UNIT-III

XML
(Extensible Markup Language)

XML (Extensible Markup Language)

- It is a text-based markup language derived from Standard Generalized Markup Language (SGML).
- XML tags identify the data and are used to store and organize the data, rather than specifying how to display it like HTML tags, which are used to display the data. XML is not going to replace HTML in the near future, but it introduces new possibilities by adopting many successful features of HTML.

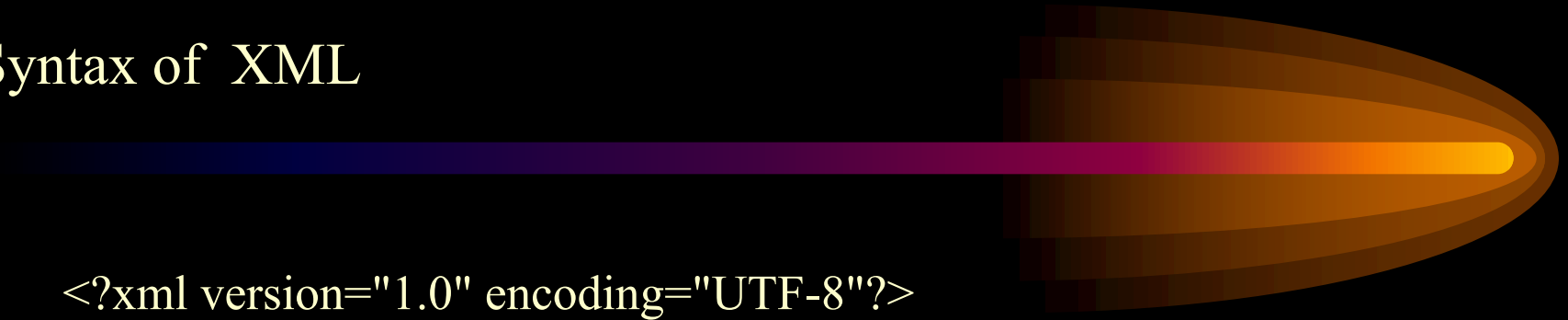
There are three important characteristics of XML

- **XML is extensible** − XML allows you to create your own self-descriptive tags, or language, that suits your application.
- **XML carries the data, does not present it** − XML allows you to store the data irrespective of how it will be presented.
- **XML is a public standard** − XML was developed by an organization called the World Wide Web Consortium (W3C) and is available as an open standard.

Some Points of XML

- **Xml** (eXtensible Markup Language) is a mark up language.
- XML is designed to store and transport data.
- Xml is created to provide an easy to use and store self describing data.
- XML became a W3C Recommendation on February 10, 1998.
- XML is not a replacement for HTML.
- XML is designed to be self-descriptive.
- XML is designed to carry data, not to display data.
- XML tags are not predefined. You must define your own tags.
- XML is platform independent and language independent.

Syntax of XML



```
<?xml version="1.0" encoding="UTF-8"?>
```

Where,

- *version* is the XML version and
- *encoding* specifies the character encoding used in the document.

Syntax of XML

<?xml version = "1.0" encoding="UTF-8"? >

<contact-info>

<name>Raaj</name>

<company>IBM</company>

<phone>01177777777</phone>

</contact-info>

There are two kinds of information in the above example

- Markup, like <contact-info>
- The text, or the character data: *Raaj, IBM* and *77777777*

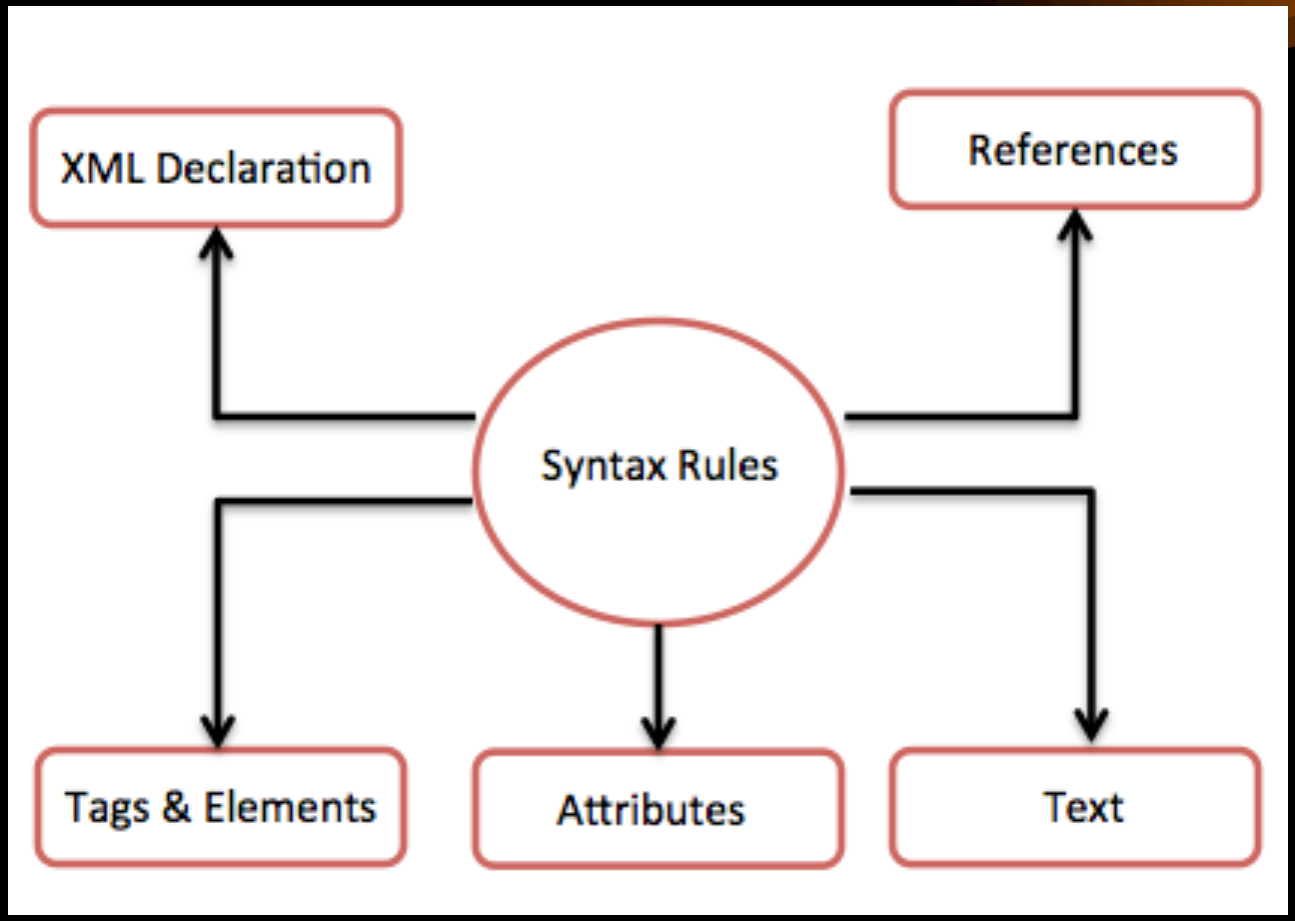
Advantages of XML

- It made it easy to transport and share data.
- XML improves the exchange of data between various platforms.
- It is a markup language, which is a set of characters or/and symbols placed in a text document.
- XML indicates how the XML document should look after it is displayed.
- It simplifies the platform change process.
- It enhances data availability.
- It supports multilingual documents and Unicode.
- Provide relatively easy to learn and code.
- It is a markup language, which is a set of characters or/and symbols placed in a text document.
- It performs validation using DTD and Schema.
- Makes documents transportable across systems and applications. With the help of XML, you can exchange data quickly between different platforms.
- XML separates the data from HTML.

