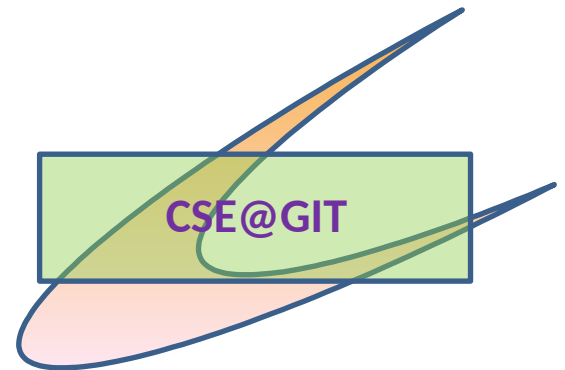


Experiment No. 4

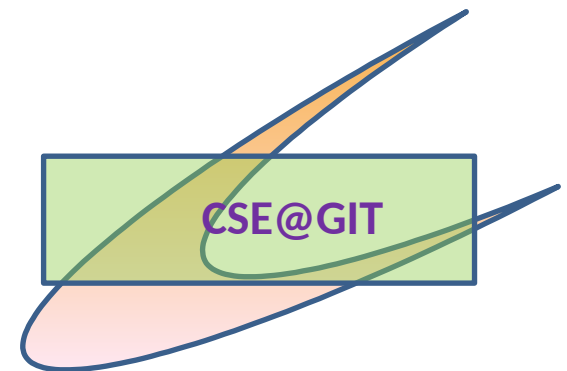
Problem Definition:

4. a) Design an XML document to store information about a student in an engineering college affiliated to VTU. The information must include USN, Name, Name of the College, Branch, Year of Joining, and e-mail id. Make up sample data for 3 students. Create a CSS style sheet and use it to display the document.
4. b) Create an XSLT style sheet for one student element of the above document and use it to create a display of that element.



Objectives of the Experiment:

1. To demonstrate the use of XML, XSLT and CSS
2. To develop an understanding of change of structured document from one form to another using XSLT
3. To be able to build simple database using XML and query with XSLT



Introduction

- XML - Extensible Markup Language (XML)
- Markup language
- XML is markup language similar to HTML
- XML is not predefined so you must define your own tags
- The primary purpose of the language is the **sharing of data** across different systems, such as the Internet

```
<?xml version="1.0"?>
<quiz>
  <qanda seq="1">
    <question>
      Who was the forty-second
      president of the U.S.A.?
    </question>
    <answer>
      William Jefferson Clinton
    </answer>
  </qanda>
  <!-- Note: We need to add
  more questions later.-->
</quiz>
```

XML

[Author en>User:Dreftymac] [https://developer.mozilla.org/en-US/docs/XML_Introduction]

"Correct" XML (valid and well-formed)

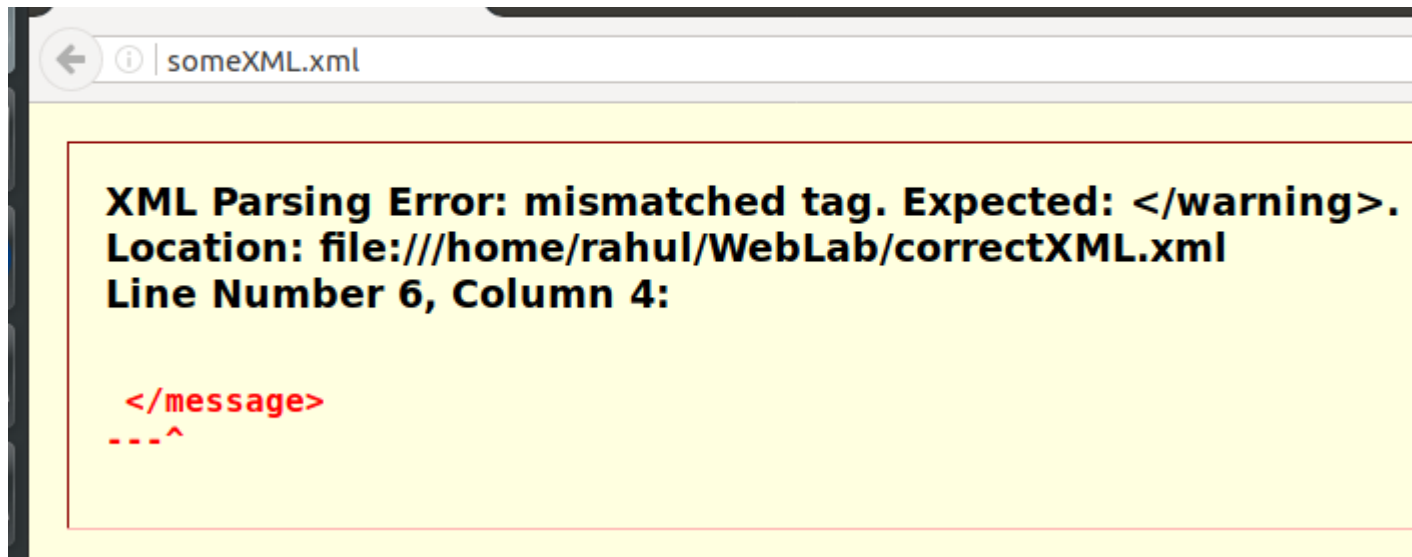
- Well-formed document
- Conforming to all of XML's syntax rules, and valid
- Example of one of syntax for well formed document : every opening tag has closing tag or is self-closing
- Is following markup valid ?

```
<message>  
    <warning>  
        Hello World  
</message>
```

"Correct" XML (valid and well-formed)

- Is following markup valid ?

