XML Technologies

Lecture 02

SVG
Scalable Vector Graphics

Vector graphics

It consists of simple lines and curves defined by mathematical objects – vectors

Vectors describe graphics in geometric terms

It does not depend on the screen resolution

It is suitable for creating text graphics, business cards, emblems, logos, where lines must be sharp and clear

SVG

The graphics resulting from the code interpretation are fully scalable and the code file is small

It is easily editable (because it is written using XML)

Every element and every attribute can be animated

An additional advantage is the excellent print quality of the page containing SVG elements

Use of SVG

Graphic elements for websites

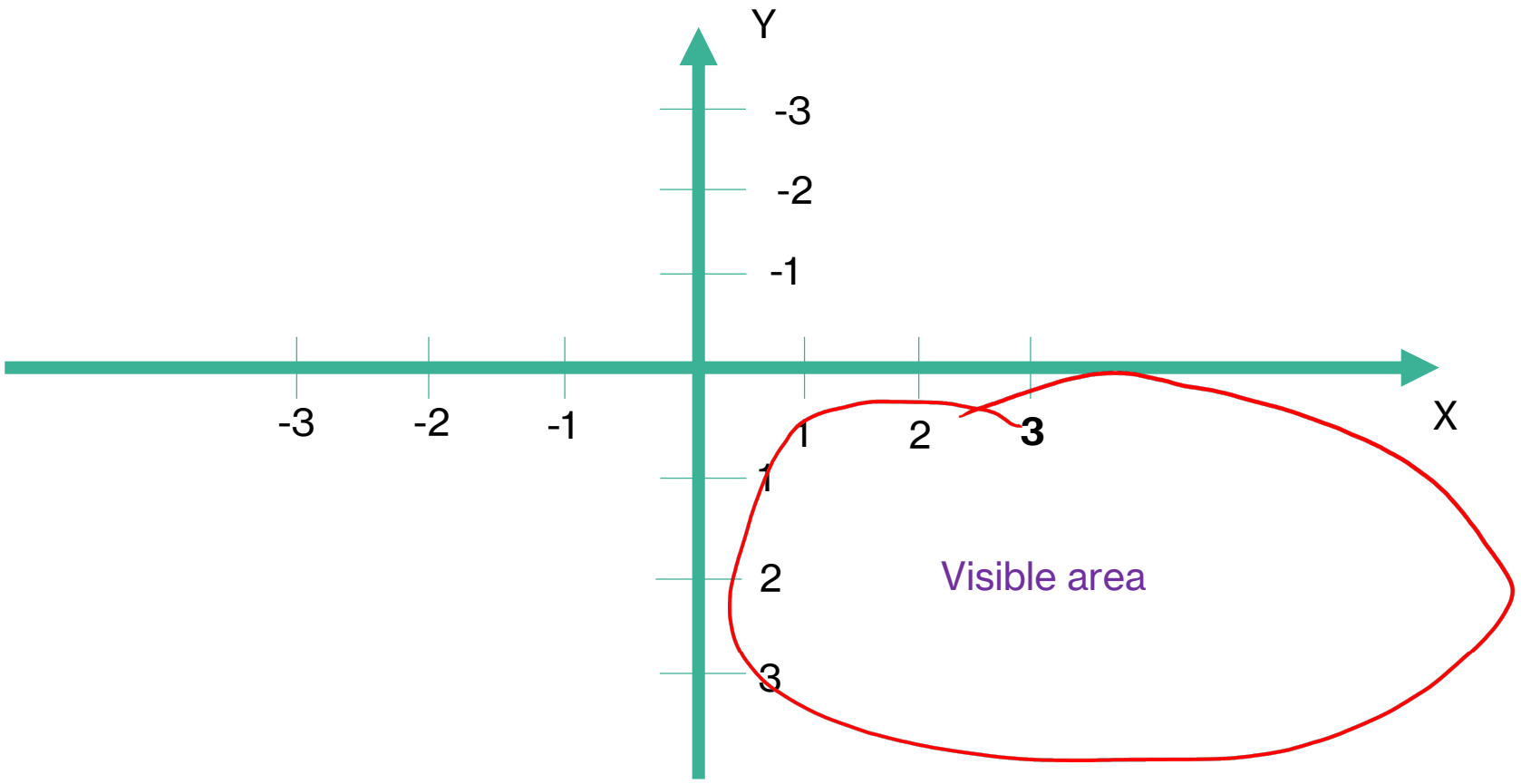
Characters for internet games, animations for websites

Create business (2D) diagrams

Creating charts (column, pie)

Creation of logos, icons

Coordinate system

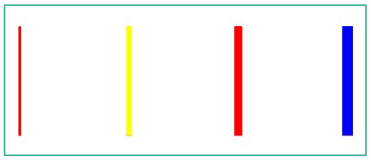
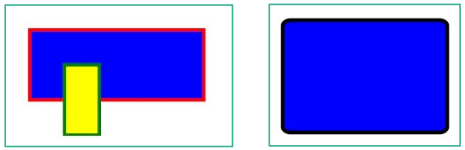
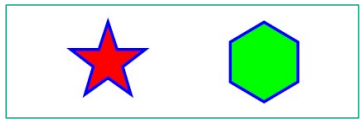
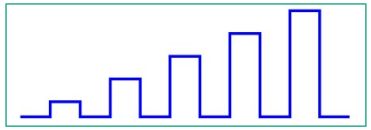


