

블록 코딩 이해하기



Contents Title

2

주 차	수업 내용
1	수업 소개
2	앱 인벤터 개발환경과 디자인 편집기
3	블록 코딩 이해하기(Hello 앱 인벤터)
4	안녕 야옹이
5	페인터 통
6	파리 관광
7	잡아라 두더지
8	중간고사
9	운전 중 문자 금지(or 바로 콜)
10	무당벌레 추적
11	내 차를 찾아줘
12	실로폰
13	서점에서 온라인 검색하기
14	메모장 만들기
15	기말고사



경태

thinkdata.co.kr/wordpress/?page_id=27

Bookmarks English Grammar Gar ozdic.com - the Engl Templates - Free tem All IT eBooks - Free Gurubee 안드로이드 [안드로이드 스튜디오 'Spring' 카테고리의

THINKDATA Creative...

Home Blog Lecture QnA 게시판 PW: 3289

앱 인벤터2

앱 인벤터2 강의 게시판입니다. 반드시 분반 확인을 하세요

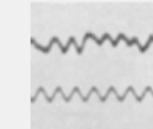
전체 Class A Class B

번호	제목	작성자	날짜	조회
4	[Class B] B.Week03.블록 코딩 이해하기	ktpark	23:04	1
3	[Class A] A.Week03.블록 코딩 이해하기	ktpark	23:03	1
2	[Class B] 강의 자료-Week02	ktpark	2017-03-14	2
1	[Class A] 강의 자료-Week02	ktpark	2017-03-14	2

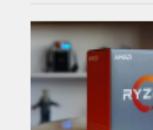
제목 검색

FOLLOW: [f](#)

① ★ ⌂

 BIOSIGNAL
Biosignal – Wikipedia
73월, 2017

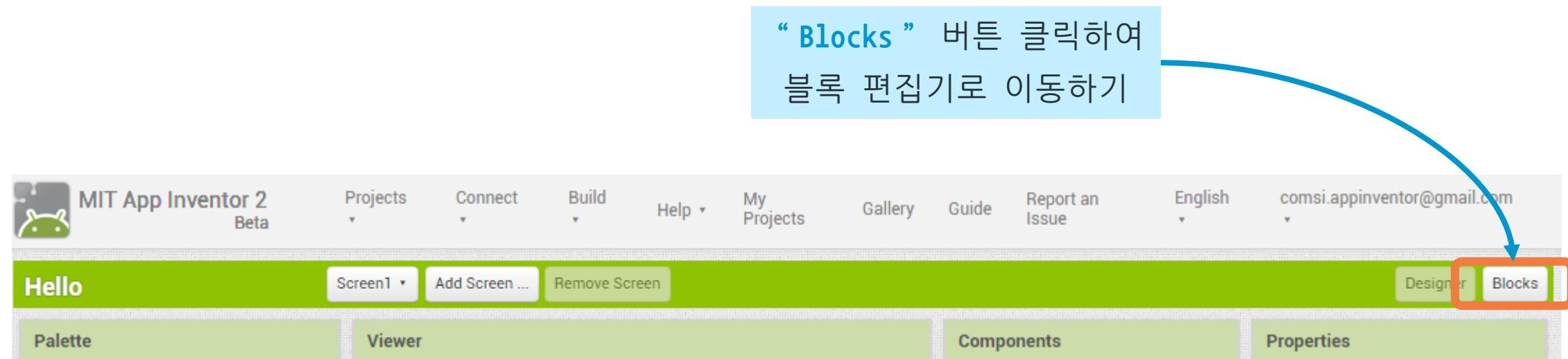
 BIOSIGNAL
Heart rate – Wikipedia
73월, 2017

 NEWS
“AMD가 돌아 왔다” 라이젠
CPU 집중 해부 벤치마크 –

2. 블록 코딩 알아보기

2. 블록 코딩 알아보기

- 디자인 편집기에서 **[Blocks]버튼**을 클릭하여 블록 편집기로 이동
- 블록 편집기는 **Blocks** 패널과 **Viewer** 패널로 구성



• 블록 편집기 화면

The screenshot shows the MIT App Inventor 2 Designer interface. At the top, there's a navigation bar with links for Projects, Connect, Build, Help, My Projects, Gallery, Guide, Report an Issue, English, and a user email (comsi.appinventor@gmail.com). Below the navigation bar, the title "HelloWorld" is displayed, along with buttons for Screen1, Add Screen ..., Remove Screen, Designer, and Blocks.

The main area is divided into two sections:

- Blocks Panel (Left):** Labeled with a blue circle containing the number 1. It contains a tree view of blocks categorized by type:
 - Built-in: Control, Logic, Math, Text, Lists, Colors, Variables, Procedures
 - Screen1: Button1, Label1, Button2
 - Any component: Any Button, Any LabelAt the bottom of this panel are buttons for Rename and Delete.
- Viewer Area (Right):** Labeled with a blue circle containing the number 2. This is where the blocks from the Blocks panel are assembled into a visual script. It features a large workspace, a trash can icon in the bottom right corner, and status indicators at the bottom left (yellow warning icon 0, red error icon 0, and a Show Warnings button).

① **Blocks:** 앱의 기능을 만드는 데 사용되는 블록이 종류별로 모여 있는 공간

② **Viewer:** Blocks 패널에 있는 블록을 가져와 앱이 사용자의 행동에 반응해서 작동하게 만드는 작업 공간

1) 블록 코딩 vs. 텍스트 코딩

7

“안녕하세요” 팝업 출력

```
package com.example.park.myapplication;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btnHelloWorld = (Button) findViewById(R.id.btnHelloWorld);
        btnHelloWorld.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Toast.makeText(getApplicationContext(), "안녕하세요", 1000).show();
            }
        });
    }
}
```

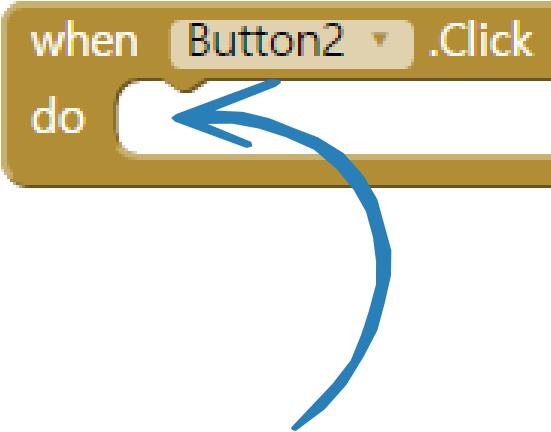
텍스트 코딩



블록 코딩

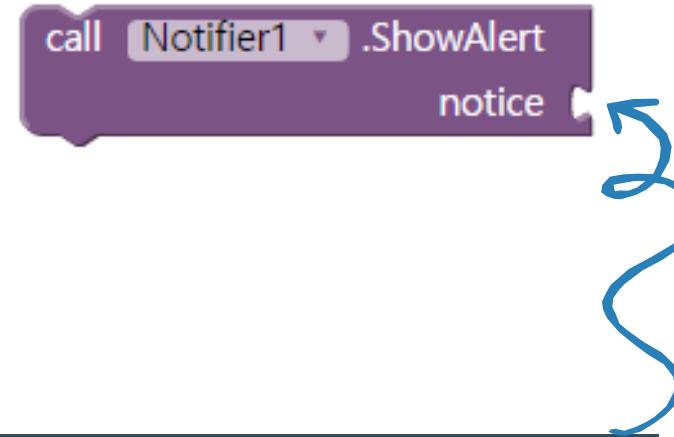
블록의 종류와 명칭

8



섹션(Section)

실행할 블록을 여러 개 붙여 순차적 처리를 할 수 있는 블록



소켓(Socket)

실행할 블록을 하나 끼워 처리할 수 있는 블록으로 소켓 블록을 끼운다.

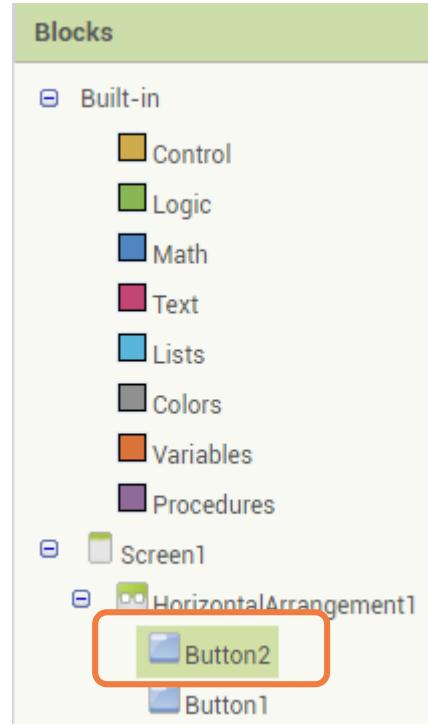


플러그(Plug)

소켓에 끼워 사용할 수 있는 블록

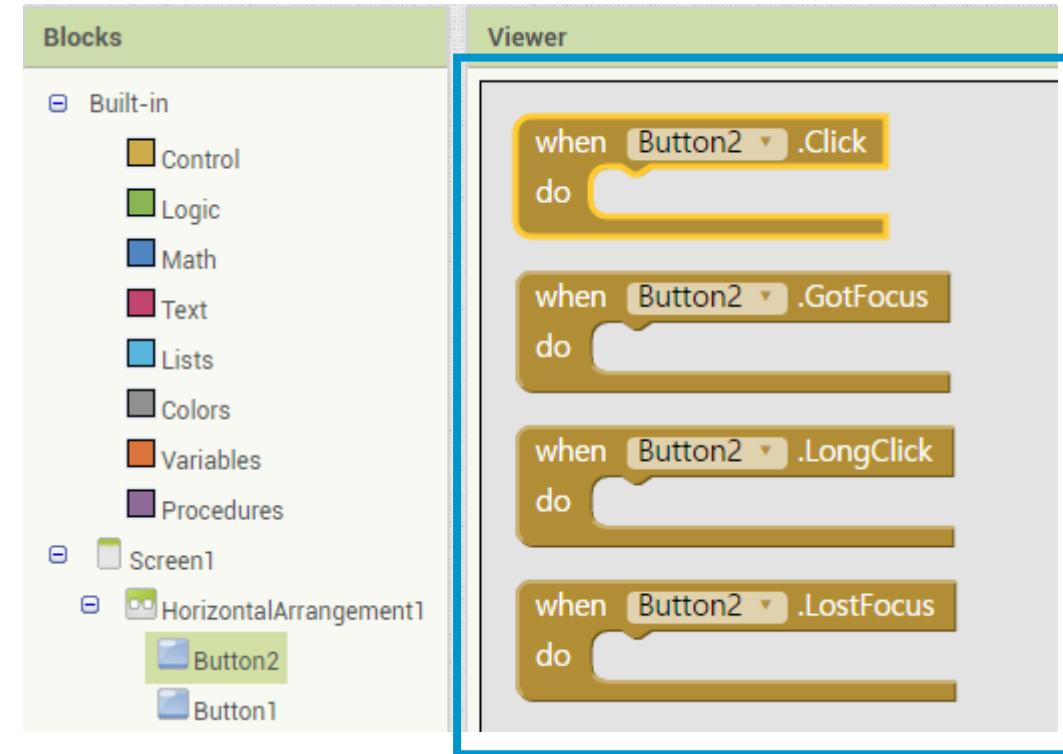
2] 블록 코딩 방법 – 3단계

9



Step 1

Blocks 패널에서 코딩을 위한 요소(**Button2**)를 클릭하여 사용 가능한 팝업창(**step2 사각형**)을 띄운다.



