XML

eXtensible Markup Language

What is XML?

- XML (eXtensible Markup Language)
- Developed by W3C (World Wide Web Consortium)
- Based on SGML (Standard Generalized Markup Language)
- It is used to store and Exchange structured data between different platforms.
- It is a metalanguage used to define other languages, called XML dialect: GML(Geographical ML), MathML, RSS, SVG, XHTML,…

Elements

- XML documents are made up of plain text (without format) and contain marks (labels) defined by the developer.

```
<name>Elsa</name>
```

- Syntax:

```
<label>value</label>
```

Empty elements

- An element can contain no value.

```
<label></label>
<label/>
```

- Example:

```
<name></name>
<name/>
```

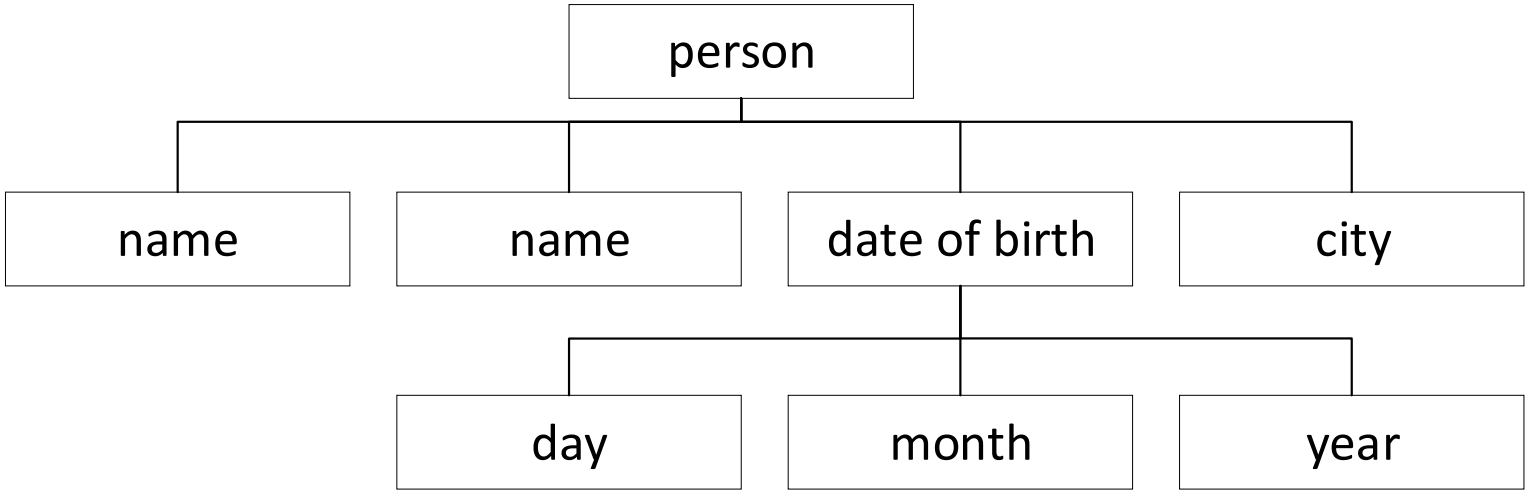
Parent-Child relationship between elements

- An element (parent) can contain zero, one or many other elements (children)

```
<person>
  <name>Elsa</name>
  <woman/>
  <date-of-birth>
    <day>18</day>
    <month>6</month>
    <year>1996</year>
  </date-of-birth>
  <city>Seville</city>
</person>
```

Root element in a XML document

- Every XML document must have a single root element (parent) from which all the others descend.



- The elements are those that give semantic structure to the document.

Elements with mixed content

- An element can contain mixed content, that is, text and other elements.

```
<person>
   <name>Elsa</name> lives in <city>Seville</city>.
</person>
```

- The "person" element contains the elements "name" and "city", in addition to the texts "lives in" and "."

Basic syntax rules

- All the names of the elements are case sensitive.
- They can contain:
  - lowercase letters,
  - capital letters,
  - numbers,
  - period".",
  - middle dashes "-“
  - underscores "_".
- They can contain the character colon ":". However, its use is reserved for when namespaces are defined.
- The first character must be a letter or a hyphen "_".

Basic syntax rules

- Behind the name of a label is allowed to write a blank space or a line break.

```
<city>Seville</ city
>
```

- There can not be a line break or a blank space before the name of a label.

```
<
city> Pamplona </ city>
Tutorial
```

Examples
Elements written incorrectly

```
<City>Seville</city>
<día>18</dia>
<month>6<month />
<city>Seville</endcity>
<_red>
<2colors>Red and Orange</2 colors>
< Hobbies > Cinema,Dancing,Swimming </ Hobbies >
<person><name>Elsa</person></name>
<favorite color>blue</favorite color>
```

Examples
Elements written correctly

```
<City>Seville</City>
<día>18</día>
<month>6</month>
<city>Seville</city>
<_red/>
<colors2>Red and Orange</colors2>
<Hobbies > Cinema,Dancing,Swimming </Hobbies >
<person><name>Elsa</name></person>
<favorite_color>blue</favorite_color>
```

Basic syntax rules

- The non-English letters (á, Á, ñ, Ñ ...) are allowed.
- However, it is advisable not to use them to reduce possible incompatibilities with programs that may not recognize them.
- Likewise, it is advisable to avoid using the hyphen character "-" and period "."

Attributes

- An attribute provides extra information about the element that contains it.

```
<product code = "G45">
    <name color="black" price="12.56">Wool cap</name>
</product>
```

- The values of the attributes can be written in double quotes (") or simple (').

Syntax rules (Attributes)

- The names of the attributes must comply with the same rules of syntax as the names of the elements.

- In addition, all the attributes of an element have to be unique. For example, it is incorrect to write:

```
<data x="3" x="4" y="5" />
```

- It is correct to write (because the second X is capital)

```
<data x="3" X="4" y="5" />
```

Avoid XML attributes?

- Some things to consider when using attributes are:
  - attributes cannot contain multiple values (elements can)
  - attributes cannot contain tree structures (elements can)
  - attributes are not easily expandable (for future changes)

```
<note>
  <date>
    <year>2008</year>
    <month>01</month>
    <day>10</day>
  </date>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don’t forget me this weekend!”</body>
</note>
```

We should try to avoid this.

```
<note day="10" month="01" year="2008"
to="Tove" from="Jani" heading="Reminder"
body="Don't forget me this weekend!">
</note>
```

Avoid XML attributes?