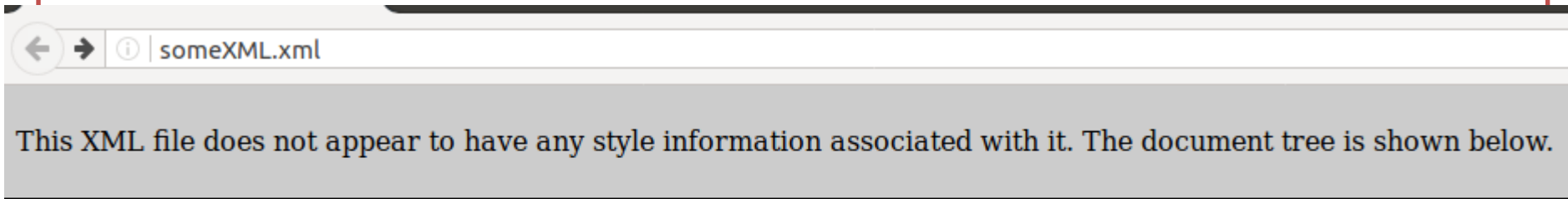
```
<message>
  <warning>
    Hello World
  </message>
```



"Correct" XML (valid and well-formed)

- Valid :

```
<message>  
  <warning>  
    Hello World  
  </warning>  
</message>
```



```
–<message>  
  <warning> Hello World </warning>  
</message>
```

"Correct" XML (valid and well-formed)

- To be valid, an XML document needs to conform to some semantic rules which are usually set in an XML schema or a Document Type Definition (DTD)
- Like programs follow syntax , so do XML
- What Context Free Grammar (CFG) is for program , Schema / DTD is for XML
- Most browsers offer a debugger that can identify poorly-formed XML documents, hence also called as parsing
- Is HTML a well formed XML ?

"Correct" XML (valid and well-formed)

- To be valid, an XML document needs to conform to some semantic rules which are usually set in an XML schema or a Document Type Definition (DTD)
- Like programs follow syntax , so do XML
- What Context Free Grammar (CFG) is for program , Schema / DTD is for XML
- Most browsers offer a debugger that can identify poorly-formed XML documents, hence also called as parsing
- HTML is well formed XML , if it follows set rules
- Rules to follow is mentioned in DOCTYPE, before **html** tag

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
    "http://www.w3.org/TR/html4/loose.dtd">
```


"Correct" XML (valid and well-formed)

- Rules to follow is mentioned in DOCTYPE, before **html** tag
- DTD

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
    "http://www.w3.org/TR/html4/loose.dtd">  
  
<html>  
  <head>  
    ...  
  </head>  
  <body>  
    ...  
  </body>  
</html>
```

|

"Correct" XML (valid and well-formed)

- Rules mentioned as DTD

```
<!-- Parameter Entities -->
```

```
<!ENTITY % head.misc "SCRIPT|STYLE|META|LINK|OBJECT" >  
    <!-- repeatable head elements -->
```

```
<!ENTITY % heading "H1|H2|H3|H4|H5|H6">
```

```
<!ENTITY % list "UL | OL | DIR | MENU">
```

```
<!ENTITY % preformatted "PRE">
```

```
<!ENTITY % Color "CDATA">
```

```
<!-- a color using sRGB: #RRGGBB as Hex values -->
```

"Correct" XML (valid and well-formed)

```
<!-- To avoid problems with text-only UAs as well as
      to make image content understandable and navigable
      to users of non-visual UAs, you need to provide
      a description with ALT, and avoid server-side image maps -->
<!ELEMENT IMG - 0 EMPTY -- Embedded image -->
<!-- ATTLIST IMG
%attrs; -- %coreattrs, %i18n, %events --
src      %URI;      #REQUIRED -- URI of image to embed --
alt      %Text;      #REQUIRED -- short description --
longdesc %URI;      #IMPLIED -- link to long description
                                (complements alt) --
name      CDATA      #IMPLIED -- name of image for scripting --
height    %Length;   #IMPLIED -- override height --
width     %Length;   #IMPLIED -- override width --
usemap    %URI;      #IMPLIED -- use client-side image map --
ismap     (ismap)    #IMPLIED -- use server-side image map --
align     %IAAlign;  #IMPLIED -- vertical or horizontal alignment --
border    %Pixels;   #IMPLIED -- link border width --
hspace    %Pixels;   #IMPLIED -- horizontal gutter --
vspace    %Pixels;   #IMPLIED -- vertical gutter --
>
```

Typical Usage - HTML

- Like HTML, XML also has only one root element
- In HTML root element is **html**, in XML – user defined / follows schema

<!--

Typical usage:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"  
    "http://www.w3.org/TR/html4/strict.dtd">
```

```
<html>
```

```
<head>
```

```
...
```

```
</head>
```

```
<body>
```

```
...
```

```
</body>
```

```
</html>
```

The URI used as a system identifier with the public identifier allows the user agent to download the DTD and entity sets as needed.

Typical Usage - XML

- First line is optional, know as XML prolog
- Unicode Transformation Format-8 is an encoding standard
- UTF-8 also standard for HTML5, CSS, JavaScript, PHP, SQL

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

```
<message>  
  <warning>  
    Hello World  
  </warning>  
</message>
```

XML namespace

- XML namespace : allows disambiguation of items having the same name by providing containers
- If thought was of namespaces in C++ or Java packages, then good ! Same concept