Step 2

블록 창(**사각형**)에서 사용하고자 하는 블록을 선택하여 드래그 한다.



Step 3

드래그 한 블록을 Viewer에 드롭한다.

3] 블록 추가하기와 삭제하기

- 앱 종료 블록을 추가해 보자

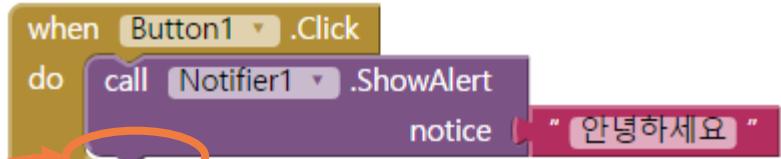
Blocks

- Built-in
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Colors
 - Variables
 - Procedures
- Screen1
 - Button1
 - Notifier1
- Any component

Viewer

- if
then
else
- do
result
- evaluate but ignore result
- open another screen screenName
- open another screen with start value screenName
startValue
- get start value
- close screen
- close screen with value result
- close application
- get plain start text

Rename Delete



블록 추가

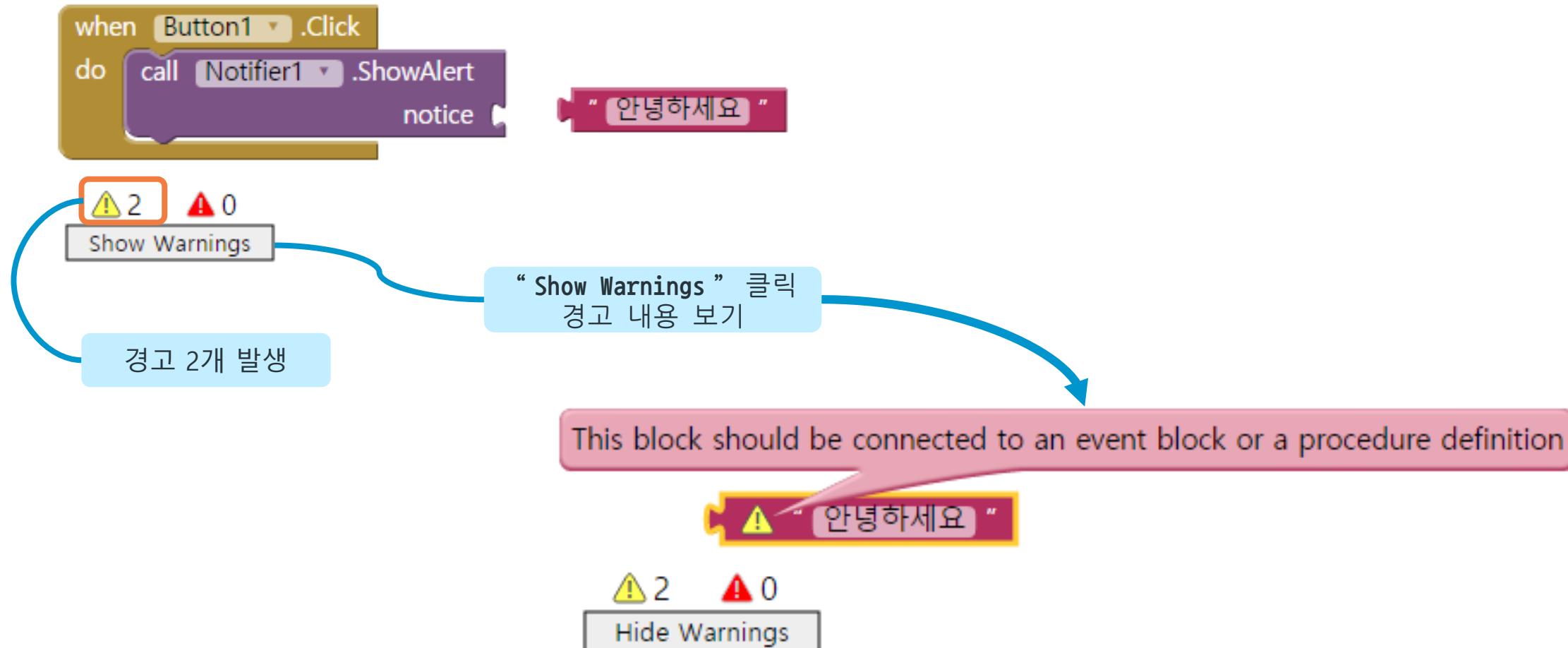
블록 삭제

쓰레기통에 버림



4) 오류 처리 – 경고와 에러

☞ 오류 - 경고 메시지(앱을 빌드하여 스마트폰에 정상적으로 설치하여 실행할 수 있다)



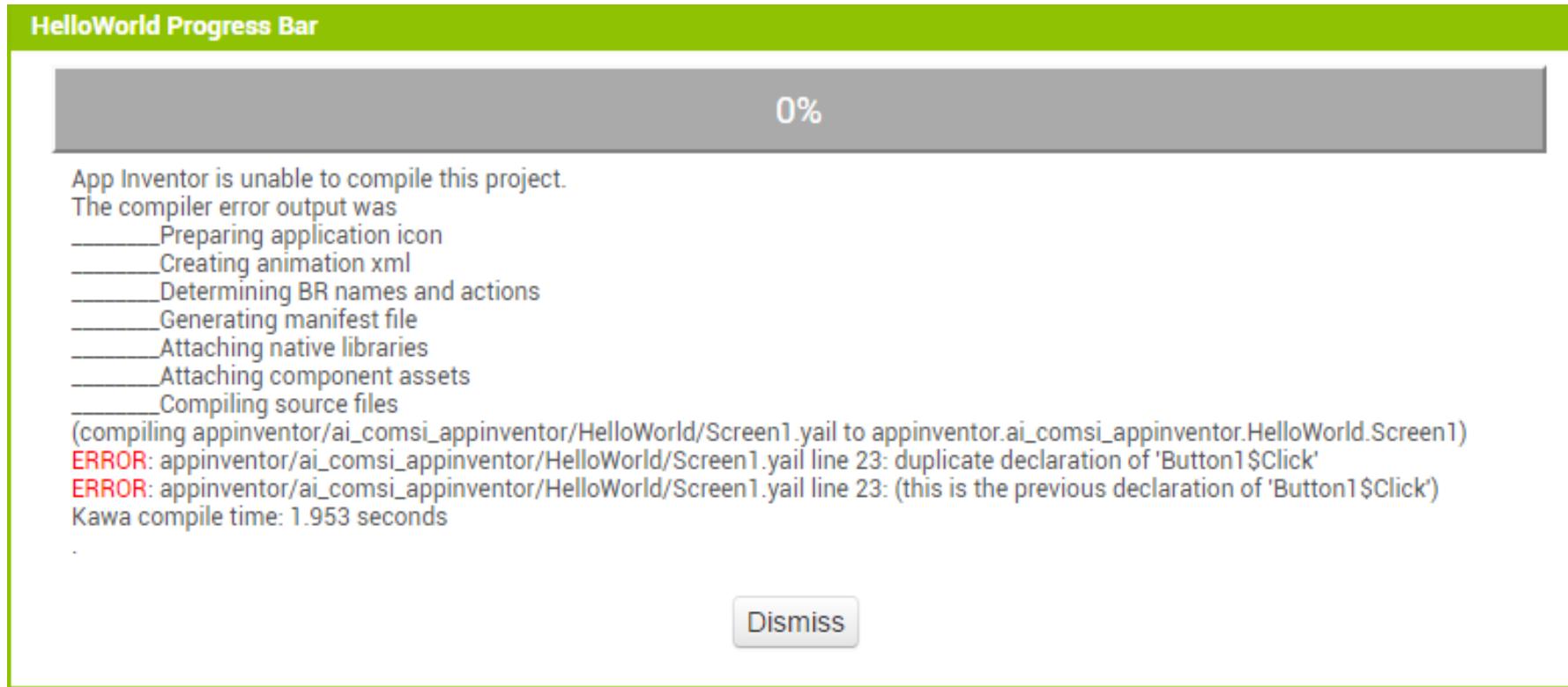


오류 - 에러 메시지(앱을 빌드할 수 없다)





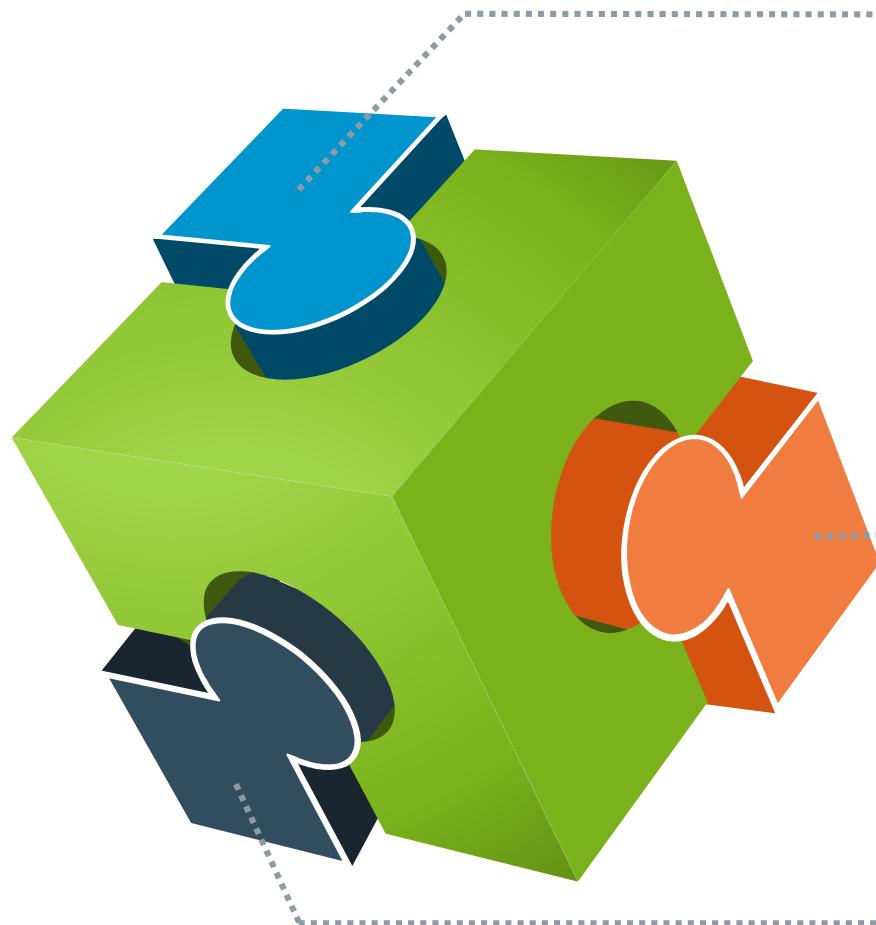
에러 상태에서 빌드(Build → App(provide QR code for .apk) 메뉴 실행) 할 경우 오류