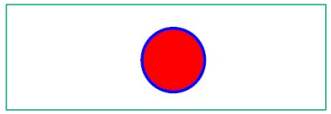
The structure of the document

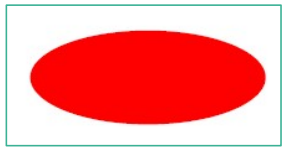
```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
         "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="350px" height="80px" xmlns="http://www.w3.org/2000/svg">
  <title> The first SVG document </title>
  <!-- Comments like in HTML -->
  <g id="tekst1">
    <text x="120" y="40" font-face="Verdana" font-size="24">
          Hello World </text>
  </g>
</svg>
```

```
<!DOCTYPE html>
<html>
  <body>
    <svg width="500" height="500">
         <rect x="150" y="350" rx="30" ry="20" width="200" height="100"
                  style="stroke:red; fill:green; stroke-width: 5; opacity:0.7"/>
     </svg>

     <img src="fig1.svg" border="1" />
  </body>
</html>
```

Basic shapes

- Lines 
- Rectangle 
- Poligon 
- Curve 

Circle 

Ellipse 

Paths 

Text

Paths

```
<path d="M100,100 m100,100" />
```

Creates a virtual pen in the d attribute and draws whatever we want

Characteristic elements:

M or **m** -> move to (set the feathers at a specific point - we give the coordinates)

L or **l** -> line to command (draws a line from a given point to a new one given by coordinates)

A or **a** -> draw arcs

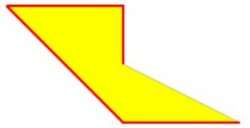
Q or **q** -> quadratic bezier curves

C or **c** -> cubic bezier curves

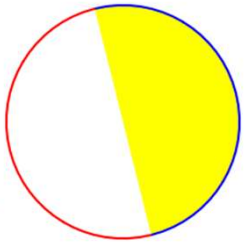
M - absolute pen position

m - relative pen position (x = 100 + 100)

Paths

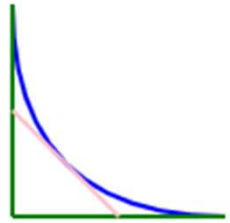


```
<svg width="500" height="500">
        <path d="M100,100 L200,100 L200,150 M300,200 L200,200 L100,100"
                style="stroke:red; stroke-width:2; fill:yellow"/>
</svg>
```



```
<svg xmlns="http://www.w3.org/2000/svg" width="500" height="500">
        <path d="M100,100 A50,50 0 0,1 150,300" style="stroke:blue; stroke-width:2; fill:yellow"/>
        <path d="M100,100 A50,50 0 0,0 150,300" style="stroke:red; stroke-width:2; fill:none"/>
</svg>
```

Paths



```
<svg width="500" height="500">
        <path d="M100,100 Q100,200 200,200 " style="stroke:blue; stroke-width:2; fill:none" />

        <line x1="100" y1="100" x2="100" y2="200" style="stroke:green; stroke-width:2;" />
        <line x1="100" y1="200" x2="200" y2="200" style="stroke:green; stroke-width:2;" />
        <line x1="100" y1="150" x2="150" y2="200" style="stroke:pink; stroke-width:2;" />
</svg>
```

```
<svg width="500" height="500">
        <path d="M100,100 C125,150 175,50 200,100 " style="stroke:blue; stroke-width:2; fill:none" />
        <path d="M100,100 C125,50 175,150 200,100 " style="stroke:yellow; stroke-width:2; fill:none" />
        <path d="M100,100 C155,30 175,150 200,100 " style="stroke:green; stroke-width:2; fill:none" />
        <path d="M100,100 C125,50 175,50 200,100 " style="stroke:pink; stroke-width:2; fill:none" />
</svg>
```

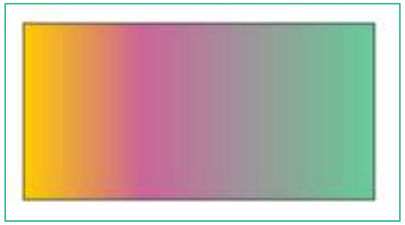
Fillings

- Filling the area with a solid color through the `fill` attribute
  - The value of this attribute: name e.g. `blue` or in `rgb(0, 0, 255)`
    or `#0000FF`
- Filling the area with a linear gradients or radial gradient
  - We define the gradient in the `<defs>` element

Linear gradient

- Definition in the `<linearGradient>` element
- Attributes
  - `id` - definition of the gradient name
  - `offset` - an attribute defining the color limit, i.e. the place of its occurrence (defined in percent)
  - `stop-color` - the name of the color in this place

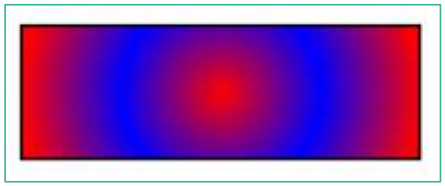
```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="400px" height="200px" viewBox="0 0 400 200" preserveAspectRatio="xMinYMin meet"
          xmlns="http://www.w3.org/2000/svg">
  <title>Three-stop gradient</title>
  <defs>
    <linearGradient id="three_stops">
      <stop offset="0%" style="stop-color: #ffcc00;"/>
      <stop offset="33.3%" style="stop-color: #cc6699"/>
      <stop offset="100%" style="stop-color: #66cc99;"/>
    </linearGradient>
  </defs>
  <rect x="20" y="20" width="200" height="100"
          style="fill:url(#three_stops); stroke: black;"/>
</svg>
```



