654,614 51.3%
►Franklin County 29,467 14,228 15,239	Under 5 years 70,726 5.5%
►Hancock County 51,791 25,324 26,467	5 to 9 83,022 6.5%
►Kennebec County 117,114 56,746 60,368	10 to 14 92,252 7.2%
Expand Collapse 2	15 to 19 89,485 7.0%
TaskList Project CENSUS Override	20 to 24 69,656 5.5%
	TaskList Project CENSUS Override

TouchTreeGrid

SAMPLE: US Census – Tablet Landscape

TouchTreeGrid Demo									
	Population	Males	Females	Median	18+	21+	62+	65+	
▼ Maine	1,274,923	620,309	654,614	39	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	41	57,218	54,367	14,772	12,551	→
▶ Cumberland County	265,612	128,589	137,023	38	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	→
▶ Hancock County	51,791	25,324	26,467	41	40,248	38,366	9,656	8,285	→
▶ Kennebec County	117,114	56,746	60,368	39	89,187	84,262	19,574	16,605	→
Expand Collapse 2									
TaskList Project CENSUS Override									

TouchTreeGrid

SAMPLE: US Census – Tablet Portrait

(# of columns auto-updated as orientation is changed)

TouchTreeGrid Demo						
	Population	Males	Females	Median	18+	21+
▼ Maine	1,274,923	620,309	654,614	39	973,685	924,108
▶ Androscoggin Coun	103,793	50,385	53,408	37	78,957	74,488
▶ Aroostook County	73,938	36,095	37,843	41	57,218	54,367
▶ Cumberland County	265,612	128,589	137,023	38	203,650	193,206
▶ Franklin County	29,467	14,228	15,239	38	22,538	20,827
▶ Hancock County	51,791	25,324	26,467	41	40,248	38,366
▶ Kennebec County	117,114	56,746	60,368	39	89,187	84,262
▶ Knox County	39,618	19,327	20,291	41	30,759	29,653
▶ Lincoln County	33,616	16,389	17,227	43	25,989	25,092
Expand Collapse 2						
TaskList Project CENSUS Override						

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 TreeGridView component automatically creates:
 1. Optional column header row
 2. Grid displaying Detail and collapsible Category rows
 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.

- Option to assign unique colors to Category rows by expansion depth:

```
categDepthColorsArr: ['#808127', '#949569', '#C5C678']
```

- 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
 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
 6. headerStyle - Additional CSS styling commands for column header row
- 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/TouchTreeGrid_Advanced/app.html?deviceType=Phone
```

```
http://localhost/TouchTreeGrid/TouchTreeGrid_Advanced/app.html?deviceType=Tablet
```

For larger applications it is **HIGHLY** 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

TouchTreeGrid

footprint size that is downloaded on the device and allows for easy configuration changes without a code change. Note that we do the same thing when implementing Mitchell Simoen's Ext.ux.touch.grid.

- 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)

Advanced Features

- 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.
- 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)
- Custom column render functions are supported a little differently than in Mitchell Simoen's Ext.ux.touch.grid.
 - Instead of defining the function within the Columns Array for each column, custom render functions are defined in 'renderers' object within each linked instance of TouchTreeGrid²:

```
{ 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)

² 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

- You could add all common functions used across your application directly in TouchTreeGrid 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.
- 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)'
],
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 = ','

TouchTreeGrid

- 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 onFirstExampleLeafItemTap() and onFirstExampleNodeItemTap()
- PullRefresh implemented as follows:
 - Define ‘plugins’ object in linked TouchTreeGrid (also refer to Example2)

```
itemId: myExample,
listPlugins: {
  xtype: 'component',
  refreshFn: function(plugin)
  {this.up('touchtreegrid').fireEvent('pullrefresh')};,
  type: 'pullrefresh'
}
```
 - Within controller:

```
refs: {
  myExample: '#myExample',

  control: {
    "touchtreegrid#myExample": {
      pullrefresh: 'onMyExampleListPullrefresh'
    },
  }
}
```
 - onMyExampleListPullrefresh() function
 - update myData object with refreshed tree data
 - myGrid = this.getMyExample();
 - myStore = myGrid.getStore();
 - Reload store via: myStore.setData(myData);
Note: setRoot() not working =>
<http://www.sencha.com/forum/showthread.php?242257>
 - Refresh grid via: myGrid.doRefreshList();
- 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.
 - Within linked TouchTreeGrid instance