Disadvantages of XML

- XML requires a processing application.
- The XML syntax is similar to another alternative ‘text-based’ data transmission formats, which is sometimes confusing.
- No intrinsic data type support
- The XML syntax is redundant.
- Does not allow the user to create his tags.

Syntax rules to write different types of markup and text in an XML document.



XML Declaration

- The XML document can optionally have an XML declaration. It is written as follows

```
<?xml version="1.0" encoding="UTF-8"?>
```

Where,

- *version* is the XML version and
- *encoding* specifies the character encoding used in the document.

Syntax Rules for XML Declaration

- The XML declaration is case sensitive and must begin with "**<?xml>**" where "**xml**" is written in lower-case.
- If document contains XML declaration, then it strictly needs to be the first statement of the XML document.
- The XML declaration strictly needs be the first statement in the XML document.
- An HTTP protocol can override the value of *encoding* that you put in the XML declaration.

Tags and Elements

- An XML file is structured by several XML-elements, also called XML-nodes or XML-tags. The names of XML-elements are enclosed in triangular brackets < > as shown below :

<element>

Syntax Rules for Tags and Elements

- **Element Syntax** − Each XML-element needs to be closed either with start or with end elements as shown below :

<element>……...</element>

OR

<element/>

Nesting of Elements

- An XML-element can contain multiple XML-elements as its children, but the children elements must not overlap. i.e., an end tag of an element must have the same name as that of the most recent unmatched start tag.

Example:

```
<?xml version = "1.0"?>
    <contact-info>
            <company>IBM</company>
    <contact-info>
```

Root Element

- An XML document can have only one root element. For example, following is not a correct XML document, because both the **x** and **y** elements occur at the top level without a root element

Example:

```
<root>
    <x>.....</x>
    <y>.....</y>
</root>
```

Case Sensitivity

- The names of XML-elements are case-sensitive. That means the name of the start and the end elements need to be exactly in the same case

Example:

<name> is different from <Name>

XML Attributes

- An **attribute** specifies a single property for the element, using a name/value pair. An XML-element can have one or more attributes.

For example:

 Medi-Caps University

Here ,

href is the attribute name and https://www.medicaps.ac.in / is attribute value.

Syntax Rules for XML Attributes

- Attribute names in XML (unlike HTML) are case sensitive. That is, *HREF* and *href* are considered two different XML attributes.
- Same attribute cannot have two values in a syntax.
- The following example shows incorrect syntax because the attribute *b* is specified twice

....

- Attribute names are defined without quotation marks, whereas attribute values must always appear in quotation marks. Following example demonstrates incorrect xml syntax

....

XML References

- References usually allow you to add or include additional text or markup in an XML document.
- References always begin with the symbol **"&"** which is a reserved character and end with the symbol **";"**. XML has two types of references
  1. **Entity References**
  2. **Character References**

Entity References −

- An entity reference contains a name between the start and the end delimiters. For example **&** where *amp* is *name*. The *name* refers to a predefined string of text and/or markup.

Character References −

- These contain references, such as **A**, contains a hash mark (“#”) followed by a number. The number always refers to the Unicode code of a character. In this case, 65 refers to alphabet "A".

XML Text

- The names of XML-elements and XML-attributes are case-sensitive, which means the name of start and end elements need to be written in the same case.
- To avoid character encoding problems, all XML files should be saved as Unicode UTF-8 or UTF-16 files.
- Whitespace characters like blanks, tabs and line-breaks between XML-elements and between the XML-attributes will be ignored.
- Some characters are reserved by the XML syntax itself. Hence, they cannot be used directly. To use them, some replacement-entities are used, which are listed below

Not Allowed Character	Replacement Entity	Character Description
<	<	less than
>	>	greater than
&	&	ampersand
'	'	apostrophe
"	"	quotation mark

XML - Declaration

- **XML declaration** contains details that prepare an XML processor to parse the XML document.
- It is optional, but when used, it must appear in the first line of the XML document

Syntax

```
<?xml version = "version_number"
encoding = "encoding_declaration"
standalone = "standalone_status"
?>
```

- Each parameter consists of a parameter name, an equals sign (=), and parameter value inside a quote

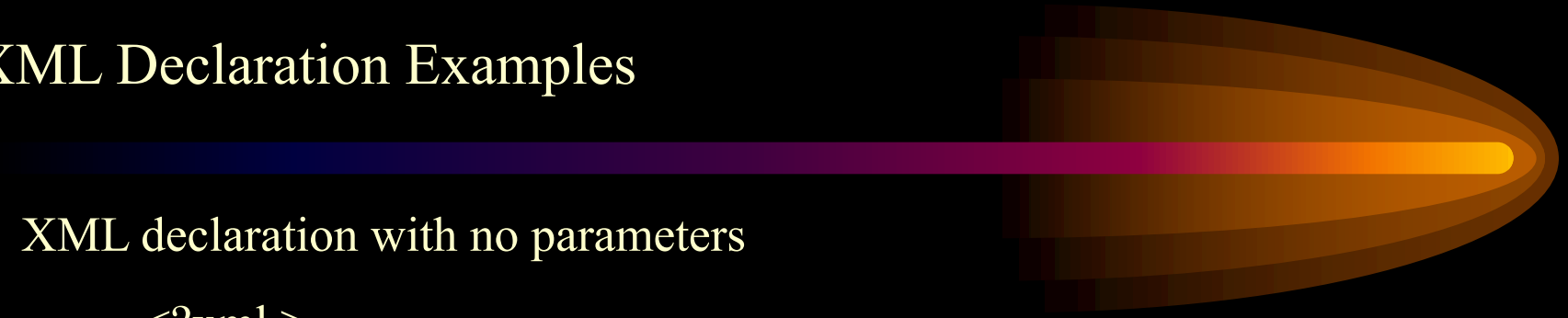
Following table shows the above syntax in detail

Parameter	Parameter_value	Parameter_description
Version	1.0	Specifies the version of the XML standard used.
Encoding	UTF-8, UTF-16, ISO-10646-UCS-2, ISO-10646-UCS-4, ISO-8859-1 to ISO-8859-9, ISO-2022-JP, Shift_JIS, EUC-JP	It defines the character encoding used in the document. UTF-8 is the default encoding used.
Standalone	yes or no	It informs the parser whether the document relies on the information from an external source, such as external document type definition (DTD), for its content. The default value is set to *no*. Setting it to *yes* tells the processor there are no external declarations required for parsing the document.