```
using namespace std; // std is the namespace
cout << ""; // or
std::cout << ""; // cout of std namespace
```

- Duplicates element and attribute names can occur when an XML document contains elements and attributes from **two or more** different XML schemas (or DTDs)
- Every XML element has a "default namespace"

[<https://developer.mozilla.org/en-US/docs/Namespaces>]

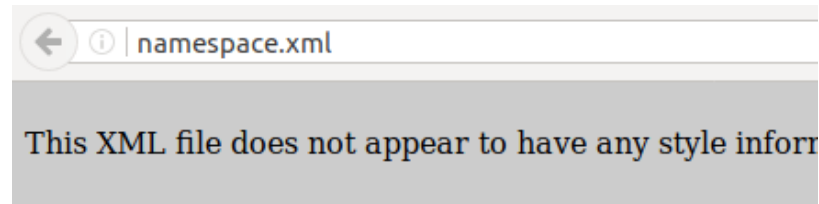
XML namespace

- Element type or attribute name in an XML namespace can be uniquely identified by its XML namespace and its "local name"
- Like in C++ namespace **std**, `std::cout`
- Suppose there are two different **textbox** element,
- Both have same names , **but** different namespace
- Namespaces must be defined at the top of the XML document in which they are used

```
<someElement  
  xmlns:xul="http://www.mozilla.org/xulNamespace"  
  xmlns:foobar="the-foobar-namespace" >
```

XML namespace

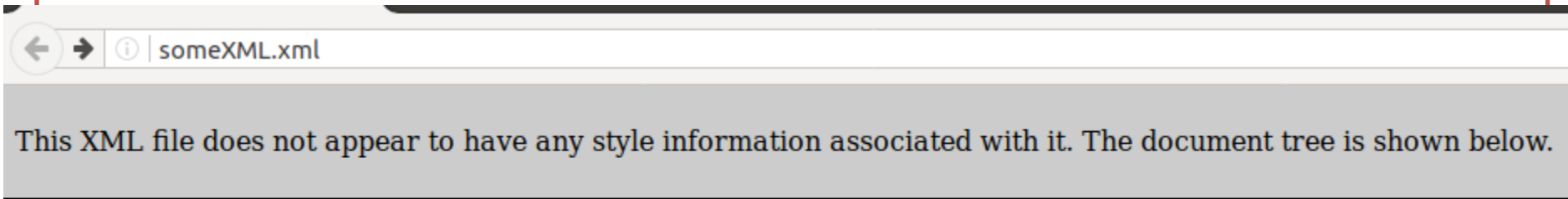
```
<?xml version="1.0" ?>
  <someElement
    xmlns:xul="http://www.mozilla.org/xulNamespace"
    xmlns:foobar="the-foobar-namespace" >
      <xul:textbox id="foo" value="bar" >
        This is textbox element, but xul namespace.
      </xul:textbox>
      <foobar:textbox favorite-food="pancakes">
        This is textbox element, but foobar namespace.
      </foobar:textbox>
    </someElement>
```



```
- <someElement>
  - <xul:textbox id="foo" value="bar">
    This is textbox element, but xul namespace.
  </xul:textbox>
  - <foobar:textbox favorite-food="pancakes">
    This is textbox element, but foobar namespace.
  </foobar:textbox>
</someElement>
```


Displaying XML

- XML is usually used for descriptive purposes, but there are ways to display XML data
- If you don't define a specific way for the XML to be rendered, the raw XML is displayed in the browser



```
-<message>  
  <warning> Hello World </warning>  
</message>
```

Displaying XML

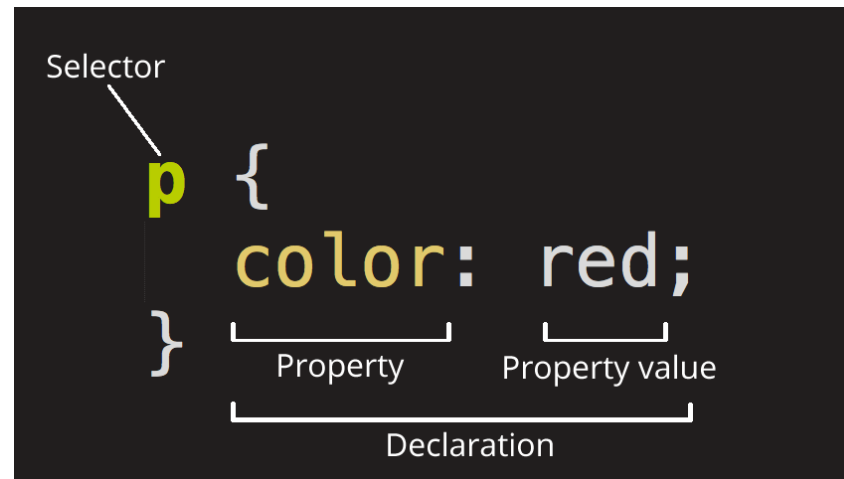
- XML is usually used for descriptive purposes, but there are ways to display XML data
- If you don't define a specific way for the XML to be rendered, the raw XML is displayed in the browser
- To style XML output is to specify CSS to apply to the document using the **xml-stylesheet** processing instruction

```
<?xml-stylesheet type="text/css" href="stylesheet.css"?>
```

```
<message>  
  <warning>  
    Hello World  
  </warning>  
</message>
```

Displaying XML - using xml-stylesheet

- XML is usually used for descriptive purposes, but there are ways to display XML data
- If you don't define a specific way for the XML to be rendered, the raw XML is displayed in the browser
- To style XML output is to specify CSS to apply to the document using the **xml-stylesheet** processing instruction



[https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics]

XML document to store information about a student in an engineering college affiliated to VTU

- Information including USN, Name, Name of the College, Branch, Year of Joining and e-mail id