- Use attributes only to provide information that is not relevant to the data.

```
<messages>
  <note id="p501">
    <to>Tove</to><from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
  </note>
  <note id="p502">
    <to>Jani</to><from>Tove</from>
    <heading>Re: Reminder</heading>
    <body>I will not!</body>
  </note>
</messages>
```

XML declaration

- The XML declaration is optional.
- If it is included, it must appear on the first line of the document, and the "<" character must be the first one of the line.
- It is just this line:
  ```
  <?xml version="1.0"?>
  ```
- In an XML document, it is not mandatory that the XML declaration appears.

XML declaration

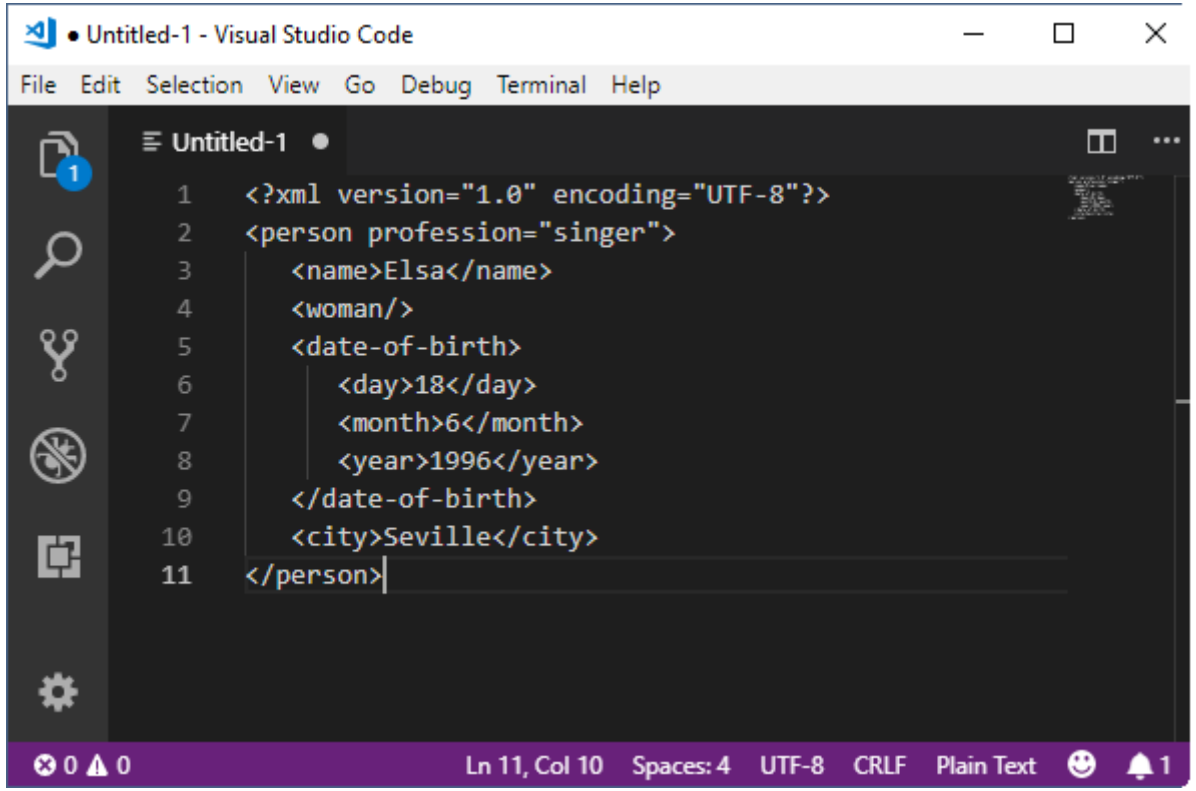
- An attribute “encoding” can be added to the XML declaration.

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

- That “encoding” attribute indicates, in a vaguely way, how the characters of the file are saved in the file.
- Do not worry about this too much at the moment.

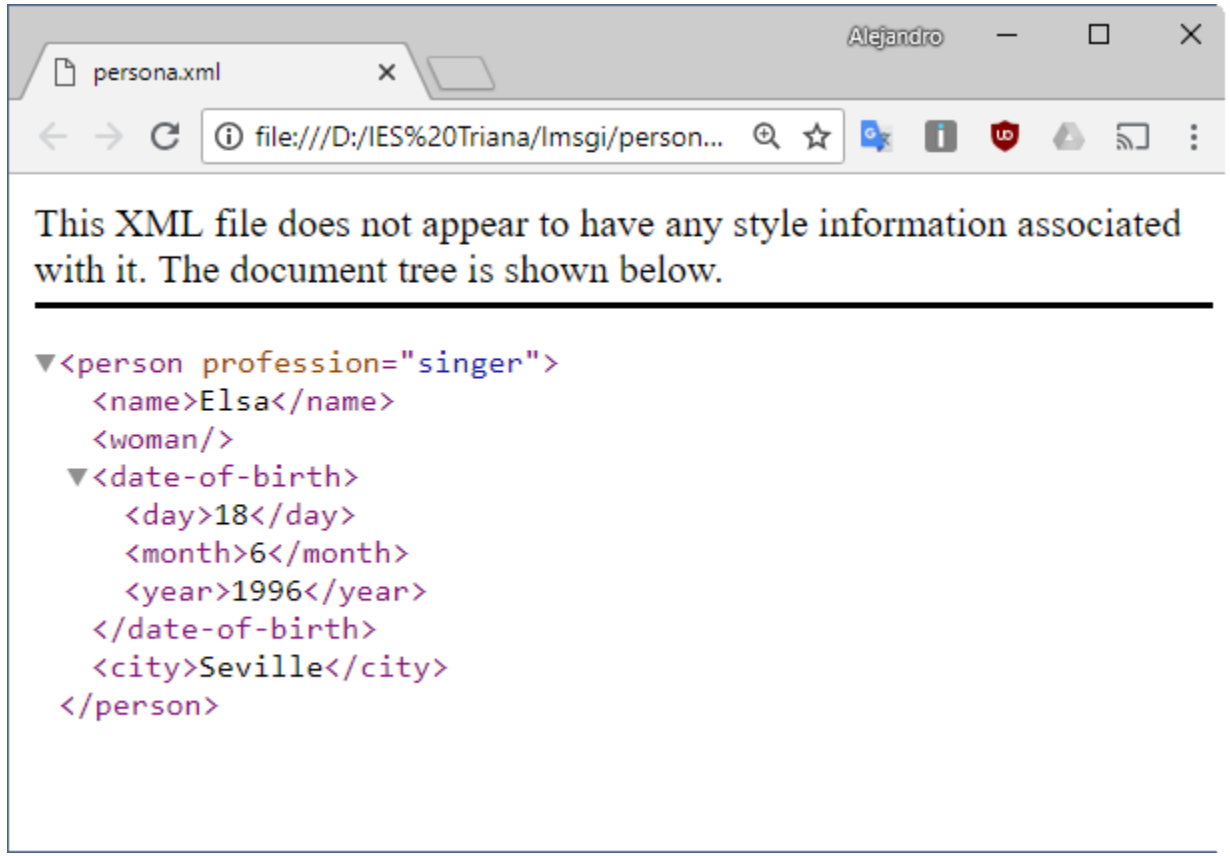
How to create a XML document

- Example. Using any editor. In this case Visual Studio Code. Encoded with UTF-8.