Rules of XML declaration

- If the XML declaration is present in the XML, it must be placed as the first line in the XML document.
- If the XML declaration is included, it must contain version number attribute.
- The Parameter names and values are case-sensitive.
- The names are always in lower case.
- The order of placing the parameters is important. The correct order is: *version, encoding and standalone.*
- Either single or double quotes may be used.
- The XML declaration has no closing tag i.e. **</?xml>**

XML Declaration Examples



- XML declaration with no parameters

  <?xml >

- XML declaration with version definition

  <?xml version = "1.0">

- XML declaration with all parameters defined

  <?xml version = "1.0" encoding = "UTF-8" standalone = "no" ?>

- XML declaration with all parameters defined in single quotes

  <?xml version = '1.0' encoding = 'iso-8859-1' standalone = 'no' ?>

XML - Tags

- **XML tags** form the foundation of XML. They define the scope of an element in XML. They can also be used to insert comments, declare settings required for parsing the environment, and to insert special instructions.

Start Tag

- The beginning of every non-empty XML element is marked by a start-tag. Following is an example of start-tag −

<address>

Empty Tag

- The text that appears between start-tag and end-tag is called content. An element which has no content is termed as empty. An empty element can be represented in two ways as follows −
- A start-tag immediately followed by an end-tag as shown below −

<hr></hr>

- A complete empty-element tag is as shown below

<hr />

Differences between HTML and XML

HTML	XML
HTML is used **to display data** and focuses on how data looks.	XML is a software and hardware independent tool used **to transport and store data**. It focuses on what data is.
HTML is a **markup language** itself.	XML provides a **framework to define markup languages**.
HTML is **not case sensitive**.	XML is **case sensitive**.
HTML is a presentation language	XML is neither a presentation language nor a programming language.

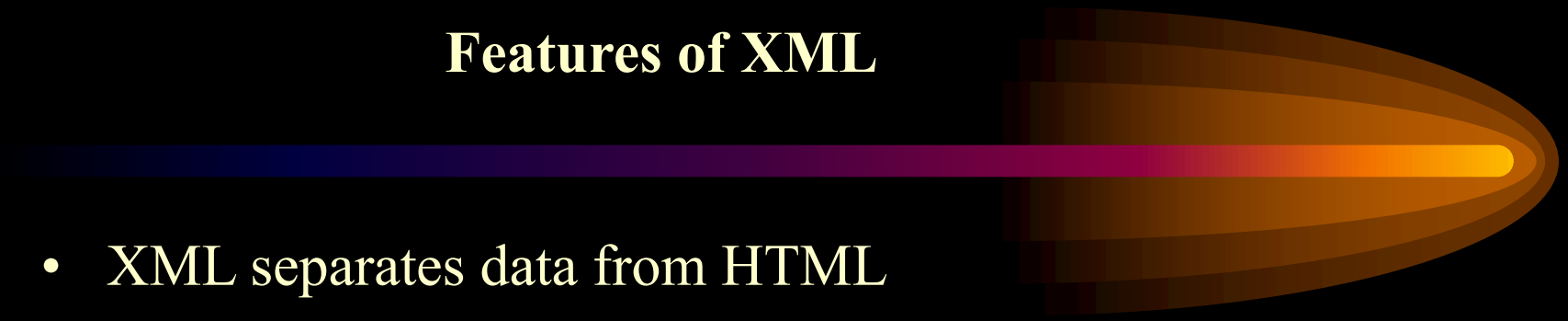
Cont…

HTML	XML
HTML **has its own predefined tags**.	You **can define tags according to your need**.
In HTML, it is **not necessary to use a closing tag**.	XML **makes it mandatory to use a closing tag**.
HTML is **static** because it is used to display data.	XML is **dynamic** because it is used to transport data.
HTML **does not preserve whitespaces**.	XML **preserve whitespaces**.

Similarities between HTML and XML

- Both are open formats.
- Both are markup languages.
- Both use tags and attributes to describe the content.

Features of XML



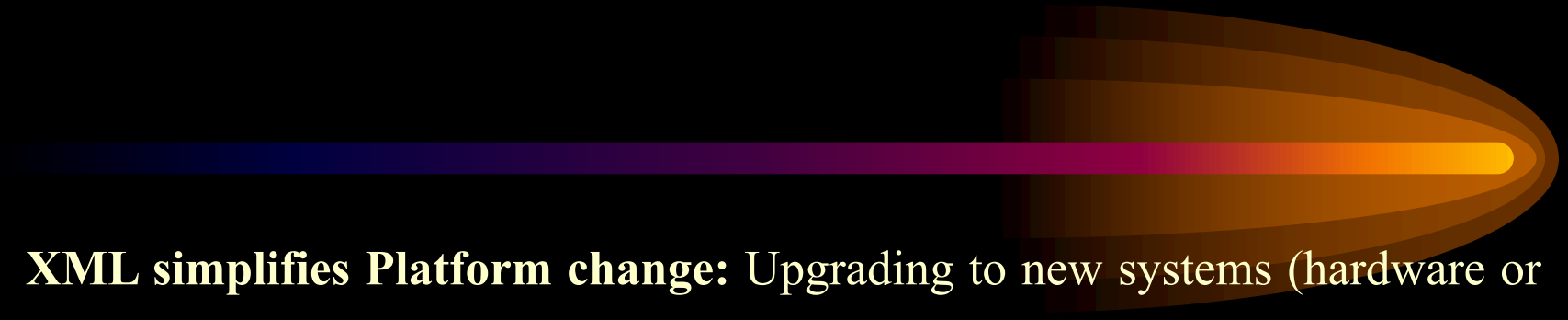
- XML separates data from HTML
- XML simplifies data sharing
- XML simplifies data transport
- XML simplifies Platform change
- XML increases data availability
- XML can be used to create new internet languages

- **XML separates data from HTML:** If you need to display dynamic data in your HTML document, it will take a lot of work to edit the HTML each time the data changes. With XML, data can be stored in separate XML files.
- This way you can focus on using HTML/CSS for display and layout, and be sure that changes in the underlying data will not require any changes to the HTML. With a few lines of JavaScript code, you can read an external XML file and update the data content of your web page

- **XML simplifies data sharing:** In the real world, computer systems and databases contain data in incompatible formats. XML data is stored in plain text format.
- This provides a software- and hardware-independent way of storing data. This makes it much easier to create data that can be shared by different applications.

- **XML simplifies data transport:** One of the most time-consuming challenges for developers is to exchange data between incompatible systems over the Internet.

- Exchanging data as XML greatly reduces this complexity, since the data can be read by different incompatible applications



- **XML simplifies Platform change:** Upgrading to new systems (hardware or software platforms), is always time consuming. Large amounts of data must be converted and incompatible data is often lost.

- XML data is stored in text format. This makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data.

XML increases data availability:

- Different applications can access your data, not only in HTML pages, but also from XML data sources. With XML, your data can be available to all kinds of "reading machines" (Handheld computers, voice machines, news feeds, etc), and make it more available for blind people, or people with other disabilities.

