A Calendar in XForms

March 7, 2019

Steven Pemberton (/authors/steven-pemberton/)

Steven Pemberton continues his series on XForms with an example of building a calendar.

Introduction

As the name suggests, XForms was originally designed for dealing with forms. However, thanks to its generalised design it is suitable for much more.

In this example, we will show how to display a calendar.

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>
```

Now to fill in the day numbers. To start off, we'll fill them starting from 1:

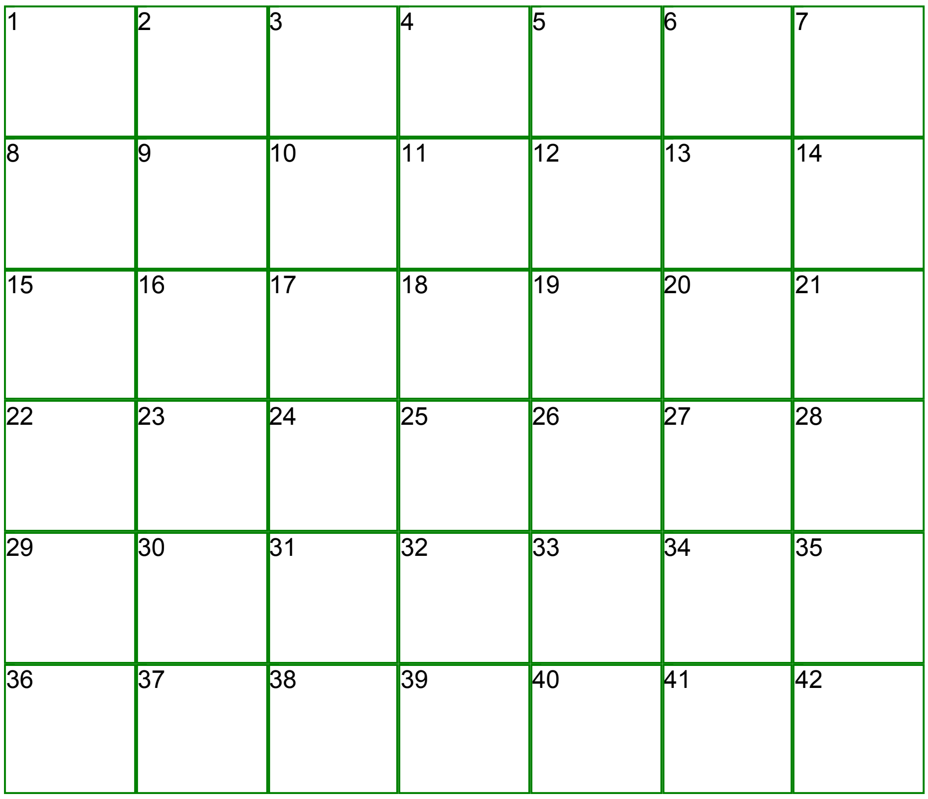
```
<bind ref="week/day"
      calculate="1 + 7*count(../preceding-sibling::week)
                   + count(preceding-sibling::day)"/>
```

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.

We can display this like this (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 (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/calendar/calendar1.xhtml)

With this basic structure, we now need to:

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.

So, add some data to the instance:

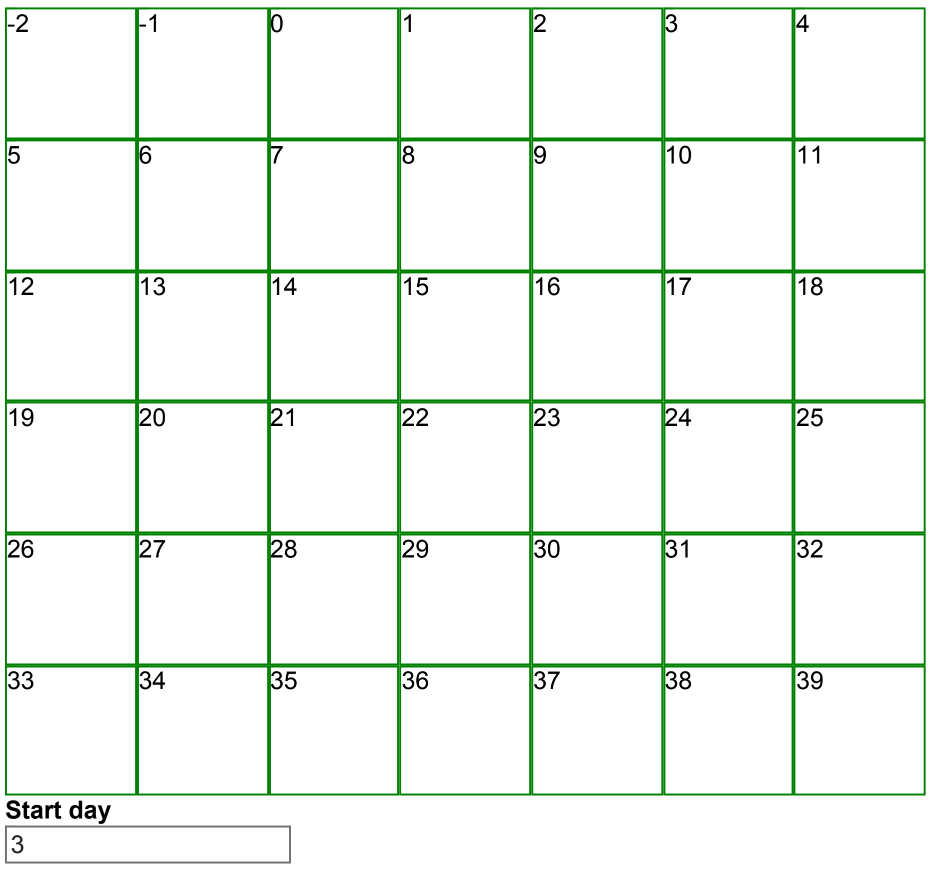
```
<instance>
  <cal xmlns="">
     <y/><m/>
     <monthlength/>
     <startday/>
     <week><day/><day/>...
     ...
  </cal>
</instance>
```

Clearly, `y`, and `m` will give the month we are interested in, `monthlength` will be the length of that month in days, and `startday` the day of the week (from 0 to 6) that the month begins on. We'll see how to calculate those shortly, but let's assume for now that that has been done.

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

