Special Lecture (406)
Spoken Language Dialog Systems
JavaScript for (HTML and) VoiceXML

Rolf Schwitter
{schwitt}@ics.mq.edu.au
Today’s Program

• What is ECMAScript?
• What is JavaScript?
• What is JavaScript good for?
• JavaScript for HTML
• JavaScript for VoiceXML
What is ECMAScript?

- ECMAScript\(^1\)
  - is a scripting language specification, and
  - not a scripting language.

- The ECMAScript specification
  - is an international standard
  - was developed retrospectively
  - was based around version 1.1 of JavaScript.

\(^1\) ECMA: European Computer Manufacturer’s Association.
What is JavaScript?

- JavaScript is an object-oriented scripting language.
- Client-side JavaScript is
  - an implementation of ECMAScript
  - usually embedded directly in HTML pages
  - interpreted.
- Server-side JavaScript is
  - used with Web servers
  - proprietary and vendor-specific
  - interpreted or compiled.
What is JavaScript good for?

- Basically JavaScript is a scripting language for HTML designers.
- JavaScript language has a very simple syntax.
- JavaScript can
  - put dynamic text into an HTML page,
  - react to events,
  - read and write HTML elements,
  - be used to validate data.
Are JavaScript and Java identical?

• No!
• JavaScript and Java are two completely different things!
• JavaScript is a lightweight scripting language.
• Java is a powerful programming language.
• Java belongs to the same category as C and C++. 
• The `<script>` tag is used to embed JavaScript code into HTML:

```html
<html>
  <head>
    <title> JavaScript Example 1 </title>
  </head>
  <body>
    <script type = "text/javascript">
      document.write("Hello World!")
    </script>
  </body>
</html>
```
Non-JavaScript Browsers

- Non-JavaScript browsers will display the script as page content.
- To prevent them from doing this, use the HTML comment tag:

  `<script type = "text/javascript">
     <!--
     some statements
     //-->
  </script>`

- Note `//` is a JavaScript comment symbol.
Executing a JavaScript

- A JavaScript in the head section of an HTML page will be executed
  - when it is called
  - when an event is triggered.
- A JavaScript in the body section of an HTML page will be executed
  - while the page loads.
Calling an external JavaScript

```html
<html>
<head>
    <title>JavaScript Example 2</title>
</head>
<body>
    <script src="external.js"> </script>
</body>
</html>

```
Variables

• Variable names
  – are case sensitive
  – must begin with a letter or the underscore character.

• You can
  – declare a variable with or without a "var" statement, and
  – assign a value in the following ways

    var myname = "Martin"  Or  myname = "Martin"
Functions

- Basically functions are a way for bundling commands together.
- Functions that will return a result must use the `<return>` statement:

```javascript
function total(a,b) {
    result = a + b
    return result
}
```
- Calling a function:

```javascript
sum = total(5,7)
```
A Function at Work

```html
<html>
<head>  
<title> JavaScript Example 3 </title>  
</head>  
<body>
  <script type = "text/javascript">
    <!--
    function calculation() {
      var x = 12;
      var y = 5;
      var result = x + y;
      document.write(result);
    }
    calculation()
    //-->
  </script>
</body>
</html>
```
<html>
<head>
  <script type="text/javascript">
  <!--
    function calculation() {
      var x = 12;
      var y = 5;
      var result = x + y;
      alert(result);
    }
    //-->  
  </script>
</head>
...
Combining Functions & Events

```html
<body>
  <form>
    <input type = "button" 
           value   = "Calculate"
           onClick = "calculation()"/>
  </form>
</body>
</html>
```
Combining Functions & Events

![Diagram of a computer interface with a button labeled 'Calculate']

