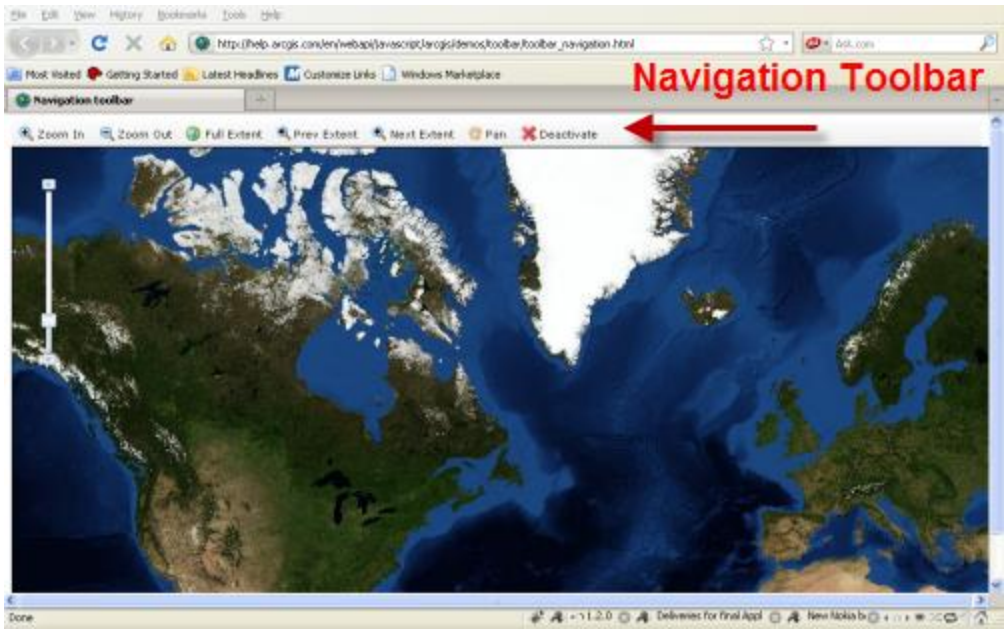
Chapter 8: Using Toolbars

As a GIS web application developer you want to focus on building functionality specific to the application you are constructing. Spending valuable time and effort adding basic GIS functions such as zooming and panning to your application detract from what should be your primary focus.

The ArcGIS Server API for JavaScript includes helper classes for adding navigation and drawing toolbars to your application. In this chapter you'll learn how easy it is to add these user interface components to an application.

Let's start by examining a navigation sample that ESRI has placed on their resource center website. Open a web browser and go to:
http://help.arcgis.com/en/webapi/javascript/arcgis/demos/toolbar/toolbar_navigation.html



At first glance you'd think that the navigation toolbar is simply a user interface component that you drop into your application. Not exactly. The ArcGIS Server API for JavaScript provides a toolbar helper class called esri.toolbars.Navigation to assist with accomplishing this task. In addition, the API also provides a class to handle drawing tasks such as lines and polygons. What these helper classes do is save you the work of drawing zoom boxes, capturing mouse clicks and other user initiated events. As any experienced GIS web developer can tell you this is no small accomplishment. Developing these basic navigation capabilities into the helper classes provided with the API can easily save dozens of hours of development work.

Adding Toolbars to an Application

There are two basic types of toolbars that you can add to your application using helper classes provided by the API: navigation and drawing. There is also an editing toolbar that can be used to edit features through a web browser. We'll discuss this toolbar in the Editing Data chapter.

Steps for Creating a Toolbar

As I mentioned on the previous slide the navigation and draw toolbars are not simply user interface components that you can drop into your application. They are helper classes and there are several steps that you need to take to actually create your toolbar with the appropriate buttons. This "to do" list for your toolbars may seem a little intimidating but once you've done this once or twice it becomes pretty simple. The steps are listed below and we'll discuss each item in detail.