Display a XML document

- Example. With Google Chrome.



Applications to play with XML

- Windows/Linux: XML Copy Editor.
  - The first app we are going to use to write XML.
  - Press F2 to validate if your XML is correct.
- XML viewers online:
  - https://codebeautify.org/xmlviewer
  - http://countwordsfree.com/xmlviewer
- Find yours online.

Processing instructions

- A processing instruction is used to indicate certain information to the program that processes the document.
- EXAMPLE. To associate a CSS file with an XML document:

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

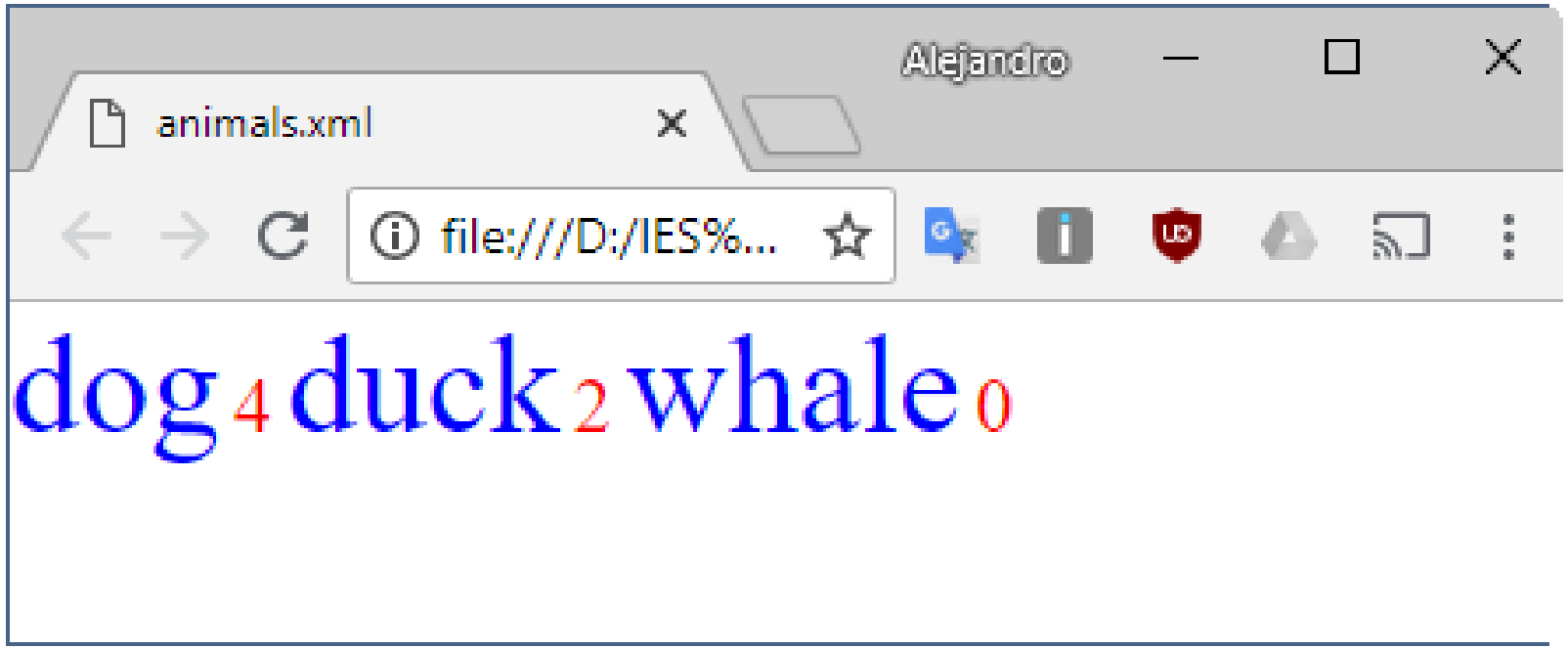
- EXAMPLE. Content of the file "animal-style.css".

```
name{color:blue;font-size:40px}
legs{color:red;font-size:22px}
```

Example “animals.xml”

```
<?xmlversion="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="animal-style.css"?>
<animals>
    <animal>
        <name>dog</name>
        <legs>4</legs>
    </animal>
    <animal>
        <name>duck</name>
        <legs>2</legs>
    </animal>
    <animal>
        <name>whale</name>
        <legs>0</legs>
    </animal>
</animals>
```

“animals.xml”