```
<?xml-stylesheet type="text/css" href="4a.css" ?>
<studentInformation>
  <student>
    <usn> </usn>
    <name> </name>
    <college> </college>
    <branch> </branch>
    <year> </year>
    <email> </email>
  </student>
</studentInformation>
```

- Is this similar to database schema ?

XML document to store information about a student in an engineering college affiliated to VTU

- To add details of one more student- add new **student** element

```
<?xml-stylesheet type="text/css" href="4a.css" ?>
<studentInformation>
  <student>
    <usn> </usn>
    <name> </name>
    <college> </college>
    <branch> </branch>
    <year> </year>
    <email> </email>
  </student>
  <student>
    <usn> </usn>
    <name> </name>
    <college> </college>
    <branch> </branch>
    <year> </year>
    <email> </email>
  </student>
</studentInformation>
```

4a. Make up sample data for 3 students

- Consider engineers :

```
<?xml-stylesheet type="text/css" href="4a.css" ?>
<studentInformation>
  <student> <!-- Assume Punjab Engineering College is affiliated -->
    <usn> 2PE80AE035 </usn> <!-- to VTU, located in North region -->
    <name> Kalpana Chawla </name>
    <college> Punjab Engineering College </college>
    <branch> Aeronautical Engineering </branch>
    <year> 1984 </year>
    <email> kalpanachawla@nasa.org </email>
  </student>
  <student>
    <usn> 4JC87IT020 </usn>
    <name> Javagal Srinath </name>
    <college> Sri Jayachamarajendra College of Engineering </college>
    <branch> Instrumentation Technology </branch>
    <year> 1991 </year>
    <email> javagalsrinath@gmail.com </email>
  </student>
```

4a. Make up sample data for 3 students

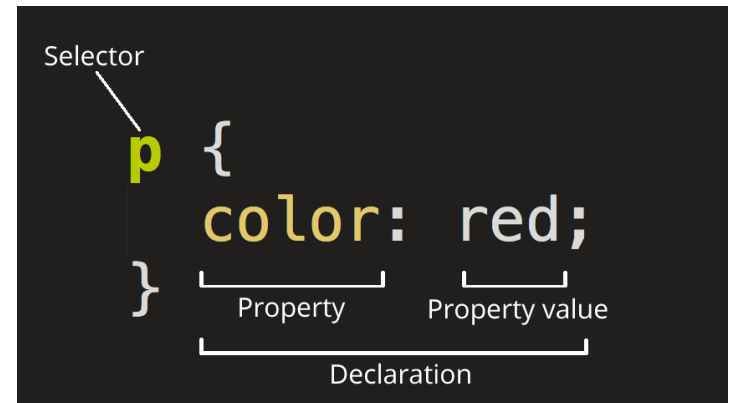
```
<student>
  <usn> 4RV88ME008 </usn>
  <name> Anil Kumble </name>
  <college> Rashtreeya Vidyalyaya College of Engineering </college>
  <branch> Mechanical Engineering </branch>
  <year> 1992 </year>
  <email> anilkumble@live.com </email>
</student>
</studentInformation>
```

- Thoughts on readability, self-describing nature of XML ?

4a. CSS style sheet to display XML

- 4a.css

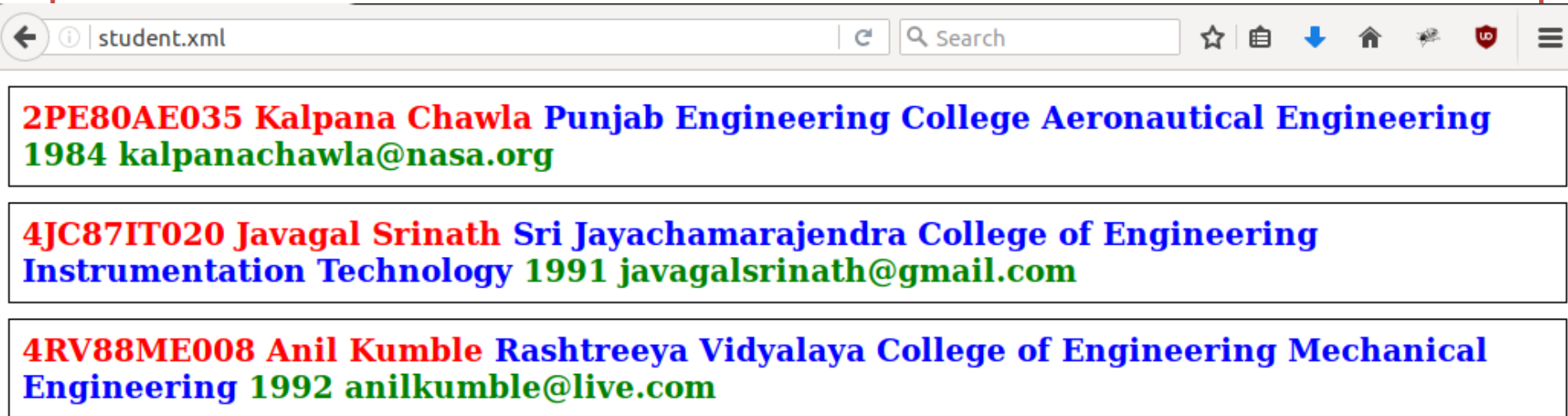
```
/* Define styling properties for
   each tag of the xml file */
studentInformation
{
    font-weight:bold;
    color:grey;
}
student
{
    border: solid black 1px;
    padding:8px;
    display:block;
    margin:10px;
}
```



4a. CSS style sheet to display XML

