Using ECMA-323 (CSTA XML) in a Voice Browser Environment - Port to E4X

Summary of Observations

- Use of selectSingleNode very common – what is E4X equivalent?
- Test for 'empty result' is ugly – if (answer.length() == 0)
- Use of "." for context object looks weird – too much punctuation
- It is important for host provided object model to be typed. Early coercision out of XML type makes it easier to reason how a program behaves.
- XML literal { } syntax is very useful
- Method calls for properties is ugly. Perhaps we can denote reserved namespace for these?

```
namespace e4x("http://www.e4x.com")
if (msg.e4x::name == "test")…
```

- XML embedded in HTML/XML is very confusing
- There are quite a few casts to XML type? Do we need a shortcut for this cast?

Sample Ported to E4X

Data for the application:

```
<input name="transferTarget" />
<input name="callerID" />
<input name="callID" />
<input name="deviceID" />
<input name="monitorObject" type="hidden" value="2234" />
<input name ="monitorCrossRefID" />
Speech objects in English (only section affected by natural language):
<listen id="recNumber" onreco ="procRecNumber()" onnoreco="procNoReco()
onsilence="procNoReco()">
<grammar src="..."/>
</listen>
<listen id="recYesNo" onreco="procYesNo()" onnoreco="procNoReco()"
onsilence="procNoReco()">
<grammar src="..."/>
</listen>
<prompt id="sayWelcome">Hello! Please say the phone number to transfer to. </prompt>
<prompt id="askAgain"> Sorry, I missed that. Please say the number again. </prompt>
<prompt id="confirm"> Did you say <value href="transferTarget"/>? </prompt>
<prompt id="sayBye"> Thank you. Your call is being transferred. </prompt>
<prompt id="tryAgain">
The number, <value href="transferTarget"/>, cannot be
reached for transfer. Please try again later.
</prompt>
```

Speech event handlers (dialog logic) in ECMAScript:

```
<script><!--
function procRecNumber() {
   var msg = XML(event.srcElement.recoresult);
   transferTarget.value = String(msg.*.phoneNumber);
   // read recognized phone number
   if (msg.@confidence < 0.5) {
      confirm.Start(); recYesNo.Start();
   }
```

```
   else {
      sayBye.Start(); ccTransfer();
   }
}
function procYesNo() {
   var answer = XML(event.srcElement.recoresult).*.yes.(.@confidence>0.5);
   // accept only yes with confidence
   if (answer.length() == 0) {
      procNoReco();
   }
   else {
      sayBye.Start(); ccTransfer();
   }
}
function procNoReco() {
   transferTarget.value = "";
   askAgain.Start(); recNumber.Start();
}
--></script>
```

