Intorduction to VoiceXML (2)

Main Topics

- Declearing variable and reteieving values
- Variable scopes
- Branching elements
- Math functions in JavaScript

Variables

- Rather than hardcoding strings and numbers throughout a program, variables provide a useful alternative that helps increase the reusability and maintainability of an application.
- Variables are a basic feature of most programming languages.
- A variable provides a mechanism for the programmer to store and retrieve data by name.
- VoiceXML supports the declaration, the setting, and the retrieval of variables.

Variable Types

- Variables typically store data of a particular type.
- Basic data types include strings, numbers, or boolean (logical) values.
- Most programming languages support the creation of a collection or list of variables of the same type known as an array.
- Sophisticated programming languages allow the programmer to create new data types composed of groupings of the more basic data types. These are known as data structures.

Variable Declaration

```
<var name="user"/>
```

- Declare a variable by using <var>

```
<var name="user" expr="'George W. Bush'"/>
```

- Declare a variable and intialize it.
- Note the two sets of quotes, single, and double

```
<field name="user">
…
</field>
```

- Declare a field item variable when declaring a field in a form

Example 1

```
<vxml version="2.0">
  <form>
    <block> The president of the United States is
        <var name="user" expr="'George W. Bush'" />
        <prompt>  <value expr="user"/>
        </prompt>
    </block>
  </form>
</vxml>
```

Example 2

```
<vxml version="2.0">
  <form>
    <field name="ph" type="phone">
        <prompt>
           Thank you for calling National
             Chi Nan University.
           What is your phone number?
        </prompt>
        <filled>
           <prompt>
             You say your phone number is
              <value expr="user"/>
           </prompt>
        </filled>
    </field>
  </form>
</vxml>
```

<assign> Element

- After a variable is declared, it can be assigned a new value

```
<assign name="user" expr="'Bill Clinton'"/>
```

- This overwirtes the old value of the variable (if it has one)

Variable Scope

- A variable can be declared in five different scopes:
  - Anonymous
  - Dialog
  - Document
  - Application
  - Session

Anonymous Scope

- When a variable is declared inside a <block>, <filled>.
- The variable can only be used inside that particular element.

```
<vxml version="2.0">
<form>
   <block>
      <!-- Delcare the variable user. -->
      <var name="user" expr="'Harrison Ford'"/>
      <prompt>
         Hi, <value expr="user"/>
      </prompt>
   </block>
</form>
</vxml>
```

Out of Scope

```
<vxml version="2.0">
<form>
   <block>
      <var name="user" expr=“Jackie Chan'"/>
   </block>
   <block>
      <prompt>
         <!-- user can not be accessed here. -->
         Hi, <value expr="user"/>
      </prompt>
   </block>
</form>
</vxml>
```

Dialog Scope

- When a variable is decalred inside a <form> or <menu> element
  - Using the `name` attribute, or
  - A child `<var>` element
- A dialog-scoped variable can be used anywhere inside the dialog in which it was declared, but cannot be accessed outside that dialog

Example 3

```
<vxml version="2.0">
  <form>
    <!-- delcare the variable user in a dialog -->
    <var name="user" expr="'Harrison Ford'" />

    <field name="answer" type="boolean">
       <prompt>
       <!-- user has dialog scope.  It can be accessed here. -->
       <value expr="user"/>, is that true?
       </prompt>
    </field>

    <block>
        <prompt>
           <!-- answer has dialog scope.  It can be accessed here. -->
           You told me <value expr="answer"/>
        </prompt>
    </block>
  </form>
</vxml>
```

Document Scope

- When a variable is declared by a <var> element that is a child of the <vxml> root element.

- A variable with document scope can only be accessed from within the same document:

```
<vxml version="2.0">
<var name="user" expr="'Jackie Chan'"/>
<form>
   <block>
      <prompt>
         Hi, <value name="user"/>.
         This variable can be used anywhere
          in this document.
      </prompt>
   </block>
</form>
</vxml>
```

Application Scope

- When a variable is declared by a <var> that is a child of the <vxml> element in the root document.
- It is available to any documents of the application.

Overlapping Scopes & Hidden Variables

- If two variables in different scopes are declared to have the same name, the VoiceXML interpreter will use the one that has the narrower scope.

```
<vxml version="2.0">
<form>
   <!-- user has a dialog scope. -->
   <var name="user" expr="'Jackie Chan'"/>
   <block>
      <!-- user has an anonymous scope. -->
      <var name="user" expr="'Jet Li'"/>
      <prompt>
        The user is <value name="user"/>.
        The dialog user is <value name="dialog.user"/>.
      </prompt>
   </block>
</form>
</vxml>
```

The document plays "The user is Jet Li", not "The user is Jackie Chan".

"The user is Jackie Chan".

Expression

- VoiceXML follows the expression syntax of JavaScript.

- An expression is a representation that evaluates to a value.
- The expr attribute works with many VoiceXML elements including `<assign>` and `<value>`.
  - The value of the `expr` attribute can be any valid JavaScript expression.
    - ☞"3.14"
    - ☞"'Jackie Chan'"
    - ☞'true"
    - ☞"var * 2 * 45"
    - ☞"var != 0"
    - ☞"var + ' ' + 'Jackie Chan'"
    - ☞"var1 > 0 && var2 < 0"

Example 4

```
<vxml version="2.0">
  <form>
    <var name="a" expr="10"/>
    <var name="b" expr="20"/>
    <var name="c" expr="a + b" />
    <block>
      <prompt>
         summation of
         <value expr="a"/>
         and
         <value expr="b"/>
         is equal to
         <value expr="c"/>
      </prompt>
    </block>
  </form>
</vxml>
```

Math Functions

- http://javascript-reference.info/#math
  - abs()
  - max()
  - min()
  - random()
  - ceil()
  - round()
  - floor()
  - sqrt()
  - pow()
  - E
  - PI

Example 5

```
<vxml version="2.0">
  <form>
    <block>
      <prompt>
      <assign name="a" expr="Math.ceil(Math.random()*10)"/>
      <assign name="b" expr="a * a"/>
      <assign name="c" expr="Math.sqrt(b)"/>
      The square root of <value expr="b"/> is <value expr="c"/>.
      </prompt>
      <exit/>
    </block>
  </form>
</vxml>
```

Example 6

```
<vxml version="2.0">
  <form>
    <var name="a" expr="Math.ceil(Math.random()*6)"/>
    <var name="b" expr="Math.ceil(Math.random()*6)"/>
    <var name="c" expr="a + b" />
    <block>
      <prompt>
         <value expr="a"/>
         plus
         <value expr="b"/>
         is equal to
         <value expr="c"/>
      </prompt>
      <exit/>
    </block>
  </form>
</vxml>
```

Branching Elements

- <if>
- <else>
- <elseif>

<if>…<else>…

```
<vxml version="2.1" xmlns="http://www.w3.org/2001/vxml">
  <form>
    <field name="famous" type="boolean">
        <prompt> would you like to be famous?
        </prompt>

        <filled> got it!
        <if cond="famous">
            let's schedule an audition
            <goto next="schedule.vxml" />
        <else /> infamous is a reasonable alternative.
            <goto next="infamous.vxml" />
        </if>
        </filled>
    </field>
  </form>
</vxml>
```

The value of the attribute `cond` evaluates to a Boolean value `true` or `false`.

Example 7

```
<vxml version="2.0">
<form>
   <!-- Assign a random number to the variable -->
   <var name="number" expr="Math.random()"/>
   <block>
      <!-- Access number in cond attribute -->
      <if cond="number ==0">
         <prompt>
         The value of the variable is zero.
         </prompt>
      <else/>
         <prompt>
            <!-- Access number in expr attribute -->
            The value of the non-zero variable
            is <value expr="number"/>.
         </prompt>
      </if>
   </block>
</form>
</vxml>
```

The value of the attribute `cond` evaluates to a Boolean value `true` or `false`.