```
usn, name
{
    color:red;
    font-size:15pt;
}
college, branch
{
    color:blue;
    font-size:15pt;
}
year, email
{
    color:green;
    font-size:15pt;
}
```

Displaying XML - using xml-stylesheet



- Problem – not intuitive
- Benefit of CSS : can give better display but only for existing content
- Limitation of CSS : new content can not be added on context
- If restricting to only CSS , then to improve understandability of output, add meaning details to XML

Adding semantics to XML

- CSS is the same, but changes to XML :

```
<?xml-stylesheet type="text/css" href="4a.css" ?>
<studentInformation>
  <student> <!-- Assume Punjab Engineering College is affiliated -->
    Student 1 details:
    <usn> 2PE80AE035 </usn> <!-- to VTU, located in North region -->
    <name> Kalpana Chawla </name>
    <college> Punjab Engineering College </college>
    <branch> Aeronautical Engineering </branch>
    <year> 1984 </year>
    <email> kalpanachawla@nasa.org </email>
  </student>
  <student>
    Student 2 details:
    <usn> 4JC87IT020 </usn>
```

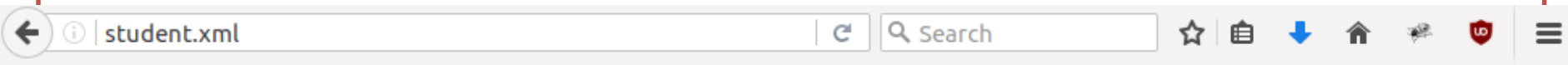
4a. Adding semantics to XML

- CSS is the same, but additions to XML :

```
<?xml-stylesheet type="text/css" href="4a.css" ?>
<studentInformation>
  <student> <!-- Assume Punjab Engineering College is affiliated
    Student 1 details:
      USN: <usn> 2PE80AE035 </usn> <!-- to VTU, located in North
      Name: <name> Kalpana Chawla </name>
      College : <college> Punjab Engineering College </college>
      Branch : <branch> Aeronautical Engineering </branch>
      Year of joining : <year> 1984 </year>
      Email-ID : <email> kalpanachawla@nasa.org </email>
    </student>
    <student>
      Student 2 details:
        USN: <usn> 4JC87IT020 </usn>
        Name: <name> Javagal Srinath </name>
```

- Is above solution redundant ?

4a. Sample Run



Student 1 details: USN: **2PE80AE035** Name: **Kalpana Chawla** College : **Punjab Engineering College** Branch : **Aeronautical Engineering** Year of joining : **1984** Email-ID : **kalpanachawla@nasa.org**

Student 2 details: USN: **4JC87IT020** Name: **Javagal Srinath** College : **Sri Jayachamarajendra College of Engineering** Branch : **Instrumentation Technology** Year of joining : **1991** Email-ID : **javagalsrinath@gmail.com**

Student 3 details: USN: **4RV88ME008** Name: **Anil Kumble** College : **Rashtreeya Vidyalaya College of Engineering** Branch : **Mechanical Engineering** Year of joining : **1992** Email-ID : **anilkumble@live.com**

Displaying XML

- XML is usually used for descriptive purposes, but there are ways to display XML data
- If you don't define a specific way for the XML to be rendered, the raw XML is displayed in the browser
- To style XML output is to specify CSS to apply to the document using the **xml-stylesheet** processing instruction
- Another way to display XML: the Extensible Stylesheet Language Transformations (**XSLT**) which can be used to transform XML into other languages such as HTML

```
<?xml-stylesheet type="text/xsl" href="transform.xsl"?>
```

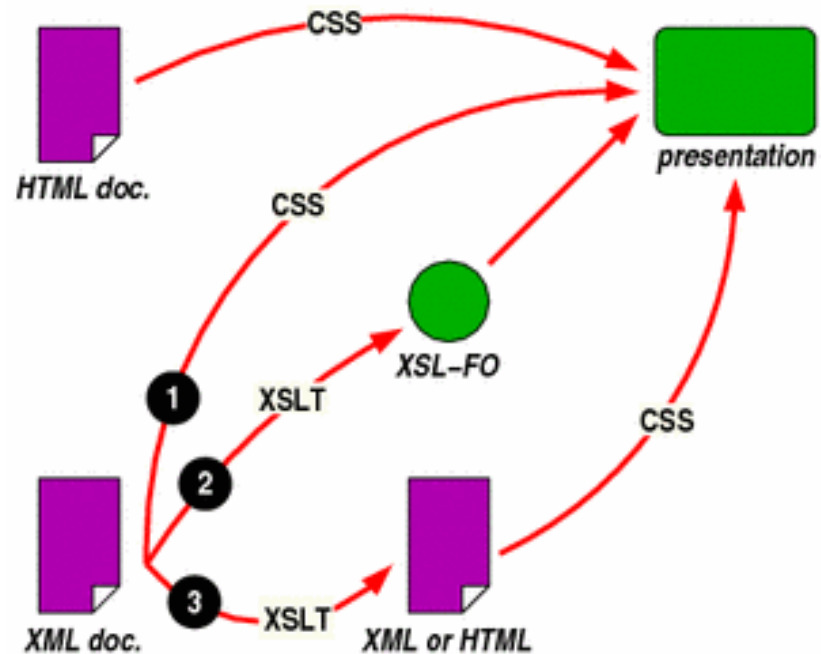
- With XSLT, XML becomes incredibly versatile

CSS & XSL

- Use CSS when you can, use XSL when you must
- CSS is much easier to use, easier to learn, easier to maintain
- In general there are more tools for CSS than for XSL
- CSS's simplicity means it has its limitations
- Use **XSL** when there is **need for transformation**
- Example, if you have a list and want it displayed in lexicographical order, use XSL

[<https://www.w3.org/Style/CSS-vs-XSL.en.html>]

Role of XSL and CSS



- (1) if the document doesn't have to be transformed, use CSS **else** use
- (2) generate the style properties together with the rearranged text, using a sub-language of XSL called XSL-FO (XSL Formatting Objects)
- (3) generate a new XML or HTML document and provide a CSS style sheet for that new document

XSLT

b) Create an XSLT style sheet for one student element of the above document and use it to create a display of that element

- Apply XSLT called as **4b.xsl** , instead of 4a.css to XML
- Transform from XML to HTML

