A Calendar in XForms

Contents

A Month

A month consists of a number of weeks of seven days, displayed as a grid. In fact there can be up to six weeks displayed for a month, if the first day of the month falls on one of the two last days of the first week.

```
<instance>
  <cal xmlns="">
     <week><day/><day/><day/><day/><day/><day/><day/></week>
     <week><day/><day/><day/><day/><day/><day/><day/></week>
     <week><day/><day/><day/><day/><day/><day/><day/></week>
     <week><day/><day/><day/><day/><day/><day/><day/></week>
     <week><day/><day/><day/><day/><day/><day/><day/></week>
     <week><day/><day/><day/><day/><day/><day/><day/></week>
  </cal>
</instance>
```

Day numbers

To start off, we'll fill them starting from 1:

```
<bind ref="week/day"
      calculate="1 + 7*count(../preceding-sibling::week)
                   + count(preceding-sibling::day)"/>
```

This sets the day number to the week number times seven, plus the day number in the week, where both start counting from zero, plus 1. So the very first day will be numbered 1 and so on.

Display

(Plus a little bit of CSS, not shown here):

```
<repeat ref="week">
   <repeat ref="day">
      <output ref="."/>
   </repeat>
</repeat>
```

which looks like this:

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	32	33	34	35
36	37	38	39	40	41	42

Source

Next steps

1. Decide which month will be displayed;
2. find out what day of the week that month starts on;
3. move the numbers up so that day 1 is in the right place;
4. hide the days that shouldn't be displayed for the month;
5. make it pretty.

Add data

Add some data to the instance:

```
<instance>
  <cal xmlns="">
     <y/><m/>
     <monthlength/>
     <startday/>
     <week><day/>...
     ...
  </cal>
</instance>
```

- `y`, and `m`: the month we are interested in,
- `monthlength`: the length of that month in days,
- `startday`: the day of the week (from 0 to 6) that the month begins on.

We'll see how to calculate those shortly, but assume for now that that has been done.

Move day 1 up

So let's move the numbers up, so that day 1 falls on the right day of the week. To do that we change the bind that calculates the day number, so that it also subtracts the start day:

```
<bind ref="week/day"
      calculate="1 + 7*count(../preceding-sibling::week)
                   + count(preceding-sibling::day)
                   - /cal/startday"/>
```

Suppose the month starts on day 3 of the week, then this is what it looks like:

-2	-1	0	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	32
33	34	35	36	37	38	39

Start day

3

Source

Unwanted days

If the day number is less than 1, or more than the length of the month, then output nothing, and otherwise the number.

&#xa0; is a non-breaking space, and is needed because without it, on some browsers the CSS layout gets messed up.

```
<repeat ref="week" class="week">
   <repeat ref="day">
      <output value="if(. &lt; 1 or . &gt; /cal/monthlength, '&#xa0;', .)"
              class="day"/>
   </repeat>
</repeat>
```

It looks like this now:

			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Start day	**Month length**
3	31

Source

Unwanted weeks

You can see that there's now a blank week at the end; we can easily get rid of that with a bind:

```
<bind ref="week" relevant="day[1] &lt;= /cal/monthlength"/>
```

which says that a week is only relevant if its first day is less than or equal to the month length. Non-relevant values are never displayed:

			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Start day	**Month length**
3	31

Source

Calculating the Month and its Values

Let's start off initially with displaying the current month. To get that, we use the function `local-date()`, which returns the current date (plus how many hours different your timezone is from UTC):

2018-09-07+02:00

We want to extract the year and month from this string, and we want to set it on initialisation of the form.

Source

Initialising

So we add the following `action` to respond to the `xforms-ready` event:

```
<action ev:event="xforms-ready">
   <setvalue ref="y" value="substring-before(local-date(), '-')"/>
   <setvalue ref="m"
             value="substring-before(substring-after(local-date(), '-'), '-'
</action>
```

So the year is the substring before the first hyphen, and the month is the substring after the first hyphen, and before the second.

