CHAPTER 11

Text to Speech and Text Messaging

GLOBAL EDITION
Starting Out with App Inventor for Android
Tony Gaddis • Rebecca Halsey
ALWAYS LEARNING
PEARSON

Topics

- `TextToSpeech` Component
- The `Texting` Component
- Receiving Messages
- Sending Messages

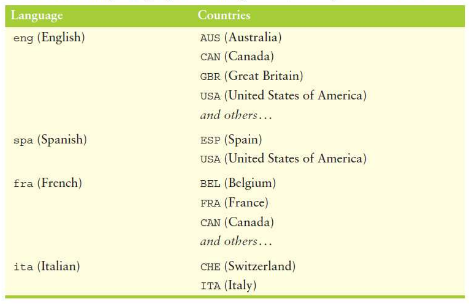
The `TextToSpeech` Component

`TextToSpeech` Component Properties

- This component uses advanced technology that allows your app to speak a block of text.
- The `TextToSpeech` component has properties that you can set for the language and country.
- To select a language, you set the `Language` property to the three-letter code that stands for that language.

The `TextToSpeech` Component

Table 11-1 Example language and country codes (*Source:* Tony Gaddis/Pearson Education, Inc.)

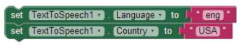


Language	Countries
eng (English)	AUS (Australia)
	CAN (Canada)
	GBR (Great Britain)
	USA (United States of America)
	and others…
spa (Spanish)	ESP (Spain)
	USA (United States of America)
fra (French)	BEL (Belgium)
	FRA (France)
	CAN (Canada)
	and others…
ita (Italian)	CHE (Switzerland)
	ITA (Italy)

The TextToSpeech Component

Use a `text` block to set the value of the `Language` and `Country` properties.

Figure 11-1 Setting Language and Country in Code (*Source:* MIT App Inventor 2)


set TextToSpeech1 . Language to "eng"
set TextToSpeech1 . Country to "USA"

The TextToSpeech Component

Pitch and Speech Rate

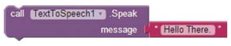
- The `pitch` property will lower or raise the pitch of the speech based on a number between 0 and 2.
- If set to zero, the voice is low.
- The `speechrate` property will slow down or speed up the rate based on a number between 0 and 2.
- If set to zero, the rate is slow.

The TextToSpeech Component

The `Speak` Method

- The `TextToSpeech.Speak` method makes the app speak.
- It has one argument message.

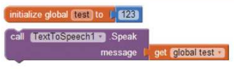
Figure 11-3 `Speak` Method with Literal Text (*Source:* MIT App Inventor 2)



The TextToSpeech Component

- Figure 11-4 demonstrates how to use a global variable for the `Speak` method.
- If you were to use a math expression such as 5*5, the result would be for the app to speak "twenty-five".

Figure 11-4 `Speak` Method with Variable Data (*Source:* MIT App Inventor 2)



The `TextToSpeech` Component

`TextToSpeech` Events

- The `TextToSpeech` component has two events, `BeforeSpeaking` and `AfterSpeaking`.
- They are self-explanatory.

Texting Component

- App Inventor provides blocks allowing us to program apps that send and receive text messages.
- For best results, use a device for apps built with the `Texting` Component.
- If you have a Google Voice account, the emulator will work.
- For more information about Google voice see https://support.google.com/voice/answer/115061?hl=en

Texting Component

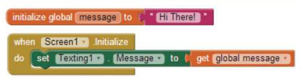
- The Texting Component is found in the Social palette.
- It has one method, one event, and just a few properties.

Texting Component

Texting Component Properties

- The `Message` property holds the message text that `SendMessage` will send.
- Before sending a message, you set the `Message` property to a value that can be literal or variable data.

Figure 11-15 Texting `Message` Property (*Source:* MIT App Inventor 2)



```
initialize global message to " Hi There! "
when Screen1 .Initialize
do  set Texting1 . Message to  get global message
```

Texting Component

Texting Component Properties

- The `PhoneNumber` property holds the number that the `SendMessage` method will use.
- This property can include only digits, dashes, dots, and parentheses.
- Figure 11-16 shows a valid phone number.

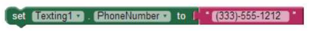


Figure 11-16 PhoneNumber Property (*Source:* MIT App Inventor 2)

```
set Texting1 . PhoneNumber to " (333)-555-1212 "
```

Texting Component

Texting Component Properties

The `ReceiveEnabled` property takes the numeric values 1, 2, and 3. They are defined as follows:

1-Off – the app will ignore all messages.

2-Foreground – the messages will be received when the app is running.

3-Always – the app will receive the messages while running and queue the messages if it is not running.

