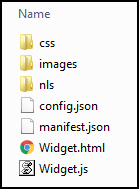
# **Custom Widget #1 – The Terrible Widget**

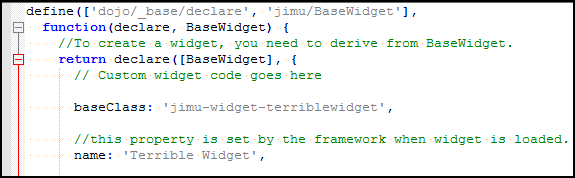
1. Navigate to **\\client\stemapp\widgets\samplewidgets**
2. Make a copy of **CustomWidgetTemplate** and call it **TerribleWidget.** This template will contain the more commonly used files in the widget file structure.



1. Make sure you a have a preferred text editor installed on your machine. There are many to choose from. If you are unsure which to use start out use [NotePad ++ for Windows](https://notepad-plus-plus.org), or [Atom](https://atom.io) for Mac.
2. The folder structure should look like this. Let’s open up the **Widget.js** file.



1. Un-comment the following:
   1. **baseClass:** change **jimu-widget-customwidget** to **jimu-widget-terriblewidget.** Note that JavaScript is case-sensitive. Also make sure a comma exists at the end of this line.
   2. **name:** un-comment this line and change to ‘Terrible Widget’



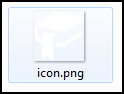
1. Save the **Widget.js** file and close it.
2. Open **Widget.html** in your text editor. Copy the following HTML syntax (Change the name to your name or change the whole phrase if you so choose):

***<div>***

***<div>Your name’s Terrible Widget</div>***

***</div>***

1. Save the **Widget.html** and close it.
2. Navigate to the **images** folder. It will contain an image **icon.png** that is of a bookmark with width and height of 50x50. Replace this icon with one of your own. Have fun with it find something from free online sources like <http://www.iconarchive.com/> , [www.findicons.com](http://www.findicons.com), etc. Use something from your phone if you want. For best display results make the height and weight the same length preferably 50x50. Example below:

1. Add the widget information to the app config JSON file. This is done by navigating to **\\client\stemapp\sample -configs\config-demo.json.** Find widgetPool -> widgets and add the following JSON to the end if it. Make sure all required commas exist to insure proper JSON format.

***{***

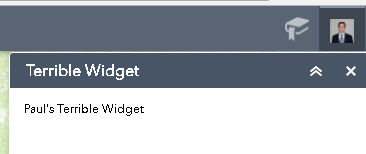
***"label": "Terrible Widget",***

***"uri": "widgets/samplewidgets/TerribleWidget/Widget"***

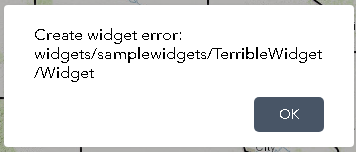
***}***



1. At this point let’s test the widget to see your progress. Test the widget via this path **http://[your host name:3344]/webappviewer/?config=sample-configs/config-demo.json**
2. In the viewer, you should see a test button with your icon. Click on that button to verify your widget appears with the correct HTML.



1. What if you get an error like this???



In this case, there may be an error in your code syntax. View your browsers developer tools by clicking **F12**. Read the errors in the console to debug the error.

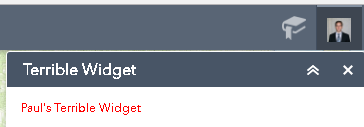
1. Let’s change the color of the text with CSS. Navigate to **\\client\stemapp\widgets\samplewidgets\TerribleWidget\css\style.css**
2. Add the following text to change the text to red.

***.jimu-widget-terriblewidget div:first-child{***

***color: red;***

***}***

1. The text should turn to red:



1. Next let’s make the text configurable. **\\client\stemapp\widgets\samplewidgets\TerribleWidget\config.json**
2. Add the following text to make to tool configurable.

**{**

**"configText": "Your Name Awesome Widget!"**

**}**

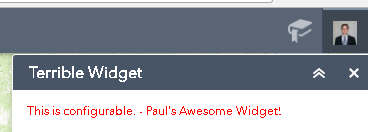
1. Open **Widget.html** file again and modify as seen here.