- 첫번째 빨간색 **ERROR:** duplicate declaration of 'Button1\$Click'
- 해결:** 중복된 [when Button1.Click] 블록을 쓰레기통에 버리면 오류가 제거되고 빌드가 정상적으로 진행된다.

3. 블록의 구분

3. Blocks의 구분



Built-in

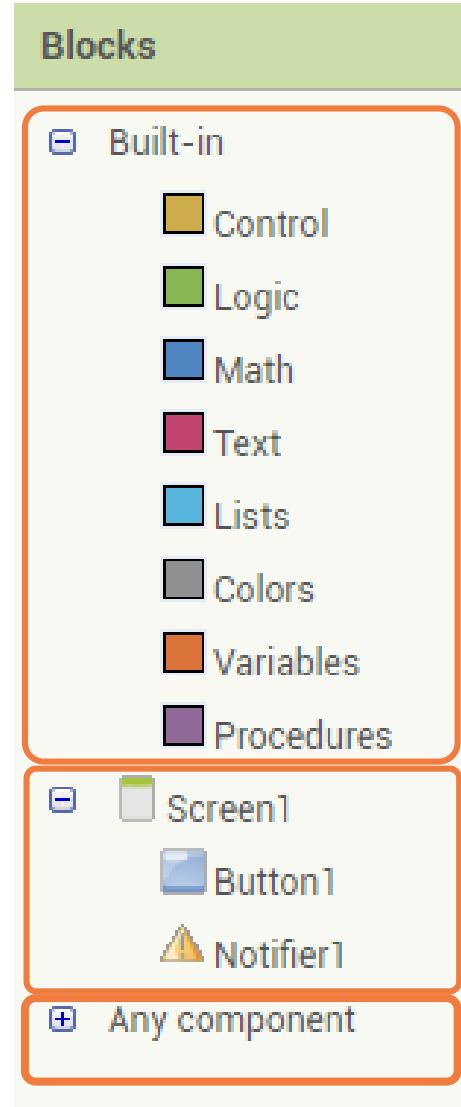
기능에 따라
[Control], [Logic], [Math],
[Text], [List], [Colors],
[Variables], [Procedures]
8가지로 분류.

Screen1

디자인 편집에서
사용한
컴포넌트들이
트리구조로 나열

Any component

Screen1에 있는 같은
종류의 컴포넌트들을
한꺼번에 제어할 때 사용





DatePicker 컴포넌트의 블록 목록(블록의 기능에 따라 분류-색상구분)

The image shows a Scratch script window with the following blocks:

- Blocks palette (Left):**
 - Built-in:
 - Control (Yellow)
 - Logic (Green)
 - Math (Blue)
 - Text (Red)
 - Lists (Light Blue)
 - Colors (Grey)
 - Variables (Orange)
 - Procedures (Purple)
 - Screen1
 - Button1
 - DatePicker1 (highlighted in green)
 - Notifier1
 - Any component- Viewer (Right):**
 - Event Handler Block:** when [DatePicker1].TouchUp do []
 - Procedure Call Block:** call [DatePicker1].LaunchPicker
 - Procedure Call Block:** call [DatePicker1].SetDateToDisplay [year, month, day]
 - Procedure Call Block:** call [DatePicker1].SetDateToDisplayFromInstant [instant]
 - Property Block:** [DatePicker1].BackgroundColor
 - Property Set Block:** set [DatePicker1].BackgroundColor to []

Annotations on the right side categorize the blocks:

- Yellow box: 이벤트 핸들러 블록 (Event Handler Block)
- Purple box: procedure 호출 블록 (Procedure Call Block)
- Green box: 속성 블록 (Property Block)

1] 이벤트 핸들러 블록

- 사용자가 특정 행동을 했을 때 지정된 동작을 하도록 사용하는 블록
- 이벤트는 사용자가 발생시키는 경우와 내부적으로 특정한 일을 수행하기 전이나 후에 발생하기도 한다.
- 황토색으로 되어 있고, 시작 부분에 “when”이라는 단어로 시작하고 이벤트가 발생했을 때 수행하고자 하는 블록은 “do” 섹션에 삽입



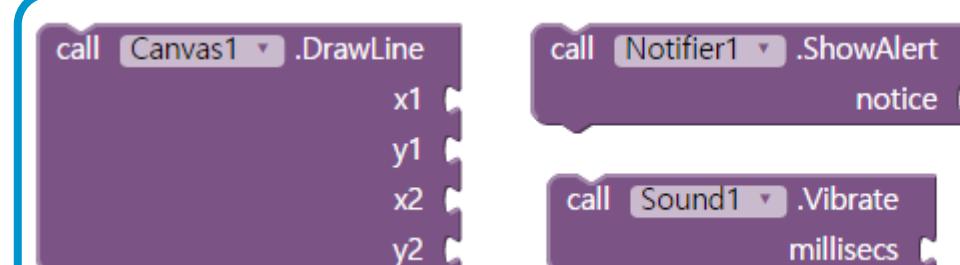
다양한 컴포넌트에서 발생하는 이벤트 블록

이벤트 핸들러 블록에서 매개변수 받기

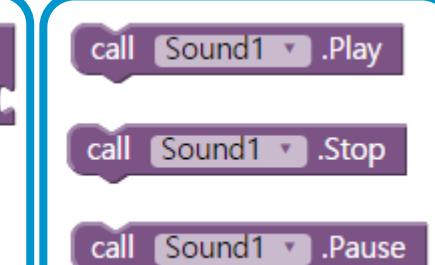
2] procedure 블록

- **procedure**는 컴포넌트가 수행해야 할 동작을 미리 만들어 놓은 보라색 블록
- 소켓이 있어서 **매개변수를 받아서 처리하는 경우**와 **단순하게 정해진 동작만 하는 경우**

매개변수를 받아서 처리하는 경우



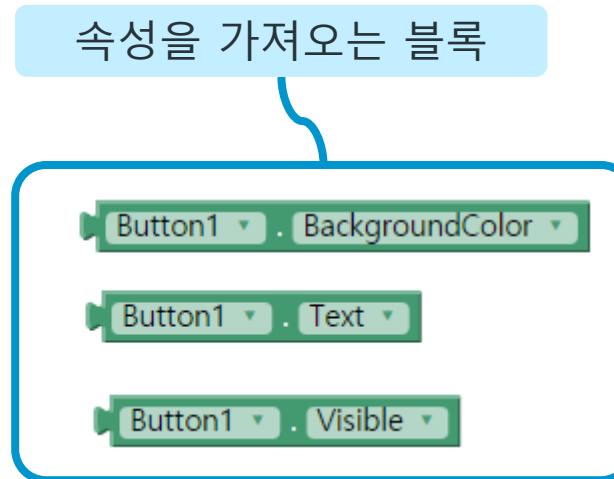
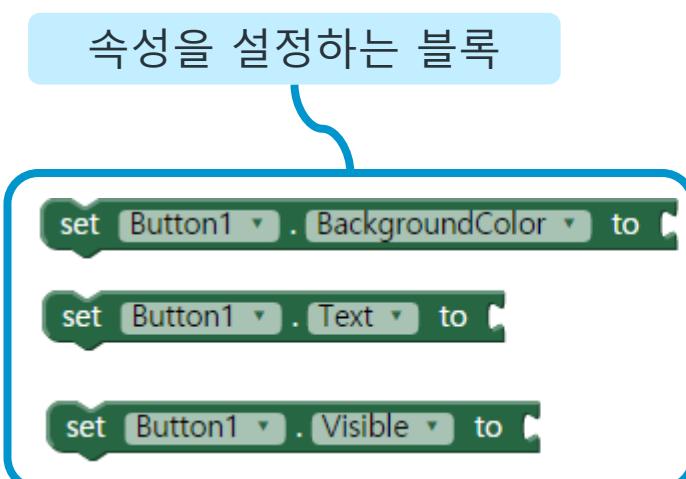
단순하게 정해진 동작만 하는 경우



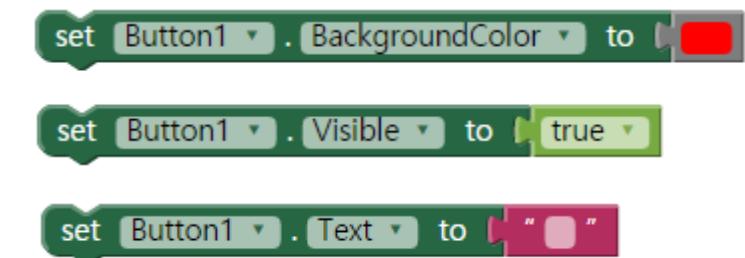
다양한 컴포넌트에서 사용하는 procedure 블록

3] 속성 블록

- 앱 인벤터에 사용되는 컴포넌트의 속성을 설정하거나 가져오는 초록색 블록
- 속성을 설정하는 블록은 오른쪽에 소켓이 있고, 가져오는 블록은 왼쪽에 플러그가 구성되어 있음
- 속성을 설정하기 위한 블록은 “set”으로 시작하고 “to” 소켓으로 끝나는 블록