XML can be used to create new internet languages:

- A lot of new Internet languages are created with XML

Here are some examples:

- XHTML
- WSDL for describing available web services
- WAP and WML as markup languages for handheld devices
- RSS languages for news feeds
- RDF and OWL for describing resources and ontology
- SMIL for describing multimedia for the web

Usages of XML

- XML can work behind the scene to simplify the creation of HTML documents for large web sites.
- XML can be used to exchange the information between organizations and systems.
- XML can be used for offloading and reloading of databases.
- XML can be used to store and arrange the data, which can customize your data handling needs.
- XML can easily be merged with style sheets to create almost any desired output.
- Virtually, any type of data can be expressed as an XML document.

Simple Example of XML

```
<?xml version = "1.0" encoding = "UTF-8" ?>
 <book>
<name>Web Technology</name>
 <author>Gopalan N. P.</author>
 <publisher>PHI Learning</publisher>
 </book>
```

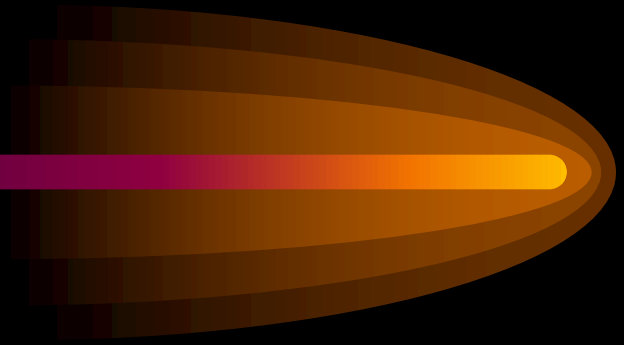
XML Comments

- XML comments are just like HTML comments. We know that the comments are used to make codes more understandable other developers.

Syntax

```
<!-- Write your comment-->
```

Example Of XML Comments

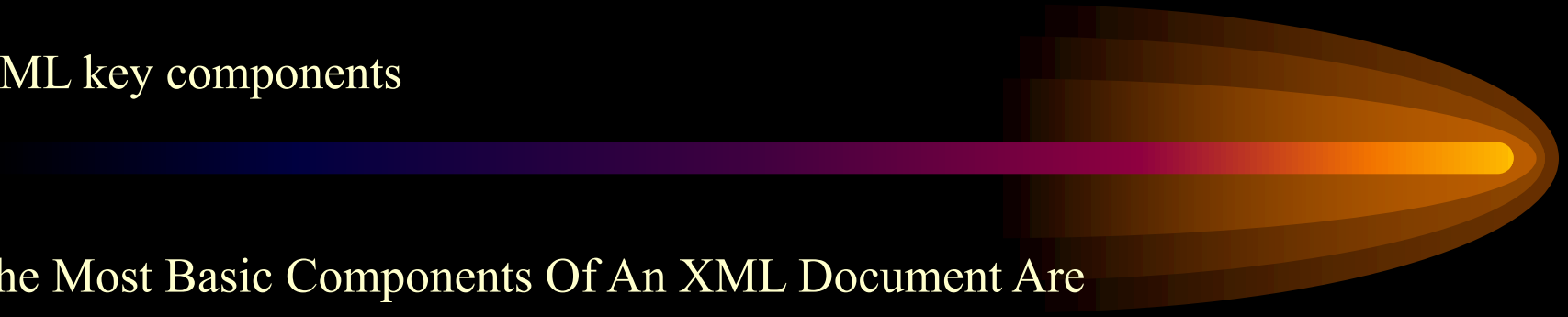


```
<?xml version="1.0" encoding="UTF-8" ?>
<!--Students marks are uploaded by months-->
<students>
  <student>
    <name>Vikash</name>
    <marks>90</marks>
  </student>
  <student>
    <name>Amit</name>
    <marks>80</marks>
  </student>
</students>
```

Rules of XML comments

- Don't use a comment before an XML declaration.
- You can use a comment anywhere in XML document except within attribute value.
- Don't nest a comment inside the other comment.

XML key components



The Most Basic Components Of An XML Document Are

1. *Elements,*
2. *Attributes,*
3. *Comments*.

Elements

- The XML elements are the basic building block of the XML document. It is used as a container to store text elements, attributes, media objects etc.
- Every XML documents contain at least one element whose scopes are delimited by start and end tags or in case of empty elements it is delimited by an empty tag.

Syntax:

<element-name attributes> Contents </element-name>

Here, element-name: It is the name of element.

attributes: The attributes are used to define the XML element property and these attributes are separated by white space. It associates the name with a value, which is a string of characters.

Example:

name=”Medicaps"

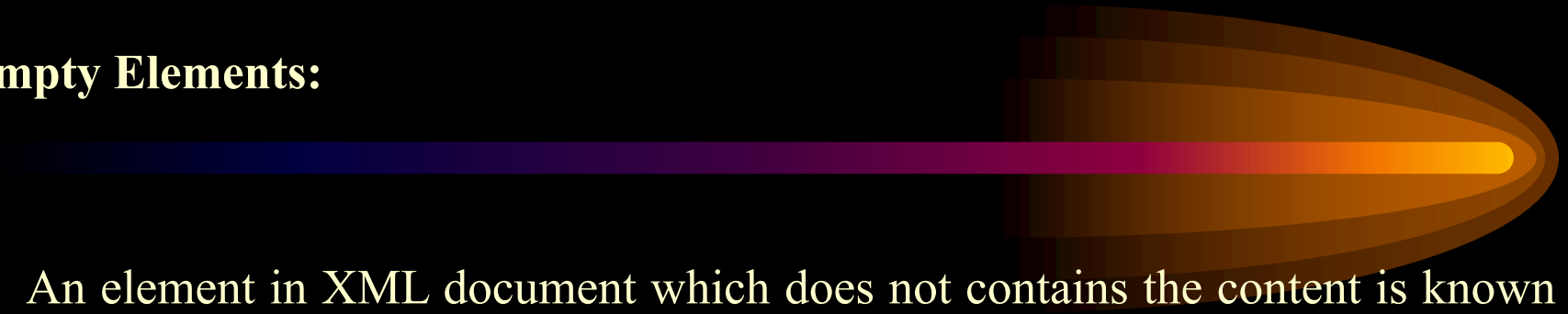
Here, Medicaps represents the value of attribute

Rules to define XML elements:

There are some rules to create XML elements which are given below:

- An element an contain alphanumeric values or characters. But only three special characters are required in the names these are hyphen, underscore and period.
- Names are case sensitive. It means lower case letters have different meaning and upper case characters have different meaning. For example address, Address are different names.
- Both start and end tags for elements need to be same.
- An element, which is a container, can contain text or elements

Empty Elements:



- An element in XML document which does not contains the content is known as Empty Element.
- The basic syntax of empty element in XML as follows:

  <elements-name attributename/>

Example

```
<?xml version = “1.0”?>
<contact-info>
        <address  = “RAU INDORE”>
                <name>VIKASH</name>
                <College>MEDICAPS</College>
                <mobile>077777777</mobile>
        </address>
</contact-info>
```

Output:

VIKASH

MEDICAPS

07777777777

Attributes

- The XML attribute is a part of an XML element. The addition of attribute in XML element gives more precise properties of the element i.e, it enhances the properties of the XML element.