The call control section (unaffected by locale, dialog logic):

```
<smex id="callControl" onreceive="ccHandler()">...</smex>
<script><!--
// The cchandler handles the ECMA-323 events.
//
// Once the connection is answered, a welcome prompt
// is played and the transfer target telephone number is solicited.
//
// When the speech event handler detects and confirms the correct
// speech input, an ECMA-323 SingleStepTransfer service is used to
// transfer the caller to the new transfer target.
//
function ccHandler() {
   var msg = XML(event.srcElement.received);
   if (msg.name() == "DeliveredEvent") { // incoming call notification
      //
      // If the connection is alerting (DeliveredEvent, ECMA-323, 15.2.5) the
      // connection information from the Delivered event is saved
      // called.value and deviceID.value) and the call is answered by using the
      // ECMA-323 AnswerCall service with the saved connection information.
      // If the application needed the ANI and DNIS, it could also obtain
      // this information from this event.
      //
      callID.value = msg.connection.callID;
      deviceID.value = msg.connection.deviceID;
      ccAnswer();
   }
   else if (msg.name() == "EstablishedEvent") { // call answered
      //
      // Once the connection is answered (EstablishedEvent, ECMA-323, 15.2.8)
      // a welcome prompt is played and the transfer target telephone number
      // is solicited.
      //
      callerID.value = msg.callingDeviceDeviceIdentifier;
      sayWelcome.Start(); recNumber.Start();
   }
   else if (msg.name() == "TransferredEvent") { // call transferred
      //
      // The TransferredEvent (ECMA-323, 15.2.18) is received when
      // the transfer has been completed. ccCleanup is called to clean up
      // the application data.
      //
      ccCleanUp();
   }
   else if (msg.name() == "ConnectionClearedEvent") { // user hang up
      //
      // A user hang up is indicated by a ConnectionClearedEvent (ECMA-323,
      // 15.2.4) which flushes the prompt queue and cleans the application
```

```
      // data. This could happen at any time during the call.
      //
      promptQueue.Flush();
      ccCleanUp();
   }
   else if (msg.name() == "CSTAErrorCode") { // service failure event
      //
      // The ccError function handles any failure responses from any of the
      // ECMA-323 services that may have failed.
      //
      ccError();
   } // feel free to handle other events here
}
function ccTransfer() { // transferring a call
   //
   // The SingleStepTransferCall service (ECMA-323, 15.1.24) is used to
   // invoke the transfer. There are two elements provided. The first
   // element is the connection information that was obtained from
   // the DeliveredEvent. The second element is the transfer target that
   // was solicited from the caller.
   //
   callControl.sent =
         <SingleStepTransferCall xmlns='http://www.ecma.ch/standards/ecma-323/csta/ed2'>
           <activeCall>
             <callID>{callID.value}</callID>
             <deviceID>{deviceID.value}</deviceID>
           </activeCall>
           <transferredTo>{transferTarget.value}</transferredTo>
         </SingleStepTransferCall>;
}
function ccStartListening() { // listening for call events
   //
   // The MonitorStart service (ECMA-323, 13.1.2) is used to place a
   // monitor on a device so that events can be generated when activity
   // happens at that device. The single element provided indicates the
   // identifier of the device that is to be monitored. In this example
   // it was part of the application data.
   //
   callControl.sent =
         <MonitorStart xmlns='http://www.ecma.ch/standards/ecma-323/csta/ed2'>
           <monitorObject>
             <deviceObject>{monitorObject.value}</deviceObject>
           </monitorObject>
         </MonitorStart>;
}
function ccAnswer() { // answering a call
   //
   // The AnswerCall service (ECMA-323, 15.1.3) is used to answer the
   // alerting connection. The single element provided is the connection
   // information that was obtained in the Delivered event.
   //
   callControl.sent =
         <AnswerCall xmlns='http://www.ecma.ch/standards/ecma-323/csta/ed2'>
           <callToBeAnswered>
             <callID>{callID.value}</callID>
             <deviceID>{deviceID.value}</deviceID>
           </callToBeAnswered>
         </AnswerCall>;
}
function ccCleanUp() {
   callerID.value = ""; transferTarget.value = ""; callID.value = "";
   recNumber.Stop(); recYesNo.Stop(); ...
}
function ccHangup() { // clearing a connection
   //
   // The ClearConnection service (ECMA-323, 15.1.8) is used to clear
   // a connection. In this example, this is used when the transfer is
   // unable to be completed.
   //
   callControl.sent =
         <ClearConnection xmlns='http://www.ecma.ch/standards/ecma-323/csta/ed2'>
```

```
          <connectionToBeCleared>
            <callID>{callID.value}</callID>
            <deviceID>{deviceID.value}</deviceID>
          </connectionToBeCleared>
        </ClearConnection>;
}
function ccError() {
   //
   // The ccError function is called to handle any failure responses to
   // ECMA-323 service requests. If there was an error starting a
   // monitor, the application logs an error. If there was an error
   // response to the SingleStepTransfer service, a message is played
   // for the caller and the connection is cleared.
   //
   var request = callControl.sent.substr(1, 7);
   // read the first 7 characters of service requested
   if (request == "Monitor") { // error starting a monitor
      logMessage("ccError", callControl.sent);
   }
   else if (request == "SingleS") { // error in transfer
      tryAgain.Start();
      ccHangup();
   } // feel free to handle other errors here
}
--> </script>
```

