Hypertext and hypermedia
Project

Theme:

MY HOBBY

Stages:

Stage	Score [pts]
HTML, XML document, XML Schema, DTD	25
XSLT	15

Stage 1: HTML (11pts), XML document (1pts), XML Schema (11pts), DTD (2pts)

HTML: (11 pts)
Requirements:

- contents of the page adequate for the subject of the project
- HTML5,
- responsive website (at least it supports two different screen sizes) (**1,6pts**)
    - viewport
    - media queries (@media)
    - repositioning the menu
    - using viewport-width property
- validation of the HTML **(0,9pts) (**https://validator.w3.org/**)**
- validation of the css (**0,5pkt**) (http://jigsaw.w3.org/css-validator/)
- page layout:
    - dividing the page into several items (header, menu, footer, content field)**(1 pts)**
    - using semantic tags (header, footer, nav, figure …) **(0,6 pts)**
- dividing the contents into several files (at least 3) **(0,6pts)**
- menu containing at least 3 options, and one of them with subordinate options; options must be available for selecting **(0,6 pts)**
- inserting multimedia:
    - graphics
        - photo gallery (raster graphics) (at least 5) must be organized as icons that can be enlarged **(0,6 pts)**
        - simple SVG vector graphics, embedded in the HTML file in <svg></svg> tags **(0,5 pts)**
        - Animation using CSS3 keyframes (@keyframes) and transition effect **(0,6pts)**
- putting on the page:
    - table **(0,2 pts)**
    - links to external web pages **(0,3 pts)**
    - links (min 2) to selected locations in the text or to the top of the page (the text should be sufficiently long to demonstrate this) **(0,4 pts)**
- styles must be defined in a separate style sheet, use CSS mechanism
    - different styles for at least 4 selectors (groups of selectors) **(0,6 pts)**
    - classes (at least 3) **(0,6 pts)**
    - identifiers (at least 1) **(0,2 pts)**
    - making use of a pseudo-class (min 2) **(0,2 pts)** and
    - pseudo-element **(0,1 pts)**
- creating a simple survey form **(0,8 pts)**
    - at least 7 fields for data entry

- at least 5 different sorts of fields for data input
- buttons for clearing the form and for sending the data (don't use the mailto in *action* attribute of form)

– taking care for aesthetic look of the page

XML: (1pts)

Requirements:

– create an XML file containing data related to the project topic. The data contained in the XML document should correspond to the data presented on the HTML page. It is an error to create an XML document containing tags responsible for the presentation of data - reflecting the arrangement of data on the HTML page created in the previous point. An XML file should be created paying attention to the data placed in it, the relationship between them and not how they were placed on the HTML page.

In XML file:

- tag names, hierarchy are to be information about stored data
- there must be at least 4 levels, excluding the root (**0,3pts**)
- use at least 5 different attributes (**0,3pts**)
- the XML document shall contain at least 12 different element names (**0,2pts**)
- data must be provided for at least three root sub-elements
- photographs must be included (at least 3) (**0,1pts**)
- there must be links (at least 3) (**0,1pts**)

XML SCHEMA: (11pts)

– To enforce the appropriate syntax of the XML file, design and create an XML Schema file.

– The XML file must be syntactially and semantically correct. The structure of the XML file must be compatible with that defined in the XML schema. To check the correctness, use a validator at (**https://www.corefiling.com/opensource/schemavalidate/**).

– Take a closer look at the form of the document, i.e. the notation, use of indentation showing the structure of the document, appropriate names of tags and attributes adequate to their contents.