Syntax:

<element_name attribute1 attribute2 ... > Contents... </element_name>

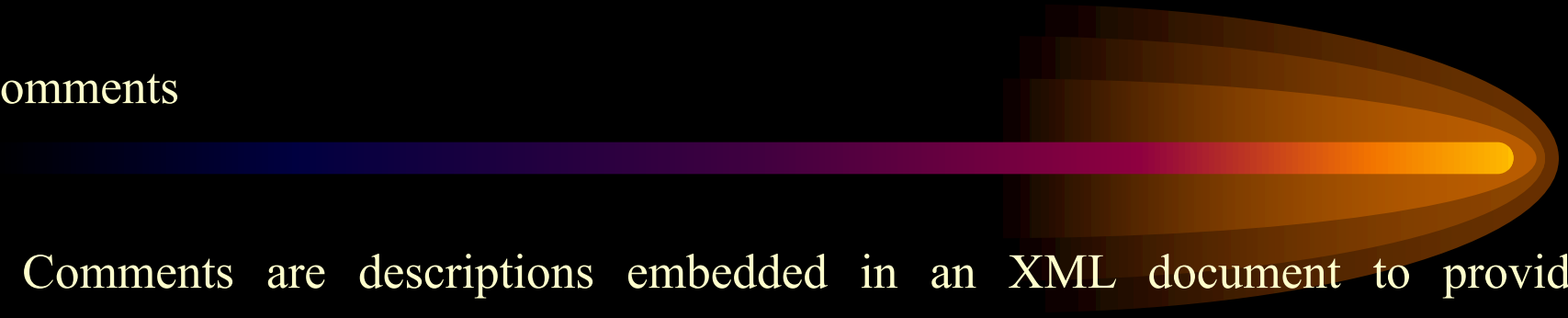
- In the above syntax element_name is the name of an element which can be any name. The attribute1, attribute2, … is XML attribute having unique attribute name. Then in the content section, any message can be written and at the end, the element name is ended.

Example

<text category = "message">Welcome to Medi-caps University</text>

- In the above example, XML element is text, the **category** is the attribute name and **message** is the attribute value, Attribute name and its value always appear in pair.
- The attribute name is used without any quotation but attribute value is used in single (‘ ‘) or double quotation (” ”).

Comments



- Comments are descriptions embedded in an XML document to provide additional information about the document.
- Comments in XML use the same syntax as HTML comments and are formatted so that they are ignored by the application processing the document, as shown here:

```
<!-- Comment text -->
```

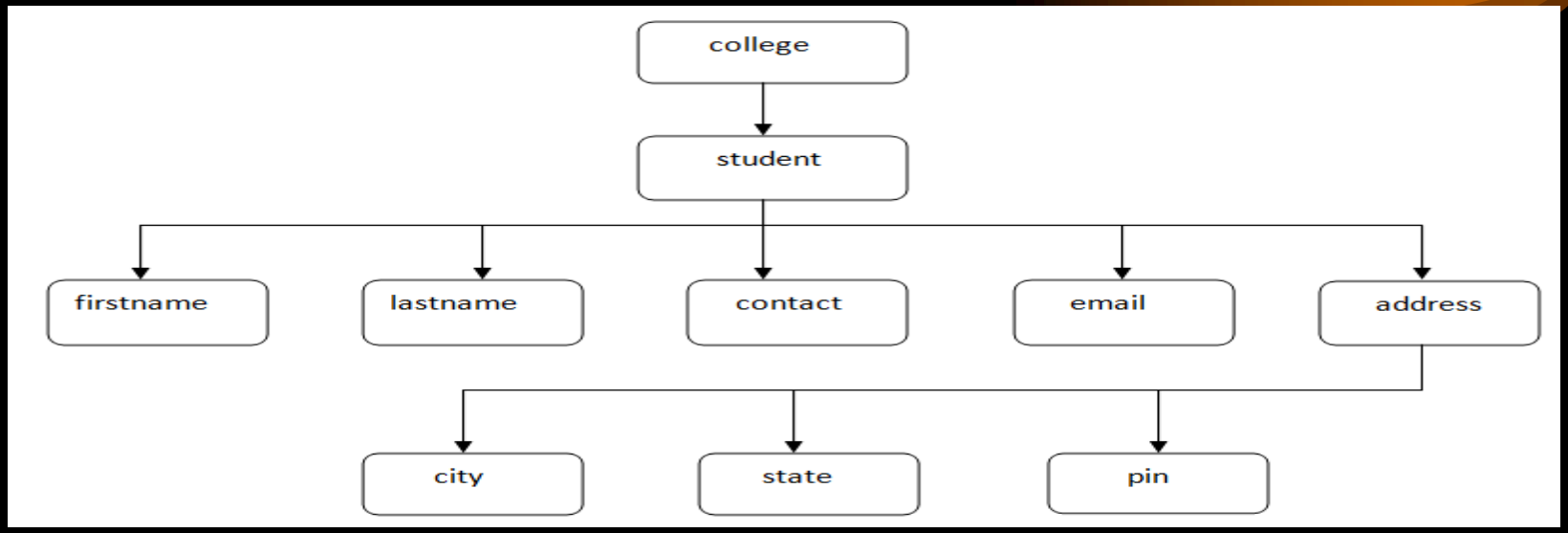
XML Tree

- An XML document has a self descriptive structure. It forms a tree structure which is referred as an XML tree. The tree structure makes easy to describe an XML document.
- A tree structure contains root element (as parent), child element and so on. It is very easy to traverse all succeeding branches and sub-branches and leaf nodes starting from the root.

Example of an XML document

```
<?xml version="1.0"?>
<college>
  <student>
     <firstname>Vikash</firstname>
     <lastname>Kumar</lastname>
     <contact>077777777</contact>
     <email>vikash@gmail.com</email>
     <address>
        <city>Indore</city>
        <state>Madhya Pradesh</state>
        <pin>453331</pin>
     </address>
  </student>
</college>
```

The tree-structure representation of the above example



- In the above example, first line is the XML declaration. It defines the XML version 1.0. Next line shows the root element (college) of the document. Inside that there is one more element (student). Student element contains five branches named <firstname>, <lastname>, <contact>, <Email> and <address>. <address> branch contains 3 sub-branches named <city>, <state> and <pin>.

Rules of XML Tree

- These rules are used to figure out the relationship of the elements. It shows if an element is a child or a parent of the other element.
- **Descendants:** If element A is contained by element B, then A is known as descendant of B. In the above example "College" is the root element and all the other elements are the descendants of "College".

- **Ancestors:** The containing element which contains other elements is called "Ancestor" of other element. In the above example Root element (College) is ancestor of all other elements.

DTD

- DTD stands for **Document Type Definition**.
- It defines the legal building blocks of an XML document.
- It is used to define document structure with a list of legal elements and attributes.

