mapserv.js

Purpose:

mapserv.js is a javascript library for managing coordinates/layers for interactive mapping applications. It was designed to work with the MapServer CGI application versions 3.6+ although it could be adapted to work with any server-based mapping engine or service. At present two interfaces: 1) Java applet (jBox) and 2) DHTML (dbox.js) have been developed to sit atop this software and provide simple functionality like “rubber-band” zoom and query.

Usage:

Step 1: include mapserv.js
Step 2: pick an interface, DHTML or Java
Step 3: set global variables to point to MapServer
Step 4: create new Mapserv object
Step 5: if necessary, create Mapserv reference map
Step 6: create Layers, run buildlayers()
Step 7: if necessary, write any mapserv.js extension functions (predraw/postdraw)
Step 8: write any required interface functions
Step 9: create map components, layer controls and such in a web document

(see appendix for a basic example)

Global Variables:

- Interface [*java*|*dhtml*] – which interface is being used for the application (*default is java*)
- MapServer [*string*] – URL for the engine (e.g. /cgi-bin/mapserv) for drawing
- QueryServer [*string*] – URL for the engine (e.g. /cgi-bin/mapserv) for query

- PixelsPerInch [*double*] – number of pixels/inch, used for scale computations (*default=72*)
- InchesPerMapUnit [*double*] – number of inches/map unit, used for scale computations (*default=39.3701, meters*)

Classes, Methods and Attributes:

Layer – holds a single layer definition

Constructor –

new Layer(name, longname, group, status, image, metadata) where:

- name [*string*] – name of the layer, typically the same as in a map file
- longname [*string*] – descriptive layer name
- group [*string*] – allows a logical grouping of mapserv.js Layers (not the same as a MapServer group), typically this is used to group layers that would share a common set of form controls (e.g. background or foreground layers)
- status [*true*|*false*] – starting visibility for this layer
- image [*string*] – URL to a reference image for this layer
- metadata [*string*] – metadata (either text or a URL) for this layer

Attributes – none other than those listed above

Methods – none

Mapserv – maintains state for a single application

Constructor –

new Mapserv(name, mapfile, minx, miny, maxx, maxy, width, height) where:

- name [*string*] – name of this object, this is used to link the object to a main mapping window in the interface (e.g. they MUST share the same name)
- mapfile [*string*] – which map file do you want to use for rendering maps
- minx/miny/maxx/maxy [*double*] – default map extent
- width/height [*int*] – size of the mapping window (in pixels)

Attributes – in addition to those listed above

- box [*true*|*false*] – is "rubber-band" box processing enabled
- cellsize [double] – read-only, cellsize of currently displayed map
- extent {array of doubles} – holds extent of currently displayed map
- defaultextent {array of doubles} – holds extent as initialized
- layers {array of Layers} – see above
- layerlist [*string*] – read-only, space delimited list of visible layers
- mode [*map*|*query*|*nquery*|*...*] – current mode of operation (*default is 'map'*)
- options [*string*] – user defined parameters to tag on to a computed rendering URL
- pansize [*double*] – when panning how far (in images) do you want to move (*default is .8*)
- queryextent {array of doubles} – holds extent of current query
- queryfile [*string*] – which map file do you want to use for queries *(default=mapfile)*
- queryoptions [*string*] – user defined parameters to tag on to a computed query URL
- referencemap {Mapserv object} – a Mapserv object for corresponding reference map
- url [*string*] – read-only, computed URL for a given operation, in most cases mapserv.js will not actually do anything besides compute a rendering or query URL, the application must use this information as it sees fit (e.g. push query to a popup window or frame)
- zoomsize [*int*] – zoom magnitude (*default is 2*)
- zoomdir [*1*|*0*|*-1*] – zoom direction (*default is 0 or pan*)

Methods –

- applybox(minx, miny, maxx, maxy) – sets extent given "rubber-band"
- applyquerybox(minx, miny, maxx, maxy) – sets queryextent given a "rubber-band" box
- applyquerypoint(x,y)- sets queryextent given a mouse click
- applyreference(x,y) – sets extent given a mouse click in the reference map
- applyzoom(x,y) – sets extent given a mouse click (using zoomsize and zoomdir)
- boxoff() – wrapper for interface specific method to turn "rubber-band" functionality off
- boxon() – wrapper for interface specific method to turn "rubber-band" functionality on
- buildlayers() – computes layerlist based on all layer status values
- draw() – sets url, and triggers interface specific draw methods
- getlayerindex(name) – gets array index by layer name
- getlayerstatus(name) – gets layer status by name
- getscale() – returns current map's scale