버튼 컴포넌트의 다양한 속성 블록들



버튼 컴포넌트의 속성 설정

4) Built-in 블록

- 일반적인 프로그래밍 언어에서 제공하는 명령어와 함수를 제공

■ Built-in



Control

조건에 따라 프로그램의 흐름을 제어하는 **황토색** 블록



Logic

흐름 제어에 필요한 논리 조건을 처리하는 **초록색** 블록



Math

기본적인 숫자자체, 비교연산, 사칙연산, 난수 발생 등의 **파란색** 블록



Text

문자열 자체, 문자열의 공백 제거나 길이, 문자열을 연결하거나 자르거나 대체하는 **자주색** 블록



Lists

여러 개의 값을 저장하는 변수 공간으로 **청록색** 블록



Colors

컴포넌트의 배경이나 글자색을 설정할 수 있는 **회색**의 색상블록
(기본 13가지 색상과 Red, Green, Blue의 농도 조절로 색상 조절 가능)



Variables

값을 임시로 저장할 수 있는 변수를 만드는 **주황색** 블록



Procedures

반복적으로 사용되는 기능을 모아 만든 **보라색** 블록
(기본 내장 procedure와 사용자 정의 procedure를 생성 가능)

[1] Control 서랍

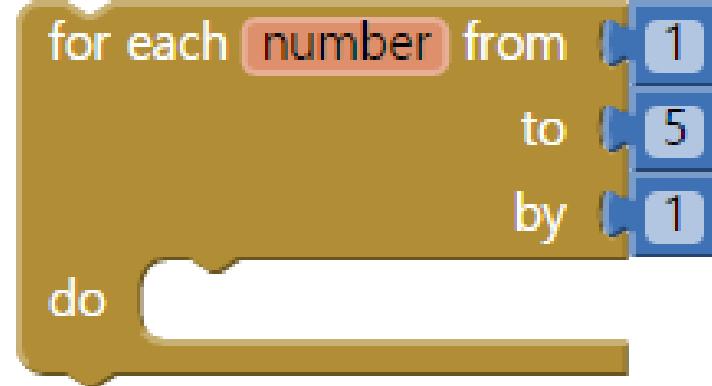
21

- 조건에 따라 프로그램의 흐름을 제어하는 **황토색** 블록

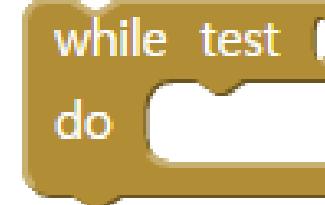
① if then



② for each ~ from to by



③ while test



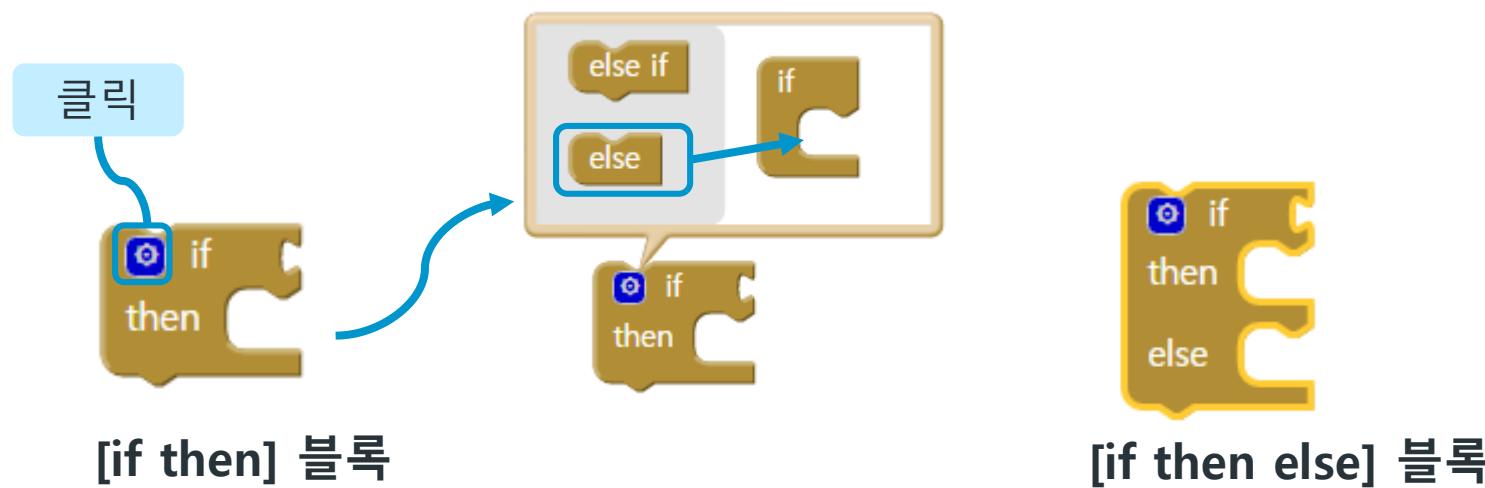
④ for each ~ in list



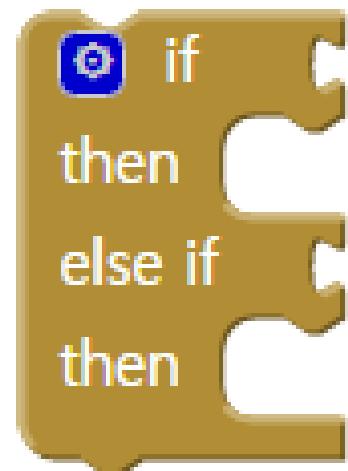
- if 소켓의 조건이 참일 경우에만 then 섹션의 블록을 실행
- by 소켓의 값을 증가양으로 하여 from 소켓의 값부터 to 소켓의 값까지 do 섹션의 블록을 반복적으로 수행. 이때, number에는 현재 값이 저장됨
- test 소켓의 조건이 참일 동안 do 섹션을 반복 실행
- list 소켓의 내용을 하나씩 가져와 모두 사용될 때까지 do 섹션 실행. item에서 list에서 가져온 내용이 저장됨

[if then] 블록으로 [if then else] 블록으로 확장하기

22



- 같은 방법으로 **[else]** 블록 대신에 **[else if]** **[if then else if then]** 블록도 만들 수 있다.



[2] Logic 서랍

23

- 흐름 제어에 필요한 논리 조건에 사용되는 초록색 블록



- ① “true” 값 블록
- ② “false” 값 블록
- ③ “not” 블록. 뒤쪽 소켓에 대한 not값을 돌려줌
- ④ 양쪽의 값을 비교하여 같으면 “true”, 다르면 “false”
- ⑤ 양쪽의 논리값이 둘 다 “true” 이면 “true”, 다르면 “false”
- ⑥ 양쪽의 논리값이 둘 중 하나가 “true” 이면 “true”, 다르면 “false”

[3] Math 서랍

- 기본적인 숫자자체, 비교연산, 사칙연산, 난수 발생 등의 **파란색** 블록



- ① 기본적인 숫자자체
- ② 비교연산($=, \neq, <, \leq, >, \geq$)
- ③ 사칙연산과 멱수($+, -, \times, /, ^$)
- ④ 난수 발생(정수 1부터 100까지 중에서 무작위 수 1개)

(4) Text 서랍

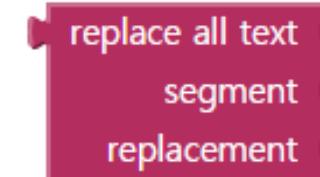
25

- 문자열 처리를 위한 **자주색** 블록

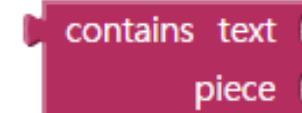
① “ ” ② trim, length



③ join, split text, replace all text



④ compare text



⑤ contains text

- 문자열 자체
- 문자열의 공백 제거나 길이
- 문자열을 연결하거나 자르거나 대체
- 문자열 비교
- 문자열 포함 비교

(5) List(목록) 서랍

26

- 여러 개의 값을 저장하는 변수 공간으로 **청록색** 블록

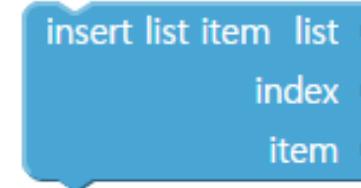
① length of list



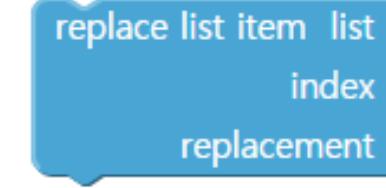
② create empty



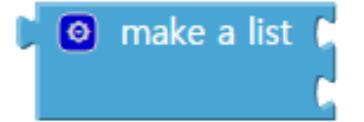
③ insert list item



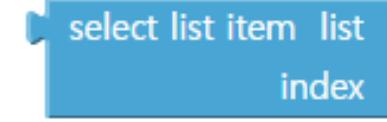
④ replace list item



⑤ make a list



⑥ select list item



- list의 크기(length)를 가져온다.
- 비어 있는 list를 만든다.
- list에서 정해진 위치(index)에 값(item)을 넣는다.
- list에서 정해진 위치(index)의 값을 다른 것(replacement)으로 대체한다.
- 소켓에 추가된 값으로 list를 만든다.
- list에서 정해진 위치(index)의 값을 가져온다.

[6] Colors 서랍

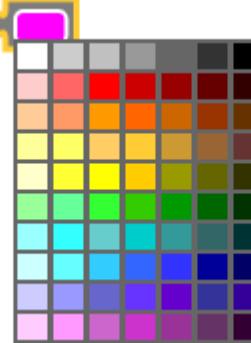
27

- 컴포넌트의 배경이나 글자색을 설정할 수 있는 회색의 색상블록

① default colors



② make color



- ① 기본 13가지 색상의 블록을 가지고 있으며, 색상 블록을 클릭하면 색상 팝업창을 띄워 색상을 선택할 수 있다.
- ② Red, Green, Blue 3가지 색상으로 혼합하여 색상을 만든다. 각각 256단계의 색을 가지고 있으므로 $256 \times 256 \times 256$ (16,777,216) 가지의 색상을 표현할 수 있다.

[7] Variables 서랍

28

- 값을 임시로 저장할 수 있는 변수를 만드는 **주황색** 블록

① initialize global ~ to



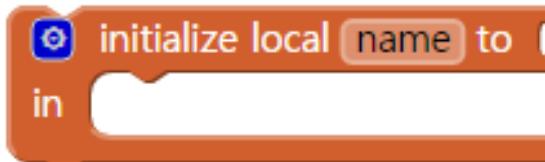
② get



③ set to



④ initialize local ~ to



⑤ initialize local ~ to



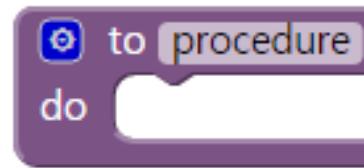
- “name”이라는 전역 저장 변수를 만든다. “name”은 변경 가능하다.
- get 뒤의 목록 변수 이름에서 저장된 값을 가져온다.
- to 소켓의 값을 set 뒤의 목록 변수 이름에 저장한다.
- “name”이라는 지역 저장 변수를 만들어 사용한다. 소켓을 끼울 수 있는 형태로 만든다. “name”은 변경 가능하다.
- “name”이라는 지역 저장 변수를 만들어 사용한다. 플러그를 끼울 수 있는 형태로 만든다. “name”은 변경 가능하다.

