|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **XUL** | **EPUB** | **ATOM** | **Android SDK** | **XBRL** |
| **Application domain** | an XML dialect for writing graphical user interface, enabling developers to write user interface elements in a manner similar to web pages.  More info: [XUL:Home Page - MozillaWiki](https://wiki.mozilla.org/XUL:Home_Page) | defines a distribution and interchange format for digital publications and documents. The EPUB format provides a means of representing, packaging and encoding structured and semantically enhanced Web content—including HTML, CSS, SVG and other resources—for distribution in a single-file container.  More info: [EPUB 3.2 (w3.org)](https://www.w3.org/publishing/epub32/) | an XML-based Web content and metadata syndication format, and an application-level protocol for publishing and editing Web resources belonging to periodically updated websites.  More info:  [https://validator.w3.org/ feed/docs/atom.html](https://validator.w3.org/%20feed/docs/atom.html) | used to implement the UI-related data  More info: [developer. android.com/ sdk/index.html](https://developer.android.com/sdk/index.html) | XBRL often termed “bar codes for reporting”, XBRL makes reporting more accurate and more efficient.  More info:  [https://www. xbrl.org/](https://www.xbrl.org/) |
| **Purpose** | Writing graphical user interface, enabling developers to write user interface elements in a manner similar to web pages | An ebook file format.became an official standard of the IDPF in September 2007, superseding the older Open eBook (OEB) standard | Used for web feeds  The Atom format was developed as an alternative to RSS | Building interfaces for android mobile applications | A freely available and global framework for exchanging business information. |
| **Function** | Widgets running on the Mozilla platform can be developed using XUL and ported across different platforms. Components of XUL documents include:  -content: The arrangement of UI components is defined in the documents that together constitute the contents of XUL files.  - skin: skin is regarded as the customizable set of colors or patterns that can be associated with a given XUL user interface defined in the form of skin files. It may include CSS and image files.  - locale: language changes can be incorporated in the user interface using internationalization and localization features provided by XUL. | supported by almost all hardware readers.  optimize text for a particular display.  Like a HTML web site, the format supports inline raster and vector images, metadata and css styling.  Better analytical support with compatible platforms | The atom content element is designed to support the direct inclusion of other XML vocabularies. | only contains tags, while implementing they need to be just invoked. | Allow:  - publishing reports to do so with confidence that the information contained in them can be consumed and analysed accurately.  - consuming reports to test them against a set of business and logical rules, in order to capture and avoid mistakes at their source.  - consuming the information to do so confident that the data provided to them conforms to a set of sophisticated pre-defined definitions |
| **Standard defining organization of the application** | **Mozilla** | **W3C** | **ISO** | **Google** | **Microsoft** |

1. **XUL**

XUL code starts with a definition of a namespace with the xmlns property which is always the same one:

xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul"

It a property of a window, which is created with the <window> tag:

<?xml version="1.0">

<window

xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">

</window>

**Displaying a text**

The component is the "description" tag and the text is assigned with the attribute "value" of this tag:

<description value= "Hello the World!"/>

**Label and Text box**

<label control="tb" value="Enter a text:" />

<textbox id="tb" multiline="true" rows="1" size="40" value="empty" />

<button label="Display" oncommand="document.getElementById('tb').value);" />

**Button**

<button label="Click" image="save.gif" orient="vertical" />

**Grid (Organizing the interface as a table)**

<grid flex="1">

<columns>

<column flex="1" />

<column flex="12" />

<columns>

<rows>

<row flex="12" />

<row />

</rows>

<gird>

1. **EPUB**

**An example EPUB file structure:**

--ZIP Container--

mimetype

META-INF/

container.xml

OEBPS/

content.opf

chapter1.xhtml

ch1-pic.png

css/

style.css

myfont.otf

**The epub:type Attribute**

1. Example 1

The following example shows how a premable could be marked up with the ***epub:type*** attribute on its containing HTML ***section*** element:

<html … xmlns:epub="http://www.idpf.org/2007/ops">

…

<section epub:type="preamble">

…

</section>

…

</html>

1. Example 2

The following example shows the ***epub:type*** attribute used to add glossary semantics on an HTML definition list:

<html … xmlns:epub="http://www.idpf.org/2007/ops">

…

<dl epub:type="glossary">

…

</dl>

…

</html>

1. Example 3

The following example shows the ***epub:type*** attribute used to add pagebreak semantics:

<html … xmlns:epub="http://www.idpf.org/2007/ops">

…

<p> … <span epub:type="pagebreak" id="p234"/> … </p>

…

</html>

1. **ATOM**

**The <feed> element**

A Feed consists of some metadata, followed by any number of entries.