Radial gradient

- Definition in the `<radialGradient>` element
- Attributes
  - `cx` and `cy` - the coordinates of the center of the outer gradient circle
  - `fx` and `fy` - the coordinates of the gradient center (if there are no parameters, values equal to cx and cy are assumed)
  - `r` - radius
  - `id` - definition of the gradient name
  - `offset` - an attribute defining the color limit, i.e. the place of its occurrence (defined in percent)
  - `stop-color` – the name of the color in this place

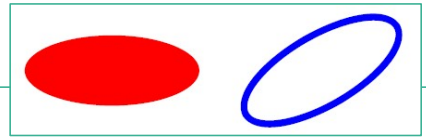
```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="8cm" height="4cm" viewBox="0 0 800 400" xmlns="http://www.w3.org/2000/svg">
  <g>
    <defs>
      <radialGradient id="MyGradient" gradientUnits="userSpaceOnUse"
                      cx="400" cy="200" r="300" fx="400" fy="200">
        <stop offset="0%" stop-color="red" />
        <stop offset="50%" stop-color="blue" />
        <stop offset="100%" stop-color="red" />
      </radialGradient>
    </defs>
    <rect fill="url(#MyGradient)" stroke="black" stroke-width="5"
          x="100" y="100" width="600" height="200"/>
  </g>
</svg>
```



Transformations

- Objects can be transformed by changing their position, rotation, inclination
- Implemented by the `<transform>` attribute
- Attribute values
  - `translate(tx [ty])` - object offset of `<tx>` units relative to x-axis and o `<ty>` units along the y axis
  - `scale(sx [sy])` - resizing the object; if `<sy>` is not given, it is assumed to be the same as `<sx>` (e.g. value `<sx>` is 0.5, then the object is reduced by 50% of the original width)
  - `rotate(rotate-angle)` - rotation of the object by a given angle around the central point (0,0) (e.g. if the rotate-angle is 45, the object will rotate 45 degrees clockwise)
  - `skewX(angle)` - tilts the element relative to the x axis by `angle`
  - `skewY(angle)` - bends the element relative to the y axis by `angle`

Transformations



```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="12cm" height="4cm" viewBox="0 0 1200 400" xmlns="http://www.w3.org/2000/svg">
  <rect x="1" y="1" width="1198" height="398" fill="none" stroke="blue" stroke-width="2" />
  <g transform="translate(300 200)">
    <ellipse rx="250" ry="100" fill="red"  />
  </g>
  <ellipse transform="translate(900 200) rotate(-30)" rx="250" ry="100" fill="none"
        stroke="blue" stroke-width="20"  />
</svg>
```

Filters

- We create using the `<filter>` element with the `id` parameter in the `<defs` element
- By creating our own filters, we use the following functions
  - `feBlend` - is used to mix two objects - merge
  - `feColorMatrix` - used for discoloring, brightening colors, luminance
  - `feComponentTransfer` - is used to adjust brightness, contrast, object thresholds
  - `feComposite` - decides about the mutual location of two objects and their mutual interpenetration
  - `feConvolveMatrix` - used to smudge objects
  - `feDiffuseLighting` - is used to control the light source on objects. This light can be
    - `feDistantLight` - a distant source of light
    - `fePointLight` - point light source
    - `feSpotLight` - a source of scattered light
  - `feDisplacementMap` - defines the movement of colors from one object to another
  - `eFlood` - is used to fill the object with colors
  - `feGaussianBlur` - applies the GaussianBlur effect on the object
  - `feImage` - allows you to load another object into the object
  - `feMerge` - allows you to combine different effects
  - `feMorphology` - is used to increase or blend effects on an alpha channel
  - `feOffset` - specifies the location of the object relative to other objects
  - `feSpecularLighting` - creates a light source similar to a reflector
  - `feTile` - used to replicate the object (duplicate)
  - `feTurbulence` - used to create textures

Animations

- The `<animate>` element changes the attributes of the element you want to animate
- Attributes (common to all `animate` element)
  - `attributeName` - the name of the attribute
  - `attributeType` - attribute type (CSS, XML or auto)
  - `begin` - start time of change
  - `end` - end of the attribute change
  - `dur` - duration of change
  - `fill` - final state parameter (`freeze`, `remove`)

```
<svg width="710" height="210">
 <rect x="0" y="0" width="0" height="0" fill="rgb(255,255,0)" >
   <animate attributeName="width" attributeType="XML" begin="0s" dur="9s" fill="freeze"   from="0" to="600" />
   <animate attributeName="height" attributeType="XML" begin="0s" dur="9s" fill="freeze"  from="0" to="220" />
  </rect>
</svg>
```

```
<svg width="500" height="500">
  <rect x="10" y="20" width="90" height="60">
    <animate id="a1" attributeName="fill" from="red" to="blue" dur="3s" fill="freeze" />
  </rect>
  <rect x="10" y="120" width="90" height="60">
    <animate id="a2" attributeName="fill" from="blue" to="yellow" begin="a1.end" dur="3s" fill="freeze" />
  </rect>
</svg>
```

Animations – movement on the track

- The `<animateMotion>` element allows you to move the beginning of the object along the specified path
- attributes
  - `path` - specifies the path on which the move takes place

```
<svg width="710" height="210">
  <text x="0" y="0" font-family="Verdana" font-size="45" >
    Ruchomy tekst
    <animateMotion path="M40,10 L40,100 L40,10" dur="5s" repeatCount="indefinite"/>
  </text>
</svg>
```