[8] Procedures 서랍

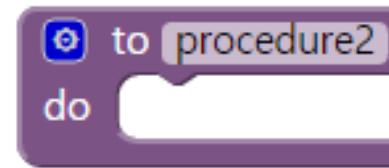
29

- 반복적으로 사용되는 기능을 모아 만든 **보라색** 블록
- 기본 내장 procedure와 사용자 정의 procedure를 생성 가능

① to ~



① to ~



call [procedure v]

call [procedure2 v]

② call ~

② call ~

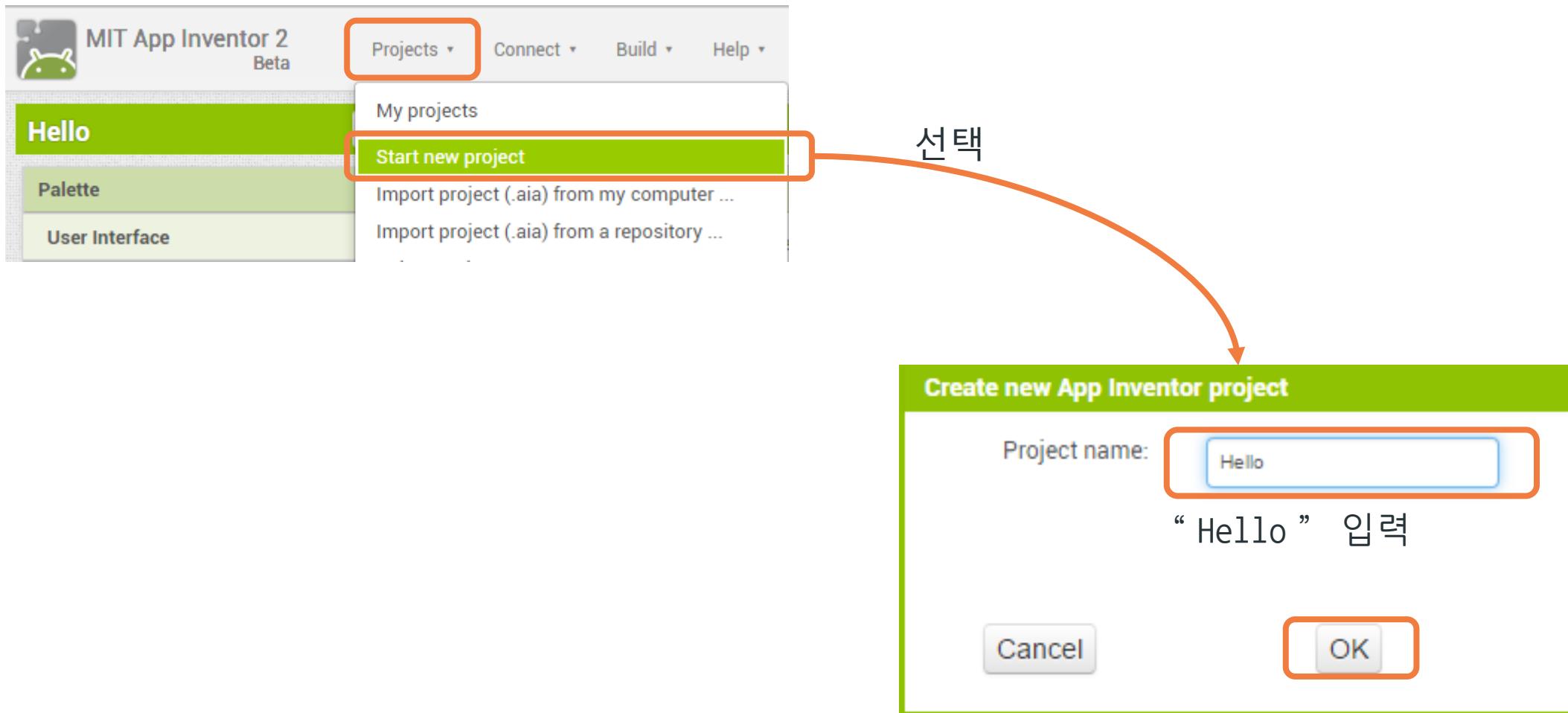
① to 뒤의 “procedure” 이름으로 프로시저를 만든다.
“procedure” 이름은 변경 가능하다.

② call 뒤의 “procedure” 이름의 프로시저를 호출하여 실행한다.

“안녕하세요”
인사말 띄우기

“안녕하세요” 인사말 띄우기 프로젝트

- “Hello”라는 프로젝트를 만든다.



Step 1. 디자이너 편집하기

32

Palette

User Interface

- Button
- CheckBox
- DatePicker
- Image
- Label
- ListView
- Notifier
- PasswordTextBox
- Slider
- Spinner
- TextBox
- TimePicker
- WebViewer

Layout

Media

Viewer

Display hidden components in Viewer

Check to see Preview on Tablet size.

Screen1

Components

Properties

Screen1

AboutScreen

AlignHorizontal
Left : 1 ▾

AlignVertical
Top : 1 ▾

AppName
Hello

BackgroundColor
□ White

BackgroundImage
None...

CloseScreenAnimation
Default ▾

Icon
None...

OpenScreenAnimation
Default ▾

ScreenOrientation
Unspecified ▾

Scollable

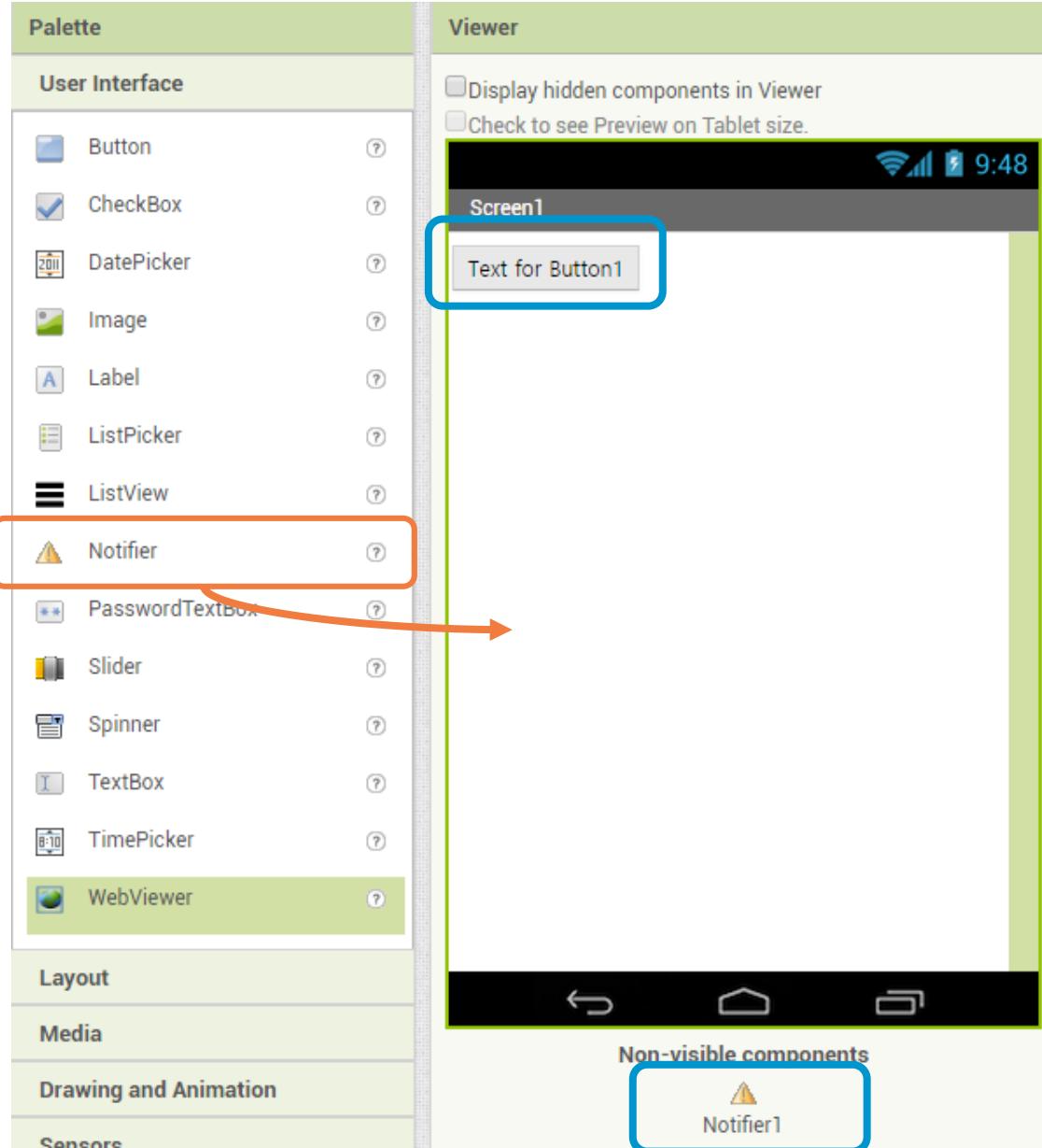
Screen1

Rename Delete

Media

The screenshot shows the App Inventor Designer interface. The left side has a 'Palette' with categories 'User Interface' and 'Layout/Media'. In 'User Interface', 'Button' is highlighted with an orange border. A red arrow points from this button icon to the 'Button' component in the 'Components' panel. The 'Components' panel shows a single component named 'Screen1'. The right side is the 'Properties' panel, which lists various properties for 'Screen1': 'AboutScreen' (empty), 'AlignHorizontal' (Left: 1), 'AlignVertical' (Top: 1), 'AppName' (Hello), 'BackgroundColor' (White), 'BackgroundImage' (None...), 'CloseScreenAnimation' (Default), 'Icon' (None...), 'OpenScreenAnimation' (Default), 'ScreenOrientation' (Unspecified), and 'Scollable' (Unspecified). The 'Properties' panel also includes 'Rename' and 'Delete' buttons at the bottom.

