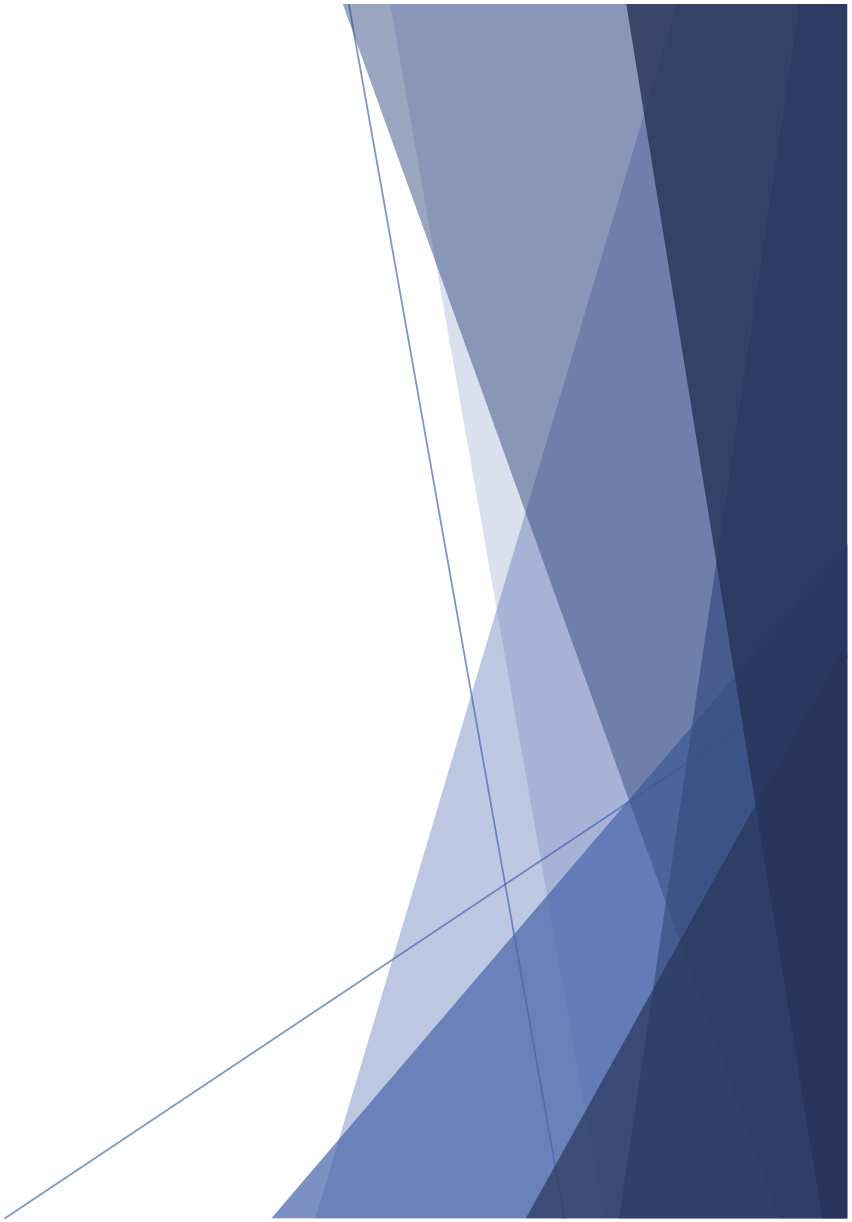
Android Programming

기본 위젯과 레이아웃

Android Programming

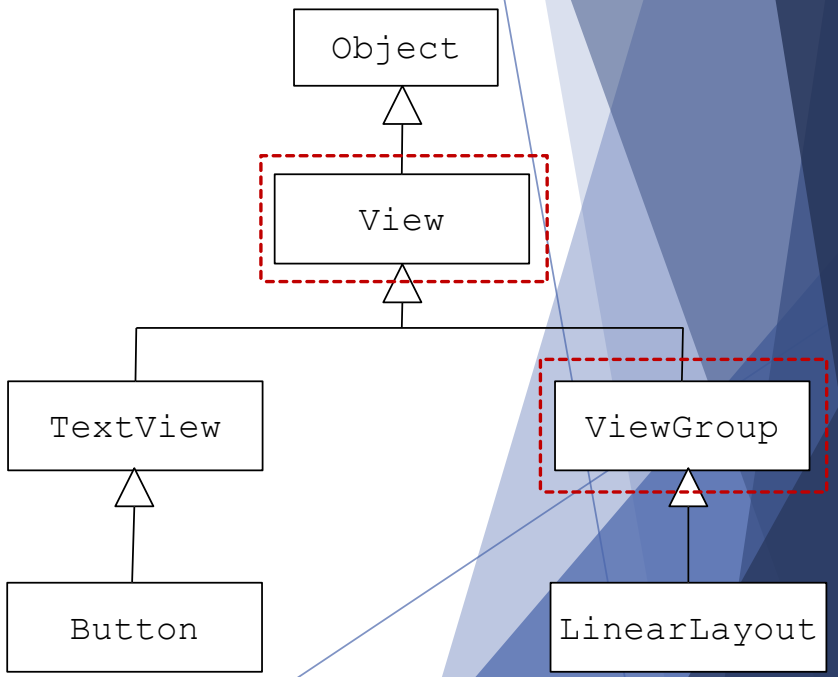
View와 ViewGroup



View와 ViewGroup

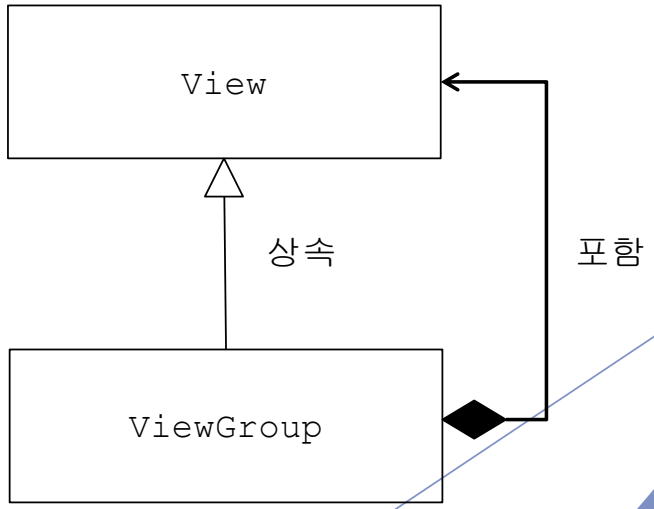
: View

- View 클래스는 안드로이드 UI를 구성하는 기본 단위
- Widget: View 중에서 일반적으로 컨트롤 역할을 하고 있는 것
- 안드로이드에서 제공하는 대부분의 UI 요소들은 Widget이라 불리는 View 클래스의 서브 클래스이다.
- 사용자와 상호작용하여 다양한 이벤트를 수신하는 역할을 수행한다
- 화면을 구성하는 요소이므로 정의된 클래스 수가 많다
- 대표적인 예는 Button, TextView 등이다.



View와 ViewGroup
: ViewGroup

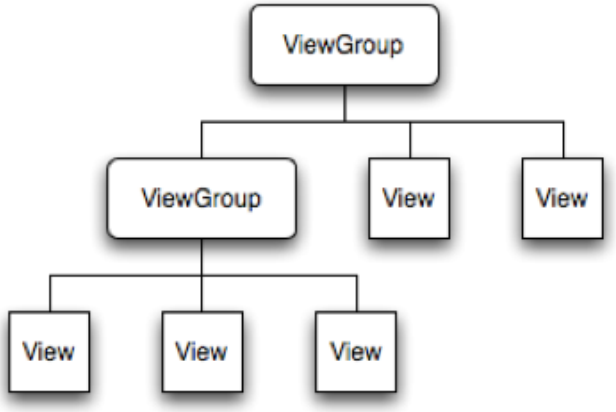
- View 들을 여러 개 포함하고 있는 것이 ViewGroup이다
- ViewGroup도 하나의 View이다.
- ViewGroup 중에서 View들을 포함하고 있으면서 배치하는 역할을 하는 것을 Layout이라 한다
- ViewGroup의 대표적인 예는 Layout



View와 ViewGroup

: View와 ViewGroup

- ViewGroup 클래스는 여러 개의 뷰를 구조화하는 단위
- 하나의 ViewGroup에는 여러 View 와 또 다른 ViewGroup으로 구성된다
- 즉, 뷰를 계층화하여 복잡하고 다양한 화면 구조를 표현할 수 있다



