Portable Data Components

[Definition] **Portable Data Component**: a chunk of markup that can be taken out of one context and dropped into another context, unchanged. The component does not depend on its context for evaluating its constraints.

First, let's see an example of portable data from HTML5. The example comes from *HTML5 for Web Designers* by Jeremy Keith.

Portable Blog Post

Suppose I have a blog post titled *Sudden Insights*. Before HTML5, I would need to know the context of the blog post in order to decide which heading level to use for the title of the post. If the post is on the front page, then it appears after an `h1` element containing the title of my blog:

```
<h1>Diverse Thoughts</h1>
<h2>Sudden Insights</h2>
<p>Got a new idea?  Stop what you are doing and give
   attention to it immediately.</p>
```

But if I'm publishing the blog post on its own page, then I want the title of the blog post to be a level one heading:

```
<h1>Sudden Insights</h1>
<p>Got a new idea?  Stop what you are doing and give attention
to it immediately.</p>
```

The blog post is not portable data.

The blog post cannot be dropped into new contexts without modification.

However, in HTML5 I don't need to worry about which heading level to use. I just embed the blog post in the new `article` element:

```
<article>
    <h1>Sudden Insights</h1>
    <p>Got a new idea?  Stop what you are doing and give
       attention to it immediately.</p>
</article>
```

Now the blog post is portable.

It doesn't matter whether it's appearing on its own page or on the home page:

```
<h1>Diverse Thoughts</h1>
<article>
    <h1>Sudden Insights</h1>
    <p>Got a new idea?  Stop what you are doing and give
       attention to it immediately.</p>
</article>
```

HTML5's new outline algorithm produces the correct results:

- Diverse Thoughts
    - Sudden Insights

This is one example of a portable chunk of markup. Prior to HTML5 a chunk of markup depended on its context. Now with the HTML5 `article` element a chunk of markup can be made portable.

Now let's see examples of portable data outside the realm of HTML5.

A Portable Data Component -- start/end

There are many examples where the start and end time (date, dateTime) must be recorded:

Example #1

Record the start and end time of a meeting. The meeting starts at 9am and ends at 10am:

```
<meeting>
    <start>09:00:00</start>
    <end>10:00:00</end>
</meeting>
```

Example #2

Record the start and end date of a conference. The conference starts on August 3, 2011 and ends on August 6, 2011:

```
<conference>
    <start>2011-08-03</start>
    <end>2011-08-06</end>
</conference>
```

Example #3

Sometimes there is a start date but no end date. The person started working for a company on October 9, 1990 and is still employed:

```
<employment>
    <start>1990-10-09</start>
</employment>
```

Example #4

A person left Beijing at 4pm on Saturday and arrived in New York at 2pm the same day!

```
<flight>
    <start>2007-07-08T16:00:00+08:00</start>
    <end>2007-07-08T14:00:00-05:00</end>
</flight>
```

Rule: End Value is Greater Than Start Value

Common among all the examples is that the end value is greater than the start value:

- The meeting `end` time is greater than the meeting `start` time.
- The conference `end` date is greater than the conference `start` date.
- The flight `end` date/time, when adjusted for time zone differences, is greater than the flight `start` date/time.

There is a co-constraint between the value of `end` and the value of `start`.

This is an ideal candidate for a portable data component. We can create an XML Schema complexType that declares the `start` and `end` elements, and use the new `assert` element to express the co-constraint:

```
<xs:complexType name="start-end-date-time">
    <xs:sequence>
        <xs:element name="start">
            <xs:simpleType>
                <xs:union memberTypes="xs:date xs:time
                                       xs:dateTime" />
            </xs:simpleType>
        </xs:element>
        <xs:element name="end" minOccurs="0">
            <xs:simpleType>
                <xs:union memberTypes="xs:date xs:time
                                       xs:dateTime" />
```

```
            </xs:simpleType>
        </xs:element>
    </xs:sequence>
    <xs:assert test="
          if (exists(end)) then
              if (start castable as xs:dateTime) then
                  xs:dateTime(end) gt xs:dateTime(start)
              else if (start castable as xs:date) then
                  xs:date(end) gt xs:date(start)
              else if (start castable as xs:time) then
                  xs:time(end) gt xs:time(start)
              else true()
          else true()" />
</xs:complexType>
```

A Portable Data Component -- time interval (start/end or start/duration)

This is an extension of the start/end portable data component.

