# Part 2: Legal Building Blocks of XML

- A Document Type Definition (**DTD**) allows the developer to create a set of rules to specify legal content and place restrictions on an XML file
- If the XML document does not follow the rules contained within the DTD, a parser generates an error
- An XML document that conforms to the rules within a DTD is said to be **valid.**
- A DTD defines the structure and the legal elements and attributes of an XML document.

# Why Use a DTD?

- A single DTD ensures a common format for each XML document that references it
- An application can use a standard DTD to verify that data that it receives from the outside world is valid
- A description of legal, valid data further contributes to the interoperability and efficiency of using XML

# An Internal DTD Declaration

If the DTD is declared inside the XML file, it must be wrapped inside the <!DOCTYPE> definition:

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend</body>
</note>
```

- **!DOCTYPE note** defines that the root element of this document is note
- **!ELEMENT note** defines that the note element must contain four elements: "to,from,heading,body"
- **!ELEMENT to** defines the to element to be of type "#PCDATA"
- **!ELEMENT from** defines the from element to be of type "#PCDATA"
- **!ELEMENT heading** defines the heading element to be of type "#PCDATA"
- **!ELEMENT body** defines the body element to be of type "#PCDATA"

# An External DTD Declaration

- If the DTD is declared in an external file, the <!DOCTYPE> definition must contain a reference to the DTD file:
- <?xml version="1.0"?>
  <!DOCTYPE note SYSTEM "note.dtd">
  <note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
  </note>

And here is the file "note.dtd", which contains the DTD:

- <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to (#PCDATA)>
  <!ELEMENT from (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body (#PCDATA)>

# DTD - Elements

- **Declaring Elements**
- In a DTD, XML elements are declared with the following syntax:
- <!ELEMENT element-name category>
  or
  <!ELEMENT element-name (element-content)>

- a

**Empty Elements**

- Empty elements are declared with the category keyword EMPTY:
- <!ELEMENT element-name EMPTY>

  Example:

  <!ELEMENT br EMPTY>

  XML example:

  <br />

**Elements with Parsed Character Data**

- Elements with only parsed character data are declared with #PCDATA inside parentheses:
- <!ELEMENT element-name (#PCDATA)>

  Example:

  <!ELEMENT from (#PCDATA)>

**Elements with any Contents**

- Elements declared with the category keyword ANY, can contain any combination of parsable data:
- <!ELEMENT element-name ANY>

  Example:

  <!ELEMENT note ANY>

**Elements with Children (sequences)**

- Elements with one or more children are declared with the name of the children elements inside parentheses:
- <!ELEMENT element-name (child1)>
  or
  <!ELEMENT element-name (child1,child2,...)>

  Example:

  <!ELEMENT note (to,from,heading,body)>

When children are declared in a sequence separated by commas, the children must appear in the same sequence in the document

**Declaring Only One Occurrence of an Element**

- <!ELEMENT element-name (child-name)>

  Example:
  <!ELEMENT note (message)>
- The example above declares that the child element "message" must occur once, and only once inside the "note" element.

**Declaring Minimum One Occurrence of an Element**

- <!ELEMENT element-name (child-name+)>

  Example:
  <!ELEMENT note (message+)>
- The + sign in the example above declares that the child element "message" must occur one or more times inside the "note" element.

**Declaring Zero or More Occurrences of an Element**

- <!ELEMENT element-name (child-name*)>

  Example:
  <!ELEMENT note (message*)>
- The * sign in the example above declares that the child element "message" can occur zero or more times inside the "note" element.

**Declaring Zero or One Occurrences of an Element**

- <!ELEMENT element-name (child-name?)>

  Example:
  <!ELEMENT note (message?)>
- The ? sign in the example above declares that the child element "message" can occur zero or one time inside the "note" element.

**Declaring either/or Content**

- <!ELEMENT note (to,from,header,(message|body))>
- The example above declares that the "note" element must contain a "to" element, a "from" element, a "header" element, and either a "message" or a "body" element.

**Declaring Mixed Content**

- <!ELEMENT note (#PCDATA|to|from|header|message)*>
- The example above declares that the "note" element can contain zero or more occurrences of parsed character data, "to", "from", "header", or "message" elements.