< 뷰 계층 (View Hierarchy) >

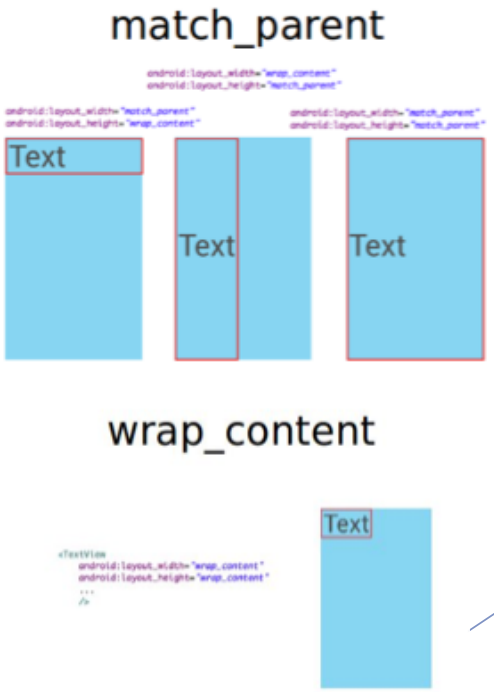
View와 ViewGroup
: View의 공통 속성

- Id
  - 뷰의 고유한 이름
  - XML 레이아웃에 정의된 뷰들을 참조할 때 사용
    - @+id/{아이디}
    - @은 아이디를 리소스(R.java)에서 참조하거나 정의한다는 의미
    - +는 정의할 때만 붙이고, 참조할 때는 붙이지 않는다
    - / 뒤에 개발자가 지정한 아이디가 오는데, 가급적 직관적 이름을 붙이도록 한다
  - XML 레이아웃에 정의된 뷰들은 화면이 보이기 이전에 객체로 만들어진다. 코드에서는 XML에 정의된 ID를 참고로, 이 객체를 찾아내어 참조한다
  - 반드시 지정해야 하는 것은 아니다. 코드에서 참조할 필요가 없는 위젯은 id를 생략하는 경우가 많다

View와 ViewGroup

: View의 공통 속성

- layout_width, layout_height
  - 뷰의 폭과 높이를 지정
  - 다음과 같은 값을 가진다
    - match_parent(fill_parent) : 부모의 크기에 맞춘다
    - wrap_content : 내부 콘텐츠의 크기만큼 채운다
    - 상수 : 지정한 크기에 맞춘다
    - match_constraint : ConstraintLayout의 제약에 맞춘다

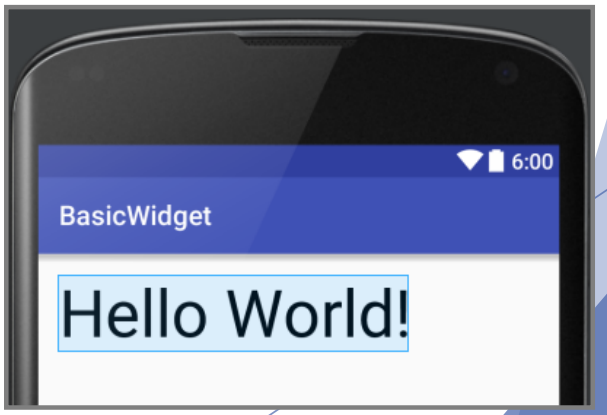


View와 ViewGroup

: View의 공통 속성

- [실습] BasicWidget
  - activity_main.xml에 TextView를 하나 추가하고 layout_width, layout_height 속성 값에 따라 위젯의 변화되는 크기를 확인해 봅시다.
  - 폭과 높이를 모두 wrap_content, match_parent, match_constraint로 변경해 보면서 변화되는 위젯의 크기와 영역을 확인해 봅시다.
  - 폭과 높이를 정수 값으로 설정하고 변화되는 위젯의 크기와 영역을 확인해 봅시다.

```
<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
```



View와 ViewGroup

: [참고] 크기를 지정하는 단위

- px(픽셀), in(인치), mm(밀리미터), pt(포인트), dp(dip: 해상도에 독립적 단위), sp(sip: 폰트 가변 크기)
  - 픽셀은 직관적이며 결과를 보기 편하다
  - 하지만 절대적 단위여서 디바이스 해상도에 따라 실제 크기가 다르게 표시될 가능성 높음
  - in, mm, pt, **dp, sp**와 같은 논리 단위를 사용 권장

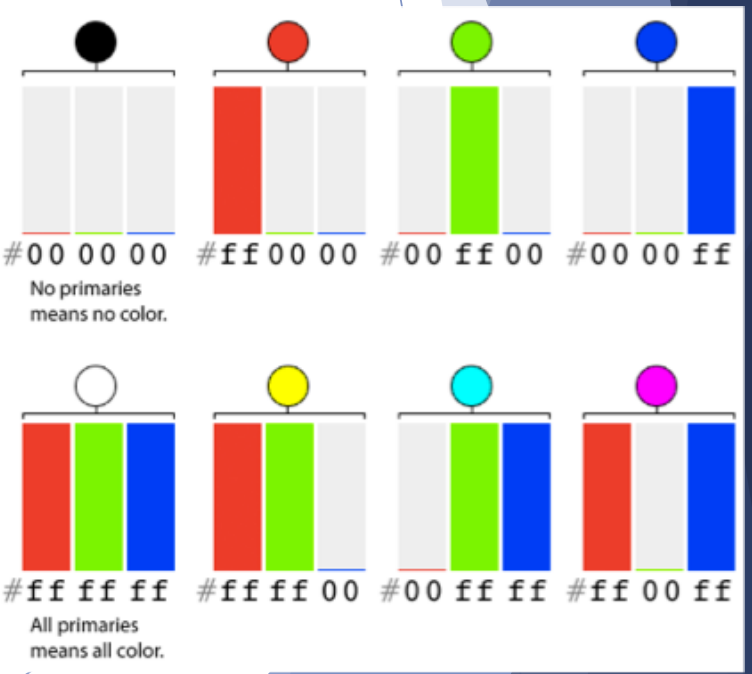
Dimension	Description	Units / Physical Inch	Density Independent	Same Physical Size on Every Screen
px	Pixels	Varies	No	No
in	Inches	1	Yes	Yes
mm	Millimeters	25.4	Yes	Yes
pt	Points	72	Yes	Yes
dp	Density independent pixels	~160	Yes	No
sp	Scale independent pixels	~160	Yes	No

View와 ViewGroup

: View의 공통 속성

[참고] Adobe Color CC: https://color.adobe.com

- background
  - 뷰의 배경을 어떻게 채울 것인지 지정
  - 별다른 지정이 없으면 디폴트 배경이 그려진다
  - 여러 객체로 배경을 지정할 수 있는데, 가장 흔한 형태는 색상
    - **이미지**, **selector**, **drawable** 등 복잡한 객체도 사용할 수 있다
  - 16진 컬러의 형식
    - 기본형 #RRGGBB : 각 색상은 0~255 (0x00 ~ 0xFF)까지의 값
    - 축약형 #RGB : 예) #FF0000 -> #F00, #FFAACC -> #FAC
    - 알파값: 투명도
      - #AARRGGBB
      - #ARGB
- [실습] BasicWidget
  - 배경색을 지정해 봅시다



View와 ViewGroup

: View의 공통 속성

- padding, margin
  - padding : 뷰 내부에 콘텐츠가 위치하는 간격
    - 4방향 지정 가능
  - margin : 뷰와 다른 뷰, 또는 부모 뷰와의 간격
    - 4방향 지정 가능

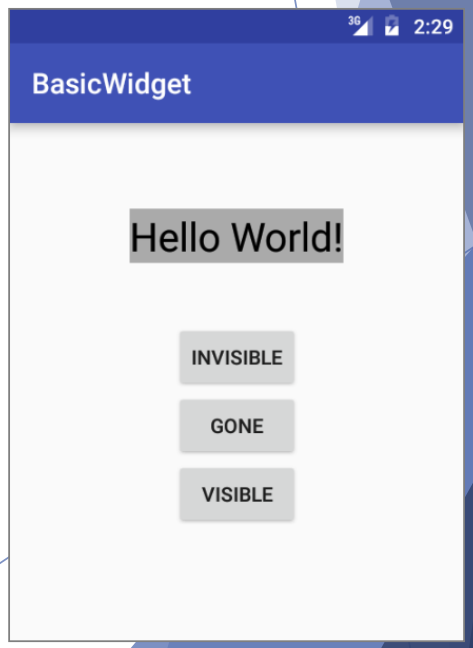
- [실습] BasicWidget
  - padding, margin 값을 4방향에 대해 각각 지정해 보면서, 위젯의 위치와 영역의 변화를 확인해 봅시다