![Message box with an exclamation mark and the number 17]
A Recursive Function at Work

```html
<html>
<head>
    <title>JavaScript Example 4</title>
    <script type = "text/javascript">
        <!--
        var n = 5;
        function factorial(n) {
            return (n <= 1) ? 1 : n * factorial(n-1);
        }
        //-->  
    </script>
</head>
...

"?:" If first operand it true, use the second operand, else use the third operand.
A Recursive Function at Work

<body>
  <h3>Factorial</h3>
  <script type="text/javascript">
    <!--
    document.write("The result is: ", factorial(n));
    //-->
    </script>
</body>
Objects

- Objects are based on functions called constructor functions:
  ```javascript
  function Animal(type, sound, movement) {
    this.animal_type = type;
    this.animal_sound = sound;
    this.animal_movement = movement;
  }
  ```

- Creating object from constructor functions:
  ```javascript
  myDog = new Animal("dog", "woof", "walk/run");
  myDog.animal_size = "fat"; 
  ```
JavaScript for Form Validation

Enter your e-mail:  
Enter your code, value from 1 to 5:  
Enter your first name, max 10 letters:  

Send input
JavaScript for Form Validation

<html>
<head><title> JavaScript Example 5 </title>
<script type = "text/javascript">
function validate() {
    form = document.myForm
    at = form.myEmail.value.indexOf("@")
    code = form.myCode.value
    firstname = form.myName.value
    submitOK = "True"
...
</script>
</head>
</html>
if (at == -1) {
    alert("Not a valid e-mail")
    submitOK = "False"
}
if (code < 1 || code > 5) {
    alert("Your code must be between 1 and 5")
    submitOK = "False"
}
if (firstname.length > 10) {
    alert("Your name must be less than 10 letters")
    submitOK = "False"
}
if (submitOK == "False") {
    return false
}
</script>
</head>
<body>
    <form name="myForm"
        action="submitdata.html"
        onsubmit="return validate()">
        <pre>
            Enter your e-mail:
            <input type="text" name="myEmail">
            Enter your code, value from 1 to 5:
            <input type="text" name="myCode">
            Enter your first name, max 10 letters:
            <input type="text" name="myName">
            <br>
            <input type="submit" value="Send input">
        </pre>
    </form>
</body>
JavaScript and VoiceXML

• VoiceXML has very few programming features of its own.
• JavaScript is the required scripting language for VoiceXML.
• VoiceXML documents can contain JavaScripts in two contexts:
  – as values of many attributes
  – as arbitrary code in <script> elements.
• VoiceXML documents can refer to external JavaScripts via the "src" attribute of the <script> element.
Special Characters

- JavaScript has 3 characters which have also meaning in VoiceXML:

<table>
<thead>
<tr>
<th>Character</th>
<th>Escape sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>&amp;</td>
<td>&amp;</td>
</tr>
</tbody>
</table>

- You must escape these characters or wrap them in a CDATA section.

```xml
<![CDATA[ JavaScript Statements ]]>```

© Macquarie University 2004
JavaScript Expressions as Attributes

- Many VoiceXML attributes use JavaScript expressions directly.
- For instance, the "expr" attribute of elements such as
  - `<var>`
  - `<field>`
  - `<assign>`
- For example:

```xml
<var name = "one" expr = "1"/>
<field name = "two" expr = "one + 1">
<assign name = "result" expr = "Math.sqrt(a)"/>
```
Example

- The following example assigns a value to the variable "ui_path":

  `<var name = "ui_path">`  
  `<assign name = "ui_path" expr = "'ui/'" />`

- The next assignment references "ui_path" and uses the JavaScript concatenation operator "+":

  `<assign name = "intro_path" expr = "ui_path + 'welcome.wav'" />`
<block>
  <var name = "d" expr = "new Date()" />
  <var name = "iHour" expr = "d.getHours()" />
  <var name = "iMin" expr = "d.getMinutes()" />

  <if cond = "iHours == 0 &amp;&amp; iMin == 0">
    <audio src = "ui/time/midnight.wav" />
  </if>

  <elseif cond = "iHours == 12 &amp;&amp; iMin == 0" />
    <audio src = "ui/time/noon.wav" />