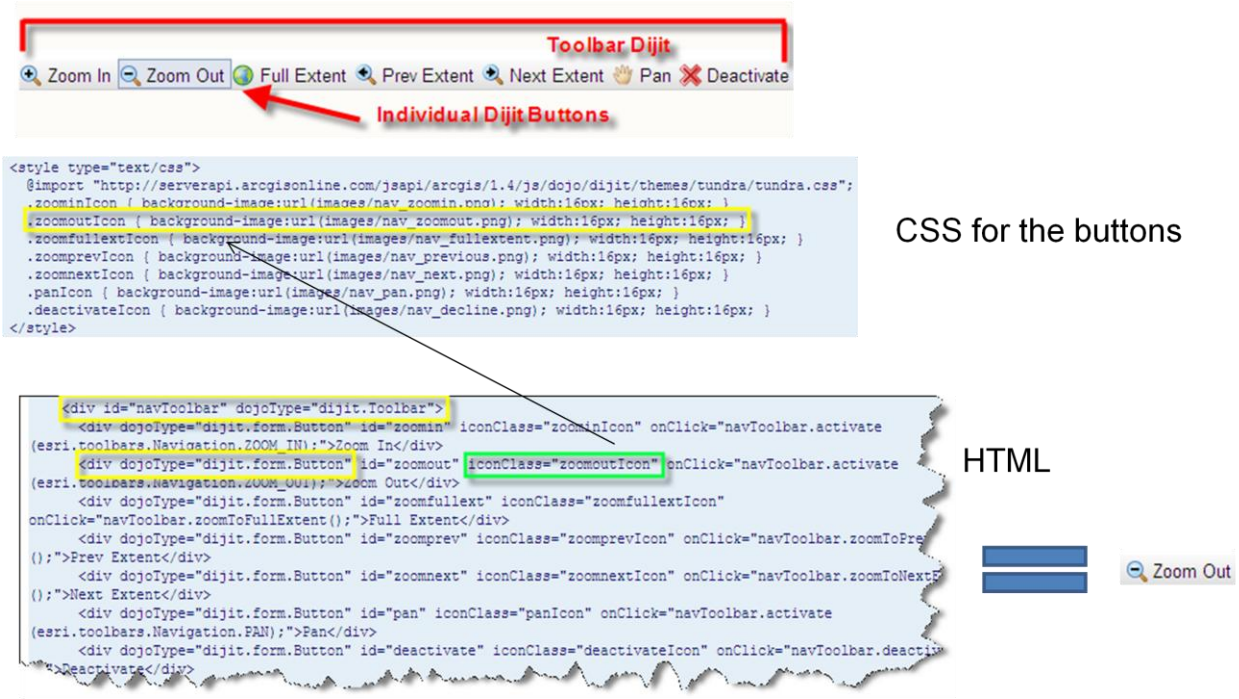
1. Define the CSS styles for each button
2. Create the Toolbar
3. Create the Buttons inside the toolbar
4. Reference the CSS styles for each button in HTML
5. Define functionality that will be enabled when button is clicked
6. Create an instance of esri.toolbars.Navigation or esri.toolbars.Draw
7. Connect button events to handler functions

The first thing you'll need to do is define the CSS styles for each button that you intend to include on the toolbar. Each button on your toolbar will need an image, text, or both along with a width and height for the button. Each of these properties is defined within the CSS.

Next, you'll need to use a Dojo Dijit called Toolbar and inside this toolbar you'll have one or more Dijit Buttons, one for each button you intend to include. Each button will need to reference the corresponding CSS style that you defined in step 1 using the iconClass property.

Finally, you'll want to create an instance of either esri.toolbars.Navigation or esri.toolbars.Draw and connect the button events to a handler function.

In the figure below you'll see code examples showing the first five steps for creating a toolbar. Here we are creating the CSS that defines how each button will appear, constructing the toolbar and buttons, and referencing the CSS styles for each button via the iconClass property on each button.