TouchTreeGrid

- customExpCollapseEvent : 'myExampleExpCollapseEvent'
- itemId : 'myExample'
- Within Controller:
 - ```
refs: {
 myExample: '#myExample', etc...}

control : {
 "container#myExample": {
 myExampleExpCollapseEvent: 'onMyExampleExpCollapse'
 }, etc...}
```
  - 'onMyExampleExpCollapse' function
    - Rebuild tree specifying 'expanded' attribute
    - Refer to documentation above on pullrefresh on how to load store and refresh grid
- 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 Sencha default: 'x-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.
- 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 you can specify the data type as Object to edit formatted text
  - Refer to "Manual" example.

# TouchTreeGrid

---

- 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.
- If you want to change the "Recycle" icon when rotating phones to landscape you can specify your own url in the linked instance of TouchTreeGrid via,  
landscapelcon = './resources/images/myIcon.png'

|                       | 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

## Summary of Provided Examples

(refer below for detailed configuration documentation)

1. Task List Grid
  - a. Basic accordion example
  - b. Fixed data stored directly in Root of store: TouchTreeGrid.store.Task
2. Project Status Grid (#1)
  - a. Initially expands nodes as individually defined in JSON tree store
  - b. color-specific category rows with matching colors on collapse-to buttons
  - c. overridden color scheme for expand/collapse button and toolbar (treegriddemo.css)
  - d. custom iPhone-style disclose icon (treegriddemo.css).
    - i. `.x-touchtreegrid-list-example2 .x-list-disclosure{...}`
    - ii. `.x-touchtreegrid-list-example2 .touchtreegrid-disclose-spacer {width: 97%;}`  
This reduces default header spacer of 95% for thinner disclose icon
    - iii. 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 itemtap hold event which triggers when user presses

# TouchTreeGrid

---

anywhere on a disclosure row for more than one second as an alternative (refer to `onExample2ListItemTaphold()` function within controller.

- e. On Disclose slides in detail panel showing all data columns for selected row
- f. Disclose only implemented for detail (leaf) rows for this particular example.
- g. Implements pull-refresh which sends “pullrefresh” event to controller to re-load data

## 3. Project Status Grid (#2)

- a. Variation of Project #1 where `itemHeight=32` and `variableHeights=false` to tighten space between rows. Also refer to CSS customization in `treegriddemo.css` to hide horizontal lines, and to center iPhone-style disclosure icon and detail row text in thinner row :

```
.x-touchtreegrid-list-example2B .x-list-normal .x-list-item .x-dock-horizontal {
 border: none; }
.x-touchtreegrid-list-example2B .x-list-disclosure{
 margin-top: .1em;
}
.x-touchtreegrid-list-example2B .touchtreegrid-list-content{
 padding:.6em 0 0 0;}
```

- b. Adds discloser icon to Category rows

## 4. Project Status Grid (#3)

- a. Variation of Project #2 where row shading is removed. Refer to CSS customization in `treegriddemo.css`:

```
.x-touchtreegrid-list-example2C .touchtreegrid-list-categ {
 border-top: none;
 border-bottom: none;
 -webkit-box-shadow: none;
}
.x-touchtreegrid-list-example2C .x-list-normal .x-list-item .x-dock-horizontal {
 border: none;
}
.x-touchtreegrid-list-example2C .touchtreegrid-list-content{
 padding:.6em 0 0 0;
}
```

- b. Disclosure icons removed from all rows

## 5. Census Grid

- a. ~600 total row grid example with 28 data columns per row (no performance issues)
- b. Uses TouchTreeGrid default color schemes for category rows and collapse-to buttons
- c. Loads data within controller via JSON tree store
- d. Initially collapses to depth 2.
- e. 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 .
- f. Grid and detail window utilize `formatNumbers()` function included with TouchTreeGrid to format numbers with commas.

