1 Digital Innovation

1.1 Intelligent Interface of Human-Machine

Tutor: cch (diffusion)

1.2 Welcome to the WarField, Skywalkers!

1.3 Textbook

1. Kujira Flying Machine: Android Programming by HTML5 and JavaScript, 旗標出版公司 F2774. 2012
2. Chen WA: From HTML5, CSS3 and JavaScript to jQuery, PhoneGap and Android, 旗標出版公司 F2472. 2012
3. Rick Rogers, Learning Android Game Programming,, Pearson Edu Inc., 2012
4. Massimo Nardone and Vladimir Silva, Pro Android Games, Third Edition, 2015
5. Li Ning, Introduction to Kotlin Programming, Post&Telecom Publ, 2018. Reference book

---

1. [Github Resource](http://github.com/cchuang2009/2016-1)
2. Wei-Meng Lee, Beginning Android Tablet Application Development, Wrox, 2011
3. Wei-Meng Lee, Beginning Android Application Development, Wrox
4. Tim Kadlec, Implementing Responsive Design: Building sites for an anywhere, everywhere web, Riders, 2013

2 Websites

---

1. http://diffusion.cgu.edu.tw/android

2. https://github.com/cchuang2009/AndroidProgram
3. App Inventor 2 (AI2): http://www.appinventor.org/book2

2.1 2013

- Local Machine (../android/index.html)
- Students' Final Project (student-2013-2/www/index.html)

2.2 2014

- Dice with jsWaffle (jsWaffle/dice.ipynb): Training for animation effect, sound embedding .
- Google Map (googleMap/GoogleMap.ipynb): HTML5 Introduction, BMI jQuery-enhanced, Canvas, Json, Google Map eamples, ...
- Web Media API's (xylophone/2014-2-8.ipynb)
- App Inventor2: Google killer App (../android/lecture-html/appInventor2/index.html) - App Inventor2: example2 (AI2DEMO/1/lightbox/index.html)

2.3 2015

- App for Wear device
- ...

2.4 2016 ~2017

- VR, from cardboard SDK to gsr SDK

2.5 2018

- kotlin introduced
  - First Introduction (kotlinIntro.ipynb)

2.6 Course Description:

Android has been widely integrated and be found in hand-held devices, especially in smart phones since it first released at 2007. Java development environment on Linux system makes its capable of potential to run app's on light-weight cell phone. Its success directs the possible and right choice for another future development in cloud computing, "end-use".

```
 However, Java and Linux teaching and learning will be not foc
us of lecture but, insteadly, cirricula resources of Java  wil
l be prepared for self-learning and the Android App developed
by HTML5-compiant web techniques will be introduced in this co
urse.
```

Since the HTML-orientated programming is the focus of lectures, we will introduce basic HTML5 syntax, Javascript and CSS application by examples.

Briefly, the topics in this course include:

- HTML5

```
new features and functions for mobile devices;
```

- CSS3

```
Style refinement and reponsive web page design;
```

- Javascript

```
tools for implemnting interactivity, for instance, work
with Google map API's.
```

The course attendants will be encouraged to implement the group project by the knowledge and techniques learned in this lecture.

Make your idea come TRUE!

2.7 Grading

- Lecture Attendance: 20%
- Computer Pratices and Conceptual Quizzes: 60%
- Group projects AND Term Report: 20%

2.8 Office hour

- Tueday, p.m.12:00~13:00
- Wednesday, p.m.12:00~13:00

2.9 Survey

You have to follow the link to complete the survey:

https://docs.google.com/forms/d/1TXTlDn9IU5CGN6Qj6FCCC8RYLM4lvLiA7qqBI6OAz6c/viec=0&w=1 (https://docs.google.com/forms/d/1TXTlDn9IU5CGN6Qj6FCCC8RYLM4lvLiA7qqBI6OAz6c/viec=0&w=1)

2.10 Marilvn vos Savant

```
To acquire knowledge, one must study;
but to acquire wisdom, one must observe.
```

2.11 Lectures

1. Android Studio (defaulted);

2. HTML, Javascript;

3. Intel XDK

4. App Inventor 2;

2.12 First Lecture

2.13 Before we go

Setup the required environment

Where are they installed best? Anywhere but not within folder in name included Chinese characters! For Window users, "D:\AndroidStudio" is a good choice.

