Theme 3 Namespaces

XML Namespaces

- Sometimes more than one custom XML language...
- ...is used in the same XML doc.

- If both tag sets have a tag with the same name...
- ...it causes **ambiguity** in the doc.

- Solve it by assigning each language a unique **namespace**.

XML Namespaces

A **namespace**:

- A collection of **related elements**.
- They are identified by a unique **namespace name**.

Designing a Namespace Name

- Each namespace must have a **unique**, **permanent** name.

- A **URI** is the most unique kind of name on the web.
  - **URI** = **U**niform **R**esource **I**dentifier.
  - **E.g.:** `http://www.domain.com/ns/1.0`

- The URI may point to an existing location, but it **doesn't have to**. Any such location is ignored during processing.

Designing a Namespace Name

- The name usually contains some info about the language...
- ...such as its origin, version number, etc.
  - Think about DTD FPI

- Any URI is acceptable, as long as it is **unique** on the web.

Using Namespaces

Declaring a Default Namespace

- Usually a namespace has a **prefix**.
  - Used to label all tags that belong in the namespace.
  - Defined by the author **using** the namespace.

- You can also declare a **default namespace** (no prefix).
  - Declare it in the starting tag of an element...
  - ...then it and its descendants belong to that namespace.

Declaring a Default Namespace

- **E.g. declaring a default namespace for all elements:**

```
<theRoot xmlns="http://www.domain.com/ns/1.0">
```

- **E.g. only for a certain element and its descendants:**

```
<someElement xmlns="http://www.domain.com/ns/1.0">
```

- Declaration are the same, only difference is the placement

Using Prefixes

- Use a **prefix** to explicitly label elements...
- ...that should belong to the associated namespace.

- **E.g. to declare a prefix and label elements:**

```
<wat:someElement xmlns:wat="http://www.lolwut.com">
        <wat:someDescendant/>
</wat:someElement>
```

Using Prefixes

- You can label **someElement**…
- …and its **descendants** with that prefix.

- You can’t use the prefix outside **someElement**.

- Declare it in the root to be able to use it anywhere.

Using Prefixes

- You may declare as many prefixed namespaces as needed.

- The prefix may not begin with any case combo of “xml”.

- An element’s **prefix** + its element **name**...
- ...is called the element’s **expanded name**.
  - A.k.a. universal name, qualified name, or QName.
  - e.g. **xs + element= xs:element**

Namespaces and Attributes

- Attribute names in a single element must be **unique**.
- Each attribute in a doc is **associated** with its element.
- I.e. unique elements with attributes of the same name...
- ...won’t cause **ambiguity** in the doc.
- Thus, no need to put attributes into **namespaces**.

Namespaces and Attributes

- **However**, you must use namespaces to use...
- ...two attributes with the same name...
- ...but from two different languages...
- ...in the **same** element (to make their names unique).

```
<image jpg:src="bah.jpg" png:src="humbug.png"/>
```

Creating a Namespace

Populating a Namespace

- When creating an XML Schema and you **specify**...
- ...which elements belong to a **certain namespace**...
- ...it is known as **populating a namespace**.

- To start, first define the elements as usual in the schema.

- Then, specify a **target namespace** for the schema.

Populating a Namespace

- Adding a **target namespace** means that...
- ...**all globally-defined** elements in this schema...
- ...belong to that namespace.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.domain.com/ns/1.0">
           ...
</xs:schema>
```

Populating a Namespace

- By default, **locally-defined** elements...
- ...are **not associated** with the target namespace.

- To override this (i.e. add locally-defined elements too)...
- ...add **elementFormDefault="qualified"** to **xs:schema**.
  - attributeFormDefault is the attribute counterpart

- By default this is set to "unqualified"

Populating a Namespace

- To associate only a specific local element (not all)...
- ...to the target namespace...
- ...add **form="qualified"** to its **xs:element**.

- Use **form="unqualified"** to disassociate an element...
- ...if **elementFormDefault="qualified"** is in **xs:schema**.

Validating Namespaces

Validating with Namespaces

- To validate an XML doc that uses multiple namespaces...
- ...you must create a **separate schema** per namespace...
- ...and if there are tags without a namespace...
- ...you need to create a schema for those tags, too.

- Each Namespace is an additional XSD document you are adding to you XML

Validating with Namespaces

- **Example 1**
  - Your XML use two namespaces for specifying
    ns1:d_element
    ns2: d_element
  - ...and the root and other elements has **no namespace**.

  - To validate this XML, you need:
    - A schema for no-namespace elements (**main.xsd**)...
    - ...a second schema for the first namespace (**ns1**)...
    - ...and a third schema for the second namespace (**ns2**).

Validating with Namespaces

- In the **main.xsd**

- We **import** the schemas for each namespace.

- We only need to define **root**.
  - Inside, we **reference** each **d_element** child
  - ...using their **expanded names**.