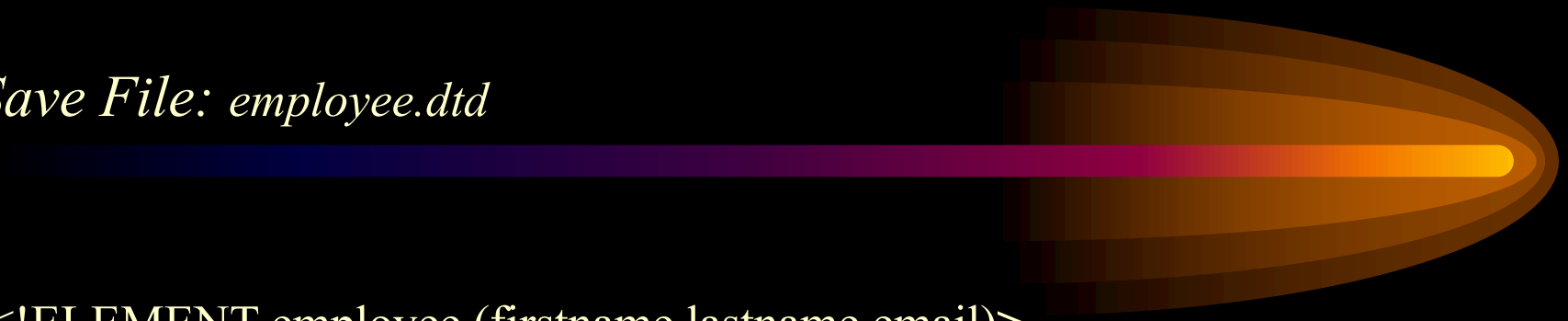
XML DTD

- A DTD defines the legal elements of an XML document
- In simple words we can say that a DTD defines the document structure with a list of legal elements and attributes.
- XML schema is a XML based alternative to DTD.
- Actually DTD and XML schema both are used to form a well formed XML document.
- We should avoid errors in XML documents because they will stop the XML programs.

Save File: employee.xml

```
<?xml version="1.0"?>
<!DOCTYPE employee SYSTEM "employee.dtd">
<employee>
  <firstname>Vikash</firstname>
  <lastname>Kumar</lastname>
  <email>Vikash@gmail.com</email>
</employee>
```

Save File: employee.dtd



```
<!ELEMENT employee (firstname,lastname,email)>

<!ELEMENT firstname (#PCDATA)>

<!ELEMENT lastname (#PCDATA)>

<!ELEMENT email (#PCDATA)>
```

- **<!DOCTYPE employee :** It defines that the root element of the document is employee.
- **<!ELEMENT employee:** It defines that the employee element contains 3 elements "firstname, lastname and email".
- **<!ELEMENT firstname:** It defines that the firstname element is #PCDATA typed. (parse-able data type).
- **<!ELEMENT lastname:** It defines that the lastname element is #PCDATA typed. (parse-able data type).
- **<!ELEMENT email:** It defines that the email element is #PCDATA typed. (parse-able data type)

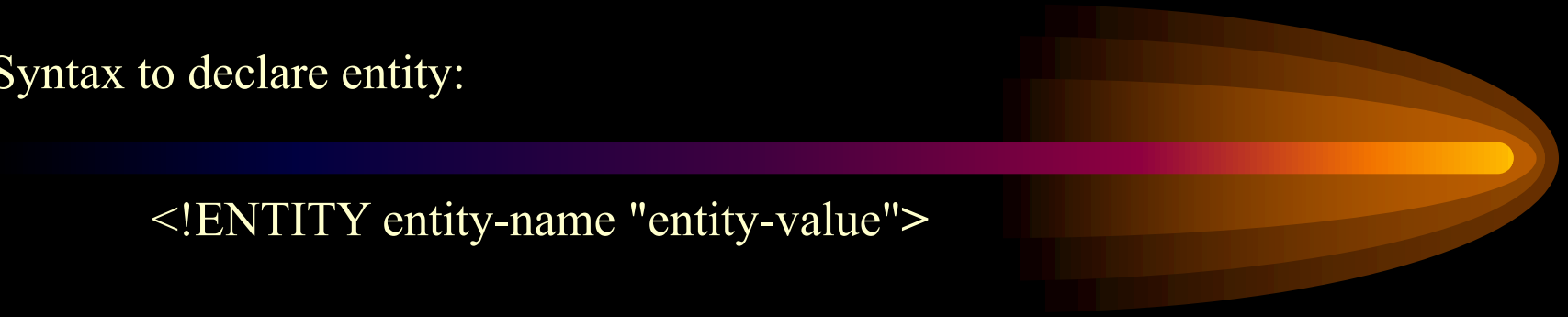
XML DTD with entity declaration

A doctype declaration can also define special strings that can be used in the XML file.

An entity has three parts:

- An ampersand (&)
- An entity name
- A semicolon (;)

Syntax to declare entity:



```
<!ENTITY entity-name "entity-value">
```

Example:

```
<?xml version="1.0" standalone="yes" ?>
<!DOCTYPE author [
  <!ELEMENT author (#PCDATA)>
  <!ENTITY ak ”Amit Kumar">
]>
<author>&ak;</author>
```

- In the above example, ak is an entity that is used inside the author element.
- In such case, it will print the value of sj entity that is ”Amit Kumar"

XML schema

- XML **schema** (also known as **XML schema definition** or **XSD**) use to describe the XML document structure. It is an alternative to DTD.

Example

```
xs:schema xmlns:xs = "http://www. medicaps.ac.in/XMLSchema">
<xs:element name = "employee">
<xs:complexType>
        <xs:sequence>
                <xs:element name = "firstname" type = "xs:string" />
                <xs:element name = "lastname" type = "xs:string" />
                <xs:element name = "phone" type = "xs:int" />
        </xs:sequence>
</xs:complexType>
 </xs:element>
</xs:schema>
```

Elements

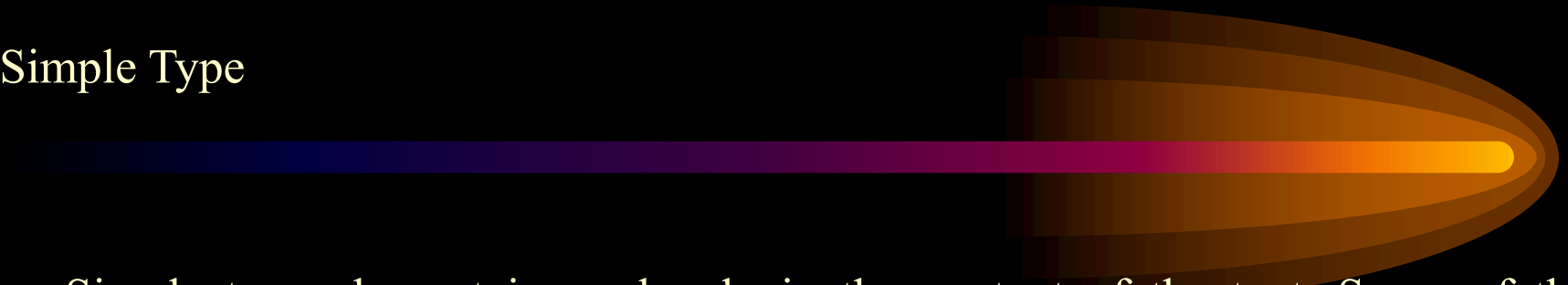
- Elements are the building blocks of XML document. An element can be defined within an XSD as follows

  <xs:element name = "x" type = "y"/>

Definition Types

1. Simple Type
2. Complex Type
3. Global Types

Simple Type



- Simple type element is used only in the context of the text. Some of the predefined simple types are: xs:integer, xs:boolean, xs:string, xs:date. For example −

```
<xs:element name = "phone_number" type = "xs:int" />
```

Complex Type

- A complex type is a container for other element definitions. This allows you to specify which child elements an element can contain and to provide some structure within your XML documents.