- Java SE Development Kit 8: http://www.oracle.com/technetwork/java/javase/downloads/ (http://www.oracle.com/technetwork/java/javase/downloads/), Android Studio require JDK-1.8;
- Android Studio +3.0 (http://developer.android.com/intl/zh-tw/sdk/index.html) (3.5 installed 2019/9/12), the best is installing the package with Android Development tools (ADT). To run or test kotlin staff requires install Android platform- 8.0+.
- Android SDK/NDK
  - SDK: http://developer.android.com/intl/zh-tw/sdk/index.html#Other (http://developer.android.com/intl/zh-tw/sdk/index.html#Other)

    ```
    which has been bundled with Andrio Studio.
    ```

  - NDK: http://developer.android.com/intl/zh-tw/tools/sdk/ndk/index.html (http://developer.android.com/intl/zh-tw/tools/sdk/ndk/index.html)

```
should be not required in general and now.
```

- Unity: https://unity3d.com (https://unity3d.com)

```
one of best software makes AR apps.
```

- Intel XDK: https://software.intel.com/en-us/intel-xdk (https://software.intel.com/en-us/intel-xdk)

```
install this if you want to share the power of HTML/Javascript codes with apps.
```

- Google Cardboard:
  - WebSite: https://www.google.com/get/cardboard/ (https://www.google.com/get/cardboard/)
  - Cardboard-java: https://github.com/googlesamples/cardboard-java (https://github.com/googlesamples/cardboard-java)
  - Cradboard-unity: https://github.com/googlesamples/cardboard-unityCardboad (https://github.com/googlesamples/cardboard-unityCardboad)

Further ...

3 Installation

- Download and extract **Java SE Development Kit (JDK 1.8u221, 2019/9/12) (http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html)**
- Install **Android Studio** (https://developer.android.com/studio/index.html), (3.5 suggested);

```
Note: Had better to choose the bundle set which should contain Android Develop Tools; if so, no need to install Android SDK and remember to update at the first startup.
```

- Download and extract **Android SDK/NDK** () into certain directory, says `$AndroidSDK` ,
  - extracted NDK put in the directory, `$AndroidSDK` /sdk/ndk-bundle;
- Install **Unity**.

Soon you should have a well-installed working environment for Android or others mobil systems.

3.1 Setting PATH

1. **(IMPORTANT)** Windows: supoose that `D:\Android` is the top directory at which softwares were installed as above; add the full path of the Android SDK tools and Android SDK platform-tools folders to the **PATH** variable, separated by a semi-colon. It should look something like this:

   ```
   D:\Android\tools;D:\Android\platform-tools
   ```

   Steps:

   - Hit the *Start* key on your Keyboard.

- Start typing the words *Environment Variables*.
- As you type, you'll see the choice to **Edit the system environment variables**. Choose it.
- In the Environment Variables window, select the **PATH** line item in the **User variables** for (your user name) section, then click the **Edit** button.
  Do not install within folder with name including any Chinese characters!

2. Mac/Linux: Suppose installed at `~/Android` , open a console and input:

   ```
   ~ User$ export PATH=$HOME/Android/tools:$HOME/Android/platform-tools:$PATH
   ```

   or add the last input to the `.bash_profile` .

3.2 First App

App's Object, create a new Android Studio kotlin project, at which includes one text display (edittext), two buttons, one for displaying text in edittext, the other for toasting the text directly.

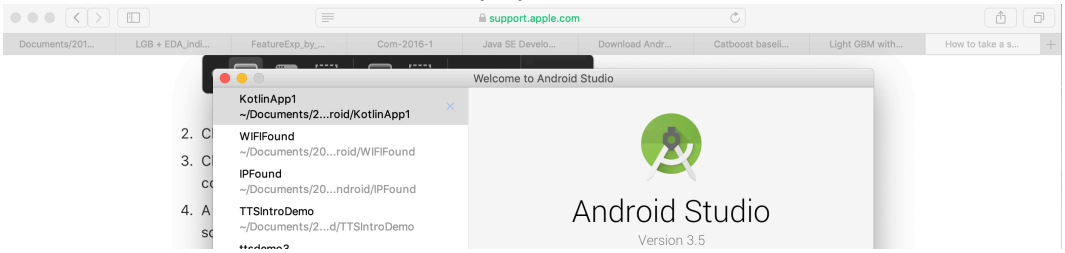
3.3 Outline

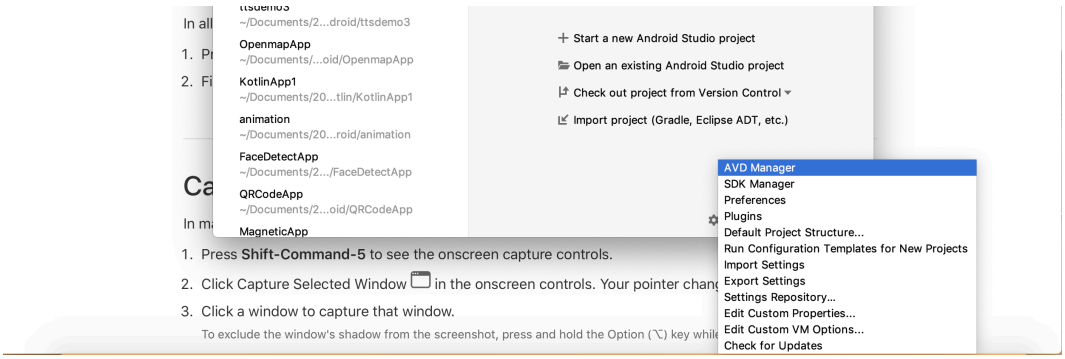
1. create project with the following settings:

