# Messenger Bot Tutorial

Rojiku, OtakoidTony, Park Hyun

January 14, 2020

Tutorials based on AutoReplyBot using javascript.

# Chapter 1

## 1.1 Setup

챗봇을 작성하기에 앞서 구글 플레이스토어에서 아래 나열한 어플리케이션을 모두 설치를 하여라. 만약에 휴대폰 운영체제가 Android계열이 아닌 경우에는 컴퓨터를 이용해 Android을 실행하여 챗봇을 구현할 수 있다. 컴퓨터가 만일 Odroid[1]를 사용중이라면 스마트폰처럼 사용하여도 문제는 없다.rrrrr

**MessengerBot**
com.xfl.kakaotalkbot

**Wear OS by Google**
com.google.android.wearable.app

[1]Odroid : Single Board Computer 중 하나로 Android OS 가 주로 사용되며 이 때문에 Android 관련 개발시 주로 사용되는 컴퓨터이다.

두 어플리케이션의 관계에 대해 설명하자면 MessengerBot에서 Wear OS by Google을 이용해 메신저에 접근하는 관계라 할 수 있으리라. 독자도 알다시피 Wear OS by Google라는 어플리케이션은 스마트폰의 알림을 Wearable Device에 전송하는 기능과 스마트폰 대신 Wearable Device만을 이용해 메신저에 답장을 보내는 기능이 탑재되어 있다. 이를 이용하여 MessengerBot에서 Wear OS by Google를 이용해 자동응답을 할 수 있도록 구현한 것이다. 두 어플리케이션에 관한 설명은 차후 이어서 하도록 하고, 만약에 두 어플리케이션 모두 설치를 하였다면, 손 아이콘을 눌러 알림권한설정에서 앞에서 설치한 MessengerBot과 Wear OS by Google를 활성화를 하여라. 이걸로 준비 작업을 마친다.

## 1.2 Make your script

준비 작업을 마친 후, 우측 하단의 십자버튼을 누르고 독자가 원하는 스크립트 이름을 정하여라. 저자의 경우 'normal'이라고 정하였다. 아래 소스는 스크립트를 처음 작성할 때 자동으로 생성되는 스크립트이며, 본 책에서는 여러 이유로 주석을 제거한 상태이다.

```
1 const scriptName="normal.js";
2
```

```
3  function response(room, msg, sender, isGroupChat,
       replier, ImageDB, packageName, threadId){
4
5  }
6
7  function onStartCompile(){
8
9  }
10
11 function onCreate(savedInstanceState,activity) {
12     var layout=new android.widget.LinearLayout(
           activity);
13     layout.setOrientation(android.widget.
           LinearLayout.HORIZONTAL);
14     var txt=new android.widget.TextView(activity);
15     txt.setText("액티비티 사용 예시입니다.");
16     layout.addView(txt);
17     activity.setContentView(layout);
18 }
19 function onResume(activity) {}
20 function onPause(activity) {}
21 function onStop(activity) {}
```

Listing 1.1: normal.js

우리는 개발을 그렇게 깊게 하지는 않을 것이기 때문에 몇 줄을 삭제하고자 한다. 내가 챗봇을 개발할 때는 자동으로 작성된 함수 중에서는 오직 response()함수만을 이용했다.

```
1  function response(room, msg, sender, isGroupChat,
       replier, ImageDB, packageName, threadId){
2
3  }
```

이제 한결 나아진 것 같다. 이제, Hello World! 예제를 작성해보자.