View와 ViewGroup

: View의 공통 속성

- visibility
  - 뷰를 보일 것인가 아닌가를 결정
  - 일반적으로 디자인 타임에 숨겨두고, 런타임시 필요할 때 보여주고 사라지게 하는 효과를 줄 수 있다
  - 다음 세 가지 값을 갖는다
    - visible : 보이는 상태
    - invisible : 숨겨진 상태 (자리는 차지함)
    - gone : 숨겨진 상태 (자리도 차지 안함)

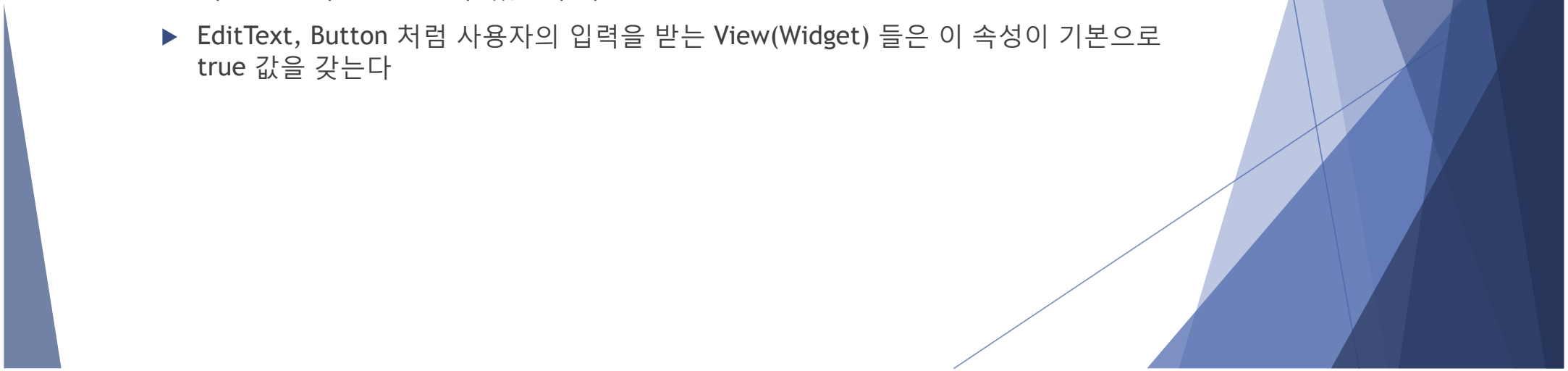
- [실습] BasicWidget
  - 3가지 visibility 속성을 설정하며 변화되는 모습을 확인합니다.
  - 버튼을 3개 추가하여 버튼 Click 이벤트에서 세 가지 속성을 변화시켜 봅시다
    - 예) view.setVisibility(View.Gone)



View와 ViewGroup

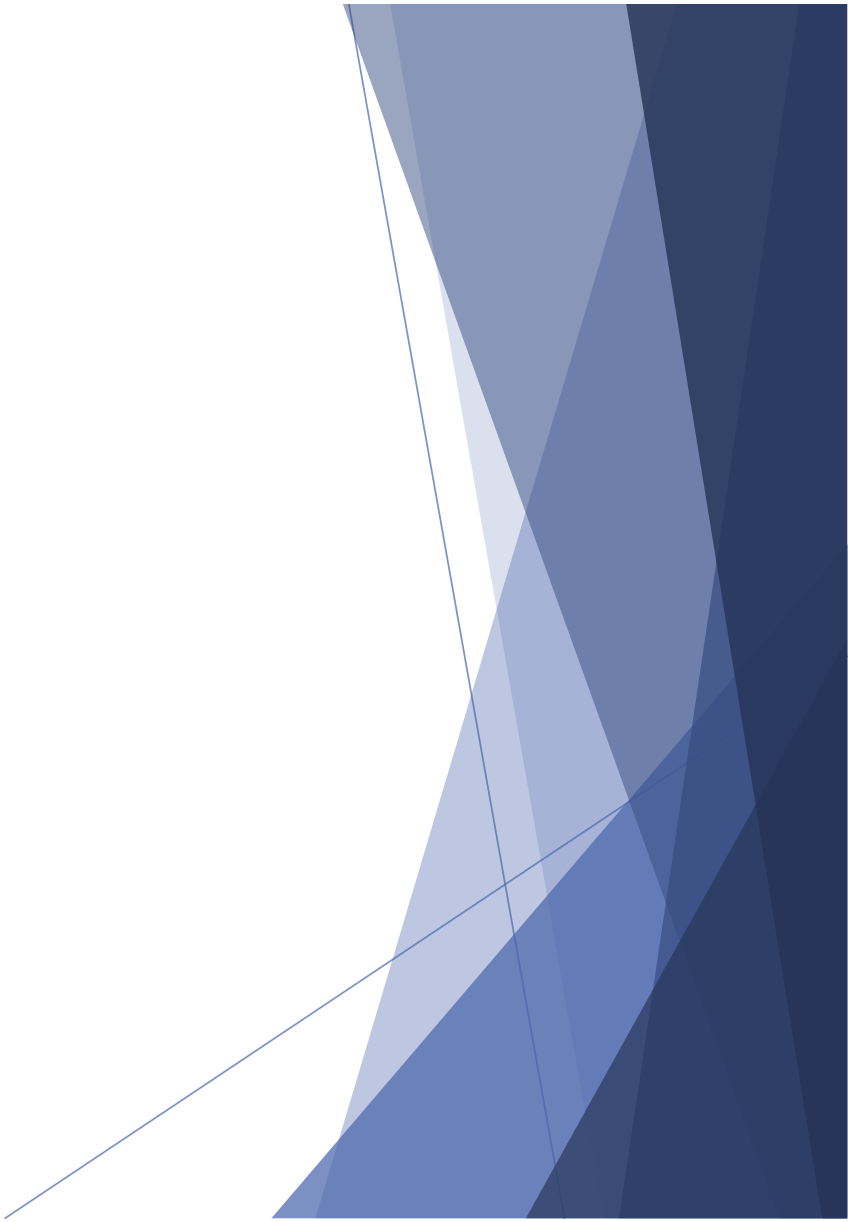
: View의 공통 속성

- clickable, longClickable
  - click 또는 long click 이벤트를 받을지 설정
- focusable
  - 키보드 포커스를 받을 수 있는지 지정
  - EditText, Button 처럼 사용자의 입력을 받는 View(Widget) 들은 이 속성이 기본으로 true 값을 갖는다