   ```
   Name
   ---
   MyApplicationKotlin

   Package Name
   ---
   com.kotlin.first.myapplicationkotlin

   ...

   Language
   ---
   Kotlin

   Minimum API level
   ---
   API 19: Android 4.4 (KitKat)
   ```

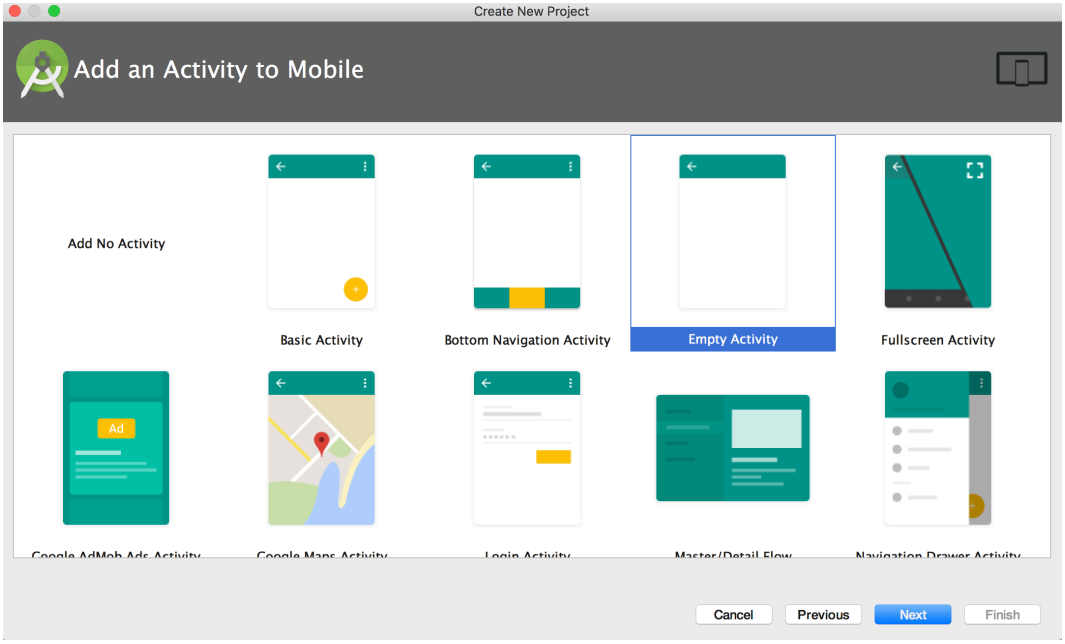
2. choose `Empty Activity` and create.

