Session 1 – Intro to Android

Practis הכנה פרקטית להי-טק

Course overview

Course goal –

Learn and obtain hands-on experience in developing basic to advanced Android apps, utilizing various modern Android device capabilities.

Course overview

Very important prerequisites –

- Android development is based on Java
- We are assuming you are familiar with:
  - Variables
  - Flow control: if, for, while
  - Arrays, matrices and Strings
  - Exploring Java APIs
  - ArrayList
  - Exceptions
  - Basic multithreading concepts
  - OOP concepts

Course overview

Course structure –

- Three daily meetings:
  - 09:00-10:30 First session
  - 10:30-10:45 Recess
  - 10:45-12:00 Second session
  - 12:00-13:00 Lunch break
  - 13:00-15:00 Third Session
  - 15:00-15:15 Recess
  - 15:15-17:00 Fourth Session

Course syllabus

Android development – course syllabus

- **Session 1 – First android app**
- Session 2 – Multiple screens & Menus
- Session 3 – Files & Network
- Session 4 – Touch, Permission and feedback
- Session 5 – Location And basic animations
- Session 6 – Ads, Uploading to Google Play

Session overview

- **What is Android**
- **First Android App**
- **Testing applications**
- **Android project structure**
- **XML**
- **Enhancing First Android App**

What is Android

Android

- Google’s smart phones OS
- First release in 2008
- Based on Linux Kernel
- Written in C, C++ and Java
- Supports 100+ languages

What is Android

Android

- More common than Apple and MS
- Over 3.5M applications in the market
- 71% of mobile developers develop for Android
- Not only for smart phones
  - Android TV
  - Android Car
  - Android Wear

What is Android

APK

- Android application package
- Format for copying and installing Android apps (Based on JAR/ZIP)
  - Try to open it using WinRAR
- Can be copied or downloaded from

