Android Basics: User Interface

Building Layouts: Part 1 – Views

1. View – a rectangular area visible on the screen, it has a width, height and color.
2. ImageView – displays an image such as an icon or a photo.
3. TextView – displays text
4. Button - TextView that is sensitive to touch
5. ViewGroup – a big view that contains and positions the smaller Views inside it

Views are the basic building blocks that you use to create visuals on your screen.

Textview

XML Syntax

```
<TextView
  android:text="Happy Birthday!"
  android:background="@android:color/darker_gray"
  android:layout_width="150dp"
  android:layout_height="75dp" />
```

Linear Layout

```
<LinearLayout
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:orientation="vertical">

<TextView
  android:text="Happy Birthday!"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content" />
```

```
<TextView
  android:text="You're the best!"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content" />

</LinearLayout>
```

####

Change the TextView

We use the unit 'dp' to describe the size of views in android, as well as the space between views.

Density-Independent Pixels

(Example)

- Medium Resolution Device : 2 Pixels takes up a quarter of the device
- High Resolution Device : 2 pixels takes up a smaller portion of the device, because the screen has more pixels on the same size screen
- Extra-High Resolution Device: 2 pixels takes up even less space on this screen than the previous ones, but still only takes up 2 pixels

By using `dp`, we avoid using the individual pixels on the screen, and instead we use pixels that are *density independent*.

```
Note – Make touch targets at least 48dp
```

Getting Past Errors

Debugging Steps:

1. Read the error message
2. Compare to working code
3. Undo
4. Ask for help

Setting Wrap Content

Instead of setting the width and height as 'dp', you can use `wrap_content`, to ensure that the content will always fit correctly.

TextView Text Size

`android:textsize="45sp"`

`sp` is a unit of measurement, like `dp`. It stands for scale-independent pixels. It makes the app look consistent across different devices.

`sp` is only used for fonts

> If you look at the Material Design Spec on Google's website, it gives you lots of information about how to make your app look good across devices. And important part is under Style > Typography. It gives you the recommended font size for different font settings like Headline, Title, Body, Caption, Display 1, etc
> `android:textAppearanceLarge="?android:textAppearanceLarge`

TextView Text Color

- `android:textColor="#325AEF"`

Simple ImageView

```
<ImageView
  android:src="@drawable/cake"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:scaleType="center" />
```

- We use the @ symbol to say that we're referencing a resource in the Android app, and `drawable` is an image type.
- `scaleType` – tells the device how to scale up or down the image based on the bounds of the image
  - center – zooms in on the center of the image

- centerCrop - crops out the very edge of the image and makes the image 'full-bleed' (all the way from one end to the other)

Documentation

https://developer.android.com – has all the info you could possibly need about android development

---

Building Layouts: Part 2

We have gone over Select Views, and Style Views. Up next: View Groups

ViewGroups

There can only be one Root Group on a page. Everything else must be placed into ViewGroups.

A **ViewGroup** is a container for Views.

1. Parent Views
    1. Relative Layout
    2. Linear Layout
2. Children Views
    1. TextView
    2. ImageView

Types of ViewGroups

1. Linear Layout
    1. Vertical Column
    2. Horizontal Row
2. Relative Layout
    1. Relative to parent
    2. Relative to other children

Linear Layout

`android:orientation` – can be horizontal for a row or vertical for a column

Always add this namespace declaration in the opening tag of the root crew of your XML file

`xmln:android="http://shemas.android.com/apk/res/android"`

Width and Height

`match_parent` – matches the size of the parent
`wrap_content` – wraps the content

NOTE: Anytime you see an attribute that starts with layout_, these are actually view group parameters, and they are handled by the parent

Evenly Spacing Out Children

Taking advantage of the space on the screen:
If you set `layout_width` to 0dp and `layout_weight` of the buttons to 1, the space will be shared equally between all buttons

Layout

- In order to work with `layout_weight`, you add up all the `layout_weights` in the ViewGroup. This is the denominator of your fraction. Now put any `layout_weight` over the calculated denominator and you now have the percentage of the parent group that the element will take up.

Relative Layout

Relative to parent

1. Children can be positioned relative to the parent's top, bottom, right or left edge
2. `android:layout_alignParentTop="true"`
3. `android:layout_alignParentBottom="true"`
4. `android:layout_alignParentLeft="true"`
5. `android:layout_alignParentRight="true"`
6. In order to position something in the corner, you can set combinations of 2 of the alignParent attributes to be true

7. `android:layout_centerHorizontal`
8. `android:layout_centerVertical`

Relative to Other Views

Assigning View ID Names

```
android:id="@+id/ben_text_view"
```

`@` refers to a resource in our Android app
`+` means that we are declaring this id for the first time

1. You can set constraints on Views with relative layouts.
2. `android:layout_toLeftOf="ben_text_view"`
3. `android:layout_above="ben_text_view"`

List Item with RelativeLayout

Padding vs. Margin

1. Padding
    1. Padding gets applied to the TextView itself
    2. `android:padding="8dp"`
    3. `android:paddingBottom="8dp"`
    4. `android:paddingTop="8dp"`
    5. `android:paddingRight="8dp"`
    6. `android:paddingLeft="8dp"`
2. Margin
    1. Margin gets applied to the Parent view
    2. `android:layout_margin="8dp"`
    3. `android:layout_marginTop="8dp"`
    4. `android:layout_marginBottom="8dp"`
    5. `android:layout_marginRight="8dp"`
    6. `android:layout_marginLeft="8dp"`

The material design guidelines recommend 8dp in on each side
16dp from the edge.

Practice Set: Building Layouts

Application Name

Company Domain

Package Name – Becomes the package name, must be unique in the App Store

#programming/java/android/beginner/1-user-interface