# TouchTreeGrid

- g. Different COLUMN configurations for Phone-Portrait, Phone-Landscape, Tablet-Portrait, Tablet-Landscape. Test using webkit-enabled browser by manually resizing window:
  - i. [http://localhost/TouchTreeGrid/TouchTreeGrid\\_Advanced/app.html?deviceType=Phone](http://localhost/TouchTreeGrid/TouchTreeGrid_Advanced/app.html?deviceType=Phone)
  - ii. [http://localhost/TouchTreeGrid/TouchTreeGrid\\_Advanced/app.html?deviceType=Tablet](http://localhost/TouchTreeGrid/TouchTreeGrid_Advanced/app.html?deviceType=Tablet)
- h. Also note custom implementation in Phone-Landscape mode where Titlebar and TabBars are hidden allowing more display (rotate to Portrait to restore):

|                       | 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.

## 6. Manual Grid

- a. Same as Basic example except HTML strings for category and detail rows are provided in the configuration. COLUMN array is not used.
- b. This provides more flexibility in use of this component to support custom needs

## Notes on Implementation

1. Setup if using Sencha Architect
  - a. Import TouchTreeGrid.xdc into your toolbox (via right-click).  
Created using Architect v.2.2.0
  - 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
    - ii. ./resources/images/Recycle.png
  - 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

# TouchTreeGrid

---

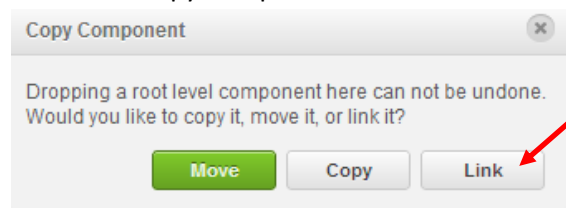
- ii. Definitions in subsequent stack will override prior definitions. When project is saved you will notice in APP.HTML that treegriddemo.css is loaded after TouchTreeGrid.css

## 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-2.1.1/sencha-touch-all-debug.js">
</script>
 <link rel="stylesheet" href="http://cdn.sencha.com/touch/sencha-touch-2.1.1/
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>
</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”

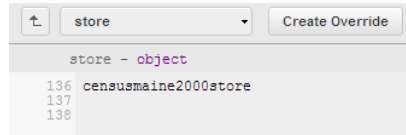


- i. Click on the added “linked” component under your container and start updating your configs. Minimum required definitions:
  - i. **xtype** : ‘touchtreegrid’ – automatically created when you link the view
  - ii. **itemId** (unique identifier for this grid)

# TouchTreeGrid

---

- iii. **columns** - array defining columns for grid (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.

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 !
2. 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 above.
  - d. I’m assuming you know what to do from here if not using Architect...
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/>

# TouchTreeGrid

---

- 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 `TouchTreeGridController` 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/TouchTreeGrid\\_Advanced/app.html](http://192.168.1.??/TouchTreeGrid/TouchTreeGrid_Advanced/app.html)

## Configuration: Project Status Grid #1

```
items: [
{
 xtype: 'touchtreegrid', <= "LINKED" TOUCHTREEGRID COMPONENT
 listPlugins: { <= Example of how to implement PullRefresh
 xtype: 'component',
 refreshFn: function(plugin) {this.up('touchtreegrid').fireEvent('pullrefresh');},
 type: 'pullrefresh'
 },
 cateDepthColors: true, <= if false standard Sencha colors would be used
 cateDepthColorsArr: [
 '#808127', <= depth 1 category row color
 '#949569', <= depth 2 category row color
 '#C5C678' <= depth 3 category row color, etc.. (if there are more levels white is used)
],
 store: 'Task2Store',
 onItemDisclosure: true, <= default is false
 columns: [
 {
 header: 'Task',
 dataIndex: 'task',
 width: '35%',
```

# TouchTreeGrid

---

```
 style: 'text-align: left;',
 categStyle: 'font-weight: bold; text-align: left; color: #692047;',
 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;'
 }
 /*
 {
 header: 'Done?',
 dataIndex: 'done',
 width: '15%',
 style: 'text-align: right; font-size: .6em;',
 categStyle: 'text-align: right; font-size: .6em;',
 headerStyle: 'text-align: right; color: #ccc;'
 }
 */
],
disclosureProperty: 'leaf', <= DISCLOSE is default to disclose category+leafs
colNumberToTruncateForIndents: 1, <= could specify different column to truncate
applyDefaultCollapseLevel: false, <= if false then initial "expanded" attribute defined via
 treestore data request
cls: [
 'x-touchtreegrid-list', <= default for CSS styling
 'x-touchtreegrid-list-example2' <= optional when overriding default CSS stylings
 (see treegriddemo.css for example overrides)
],
listItemId: 'example2list', <= allows itemId assignment for generated list (optional)
itemId: 'example2' <= user assigned unique itemId referenced by controller
}]
```