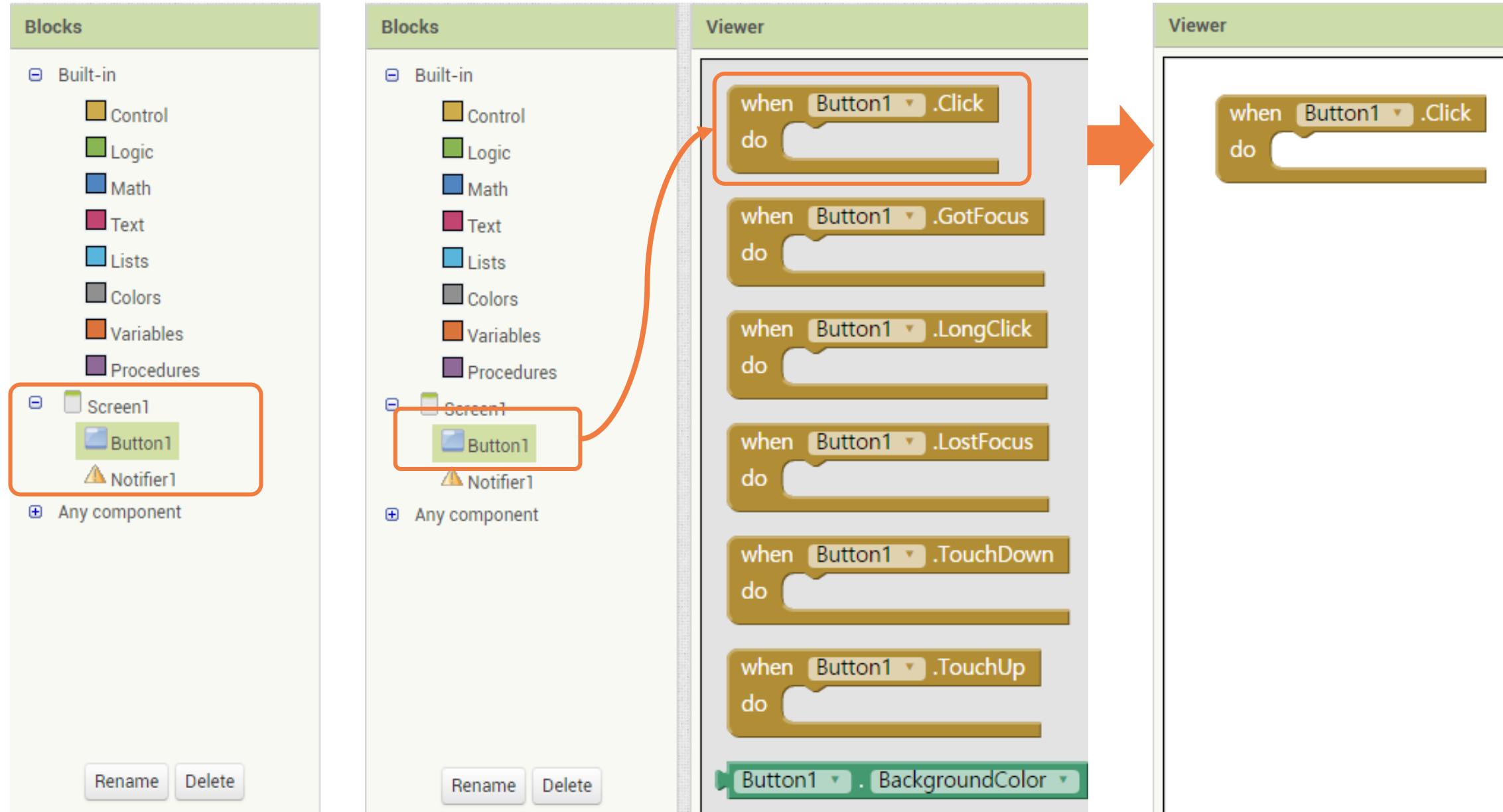
Step 1. 디자이너 편집하기



- ① 디자이너 편집기에서 Button와 Notifier 컴포넌트를 끌어와 Screen1 영역에 삽입
→ 컴포넌트들은 화면 위에서 순차적으로 표시
- ② Notifer 컴포넌트는 보이지 않는 요소이므로 화면 아래에 표시
- ③ [Blocks] 버튼을 눌러 블록 편집기로 이동한다.

Step 2. 블록 편집하기

34



Blocks

- Built-in
 - Control
 - Logic
 - Math
 - Text
 - Lists
 - Colors
 - Variables
 - Procedures
- Screen1
 - Button1
 - Notifier1
- Any component

Viewer

```

response
do when Button1 .Click
  do [
    call Notifier1 .DismissProgressDialog
    call Notifier1 .LogError
      message
    call Notifier1 .LogInfo
      message
    call Notifier1 .LogWarning
      message
    call Notifier1 .ShowAlert
      notice
    call Notifier1 .ShowChooseDialog
      message
      title
      button1Text
      button2Text
  ]
end
  
```

Viewer

```

when Button1 .Click
do call Notifier1 .ShowAlert
  notice
  
```

[when Button1.Click]
블록의 do 소켓에 키운다
(“딸깍” 소리가 난다).

플러그 끼우기
(인사말 추가하기)

Blocks

- Built-in
 - Control
 - Logic
 - Math
 - Text**
 - Lists
 - Colors
 - Variables
 - Procedures
- Screen1
 - Button1
 - Notifier1
- Any component

Viewer

```

when [Button1 v].Click
do
  call [Notifier1 v].ShowAlert
  notice [안녕하세요 v]
end
  
```

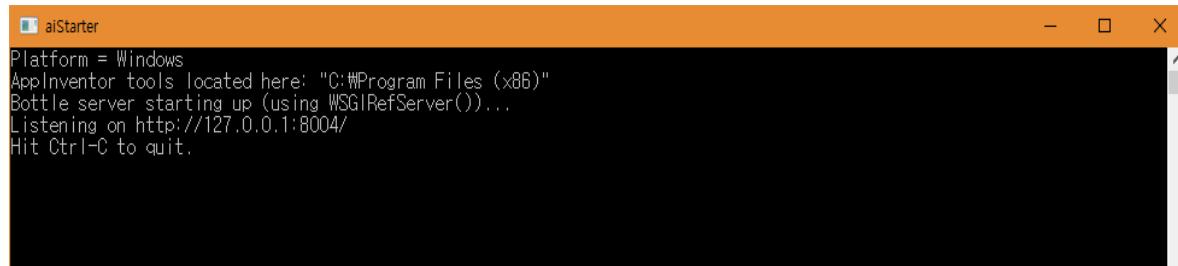
notice 소켓에 끼운다.

플러그 블록에 “안녕하세요” 입력

Congratulations Trophy

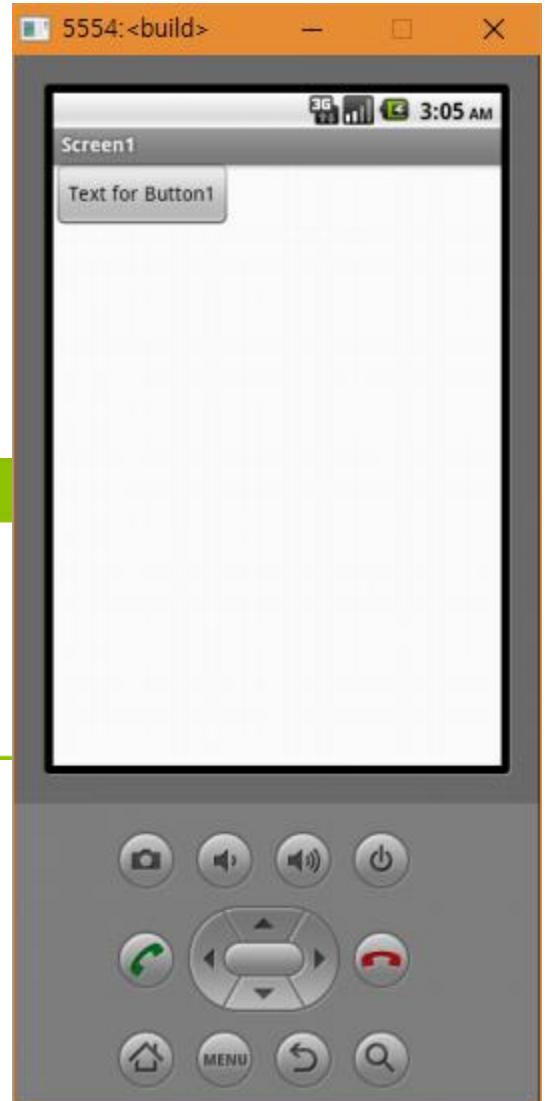
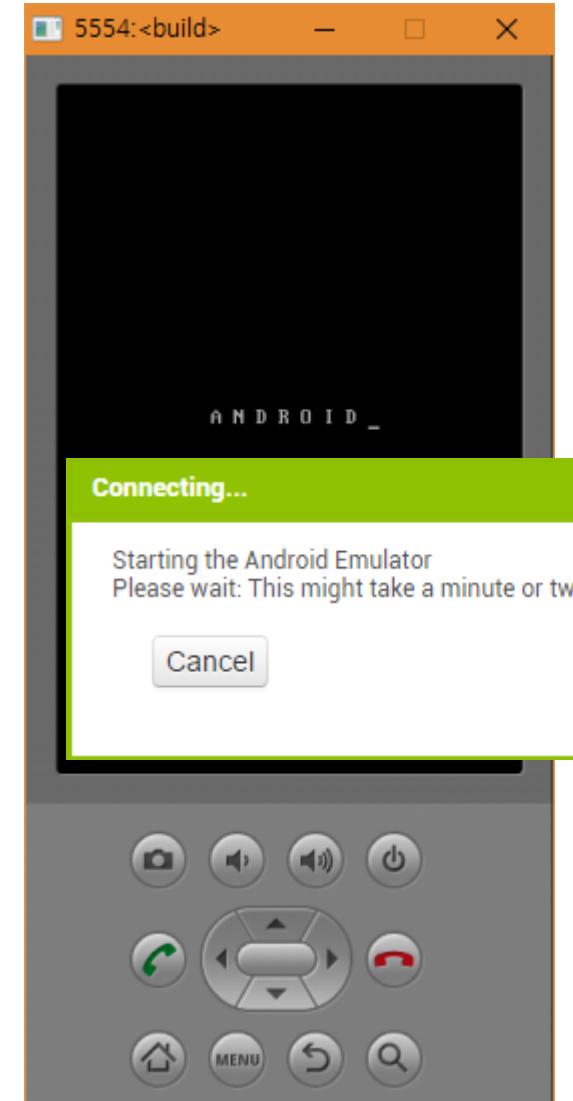
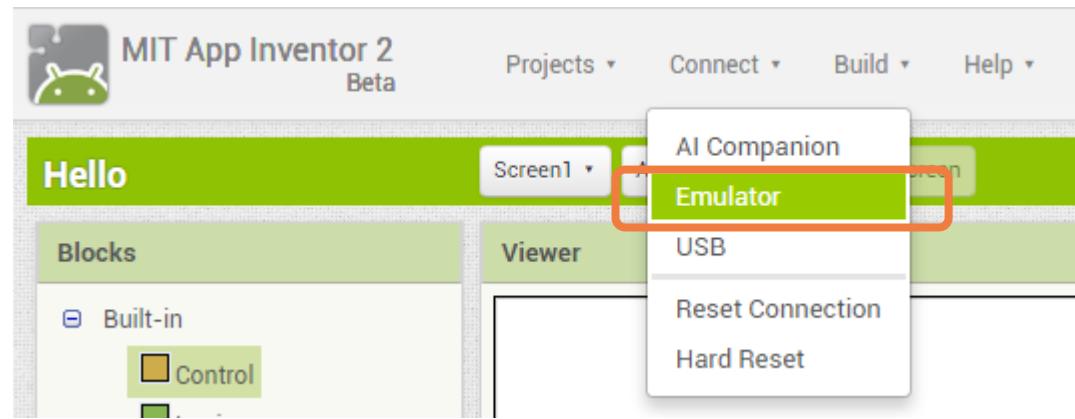
Step 3. 결과 보기 - 에뮬레이터 실행