Animation – transformations

- The `<animateTransform>` element is used to give objects such transformations as `translate`, `scale`, `rotate`, `skewX`, `skewY`
- attributes
  - `type` - specifies the type of transformation
  - `from` - specifies the initial value of the transformation parameter
  - `to` - specifies the final value of the transformation parameter

```
<svg width="500" height="500">
  <text style="font-family: Verdana, Arial, Helvetica, sans-serif; font-size:12pt;
        font-style:normal" x="0" y="0">
    Tekst obracający się ...
    <animateTransform attributeName="transform" attributeType="XML" dur="5s"
       repeatCount="indefinite" type="rotate" from="0" to="360" fill="freeze" />
  </text>
</svg>
```

XML

Extensible **M**arkup **L**anguage

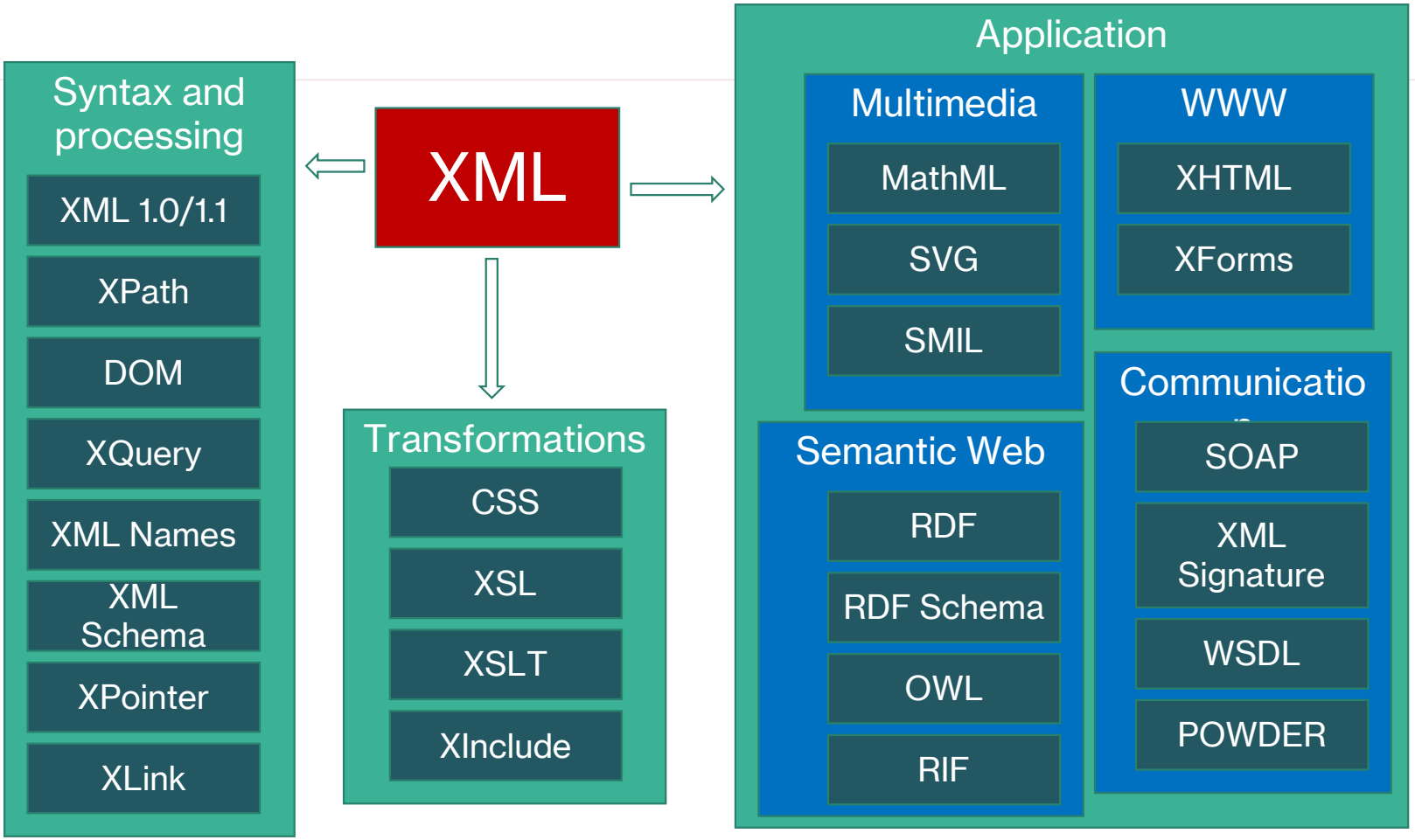
Road to XML

- 1967–1970s – William Tunnicliffe, **GenCode**
- Late 1960s – IBM – SCRIPT project, INTIME experiment
  - Charles **G**oldfarb, Edward **M**osher, Raymond **L**orie
  - **G**eneralized **M**arkup **L**anguage (**GML**)
- 1974–1986 – Standard Generalized Markup Language (**SGML**)
  - ISO 8879:1986
- Late 1990s – E**x**tensible **M**arkup **L**anguage (XML)
  - W3C Recommendation 1998
  - Simplification(!) and subset of SGML