Android Basic Widgets

TextView



Android Basic Widgets

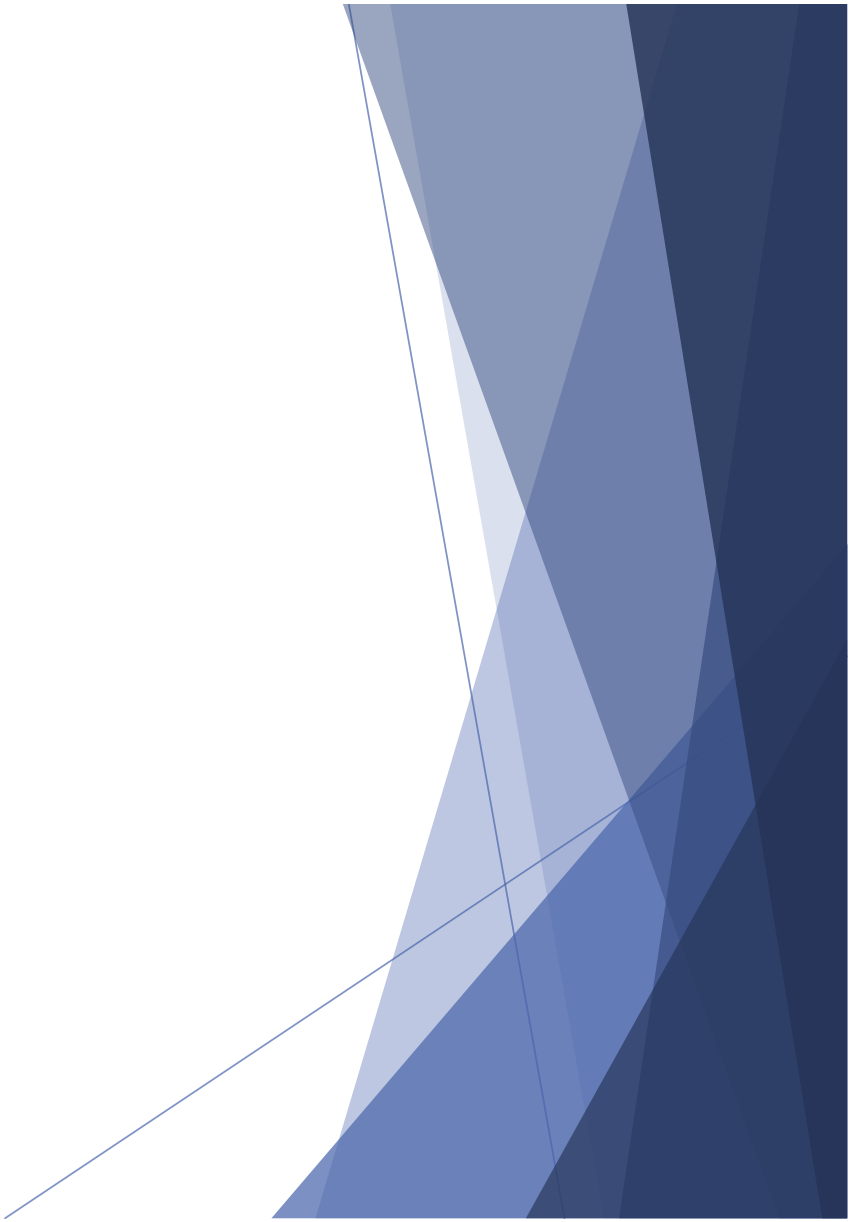
: TextView

- 화면에 텍스트를 표시하는 가장 기본적인 위젯
- 단순하지만 많은 기능과 속성을 제공
- 고정된 텍스트를 보여주고, 다른 위젯의 제목을 표시할 때 사용된다

- 주요 속성
  - text
    - 출력할 문자열을 지정
    - 형식
      - “문자열” : 문자열 리터럴로 대입
      - @string/id : string.xml에 정의해 놓은 문자열을 지정
    - 일관된 메시지 관리 또는 다국어 버전 작성을 위해서 리터를 직접 대입보다 string.xml 에 문자열을 정의해 id를 지정하는 것을 권장

Android Basic Widgets
: TextView

- 주요 속성
  - textColor
    - 문자열의 색상을 지정
  - textSize
    - 텍스트의 폰트 크기를 지정
    - 정밀한 크기 지정을 위해 실수 타입으로 지정할 수 있다
    - 숫자 뒤에 단위 (sp, dp, px, in, mm 등)를 지정한다
    - 일반적으로 폰트 크기 가변 sp 단위를 쓰는 것을 추천
  - textStyle
    - 폰트의 속성(normal, bold, italic)을 지정
    - 두 개 이상의 상수값 지정도 가능 예) bold|italic (o), bold | italic (x)



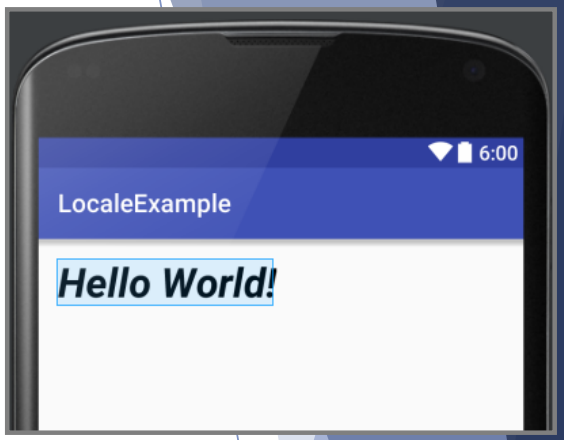
Android Basic Widgets

: TextView

- 주요 속성
  - typeface
    - 글꼴의 모양을 지정
    - 모바일 환경은 내장 폰트 수의 제약이 있으므로, 데스크탑처럼 다양한 글꼴을 제공하지는 못한다
    - 보통 normal, sans, serif, monospace 중 하나로 글꼴을 지정할 수 있다
  - width, height
    - 폭과 높이를 크기 값과 단위로 지정한다
    - 이 속성은 잘 사용하지 않는다
    - 절대적 크기뿐 아니라 상대적 크기도 지정할 수 있는 View의 layout_width, layout_height 속성을 더 선호하고 이 방식이 범용성이 훨씬 높다

Android Basic Widgets

: TextView



[실습] LocaleExample

안드로이드의 로케일 변화에 따라 TextView 내용이 변화 되도록 해 보자.
다국어 버전의 앱을 작성할 때 strings.xml 작성 요령을 알아본다.

1) 액티비티가 하나인 프로젝트 LocaleExample 를 만든다.

2) activity_main.xml 레이아웃에 TextView 위젯을 추가한다.

3) textColor, textSize, textStyle 등을 직접 값을 세팅한다.

4) text 속성에는 "@string/message" 로 설정할 수 있도록 string.xml에
"Hello World!" 문자열을 정의하자.

5) /res 밑에 /values-ko 폴더를 생성 한다 (프로젝트 도구 창에 Android로 보면 보이지 않으니 Project 보기로 본다)

6) /res/values-ko 폴더에 strings.xml 를 복사하고 다음과 같이 수정 한다.

```
<resources>
    <string name="app_name">로케일 예제</string>
    <string name="message">헬로우 월드!</string>
</resources>
```

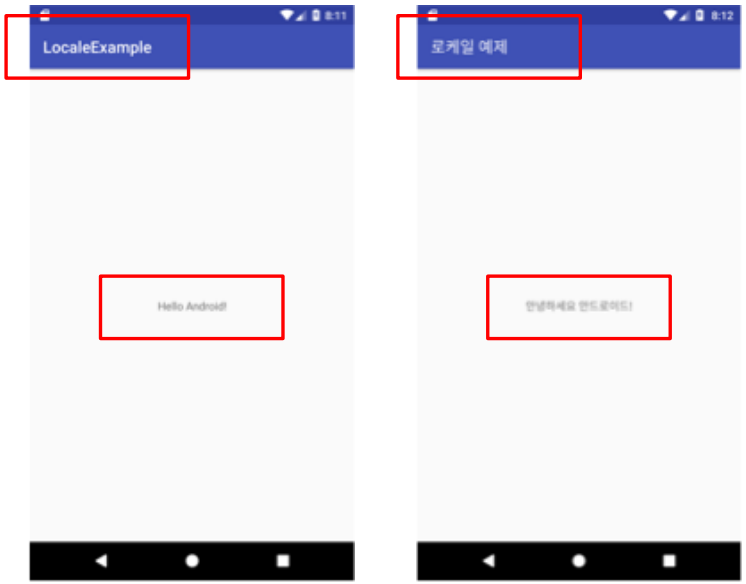
Android Basic Widgets

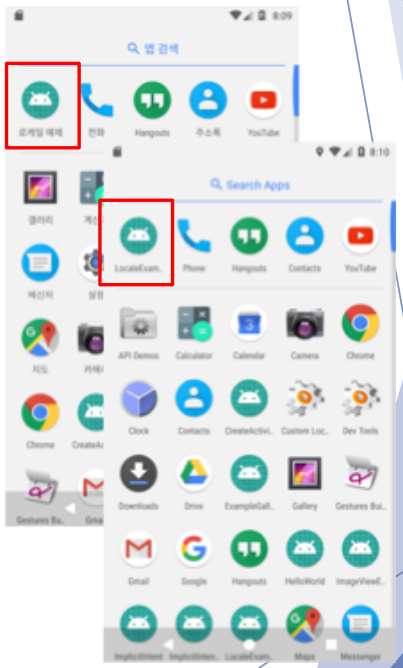
: TextView

[실습] LocaleExample

설정(settings) > 언어 및 키보드 (Language & input) > 언어(Language) 에서 **“English(United State)”** 와 **“한국어”** 를 각각 선택해보자.

앱을 실행 시켜 화면 변화와 론처에서 앱 아이콘의 변화를 확인해 보자.





Android Basic Widgets

: TextView

[실습] TextViewExample]
TextViewExample에서 TextView의 몇 가지 유용한 기능과 속성을 더 알아봅니다

1) text에 다음문장을 입력 합니다.