Google Play

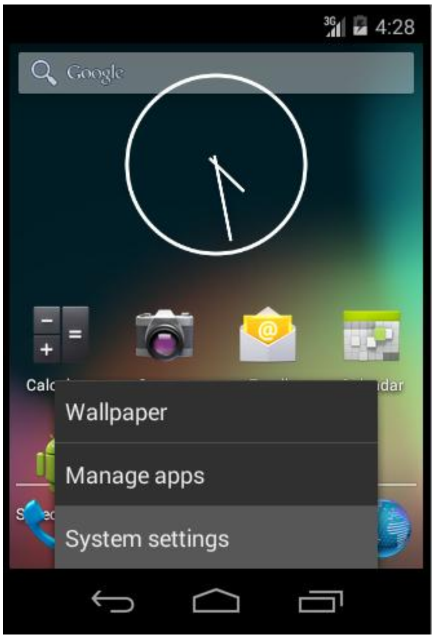
What is Android

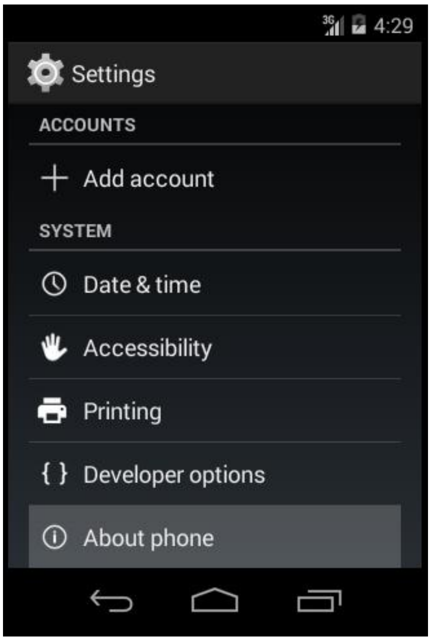
Android API

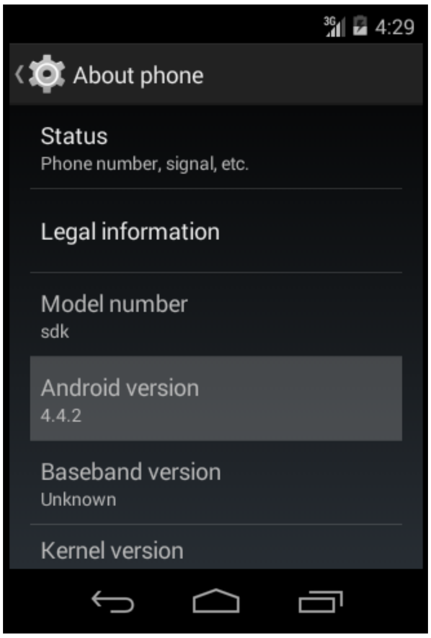
- Google releases new Android versions
- Every version has unique API number
- Each version contains –
  - Bug fixes
  - Improved performance
  - New features
  - **Removal of old features**

What is Android

Android API – Knowing my version







Status
Phone number, signal, etc.

Legal information

Model number
sdk

Android version
4.4.2

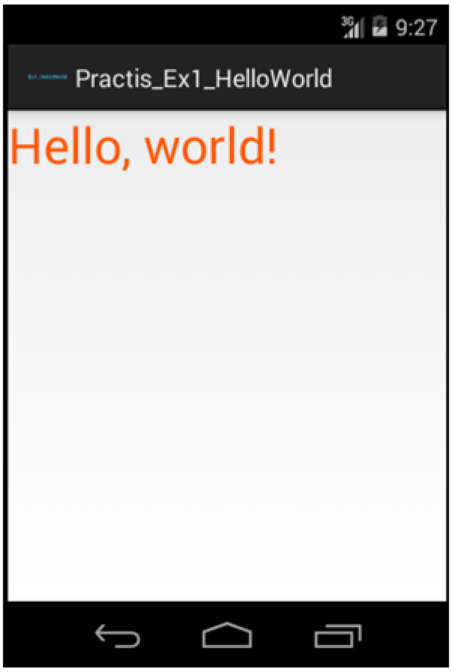
Baseband version
Unknown

Kernel version

First Android App

Hello, World!

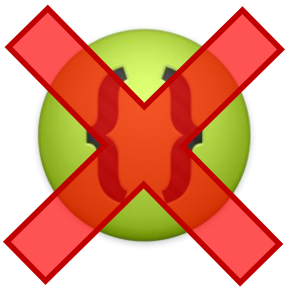
- Typical first application
- Relatively simple to create
- Outputs “Hello, world!” to the screen
- Android Studio as IDE



First Android App

Android studio –

- Google’s official Android IDE
- Based on IntelliJ IDEA community
- Only supports Android development
- Replaces previous Eclipse ADT



First Android App

JetBrains –

- Czech company (original name: IntelliJ)
- Many excellent development tools
  - IDEA (Java + Android)
  - CodeX (iPhone)
  - .Net tools – ReSharper, dotTrace
  - TeamCity (continuous integration)
  - Many more

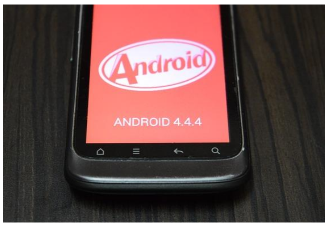
First Android App

Android theme –

- Style to set to the entire application
- Affects how the app looks like, examples:
  - Default font (type, color, size..)
  - Foreground and background colors
  - Window style (border, title, full-screen)
- **I recommend starting with no special theme (none)**

Testing Applications

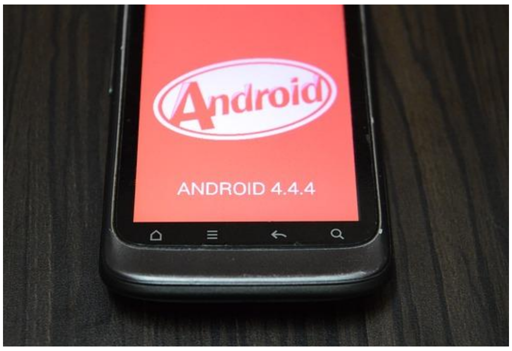
Using real device



- Connect phone to development machine
- Need to allow debugging in phone
- **Need correct drivers on computer**