**Required <feed> element**

* id: Identifies the feed using a universally unique and permanent URI. If you have a long-term, renewable lease on your Internet domain name, then you can feel free to use your website's address.

<id>http://example.com/</id>

* title: Contains a human readable title for the feed. Often the same as the title of the associated website. This value should not be blank.

<title>Example, Inc.</title>

* updated: Indicates the last time the feed was modified in a significant way.

<updated>2003-12-13T18:30:02Z</updated>

* author: Names one author of the feed. A feed may have multiple author elements. A feed must contain at least one author element unless **all** of the entry elements contain at least one author element.

<author>

   <name>John Doe</name>

   <email>JohnDoe@example.com</email>

   <uri>http://example.com/~johndoe</uri>

</author>

* link: Identifies a related Web page. The type of relation is defined by the rel attribute. A feed is limited to one alternate per type and hreflang. A feed should contain a link back to the feed itself

<link rel="self" href="/feed" />

**The <entry> element**

An example of an entry would be a single post on a weblog

**Require element of <entry>**

* id: Identifies the entry using a universally unique and permanent URI. Two entries in a feed can have the same value for id if they represent the same entry at different points in time.

<id>http://example.com/blog/1234</id>

* title: Contains a human readable title for the entry. This value should not be blank.

<title>Atom-Powered Robots Run Amok</title>

* updated: Indicates the last time the entry was modified in a significant way. This value need not change after a typo is fixed, only after a substantial modification. Generally, different entries in a feed will have different updated timestamps.

<updated>2003-12-13T18:30:02-05:00</updated>

* author: Names one author of the entry. An entry may have multiple authors. An entry must contain at least one author element unless there is an author element in the enclosing feed, or there is an author element in the enclosed source element.

<author>

<name>John Doe</name>

</author>

* content: Contains or links to the complete content of the entry. Content must be provided if there is no alternate link, and should be provided if there is no summary

<content>complete story here</content>

* link: Identifies a related Web page. The type of relation is defined by the rel attribute. An entry is limited to one alternate per type and hreflang. An entry must contain an alternate link if there is no content element.

<link rel="alternate" href="/blog/1234"/>

* summary: Conveys a short summary, abstract, or excerpt of the entry. Summary should be provided if there either is no content provided for the entry, or that content is not inline (i.e., contains a src attribute), or if the content is encoded in base64.

<summary>Some text.</summary>

1. **Android SDK**

**XML file example:**

<?xml version="1.0" encoding="utf-8"?>  
<[**ViewGroup**](https://developer.android.com/guide/topics/resources/layout-resource#viewgroup-element)  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:id="@[+][package:]id/resource\_name"  
    android:layout\_height=["dimension" | "match\_parent" | "wrap\_content"]  
    android:layout\_width=["dimension" | "match\_parent" | "wrap\_content"]  
    [ViewGroup-specific attributes] >  
    <[**View**](https://developer.android.com/guide/topics/resources/layout-resource#view-element)  
        android:id="@[+][package:]id/resource\_name"  
        android:layout\_height=["dimension" | "match\_parent" | "wrap\_content"]  
        android:layout\_width=["dimension" | "match\_parent" | "wrap\_content"]  
        [View-specific attributes] >  
        <[**requestFocus**](https://developer.android.com/guide/topics/resources/layout-resource#requestfocus-element)/>  
    </View>  
    <[**ViewGroup**](https://developer.android.com/guide/topics/resources/layout-resource#viewgroup-element) >  
        <[**View**](https://developer.android.com/guide/topics/resources/layout-resource#view-element) />  
    </ViewGroup>  
    <[**include**](https://developer.android.com/guide/topics/resources/layout-resource#include-element) layout="@layout/layout\_resource"/>  
</ViewGroup>

**<ViewGroup> elements**

A container for other [View](https://developer.android.com/reference/android/view/View) elements. There are many different kinds of [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup) objects and each one lets you specify the layout of the child elements in different ways. Different kinds of [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup) objects include [LinearLayout](https://developer.android.com/reference/android/widget/LinearLayout), [RelativeLayout](https://developer.android.com/reference/android/widget/RelativeLayout), and [FrameLayout](https://developer.android.com/reference/android/widget/FrameLayout).

* Attributes
  + android:id

Resource ID. A unique resource name for the element, which you can use to obtain a reference to the [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup) from your application.

