

xlmns: (namespace URI)

Uniform Resource Indicator(URI):

- In computing, it 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.



xmlns: (namespace URI)

xmlns:prefix="URI"

- Ex: <http://schemas.android.com/apk/res/android:id> is the URI here
- The most common URI is the Uniform Resource Locator (URL) which identifies an Internet domain address. Another, not so common type of URI is the [Uniform Resource Name \(URN\)](#).

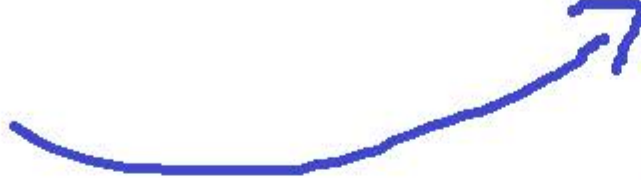



XML Namespace

- XML namespaces are used for providing uniquely named elements and attributes in an XML document.
- xmlns:android describes the android namespace.
- Also suppose we write our own textview widget with different features compared to android textview, and namespace helps to distinguish between our custom textview widget and android textview widget.



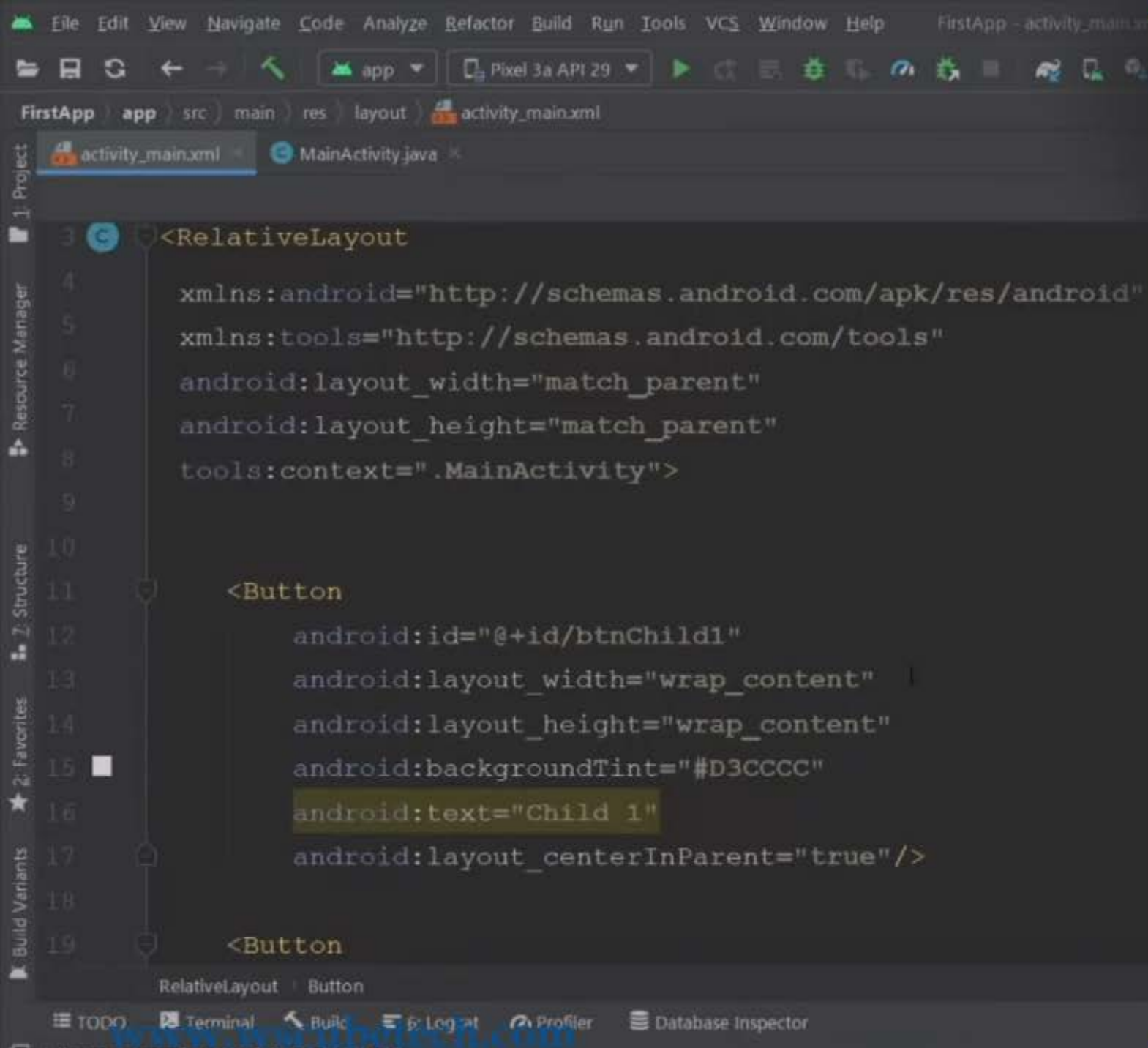
Important Attributes

- android:id 
- android:layout_width (required)
- android:layout_height (required)
- android:text 

--> view ko java & XML sai connect karna kai liya use karta hai .

--> Kisi specific view ko identify kar kai usa call karna kai liya use hota hai.

--> View mai text add karna kai liya use hota hai.



--> In Android development, size types typically refer to the different ways. They are as follow:

1. Fixed Size: Dimensions are set to specific pixel values(px) or density pixel(dp), providing a fixed size for UI elements. While this ensures precision, it may not scale well on devices with different screen sizes or resolutions.

2. Wrap Content: UI elements adjust their size based on their content, wrapping around the content they contain. This is useful when you want an element to take up only as much space as necessary.

3. Match Parent: UI elements expand to fill the available space within their parent container. This is useful for creating layouts where a particular view should occupy the entire available width or height.

--> Layout ya root element wrap_content nahi hota kyunki yeh design aur layout ka structure define karta hai. Jab aap wrap_content ka istemal karte hain, toh woh apne content ke hisab se expand ho jata hai. Lekin, agar root element apne content ke hisab se badhta rahe, toh yeh unpredictable hoga aur aapki UI ka layout sahi taur par maintain nahi hoga.