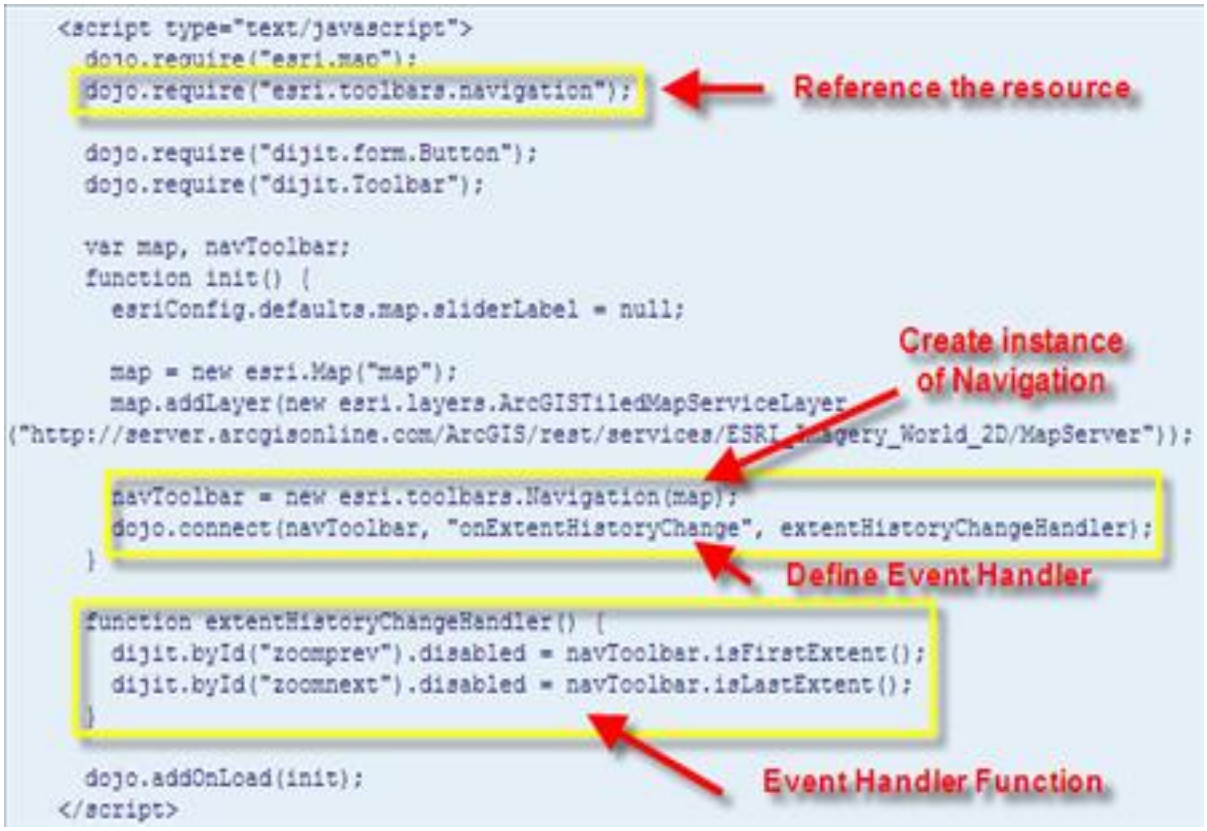
The first code block defines the CSS styles to use for each button. Notice the reference to the Zoom Out button. As with all the other buttons we define an image to use for the button (nav_zoomout.png) along with the width and height of the button.

The second code block defines the toolbar and buttons on the toolbar. The toolbar you see in the example is created using the Toolbar and Button user interface controls provided by Dijit (a sub-project of Dojo). Each control is enclosed within a tag inside the <body> of the web page with all buttons being enclosed by the surrounding that contains the Toolbar dijit.

Each button can be assigned both an icon and text. Notice in the figure above that the 'Zoom Out' Button assigns a value of 'zoomoutIcon' to the 'iconClass' attribute. This simply points to a CSS style that has been defined at the top of the file. Notice in the example that we have a style called 'zoomoutIcon' which defines an image to use for our button.

Now that the visual interface for the toolbar and buttons is complete we need to create an instance of esri.toolbars.Navigation or esri.toolbars.Draw and wire up the events and event handlers.

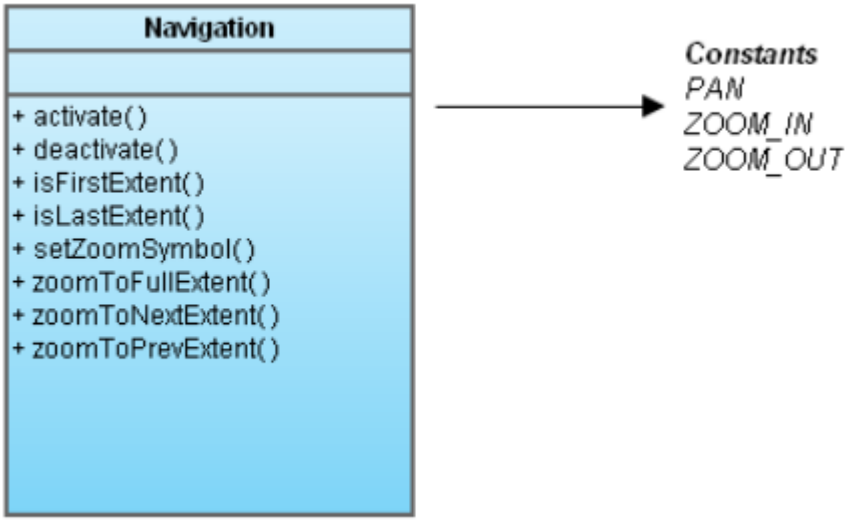
Creating an instance of the Navigation class is as easy as calling the constructor and passing in a reference to esri.map as seen in the code example below. However, you'll first want to make sure that you add a reference to esri.toolbars.navigation.




```
<script type="text/javascript">
  dojo.require("esri.map");
  dojo.require("esri.toolbars.navigation");

  dojo.require("dijit.form.Button");
  dojo.require("dijit.Toolbar");

  var map, navToolbar;
  function init() {
    esriConfig.defaults.map.sliderLabel = null;

    map = new esri.Map("map");
    map.addLayer(new esri.layers.ArcGISTiledMapServiceLayer
("http://server.arcgisonline.com/ArcGIS/rest/services/ESRI_Imagery_World_2D/MapServer"));

    navToolbar = new esri.toolbars.Navigation(map);
    dojo.connect(navToolbar, "onExtentHistoryChange", extentHistoryChangeHandler);
  }

  function extentHistoryChangeHandler() {
    dijit.byId("zoomprev").disabled = navToolbar.isFirstExtent();
    dijit.byId("zoomnext").disabled = navToolbar.isLastExtent();
  }

  dojo.addOnLoad(init);
</script>
```

