XML Schema Description Language (XSD or XSDL)

# INTRODUCTION TO XML Schema (XSD)

# Review: Formal Grammars

- When we describe languages we use a grammar
- Typically use a Backus-Naur Form (BNF) Grammar
- Ex: US Postal Address

```
John Doe IV
8840 Stanford Blvd Ste 200
Columbia MD 12345
```

postal-address ::= name-part street-address zip-part
name-part ::= personal-part *last-name* opt-suffix-part *EOL*
personal-part ::= *first-name* | *initial* "."
street-address ::= *house-num street-name* opt-apt-num *EOL*
zip-part ::= *town-name* "," *state-code ZIP-code EOL*
opt-suffix-part ::= "Sr." | "Jr." | roman-numeral | ""
opt-apt-num ::= *apt-num* | ""

## XML Schema is a Formal Grammar

- Grammar of an XML document
- In a very verbose notation
- Assumes XML document is well-formed

postal-address ::= name-part street-address zip-part

```
<element name="postal-address">
  <complexType>
    <sequence>
      <group ref="name-part"/>
      <element
          name="street-address"
          type="street-address-type"/>
      <group ref="zip-part"/>
    </sequence>
  </complexType>
</element>
```

```
<postal-address>
  <first-name>John</first-name>
  <last-name>Doe</last-name>
  <name-suffix>IV</name-suffix>
  <street-address>
    <street>
      8840 Stanford Blvd
    </street>
    <apt-num>Ste 200<apt-num>
    <city>Columbia</city>
    <state>MD</state>
  </street-address>
  <zipcode>12345</zipcode>
</postal-address>
```

## XML Schema as a Formal Grammar

personal-part ::= *first-name* | *initial* "."

```
<group name="personal-part">
  <choice>
    <element name="first-name" type="xs:string"/>

    <element name="initial">
      <simpleType>
        <restriction base="xs:string">
          <pattern value="[A-Z]\."/>
        </restriction>
      </simpleType>
    </element>

  </choice>
</group>
```

## XML Schema Defining Forms

- An XML Schema is a collection of *Defining Forms*

XSD Terminology:
Elements have *declarations*.
Types and Groups have *definitions*.

- Element
  - always named, can be *nillable*
- SimpleType - int, boolean, string, float, date, time, etc.
  - named or anonymous (inline)
- ComplexType - contains child elements
  - named or anonymous (inline)
- Group
  - named for reuse or anonymous (inline)
  - Sequence
  - Choice

## XML Schema (XSD) is Verbose

Compare this BNF:

personal-part ::= *first-name* | *initial* "."

To this XSD:

```
<group name="personal-part">
  <choice>

    <element name="first-name" type="xs:string"/>

    <element name="initial">
      <simpleType>
        <restriction base="xs:string">
          <pattern value="[A-Z]\."/>
        </restriction>
      </simpleType>
    </element>

  </choice>
</group>
```

# XSD is Verbose for One Very Good Reason

- *Standardized Annotation Syntax*
  - non-native attributes
  - appinfo annotations

- Every part of the XML Schema has these. Consider:

```
<group name="personal-part">
  <choice dfdl:choiceLengthKind='implicit'>
    <annotation>
      <appinfo source="http://www.ogf.org/dfdl/">
        <dfdl:choice choiceDispatchKey='{
              ....
        }'/>
      </appinfo>
    </annotation>
    ...
```

- BNF provides no place to hang annotations. It is too dense. No flexibility.

personal-part ::= *first-name* | *initial* "."