aiStarter



```
Platform = Windows
AppInventor tools located here: "C:\Program Files (x86)"
Bottle server starting up (using MSGRefServer())
Listening on http://127.0.0.1:8004/
Hit Ctrl-C to quit.
```

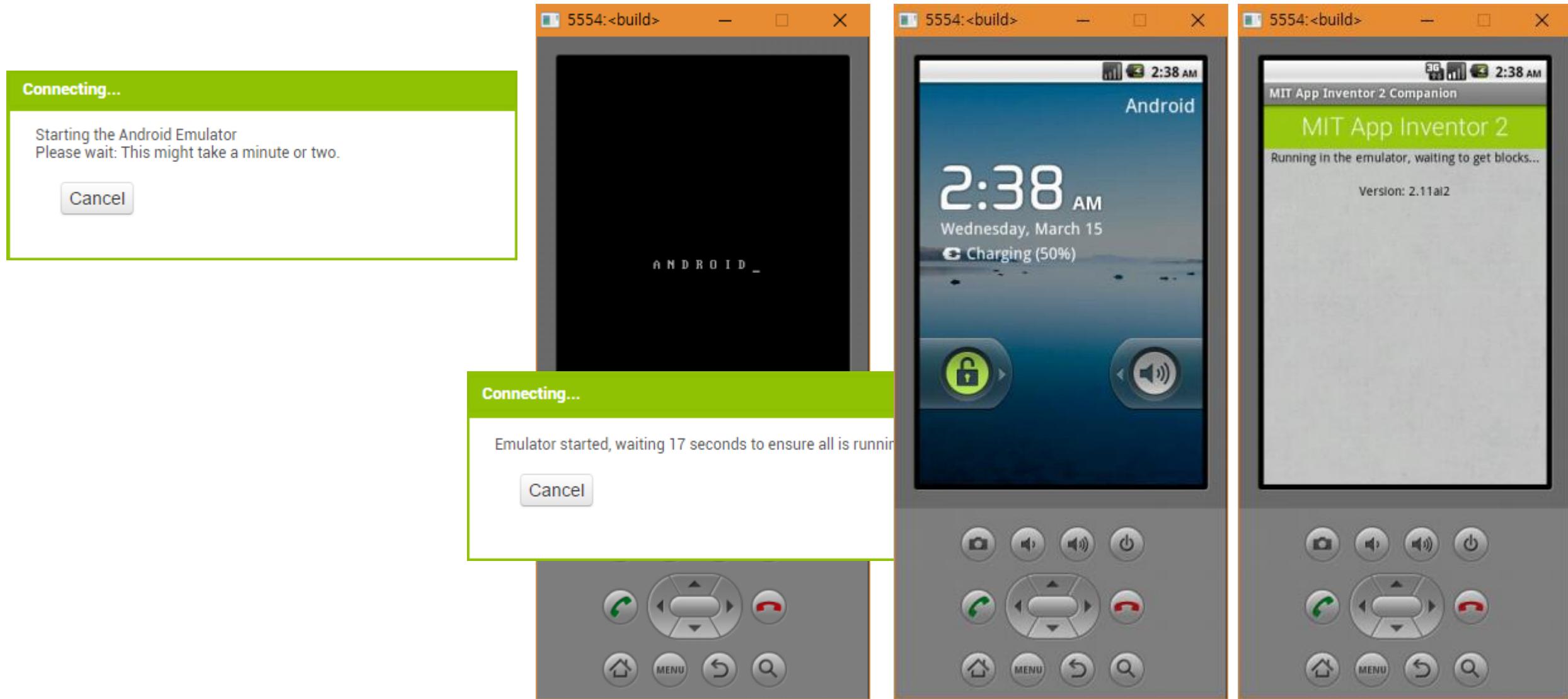
Emulator 실행



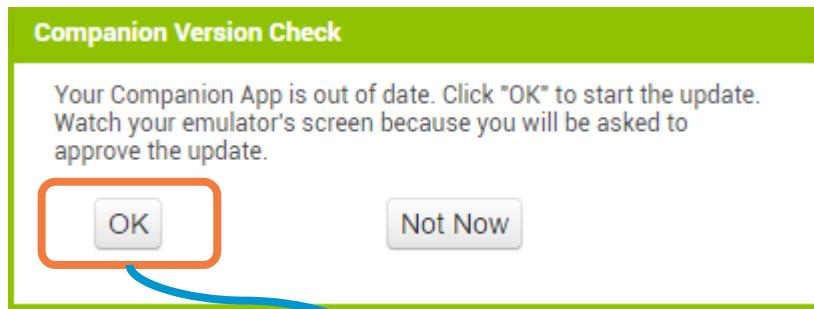
에뮬레이터의 컴파니언 앱 업데이트



컴파니언 앱의 업데이트가 발생하는 경우

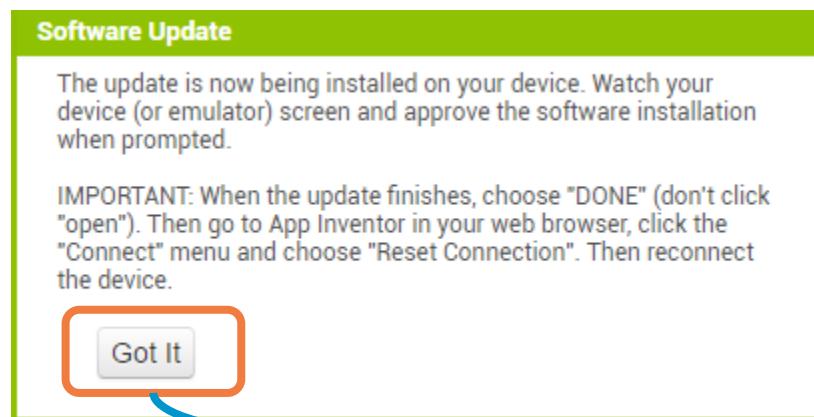


컴패니언 버전 체크



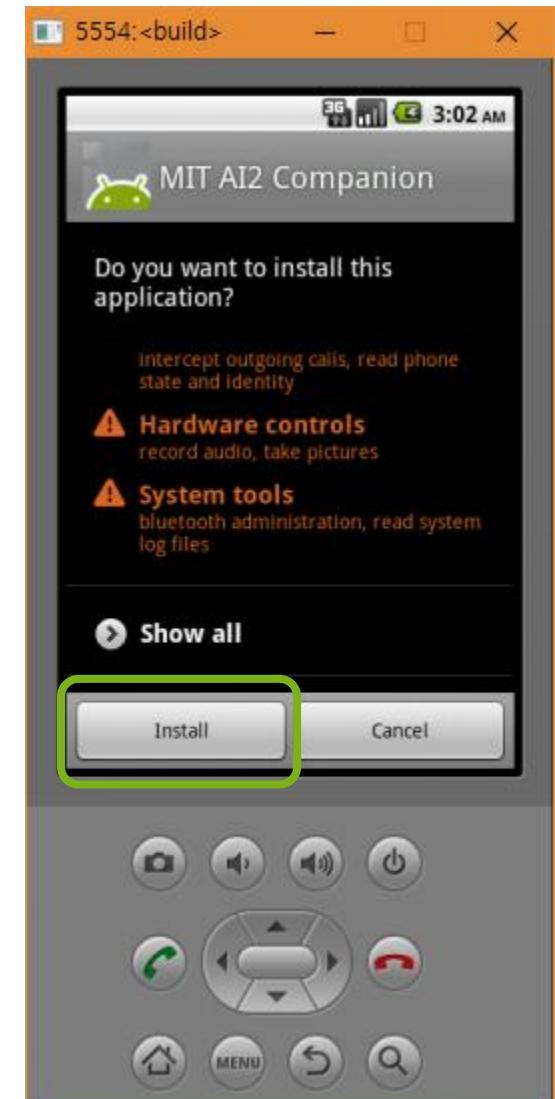
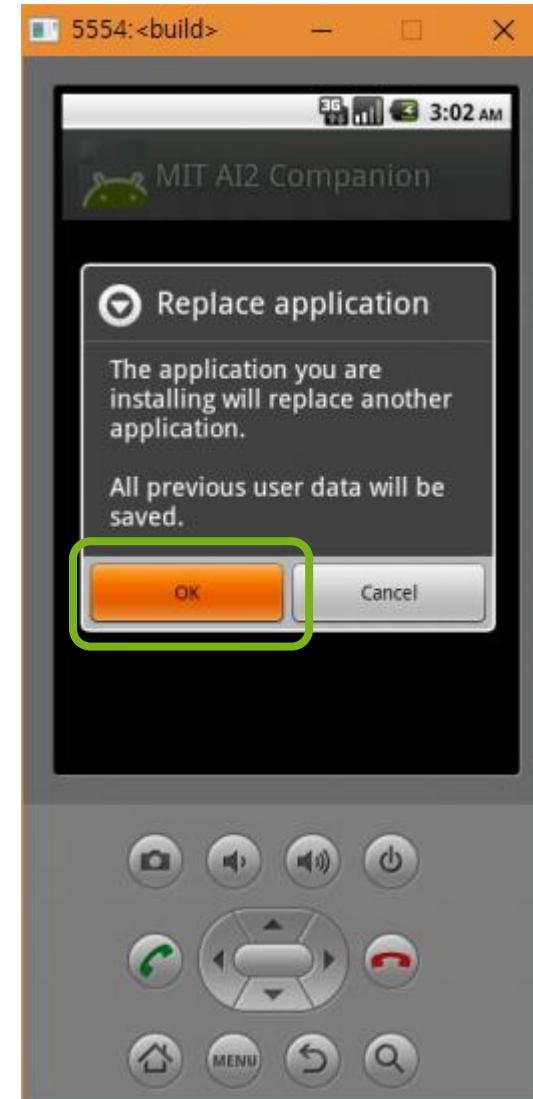
“OK” 버튼을 클릭