- layersoff() – sets all layer's status attribute to false
- pan(direction) – sets extent based on a direction, valid values are nw|n|ne|w|e|sw|s|se
- query() – sets url based on mode, queryextent, extent, and so on...
- setextent(minx, miny, maxx, maxy) – sets extent to supplied values and adjusts it to fit current map geometry
- setextentfromradius(x, y, radius) – sets extent based on a point (in map coordinates) and a radius (in map units), extent is adjusted to fit current map geometry
- setextentfromscale(x, y, scale) – sets extent based on a point (in map coordinates) and a scale value (e.g. 24000 = 1:24,000), extent is adjusted to fit current map geometry
- setlayerstatus(name, status) – sets a layer's status by name
- togglelayers(element) – controls form element (checkbox/select list/radio button) based on layer status values (e.g. if layer.status=true then the checkbox is checked), element values must match layer names
- zoomin(x,y) – sets extent given a image coordinate (zoomdir=1)
- zoomdefault() – sets extent to original extent
- zoomout(x,y) – sets extent given a image coordinate (zoomdir=-1)

dbox.js

Purpose:

dbox.js is a javascript library for creating DHTML rubber-band box components that utilize the mapserv.js coordinate management library. The code is built upon the CBE (www.cross-browser.com) cross-browser library so it is quite portable. The component is patterned after the old mapplet (now called jbox) applet so that the DHTML or Java-based controls could easily be used with mapserv.js.

Usage:

Step 1: include CBE (cbe_core.js and cbe_event.js)
Step 2: include dbox.js
Step 3: create a dbox object
Step 4: create the relatively positioned anchor layer amongst the HTML content
Step 5: create the absolutely positioned layer to hold the component
Step 6: call the objects initialize method in the pages onLoad method (or use CBE windowOnload function)

(see appendix for a basic example)

Global Variables:

- dBox_BusyMessage [*string*] – message to display in browser status area when a component is busy (*default is 'fetching map'*)
- dBox_NotBusyMessage [*string*] – message to display in browser status area when a component is busy (*default is nothing*)

Classes, Methods and Attributes:

dBox – DHTML-based map component

Constructor –

new dBox(name, width, height, color, thickness) where:

- name [*string*] – name of this component, matches the name of the main Mapserv object
- width/height [*int*] – height and width of this component, matches the main Mapserv object
- color [*string*] – color of the "rubber-band" box, given as a hexadecimal string (e.g. #ff0000=red) or named javascript color (e.g. white)
- thickness [*int*] – thickness (in pixels) of the "rubber-band" box

Attributes – in addition to those listed above

- box [*true*|*false*] – is "rubber-band" box functionality enabled (*default is true*)
- cursorsize [*int*] – size (in pixels) of the cross-hair cursor
- jitter [*int*] – sets a minimum "rubber-band" box size, prevents poor mouse clicking from triggering unwanted zooming

- verbose [*true*|*false*] – should the component report all mouse movement (*default is false*)

Methods –

- boxoff() – sets box attribute to false, hides any necessary layers
- boxon() – sets box attribute to true
- initialize() – creates all the layers necessary to do “rubber-band” boxes
- setimage(imgurl) – swaps the current content of the component with a new image
- sync() – aligns the absolutely positioned layer with its anchor