3. Launch Android Studio and create new project:

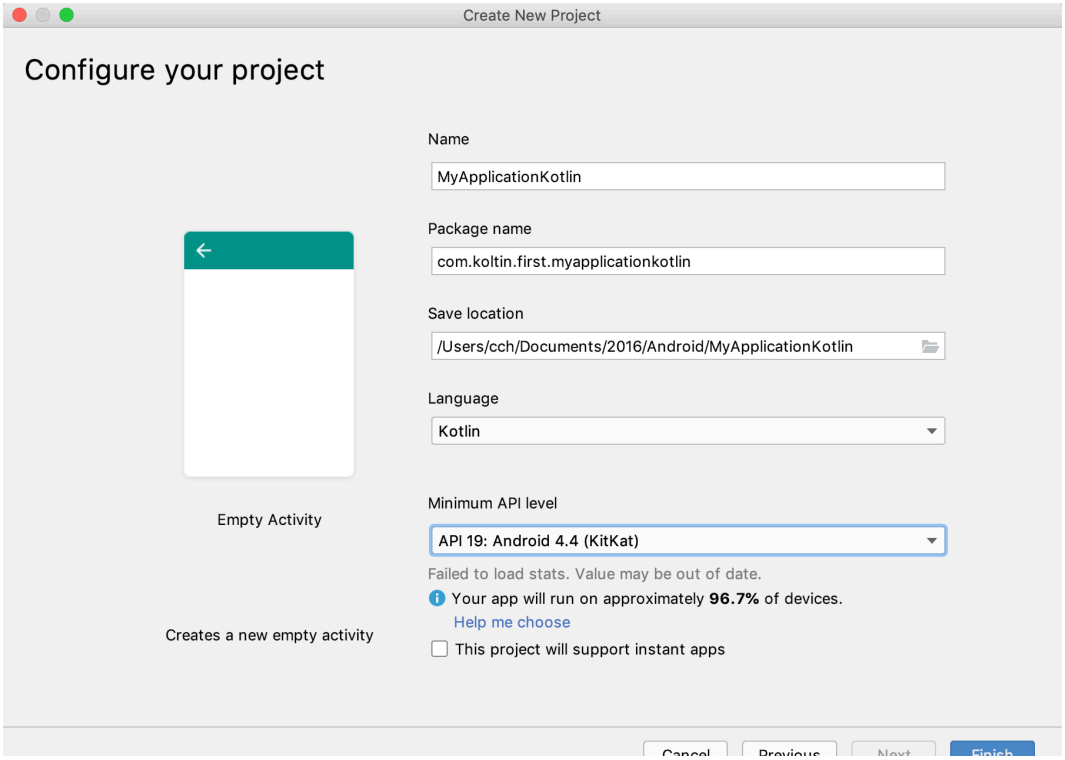




- Type of Activities, blank activity

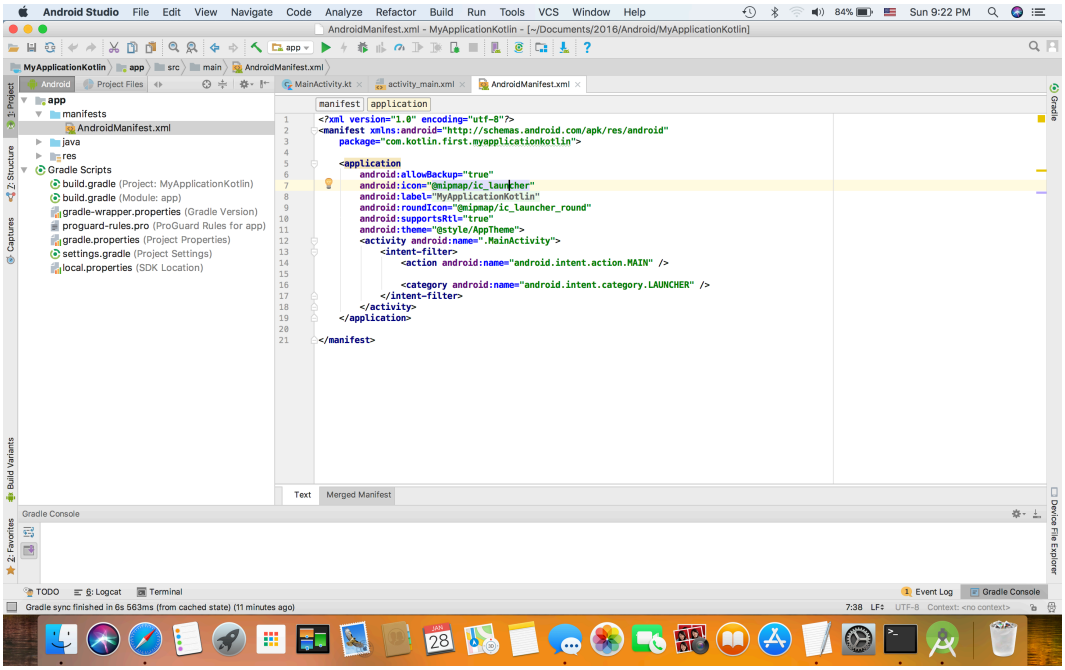


- Create New Project: the first project creation, it had better to enable option `[include kotlin support]` as belows which should install kolin-related packages automatically:

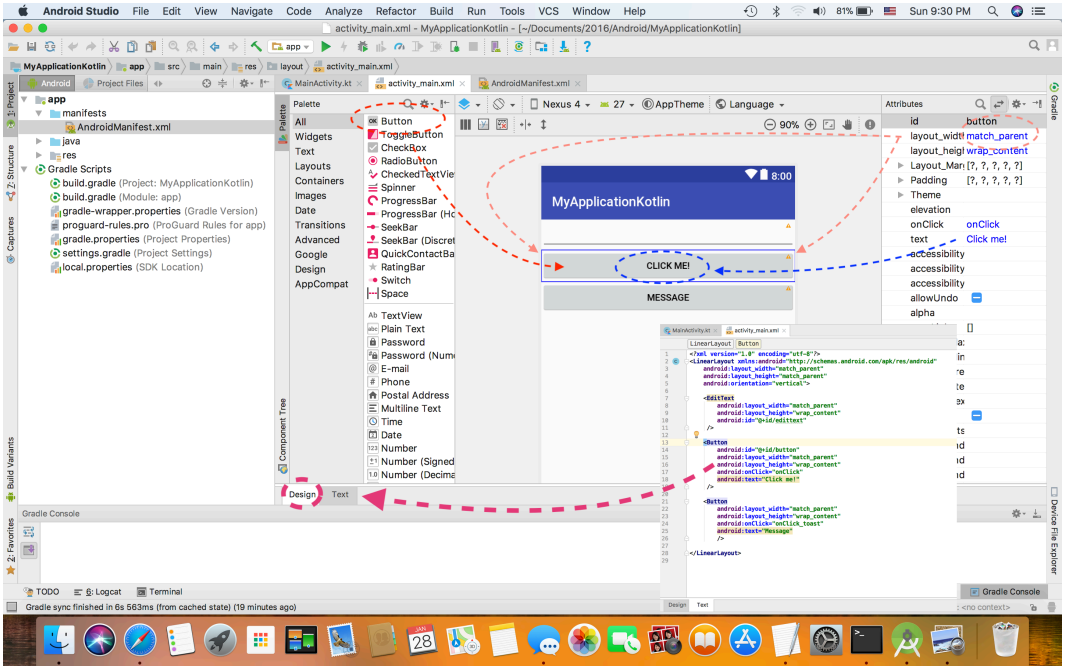


```
- Main Menu of Android Studio
<img src="imgs/1/AndroidStudio-7.png" width=%80 />
```