소프트웨어 업데이트

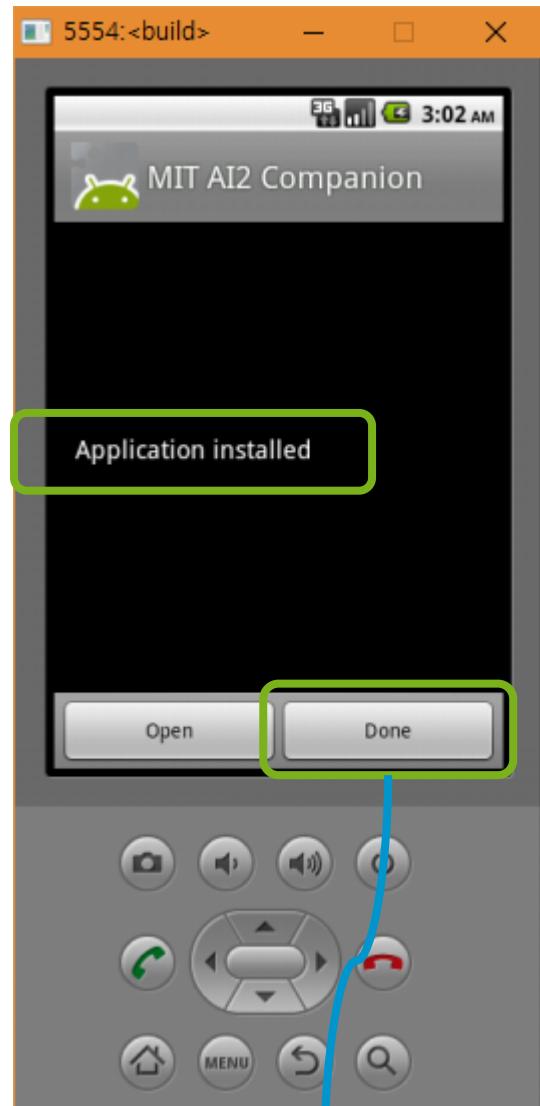


“Got It” 클릭

컴패니언 업데이트 진행...

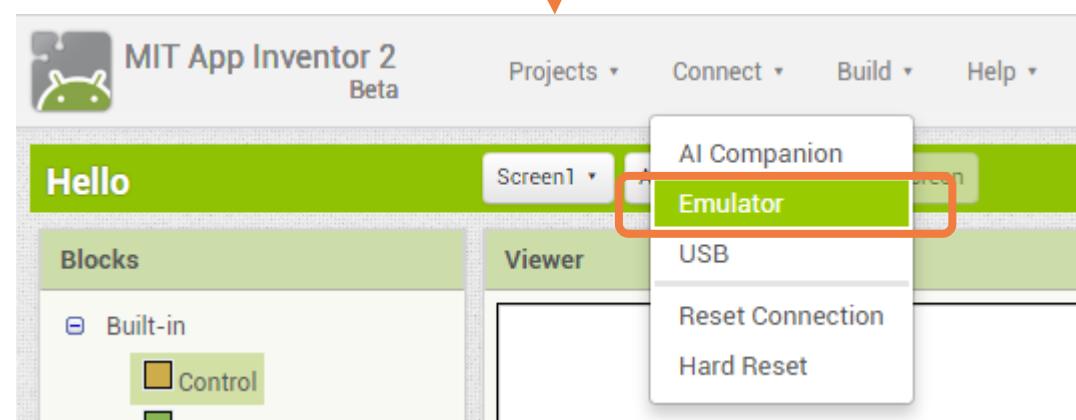
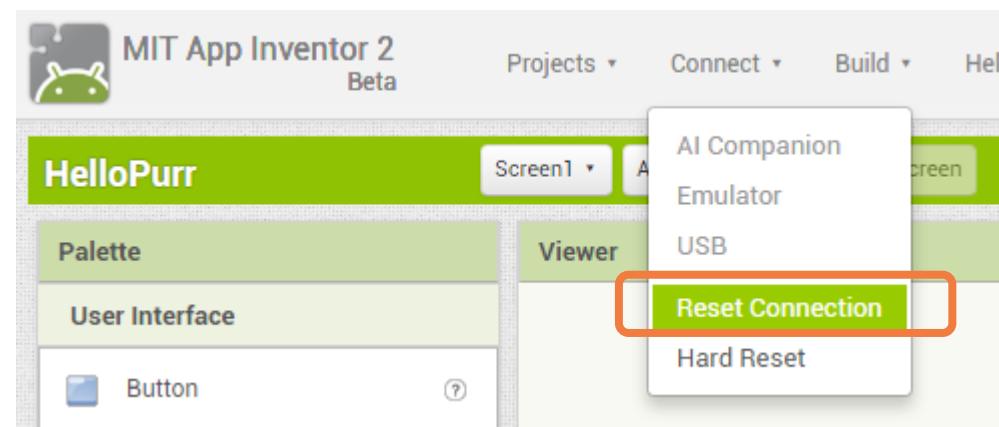


설치 완료.....



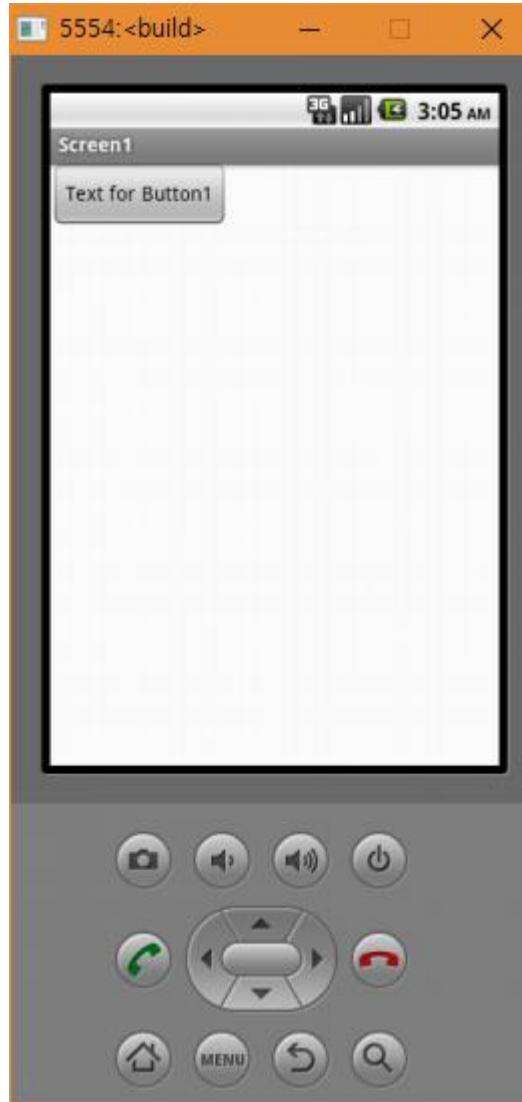
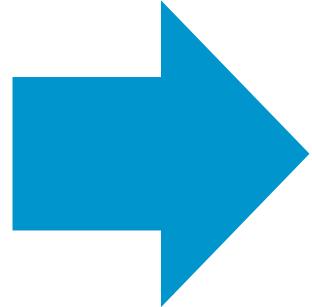
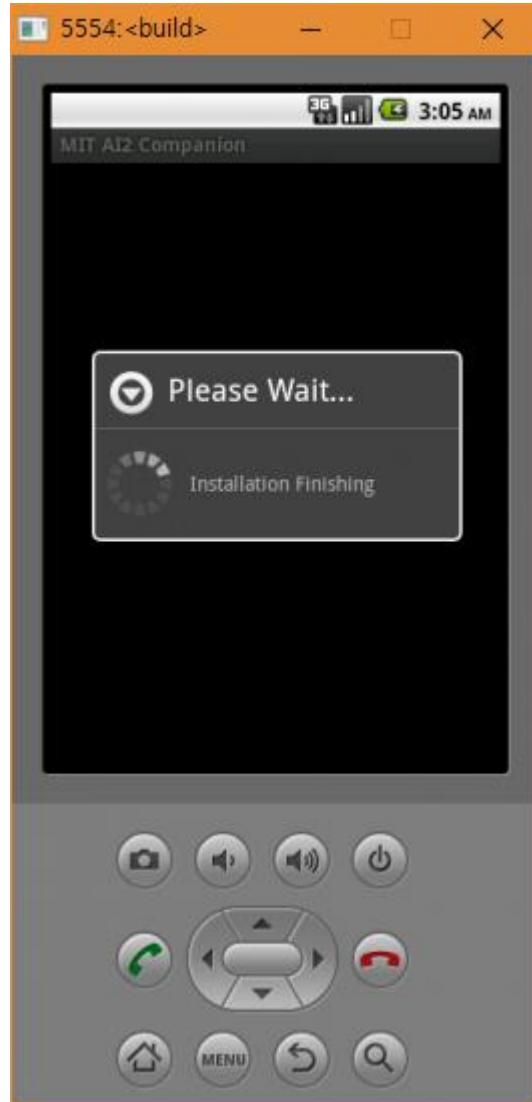
반드시 “Done” 선택

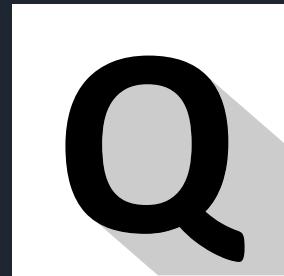
에뮬레이터 연결 리셋.....





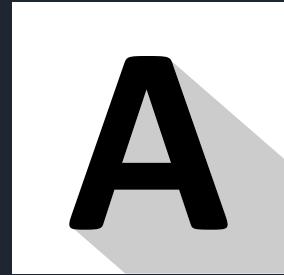
설치 마무리 후 앱 실행





question

&



answer

43