## 1.3 Hello World!

```
1 function response(room, msg, sender, isGroupChat,
      replier, ImageDB, packageName, threadId){
2     replier.reply("Hello World!")
3 }
```

위와 같이 작성한 다음, 컴파일을 하여라. 컴파일은 저장버튼을 길게 누르면 할 수 있으며, 다른 방법으로는 스크립트가 나열된 메인화면에서 새로고침버튼을 누르면 컴파일할 수 있다. 컴파일을 하였으면, 메인화면으로 나와 벌레버튼을 눌러 Debug Room으로 진입하여라. 우선은, Debug Room에서 정상적으로 작동하는지를 살펴볼 것이다. Debug Room에 진입하였으면 아무 말이나 입력하고 반응을 살펴보자. 그러면, 독자가 무엇을 입력하든간에 "Hello World!"라고만 응답하는 것을 볼 수 있을 것이다. 아직까지는 기능이 없는데다가 누가 무슨 말을 하든간에 "Hello World!"라고 메시지를 전송하는 문제가 다분한 챗봇이다. 따라서 이 봇이 특정 키워드에만 반응을 하도록 수정을 해야한다. 어떻게 하면 특정 키워드에만 챗봇이 작동할 수 있을까? 이에 대해서 다음 장에서 설명하고자 한다.

# Chapter 2

## 2.1 Specifical Replication

우리는 앞에서 챗봇이 무슨 말을 하든간에 "Hello World!" 라고 메시지를 전송하는 문제를 마주하였다. 하지만, 이는 if 문이라는 것을 이용하여 쉽게 해결할 수 있다. if문은 매우 많이 사용되기에 반드시 알아야하는 구문 중 하나이다. 다음은 if문을 이용한 매우 간단한 예시이다.

```
1 if (true) {
2     print("Hello World!");
3 }
```

이것 뿐이다. 조건이 참(true)이기 때문에 당연히 Hello World! 를 출력할 것이다. 이를 이용하여 우리는 위에서의 문제를 해결할 수 있다. 다음 따라오는 스크립트를 따라해보자.

```
function response(room, msg, sender, isGroupChat,
    replier, ImageDB, packageName, threadId){
    if (msg=='Hello'){
        replier.reply("Hello! Nice to meet you!");
    }
}
```

그러면, 발신자가 오직 'Hello'라고 발신하였을 때만, 챗봇은 'Hello! Nice to meet you!' 라고 답장할 것이다. 이와 같이 조건문을 사용하여 발생할 수 있는 상황에 대해 꾸준히 작성하면 챗봇도 같이 점차 적절한 발화를 할 수 있게 될 것 이다. 이 책에서는 조건문에 대해서 따로 구문을 설명하지는 않으니, 이 책에서 나와있는 예제들을 통해 몸으로 익히도록 하자. 물론, 따로 프로그래밍 관련 서적을 참고하는 것도 매우 좋다. 아니, Best다!

## 2.2 Long Scripts Problem

조건문을 이용하여 열심히 챗봇을 개선해 나가다 보면 다음과 같은 코드를 만나기 십상이다.

```
function response(room, msg, sender, isGroupChat,
    replier, ImageDB, packageName, threadId){
    if (msg=='Hello'){
        replier.reply("Hello! Nice to meet you!");
    }
    if (msg=='loli is'){
        replier.reply("LIFE!!!");
    }
```

```
8       if (msg=='XD'){
9           replier.reply("lol");
10      }
11  }
```

단지 3가지 경우에 적절히 응답하도록 만든 간단한 챗봇임에도 불구하고 11줄이나 차지하는 너무 더럽다고 해도 과언이 아닌 스크립트가 되어 있다. 하지만, 자바스크립트에서는 파이썬과는 달리 줄에 대해서 관대하기에 다음과 같이 쓸 수도 있다.

```
1   function response(room, msg, sender, isGroupChat,
        replier, ImageDB, packageName, threadId){
2       if (msg=='Hello'  ){replier.reply("Hello! Nice
            to meet you!");}
3       if (msg=='loli is'){replier.reply("LIFE!!!");}
4       if (msg=='XD'     ){replier.reply("lol");}
5   }
```