## Configuration: Project Status Grid #2 (differences from #1)

```
{
 xtype: 'touchtreegrid',

 SEE CODE...

 columns: [
 {
 header: 'Task',
 dataIndex: 'task',
 width: '35%',
 style: 'text-align: left;',
 categStyle: 'font-weight: bold; text-align: left; color: #f4f4f4;',
 headerStyle: 'text-align: left; color: #ccc;'
 },
 SEE CODE...
],
 variableHeights: false,
 itemHeight: 32,
 listItemId: 'example2Blist',

 SEE CODE...
}
```



# TouchTreeGrid

---

```
cls: [
 'x-touchtreegrid-list',
 'x-touchtreegrid-list-example2',
 'x-touchtreegrid-list-example2B'
],
itemId: 'example2B'
}
```

## Configuration: Project Status Grid #3 (differences from #2)

```
{
 xtype: 'touchtreegrid',
 onItemDisclosure: false,

 SEE CODE...

 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;'
 },
 SEE CODE...
],

 listItemId: 'example2Clist',

 SEE CODE...

 cls: [
 'x-touchtreegrid-list',
 'x-touchtreegrid-list-example2',
 'x-touchtreegrid-list-example2C'
],
 itemId: 'example2C'
}
```

## Configuration: CENSUS Grid

```
items: [{
 xtype: 'touchtreegrid', <= "LINKED" TOUCHTREEGRID COMPONENT
 categIndentPct: 2, <= Indent 2% of screen width per depth
 colNumberToTruncateForIndents: 1,
 store: 'censusmaine2000store',
 onItemDisclosure: true,
 listItemId: 'censusmainelist',
 categDepthColors: true, <= categDepthColorsArr not provide so default colors will be
 applied ('#a6a6a6', '#dddddd', 'white')

 columns : {}, <= array definitions within controller and applied based on device/orientation

 renderers: { <= user-defined functions to format data display stored in "renderers" object
 renderer_displayIn1000s: function(value)
 {return this.formatNumbers(Math.round(Number(value)/1000), 0);}
 , renderer_anotherone: function(value) {...}
 ** we could include additional functions separated by commas as shown
 },
}
```

### DISCUSSION:

```
>formatNumbers() function is provided with TOUCHTREEGRID component
 (refer to "Advanced Features")
>renderer_displayIn1000s() is user-defined, and is not actually used in this example.
 We could for example display numbers in 1000's and format result with comma's
 To display MYCOL ="1234567" as "1,235" the column array would include this attribute:
 renderer: 'this.renderer_displayIn1000s(values.MYCOL)'
```

# TouchTreeGrid

---

```
defaultCollapseLevel: 2, <= auto-expand to depth 2 upon launch or refresh
cls: [
 'x-touchtreegrid-list',
 'x-touchtreegrid-list-censusmaine' <= optional when overriding default CSS stylings
 (not actually used in provided examples)
],
itemId: 'censusmaine'
}]
```

## Configuration: Manual Grid

```
items: [{
 xtype: 'touchtreegrid',
 listItemId: 'overrideexamplelist',
 includeFooter: false, <= do not include expand/collapse footer toolbar
 includeFooterLevels: false,
 includeHeader: false, <= do not include titlebar containing column headers
 applyDefaultCollapseLevel: true, <=apply 'defaultCollapseLevel'
 defaultCollapseLevel: 1, <= expand to depth 1
 (0 is fully collapsed, same as 1 if no single root)

 store: 'TaskStore',

 categorItemTplOverride: <= User specific HTML for category rows follows...

 '<div>
<tpl if="this.isExpanded(values)">

 <span class="x-button-icon arrow_down x-icon-mask"
 style="width: .75em; height: .75em; margin-right:0.4em;">

 {text}

<tpl else>

 <span class="x-button-icon arrow_right x-icon-mask"
 style="width: .75em; height: .75em; margin-right:0.4em;">

 {text}

</tpl>
</div>',

 contentItemTplOverride: <= User specific HTML for detail rows follows...

 '<div style="background-color:#fff; border-bottom: 1px solid #dedede;">{text}</div>',

 cls: [
 'x-touchtreegrid-list',
 'x-touchtreegrid-list-override' <= see treegriddemo.css for example CSS override
 Of .touchtreegrid-list-categ styling
],
 itemId: 'overrideexample'
}]
```