Entity references

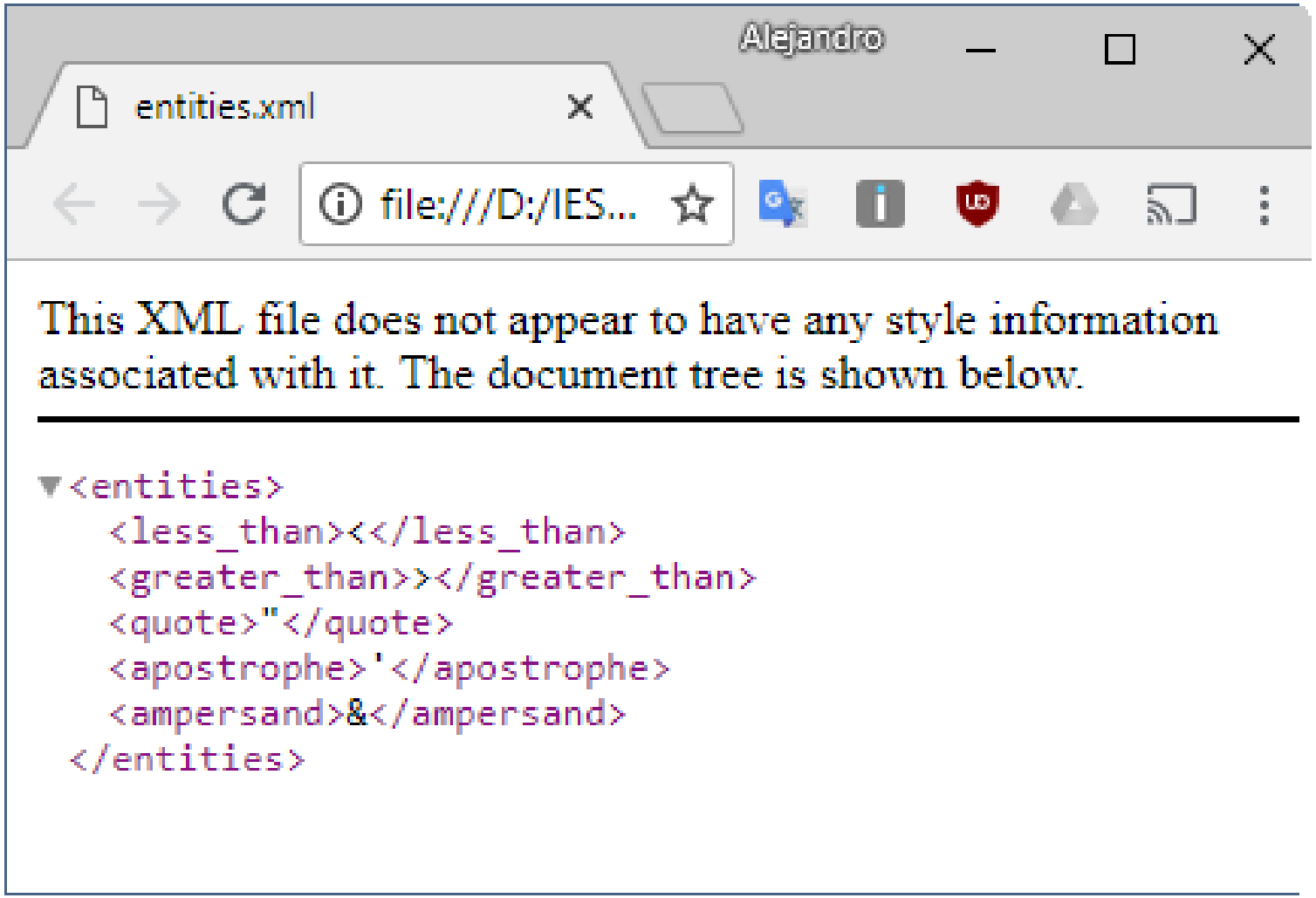
XML entity references		
Character	Entity	Entity reference
< (less than)	lt	<
> (greater than)	Gt	>
“ (quotation mark)	quot	"
‘ (apostrophe)	apos	'
& (ampersand)	amp	&

Entity references

- Example: “entities.xml”

```
<?xml version=“1.0” encoding=“UTF-8”?>
<entities>
  <less_than>&lt;</less_than>
  <greater_than>&gt;</greater_than>
  <quote>&quot;</quote>
  <apostrophe>&apos;</apostrophe>
  <ampersand>&amp;</ampersand>
</entities>
```

Entity references



Problematic characters in XML: less than (<) and ampersand (&)

- It is not correct:

```
<condition>a<b</condition>
<condition>a=1 && b=2</condition>
```

- It is right:

```
<condition>a&lt;b</condition>
<condition>a=1 && b=2</condition>
<condition>a>b</condition>
```

Use of the double quote (") and the single quote (') in attributes

- It is not correct:

```
<data character="double quote (") "/>
<data character='single quote (') '/>
```

- It is right:

```
<data character="double quote (&quot;)" />
<data character='single quote (&apos;)’ />

<data character="single quote (')" />
<data character='double quote (")' />
```

Character references

- EXAMPLE "products.xml“

```
<?xml version="1.0" encoding="UTF-8"?>
<products>
  <name price="12.56&#8364;">Woolcap</name>
  <name price="16.99&#x20AC;">Polarcap</name>
</products>
```