위와 같이 작성하면 5줄만으로도 작성할 수 있다. 그러나, 컴퓨터에서는 사실상 같은 스크립트일 것이며, 이 스크립트 또한 응답할 상황이 많아지면 많아질 수록 스크립트의 내용은 방대해 질 것 이다. 이를 해결하기 위해서는 어떻게 작성을 하면 좋을까? 이에 대한 답은 객체(혹은 딕셔너리)를 이용하는 것이다.

```
1   var msgDict = {'Hello': "Hello World!", 'loli is':
        "LIFE!!!", 'XD': "lol"};
2   function response(room, msg, sender, isGroupChat,
        replier, ImageDB, packageName, threadId){
3       if (msg in msgDict){
```

```
4           replier.reply(msgDict[msg]);
5       }
6   }
```

물론 독자가 Switch문에 대해서 알고 있다면, 다음과 같이 작성할 수는 있다.

```
1   function response(room, msg, sender, isGroupChat,
        replier, ImageDB, packageName, threadId){
2       switch(msg){
3           case 'Hello' :
4               replier.reply("Hello! Nice to meet you
                    !");
5               break;
6           case 'loli is' :
7               replier.reply("LIFE!!!");
8               break;
9           case 'XD' :
10              replier.reply("lol");
11              break;
12      }
13  }
```

음... 그냥 객체를 이용하자. 객체에 대한 내용은 차후에 다시 언급될 것이고 무엇보다 매우 편리한 것이니 반드시 숙지해두자! 이것으로 단순한 자동응답 챗봇은 구현할 수 있게 되었다. 이제 본격적으로 챗봇을 개발해보자! 다음 챕터부터는 챗봇에 여러 기능을 추가하게 될 것이다.

혹시나 한시라도 빨리 챗봇을 만들어 실행시켜보고 싶은 독자를 위해 실행하는 방법을 적어보자면, 메인화면에서 해당 스크립트에 토글 스위치를 누르고 메인화면 상단에 있는 토글 스위치를 눌러 활성화하면 해당 스크립트를 활성화할 수 있다. 단, 메신저를 사용 중일 때는 자동응답을 할 수 없다.

# Chapter 3

## 3.1 Random Replication

만약에 발신자가 '뭐임'라고 할 때마다 '글쎄'라고 챗봇이 답장을 하게 되면, 답장자가 챗봇이라는 느낌을 강하게 받을 뿐만 아니라 if문에 대해서 알고 있는 발신자 또한 개발자라면, 바로 if문을 이용해 구현했다는 것을 알아낼 것이고 너무 무분별한 답장기능은 때로는 특정 채팅방으로부터 강퇴 사유가 되기도 한다. 따라서 각 상황별 여러가지 답장을 할 수 있도록 구현해야 한다. 이 때 사용되는 것이 바로 Math라는 라이브러리이다. 맞다. 독자가 매우 싫어하는 과목, 그 Math이다. 물론, 수학을 보고 흥분하는 변태일 가능성도 있긴 하다. 어찌됐든, 이상한 소리는 여기까지하고 하던 말을 이어가자면 Math라는 라이

브러리에는 당연히 수학에 관한 함수나 여러 기능이 포함되어 있는데 이를 이용하여 챗봇이 랜덤하게 답장을 하도록 구현할 것이다. 우선 `Math.random()`라는 함수는 0부터 1사이에서의 랜덤한 수를 제공한다. 이 기능은 공학용 계산기에서의 Ran♯과 동일하다. 이를 이용하여 1차원 배열에서 랜덤한 성분을 뽑아내는 함수를 다음과 같이 구현할 수 있다.

```
function randomItem(a) {
    return a[Math.floor(Math.random() * a.length)
        ];
}
```