For example

```
<xs:element name = "Address">
<xs:complexType>
 <xs:sequence>
        <xs:element name = "name" type = "xs:string" />
        <xs:element name = "company" type = "xs:string" />
         <xs:element name = "phone" type = "xs:int" />
</xs:sequence>
</xs:complexType>
</xs:element>
```

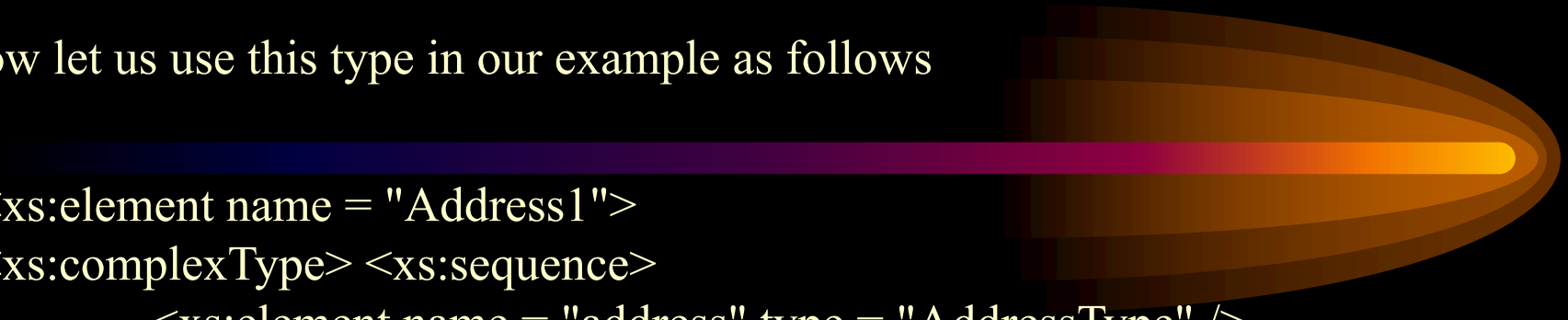
- In the above example, *Address* element consists of child elements. This is a container for other **<xs:element>** definitions, that allows to build a simple hierarchy of elements in the XML document

Global Types

- With the global type, you can define a single type in your document, which can be used by all other references.
- For example, suppose you want to generalize the *person* and *company* for different addresses of the company.
- In such case, you can define a general type as follows

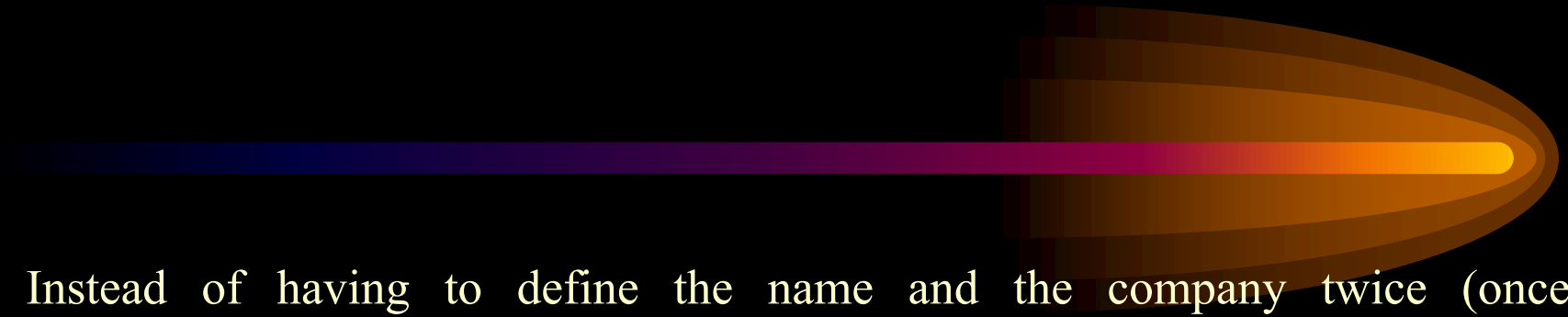
```
<xs:element name = "AddressType">
<xs:complexType>
 <xs:sequence>
<xs:element name = "name" type = "xs:string" />
<xs:element name = "company" type = "xs:string" />
 </xs:sequence>
 </xs:complexType>
</xs:element>
```

Now let us use this type in our example as follows



```
<xs:element name = "Address1">
<xs:complexType> <xs:sequence>
        <xs:element name = "address" type = "AddressType" />
         <xs:element name = "phone1" type = "xs:int" /> </xs:sequence>
 </xs:complexType>
</xs:element>

<xs:element name = "Address2">
<xs:complexType>
 <xs:sequence>
        <xs:element name = "address" type = "AddressType" />
        <xs:element name = "phone2" type = "xs:int" />
</xs:sequence>
 </xs:complexType>
</xs:element>
```



- Instead of having to define the name and the company twice (once for *Address1* and once for *Address2*),
- we now have a single definition. This makes maintenance simpler, i.e., if you decide to add "Postcode" elements to the address, you need to add them at just one place

Attributes

- Attributes in XSD provide extra information within an element. Attributes have *name* and *type* property as shown below :

```
<xs:attribute name = "x" type = "y"/>
```

Displaying XML Using CSS

- XML stands for **E**xtensible **M**arkup **L**anguage. It is a dynamic markup language. It is used to transform data from one form to another form.

An XML file can be displayed using two ways. These are as follows

1. Cascading Style Sheet
2. Extensible Stylesheet Language Transformation

Displaying XML file using CSS

- CSS can be used to display the contents of the XML document in a clear and precise manner. It gives the design and style to whole XML document

Basic steps in defining a CSS style sheet for XML

- For defining the style rules for the XML document, the following things shoulde be done :-Define the style rules for the text elements such as font-size, color, font-weight, etc.
- Define each element either as a block, inline or list element, using the display property of CSS.
- Identify the titles and bold them.