# DTD - Attributes

In a DTD, attributes are declared with an ATTLIST declaration.

**Declaring Attributes**

An attribute declaration has the following syntax:

<!ATTLIST element-name attribute-name attribute-type attribute-value>

DTD example:

<!ATTLIST payment type CDATA "check">

XML example:

<payment type="check" />

# The **attribute-type** can be one of the following:

| Type | Description |
|---|---|
| CDATA | The value is character data |
| (*en1*\|*en2*\|..) | The value must be one from an enumerated list |
| ID | The value is a unique id |
| IDREF | The value is the id of another element |
| IDREFS | The value is a list of other ids |
| NMTOKEN | The value is a valid XML name |
| NMTOKENS | The value is a list of valid XML names |
| ENTITY | The value is an entity |
| ENTITIES | The value is a list of entities |
| NOTATION | The value is a name of a notation |
| xml: | The value is a predefined xml value |

# The **attribute-value** can be one of the following:

| Value | Explanation |
| --- | --- |
| *value* | The default value of the attribute |
| #REQUIRED | The attribute is required |
| #IMPLIED | The attribute is optional |
| #FIXED *value* | The attribute value is fixed |

## A Default Attribute Value

- DTD:
  <!ELEMENT square EMPTY>
  <!ATTLIST square width CDATA "0">

  Valid XML:
  <square width="100" />
- In the example above, the "square" element is defined to be an empty element with a "width" attribute of type CDATA. If no width is specified, it has a default value of 0.

# #REQUIRED

**Syntax**

- <!ATTLIST element-name attribute-name attribute-type #REQUIRED>

**Example**

- DTD:
  <!ATTLIST person number CDATA #REQUIRED>

  Valid XML:
  <person number="5677" />

  Invalid XML:
  <person />
- Use the #REQUIRED keyword if you don't have an option for a default value, but still want to force the attribute to be present.

# #IMPLIED

**Syntax**

- <!ATTLIST element-name attribute-name attribute-type #IMPLIED>

**Example**

- DTD:
  <!ATTLIST contact fax CDATA #IMPLIED>

  Valid XML:
  <contact fax="555-667788" />

  Valid XML:
  <contact />
- Use the #IMPLIED keyword if you don't want to force the author to include an attribute, and you don't have an option for a default value.

# #FIXED

**Syntax**

- <!ATTLIST element-name attribute-name attribute-type #FIXED "value">

**Example**

- DTD:
  <!ATTLIST sender company CDATA #FIXED "Microsoft">

  Valid XML:
  <sender company="Microsoft" />

  Invalid XML:
  <sender company="W3Schools" />
- Use the #FIXED keyword when you want an attribute to have a fixed value without allowing the author to change it. If an author includes another value, the XML parser will return an error.

# Enumerated Attribute Values

**Syntax**

- <!ATTLIST element-name attribute-name (en1|en2|..) default-value>

**Example**

DTD:
<!ATTLIST payment type (check|cash) "cash">

XML example:
<payment type="check" />
or
<payment type="cash" />

- Use enumerated attribute values when you want the attribute value to be one of a fixed set of legal values.

# DTD - Entities

- Entities are used to define shortcuts to special characters.
- Entities can be declared internal or external.
- **An Internal Entity Declaration**
- **Syntax**
- <!ENTITY entity-name "entity-value">
- **Example**
- DTD Example:

  <!ENTITY writer "Donald Duck.">
  <!ENTITY copyright "Copyright W3Schools.">

  XML example:

  <author>&writer;&copyright;</author>
- **Note:** An entity has three parts: an ampersand (&), an entity name, and a semicolon (;).

## An External Entity Declaration

- **Syntax**
- <!ENTITY entity-name SYSTEM "URI/URL">
- **Example**
- DTD Example:

  <!ENTITY writer SYSTEM
  "http://www.w3schools.com/entities.dtd">
  <!ENTITY copyright SYSTEM
  "http://www.w3schools.com/entities.dtd">

  XML example:

  <author>&writer;&copyright;</author>

# Some Example DTD Declarations

- Example 1: The Empty Element

```
<!ELEMENT Bool (EMPTY)> <!--DTD declaration of empty
   element-->
<Bool Value="True"></Bool> <!--Usage with attribute
   in XML file-->
```

- Example 2: Elements with Data

```
<!ELEMENT Month (#PCDATA)> <!--DTD declaration of an element->
<Month>April</Month> <!—Valid usage within XML file-->
<Month>This is a month</Month> <!—Valid usage within XML file-->
<Month> <!—Invalid usage within XML file, can't have
   children!-->
<January>Jan</January>
<March>March</March>
</Month>
```

# Some Example DTD Declarations

Example 3: Elements with Children

To specify that an element must have a single child element, include the element name within the parenthesis.