"For More Information,₩nPlease Visit www.android.com or call 1234-5678"

2) TextView너비를 레이아웃에 맞춥니다.

android:layout_width = "match_parent"

3) 텍스트의 사이즈를 변경한다.

android:textSize="25sp"

4) 다음 속성을 설정하고 어떤 속성인지 확인해 봅니다

android:autoLink="web|phone"

Android Basic Widgets

: TextView

[실습] TextViewExample

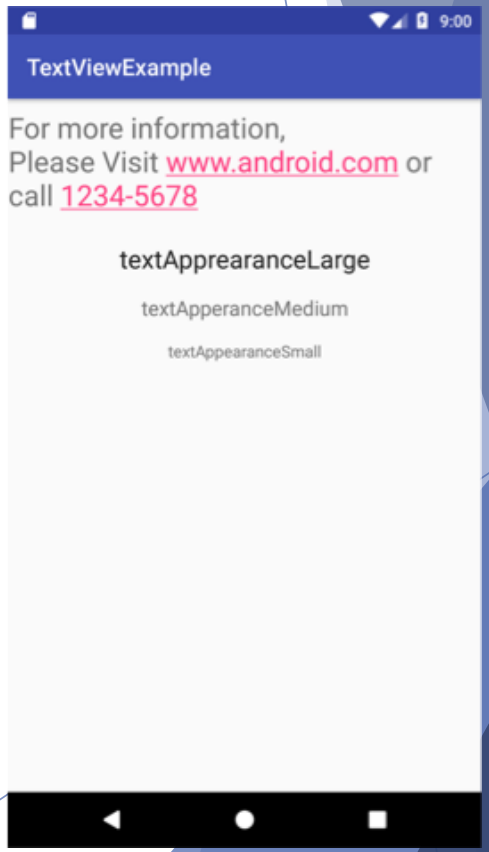
5) 다음 3개 TextView 를 추가한다.

```
android:text = "textAppearanceLarge"
android:text = "textAppearanceMedium"
android:text = "textAppearanceSmall"
```

6) 3 문장에 다음 속성들을 순서대로 적용해 봅니다. (테마 속성으로 지정)

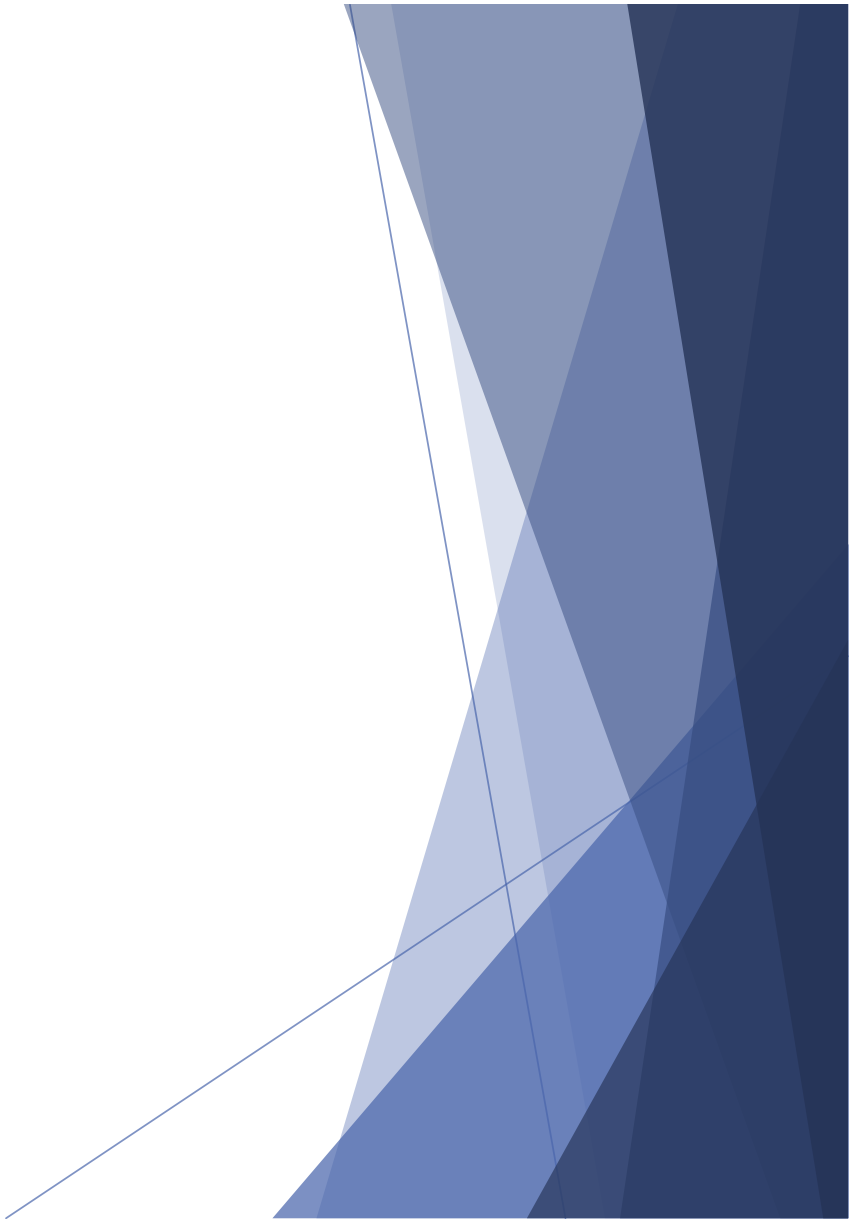
```
android:textAppearance = "?android:attr/textAppearanceLarge"

android:textAppearance = "?android:attr/textAppearanceMedium"

android:textAppearance = "?android:attr/textAppearanceSmall"
```

?attr 지정자: 현재 테마에 지정되어 있는 값을 사용하라는 의미



Android Basic Widgets

ImageView



Android Basic Widgets
: ImageView

- 아이콘이나 비트맵을 출력하는 위젯
- 리소스, 파일 등은 물론, 웹 상의 이미지도 표시할 수 있다

- 주요 속성
  - src
    - 출력할 이미지를 지정하는 가장 중요한 속성
    - ##RRGGBB 형태의 색상 값을 출력할 수 있다
    - 외부의 이미지 리소스를 지정할 수 있다
    - **리소스에 미리 준비해 놓은 이미지를 출력 (@drawable/ID)**
  - adjustViewBounds
    - 표시될 이미지의 종횡비를 맞춘다 : **true** 또는 **false**

Android Basic Widgets

: ImageView

- 주요 속성
  - scaleType
    - 이미지 출력 형식
    - CENTER, CENTER_CROP, CENTER_INSIDE, MATRIX, FIT_XY 등

Android Basic Widgets

: ImageView

[실습] ImageViewExample

ImageView의 속성을 확인하는 실습과제입니다. 액티비티가 하나인 프로젝트 ImageViewExample를 만들고 activity_main 레이아웃에 ImageView 위젯을 추가한 후, 다음 속성들을 추가해 보고 확인한다.

1) android:src = "@drawable/image" image.jpg 또는 image.png 이름의 이미지 파일을 추가 해보세요.

2) android:scaleType = "centerInside" 속성을 주고 그림을 확인해 보세요.

3) 다음의 속성값을 바꿔가면서 확인해 보세요.

centerInside
ImageView 틀의 높이나 넓이에 맞추어 이미지의 크기를 비율에 맞추어 확대 또는 축소하여 이미지가 ImageView 에 높이 또는 넓이에 맞추어 보여준다.

center
이미지 크기 그대로 ImageView 의 중앙에 위치한다.

centerCrop
ImageView 틀의 높이나 넓이에 맞추어 이미지의 크기를 비율에 맞추어 확대 또는 축소하여 이미지의 중앙 부분을 ImageView 에 꽉차게 보여준다.

Android Basic Widgets

: ImageView

[실습 ImageViewExample

3) 다음의 속성값을 바꿔가면서 확인해 보세요.

.

fitCenter
ImageView 틀에 맞게 이미지 비율에 맞추어 확대 또는 축소 되어 보이며 이미지가 중단에 위치하게 된다.

fitEnd
ImageView 틀에 맞게 이미지 비율에 맞추어 확대 또는 축소 되어 보이며 이미지가 하단에 위치하게 된다.