***<div>***

***<div>This is configurable. - ${config.configText}</div>***

***</div>***

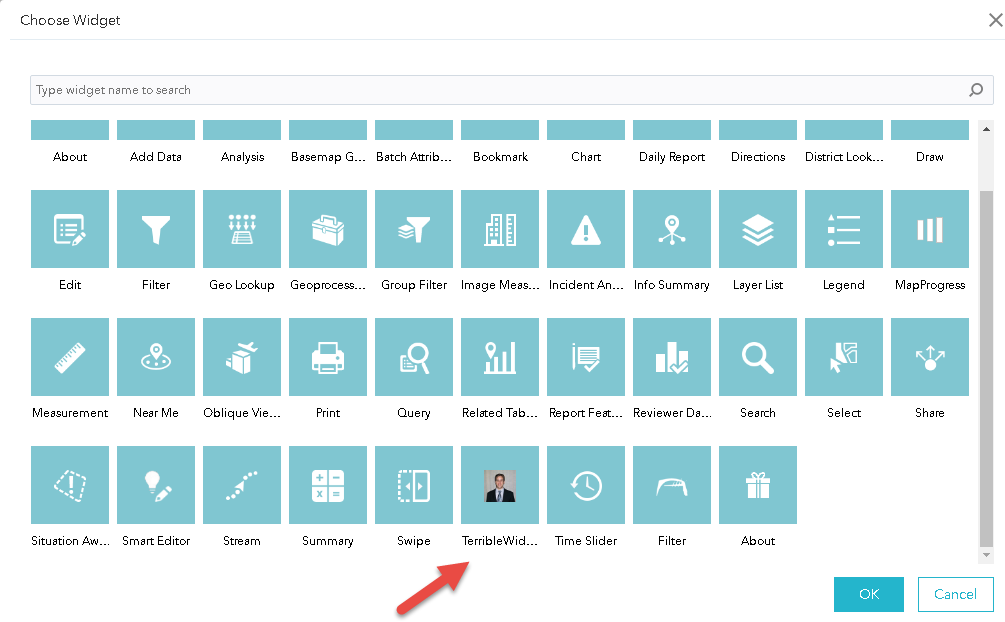
1. Test the widget again in the viewer.



1. Edit the **Manifest.json** file. The name should match the name of your widget folder. You can change the **author** and **description**. Also, change the properties settings as seen below.



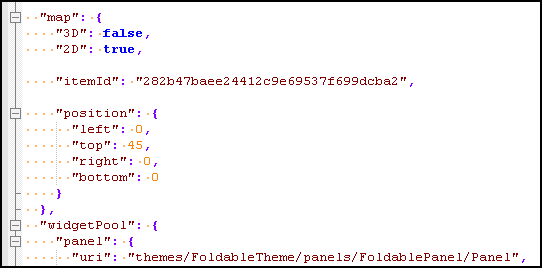
1. Finally, to deploy this widget for use with an app place the entire folder in [\\client\stemapp\widgets](file:///\\client\stemapp\widgets)
2. Before the changes take effect you will need to restart your node session. To close your node session, close the startup.bat file and start a new node session by opening the startup.bat file.
3. Once you’ve started a new node session, add a new widget and the new widget should appear in the list of available widgets.



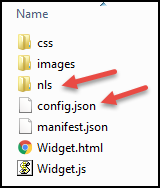
1. For more information, check out [Esri’s ‘Create a custom in-panel widget’ tutorial](https://developers.arcgis.com/web-appbuilder/sample-code/create-custom-in-panel-widget.htm).