Testing Applications

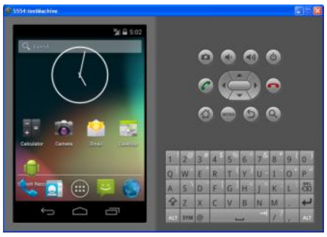
Using real device



- Pros:
  - Accurate
  - Fast
  - Comfortable
- Cons:
  - Many devices for many resolutions
  - Expensive

Testing Applications

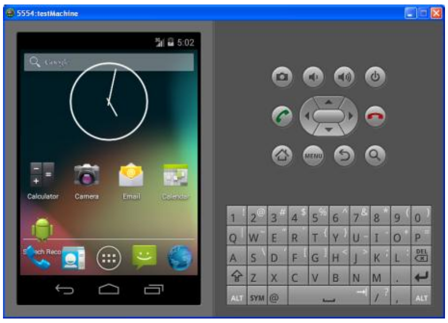
Using the Emulator



- An application simulating real device
- There is built in Emulator in ADT
- Other Emulators might be used (genyMotion)

Testing Applications

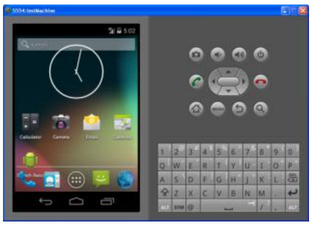
Using the Emulator



- Pros:
  - Test various resolutions easily
  - Test various APIs easily
  - Cheap
- Cons:
  - **SLOW!!**
  - Never 100% accurate

Testing Applications

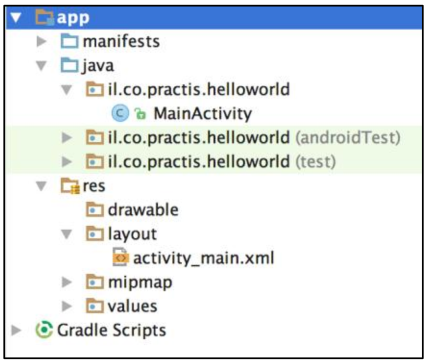
Tips for faster Emulator



- **Intel x86 Emulator Accelerator**
- Use snapshots
- Emulate simpler devices
- Give sufficient RAM to device
- Close only the app, not the Emulator

Android Project Structure

Project Structure –



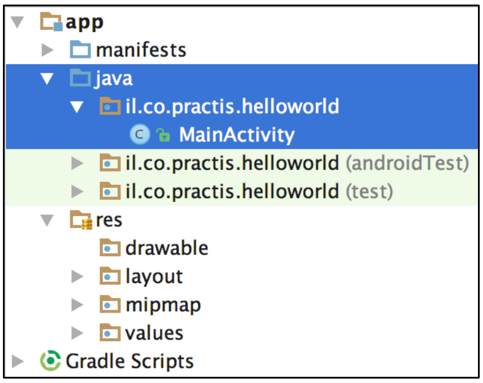
```
▼ app
    ▶ manifests
    ▼ java
        ▼ il.co.practis.helloworld
            MainActivity
        ▶ il.co.practis.helloworld (androidTest)
        ▶ il.co.practis.helloworld (test)
    ▼ res
          drawable
        ▼ layout
              activity_main.xml
        ▶ mipmap
        ▶ values
▶ Gradle Scripts
```

- Might get confusing at first
- Contains many files
- Many Meta Data files
- Some updated automatically