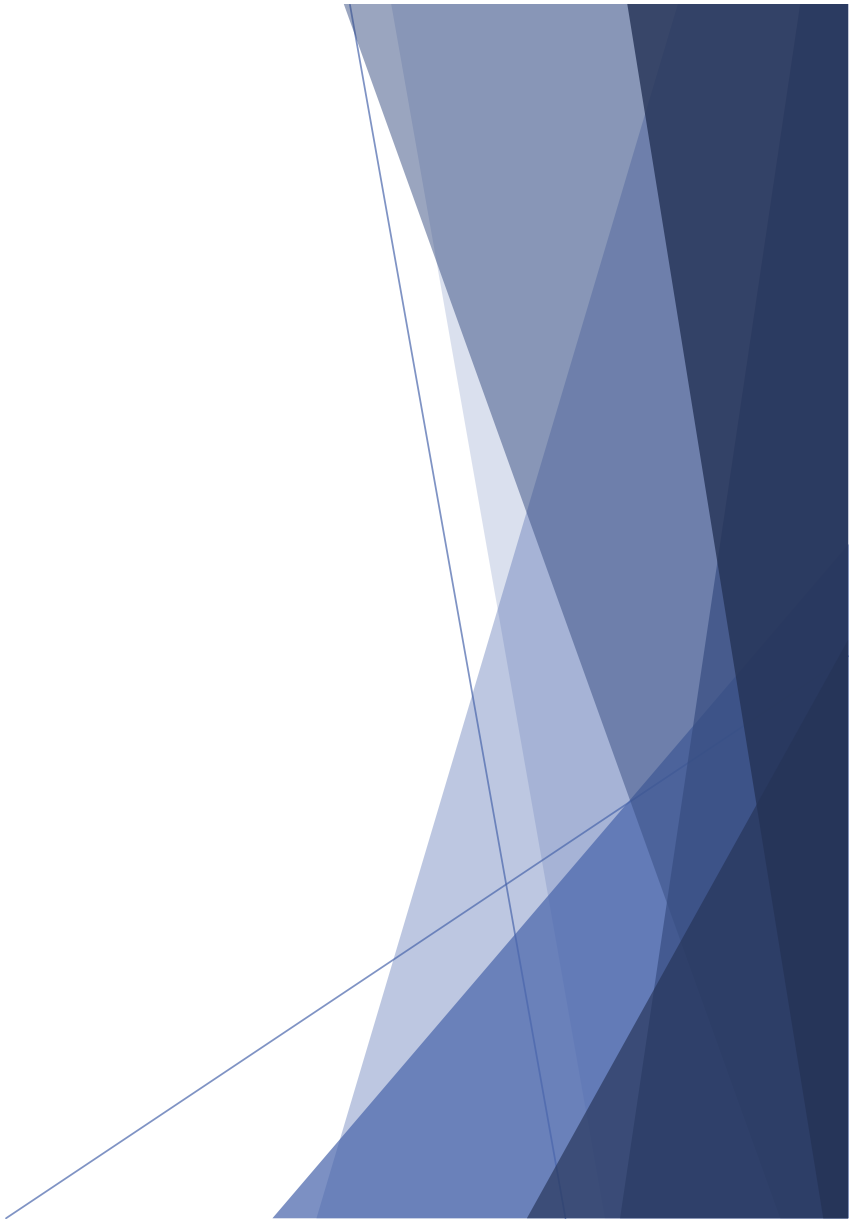
fitStart
ImageView 틀에 맞게 이미지 비율에 맞추어 확대 또는 축소 되어 보이며 이미지가 상단에 위치하게 된다.

fitXY
ImageView 틀에 맞게 이미지가 비율에 상관없이 틀에 꽉차게 확대 또는 축소 되어 보인다.
이미지의 좌우 크기가 맞지 않는 경우 이미지가 변형되어 보일 수 있다.

matrix
ImageView 틀에 기준되어 왼쪽 최상단에 맞추어 보여진다. 이미지가 변형이 되지 않기 때문에 틀이 작게 되면 이미지가 잘려서 보인다.

Android Basic Widgets

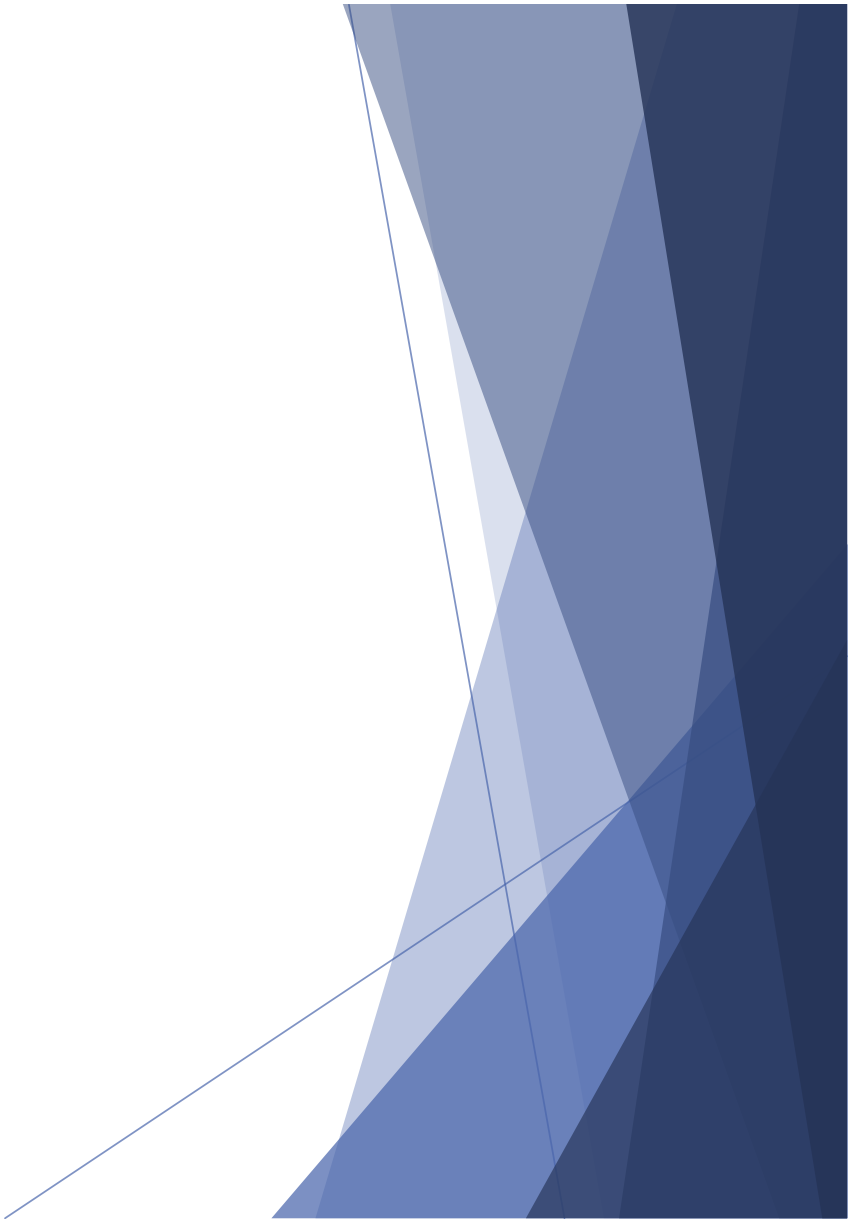
입력 처리



Android Basic Widgets

: 입력처리 - EditText

- 개요
  - EditText는 글자를 입력 받음
  - 입력 받을 문자의 종류를 제한한다
  - 비밀번호를 입력받을 때 내용을 안보이게 할 수 있다
  - 행의 개수 제한
  - 정렬 방법 지정
- 주요 속성
  - android:hint : 입력해야 할 내용에 대한 설명
  - android:inputType
  - android:line : 행의 개수
  - android:gravity : 기본 정렬 방향 지정



Android Basic Widgets

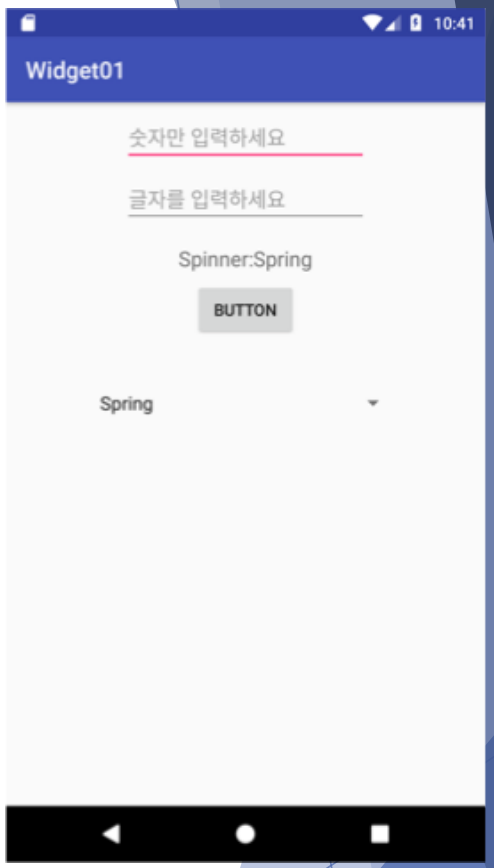
: 입력처리 - Spinner

- 개요
  - Spinner는 콤보박스와 유사
  - 미리 지정된 몇 개의 항목 중 하나를 선택하도록 함
- 주요 속성
  - prompt : Spinner를 클릭했을 때 제목으로 표시될 문자열
  - entries : 목록에 표시될 항목들. 문자열 배열 리소스이 ID
- 리스너의 구현 예

```
spinner = findViewById(R.id.spinner);
spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
        String itemName = (String)adapterView.getItemAtPosition(i);
        tvMessage.setText("Spinner:" + itemName);
    }
```

Android Basic Widgets

: 입력처리 실습



[실습] Widget01

EditText, Spinner, Button의 속성을 확인하는 실습과제입니다.
액티비티가 하나인 프로젝트 Widget01을 만듭니다.