# **Custom Widget #2 – Zoom to Minnesota County**

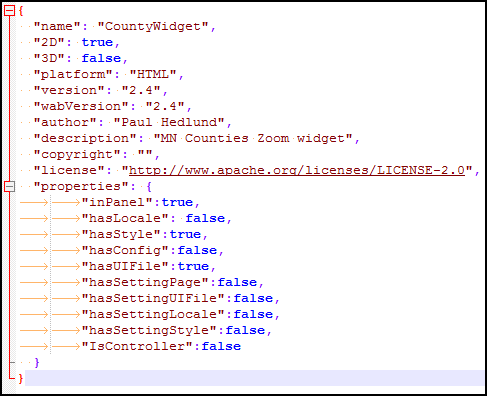
1. Start by going through steps #1 to #13 in **The Terrible Widget** except call it the widget **CountyWidget**. Select a different Icon as well.
2. Add AGO map to the sample config. This is done by navigating to **\\client\stemapp\sample -configs.** Go to **map -> itemId** and change the Web Map ID to ***“bbd3477a3f0944a48881f681a8530be2”***



1. This widget will not require some template files. Therefore, remove the **config.json** file and **nls** folder.



1. Edit the **Manifest.json** file to not include some properties. Have the parameters similar to what is seen below.



1. This widget will require some HTML syntax containing some DOJO/DIJIT widgets. First, remove any existing text in the **widget.html** file. Then, copy the text below.

***<div>***

***<span class="sectionTitles">Minnesota County List</span>***

***<div id="filterBlock">***

***<table width=”100%”>***

***<tr>***

***<td class=tableAlignLeft><select dojotype="dijit/form/FilteringSelect" data-dojo-props="id:'MNcountylist', autoComplete:false, value:'', placeHolder: 'Select county ...'" required="false" />***

***</tr>***

***</table>***

***</div>***

***<div id="buttonBlock" style=”margin-top:10px;”>***

***<table width=”100%”>***

***<tr>***

***<td style="text-align:center;"><div class="jimu-btn" data-dojo-attach-point="btnVote" data-dojo-attach-event="click:\_ZoomCounty">Zoom to County</div></td>***

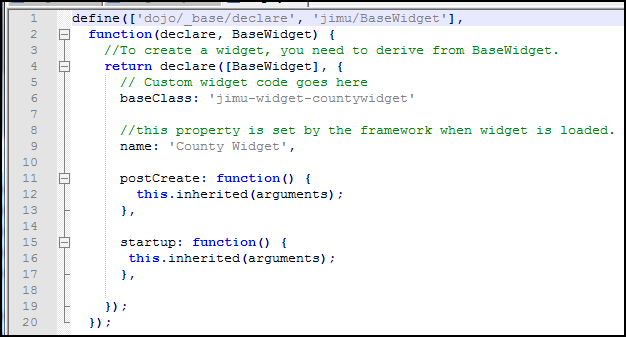
***</tr>***

***</table>***

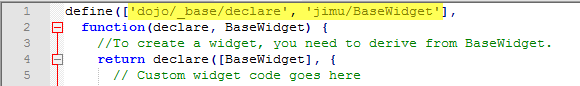
***</div>***

***</div>***

1. Now let’s spend some time adding code to the **widget.js** file. This file will contain the most logic of any file for the widget. First un-comment the **postCreate** and **startup** event functions. Remove all other commented functions including **onOpen**, **onClose**, etc. These events will not be used for this widget.



1. Part of the ArcGIS JavaScript API framework is something called AMD (Asynchronous Module Definition). Part of the AMD process is adding ESRI libraries ins declarative process. Add this code to the top of the **widget.js** file under define.



***'dijit/\_WidgetsInTemplateMixin',***

***'esri/graphicsUtils',***

***'esri/graphic',***

***'dojo/store/Memory',***

***'esri/tasks/query',***

***'esri/tasks/QueryTask',***

***'esri/symbols/SimpleFillSymbol',***

***'esri/symbols/SimpleLineSymbol',***

***'dojo/\_base/Color',***

***'dijit/registry',***

***'dijit/form/FilteringSelect',***

***'jimu/dijit/Message',***

***'dojo/on'***

1. Then in the main function add the following.

***function(declare,BaseWidget,\_WidgetsInTemplateMixin,graphicsUtils,Graphic,Memory,Query,QueryTask,SimpleFillSymbol,SimpleLineSymbol,ColorDojo,registry,FilteringSelect,Message,on)***

1. Modify the declare line from **return declare([BaseWidget]** to the text here below.

***return declare([BaseWidget, \_WidgetsInTemplateMixin]***

1. The code at the top of the page should look like this. Verify all commas and semicolons in the correct place.



1. There will be two more custom functions to add to the **widget.js** file. The first on is called **\_GetProjectList**. This will add all counties into a dropdown FilteringSelect dojo widget. Add the syntax below for this code. It can be added below the **startup** function.

