data-vis\gallery.js

```
function Gallery() {

  this.visuals = [];
  this.selectedVisual = null;
  var self = this;


  // Add a new visualisation to the navigation bar.
  this.addVisual = function(vis) {

    // Check that the vis object has an id and name.
    if (!vis.hasOwnProperty('id')
        && !vis.hasOwnProperty('name')) {
      alert('Make sure your visualisation has an id and name!');
    }

    // Check that the  vis object has a unique id.
    if (this.findVisIndex(vis.id) != null) {
      alert(`Vis '${vis.name}' has a duplicate id: '${vis.id}'`);
    }

    this.visuals.push(vis);


    // Create menu item.
    var menuItem = createElement('li', vis.name);
    menuItem.addClass('menu-item');
    menuItem.id(vis.id);

    menuItem.mouseOver(function(e)
    {

        var el = select('#' + e.srcElement.id);
        el.addClass("hover");
    })

    menuItem.mouseOut(function(e)
    {
        var el = select('#' + e.srcElement.id);
        el.removeClass("hover");
    })

    menuItem.mouseClicked(function(e)
    {
        //remove selected class from any other menu-items

        var menuItems = selectAll('.menu-item');

        for(var i = 0; i < menuItems.length; i++)
        {
            menuItems[i].removeClass('selected');
        }

        var el = select('#' + e.srcElement.id);
        el.addClass('selected');

        self.selectVisual(e.srcElement.id);

    })


    var visMenu = select('.visuals-menu');
    visMenu.child(menuItem);

    // Preload data if necessary.
    if (vis.hasOwnProperty('preload')) {
      vis.preload();
    }
  };

  this.findVisIndex = function(visId) {
    // Search through the visualisations looking for one with the id
    // matching visId.
    for (var i = 0; i < this.visuals.length; i++) {
      if (this.visuals[i].id == visId) {
        return i;
      }
    }

    // Visualisation not found.
    return null;
  };

  this.selectVisual = function(visId){
    var visIndex = this.findVisIndex(visId);

    if (visIndex != null) {
      // If the current visualisation has a deselect method run it.
      if (this.selectedVisual != null
          && this.selectedVisual.hasOwnProperty('destroy')) {
        this.selectedVisual.destroy();
      }
      // Select the visualisation in the gallery.
      this.selectedVisual = this.visuals[visIndex];

      // Initialise visualisation if necessary.
      if (this.selectedVisual.hasOwnProperty('setup')) {
        this.selectedVisual.setup();
      }

      // Enable animation in case it has been paused by the current
      // visualisation.
      loop();
    }
  };
  }
```