## Additional Functionality included in Advanced Example:

1. How JSON data is loaded into TreeStore
  - a. Tasklist store is loaded upon launch as fixed data is defined in root of 'TouchTreeGrid.store.Task'

# TouchTreeGrid

---

- b. 'TouchTreeGrid.store.Task2' and 'TouchTreeGrid.store.CensusMaine2000' are loaded when user presses respective tab (and only if no data already)
- c. **onMainTabpanelActiveItemChange()** function within controller listens for tab change and calls either **loadExample2Store(gridcont)** or **loadCensusMaine2000Store()**
- d. Both functions are similar and call common **loadStore()** function:

```
loadCensusMaine2000Store: function() {
 var me = this;
 var gridcont = me.getCensusmaine();
 var gridurl = 'data/censusmaine2000TREE.json';

 me.loadStore(me, gridcont, gridurl, 'Loading Census...');
}
```

- e. **loadStore()** function makes AJAX call to load JSON file and then calls **postLoadProcess()** function with Callback success function (since data is loaded asynchronously)
  - i. Note: similar function can be used to load data from server

```
loadStore: function(me, gridcont, gridurl, loadmask) {
 // Load data from JSON file within Controller since doesn't seem to work from
 // within Store itself.
 // NOTE: autoload=true -and- dummy root initialization required in Store=>
 // root: {children: []} <= important !

 if (loadmask) {
 Ext.Viewport.setMasked({
 xtype: 'loadmask',
 message: loadmask
 });
 }

 // Change this to reload function with Pull Refresh
 var myRequest = Ext.Ajax.request({
 url: gridurl,
 method: 'GET',
 timeout: 10000,
 cache: false,
 dataType: 'json',
 reader: {
 type: 'json'
 },

 success: function(response) {
 var griddata = Ext.JSON.decode(response.responseText);

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

 gridstore.removeAll();
 var gridloaded = gridstore.setData(griddata);
 // setRoot() not working =>
 // http://www.sencha.com/forum/showthread.php?242257

 if (loadmask) {Ext.Viewport.setMasked(false);}

 me.postLoadProcess(gridListItemId, gridcont);
 },
 },
```

# TouchTreeGrid

---

```
failure: function(response, opts) {
 if (loadmask) {Ext.Viewport.setMasked(false);}
 Ext.Msg.alert('Data not loaded: '+gridurl);
}
});
}
```

- f. **PostLoadProcess()** function refreshes grid and applies default collapse-to level
  - i. It also calls **loadColumnsCensusMaine()** function which applies appropriate column configuration for Census example based on Phone vs. Tablet and Portrait vs. Landscape orientations. When columns config is updated for particular list, **applyColumns()** function within TouchGridList component is automatically called to redefine the HTML for the Category and Detail rows in the list.
  - ii. **gridcont.doRefreshList()** function is finally called to
  - iii. redefine the Header columns
  - iv. update the itemTpl based on updated HTML above (which refreshes the list)
  - v. applies collapse-to level to the treestore
  - vi. updates footer with auto-generated collapse-to buttons based on depth level of refreshed data

```
postLoadProcess: function(gridListItemId, gridcont) {
 var refreshed;

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

 this.loadColumnsCensusMaine(); // also refreshes list
 }
 else if ((gridListItemId === 'example2list') ||
 (gridListItemId === 'example2Blist') ||
 (gridListItemId === 'example2Clist')) {
 this.getMain().down('#example2list').up('touchtreegrid').doRefreshList();
 this.getMain().down('#example2Blist').up('touchtreegrid').doRefreshList();
 this.getMain().down('#example2Clist').up('touchtreegrid').doRefreshList();
 }
 else {
 refreshed = gridcont.doRefreshList();
 }
},
```