A time interval may be expressed using either of these:

- start and end values, or
- start and duration values

Here are examples using start/duration.

Example #1

The class starts on May 3, 2011 at 8am and runs for 8 hours:

```
<class>
  <start>2011-05-03T08:00:00</start>
  <duration>PT8H</duration>
</class>
```

Example #2

My vacation begins June 22, 2011 and is for 3 days:

```
<vacation>
  <start>2011-06-22</start>
  <duration>P3D</duration>
</vacation>
```

The constraints on the start/end values are those described earlier.

There is one constraint on the start/duration values: the duration must not be a negative value.

Determining that duration is not a negative value is not straightforward. One would think that this XPath expression would do the job:

```
xs:duration(duration) gt xs:duration("P0Y")
```

However, that is not correct. It will generate this error:

```
xs:duration is not an ordered type
```

Instead, what is required is to treat duration as a string and check that the string starts with the letter 'P' (*i.e.,* not the letter '-'):

```
starts-with(duration, 'P')
```

Here is the time-interval portable data component:

```
<xs:complexType name="time-interval">
  <xs:sequence>
    <xs:element name="start">
      <xs:simpleType>
        <xs:union memberTypes="xs:date xs:time
                               xs:dateTime" />
      </xs:simpleType>
    </xs:element>
    <xs:choice>
      <xs:element name="end" minOccurs="0">
        <xs:simpleType>
          <xs:union memberTypes="xs:date xs:time
                                 xs:dateTime" />
        </xs:simpleType>
      </xs:element>
      <xs:element name="duration" type="xs:duration" />
    </xs:choice>
  </xs:sequence>
  <xs:assert test="if (exists(end)) then
                     if (start castable as xs:dateTime) then
                       xs:dateTime(end) gt xs:dateTime(start)
                     else if (start castable as xs:date) then
                       xs:date(end) gt xs:date(start)
                     else if (start castable as xs:time) then
                       xs:time(end) gt xs:time(start)
                     else true()
                   else if (exists(duration)) then
                     starts-with(duration, 'P')
```

```
            else true()" />
</xs:complexType>
```

A Portable Data Component -- length

Note: the following "portable data component" message was posted on the world-wide xml-dev list on April 9, 2011. A lot of discussion ensued. Many people were unconvinced that the portable length component is useful. So, let's first see the component and afterward is a summary of the xml-dev discussion.

Example #1

Here is the altitude of an aircraft, expressed in both feet and meters:

```
<altitude>
    <feet>12000</feet>
    <meters>3657.6</meters>
</altitude>
```

It is important that the two length values are consistent:

```
meters = feet * 12 inches/ft * 2.54 cm/in * m/100cm
       = 12000 * 12 * 2.54 * 0.01
       = 3657.6
```

The following is invalid because the values are not consistent:

```
<altitude>
    <feet>12000</feet>
    <meters>4000</meters>
</altitude>
```

This example shows that there is a constraint between the value of the `<feet>` element and the value of the `<meters>` element.

Example #2

Here is the distance from my home to Boston, expressed in both miles and kilometers:

```
<distance-to-Boston>
    <miles>40</miles>
    <kilometers>64.37376</kilometers>
</distance-to-Boston>
```

It may be desired to express the distance in yards as well:

```
<distance-to-Boston>
    <miles>40</miles>
    <kilometers>64.37376</kilometers>
    <yards>70400</yards>
</distance-to-Boston>
```

There is a constraint between the value of the <miles> element, the <kilometers> element, and the <yards> element.