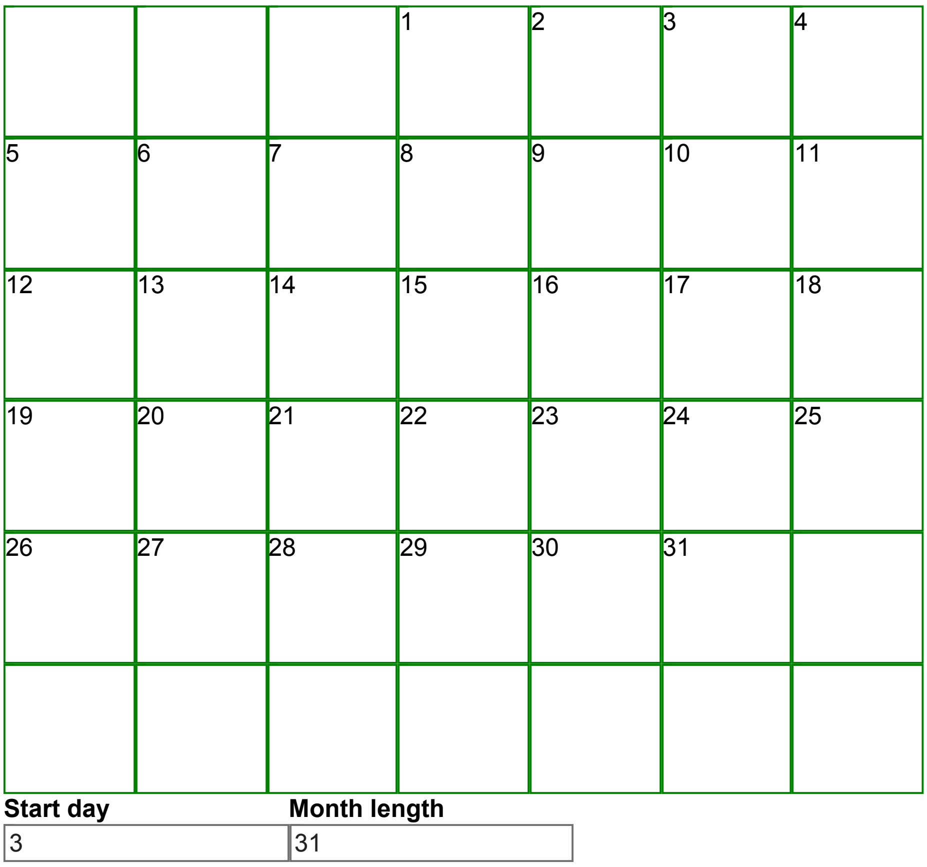
Source (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/calendar/calendar2.xhtml)

You can see that there's an input for the start day, so that you can play around with it.

To get rid of the days we don't want, we can now change the output:

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

which says that 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). 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 3

Month length 31

Source (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/calendar/calendar3.xhtml)

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 (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/calendar/calendar4.xhtml)

The inputs for the start day and month length let you play around with them to see what happens. For instance, try putting the start-day to 0 and the month length to 28.

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):

2019-12-03+01:00

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

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

Now to calculate the length of the month.

We can 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):

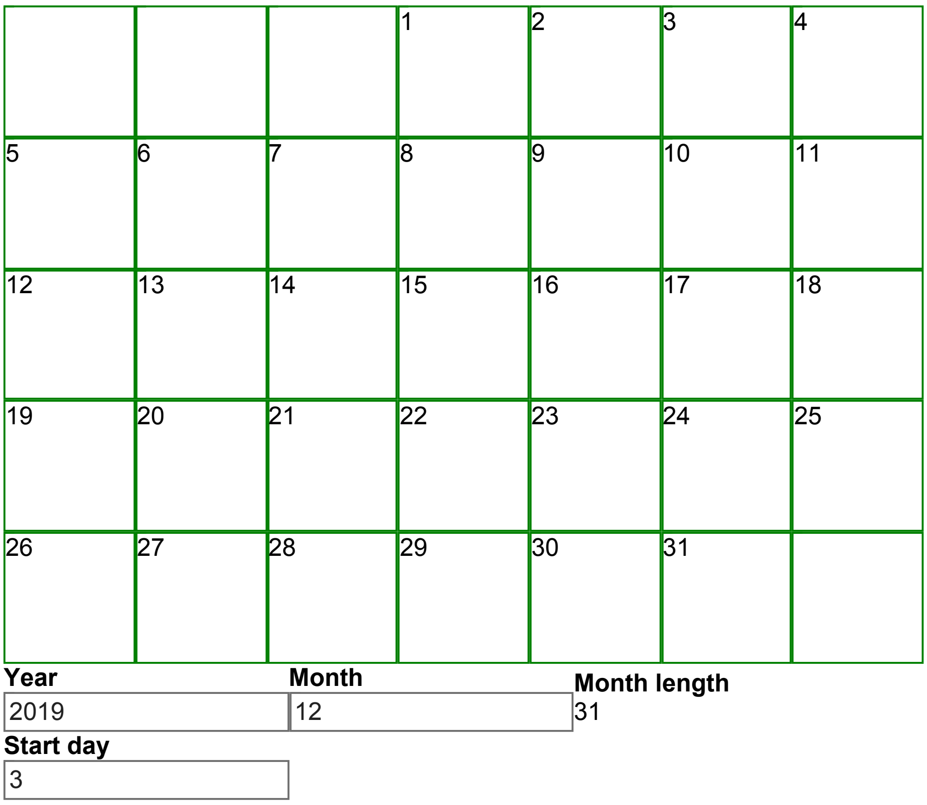
```
<bind ref="leap"
      calculate="if(((../y mod 4 = 0) and (../y mod 100 != 0)) or (..
```

and calculate February accordingly:

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

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	31	

Year 2019
Month 12
Month length 31
Start day 3

Source (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/calendar/calendar5.xhtml)