```
<?xml-stylesheet type="text/xsl" href="4b.xsl" ?>
<studentInformation>
  <student> <!-- Assume Punjab Engineering College is affiliated -->
    <usn> 2PE80AE035 </usn> <!-- to VTU, located in North region -->
    <name> Kalpana Chawla </name>
    <college> Punjab Engineering College </college>
    <branch> Aeronautical Engineering </branch>
    <year> 1984 </year>
    <email> kalpanachawla@nasa.org </email>
  </student>
  <student>
    <usn> 4JC87IT020 </usn>
```

XSLT

- **4a.xsl** is a Extensible Stylesheet Language(**XSL**)
- XML-based language used, in conjunction with specialized processing software, for the transformation of XML documents : Extensible Stylesheet Language Transformations (**XSLT**)
- The original document is not changed, new XML document is created based on the content of an existing document
- XSLT allows a stylesheet author to transform a primary XML document in two ways:
 1. Manipulating and sorting the content, including a wholesale reordering of it if so desired
 2. Transforming the content into a different format (and in the case of Firefox, the focus is on converting it on the fly into HTML which can then be displayed by the browser)

Transforming XML with XSLT

- Separation of content and presentation
- Content stored in XML documents will need to be presented to human readers
- Specific example : Mozilla Firefox : Gecko (the layout engine behind Firefox) supports XML stylesheets



[https://developer.mozilla.org/en-US/docs/Transforming_XML_with_XSLT ,
Thierry Caro]

stylesheet

- **<xsl:stylesheet>** element or equivalent **<xsl:transform>** element is the outermost element of a stylesheet
- Defines that the document is an XSLT style sheet document (along with the version number and XSLT namespace attributes)
- pseudo-attribute required to identify the document as an XSLT stylesheet , generally -
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
- Required outermost element of stylesheet

[<https://developer.mozilla.org/en-US/docs/Web/XSLT/stylesheet>]

stylesheet

- **<xsl:stylesheet>** element or equivalent **<xsl:transform>** element is the outermost element of a stylesheet

```
<xsl:stylesheet  
  version=NUMBER  
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"  
  id="NAME"  
  extension-element-prefixes="LIST-OF-NAMES"  
  exclude-result-prefixes="LIST-OF-NAMES">
```

ENTIRE STYLESHEET

```
</xsl:stylesheet>
```

```
<xsl:stylesheet version="1.0"  
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
```

```
</xsl:stylesheet>
```

template

- **<xsl:template>** element defines an output producing template
- Must be the child of **<xsl:stylesheet>** or **<xsl:transform>**
- This element must have either the **match attribute** or the **name attribute** set

```
<xsl:template  
  match=PATTERN  
  name="NAME"  
  mode="NAME"  
  priority="NUMBER">  
  <xsl:param> [optional]
```

TEMPLATE

```
</xsl:template>
```

[<https://developer.mozilla.org/en-US/docs/Web/XSLT/template>]

template

- **<xsl:template>** element defines an output producing template
- Must be the child of **<xsl:stylesheet>** or **<xsl:transform>**
- This element must have either the **match attribute** or the **name attribute** set

```
<xsl:template  
  match=PATTERN  
  name="NAME"  
  mode="NAME"  
  priority="NUMBER">  
  <xsl:param> [optional]
```

TEMPLATE

```
</xsl:template>      <xsl:template match="/studentInformation">  
                      </xsl:template>
```

stylesheet

So far

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >

  <xsl:template match="/studentInformation">
    <!-- match all -->
  </xsl:template>

</xsl:stylesheet>
```


template

- **<xsl:template>** element is used to build templates
- Template **contains rules** to apply when a specified node is **matched**
- Benifit of matching ?

template

- **<xsl:template>** element is used to build templates
- Template **contains rules** to apply when a specified node is **matched**
- Benifit of matching ?
- Matched value can be tested / compared , sorted or presented as html file
- What if multiple nodes/elements have been matched , then guess what has been provided to go through each matched nodes ?

for-each

- for-each can be used to select every XML element of a specified node-set (like node-set is provided by template)
- **<xsl:for-each>** element selects a set of nodes and processes each of them in the same way
- Used to iterate through a set of nodes
- If **<xsl:sort>** elements appear as the children of this element, sorting occurs before processing. Otherwise, nodes are processed in document order

[Which sort may have been used ?]