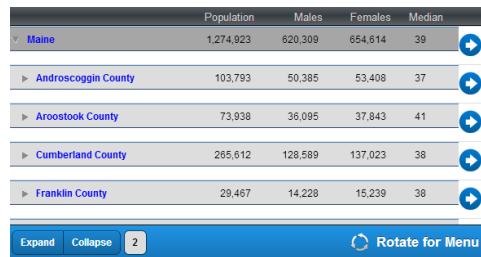
## 2. Example for handling changes in device orientation

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

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

# TouchTreeGrid

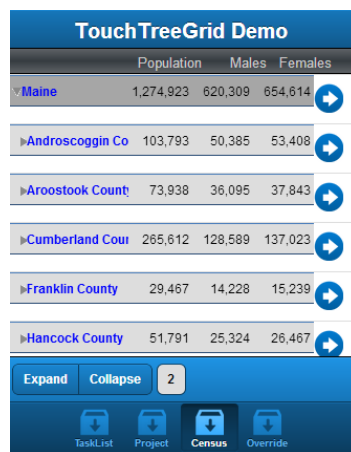
- b. `onOrientationChange()` function calls `loadColumnsCensusMaine()` function which loads different column configurations for phone (portrait vs. landscape) and tablet (portrait vs. landscape). The number of grid columns dynamically change simply by rotating the device.
- c. `onOrientationChange()` function calls `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.



	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

- 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.
- 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	➔
▶ Aroostook Count	73,938	36,095	37,843	➔
▶ Cumberland Cour	265,612	128,589	137,023	➔
▶ Franklin County	29,467	14,228	15,239	➔
▶ Hancock County	51,791	25,324	26,467	➔

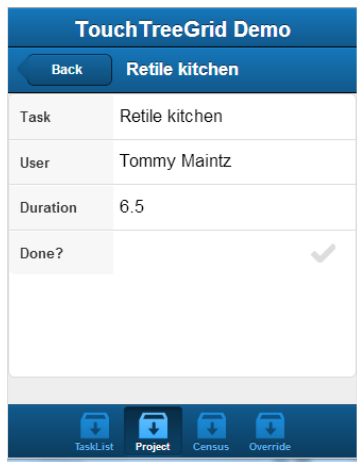
Expand Collapse 2

TaskList Project Census Override

# TouchTreeGrid

---

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



- 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. TouchTreeGridController defines references to parent grid panel (main) and griddetailpanel:

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

- c. TouchTreeGridController also 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) {
```

# TouchTreeGrid

---

```
// 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
 }
}
```

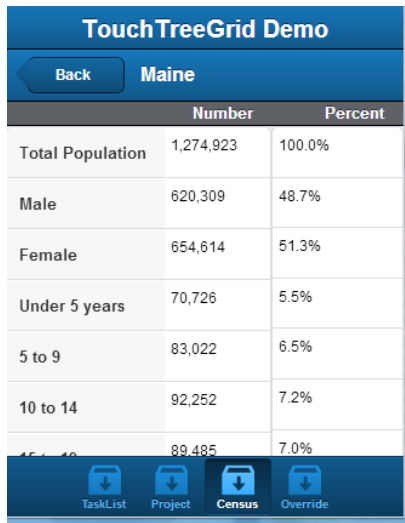
- f. onGridDetailBackButtonTap() function handles Back button press to navigate from griddetailpanel back to the grid

```
// 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);
}
```

# TouchTreeGrid

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



The screenshot shows a mobile application interface titled "TouchTreeGrid Demo". At the top, there is a blue header bar with a "Back" button and the word "Maine". Below this is a table with two columns: "Number" and "Percent". The table contains the following data:

	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%

At the bottom of the screen, there is a blue bar with four icons and labels: "TaskList", "Project", "Census", and "Override".

- Architect is used to quickly create the layouts with dummy fieldsets for Number and for Percent columns.
- Censusdetailpanel is then instantiated similar to griddetailpanel
- TouchTreeGridController 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...
}
```

- Fieldsets for Number and Percent are manually updated within onCensusMaineListDisclose() function.
  - Back button functionality handled within onCensusDetailBackButtonTap()
5. **onMainTabpanelActiveItemChange()** within controller contains logic to load Project and Census data only when user clicks on those tabs to speed up initial launch time.
- Data only loaded if Store is empty
6. **onTitlebarGridhelp()** function with controller traps touches on TitleBar and launches modal information window custom for each Grid



# TouchTreeGrid

---

- 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. xx

# 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 rows
- categStyle - Additional CSS styling commands for category rows
- headerStyle - Additional CSS styling commands for column header row
- renderers - refer to "Advanced Features" sections for discussion on use

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 TouchTreeGridController in Advanced example for additional number of column variations for Phone (portrait vs. landscape) and Tablet (portrait vs. landscape).