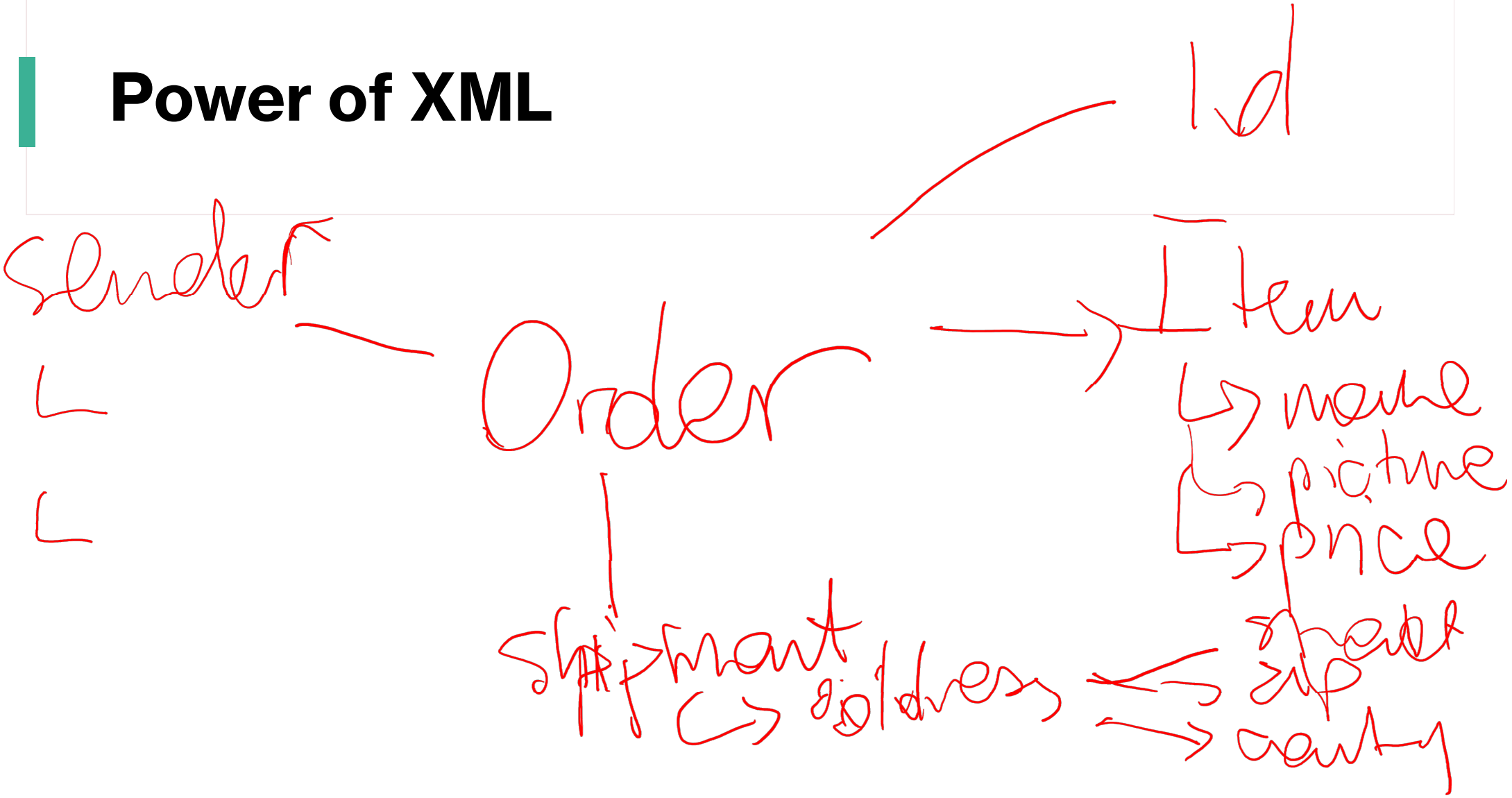
What is XML?

- **Standard** – Extensible Markup Language World Wide Web Consortium (W3C) Recommendation
  - version 1.0 – 1998
  - version 1.1 – 2004
- **Language** – a format for writing structural documents in text files
- **Metalanguage** – an extensible and growing family of concrete languages (XHTML, SVG, etc...)
- Means of: (two primary applications)
  - document markup
  - carrying data (for storage or transmission)