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help FirstApp activity_main.xml [FirstApp.app] - Android Studio

Pixel 3a API 29

FirstApp > app > src > main > res > layout > activity_main.xml

activity_main.xml MainActivity.java

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

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:backgroundTint="#DEADB8"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

LinearLayout Button

Component Tree

Flutter Outline Flutter Inspector Flutter Performance Layout Validator

WScUBE TECH®
System For Satisfaction


Code Split Design

Pixel 30 FirstApp

Attributes

daemon started successfully (today 10:54)

TODO Terminal Build Logcat Profiler Database Inspector



Looking for Android training? Call us at : +91 9024244886 / 9269698122 or visit - www.wscubetech.com

Feature	LinearLayout	RelativeLayout
Definition in XML	<LinearLayout>	<RelativeLayout>
Child View Placement	Linear arrangement (either horizontal or vertical)	Relative placement (based on relationships between views)
Orientation	android:orientation attribute (horizontal or vertical)	Not applicable; Views are placed relative to each other
Flexibility	Simple and easy to use for linear arrangements	More complex, but offers greater flexibility in positioning views
Performance	Generally efficient for simple linear layouts	May be less efficient for complex layouts due to the need to calculate relationships
Nesting Views	Easily nests other layouts, including additional LinearLayouts	Supports nested layouts, and child views can be positioned relative to views in parent or sibling RelativeLayouts
Gravity Attribute	android:gravity attribute to control alignment of children within the layout	Not applicable; Alignment controlled through positioning attributes
Positioning of Views	Limited to linear arrangements; views are positioned based on order	Offers more control over the positioning of views relative to each other
Use Cases	Well-suited for simple linear arrangements of views, such as rows or columns	Suitable for more complex layouts where views need to be positioned relative to each other