1) EditText 위젯을 추가하고

android:inputType = "number“ 가 숫자만 받는지 확인

android:hint = “숫자만 입력하세요” 를 추가한 후, 어떤 변화가 있는지 확인

2) EditText 를 하나 더 추가하고 이번에는 문자만 입력 받도록 수정

hint 텍스트: “글자를 입력하세요”

3) TextView 와 Button을 추가하고 버튼을 클릭 했을 때 EditText의 내용을 TextView에 나타나게 처리하기

4) Spinner를 추가하고

android:entries 속성에 @array/lists 를 추가한다 (lists란 이름의 스트링 배열을 만들어야 함)

5) Spinner에 선택이 변화되면 발생하는 이벤트의 리스너를 달고 바뀐 값을 TextView에 나타나게 처리

Android Basic Widgets

: 입력처리 – RadioButton, CheckBox

- RadioButton
  - 여러 항목 중 단 ‘하나’만을 선택할 때 사용
  - 여러 항목 중 하나만을 골라야 하기 때문에 선택 가능한 항목의 범위를 지정해야 함
  - RadioGroup이라는 컨테이너를 사용한다
- CheckBox
  - 여러 항목 중 원하는 항목을 중복하여 선택할 때 사용

Android Basic Widgets

: 입력처리 실습

[실습] Widget02

RadioButton과 CheckBox의 사용방법을 실습합니다. 액티비티가 하나인 프로젝트 Widget02를 만들고 레이아웃은 다음을 참고합니다.

Widget02 > activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <RadioGroup
        android:id="@+id/radioGroup"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <RadioButton
            android:id="@+id/radio1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Female"
            android:checked="true"
            />
        <RadioButton
            android:id="@+id/radio2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Male"
            />
    </RadioGroup>
        <CheckBox
            android:id="@+id/checkbox"
            android:text="Check Me!"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <TextView
            android:id="@+id/radioChecker"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <TextView
            android:id="@+id/checkboxChecker"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

</LinearLayout>
```

Android Basic Widgets

: 입력처리 실습

[실습] Widget02

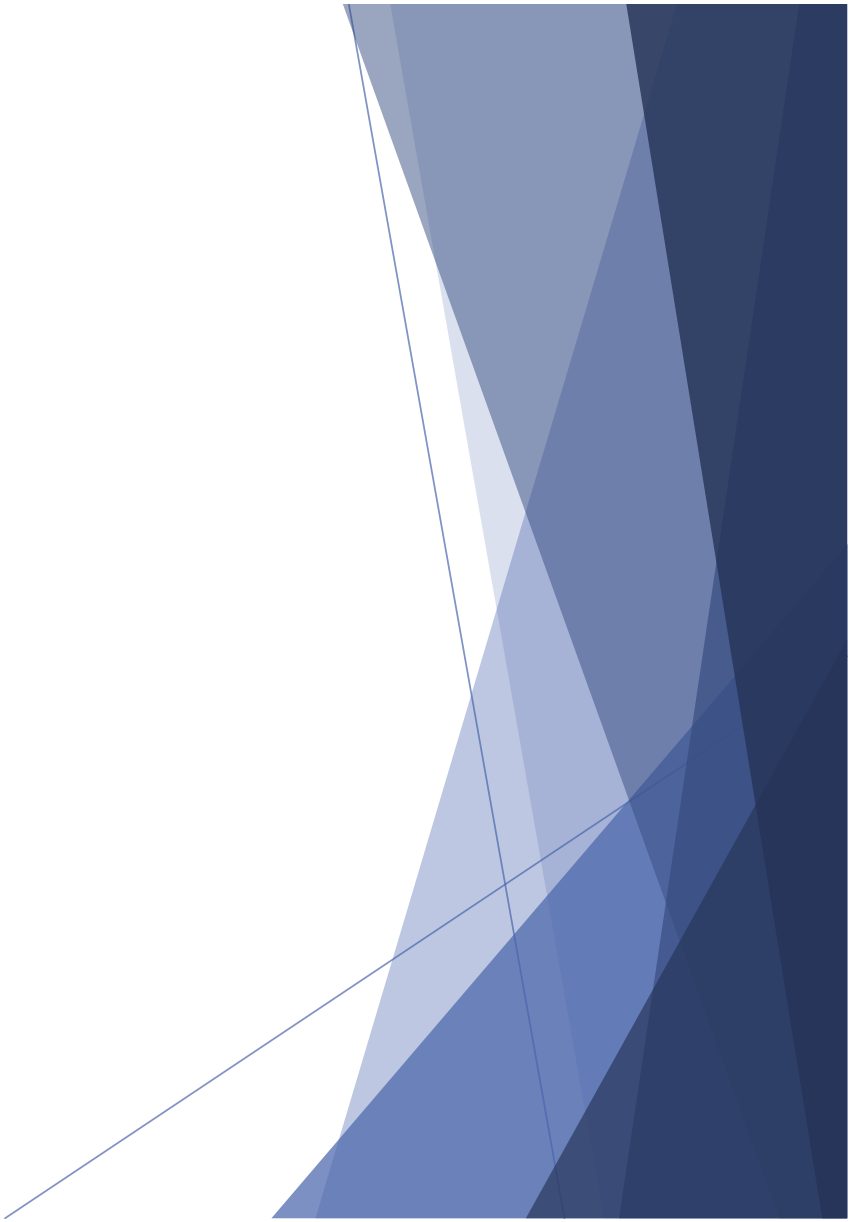
1) RadioButton 선택 이벤트 처리하기

```
RadioGroup radioGroup = findViewById(R.id.radioGroup);
radioGroup.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(RadioGroup radioGroup, int i) {
        //  ...
    }
});
```

2) CheckBox 이벤트 처리하기

```
CheckBox checkBox = findViewById(R.id.checkbox);
checkBox.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //  ...
    }
});
```

Android Layout



Android Layout

: Layout 이란?

- 레이아웃은 다양한 위젯들을 화면에 배치하여 UI를 구성하고 설계할 수 있게 한다.
- 레이아웃은 ViewGroup으로부터 파생되어 여러 뷰들을 계층화하고 구조화할 수 있다.
- 레이아웃은 다른 레이아웃의 자식 뷰로도 배치될 수 있으며, 이를 이용하여 화면의 UI 구성을 다양하고 풍부하게 할 수 있다
- 안드로이드 플랫폼 레이아웃 생성 방법 2가지
  - XML 리소스 파일 이용
  - Activity 클래스 내에서 해당 레이아웃을 생성

Android Layout

: 레이아웃 생성 – XML 리소스 파일의 이용

- XML 리소스 파일을 이용한 레이아웃 생성은 레이아웃을 정의하는 가장 일반적인 방법
- XML 내의 모든 UI 구성에 대한 설계가 이루어지기 때문에 HTML을 이용한 웹 화면 개발과 비슷한 익숙함을 제공
- 레이아웃을 정의하는 XML 리소스 파일은 /res/layout 디렉터리에 위치하여 관리
- UI 화면들을 각 레이아웃별로 XML 리소스 파일로 정의
- 각 UI 화면별 위젯의 정적인 생성과 배치, 그리고 속성값 설정을 대부분 코드가 아닌 XML에서 정의할 수 있다
- 레이아웃 XML 리소스 파일을 이용해서
  - UI 설계가 간단하게 진행될 수 있으며,
  - UI 레이아웃 관련 불필요한 코딩이 줄어들고
  - 해당 Activity 클래스 코드를 깔끔하게 정리할 수 있다.

Android Layout

: 레이아웃 생성 - 직접 코드 작성

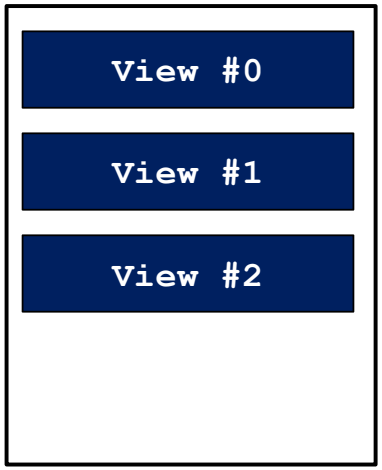
- 실행 시점에 몇몇 속성 값들을 변경할 때 유용한 방식
- 실수를 유발하기 쉬우며, 유지보수가 어렵고, UI 설계상 직관적이지 않다
- 코드를 이용해서는 동적으로 레이아웃이나 위젯의 속성 값들을 변경하는 용도로 사용하는 것이 바람직