Finally, we use dojo.connect to wire up an event to an event handler. In this case we are connecting the 'onExtentHistoryChange' event to the 'extentHistoryChangeHandler' function which will run when the extent history changes.

In an earlier step we created tags to serve as containers for the toolbar and buttons. The Navigation class contains a number of constants and methods as seen below.



Each of the buttons that we defined within tags has an 'onClick' attribute which refers to an event that fires when the button is clicked. In the case of the 'Zoom Out' button seen in the code example, a click of the button calls the 'activate' method on Navigation. A navigation type is passed into the 'activate' method. The navigation type can be one of three constants: PAN, ZOOM_IN, ZOOM_OUT. In each case these buttons are really tools that will require an additional map action by the user. For example, in the case of 'ZOOM_IN' the user would then draw a rectangle on the map which would trigger the generation of a new map displayed to the user.

```
navToolbar = new esri.toolbars.Navigation(map);
```

```
<div dojoType="dijit.form.Button" class="btn" id="zoomout" iconClass="zoomoutIcon"
    onClick="navToolbar.activate(esri.toolbars.Navigation.ZOOM_OUT);" showLabel="false">Zoom Out</div>
```

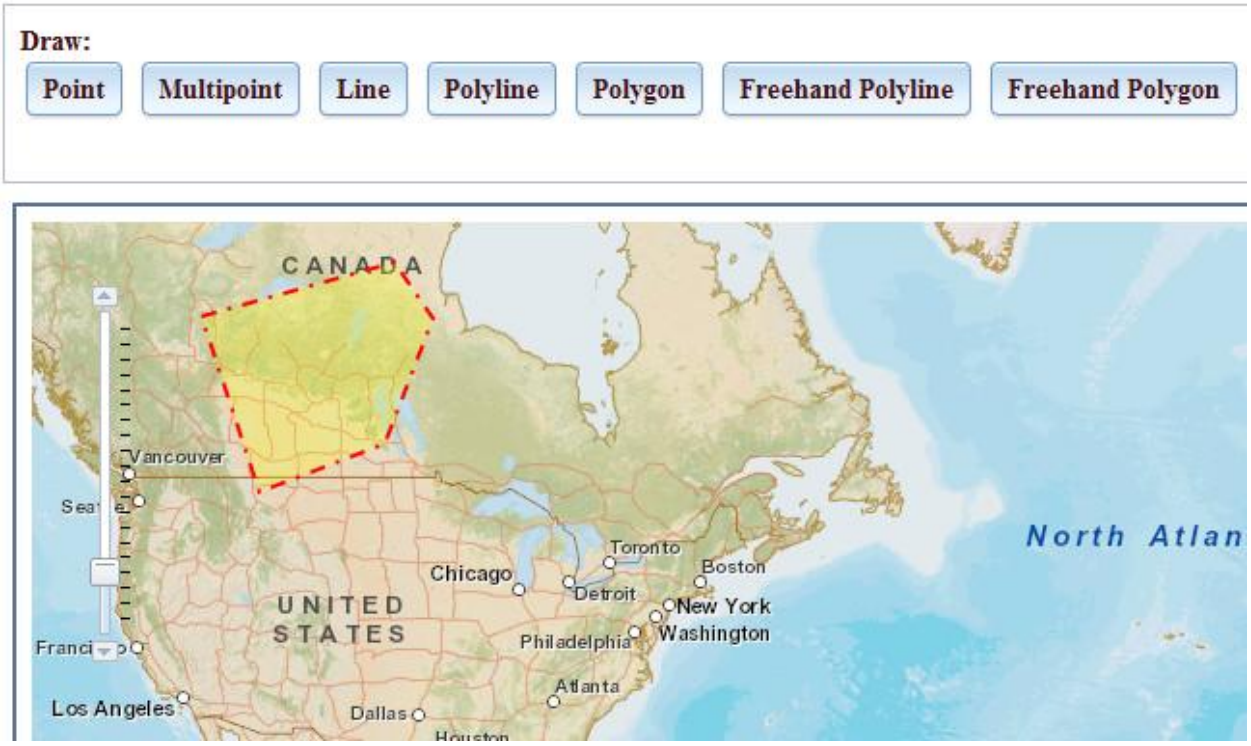
We've already discussed the details of creating toolbars and the buttons contained therein.