***\_GetProjectList: function () {***

***//Query the counties***

***var item = "NAME";***

***var queryTask = new QueryTask***

***("https://services2.arcgis.com/rwqARsO7kmPlpMQS/ArcGIS/rest/services/MNcounties/FeatureServer/0"); //service***

***var query = new Query();***

***query.where = "1=1"; //query to get all counties***

***query.returnGeometry = false;***

***query.outFields = [ item ];***

***query.orderByFields = [ item ];***

***queryTask.execute(query, getStoreQueryValues);***

***//Loop through and store all MN counties in dropdown box***

***function getStoreQueryValues(results){***

***var data = {***

***identifier: 'id', //This field needs to have unique values***

***label: 'name', //Name field for display.***

***items: []***

***};***

***var storeItem = new Memory({***

***data: data***

***});***

***var itemStr = "";***

***for (var i = 0; i < results.features.length; i++) {***

***if(itemStr.indexOf(results.features[i].attributes[item]) == -1){***

***itemStr += "{id: '" + (i + 1) + "',name:'" + results.features[i].attributes[item] + "'},";***

***storeItem.put({'id': (i + 1),'name': results.features[i].attributes[item]});***

***}***

***}***

***if (registry.byId("MNcountylist")) {***

***registry.byId("MNcountylist").reset();***

***registry.byId("MNcountylist").store = storeItem;***

***}***

***}***

***},***

1. The second function is called **\_ZoomCounty** and is the code that will zoom the map to the county and highlight it.

***//Function to zoom to county***

***\_ZoomCounty: function () {***

***//Query the selected county***

***var queryTask = new QueryTask("https://services2.arcgis.com/rwqARsO7kmPlpMQS/ArcGIS/rest/services/MNcounties/FeatureServer/0"); //service***

***var query = new Query();***

***query.where = "NAME = '" + registry.byId("MNcountylist").displayedValue + "'"; query.returnGeometry = true;***

***query.outFields = ["NAME"];***

***map = this.map;***

***query.outSpatialReference = {***

***wkid: 102100***

***};***

***on(queryTask, "complete", function(evt){***

***zoomToResults(evt, map);***

***});***

***queryTask.execute(query, showSpatialResults);***

***//Create graphic for zoom results***

***function showSpatialResults(results){***

***ClearSearchGraphics(map);***

***var fieldsSelectionSymbol = new SimpleFillSymbol(SimpleFillSymbol.STYLE\_SOLID,new SimpleLineSymbol(SimpleLineSymbol.STYLE\_DASHDOT, new ColorDojo([255,255,0]), 6),new ColorDojo([255,255,0,0.35]));***

***map.graphics.add(new Graphic(results.features[0].geometry, fieldsSelectionSymbol));***

***}***

***//zoom to the location of the result***

***function zoomToResults(evt,map){***

***var featureSet = evt || {};***

***var features = featureSet.featureSet.features || [];***

***if (features.length > 0) {***

***var extent = graphicsUtils.graphicsExtent(features);***

***if (extent) {***

***map.setExtent(extent.expand(1.5), true);***

***}***

***}***

***else {***

***new Message({message: 'The county was not found on the map!',titleLabel: "County Widget Error!",autoHeight: true});***

***}***

***}***

***//clear previous search yellow polygon***

***function ClearSearchGraphics(map){***

***if (map.graphics) {***

***map.graphics.clear();***

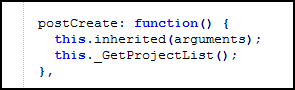
***}***

***}***

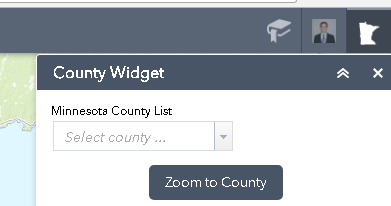
***}***

1. Then, in the **postCreate** event add the function call.

***this.\_GetProjectList();***



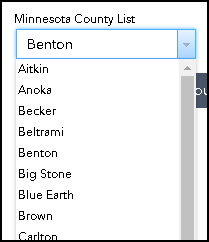
1. Now let’s test the display of the widget. Start the test URL again at **“http://[your host name:3344]/webappviewer/?config=sample-configs/config-demo.json”.**  Find the County Widget and test it.