XML-related family of standards



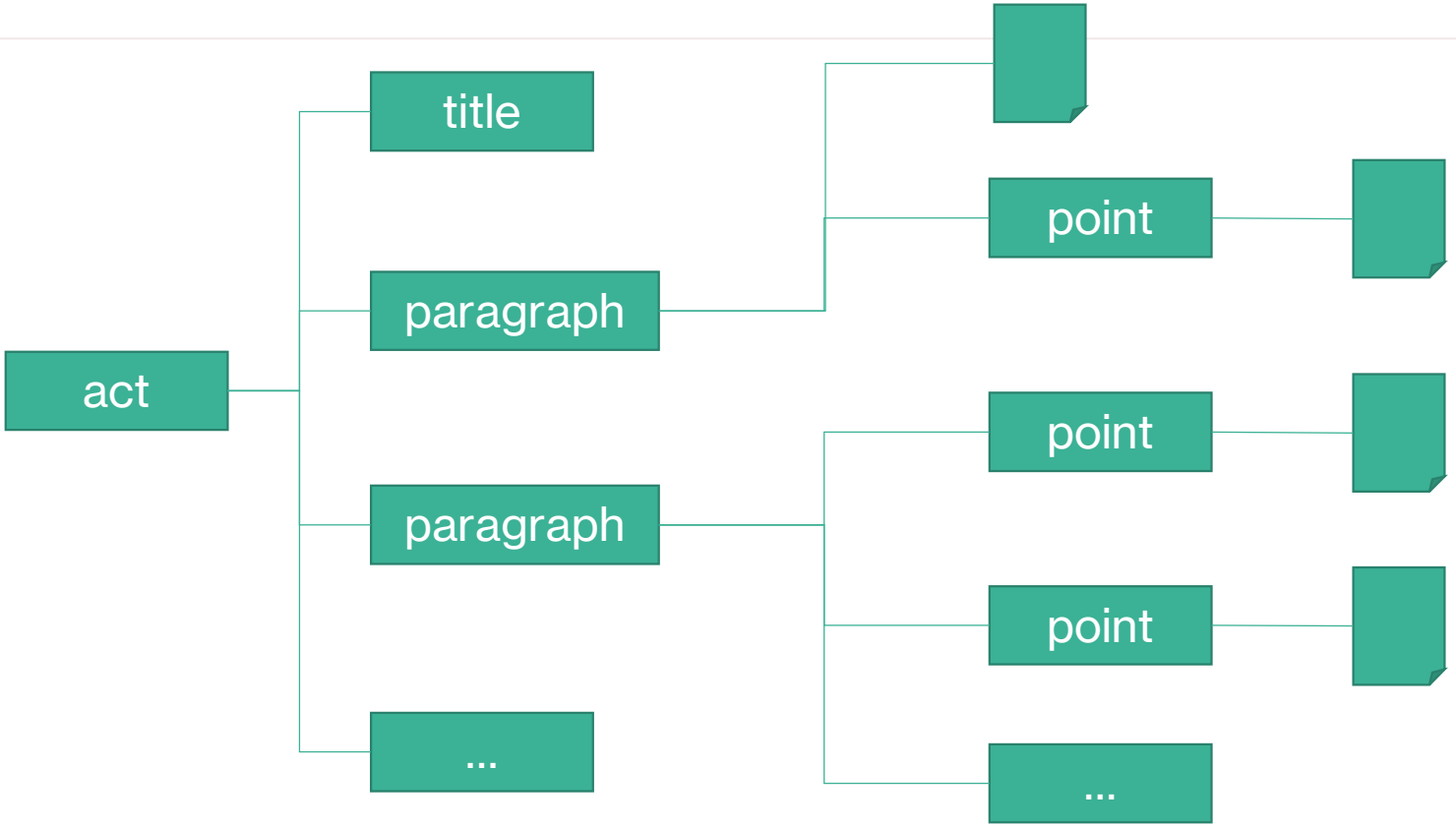
Power of XML


id
sender
Order
Item
name
picture
price
shipment
address
zip
country

What does XML look like?

```
<?xml version="1.0" encoding="UTF-8"?>
<act>
  <title> o realizacji przedpłat na samochody </title>
  <paragraph no="1">
   Osoby, które w 1981 r. wniosły przedpłaty na zakup
   samochodów i dotychczas samochodu nie odebrały, mają
   prawo do otrzymania rekompensaty w wysokości:
    <point no="1">
      5.930 zł, jeśli przedpłata została wniesiona na
      samochód marki Fiat 126p,
    </point>
    <point no="2">
      8.400 zł dla samochodu marki FSO 1500.
    </point>
  </paragraph>
</act>
```

XML document – tree

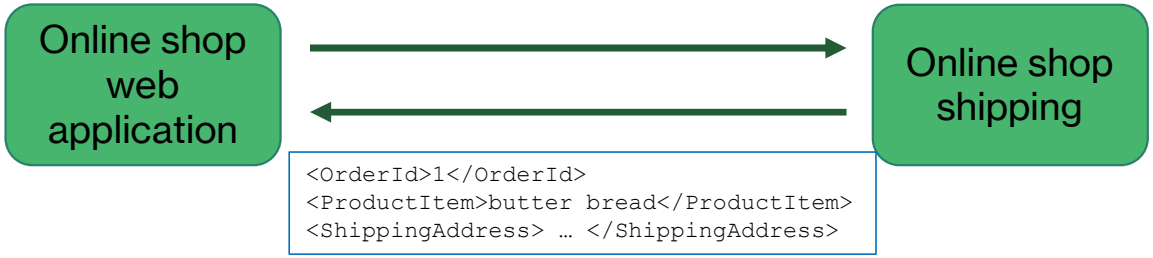


Why XML?

- We can create our own language

```
<Order>
      <OrderId>1</OrderId>
      <ProductItem>butter bread</ProductItem>
      <ShippingAddress> … </ShippingAddress>
</Order>
```

- It contains data as well as metadata

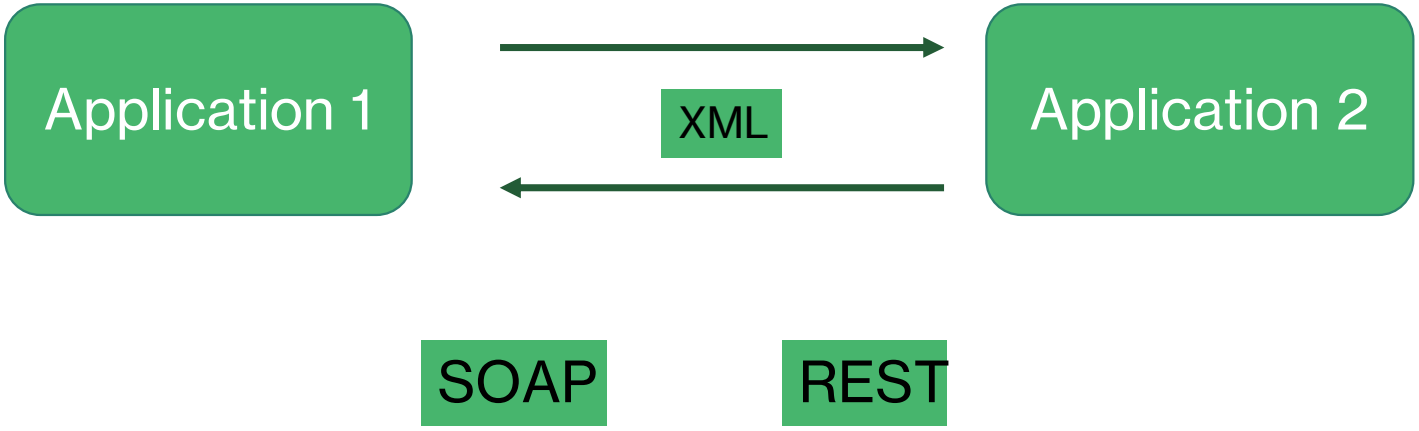


- Checked correctness of documents

Well-Formedness: XML rules how to create XML documents

Validation: XML + DTD/XML Schema definition

When XML?