Appendix – Minimal Setup Example (Itasca Demo)

```
<html><head>
  <title>MapServer - Itasca Application</title>

  <!-- the DHTML JavaScript library includes -->
  <script type="text/javascript" src="javascript/cbe/cbe_core.js"></script>
  <script type="text/javascript" src="javascript/cbe/cbe_event.js"></script>

  <!-- MapServer specific JavaScript library includes -->
  <script language="javascript" src="javascript/mapserv.js"></script>
  <script language="JavaScript" src="javascript/dbox.js"></script>

  <!-- scripting specific to the application -->
  <script language="javascript">
    // the DHTML main mapping window (note the significance of the name "main"
    // here and    with the Mapserv object)
    var main = new dBox("main", 600, 600, "#FF0000", 2);
    main.verbose = true;

    // the DHTML reference map window
    var reference = new dBox("reference", 120, 120, "#989898", 1);
    reference.box = false;
    reference.cursorsize = 0;

    // mapserv.js global variables
    var MapServer = "/cgi-bin/mapserver";
    var QueryServer = MapServer;
    var Interface = "dhtml";

    // create the Mapserv object
    var ms = new Mapserv("main", "itasca.map", 388107, 5203120, 500896, 5310243, 600, 600);
    ms.minscale = 1000;
    ms.maxscale = 1550000;

    // layer definitions
    ms.layers[0] = new Layer('airports', 'Airports', 'layers', false, null, null);
    ms.layers[1] = new Layer('cities', 'Cities', 'layers', false, null, null);
    ms.layers[2] = new Layer('lakespy2', 'Lakes & Rivers', 'layers', true, null, null);
    ms.layers[3] = new Layer('dlgstln2', 'Streams', 'layers', true, null, null);
    ms.layers[4] = new Layer('roads', 'Roads', 'layers', false, null, null);
    ms.layers[5] = new Layer('twprgpy3', 'Townships', 'layers', false, null, null);
    ms.layers[6] = new Layer('drgs', 'USGS 1:250,000 Quads', 'layers', false, null, null);
    ms.buildlayers();

    // add the reference map
    ms.referencemap = new Mapserv("reference", “itasca.map”, 393234, 5205405, 495769, 5307959, 120, 120);
```

```
//
// Extensions to Mapserv.draw(): this allows you to exend the capabilties of
// of the default draw method. There are post and pre draw functions available.
//
function predraw() {
  // update the scalebars
  document.scalebar_miles.src = MapServer + "?map=" + ms.mapfile + "&mode=scalebar&mapext=0+0+" + (ms.extent[2] - ms.extent[0])
                              + "+" + (ms.extent[3] - ms.extent[1]) + "&mapsize=" + ms.width + "+" + ms.height;
  document.scalebar_kilometers.src = MapServer + "?map=" + ms.mapfile + "&map_scalebar_units=kilometers&mode=scalebar&mapext=0+0+"
                                   + (ms.extent[2] - ms.extent[0]) + "+" + (ms.extent[3] - ms.extent[1]) + "&mapsize="
                                   + ms.width + "+" + ms.height;

  // update the legend
  document.legend.src = MapServer + "?map=" + ms.mapfile + "&mode=legend&layers=" + ms.layerlist;

  // because the legend can change size AND is above the reference map we need to re-sync it with its anchor
  reference.sync();
}

//
// Functions that are called by the jBox applet or the dBox javascript code:
// basically these provide the gateway from the applet/layers to the rest of
// the application. Note that they are the same regardless of implementation.
//
// Bottom line: you may want to swipe some of this code.
//
// jBox/dBox errors are passed to the browser via this function
function seterror_handler(name, message) { alert("Component " + name + " error: " + message); }

// allows jBox/dBox to reset without redrawing
function reset_handler(name, minx, miny, maxx, maxy) { }

// called from jBox/dBox when the user initiates change
function setbox_handler(name, minx, miny, maxx, maxy) {
  if(name == 'reference') {
    ms.applyreference(minx, miny);
    ms.draw();
  } else {
    if(ms.mode == 'map') {
      if(minx != maxx && miny != maxy)
        ms.applybox(minx, miny, maxx, maxy);
      else
        ms.applyzoom(minx, miny);
      ms.draw();
    } else if(ms.mode != 'map') {
      ms.applyquerybox(minx, miny, maxx, maxy); // these just set members
      ms.applyquerypoint(minx, miny);
      ms.query(); // builds query URL
      window.open(ms.url); // open the query in a new window
```

```
          }
        }
      }

      // various event handlers called by jBox/dBox
      function mousemove_handler(name, x, y) {
        window.status = "UTM Coordinates: x=" + Math.round(Number(ms.extent[0] + x*ms.cellsize)) + " and y="
                      + Math.round(Number(ms.extent[3] - y*ms.cellsize));
      }
      function mouseexit_handler(name) { window.status = ""; }
      function mouseenter_handler(name) { window.status = ""; }

      // page initialization function
      function windowOnload() {
        main.initialize();
        reference.initialize();

        ms.zoomdir=1;
        ms.draw();
      }
    </script>

  </head>