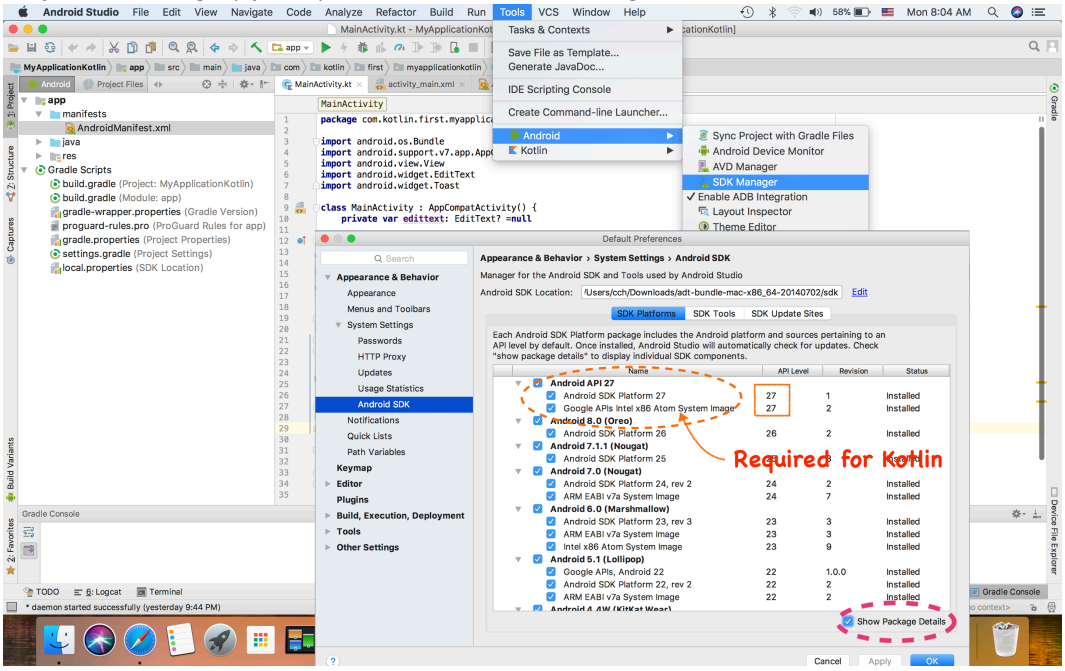
- BluePrint of Project, AndroidManifest.xml; where you can find a). the app icon (arrow at center) b) if any error (displayed in right red dash shown in the right border of table) , reference the last snapshot in this section:

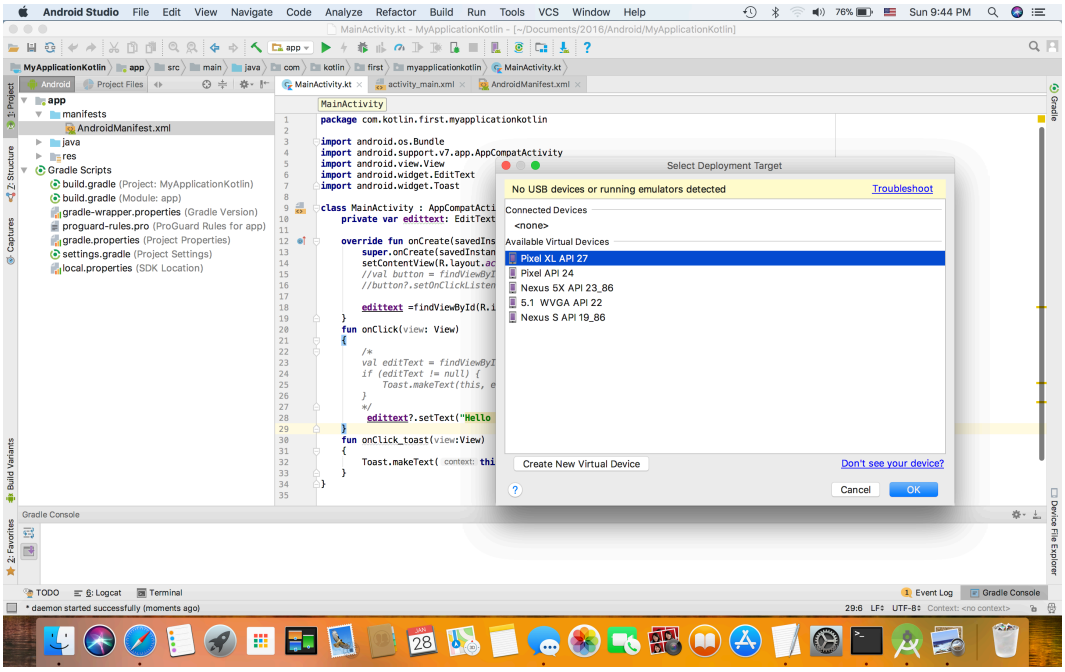


- Layout of App, graphical and text modes



- App Test via Android Virtual Machine (AVM); running Kotlin app requires install Android-8+ platform image (api27+), here intel-x86 atom image installed,



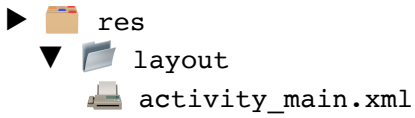


4 Coding Part

UI's design and code are seperated but are highly related.