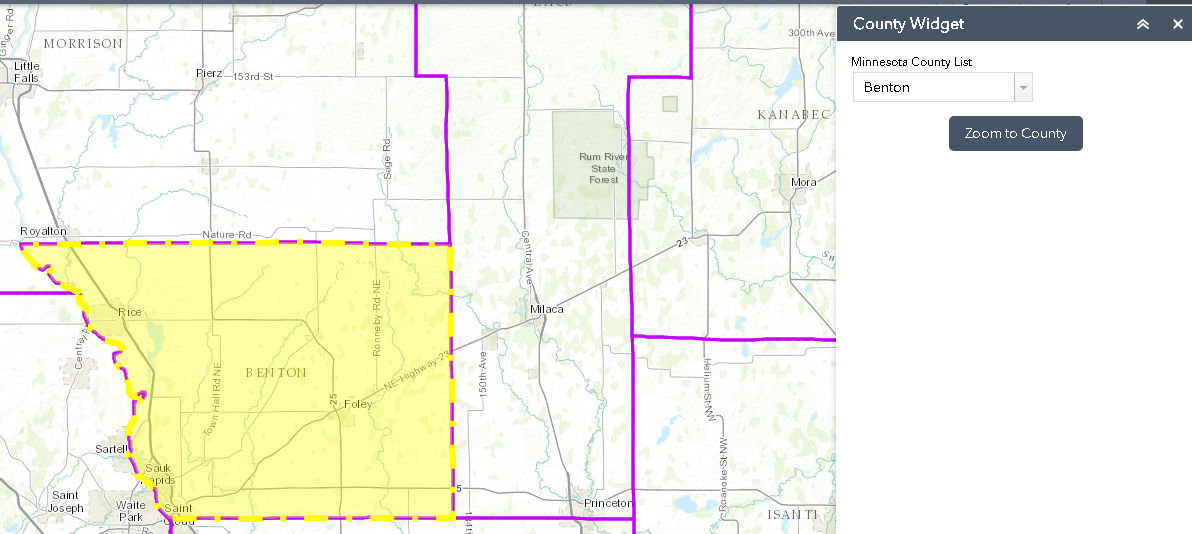
1. Did you get this? Use the developer tools via F12 to debug the error. A missing comma or semicolon can cause the error.



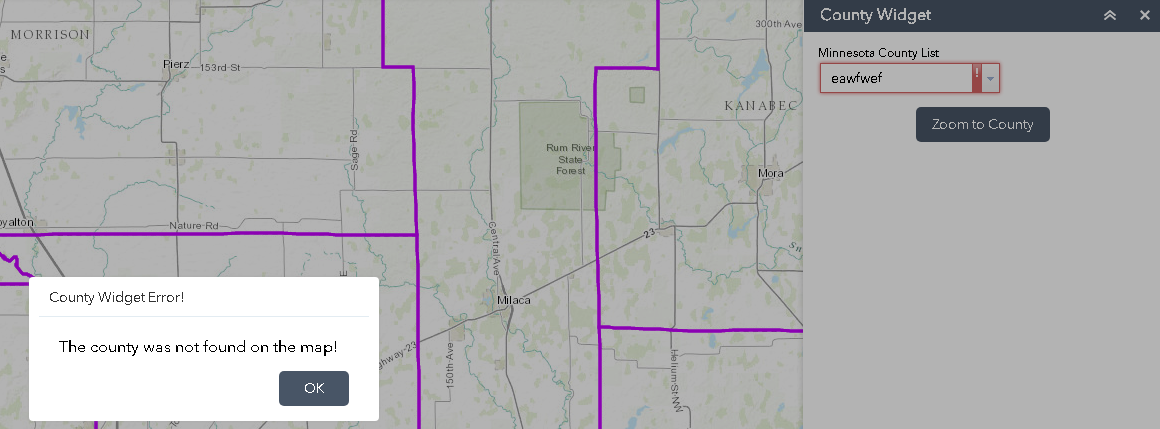
1. The county list dropdown should contain a list of all Minnesota counties in alphabetical order.