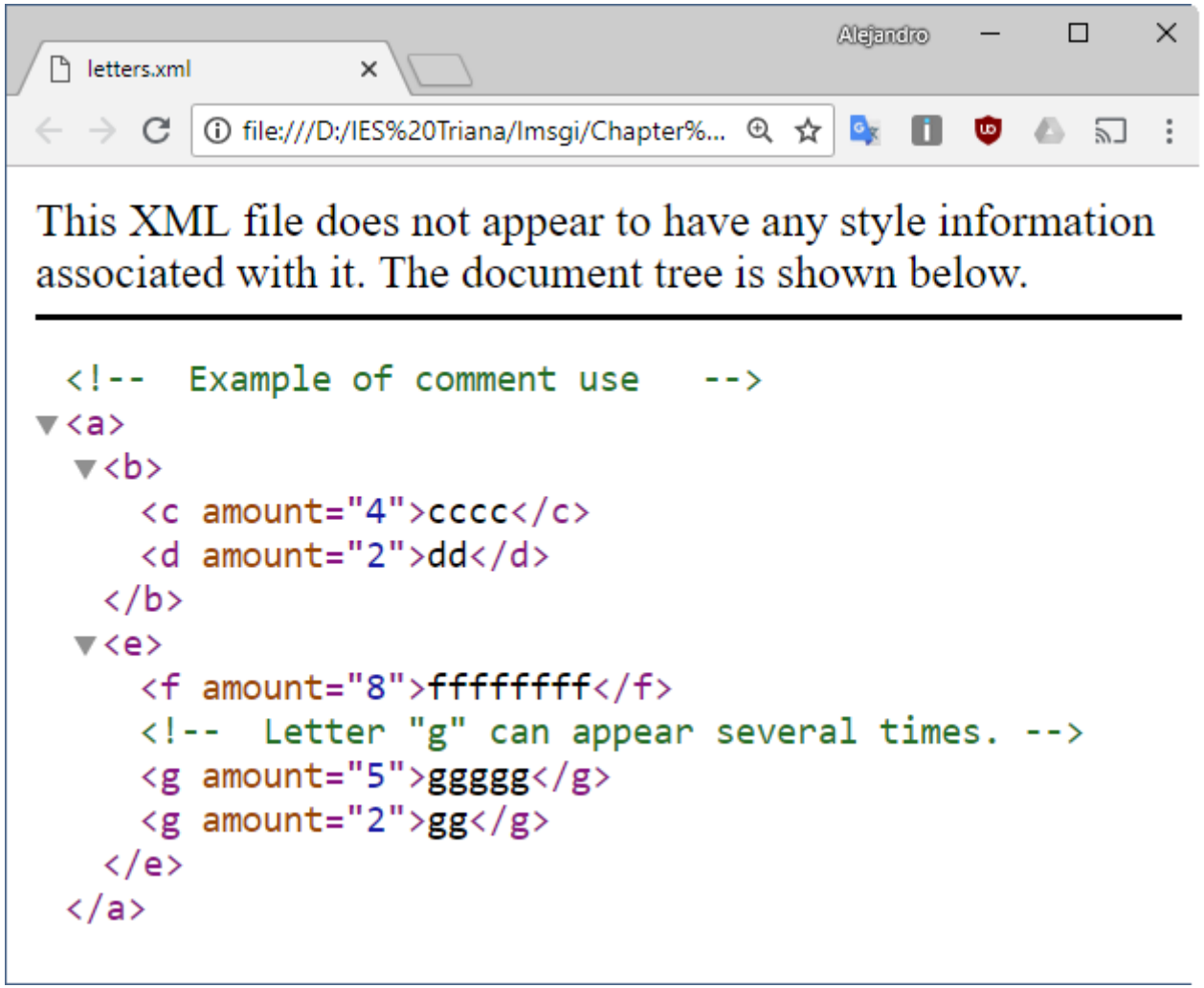
- See: https://unicode-table.com/

“products.xml”

Comments. Example “letters.xml”

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Example of comment use  -->
<a>
    <b>
        <c amount="4">cccc</c>
        <d amount="2">dd</d>
    </b>
    <e>
        <f amount="8">ffffffff</f>
        <!-- Letter "g" can appear several times.-->
        <g amount="5">ggggg</g>
        <g amount="2">gg</g>
    </e>
</a>
```

“letters.xml”


letters.xml
Alejandro
file:///D:/IES%20Triana/Imsgi/Chapter%...
This XML file does not appear to have any style information associated with it. The document tree is shown below.
<!-- Example of comment use -->
<a>

<c amount="4">cccc</c>
<d amount="2">dd</d>

<e>
<f amount="8">ffffffff</f>
<!-- Letter "g" can appear several times. -->
<g amount="5">ggggg</g>
<g amount="2">gg</g>
</e>

Comments

- You can not write comments inside the tags.

```
<woman <!-- empty element --> />
```

- In the comments it is not allowed to use two dashes in a row:

```
<!-- Two dashes in a row -- in a comment, gives an error -->
```

- It is not possible to nest comments in an XML document.

CDATA section. “cdata_example.xml”

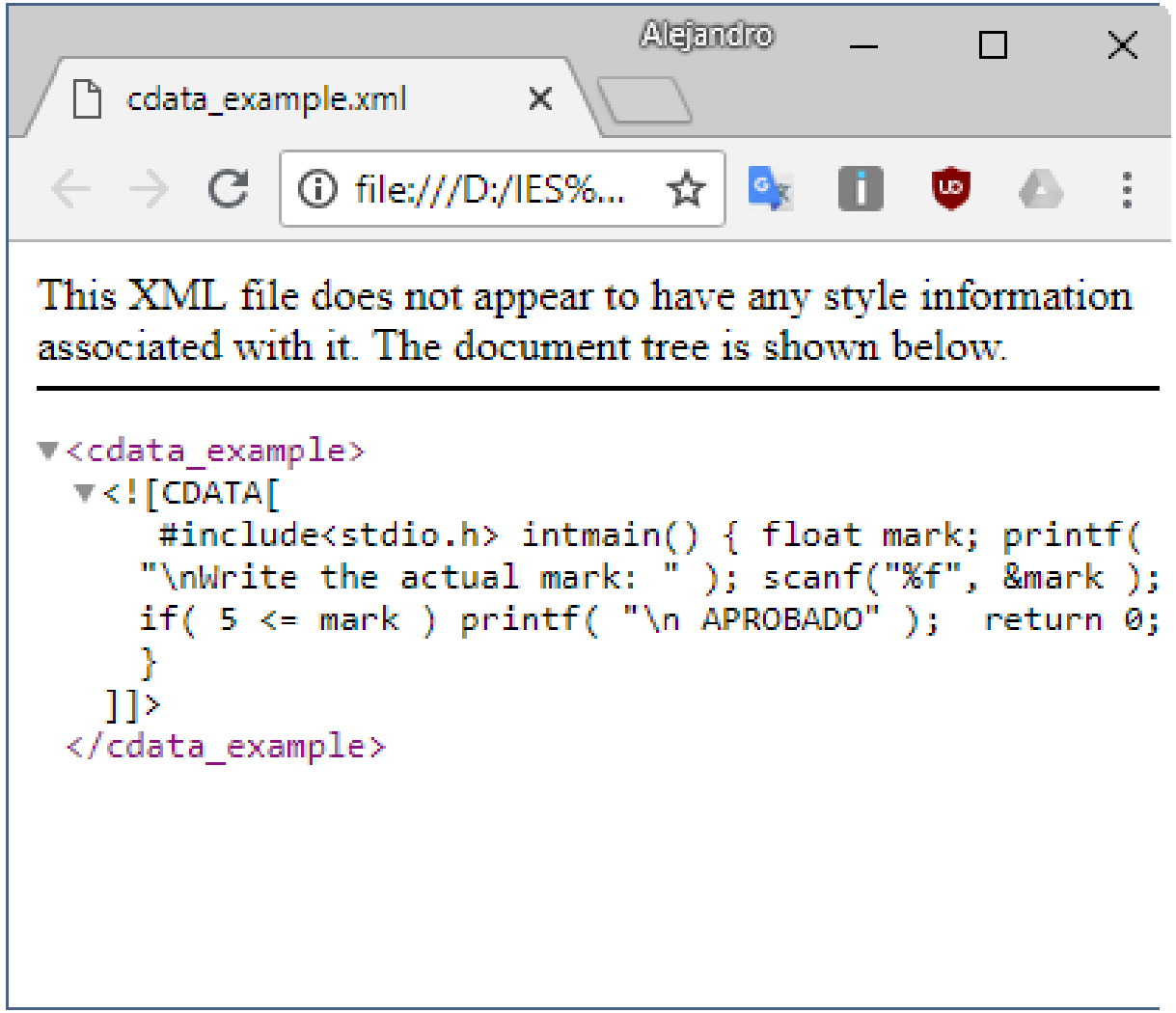
```
<?xml version="1.0" encoding="UTF-8"?>
<cdata_example>
<![CDATA[
#include<stdio.h>

int main()
{
   float mark;
   printf( "\nWrite the actual mark: " );
   scanf("%f", &mark );

   if( 5 <= mark )
      printf( "\nPASSED" );

   return 0;
}
]]>
</cdata_example>
```

“cdata_example.xml”



CDATA sections

- Within a CDATA section you can not write the string "]]>".
- As a result, CDATA sections can not be nested.
- It is not allowed to write blank spaces or line breaks in the "<![CDATA [" or end "]]>" start strings of a CDATA section.

Namespaces

- EXAMPLE. Two XML documents could contain an element called “carta", but with different meaning