```
<!ELEMENT House (Address)> <!—A house has a single address-->
<House> <!—Valid usage within XML file-->
<Address>1345 Preston Ave Charlottesville Va 22903</Address>
</House>
```

An element can have multiple children. A DTD describes multiple children using a *sequence,* or a list of elements separated by commas. The XML file must contain one of each element in the specified order.

```
<!--DTD declaration of an element-->
  <!ELEMENT address (person,street,city, zip)>
  <!ELEMENT person  (#PCDATA)>
  <!ELEMENT street  (#PCDATA)>
  <!ELEMENT city    (#PCDATA)>
  <!ELEMENT zip     (#PCDATA)>
  <!—Valid usage within XML file-->
  <address>
  <person>John Doe</person>
  <street>1234 Preston Ave.</street>
  <city>Charlottesville, Va</city>
  <zip>22903</zip>
  </address>
```

# Cautions concerning DTDs

- All element declarations begin with <!ELEMENT and end with >
- The ELEMENT declaration is *case sensitive*
- The programmer must declare all elements within an XML file
- Elements declared with the #PCDATA content model can not have children
- When describing sequences, the XML document must contain exactly those elements in exactly that order.

# Walking Through an Example

- Using the file “music.xml” contained in the extras folder, create a Document Type Definition that describes all of the elements. The goals for this exercise are to:
  - map out the elements
  - define each element
  - pick the best content model for each element
  - correctly order the elements using sequences
- Internally embed the DTD within the XML document

```
<!DOCTYPE TVSCHEDULE [

< !ELEMENT TVSCHEDULE (CHANNEL+)>
< !ELEMENT CHANNEL (BANNER,DAY+)>
< !ELEMENT BANNER (#PCDATA)>
< !ELEMENT DAY (DATE,(HOLIDAY|PROGRAMSLOT+)+)>
< !ELEMENT HOLIDAY (#PCDATA)>
< !ELEMENT DATE (#PCDATA)>
< !ELEMENT PROGRAMSLOT (TIME,TITLE,DESCRIPTION?)>
< !ELEMENT TIME (#PCDATA)>
< !ELEMENT TITLE (#PCDATA)>
< !ELEMENT DESCRIPTION (#PCDATA)>

< !ATTLIST TVSCHEDULE NAME CDATA #REQUIRED>
< !ATTLIST CHANNEL CHAN CDATA #REQUIRED>
< !ATTLIST PROGRAMSLOT VTR CDATA #IMPLIED>
< !ATTLIST TITLE RATING CDATA #IMPLIED>
< !ATTLIST TITLE LANGUAGE CDATA #IMPLIED>
]>
```

```
<!DOCTYPE CATALOG [
< !ENTITY AUTHOR "John Doe">
< !ENTITY COMPANY "JD Power Tools, Inc.">
< !ENTITY EMAIL "jd@jd-tools.com">

< !ELEMENT CATALOG (PRODUCT+)>

< !ELEMENT PRODUCT
(SPECIFICATIONS+,OPTIONS?,PRICE+,NOTES?)>
< !ATTLIST PRODUCT
NAME CDATA #IMPLIED
CATEGORY (HandTool|Table|Shop-Professional)
"HandTool"
PARTNUM CDATA #IMPLIED
PLANT (Pittsburgh|Milwaukee|Chicago) "Chicago"
INVENTORY (InStock|Backordered|Discontinued)
"InStock">

< !ELEMENT SPECIFICATIONS (#PCDATA)>
< !ATTLIST SPECIFICATIONS
WEIGHT CDATA #IMPLIED
POWER CDATA #IMPLIED>
```

```
< !ELEMENT OPTIONS (#PCDAT
< !ATTLIST OPTIONS
FINISH (Metal|Polished|Matte
ADAPTER
(Included|Optional|NotApplic
"Included"
CASE (HardShell|Soft|NotApp
"HardShell">

< !ELEMENT PRICE (#PCDATA)>
< !ATTLIST PRICE
MSRP CDATA #IMPLIED
WHOLESALE CDATA #IMPLIED
STREET CDATA #IMPLIED
SHIPPING CDATA #IMPLIED>

< !ELEMENT NOTES (#PCDATA)

]>
```

# Questions