  <body bgcolor=#FFFFFF onResize="main.sync();reference.sync()">

    <center><h1>MapServer - Itasca Application</h1></center>
    <hr>

    <center>
    <table border=0 cellspacing=0 cellpadding=4>
    <tr>
    <td valign="top" align=center>
      <table width="390" border="0" cellspacing="0" cellpadding="0" align="center" bgcolor="#666666">
        <tr>
          <td align="right" width="18"><a href="javascript:ms.pan('nw')"><img src="graphics/nw.gif" border="0"></a></td>
          <td align="center"><a href="javascript:ms.pan('n')"><img src="graphics/n.gif" border="0"></a></td>
          <td align="left" width="18"><a href="javascript:ms.pan('ne')"><img src="graphics/ne.gif" border="0”></a></td>
        </tr>
        <tr>
          <td align="right" width="18"><a href="javascript:ms.pan('w')"><img src="graphics/w.gif’ border="0"></a></td>
          <td align="center" bgcolor="#cccccc">
            <!-- this is the holding spot (the anchor) for the map -->
            <DIV id="main_anchor" style="position:relative; visibility:visible; width:100%; height:100%; left:0px; top:0px;"><IMG
src="graphics/red_pixel.gif" height="600" width="600"></DIV>
            <!-- absolutely positioned layer to hold the map -->
            <DIV id="main" style="position:absolute; visibility:visible; width:100%; height:100%; clip:rect(100%,100%,100%,100%);
background:transparent;"><IMG name="main" src="graphics/transparent_pixel.gif" height="600" width="600"></DIV>
          </td>
          <td align="left" width="18"><a href="javascript:ms.pan('e')"><img src="graphics/e.gif" border="0"></a></td>
        </tr>
```

```
    <tr>
      <td align="right" width="18"><a href="javascript:ms.pan('sw')"><img src="graphics/sw.gif" border="0”></a></td>
      <td align="center"><a href="javascript:ms.pan('s')"><img src="graphics/s.gif" border="0"></a></td>
      <td align="left" width="18"><a href="javascript:ms.pan('se')"><img src="graphics/se.gif" border="0”></a></td>
    </tr>
    <tr><td bgcolor="#666666" colspan="3">
      <img name="scalebar_kilometers" align="right" title="scalebar (km)" alt="scalebar (km)" src="graphics/red_pixel.gif">
      <img name="scalebar_miles" title="scalebar (mi)" alt="scalebar (mi)" src="graphics/red_pixel.gif">
    </td></tr>
  </table>
</td>
<td valign="top" bgcolor=#ffffff>
  <table cellpadding="5" cellspacing="0" border="0" bgcolor="#ffffff">
    <tr><td>
      <!-- Note that we don't have a submit action for this form, we only need the form for some controls -->
      <form name="mapserv" action="javascript:void(0)">
        <p><b>Choose an Action: </b><br>
        <input onClick="ms.mode='map'" type="radio" name="mode" checked> Browse map<br>
        <input onClick="ms.mode='query'" type="radio" name="mode"> Query feature<br>
        <input onClick="ms.mode='nquery'" type="radio" name="mode"> Query multiple features
        <hr>

        <p><b>Select Layers to Display: </b><br>
        <select multiple name="layers" size=3 onChange="ms.togglelayers(this)">
          <option value="airports"> Airports
          <option value="cities"> Cities
          <option value="lakespy2" selected> Lakes & Rivers
          <option value="dlgstln2" selected> Streams
          <option value="roads"> Roads
          <option value="twprgpy3"> Townships
          <option value="drgs"> USGS 1:250,000 Quads
        </select><br>
        <input type="button" value="Refresh Map" onClick="ms.draw()">
        <hr>

        <p><b>Zoom Controls: </b><br>
        Zoom In <input onClick="ms.zoomdir=1" type=radio name=zoomdir checked>
        Pan <input onClick="ms.zoomdir=0" type=radio name=zoomdir>
        Zoom Out <input onClick="ms.zoomdir=-1" type=radio name=zoomdir>
        <p>
        Zoom Size <input type=text name=zoomsize size=4 value=2 onChange="ms.zoomsize=this.value">
        <hr>
      </form>

      <p><b>Legend:</b><br>
      <!-- How you'd do a legend varies by browser. With some browsers that support dynamic image size you could
      handle like the scalebars. Otherwise you need to combine the select list above with pre-computed
      images or use a popup window. -->
      <img name="legend" src="graphics/red_pixel.gif">
```

```
            <p>
            <!-- this is the holding spot (the anchor) for the reference map -->
            <DIV id="reference_anchor" style="position:relative; visibility:visible; width:100%; height:100%; left:0px; top:0px;"><img
src="graphics/red_pixel.gif" height="120" width="120"></DIV>
            <!-- absolutely positioned layer to hold the reference map -->
            <DIV id="reference" style="position:absolute; visibility:visible; width:100%; height:100%; clip:rect(100%,100%,100%,100%);
background:transparent;"><IMG name="reference" src="graphics/reference.gif" height="120" width="120"></DIV>

          </td></tr></table>

        </td></tr>
      </table>
      </center>

      <p><hr><p>

    </body></html>
```