Linking XML with CSS

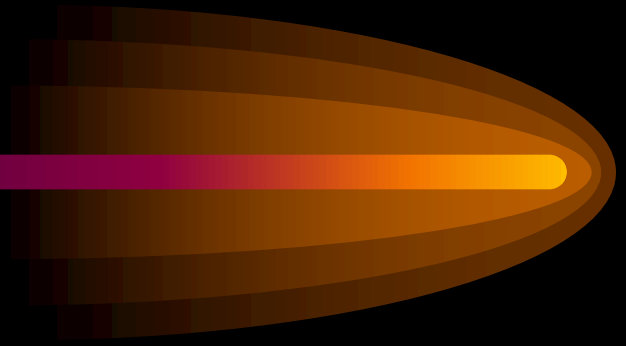
- In order to display the XML file using CSS, link XML file with CSS. Below is the syntax for linking the XML file with CSS:

```
<?xml-stylesheet type="text/css" href="name_of_css_file.css"?>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="Rule.css"?>
<books>
    <heading>Welcome To GeeksforGeeks </heading>
    <book>
        <title>Title -: Web Programming</title>
        <author>Author -: Chrisbates</author>
        <publisher>Publisher -: Wiley</publisher>
        <edition>Edition -: 3</edition>
        <price> Price -: 300</price>
    </book>
    <book>
        <title>Title -: Internet world-wide-web</title>
        <author>Author -: Ditel</author>
        <publisher>Publisher -: Pearson</publisher>
        <edition>Edition -: 3</edition>
        <price>Price -: 400</price>
    </book>
```

- In the above example, Books.xml is linked with Rule.css which contains the corresponding style sheet rules

Creating Rule.css as



```
books {
        color: white;
        background-color : gray;
        width: 100%;
}
heading {
        color: green;
        font-size : 40px;
        background-color : powderblue;
}
heading, title, author, publisher, edition, price {
        display : block;
}
title {
        font-size : 25px;
        font-weight : bold;
}
```

Advantages of displaying XML using CSS

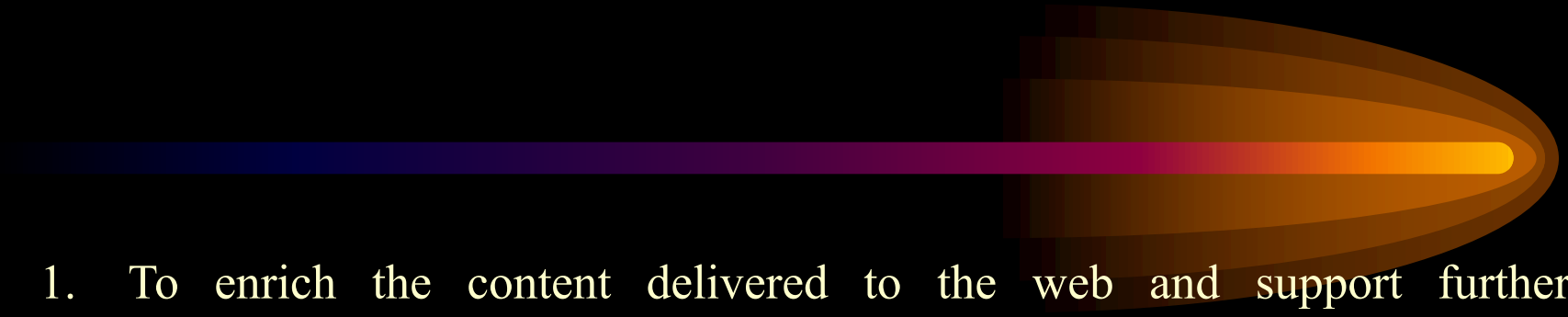
- CSS is used in XML or HTML to decorate the pages.
- CSS is used for interactive interface, so it is understandable by user.
- CSS enable multiple pages to share formatting, and reduce complexity and repetition in the structural content. So page loader is faster

Disadvantages of displaying XML using CSS

- Using CSS, no transformation can be applied to the XML documents.
- CSS uses different dimensions with different browsers. So the programmer has to run the code in different browser and test its compatibility to post it live.
- CSS have different level of versions, so it is confusing for the browser and user.

Embedding XML into HTML document

- HTML documents to support the inclusion and processing of XML data. This would allow an author to embed within a standard HTML document some well delimited, well defined XML object.
- The HTML document would then be able to support some functions based on the special XML markup.
- This strategy of permitting "islands" of XML data inside an HTML document would serve at least two purposes:



1. To enrich the content delivered to the web and support further enhancements to the XML-based content models.
2. To enable content developers to rely on the proven and known capabilities of HTML while they experiment with XML in their environments

Example

<HTML>
<body>
<!-- some typical HTML document with
<h1>, <h2>, <p>, etc. -->
<xml>
<!-- The <xml> tag introduces some XML-compliant markup for some specific purpose. The markup is then explicitly terminated with the </xml> tag. The user agent would invoke an XML processor only
on the data contained in the <xml></xml> pair. Otherwise the user agent would process the containing document as an HTML document. -->
</xml>
<!-- more typical HTML document markup -->
</body>
</html>

XSLT

- **XSLT** stands for **E**xtensible **S**tylesheet **L**anguage **T**ransformation.
- XSLT is used to transform XML document from one form to another form.
- XSLT uses Xpath to perform matching of nodes to perform these transformation .
- The result of applying XSLT to XML document could be an another XML document, HTML, text or any another document from technology perspective.
- The XSL code is written within the XML document with the extension of *(.xsl)*.
- In other words, an XSLT document is a different kind of XML document.

Displaying XML Using XSLT

XML file: Save: student.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl "href=”student.xsl" ?>
<student>
<s>
<name> Divyank Singh Sikarwar </name>
<branch> CSE</branch>
<age>18</age>
<city> Agra </city>
</s>
<s>
<name> Aniket Chauhan </name>
<branch> CSE</branch>
<age> 20</age>
<city> Shahjahanpur </city>
</s>
<s>
```

Cont….

```
<name> Simran Agarwal</name>
<branch> CSE</branch>
<age> 23</age>
<city> Buland Shar</city>
</s>
<s>
<name> Abhay Chauhan</name>
<branch> CSE</branch>
<age> 17</age>
<city> Shahjahanpur</city>
</s>
<s>
<name> Himanshu Bhatia</name>
<branch> IT</branch>
<age> 25</age>
<city> Indore</city>
</s>
</student>
```

XSLT Code: Save student.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h1 align="center">Students' Basic Details</h1>
<table border="3" align="center" >
<tr>
        <th>Name</th>
        <th>Branch</th>
        <th>Age</th>
        <th>City</th>
</tr>
        <xsl:for-each select="student/s">
```

```
<tr>
        <td><xsl:value-of select="name"/></td>
        <td><xsl:value-of select="branch"/></td>
        <td><xsl:value-of select="age"/></td>
        <td><xsl:value-of select="city"/></td>
</tr>
        </xsl:for-each>
        </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

OUTPUT

Students' Basic Details

Name	Branch	Age	City
Divyank Singh Sikarwar	CSE	18	Agra
Aniket Chauhan	CSE	20	Shahjahanpur
Simran Agarwal	CSE	23	Buland Shar
Abhay Chauhan	CSE	17	Shahjahanpur
Himanshu Bhatia	IT	25	Indore

Compare XML and XSLT

XML	XSLT
• XML is used for storing data in a structured format.	• XSLT is used for transforming and also for formatting XML file to display the content or to process the content.
• XML does not perform transformation of data.	• XSLT is a specialized language that performs transformation of one XML document into a different XML document. XSLT can also perform transformation of XML document into HTML document and XHTML document.
• XPath is a specialized language that is used to address portions of the XML file.	• XSLT will be using XPath for transformation and formatting of XML file.

THANK YOU !