* + android:layout\_height

Dimension or keyword. **Required**. The height for the group, as a dimension value (or [dimension resource](https://developer.android.com/guide/topics/resources/more-resources#Dimension)) or a keyword ("match\_parent" or "wrap\_content").

* + android:layout\_width

Dimension or keyword. **Required**. The width for the group, as a dimension value (or [dimension resource](https://developer.android.com/guide/topics/resources/more-resources#Dimension)) or a keyword ("match\_parent" or "wrap\_content").

**<View> elements**

An individual UI component, generally referred to as a "widget". Different kinds of [View](https://developer.android.com/reference/android/view/View) objects include [TextView](https://developer.android.com/reference/android/widget/TextView), [Button](https://developer.android.com/reference/android/widget/Button), and [CheckBox](https://developer.android.com/reference/android/widget/CheckBox).

* Attributes:
  + android:id

Resource ID. A unique resource name for the element, which you can use to obtain a reference to the [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup) from your application.

* + android:layout\_height

Dimension or keyword. **Required**. The height for the group, as a dimension value (or [dimension resource](https://developer.android.com/guide/topics/resources/more-resources#Dimension)) or a keyword ("match\_parent" or "wrap\_content").

* + android:layout\_width

Dimension or keyword. **Required**. The width for the group, as a dimension value (or [dimension resource](https://developer.android.com/guide/topics/resources/more-resources#Dimension)) or a keyword ("match\_parent" or "wrap\_content").

**<requestFocus> elements**

Any element representing a [View](https://developer.android.com/reference/android/view/View) object can include this empty element, which gives its parent initial focus on the screen. You can have only one of these elements per file.

**<include> elements**

Includes a layout file into this layout.

* Attributes

* + layout:

Layout resource. **Required**. Reference to a layout resource.

* + android:id

Resource ID. Overrides the ID given to the root view in the included layout.

* + android:layout\_height:

Dimension or keyword. Overrides the height given to the root view in the included layout. Only effective if android:layout\_width is also declared.

* + android:layout\_width:

Dimension or keyword. Overrides the width given to the root view in the included layout. Only effective if android:layout\_height is also declared.

**<merge> elements**

An alternative root element that is not drawn in the layout hierarchy. Using this as the root element is useful when you know that this layout will be placed into a layout that already contains the appropriate parent View to contain the children of the <merge> element. This is particularly useful when you plan to include this layout in another layout file using [<include>](https://developer.android.com/guide/topics/resources/layout-resource#include-element) and this layout doesn't require a different [ViewGroup](https://developer.android.com/reference/android/view/ViewGroup) container. For more information about merging layouts, read [Re-using Layouts with <include/>](https://developer.android.com/training/improving-layouts/reusing-layouts).

1. **XBRL**

**XLINK in XBRL**

This specification defines the XML Linking Language (XLink), which allows elements to be inserted into XML documents in order to create and describe [links](https://www.w3.org/TR/xlink/#dt-link) between resources.

XLink provides a framework for creating both basic unidirectional links and more complex linking structures

1. Simple links

A simple link is a link that points from one resource to another [[XLINK]](http://www.xbrl.org/specification/xbrl-2.1/rec-2003-12-31/xbrl-2.1-rec-2003-12-31+corrected-errata-2013-02-20.html#XLINK)

* The @xlink:type attribute on simple links

The @xlink:type attribute **MUST** occur and **MUST** have the fixed content "simple".

* The @xlink:href attribute on simple links

A simple link **MUST** have an @xlink:href attribute. The @xlink:href attribute **MUST** be a URI. The URI **MUST** point to an XML document or to an XML fragment within an XML document. If the URI is relative, it **MUST** be resolved to obtain an absolute URI as specified in XML Base specification [[XML Base]](http://www.xbrl.org/specification/xbrl-2.1/rec-2003-12-31/xbrl-2.1-rec-2003-12-31+corrected-errata-2013-02-20.html#XMLBASE).

* The @xlink:role attribute on simple link (optional)

The optional @xlink:role attribute **MUST** take URI values. If it is provided, the @xlink:role attribute **MUST NOT** be empty.

* The @xlink:arcrole attribute on simple link (optional)

If it occurs, the @xlink:arcrole attribute **MUST NOT** be an empty string

* The @xml:base attribute on simple link (optional)

The @xml:base attribute [[XML Base]](http://www.xbrl.org/specification/xbrl-2.1/rec-2003-12-31/xbrl-2.1-rec-2003-12-31+corrected-errata-2013-02-20.html#XMLBASE) **MAY** appear on the simple links, participating in the resolution of relative URIs specified in their @xlink:href attributes