Listing 3.1: `randomItem(array)`

이제 다음과 같이 작성하자.

```
function randomItem(a) {
    return a[Math.floor(Math.random() * a.length)
        ];
}
var msgDict = {
    '안녕하세요': [
        "안녕하세여~!",
        "어서와여~!",
        "안녕~ 이에여~!"
    ],
    '냥!': [
        "냐아앙~!",
        "냥~! 냥~!",
        "냐아~!"
    ],
    '박제': [
        "어휴;;;",
        "증! 거! 확! 보!"
    ]
```

```
19 };
20
21 function response(room, msg, sender, isGroupChat,
       replier, ImageDB, packageName, threadId){
22     var probability = Math.random() * 100;
23     if (probability <= 70){
24         if (msg in msgDict){
25             replier.reply(randomItem(msgDict[msg])
                   );
26         }
27     }
28 }
```

Listing 3.2: RandomReply.js

이와 같이 작성을 하면, 입력이 들어 올 때마다 응답을 하지 않고 70%확률로 미리 정한 1차원 배열 중 랜덤으로 답장을 하게 될 것이다. 우선 msgDict라는 객체를 선언한 것에 대해 설명을 하자면 msgDict는 객체(혹은 딕셔너리)이므로 msgDict['박제'] 처럼 사용할 수 있으며 이 때 해당하는 값은 크기가 2인 1차원 배열, ["어휴;;;", "증! 거! 확! 보!"]가 될 것이다. 이것만으로는 설명이 매우 부족한 관계로 위 스크립트를 분석해보자.

```
1 function response(room, msg, sender, isGroupChat,
      replier, ImageDB, packageName, threadId){
2     var probability = Math.random() * 100;
3     if (probability <= 70){
4         if (msg in msgDict){
5             replier.reply(randomItem(msgDict[msg])
                  );
6         }
7     }
```

```
8 }
```

무엇이 되었든간에 챗봇에 있어서 메인이 되는 함수는 위 함수, response()일 것이다. 우선, 발신자가 '안녕하세요' 라고 보냈다고 가정하자. 그러면 msg = '안녕하세요'일 것이다. 이제 계속해서 response()내부에 선언된 변수 probability에 대해서 살펴보면 probability=Math.random()*100 이다. 이 때, Math.random()의 값이 0.26886278881818315가 나왔다고 하자. 그러면 이 때는(왜냐하면 random()은 매 호출시마다 값이 바뀌기 때문이다.) probability의 값은 대략 26.xx일 것이다. 따라서 probability<=70의 값은 true이며 그러므로 다음 절차를 밟게 될 것이다.

```
1 if (msg in msgDict){
2     replier.reply(randomItem(msgDict[msg]));
3 }
```

이제 msg in msgDict의 값을 알아보자면, 결론적으로는 true이다. 그 이유는 앞에서 선언한 msgDict에 '안녕하세요' 라는 값이 존재하기 때문이다. 마찬가지로 msg='냥!' 이었으면, 이 때도 동일하게 true이다. 반대로 msg='안녕' 인 경우에 msg in msgDict의 값은 false이다.

```
1 var msgDict = {
2     '안녕하세요': [
3         "안녕하세여~!",
4         "어서와여~!",
```

```
5           "안녕~ 이에여~!"
6       ],
7       '냥!': [
8           "냐아앙~!",
9           "냥~! 냥~!",
10          "냐아~!"
11      ],
12      '박제': [
13          "어휴;;;",
14          "증! 거! 확! 보!"
15      ]
16  };
```

Listing 3.3: `msgDict`

이제 if문에서 조건이 참임을 알았으니, 이제 안에 작성된 내용을 수행하는 것만 관찰하면 된다.

```
1  replier.reply(randomItem(msgDict[msg]));
```

`msg`='안녕하세요' 이므로 `msgDict[msg]` = `msgDict['안녕하세요']`이라 할 수 있다. 따라서 `msgDict[msg]`의 값은 크기가 3인 배열, ["안녕하세여~!", "어서와여~!", "안녕~ 이에여~!"] 이다. 그렇다면 도대체 `randomItem(msgDict[msg])`의 값이 무엇일까? 다시 위에서 정의했던 `randomItem()`을 살펴보자.

```
1  function randomItem(a) {
2      return a[Math.floor(Math.random() * a.length)
           ];
3  }
```