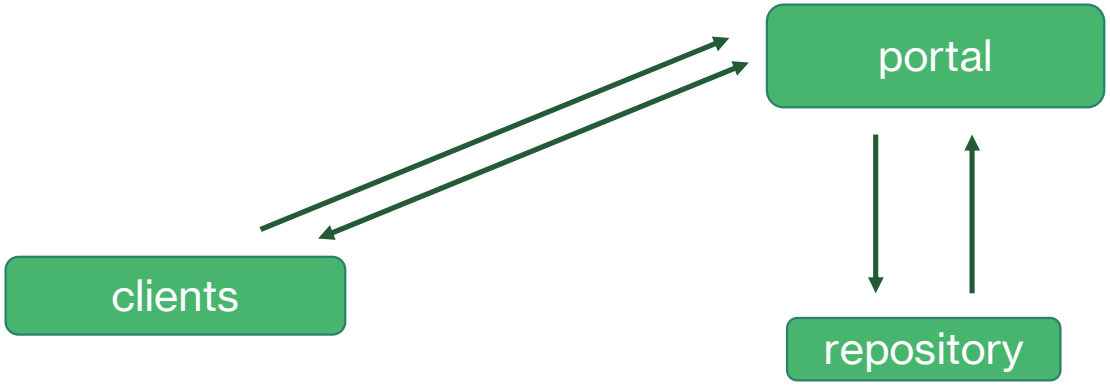


When XML?

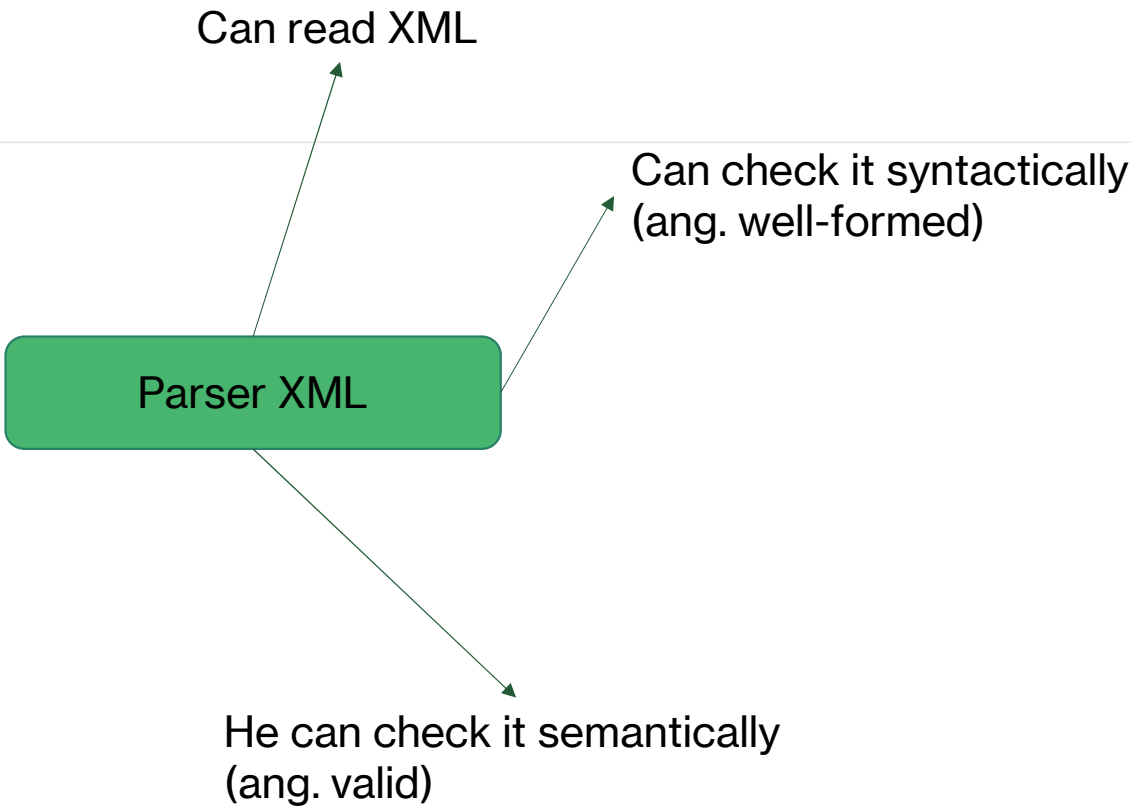
Configuration files

web.xml | server.xml | pom.xml | build.xml

Record, modification, presentation



Processor XML



XML declaration

- Always at the beginning of the document (on the first line, from the first character)
- The declaration can be omitted if the document is version 1.0 and its encoding is UTF-8 or UTF-16

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
```

Control / processing instructions

- They can occur outside the main element
- The name of the processing instruction must be different from the string "XML" (and variants with different case)

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<?xml-stylesheet href="style.css" type="text/css" ?>

<student>John Doe</student>
```