4.1 UI

1. **activiti_main.xml**, define all the visualized units in app,



- change layout to LinearLayout
- create one `EditText` and two `Button` 's

2. Code

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/ap
k/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical">

 <EditText
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:id="@+id/edittext"
  />

 <Button
     android:id="@+id/button"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:onClick="onClick"
     android:text="Click me!"
  />

 <Button
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:onClick="onClick_toast"
     android:text="Message"
     />
</LinearLayout>
```

4.2 Kotlin Part

The way of app comes as follows:

```
package com.kotlin.first.myapplicationkotlin

// standard imports
import android.app.Activity
import android.os.Bundle
import android.view.View
```

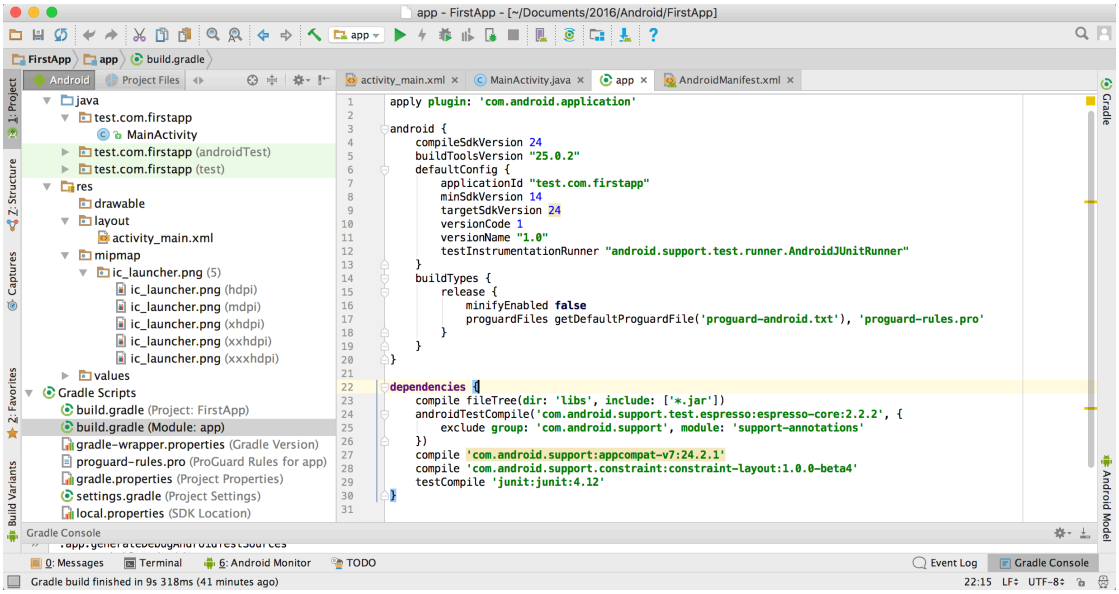
```
import android.widget.EditText
import android.widget.Toast

class MainActivity : Activity() {
    private var edittext: EditText? =null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // related to UI's
        setContentView(R.layout.activity_main)
        // define var with espect to UI
        edittext =findViewById(R.id.edittext)
    }

    // define the functions waiting service while button's, in
UI's Buttons', clicked,
    fun onClick(view: View)
    {
         edittext?.setText("Hello Kotlin!")
    }
    fun onClick_toast(view:View)
    {
        Toast.makeText(this,edittext?.text, Toast.LENGTH_LONG)
.show()
    }
}
```

In brief, display the UI and wait for button clicking for service.

4.3 Note

Should Android Studio run *off-line* possibly? Theoretically, yes. However, running online at the project first created is better since it could download the absent packages automatically if has. The error due to such kind of failure to create project can be removed by deleting unnecessary dependencies setting in grandle.build(app):

```
    ...

androidTestCompile('com.android.support.test.espresso:espresso
-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support
-annotations'
    })
    ...
```



4.4 Image Staff

Put the image in the the layout, add ImageView in activity_main.xml:

```
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:srcCompat="@mipmap/flower"
            android:layout_marginStart="24dp"
            android:id="@+id/flower" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_toRightOf="@+id/flower"
            android:text="How wonderful the flower of cactus is al
    ive!"
            android:layout_marginLeft="5dp"
            android:id="@+id/textView" />
```

the introduced text was put next to the right site of Image.

4.5 Self Train

Make your own message (card).

In [ ]:

- How do you work with data (the simplest way)
  - Ms Office Suit
  - T_EX
  - ...

  Do you satisfy?

