# XML / JSON Exercises

## Exercise 1

The XML file is saved into the XML folder.

## Exercise 2

XML NameSpaces

The XML 1.0 version specification did not have namespace support thus it was very difficult to create a unique name identifiers for the XML document structure. As a result, the developer used only one global namespace for the XML documentation, which was not a bulletproof solution because two pieces of software couldn’t avoid name collision based on the data type representation. The solution is that to embed an extra information into n XML structure, called the Namespace. This uniquely identifies set of names so that technique provides source origin validation, apparently.

The XML namespace convention must follow the generic syntax for URIs defined by RFC 2396. In XML URI references identify abstract resources. The W3C introduced the formal specification of the XML document along with names of element and attributes, however, all the names are defined by developers and followed by certain name convention/rules. These details provide a unique origin id of the document structure therefore the software can recognize the producer/owner of the XML data since the namespaces point to the correct location. The Namespace can be a fiction name, however, good practise to apply with a valid unique URI name.

**Default Namespace:** It is declared with xmlns attribute without prefix, however, it can be applied only to un-prefixed element and its descendant elements.

e.g : “xmlns=<http://www.w3.org/1999/html>” 🡪

<svg “xmlns=<http://www.w3.org/1999/html>” width=”12” height=”10cm” >

<circle r=”4cm” border=”1px solid” />

</svg>

**User Defined Namespace:** It is declared with a xmlns attribute element with a prefix value. (xmlns: thisIsThePrefixValue)

“xmlns: thisIsThePrefixValue =<http://www.w3.org/1999/html>” 🡪

< “xmlns**: thisIsThePrefixValue** =<http://www.w3.org/1999/html>” width=”12” height=”10cm” >

< **thisIsThePrefixValue** : circle r=”4cm” border=”1px solid” />

</svg>

**Schema Namespace:** every schema has one target namespace and possibly many source namespaces.

* **Target Namespaces** for the XML schema document itself. It is going to be assigned to the schema which is created by developer.
* **Source Namespaces** external to the XML schema document

<xyz:schema targetNamespace=<http://sabi11.ddns.net/watchEyeApp>

xmlns:xyz=<http://sabi11.ddns.net/XMLSchema>

xmlns:demo=http://sabi11.ddns.net/watchEyeApp

<xyz:element name=”id” type=”xyz:positiveInteger”/>

<xyz:element name=”name” type=”demo:String” />

……..

</xyz:schema>

The following XML fragment shows characteristics of the namespaces.

**Xmlns** is just a shorthand of the placeholders for the full names.

XYZ represents the namespace of the URL : whose name is <http://www.sunshine.com/holidays>

The h letter is belong to the <http://www.w3.org/SH/2016/html4> domain address.

<h:html xmlns:**xyz**="http://www.sunshine.com/holidays"

xmlns:h=<http://www.w3.org/SH/2016/html4>

xmlns=http://www.w3.org/defaultNamespace >

<h:head>

<h:title>Booking Holiday</h:title>

</h:head>

<h:body>

<**xyz**:bookingreview>

<**xyz**:title>Details</**xyz**:title>

<h:table>

<h:tr align="center">

<h:td>Customer</h:td>

<h:td>Price</h:td>

<h:td>Dest.</h:td>

<h:td>Date</h:td>

</h:tr>

<h:tr align="left">

<h:td><**xyz**:customer>Joe</**xyz**:customer></h:td>

<h:td><**xyz**:price>319</**xyz**:price></h:td>

<h:td><**xyz**:dest>352</**xyz**:dest></h:td>

<h:td><**xyz**:date>2016/01/31</**xyz**:date></h:td>

</h:tr>

</h:table>

</**xyz**:bookingreview>

</h:body>

</h:html>

## Exercise 3

Schema files are saved into the SCHEMA folder.

## Exercise 4

JSON file is saved into the JSON folder.