Month length

We add the length of each month to the instance:

```
<ml>31</ml><ml>28</ml><ml>31</ml><ml>30</ml><ml>31</ml><ml>30</ml>
<ml>31</ml><ml>31</ml><ml>30</ml><ml>31</ml><ml>30</ml><ml>31</ml>
```

But February changes in leap years. So we add `leap` to the instance, and calculate if the current year is a leap year (if the year is divisible by 4, but not by 100, or divisible by 400):

```
<bind ref="leap"
      calculate="if(((../y mod 4 = 0) and (../y mod 100 != 0)) or (../y mod
```

and calculate February accordingly:

```
<bind ref="ml[2]" calculate="28+../leap"/>
```

Month length

This now allows us to calculate the month length of the current month:

```
<bind ref="monthlength" calculate="../ml[position()=../m]"/>
```

			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

Year	**Month**	**Month length**	**Start day**
2018	09	30	3

(Remember that we haven't calculated the true start day for the month yet, but you can play around with it, to see how the display changes.)

Source

What Day of the Week does a Month Start On?

Answer: its day number in the year, less the 'Dominic' number for the year, plus a constant, all modulo 7 so that we get a value in the range 0 to 6.

The dominic number for any year is: 7 - ((year + floor(year ÷ 4) - century + floor(century ÷ 4) - leap) mod 7).

Well, we've already calculated `leap`. Let's do the rest.

Dominic

Add century and dominic to the instance:

```
<bind ref="century" calculate="floor(../year div 100)"/>
<bind ref="dominic"
      calculate="7-((../y+floor(../y div 4)-../century+floor(../century div
```

Now calculate the day in the year of the first of the month:

```
<bind ref="dayinyear"
      calculate="sum(../ml[position() &lt; ../m]) + 1"/>
```

That is, add the month lengths for all months before the current month, and add one.

The *real* start day of the month

```
<bind ref="startday" calculate="(11 + ../dayinyear - ../dominic) mod 7"/>
```

See it in action here:

					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Year	**Month**
2018	09

Month length	**Dominic**	**Day in year**
30	5	244

This is the first moment we have made any decision about which weekday is the first day of the week. In this case we have used Monday; however, if your week starts with a different day, it is easy to add the necessary offset (you change that 11 to adjust).

Source

What's Next?

Clearly, we need to add some decoration, like the names of the weekdays.

Controls to step through the calendar.

Year:

```
<trigger appearance="minimal"><label>→</label>
   <setvalue ref="y" value=". + 1" ev:event="DOMActivate"/>
</trigger>
```

Month:

```
<trigger appearance="minimal"><label>→</label>
   <action ev:event="DOMActivate">
      <setvalue ref="m" value="if(.=12, 1, . + 1)"/>
      <setvalue ref="y[../m=1]" value=". + 1"/>
   </action>
</trigger>
```

Today

I'll calculate it in a different way this time, just to show the options. Since `local-date()` has a fixed format, you can select fixed parts of the result:

```
<today><y/><m/><d/></today>
 ...
<bind ref="today">
   <bind ref="y" calculate="substring(local-date(), 1, 4)"/>
   <bind ref="m" calculate="substring(local-date(), 6, 2)"/>
   <bind ref="d" calculate="substring(local-date(), 9, 2)"/>
</bind>
```

There is a race condition in the calculation, since `local-date` gets called three times, and the date might just change between the calls. It will hardly ever happen, but it could. So better to add `local-date` to the instance as well, and call the function just once:

```
<today><date/><y/><m/><d/></today>
 ...
<bind ref="today">
   <bind ref="date" calculate="local-date()"/>
   <bind ref="y" calculate="substring(../date, 1, 4)"/>
   <bind ref="m" calculate="substring(../date, 6, 2)"/>
   <bind ref="d" calculate="substring(../date, 9, 2)"/>
</bind>
```

Displaying today specially

Now in the main output, we will add a special class on the output element:

```
<repeat ref="week" class="week">
   <repeat ref="day">
      <output value="if(. &lt; 1 or . &gt; /cal/monthlength, '&#xa0;', .)"
              class="{if(. = /cal/today/d and /cal/m = /cal/today/m and /cal
                         'today', 'day')}"/>
   </repeat>
</repeat>
```

which sets the class of the output to 'today' if it represents today's date, and otherwise to 'day', and we can use any CSS rules we want to display them differently:

← September→ ← 2018→

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Source

Events

We will have an instance to hold events, which we will load from an external source:

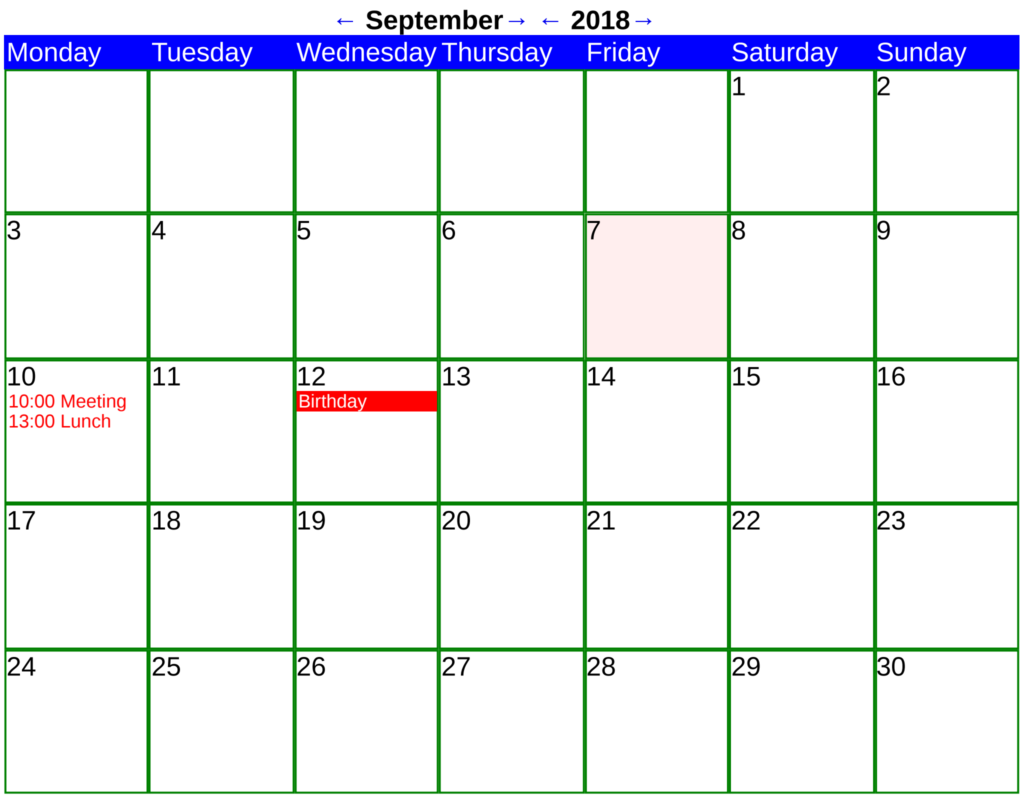
```
<instance id="events" src="events.xml"/>
```

This will hold our calendar events. For now a simple version:

```
<events>
   <event y="2018" m="3" d="20" t="10:00">Write program</event>
   <event y="2018" m="3" d="20" t="13:30">Meeting</event>
   <event y="2018" m="3" d="24">Clocks go forward</event>
   ...
</events>
```

Display with events

```
<repeat ref="week" class="week">
   <repeat ref="day">
      <group class="{if(. = /cal/today/d and /cal/m = /cal/today/m and /cal/
         <output value="if(. &lt; 1 or . &gt; /cal/monthlength, '&#xa0;', .)
         <repeat ref="instance('events')/event[@y=/cal/y and @m = /cal/m and
            <output class="event" value="concat(@t, ' ', .)"/>
         </repeat>
      </group>
   </repeat>
</repeat>
```



← September → ← 2018 →

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
					1	2
3	4	5	6	7	8	9
10 10:00 Meeting 13:00 Lunch	11	12 Birthday	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Actually I also added

Source

```
<output class="{if(@t='', 'dayevent', 'event')}" value="concat(@t, ' ', .)"/
```

to allow different styling for events with no time given.

There. Depending on what you count, 75 lines of XForms. A simple, direct implementation of a calendar viewer.