Hopefully the previous Navigation Toolbar example has illustrated how easy it is to add a navigation toolbar to your ArcGIS Server application through the JavaScript API. You no longer have to be concerned with adding in JavaScript code to draw and handle the extent rectangle or capture mouse coordinates for a pan operation. In addition, the user interface components of the toolbar can be created easily through various user interface controls supplied by the Dijit library. The Draw class makes it equally easy to support the drawing of geometries such as points, lines, and polygons within a similar toolbar. The creation of this toolbar will follow the same process as we examined with the Navigation toolbar.

Drawing Toolbar

Hopefully the previous section on the Navigation Toolbar has illustrated how easy it is to add a navigation toolbar to your ArcGIS Server application through the JavaScript API. You no longer have to be concerned with adding in JavaScript code to draw and handle the extent rectangle or capture mouse coordinates for a pan operation. In addition, the user interface components of the toolbar can be created easily through various user interface controls supplied by the Dijit library.

The Draw class makes it equally easy to support the drawing of geometries such as points, lines, and polygons within a similar toolbar. The creation of this toolbar will follow the same process as we examined with the Navigation toolbar.



Creating the Drawing Toolbar

The Drawing toolbar requires the use of the esri.toolbars.draw resource which you'd import into your application through the use of dojo.require().

```
dojo.require("esri.toolbars.draw");
```

To create a new instance of the Drawing toolbar simply call the constructor, passing in the map that it will be associated with.

```
toolbar = new esri.toolbars.Draw(map);
```

The toolbar can then be activated with specific mouse behavior for drawing features. The activate() method on the toolbar accepts a geometry type as well as various options. Only a single geometry type is passed in to the activate() method each time the toolbar is activated. Geometry types can include the following:

Constant	Description
ARROW	Draws an arrow.
CIRCLE	Draws a circle.
DOWN_ARROW	Draws an arrow that points down.
ELLIPSE	Draws an ellipse.
EXTENT	Draws an extent box.
FREEHAND_POLYGON	Draws a freehand polygon.
FREEHAND_POLYLINE	Draws a freehand polyline.
LEFT_ARROW	Draws an arrow that points left.
LINE	Draws a line.
MULTI_POINT	Draws a Multipoint.
POINT	Draws a point.
POLYGON	Draws a polygon.
POLYLINE	Draws a polyline.
RECTANGLE	Draws a rectangle.
RIGHT_ARROW	Draws an arrow that points right.
TRIANGLE	Draws a triangle.
UP_ARROW	Draws an arrow that points up.

The code example below illustrates the activation of the drawing toolbar with a freehand polygon geometry type. This geometry type allows the user to enter a polygon by clicking multiple points on the map.

```
toolbar.activate(esri.toolbars.Draw.FREEHAND_POLYGON);
```

Various options can also be supplied as the second parameter in the constructor for the Draw toolbar. A code example showing various options for the Draw toolbar is shown below. These include a boolean parameter (showTooltips) that specifies whether or not to show tooltips when creating new graphics. The tooltip is attached to the mouse as it moves around the screen. The tooltipOffset option specified how far, in pixels, the tooltip should be offset from the mouse pointer.

When working with freehand tools such as freehand polygon or freehand line the tolerance option can be used to define how often a vertice should be placed. Finally, the drawtime option determines how much time to wait before adding a new point when using a freehand tool. The default value is 75.

```
toolbar = new esri.toolbars.Draw(map, {
    tooltipOffset:20,
    drawTime:90
  });
```

In most cases with the Draw toolbar you'll create a number of buttons using Dojo just as was the case with the Navigation toolbar. A combination of CSS and Dojo dijits can be used to create these tools.