Android App Resources

- These are additional files and static content that your code uses
  - Example
    - bitmaps,
    - layout definitions,
    - user interface strings,
    - animation instructions
- externalize app resources from your code to maintain them independently
- We must provide alternative resources for specific device configurations
  - At runtime, Android uses the appropriate resource based on the current configuration.
  - Example - provide a different UI layout depending on the screen size
- Externalized app resources can be accessed by using resource IDs that are generated in your project's `R` class.
- **Grouping resource types**
  - Place each type of resource in a specific subdirectory of your project's `res/` directory.

```
MyProject/
    src/
        MyActivity.java
    res/
        drawable/
            graphic.png
        layout/
            main.xml
            info.xml
        mipmap/
            icon.png
        values/
            strings.xml
```

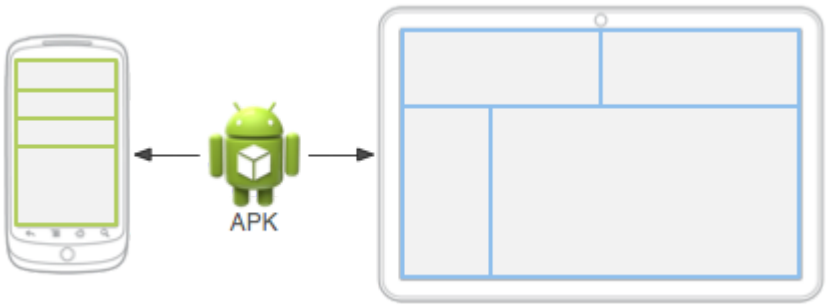
- Resource directories supported inside project `res/` directory.

Directory	Resource Type
`animator/`	XML files that define property animations.
`color/`	XML files that define a state list of colors.
`drawable/`	Bitmap files (`.png`, `.9.png`, `.jpg`, `.gif`)
`mipmap/`	Drawable files for different launcher icon densities.
`layout/`	XML files that define a user interface layout.
`menu/`	XML files that define app menus, such as an Options Menu, Context Menu, or Sub Menu.
`raw/`	Arbitrary files to save in their raw form. To open these resources with a raw `InputStream`, call`Resources.openRawResource()` with the resource ID, which is `R.raw.filename`.

`values/`	XML files that contain simple values, such as strings, integers, and colors.
`xml/`	Arbitrary XML files that can be read at runtime by calling `Resources.getXML()`. Various XML configuration files must be saved here.
`font/`	Font files with extensions such as `.ttf`, `.otf`, or `.ttc`, or XML files that include a `<font-family>` element.

Providing alternative resources

- Alternative resources - to support specific device configurations.
- Example:-
    - alternative drawable resources for different screen densities and
    - Alternative string resources for different languages.
- Note:-
    - At runtime, Android detects the current device configuration and loads the appropriate resources for your app.



Two different devices, each using different layout resources.

To specify configuration-specific alternatives for a set of resources:

Create a new directory in `res/` named in the form `<resources_name>-<config_qualifier>`.

- `<resources_name>` - directory name of the corresponding default resources
- `<qualifier>` - name that specifies an individual configuration for which these resources are to be used

```
res/
    drawable/
        icon.png
        background.png
    drawable-hdpi/
        icon.png
        background.png
```

Accessing your app resources

- Apply/use in code it by referencing its resource ID.
- All resource IDs are defined in your project's `R` **class**, which the `aapt` **tool automatically generates.**
- For each type of resource, there is an `R` **subclass**
    - for example,
        - `R.drawable` for all drawable resources
- a static integer - for each resource (of that type) – Resource ID
    - for example,
        - `R.drawable.icon`

A **resource ID** is always composed of:

- **The *resource type*:** (Each resource is grouped into a "type,")
    - Ex:- `string`, `drawable`, and `layout`.
- **The *resource name*:**
    - filename (excluding the extension) or
    - the value in the XML `android:name` attribute, if the resource is a simple value (such as a string).
- two ways you can access a resource:
    - **In code:** Using a static integer from a sub-class of your `R` class, such as:
        - `R.string.hello`
            - R – Class name
            - `string` - resource type and
            - `hello` - resource name
    - **In XML:**
        - `@string/hello`

Accessing resources in code

```
ImageView imageView = (ImageView) findViewById(R.id.myimageview);
```

```
[<package_name>.]R.<resource_type>.<resource_name>
```

Accessing resources from XML

```
@[<package_name>:]<resource_type>/<resource_name>
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <color name="opaque_red">#f00</color>
   <string name="hello">Hello!</string>
</resources>
```

You can use these resources in the following layout file to set the text color and text string:

```
<?xml version="1.0" encoding="utf-8"?>
<EditText xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:textColor="@color/opaque_red"
    android:text="@string/hello" />
```

Referencing style attributes

To reference a style attribute, the name syntax is almost identical to the normal resource format, but instead of the at-symbol (`@`), use a question-mark (`?`), and the resource type portion is optional. For instance:

```
?[<package_name>:][<resource_type>/]<resource_name>
```

For example, here's how you can reference an attribute to set the text color to match the "primary" text color of the system theme:

```
<EditText id="text"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="?android:textColorSecondary"
    android:text="@string/hello_world" />
```