Android Project Structure

Project Structure – Code

- Under the java folder
- Organized by packages
- We start with one activity

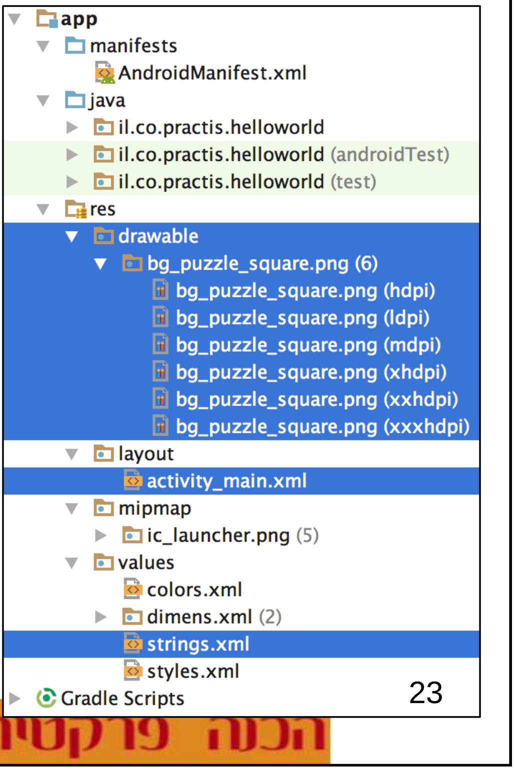


```
▼ app
   ▶ manifests
   ▼ java
      ▼ il.co.practis.helloworld
            MainActivity
      ▶ il.co.practis.helloworld (androidTest)
      ▶ il.co.practis.helloworld (test)
   ▼ res
         drawable
      ▶ layout
      ▶ mipmap
      ▶ values
▶ Gradle Scripts
```

Android Project Structure

Project Structure – res

- Holds various meta data files (Mostly XML):
  - Screen definitions (Layout)
  - Strings (languages)
  - Styles
- Images in variant resolutions



```
▼ app
  ▼ manifests
      AndroidManifest.xml
  ▼ java
    ▶ il.co.practis.helloworld
    ▶ il.co.practis.helloworld (androidTest)
    ▶ il.co.practis.helloworld (test)
  ▼ res
    ▼ drawable
      ▼ bg_puzzle_square.png (6)
          bg_puzzle_square.png (hdpi)
          bg_puzzle_square.png (ldpi)
          bg_puzzle_square.png (mdpi)
          bg_puzzle_square.png (xhdpi)
          bg_puzzle_square.png (xxhdpi)
          bg_puzzle_square.png (xxxhdpi)
    ▼ layout
        activity_main.xml
    ▼ mipmap
      ▶ ic_launcher.png (5)
    ▼ values
        colors.xml
      ▶ dimens.xml (2)
        strings.xml
        styles.xml
▶ Gradle Scripts
```

Android Project Structure

Project Structure – Screen resolutions

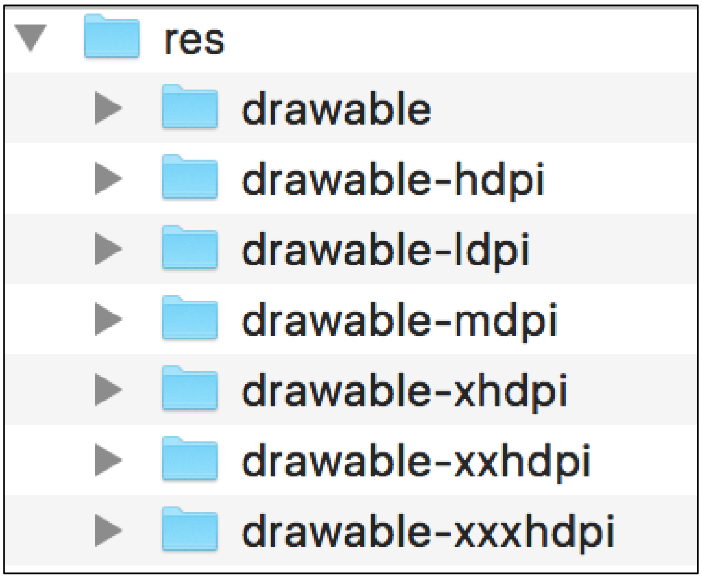
- ldpi ~120dpi / 36x36 (0.75x)
- mdpi ~160dpi / 48x48 (1.0x baseline)
- hdpi ~240dpi / 72x72 (1.5x)
- xhdpi ~320dpi / 96x96 (2.0x)
- xxhdpi ~480dpi / 180x180 (3.0x)
- nodpi

http://romannurik.github.io/AndroidAssetStudio/nine-patches.html

Android Project Structure

Project Structure – Screen resolutions

- Saved in different res folders:

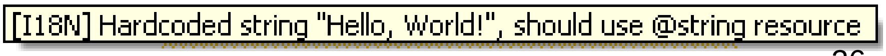


res
- drawable
- drawable-hdpi
- drawable-ldpi
- drawable-mdpi
- drawable-xhdpi
- drawable-xxhdpi
- drawable-xxxhdpi

Android Project Structure

Project Structure – strings.xml

- Holds all texts for the app
- Used to support the 70 languages
- We have one file per language
- Android will choose which to use
- Any string outside the file will cause a warning:

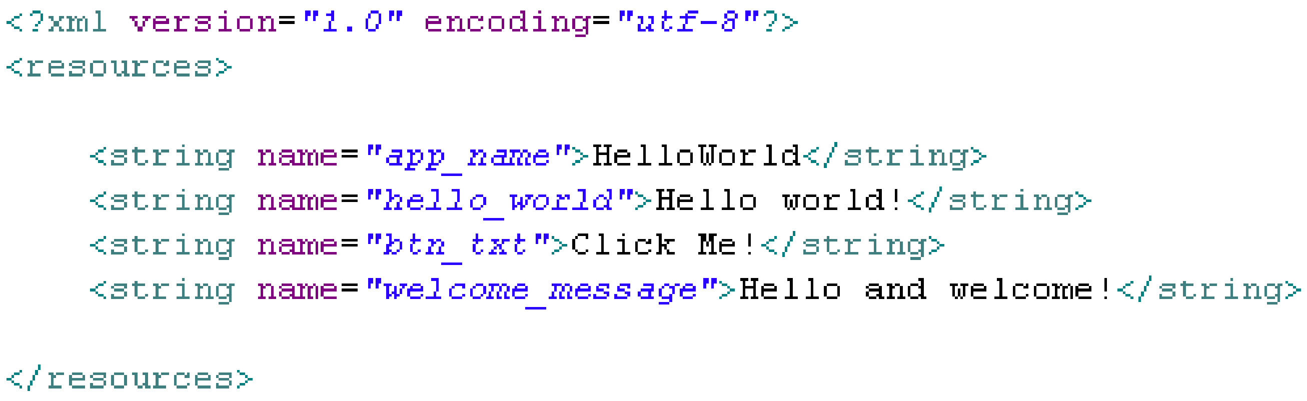


[I18N] Hardcoded string "Hello, World!", should use @string resource

Android Project Structure

Project Structure – strings.xml

Declaration example –



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

    <string name="app_name">HelloWorld</string>
    <string name="hello_world">Hello world!</string>
    <string name="btn_txt">Click Me!</string>
    <string name="welcome_message">Hello and welcome!</string>