  <elseif cond = "iHour &lt; 12" />
    <audio src = "ui/time/am.wav" />

  <else />
    <audio src = "ui/time/pm.wav" />
  </else>

</block>
Another Example

- The example uses a JavaScript expression in a condition to determine whether to play a recorded file representing "midnight," "noon," "am," or "pm".
- It uses the "Date" object that copies the date and time from the local machine into a specified object name.
JavaScript within Script Elements

• A <script> element may occur
  – in the <vxml> and <form> elements, or
  – in executable content (in <filled>, <if>, <block>, <catch>).
• The VoiceXML <script> element does not have a language type attribute.
Evaluating JavaScripts

- A JavaScript in the `<vxml>` element is evaluated just after the document is loaded, along with the `<var>` elements, in document order.

- A JavaScript in the `<form>` element is evaluated in document order, along with `<var>` elements and form item variables, each time execution moves into the `<form>` element.

- A `<script>` element in executable content is executed, like other executable elements, as it is encountered.
Accessing Variables

- A variable declared in VoiceXML is accessible from JavaScript:

```xml
<block>
  <var name = "iCurrentMonth" />

  <script>
    var d = new Date();
    iCurrentMonth = d.getUTCMonth();
  </script>

  <prompt>
    <audio src = "curmonth.wav"/>
    <!-- The current month is ... -->
    <audio expr = "'ui/months/' + iCurrentMonth + '.wav'"/>
  </prompt>
</block>
```
Accessing Variables

- In the previous example the VoiceXML variable is set to the numeric equivalent of the current month.
- The month is played back to the user by referencing an audio file.
- An application developer using this code would create audio files with names that correspond to the numbers returned by the "getUTCMonth" method of the "Date" object.
The factorial Function in VoiceXML

- The following function "factorial" accepts an argument "n" and use a "return" statement to return a value back.

```xml
<script> <![CDATA[
    function factorial(n)
    {
        return (n <= 1) ? 1 : n * factorial(n-1);
    } ]]> </script>
```
Example: Factorial Dialog

- Computer: Tell me a number and I'll tell you its factorial.
- Caller: 5
- Computer: The factorial of 5 is 120.
Factorial in VoiceXML

```xml
<?xml version = "1.0">  
<vxml version = "2.0">  

<script> <![CDATA[

    function factorial(n)
    {
        return (n <= 1)? 1 : n * factorial(n-1); 
    }

</script>

```
Factorial in VoiceXML

<form id="form">
  <field name="n">
    <grammar type="application/srgs+xml"
      src="/grammars/number.grxml"/>
    <prompt>
      Tell me a number and I'll tell you its factorial.
    </prompt>
    <filled>
      <prompt> The factorial of <value expr="n"/>
        is <value expr="factorial(n)"/>.
      </prompt>
    </filled>
  </field>
</form>
Another Example: Factorial Dialog

Computer: Say a number.
Caller: 3
Computer: The factorial of 3 is 6.
This is the first factorial that has been computed.
Computer: Say a number.
Caller: 5
Computer: The factorial of 5 is 120.
The last factorial that has been computed was 5.
Computer: Say a number.
Factorial in VoiceXML

...<form id = "form">
   <var name = "lastresult"
       expr = "'This is the first factorial that has been computed.'"/>

   <field name = "n" type = "number">
      <prompt> Say a number. </prompt>

</form>
<filled>
   <prompt> The factorial of <value expr = "n"/> is
   <value expr = "factorial(n)"/>
   <value expr = "lastresult"/>
   </prompt>
   <assign name = "lastresult"
   expr = "'The last factorial
           that has been computed was ' + n + '.'" />
   <clear namelist = "n"/>
</filled>
</field>
</form>
A String Object Tutorial in VoiceXML