[<https://developer.mozilla.org/en-US/docs/Web/XSLT/for-each>]

for-each

- for-each can be used to select every XML element of a specified node-set (like node-set is provided by template)
- **<xsl:for-each>** element selects a set of nodes and processes each of them in the same way
- Used to iterate through a set of nodes
- If **<xsl:sort>** elements appear as the children of this element, sorting occurs before processing, otherwise, nodes are processed in document order

```
<xsl:for-each select=EXPRESSION>  
  <xsl:sort> [optional]  
  TEMPLATE  
</xsl:for-each>
```

- What can be done for each node ?

[<https://developer.mozilla.org/en-US/docs/Web/XSLT/for-each>]

stylesheet

So far

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >

  <xsl:template match="/studentInformation">
    <!-- match all -->

      <xsl:for-each select="student" >
        TEMPLATE
      </xsl:for-each>

    </xsl:template>

  </xsl:stylesheet>
```

value-of

- **<xsl:value-of>** element evaluates an XPath expression, converts it to a string, and writes that string to the result tree

```
<xsl:value-of select=EXPRESSION  
             disable-output-escaping="yes" || "no" />
```

- disable-output-escaping - specifies whether special characters are escaped when written to the output
- Example, the character `<tt>></tt>` is output as `>`, not as `">"`
- This element is used to extract the value of a selected node

```
<xsl:value-of select="usn" />
```

[<https://developer.mozilla.org/en-US/docs/Web/XSLT/value-of>]

stylesheet

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >

  <xsl:template match="/studentInformation">

    <xsl:for-each select="student" >

      <xsl:value-of select="usn" />

    </xsl:for-each>

  </xsl:template>

</xsl:stylesheet>
```

stylesheet

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >

  <xsl:template match="/studentInformation">


    <xsl:for-each select="student" >

      <xsl:value-of select="usn" />

    </xsl:for-each>

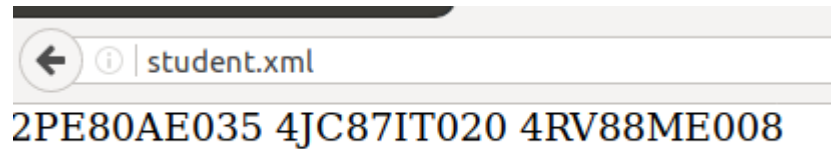
  </xsl:template>

</xsl:stylesheet>
```



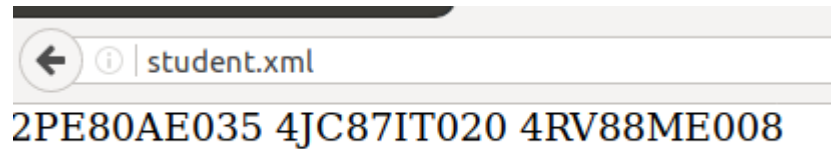
The screenshot shows a web browser address bar. On the left is a back button icon. Next to it is an information icon (i) followed by the text 'student.xml'. Below this, a long alphanumeric string is displayed: '2PE80AE035 4JC87IT020 4RV88ME008'.

Output



- Compared to CSS applied on XML, XSL was able to select specific node/element
- Can content selected be displayed inside a table of a html document ?

Output



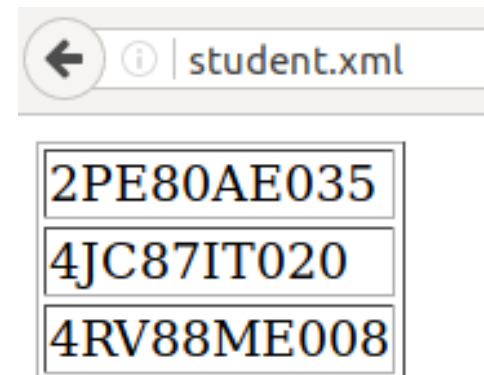
- Compared to CSS applied on XML, XSL was able to select specific node/element
- Can content selected be displayed inside a table of a html document ?
- Yes : enclose content in html , table , tr , td tag
- Where to mention open and close tags of html , table , tr , td ?

stylesheet

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
  <xsl:template match="/studentInformation">
    <html>
      <body>
        <table border="1" >
          <xsl:for-each select="student" >
            <tr>
              <td> <xsl:value-of select="usn" /> </td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

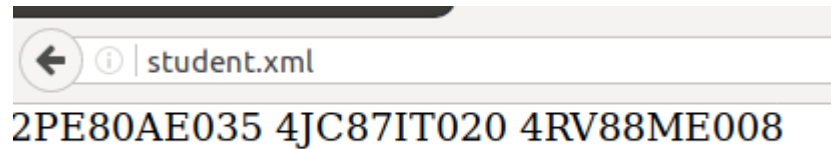
stylesheet

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
  <xsl:template match="/studentInformation">
    <html>
      <body>
        <table border="1" >
          <xsl:for-each select="student" >
            <tr>
              <td> <xsl:value-of select="usn" /> </td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```



2PE80AE035
4JC87IT020
4RV88ME008

Output



- Compared to CSS applied on XML, XSL was able to select specific node/element
- Can content selected be displayed inside a table of a html document ?
- Yes : enclose content in html , table , tr , td tag
- Where to mention open and close tags of html , table , tr , td ?
- Could conditional output be generated using CSS?
- What would be available in XSLT for testing values?

if

- **<xsl:if>** element contains a test attribute and a template
- If the test evaluates to true, the template is processed
- Similar to an if statement in other languages

```
<xsl:if test=EXPRESSION>  
    TEMPLATE  
</xsl:if>
```

- For if-then-else statement, use **<xsl:choose>** element with one **<xsl:when>** and one **<xsl:otherwise>** children