```
 public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        TextView text = new TextView(this);
        text.setText(R.string.hello);

        Button button = new Button(this);
        button.setText(R.string.hello);

        LinearLayout layout = new LinearLayout(this);
        layout.setOrientation(LinearLayout.VERTICAL);

        layout.addView(text);
        layout.addView(button);

        setContentView( layout );
}
```

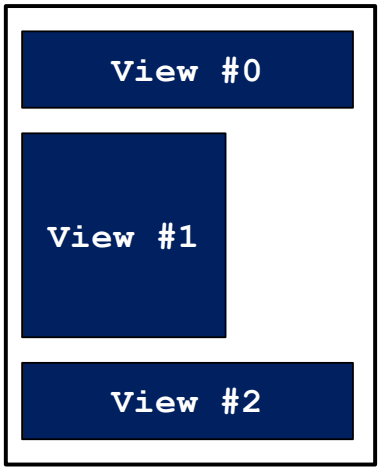
Android Layout

: 레이아웃의 종류

- LinearLayout:
  - 수평 또는 수직으로 UI 구성 요소를 배치하는 레이아웃. 가장 기본적인 레이아웃
- RelativeLayout:
  - UI 구성요소들 간의 상대적 위치 관계에 따라 UI 구성요소를 배치하는 레이아웃
  - 각 구성요소를 세밀하게 배치할 수 있어, 좀 더 복잡한 화면을 구성하는데 적합



LinearLayout

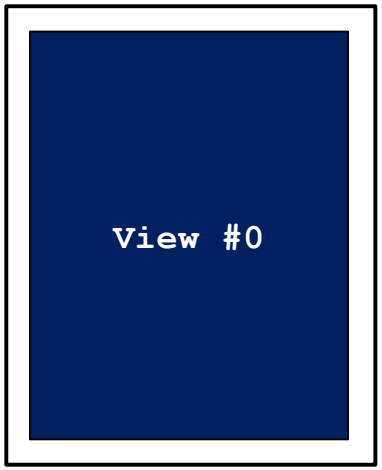


RelativeLayout

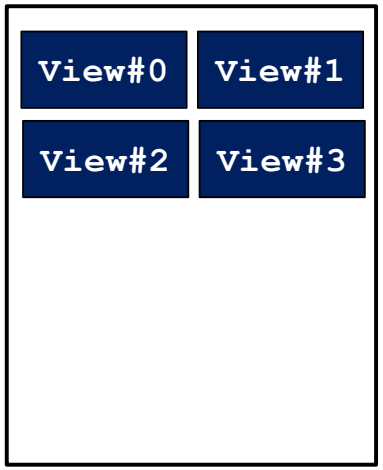
Android Layout

: 레이아웃의 종류

- FrameLayout:
  - 각 UI 구성요소를 왼쪽 상단으로부터 겹쳐서 배치. Tab 등을 구성할 때 적합
- TableLayout:
  - 표 형태로 UI 구성요소를 배치.
- AbsoluteLayout:
  - UI 구성요소를 화면의 절대 좌표에 배치. 현재는 폐기



FrameLayout



TableLayout

Android Layout

: LinearLayout

- 가장 흔히 사용하며 가장 중요한 레이아웃으로 거의 모든 레이아웃을 구성할 수 있다
- 방향(orientation)
  - 뷰를 차례로 추가할 때 가로(horizontal) 또는 세로(vertical) 방향을 결정
  - 기본은 세로(vertical)

[실습] LinearLayoutExample

앱의 기본 레이아웃(activity_main.xml)과 일반적인 경우 버튼을 눌렸을 때 변화되는 레이아웃(normal.xml)를 비교해 봅시다.

Android Layout

: LinearLayout

- 정렬 방향(gravity)
  - layout_gravity : 부모 뷰에서 자신의 뷰 위치 지정
  - gravity : 뷰 안의 콘텐츠를 정렬할 때

[실습] LinearLayoutExample

1. 화면의 'gravity 사용하기' 버튼을 눌렸을 때 변화되는 레이아웃(gravity.xml)에서 세 개의 버튼의 layout_gravity 속성의 값들과 배치 모습을 비교해 보세요.

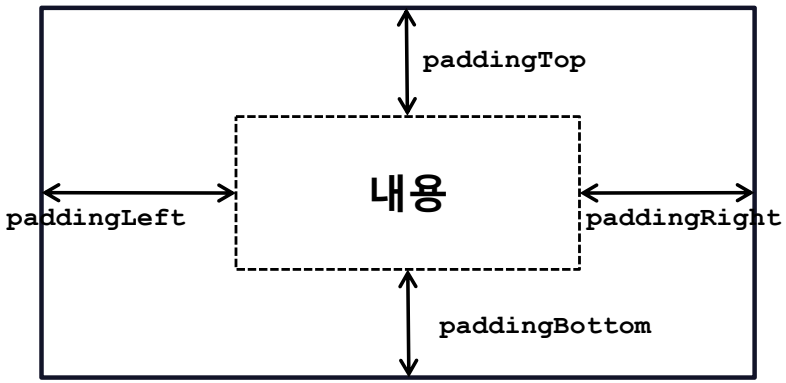
2. gravitytext01.xml, gravitytext02.xml, gravitytext03.xml 에서 garivity 속성 left, center_horizontal, right, bottom, center_vertical, top 의 차이점을 확인해 보세요.

.

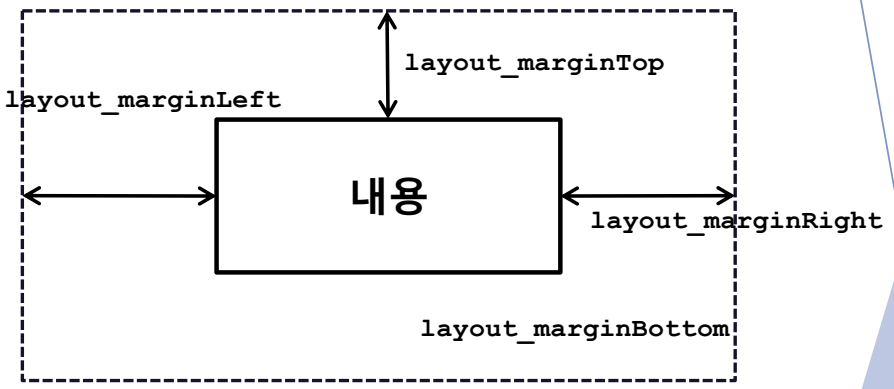
Android Layout

: LinearLayout

- 여유공간(layout_margin, padding)



뷰(view)



widget cell

[실습] LinearLayoutExample

1. 화면의 'gravity 사용하기' 버튼을 눌렸을 때 변화되는 레이아웃(gravity.xml)에서 세 개의 버튼의 layout_gravity 속성의 값들과 배치 모습을 비교해 보세요.

2. gravitytext01.xml, gravitytext02.xml, gravitytext03.xml 에서 garivity 속성 left, center_horizontal, right, bottom, center_vertical, top 의 차이점을 확인해 보세요.

Android Layout

: LinearLayout

- 공간 가중치(layout_weight)
  - 두 개 이상의 뷰가 같은 부모의 남아있는 여유공간을 가중치로 할당받음
  - 두 뷰의 layout_weight가 1로 설정되어 있으면 1 / 2씩 공간을 할당받음
  - 두 뷰의 layout_weight가 1, 2로 설정되어 있으면 각각 1 / 3, 1 / 2씩 공간을 할당받음

[실습13] LinearLayoutExample

화면의 'weight 사용하기' 버튼을 눌렸을 때 변화되는 레이아웃(weight.xml)에서 세 개의 버튼의 layout_weight 속성의 값들과 배치 모습을 비교해 보세요.