- The following tutorial demonstrates some string object methods.

```javascript
Prompt >> Welcome to the Java Script string tutorial.
     After each test, please say "next" to continue. <<

Prompt >> Testing "char at" <<

Prompt >> My string is Catfish seeking the character at three
     returns f. <<
```
Tutorial in VoiceXML

```xml
<?xml version = "1.0" ?>
<vxml version = "2.0">

<form id = "test_string">

<var name = "myString"/>
<var name = "myString2"/>

<script>
  <![CDATA[
    myString = new String("Catfish");
    myString2 = new String(" Grouper Minnow Tadpole Cod");
  ]]> 
</script>

</form>
</vxml>
```
Welcome to the <emp>Java Script</emp> string tutorial. After each test, please say "next" to continue.
charAt()

<block> Testing "char at" </block>

<field name = "test_char">
  <prompt>
    My string is <value expr = "myString"/>
    seeking the character at three
    returns <value expr = "myString.charAt(3)"/>
  </prompt>
  <grammar type = "application/x-gsl">
    [next continue]
  </grammar>
  <filled> <prompt> Ok </prompt> </filled>
</field>
<block> Testing "concat" </block>

<field name = "test_concat">
    
    <prompt>
    My string is <value expr = "myString"/>.
    My second string is <value expr = "myString2"/>.
    Concatenated they are
    <value expr = "myString.concat(myString2)"/>.
    
    </prompt>

    <grammar type = "application/x-gsl">
        [next continue]
    </grammar>

    <filled> <prompt> Ok </prompt> </filled>

</field>
<block> Testing "length" </block>

<field name = "test_length">
    <prompt>
        My string is <value expr = "myString"/>.
        Its length is <value expr = "myString.length"/>.
    </prompt>
    <grammar type = "application/x-gsl">
        [next continue]
    </grammar>
    <filled> <prompt> Ok </prompt> </filled>
</field>
<block> Testing "split" </block>

<field name = "test_split">

<prompt>
    My string two is <value expr = "myString2"/>
    Splitting on white space
    returns <value expr = "myString2.split(' ')"/>
    My string two is still <value expr = "myString2"/>
</prompt>

</field>

<grammar type = "application/x-gsl">
    [next continue]
</grammar>

<filled> <prompt> Ok </prompt> </filled>

</field>
replace()

[block] Testing "replace" [/block]
[field name = "test_replace">
<prompt>
My string is <value expr = "myString"/>.
Replacing "ay" with "ee"
returns <value expr = "myString.replace('a','e')"/>.
My string is still <value expr = "myString"/>.
</prompt>
[grammar type = "application/x-gsl">
[next continue]
</grammar>
[filled] <prompt> Ok </prompt> </filled>
[/field]
End Tutorial

<block> Good-bye </block>
</form>
</vxml>
Regular Expressions

• Execute a search.

```<script>
  <![CDATA[
    var re = /d(b+)(d)/ig;
    var s = new String("cdbBdbbsbz");
    var result = re.exec(s);
  ]]>  
</script>```
Regular Expressions

• Test for a match.

<script>
  <![CDATA[
    var s = new String("salary supplementation");
    var re = /salary/;
  ]]> 
</script>

<value expr = "re.test(s)"/>
Example: Say as Digits

```javascript
<![CDATA[
function sayasDigits(number) {
    var digitNumber = number.charAt(0);
    for(var i = 1; i < number.length; i++) {
        digitNumber += ' ' + number.charAt(i);
    }
    return digitNumber;
}
]]>
</script>
Speech Synthesis Markup Language

• All written languages have special constructs that require a conversion of the written form into the spoken form.

• Text normalization is an automated process of the synthesis processor that performs this conversion.

• The `<say-as>` element allows the author to specify the explicit pronunciation of words (if implemented).

• For example:

```html
<say-as interpret-as = "digits"> 123 </say-as>
```

Output: One two three
Take-Home Message

• JavaScript is a scripting language suitable for HTML and VoiceXML.
• JavaScripts are embedded via the <script> tag.
• VoiceXML has very few programming features of its own.
• VoiceXML uses JavaScript as scripting language.
• Variables declared in VoiceXML are accessible from JavaScript.
• For more details about JavaScript check out:
  – http://devedge.netscape.com/central/javascript/