Detailed requirements:

In the XML Schema file, declare and use:

- at least 6 definitions of global complex types **(1,6 pts)**
- at least 5 definitions of global simple types **(1,6 pts)**
- at least 2 definitions of local complex types **(0,8 pts)**
- at least 2 definitions of local simple types **(0,8 pts)**
- using different order indicators (choice, sequence, all), mixed content models **(0,3 pts)**
- at least one definition of a group (of elements or attributes) **(0,2 pts)**
- presence of at least 4 nested levels in the structure of the XML document **(0,4 pts)**
- declaration of at least 6 attributes, of which at least 1 is defined globally and used at least twice **(1,2 pts)**
- various declarations of at least 12 different elements (0,6 pts), of which at least 5 should contain sub-elements (1 pts) **(1,6 pts)**
- using facets (restrictions on elements and attributes)
  - *length, minLength, maxLength, maxInclusive, minInclusive,maxExclusive, minExclusive,* (min 4 chosen) **(0,4 pts)**
  - *pattern* and *enumeration* **(0,6 pts)**
- derivation of types **(0,3 pts)**
  - *extension* (extending an existing type with additional elements)
- at least 3 references to elements and/or attributes (there must be both a reference to an element and to an attribute) **(0,6 pts)**
- using a list (0,05 pts), setting its length, i.e. the number of items ((0,05 pts) and restricting their values, (0,05 pts), the list must also be used in the XML file (0,05 pts) **(0,2 pts)**

- using a union **(0,2 pts)**
- using at least 4 different built-in types (**0,2 pts**)
- validation of the file
- at least 3 filled sub-elements of the root in the XML file

XML, DTD: (2pts)

Requirements:

– For an XML file (used in an XML Schema task), in order to force its proper syntax, design and create a DTD file.

– The XML file must be syntactically and semantically correct. The structure of the XML file must be in accordance with the DTD. To verify the correctness, use the validator given on the enauczanie (Moodle).

– It is also important to pay attention to the form of the document, i.e. the notation, the use of spacing that illustrate the structure of data, appropriate (adequate to the content contained in them) naming tags, attributes.

Detailed requirements:

In the DTD file, declare/define and use::

- elements declarations, you should use sequence and choice (**0,5pts**)
- using at least 6 attributes (**0,5pts**)
- attributes declarations, you should use:
  - enumeration type (**0,3pts**)
  - default value (**0,3pts**)
  - use attribute (**0,2pts**)
- parameter entity at least 2 times (**0,2 pts**)
- the using of ANY is forbidden

Selected examples of errors occurring in the schemes:

- creating an XML document containing tags reflecting the way data is presented on the page, e.g. <subpage> < paragraph ></ paragraph > </suppage> and not the data itself (**-2pts**)
- validation errors XML(the file will not validate) (**up to -10 pts**)
- trivial definition of a simple type (e.g. a simple type being a simple string type) (up to **-2 pts**)
- repetition of type definitions (multiple definitions of the same types) (up to **-2 pts**)
- using anyType (**up to -10 pts**)
- slightly modified, generated XSD file (**up to -10 pts**)
- incorrect definition of elements, attributes, structure (**up to -6 pts**)
  - e.g. instead of using maxOccurs=4, quadruple declaration of the same type
- missing photos (in XML (min 4) and in Schema) (**-0,5 pts**)
- missing links (in XML (min 4) and in Schema) (**-0,5 pts**)
- missing at least 3 filled subelements of the root in the XML file (**-2 pts**)
- using ANY in DTD (**-1pts**)

Note

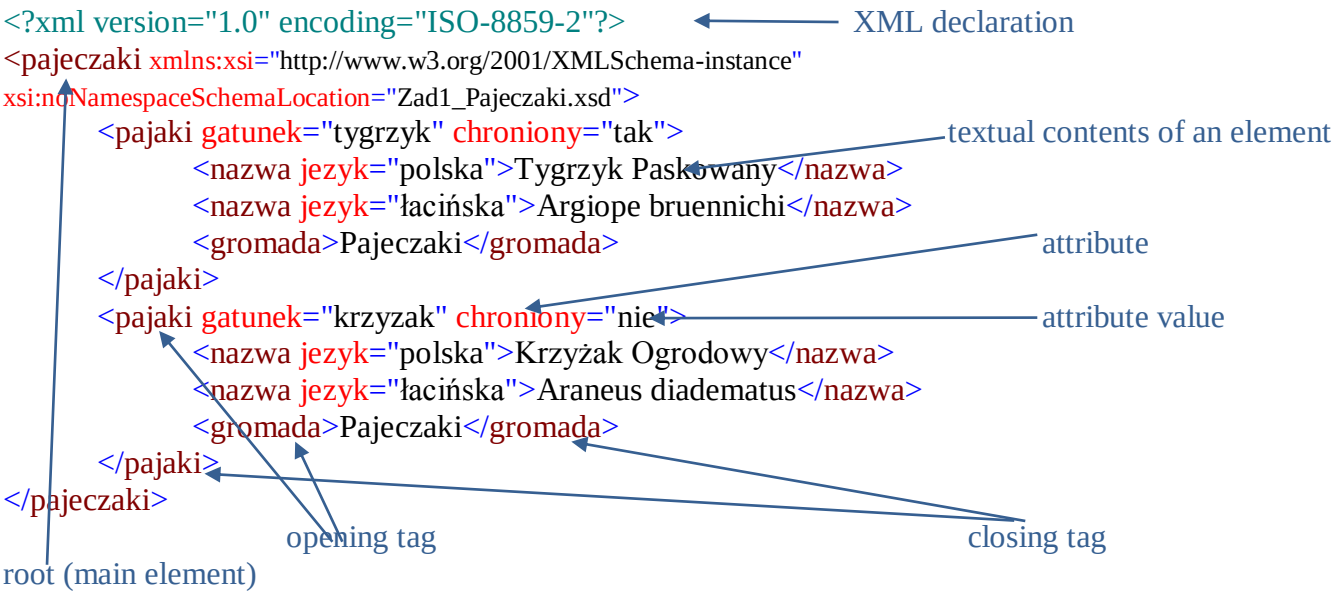
- The final number of points for the project depends on the answers given during project evaluation, grasp of the project and of the theory.
- During the evaluation the project **code** must be **free** of any **comments.**

Evaluation of projects

- every person has a specific appointment time slot: day, time and an examiner. Evaluation of the project is carried out only during that time slot.
- Evaluation of the project will be held online
- correction of already evaluated projects is not possible

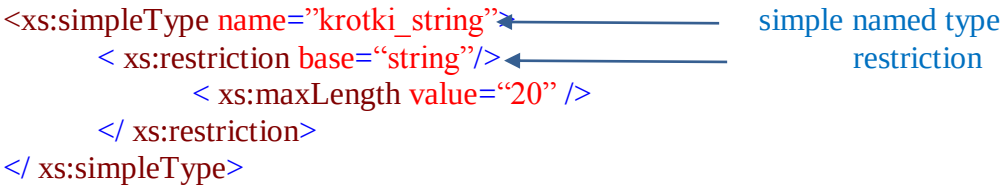
- **XML and XML Schema**

XML
- all nonempty elements should have an opening and a closing tag
- elements can be nested, but they cannot overlap
- case sensitive

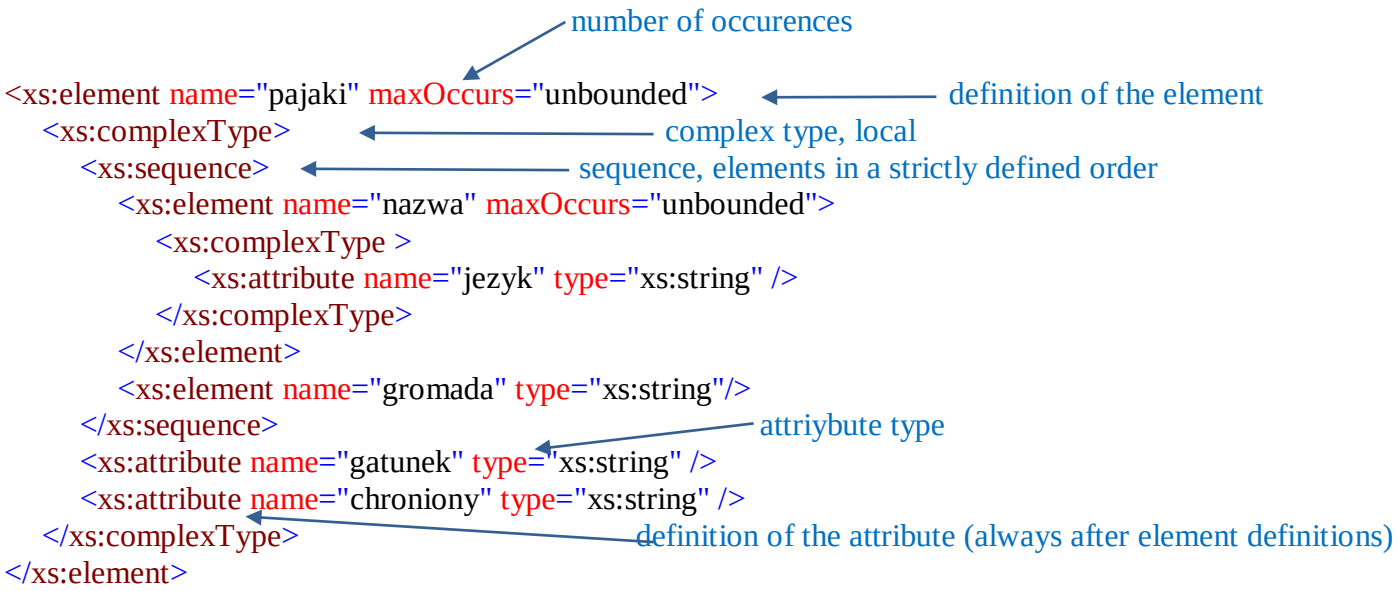


XML Schema
To create:
- only an element with a textual contents
  - simple type
- element with subelements
  - complex type
- element z podelementami i atrybutami
  - typ złożony
- element with mixed contents (subelements and text)
  - complex type with mixed=true attribute
- element with attributes
  - complex type
- element with attributes and simple contents
  - complex type with simpleContent

1) Definition of a simple named type



2) Definition of an element



3) Enumerations – lists of predefined values

```
<xs:simpleType name=“nazwa_typu” >
  < xs:restriction base=“string”>
     < xs:enumeration value=“wartosc1” />
     < xs:enumeration value=“ wartosc2” />
     < xs:enumeration value=“ wartosc3” />
  </ xs:restriction>
</ xs:simpleType>
```

4) SimpleContent

When creating a type derived from a simple type or other complex type with a simple content. In this way, attributes can be added to the base type.

```
<xs:complexType name="nazwa_typu">
   <xs:simpleContent>
      <xs:extension base="xs:string">
         <xs:attribute name="nazwa_atrybutu" type="xs:string"/>
      </xs:extension>
   </xs:simpleContent>
</xs:complexType>
```

5) References to an element

```
<xs:element name="data" type="xs:date"/>          global element definition

<xs:element ref="data" minOccurs="0"/>            reference to the globally defined element
```

DTD

1) Element declaration

```
<!ELEMENT name type>
```

2) the number of occurrences

? 0 or 1
\+ 1 or unbounded
* 0 or unbounded

3) attribute declaration

<!ATTLIST element-name
attribute-name1 attribute1-type *description1*
attribute-name2 attribute2-type *description2*
...>

Attribute type

CDATA	text,
ID	name unique in the whole document, for a given type of element only one attribute can be declared as ID,
IDREF	a name appearing somewhere in the document as an ID value

description: określa czy atrybut jest wymagany i jaką ma wartość domyślną

#REQUIRED	#IMPLIED
"default-value"	#FIXED "value"

4) Entity

- internal
  <!ENTITY *name* '*text*'**>**
- external
  <!ENTITY *name* **SYSTEM '***file_name***'>**
- parameters
  <!ENTITY % *name* '*text*'>