Putting it all together:

```
<html>
...
<body>
// data section here
<div xmlns="http://www.saltforum.org/2002/SALT" style="visibility:hidden">
// put the speech objects here
</div>
// speech event handlers here
// call control section here
// finally, when the page is loaded...
<script>
ccStartListening();
</script>
</body>
</html>
8 CCXML/CSTA XML Programming Example
The following example demonstrates the use of ECMA-323 with CCXML. The example shows how
the ECMA-323 services and events used in the previous example are created and exposed via
CCXML.
<?xml version="1.0" encoding="UTF-8"?>
<ccxml version="1.0"
xmlns="http://www.w3.org/2002/09/ccxml"
xmlns:ccxml="http://www.w3.org/2002/09/ccxml"
xmlns:csta="http://www.ecma.ch/standards/ecma-323/csta/ed2">
<var name="state0"/>
<!-- CSTA Vars -->
<var name="callID"/>
<var name="deviceID"/>
<eventhandler statevariable="state0">
<transition event="ccxml.loaded">
<!-- inline ECMA-323 -->
<csta:MonitorStart>
<monitorObject>
<deviceObject>
9999
</deviceObject>
</monitorObject>
</csta:MonitorStart>
<!-- ECMA-323 as a external event source -->
<script>
var monitorRequest =
    <MonitorStart xmlns='http://www.ecma.ch/standards/ecma-323/csta/ed2'>
```

```
      <monitorObject><deviceObject>9999</deviceObject></monitorObject>
    </MonitorStart>;
</script>
<send dest="http://www.csta-webservice.com/ecma323" name="sendRequest"
 namelist="monitorRequest" />
</transition>
<transition event="csta.DeliveredEvent" name="evt">
<!-- Simple CCXML style shortcut -->
<accept/>
<!-- save off event props -->
<!-- simple object mapping -->
<assign name="callID" expr="evt.connection.callID"/>
<assign name="deviceID" expr="evt.connection.deviceID"/>
<!-- DOM API -->
<script>
callID.value = XML(evt).connection.callID;
deviceID.value = XML(evt).connection.deviceID;
</script>
<!-- Big inline ECMA-323 style request -->
<AnswerCall xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
<callToBeAnswered>
<callID> <ccxml:value expr="evt.connection.callID"/> </callID>
<deviceID> <ccxml:value expr="evt.connection.deviceID"/>
</deviceID>
</callToBeAnswered>
</AnswerCall>
<!-- ECMA-323 as a external event source -->
<script>
var answerRequest =
      <AnswerCall xmlns='http://www.ecma.ch/standards/ecma-323/csta/ed2'>
        <callToBeAnswered><callID>{evt.connection.callID}</callID>
        <deviceID>{evt.connection.deviceID}</deviceID></callToBeAnswered>
      </AnswerCall>;
</script>
<send dest="http://www.csta-webservice.com/ecma323" name="sendRequest"
namelist="answerRequest" />
</transition>
<transition event="csta.EstablishedEvent" name="evt">
<dialogstart src="'hello.vxml'"/>
</transition>
<transition event="dialog.exit">
<disconnect/>
<!-- Big inline ECMA-323 style request -->
<ClearConnection xmlns="http://www.ecma.ch/standards/ecma-323/csta/ed2">
<connectionToBeCleared>
<callID><ccxml:value expr="callID"/></callID>
<deviceID><ccxml:value expr="deviceID"/></deviceID>
</connectionToBeCleared>
</ClearConnection>
<!-- ECMA-323 as a external event source -->
<script>
var clearRequest =
    <ClearConnection xmlns='http://www.ecma.ch/standards/ecma-323/csta/ed2'>
      <connectionToBeCleared>
        <callID>{evt.connection.callID}</callID>
        <deviceID>{evt.connection.deviceID}</deviceID>
      </connectionToBeCleared>
    </ClearConnection>;
</script>
<send dest="http://www.csta-webservice.com/ecma323" name="sendRequest"
  namelist="clearRequest" />
</transition>
<transition event="csta.ConnectionClearedEvent" name="evt">
<exit/>
</transition>
<transition event="error.*" name="evt">
<log expr="'an error has occured (' + evt.error + ')'"/>
<exit/>
</transition>
</eventhandler>
</ccxml>
```