```
<carta>
  <palo>Corazones</palo>
  <numero>7</numero>
</carta>

<carta>
  <carnes>
    <filete_de_ternera precio="12.95"/>
    <solomillo_a_la_pimienta precio="13.60"/>
  </carnes>
  <pescados>
    <lenguado_al_horno precio="16.20"/>
    <merluza_en_salsa_verde precio="15.85"/>
  </pescados>
</carta>
```

Using namespaces

```
<?xmlversion="1.0" encoding="UTF-8"?>
<e1:ejemplo xmlns:e1="http://www.iestriana.com/ejemplo1"
xmlns:e2="http://www.iestriana.com/ejemplo2">
    <e1:carta>
        <e1:palo>Corazones</e1:palo>
        <e1:numero>7</e1:numero>
    </e1:carta>
    <e2:carta>
        <e2:carnes>
            <e2:filete_de_ternera precio="12.95"/>
            <e2:solomillo_a_la_pimienta precio="13.60"/>
        </e2:carnes>
        <e2:pescados>
            <e2:lenguado_al_horno precio="16.20"/>
            <e2:merluza_en_salsa_verde precio="15.85"/>
        </e2:pescados>
    </e2:carta>
</e2:ejemplo>
```

Syntax to define a namespace

```
xmlns:prefix="URI"
xmlns:e1="http://www.iestriana.com/ejemplo1"
xmlns:e2="http://www.iestriana.com/ejemplo2"
```

URIs do not have to contain anything, their function is to be unique. However, in a URI you can display information if it is considered appropriate:

– http://www.w3.org/1999/xhtml/

– http://www.w3.org/1999/XSL/Transform

– http://www.w3.org/2000/svg

Definition of namespaces in elements other than the root

```
<?xmlversion="1.0" encoding="UTF-8"?>
<e1:ejemplo xmlns:e1="http://www.iestriana.com/ejemplo1">
    <e1:carta>
        <e1:palo>Corazones</e1:palo>
        <e1:numero>7</e1:numero>
    </e1:carta>
    <e2:carta xmlns:e2="http://www.iestriana.com/ejemplo2">
        <e2:carnes>
            <e2:filete_de_ternera precio="12.95"/>
            <e2:solomiIlo_a_la_pimienta precio="13.60"/>
        </e2:carnes>
        <e2:pescados>
            <e2:lenguado_al_horno precio="16.20"/>
            <e2:merluza_en_salsa_verde precio="15.85"/>
        </e2:pescados>
    </e2:carta>
</e1:ejemplo>
```

Definition of a default namespace

- Syntax: `xmlns="URI“`

- Example:

```
<?xmlversion="1.0" encoding="UTF-8"?>
<ejemplo xmlns="http://www.iestriana.com/ejemplo1">
<carta>
   <palo>Corazones</palo>
   <numero>7</numero>
   </carta>
</ejemplo>
```

Definition of a default namespace

```
<?xmlversion="1.0" encoding="UTF-8"?>
<ejemplo xmlns="http://www.iestriana.com/ejemplo1">
   <carta>
      <palo>Corazones</palo>
      <numero>7</numero>
   </carta>
   <carta xmlns="http://www.iestriana.com/ejemplo2">
      <carnes>
         <filete_de_ternera precio="12.95"/>
         <solomillo_a_la_pimienta precio="13.60"/>
      </carnes>
      <pescados>
         <lenguado_al_horno precio="16.20"/>
         <merluza_en_salsa_verde precio="15.85"/>
      </pescados>
   </carta>
</ejemplo>
```

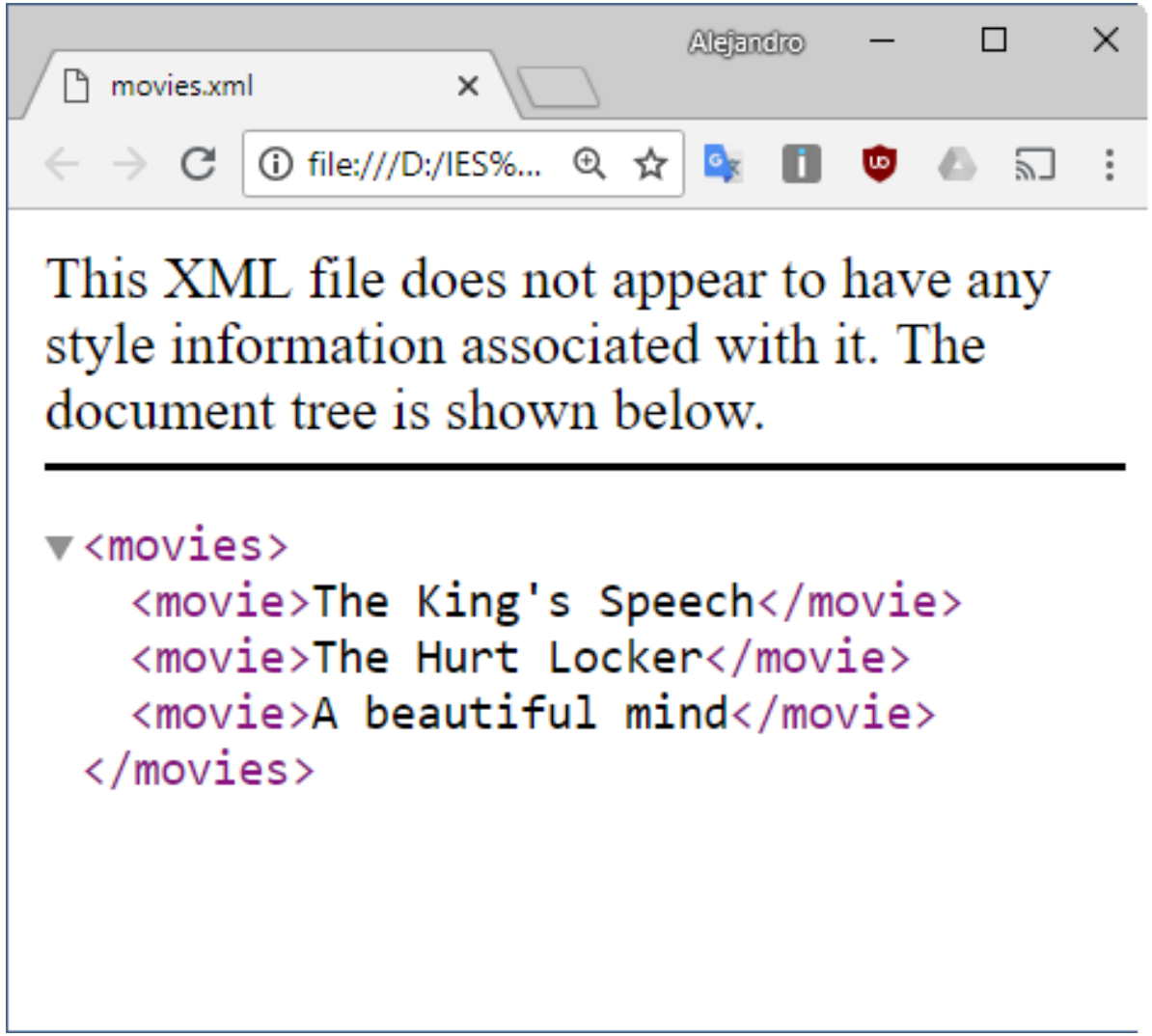
Indicating an element does not belong to any namespace