# 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;  
}
- 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;  
}
- 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;  
    margin: 0;  
    font-size: .8em !important;  
}
- Default layout for Header row  
**.x-touchtreegrid-list .touchtreegrid-header {**  
    width: 100%;

# TouchTreeGrid

---

```
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 Detail row individual “cells”

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

- 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;
}
```

- 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 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)

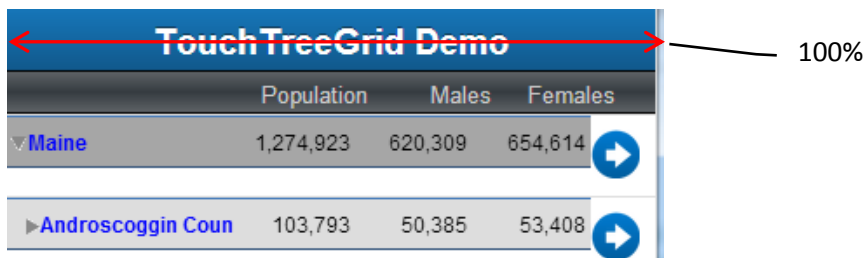
```
.x-touchtreegrid-list-example2 .x-list-disclosure{
 width: 12px;
 height: 1.5em;
 margin: .4em 0 0 .5em;
 -webkit-mask: none;
 -webkit-mask-box-image: (see file for embedded image)
```

# TouchTreeGrid

}

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 itemtaphold event as secondary means to react to user's attempt to navigate.

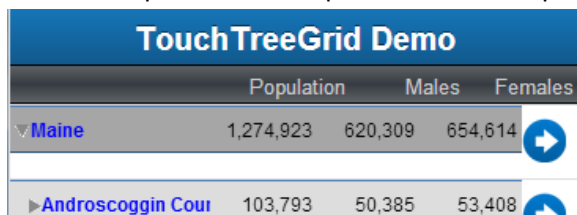
- Define spacer width when using disclose icons such that header columns line up with category/detail columns:



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

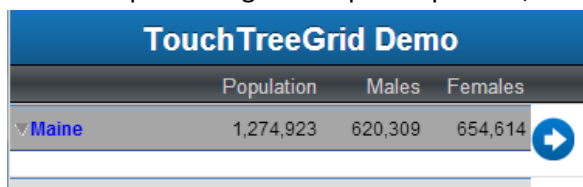
```
.touchtreegrid-disclose-spacer {
 max-width: 95%;
}
```

100% example reveals the problem with no spacer:

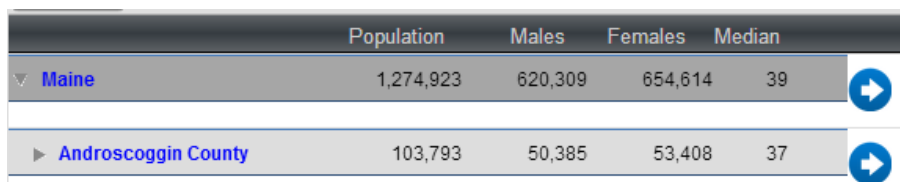


	Population	Males	Females
▼ Maine	1,274,923	620,309	654,614
▶ Androscoggin Cour	103,793	50,385	53,408

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



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



	Population	Males	Females	Median
▼ Maine	1,274,923	620,309	654,614	39
▶ Androscoggin County	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

# TouchTreeGrid

---

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 prior to updating linked instances as "Inline Class" as you could observe issues running your code.
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')
9. Drag the newly imported component ontop of "View" within Project Inspector.

## 10. Save the Project !

11. 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.