Listing 3.4: `randomItem(array)`

혹시나 `Math.floor(x)`에 대해 궁금할 독자를 위해 설명하자면 다음과 같다.

$$\begin{aligned} \texttt{Math.floor}(x) &= \lfloor x \rfloor &&= [x] \\ &= \max\{n \in \mathbb{Z} \colon n \leq x\} &&= \operatorname{int}(x) \end{aligned}$$

`randomItem(array)`를 살펴보면, `array`의 원소 중 `floor(random()×array.length]`+1번째 원소를 내보낸다는 것을 알 수 있다. 따라서 `randomItem(msgDict[msg])`의 값은 `msgDict[msg]`의 원소 중 하나일 것이며, 그 값 또한 매 호출시마다 바뀔 것이다. 물론, 이 때도 `msgDict[msg]`의 원소 중 하나이다. 본 스크립트는 이제 어느정도 살펴본 것 같으니 여기서 Random Replication 섹션를 마치도록 하겠다.

## 3.2 Calculator

이번에는 `eval()`함수를 이용하여 계산기를 만들어 보고자 한다. 계산기 프로그램을 구현시 계산식을 처리하는 알고리즘에 대해서 숙지를 해야 정석이나, 이 섹션에서는 `eval()`를 이용하여 간단히 구현할 것이다. 다음과 같이 작성해보자.

```
1 function response(room, msg, sender, isGroupChat,
      replier, ImageDB, packageName, threadId){
2     replier.reply(eval(msg));
```

```
3 }
```

이것만으로도 계산기 챗봇을 구현한 것이다. 다만, 매우 위험한 것 뿐이다. 발신자가 위험한 것이 아니라, 개발자, 즉 독자가 위험하다는 것이다! 그 이유는 `eval()`에 있다. `eval()`는 매우 편리한 함수이자, 매우 치명적인 독약과도 같은, 양날의 검이라 할 수 있다. `eval()`는 매개변수를 실행하여 나온 값을 내보낸다. 예를 들어, 매개변수로 3+6을 넣으면 9가 나오는 매우 편리한 함수이다. 겨우 사칙연산? 아니다. 내장 함수도 사용할 수 있다! 그 뿐 만이 아닌 무엇이 됐든지간에 실행을 할 수 있는 매우 무시무시한 함수이다. 이 때문에 eval은 evil이라는 말도 나올 정도이다. (갑자기 tiny evil이 생각나는 것은 왜일지...) 아무튼 evil... 아니 `eval()`는 사용할 때 매우 조심해야 하기에 안전하게 `eval()`를 사용해야 한다. 본 섹션에서는 안전하게 `eval()`를 사용하는 법도 또한 매우 간단한 알고리즘을 통해 구현할 것이다. 어떻게 하면 발신자가 함수를 사용하지 못하게 할 수 있을까? 이는 다음과 같은 알고리즘을 통해 구현할 수 있다.

**Algorithm 1** Using Whitelist

```
1: procedure isFine(sentence)
2:     Whitelist ← {Allowed Words}
3:     for i ∈ Whitelist do
4:         sentence ← sentence.replaceAll(i, Blank)
5:     if sentence = Blank then
6:         return True
7:     else
8:         return False
```

위 알고리즘을 스크립트에 적용해보면 다음과 같다.

```
1  String.prototype.replaceAll = function(org, dest)
       {
2      return this.split(org).join(dest);
3  }
4
5  function response(room, msg, sender, isGroupChat,
       replier, ImageDB, packageName, threadId) {
6      if (   msg.charAt(0) == "계"
7          && msg.charAt(1) == "산") {
8          var input = msg.split(" ", 2)[1];
9          var test = input;
10         test = test.replace(/ /gi, "");
11         test = test.replace(/[0-9]/g, "");
12         var i = 0;
13         var WhiteList = new Array("\(", "\)", "+",
               "-", "*", "/");
14         while (i<=WhiteList.length){
15             test = test.replaceAll(WhiteList[i], '
                   ');
16             i = i+1;
17         }
18         if (test==""){
19             replier.reply(eval(input));
20         }
```

```
21      }
22  }
```

글로 풀어서 설명하자면, 저자가 사용한 안전하게 `eval()`사용하는 방법은 이렇다.