```
<?xmlversion="1.0" encoding="UTF-8"?>
<ejemplo xmlns="http://www.iestriana.com/ejemplo1">
   <carta>
      <palo>Corazones</palo>
      <numero>7</numero>
   </carta>
   <carta xmlns="http://www.iestriana.com/ejemplo2">
      <carnes>
         <filete_de_ternera precio="12.95"/>
         <solomillo_a_la_pimienta precio="13.60"/>
      </carnes>
      <pescados xmlns="">
         <lenguado_al_horno precio="16.20"/>
         <merluza_en_salsa_verde precio="15.85"/>
      </pescados>
   </carta>
</ejemplo>
```

Blank spaces in the content (text) of an element. EXAMPLE "movies.xml"

```
<?xml version="1.0" encoding="UTF-8"?>
<movies>
    <movie>The King's Speech</movie>
    <movie>The Hurt Locker</movie>
    <movie>A
beautiful
mind</movie>
</movies>
```

“movies.xml”


Alejandro
movies.xml
file:///D:/IES%...
This XML file does not appear to have any style information associated with it. The document tree is shown below.
<movies>
<movie>The King's Speech</movie>
<movie>The Hurt Locker</movie>
<movie>A beautiful mind</movie>
</movies>

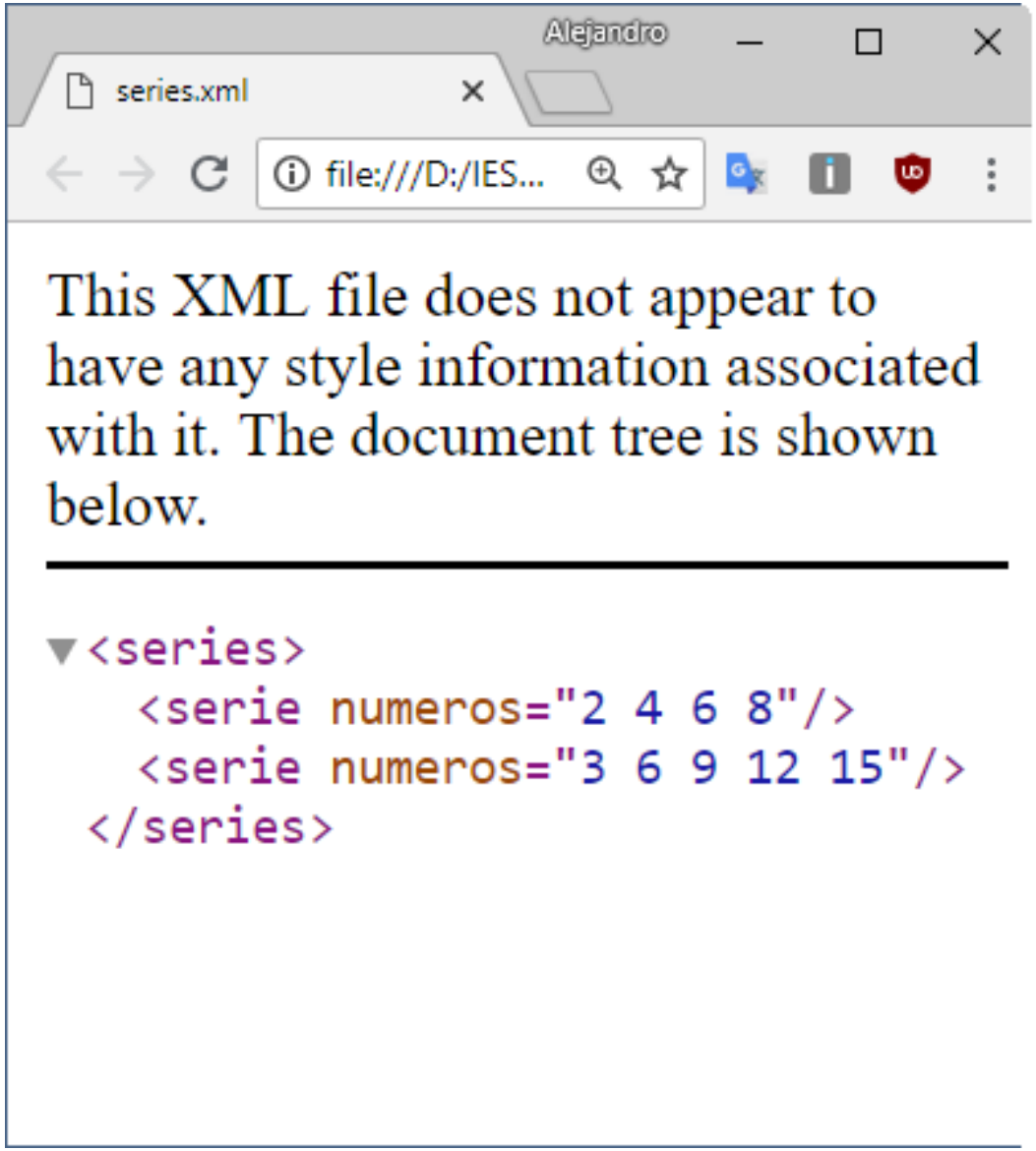
Blank spaces in attributes. Example “series.xml”

