# [Why this line: xmlns:android=“http://schemas.android.com/apk/res/android”?](Why this line: xmlns:android=\“http://schemas.android.com/apk/res/android\”?)

In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications. A user or an XML application will not know how to handle these differences. Name conflicts in XML can easily be avoided using a name prefix. When using prefixes in XML, a namespace for the prefix must be defined. Thus, this Name Space declaration must be included in the opening tag of the root view of our XML file.

The namespace declaration has the following syntax. xmlns:prefix="URI".

In computing, a uniform resource identifier (URI) is a string of characters used to identify a name of a resource. Such identification enables interaction with representations of the resource over a network, typically the World Wide Web, using specific protocols.

In Android, this is just the XML Name Space declaration. We use this Name Space in order to specify that the attributes belongs to Android. Thus they starts with "*android:*". We can actually create our own custom attributes. So to prevent the name conflicts where 2 attributes are named the same thing, but behave differently, we add the prefix "*android:*" to signify that these are Android attributes.

**Points Summary:**

1. xmlns means xml namespace.
2. It is created to avoid naming conflicts in the xml’s.
3. In order to avoid naming conflicts by any other way we need to provide each element with a prefix.
4. To avoid repetitive usage of the prefix in each xml tag we use xmlns at the root of the xml. Hence we have the tag **xmlns:android=”**[**http://schemas.android.com/apk/res/android**](http://schemas.android.com/apk/res/android)**”**
5. Now **android** here simply means we are assigning the namespace “<http://schemas.android.com/apk/res/android>” to it.
6. This namespace is not a URL but a URI also known as URN (universal resource name) which is rarely used in place of URI.
7. Due to this android will be responsible to identify the android related elements in the xml document which would be android:xxxxxxx etc. Without this namespace android:xxxxxxx will be not recognized.

[**Meaning of - <?xml version=“1.0” encoding=“utf-8”?>**](https://stackoverflow.com/questions/13743250/meaning-of-xml-version-1-0-encoding-utf-8)

(UTF - Unicode Transformation Format)

**Encoding** is the process of converting unicode characters into their equivalent binary representation. When the XML processor reads an XML document, it encodes the document depending on the type of encoding. Hence, we need to specify the type of encoding in the XML declaration.

## **Encoding Types**

There are mainly two types of encoding:

* UTF-8
* UTF-16

UTF (Unicode Transformation Format) stands for UCS Transformation Format, and UCS itself means Universal Character Set. The number 8 or 16 refers to the number of bits used to represent a character. They are either 8 (one byte) or 16 (two bytes). For the documents without encoding information, UTF-8 is set by default.

Optional preamble XML are:

* version="1.0" means that this is the XML standard this file conforms to
* encoding="utf-8" means that the file is encoded using the UTF-8 Unicode encoding

## **Syntax**

Encoding type is included in the prolog section of the XML document. The syntax for UTF-8 encoding is as follows:

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

The syntax for UTF-16 encoding is as follows:

<?xml version = "1.0" encoding = "UTF-16" standalone = "no" ?>

**Validation** is a process by which an XML document is validated. An XML document is said to be valid if its contents match with the elements, attributes and associated document type declaration (DTD), and if the document complies with the constraints expressed in it. Validation is dealt in two ways by the XML parser. They are:

* Well-formed XML document
* Valid XML document

## **Well-formed XML Document**

An XML document is said to be **well-formed** if it adheres to the following rules:

* Non DTD XML files must use the predefined character entities for **amp(&)**, **apos(single quote)**, **gt(>)**, **lt(<)**, **quot(double quote)**.
* It must follow the ordering of the tag. i.e., the inner tag must be closed before closing the outer tag.
* Each of its opening tags must have a closing tag or it must be a self ending tag.(<title>....</title> or <title/>).
* It must have only one attribute in a start tag, which needs to be quoted.
* **amp(&)**, **apos(single quote)**, **gt(>)**, **lt(<)**, **quot(double quote)**entities other than these must be declared.

### **Example**

Following is an example of a well-formed XML document −

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

<!DOCTYPE address

[

<!ELEMENT address (name,company,phone)>

<!ELEMENT name (#PCDATA)>

<!ELEMENT company (#PCDATA)>

<!ELEMENT phone (#PCDATA)>

]>

<address>

<name>Tanmay Patil</name>

<company>TutorialsPoint</company>

<phone>(011) 123-4567</phone>

</address>

The above example is said to be well-formed as −

* It defines the type of document. Here, the document type is **element** type.
* It includes a root element named as **address**.
* Each of the child elements among name, company and phone is enclosed in its self explanatory tag.
* Order of the tags is maintained.

## **Valid XML Documents**

An XML document is valid if there is a document type definition (DTD) or XML schema associated with it, and if it conforms to all rules of its DTD or schema. The rules an XML document must obey are:

* **The root element** of the document must be the one mentioned in DTD.
* A document should not contain arbitrary tags, while a well-formed document might contain **arbitrary tags**.
* Any tags used in document must be declared in DTD and must be used in the way permitted by DTD.

A validating parser reads DTD and checks the document for the validity. If it finds an error, it reports the same. Most of the browsers do not check for validity of document hence it is good practice to validate the XML document using parser before using.

## **The Unicode Character Sets**

Unicode can be implemented by different character sets. The most commonly used encodings are UTF-8 and UTF-16:

|  |  |
| --- | --- |
| **Character-set** | **Description** |
| UTF-8 | A character in UTF8 can be from 1 to 4 bytes long. UTF-8 can represent any character in the Unicode standard. UTF-8 is backwards compatible with ASCII. UTF-8 is the preferred encoding for e-mail and web pages |
| UTF-16 | 16-bit Unicode Transformation Format is a variable-length character encoding for Unicode, capable of encoding the entire Unicode repertoire. UTF-16 is used in major operating systems and environments, like Microsoft Windows, Java and .NET. |

* HTML 4 supports UTF-8. HTML 5 supports both UTF-8 and UTF-16.