```
Nothing is no more popular than HTML!
```

5 HTML

A popular format of data suit for any device and anybody!

5.1 Template

```
<html>
   <meta name="viewport" content="user-scalable=no, initial-sc
ale=1, maximum-scale=1, minimum-scale=1, width=device-width, h
eight=device-height, target-densitydpi=device-dpi" />
   <head>Page Title</head>
   <script type="text/javascript">...</script>
   <style type="text/css">...</style>
   <body>
      I'm here.
   </body>
</html>
```

- (verbatim mode) I'm here,
- <big>I'm here</big>, (by <big>... </big> tag)
- *I'm here*, (by <i>... <i> tag)
- I'm here, (by <font color="red">... </color> tag)

In [7]:

```
from IPython.display import HTML
```

In [7]:
```
from IPython.display import HTML
HTML("<iframe src=1/here.html width=50% height=20%></iframe>")
```

Out[7]:

5.2 Code

```
<html>
   <meta name="viewport" content="user-scalable=no, initial-sc
ale=1,
     maximum-scale=1, minimum-scale=1,
     width=device-width, height=device-height" />
   <body>
      I'm here.<br>
       <font color="red"> I'm here.</font><br>
       <code style="background-color:red;color:white;"> I'm he
re.</code><br>
   </body>
</html>
```

5.3 Note

The *meta* tag with "viewport" used above is initially introduced for recent mobile devices device but is also popular in general HTML codes.

5.4 HTML Programming Environment

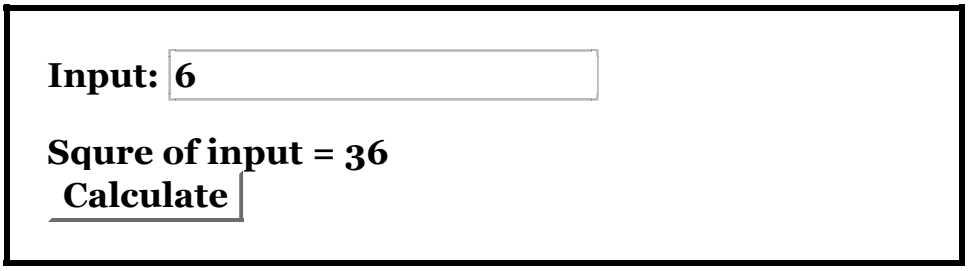
- HTML, CSS and Javascript
- Development Tool: Seamonkey,Gimp

6 Second Lecture (Interactivity)

Input a number, X, and calculate its square power, X^2:

In [2]:
```
from IPython.display import HTML
HTML("<iframe src=1/Square.html width=50% height=50%></iframe>")
```

Out[2]:



Main HTML Part

define a form, waiting for data input to work:

```
<form action="#">
  Input:
  <input id="real" type="numeric" name="real"
         min="0" max="100" step="1" value="6">
  <br>
     Squre of input = <b id="boldStuff">36</b>
  <br>
  <input value="Calculate"
     onclick="calcSquare(this.form.real.value)"
     type="button">
</form>
```

Javascript Function

Calculate the square of Input:

```
<script type="text/javascript">
    function calcSquare(real)
    {
```

```
            var result=real*real;
            document.getElementById('boldStuff').innerHTML = res
ult;
        }
    </script>
```

or

```
    <script type="text/javascript">
        function calcSquare()
        {
          var realval=document.getElementById('real').value;
          var result=realval*realval;
          document.getElementById('boldStuff').innerHTML = res
ult;
        }
    </script>
```

In [2]:
```
from IPython.display import HTML
```

In [3]:
```
HTMLcode="""
<form action="#">
   Input:
   <input id="real" type="numeric" name="real" min="0" max="100" ste
   <br>
   <p>
   Squre of input = <b id="boldStuff">36</b>
   <br>
   <input value="Calculate" onclick="calcSquare(this.form.real.value
</form>
"""
JScode="""
<script>
      function calcSquare(input)
      {
        // deal the values of variables passed from HTML codes
        var result=input*input;
        document.getElementById('boldStuff').innerHTML = result;
      }
</script>
"""
```

In [4]:
```
HTML(JScode+HTMLcode)
```

Out[4]:

Input: 6

Squre of input = **36**

Calculate

Note

- above result enhanced by some CSS

```
   <div style="font-family: Georgia, serif;background-color:ga
insboro; border:solid black; width:300px; padding:20px;">
```

- same effect by standard CSS syntax

```
   <style type="text/css">
      form,input {
            font-family:  Georgia, serif;
      }
   </style>
```

In [5]:

```
HTMLcode="""
<div style="font-family: Georgia, serif;background-color:gainsboro;
<form action="#">
   Input:
   <input id="real" type="numeric" name="real" min="0" max="100" ste
   <br>
   <p>
   Squre of input = <b id="boldStuff">36</b>
   <br>
   <input value="Calculate" onclick="calcSquare(this.form.real.value
</form></div>
"""
JScode="""
<script>
      function calcSquare(input)
      {
        // deal the values of variables passed from HTML codes
        var result=input*input;
        var kernel = IPython.notebook.kernel;
        var callbacks = {'output' : handle_output};
        document.getElementById('boldStuff').innerHTML = result;
      }
</script>
"""
```

In [6]:

```
HTML(JScode+HTMLcode)
```

Out[6]:

Input: 6

Squre of input = **36**

Calculate

Code (HTML+CSS+Javascript)

```
<html>
   <meta name="viewport" content="user-scalable=no, initial-sc
ale=1, maximum-scale=1, minimum-scale=1, width=device-width, h
eight=device-height" />
   <style type="text/css">
      form {
            font-family:  Georgia, serif;
            background-color:gainsboro;
            border:solid black;
            width:300px;
            padding:20px;
      }
      input {
            font-family:  Georgia, serif;
      }
   </style>

   <body>
      <form action="#">
     Input:
     <input id="real" type="numeric" name="real"
            min="0" max="100" step="1" value="6">
     <p />
        Squre of input = <b id="boldStuff">36</b>
     <br>
     <input value="Calculate"
        onclick="calcSquare(this.form.real.value)"
        type="button">
   </form>
   <script type="text/javascript">
        function calcSquare(real)
        {
          var result=real*real;
          document.getElementById('boldStuff').innerHTML = res
ult;
        }
    </script>
   </body>
</html>
```

Computer Practice, App Packing

1. create new project:

```
Application Name:  first
Module Name:       first
Package Name:      android.com
```

2. Edit your application's res/ `activity_main.xml` file:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/re
s/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="android.com.first.MainActivity">

    <WebView android:id="@+id/webview1"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >
    </WebView>
</LinearLayout>
```

a). change layout to LinearLayout b). create a webview, named "webview1", for HTML output.

3*. *modify AndroidManifest.xml by adding the following permissions after* *manifest tag:

```
    <uses-permission android:name="android.permission.INTERNET
" />
```

4. Add HTML codes

copy all the files into the directory $Project/app/src/main/assets, especially the html, index.html;

5. Modify Main Java code:

```
package android.com.first

import android.support.v7.app.AppCompatActivity;
//import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebView;

public class MainActivity extends AppCompatActivity {
```

```
    private WebView mwebview;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mwebview = (WebView) findViewById(R.id.webview1);
        mwebview.getSettings().setJavaScriptEnabled(true);
        mwebview.loadUrl("file:///android_asset/index.html");
    }

}
```

And last, run the app by emulator or on your mobile device.

- Complete all the steps of the example.

Next Lecture

How to make an App with the HTML contents:

- Hello World (ipynb) (1/index.ipynb)
- Hello World (HTML) (1/index.html)

App's Internationalization (i18n)

1. Startup **Android Studio** (or **Eclipse**) and create a project, named "test"; - Structure of projects, *src,res/[layout,values]*; - Setup "run configuration", create a new AVD, and test it; - copy whole the directory, res/values, and rename as "res/values-zh-rTW" for **Traditional Chinese Language**; - modify **string.xml** in the directory ceated by last step into Traditional Chinese; - change language of AVD to test the locale. - use mobile device as the test system.

Note

1. Change layout Orientation:
   - Switch layout orientation portrait/landscape backwards [AVD]
   - modify AndroidManifest.xml as follows:

```
<activity
   android:name=".ActivityName"
   android:screenOrientation="landscape" >
```

2. Remove title bar: open `res/values/style.xml` and change `.DarkActionBar` for `.NoActionBar` .
3. prevant to restart activity while orientation was changed,

```
<activity
```

```
android:name= .ActivityName
android:screenOrientation="landscape"
android:configChanges="orientation|screenSize" >
```

Kotlin Android

- start from Android Studio -3,* Convert Java to Kotlin manually:

---

1. [Pref > Plugins > search “Kotlin”]

   and install it;
2. restart Android Studio, and click `[shift]` -key twice to activae help windows and input (`"convert to java ..."`) to convert java to kotlin automatically and this step should ask to set up kotlin's configuration too.

   - Windows: [ctrl + alt + shift +k]
   - Mac OS: [shift+option+command+k]
   - menu: choose [project ▶ app ▶ java], and Main munu [code] ▷ [convert java to kotlin]
   - Android should remind to synchronize the project to work.
3. try AVD again.

Note

Assignment this week, News about HTC and Google deal (m.appledaily.tw/realtimenews/article/hot/20170921/1208209),

1. why does the HTC depressed so much recently?
2. Compare all the specification of the flahship handsets of the market leaders in the mobile-device market.
3. Your opion in such deal.

In [2]:
```
%%bash

jupyter nbconvert Com-2016-1.ipynb
```

```
[NbConvertApp] Converting notebook Com-2016-1.ipynb to html
[NbConvertApp] Writing 285435 bytes to Com-2016-1.html
```

Android Programming

Dept: Sequence Number:

Dept: Sequence Number:
Name:

Week	Check Point	Yes/No	Date
1	Android Studio,JDK-8, Android platform 28 with atom image installation	/	
2	Hello World App		
3			
4			
5			
6			
7			
8			
9			

10

11

12

13

14

15

Fin

In [ ]: