A Game in XForms

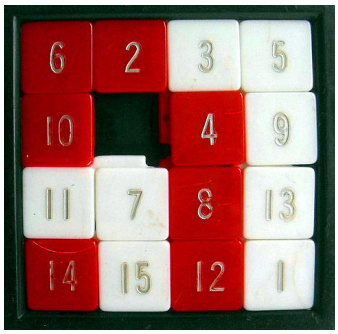
May 20, 2019

Steven Pemberton (/authors/steven-pemberton/)

Steven Pemberton continues his XForms series with an example of coding a game.

Introduction

My youngest asked me how you would program a game. I asked what sort of game, and he described the game where you have to slide numbered tiles around to get them in order.



Take some data

We start with four rows of four cells, containing the numbers 1-15 in any old order, plus one blank cell:

```
<instance>
  <data xmlns="">
    <row><cell>1</cell><cell>5</cell><cell> 8</cell><cell>12</cell></row>
    <row><cell>2</cell><cell>6</cell><cell> 9</cell><cell>13</cell></row>
    <row><cell>3</cell><cell>7</cell><cell>10</cell><cell>14</cell></row>
    <row><cell>4</cell><cell>.</cell><cell>11</cell><cell>15</cell></row>
  </data>
</instance>
```

Display it

We can display the numbers like so (with a suitable bit of CSS to format the cells, setting height and width and adding a border, not shown here):

```
<repeat ref="row">
    <repeat ref="cell">
        <output value="."/>
    </repeat>
</repeat>
```

which would look like this:

1	5	8	12
2	6	9	13
3	7	10	14
4	.	11	15

Source (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/game/game.xhtml)

Adding interaction

We want to be able to click on the tiles in order to move them, so we wrap the output of the number with a `trigger`. This is the similar to a `button` in HTML, except that XForms tries to be representation-neutral, and so avoids using naming that suggests a particular representation:

```
<repeat ref="row">
    <repeat ref="cell">
        <trigger appearance="minimal">
            <label><output value="."/></label>
        </trigger>
    </repeat>
</repeat>
```

The `appearance="minimal"` is an indication that you don't want it formatted as a button, just as regular text, but still *acting* as a button.

However, this trigger doesn't do anything yet. To achieve this we add an `action` within the trigger:

```
<trigger appearance="minimal">
    <label><output value="."/></label>
    <action ev:event="DOMActivate">
        ...
    </action>
</trigger>
```

This responds to the `DOMActivate` event on the `trigger`, which is the event that occurs when a trigger is clicked on.

What we want the action to do is copy the value in the clicked-on cell to the empty cell:

```
<setvalue ref="//cell[.='.']" value="context()"/>
```

and make the clicked-on cell empty:

```
<setvalue ref="." value="'.'"/>
```

In total:

```
<trigger appearance="minimal">
    <label><output value="."/></label>
    <action ev:event="DOMActivate">
        <setvalue ref="//cell[.='.']" value="context()"/>
        <setvalue ref="." value="'.'"/>
    </action>
</trigger>
```

This then looks like this (try clicking on the cells):

1	5	8	12
2	6	9	13
3	7	10	14
4	.	11	15

Source (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/game/game1.xhtml)

However, this allows you to click on *any* square, and we only want to allow swapping a square if the empty square is one of its (up to) four directly adjacent ones.

To do this, we add a condition to the action:

```
<action ev:event="DOMActivate" if="...">
```

The condition is the tricky bit. The preceding or following cell in a row is easy:

```
preceding-sibling::cell[1]='.' or
following-sibling::cell[1]='.'
```

The cell at the same position in the preceding or following row is slightly harder. The following row is:

```
../following-sibling::row[1]
```

We want to find the cell within that row at the same position:

```
../following-sibling::row[1]/cell[...position calculation here...]
```

The position in the row of the cell clicked on is one plus the number of preceding cells there are:

```
1+count(context()/preceding-sibling::cell)
```

So putting it together:

```
../following-sibling::row[1]/cell[1+count(context()/preceding-sibling::cell)]='.'
../preceding-sibling::row[1]/cell[1+count(context()/preceding-sibling::cell)]='.'
```

Which gives us our final game. You can try it out:

1	5	8	12
2	6	9	13
3	7	10	14
4	.	11	15

Source (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/game/game2.xhtml)

Moving the condition to the model

XForms splits processing in two parts, form and content: the model contains details of the data and its relationships, and the content contains the view onto the data and deals with interaction.

It is usually preferable to have as much of the data description as possible in the model. An example of this is the description of which cells may be clicked on. Having the condition on the action as we do now describes when the action can be carried out. However, we could move it to the model to describe which cells may be clicked on. We do this in the following way.

We add a `click` attribute to each cell. This will indicate whether the cell can currently be clicked on or not.

```
<instance>
  <data xmlns="">
    <row><cell click="">1</cell><cell click="">5</cell>...
```

etc.

We move the condition up to the model. Immediately after the instance, we add a bind:

```
<bind ref="row/cell/@click" type="boolean"
      calculate="..."/>
```

where the `calculate` attribute holds the condition, with one change since the context for this expression is slightly different: we are talking about the attribute, and on the action we were talking about the cell itself. So the condition has to be applied one level higher by adding a `"../"` before its elements.

The action in the content can now look like this:

```
<action ev:event="DOMActivate" if="@click = true()">
```

Since the `click` attribute now contains `true` or `false`, just for fun, and to demonstrate the subtle difference between describing which cells may be clicked on and when the action may be done, we can display the clickable cells differently by adding `class="cell {@click}"` to the cell output, and adding different formatting to the CSS for `class="true"`, thus making it visually obvious which cells are clickable:

```
.true { ... }
```

6	2	3	5
10	.	4	9
11	7	8	13
14	15	12	1

Source (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/game/game3.xhtml)

Change the output

"It would be more fun if it was like a jigsaw."

OK. Easy peasy. Almost a single change:

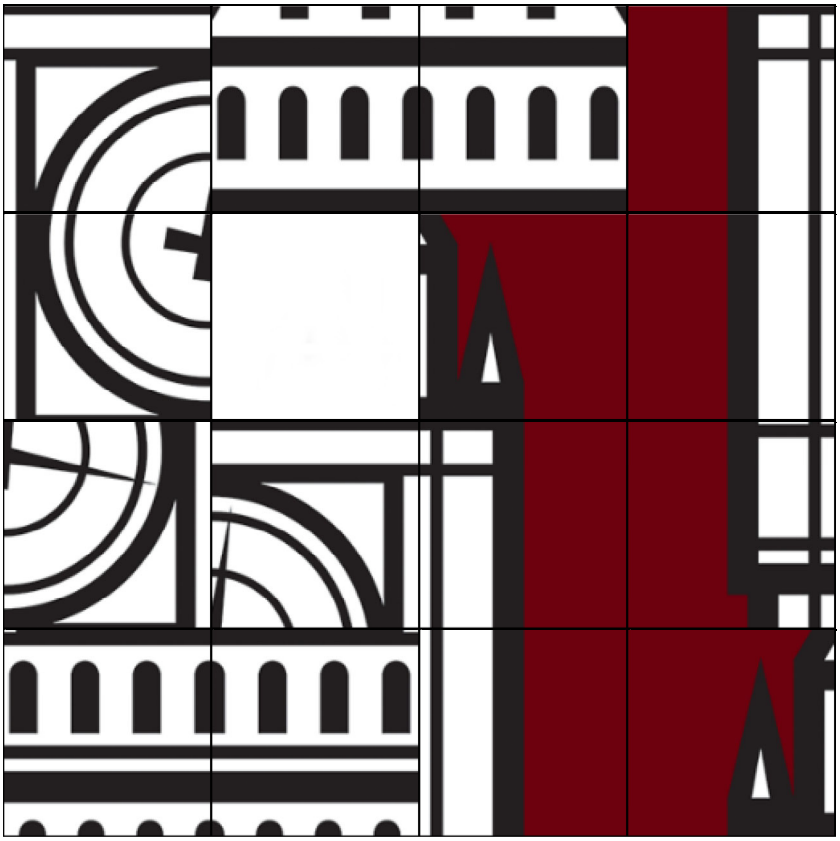
```
<output value="." mediatype="image/*"/>
```

This says, instead of outputting a value such as "1", interpret the 1 as a filename, and display that file as an image.

The only problem is that you are not allowed to use "." as a filename, so we'll have to catch that case, and we'll use "blank" as the filename instead:

```
<output value="if(.='.', 'blank', .)" mediatype="image/*"/>
```

That gives a version of the game as a sort of jigsaw (*audience gasps*):



Source (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/game/game4.xhtml)

Related links

- A Clock in XForms (/articles/2019/04/07/clock-xforms/)
- A Calendar in XForms (/articles/2019/03/07/calendar-xforms/)
- Viewing Data with XForms (/articles/2019/02/08/viewing-data-xforms/)
- An Introduction to XForms (/articles/2018/11/27/introduction-xforms/)