```
<?xml version="1.0" encoding="UTF-8"?>
<series>
   <serie numeros="2 4 6 8"/>
   <serie numeros="3
6



   9
   12 15"/>
</series>
```

“series.xml”


Alejandro
series.xml
file:///D:/IES...
This XML file does not appear to have any style information associated with it. The document tree is shown below.
<series>
<serie numeros="2 4 6 8"/>
<serie numeros="3 6 9 12 15"/>
</series>

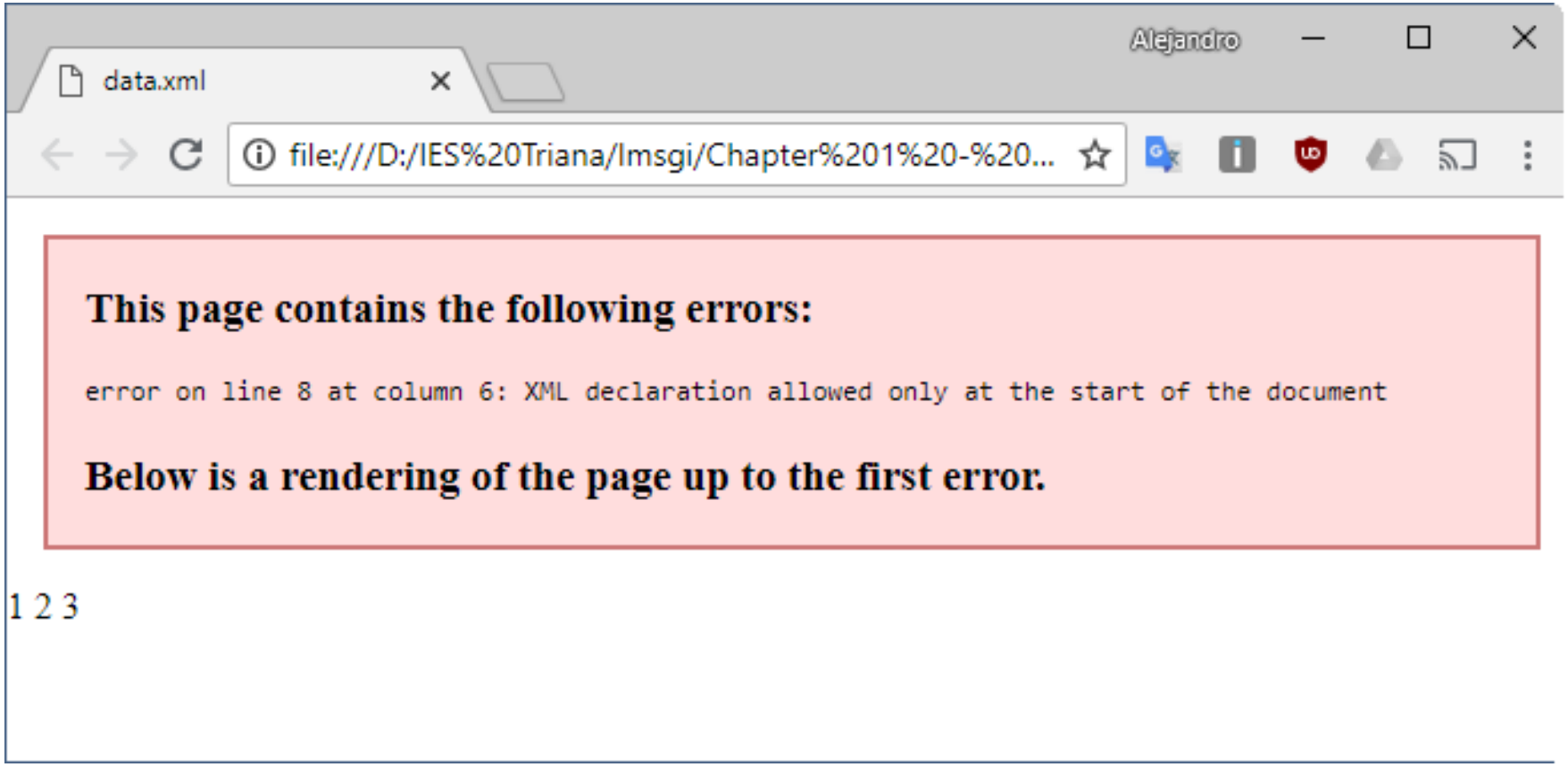
Blank spaces between elements. Example “data.xml”

```
<?xml version="1.0" encoding="UTF-8"?>
<datos>
<dato>1</dato>
<dato>2</dato>
<dato>3</dato>
</datos>

<?xml version="1.0" encoding="UTF-8"?>
<datos><dato>1</dato><dato>2</dato><dato>3</dato></datos>

<?xml version="1.0" encoding="UTF-8"?>
<datos><dato>1</dato> <dato>2</dato>
<dato>3</dato></datos>
```

“data.xml”


data.xml
Alejandro
file:///D:/IES%20Triana/lmsgi/Chapter%201%20-%20...
This page contains the following errors:
error on line 8 at column 6: XML declaration allowed only at the start of the document
Below is a rendering of the page up to the first error.
1 2 3

Using the attribute xml:space

```
<classification xml:space="preserve">
1            Fernando Alonso              1:55.341
2            Lewis Hamilton                 1:55.729
3            SebastianVettel                 1:56.122
</classification >
```

- The only values that the attribute **xml:space** admits are "preserve" and "default", the latter being its default value when that attribute is not written.
- The value "default" indicates that the application that makes use of the XML document is responsible for deciding how to deal with the blanks.
- Not all programs recognize this attribute.

Well-formed XML documents (without syntax errors)

- The names of the elements and their attributes must be written correctly.
- The values of the attributes must be written in double or single quotes.
- The attributes of an element must be separated with blanks.
- You have to use references to entities where necessary.
- There must be a single root element.
- Every element must have a parent element, except the root element.
- All items must have an opening tag and a closing tag.
- The labels must be correctly nested.
- The process instructions must be written correctly.
- The XML declaration must be in the first line written correctly.
- CDATA sections and comments must be correctly written.

Valid XML documents

- An XML document is valid when, in addition to not having syntax errors, it does not violate any of the rules established in its structure.

- This structure can be defined using different methods:
  - DTD (Document Type Definition).
  - XML Schema.
  - RELAX NG (Regular Language for XML Next Generation).