(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.)

What Day of the Week does a Month Start On?

OK, you're just going to have to believe me on this one. The day of the week that any day falls on is 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. Add century and dominic to the instance, and the two binds:

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

Now to calculate the day in the year. Add that to the instance, with the following bind:

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

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

Now we can calculate the *real* start day of the month:

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

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	31					

Year	**Month**
2019	12

Month length	**Dominic**	**Day in year**
31	4	335

Source (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/calendar/calendar6.xhtml)

You may have noticed that 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).

What's Next?

We now have the working infrastructure to display the calendar for any given month. Clearly, we need to add some decoration, like the names of the weekdays, and controls to step through the calendar, and I'll describe some of that at the end.

In the meantime, I will describe two other things: displaying today specially, and adding events to the calendar.

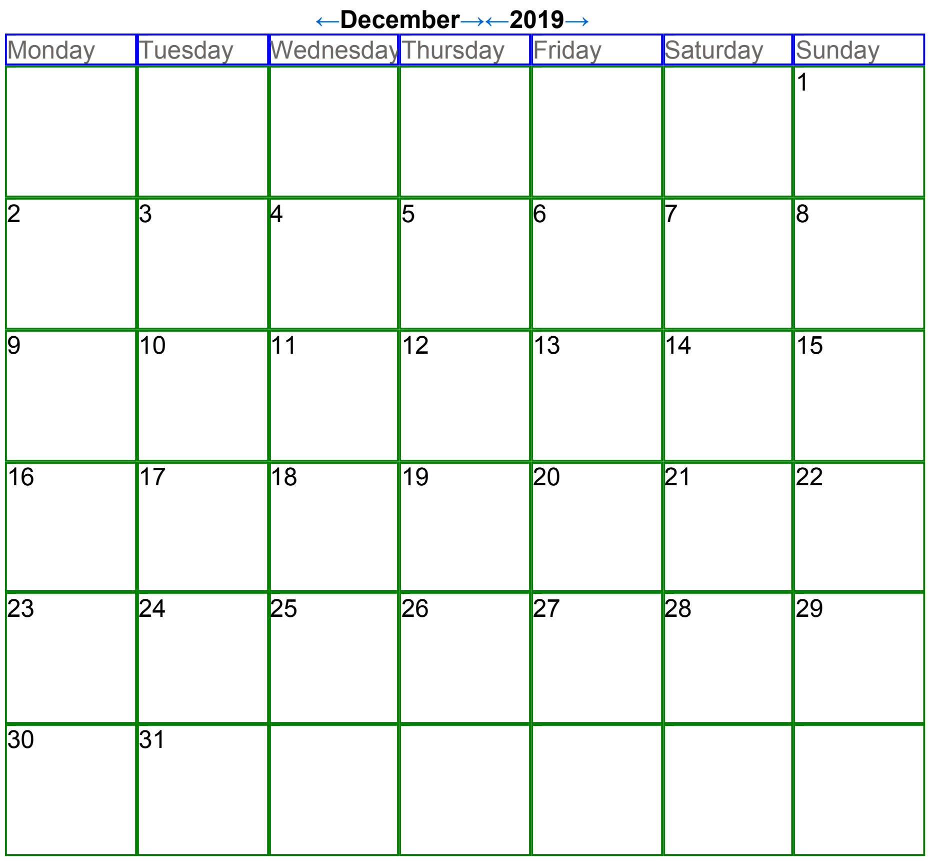
We already know how to get today's date, so let's add it to the instance. I'll calculate it in a different way this time, just to show the options. Since `local-date()` has a fixed format, you can select fixed parts of the result:

```
<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>
```

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 a
                         '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:



←December→←2019→

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	31					

Source (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/calendar/calendar7.xhtml)

It's worth pointing out that strictly speaking there is a race condition in the calculation of today's date, 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>
```

Events

And finally to add 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="2019" m="3" d="20" t="10:00">Write program</event>
   <event y="2019" m="3" d="20" t="13:30">Meeting</event>
   <event y="2019" m="3" d="31">Clocks go forward</event>
   ...
</events>
```

Now all we have to do is display them. We replace the main display loop with this:

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

←December→←2019→

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	31					

Source (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/calendar/calendar8.xhtml)

Actually I also added

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

to allow different styling for events with no time given.

Stepping through the calendar

The only interactive part of the calendar is stepping through months and years.

If you want to click on something to have an effect in XForms, you use the `trigger` element:

```
<trigger label="your label here">
 ...some action here ...
</trigger>
```

So to step through the year value, you can use:

```
<trigger label="←">...</trigger><output ref="y"/><trigger label="→">.
```

The action for the first trigger would be:

```
<trigger label="←">
   <action ev:event="DOMActivate">
      <setvalue ref="y" value=". - 1"/>
   </action>
</trigger>
```

A click generates a `DOMActivate` event, and this causes the value of `y` to be reduced by 1.

If an `action` element contains a single action, then you can move the `ev:event` attribute onto the sub-element:

```
<trigger label="←">
   <setvalue ref="y" value=". - 1" ev:event="DOMActivate"/>
</trigger>
```

The result looks like this:

← **2019** →

[Source (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/calendar/trigger1.xhtml)](https://homepages.cwi.nl/~steven/forms/examples/xmlcom/calendar/trigger1.xhtml)

As you can see, the default presentation of a trigger is as a button. Changing this to

```
<trigger appearance="minimal" label="←">...</trigger>
```

just displays it as text:

←**2019**→

[Source (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/calendar/trigger2.xhtml)](https://homepages.cwi.nl/~steven/forms/examples/xmlcom/calendar/trigger2.xhtml)

The months are slightly more complicated, since if you go back one month earlier than January, you want to decrement the year as well:

```
<action ev:event="DOMActivate">
  <setvalue ref="m" value="if(. = 1, 12, . - 1)"/>
  <setvalue ref="y" value="if(../m = 12, . - 1, .)"/>
</action>
```

Actually, there is a shorter idiom you can use here. We only want to alter the value of `y` if the new value of `m` is 12:

```
<action ev:event="DOMActivate">
  <setvalue ref="m" value="if(.=1, 12, . - 1)"/>
   <setvalue ref="y[../m=12]" value=". - 1"/>
</action>
```

←**December**→←**2019**→

Source (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/calendar/trigger3.xhtml)

There. Around 75 lines of XForms: a simple, direct implementation of a calendar viewer.

Related links

- An Introduction to XForms (/articles/2018/11/27/introduction-xforms/)
- Viewing Data with XForms (/articles/2019/02/08/viewing-data-xforms/)