CSCU9T4/CSCU9TF **Spring 2017**

XML Practical 1 - Well-formed XML and SVG

The purpose of this practical is to build your first XML files, see how they are displayed in a browser, and check if they are well-formed. We will use a browser and an online tool to help us find the errors in XML files.

Part I

1. Creating your first XML file

- Create a simple text file. Use Notepad (preferably Notepad++ if available) to create a file `appUsers.txt` where you will store 3 names of your choice with the *name surname* format, one on each line. For example the contents of the file might be:

```
William Smith
Rachel Maddow
Tom Jones
```

- Open `appUsers.txt` using a web browser. You should see just a simple list of the names.
- Create a second file named `appUsers.xml` containing an XML version of this data with suitably named *root* and *child* elements and where the *name* and *surname* are elements nested within a child element.
- Note: If you are doing this in standard Notepad make sure you put quotes around the full filename before saving it as an `.xml` file or otherwise you'll get an unwanted `.txt` extension added.
- Open the xml file using a web browser. You should see that it is treated very differently than the standard text file. The browser will show the metadata in a different colour than the base data, and also allows expansion and contraction of the nodes.
- Add some errors to your xml file and check how the browser will display these errors. Correct the errors and check again.

CHECKPOINT
`appUsers.xml` without errors

Part II

SVG (Scalable Vector Graphics) is a vector graphics format based on XML. Because of this, an SVG file can be created and edited with a text editor. A simple SVG document is composed of the <svg> root element and some basic shapes that make up a graphic together. It can be visualised in a modern browser.

Here is some code for a simple SVG document. Copy and paste the code to a new file in a text editor (Notepad or, better, Notepad++) and save it with a .svg extension. You may also download the file here.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is a comment -->
<svg version="1.1" width="300" height="200"
     xmlns="http://www.w3.org/2000/svg">
<rect width="100%" height="100%" fill="deepskyblue" />
<circle cx="150" cy="100" r="80" fill="orange" />
<text x="150" y="125" font-size="60" text-anchor="middle"
      fill="white">SVG</text>
</svg>
```

Open the file with a browser or image viewer. You should see something like this:

Let's look at the file line by line (see https://www.w3.org/TR/SVG/ for more details).

```
<?xml version="1.0" encoding="UTF-8"?>
```

It is good practice to have an XML declaration. It gives the version of the XML specification used and the character encoding.

```
<!-- This is a comment -->
```

Comments are delimited by `<!--` and `-->`

```
<svg version="1.1" width="300" height="200"
     xmlns="http://www.w3.org/2000/svg">
…
</svg>
```

The <svg> element is the root element. It needs to be closed with </svg>. The version attribute defines the SVG specification used. The width and height define the dimensions of

the image in pixels. The xmlns attribute specifies the SVG namespace so that all SVG elements are identified as belonging to the SVG namespace.

```
<rect width="100%" height="100%" fill="deepskyblue" />
```

The <rect> element defines a rectangle. Here the width and height are defined relative to the size of the image but the percentage signs could be removed to obtain absolute values. Try it! The fill attribute specifies the colour of the shape. Try a different colour. You can use the colour names from here: https://en.wikipedia.org/wiki/Web_colors

```
<circle cx="150" cy="100" r="80" fill="orange" />
```

The <circle> element defines a circle whose centre is at coordinate (cx, cy) and with radius r. See what happens when you change those values.

```
<text x="150" y="125" font-size="60" text-anchor="middle"
      fill="white">SVG</text>
```

The <text> element defines a text object for the “SVG” string. It needs to be closed with </text>. The text is positioned by setting an anchor (text-anchor). Here it is the middle of the text and this anchor has coordinates (x, y). See what happens when you change those values. Other text-anchor options are start and end.

Remember that each time you edit the code, you need to save the file and refresh your browser or reopen the file with an image viewer to see the new version.

2. Play around with the code above, try different dimensions and colours. Try to add a new shape to the picture.
3. Explore what happens when there is a mistake in the file. Pay attention to the error message if there is one and consider whether you find it helpful.
    a. Delete the height attribute of the <rect> element.
    b. Delete one of the /> that closes an element.
    c. Delete the </text> that closes the <text> element.
4. Repeat step 3 but this time use an XML validator (for example http://www.freeformatter.com/xml-validator-xsd.html).

5. The code below is for the flag of Sweden but contains three errors. Fix the errors. You may download the file here.

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg" width="1600"
height="1000">
<rect width="1600" height="1000" fill="#006aa7"
<rect width="200" height="1000" x="500" fill="#fecc00"/>
<rect width=1600 height="200" y="400" fill="#fecc00"/>
</SVG>
```

6. Create an SVG file that represents the flag of Laos and has a caption under the flag, see below. The image you create does not need to be perfect, it only needs to contain the necessary shapes and text in roughly the right positions.

Flag of Laos

CHECKPOINT
Flag of Sweden without errors