```
<xs:element ref="ns1:d_element">
```

Validating with Namespaces

- Because we’re referring to the **ns1** and **ns2** prefixes...
- ...we must declare their namespaces in **xs:schema** root.
  - I.e. **xmlns:ns1** and **xmlns:ns2**

- The references in the element definition...
- ...refer to elements defined in the imported schemas.

Validating with Namespaces

- Then in **ns1.xsd** and **ns2.xsd**.
- We give each schema the appropriate **target namespace**.
- We define **d_element** in each schema.
  - Here we use their element names, not expanded names.

```
<xs:element name="d_element" type="xs:string">
```

- Now the XML doc will validate successfully!

Expanding Namespaces

XML Schemas in Multiple Files

- It’s possible to split large schema docs...
- ...with the **same target namespace**...
- ...into separate schema files.

- Add a global tag to the “root” doc for each “sub” doc:

```
<xs:include schemaLocation="subSchema1.xsd"/>
```

XML Schemas in Multiple Files

- In the “root” schema, if you reference an element...
- ...defined in a “sub” schema...
- ...you must use that element’s **expanded name**.

- As always, if using a particular namespace prefix in a doc...
- ...you must declare its namespace with **xmlns**.

XML Schemas in Multiple Files

- The “root” and “sub” docs are complete schema docs...
- ...but each one only defines some of the XML elements.

- **Take care:**
  - **xs:include** is for schemas with the same target namespace.
  - **xs:import** is for schemas with different target namespaces.

Namespace usage

three reasons for using namespaces

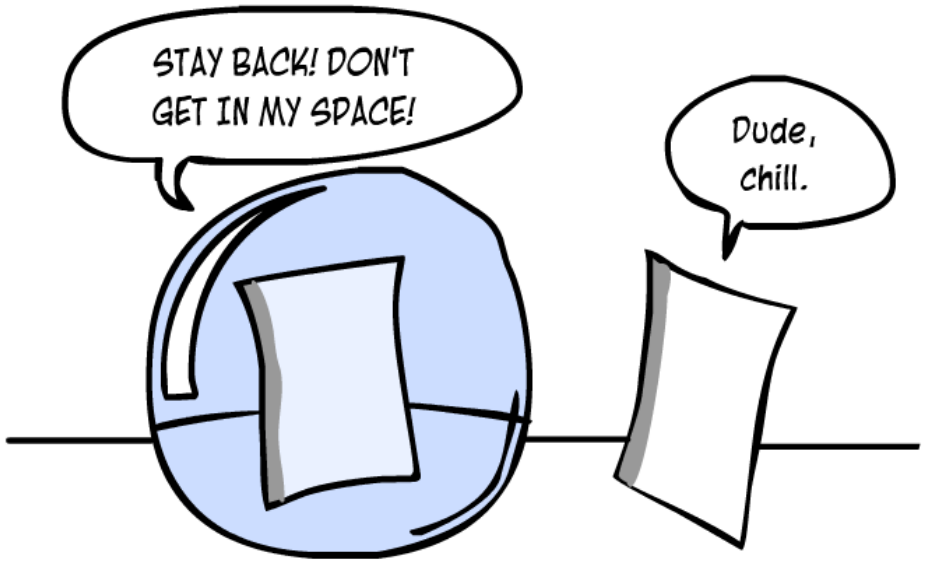
- XML Schemas
  - Defining the structure of a document.

- Combination documents
  - Merging documents from more than one source.

- Versioning
  - Differentiating between different versions of an XML format.

- **Reminder**: DTD cannot validate XML docs that use namespaces.

Theme 3: Namespaces



END OF THEME 3