E4X Use Case – Chart Generator

Chart Generator

This HTML based solution is used to display performance measurement data over time against goals. The code is based off some real world code. It demonstrates one way developers use and manipulate XML data from web pages.

The code and data for this is in attached file perfchart.zip which contains the following:

File	Description
perfchart.htm	Main file which loads up XML data and renders charts.
data\scenarios.xml	Descriptions of tests and corresponding performance goals.
data\259.xml	Perf data for test scenario “HelloWorld – Startup Working Set”
data\262.xml	Perf data for test scenario “HelloWorld – Startup Time”
data\updates.xml	Data used to normalize quantities

The sample can be run by loading up perfchart.htm in IE. I have not tried in Navigator, but suspect it will not work due to the HTML DOM calls being used.

How the Code Access XML

Mapping of XML data to OM

Because the DOM is so cumbersome, the developer decided it was worthwhile to map the XML to objects and process the data in the objects. The code does this through the functions ListLoaded, RetrieveData, and UpdateLoaded.

The data is simple and the operation needed for mapping aren’t terribly taxing. The code uses coercion functions along with the following operations to manipulate the XML.

- node selection – documentElement.selectNodes(…)
- access of attribute values - attributes.getNamedItem(…).nodeValue

```
function ListLoaded()
{
   if (_scenarios.readyState != "complete")
      return;
   var nodes = _scenarios.documentElement.selectNodes("TESTS/TEST");
   _aryScenarios = new Array();
   for (var i = 0; i < nodes.length; i++)
   {
      var testName = nodes[i].attributes.getNamedItem("NAME").nodeValue;
      var o = new Object();
      _aryScenarios[_aryScenarios.length] = o;
      o.name = testName;
      o.id = nodes[i].attributes.getNamedItem("TESTID").nodeValue;
      var milestone = nodes[i].attributes.getNamedItem(_goalAttribute+"GOAL");
      o.glideEnd = milestone == null ? -1 : parseFloat(milestone.nodeValue);
      var e = document.createElement("OPTION");
      e.innerText = testName;
      _selNames.appendChild(e);
   }

   _milestone.innerText = _goalAttribute;
   _selNames.onchange = ScenarioSelected;
   _selNames.selectedIndex = 0;

   if (_aryUpdates != null)
      ScenarioSelected();
}

function RetrieveData()
{
   if (_data.documentElement == null)
```

```
      return;
   var o = _aryScenarios[_selNames.selectedIndex];
   o.data = new Array();
   var nodes = _data.documentElement.selectNodes("RESULT");
   for (var i = 0; i < nodes.length; i++)
   {
      var bn = nodes[i].attributes.getNamedItem("BUILDNAME").nodeValue;
      if (bn.indexOf("Trash") > 0)
         continue;
      var bt = nodes[i].attributes.getNamedItem("BUILDTIME").nodeValue;
      var date = ParseDate(bt);
      if (date < _startTime)
         continue;
      var d = new Object();
      d.date = date;
      o.data[o.data.length] = d;
      d.data=parseFloat(nodes[i].attributes.getNamedItem("VALUE").nodeValue);
      d.data = GetUpdatedData(o.id, d);
   }
   o.minMax = GetMinMax(o.data);
}

function UpdateLoaded()
{
   if (_updates.readyState != "complete")
      return;

   var nodes = _updates.documentElement.selectNodes("Update");
   _aryUpdates = new Array();
   for (var i = 0; i < nodes.length; i++)
   {
      var sDate = nodes[i].attributes.getNamedItem("DateTime").nodeValue;
      var date = ParseDate(sDate).getTime();
      _aryUpdates[date] = new Array();
      for (var j = 0; j < nodes[i].childNodes.length; j++)
      {
         var kid = nodes[i].childNodes[j];
         var name = kid.attributes.getNamedItem("Name").nodeValue;
         _aryUpdates[date][name]
            = parseFloat(kid.attributes.getNamedItem("Scale").nodeValue);
      }
   }

   if (_aryScenarios != null)
      ScenarioSelected();
}
```

Generate of HTML by combining DOM calls with strings

The code uses a combination of DOM calls and strings of “HTML” to generate the chart. Although the code here is generating HTML, the generation of XML is not a huge stretch. The point here is that strings are quite often used by scripters to serialize objects and take care of mapping from one object model to another.

```
   for (var i = 0; i <= _numWeeks; i++)
   {
      var oBar = document.createElement("DIV");
      oBar.className = "GLIDE_BAR";
      if (i >= _numWeeks)
         oBar.style.borderBottomWidth = 0;
      oBar.style.height = i < _numWeeks ? dHeight : 3;
      oBar.style.width = inc * (i + 1);

      var oPadding = document.createElement("SPAN");
      oPadding.className = "GLIDE_PADDING";
      oPadding.style.width = Math.round(inc);
      oBar.appendChild(oPadding);
      _divGlide.appendChild(oBar);
```

```
    //
    // display date
    //
    oDate = document.createElement("SPAN");
    var date = new Date();
    date.setTime(_startTime.getTime() + (i * 7 * 24 * 60 * 60 * 1000));
    oDate.innerHTML = "<CENTER CLASS=DATE_POINTER>&#124;</CENTER>"
                + "<SPAN CLASS=DATE_SHORT>" + (date.getMonth()+1)
                + "-" + (date.getDate()) + "</SPAN>";
    oDate.style.position = "absolute";
    _divGlide.appendChild(oDate);
    oDate.style.left = GetOriginXY(oPadding).x - oDate.offsetWidth / 2;
    oDate.style.top = GetOriginXY(_divGlide).y + _viewHeight;

    //
    // display path values
    //
    var value = initValue - i * totalGlide / _numWeeks;
    oValue = document.createElement("SPAN");
    oValue.innerHTML = "<SPAN CLASS=VALUE_POINTER>"
              + (Math.round(value * 100) / 100) + " "
              + "<SPAN STYLE='font-family:Symbol'>&#190;</SPAN><SPAN>";
    oValue.style.position = "absolute";
    _divGlide.appendChild(oValue);
    oValue.style.top = GetOriginXY(oBar).y - oValue.offsetHeight + 4;
    oValue.style.left = GetOriginXY(_divGlide).x - oValue.offsetWidth;
}
```

Mapping To E4X

This use case provides two alternatives approaches to consider in an E4X based solution:

1. Keeping architecture with an XML to object model transformation layer. E4X is there to makes it easier to create this layer.
2. Remove the mapping layer so code directly processes XML.

People would continue to use option 1 in cases where the format of the XML data isn't in a format that they'd use in their program. The cost of this option is the duplication of data.

Some example write ups of possible E4X solutions are pending.