</resources>
```

Android Project Structure

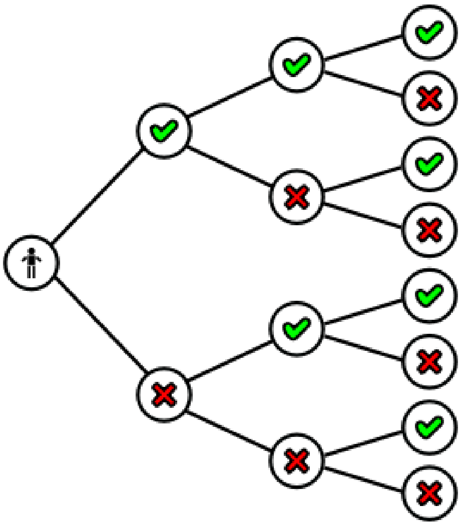
Project Structure – Layout files

- Used to describe different screens
- **Hierarchical**
- Like SWT form designer
- Two edit modes –
  - Raw XML
  - Graphical layout

Android Project Structure

Project Structure – Layout files

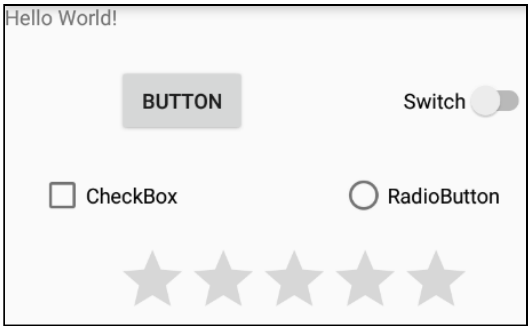
- Consists of layouts
  - RelativeLayout
  - GridLayout
  - LinearLayout
- Consists of widgets
  - TextView
  - ImageView
  - Button



Android Project Structure

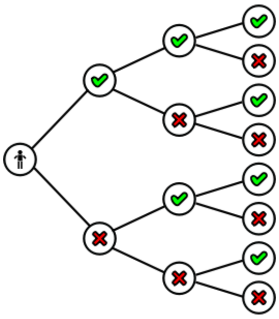
Layout files – Widgets –

- Elements visible to the user
- He interacts with them
- Many are build-in
- Custom ones can be added as well



Android Project Structure

Layout files – Layouts –

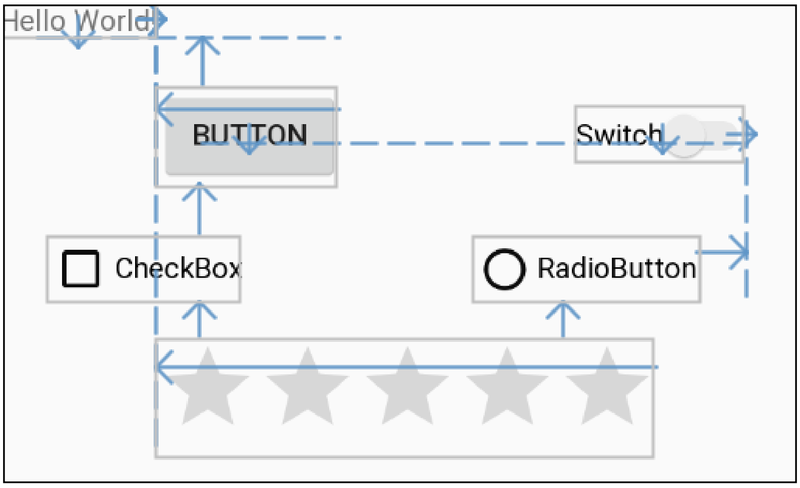


- Automatically aligns widgets
- Each layout by its own rules
- Helps support multiple screen resolutions
- Several can be hierarchically added

Android Project Structure

Layout files – Relative layout –

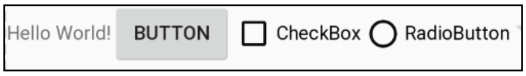
- Organizes widgets relatively to one another
  - Above
  - Below
  - Beginning of
  - End of



Android Project Structure

Layout files – Linear layout –

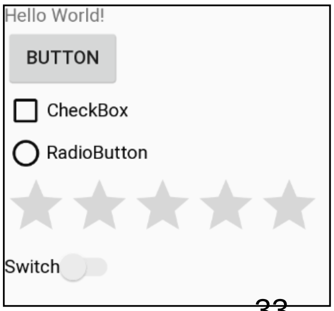
- Organizes widgets in a row / column
  - Horizontal: all in one row



```
<LinearLayout
    android:orientation="horizontal"
```

  - Vertical: all in one column

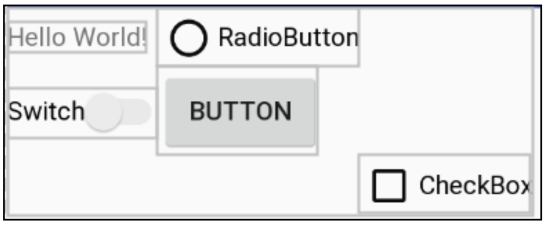
```
<LinearLayout
    android:orientation="vertical"
```



Android Project Structure

Layout files – Grid layout –

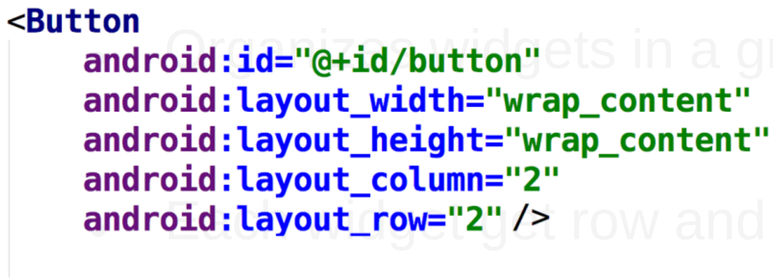
- Organizes widgets in a grid
- Each widget get row and column number
- Automatically aligns elements



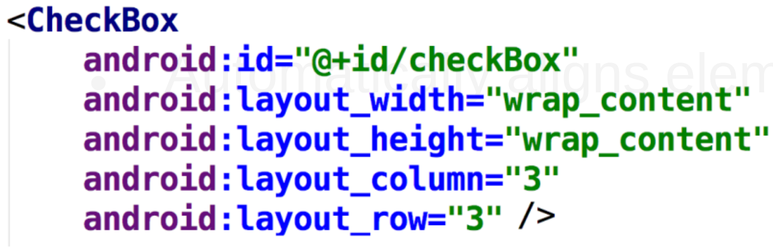
Android Project Structure

Layout files – Grid layout –

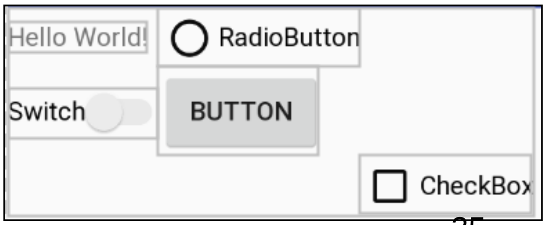
- Organizes widgets in a grid
- Each widget get row and column number
- Automatically aligns elements



```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="2"
    android:layout_row="2" />
```



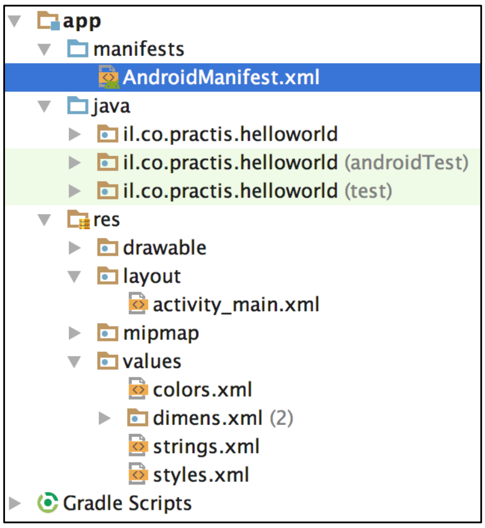
```
<CheckBox
    android:id="@+id/checkBox"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="3"
    android:layout_row="3" />
```



Android Project Structure

Project Structure – AndroidManifest.xml

- Many app configurations
- List of required permissions
- Supported SDK versions
- List of screens/activities



```
▼ app
  ▼ manifests
      AndroidManifest.xml
  ▼ java
    ▶ il.co.practis.helloworld
    ▶ il.co.practis.helloworld (androidTest)
    ▶ il.co.practis.helloworld (test)
  ▼ res
    ▶ drawable
    ▼ layout
        activity_main.xml
    ▶ mipmap
    ▼ values
        colors.xml
      ▶ dimens.xml (2)
        strings.xml
        styles.xml
▶ Gradle Scripts
```

XML

Extensible Markup Language (XML)

- Developer in 1997
- Developed by the World Wide Web Consortium
- Simplified version of SGML
- Eliminated many complex SGML features
- Used for inter-application data transfer

Basic XML Rules #1-

- Free to choose tags names:

```
<MyTag>


<MySecondTag>


<Tag111>
```

Basic XML Rules #2-

- Must close each tag:

```
<MyTag></MyTag>


<MySecondTag/>


<Tag111></Tag222> ✗
```

Basic XML Rules #3-

- Each tag might have a value:

```
<MyTag>value</MyTag>


<MySecondTag/>


<Tag111>value value value</Tag111>
```

Basic XML Rules #4-

- Each tag might have child tags:

```
<MyTag>
   <ChildTag>childValue</ChildTag>
</MyTag>



<MySecondTag/>
```

Basic XML Rules #5-

- Must close tags in the correct order:

```
<MyTag>
   <ChildTag>childValue</ChildTag>
</MyTag>



<MySecondTag>
   <ChildTag>
</MySecondTag>  ✗
   </ChildTag>
```

Basic XML Rules #6-

- Each tag might have values and attributes:

```
<MyTag attribute="attributeValue">value</MyTag>


<MySecondTag onlyAttribute="attributeValue"/>


<Tag111>onlyValue</Tag111>
```

Basic XML Rules #7-

- Some characters need to be escaped:

```
<MyTag>value1 & value2</MyTag>  ✗


<MyTag>value1 &amp; value2</MyTag>  ✓


"    &quot;
'    &apos;
<    &lt;
>    &gt;
&    &amp;
```

XML

Basic XML Rules #8-

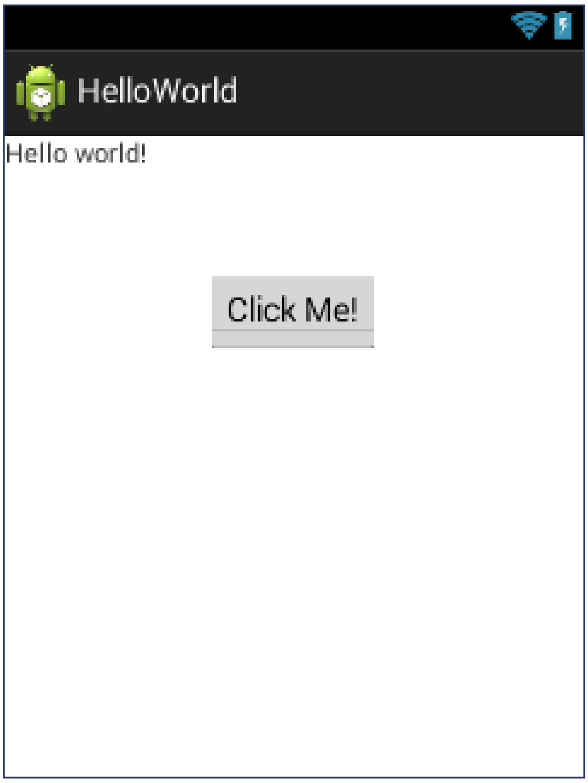
- Sometimes a fixed header is provided:

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

<MyTag/>



<MySecondTag/>



<Tag111/>
```

Enhancing First Android App

We will –

- Add a button
- Change button attributes
- Add some code



Enhancing First Android App

Change #1 – Change label text on button click

- Make sure label and button have Ids:
  - @+id/lblHello
  - @+id/btnClickMe

- Set value in button property: OnClick

- Add following code

Enhancing First Android App

Change #1 – Change label text on button click

```
public void DoClick(View v) {
    TextView label = (TextView)findViewById(R.id.lblHello);
    label.setText("Button was clicked!");
}
```

Enhancing First Android App

Change #1 – Better option – use strings.xml!

```
public void DoClick(View v) {
    TextView label = (TextView)findViewById(R.id.lblHello);
    label.setText(R.string.welcome_message);
}
```

Enhancing First Android App

Change #2 – add toast

```
public void DoClick(View v) {
    TextView label = (TextView)findViewById(R.id.lblHello);
    label.setText(R.string.welcome_message);

    // we will discuss Context in next classes
    Context context = getApplicationContext();
    Toast toast = Toast.makeText(context,
                                      "My First Toast!",
                                      Toast.LENGTH_SHORT);
    toast.show();
}
```

Enhancing First Android App

Change #2 – add toast

```
public void DoClick(View v) {
    TextView label = (TextView)findViewById(R.id.lblHello);
    label.setText(R.string.welcome_message);

    // we will discuss Context in next classes
    Context context = getApplicationContext();
    Toast toast = Toast.makeText(context,
                                          getString(R.string.helloMsg),
                                          Toast.LENGTH_SHORT);
    toast.show();
}
```

For Questions:
yoav@practis.co.il