1. Select a county and zoom to it by clicking **Zoom to County**. The map should zoom to the county selected and highlight it in yellow.



1. Try entering an invalid county name. A trapped error message should appear.



1. Now let’s clean up the styling a little bit by adding the following to the **style.css** file. Add the following syntax to the file.

***#filterBlock{***

***border-top: 3px solid;***

***padding-top:10px;***

***}***

***#buttonBlock{***

***margin-top:10px;***

***}***

***#MNcountylist{***

***width:190px;***

***}***

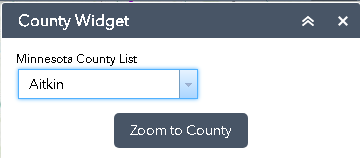
***#btnVote{***

***width:175px;***

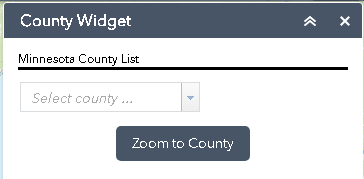
***text-align:center;***

***}***

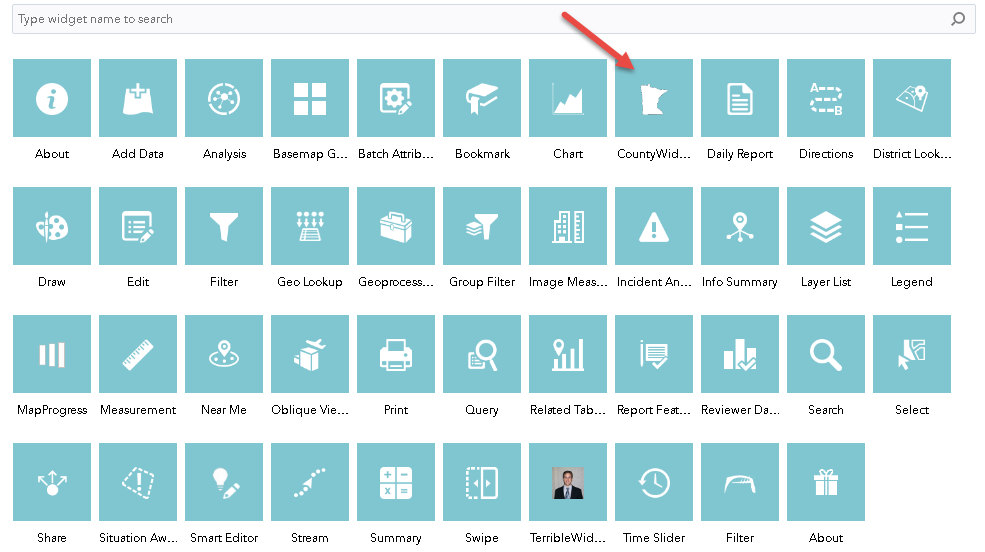
1. The change in the CSS should be as seen below.



CSS:



1. Finally, to deploy this widget for use with an app place the entire folder in [\\client\stemapp\widgets](file:///\\client\stemapp\widgets)
2. Before the changes take effect you will need to restart your node session. To close your node session, close the startup.bat file and start a new node session by opening the startup.bat file.
3. Once you’ve started a new node session, add a new widget and the new widget should appear in the list of available widgets.



1. That’s it! If you have some time remaining play with modifying the widget. Look into the ESRI libraries contained in the **widget.js** file. Guides on this can be found on the [ArcGIS API for JavaScript website](https://developers.arcgis.com/javascript/3/jsapi).