만약, 입력값이 화이트리스트의 요소들로만 구성된 문자열이라면, 다시말해 여기서는 숫자와 사칙연산자 및 소괄호들로 이루어진 문자열이라면, 입력값에서 숫자와 사칙연산자 및 소괄호들을 모두 제거하면 입력값은 ”이 될 것이다. 따라서 입력값에서 화이트리스트의 요소들에 해당하는 문자열들을 전부 삭제해 ”가 된 문자열들만 `eval()`를 이용해 계산하면 사용자가 자바함수를 호출하는 것을 막을 수 있을 것이다.

간단한 계산기 챗봇의 구현은 여기서 마치고 잠깐 다른 말 좀 하고자 한다. 다른 것은 아니고 혹시나 독자 중 아직 고등교육을 이수하지 않았거나 이수 중인 독자가 있다면, 그 중에서도 이과생인 경우에는 `sin()`함수를 구현해 위의 간단한 계산기 챗봇을 한 단계 수준을 높여보는 것도 도움이 될 것이라 생각한다. 비록 고등교육이 아니더라도 학생인 경우에 학교에서 습득한 내용을 챗봇으로 구현해보는 것은 어떨까? 하고자 한 말은 여기까지이다. 다음 섹션에서는 단계를 높여 간단한 미

니게임 챗봇을 만들어보고자 한다.

## 3.3 Slot Machine

미니게임 챗봇을 만들고 플레이어가 활동한 데이터를 기록하기 위해서는 Android 라이브러리를 이용할 필요가 있다. 물론, javascript에서는 JAVA 라이브러리나 Android 라이브러리를 사용할 수 없다. 그러나 이미 눈치를 챘을 독자들도 많겠지만, 챗봇을 만들 때 사용하는 javascript는 조금 특수한 케이스인지라 Android 개발시 사용되는 API가 사용가능하다! 따라서, 챗봇을 작성할 때에도 Android Package를 개발하는 것처럼 Android 라이브러리를 사용할 수 있다. 우선, 데이터를 기록할 필요가 있으니 플레이어가 활동한 데이터를 독자의 휴대폰 내장 메모리에 접근하는 함수를 작성하자.

```
1  function save(folderName, fileName, str) {
2      var c = new java.io.File(sdcard + "/" +
           folderName + "/" + fileName);
3      var d = new java.io.FileOutputStream(c);
4      var e = new java.lang.String(str);
5      d.write(e.getBytes());
6      d.close();
7  }
8
9  function read(folderName, fileName) {
10     var b = new java.io.File(sdcard + "/" +
           folderName + "/" + fileName);
11     if (!(b.exists())) return null;
```

```
12      var c = new java.io.FileInputStream(b);
13      var d = new java.io.InputStreamReader(c);
14      var e = new java.io.BufferedReader(d);
15      var f = e.readLine();
16      var g = "";
17      while ((g = e.readLine()) != null) {
18          f += "\n" + g;
19      }
20      c.close();
21      d.close();
22      e.close();
23      return f.toString();
24  }
```

도대체 내가 자바스크립트를 하는건지... 자바를 하는건지... 이쯤되면 확장명만 .js인게 아닌가 싶기도 하지만, 쓸데없는 소리는 여기까지 하고. 아니, 애당초에 자바스크립트와 자바는 이름만 비슷할 뿐, 엄연히 다른 프로그래밍 언어이다! 이제, 플레이어의 활동을 저장하고 읽는 것도 가능해졌으니, 게임을 구현할 차례다!