```
<xsl:choose>  
    <xsl:when test="[whatever to test1]"></xsl:when>  
    <xsl:when test="[whatever to test2]"></xsl:when>  
    <xsl:otherwise></xsl:otherwise> [optional]  
</xsl:choose>
```

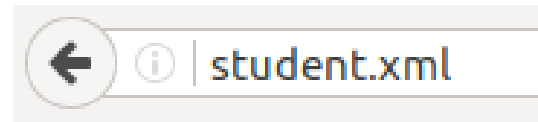
[<https://developer.mozilla.org/en-US/docs/Web/XSLT/if> ,
<https://developer.mozilla.org/en-US/docs/Web/XSLT/choose>]

4b. stylesheet

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
  <xsl:template match="/studentInformation">
    <html>
      <body>
        <table border="1" >
          <xsl:for-each select="student" >
            <xsl:if test="name[text()='Anil Kumble ']">
              <tr>
                <td> <xsl:value-of select="usn" /> </td>
              </tr>
            </xsl:if>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

4b. stylesheet

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
  <xsl:template match="/studentInformation">
    <html>
      <body>
        <table border="1" >
          <xsl:for-each select="student" >
            <xsl:if test="name[text()='Anil Kumble ']">
              <tr>
                <td> <xsl:value-of select="usn" /> </td>
              </tr>
            </xsl:if>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```



4RV88ME008

stylesheet

- Select remaining details of name, college, branch, year, email:
- Use value-of and node/element name

4b. stylesheet

- Select remaining details of name, college, branch, year, email:
- Use value-of and node/element name

```
<xsl:for-each select="student" >
  <xsl:if test="name[text()=' Anil Kumble ']">
    <tr>
      <td> <xsl:value-of select="usn" /> </td>
      <td> <xsl:value-of select="name" /> </td>
      <td> <xsl:value-of select="college" /> </td>
      <td> <xsl:value-of select="branch" /> </td>
      <td> <xsl:value-of select="year" /> </td>
      <td> <xsl:value-of select="email" /> </td>
    </tr>
  </xsl:if>
</xsl:for-each>
```

stylesheet

- Select remaining details of name, college, branch, year, email:
- Use value-of and node/element name
- Add table header

4b. stylesheet

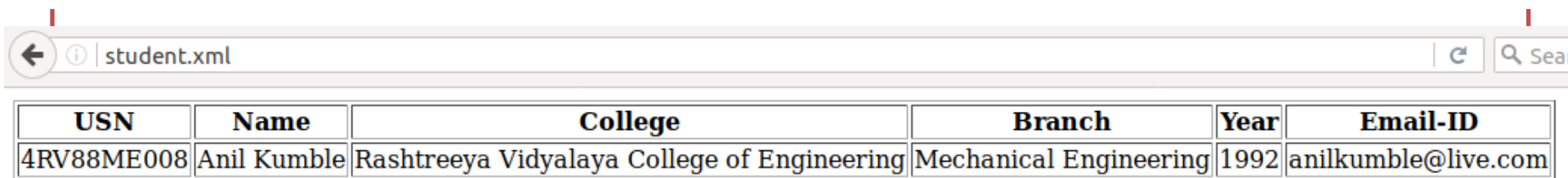
- Select remaining details of name, college, branch, year, email:
- Use value-of and node/element name
- Add table header

```
<tr>  
  <th>USN</th><th>Name</th><th>College</th><th>Branch</th>  
  <th>Year</th><th>Email-ID</th>  
</tr>
```

4b. stylesheet

- Select remaining details of name, college, branch, year, email:
- Use value-of and node/element name
- Add table header

```
<tr>
  <th>USN</th><th>Name</th><th>College</th><th>Branch</th>
  <th>Year</th><th>Email-ID</th>
</tr>
```



USN	Name	College	Branch	Year	Email-ID
4RV88ME008	Anil Kumble	Rashtreeya Vidyalaya College of Engineering	Mechanical Engineering	1992	anilkumble@live.com

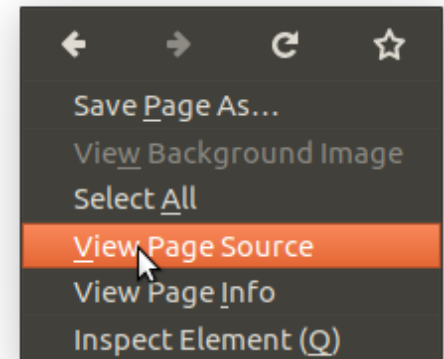
- Now add heading and title

4b. Sample Run

Student Information

USN	Name	College	Branch	Year	Email-ID
4RV88ME008	Anil Kumble	Rashtreeya Vidyalaya College of Engineering	Mechanical Engineering	1992	anilkumble@live.com

- Right click on displayed web page and select “View Properties”
- What do you see ?



XML

- XML is one of the most widely-used formats for sharing structured information today:
 - between programs
 - between people
 - between computers and people
 - both locally and across networks
- Universal Business Language (UBL); of Universal Plug and Play (UPnP) used for home electronics; word processing formats such as ODF and OOXML;
- Graphics formats such as SVG;
- Communication with XMLRPC and Web Services
- Supported by computer programming languages and databases, servers, mobile telephones

[<https://www.w3.org/standards/xml/core>]

Parsing XML with Javascript

- XML DOM (Document Object Model)
- document.getElementById method
- Node.childNodes properties

Learning Outcomes of the Experiment

At the end of the session, students should be able to :

1)

2)