Texting Component

Texting Component Properties

Figure 11-17 shows a combination of blocks you might use to set a “do not disturb” feature.

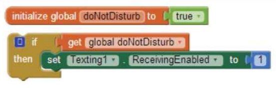


Figure 11-17 `ReceivingEnabled` Property (*Source:* MIT App Inventor 2)

```
initialize global doNotDisturb to true
if get global doNotDisturb
then set Texting1 . ReceivingEnabled to 1
```

Texting Component

Send Message Method

- The `Texting` component has one method, `SendMessage`.
- It is important to set both the `PhoneNumber` and message properties before calling the `SendMessage` method.

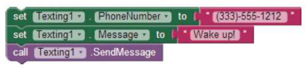


Figure 11-18 Using the `SendMessage` Method (*Source:* MIT App Inventor 2)

```
set Texting1 . PhoneNumber to " (333)-555-1212 "
set Texting1 . Message to " Wake up! "
call Texting1 .SendMessage
```

Texting Component

The `ReceiveMessage` Event

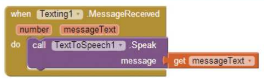
- The `ReceiveMessage` event is triggered by a text message coming into your device.
- This event will listen for text messages when the app is active or dormant.

Texting Component

The `ReceiveMessage` Event

- `ReceiveMessage` allows you to program behavior when a text message comes in.
- Use this event to filter incoming messages.
- For example, you can use a `TextToSpeech` component to program the app so it will speak text messages from family members and ignore all others.

Figure 11-19 `MessageReceived` Example (*Source:* MIT App Inventor 2)



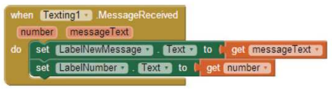
```
when Texting1 .MessageReceived
  number  messageText
do  call TextToSpeech1 .Speak
                  message  get messageText
```

Receiving Text Messages

- If you'd like your app to "do" something when text messages comes in, add the `Texting` Component and use the `MessageReceived` event handler.
- The event handler has two arguments passed to it, `number` and `messageText`.
- The `number` stores the phone number from which the message was sent.
- The `messageText` is the text that was sent.

Receiving Text Messages

Figure 11-20 `MessageReceived` Event Handler (*Source:* MIT App Inventor 2)

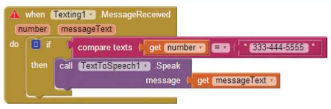


```
when Texting1 .MessageReceived
  number  messageText
do  set LabelNewMessage . Text to  get messageText
    set LabelNumber . Text to  get number
```

Receiving Text Messages

- Figure 11-21 shows the `MessageReceived` event handler you will create in Tutorial 11-2.
- An `if then` block is used to evaluate who the text message is from.
- If it meets the condition, then use a `TextToSpeech` component.

Figure 11-21 Receiving Text (*Source:* MIT App Inventor 2)



```
when Texting1 .MessageReceived
  number  messageText
do  if    compare texts  get number  =  " 333-444-5555 "
    then  call TextToSpeech1 .Speak
                            message  get messageText
```

Sending Text Messages

Sending a text message is a process of telling the app:

- *Who* to send it to.
- *What* message to send.
- *Calling* the `SendMessage` method.

Sending Text Messages

- Let's look at an example of an app that sets up a list of numbers belonging to a group.
- The app will use a `foreach` loop to iterate through the list of numbers and send the message to each number. See Figures 11-29 and 11-30.

Sending Text Messages

Figure 11-29 User Interface (*Source:* MIT App Inventor 2)

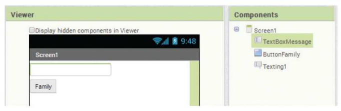
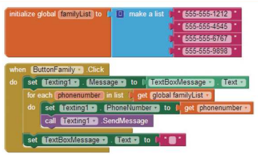


Figure 11-30 Blocks Editor Workspace (*Source:* MIT App Inventor 2)



```
initialize global familyList to  make a list  " 555-555-1212 "
                                              " 555-555-4545 "
                                              " 555-555-6767 "
                                              " 555-555-9898 "

when ButtonFamily .Click
do  set Texting1 . Message to  TextBoxMessage . Text
    for each phonenumber in list  get global familyList
    do  set Texting1 . PhoneNumber to  get phonenumber
        call Texting1 .SendMessage
    set TextBoxMessage . Text to  " "
```

Sending Text Messages

- See that we set the `Texting` component's `Message` property to the `Text` property of the `TextBoxMessage` textbox.
- Set the `Texting` component's `PhoneNumber` property to an element in the list.
- Once we have those two things then call `SendMessage`.