Desired Characteristics of a "Length" Data Component

It would be useful to create a data component that:

1. Permits a length to be expressed in various units
2. Ensures that the values in different units are consistent
3. Can be dropped into different contexts without alteration

We have created a data component that possesses these characteristics!

Portable Length Data Component

Here is the portable length data component:

http://www.xfront.com/xml-schema-1-1/best-practice/portable-data-components/length.xsd

Here is a sample XML instance document:

http://www.xfront.com/xml-schema-1-1/best-practice/portable-data-components/test.xml

Rebuttal of the Portable Length Data Component

During discussions on the xml-dev list multiple people expressed doubt about the usefulness of the "portable length data component." They recommended finding other, more useful "portable data components."

Let's examine their reasons. Recall the example that was provided:

```
<altitude>
    <feet>12000</feet>
    <meters>3657.6</meters>
</altitude>
```

Notice that the same value is expressed in different units.

Arguments against this Design

1. Store only one value and convert as needed. That way it can't get out of sync. This is in accordance with Normal Form in database theory which says that information should never be duplicated.

2. During a workflow, if the values accidentally get out of synch, how would one determine which is the correct value?

3. All measurements have uncertainty. Suppose the altitude measurement has an uncertainty of +/- 50 feet. Then the value of <meters> shouldn't have to be precisely 3657.6

4. Suppose that the measured distance is 1/3 mile. What do you put down so that the equations come out exactly right?

5. It is important to have some indication as to which value was an actual measurement (there might be more than one) vs. which values were computed from the measurement(s), so one could distinguish between potential measurement errors and potential computational issues (roundoffs, etc.).

Arguments in Favor of this Design

1. There are many real-world examples where the same value is expressed in different units:

- In the US the distance to the next town is shown in both miles and kilometers
- US cars show the speed in both miles-per-hour and kilometers-per-hour
- Shops in the UK show the price of loose fruit and vegetables in both price per imperial and price per metric weight
- Shoes and clothes sold in the UK show both UK and EU sizes
- Many large hotels have multiple clocks, each showing the same time but in different locales (e.g., New York, London, Rome, Hong Kong)

It is easier for the author of the information to compute and show it in different units than it is for typical shoppers/consumers to do the calculation.

2. The redundancy allows for cross-checking. Suppose a sign says that the distance to the next town is 10 miles and 300 kilometers:

```
<distance-to-next-town>
    <miles>10</miles>
    <kilometers>300</kilometers>
</distance-to-next-town>
```

We can validate the data to discover that they are not consistent. Suppose the actual distance is 10 miles. If there was just one sign, which says Distance: 300 kilometers then we would have no way of realizing that something is wrong.

Guidelines for when to Express the Same Value in Multiple Different Units

Express a value in multiple different units when the cost of deriving/computing the value in different units by the recipients is greater than the cost of computing/including the value by the sender.

Example: Consider the issue of prices and currency conversion:

```
<price>
    <USD>34.52</USD>
</price>
```

If the recipients of the XML want to convert the same price into pounds sterling, it's not a straightforward fixed conversion rate. It will depend upon multiple factors such as the date that the transaction took place and any conversion charges that apply.

```
<price>
    <USD>34.52</USD>
    <UKP>18.32</UKP>
</price>
```

If this is recording a transaction and the conversion rate is not discoverable by the recipients, then expressing the value in different units into the instance is imperative.

The transaction costs involved in discovering the equivalences are sometimes higher than the cost of carrying the data in the first place.

When deciding whether to replicate information, the determinant is this:

> Provide the data in multiple different units when the cost of replication is less than the cost of computation when the data is re-referenced.

Acknowledgements

Thanks to the following people for participating in these "portable data component" discussions:

- Paul Birkel
- Mike Brenner
- Kurt Cagle

- Tony Considine
- Pete Cordell
- John Cowan
- Roger Costello
- Michael Fletcher
- Mukul Gandhi
- Stephen Green
- Jasen Jacobsen
- Bill Kearney
- Ken Laskey
- Amelia Lewis
- Frank Manola
- Bob Natale
- Liam Quin
- Simon St. Laurent