Elements

- The document contains exactly one root element
- Elements can contain sub-elements and text nodes

```
<student>John Doe</student>
```

- Name of the element
  - Case sensitive
  - May contain
    - Alphanumeric characters and numbers
    - National characters (e.g. ąęśćńś)
    - Punctuation marks (_ - .)
    - It can start with a letter or an underscore

Attributes

- Values must be enclosed in double quotes or single apostrophes
- One element cannot have two attributes with the same name
- The attribute value cannot contain the <

```
<student semester="3">John Doe</student>
```

```
<registration forSemester="zimowy" academicYear="2022/2023"/>
```

Comment

- They can be multi-line
- They can contain any characters <,>, &
- They cannot contain two - in the content
- They cannot be inserted into tag labels
- They can occur outside the main element

```
<!-- this is my comment and no one will process it -->
<student>John Doe</student>
```

PREDEFINED entities

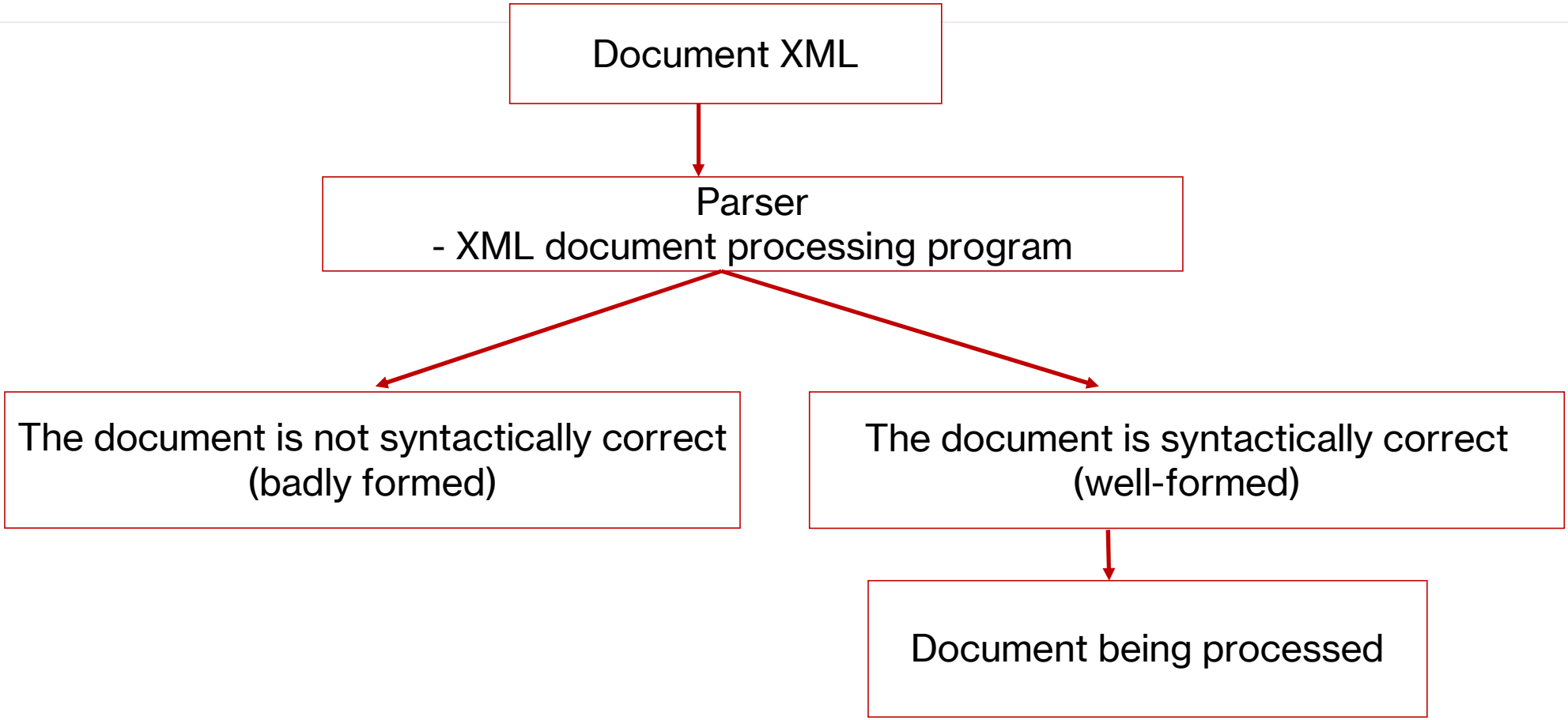
- Characters reserved for XML syntax
  - `&` for &
  - `<` for <
  - `>` for >
  - `&apos;` for '
  - `&quot;` for "

CDATA section

- One forbidden string is]]> (also in text content – in the CDATA section you can quote an XML document without the CDATA section)
- CDATA sections cannot be nested

```
<student>
   <![CDATA[
         <svg xmlns="http://www.w3.org/2000/svg"
              width="12cm" height="10cm">
            <ellispe rx="110" rx="130"/>
            <rect x="4cm" y="1cm" width="6cm" height="3cm" />
         </svg>
  ]]>
</student>
```

Syntactic correctness



Syntactic correctness (well-formed)

- There must be only one root element (root, root)
- XML elements must be closed
- XML elements must be properly nested
- Elements names are case sensitive
- XML attribute values must be enclosed in quotation marks or apostrophes

Summary

In XML we can define our own language, build a hierarchy of elements and relationships between components

Other **XML**-based languages create their own tags and define its meaning

XML is supported by the **W3C** (**World Wide Web Consortium**), which supports languages related to technologies found on the Web